

Data-driven modelling of non-linear systems by means of artificial neural network hybrids

Dissertation
for the award of the degree
Doctor rerum naturalium
of the Georg-August-Universität Göttingen

within the doctoral program Physics of Biological and Complex
Systems (PBCS) of the Georg-August University School of Science
(GAUSS)

submitted by
Sebastian Herzog
from Bromberg

Göttingen 2021

Thesis committee

Prof. Dr. Florentin Wörgötter,

Third Institute of Physics and Bernstein Center for Computational Neuroscience,
Georg-August-Universität Göttingen

apl. Prof. Dr. Ulrich Parlitz,

Max Planck Institute for Dynamics and Self-Organization and Institute for the
Dynamics of Complex Systems, Georg-August-Universität Göttingen

Prof. Dr. Stefan Klumpp,

Institute for the Dynamics of Complex Systems, Georg-August-Universität Göttingen

Members of the examination board

First Reviewer: **Prof. Dr. Florentin Wörgötter,**

Third Institute of Physics and Bernstein Center for Computational Neuroscience,
Georg-August-Universität Göttingen

Second Reviewer: **apl. Prof. Dr. Ulrich Parlitz,**

Max Planck Institute for Dynamics and Self-Organization and Institute for the
Dynamics of Complex Systems, Georg-August-Universität Göttingen

Other members of the examination board:

Prof. Dr. Stefan Klumpp,

Institute for the Dynamics of Complex Systems, Georg-August-Universität Göttingen

Prof. Dr. Timo Betz,

Third Institute of Physics, Georg-August-Universität Göttingen

Prof. Dr. Carsten Damm,

Institute for Informatics - Theoretical Computer Science Group, Georg-August-Universität Göttingen

Prof. Dr. Alexander Ecker,

Institute for Informatics - Neural Data Science Group, Georg-August-Universität Göttingen

Date of the oral examination: 15.03.2021

Acknowledgements

First, I have to thank my supervisors Prof. Dr. Florentin Wörgötter and apl. Prof. Dr. Ulrich Parlitz, for the personal support, countless discussions and the opportunity to realise my scientific ideas. I would also like to thank Prof. Dr. Stefan Luther and Prof. Dr. Claus Wagner to be part of their work groups. Special thanks to my colleagues Dr. Tomas Kulvicius, Dr. Christian Tetzlaff, Dr. Michael Fauth, Dr. Alejandro Agostini, Axel Dannhauer, Tobias Feilke, Dr. Daniel Schiepel, Dr. Thomas Lilienkamp and Dr. Isabella Guido. I would also like to thank the secretaries Ursula Hahn-Wörgötter, Regina Wunderlich, Doris Henze and Annika Köhne from the different work groups. As well as Thomas Geiling for the great information technology support. Last but not least i would like to thank Frauke Bergmann and Antje Erdmann from the GGNB for administrative support.

Thank you all!

Abstract

In the natural sciences, theory and experiment are in permanent interaction with each other. Experimental data provide impulses for new theories and theories suggest new experimental set-ups. Earlier, these two areas had been fairly balanced. However, the rapid increase in performance in semiconductor technology makes modern measurement methods possible. This leads currently to the accumulation of gigantic amounts of data that no human being can process by sight and thought alone. Thus, at the moment, the question of how such floods of data can ever be condensed into models, refined theories, and finally into knowledge is sometimes pushed into the background.

The work presented in this thesis addresses this issue with a focus on data from non-linear systems. This study can be classified as belonging to the field of data-driven modelling with a methodological focus on hybrid artificial neural networks.

The hybrid models presented here are combinations of artificial neural networks, stochastic graphical models and numerical solvers for ordinary differential equations with two goals: The first one is to predict the spatial-temporal dynamics, of the non-linear system, from which the data are coming, over a long time as accurately as possible. While the second one is to find more explicit representations for the data, which are easier to interpret.

With respect to the prediction of the spatial-temporal dynamics of non-linear systems, the idea was to employ a well-known artificial neural network architecture that can encode the data in such a way that it can be well predicted by a stochastic graphical model. A specific advantage of this approach is that the artificial neural network has processing properties that can be used for tasks like state reconstruction.

This artificial neural network+stochastic graphical model hybrid was evaluated on different non-linear systems and was able to achieve prediction horizons that exceeded the former state of the art, sometimes substantially, in all cases.

The second hybrid, is a demonstration how an artificial neural network can be combined with with an numerical equation solver for ordinary differential equations. The goal was to characterise the underlying dynamics of a system as a vector field based on a predefined system of equations given by the user. This approach was applied (under the assumption of Hamilton equations), to the case of a multi agent system and was able to predict a vector field describing the motions of the agents.

Therefore this hybrid approach is not trained to make a spatial-temporal prediction as accurately as possible, but to parameterize the derivative of the hidden states from the assumed equations (e.g. Hamilton equations) so that the predicted vector field represents the data with the smallest possible error. In particular, this approach can be used to validate whether assumptions about the physics (here we assumed that the data can be represented by the Hamilton equations) are applicable.

Finally, a demonstration is given how to apply these two hybrids methods to real experimental data from a Rayleigh–Bénard convection cell and the motion of *Dictyostelium discoideum* (a soil-dwelling amoeba) responding to electric fields of continuous current. For this purpose, the measured raw data must be transferred into a format that is suitable for training the hybrid system. This has been achieved with a newly-developed particle tracking method that is able to reconstruct even high particle densities and assemble them into two or three dimensional trajectories. These trajectories then serve as input for the here-introduced approaches.

In both experimental cases, the two hybrid approaches were able to reproduce the data and make predictions that are easier to interpret than the reconstructed data, for example due to the fact that they are essentially noise free.

In summary this thesis quantifies how different novel combinations of artificial neural networks with stochastic graphical models and numerical equation solvers for ordinary differential equations perform in predicting and explaining data from a variety of complex non-linear systems.

Contents

List of Figures	xi
List of Abbreviations	xiii
1 Introduction	1
1.1 Introductory words	1
1.2 Motivation	2
1.2.1 History of simulation technologies	4
1.2.2 Challenges in the field of numerical simulations	6
1.3 Machine learning	8
1.3.1 Learning	8
1.3.2 Data-driven modelling as an application of machine learning	10
1.4 Dynamical systems	11
1.5 Contribution	11
2 Data processing	15
2.1 Introduction to artificial neural networks and data processing	15
2.1.1 Learning as function approximation	17
2.2 Artificial neural networks	18
2.2.1 Convolutional neural networks	19
2.2.2 Autoencoder	20
2.3 Publication: (Herzog et al. 2021c)	21
2.3.1 Conclusions from (Herzog et al. 2021c)	46
2.4 Publication: (Herzog et al. 2020a)	46
2.4.1 Conclusions from (Herzog et al. 2020a)	57
3 Spatio-temporal data prediction	61
3.1 Introduction to spatio-temporal prediction	61
3.2 Stochastic modelling	62
3.2.1 Stochastic process	62
3.2.2 Graphical models	64
3.3 Spatio-temporal prediction of non-linear dynamics	67
3.3.1 Publication: (Herzog et al. 2019)	67

3.3.1.1	Conclusions from (Herzog et al. 2019)	79
3.3.2	Publication: (Herzog et al. 2018)	79
3.3.2.1	Conclusions from (Herzog et al. 2018)	90
3.3.3	Publication: (Herzog et al. 2020b)	90
3.3.3.1	Conclusions from (Herzog et al. 2020b)	100
3.4	Summary and outlook	100
4	From data to symbols	103
4.1	Introduction data to symbols	103
4.1.1	Learning as a search	104
4.2	Learning symbolic representations	105
4.2.1	Publication: (Herzog et al. 2021a)	106
4.2.1.1	Conclusion from (Herzog et al. 2021a)	115
5	Application to experimental data	117
5.1	Introduction to experimental data application	117
5.2	Publication: (Herzog et al. 2021b)	118
5.3	Conclusions from (Herzog et al. 2021b)	137
5.3.1	Application of the presented methods	137
5.4	Summary	142
6	Conclusion and outlook	145
6.1	Conclusion	145
6.2	Outlook	147
Appendices		
A	Formal definitions for artificial neural networks	155
A.1	Artificial neural networks	155
A.1.1	Network structures	156
A.1.2	Multilayer feed-forward network	157
A.1.3	Recursive definition	157
A.1.4	Network state	158
A.1.5	Topological network dynamics	158
B	Theoretical background for graphical models	161
B.1	Random vector	161
B.1.1	Joint distribution	162
B.1.2	Density	162
B.1.3	Multivariate random variable	162
B.2	Stochastic independence	164

B.3 Graph	164
B.3.1 (Un)directed graph	165
B.4 Gaussian process	165
References	169
C Statement of individual contributions	173

List of Figures

1.1	Structure overview: Introduction chapter 1	2
2.1	Structure overview: Data processing chapter 2	16
2.2	Example autoencoder architecture	20
2.3	Comparison CAE vs CAE+FB for noisy data	57
2.4	Comparison CAE vs CAE+FB for blurred data	58
2.5	Comparison CAE vs CAE+FB for under-sampled data	58
3.1	Structure overview: Spatio-temporal data prediction chapter 3	62
3.2	Example of a factor graph	65
3.3	Example for a CRF with three variables	66
4.1	Structure overview: From data to symbols chapter 4	104
5.1	Structure overview: Application to experimental data chapter 5	118
5.2	Dictyostelium discoideum tracks	138
5.3	Dictyostelium discoideum vector fields	138
5.4	RBC prediction starting from a sphere	140
5.5	RBC prediction over the entire volume	140
5.6	Predicted LSC slices	141
6.1	Structure overview: Conclusion and outlook chapter 6	146
B.1	(Un)directed graph	165

List of Abbreviations

ANN	Artificial neural network
AE	Autoencoder
CFD	Computational fluid dynamics
CRF	Conditional random field
CAE	Convolutional autoencoder
CNN	Convolutional neural network
ESN	Echo state network
ENIAC	Electronical Numerical Integrator and Computer
ERM	Empirical risk minimisation
FB	Feedback
FEM	Finite element method
GM	Graphical model
GPU	Graphics processing unit
IQR	Interquartile range
LSC	Large-scale circulation
MCM	Monte Carlo method
nODE	Neural ordinary differential equations
PDE	Partial differential equations
RBC	Rayleigh-Bénard convection
s.i.	Stochastically independent
SID	System identification

The complexity for minimum component costs has increased at a rate of roughly a factor of two per year. Certainly over the short term this rate can be expected to continue, if not to increase. Over the longer term, the rate of increase is a bit more uncertain, although there is no reason to believe it will not remain nearly constant for at least 10 years

— Gordon Moore 1965, reprinted in (Feynman 1965)

1

Introduction

Contents

1.1	Introductory words	1
1.2	Motivation	2
1.2.1	History of simulation technologies	4
1.2.2	Challenges in the field of numerical simulations	6
1.3	Machine learning	8
1.3.1	Learning	8
1.3.2	Data-driven modelling as an application of machine learning	10
1.4	Dynamical systems	11
1.5	Contribution	11

1.1 Introductory words

This dissertation consists of several parts with the intention to present the manuscripts that were written during my studies as well as some unpublished results. The first chapter deals with the introduction and general motivation, especially the interplay between theory, experiment and simulation. This is followed by the second part, in which problems of numerical simulations are motivated. Then data-driven modelling will be addressed, which in this work is achieved through machine learning, more specifically the use of artificial neural networks and hybrid forms of them, to deal with some of the problems in the cross-section of experiments, simulations and theory.

The following chapters 2, 3 and 4 represent the core of the work and address three specific problems, namely processing data *(Herzog et al. 2021c; Herzog et al. 2020a), learning non-linear dynamics for spatial and temporal prediction *(Herzog et al. 2019; Herzog et al. 2018; Herzog et al. 2020b), and finding symbolic representations from data *(Herzog et al. 2021a). Chapter 5 shows the application of my methods to real experimental data. For this purpose, a particle tracking method has been developed in order to obtain a ground truth that is as accurate as possible *(Herzog et al. 2021b). The last chapter 6 deals with the conclusion and an outlook.

The structure of the work is illustrated in the diagram in figure 1.1.

1.2 Motivation

In the last centuries, science has been based on two components: experiment and theory. Starting with the observation of phenomena, science is supposed to deduce the underlying principles and finally the general theory behind them. Observation began with simple means, e.g. with magnifying glasses and telescopes. These tools then developed over time into more complex tools such as stimulated emission depletion microscopy, magnetic resonance imaging, space telescopes or particle accelerators (just to name a few). All these tools have in common that they are used to search for laws and rules behind the processes in nature in order to describe and ultimately understand them. Science that observes, counts, measures, registers and classifies falls within the realm of empiricism. Empiricists strive to obtain the most accurate information possible about how processes in nature take place.

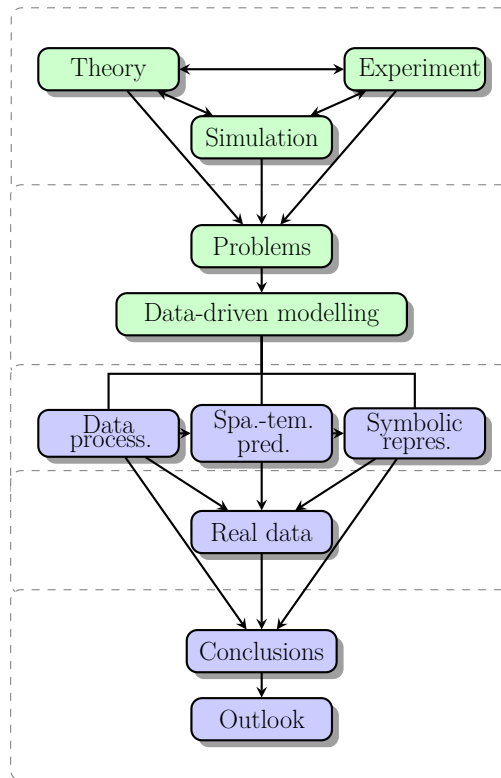


Figure 1.1: Structure overview: Introduction chapter 1

However, this is not the only form of science. Experimental observations with tools are under some circumstances not even enough, to describe nature in a generally valid way. The reverse process is at least as important and has a long tradition: laws and formulas that determine our current view of the world often have their origin in theory. Recognising regularities and formulating them is the task of theoreticians, who often choose the language of mathematics to describe things as precisely as possible. This should be done in such a way that these mathematical descriptions do not only agree with existing observations but can also predict the outcome of experiments that have not yet been carried out. Be it Schrödinger, Euler, Maxwell or Einstein; their equations and systems of equations were not developed in laboratories with the help of measuring tools, but with logical considerations and symbolic descriptions. The validity of Einstein's general relativity theory was proven only years later by observations and experiments. The case of Johannes Kepler shows that it also works the other way round. He used Tycho Brahe's extensive observation material to formulate his planetary laws, and thus measurement results led to new insights through theory building. The subdivision of science into groups of experimentalists and theoreticians is common, although this classification is not to be understood as an absolute binary classification. However, it is important to recognise that these two groups complement each other in an enormous way. With the advancing knowledge of science, the still unknown areas of nature are becoming more complex and harder to describe. Systems that are present in the everyday life, such as the human brain or the weather, are beyond direct theorising through first principles. Many of the hypotheses and theories in this two fields are therefore difficult or impossible to prove experimentally on the object. This requires an even more pronounced interplay between theorists and experimentalists in order to advance science in these fields.

If experiment and theory are considered the two pillars of science, a third pillar is becoming increasingly important and has been indispensable in many fields: numerical simulation. Computer models and simulations create virtual representations of nature by applying the effects of the known laws of nature to

almost any type and number of observation data and parameters. These models have the advantage of being fully observable, parameters and boundary conditions can be changed at will, and one has a control that is not there in reality. With enough computing power and time one could observe every eventuality. At the same time, these results can be matched with observations from the real world, allowing the model to be adjusted, extended and improved again and again. The extension of scientific methodology to include simulation allows a form of experimentation that cannot be done or one that should not be done in reality. This types of numerical experiments bring the field of theory and experiment closer together.

For such simulations to work, numerical models are needed that can be calculated by computers. These numerical models are intended to reproduce real processes or systems in a simplified manner, which often represents an abstraction or idealisation, in order to be able to describe, explain or even predict the dynamics of the system under investigation. Due to the need for numerical models new questions arise that are at the core of many *simulation technologies*. The use of simulations and the technologies based on them provides new challenges. Unfortunately most *simulation technologies* are technically rather complex, being almost always computer simulations consisting of increasingly detailed models, powerful algorithms and powerful computing and storage architectures. This complexity keeps increasing and will increase further and in order to understand this I would like to start with a brief outline concerning *simulation technologies*.

1.2.1 History of simulation technologies

The mathematical beginnings of modelling, i.e. the basis for simulation, can already be found in the so-called *Buffon's needle problem* by Buffon and Laplace in the 18th century (Aigner et al. 2004). In the needle problem, the probability is to be calculated with which a needle lands on the line of a pre-drawn grid. A solution to *Buffon's needle problem* can be found by many repetitions of a random experiment: describe the position in which the needle falls by a random variable and count how many times it falls on one of the grid lines. This idea led to a

basic simulation method: the so-called *Monte Carlo method* (MCM). Defined by John von Neumann and Stanislaw Ulam in 1946 (Eckhardt 1987) this is a method for calculating the probability or uncertainty of results based on a large number of independent random experiments. This method is still essential for numerical modelling and simulation today. For example, MCMs can even be used for the solution of partial differential equations (PDE) (Barth et al. 2011). A PDE is a differential equation that contains partial derivatives, i.e. equations involving processes with more than one independent variable. Such equations are used for the mathematical modelling of many physical processes. The solution theory of partial differential equations has been largely researched for linear equations, but still contains many gaps for non-linear equations. Therefore, these are solved numerically by approaches such as MCMs.

The spatial and temporal discretisation of PDEs quickly became a new field of interest for many researchers. As a consequence, only a few years later in the 1960s (Argyris et al. 1969) a methodological scheme had been developed that still has great relevance today. Again, the goal was to solve PDEs, but without the stochastic character of MCMs and computationally more efficient and in a scalable manner. The method developed was the so-called finite element method (FEM), initially developed for the simulation of solids, but today also used in many other physical problems, such as weather forecasting or medical simulations.

However, it were not only the mathematical methods that improved rapidly. In addition to this, also the ground-breaking technical developments in semiconductor technology boosted computer technology. The complexity of integrated circuits doubled after regular intervals (Moore 2006), which often also implied a doubling of computing power, this growth was crucial for the rise of simulation methods, allowing to run simulations at a speed and with a complexity that today makes models possible of dynamical systems from the nanoscale to supernovae.

The technical basis for this was provided by Konrad Zuse in the 1940s with the Z3 (Alex 2000). The Z3 was the first functioning computer. Electromagnetic relays had been used to perform the arithmetic operations, which were rather slow and

resulted in a correspondingly low computing power. However, only a few years later the first fully electronic digital universal computer ENIAC (Electronical Numerical Integrator and Computer) (Haigh et al. 2016), had been presented in 1947 at the University of Pennsylvania with a tremendously increased computational power. The first simulations on ENIAC were stochastic simulations of nuclear fusion, which were the basis for the hydrogen bomb. Even though this was military research, a general advantage is already apparent here: experiments that would be too dangerous or ethically questionable can be tested in a simulation. Virtual test runs make it possible to selectively tweak individual parameters and check effects without risk. This made simulations attractive not only for military purposes, but also in all those areas where costly or risky experiments could be minimised or even avoided this way.

Up to this point, simulations served less to describe already existing systems but more to specifically design individual material functions or to forecast (un-)desired behaviour. With the increasing power of computers since the 1970s, it became possible to simulate increasingly complex scenarios and models - the most prominent areas of application here are weather forecasting (Shuman 1989), election predictions or the development of high-risk technologies. Another advantage of the increasing computing power was the possibility to visualise the results of the simulations. Often only the essential components of a simulation can be visualised, but this can be crucial to lead to new insights, for example, as in the case of the recently presented and so far most precise simulation of the formation of our universe (Pillepich et al. 2019).

The hopes and partial promises of simulation as a scientific methodology are thus very high, but none of this "comes for free". There are still many challenges that need to be addressed to allow for effective use of simulations.

1.2.2 Challenges in the field of numerical simulations

As already explained above, simulations need numerical models. Only with those it is possible to use simulations in cases where theory and experiment fall short. Sometimes with this it is even possible to arrive at ethically safe verification of

certain assumptions and scenarios, for example in the development of new drugs. On the other hand, lack of a good model makes simulation mostly useless. In some simple cases where the theory is already very well understood it is possible to derive numerical models from first principles. Models consider only sub-aspects of reality and are, as already mentioned, always an abstraction or idealisation. They never describe reality with absolute accuracy, but capture certain relevant aspects "sufficiently well", neglecting other details irrelevant to the question at hand. Seen in this light, "all models are wrong", as the statistician George Box (1976) from the University of Wisconsin in Madison provocatively put it.

To take up this provocation, one can consider weather forecasts. These forecasts allow the approximate calculation of effects for which measurements are not yet available. However, they are based on many simplifications and are, while in general useful, not always correct. Weather modelling is one of the oldest application fields for simulations, where countless experts have invested an immense amount of time to come up with different models.

To further complicate matters, the balance between experiment and theory has shifted over the last 20 years to an extent that in some cases rises the fears of decoupling. The main reason for this is in my opinion that considerably more funding has gone into experimental facilities such as accelerators, telescopes, sequencers or computers than into theory building. In conjunction with the rapid increase in performance in semiconductor technology, this has led to the accumulation of gigantic amounts of data that no human being can process by sight and thought alone. The question of how such floods of data can ever be condensed into theories, refined models, and finally into knowledge, was often pushed into the background.

The work presented in this thesis addresses this issue with a focus on data coming from non-linear systems. It can be classified as belonging to the field of data-driven modelling and has a methodological focus on hybrid artificial neural networks. This is summarised by the title "Data-driven modelling of non-linear systems by means of artificial neural network hybrids".

1.3 Machine learning

From the first successful use of ENIAC, it was clear that the need for programmers has been a bottleneck. In order to relieve programmers of routine work, the use of machine learning was formulated as a goal. Alan Turing considered the learning ability of a computer to be the most important achievement. His recommendation was to "educate" a computer, so that it improves its performance, since it is impossible to program everything (Turing et al. 1997). In 1957, Alonzo Church defined the task to synthesise a circuit from mathematical requirements (Friedman 1963). This later became known as *Church's problem*. It was one of the earliest descriptions of *program synthesis* and this led to the concept of machine learning being narrowed down to the automatic acquisition of rules or the improvement of rule sets.

1.3.1 Learning

When talking about machine learning, the question of what one defines as learning arises quite quickly. As simple as this question sounds, problems became visible immediately after the first attempts at a definition.

A well-known definition by Simon (1983) is:

Definition 1.3.1 (Simons definition of learning (Simon 1983)). Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.

This has led to two points of criticism. It covers phenomena that are not usually called learning while at the same time it does not cover all phenomena attributed to learning. An example of how learning is not the only reason for improved performance comes from Michalski (2002). If the task is to cut something, performance is improved by taking a sharper knife, this could also happen for coincidental reasons. Based on Simons definition, this would be a learning process. On the other hand, the *realisation* that cutting is more efficient with a sharper knife would indeed be a learning process. According to Simon's definition, the ability of

programmes to learn could be demonstrated by running the same programme on a faster computer. The system, computer and programme, would then solve the same task faster. But it is obvious that this is not learning. Learning as such is a more complex process, Michalski gives a drastic example that performance reduction can be a learning outcome (Michalski 2002). One could learn how to do less and still look equally busy. Michalski, thus, wants to raise awareness of the goal-dependency of the concept of performance. Depending on how one defines the task, this may or may not fall under Simon's definition. Scott in 1983 argues against performance measurements in the definition of learning (Shalin et al. 1988). He gives the example of a walker in a city that is still unknown to him, who passes by the public library. While he perceives it, he learns something about the city without having any task for which he would have to know whether and where there is a library. Only when someone else asks the walker for direction to the library than the walker uses what he has learned. But even without the question of direction by someone else, the walker has learned. The walker learns regardless of whether it is tested. Based on this, Scott defines in 1983 learning without a given performance measure:

Definition 1.3.2 (Scotts definition of learning (Shalin et al. 1988)). Learning is a process by which a system builds a retrievable representation of past interactions with its environment.

Thus, performance is potentially observable, because the new representation is retrievable. However, the learning itself is independent of whether its result is ever needed. Nor is a reduction in performance through learning ruled out. For example, someone who knows only one statement about something when asked about it might be able to answer more quickly than someone who has to search for the right one from a wealth of information. Later in 1986, Michalskis definition is similar:

Definition 1.3.3 (Michalski definition of learning (Michalski 2002)). Learning is the construction or modification of representations of experience.

Both definitions presuppose a process that uses representations. The extent to which this is built up or changed through learning remains open. Even from this brief discussion of the definition of learning, it is clear that learning is similarly difficult to grasp as intelligence. We are left with our colloquial understanding of what learning is for us as a suggestion and a guide.

Machine learning, like all other sub-fields of artificial intelligence, has three different motivations: a cognitive-scientific, a theoretical-technical, and a practical, application-oriented one. The work here focuses in particular on the last two points, where inductive conclusions are particularly important.

1.3.2 Data-driven modelling as an application of machine learning

For us humans (and for scientific work), the formation of concepts can be divided, very crudely and naive, into two phenomena: aggregation and characterisation (or definition). Aggregation groups objects, events and facts of the world into classes or categories. A category is the extension of a concept. Characterisation describes a category so that it can be decided for new objects in which category they belong. The intentional description of the category thus serves to determine the class membership. An object is recognised as an example of a concept if the characterisation of the concept covers the object. A concept is a mental, cognitive unit that refers to a category. Leading to the chain of aggregation, characterisation and classification which is a kind of recognition, finally leading to theory building.

The extraction of this kind of knowledge from observable data with the intention of modelling is called data-driven modelling.

Machine learning is predominantly used to extract a set of rules from data or to improve a given set of rules. The rules are then either used directly by humans or embedded in a system and thus made available to its users, making it possible to inspect the data and get an overview (*description*). On the other hand, the rules can be used to solve new cases (*prediction*). Both *description* and *prediction* with the

application to dynamical systems are components of the investigations carried out here. For this purpose, the concept of a dynamical system should also be discussed.

1.4 Dynamical systems

A dynamical system can be defined as a deterministic mathematical concept, consisting of a *state space*, a set of times and a family of *evolution functions*, to describe the evolution of system states through time. Time is either continuous or discrete. If the time t is continuous then a dynamical system typically consist of a set of N first-order, autonomous, ordinary differential equations given by

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{F}(\mathbf{x}(t)),$$

where $\mathbf{x} \in \mathbb{R}^N$ denotes the state vector. For the case that t is discrete a dynamical system is given by a map

$$\mathbf{x}_{t+1} = \mathbf{M}(\mathbf{x}_t).$$

In both cases \mathbf{F} , \mathbf{M} are the so-called *evolution functions*. A dynamical systems is called *non-linear* if the *evaluation functions* \mathbf{F} or \mathbf{M} are non-linear. The evolution of \mathbf{x} over time t results in a *trajectory*. In this thesis dynamical systems with dissipative structures are considered. The asymptotic dynamics of this dissipative systems is governed by attracting bounded subsets of the state space, like stable fixed points, stable period orbits, attracting tori (quasi-periodic dynamics), and chaotic (strange) attractors (characterized by aperiodic oscillations and sensitive dependence on initial conditions).

1.5 Contribution

In order to support the process of theory and model building, algorithms that are purely data-driven have been developed in this work. All these approaches are based on artificial neural networks or are hybrid forms of them. The previous part of this cumulative dissertation dealt with the introduction as well as the motivation of the subject matter. These are the first two parts of figure 1.1.

The following parts will now present several papers which form the core of this thesis:

1. Herzog, S., Zimmermann, R. S., Abele, J., and Parlitz, U. (2021c). “Reconstructing Complex Cardiac Excitation Waves From Incomplete Data Using Echo State Networks and Convolutional Autoencoders”. In: *Frontiers in Applied Mathematics and Statistics*. accepted version 2020/12/07 - shown in this work, published version 2021/03/18 - available online: <https://www.frontiersin.org/articles/10.3389/fams.2020.616584/full>
2. Herzog, S., Tetzlaff, C., and Wörgötter, F. (2020a). “Evolving artificial neural networks with feedback”. In: *Neural Networks : the official journal of the International Neural Network Society* 123, pp. 153–162
3. Herzog, S., Wörgötter, F., and Parlitz, U. (2019). “Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos”. In: *Chaos (Woodbury, N.Y.)* 29.12
4. Herzog, S., Wörgötter, F., and Parlitz, U. (2018). “Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media”. In: *Frontiers in Applied Mathematics and Statistics* 4
5. Herzog, S. and Wagner, C. (2020b). “Development of Artificial Neural Networks with Integrated Conditional Random Fields Capable of Predicting Non-linear Dynamics of the Flow Around Cylinders”. In: *New Results in Numerical and Experimental Fluid Mechanics XII*. Cham: Springer International Publishing, pp. 71–79
6. Herzog, S., Schiepel, D., Guido, I., and Wagner, C. (2021b). “A probabilistic particle tracking framework for guided and Brownian motion systems with high particle densities”. submitted to *Int J Comput Vis*
7. Herzog, S. and Wörgötter, F. (2021a). “Application of neural ordinary differential equations to the prediction of multi-agent systems”. accepted

for SWARM 2021 (to be considered for full publication in Artificial Life and Robotics)

These works will be cited in my thesis marked with an "*" to point their specific contribution out as required.

*My greatest concern was what to call it. I thought of calling it *information*, but the word was overly used, so I decided to call it *uncertainty*.*

*When I discussed it with John von Neumann, he had a better idea. Von Neumann told me, 'You should call it *entropy*, for two reasons. In the first place your uncertainty function has been used in statistical mechanics under that name, so it already has a name. In the second place, and more important, no one really knows what entropy really is, so in a debate you will always have the advantage.'*

—Claude E. Shannon, reprinted in (Tribus et al. 1971a)

2

Data processing

Contents

2.1	Introduction to artificial neural networks and data processing	15
2.1.1	Learning as function approximation	17
2.2	Artificial neural networks	18
2.2.1	Convolutional neural networks	19
2.2.2	Autoencoder	20
2.3	Publication: (Herzog et al. 2021c)	21
2.3.1	Conclusions from (Herzog et al. 2021c)	46
2.4	Publication: (Herzog et al. 2020a)	46
2.4.1	Conclusions from (Herzog et al. 2020a)	57

2.1 Introduction to artificial neural networks and data processing

If no information about a system is available, model and theory building is not possible, in which case data acquisition is necessary in the hope of obtaining information about the system under study. However, collecting data does not mean obtaining compelling information (more specific new knowledge). The process of collecting data is exposed to many sources of interference, such that many factors can play a role that ultimately determine whether it is possible to obtain

information about the studied system, information that is necessary to form models, models from which theories follow. One of these cases is that during the collection of data, effects are also measured that have nothing to do with the system under observation and come from a different source. In such a case, one often speaks of noise. In addition to noise, there are also other problems, for example the resolution (both spatially and temporally) may not be sufficient to observe the system at a level to obtain enough information to form theories. Likewise, the system under observation may have characteristics that can only be partially observed. In the worst case, there may even be a combination of all of these problems, which - alas -

happens often. The first publication in this thesis Herzog, S., Zimmermann, R. S., Abele, J., and Parlitz, U. (2021c). “Reconstructing Complex Cardiac Excitation Waves From Incomplete Data Using Echo State Networks and Convolutional Autoencoders”. In: *Frontiers in Applied Mathematics and Statistics*. accepted version 2020/12/07 - shown in this work, published version 2021/03/18 - available online: <https://www.frontiersin.org/articles/10.3389/fams.2020.616584/full> deals with this question. In this paper, two artificial neuronal networks are examined for their ability to reconstruct noisy, blurred, under-sampled data and later on even to recover patterns from impaired observations.

Before the paper is presented, I would like to put it in relation to the introduction and also the concept of learning should be more specified.

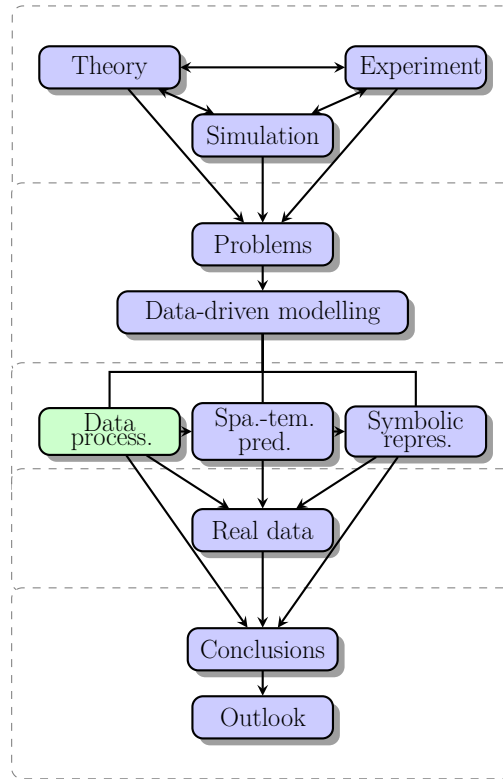


Figure 2.1: Structure overview: Data processing chapter 2

2.1.1 Learning as function approximation

Many learning problems can be understood as function approximation. The goal is to find a hypothesis that can be used to transfer one set of states to another set of states. The first case presented in this chapter will be the case of predicting a set of impalpable properties based on a set of observable ones. This can be considered as *state-space reconstruction*. In chapter 3, the second case will be discussed, where the goal will be to predict spatial–temporal dynamics. For both cases pairs of examples of the function to be learned are needed. The hypothesis that best approximates this function is sought. A hypothesis approximates the objective function well if its predictions are fulfilled as often as possible. This model can be formally written down as follows:

Definition 2.1.1. Assuming a generator G , that generates sample descriptions $x_i \in K$, where K is some arbitrary space, based on a probability distribution $p(x_i)$. An oracle O , that assigns a value $y_i = t(x_i)$ to each example description x_i generated by G , in a hypothesis language LH . The goal is to find the hypothesis $h \in LH$, which minimises the following expression:

$$R(h) = \sum_{i=1}^{|K|} L(x_i, h) \cdot p(x_i).$$

$p(x_i)$ is the probability that the example x_i is drawn from the example description language. It is therefore important to make fewer errors on likely examples x_i than on unlikely examples. $L(x_i, h)$ is an error function (so-called *loss function*). It describes the quality of the prediction of hypothesis h , for example x_i . Based on the form of L , one distinguishes between the following two tasks, among others: *Classification* of examples into a fixed and given number of classes (e.g. classification between cats and dogs). The following error function is normally used here, which returns the value 1, if the prediction $h(x_i)$ is false (unequal to the true class of x_i denoted by $t(x_i)$).

$$L(x_i, h) = \begin{cases} 0 & h(x_i) = t(x_i) \\ 1 & h(x_i) \neq t(x_i) \end{cases}.$$

The second task is the *regression* task, here O is supposed to approximate real-valued function (e.g. prediction of stock prices). Often L here is the squared deviation of the predicted value $h(x) \in \mathbb{R}$ from the target value $t(x) \in \mathbb{R}$.

$$L(x_i, h) = (t(x_i) - h(x_i))^2$$

Direct minimisation of the expected error $R(h)$ is not possible, because neither $p(x_i)$ nor $t(x_i)$ for all i is known. However, examples drawn by the generator exist using $p(x_i)$ for which we know $t(x_i)$. These examples are used to approximate the expected error $R(h)$ with the observed error $R_o(h)$. The observed error for a set of examples $(x_1, t(x_1)), \dots, (x_n, t(x_n))$ and a hypothesis h is calculated as:

$$R_o(h) = \frac{1}{n} \sum_{i=1}^n L(x_i, h).$$

Definition 2.1.2 (Empirical risk minimisation (ERM) learning). From this, the following learning problem can be formulated. Given a set of examples

$$(x_1, t(x_1)), \dots, (x_n, t(x_n))$$

and the hypothesis language LH . The objective is to find the hypothesis $h \in LH$, for which the observed error $R_o(h)$ is minimal.

These definitions, which are based on information theory, do not lose their meaning for artificial neural networks.

2.2 Artificial neural networks

Learning often, but in artificial neural networks particularly, corresponds to empirical risk minimisation. In the following, neural networks are treated from the point of view of machine learning and no attempt is made to model biological processes in the brain. In the field of machine learning an artificial neural network (ANN) is a tuple of nodes, some given network structure, a set of weights, biases and activation functions which process an input to an output, a more formal definition is given in the appendix A.1.1. Based on the network structure (as defined in A.1.2) and

the weights, one gets the architecture of the ANN. Depending on the topological network dynamics defined in A.1.5 ANNs are often divided into *recurrent* and *feed-forward* networks.

2.2.1 Convolutional neural networks

With the presentation of the *AlexNet* by Krizhevsky et al. (2012) ANNs gained great attention. AlexNet is a variant of a *feed-forward* convolutional neural network (CNN) design introduced by LeCun et al. (1989) and further improved in (LeCun et al. 1998). In many reports that appear in the media, CNNs are often mentioned when talking about artificial neural networks. They are used in many areas like in the field of radiology (Yamashita et al. 2018), epidemiological imaging for image analysis (Ivanovska et al. 2019) but also speech recognition (Grefenstette et al. 2014) and many more. As the name suggests, CNNs use convolutional operations to process the input.

Definition 2.2.1 (Convolution). Let $f, g : \mathbb{R}^p \rightarrow \mathbb{R}$ be two functions with $\int |f|^2 dP < \infty$ and $\int |g|^2 dP < \infty$. Then, the function

$$(f * g)(t) := \int_{\mathbb{R}^p} f(y)g(t - y)dy$$

is called *convolution* of f and g . For the discrete case, i.e. $t \in \mathbb{Z}$, the *discrete convolution* is defined by

$$(f * g)(t) := \sum_{y=-k}^k f(y)g(t - y),$$

where $k \in \mathbb{N}$ is the size of the *convolution kernel* used ($k = \pm\infty$ is in theory also possible).

These *feed-forward* networks were primarily developed to process static images, like for image classification or labelling. As will be shown later, however, they are also very useful for cases where fields or similar data can be displayed as images. CNNs also have an architecture and this is decisive for which functions the network can approximate and how.

2.2.2 Autoencoder

For this and the next chapter, the *feed-forward* neural network has the architecture of an autoencoder (AE) (Kramer 1991). AEs are structured in 3 parts:

1. The encoder part: This is the first part of the neural network and aims to compress the input to a more compact representation.
2. The compact representation is the bottleneck of the AE and often called latent space.
3. The third part of the network is the decoder part, it is the inverse function of the encoder part and should transform the data back into the original representation.

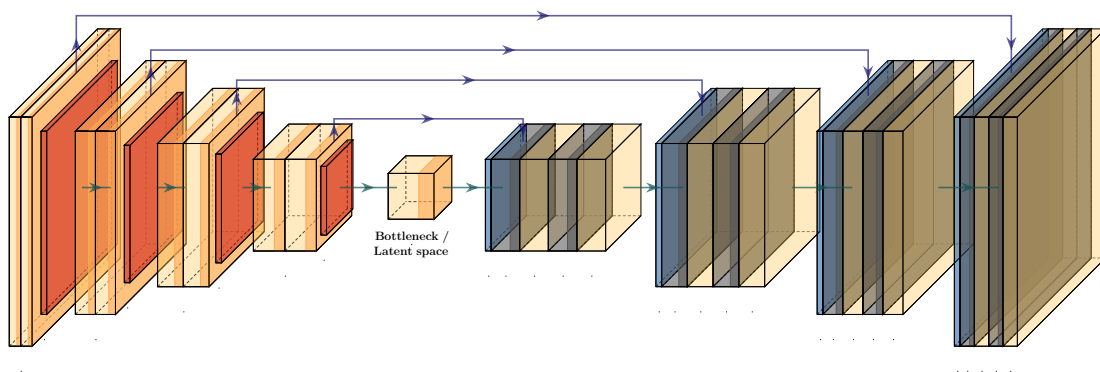


Figure 2.2: Exemplary representation of an autoencoder neural network with *convolutional* layers. This illustration was generated with software library of Iqbal (2018). The rectangles in the light yellow are the convolutional-, the rectangles in orange are the pooling-layers. These are simply layers that reduce the dimension, e.g. by always outputting only the pixel with the largest intensity of the neighbouring pixels. The arrows in blue, which skip over neighbouring layers are the so called *residual connections*.

The objective of these architectures is to learn an efficient encoding and other useful properties (Goodfellow et al. 2016). This type of function approximation (section 2.1), is very powerful and an elementary building block for the following methods. A visualisation of this architecture is presented in figure 2.2. However, it only serves as an illustration. Connections that go beyond neighbouring layers should also be noted, they are called *residual connection* (He et al. 2016). The

hourglass-shaped structure of the autoencoder is characteristic for this architecture and there are still some variations like *stacked autoencoders* (Vincent et al. 2010), *variational autoencoders* (Kingma et al. 2019), *transforming auto-encoders* (Hinton et al. 2011), just to name of few of them.

2.3 Publication: (Herzog et al. 2021c)

The first publication that is now presented is Herzog, S., Zimmermann, R. S., Abele, J., and Parlitz, U. (2021c). “Reconstructing Complex Cardiac Excitation Waves From Incomplete Data Using Echo State Networks and Convolutional Autoencoders”. In: *Frontiers in Applied Mathematics and Statistics*. accepted version 2020/12/07 - shown in this work, published version 2021/03/18 - available online: <https://www.frontiersin.org/articles/10.3389/fams.2020.616584/full>. This work had two intentions. The first was to see how well ANNs could be used to represent data of non-linear systems. The second question was whether a *feed-forward network* in the form of a *convolutional autoencoder* (CAE) or a recurrent network in the form of an *recurrent echo state network* (ESN) is a better function approximator to reconstruct noisy, blurred, under-sampled data and later on even to recover patterns from impaired observation. For the synthetic data sets used to evaluate both methods in *(Herzog et al. 2021c) both networks generated satisfying solutions clearly indicating that such data reconstruction tasks can be solved by means of ANNs.

Reconstructing complex cardiac excitation waves from incomplete data using echo state networks and convolutional autoencoders

Sebastian Herzog^{1,2}, Roland S. Zimmermann^{3,4}, Johannes Abele^{1,4}, Stefan Luther^{1,5,6} and Ulrich Parlitz^{1,4,6*}

¹ Max Planck Institute for Dynamics and Self-Organization, Germany

² Third Institute of Physics and Bernstein Center for Computational Neuroscience, University of Göttingen, Germany

³ Tübingen AI Center, University of Tübingen Germany

⁴ Institute for the Dynamics of Complex Systems, University of Göttingen, Germany

⁵ Institute of Pharmacology and Toxicology, University Medical Center Göttingen, Germany

⁶ DZHK (German Center for Cardiovascular Research), partner site Göttingen, Germany

Correspondence*:

Corresponding Author

ulrich.parlitz@ds.mpg.de

2 ABSTRACT

3 The mechanical contraction of the pumping heart is driven by electrical excitation waves running
4 across the heart muscle due to the excitable electrophysiology of heart cells. With cardiac
5 arrhythmias these waves turn into stable or chaotic spiral waves (also called rotors) whose
6 observation in the heart is very challenging. While mechanical motion can be measured in
7 3D using ultrasound, electrical activity can (so far) not be measured directly within the muscle
8 and with limited resolution on the heart surface, only. To bridge the gap between measurable
9 and not measurable quantities we use two approaches from machine learning, echo state
10 networks (ESNs) and convolutional autoencoders (CAEs), to solve two relevant data modelling
11 tasks in cardiac dynamics: Recovering excitation patterns from noisy, blurred or undersampled
12 observations and reconstructing complex electrical excitation waves from mechanical deformation.
13 For the synthetic data sets used to evaluate both methods we obtained satisfying solutions with
14 ESNs and good results with CAEs, both clearly indicating that the data reconstruction tasks can
15 in principle be solved by means of machine learning.

16 **Keywords:** reservoir computing, echo state networks, convolutional autoencoder, image enhancement, cross-prediction, cardiac
17 arrhythmias, excitable media, electro-mechanical coupling, cardiac imaging

1 INTRODUCTION

18 Cardiac arrhythmias, such as ventricular or atrial fibrillation, are electro-mechanical dysfunctions of the
19 heart that are associated with complex, chaotic spatio-temporal excitation waves within the heart muscle
20 resulting in incoherent mechanical contraction and a significant loss of pump function [1, 2, 3]. Ventricular
21 fibrillation (VF) is the most common deadly manifestation of a cardiac arrhythmia and requires immediate
22 defibrillation using high-energy electric shocks. Atrial fibrillation (AF) is the most common form of a

23 cardiac arrhythmia, affecting 30 million patients worldwide. While not immediately life-threatening, AF is
24 considered to be responsible for 15% of strokes if left untreated. The structural substrate and functional
25 mechanisms that underlie the onset and perpetuation of VF and AF are not fully understood. It is generally
26 agreed that imaging of the cardiac electrical and mechanical function is key to an improved mechanistic
27 understanding of cardiac disease and the development of novel diagnosis and therapy. This has motivated
28 the development of non-invasive and invasive electrophysiological measurement and imaging modalities.
29 Electrical activity of the heart can (so far) be measured on its surface, only. Direct measurements can be
30 made in-vivo inside the heart using so-called basket catheters with typically 64 electrodes or in ex-vivo
31 experiments, where an extracted heart in a Langendorff perfusion set-up is kept beating and the cell
32 membrane voltage on the epicardial surface is made visible using fluorescent dyes (a method also known
33 as optical mapping) [4]. A method for indirect observation of electrical excitation waves is ECG imaging
34 where an array of EEG-electrodes is placed on the body surface and an (ill-posed) inverse problem is solved
35 to estimate the potential on the surface of the heart. Mechanical contraction and deformation of the heart
36 tissue can be studied in full 3D using ultrasound, in 2D using real-time MRT [5] or (on the surface only) by
37 motion tracking in Langendorff experiments.

38 The reconstruction of patterns of action potential wave propagation in cardiac tissue from ultrasound has
39 been introduced by Otani et al. in 2010 [6, 7]. They proposed to use ultrasound to visualise the patterns of
40 propagation of these waves through the mechanical deformations they induce and to reconstruct action
41 potential-induced active stress from the deformation. Provost et al. [8] introduced electromechanical wave
42 imaging to map the mechanical deformation of cardiac tissue at high temporal and spatial resolutions.
43 The observed deformations resulting from the electrical activation were found to be closely correlated
44 with electrical activation sequences. The cardiac excitation-contraction-coupling (ECC) [9] has also
45 been studied in optical mapping experiments in Langendorff-perfused isolated hearts [10, 11, 12]. Using
46 electromechanical optical mapping [12], it was shown that during ventricular tachyarrhythmias electrical
47 rotors introduce corresponding rotating mechanical waves. These co-existing electro-mechanical rotors
48 were observed on the epicardial surface of isolated Langendorff-perfused intact pig and rabbit using optical
49 mapping [13]. Using high-resolution ultrasound, these mechanical rotors were also observed inside the
50 ventricular wall during ventricular tachycardia and fibrillation [13].

51 All these measurement modalities are limited, in particular those suitable for in vivo applications.
52 Measurements with basket catheters are effectively undersampling the spatio-temporal wave pattern. Inverse
53 ECGs suffer from ill-posedness and required regularisation that may lead to loss of spatial resolution and
54 blurring. Limited spatial resolution is also an issue with ultrasound measurements, but they are currently
55 the only way to “look inside” the heart, albeit measuring only mechanical motion. Electrical excitation
56 waves inside the heart muscle are so far not accessible by any measurement modality available.

57 These limitations motivated the search for algorithms to reconstruct electro-mechanical wave dynamics
58 in cardiac tissue from measurable quantities. Berg et al. [14] devised synchronization-based system
59 identification of extended excitable media, in which model parameters are estimated by minimizing
60 the synchronization error. Using this approach, Lebert and Christoph [15] demonstrated that electro-
61 mechanic wave dynamics of excitable-deformable media can be recovered from a limited set of observables
62 using a synchronization-based data assimilation approach. Hoffman et al. reconstructed electrical wave
63 dynamics using ensemble Kalman filters [16, 17]. In an another approach, it was shown that echo state
64 networks [18] and deep convolutional neural networks [19, 20] provide excellent cross estimation results
65 for different variables of a mathematical model describing complex electrical excitation waves during
66 cardiac arrhythmias. Following this approach, Christoph and Lebert [21] demonstrated the reconstruction of
67 electrical excitation and active stress from deformation using a simulated deformable excitable medium. To

68 continue this research and to address the general challenge of missing or impaired observations we consider
 69 in this article two tasks: (i) recovering electrical excitation patterns from noisy, blurred or undersampled
 70 observations and (ii) reconstructing electrical excitation waves from mechanical deformation. To solve the
 71 corresponding data processing and cross-prediction tasks two machine learning methods are employed and
 72 evaluated: echo state networks and convolutional autoencoders. Both algorithms are applied to synthetical
 73 data generated by prototypical models for electrophysiology and electromechanical coupling.

2 METHODS

74 In this section we will first introduce in section 2.1 and section 2.2 the mathematical models describing
 75 cardiac dynamics which were used to generate the example data for the two tasks to be solved: (i) recovering
 76 electrical wave pattern from impaired observations and (ii) cross-predicting electrical excitation from
 77 mechanical deformation. Then in section 2.3 both machine learning methods used for solving these
 78 tasks, echo state networks (section 2.3.1) and convolutional autoencoders (section 2.3.2), will be briefly
 79 introduced.

80

81 2.1 Recovering complex spatio-temporal wave patterns from impaired observations

82 For motivating, illustrating, and evaluating the employed methods for dealing with incomplete or distorted
 83 observations we shall use spatio-temporal time series generated with the Bueno-Orovio-Cherry-Fenton
 84 (BOCF) model [22] describing complex electrical excitation patterns in the heart during cardiac arrhythmias.
 85 The BOCF model is a set of partial differential equations (PDEs) with four variables and will be introduced
 86 in section 2.1.1. In section 2.1.2 a formal description of the data recovery tasks will be given.

87

88 2.1.1 Bueno-Orovio-Cherry-Fenton Model

89 Cardiac dynamics is controlled by electrical excitation waves triggering mechanical contractions of the
 90 heart. In the case of cardiac arrhythmias like lethal ventricular fibrillation, wave break-up and complex
 91 chaotic wave patterns occur resulting in significantly reduced pump performance of the heart. From the
 92 broad range of mathematical models describing this spatio-temporal dynamics [23] we chose the Bueno-
 93 Orovio-Cherry-Fenton (BOCF) model [22] to generate spatio-temporal time series that are used as a
 94 benchmark to validate our approaches for reconstructing complex wave patterns in excitable media from
 95 incomplete data. The BOCF model consists of four system variables whose evolution is given by four
 96 (partial) differential equations

$$\begin{aligned}
 \frac{\partial u}{\partial t} &= D \cdot \nabla^2 u - (J_{si} + J_{fi} + J_{so}) \\
 \frac{\partial v}{\partial t} &= \frac{1}{\tau_v^-} (1 - H(u - \theta_v)) (v_\infty - v) - \frac{1}{\tau_v^+} H(u - \theta_v) v \\
 \frac{\partial w}{\partial t} &= \frac{1}{\tau_w} (1 - H(u - \theta_w)) (w_\infty - w) - \frac{1}{\tau_w^+} H(u - \theta_w) w \\
 \frac{\partial s}{\partial t} &= \frac{1}{2\tau_s} ((1 + \tanh(k_s(u - u_s))) - 2s).
 \end{aligned} \tag{1}$$

97 The variable u represents the continuum limit representation of the membrane voltage of cardiac cells
 98 and the variables v , w , and s are gating variables controlling ionic transmembrane currents J_{si} , J_{fi} and J_{so}
 99 given by the equations

$$\begin{aligned}
J_{si} &= -\frac{1}{\tau_{si}}H(u - \theta_w)ws \\
J_{fi} &= -\frac{1}{\tau_{fi}}vH(u - \theta_v)(u - \theta_v)(u_u - u) \\
J_{so} &= \frac{1}{\tau_o}(u - u_o)(1 - H(u - \theta_w)) + \frac{1}{\tau_{so}}H(u - \theta_w).
\end{aligned} \tag{2}$$

100 Here $H(\cdot)$ denotes the Heaviside function and the currents depend on the following seven voltage
101 controlled variables

$$\begin{aligned}
\tau_v^- &= (1 - H(u - \theta_v^-))\tau_{v1}^- + H(u - \theta_v^-)\tau_{v2}^- \\
\tau_w^- &= \tau_{w1}^- + \frac{1}{2}(\tau_{w2}^- - \tau_{w1}^-)(1 + \tanh(k_w^-(u - u_w^-))) \\
\tau_{so}^- &= \tau_{so1} + \frac{1}{2}(\tau_{so2} - \tau_{so1})(1 + \tanh(k_{so}(u - u_{so}))) \\
\tau_s &= (1 - H(u - \theta_w))\tau_{s1} + H(u - \theta_w)\tau_{s2} \\
\tau_o &= (1 - H(u - \theta_o))\tau_{o1} + H(u - \theta_o)\tau_{o2}
\end{aligned} \tag{3}$$

$$v_\infty = \begin{cases} 1, & \text{if } u \leq \theta_v^- \\ 0, & \text{if } u \geq \theta_v^- \end{cases}$$

$$w_\infty = (1 - H(u - \theta_o))(1 - \frac{u}{\tau_{w\infty}}) + H(u - \theta_o)w_\infty^*.$$

102 For simulating the dynamics we used the set of parameters given in Table 1 for which the BOCF model
103 was found [22] to exhibit excitation wave dynamics similar to the *Ten Tusscher-Noble-Noble-Panfilov*
104 (TNNP) model [24] describing human heart tissue.

u_o	0	τ_{v2}^-	1150	τ_{fi}	0.11	τ_{s1}	2.7342	τ_{s2}	3	τ_{o1}	6	τ_{o2}	6
u_u	1.58	τ_v^+	1.4506	τ_{w1}^-	70	τ_{w2}^-	20	τ_{so1}	43	τ_{so2}	0.2	τ_{si}	2.8723
θ_v	0.3	$\tau_{w\infty}$	0.07	τ_{v1}^-	60	τ_w^+	280	k_s	2.0994	w_∞^*	0.94	θ_w	0.015
u_s	0.9087	θ_v^-	0.015	k_w^-	65	θ_o	0.006	u_w^-	0.03	k_{so}	2	u_{so}	0.65

Table 1. TNNP model parameter values for the BOCF model [22].

105 Typical snapshots of the four variables during a chaotic evolution are shown in Figure 1. The spatio-
106 temporal chaotic dynamics of this system is actually transient chaos whose lifetime grows exponentially
107 with system size [25, 26]. To obtain chaotic dynamics with a sufficiently long lifetime the system has
108 been simulated on a domain of 512×512 grid points with a grid constant of $\Delta x = 1.0$ space units and
109 a diffusion constant $D = 0.2$. Furthermore, an explicit Euler stepping in time with $\Delta t = 0.1$, a 5 point
110 approximation of the Laplace operator, and no-flux boundary conditions were used for solving the PDEs.

111

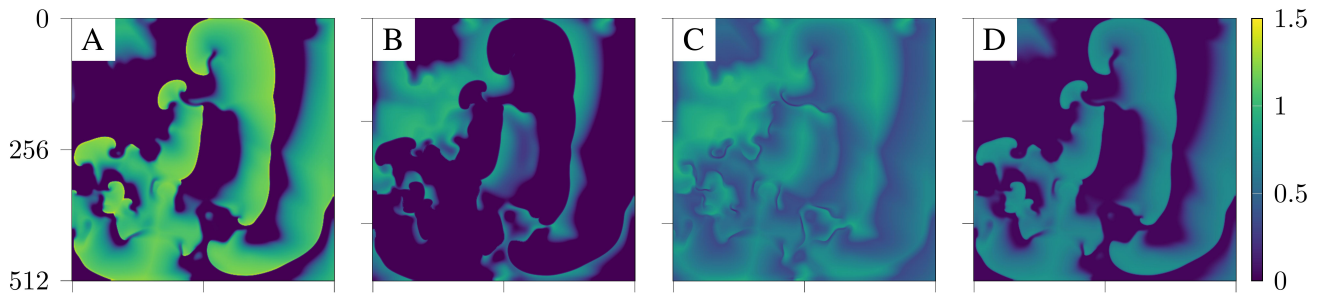


Figure 1. Snapshots of the four fields u, v, w, s of the BOCF model Eq. (1) (from left to right).

112 2.1.2 Reconstruction tasks

113 Experimental measurements of the dynamics of a system of interest often allow only the observation of
 114 some state variables (e.g., the membrane voltage) and may provide only incomplete or distorted information
 115 about the measured observable. Typical limitations are (additive) measurement noise and low-spatial
 116 resolution (due to the experimental conditions and/or the available hardware). Formally, measurements
 117 impaired due to noise, blurring or undersampling can be described as follows: Let $\mathbf{X}_n \in \mathbb{R}^{r \times c}$ be the
 118 measured data (here: snapshots of the field u) where r and c specify the two spatial dimensions. Each sample
 119 \mathbf{X}_n with $n = 1, \dots, N$ corresponds to a true system output $\mathbf{X}'_n \in \mathbb{R}^{r' \times c'}$ that is assumed to be known only
 120 during the training phase in terms of a training set $\mathcal{D} = \{\mathbf{Z}_1 = (\mathbf{X}_1, \mathbf{X}'_1), \dots, \mathbf{Z}_N = (\mathbf{X}_N, \mathbf{X}'_N)\}$. Note
 121 that with coarse graining $r \leq r'$ and $c \leq c'$. The task is to predict the true system output \mathbf{X}' from impaired
 122 observations \mathbf{X} which belong to one of the following three cases:

- 123 1. *Noisy data:* To add noise each element of \mathbf{X}' is replaced with probability p by 0 or 1 drawn from
 124 a Bernoulli distribution $\mathcal{B}(0.5)$ (note that in our case \mathbf{X}' is given by the variable u of the BOCF
 125 model which has a range of $[0, 1]$). To simulate different levels of noise different probabilities $p =$
 126 $0.1, 0.2, \dots, 0.9$ are used to generate noisy data sets $\{\mathbf{X}_n\}$. In the following p is called the noise level.
- 127 2. *Blurred data:* Data with reduced spatial resolution are obtained as Fourier low-pass filtered data
 128 $\mathbf{X} = \mathcal{F}^{-1}(\mathcal{P}^m(\mathcal{F}(\mathbf{X}')))$ where \mathcal{F} and \mathcal{F}^{-1} denote the Fourier transform and its inverse, respectively,
 129 and \mathcal{P}^m is a projection where frequencies outside a radius $m \in [2, 4, 8, \dots, 18]$ (Manhattan distance)
 130 centered at frequency zero are set to zero.
- 131 3. *Undersampled data:* To generate undersampled data \mathbf{X}' is down-sampled $\mathbb{R}^{r' \times c'} \rightarrow \mathbb{R}^{r \times c}$ with $r < r'$
 132 and $c < c'$ by accessing every 2^i -th value of \mathbf{X}' , where $i \in [1, 7]$.

Figure 2 shows examples of the three types of impaired observations.

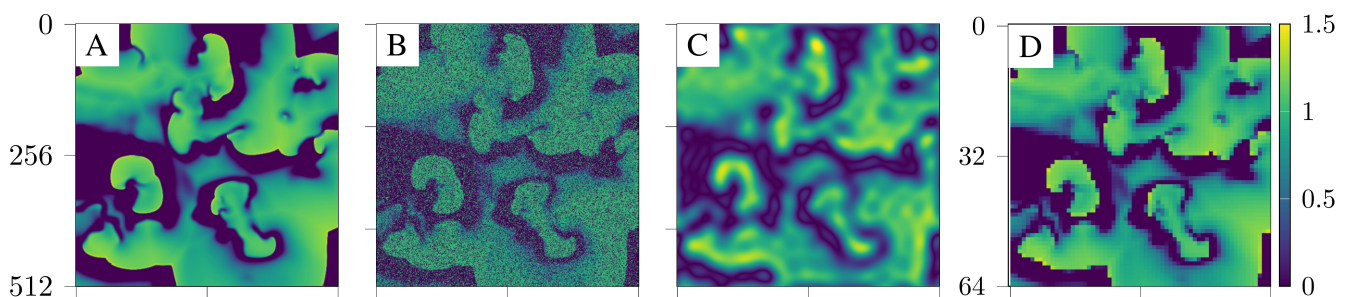


Figure 2. Snapshots of the three cases of impaired data based on u (from left to right: **A:** reference data u , **B:** noisy data, **C:** blurred data, and **D:** undersampled data.)

133 2.2 Predicting electrical excitation from mechanical contraction

134 To learn the relation between mechanical deformation and electrical excitation *inverse modelling data*
 135 were generated by a conceptual electro-mechanical model consisting of an Aliev-Panfilov model describing
 136 the electrical activity and a driven mass-spring-system [15].

137

138 2.2.1 Aliev-Panfilov Model

Specifically developed to mimic cardiac action potential in the myocardium, the Aliev-Panfilov model is a modification of the FitzHugh-Nagumo model, which reproduces the characteristic shape of electric pulses occurring in the heart [27]. It is given by a set of two differential equations,

$$\frac{\partial u}{\partial t} = \nabla(\mathbf{D} \cdot \nabla u) - ku(u - a)(u - 1) - uv \quad (4)$$

$$\frac{\partial v}{\partial t} = \epsilon(u, v) \cdot (-v - ku(u - b - 1)) \quad (5)$$

$$\epsilon(u, v) = \epsilon_0 + \frac{\mu_1 v}{\mu_2 + u}$$

in which u and v are the normalised membrane voltage and the recovery variable, respectively, and a , b and k are model parameters. The term $\nabla(\mathbf{D} \cdot \nabla u)$ accounts for the diffusion, in which the tensor \mathbf{D} can be used to model anisotropies in the myocardial tissue. In addition, the term $\epsilon(u, v)$ is introduced to adjust the shape of the restitution curve by modulating the parameters μ_1 and μ_2 . The computational advantage of the Aliev-Panfilov model lies in its simplicity over other ion-flow-based models which allows shorter runtimes and combined with the elastomechanical model, keeps computational costs fairly reasonable. For this reason, the Aliev-Panfilov model was chosen for generating synthetic data from complex chaotic electromechanical wave dynamics.

Within the heart muscle, the myocardium, cells contract upon electrical excitation through a passing action potential. At this point it is important to note that muscle fibre contracts along its principal orientation which has to be considered during the implementation of the mechanical part of the simulation. To couple the mechanical contraction of the muscle fibre to electrical excitation of a cell, as an extension to the Aliev-Panfilov model the *active stress* T_a was introduced by Nash and Panfilov [28] which leads to contraction in the principal orientation of the muscle fibre. The change of the active stress is described by

$$\frac{\partial T_a}{\partial t} = \epsilon_T(u) \cdot (k_T u - T_a), \quad (6)$$

where k_T controls the strength of the build-up of active stress. The term $\epsilon(u)$ regulates the influence of u on T_a for large u . In our simulations we use a smooth function introduced by Göktepe and Kuhl [29] given by

$$\epsilon_T(u) = \epsilon_{T,0} + (\epsilon_\infty - \epsilon_{T,0}) \cdot \exp(-\exp(-\xi_T \cdot (u - u_0))). \quad (7)$$

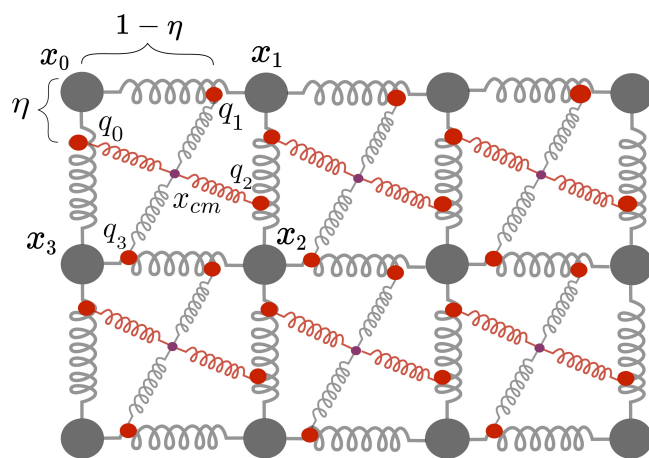
139 Here, ξ_T controls the steepness of the transition between ϵ_∞ and $\epsilon_{T,0}$ and u_0 denotes the potential threshold
 140 for the activation of the active stress, with $\epsilon_\infty < \epsilon_{T,0}$ to achieve a physiological time course [30].

141 2.2.2 Mass-Spring Damper System

142 The elasto-mechanical properties of the cardiac muscle fibre were implemented using a modified
 143 two-dimensional mass-spring damper system [31]. For the current study the mass-spring system was

144 implemented in two dimensions because this allows shorter runtimes and primarily serves as a proof-of-
 145 principle for the evaluated reconstruction approach. In its two-dimensional form this mechanical model
 146 might correspond best to a cut-out of the atrium’s wall, since there the muscle tissue is less than 4 mm
 147 thick. However, this mass-spring system can easily be expanded to three dimensions (see [15]).
 148 Placed on a regular lattice, one mechanical cell is made up of four particles x_i at the corners connected by
 149 *structural springs* and two sets of orthogonal springs connecting the centre of mass \vec{x}_{cm} to each side of the
 150 cell (see Figure 3). The springs in the middle of the cell are called *axial springs*, of which one is made to
 151 be active (red). Here it is important to point out that one cell in the electrical model corresponds to one cell
 152 in the mechanical mass-spring system. For setting the fibre orientation through the active axial spring, the
 153 orientation parameter $\eta \in [0, 1]$ has been introduced, with which the four points of attachment q_i can be
 154 computed easily. This parameter can be set individually for each cell, so that various fibre orientations can
 be modelled.

Figure 3. Two dimensional mass-spring damper system with one active spring modelling fibre orientation (red) and one passive spring (grey), the centre of mass x_{cm} , the four points of attachment q_i to the structural springs and the orientation parameter η .



155

Using $\vec{x}_{cm} = \frac{1}{4} \sum_{i=0}^3 \vec{x}_i$ the forces from the passive spring \vec{f}_j and the active spring \vec{f}_a are obtained as

$$\vec{f}_j = -k_j (\|\vec{q}_j - \vec{x}_{cm}\| - l_{j,0}) \cdot \vec{e}_j, \tag{8}$$

$$\vec{f}_a = -k_a \left(\|\vec{q}_a - \vec{x}_{cm}\| - \frac{l_{a,0}}{1 + c_a \cdot T_a} \right) \cdot \vec{e}_a. \tag{9}$$

Here $l_{j,0}$, k_j and $l_{a,0}$, k_a denote the resting lengths and spring constants of the passive and active spring, respectively. From Equation (9) it can be seen that, upon a rise in active stress T_a from Equation (6), the active spring contracts and an inward force is generated. The parameter c_a represents a scaling factor to modulate the influence of the active stress. Through the orientation parameter the forces from the active and passive spring can be redistributed to the corresponding particles at the corners. For example for \vec{q}_1 , the force on x_0 would be $\vec{f}_0 = \eta \vec{f}_{q_1}$ and on x_1 it would amount to $\vec{f}_1 = (1 - \eta) \vec{f}_{q_1}$. In addition, the mechanical grid is held together by structural forces between the corner particles, which can be computed using

$$\vec{f}_{ij} = -k_{ij} (\|\vec{x}_i - \vec{x}_j\| - l_{ij}) \cdot \vec{e}_{ij}, \tag{10}$$

$$\vec{f}_{ji} = -\vec{f}_{ij}, \tag{11}$$

with l_{ij} being the resting length between particle x_i and x_j .

Finally, with all the above forces acting on particle x_i with mass m_i , its motion is determined according to

$$m_i \frac{d^2 \vec{x}_i}{dt^2} = \sum_{\{a\}\{j\}\{ij\}} \vec{f}_k - \nu \frac{d\vec{x}_i}{dt}, \quad (12)$$

with the sum $\sum_{\{a\}\{j\}\{ij\}} \vec{f}_k$ of all relevant springs pulling or pushing the particle. The damping constant ν sets the strength of the damping to increase the stability of the mechanical system as a whole.

The area of each cell was calculated with a simple formula for a general quadrilateral using the positions of its four corners. As a measure of contraction, the relative change of area

$$\Delta A(t) = \frac{A(t)}{A_{\text{undeformed}}} - 1 \quad (13)$$

156 has been used. The numerical algorithm for solving the full set of electro-mechanical ODEs is summarized
157 in the appendix.

158 2.2.3 Reconstruction task

159 The inverse modelling data are generated by forward modelling $M : u \mapsto \Delta A$ using the output of
160 Equations (4) and (13). The task is to train an ESN or CAE to approximate $M^{-1} : \Delta A \mapsto u$. To fulfill
161 this task we use the membrane voltages and the local deformations at all $r \times c$ grid points sampled at
162 times t_n . The training data set $\mathcal{D} = \{\mathbf{Z}_1 = (\mathbf{X}_1, \mathbf{X}'_1), \dots, \mathbf{Z}_N = (\mathbf{X}_N, \mathbf{X}'_N)\}$ thus consists of snapshots
163 $\mathbf{X}_n \in \mathbb{R}^{r \times c}$ and $\mathbf{X}'_n \in \mathbb{R}^{r \times c}$ of the relative mechanical deformation $\Delta A(t_n)$ and the membrane voltage
164 $u(t_n)$, respectively, and we aim at approximating $M^{-1} : \mathbf{X}_n \mapsto \mathbf{X}'_n$ with $r, c = 100$.

165 2.3 Machine Learning Methods

166 In this section we will introduce the two machine learning approaches, echo state networks (ESN) [32]
167 and convolutional autoencoders (CAE) [33], that will be applied to solve the reconstruction tasks defined
168 in section 2.1.2 and section 2.1.2.

169 2.3.1 Echo State Network

170 Echo state networks have been introduced in 2001 by H. Jaeger [32] as a simplified type of recurrent
171 neural network, in which the weights describing the strength of the connections within the network are
172 fixed. In its general composition an ESN subdivides into three sections [32], as illustrated in Figure 4.
173 First of all, there is the input layer into which the input signal $\vec{u}_n \in \mathbb{R}^{N_u}$ and a constant bias b_{in} are
174 fed. Secondly, the intermediate reservoir consists of N nonlinear units and its state is given by $\vec{s}_n \in \mathbb{R}^N$.
175 And lastly, the output layer provides the output signal $\vec{y}_n \in \mathbb{R}^{N_y}$. Here, n denotes the discrete time steps
176 $n = 1, \dots, T$.

177

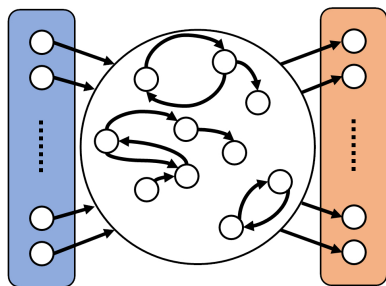


Figure 4. Schematic representation of an ESN. On the left side (colored in blue) is the input layer where the input signal \vec{u}_n and a constant bias b_{in} are fed in. The reservoir is represented as the large circle in the middle, where the small circles are the nodes. The output layer on the right (colored in orange) provides the reservoir signals \vec{s}_n that are part of the vector $\vec{x}_n = [b_{out}; \vec{s}_n; \vec{u}_n]$ used for computing the output $\vec{y}_n = \mathbf{W}_{out} \vec{x}_n$.

The concatenated bias-input vector $[b_{in}; \vec{u}_n]$ is fed into to the reservoir through the input matrix $\mathbf{W}_{in} \in \mathbb{R}^{N \times (1+N_u)}$. Inside the reservoir connections are given by the weight matrix $\mathbf{W} \in \mathbb{R}^{N \times N}$, where N is the reservoir size. Together with the input matrix it is possible to determine the state of the reservoir at time n through the update rule

$$\vec{s}_n = (1 - \alpha)\vec{s}_{n-1} + \alpha f_{in}(\mathbf{W}_{in}[b_{in}; \vec{u}_n] + \mathbf{W}\vec{s}_n), \quad (14)$$

in which $[\cdot; \cdot]$ denotes a concatenated vector. The input bias b_{in} , as well as the later introduced output bias b_{out} were both set to 1 in the following. The parameter $\alpha \in (0, 1]$ in Equation (14) represents the *leaking rate* which controls how much of a neuron's activation is carried over to the next time step and can be used as a parameter to enhance predictions. As for the transfer function $f_{in}(\cdot)$ we use $\tanh(\cdot)$ and the network dynamics has no feedback loop. Only the weights \mathbf{W}_{out} providing the output signal

$$\vec{y}_n = \mathbf{W}_{out}\vec{x}_n \quad \text{with} \quad \vec{x}_n = [b_{out}; \vec{s}_n; \vec{u}_n] \quad (15)$$

are adapted during the training process by minimizing the cost function [34]

$$C(\mathbf{W}_{out}) = \sum_n \|\vec{y}_n^{true} - \mathbf{W}_{out}\vec{x}_n\|^2 + \lambda \text{Tr}(\mathbf{W}_{out}\mathbf{W}_{out}^T) \quad (16)$$

178 where λ controls the impact of the regularization term that prevents overfitting [35]. The final output matrix
179 is given by the minimum of the cost function at $\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T + \lambda\mathbf{1})^{-1}$ where \mathbf{X} and \mathbf{Y} are matrices
180 whose columns are given by the vectors \vec{x}_n and \vec{y}_n^{true} , respectively.

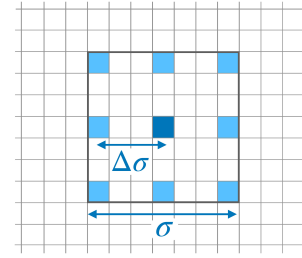
181 Both matrices \mathbf{W}_{in} and \mathbf{W} , are initialised with random values from the interval $[-0.5, 0.5]$. Since in
182 experiments it turned out that more diverse dynamics could be modelled using networks in which only a
183 small percentage ϵ of weights inside the reservoir remained non-zero [32], the weight matrix \mathbf{W} is made
184 sparse with only a portion ϵ of its values remaining non-zero. Furthermore, it is scaled by a factor $\frac{\rho}{|\mu_{max}|}$
185 where $|\mu_{max}|$ denotes here the largest eigenvalue of \mathbf{W} and ρ is a hyperparameter for optimizing the
186 performance (by ensuring the so-called echo state property [36]). To reduce the probability of drawing an
187 dysfunctional set of matrix entries the randomly generated matrices \mathbf{W}_{in} and \mathbf{W} were selected from four
188 different realisations. To optimize the performance of the ESN five hyperparameters (N , ϵ , ρ , α and λ) are
189 tuned.

190 Reservoir computing using ESNs for predicting chaotic dynamics has already been demonstrated in 2004
191 by Jaeger and Haas [37]. Since then many studies appeared analyzing and optimizing this approach (see,
192 for example, [38, 39, 40, 41, 42, 43, 44] and references cited therein). In particular, it has been pointed
193 out how reservoir computing exploits generalized synchronization of uni-directionally coupled systems
194 [45, 46].

195 Recently, applications of ESNs to spatio-temporal time series have been presented [47, 18] employing
196 many networks operating in parallel at different spatial locations based on the concept of (reconstructed)
197 local states [48]. In particular, using this mode of reservoir computing it was possible to perform a cross-
198 prediction between the four different variables of the BOCF model [18]. Therefore, for the current task
199 of reconstructing data from impaired observations we build on the previous ESN design and modelling
200 procedure. For each pixel an ESN is trained receiving input from neighbouring pixels, only, representing
201 the local state at the location of the reference pixel as illustrated in Figure 5. This design introduces two
202 new hyperparameters σ and $\Delta\sigma$ to the default ESN, where σ is the size of the stencil to define the local

203 state and $\Delta\sigma$ specifies the spatial distance of adjacent pixels included in the local state. Optimal values for
 204 all hyperparameters are determined by a grid search.

Figure 5. Stencil for locally sampling data used as input of the ESN operating at the location of the dark blue pixel in the center. The stencil is characterized by its width σ and the spatial separation $\Delta\sigma$ of sampling points.



205 2.3.2 Convolutional autoencoder

206 A convolutional autoencoder [33] is a special architecture of a feed forward network (FFN) with
 207 convolutional layers similar to convolutional neural networks (CNNs) [49]. Generally a CAE learns a
 208 representation of the training set \mathcal{D} with the purpose of dimensionality reduction. For each pair $\mathbf{Z}_i =$
 209 $(\mathbf{X}_i, \mathbf{X}'_i) \in \mathcal{D}$ the CAE is trained to perform a nonlinear transformation from the input representation of
 210 \mathbf{X}_i to the output representation of \mathbf{X}'_i . Like CNNs a CAE is a partially locally connected feed forward
 211 network, which is typically composed of the following layers:

- 212 • Convolutional layers: Convolution of the input by a kernel sliding over the input. The number of rows
 213 and columns of the kernel are hyper-parameters, in this work they are set to be 3×3 .
- 214 • Batch normalization layer: Normalization of the activations of the previous layer during training and
 215 for each batch. Batch normalization allows the use of higher learning rates, being computationally
 216 more efficient, and also acts as a regularizer [50].
- Leaky ReLU [51] layer: Leaky version of a rectified linear unit (ReLU) [52], such that:

$$\nu(x) = \begin{cases} \alpha x & \text{for } x < 0 \\ x & \text{for } x \geq 0. \end{cases}$$

- 217 • Max pooling layer: Sample-based operation for discretization based on a kernel that slides over the
 218 input like the convolutional operator but only the maximum value of the kernel is passed to the next
 219 layer. Width and height of the kernel are hyper-parameters (in this work 2×2). In contrast to the
 220 convolutional layer a pooling layer is not trainable.
- 221 • Dropout layer: Regularization method to prevent overfitting where during training some weights are
 222 set randomly to zero [53]. In this work the probability of setting the weights to zero is 0.05.

223 The eponymous part of the CAEs are the convolutional layers, a convolution of $A = (a_{ij}) \in K^{n \times n}$ with a
 224 kernel $F = (f_{ij}) \in K^{k \times k}$, where $k < n$, is given by:

$$(A * F)_{xy} = \sum_{i=\lfloor -k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{j=\lfloor -k/2 \rfloor}^{\lfloor k/2 \rfloor} a_{xy} f_{(i-x)(j-y)}, \quad (17)$$

225 with $x, y \in 1, \dots, n$. If $i - x -$ or $j - y$ exceeds the range of A zero-padding is applied [54].
 226 In this work two architectures are used. The first one employed to reconstruct the data from noisy, blurred
 227 and inverse modelling data is illustrated in Figure 6. The architecture is the same for the tasks in section 2.1.2

228 and section 2.2.3 but the sizes of \mathbf{X} and \mathbf{X}' are different, with $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{512 \times 512}$ and $\mathbf{X}, \mathbf{X}' \in \mathbb{R}^{100 \times 100}$,
 229 respectively. Due to the smaller input size, the data for the inverse modelling reconstruction is transformed
 230 into a latent space with the size of 25×25 . The second architecture is sketched in Figure 7 and deals with
 231 the undersampled data reconstruction.

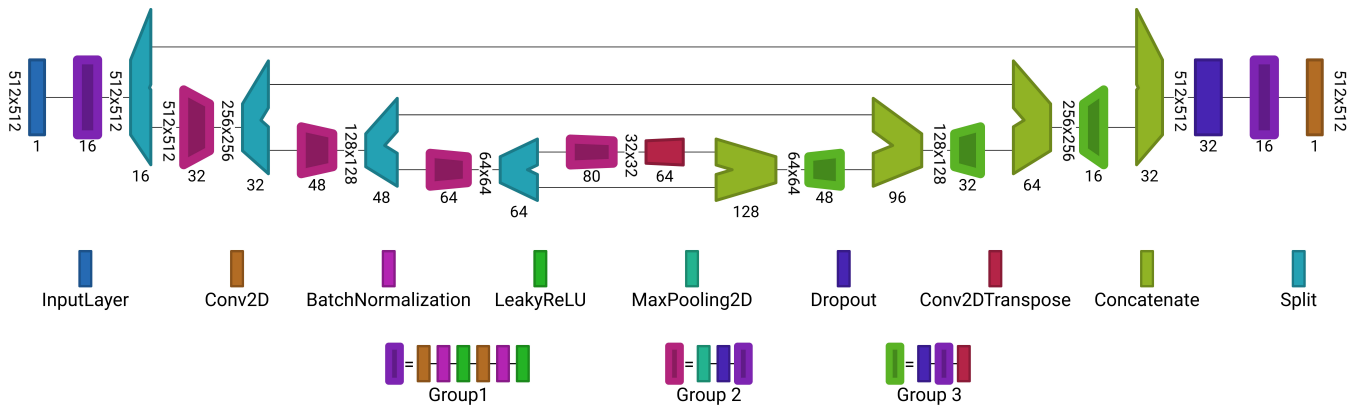


Figure 6. Proposed autoencoder architecture for reconstruction of data from noisy or blurred input. Each block is a set of layers. The values written vertically describe the dimension of the input for each layer, e.g.: for noisy and blurred data $r = 512, c = 512$ and for the inverse modelling data $r = 100, c = 100$. The horizontally written values at the layers are the number of channels or number of filters. Group 1 is a combination of layers, consisting of: Conv2D, BatchNormalization, LeakyReLU, Conv2D, BatchNormalization and LeakyReLU layers. Group 2 is an extension of Group 1 where a MaxPooling2D and Dropout layer are placed before Group 1. Similar applies to Group 3, it consists of a Dropout layer followed by the layers from Group 1 and finalized by a Conv2DTranspose layer follows. The architecture was visualized with *Net2Vis* [55].

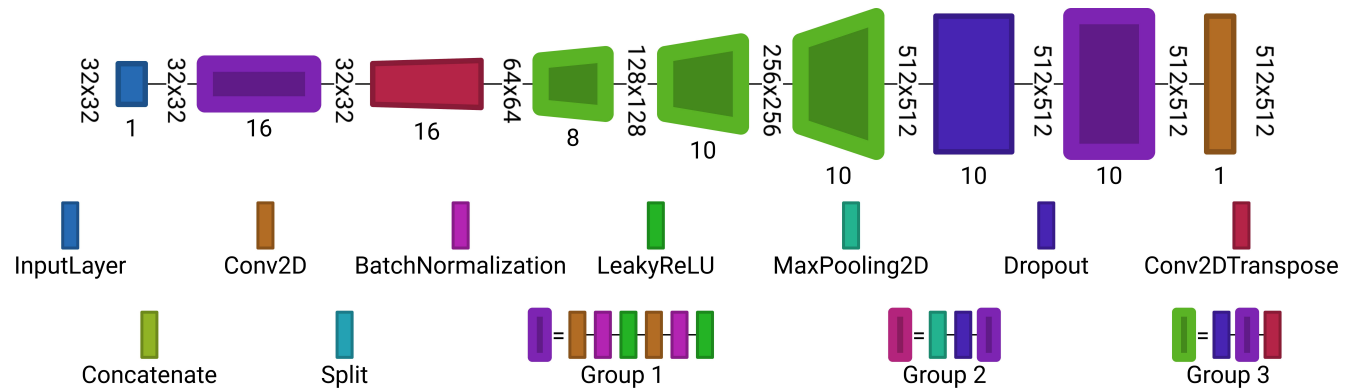


Figure 7. Autoencoder architecture used for reconstruction from undersampled observations. Each block is a set of layers. The layer labeling is the same as in Figure 6. Visualized with *Net2Vis* [55].

3 RESULTS

232 In the following both machine learning methods will be applied to two tasks: (i) Reconstructing electrical
 233 excitation waves from noisy blurred and under sampled data (section 3.1) and (ii) Predicting electrical
 234 excitation from mechanical contraction (section 3.2).

235

236 3.1 Recovering complex spatio-temporal wave patterns from impaired observations

237 To benchmark both reconstruction methods, using ESNs and CAEs, we use time series generated by
 238 the BOCF model introduced in section 2.1.1. The same data were used for both methods, consisting of
 239 5002 samples in the training data set, 2501 samples in the validation data set, and 2497 samples in the
 240 test data set. The sampling time of all time series equalled $10\Delta t = 1$. We considered nine cases of noisy
 241 data (with different noise levels), ten cases of (differently) blurred data and seven examples of (spatially)
 242 undersampled time series.

243 To determine the optimal ESN hyperparameters a grid search is performed as described in [18] using the
 244 training and validation subsets of the data. This search consists of two stages: first, for each combination
 245 of the local states' hyperparameters σ and $\Delta\sigma$ as listed in Table 2 a grid search is performed to find the
 246 optimal five hyperparameters of the ESN resulting in 37 sets of optimal hyperparameters. To make these
 247 grid searches more feasible, they were performed just for a single input patch (area covered by the stencil,
 248 see Fig. 5) in the spatial center of the training set and thus not use the full spatial data.

σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$	σ	$\Delta\sigma$
25	2	29	2	33	2	37	2	41	2	45	2	49	4	101	10
25	4	29	7	33	8	37	4	41	4	45	4	49	8	101	20
25	8	29	14	33	16	37	9	41	8	45	11	49	16	101	25
25	24	29	28	33	32	37	12	41	20	45	22	49	24	101	50
						37	18	41	40	45	44	49	48		
						37	36								

Table 2. The examined set of hyperparameters σ and $\Delta\sigma$ for the local states.

249 In the second stage, for each of the 37 sets of optimal hyperparameters determined before (for each
 250 combination of σ and $\Delta\sigma$), an ESN is trained on a larger subset of the training data and not just on a single
 251 patch. Ideally, this step should be performed on the entire spatial domain of the training set, however, as we
 252 did not notice significant differences in the results when the ESNs were trained on a spatial subset of size
 253 250×250 to speed up the training process. Following the same methodology as in [18], for each pixel
 254 from this spatial subset a single ESN is trained and then the obtained output matrices \mathbf{W}_{out} of these ESNs
 255 are averaged over all pixels. Compared to the procedure used in [18], the handling of boundary values has
 256 been changed. As for boundary pixels fewer adjacent pixels exist than for those inside, the creation of local
 257 states is obstructed, and boundary pixels require special treatment. In our previous work [18] individual
 258 ESNs have been trained for the boundary pixels using local states of lower dimensionality. In the following
 259 we use an alternative approach based on padding the boundary pixels by mirroring their values (motivated
 260 by the no-flux boundary conditions used). In this way, local states can be formally defined for boundary
 261 pixels in the same way as for inner pixels.

262 Next, the different optimal ESNs obtained for different stencils ($\sigma, \Delta\sigma$) were evaluated by comparing their
 263 performance on the validation subset. In this way optimal values for σ and $\Delta\sigma$ were selected by choosing
 264 the combination ($\sigma, \Delta\sigma$) with the lowest ℓ_2 difference between the prediction and ground truth on the
 265 validation set. This process yields an ESN whose hyperparameters and weights are optimized to yield
 266 minimal ℓ_2 error. Finally, without training the network again on the entire training set, the optimal ESN
 267 found before is used to perform the prediction on the entire test set. As a pre-processing step, both the input
 268 and target data of the training, validation and test set are rescaled with min-max scaling, where the minimal
 269 and maximal value are determined over all pixels of the training set.

270 The CAE was trained using the ADAM optimizer [56], implemented with *TensorFlow* [57] in version 2.3,
 271 with early stopping when the validation loss has not improved at least by 10^{-6} for 20 epochs. The learning

272 rate was reduced by a factor of 0.2 when the loss metric stopped improving at least by 10^{-5} for ten epochs.
 273 Dropout was set to be 0.05 in all cases. As loss function the mean absolute error (MAE) was chosen:

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^N \left| \hat{\mathbf{X}}_i - \mathbf{X}'_i \right|, \quad (18)$$

274 where N is the number of elements in the data set, $\hat{\mathbf{X}}_i$ the network output, \mathbf{X}'_i desired ground-truth and $|\cdot|$
 275 stands for the absolute values.

276 3.1.1 Noisy data

278 Figure 8 shows snapshots of the noisy input data, the corresponding ground truth, the outputs provided
 279 by the CAE and the ESN, respectively, and the absolute values of their prediction errors with respect to
 the ground truth. The evolution of the loss function during the training epochs is shown in Figure 9. In

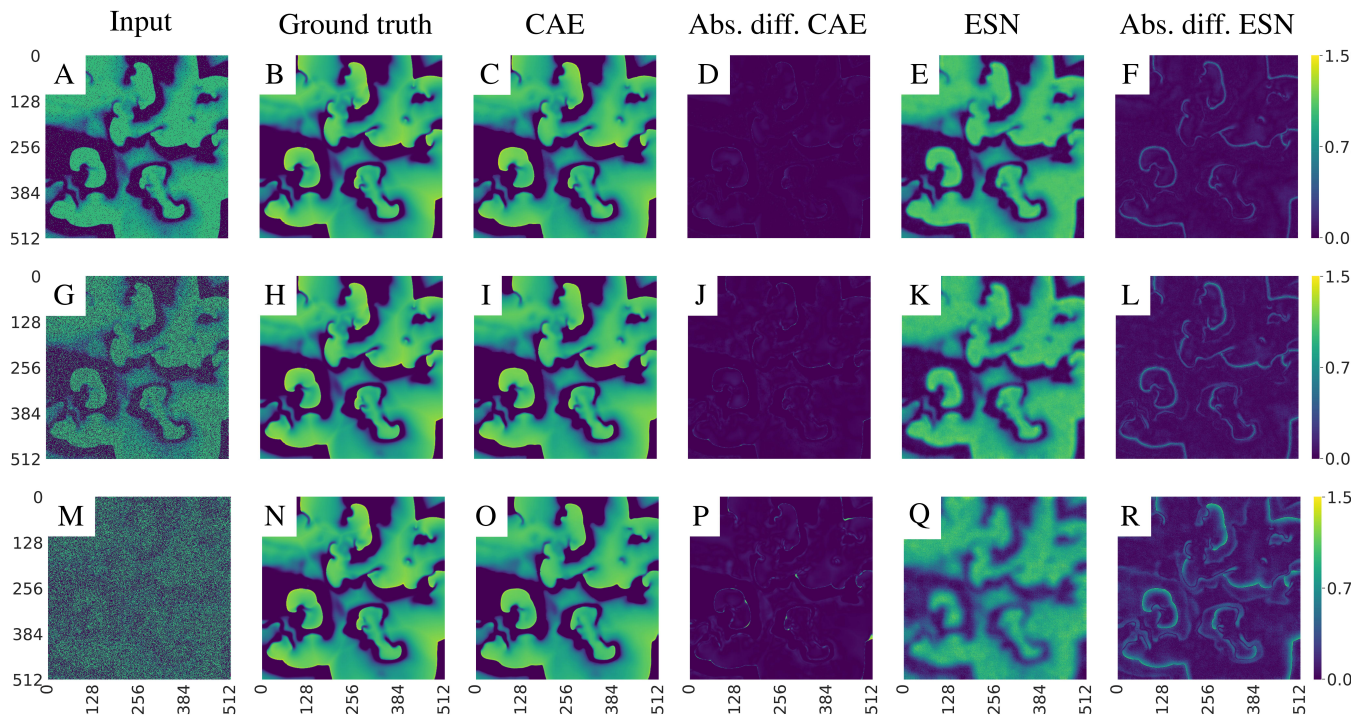


Figure 8. Exemplary visualization of the input and output for both networks for data with different noise levels p : **A-F** $p = 0.1$, **G-L** $p = 0.5$, and **M-R** $p = 0.9$. Comparing the absolute differences between the prediction and the ground truth (**D,J,P** for the CAE and **F,L,R** for the ESN) one can see that the CAE is less sensitive to noise. Note that the errors develop primarily on the fronts of the waves.

280 all cases the error decreases and the training converges, but the duration of the training depends on the
 281 complexity of the case.
 282

283 Figures 10 shows a comparison of the performance of the CAE and the ESN for noisy data with nine
 284 different noise levels $p = 0.1, \dots, 0.9$. While the mean absolute error of the CAE remains below 0.02, the
 285 reconstruction error of the ESN increases from 0.06 for $p = 0.1$ to 0.18 for $p = 0.9$, the associated ESN
 286 hyperparameter can be found in the appendix (Table 5).

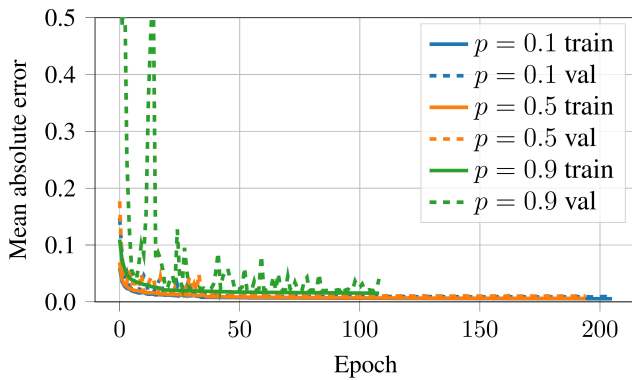


Figure 9. Evolution of the loss function values over the epochs for noisy input data generated with noise levels $p = 0.1$, $p = 0.5$, and $p = 0.9$ (compare Fig. 8). It can be seen that the training always ran up to the point where early stopping, as defined in section 3.1, terminated it. The solid lines are the values of the loss function during training on the training data, while the dotted lines are the values of the loss function obtained when the trained model is applied to the validation data. One epoch trained approximately 110 seconds on a GTX 1080 Ti.

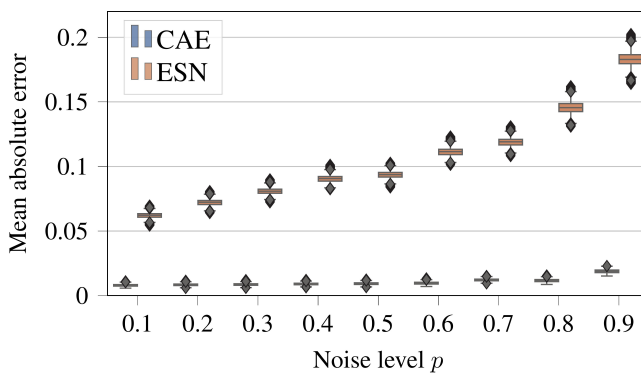


Figure 10. Comparison between CAE and ESN performance with noisy input data showing boxplots of mean absolute errors (18) for different noise levels $p \in [0.1, 0.2, \dots, 0.9]$. Each discrete value on the x-axis is assigned to the boxes of the CAE and ESN, where the ESN boxplots are coloured in orange and the CNN boxplots are coloured in blue. Note that for better visibility the CAE boxes and the ESN boxes a slightly shifted to the left and to the right, respectively. A tabular overview of the values can be found in the appendix (Table 4).

287 3.1.2 Blurred data

288 To evaluate the performance of CAE and ESN for recovering full resolution (ground truth) data from
 289 blurred observations we consider nine cases where the radius m of Fourier low-pass filtering ranges from
 290 $m = 2$ to $m = 18$ (in steps of 2). Figure 11 shows snapshots of reconstructions of the u -variable of the
 291 BOCF model using CAE and ESN with filter parameters $m = 20$ (A-F), $m = 14$ (G-L), and $m = 8$ (M-R).
 292 Similar to Figure 8 the errors are largest at fronts of the excitation waves, but in contrast to noisy images
 293 the performances of CAE and ESN differ not much for blurred data. This observation is also confirmed by
 294 a systematic comparison of the mean absolute errors of both methods for different manhattan distances m
 295 given in Figure 12. The errors decrease with m because the larger m the less blurred are the input data of
 296 the CAE or ESN (for hyperparameter see appendix (Table 6)). Figure 13 shows the evolution of the loss
 297 function during training of the CAE.

298 299 3.1.3 Undersampled data

300 Figure 14 shows examples of data reconstructed from undersampled data. In total we considered seven
 301 cases of undersampling by 2^i pixels, where i ranges from $i = 1$ to $i = 7$. For $i = 1$ input images have a
 302 resolution of $\mathbf{X} \in \mathbb{R}^{256 \times 256}$ and for $i = 7$, $\mathbf{X} \in \mathbb{R}^{4 \times 4}$. In all cases the desired output (ground truth) \mathbf{X}'
 303 has a size of 512×512 pixels. The used hyperparameter for the ESN can be found in the appendix (Table 7).
 304

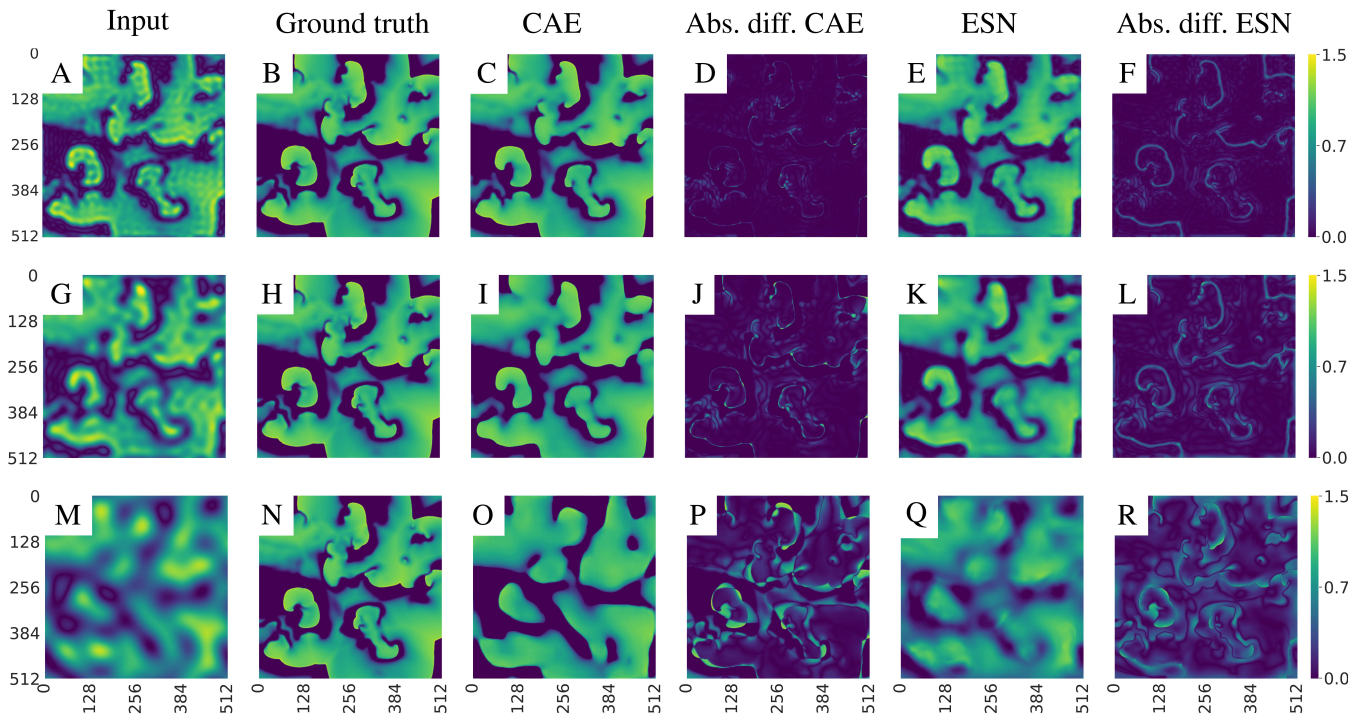


Figure 11. Exemplary visualization of the input and output for both networks, CAE and ESN, when recovering the original data from blurred measurements. **A-F** corresponds to case one, where $m = 20$, **G-L** to case six with $m = 14$ and **M-R** to seven, $m = 8$. Comparing the absolute differences between the prediction and the ground truth (**D,J,P** for the CNN and **F,L,R** for the ESN) one recognizes that the CAE and ESN exhibit different patterns. The errors of the CAE are rather pointwise distributed at some locations (see **D,J**) on the front while they are more evenly distributed when using ESNs (**F,L**). This pattern is even more pronounced in **P** vs **R**.

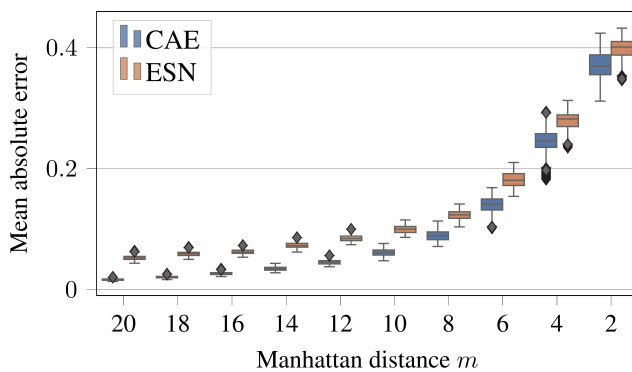


Figure 12. Mean absolute errors of reconstructions using CAE or ESN from data blurred with different values of the low-pass filter parameter m . Like in Figure 10 boxes are horizontally shifted for better visibility. A tabular overview of the values can be found in the appendix (Table 4)

305 **3.2 Predicting electrical excitation from mechanical contraction**

306 Echo state networks as well as convolutional autoencoders have been trained with time series generated by
 307 the electromechanical model introduced in section 2.2 to predict the membrane voltage $u(x, t)$ Equation (4)
 308 from the local contraction $\Delta A(x, t)$ given in Equation (13). The sampling time of all time series equalled
 309 $6\Delta t = 0.48$. Since for periodically rotating spiral waves this cross-prediction task is quite straightforward
 310 we focus here on the much more demanding case of an example exhibiting spatio-temporal chaos
 311 (corresponding to atrial or ventricular fibrillation). Figure 17 shows at three instants of time snapshots
 312 of the observed contraction ΔA (first column), the ground truth of the voltage u (second column), the
 313 prediction of the CAE u_{CAE} and its absolute error $|u - u_{CAE}|$ (third and fourth column) and the prediction

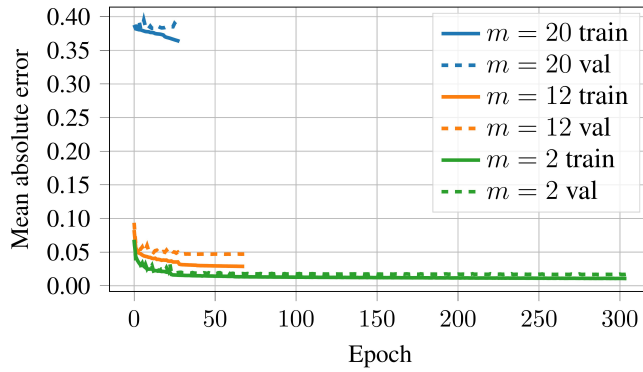


Figure 13. Evolution of the loss function values over the training epochs for blurred data. Training was always terminated by reaching the early stopping criterion, defined in section 3.1. One epoch trained approximately 108 seconds on a GTX 1080 Ti.

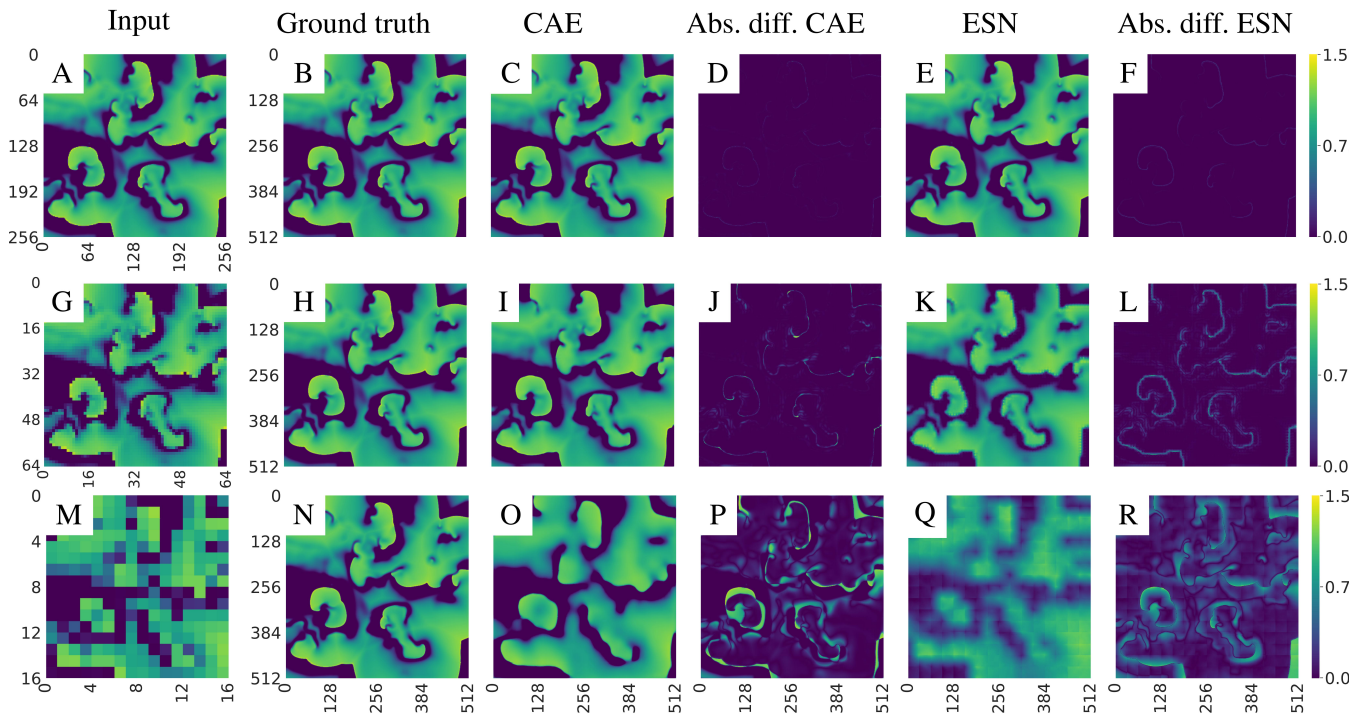


Figure 14. Exemplary visualization of the reconstruction of the u -field of the BOCF model from undersampled data. **A-F** corresponds to a sampling parameter $i = 1$ resulting in $\mathbf{X} \in \mathbb{R}^{256 \times 256}$, **G-L** to case $i = 3$, $\mathbf{X} \in \mathbb{R}^{64 \times 64}$ and **M-R** to $i = 5$, $\mathbf{X} \in \mathbb{R}^{16 \times 16}$. With downsampling by a factor of 2^1 reconstructions by both networks, CAEs and ESNs, are both very successful and the absolute differences shown in **D** and **F** are nearly zero. Similar to reconstructions from noisy or blurred data errors occur mainly at the fronts of the waves, and reconstruction errors of the CAE appear to be more localized compared to ESN results.

314 of the ESN u_{ESN} and the corresponding absolute error $|u - u_{\text{ESN}}|$ (columns five and six, respectively).
 315 Both, the ESN and the CAE were trained and tested with the same spatio-temporal time series (lengths
 316 of training, validation and test sets are 15000, 2000 and 2000 samples). Hyperparameters of the ESN are
 317 $N = 600$, $\alpha = 0.5$, $\rho = 1.1$, $\epsilon = 0.05$, $\lambda = 5 \cdot 10^{-3}$, $\sigma = 7$ and $\Delta\sigma = 1$. To make the ESN more robust
 318 normally distributed noise with zero mean and a variance of 10^{-4} was added to the arguments of the
 319 activation function. As illustrated in Figure 17 both networks can solve the inverse problem and reconstruct
 320 the electrical potential field u from Equation (4). However, the reconstruction of the CAE is more precise,
 321 which is particularly noticeable at the edges of the reconstructed electrical potential field. Considering the

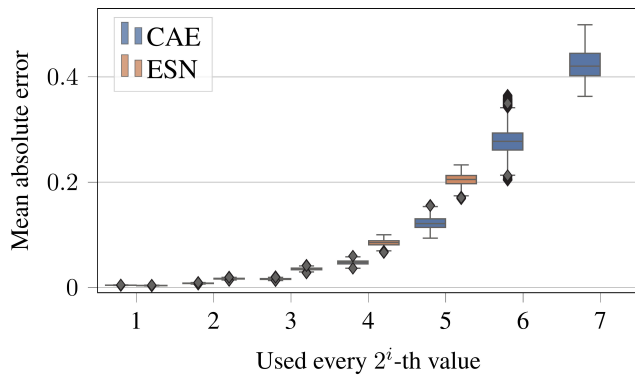


Figure 15. Same as Figure 10 but for the case of the undersampled data. A tabular overview of the values can be found in the appendix (Table 4).

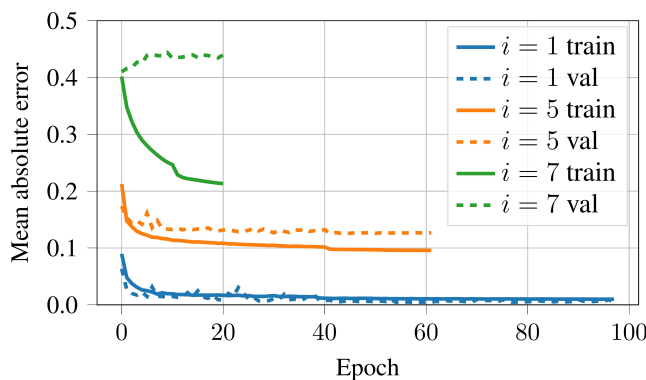


Figure 16. Evolution of the loss function values over the epochs for undersampled input data.

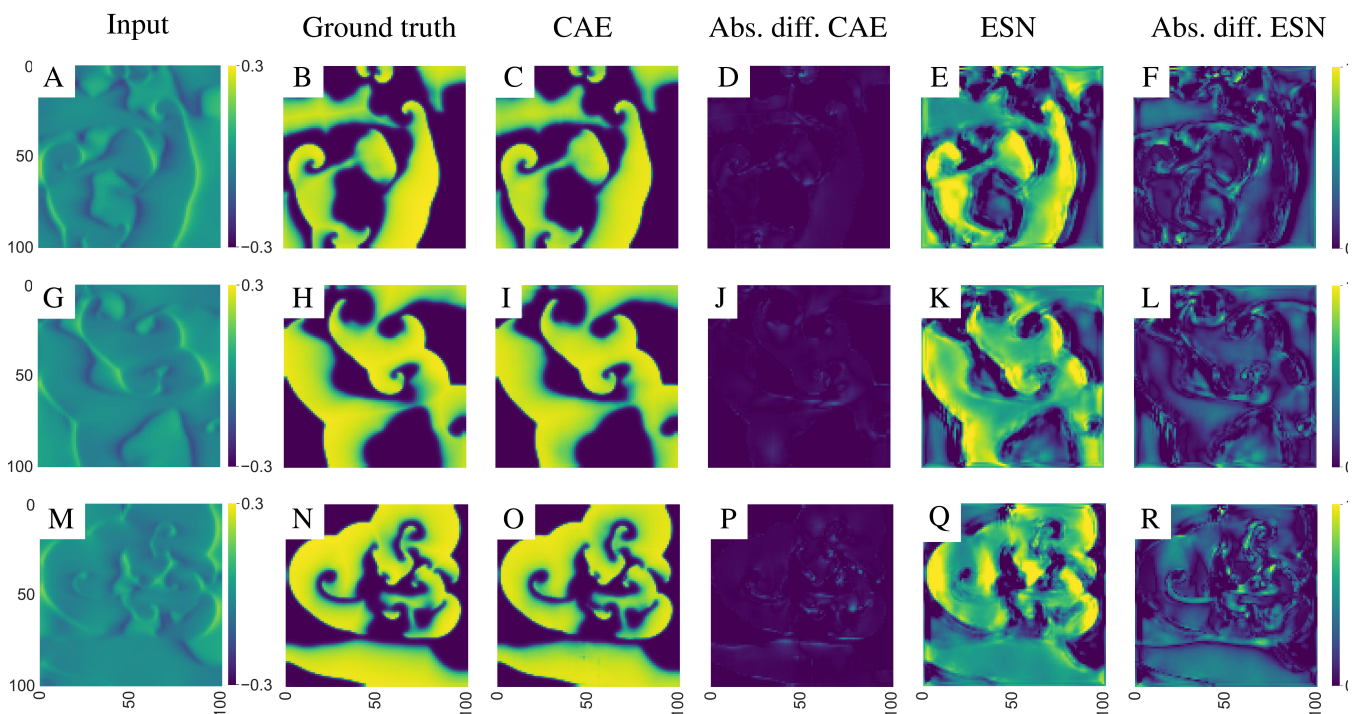


Figure 17. Exemplary visualization of the input and output for both networks for the inverse reconstruction of the membrane voltage *u* (Equation (4)) based on the mechanical deformation ΔA (Equation (13)). **A-F** corresponds to *t* = 1000, **G-L** to *t* = 1500 and **M-R** to *t* = 2000 of the test data set. The input is the mechanical deformation given by Equation (13) caused by the electrical potential *u* from Equation (4).

322 median of the MAE over the entire test data the ESN approach achieves an error of 0.1963 ± 0.0260 while
 323 the median error of the CAE equals 0.0164 ± 0.0028 .

4 CONCLUSION

324 Using synthetic data generated with conceptual models describing complex cardiac dynamics we have
325 demonstrated possible applications of machine learning methods to complete and enhance experimental
326 observations. It was shown that echo state networks as well as convolution autoencoders provide promising
327 results, where the latter turned out to be the method of choice in terms of more faithful reconstructions.
328 At this point, however, we would like to stress, that we didn't try to fully optimize the algorithms
329 employed. One could, for example, increase the size of the ESNs used or extend and refine the grid
330 search of hyperparameters. Also with the CAE several options exist to improve the performance even
331 more. Instead of the MAE in the loss function one could use an adaptive robust loss function [58] or the
332 Jensen-Shannon divergence [19]. The weights of the CAE could be optimized with a stochastic gradient
333 descend approach instead of the ADAM algorithm [56]. But we expect such modifications would show
334 only minor improvements (if at all). In future work, using more realistic numerical simulations (and
335 experimental data) such an optimization should be performed to achieve the best possible result for the
336 intended medical application. Since here we used only data from conceptual models we refrained from
337 fully optimizing the machine learning methods applied. The fact that already a straight-forward application
338 of known algorithms and architectures provided very good results for the considered reconstruction tasks is
339 very promising and encourages to address in future work extended tasks (including other variables, like
340 calcium concentration, mechanical stress and strain, etc.) and reconstruction tasks with more realistic
341 synthetic data (from 3D models, for example) combined with experimental measurements.

CONFLICT OF INTEREST STATEMENT

342 The authors declare that the research was conducted in the absence of any commercial or financial
343 relationships that could be construed as a potential conflict of interest.

AUTHOR CONTRIBUTIONS

344 SH performed the CAE simulations and RSZ and JA did the ESN modelling. All authors designed the
345 study, analyzed the results, and wrote the manuscript.

FUNDING

346 SH acknowledges funding by the International Max Planck Research Schools of Physics of Biological and
347 Complex Systems. UP and SL acknowledge funding by the Max Planck Society, the German Center for
348 Cardiovascular Research (DZHK) partner site Goettingen, and the SFB 1002 Modulatory Units in Heart
349 Failure (project C03).

ACKNOWLEDGMENTS

350 SH and UP thank Florentin Wörgötter and Thomas Lilienkamp for continuous support and inspiring
351 scientific discussions.

DATA AVAILABILITY STATEMENT

352 The datasets generated for this study are available on request to the corresponding author. The
353 source code will be available after publication at [https://gitlab.gwdg.de/sherzog3/
354 reconstructing-cardiac-excitation-waves.git](https://gitlab.gwdg.de/sherzog3/reconstructing-cardiac-excitation-waves.git)

APPENDIX 1: NUMERICAL SOLUTION OF THE ELECTRO-MECHANICAL DYNAMICS

355 The differential equations from the extended Aliev-Panfilov model in Equations (4), (5) and (6) have been
356 integrated using the forward Euler method

$$y_i^{t+\Delta t} = y_i^t + \Delta t \cdot f(y) + \mathcal{O}(\Delta t^2), \quad (19)$$

in which y is a place holder for the model variables u , v and T_a ; $f(y)$ represents the right-hand side of the respective equation. Even though the Euler method shows low accuracy compared to other integration methods, it was chosen because it allows short runtimes, while high accuracy is not extremely important for this simulation. The diffusion tensor \mathbf{D} in Equation (4) was set to a scalar constant D and the diffusion term was approximated with a nine-point stencil [59]

$$\begin{aligned} \nabla(\mathbf{D} \cdot \nabla u_{ij}) = D \cdot \nabla^2 u_{ij} = & \frac{D}{6h^2} (4u_{i+1,j} + 4u_{i-1,j} + 4u_{i,j+1} + 4u_{i,j-1} + u_{i+1,j+1} \\ & + u_{i+1,j-1} + u_{i-1,j+1} + u_{i-1,j-1} - 20u_{ij}) + \mathcal{O}(h^4) \end{aligned} \quad (20)$$

357 where i, j are the indices of the grid points and h denotes the spacing constant h between the cells.
358 For the excitation variable u in the electrical part of the simulation, no-flux boundary conditions have
359 been used which were imposed by setting the two outermost cells to the same value, i.e. $u_{0,j} = u_{1,j}$. In
360 the mechanical part of the simulation, numerical calculations were carried out according to the following
361 scheme for each time step:

- 362 1. update of the position of the centre of mass x_{cm}
- 363 2. calculation of all four points of attachment q_i for each cell
- 364 3. computing of forces from structural and axial springs for each particle x_i
- 365 4. update of the positions of all particles using the Verlet method
- 366 5. determine change of area for each cell

Here the Verlet method refers to the standard Verlet algorithm which is given as [60]

$$\vec{x}_i^{t+\Delta t} = 2\vec{x}_i^t - \vec{x}_i^{t-\Delta t} + \frac{1}{m_i} \vec{F}_i^t \Delta t^2 + \mathcal{O}(\Delta t^4), \quad (21)$$

367 with the total force \vec{F}_i^t acting on the particle. Because the total force includes the damping term, it is
368 convenient to rewrite Eq. (21) with $\vec{F}_i = \vec{f}_i - \nu \frac{d\vec{x}_i}{dt}$ to

$$\vec{x}_i^{t+\Delta t} = \frac{2\vec{x}_i^t - \vec{x}_i^{t-\Delta t} (1 - \frac{\nu}{2m_i} \Delta t) + \frac{1}{m_i} \vec{f}_i^t \Delta t^2}{1 + \frac{\nu}{2m_i} \Delta t} + \mathcal{O}(\Delta t^4). \quad (22)$$

369 A padding layer of ten electrically inactive cells was implemented outside the electrical grid to account
370 for boundary effects in the mechanical network. In addition, the active stress variable T_a from Eq. (6) of
371 the last row of cells just at the edge of the simulation grid was mirrored to the two padding layers just
372 outside the simulation grid which proved to dramatically reduce mechanical boundary effects. This is likely
373 due to the fact that a proper contraction of an electrically active cell is not guaranteed if one of its sides is
374 connected to an inactive cell. Lastly, the outermost padding cell's positions were fixed to prevent the grid
375 as a whole from moving away from its original position.

376 To improve numerical accuracy for each time step Δt of the electrical equations (19) five time steps of size

377 $\Delta t/5$ were computed for the mechanical system (22). All computations have been performed on a spatial grid of 100×100 elements. The parameter values of the dynamical equations used are given in Table 3.

u_0	0.1	a	0.05	b	0.05	μ_1	0.2	μ_2	0.3	k	8	ϵ_0	0.002
D	0.22	Δt	0.08	k_T	3	k_{ij}	13	k_{ij}^{pad}	23	k_j	2	k_a	9
k_a^{pad}	23	c_a	10	m_i	0.2	ν	6.86	$\epsilon_{T,0}$	1	ϵ_∞	0.1	ξ_T	30

Table 3. Parameters of the electro-mechanical model.

378

APPENDIX 2: COMPARISON OF RECONSTRUCTION ERRORS FROM IMPAIRED DATA.

379 Table 4 shows the mean absolute errors of reconstructions obtained with ESNs and CAES for blurred, noisy, and undersampled data.

CAE (MAE \pm STD)			
Case	Blurred data	Noisy data	Undersampled data
1	0.01644 \pm 0.00136	0.00794 \pm 0.00096	0.00432 \pm 0.00020
2	0.02076 \pm 0.00170	0.00835 \pm 0.00097	0.00782 \pm 0.00053
3	0.02667 \pm 0.00227	0.00856 \pm 0.00104	0.01613 \pm 0.00119
4	0.03450 \pm 0.00318	0.00900 \pm 0.00096	0.04727 \pm 0.00393
5	0.04532 \pm 0.00407	0.00919 \pm 0.00099	0.12190 \pm 0.01061
6	0.06137 \pm 0.00585	0.00961 \pm 0.00110	0.27821 \pm 0.02823
7	0.08913 \pm 0.00864	0.01210 \pm 0.00103	0.42401 \pm 0.02800
8	0.14018 \pm 0.01261	0.01156 \pm 0.00120	
9	0.24689 \pm 0.01898	0.01873 \pm 0.00136	
10	0.37214 \pm 0.02408		
ESN (MAE \pm STD)			
Case	Blurred data	Noisy data	Undersampled data
1	0.05220 \pm 0.00347	0.06193 \pm 0.00264	0.00362 \pm 0.00031
2	0.05910 \pm 0.00365	0.07193 \pm 0.00288	0.01682 \pm 0.00110
3	0.06245 \pm 0.00394	0.08070 \pm 0.00299	0.03516 \pm 0.00242
4	0.07318 \pm 0.00469	0.09052 \pm 0.00312	0.08491 \pm 0.00561
5	0.08476 \pm 0.00536	0.09344 \pm 0.00341	0.20439 \pm 0.01105
6	0.09959 \pm 0.00644	0.11136 \pm 0.00370	n.A.
7	0.12325 \pm 0.00813	0.11889 \pm 0.00391	n.A.
8	0.18129 \pm 0.01323	0.14548 \pm 0.00596	
9	0.27925 \pm 0.01628	0.18259 \pm 0.00720	
10	0.39927 \pm 0.01717		

Table 4. Comparison of the MAE obtained when applying the CAE method and the ESN method to the test data set.

380

ESN hyperparameters for the case of noisy data					
Case	N	ρ	α	ϵ	L2 regularisation
1	500	1.25	0.5	0.05	10
2	500	0.50	0.2	0.05	10
3	250	1.00	0.2	0.2	10
4	250	0.05	0.2	0.2	10
5	250	3.00	0.2	0.1	10
6	250	1.25	0.2	0.1	10
7	250	3.00	0.2	0.05	10
8	500	1.25	0.05	0.2	10
9	500	3.00	0.05	0.05	10

Table 5. Selected ESN hyperparameters for the case of noisy data

ESN hyperparameters for the case of blurred data					
Case	N	ρ	α	ϵ	L2 regularisation
1	250	0.30	0.01	0.1	10
2	250	1.25	0.05	0.05	10
3	500	2.00	0.05	0.1	10
4	500	0.75	0.2	0.2	5
5	250	1.50	0.2	0.2	10
6	250	1.50	0.2	0.2	10
7	250	1.25	0.2	0.2	10
8	500	0.90	0.2	0.2	5
9	250	1.50	0.5	0.2	10
10	500	0.75	0.7	0.2	10

Table 6. Selected ESN hyperparameters for the case of blurred data

ESN hyperparameters for the case of undersampled data					
Case	N	ρ	α	ϵ	L2 regularisation
1	250	0.30	0.01	0.1	5
2	500	0.05	0.05	0.2	0.5
3	250	1.50	0.05	0.2	10
4	250	2.00	0.5	0.2	10
5	500	0.30	0.05	0.1	5
6	500	0.75	0.2	0.05	10
7	250	1.50	0.05	0.1	10

Table 7. Selected ESN hyperparameters for the case of undersampled data

REFERENCES

- 381 [1] Gray RA, Pertsov AM, Jalife J. Spatial and temporal organization during cardiac fibrillation. *Nature*
382 **392** (1998) 75–78. doi:10.1038/32164.
- 383 [2] Davidenko JM, Pertsov AV, Salomonsz R, Baxter W, Jalife J. Stationary and drifting spiral waves of
384 excitation in isolated cardiac muscle. *Nature* **355** (1992) 349–51. doi:10.1038/355349a0.
- 385 [3] Witkowski FX, Leon LJ, Penkoske PA, Giles WR, Spano ML, Ditto WL, et al. Spatiotemporal
386 evolution of ventricular fibrillation. *Nature* (1998). doi:10.1038/32170.
- 387 [4] Efimov IR, Nikolski VP, Salama G. Optical imaging of the heart **95** (2004) 21–33. doi:10.1161/01.
388 RES.0000130529.18016.35.

- 389 [5] Rosenzweig S, Scholand N, Holme HCM, Uecker M. Cardiac and respiratory self-gating in radial mri
390 using an adapted singular spectrum analysis (ssa-fary). *IEEE Transactions on Medical Imaging* **39**
391 (2020) 3029–3041.
- 392 [6] Otani N, Luther S, Singh R, Gilmour R. Transmural ultrasound-based visualization of patterns of
393 action potential wave propagation in cardiac tissue. *Ann Biomed Eng.* **38** (2010) 3112—3123.
- 394 [7] Otani N, Luther S, Singh R, Gilmour R. Methods and systems for functional imaging of cardiac tissue.
395 *World Intellectual Property Organization* (2010). International Patent: WO 2011/028973 A3.
- 396 [8] Provost J, Lee WN, Fujikura K, Konofagou EE. Imaging the electromechanical activity of the heart in
397 vivo. *Proceedings of the National Academy of Sciences of the United States of America* **108** (2011)
398 8565–8570. doi:10.1073/pnas.1011688108.
- 399 [9] Bers DM. Cardiac excitation-contraction coupling. *Nature* **415** (2002) 198–205. doi:10.1038/415198a.
- 400 [10] Bourgeois EB, Bachtel AD, Huang J, Walcott GP, Rogers JM. Simultaneous optical mapping of
401 transmembrane potential and wall motion in isolated, perfused whole hearts. *Journal of Biomedical*
402 *Optics* **16** (2011) 096020. doi:10.1117/1.3630115.
- 403 [11] Zhang H, Iijima K, Huang J, Walcott GP, Rogers JM. Optical Mapping of Membrane Potential and
404 Epicardial Deformation in Beating Hearts. *Biophysical Journal* **111** (2016) 438–451. doi:10.1016/j.
405 bpj.2016.03.043.
- 406 [12] Christoph J, Schröder-Schetelig J, Luther S. Electromechanical optical mapping. *Progress in*
407 *Biophysics and Molecular Biology* **130** (2017) 150–169. doi:10.1016/j.pbiomolbio.2017.09.015.
- 408 [13] Christoph J, Chebbok M, Richter C, Schroeder-Schetelig J, Stein S, Uzelac I, et al. Electromechanical
409 vortex filaments during cardiac fibrillation. *Nature* **555** (2018) 667–672. doi:10.1038/nature26001.
- 410 [14] Berg S, Luther S, Parlitz U. Synchronization based system identification of an extended excitable
411 system. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **21** (2011) 033104. doi:10.1063/1.
412 3613921.
- 413 [15] Lebert J, Christoph J. Synchronization-based reconstruction of electromechanical wave dynamics in
414 elastic excitable media. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29** (2019) 093117.
415 doi:10.1063/1.5101041.
- 416 [16] Hoffman MJ, LaVigne NS, Scorse ST, Fenton FH, Cherry EM. Reconstructing three-dimensional
417 reentrant cardiac electrical wave dynamics using data assimilation. *Chaos: An Interdisciplinary*
418 *Journal of Nonlinear Science* **26** (2016) 013107. doi:10.1063/1.4940238.
- 419 [17] Hoffman MJ, Cherry EM. Sensitivity of a data-assimilation system for reconstructing three-
420 dimensional cardiac electrical dynamics. *Philosophical Transactions of the Royal Society A:*
421 *Mathematical, Physical and Engineering Sciences* **378** (2020) 20190388. doi:10.1098/rsta.2019.0388.
- 422 [18] Zimmermann RS, Parlitz U. Observing spatio-temporal dynamics of excitable media using reservoir
423 computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **28** (2018) 043118. doi:10.
424 1063/1.5022276.
- 425 [19] Herzog S, Wörgötter F, Parlitz U. Data-driven modeling and prediction of complex spatio-temporal
426 dynamics in excitable media. *Frontiers in Applied Mathematics and Statistics* **4** (2018) 60. doi:10.
427 3389/fams.2018.00060.
- 428 [20] Herzog S, Wörgötter F, Parlitz U. Convolutional autoencoder and conditional random fields hybrid for
429 predicting spatial-temporal chaos. *Chaos* **29** (2019) 123116. doi:10.1063/1.5124926.
- 430 [21] Christoph J, Lebert J. Inverse mechano-electrical reconstruction of cardiac excitation wave patterns
431 from mechanical deformation using deep learning (2020).
- 432 [22] Bueno-Orovio A, Cherry EM, Fenton FH. Minimal model for human ventricular action potentials in
433 tissue. *Journal of Theoretical Biology* **253** (2008) 544–560. doi:10.1016/j.jtbi.2008.03.029.

- 434 [23] Clayton R, Bernus O, Cherry E, Dierckx H, Fenton F, Mirabella L, et al. Models of cardiac tissue
435 electrophysiology: Progress, challenges and open questions. *Progress in Biophysics and Molecular*
436 *Biology* **104** (2011) 22 – 48. doi:https://doi.org/10.1016/j.pbiomolbio.2010.05.008. Cardiac Physiome
437 project: Mathematical and Modelling Foundations.
- 438 [24] Ten Tusscher K, Noble D, Noble P, Panfilov AV. A model for human ventricular tissue. *American*
439 *Journal of Physiology-Heart and Circulatory Physiology* **286** (2004) H1573–H1589.
- 440 [25] Strain MC, Greenside HS. Size-dependent transition to high-dimensional chaotic dynamics in a
441 two-dimensional excitable medium. *Phys. Rev. Lett.* **80** (1998) 2306–2309. doi:10.1103/PhysRevLett.
442 80.2306.
- 443 [26] Lilienkamp T, Christoph J, Parlitz U. Features of chaotic transients in excitable media governed by
444 spiral and scroll waves. *Phys. Rev. Lett.* **119** (2017) 054101.
- 445 [27] Aliev RR, Panfilov AV. A simple two-variable model of cardiac excitation. *Chaos, Solitons & Fractals*
446 **7** (1996) 293–301. doi:10.1016/0960-0779(95)00089-5.
- 447 [28] Nash MP, Panfilov AV. Electromechanical model of excitable tissue to study reentrant cardiac
448 arrhythmias. *Progress in Biophysics and Molecular Biology* **85** (2004) 501–522. doi:10.1016/j.
449 pbiomolbio.2004.01.016.
- 450 [29] Göktepe S, Kuhl E. Electromechanics of the heart: a unified approach to the strongly coupled
451 excitation–contraction problem. *Computational Mechanics* **45** (2009) 227–243. doi:10.1007/
452 s00466-009-0434-z.
- 453 [30] Eriksson TSE, Prassl A, Plank G, Holzapfel G. Influence of myocardial fiber/sheet orientations on
454 left ventricular mechanical contraction. *Mathematics and Mechanics of Solids* **18** (2013) 592–606.
455 doi:10.1177/1081286513485779.
- 456 [31] Bourguignon D, Cani MP. Controlling anisotropy in mass-spring systems. *Eurographics* (Springer
457 Vienna) (2000), 113–123. doi:10.1007/978-3-7091-6344-3_9.
- 458 [32] Jaeger H. The ‘echo state’ approach to analysing and training recurrent neural networks – with an
459 erratum note. *GMD Report* (German National Research Institute for Computer Science) (2001), vol.
460 148, 43 pp.
- 461 [33] Cheng Z, Sun H, Takeuchi M, Katto J. Deep convolutional autoencoder-based lossy image compression.
462 *2018 Picture Coding Symposium, PCS 2018 - Proceedings* (Institute of Electrical and Electronics
463 Engineers Inc.) (2018), 253–257. doi:10.1109/PCS.2018.8456308.
- 464 [34] Lukoševičius M. A practical guide to applying echo state networks. *Lecture Notes in Computer*
465 *Science* (Springer Berlin Heidelberg) (2012), 659–686. doi:10.1007/978-3-642-35289-8_36.
- 466 [35] Lukoševičius M, Jaeger H. Reservoir computing approaches to recurrent neural network training.
467 *Computer Science Review* **3** (2009) 127–149. doi:10.1016/j.cosrev.2009.03.005.
- 468 [36] Yildiz IB, Jaeger H, Kiebel SJ. Re-visiting the echo state property. *Neural Networks* **35** (2012) 1–9.
469 doi:10.1016/j.neunet.2012.07.005.
- 470 [37] Jaeger H, Haas H. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless
471 communication. *Science* **304** (2004) 78–80. doi:10.1126/science.1091277.
- 472 [38] Lu Z, Pathak J, Hunt B, Girvan M, Brockett R, Ott E. Reservoir observers: Model-free inference of
473 unmeasured variables in chaotic systems. *Chaos: An Interdisciplinary Journal of Nonlinear Science*
474 **27** (2017) 041102. doi:10.1063/1.4979665.
- 475 [39] Carroll TL, Pecora LM. Network structure effects in reservoir computers. *Chaos: An Interdisciplinary*
476 *Journal of Nonlinear Science* **29** (2019) 083130. doi:10.1063/1.5097686.
- 477 [40] Griffith A, Pomerance A, Gauthier DJ. Forecasting chaotic systems with very low connectivity
478 reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **29** (2019) 123108.

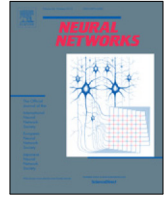
- 479 doi:10.1063/1.5120710.
- 480 [41] Thiede LA, Parlitz U. Gradient based hyperparameter optimization in echo state networks. *Neural*
481 *Networks* **115** (2019) 23 – 29. doi:https://doi.org/10.1016/j.neunet.2019.02.001.
- 482 [42] Haluszczynski A, Aumeier J, Herteux J, R ath C. Reducing network size and improving prediction
483 stability of reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **30** (2020)
484 063136. doi:10.1063/5.0006869.
- 485 [43] Carroll TL. Dimension of reservoir computers. *Chaos: An Interdisciplinary Journal of Nonlinear*
486 *Science* **30** (2020) 013102. doi:10.1063/1.5128898.
- 487 [44] Fan H, Jiang J, Zhang C, Wang X, Lai YC. Long-term prediction of chaotic systems with machine
488 learning. *Phys. Rev. Research* **2** (2020) 012080. doi:10.1103/PhysRevResearch.2.012080.
- 489 [45] Parlitz U, Hornstein A. Dynamical prediction of chaotic time series. *Chaos and Complexity Letters* **1**
490 (2005) 135–144.
- 491 [46] Lu Z, Hunt BR, Ott E. Attractor reconstruction by machine learning. *Chaos: An Interdisciplinary*
492 *Journal of Nonlinear Science* **28** (2018) 061104. doi:10.1063/1.5039508.
- 493 [47] Pathak J, Hunt B, Girvan M, Lu Z, Ott E. Model-free prediction of large spatiotemporally chaotic
494 systems from data: A reservoir computing approach. *Physical Review Letters* **120** (2018). doi:10.
495 1103/physrevlett.120.024102.
- 496 [48] Parlitz U, Merkwirth C. Prediction of spatiotemporal time series based on reconstructed local states.
497 *Physical Review Letters* **84** (2000) 1890–1893. doi:10.1103/physrevlett.84.1890.
- 498 [49] LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, et al. Handwritten digit
499 recognition with a back-propagation network. Touretzky DS, editor, *Advances in Neural Information*
500 *Processing Systems 2* (Morgan-Kaufmann) (1990), 396–404.
- 501 [50] Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal
502 covariate shift. *Proceedings of the 32nd International Conference on International Conference on*
503 *Machine Learning - Volume 37* (JMLR.org) (2015), ICML'15, 448–456.
- 504 [51] Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. *in*
505 *ICML Workshop on Deep Learning for Audio, Speech and Language Processing* (2013), 3.
- 506 [52] Hahnloser RH, Sarpeshkar R, Mahowald MA, Douglas RJ, Seung HS. Digital selection and analogue
507 amplification coexist in a cortex-inspired silicon circuit. *Nature* **405** (2000) 947.
- 508 [53] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A simple way
509 to prevent neural networks from overfitting. *Journal of Machine Learning Research* **15** (2014)
510 1929–1958.
- 511 [54] Dumoulin V, Visin F. A guide to convolution arithmetic for deep learning. *arXiv e-prints* (2016)
512 arXiv:1603.07285.
- 513 [55] B auerle A, Ropinski T. Net2vis: Transforming deep convolutional networks into publication-ready
514 visualizations. *arXiv preprint arXiv:1902.04394* (2019).
- 515 [56] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. *arXiv e-prints* (2014)
516 arXiv:1412.6980.
- 517 [57] [Dataset] Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, et al. TensorFlow: Large-scale
518 machine learning on heterogeneous systems (2015). Software available from tensorflow.org.
- 519 [58] [Dataset] Barron JT. A general and adaptive robust loss function (2019).
- 520 [59] Rosser JB. Nine-point difference solutions for poisson's equation. *Computers & Mathematics with*
521 *Applications* **1** (1975) 351–360. doi:10.1016/0898-1221(75)90035-8.
- 522 [60] Scherer POJ. *Computational physics : simulation of classical and quantum systems* (Berlin Heidelberg:
523 Springer) (2010), 147 .

2.3.1 Conclusions from (Herzog et al. 2021c)

The results from the paper above are promising for both approaches, but the CAE gives better results than the ESN on the data studied. Apart from this point, there is another technical point that is an advantage for CAE, namely the possibility of using highly optimised and customised libraries like *Tensorflow* (Abadi et al. 2015) and *PyTorch* (Paszke et al. 2019), which can run completely on the graphics processing unit (GPU). In the following chapters, the CAE is used as an elementary building block. However, before the hybrid model is presented, a possibility of further increasing the performance of the CAE should be introduced next.

2.4 Publication: (Herzog et al. 2020a)

As *(Herzog et al. 2021c) has demonstrated, a CAE can be used to approximate various functions from data from non-linear systems. A closer look at *(Herzog et al. 2021c, figures 8, 11 and 14, panals D, J, P) shows that the biggest errors occur at the wave fronts. Already in an earlier work, the question arose whether one could increase the discriminatory power of neural networks by using recurrent connections. The naive motivation behind this was connected with the question whether recurrent connections (in particular whether the feedback from a higher layer in the network) could be useful to better distinguish objects that are very similar to each other. The following work *Herzog, S., Tetzlaff, C., and Wörgötter, F. (2020a). “Evolving artificial neural networks with feedback”. In: *Neural Networks : the official journal of the International Neural Network Society* 123, pp. 153–162 does not directly address this question. It does, however, present an approach with which it is possible to add recurrent connections to the network via non-adjacent network layers to thereby drastically reduce the number of necessary network layers.



Evolving artificial neural networks with feedback

Sebastian Herzog¹, Christian Tetzlaff¹, Florentin Wörgötter^{*}

Third Institute of Physics, Universität Göttingen, Friedrich-Hund Platz 1, 37077 Göttingen, Germany
Bernstein Center for Computational Neuroscience, Friedrich-Hund Platz 1, 37077 Göttingen, Germany



ARTICLE INFO

Article history:

Received 15 November 2018
Received in revised form 27 November 2019
Accepted 2 December 2019
Available online 14 December 2019

Keywords:

Deep learning
Feedback
Transfer entropy
Convolutional neural network

ABSTRACT

Neural networks in the brain are dominated by sometimes more than 60% feedback connections, which most often have small synaptic weights. Different from this, little is known how to introduce feedback into artificial neural networks. Here we use transfer entropy in the feed-forward paths of deep networks to identify feedback candidates between the convolutional layers and determine their final synaptic weights using genetic programming. This adds about 70% more connections to these layers all with very small weights. Nonetheless performance improves substantially on different standard benchmark tasks and in different networks. To verify that this effect is generic we use 36000 configurations of small (2–10 hidden layer) conventional neural networks in a non-linear classification task and select the best performing feed-forward nets. Then we show that feedback reduces total entropy in these networks always leading to performance increase. This method may, thus, supplement standard techniques (e.g. error backprop) adding a new quality to network learning.

© 2019 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY-NC-ND license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

1. Introduction

Modern deep neural networks employ sometimes more than 100 hierarchical layers between input and output (He, Zhang, Ren, & Sun, 2015), whereas vertebrate brains achieve high levels of performance using a much shallower hierarchy. This may well be largely due to massive recurrent and feedback connections, which are dominant constituents of cortical connectivity (Markov et al., 2014). Their role remains puzzling in artificial neural networks.

Intra-layer recurrent connections, more commonly known as lateral connections, have indeed become an important aspect in several deep learning architectures, notably deep recurrent neural nets (DRNN) (Hermans & Schrauwen, 2013; Liao & Poggio, 2016) and especially also in long–short-term memory networks (LSTM) (Hochreiter & Schmidhuber, 1997), which show superior performance as compared to conventional DRNNs. Different from this, inter-layer feedback is less common and mostly employed in rather specific ways. For example, feedback connections have been introduced into deep Boltzmann Machines, which are an unsupervised method (Salakhutdinov & Hinton, 2009). Alternatively, feedback has been used in deep architectures to create the equivalent of selective attention (Stollenga, Masci, Gomez, & Schmidhuber, 2014) or to implement some aspects of long-term memory for object recognition (Tang et al., 2018; Varadarajan

& Vincze, 2013). Furthermore, a pioneering study using a small network and a set of rather simple tasks (Kietzmann et al., 2019; Spoerer, McClure, & Kriegeskorte, 2017) indicates the benefit of using recurrent convolutional architectures on the system's performance. A recent study (Spoerer, Kietzmann, & Kriegeskorte, 2020) shows that including recurrent connections that let information flow in cycles can improve performance of deep neural networks for vision tasks. Another study (Ernst, Triesch, & Burwick, 2019) shows that recurrent connections move the network's representation of an occluded object towards its unoccluded representation. Several other approaches exist that employ feedback in deep learning architectures in different ways and for different applications including connections that only influence learning (Lillicrap, Cownden, Tweed, & Akerman, 2016), for a general review see LeCun, Bengio, and Hinton (2015). It seems, however, fair to conclude that currently little is known about when and how to introduce feedback in deep network architectures.

In the vertebrate brain, feedback connectivity had been investigated in great detail for the visual system, where it can influence spatial- as well as object- and feature-oriented attention. It can affect perceptual tasks and object expectation, and it can also help to create efference copies, influence perceptual learning, and guide different aspects of memory function (reviewed in, e.g., Gilbert & Li, 2013 and Spillmann, Dresch-Langley, & Tseng, 2015).

The architectural differences between real and deep neural networks, which concern feed-forward and in particular also feedback connectivity, make it, however, difficult to compare

^{*} Corresponding author at: Third Institute of Physics, Universität Göttingen, Friedrich-Hund Platz 1, 37077 Göttingen, Germany.

E-mail address: worgott@gwdg.de (F. Wörgötter).

¹ These authors contributed equally.

deep neural networks and their performance to brain structure and function. Several relations between both substrates have been reported (Cichy, Khosla, Pantazis, Torralba, & Oliva, 2016; Güçlü & van Gerven, 2015; Khaligh-Razavi & Kriegeskorte, 2014; Yamins & DiCarlo, 2016) but processing architectures differ too much to allow for direct links. Differences and potential relations, however, are currently vividly discussed (Bengio, Lee, Bornschein, Mesnard, & Lin, 2015; Kietzmann, McClure, & Kriegeskorte, 0000; Liao & Poggio, 2016; Marblestone, Wayne, & Körding, 2016).

One dominant aspect of feedback connectivity in the brain is that it amplifies feed-forward processing pathways and increases their signal throughput. Thus, we asked whether we could positively influence the performance of a deep neural network using a similar mechanism? However, one central problem, which has so far hindered the implementation of a potentially more realistic feedback connectivity in artificial nets, is how to actually structure such connections, especially between non-neighboring layers. For example in the small AlexNet, when considering *only* the convolutional layers, there would be about 10^8 feedback connections possible. Therefore, clear guiding principles are needed to find an appropriate set of feedback connections, which would lead to performance improvement.

To achieve this, we use transfer entropy (Schreiber, 2000) between convolutional network units. Previously, this measure has been used to estimate the functional connectivity between neurons (Lizier, Heinzle, Horstmann, Haynes, & Prokopenko, 2011; Vicente, Wibral, Lindner, & Pipa, 2011). Transfer entropy between neurons is a measure of the mutual dependence between neurons, indicating relevant sub-paths in the network. Thus, initially, we use transfer entropy to identify such pathways and connect them by feedback with random weights similar to those in the feed-forward connections. For the AlexNet, this way we arrive only at about 3.5% of all possible feedback connections. Then we use a genetic algorithm to modify their connection strengths and obtain in the end a set of very small weights, similar to many feedback paths in the brain, which amplify already connected feed-forward paths only very mildly. Remarkably, this type of feedback substantially improves classification performance for different networks and data sets.

2. Methods

First we describe the core aspects on the here-used networks and data as well as how we introduce feedback by ways of an overview. Details about this are then provided directly afterwards.

2.1. Overview

2.1.1. Data and network architectures

We have first tested our algorithm on CIFAR-10 and ImageNet data using AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and ResNet (He et al., 2015) architectures. In total, CIFAR-10 contains 60,000 color images (Krizhevsky, Nair, & Hinton, 0000), while ImageNet-Challenge has more than one million images (Deng et al., 2009). Both are standard benchmarks. The AlexNet has eight layers. The here-used ResNet has 32, where we also use a 56-layer version for comparison.

The following description focuses on the CIFAR-10 set and the AlexNet. But all notations apply in a similar way to the other cases, too. Feedback connections have in both cases been implemented in the same way between the convolution layers.

The core of the AlexNet consists of five layers, which perform convolutions (Fig. 1). Note that only between those layers (Layers 1–5) feedback connections will be formed; the three dense layers (Layers 6–8) are excluded from the formation of feedback

connections. Layers 1–5 consist of 19072 units and 1376 kernels. Every unit in layer l receive inputs from a set of units located in the previous layer $l - 1$. This input set is defined by a kernel with each element of the kernel matrix defining the strength of the connection between its corresponding input unit from this set and the target unit in layer l . As usual, we call the value of a kernel-element *connection weight*. Accordingly, each unit calculates the sum of all its inputs multiplied by the connection weights resulting in o (called *raw output*). This scalar is clipped at zero defining the final *activation* by: $\psi(o) = \max\{0, o\}$; this non-linearity is often called rectified linear unit (ReLU) (Nair & Hinton, 2010). Different from the feed-forward connections, which can only connect adjacent layers, we consider feedback also for non-adjacent convolutional layers. Theoretically, in the AlexNet there are 140,820,480 such connections possible. For ResNet32 this number is far bigger and this shows that it is impossible to arrive at structured feedback without clear criteria of where to make a connection.

2.1.2. Algorithm

Algorithmic steps are:

1. Train the kernels for feed-forward net using a subset (the training-data) of the data set, employing standard error back-propagation training (Rumelhart, Hinton, & Williams, 1986; Schmidhuber, 2015). Store the resulting feed-forward connection weights (i.e. kernel elements).
2. Run the feed-forward net on the complete training-data set and calculate for each image the resulting raw output o of each unit. From this, also calculate the units' activations $\psi = \max\{0, o\}$ and store all this.
3. For all units i and j , where i is in a layer below that of j , use their activations to calculate the transfer entropy $T_{i \rightarrow j}$ between them. Use the highest transfer entropy to normalize $T_{i \rightarrow j}$ such that the normalized values are in the interval $[0, 1]$. Create and store a list of all pairs of units i and j , for which $T_{i \rightarrow j} > \Phi$, where Φ is the upper quantile of all $T_{i \rightarrow j}$.
4. Apply the genetic algorithm. Make the first population Ω by creating 100 population members consisting of networks with identical feed-forward weights (from step 1 above) and randomly assigned feedback weights with values between the minimum and maximum feed-forward weights. Assign feedback connections only for units i and j for which their transfer entropy was in the desired quantile (see step 3).
 - Calculate the *fitness*, on the training-data of all members of Ω defined as their performance on the image recognition task.
 - Perform cross-over to create new feedback weight combinations and
 - subject these new combinations also to mutations. This way create the members of a new population Ω^* .
 - Calculate the fitness of all members in Ω^* .
 - Combine the 100 fittest members from Ω and Ω^* in a new population and call this again Ω .
 - Continue with the first step until 200 generations have been processed.
5. Take the best performing network after 200 generations as the final result.

2.2. Detailed methods

First, we introduce the formal definition of Artificial Neural Networks (ANNs) and their convolutional layers to allow using the there-introduced notation for transfer entropy and genetic algorithm, which are then defined in following sub-sections.

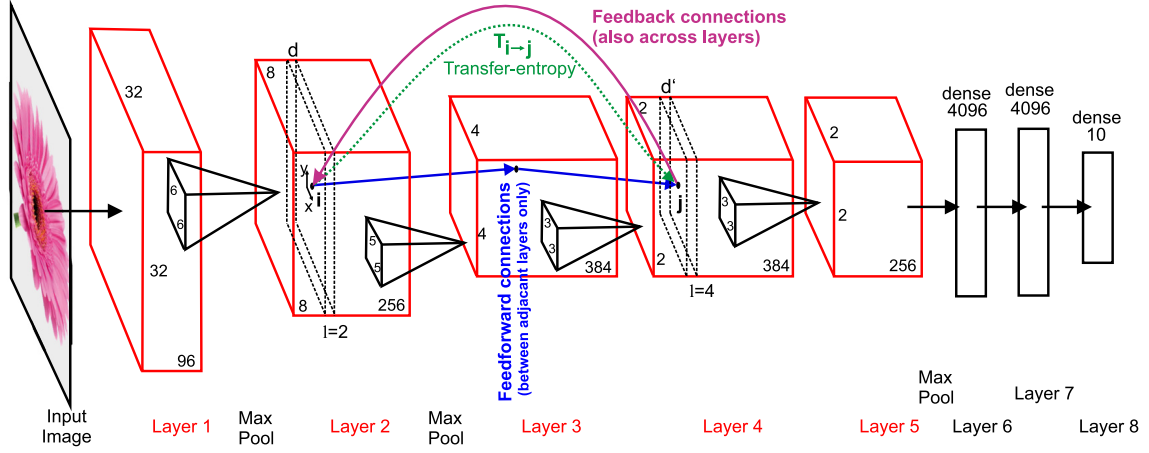


Fig. 1. Structure of and computations performed by the AlexNet. Red are convolutional layers, black dense layers; with number of units given. Pyramids show filter sizes. Feed-forward connections exist only between adjacent layers (e.g., blue arrows). Transfer entropy is calculated for all units i and j in different layers l where $l_i < l_j$ (green) and feedback connections (pink) will be made following a threshold criterion (see text). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2.1. Artificial neural network

Artificial neural network (ANNs) are parameterizable models to approximate a function F^* . The actual function of an ANN:

$$f : \mathbb{R}^O \mapsto \mathbb{R}^P, \quad (1)$$

where $O \in \mathbb{N}$ is the dimension of the input and $P \in \mathbb{N}$ the dimension of the output, is supposed to be $f \simeq F^*$. One of the first common ANNs were feed-forward neural networks (FNN). A one-layer network f depends non-linearly on the input $X \in \mathbb{R}^O$ such that:

$$f(X) = \psi(WX + b), \quad (2)$$

with nonlinear function ψ , weight matrix $W \in \mathbb{R}^{P \times O}$, and bias $b \in \mathbb{R}$. By recursively applying the output of one layer as input to the next layer, a multi-layer FNN can be constructed:

$$f(X) = f^L(\dots f^2(f^1(X; W^1, b^1); W^2, b^2) \dots; W^L, b^L). \quad (3)$$

Eq. (3) is the model for a multi-layer FNN with $L \in \mathbb{N}$ layers.

We denote \mathcal{W}^{ff} as the set of feed-forward weights of the multi-layer FNN:

$$\mathcal{W}^{ff} = \{W^1, W^2, \dots, W^L\}. \quad (4)$$

W.l.o.g., Eqs. (1)–(3) can be extended to higher dimensional systems, i.e.: images where the input X becomes $\mathbf{X}_\alpha \in \mathbb{R}^{h_X^{(0)} \times w_X^{(0)} \times d_X}$ with $h_X^{(0)} \in \mathbb{N}$ rows, $w_X^{(0)} \in \mathbb{N}$ columns, and d_X channels (see below). To improve the approximation properties of the network (Eq. (3)), FNNs can consist of convolutional layers leading to state-of-the-art models for data classification, so-called convolutional neural networks.

2.3. Convolution layer

Convolutional neural networks for image classification receive the trainings data-set

$\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m\}$, where $\mathbf{X}_\alpha \in \mathbb{R}^{h_X^{(0)} \times w_X^{(0)} \times d_X}$ is an image with $h_X^{(0)} \in \mathbb{N}$ rows, $w_X^{(0)} \in \mathbb{N}$ columns, and d_X channels (in the case of R, G, B color channels $d_X = 3$). Each channel of \mathbf{X}_α is denoted by $X_\alpha^{(k)}$, such that $\mathbf{X}_\alpha = \{X_\alpha^{(1)}, X_\alpha^{(2)}, \dots, X_\alpha^{(d_X)}\}$.

The data processing through the network is described layer-wise, i.e. in the l th convolutional layer the input $\mathbf{X}^{(l)}$ will be transformed to the raw output $\mathbf{o}^{(l)}$, which is in turn the input to the next layer $l + 1$, where the dimension changes depending on the number and size of convolutions, padding and stride of the

layers. The padding parameter $P^{(l)} \in \mathbb{N}$, for layer l , describes the number of zeros at the edges of an image with which the image is extended. This is necessary since every convolution being larger than 1×1 will decrease the output size. The stride parameter $S^{(l)} \in \mathbb{N}$ is the parameter determining how much the kernel is shifted in each step to compute the next spatial position x, y . This specifies the overlap between individual output pixels, and it is here set to 1.

Each layer l is specified by its number of kernels $\mathbf{K}^{(l)} = \{K^{(l,1)}, K^{(l,2)}, \dots, K^{(l,d_K)}\}$, where $d_K \in \mathbb{N}$ is the number of kernels in layer l , and its additive bias terms $\mathbf{b}^{(l)} = \{b^{(l,1)}, b^{(l,2)}, \dots, b^{(l,d_K)}\}$ with $b^{(l,d)} \in \mathbb{R}$. Consider that the input $\mathbf{X}^{(l)} \in \mathbb{R}^{h_X^{(l)} \times w_X^{(l)}}$ in the l th layer with size $h_X^{(l)} \times w_X^{(l)}$ and depth $d_X^{(l)}$ is processed by a set of kernels $\mathbf{K}^{(l)}$. For each kernel $K^{(l,d)} \in \mathbb{R}^{h_K^{(l)} \times w_K^{(l)}}$ with size $h_K^{(l)} \times w_K^{(l)}$ and $d \in \{1, \dots, d_K\}$, the raw output $\mathbf{o}^{(l)} \in \mathbb{R}^{\frac{h_X^{(l)} - h_K^{(l)} - 1 + P^{(l)}}{S^{(l)}} \times \frac{w_X^{(l)} - w_K^{(l)} - 1 + P^{(l)}}{S^{(l)}}}$ is computed elementwise by:

$$o_{x,y}^{(l,d)} = (\mathbf{X}^{(l)} * K^{(l,d)})_{x,y} = \left(b^{(l,d)} + \sum_{k=1}^{d_X^{(l)}} \sum_{i=1}^{h_K^{(l)}} \sum_{j=1}^{w_K^{(l)}} K_{i,j}^{(l,d)} \cdot X_{x+i-1,y+j-1}^{(l,k)} \right). \quad (5)$$

The result is modulated by an activation function to obtain the activation $\psi(o_{x,y}^{(l,d)})$ of a unit:

$$\psi(o_{x,y}^{(l,d)}) = \max\{0, o_{x,y}^{(l,d)}\}. \quad (6)$$

To obtain $\mathbf{o}^{(l)} = \{o^{(l,1)}, \dots, o^{(l,d_K)}\}$, Eq. (6) needs to be calculated $\forall d = 1, \dots, d_K$ and $\forall x, y$. Each spatial calculation of $o_{x,y}^{(l,d)}$ is considered as a unit and $\psi(o_{x,y}^{(l,d)})$ as the feed-forward activation of the unit. The value of an element of a kernel ($K_{i,j}^{(l,d)}$) between two units is the weight of the feed-forward connection.

2.4. Transfer entropy

Before calculating the transfer entropy (Schreiber, 2000), the activation for all images $\mathbf{X}_\gamma \in \mathcal{X}_\beta$, where \mathcal{X}_β is the set of images from one class, with $\gamma = \{1, \dots, q\}$ elements, is calculated by the network. The series of activations for each unit across all images

$$\Psi(o_{x,y}^{(l,d)}) = \left\{ \psi(o_{x,y}^{(l,d)})_1, \psi(o_{x,y}^{(l,d)})_2, \dots, \psi(o_{x,y}^{(l,d)})_q \right\}, \quad (7)$$

is stored for each class. Thus, for every unit, Eq. (7) calculates the activation-set across all images, which captures the occurrence frequencies of all possible activation-events and will be used to calculate the probabilities needed in the following equations. For better readability, in the following, we will drop the function argument of the activation such that $\psi_1 := \psi(o_{x,y}^{(l,d)})_1$, $\psi := \psi(o_{x,y}^{(l,d)})$, $\psi' := \psi(o_{x',y'}^{(l',d')})$, and so forth. In addition, we assume w.l.o.g. that layer $l' > l$.

A section $[t, m]$ of the series ψ is denoted by

$$\psi_{[t,m]} = \{\psi_t, \psi_{t+1}, \dots, \psi_{t+m}\}.$$

Thus, the transfer entropy $T_{\psi \rightarrow \psi'}$ between two units is calculated by

$$T_{\psi \rightarrow \psi'} = \sum_{t,n,m=1}^{m'} p[\psi_{t+1}, \psi_{[t,n]}, \psi'_{[t,m]}] \log \left(\frac{p[\psi_{t+1} | \psi_{[t,n]}, \psi'_{[t,m]}]}{p[\psi_{t+1} | \psi_{[t,n]}} \right), \quad (8)$$

where “ \rightarrow ” refers to a feed-forward connection.

To calculate the transfer entropy between units in non-adjacent layers, we have to generalize Eq. (8) by conditioning on the activations of the units being in the in-between layers (here denoted by ψ''). For one in-between layer this reads:

$$T_{\psi \rightarrow \psi'} = \sum_{t,n,m,o=1}^{m'} p[\psi_{t+1}, \psi_{[t,n]}, \psi''_{[t,o]}, \psi'_{[t,m]}] \log \left(\frac{p[\psi_{t+1} | \psi_{[t,n]}, \psi''_{[t,o]}, \psi'_{[t,m]}]}{p[\psi_{t+1} | \psi_{[t,n]}} \right). \quad (9)$$

An efficient implementation to estimate the transfer entropy can be found in Wollstadt et al. (2019). To estimate the conditional mutual information the OpenCL-implementation flag was used.

2.5. Shannon entropy

We use the Shannon Entropy to quantify the behavior of a network. To calculate it, we run the network on the complete data set and calculate and store for each input the resulting raw output o of each unit, as well as the units' activations $\psi = \max\{0, o\}$.

Then we use the stored activations to calculate the conditional probabilities $p[\psi_t | \psi'_t]$ for all pairs of units ψ_t, ψ'_t . Finally we obtain Shannon Entropy in the usual way as

$$H = - \sum p[\psi_t | \psi'_t] \log_2 p[\psi_t | \psi'_t]. \quad (10)$$

2.6. Genetic algorithm

To add specific feedback to a trained feed-forward network, a genetic algorithm (GA) (Barker, 1958; Holland & Goldberg, 0000; Koza, 1994) is utilized to find the local, optimal weights of the corresponding feedback connections. In the following, we name the network performance on the training set *fitness* to use the same common terminology as in the field of GAs. For the training data set $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m\}$ with desired labels $\mathcal{Z} = \{Z_1^*, Z_2^*, \dots, Z_m^*\}$, the fitness is calculated by

$$\text{fitness}(\mathbf{f}(*), \mathcal{X}, \mathcal{Z}) = \left(\sum_{\alpha=1}^m \delta(\mathbf{f}(\mathbf{X}_\alpha), Z_\alpha^*) \right) / m, \quad (11)$$

where \mathbf{f} is the network and $\delta(\cdot) : \mathbb{R} \mapsto \{0, 1\}$ is the Kronecker delta function being 1, if the variables are equal, and 0 otherwise.

For all of the following steps, we leave the feed-forward connectivity in the network constant, the genetic algorithm adapts only the feedback weights, the connectivity of the nodes is encoded in a way described in Stanley and Miikkulainen (2002).

For the GA, we define first a *population* by its members. A *member* of a *population* is a set of feedback weights $\mathcal{W}^{(i)}$, which are *initially* randomly assigned between any two units in the network under the only condition that the transfer entropy between these units is above some control parameter Φ . The initial random weights are chosen between the maximal and minimal value of the feed-forward network. If the transfer entropy is below Φ those two units will not be linked by a feedback connection. Hence, a *population* is a set of p such realizations given by: $\Omega = \{\mathcal{W}^{(1)}, \mathcal{W}^{(2)}, \dots, \mathcal{W}^{(p)}\}$. We always use $p = 100$. For every realization there is a corresponding network $\mathbf{f}(\cdot; \mathcal{W}^{(i)})$, which is the network which consists of the unchanged feed-forward connections together with the feedback weights $\mathcal{W}^{(i)}$.

Then, the GA works as follows (Fig. 2):

1. Calculate the *fitness* of every network $\mathbf{f}(\mathbf{X}; \mathcal{W}^{(i)})$, where $\mathcal{W}^{(i)}$ is a member of the *population* Ω . For this define the activation of the feed-forward-feedback network $\psi^{\text{ff+fb}}$, which enters in the calculation of f , by:

$$\psi^{\text{ff+fb}} = \max \left\{ 0, o_{x,y}^{(l,d)} + \sum_{l',d',x',y'}^{l \neq l'} w_{l' \leftarrow l} \psi' \right\}. \quad (12)$$

where o is the raw network output and $\psi' = \max\{0, o_{x',y'}^{(l',d')}\}$ is the feed-forward activation. The symbol $w_{l' \leftarrow l}$ defines a feedback weight.

2. Select with equal probability two networks with different feedback configurations (*parents*) $\mathcal{W}^{(a)}$ and $\mathcal{W}^{(b)}$ ($a \neq b$) from Ω .
3. Cross over both *parents* to combine the information of the *parents* to generate new offspring, by selecting a random index r_1 for each element in $\mathcal{W}^{(a)}$, such that:

$$a = \{w_{l' \leftarrow l_1}^{(a)}, w_{l' \leftarrow l_2}^{(a)}, \dots, w_{l' \leftarrow l_{r_1}}^{(a)}\} \quad (13)$$

and a random index r_2 for each element in $\mathcal{W}^{(b)}$, such that:

$$b = \{w_{l' \leftarrow l_{r_2}}^{(b)}, \dots, w_{l' \leftarrow l_B}^{(b)}\}, \quad (14)$$

where l' annotates a target unit (abbreviated from (l', d', x', y')), l_i are source units for the feedback and “ \leftarrow ” refers to the feedback connection. Now we generate the offspring $\forall w_{l' \leftarrow l}^{(c)} \in \mathcal{W}^{(c)}$:

$$w_{l' \leftarrow l}^{(c)} = a \cup b. \quad (15)$$

If the sub-sets of a and b are overlapping the overlapped values of b are dismissed and those from a are kept. Hence, after this step we have recombined the feedback weights from $\mathcal{W}^{(a)}$ and $\mathcal{W}^{(b)}$ for every target unit.

4. Next, we mutate *all individual weights* $w_{l' \leftarrow l_j}^{(c)}$ of $\mathcal{W}^{(c)}$ with probability $p_m = 0.3$. For this, we convert all $w_{l' \leftarrow l_j}^{(c)}$ into their binary representation, mutating every entry (bit) therein with probability p_m . The resulting mutated weight is then called $w_{l' \leftarrow l_j}^*$ and is transformed back to its floating point representation. The mutated

$$\mathcal{W}^{(c)*} \ni w_{l' \leftarrow l}^{(c)*} = \{w_{l' \leftarrow l_1}^{(c)*}, w_{l' \leftarrow l_2}^{(c)*}, \dots, w_{l' \leftarrow l_B}^{(c)*}\} \quad (16)$$

is placed in Ω^*

5. Repeat steps 2–4 until the new *population* Ω^* has p members.

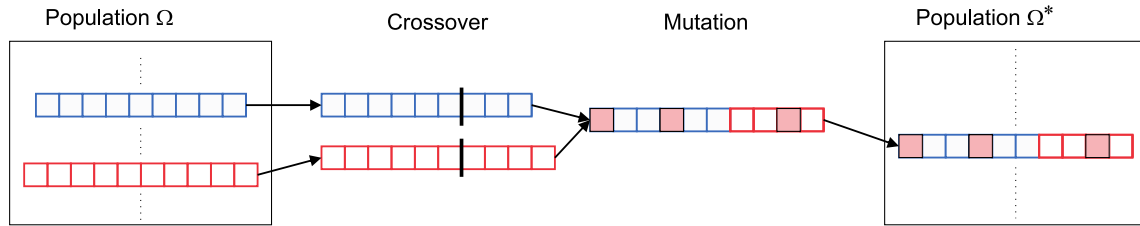


Fig. 2. Abstract representation of the genetic algorithm, starting in the left corner with a starting population Ω . From Ω always two parents are chosen randomly, on each parent a crossover point (black lines in the second column) is chosen randomly. The part on the left side of the crossover point is used for the crossover from the first parent and the right side from the second parent (third column). In the next step some values of the child are mutated by chance (red boxes) and added to the temporal population Ω^* . The new population of the next generation is then defined by the best p values from Ω and Ω^* , based on the fitness function.

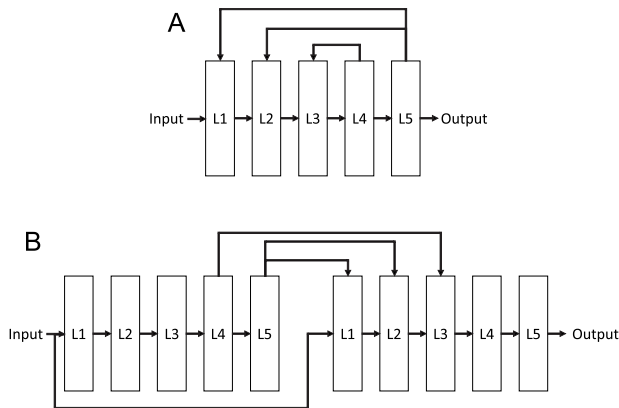


Fig. 3. Schematic representation of the unrolling of the recurrent network. (A) shows the network with feedback connections. (B) shows the unrolled network.

6. Calculate the *fitness* of all networks created from the members in Ω^* .
7. From the parent Ω and child Ω^* population, choose p corresponding networks with the highest *fitness* values to create a final new population Ω^{**} .
8. Set $\Omega = \Omega^{**}$ and go to step 1 and repeat steps 1–7 until the maximal number of *generations* is reached, where we have always used 200 generations.

After those 200 generations, select the best performing network as the final result.

The network output is computed by “unrolling” the recurrent network back to a feed forward network where the layers are cloned, see (Fig. 3).

3. Results

After the genetic algorithm has determined the weights, the network can be used for classification.

3.1. General observations

The AlexNet has eight layers, but only layers 1–5 perform convolutions and form feedback connections. Fig. 4A shows the number of feedback connections made for different values of Φ with a resolution interval of $\Delta\Phi = 0.005$ for this curve. At about 0.9 there is a very sharp kink in the curve and down to about 0.05 the number of connections varies by less than one order of magnitude. When analyzing the connections that are made for $T_{i \rightarrow j} > \Phi = 0.9$, we found that those would consist almost exclusively of feedback between units in adjacent layers, which do not have a feed-forward connection. This aspect shows up also in Fig. 6 and will be discussed there again.

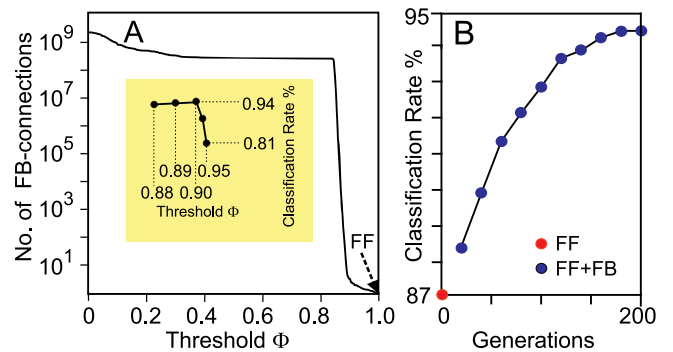


Fig. 4. Performance quantification of the FB-AlexNet. (A) Number of feedback connections made between units for which $T_{i \rightarrow j} > \Phi$. The yellow inset shows the classification performance of the FB-AlexNet in percent for different thresholds Φ on the test data. (B) Development of the classification rate over 200 generations during the genetic algorithm, based on a population of $p = 100$ networks, with mutation rate $p_m = 0.3$.

This quasi-discontinuous behavior of the transfer entropy lends itself to defining $\Phi = 0.9$ as threshold for making feedback connections. This assumption is also backed-up by a control experiment (yellow inset in Fig. 4A). Here we have calculated the classification rate of the FB-AlexNet on the CIFAR-10 data for several values of Φ near the kink and find a sharp performance decline above $\Phi = 0.9$ (drop from 94% to 81%). Exhaustive analysis of this behavior is not possible because of the exploding run-time and memory requirements for treating the huge number of connections that are formed for $\Phi \ll 0.9$.

The network has 2,332,704 connections without feedback and with $\Phi = 0.9$ we get now an additional 1,796,182 connection candidates (77% more). Those can be processed efficiently with 200 generations by the genetic algorithm (Fig. 4B). Note that the randomly initialized feedback weights of the first generation in the genetic algorithm lead on average to a performance decrease by several percent relative to the pure feed-forward AlexNet (data not shown). The genetic algorithm starts with this and leads finally to a performance increase of more than 7% for the CIFAR-10 data set (50,000 training images, 10,000 test images). The final FB-weight distribution contains rather small weights (Fig. 5C).

This way the FB-AlexNet reaches the top-performing group of network architectures (Graham, 2014; Mishkin & Matas, 2015; Springenberg et al., 2014) on the CIFAR-10 data (see Fig. 5A), based on the weights provided by Krizhevsky et al. (2012). This is remarkable, because the existing top-performing networks are far more complex (especially consisting of far more layers) than the AlexNet and current performance improvements are often in the range of less than 1% when a new architecture is proposed.

Will the here observed performance improvement be limited to this specific data set and/or to the network used so far? To address this question, we tested the influence of feedback on

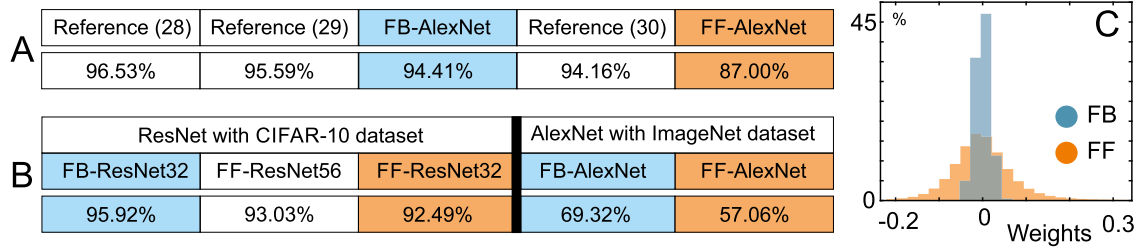


Fig. 5. Performance of FF- and FB-AlexNet (A) Comparison to other methods (Graham, 2014; Mishkin & Matas, 2015; Springenberg, Dosovitskiy, Brox, & Riedmiller, 2014) on the CIFAR-10 dataset. (B) Comparison of the performance for FB- and FF-ResNet architecture on the CIFAR-10 data (left) and for FB- and FF-AlexNet on ImageNet data (right). (C) Weight distribution (in percent) of the FB-AlexNet plotting the values of the feed-forward (orange) and feedback weights (blue). The range of the x-axis was limited to $[-0.2, 0.3]$. A few more connections exist outside this range (in $[-0.4, 0.6]$).

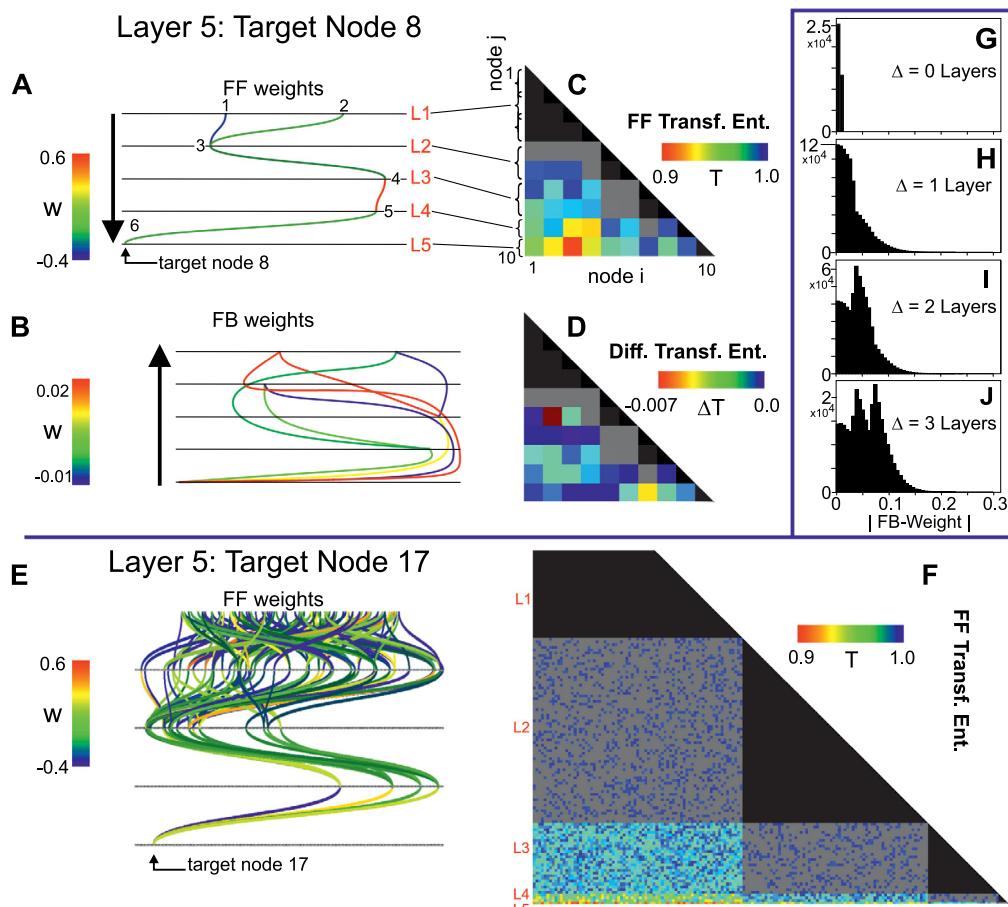


Fig. 6. Detailed analysis of two example convergence trees with different layer-5 target units in the AlexNet. Panels (A–D) Target unit 8. (A) Feed-forward weights $w_{(i) \rightarrow (j)}$. Small numbers refer to the consecutive numbering of units in diagrams C–F. (B) Feedback weights $w_{(j) \rightarrow (i)}$. Feed-forward paths are the same as in panel A. (C) Transfer entropies for $\Phi = 0.9$. (D) Difference of transfer entropies without and with feedback. Panels (E,F) Target unit 17. (E) Feed-forward weights as in (A). (F) Transfer entropies as in (C). (G–J) Distributions of the absolute FB-weight values for Δ -layers distance. Hence, $\Delta = 1$ refers to FB connections between units in layers, where there is one layer in between.

network architectures with a more complex layout and also used a much richer data set.

First, we compared the performance of ResNet on the CIFAR-10 data set with and without feedback connections. The feed-forward structure of the network was trained as described in He et al. (2015). As known, ResNet performs much better than AlexNet and the 32-layer version reaches 92.49% (Fig. 5B, left). Including feedback connections, the network performance increases by 3.43% to 95.92%, which is again to a degree remarkable, because – close to ceiling – usually performance gains are

around 1% only, as also shown here when using a feed-forward ResNet with 56 layers, that has only a performance of 93.03%.

In addition to this, we also used a pre-trained AlexNet (Imagnet, 2017) on the much larger ImageNet data set. The 8-layer AlexNet is quite challenged by this benchmark and achieves 57.06%. Modern, far more complex networks reach here maximally 82.3% (Zoph, Yuret, May, & Knight, 2016). With feedback connections the AlexNet attains 69.32% (Fig. 5B, right), which is an 12.26% increase and quite remarkable for this small network.

3.2. Detailed quantification

It is difficult to understand how feedback changes the complex deep convolutional neural networks, but analyzing subsets of the network – as shown next – can provide some insights why performance has improved. In addition to this we will below provide a statistical analysis using a large set of small (and less complex) traditional networks.

Feed-forward connectivity of individual convergence trees in the AlexNet (incorporating all paths from the input layer to a specific target unit in the output layer) can vary widely, and two characteristic examples are shown in Fig. 6A, E for layer 5 target units named “8” and “17”. The matrix diagrams in panels C and F display the transfer entropies between units along all connected pathways for $\Phi = 0.9$. Gray regions are those where transfer entropy is below Φ and feedback connections are formed between units that have a transfer entropy above than this threshold. Black regions represent intra-layer relations, which are not considered, as there are no lateral connections. Interestingly, gray “pixels”, where $T_{i \rightarrow j} < 0.9$, only occur between adjacent layers. As claimed above (see discussion of Fig. 4A), those are unit pairs, for which there are no feed-forward connections and transfer entropy is, quite intuitively, small between those.

Fig. 6B shows the feedback connections for the convergence tree of target unit 8 (feed-forward connections as in panel A). Note that the maximum feedback weight is about a factor of 100 smaller than the maximum feed-forward weight. The largest feedback weights are assigned for connections between units with biggest layer distances (panel J).

Of specific interest is the development of the transfer entropy comparing the pure feed-forward situation ($T_{i \rightarrow j}^{FF}$) with the one that also contains feedback ($T_{i \rightarrow j}^{FF+FB}$). This is shown in Fig. 6D, plotting the difference in transfer entropy: $T_{i \rightarrow j}^{FF+FB} - T_{i \rightarrow j}^{FF}$. In general, feedback connections in the network yield a drop in the transfer entropy by a quite small amount and in a rather dispersed way. This demonstrates that transfer entropy does not just show some overall decrease, instead some paths are specifically improved by feedback, which are – presumably – those that carry most information for classification.

Fig. 6B suggests that the strongest feedback connections are formed between distant layers, while those between adjacent ones are the weakest. This is quantified in Fig. 6G–J, which confirms this observation. Interestingly, fewer feedback connections exist between adjacent layers (Fig. 6G), because – as discussed – very often $T_{i \rightarrow i+1} < \Phi$ and, if feedback exists, then it is very weak. This is not unexpected. Feedback between adjacent layers is equivalent to an upregulation of the feed-forward connectivity. The genetic algorithm introduces a few small, adjacent-layer FB-connections but essentially it “is satisfied” with the connections that had been found by the error back-propagation that had initially set up the forward connectivity.

3.3. Control experiments

To raise confidence in the above results, several additional control experiments have been performed.

At first we are considering again the Cifar 10 data set for which we trained the AlexNet for each realization from scratch using (Chollet et al., 2015). All layers were initialized with the keras default parameters. We used the default categorical cross-entropy as the loss function, the Adam optimizer (Kingma & Ba, 2014) with default values (learning rate = 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and the amsgrad flag set to false). In addition, we changed the learning rate by a factor of 0.5, if in 10 epochs the accuracy did not improve further and used early stopping after 50 epochs of no improvement was found any longer. If in the first

three epochs the accuracy does not increase by at least 13%, the realization had been re-started with a new initialization. Please note that, different from Krizhevsky et al. (2012), we did not use data augmentation to save massive on training time.

Controls are as follows: (1) First, we kept all target nodes but randomly shuffled the feedback connections that came from the above algorithm, such that the feedback connection between $i \rightarrow j$ changes to $i \rightarrow j'$, where $j \neq j'$, with a certain probability. Fig. 7A shows how the accuracy on the validation data changes when the shuffling probability increases. The continuous drop of this curve shows that the originally introduced feedback connections, which came from the transfer entropy assessment, were always the best.

(2) Second, as another control we wanted to validate whether actually specifying feedback connections by using transfer entropy is central for achieving the here presented results. To do this we created random feedback connections in the FF-AlexNet and optimized their weights with the genetic algorithm. The total number of feedback connections was kept the same as when using transfer entropy. It can be seen (Fig. 7B) that the genetic algorithm under these conditions only yields an improvement of 1.01% (Rnd. GA FB-AlexNet), compared to the FF-AlexNet.

(3) The third control asks whether a similar performance gain could be achieved when keeping the basic network architecture unchanged but increasing the number of intrinsic network parameters to the same level as those found in the FF + FB configuration. Feedback had added about 77% more connections to the convolutional layers and this number needs to be roughly matched. Brute force trial and error was used to find out where in the AlexNet an increase in the number of filters would lead to the greatest advantage. Every network created by this search procedure was trained on the training data from the Cifar 10 data set. Kernels were initialized with the kernel initialization from Saxe, McClelland, and Ganguli (2013) to improve convergence and Adam Kingma and Ba (2014) was used as optimizer with early stopping if accuracy did not improve after 10 epochs. Through this extended search we found that the best-performing extension of the FF-AlexNet took the following shape: The number of filters in the second layer was increased by 30%, to 332 filters, in the third and fourth convolution layers by 50%, to 576 filters each and in the last layer by 35% to 345 filters having 86% more parameters. The whole network then had 11,943,317 trainable parameters. We call this extended network ext-AlexNet in comparison to the regular FF-AlexNet, which has 6,390,040 trainable parameters. In Fig. 7B (FF-AlexNet vs ext-AlexNet) the results of this experiment are shown. It is easy to see that the increase parameters do not lead to any significant increase of the accuracy on the validation data and all other versions of the ext-AlexNet performed worse than this one.

(4) The fourth set of controls uses a different network architecture and different inputs. Several parameters influence the performance of a deep CNN, for example how pre-training is performed, which initial conditions and which learning rates have been used and more. Due to long run-times most if not all studies in deep learning remain shy of statistical evaluations of these parameters. Similar to this, also the results obtained above carry no statistical significance and the question arises whether performance increase due to feedback is generic. To address this issue we performed a statistical evaluation of 36,000 different network configurations using conventional networks with 2 up to 10 hidden layers in a binary classification task. Each layer consists of 50 units and feed-forward as well as feedback connectivity has been determined as before. Input data are 2D-point clouds (Fig. 8A1, B1) separated by a continuous function and the network was trained on 50% of them to find the separator. Tolerance for optimization was 10^{-4} and was reached in all cases. We used Adam as solver with all intrinsic parameters as in the original

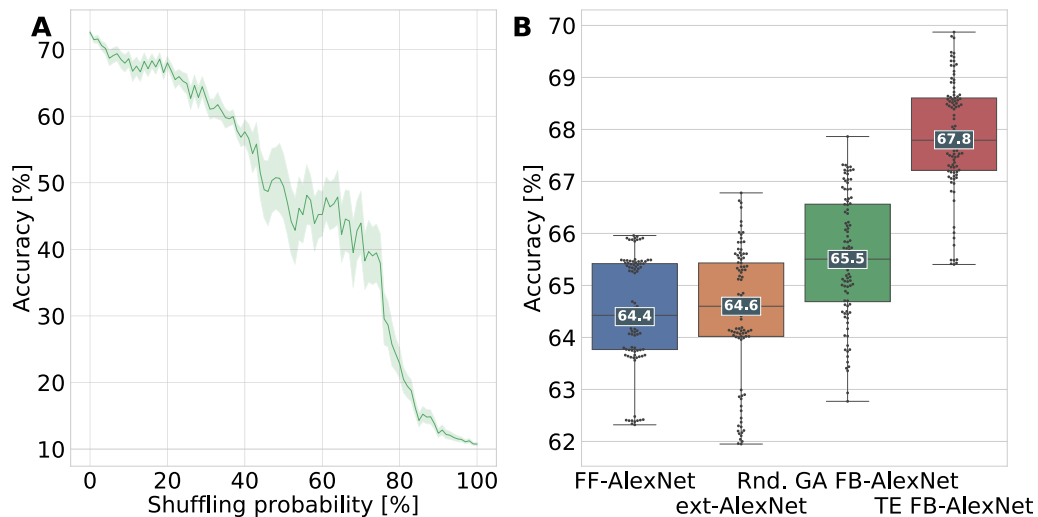


Fig. 7. Control experiments: (A) Shuffling the optimized feedback connections randomly influences the accuracy. The green line (starting at 73%) shows the mean value of the accuracy over 100,000 trials each where the green shade is the variance. (B) Comparison of the accuracy on the validation data of the AlexNet (in blue), extended AlexNet (ext-AlexNet in orange) with 86% more parameters, an AlexNet with random feedback connections where the weights of the random feedback connections were optimized by the genetic algorithm (Rnd. GA FB-AlexNet in green) and the AlexNet with transfer entropy guided feedback connections (TF FB-AlexNet). The results for 100 initializations are shown as a boxplot (median: FF-AlexNet: 64.4%, ext-AlexNet: 64.6%, Rnd. GA FB-AlexNet: 65.4% and TE FB-AlexNet: 67.8%). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

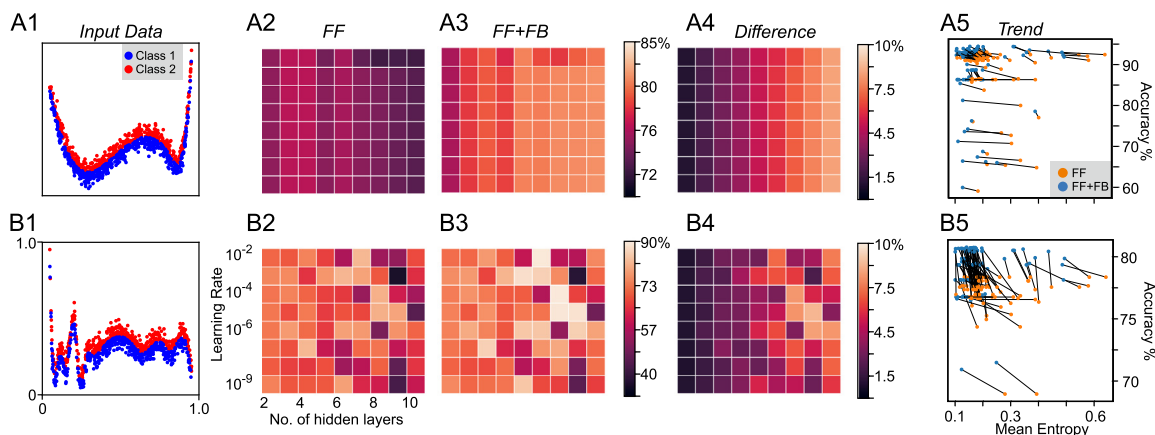


Fig. 8. Performance evaluation of 9 different conventional networks with 8 different learning rates. (Rows A and B) Performance under less (A) and more (B) complex inputs. (Panels 1) 2-D input data. Classification rates of the FF (2) and the FF+FB (3) networks and their differences (4). (Panels 5) Entropy is reduced and performance increases for all networks with feedback. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

paper (Kingma & Ba, 2014). Eight different learning rates and nine different numbers of hidden layers have been used and for every one of these 72 cases we assessed 500 different initial conditions and always selected the best performing case for further analysis. Panels 2 and 3 show the classification success for the FF- and the FF+FB networks. Too high learning rates lead to the characteristic mal-convergence (top rows) known from such networks. Case B is more complex than A and the classification rate shown in panels B2,3 is lower than in A2,3. Networks with more layers generally perform better but sometimes there is a drop. This results from the network beginning to fit the complex, steeper parts of the curves better but losing performance at the less steep parts. Such effects are also known from classification with networks. Interestingly, this effect is almost completely removed by feedback. Very consistently feedback improves performance and larger networks now perform (with few exceptions) better than smaller ones. Percent improvement is visible from panels

4. In panels 5 we show FF- (orange) and their corresponding FF+FB (blue) networks as pairs and one can see that performance improvement goes along with a reduction of the Shannon entropy in the network. Taken together, these results confirm that this type of feedback consistently improves network performance and creates “more orderly” nets.

4. Conclusion

Here we have shown that specific feedback between units in different networks substantially improves performance. This effect has first been found with different deep convolutional architectures and then we have confirmed its robustness with smaller conventional nets. Hence, this framework is neither limited to specific architectures nor to image recognition problems.

Why should one consider using transfer entropy for searching for feedback candidates in such networks? Two general observations can be made from the results of this study, which will

hold also for other architectures. First, there is a decreasing feed-forward convergence towards higher layers, common to most if not all deep convolutional networks. Second, transfer entropy is in general higher between units with larger layer-distances than between neighboring layers (see Fig. 6). This is natural due to the fact that long-range transfer entropy is calculated by conditioning on the intermediate layers. Thus, when using transfer entropy to define feedback, there is a higher probability to form long-range as compared to short-range feedback connections in the network.

If transfer entropy between distant units is small, then this is indicative of a (long) path segment, which is “meaningful” and which ought to be more strongly amplified. Thus, long-range feedback connections should be the strongest and this is in general found. While, on the other hand, pairs of units with high transfer entropy in neighboring layers are anyhow directly connected (by their feed-forward link). Their amplification should be small in order to prevent run-away activity. The genetic algorithm finds this solution as shown in Fig. 6G–I, but all connections are very small (Fig. 5C).

This is reminiscent of many feedback pathways in the vertebrate brain (Gilbert & Li, 2013; Spillmann et al., 2015). Particularly the interleaving of feed-forward with feedback connectivity discussed by several authors (Markov et al., 2014; Yamins & DiCarlo, 2016) suggests that there is also an interleaved information flow happening. It is, thus, tempting to speculate that a similar principle – an evaluation of the relevance of the different feed-forward pathways – might have been a phylo- or ontogenetic driving force for the design of different feedback structures in real neural systems. Transfer entropy can be used to measure how significantly neurons interact (Lizier et al., 2011; Vicente et al., 2011). It is, thus, an interesting question to what degree neural systems might have used this or a similar type of information to distinguish and amplify important relative to less important processing pathways.

Acknowledgments

We thank Dr. D. Miner and Dr. M. Tamosiunaite for valuable comments on the manuscript. Many thanks also go to Dr. D. Yamins who provided valuable inputs to an earlier version of this paper. The research leading to these results has received funding from the European Community’s Horizon 2020 Programme under grant agreement no. 732266, Plan4Act.

References

- Barker, J. (1958). Simulation of genetic systems by automatic digital computers. *Australian Journal of Biological Sciences*, 11(4), 603–612.
- Bengio, Y., Lee, D.-H., Bornschein, J., Mesnard, T., & Lin, Z. (2015). Towards biologically plausible deep learning. arXiv:1502.04156.
- Chollet, F., et al. (2015). Keras. <https://keras.io>.
- Cichy, R. M., Khosla, A., Pantazis, D., Torralba, A., & Oliva, A. (2016). Deep neural networks predict hierarchical spatio-temporal dynamics of human visual object recognition. arXiv:1601.02970.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer vision and pattern recognition, 2009. CVPR 2009. IEEE conference on* (pp. 248–255).
- Ernst, M. R., Triesch, J., & Burwick, T. (2019). Recurrent connections aid occluded object recognition by discounting occluders. In *Artificial neural networks and machine learning – ICANN 2019: Image processing* (pp. 294–305). http://dx.doi.org/10.1007/978-3-030-30508-6_24.
- Gilbert, C. D., & Li, W. (2013). Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5), 350–363.
- Graham, B. (2014). Fractional max-pooling. arXiv:1412.6071.
- Güçlü, U., & van Gerven, M. A. J. (2015). Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. *Journal of Neuroscience*, 35(27), 10005–10014.
- He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep residual learning for image recognition. arXiv:1512.03385.
- Hermans, M., & Schrauwen, B. (2013). Training and analysing deep recurrent neural networks. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 26* (pp. 190–198). Curran Associates, Inc..
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.
- Holland, J., & Goldberg, D. Genetic algorithms in search, optimization and machine learning, Massachusetts: Addison-Wesley.
- dataset (2017). Imagenet weights for alexnet. Accessed: 18 June 2018. URL http://files.heuritech.com/weights/alexnet_weights.h5.
- Khaligh-Razavi, S. M., & Kriegeskorte, N. (2014). Deep supervised, but not unsupervised models may explain IT cortical representation. *PLoS Computational Biology*, 10(11), e1003915.
- Kietzmann, T. C., McClure, P., & Kriegeskorte, N. Deep neural networks in computational neuroscience. In Oxford Research Encyclopedia of Neuroscience, <http://dx.doi.org/10.1093/acrefore/9780190264086.013.46>.
- Kietzmann, T. C., Spoerer, C. J., Sörensen, L. K. A., Cichy, R. M., Hauk, O., & Kriegeskorte, N. (2019). Recurrence is required to capture the representational dynamics of the human visual system. *Proceedings of the National Academy of Sciences*, 116(43), 21854–21863, URL <https://www.pnas.org/content/116/43/21854>.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.
- Koza, J. (1994). *Genetic programming: On the programming of computers by means of natural selection*. Cambridge: Bradford Book.
- Krizhevsky, A., Nair, V., & Hinton, G. Cifar-10 (Canadian institute for advanced research) Technical Report. URL <http://www.cs.toronto.edu/kriz/cifar.html>.
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, & K. Q. Weinberger (Eds.), *Advances in neural information processing systems 25* (pp. 1097–1105). Curran Associates, Inc..
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436–444.
- Liao, Q., & Poggio, T. (2016). Bridging the gaps between residual learning, recurrent neural networks and visual cortex. arXiv:1604.03640.
- Lillicrap, T. P., Cownden, D., Tweed, D. B., & Akerman, C. J. (2016). Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7, 13276.
- Lizier, J. T., Heinzle, J., Horstmann, A., Haynes, J. D., & Prokopenko, M. (2011). Multivariate information-theoretic measures reveal directed information structure and task relevant changes in fMRI connectivity. *Journal of Computational Neuroscience*, 30(1), 85–107.
- Marblestone, A. H., Wayne, G., & Körding, K. P. (2016). Toward an integration of deep learning and neuroscience. *Frontiers in Computational Neuroscience*, 10, 94.
- Markov, N. T., Vezoli, J., Chameau, P., Falchier, A., Quilodran, R., Huissoud, C., et al. (2014). Anatomy of hierarchy: feedforward and feedback pathways in macaque visual cortex. *Journal of Comparative Neurology*, 522(1), 225–259.
- Mishkin, D., & Matas, J. (2015). All you need is a good init. arXiv:1511.06422.
- Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).
- Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533–536.
- Salakhutdinov, R., & Hinton, G. E. (2009). Deep Boltzmann Machines. In *Proceedings of the twelfth international conference on artificial intelligence and statistics, AISTATS 2009* (pp. 448–455). Florida, USA: Clearwater Beach.
- Saxe, A. M., McClelland, J. L., & Ganguli, S. (2013). Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. arXiv:1312.6120.
- Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61, 85–117. <http://dx.doi.org/10.1016/j.neunet.2014.09.003>.
- Schreiber, T. (2000). Measuring information transfer. *Physical Review Letters*, 85(2), 461.
- Spillmann, L., Dresch-Langley, B., & Tseng, C. H. (2015). Beyond the classical receptive field: The effect of contextual stimuli. *Journal of Vision*, 15(9), 7.
- Spoerer, C. J., Kietzmann, T. C., & Kriegeskorte, N. Recurrent networks can recycle neural resources to flexibly trade speed for accuracy in visual recognition, bioRxiv. URL <https://www.biorxiv.org/content/early/2019/06/22/677237>.
- Spoerer, C. J., McClure, P., & Kriegeskorte, N. (2017). Recurrent convolutional neural networks: a better model of biological object recognition. *Frontiers in Psychology*, 8, 1551.
- Springenberg, J. T., Dosovitskiy, A., Brox, T., & Riedmiller, M. (2014). Striving for simplicity: The all convolutional net. arXiv:1412.6806.
- Stanley, K. O., & Miikkulainen, R. (2002). Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2), 99–127.
- Stollenga, M. F., Masci, J., Gomez, F., & Schmidhuber, J. (2014). Deep networks with internal selective attention through feedback connections. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, & K. Weinberger (Eds.), *Advances in neural information processing systems 27* (pp. 3545–3553). Curran Associates, Inc..

- Tang, H., Schrimpf, M., Lotter, W., Moerman, C., Paredes, A., Ortega Caro, J., et al. (2018). Recurrent computations for visual pattern completion. *Proceedings of the National Academy of Sciences*, 115(35), 8835–8840, URL <https://www.pnas.org/content/115/35/8835>.
- Varadarajan, K. M., & Vincze, M. (2013). Parallel deep learning with suggestive activation for object category recognition. In M. Chen, B. Leibe, & B. Neumann (Eds.), *Lecture Notes in Computer Science: vol. 7963, Computer vision systems* (pp. 354–363). Springer.
- Vicente, R., Wibral, M., Lindner, M., & Pipa, G. (2011). Transfer entropy—a model-free measure of effective connectivity for the neurosciences. *Journal of Computational Neuroscience*, 30(1), 45–67.
- Wollstadt, P., Lizier, J., Vicente, R., Finn, C., Martinez-Zarzuela, M., Mediano, P., et al. (2019). Idtxl: The information dynamics toolkit xl: a python package for the efficient analysis of multivariate information dynamics in networks. *Journal of Open Source Software*, 4(34), 1081. <http://dx.doi.org/10.21105/joss.01081>.
- Yamins, D. L., & DiCarlo, J. J. (2016). Using goal-driven deep learning models to understand sensory cortex. *Nature Neuroscience*, 19(3), 356–365.
- Zoph, B., Yuret, D., May, J., & Knight, K. (2016). Transfer learning for low-resource neural machine translation. [arXiv:1604.02201](https://arxiv.org/abs/1604.02201).

2.4.1 Conclusions from (Herzog et al. 2020a)

In *(Herzog et al. 2020a) it could be shown how that adding specific feedback connections between individual units improves performance of the CNN substantially. The following experiment demonstrates how the application of the algorithm from *(Herzog et al. 2020a) can improve the performance of the CAE from *(Herzog et al. 2021c). For this purpose, the data from *(Herzog et al. 2021c) was used and the feedback approach from *(Herzog et al. 2020a) was applied to exactly the same architecture and already trained weights as used in *(Herzog et al. 2021c). Then the three evaluations for the noisy, blurred and under-sampled data cases were re-evaluated. In the following evaluations the comparison with the ESN has been omitted in order to provide a better overview. Starting with the re-evaluation of the noisy data case. It is easy to see that the addition of feedback has further reduced

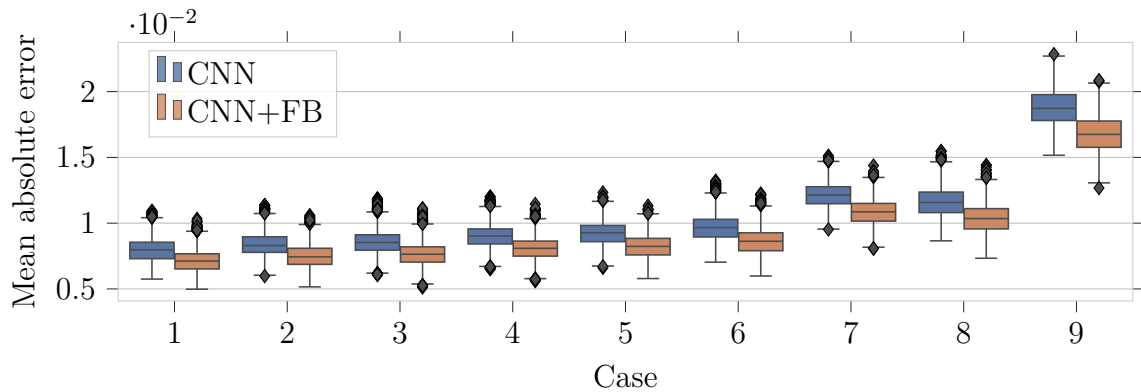


Figure 2.3: Comparison between CAE and CAE with feedback connections (CAE+FB) performance with noisy input data showing boxplots of mean absolute errors for different noise levels $p \in [0.1, 0.2, \dots, 0.9]$. Each discrete value on the x-axis is assigned to the boxes of the CAE and CAE+FB, where the CNN boxplots are coloured in blue and CAE+FB boxplots are coloured in orange. The line in the middle of the box is the median value, the box itself is the interquartile range (IQR), defined as the difference between the 75th ($Q3$) and 25th ($Q1$) percentiles. The outgoing whiskers from the box are defined as $Q1 - 1.5IQR$ and $Q3 + 1.5IQR$. The points outside of the whiskers are considered as outliers. This illustration corresponds to *(Herzog et al. 2021c, figure 10).

the mean absolute error. The mean absolute error across all cases was reduced by 5.56% in the mean with a standard deviation of 0.22%. The same experiments were also carried out for the blurred data case. Here, too, the mean absolute error across

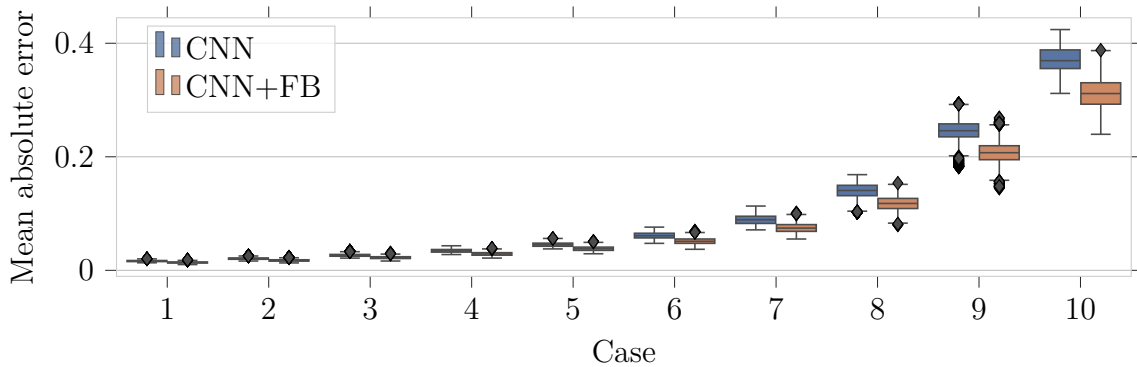


Figure 2.4: Mean absolute errors of reconstructions using CAE and CAE+FB for the blurred data case of the low-pass filter parameter m . Same representation as in figure 2.3 and corresponding to *(Herzog et al. 2021c, figure 12).

all cases was reduced by 9.13% in the mean with a standard deviation of 0.37%. Which has even led to a larger improvement than in the first case.

Finally, the approach from *(Herzog et al. 2020a) was also applied to the case of the under-sampled data. Interestingly, adding feedback to this architecture lead to no noticeable improvement. This may be because the architecture is only one part of the CAE and not a whole CAE, or because the existing information in the input data is not sufficient for better reconstructions, even when feedback is added. The mean absolute error across all cases was reduced by 0.5% in the

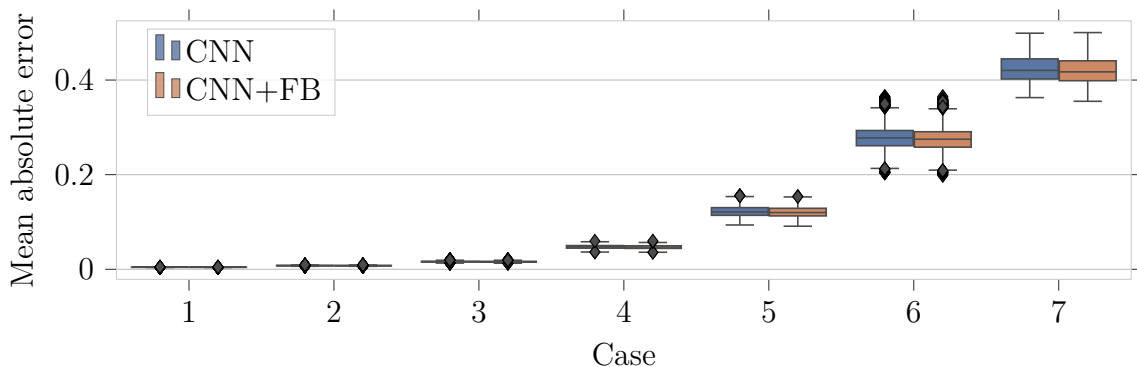


Figure 2.5: Mean absolute errors of reconstructions using CAE and CAE+FB for the case of under-sampled data. Same representation like in figure 2.3 and corresponds to *(Herzog et al. 2021c, figure 15).

mean with a standard deviation of 0.09%. All in all, these are only preliminary

studies and one must generally ask whether the improvements through feedback justify the additional computing effort. For the cases shown here, it took longer to calculate the feedback than to train the entire network. This is partly due to the fact that the necessary calculation is not carried out using optimised code. Furthermore, very large amounts of data accumulate that do not easily fit into the random-access memory of a workstation. The last point of criticism is that, the way the feedback is calculated also causes the problem that the network has to be evaluated twice. In the next chapter the feedback approach was not used, but further studies will follow in the future.

In summary, it is possible to say that the CAE architecture is a very powerful tool that allows many pre- and post-processing steps and is comparatively easy to use. On the other hand, a sufficient amount of data has to be available. Unfortunately it is not possible to say how much data is really sufficient. Since only the evaluation on the test data can provide a comparably reliable statement about the quality of the approximated function by the ANN.

A man provided with paper, pencil, and rubber, and subject to strict discipline, is in effect a universal machine.

—Alan Turing, reprinted in (Tribus et al. 1971b)

3

Spatio-temporal data prediction

Contents

3.1	Introduction to spatio-temporal prediction	61
3.2	Stochastic modelling	62
3.2.1	Stochastic process	62
3.2.2	Graphical models	64
3.3	Spatio-temporal prediction of non-linear dynamics . .	67
3.3.1	Publication: (Herzog et al. 2019)	67
3.3.2	Publication: (Herzog et al. 2018)	79
3.3.3	Publication: (Herzog et al. 2020b)	90
3.4	Summary and outlook	100

3.1 Introduction to spatio-temporal prediction

In the previous chapter, the convolutional autoencoder (CAE) architecture has been introduced. In this chapter, this architecture will be extended by a probabilistic component. This extension should enable the CAE to make long-term predictions from the data on which it has been trained. The chapter starts with some basic definitions in section 3.2 about stochastic modelling, introduces the concept of a *stochastic process* up to *random fields* to provide the basic concepts to understand the method of *graphical models* presented in subsection 3.2.2. This should provide the necessary basis to better understand the approach presented in section 3.3.

3.2 Stochastic modelling

For exact mathematical modelling it is necessary that all relevant aspects of the system to be described are known. If information about the system is missing, exact modelling is not possible and one often resorts to approximations or probabilities in order to describe the system, nevertheless. In the second case in particular, one speaks of *stochastic modelling*. Based on a set of statistical assumptions a probability of an event can be calculated. In this section, the basic concepts of stochastic modelling will be discussed, to provide a better intuition for the ansatz presented later. The mathematical principles and terminology are defined in the appendix B.

Stochastic models often consist of a set of *random vectors* (see appendix B.1.1), describing a *stochastic process*. A *stochastic process* is a mathematical model for a real process that is random and depends on a parameter (usually time). The concept of the *stochastic process* shall be presented in more detail next.

3.2.1 Stochastic process

One way of looking at time ordered data, which are used for training the presented approach, is to interpret them as a *stochastic process*. The model assumption is that given probability space $(\Omega, \mathfrak{A}, P)$, in which each $\omega \in \Omega$ is assigned a random function $x(t, \omega), t \in T$, where $T \subseteq \mathbb{R}$ is the time. Then a *stochastic process* can be defined as:

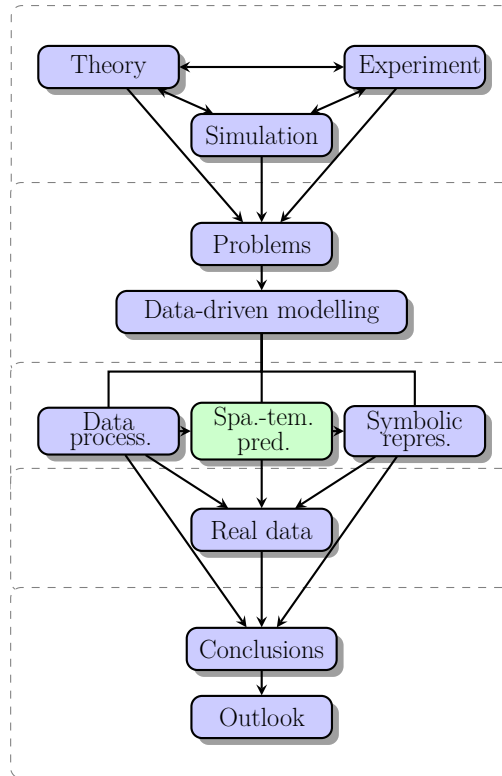


Figure 3.1: Structure overview: Spatio-temporal data prediction chapter 3

Definition 3.2.1 (Stochastic process). Let $T \subseteq \mathbb{R}$. A family of real-valued random variables $\{X(t) \mid t \in T\}$ on a joint probability space $(\Omega, \mathfrak{A}, P)$ is called a *stochastic process*.

So $X(t, \omega) : T \times \Omega \rightarrow \mathbb{R}$ is a random variable for a fixed $t \in T$ and a function of t for a fixed $\omega \in \Omega$. For each n time points $t_1, \dots, t_n \in T$ one obtains the *joint distribution* of $(X(t_1), \dots, X(t_n))^T$ by the *joint distribution function* $F_{(X(t_1), \dots, X(t_n))}(x_1, \dots, x_n)$ (see appendix B.1.2). Considering real cases to be *stochastic processes* it is mandatory to calculate probabilities, to do so underlying distributions are needed to be assumed. For the distributions in this work, it should always be assumed that the underlying distributions belong to the *exponential family*, which is defined as follows

Definition 3.2.2 (Exponential family). For density (see appendix B.1.3) $p(\mathbf{x} \mid \boldsymbol{\theta})$, with $\mathbf{x} = (x_1, \dots, x_m)^T \in \mathbb{R}^m$ and $\boldsymbol{\theta} = (\theta_1, \dots, \theta_d)^T \in \mathbb{R}^d$, is said to be in the exponential family if it is of the form

$$\begin{aligned} p(\mathbf{x} \mid \boldsymbol{\theta}) &= \frac{1}{Z(\boldsymbol{\theta})} h(\mathbf{x}) \exp[\boldsymbol{\theta}^T \phi(\mathbf{x})] \\ &= h(\mathbf{x}) \exp[\boldsymbol{\theta}^T \phi(\mathbf{x}) - A(\boldsymbol{\theta})] \end{aligned}$$

where

$$\begin{aligned} Z(\boldsymbol{\theta}) &= \sum_{i=1}^m h(x_i) \exp[\boldsymbol{\theta}^T \phi(x_i)] \\ A(\boldsymbol{\theta}) &= \log Z(\boldsymbol{\theta}). \end{aligned}$$

$h(\mathbf{x})$ is the a scaling constant, $Z(\boldsymbol{\theta})$ is the partition function, $\phi(\mathbf{x}) \in \mathbb{R}^d$ is a vector of sufficient statistics, $A(\boldsymbol{\theta})$ the log partition function and $\boldsymbol{\theta}$ are the so called *natural parameters*.

This assumption is realised in the following as a specific form of a *graphical model*. Based on the brief introduction to the stochastic concepts introduced in this section and the appendix B, *graphical models* shall be introduced next.

3.2.2 Graphical models

The CAE presented in chapter 2 is to be extended by a *graphical model* to enable the long term prediction of spatial-temporal dynamics. Generally speaking *graphical models* are *graphs* (see appendix B.3.1) whose nodes are *random variables* and in which the absence of edges between these nodes indicates their independence (see appendix B.2.1). The use of *graphical models* allows setting up models which can deal with complex inputs and also handle some uncertainty. In particular, the structure as a *graph* can be used very well to combine with ANNs, as will be shown in later in the presented publications. Starting with the definition of a *graphical model*.

Definition 3.2.3 (Graphical model (GM)). A graphical model represents a *joint distribution* (see appendix B.1.2) by a graph $G = (\mathcal{V}, \mathcal{E})$, the nodes \mathcal{V} represent random variables. The edges \mathcal{E} represent conditional dependence assumptions.

Definition 3.2.4 (Clique). For an *undirected graph* (see B.3.2), a *clique* is a set of nodes that are all neighbours of each other. A clique is called maximal if its size cannot be increased without losing the clique property

Definition 3.2.5. A distribution $p(\mathbf{x})$ from $\mathbf{X} = \{X_1, \dots, X_A\}^T$ factorizes according to a factor graph \mathcal{F} if there exists a set of local functions/factors Ψ_a such that p can be written as

$$p(\mathbf{x}) = Z^{-1}(\mathbf{x}) \prod_{a=1}^A \Psi_a(\mathbf{x}_a).$$

The following is the extension to the factor graph.

Definition 3.2.6 (Factor graph, based on (Koller et al. 2009)). A factor graph \mathcal{F} is an *undirected graph* (see B.3.2) $\mathcal{F} = \{\mathcal{V}_n, \mathcal{V}_\Psi, \mathcal{E}\}$ consisting of two types of nodes: the variable nodes \mathcal{V}_n and factor nodes \mathcal{V}_Ψ . Edges \mathcal{E} are only present between \mathcal{V}_n and \mathcal{V}_Ψ . \mathcal{F} is parameterized by a set of factors, where each factor node $\Psi \in \mathcal{V}_\Psi$ is associated with one factor representing a set of neighbouring variables of Ψ in the graph. A distribution p factorizes over \mathcal{F} if it can be represented as a set of factors.



Figure 3.2: Example of a factor graph with three variables a, b, c , denoted by circles and factor nodes denoted by squares. In the factor graph a) only one factor Ψ_1 is used. In the factor graph b) three factors Ψ_1, Ψ_2, Ψ_3 are used. The induced network for both is a clique over a, b, c according to figure B.1 a). The shaded boxes Ψ_1, \dots, Ψ_3 are factor nodes. The set of all distributions $p(a, b, c)$ over the three variables a, b, c can be factorized as $p(a, b, c) = \Psi_1(a, b, c)$ for graph a) and $p(a, b, c) = \Psi_1(a, b)\Psi_2(b, c)\Psi_3(a, c)$ for graph b).

The following definitions are based on (Sutton et al. 2012). X is a set of observable *input variables* and Y set of *output variables*, which shall be predicted. An arbitrary assignment to X is denoted by a vector \mathbf{x}

Definition 3.2.7 (Conditional random field (CRF), based on (Sutton et al. 2012)). Let \mathcal{F} be a factor graph over X and Y . (X, Y) is a conditional random field if for any value \mathbf{x} of X , the distribution $p(\mathbf{y} \mid \mathbf{x})$ factorizes according to \mathcal{F} .

If $\mathcal{V}_\Psi = \{\Psi_a\}$ is the set of factors in \mathcal{F} , then the conditional distribution for a CRF is

$$p(\mathbf{y} \mid \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{|\mathcal{V}_\Psi|} \Psi_a(\mathbf{y}_a, \mathbf{x}_a) \text{ with} \quad (3.1)$$

$$\Psi_a(\mathbf{y}_a, \mathbf{x}_a) = \exp\left(\sum_k \theta_{ak} f_{ak}(\mathbf{y}_a, \mathbf{x}_a)\right). \quad (3.2)$$

f_{ak} are the *feature functions*, which allow transformation between the variables and k is the index that enumerates the number of feature functions per factor Ψ_a . The choice of these feature functions is discussed later in the next publication included in this thesis, which is an important component that makes good prediction possible. An example of what a graph for such a CRF would look like can be found in figure 3.3. It is easy to see that the graph is much more accessible than the equations 3.1 and 3.2.

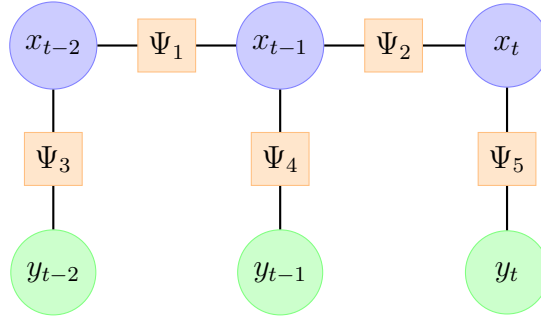


Figure 3.3: A CRF with three variables x_{t-2}, \dots, x_t which can emit three variables y_{t-2}, \dots, y_t . The same symbols are used as in figure 3.2. The circles in green symbolise the desired output of the prediction.

To reduce the number of possible parameters during training, the factor graph \mathcal{F} is further partitioned into $\mathcal{C} = \{C_1, C_2, \dots, C_P\}$, where each C_p is a so called *clique template*.

Definition 3.2.8 (Clique template, based on (Sutton et al. 2012)). A clique template is a set of factors sharing a set of feature functions $\{f_{pk}(x_c, y_c)\}_{\forall k \in \Psi_c}$ with a set of corresponding parameters $\theta_p \in \mathfrak{R}^{K(p)}$.

A CRF consisting of clique templates is given by:

$$p(\mathbf{y} | \mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(x_c, y_c; \theta_p), \quad (3.3)$$

where each clique templated factor is parameterized as

$$\Psi_c(x_c, y_c; \theta_p) = \exp \left\{ \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(x_c, y_c) \right\}. \quad (3.4)$$

The normalisation function is

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\Psi_c \in C_p} \Psi_c(x_c, y_c; \theta_p). \quad (3.5)$$

This concludes the definition of the CRF. As may be evident from the definitions 3.2.7 and 3.2.8 including figure 3.3, CRFs can be structured in arbitrary architectures similar to ANNs. The CRFs used in the following publications were built with the intention that each variable Y is described by a *Gaussian process* (for a definition and some mathematical properties see appendix B.4.1).

3.3 Spatio-temporal prediction of non-linear dynamics

Together with the definitions from chapter 2, all parts are now introduced to understand the method in Herzog, S., Wörgötter, F., and Parlitz, U. (2019). “Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos”. In: *Chaos (Woodbury, N.Y.)* 29.12. ANNs and CRFs are both discriminatively trained probabilistic models. The idea of combining ANNs and CRFs was based on the hope of being able to use the strengths of both approaches with the intention that the combination would behave better than the individual components. The ANN, more precisely the CAE, should encode the data into a form that can then be predicted particularly well by the CRF. In the past, different architectures and types of ANNs have been investigated to predict complex systems. For example ANNs were used to approximate the steady flow (Guo et al. 2016; Ribeiro et al. 2020) or more general computational fluid dynamics (CFD) tasks like in (Musil et al. 2019), but usually these were processes that were little (or not at all) chaotic. By using *Gaussian process regressions* (Wan et al. 2017) was more successful to forecast data from a chaotic dynamical system. The most successful prediction on this type of data, until our contribution *(Herzog et al. 2019), was achieved by Pathak et al. (2018).

3.3.1 Publication: (Herzog et al. 2019)

In the following paper, we presented our modified CRF and the integration into the latent space of the CAE. The performance of the approach is compared with the results from (Pathak et al. 2018).

Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos

Cite as: Chaos **29**, 123116 (2019); <https://doi.org/10.1063/1.5124926>

Submitted: 18 August 2019 . Accepted: 14 November 2019 . Published Online: 12 December 2019

S. Herzog,  F. Wörgötter, and  U. Parlitz

COLLECTIONS

Paper published as part of the special topic on [When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics MACL2020](#)



View Online



Export Citation



CrossMark

ARTICLES YOU MAY BE INTERESTED IN

[Bayesian framework for simulation of dynamical systems from multidimensional data using recurrent neural network](#)

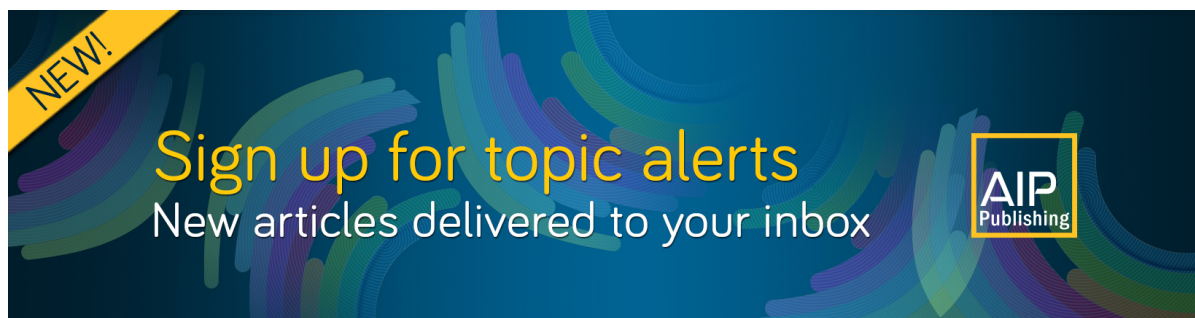
Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 123115 (2019); <https://doi.org/10.1063/1.5128372>

[Forecasting chaotic systems with very low connectivity reservoir computers](#)

Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 123108 (2019); <https://doi.org/10.1063/1.5120710>

[Using machine learning to predict extreme events in the Hénon map](#)

Chaos: An Interdisciplinary Journal of Nonlinear Science **30**, 013113 (2020); <https://doi.org/10.1063/1.5121844>



NEW!

Sign up for topic alerts
New articles delivered to your inbox

AIP
Publishing



Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos

Cite as: Chaos **29**, 123116 (2019); doi: 10.1063/1.5124926

Submitted: 18 August 2019 · Accepted: 14 November 2019 ·

Published Online: 12 December 2019



View Online



Export Citation



CrossMark

S. Herzog,^{1,2,3} F. Wörgötter,²  and U. Parlitz^{1,3,4,a)} 

AFFILIATIONS

¹Max Planck Institute for Dynamics and Self-Organization, Am Fassberg 17, 37077 Göttingen, Germany

²Third Institute of Physics and Bernstein Center for Computational Neuroscience, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

³DZHK (German Centre for Cardiovascular Research), partner site Göttingen, Robert-Koch-Str. 42a, 37075 Göttingen, Germany

⁴Institute for the Dynamics of Complex Systems, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

Note: This paper is part of the Focus Issue, “When Machine Learning Meets Complex Systems: Networks, Chaos and Nonlinear Dynamics.”

^{a)}Electronic mail: ulrich.parlitz@ds.mpg.de

ABSTRACT

We present an approach for data-driven prediction of high-dimensional chaotic time series generated by spatially-extended systems. The algorithm employs a convolutional autoencoder for dimension reduction and feature extraction combined with a probabilistic prediction scheme operating in the feature space, which consists of a conditional random field. The future evolution of the spatially-extended system is predicted using a feedback loop and iterated predictions. The excellent performance of this method is illustrated and evaluated using Lorenz-96 systems and Kuramoto-Sivashinsky equations of different size generating time series of different dimensionality and complexity.

Published under license by AIP Publishing. <https://doi.org/10.1063/1.5124926>

With the advent of novel measurement devices like dense sensor networks or high speed high resolution cameras, large datasets are available to describe complex dynamics in spatially-extended systems. Using such spatiotemporal time series to forecast high-dimensional chaotic processes remains a challenge because methods from nonlinear time series analysis for reconstructing the dynamics using delay coordinates cannot be applied straightforwardly due to the extremely high dimension of the required delay embedding space. This difficulty can be overcome by nonlinear dimension reduction and a suitable factorization of the remaining multivariable conditional probability distributions providing forecasts based on historic data. This hybrid concept has been implemented using a convolutional autoencoder combined with an extended conditional random field modeling the time evolution. When applied to different very hyperchaotic spatiotemporal benchmark time series from the literature, this prediction method turns out to be very effective.

I. INTRODUCTION

Development of explicit mathematical models for complex physical phenomena, while desirable, often remains difficult or unfeasible. This is, in particular, true for high-dimensional, nonlinear dynamical systems for which many times adequate models based on first principles do not exist. As a result, it remains hard or impossible to predict the temporal development of such systems in any reliable way. This has led to attempts that use implicit data-driven descriptions of such systems, for example, employing artificial neural networks. For example, in the field of fluid dynamics, artificial neural networks⁴⁶ were used early on to introduce reduced-order models.⁴⁸ Over time, more and more approaches have been proposed for generating models from (training) data including autoregressive models³ or adaptive fuzzy rule-based models.² The authors in Ref. 28 introduced a tree-structured, recurrent switching linear dynamical system. This probabilistically models the multiscale property of the observed data through a hierarchy of locally linear dynamic elements

that jointly approximate the global nonlinear dynamics of the system. Another current approach introduced in Ref. 15 treats time series prediction by using local state reconstructions,³⁰ dimension reduction, and nearest neighbor methods for local modeling. Furthermore, there are approaches that use echo state networks (ESNs) with very good results.³¹ In Ref. 33, possible advantages of deep learning approaches over reservoir computing are shown, where a time-delay reservoir is outperformed by a deep learning network by at least an order of magnitude. Also, investigations were made with long short-term memory (LSTM) networks, which are prominent recurrent neural networks in the field of deep learning, where the LSTM uses the short-term history of the reduced-order variable to predict the state derivative and uses it for one-step prediction.⁴⁷ The approach presented below employs a convolutional autoencoder⁴ for dimension reduction and feature extraction combined with a probabilistic prediction scheme to learn the dynamics of the considered systems only using recorded data.¹¹ Our goal is to show that, after learning, this model will describe data in different systems precisely and with a high level of predictability. This will be demonstrated using the Lorenz-96 system²⁵ as well as the Kuramoto-Sivashinsky equations,^{21,40,41} where we can show that our approach substantially increases the prediction horizon compared to the state of the art.³¹

II. METHODS

In data-driven modeling, mathematical models are not based on first principles (e.g., Newton's laws, Maxwell's equations, etc.) but are directly derived from experimental data. In the following, we present a method¹¹ which combines a convolutional autoencoder (AE)⁴ for feature extraction and dimension reduction with an extended version of conditional random fields (CRFs)²² in order to model the properties of temporal sequences.

A. Artificial neural networks

The structure of artificial neural networks is divided into three parts. First, the input data are recorded in an input layer. Then, the recorded data are processed by an arbitrary number of hidden layers and in the third step this processed data is passed to an output layer, which maps the data to a descriptive space (e.g., classification^{10,20}) or a generative space (e.g., super resolution⁷). Hereby, the hidden layers can have very different functions; for the algorithm presented in the following, the relevant parts are

1. Convolutional layers: Convolution of the input by a kernel sliding over the input. The numbers of rows and columns of the kernel are hyperparameters; in this work, they are set to be (3×3) .
2. Batch normalization layer: Normalization of the activations of the previous layer during training and for each batch. Batch normalization allows the use of higher learning rates, being computationally more efficient, and also acts as a regularizer.¹⁴
3. Leaky ReLU²⁶ layer: Leaky version of a rectified linear unit (ReLU),⁹ such that

$$v(x) = \begin{cases} \alpha x & \text{for } x < 0, \\ x & \text{for } x \geq 0. \end{cases}$$

4. Max pooling layer: Sample-based operation for discretization based on a kernel that slides over the input like the convolutional operator but only the maximum value of the kernel is passed to the next layer. Width and height of the kernel are hyperparameters (in this work 2×2). In contrast to the convolutional layer, a pooling layer is not trainable.
5. Dropout layer: Regularization method to prevent overfitting where during training some weights are set randomly to zero.⁴³ In this work, the probability of setting the weights to zero is 0.4.

A network is called feed forward network (FFN) if only the inputs from previous layers are used in a present layer. A FFN with convolutional layers is called a convolutional neural network (CNN) introduced in Ref. 23.

Let $X \in \mathcal{X} \subset \mathbb{R}^d$ and $Y \in \mathcal{Y} \subset \mathbb{R}$ be two random variables and $Y = f(X)$ for some unknown function f . Given samples $\{(x^{(r)}, y^{(r)})\}_{r=1, \dots, R}$ drawn from the joint distribution of X and Y , the goal is to find the mapping $\hat{f}: \mathcal{X} \mapsto \mathcal{Y}$ which minimizes the expected loss by a given loss function $\mathcal{L}: \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$, leading to:

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \mathbb{E}[\mathcal{L}(Y, f(X))], \quad (1)$$

where \mathcal{F} is a restricted function space and \mathbb{E} the expected value. CNNs are supposed to find solutions for Eq. (1).

A CNN with depth L consists of convolutional layers given by

$$x^{(l)} = h^{(l)}(x) = v(W^{(l)} * h^{(l-1)}(x) - b^{(l)}), \quad l = 1, \dots, L, \quad (2)$$

where $x^{(l)}$ is the output of the hidden layer, at layer l , $W^{(l)}$ is a convolutional tensor, v an activation functions (in this work the leaky ReLU²⁶) which acts componentwise, and $b^{(l)}$ is a vector of bias values for layer l . For $h^{(0)}(x) = x$ holds. $W^{(l)} * h^{(l-1)}(x)$ is the discrete convolution between W , a tensor of weights, and $h^{(l-1)}(x)$, a tensor of outputs from the $l - 1$ layer.

A classical autoencoder¹³ (AE) is a FFN, a convolutional autoencoder⁴ a CNN, both try to replicate the input data as good as possible, while processing the data through a hourglass similar architecture, where in the middle of the network the data space is reduced drastically. This reduced space is called *feature space*. The compressing part of the network, often called encoding part, tries to reduce the complexity of the data as representatively as possible in the feature space. The second part, often called the decoding part, tries to reconstruct the data from the features such that the difference between input and output is minimal. AEs are self-supervised learning methods such that no annotated data are needed, changing Eq. (1) to

$$\hat{f} = \arg \min_{f \in \mathcal{F}} \mathbb{E}[\mathcal{L}(X, f(X))]. \quad (3)$$

The loss function $\mathcal{L}: \mathbb{R}^{|\theta|} \mapsto \mathbb{R}$,

$$\mathcal{L}(\theta) = \frac{1}{R} \sum_{r=1}^R |x^{(r)} - x_L^{(r)}(\theta)|, \quad (4)$$

quantifies the difference between input $x^{(r)}$ and output $x_L^{(r)}$ of the AE by means of a suitable metric (in this case, the mean absolute error), where L denotes the output layer and $x^{(r)}$ are the different inputs. The loss function $\mathcal{L}(\theta)$ can also be used to train the weights θ of

the encoder and the decoder by a gradient descent method¹⁸ to minimize $\mathcal{L}(\theta)$. In our prediction algorithm, the features generated in the bottleneck of the AE architecture will be the input of the CRF, which predicts their temporal development.

B. Extended conditional random fields

Before introducing the extended conditional random field (eCRF) used for prediction, we shall revisit the general concept of CRFs. For this purpose, we follow the introduction to CRFs by Sutton and McCallum⁴⁴ and adopt their notation, as far as possible. The task of time series prediction for a temporal sequence of frames (fields, images) can be formulated in a probabilistic framework referring to a set of random variables $X \cup Y$, where X is a set of (known) input variables (here: features of the current and previous frames) and Y is a set of (unknown) output variables to be predicted (here: the feature representation of the next future frame). The relation between input and output variables $\mathbf{x} \in X$ and $\mathbf{y} \in Y$, respectively, can be expressed in terms of a conditional probability distribution $p(\mathbf{y}|\mathbf{x})$. Maximizing this conditional probability provides the desired predicted value

$$\mathbf{y}_p = \underset{\mathbf{y}}{\operatorname{argmax}}(p(\mathbf{y}|\mathbf{x})). \quad (5)$$

Despite the dimension reduction provided by the AE, the feature representation of spatiotemporal time series still consists of many variables (in the following examples 32×64 , see Fig. 2). To cope with this kind of high-dimensional (conditional) probability distributions, we use here conditional random fields (CRFs)²² which belong to the class of graphical models (GMs).¹⁹ GMs are a framework for dealing with multivariate probability distributions. They exploit the fact that a distribution over many variables can often be expressed as a product of so-called *factors* (also called *local functions* or *compatibility functions*) that each depend on a (small) subset of variables exploiting statistical independence between (sets of) variables. With CRFs, this decomposition is applied to the conditional probability distribution,

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^A \psi_a(\mathbf{y}_a, \mathbf{x}_a), \quad (6)$$

where $F = \{\psi_a\}$ is the set of factors $\psi_a \geq 0$ with a total number of A elements, such that $X_a \subset \mathcal{X}, Y_a \subset \mathcal{Y}$, with $\mathbf{x}_a \in X_a$ and $\mathbf{y}_a \in Y_a$. $Z(\mathbf{x})$ is a normalization term, defined later. Such factorizations of probability distributions are called GMs because their structure can efficiently be represented by means of an undirected bipartite graph $G(V, F, E)$ where the first set of nodes V consists of the random variables, F stands for the set of nodes representing the factors (or local functions), and E is the set of edges defining which variables enter which factor ψ_a , see Fig. 1.

(X, Y) is a *conditional random field* if for any $\mathbf{x} \in X$ the conditional probability distribution $p(\mathbf{y}|\mathbf{x})$ factorizes according to the graph G . Up to this point, it was assumed that the \mathbf{x}, \mathbf{y} are a pair of variables occurring at the same time or a temporal state. Assuming a stationary input, the factorization of the probability distribution and the structure of the graph G will not change in time. As will be shown in more detail below, the factors ψ_a are composed of a linear superposition of *feature functions* with corresponding *weights* (representing

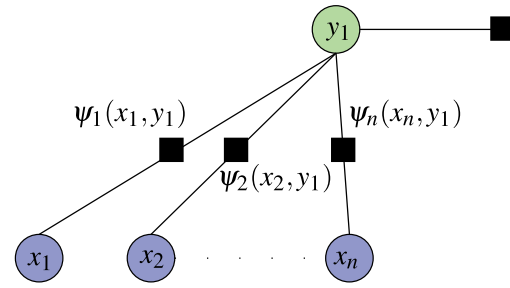


FIG. 1. Example of a graph G describing the distribution $p(y_1|\mathbf{x})$, with local functions $\psi_1(x_1, y_1), \psi_2(x_2, y_1), \dots, \psi_n(x_n, y_1)$. The circles are variable nodes, where the blue ones are the input and the green one the output, and the shaded boxes are factor nodes.

parameters to be learned). Another step for reducing the complexity of the factorization is the common use of feature functions and weights by several factors. This results in a partitioning of the graph G into a set $\mathcal{C} = \{C_1, C_2, \dots, C_q\}$ of cliques C_p such that the CRF can be written as⁴⁴

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p), \quad (7)$$

where \mathbf{x}_c and \mathbf{y}_c are the parts of \mathbf{x}, \mathbf{y} with shared weights in a clique and where each factor,

$$\Psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p) = \exp \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{y}_c), \quad (8)$$

is given by the weights θ_{pk} and the feature functions $\mathcal{F} = \{f_{pk}(\mathbf{x}_c, \mathbf{y}_c)\}_{k=1}^{K(p)}$. The normalization function reads

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{y}_c; \theta_p). \quad (9)$$

To model more than one temporal state, Eq. (7) can be extended by latent variables to a hidden conditional random field (HCRF),³⁶ where $\mathbf{h} = (h_1, \dots, h_m)$ is a vector of latent variables, with $h_j \in \mathcal{H}$, where \mathcal{H} is a finite set of possible hidden states. Practically, this means that the encoded elapsed time steps from the AE are stored in \mathbf{h} ,

$$p(\mathbf{y}, \mathbf{h}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p), \quad (10)$$

with parameterized factors

$$\Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p) = \exp \sum_{k=1}^{K(p)} \theta_{pk} f_{pk}(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c), \quad (11)$$

and the normalization function

$$Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{C_p \in \mathcal{C}} \prod_{\psi_c \in C_p} \psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p). \quad (12)$$

Extended conditional random fields (eCRFs) are defined by extending the parameterized factors to

$$\Psi_c(\mathbf{x}_c, \mathbf{h}_c, \mathbf{y}_c; \theta_p) = \exp \left[\sum_{j=1}^m \theta_p^{(h_j)} f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, \tau) + \theta_p^{(y, h_j)} + \sum_{\forall (j,k)s.t.: j \neq k} \frac{\theta_p^{(y)} f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, k, \tau)}{k} + \theta_p^{(y, h_j, h_k)} \right], \quad (13)$$

i.e., an eCRF is given by Eqs. (10), (12), and (13). The extended parameterized factors in Eq. (13) describe (measure) the conditional probability between a possible forecast \mathbf{y} , a set of observations \mathbf{x} and a configuration of past states \mathbf{h} , where m is the total number of past states. $f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, \tau)$ is a feature function with cliques that can include any subsequence of τ past states to describe the probability between some subsequence of past states and the observed current state. While $f_p(\mathbf{x}_c, \mathbf{y}_c, \mathbf{h}_c, j, k, \tau)$ describes the probability of the current state and a possible future state, it is divided by k to avoid that this term is added multiple times. The parameter vector $\theta_p = \{\theta_p^{(h_j)}, \theta_p^{(y, h_j)}, \theta_p^{(y, h_j, h_k)}, \theta_p^{(y)}\}$ consists of the parameter vectors $\theta_p^{(h_j)}$ for the parameters corresponding to the state $h_j \in \mathcal{H}$, the parameters $\theta_p^{(y, h_j)}$ describing how well a forecasted feature \mathbf{y} corresponds to a state h_j , the weights $\theta_p^{(y, h_j, h_k)}$ between two edges of the past states h_j and h_k , and some parameters $\theta_p^{(y)}$ for any element y of \mathbf{y} depending on the features over the past. Parameter $\tau = 10$ specifies the number of inputs from the AE taken into account for the forecast.

1. eCRF features

To get feature functions f_p for Eq. (13), a set of base features functions is used for the introduced eCRF model. These base features are used during the training to induce the desired features. The method of *feature induction* is described in Ref. 27. Four base functions are used, where \mathbf{y} is considered to be part of \mathbf{x} , for the *feature induction*:

- Weighted neighbor feature

$$f_{\text{WNF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = \begin{cases} \theta_{\text{WNF}}^{(1)} \sum_{i=j}^{\tau+j} \|x_j - x_i\|_2 & \text{if } \tau > 1, \\ \theta_{\text{WNF}}^{(2)} & \text{if } \tau = 0. \end{cases} \quad (14)$$

- Mean feature

$$f_{\text{DF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = \frac{x_j + \sum_{i=j}^{\tau+j} h_i}{\tau + 1}. \quad (15)$$

- Distance feature

$$f_{\text{MF}}(\mathbf{x}, \mathbf{y}, \mathbf{h}, j, \tau) = x_j - x_{j+\tau}. \quad (16)$$

- For $f_{\text{PM}}(\mathbf{x}, \mathbf{h}, j, k, \tau)$, a probabilistic movement feature based on Ref. 29 is calculated,

$$\text{pm}_t = \begin{bmatrix} x_j(t) \\ \dot{x}_j(t) \end{bmatrix} = \phi_t^T w + \varepsilon_{\text{pm}_t}, \quad (17)$$

$$p(w|\tau) = \prod_{t=0}^{\tau} \mathcal{N}(\text{pm}_t | \phi_t^T w, \Sigma_{\text{pm}_t}), \quad (18)$$

where a weight vector w is used to compactly represent a single trajectory (changes of x_j over the time window τ) given as a linear basis function model ϕ_t and $\varepsilon_{\text{pm}_t} \sim \mathcal{N}(0, \Sigma_{\text{pm}_t})$ is a zero-mean i.i.d Gaussian noise. This model is used to predict the value of x_j at time point $t + 1$,

$$f_{\text{PM}}(\mathbf{x}, \mathbf{h}, j, k, \tau) = x_j + \dot{x}_j, \quad (19)$$

where $x_j + \dot{x}_j$ is the predicted value of x_j at time $t + 1$ based on the subsequence $\{h_j, \dots, h_k\} \subseteq \{h_1, \dots, h_m\}$.

C. Training

The parameter determination of the presented approach consists of two phases: First the AE, consisting of the encoding and decoding part, is trained such that $\sum_r |f(x^{(r)}; \theta_{\text{AE}}) - x^{(r)}| \stackrel{!}{=} 0$. Then, the parameters θ_{AE} of the AE are fixed and the eCRF is deployed. The combined AE with the eCRF is then used to train the parameters θ_{eCRF} of the eCRF. For this, the likelihood function,

$$\mathcal{L}(\theta_{\text{eCRF}}) = \sum_{i=1}^R P(\mathbf{y}_i | \mathbf{x}_i, \theta_{\text{eCRF}}) - \frac{1}{2\sigma^2} \|\theta_{\text{eCRF}}\|^2, \quad (20)$$

is used, where R is the number of training examples and σ^2 the L_2 regularizer (in this work, $\sigma^2 = 11$). By maximizing the likelihood for the true prediction on the training data, the optimal parameter set θ_{eCRF}^* is determined. To find θ_{eCRF}^* , Eq. (20) can be maximized by the gradient descent method which is used for optimizing/training the θ_{AE} . The AE part of the method was implemented with Keras⁵ and Tensorflow¹ version 1.10, the eCRF was implemented with C++ and Python³⁸ version 3.6 from scratch.

D. Prediction

To predict the input sequence with the eCRF for one time step, it is necessary to find

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y}} p(\mathbf{y}, \mathbf{h} | \mathbf{x}; \theta_{\text{eCRF}}^*), \quad (21)$$

where $\hat{\mathbf{y}}$ is calculated elementwise for every element in \mathbf{y} and for each element \mathbf{x} in the feature space. The field of $\hat{\mathbf{y}}$ is then used by the AE decoding part to map the features back to generate the desired output at $t + \Delta t$. The future evolution of the spatiotemporal time series is predicted by using a feedback loop from the output of the AE back to the input.

III. RESULTS

In order to benchmark our method, spatiotemporal time series from two systems were examined. The first system is the Lorenz-96 model,²⁴ a system with an adjustable number of ordinary differential equations exhibiting high-dimensional chaotic behavior. This system is often used as a benchmark in data assimilation.^{35,37} The second set of time series was generated by the Kuramoto-Sivashinsky equation,^{21,40,41} a nonlinear PDE which was also used as benchmark

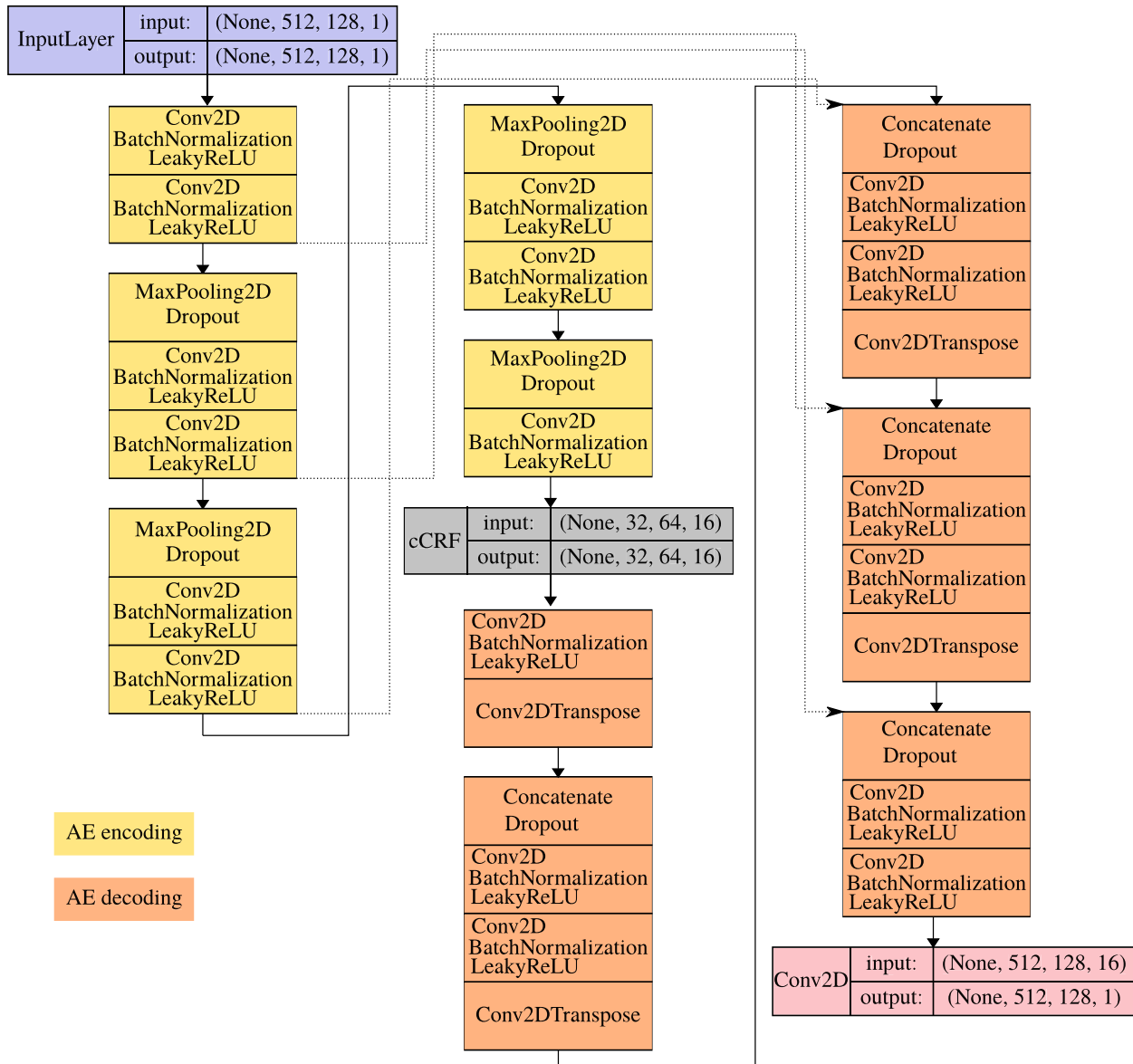


FIG. 2. Proposed architecture for prediction, consisting of a split autoencoder and the eCRF. Each rectangular block is a set of layers. The first dimension in the input and output boxes is the batch size which is **None** meaning that this dimension is variable. The second variable 512 is the size of state vector, and the 128 the number of time steps taken into account for performing one input for the eCRF, the last size is the number of channels or filters used in the layer.

system in Ref. 31. In both cases, we investigate how long the mean absolute error between the ground truth data and the predicted data is smaller than some given ϵ_{error} . This prediction horizon is then compared to the Lyapunov time, a characteristic time scale given by the inverse of the largest Lyapunov exponent Λ_{max} . As input, we always use $\tau = 10$ fields where the first dimension of each field is the system size and the second dimension is the number of time steps (here: 128).

A. Lorenz-96 model

The Lorenz-96 model is a dynamical system

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{22}$$

introduced by Lorenz²⁴, where x_i ($i = 1, \dots, N$) is a component of the system state $\mathbf{x} = [x_1, \dots, x_N]$ and $F \in \mathbb{R}$ is a forcing constant.

The components of \mathbf{x} are arranged in a ring with periodic boundary conditions $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_{N+1} = x_1$.

In the context of this manuscript, $F = 8$ and $N \in \{32, 64, 128, 256\}$. For each N , 100 cases were generated where the initial state was set to F and 10% of the components were perturbed with some random values from a uniform distribution between $[-0.01, 0.01]$. To integrate the system, we used $\Delta t = 0.01$ and $t_{\max} = 45.6$ rendering in total 4560 time steps. For integration, the python implementation of lsoda³⁴ was used. However, only the last 2560 time steps were considered and the first 2000 time steps of the transient to the attractor were discarded to avoid initialization artifacts. The data were then divided into sequences of 128 time steps, such that $X \in \mathbb{R}^{N \times 128}$, and randomly divided in 50% training, 25% validation and 25% test data. The following results are all from the test data. In Figs. 3 and 4, cases $N = 32$ and $N = 128$ are shown as examples. By setting the threshold $\varepsilon_{\text{error}} = 0.35$ for the mean absolute error, we define the prediction horizon, i.e., the point in time until successful forecasts are achieved. It should be noted that the choice of this value was subjective, it was the point where a deviation from the ground truth can be seen by bare eyes. With this threshold, the Lorenz-96 model is successfully predicted for $N = 32$ over a period of time $9.55t$ corresponding to 955 time steps. By multiplying t with the largest Lyapunov exponent Λ_{\max} (see Table I), this corresponds to over 15 Lyapunov times. A similar behavior can be seen for the case $N = 128$ (Fig. 4). Although the prediction time-interval decreases, it is found that in terms of Lyapunov time a similar prediction duration exists, since Λ_{\max} has increased. However, Figs. 3 and 4 only provide a general intuition. The duration of how long the forecast remains correct depends also on the initial state on the attractor. Therefore, in Fig. 5a, the prediction statistics for all 500 test sequences for $N \in \{32, 64, 128, 256\}$ is shown with different initial conditions in the test data. The median for the prediction horizon (in units of the Lyapunov time) is 15.48 for $N = 32$, 15.03 for $N = 64$, 15.00 for $N = 128$ and 14.86 for $N = 256$.

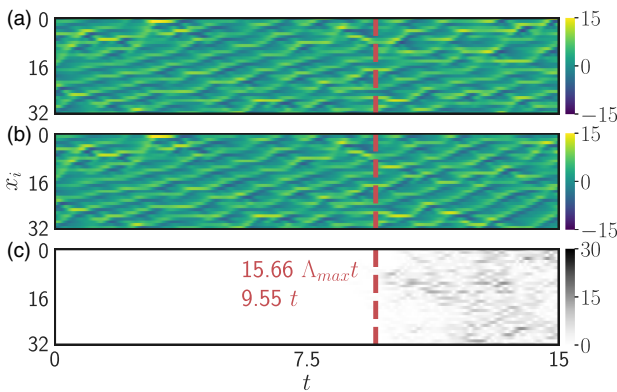


FIG. 3. Comparison between real (ground truth) data (a) with the predicted data (b) for the Lorenz-96 model with $N = 32$. (c) shows the absolute error between real data and the forecast. The red line indicates the time when the mean absolute error is larger than $\varepsilon_{\text{error}} = 0.35$ corresponding to an error of about 1.2%. Using this conservative criterion for this example, a prediction horizon of $15.66 \Lambda_{\max} t$ is achieved.

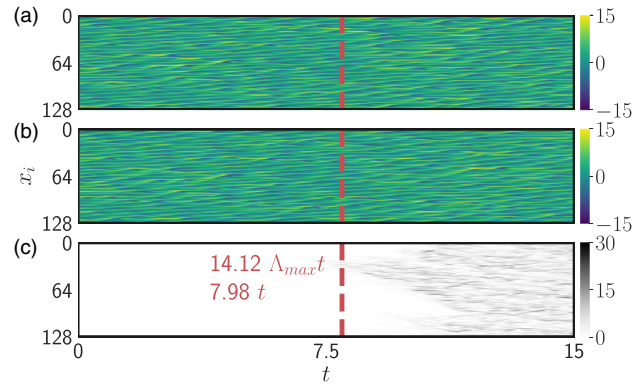


FIG. 4. Same as Fig. 3 for $N = 128$, where we achieve a prediction horizon of $14.12 \Lambda_{\max} t$ under the same constraints as in Fig. 3.

B. Kuramoto-Sivashinsky equation

The Kuramoto-Sivashinsky equation (KSE),

$$u_t + \nu \nabla^4 u + \nabla^2 u + \frac{1}{2} |\nabla u|^2 = 0,$$

$$(x, t) \in \mathbb{R} \times \mathbb{R}_+,$$

$$u(x, 0) = u_0(x), \quad u(x + L, t) = u(x, t), \quad (23)$$

is a fourth-order nonlinear partial differential equation introduced for modeling diffusive instabilities in a laminar flame front,⁴¹ where ∇^2 is the Laplacian, ∇^4 is the biharmonic operator, ∇ is the gradient, ν is a positive constant and u_0 is L -periodic, where L is the size of a pattern cell.⁴¹ In the specific limit of large nondimensional surface tension, it was proved⁴² that for $y(x, t) = \nabla u(x, t)$ and $\nu = 1$ the KSE can be written as

$$y_t + \nabla^4 y + \nabla^2 y + y \nabla y = 0. \quad (24)$$

In order to achieve a comparison with recent results by Pathak *et al.*³¹ representing the current the state of the art of data-driven time series prediction, we also extended Eq. (24) with a spatial inhomogeneity term, as in Ref. 31,

$$y_t + \nabla^4 y + \nabla^2 y + y \nabla y - \mu \cos\left(\frac{2\pi x}{\lambda}\right) = 0. \quad (25)$$

Like in Ref. 31, Eq. (24) is integrated on a grid of Q equally spaced points with $\Delta t = 0.25$ and $t_{\max} = 2500$ rendering 10 000 time

TABLE I. Largest Lyapunov exponent Λ_{\max} and Kaplan-Yorke attractor dimension (D_{KY}) ¹⁷ for Lorenz-96 systems with different dimensions N .

N	Λ_{\max}	D_{KY}
32	1.64	21.7
64	1.72	43.0
128	1.77	86.6
256	1.79	173.3

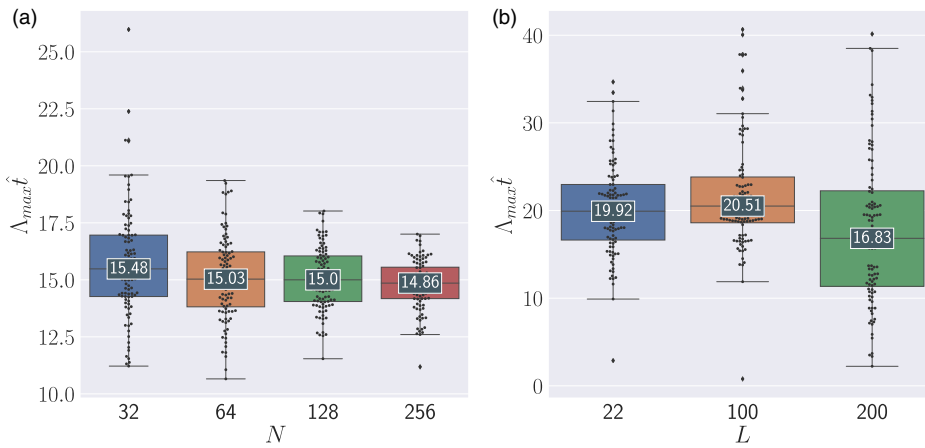


FIG. 5. Lyapunov time statistics for (a) Lorenz-96 systems of dimensions $N \in \{32, 64, 128, 256\}$ and b) KSEs with $L = 22, Q = 64, \mu = 0$, $L = 100, Q = 256, \mu = 0.01, \lambda = 100$, and $L = 200, Q = 512, \mu = 0.01, \lambda = 100$. \hat{t} is the time when the mean absolute error for the prediction exceeds $\varepsilon_{error} = 0.35$ for the Lorenz-96 system and $\varepsilon_{error} = 0.07$ for the KSE.

steps. For integration, the exponential time differencing fourth-order Runge-Kutta algorithm (ETDRK4)⁶ was used. To compare our results with Pathak *et al.*,³¹ we considered the cases $L \in \{22, 100, 200\}$.

Similar to the analysis of the Lorenz-96 model, for each L , 100 different cases were generated with random initialization and the last 7936 time steps, such that subsequences $X \in \mathbb{R}^{L \times 128}$ representing 128 time steps for later training, validation, and test purposes. Figure 6 shows the prediction results for the KSE with $L = 22, Q = 64$, and $\mu = 0$. In this specific case a prediction horizon of 22.97 Lyapunov times was achieved with an error threshold of $\varepsilon_{error} = 0.07$. For the case of $L = 200, Q = 512, \mu = 0.01$, and $\lambda = 100$ (Fig. 7), a comparable result of 22.18 $\Lambda_{max} t$ was found.

As with the Lorenz-96 model, the prediction duration of the KSE is also dependent on the initial state on the attractor. The corresponding statistics are shown in Fig. 5(b). According to our

knowledge, the current best prediction results for the KSE were reported in Ref. 31 for an echo state network which achieved ≈ 3 Lyapunov times (read from Fig. 2 in Ref. 31) for $L = 22$ and ≈ 8 Lyapunov times for $L = 200$ (reported in text of Ref. 31). Unfortunately, there it was not reported at which error threshold for the prediction horizon these values were taken. We exceed the prediction horizon obtained with the echo state network, even when only considering median values, substantially with 19.92 Lyapunov times for $L = 22$ and 16.83 for $L = 220$. Of course, training of our hybrid approach is much more time consuming than the learning procedure with reservoir computing.

IV. DISCUSSION

The presented algorithm delivers promising results, but one can ask, why this is so? Alas, this hybrid method does not easily lend itself to intuition. While the system necessarily uses the temporal history of

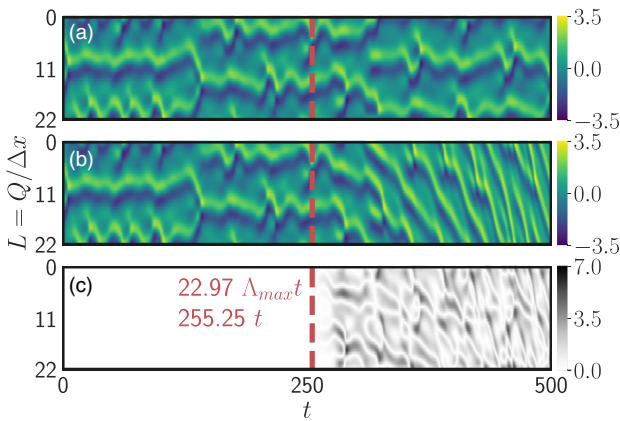


FIG. 6. Comparison between real data (a) with predicted data (b) of the KSE with $L = 22, Q = 64$, and $\mu = 0$. (c) shows the absolute error between ground truth and prediction. The red line indicates the point at which the mean absolute error is greater than $\varepsilon_{error} = 0.07$ corresponding to an error of about 1.2%, using this criterion a prediction horizon of 22.97 $\Lambda_{max} t$ is achieved.

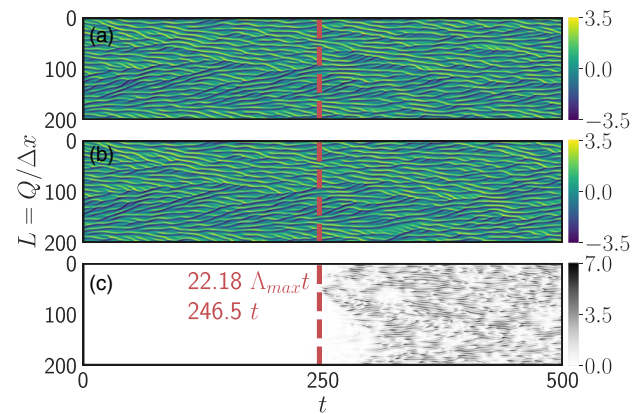


FIG. 7. Same as in Fig. 6 with $L = 200, Q = 512, \mu = 0.01$, and $\lambda = 100$, where we get a prediction horizon of 22.18 $\Lambda_{max} t$ under the same constraints as in Fig. 6.

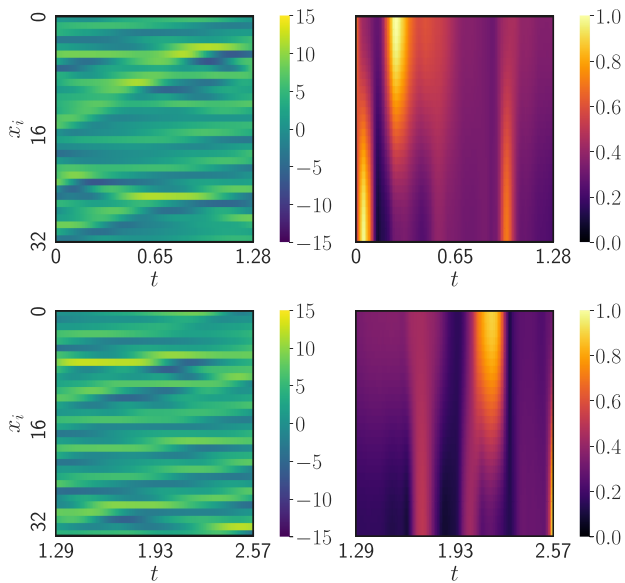


FIG. 8. Attention map for the Lorenz-96 system with $N = 32$. The left side is the input image and the right side is the corresponding attention map. The panels at the bottom show the same for a different moment in time along the time series.

the time series, this does not happen in any straightforward manner. For example, the simple assumption that there is a continuous decay of the importance of data that is longer in the past is not true.

The autoencoder (AE) compresses the data at its bottleneck layers, leading to a much reduced and importance-weighted representation of the input. To make this visible, we need to backproject the importance-weighting into the input space. For this, one can use so-called gradient-based attention maps, which are maps where the value of the positive gradients from the AE, after training, is weighted by the loss of Eq. (4) and projected back to the input space. Hence, they allow us to inspect which parts of the input are most strongly represented in the bottleneck of the AE. The method to generate such attention maps was presented in Ref. 39. Figure 8 shows the input situation (left) for a certain moment in time of the Lorenz-96 system for a period of time and the corresponding attention map (right) for $N = 32$. Hence, only those parts of the input, where the attention map has high values, are considered by the encoder-pathway of the AE. A similar diagram for $N = 128$ is shown in Fig. 9. There is no simple temporal structure visible in either of these attention maps and the resulting structures are complex. Furthermore, it is important to note that these attention maps will look different for different time points along the time series. On more theoretical grounds, a certain commonality, however, exists for the importance-weighting performed by the AE across all cases. Autoencoders have been investigated in depth in the recent past.^{8,12,45} These works show that AEs are capable of learning disentangled representations of the data to construct a latent representation that reconstructs existing (or generates new) samples from complex, high-dimensional distributions.

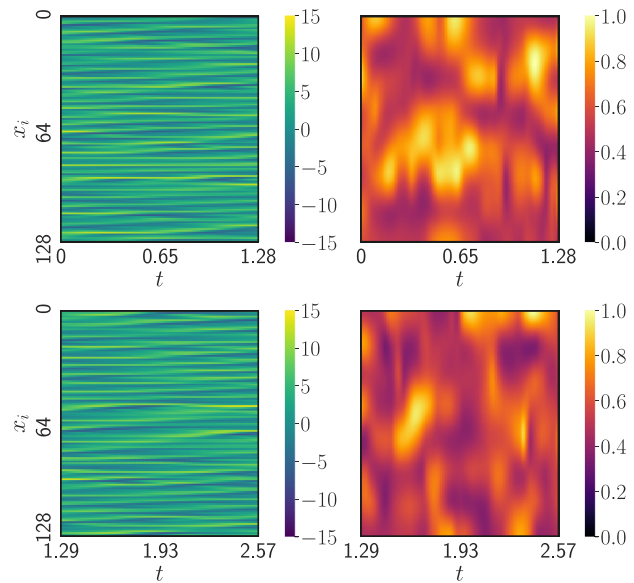


FIG. 9. As Fig. 8 but here with system size $N = 128$.

We suggest that exactly this behavior is a key component for the performance of our approach.

The disentanglement property creates features in the bottleneck layer of the AE that can effectively be understood as independent random variables. Under such conditions, conditional random fields (CRF) are known to operate with high performance due to optimal factorization and representation. We suggest that these aspects may underlie the high prediction performance observed with our hybrid prediction method.

The here-presented method delivers quite a large prediction horizon on complex time series. As a consequence, different applications for such a system are conceivable. It can be used to perform data-driven modeling for systems where no mathematical models, based on first principles, exist, or which are very expensive to solve. A second, potentially interesting application would be the acceleration of complex model calculations, where a hybrid approach is used that iterates conventional numerical simulations with predictions by our method. This way, costly and potentially slow numerical simulations do not have to be performed for every timestep. Instead, missing data would be filled in by the predictions made by our approach.

The work we have done so far is intended to be a proof of concept; there are still many components that can probably be improved to some degree. In this work, we have not tried to tune hyperparameters, including the architecture of the AE or of the eCRF. The way in which we have trained the parameters meets the state of the art in machine learning, but also, here, possibilities exist for optimization. One option would be to use a bias-variance balancing objective function.³² Another possibility, which could even be considered in addition, would be population-based training.¹⁶

ACKNOWLEDGMENTS

We thank Stefan Luther for continuous support and inspiring discussions on applications of nonlinear time series analysis in cardiac dynamics and beyond, and Jonas Iensee for support with the implementation of the Kuramoto-Sivashinsky equation and interesting exchanges about state reconstructions of nonlinear systems. S.H. acknowledges funding by the International Max Planck Research Schools of Physics of Biological and Complex Systems. This work was supported by the European Commission's H2020-Framework FET-Proactive Grant Plan4Act (No. 732266).

REFERENCES

- ¹M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems" (2015), see tensorflow.org.
- ²A. J. Abebe, D. P. Solomatine, and R. G. W. Venneker, "Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events," *Hydrolog. Sci. J.* **45**, 425–436 (2000).
- ³G. E. P. Box and G. Jenkins, *Time Series Analysis, Forecasting and Control* (Holden-Day Inc., San Francisco, CA, 1990).
- ⁴Z. Cheng, H. Sun, M. Takeuchi, and J. Katto, "Deep convolutional autoencoder-based lossy image compression," in *2018 Picture Coding Symposium, PCS 2018—Proceedings* (Institute of Electrical and Electronics Engineers Inc., 2018), pp. 253–257.
- ⁵F. Chollet *et al.*, see <https://keras.io> for "Keras" (2015).
- ⁶S. Cox and P. Matthews, "Exponential time differencing for stiff systems," *J. Comput. Phys.* **176**, 430–455 (2002).
- ⁷C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *Computer Vision—ECCV 2014*, edited by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars (Springer International Publishing, Cham, 2014), pp. 184–199.
- ⁸S. Gao, R. Brekelmans, G. V. Steeg, and A. Galstyan, "Auto-encoding total correlation explanation," e-print [arXiv:1802.05822\[cs.LG\]](https://arxiv.org/abs/1802.05822) (2018).
- ⁹R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung, "Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit," *Nature* **405**, 947 (2000).
- ¹⁰K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 770–778; see <https://ieeexplore.ieee.org/document/7780459>.
- ¹¹S. Herzog, F. Wörgötter, and U. Parlitz, "Data-driven modeling and prediction of complex spatio-temporal dynamics in excitable media," *Front. Appl. Math. Stat.* **4**, 60 (2018).
- ¹²I. Higgins, L. Matthey, X. Glorot, A. Pal, B. Uria, C. Blundell, S. Mohamed, and A. Lerchner, "Early visual concept learning with unsupervised deep learning," e-print [arXiv:1606.05579](https://arxiv.org/abs/1606.05579) (2016).
- ¹³G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science* **313**, 504–507 (2006).
- ¹⁴S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," e-print [arXiv:1502.03167\[cs.LG\]](https://arxiv.org/abs/1502.03167) (2015).
- ¹⁵J. Iensee, G. Datseris, and U. Parlitz, "Predicting spatio-temporal time series using dimension reduced local states," *J. Nonlinear Sci.* (published online).
- ¹⁶M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, K. Simonyan, C. Fernando, and K. Kavukcuoglu, "Population based training of neural networks," e-print [arXiv:1711.09846\[cs.LG\]](https://arxiv.org/abs/1711.09846) (2017).
- ¹⁷J. L. Kaplan and J. A. Yorke, "Chaotic behavior of multidimensional difference equations," in *Functional Differential Equations and Approximation of Fixed Points*, edited by H.-O. Peitgen and H.-O. Walther (Springer, Berlin, 1979), pp. 204–227.
- ¹⁸D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," e-print [arXiv:1412.6980](https://arxiv.org/abs/1412.6980) (2014).
- ¹⁹D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning Series)* (The MIT Press, 2009).
- ²⁰A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems—Volume 1*, series and number NIPS'12 (Curran Associates Inc., 2012), pp. 1097–1105.
- ²¹Y. Kuramoto, "Diffusion-induced chaos in reaction systems," *Prog. Theor. Phys. Suppl.* **64**, 346–367 (1978).
- ²²J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proceedings of the 18th International Conference on Machine Learning* (Morgan Kaufmann Publishers Inc., 2001), pp. 282–289.
- ²³Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in Neural Information Processing Systems 2*, edited by D. S. Touretzky (Morgan-Kaufmann, 1990), pp. 396–404.
- ²⁴E. Lorenz, "Predictability: a problem partly solved," in *Seminar on Predictability, 4–8 September 1995*, ECMWF (ECMWF, Shinfield Park, Reading, 1995), Vol. 1, pp. 1–18.
- ²⁵E. N. Lorenz, "Deterministic nonperiodic flow," *J. Atmos. Sci.* **20**, 130–141 (1963).
- ²⁶A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proceedings of the 30th International Conference on Machine Learning* (Atlanta, Georgia, 2013); see also: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf.
- ²⁷A. McCallum, "Efficiently inducing features of conditional random fields," in *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, series and number UAI'03 (Morgan Kaufmann Publishers Inc., San Francisco, CA, 2003), pp. 403–410.
- ²⁸J. Nassar, S. Linderman, M. Bugallo, and I. M. Park, "Tree-structured recurrent switching linear dynamical systems for multi-scale modeling," in *International Conference on Learning Representations* (International Conference on Learning Representations (ICLR), 2019).
- ²⁹A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," in *Advances in Neural Information Processing Systems 26*, edited by C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Curran Associates, Inc., 2013), pp. 2616–2624.
- ³⁰U. Parlitz and C. Merkwirth, "Prediction of spatiotemporal time series based on reconstructed local states," *Phys. Rev. Lett.* **84**, 1890–1893 (2000).
- ³¹J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," *Phys. Rev. Lett.* **120**, 024102 (2018).
- ³²M. Pavlovski, F. Zhou, N. Arsov, L. Kocarev, and Z. Obradovic, "Generalization-aware structured regression towards balancing bias and variance," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18* (International Joint Conferences on Artificial Intelligence Organization, 2018), pp. 2616–2622.
- ³³B. Penkovsky, X. Porte, M. Jacquot, L. Larger, and D. Brunner, "Coupled nonlinear delay systems as deep convolutional neural networks," *Phys. Rev. Lett.* **123**, 054101 (2019).
- ³⁴L. Petzold, "Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations," *SIAM J. Sci. Stat. Comput.* **4**, 136–148 (1983).
- ³⁵F. R. Pinheiro, P. J. van Leeuwen, and U. Parlitz, "An ensemble framework for time delay synchronization," *Q. J. R. Meteorol. Soc.* **144**, 305–316 (2018).
- ³⁶A. Quattoni, S. Wang, L. Morency, M. Collins, and T. Darrell, "Hidden conditional random fields," *IEEE Trans. Pattern Anal. Mach. Intell.* **29**, 1848–1852 (2007).
- ³⁷D. Rey, M. Eldridge, M. Kostuk, H. D. Abarbanel, J. Schumann-Bischoff, and U. Parlitz, "Accurate state and parameter estimation in nonlinear systems with sparse observations," *Phys. Lett. A* **378**, 869–873 (2014).
- ³⁸G. van Rossum and F. L. Drake, *The Python Language Reference Manual* (Network Theory Ltd., 2011).

³⁹R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-cam: Visual explanations from deep networks via gradient-based localization,” e-print [arXiv:1610.02391\[cs.CV\]](https://arxiv.org/abs/1610.02391) (2016).

⁴⁰G. Sivashinsky, “Nonlinear analysis of hydrodynamic instability in laminar flames—i. derivation of basic equations,” *Acta. Astronaut.* **4**, 1177–1206 (1977).

⁴¹G. I. Sivashinsky, “On flame propagation under conditions of stoichiometry,” *SIAM J. Appl. Math.* **39**, 67–82 (1980).

⁴²G. I. Sivashinsky and D. M. Michelson, “On irregular wavy flow of a liquid film down a vertical plane,” *Progr. Theor. Phys.* **63**, 2112–2114 (1980).

⁴³N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.* **15**, 1929–1958 (2014); available at <http://dl.acm.org/citation.cfm?id=2627435.2670313>.

⁴⁴C. Sutton and A. McCallum, “An introduction to conditional random fields,” *Found. Trends Mach. Learn.* **4**, 267–373 (2012).

⁴⁵M. Tschannen, O. Bachem, and M. Lucic, “Recent advances in autoencoder-based representation learning,” e-print [arXiv:1812.05069](https://arxiv.org/abs/1812.05069) (2018).

⁴⁶V. Vemuri, *Artificial Neural Networks: Theoretical Concepts*, Computer Society Press Technology Series: Neural networks (Computer Society Press of the IEEE, 1988).

⁴⁷P. Vlachas, W. Byeon, Z. Yi Wan, T. P. Sapsis, and P. Koumoutsakos, “Data-driven forecasting of high-dimensional chaotic systems with long-short term memory networks,” *Proc. R. Soc. A Math. Phys. Eng. Sci.* **474**, 1 (2018).

⁴⁸W. Zhang, B. Wang, Z. Ye, and J. Quan, “Efficient method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order models,” *AIAA J.* **50**, 1019–1028 (2012).

3.3.1.1 Conclusions from (Herzog et al. 2019)

As the previous publication shows, the combination of the CAE and CRF turned out to be very powerful, allowing prediction horizons clearly larger than those in the former state of the art presented by Pathak et al. (2018). The systems used here were fully observable, i.e. all variables were seen at all times. But what happens if a system is only partially observable? This is precisely the question addressed in the next publication. Herzog, S., Wörgötter, F., and Parlitz, U. (2018). “Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media”. In: *Frontiers in Applied Mathematics and Statistics* 4.

3.3.2 Publication: (Herzog et al. 2018)

The previous publication demonstrated the performance of the hybrid approach on data from non-linear systems often used as benchmarks for the prediction horizon. In the next publication, the hybrid model is applied on data from a system consisting of several variables that provides a compact description of excitable cardiac dynamics. Besides the investigation of the prediction horizon, it was also investigated whether a state reconstruction of the model is possible if only a subset of the state variables can be used. This question extends the publication from the previous chapter and demonstrates that the hybrid model preserves the reconstruction properties of the CAE. Furthermore, it is shown that even a prediction is possible if not all variables are used as an input for the prediction.



Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media

Sebastian Herzog^{1,2}, Florentin Wörgötter² and Ulrich Parlitz^{1,3,4*}

¹ Max Planck Institute for Dynamics and Self-Organization, Göttingen, Germany, ² Third Institute of Physics and Bernstein Center for Computational Neuroscience, University of Göttingen, Göttingen, Germany, ³ Institute for Nonlinear Dynamics, University of Göttingen, Göttingen, Germany, ⁴ DZHK (German Centre for Cardiovascular Research), Partner Site Göttingen, Göttingen, Germany

OPEN ACCESS

Edited by:

Axel Hutt,
German Weather Service, Germany

Reviewed by:

Christian Andreas Welzbacher,
German Weather Service, Germany
Xin Tong,
National University of Singapore,
Singapore

*Correspondence:

Ulrich Parlitz
ulrich.parlitz@ds.mpg.de

Specialty section:

This article was submitted to
Dynamical Systems,
a section of the journal
Frontiers in Applied Mathematics and
Statistics

Received: 04 September 2018

Accepted: 26 November 2018

Published: 11 December 2018

Citation:

Herzog S, Wörgötter F and Parlitz U
(2018) Data-Driven Modeling and
Prediction of Complex
Spatio-Temporal Dynamics in
Excitable Media.
Front. Appl. Math. Stat. 4:60.
doi: 10.3389/fams.2018.00060

Spatio-temporal chaotic dynamics in a two-dimensional excitable medium is (cross-) estimated using a machine learning method based on a convolutional neural network combined with a conditional random field. The performance of this approach is demonstrated using the four variables of the Bueno-Orovio-Fenton-Cherry model describing electrical excitation waves in cardiac tissue. Using temporal sequences of two-dimensional fields representing the values of one or more of the model variables as input the network successfully cross-estimates all variables and provides excellent forecasts when applied iteratively.

Keywords: deep learning, conditional random fields, artificial neural network, cross-estimation, spatio-temporal chaos, excitable media, cardiac arrhythmias, non-linear observer

1. INTRODUCTION

In life sciences mathematical models based on first principles are scarce and often a variety of approximate models of different complexity exists for describing the given (experimental) dynamical process. For example, electrical excitation waves in cardiac tissue can be described using partial differential equations (PDEs) with 2 to more than 60 variables, covering the range from simple qualitative models [1, 2] to detailed ionic cell models including not only cell membrane voltage but also different ionic currents and gating variables [3, 4]. While there are several modalities for measuring membrane voltage (electrical sensors, fluorescent dyes [5]) it is in general much more difficult and expensive (if not impossible) to directly measure the other variables of the mathematical model, such as ionic currents or gating variables. In such cases it is desirable to (cross) estimate variables, which are difficult to assess from those that can be easily measured. In control theory this task is addressed by constructing an observer based on a given mathematical model describing the process of interest. Once all state variables of the model have been estimated, the model (e.g., a PDE) can be used to simulate and forecast the future evolution of the dynamical process. This combination of cross estimation and prediction of dynamical variables is the core of all data assimilation methods [6–10] where again the model equations are involved and have to be known. In this contribution, we present a machine learning method for estimating all state variables and forecasting their evolution from limited observations. This “black-box model” consists of a convolutional neural network (CNN) combined with a conditional random field (CRF) and will be introduced in section 2. For training and evaluating the network two dimensional spatio-temporal time series are used, which were generated by the Bueno-Orovio-Fenton-Cherry (BOCF) model [11] describing complex electrical excitation waves in cardiac tissue. This model is introduced in

section 3. As modeling tasks we consider cross estimation of variables, forecasting dynamics using an iterative feedback scheme, and a combination of forecasting and cross estimation providing future values of not measured variables. These results are presented in section 4. A summary and a brief discussion of potential future developments are given in section 5.

2. DATA DRIVEN MODELING

In data driven modeling mathematical models are not based on first principles (e.g., Newton’s laws, Maxwell’s equations, ...) but are directly derived from experimental measurement data or other physical observations. The model should describe the experiment as precisely as possible but it also should possess a high level of generalizability, i.e., the ability to provide a suitable and good description for data from a very similar experiment. Therefore, overfitting has to be avoided and all irrelevant aspects that are not necessary to describe the desired effect should be discarded when generating the model (without employing human expert knowledge). Many approaches for generating (dynamical) models from (training) data have been devised including autoregressive models [12], evolutionary algorithms in particular genetic algorithms [13], local modeling [14], reservoir computing [15–19], symbolic regression [20], or adaptive fuzzy rule-based models [21]. Furthermore, Monte Carlo techniques may be used for assessing uncertainty in model parameters [22]. In this work we present a modeling ansatz which combines deep convolutional neural networks [23] for feature extraction and dimension reduction with conditional random fields (CRFs) [24] for modeling the properties of temporal sequences.

2.1. Artificial Neural Network

Artificial neural networks (ANNs) [25–27] are parameterizable models for approximating a (unknown) function F implicitly given by the data. The actual function provided by the ANN:

$$f: \mathbb{R}^O \mapsto \mathbb{R}^P, \tag{1}$$

should be a good approximation of F , i.e., $f \simeq F$. Here $O \in \mathbb{N}$ and $P \in \mathbb{N}$ denote the dimension of the input and the output of f , respectively. A widely used type of ANN are feed-forward neural networks (FNN) where, in general, f is given by

$$f(X) = \psi(WX + b), \tag{2}$$

with a non-linear function ψ applied component-wise, an input vector $X \in \mathbb{R}^O$, a weight matrix $W \in \mathbb{R}^{P \times O}$, and a bias $b \in \mathbb{R}^P$. Equation (2) is called a one-layer FNN. By recursively applying the output of one layer as input to the next layer, a multi-layer FNN can be constructed:

$$f(X) = f^L(\dots f^2(f^1(X; W^1, b^1); W^2, b^2) \dots; W^L, b^L). \tag{3}$$

Equation (3) describes a multi-layer FNN with $L \in \mathbb{N}$ layers. In the following an input with several variables is considered and the input is given by $X \in \mathbb{R}^{h \times w \times d}$, with $h \in \mathbb{N}$ rows and $w \in \mathbb{N}$ columns of the input field, and the number of variables

d . To improve the approximation properties of the network Equation (3), FNNs may contain additional convolutional layers leading to state-of-the-art models for data classification, so-called convolutional neural networks (CNNs) [23].

2.2. Network Architecture

The network used in the following for prediction of multivariate time series is built based on the architecture of a convolutional autoencoder [28], with residual connections [29] consisting of an encoding path (left half of the network, from 512×512 to 64×64) to retrieve the features of interest and a symmetric decoding path (right half of the network, from 64×64 back to 512×512). As illustrated in **Figure 1** each encoding/decoding path consists of multiple levels, i.e., resolutions, for feature extraction on different scales and noise reduction. The conditional random field block has a special role: Based on the selected feature, the CRF should map a sequence of features of a previous time step t to the next time step $t + \Delta t$. The other four components of the network are basic building blocks, like regular convolutional layers followed by rectified linear unit activation and batch normalization (these blocks are omitted in **Figure 1** for simplicity). Each residual layer consists of three convolutional blocks and a residual skip connection. A maxpooling layer is located between levels in the encoding path to perform downsampling for feature compression. The deconvolutional layer [30] is located between levels in the decoding path to up-sample the input data using learnable interpolations. The input for the network are all four system variables of the BOCF model which will be introduced in section 3.1 or a sequence of the four system variables as introduced in section 4.1. The output of the network always consists of four system variables.

2.3. Convolution Layer

Convolutional neural networks [23, 26, 27] receive a training data set $\mathcal{X} = \{X_1, X_2, \dots, X_m\}$, where $X_\alpha \in \mathbb{R}^{h \times w \times d}$. The data processing through the network is described layer-wise i.e., in the l -th convolutional layer the input $X^{(l)}$ will be transformed to the *raw output* $o^{(l)}$, which is in turn the input to the next layer $l + 1$, where the dimension changes depending on the number and size of convolutions, padding and stride of the layers as illustrated in **Figure 1**. The padding parameter $p^{(l)} \in \mathbb{N}$, for layer l , describes the number of zeros at the edges of a field by which the field is extended. This is necessary since every convolution being larger than 1×1 will decrease the output size. The stride parameter $s^{(l)} \in \mathbb{N}$ is the parameter determining how much the kernel is shifted in each step to compute the next spatial position (x, y) . This specifies the overlap between individual output pixels, and it is here set to 1. Each layer l is specified by its number of kernels $K^{(l)} = \{K^{(l,1)}, K^{(l,2)}, \dots, K^{(l,d^{(l)})}\}$, where $d^{(l)} \in \mathbb{N}$ is the number of kernels in layer l , and its additive bias terms $b^{(l)} = \{b^{(l,1)}, b^{(l,2)}, \dots, b^{(l,d^{(l)})}\}$ with $b^{(l,d)} \in \mathbb{R}$. Note that the input $X^{(l,d)} \in \mathbb{R}^{h^{(l)} \times w^{(l)}}$ in the l -th layer with size $h^{(l)} \times w^{(l)}$, kernel k , and depth $d^{(l)}$ is processed by a set of kernels $\{K^{(l,d)}\}$. For each kernel $K^{(l,d)} \in \mathbb{R}^{h_K^{(l)} \times w_K^{(l)}}$ with size $h_K^{(l)} \times w_K^{(l)}$ and $d \in \{1, \dots, d^{(l)}\}$, the *raw output* $o^{(l)} \in \mathbb{R}^{\frac{h^{(l)} - h_K^{(l)} - 1 + p^{(l)}}{s^{(l)}} \times \frac{w^{(l)} - w_K^{(l)} - 1 + p^{(l)}}{s^{(l)}}}$ is computed

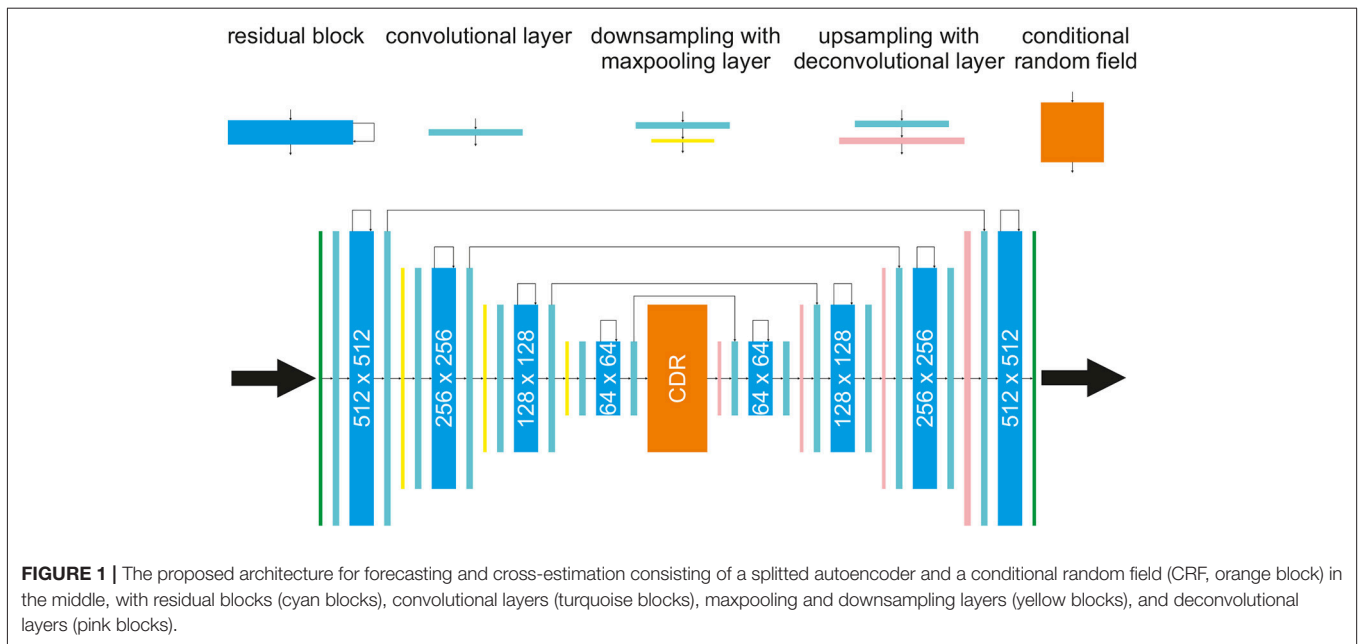


FIGURE 1 | The proposed architecture for forecasting and cross-estimation consisting of a splitted autoencoder and a conditional random field (CRF, orange block) in the middle, with residual blocks (cyan blocks), convolutional layers (turquoise blocks), maxpooling and downsampling layers (yellow blocks), and deconvolutional layers (pink blocks).

element by element as:

$$\begin{aligned}
 o_{x,y}^{(l,d)} &= b^{(l,d)} + \left(K^{(l,d)} * X^{(l,d)} \right)_{x,y} \\
 &= b^{(l,d)} + \sum_{k=1}^{d^{(l)}} \sum_{i=1}^{h_k^{(l)}} \sum_{j=1}^{w_k^{(l)}} K_{i,j}^{(l,d)} \cdot X_{x+i-1,y+j-1}^{(l,k)}. \quad (4)
 \end{aligned}$$

The result is clipped by an *activation function* ψ to obtain the *activation* $\psi(o_{x,y}^{(l,d)})$ of each unit in layer l :

$$\psi \left(o_{x,y}^{(l,d)} \right) = \max \left\{ 0, o_{x,y}^{(l,d)} \right\}. \quad (5)$$

To obtain $o^{(l)} = \{o^{(l,1)}, \dots, o^{(l,d^{(l)})}\}$, Equation (5) needs to be calculated $\forall d = 1, \dots, d^{(l)}$ and $\forall (x, y)$. Each spatial calculation of $o_{x,y}^{(l,d)}$ is considered as a *unit* and $\psi(o_{x,y}^{(l,d)})$ as the feedforward *activation* of the unit. The value of an element of a kernel $(K_{i,j}^{(l,d)})$ between two units is the weight of the feedforward connection. Such systems are well-suited for feature extraction [28], but their linear structure does not allow a direct modeling of temporal changes or the possibility to process a sequence of data. To enable temporal modeling, we employ linear-chain conditional random fields [31] that will be introduced in the next section.

2.4. Linear-Chain Conditional Random Fields

To implement a probabilistic forecasting block we consider the output of the convolutional layer \mathbf{o} and the corresponding forecast q as random variables \mathbf{O} and \mathbf{Q} , respectively. Both random variables \mathbf{O} and \mathbf{Q} are jointly distributed and in a predictive framework we aim at constructing a conditional model $P(\mathbf{Q}|\mathbf{O})$ from paired observation and forecast sequences. Let $G =$

(V, E) be a undirected graph such that $\mathbf{Q} = (\mathbf{Q}_v)_{v \in V}$, where \mathbf{Q} is indexed by the vertices of G . Each vertex in G represents a state, a history or a forecast. Then (\mathbf{O}, \mathbf{Q}) is a conditional random field (CRF), if conditioned on \mathbf{O} the random variables \mathbf{Q}_v obey the Markov property [24]. A linear-chain conditional random field, where \mathbf{o} is a sequence of historical extracted features and q a corresponding forecasted feature in the future, is given by:

$$\begin{aligned}
 P(q | \mathbf{o}, \theta) &= \sum_{\mathbf{h} \in \mathcal{H}} P(q, \mathbf{h} | \mathbf{o}, \theta) \\
 &= \frac{\sum_{\mathbf{h} \in \mathcal{H}} \exp(\Psi(q, \mathbf{h}, \mathbf{o}; \theta))}{\sum_{q' \in \mathcal{Q}} \sum_{\mathbf{h} \in \mathcal{H}} \exp(\Psi(q', \mathbf{h}, \mathbf{o}; \theta))}, \quad (6)
 \end{aligned}$$

where $q \in \mathcal{Q}$, \mathcal{Q} is a set of future events, $\mathbf{h} \in \mathcal{H}$, \mathcal{H} is the set of layers of the CRF where each element h_i of \mathbf{h} represents a historical state of an event at time t . θ is the set of parameters. $\Psi(q, \mathbf{h}, \mathbf{o}; \theta)$ is a so called *potential function* (also called *local* or *compatibility function*) which measures the compatibility (i.e., high probability) between a forecast, a set of observed features, and a configuration of historical states, such that:

$$\begin{aligned}
 \Psi(q, \mathbf{h}, \mathbf{o}; \theta) &= \sum_{j=1}^n \phi_j(\mathbf{o}, \omega) \cdot \theta_h[h_j] \\
 &+ \sum_{j=1}^n \theta_y[y, h_j] + \sum_{(i,j) \in \epsilon} \theta_\epsilon[q, h_j, h_k] + \frac{\phi(\mathbf{o}, \omega) \cdot \theta_p[q]}{k}, \quad (7)
 \end{aligned}$$

Here n is the number of historical states and $\phi_j(\mathbf{o}, \omega)$ is a vector that can include any feature of the observation specific for a specific time window ω , and $\theta = [\theta_h, \theta_q, \theta_\epsilon, \theta_p]$ are model parameters. To restrict the search space for possible parametrizations only sine, cosine, and a linear interpolation

function are allowed to be used as feature functions. $\theta_h[h_j]$ is the parameter that corresponds to the state h_j . The function $\theta_q[q, h_j]$ indicates the parameters that corresponds to the forecast q and the state h_j . $\theta_e[q, h_i, h_k]$ refers to parameters that describe the dependency relation between the nodes h_i and h_k . $\theta_p[q]$ defines the parameters for q given the features over the past, while the dot product $\phi_j(\mathbf{o}, \omega) \cdot \theta_n[h_j]$ measures the compatibility between the observed features and the state at time j . In contrast to this $\phi(\mathbf{o}, \omega) \cdot \theta_p[q]$ measures the compatibility between observation and the forecast. \mathbf{h} consists of $k = 1,024$ elements and the last term in Equation (7) captures the influence of the past features on the forecast. For training the following likelihood function is defined:

$$L(\theta) = \sum_{i=1}^n P(q_i | \mathbf{o}_i, \theta) - \frac{1}{2\sigma^2} \|\theta\|^2, \tag{8}$$

where n is the number of training examples. By maximizing the likelihood for the forecasted training data the optimal parameter set θ^* is determined. To find θ^* Equation (8) can be evaluated by the same gradient descent method which is used for optimizing/training the autoencoder. To forecast the input sequence with a linear-chain CRF it is necessary to compute the q sequence that maximizes the following equation:

$$\hat{q} = \arg \max_q P(q | \mathbf{o}; \theta^*) \tag{9}$$

The sequence maximizing this is then used by the deconvolutional part of the network to map the features back to the desired system variables at $t + \Delta t$.

3. MODELING EXCITABLE MEDIA

Excitable systems are non-linear dynamical systems with a stable fixed point. Small perturbations of the stable equilibrium decay, but stronger perturbations above some characteristic threshold lead to a high amplitude excursion in state space until the trajectory returns to the stable fixed point. In neural or cardiac cells this response leads to a so-called action potential. After such a strong response a so-called *refractory period* has to pass until the next excitation can be initialized by perturbing the system again. An excitable medium consists of excitable systems (e.g., cells), which are spatially coupled. Electric coupling of neighboring cardiac cells, for example, can be modeled by means of a diffusion term for local currents. The resulting partial differential equations (PDEs) describe the propagation of undamped solitary excitation waves. Due to the refractory time of local excitations spiral or scroll waves are very common and typical hallmarks of excitable media, which can lead to stable periodically rotating wave patterns or may break-up forming complex chaotic wave dynamics. From the large selection of different PDE models describing excitable media we have chosen the Bueno-Orovio-Cherry-Fenton (BOCF) model which was devised as an efficient model for cardiac tissue [11].

3.1. Bueno-Orovio-Cherry-Fenton Model

The Bueno-Orovio-Cherry-Fenton (BOCF) model [11] provides a compact description of excitable cardiac dynamics. We use this model as a benchmark to validate our approach for forecasting and cross-estimation of complex wave patterns in excitable media. The evolution of the four system variables of the BOCF model is given by four PDEs

$$\begin{aligned} \frac{\partial u}{\partial t} &= D \cdot \nabla^2 u - (J_{si} + J_{fi} + J_{so}) \\ \frac{\partial v}{\partial t} &= \frac{1}{\tau_v^-} (1 - H(u - \theta_v)) (v_\infty - v) - \frac{1}{\tau_v^+} H(u - \theta_v) v \\ \frac{\partial w}{\partial t} &= \frac{1}{\tau_w} (1 - H(u - \theta_w))(w_\infty - w) - \frac{1}{\tau_w^+} H(u - \theta_w) w \\ \frac{\partial s}{\partial t} &= \frac{1}{2\tau_s} ((1 + \tanh(k_s(u - u_s))) - 2s), \end{aligned} \tag{10}$$

where u represents the membrane voltage and $H(\cdot)$ denotes the Heaviside function. The three currents J_{si} , J_{fi} and J_{so} are given by the equations

$$\begin{aligned} J_{si} &= -\frac{1}{\tau_{si}} H(u - \theta_w) w s \\ J_{fi} &= -\frac{1}{\tau_{fi}} v H(u - \theta_v)(u - \theta_v)(u - u_o) \\ J_{so} &= \frac{1}{\tau_o} (u - u_o)(1 - H(u - \theta_w)) + \frac{1}{\tau_{so}} H(u - \theta_w). \end{aligned} \tag{11}$$

Furthermore, seven voltage dependent variables

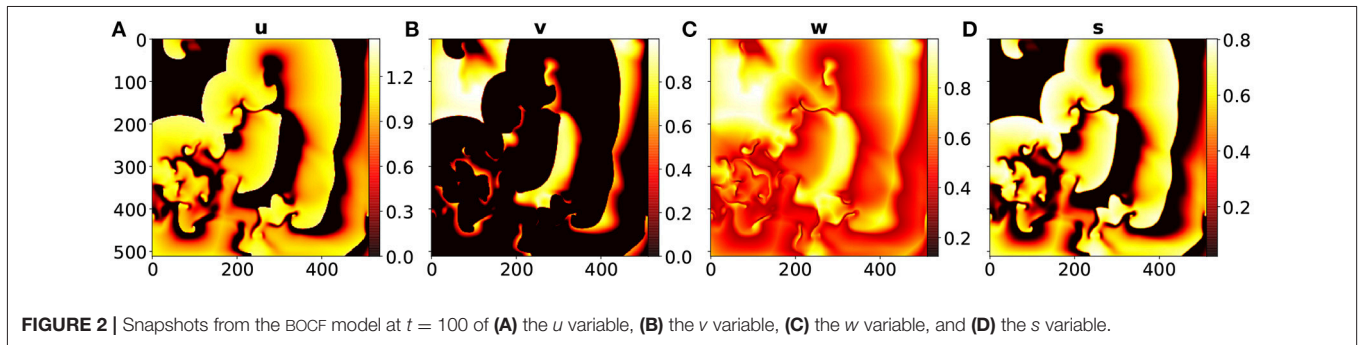
$$\begin{aligned} \tau_v^- &= (1 - H(u - \theta_v^-))\tau_{v1}^- + H(u - \theta_v^-)\tau_{v2}^- \\ \tau_w^- &= \tau_{w1}^- + \frac{1}{2}(\tau_{w2}^- - \tau_{w1}^-)(1 + \tanh(k_w^-(u - u_w^-))) \\ \tau_{so}^- &= \tau_{so1} + \frac{1}{2}(\tau_{so2} - \tau_{so1})(1 + \tanh(k_{so}(u - u_{so}))) \\ \tau_s &= (1 - H(u - \theta_w))\tau_{s1} + H(u - \theta_w)\tau_{s2} \\ \tau_o &= (1 - H(u - \theta_o))\tau_{o1} + H(u - \theta_o)\tau_{o2} \\ v_\infty &= \begin{cases} 1, & \text{if } u \leq \theta_v^- \\ 0, & \text{if } u \geq \theta_v^- \end{cases} \\ w_\infty &= (1 - H(u - \theta_o))(1 - \frac{u}{\tau_{w\infty}}) + H(u - \theta_o)w_\infty^* \end{aligned} \tag{12}$$

are required. The characteristic model dynamics is determined through 28 parameters. In our simulations we used a set of parameters [11] given in **Table 1** for which the BOCF model exhibits chaotic excitation wave dynamics similar to the *Ten Tusscher-Noble-Noble-Panfilov* (TNNP) model [32].

The spatio-temporal chaotic dynamics of this system is actually transient chaos whose lifetime grows exponentially with system size [33, 34]. To obtain chaotic dynamics with a sufficiently long lifetime the system has been simulated on a domain of 512×512 grid points with a grid constant of $\Delta x = 1.0$ space units and a diffusion

TABLE 1 | TNNP model parameter values for the BOCF model [11].

u_o	0	τ_{v2}^-	1150	τ_{fi}	0.11	τ_{s1}	2.7342
u_u	1.58	τ_{v1}^+	1.4506	τ_{o1}	6	τ_{s2}	3
θ_v	0.3	τ_{w1}^-	70	τ_{o2}	6	k_s	2.0994
θ_w	0.015	τ_{w2}^-	20	τ_{so1}	43	u_s	0.9087
θ_v^-	0.015	k_w^-	65	τ_{so2}	0.2	τ_{sj}	2.8723
θ_o	0.006	u_w^-	0.03	k_{so}	2	$\tau_{w\infty}$	0.07
τ_{v1}^-	60	τ_w^+	280	u_{so}	0.65	w_∞^*	0.94



constant $D = 0.2$. Furthermore, an explicit Euler stepping in time with $\Delta t = 0.1$ time units¹, a 5 point approximation of the Laplace operator, and no-flux boundary conditions were used. **Figure 2** shows typical snapshots of the dynamics.

4. RESULTS

The proposed network model was trained with simulated data generated by the BOCF model with parameter values given in **Table 1**. Ten trajectories with different initial conditions for the variables u, v, w , and s were generated by simulating the BOCF model for a time series of 50,000 samples spanning a period of time of 5 s. Five of these data sets randomly chosen, were used to train the network, while the other solutions were used for validation. For training the Adam optimizer [35] was used, with a learning rate $lr = 0.0001$ and $\beta_1 = 0.9$, $\beta_2 = 0.999$.

In order to quantify the performance of the estimation and prediction methods the similarity of target fields and output fields of the network has to be quantified. For this purpose we use the Jensen-Shannon divergence (JSD) [36] applied to normalized fields of the variables of the BOCF model. The JSD of two discrete probability distributions A and B is defined as

$$JSD(A||B) = \frac{1}{2}D_{KL}(A||M) + \frac{1}{2}D_{KL}(B||M), \quad (13)$$

¹We consider all variables and parameters of the BOCF model as dimensionless. The parameter values given in **Table 1** are, however, consistent with the choice of a time unit equalling 1ms. In this case all t -values given in this article would correspond to milli seconds.

where $M = \frac{1}{2}(A + B)$ and $D_{KL}(A||M)$ is the Kullback-Leibler divergence [37]:

$$D_{KL}(A||M) = - \sum_i P(i) \log \left(\frac{A(i)}{M(i)} \right). \quad (14)$$

During training the JSD was used as objective function to be minimized (for a GPU implementation of the JSD see [38]). The JSD is bounded by 0 and 1 and a value below 0.02 was considered to indicate no discernible differences between the two distributions (fields). An alternative for quantifying the deviation would be the Fractions Brier Score [39]. For training the network, for each trajectory at each time step, sequences of lengths up to $m = 10$ were used as input.

The input of the network consisted of fields of variables that were assumed to be measured and random fields representing variables that were considered to be not available.

4.1. Forecast

For forecasting the input of the network consisted of sequences of length $m = 10$ of u, v, w , and s given by $\{u_{t-m+1}, \dots, u_t\}$, $\{v_{t-m+1}, \dots, v_t\}$, $\{w_{t-m+1}, \dots, w_t\}$, and $\{s_{t-m+1}, \dots, s_t\}$. The desired output of the network is then $u_{t+\Delta t}, v_{t+\Delta t}, w_{t+\Delta t}$ and $s_{t+\Delta t}$. By using the output of the network as a new input the system can be run iteratively in a closed loop for long term prediction. The development of the JSDs of u, v, w, s through time are shown in **Figure 3A**. Since the u and the s fields look quite similar (see **Figures 2A,D**) their JSD-values are almost the same. The w -field (**Figure 2C**) exhibits relatively high values at all spatial locations and therefore the JSD of two such fields is rather low. On the other hand, the v field (**Figure 2B**) possesses only very localized structures with high values and this leads to rather high values of the JSD for (slightly) different fields. **Figure 3B**

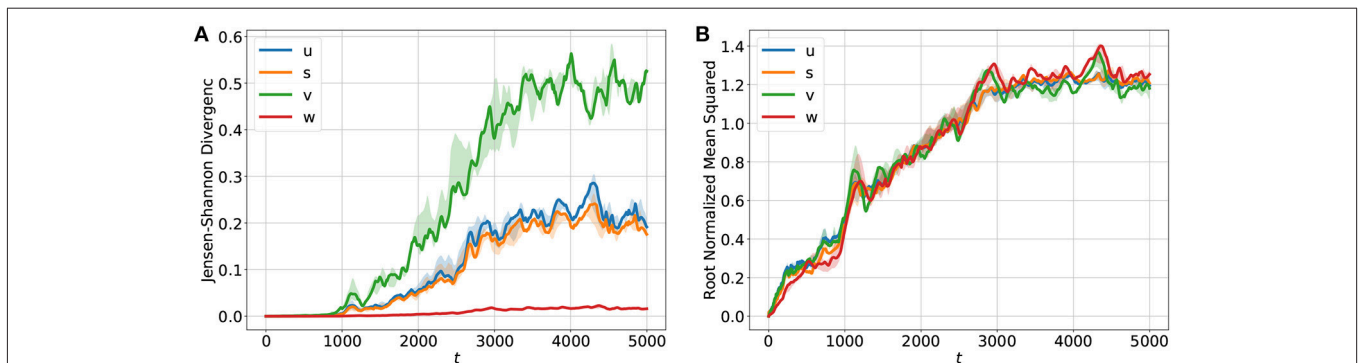


FIGURE 3 | Temporal development of (A) the Jensen-Shannon divergence (JSD) and (B) the root normalized mean squared error (RNMSE) for all variables u, v, w, s showing the deviation of the iterative network prediction (in a feedback mode) from the reference orbit obtained with the BOCF model. During the period $[0 - 1000]$ the predicted and the true fields agree very well as indicated by very small values of the JSD. In the time interval $(1000 - 3000)$ the JSD values increase until they saturate and the forecasts become very poor and useless. The RNMSE values show a similar increase in time but turn out to be more sensitive to minor deviations during the initial phase $[0 - 1000]$ of the forecast. The solid curves show median values of JSD and RNMSE obtained from ten different initial values of u, v, w, s . The transparent areas visualize the 0.25/0.75 percentile.

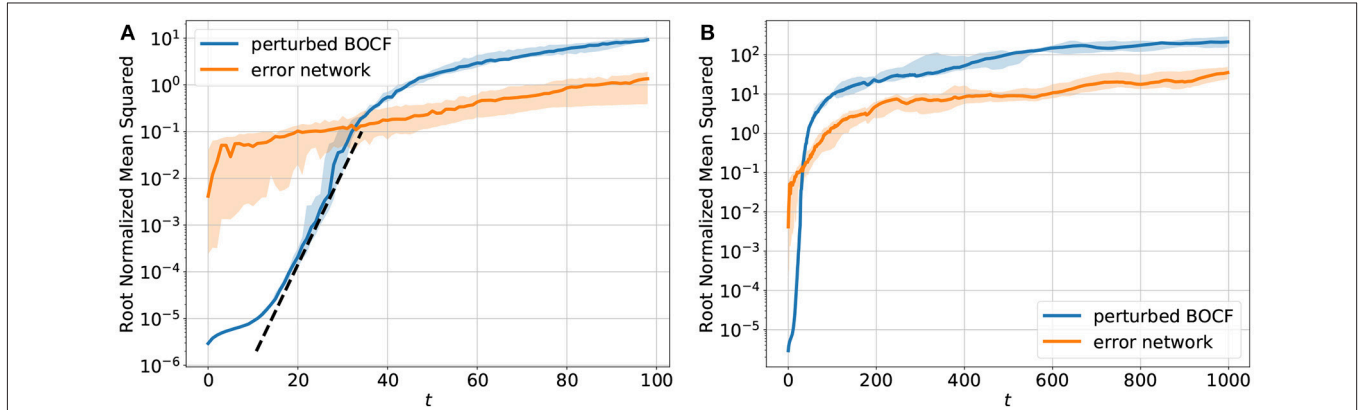


FIGURE 4 | Temporal development of the sum of the root normalized mean squared errors (RNMSE) of all variables u, v, w, s . (A) shows the NMSE for $t \in [1, 100]$ and (B) shows the NMSE for $t \in [1, 1000]$. The orange curve describes the deviation of the trajectory generated by the network from the reference orbit simulated with the BOCF model. For comparison the blue curve shows the distance between the reference orbit and a second solution of the BOCF model obtained by perturbing the initial conditions where each variable was perturbed at every spatial location using Gaussian random noise ($\mu = 0, \sigma^2 = 10^{-11}$). The error dynamics of ten perturbed trajectories was analyzed. These orbits were obtained by perturbing the reference orbit at different times $[0, 1000], [1000, 2000], \dots, [9000, 10000]$. The blue curve shows the median and the 0.25/0.75 percentile is visualized by the transparent areas. The dotted black line (A) denotes the slope the linear part of the $\log(\text{NMSE})$ vs. t curve which provides an estimate of the largest Lyapunov exponent [40] $\lambda_1 \approx 0.25$ (with respect to the natural logarithm).

shows for comparison the root normalized mean squared errors (RNMSE) of all variables u, v, w, s which is given by

$$\text{RNMSE}(v) = \sqrt{\frac{\text{MSE}(v)}{\text{MSE}(\bar{v})}} \tag{15}$$

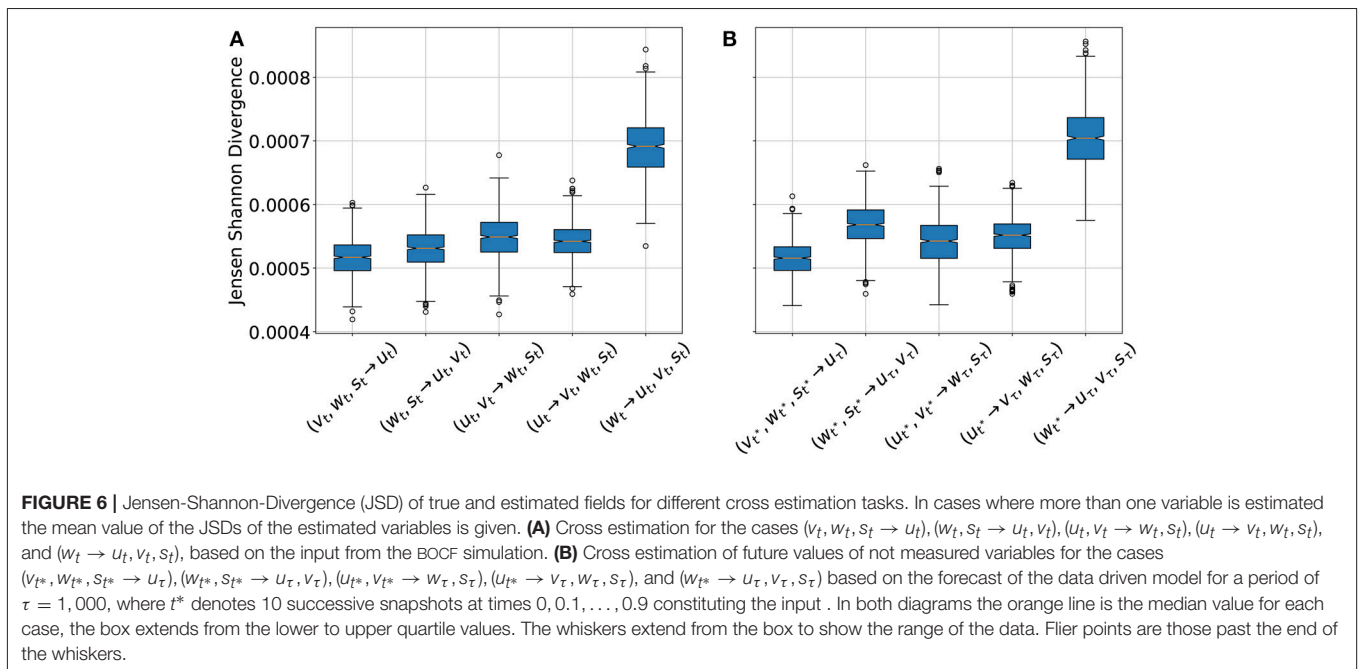
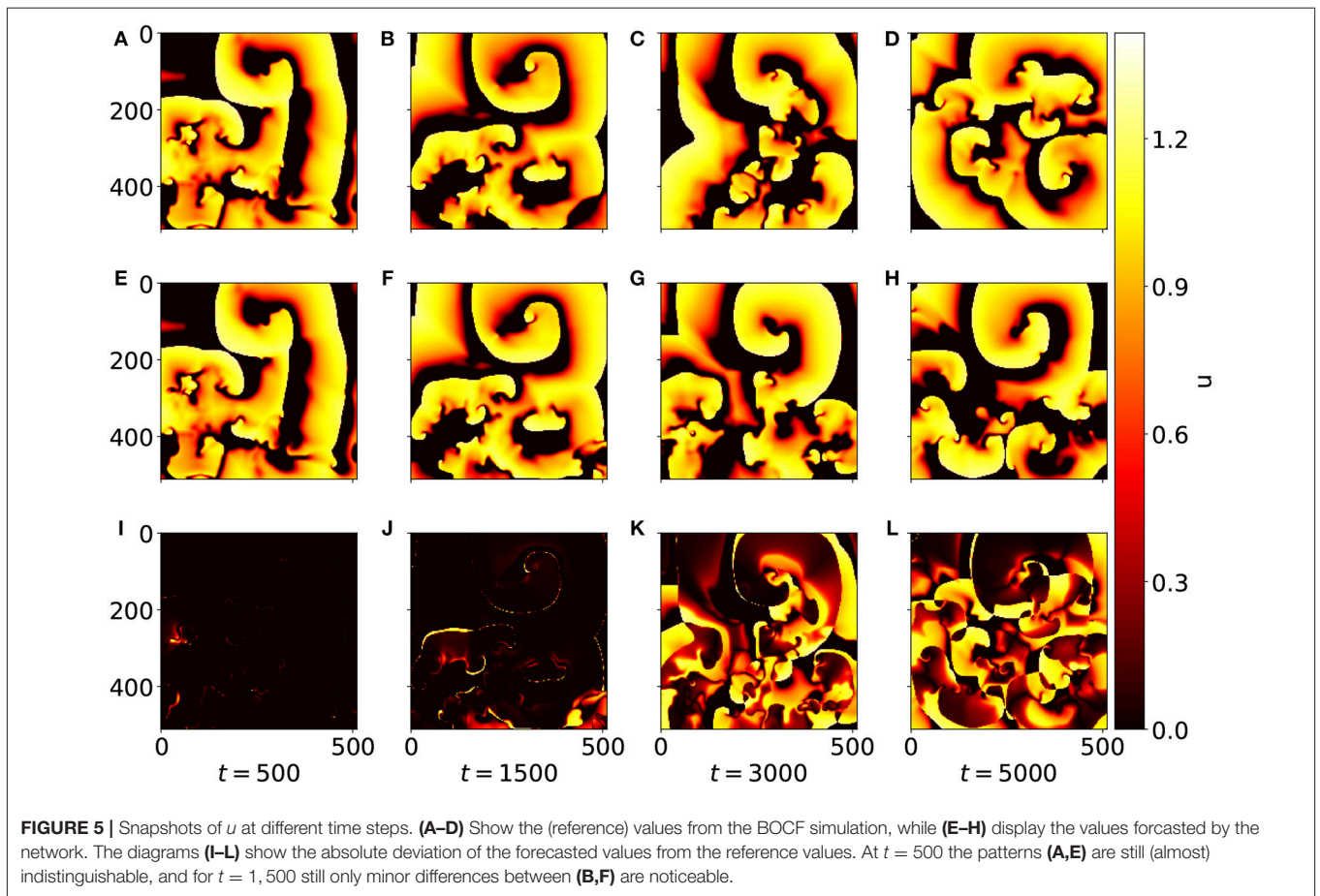
where

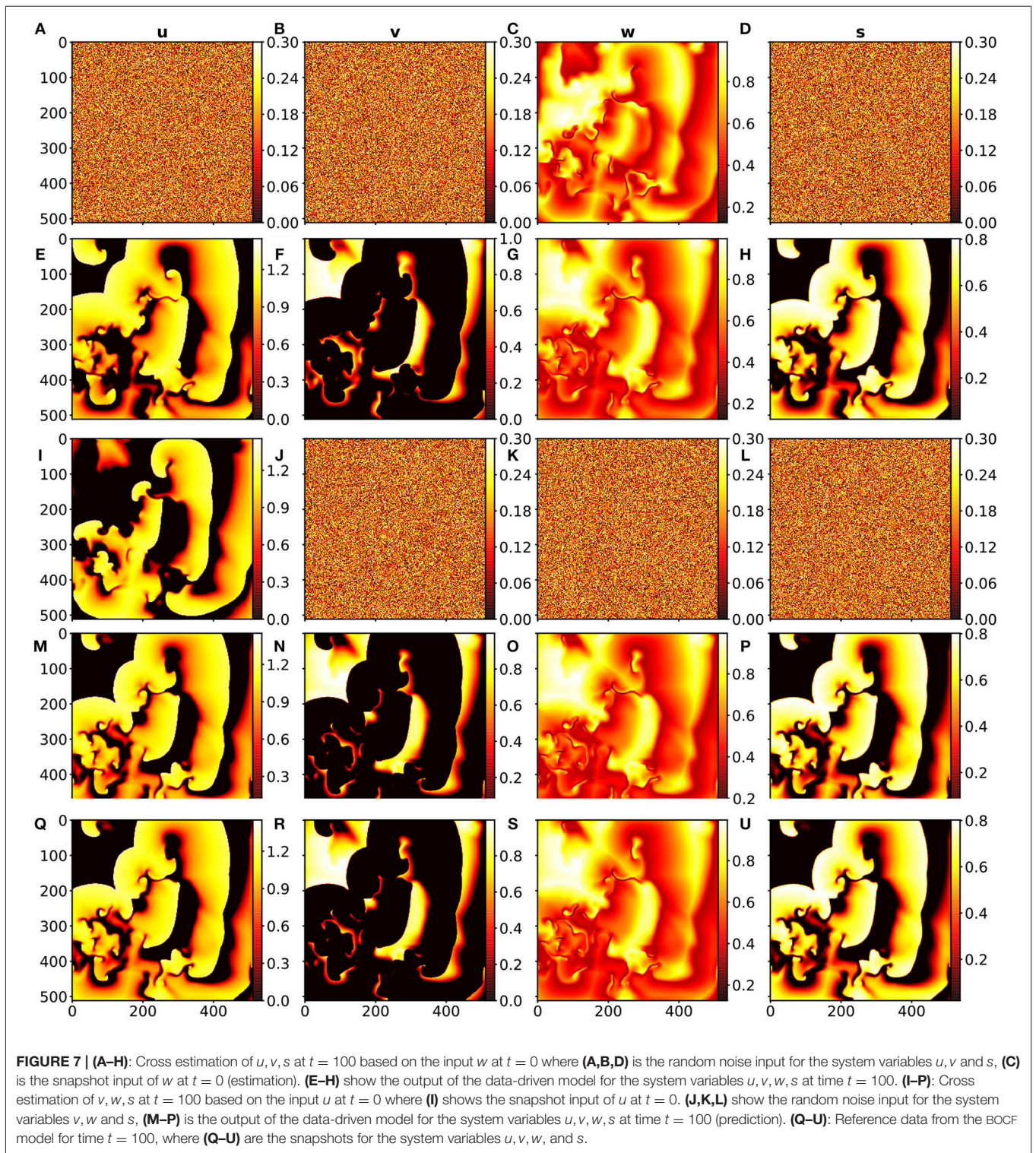
$$\text{MSE}(v) = \frac{1}{M^2} \sum_{i=1}^M \sum_{j=1}^M \left(v_{ij}^{\text{BOCF}}(t) - v_{ij}(t) \right)^2 \tag{16}$$

Here \bar{v} denotes the temporal and spatial mean values of the BOCF sequence of length T_F , $M^2 = 512 \cdot 512$ is the number of grid points of the domain and v_{ij}^{BOCF} denotes the value of variable v at

grid point (i, j) for the reference solution generated by the BOCF model. As can be seen in **Figure 3A** all four curves possess very similar values and indicate an increase of the error already during the initial period for $t \in [0, 1000]$.

Figure 4 shows a comparison of the error dynamics of the forecast obtained with the iterated network with feedback (orange curve) and the dynamics of a BOCF model starting from slightly perturbed initial conditions (blue curve). Both curves give the root normalized mean squared error (RNMSE) with respect to the same reference orbit generated by the BOCF model. The perturbation of the initial condition of the second BOCF solution with respect to the initial condition of the reference orbits was chosen to be very small. Therefore, during the initial phase the deviation still remains so small that (with semi-logarithmic axes) a linear segment of the error curve occurs that





can be used to estimate the largest Lyapunov exponent [40]. Once the error of the perturbed BOCF orbit (blue curve) reaches the level of the network prediction error (orange curve) both error curves continue to increase in the same way indicating that the network almost perfectly learned the true dynamics of the BOCF model.

To illustrate the deviation between the u field forecasted by the network and the (true) u field provided by the simulation of the BOCF PDE **Figure 5** shows snapshots at times $t = 500, t = 1,500, t = 3,000,$ and $t = 5,000$. While at $t = 500$ original (A) and forecast (E) are almost indistinguishable the snapshots at $t = 1,500$ exhibit minor differences (**Figures 5B,F**). At time

$t = 3,000$ only rough structures agree (**Figures 5C,G**) until at $t = 5,000$ forecast and simulation appear completely decorrelated (**Figures 5D,H**). The full evolution of the forecast compared to the original dynamics generated with the BOCF model is also available as a movie (**Supplemental Data**). Compared to a typical spiral rotation period of approximately $T_{sp} = 350$ good forecasting results can be obtained for about five spiral rotations corresponding to $5 \cdot 350 / 4 = 437$ Lyapunov times $T_L = 1/\lambda_1 \approx 4$ given by the largest Lyapunov exponent $\lambda_1 \approx 0.25$ (see **Figure 4**).

4.2. Cross-Estimation

For cross-estimation only a part of the system variables are considered as being directly observable or measurable. Based on these available variables the other not measurable variables have to be estimated (a task also called cross prediction). In the context of the BOCF model we shall, for example, estimate v_t, w_t, s_t from observations of u_t , only. Since the network expects all system variables as input the not observed variables were replaced by uniform noise in the range of $0 - 0.3$. For this purpose for every $t \in [0, 1000]$ the data of the BOCF model were used as single time step input for the network and the cases $(v_t, w_t, s_t \rightarrow u_t)$, $(w_t, s_t \rightarrow u_t, v_t)$, $(u_t, v_t \rightarrow w_t, s_t)$, $(u_t \rightarrow v_t, w_t, s_t)$, and $(w_t \rightarrow u_t, v_t, s_t)$ were considered as estimation tasks. **Figure 6** shows the JSD statistics for all these cases. The low JSD values for $(v_t, w_t, s_t \rightarrow u_t)$ indicated that the variable u can be very well estimated by the variables v, w, s , which could be expected because the variable u is part of the PDEs of the other variables. Similarly good estimation results are obtained for $(u_t \rightarrow v_t, w_t, s_t)$ which is remarkable, because the membrane potential u is the variable, which can be measured most easily in experiments and the result shows that this information is sufficient to recover the other variables v, w , and s of the BOCF model. The worst performance is achieved if only w is used to cross estimate all other system variables. These cross estimation results are in very good agreement with the performance of an Echo State Network applied to similar data [19].

4.3. Forecast and Cross-Estimation

This investigation represents a combination of the two previous ones. In this case, however, not for every time step the data from the BOCF model were used, but only ten time steps from the BOCF model were used to initialize the forecast of the network. Depending on the case which variable should be estimated the BOCF variables for initialization were replaced by uniform noise, as before. **Figure 6B** shows the JSD statistics for the four estimation cases considered and in **Figure 7** snapshots of the input and the true and estimated fields are presented illustrating the very good performance at time $t = 100$.

5. DISCUSSION

Spatio-temporal non-linear dynamical systems like extended systems (described by PDEs) or networks of interacting

oscillators may exhibit very high dimensional chaotic dynamics. A typical example are complex wave pattern occurring in some excitable media. As a representative of this class of systems we used the BOCF model describing electrical excitation waves in cardiac tissue where chaotic dynamics is associated with cardiac arrhythmias. For future applications like monitoring and predicting the dynamical state of the heart or the impact of interventions, mathematical models are required describing the temporal evolution or the relation between different (physical) variables. As an alternative to the large number of simple qualitative or detailed (ionic) models (incorporating many biophysical details and corresponding variables) we presented a machine learning approach for data driven modeling of the spatio-temporal dynamics. A convolutional neural network combined with a linear-chain of conditional random fields was trained and validated with data generated by a simulation of the BOCF model. To mimic experimental limitations when measuring cardiac dynamics we considered different cases where only some of the variables of the BOCF model were assumed to be available as input of the generated model and the not measurable variables were replaced by random numbers. Running the trained network in a closed loop (feedback) configuration iterated prediction provided forecasts of the complex dynamics that turned out to follow the true (chaotic!) evolution of the BOCF simulation for about five periods of the intrinsic spiral rotations. These results clearly show that machine learning methods like those employed here provide faithful models of the underlying complex dynamics of excitable media that, when suitably trained can provide powerful tools for predicting the spatio-temporal evolution and for cross-estimating not directly observed variables.

AUTHOR CONTRIBUTIONS

SH performed numerical simulations. UP and SH identified the scientific topic. UP, FW, and SH devised the strategy for solving it, and wrote the manuscript.

FUNDING

SH acknowledges funding by the International Max Planck Research Schools of Physics of Biological and Complex Systems.

ACKNOWLEDGMENTS

We thank T. Lilienkamp and S. Luther for support with the BOCF model and for inspiring discussions about spatio-temporal dynamics in excitable media.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: <https://www.frontiersin.org/articles/10.3389/fams.2018.00060/full#supplementary-material>

Supplemental Data | Movie showing the temporal evolution of the u field from the simulation, the forecast and the absolute difference of both.

REFERENCES

- Barkley D. A model for fast computer simulation of waves in excitable media. *Physica D Nonlinear Phenomena* (1991) **49**:61–70.
- Bär M, Eiswirth M. Turbulence due to spiral breakup in a continuous excitable medium. *Phys Rev E* (1993) **48**:R1635–7.
- Cherry EM, Fenton F, Krogh-Madsen T, Luther S, Parlitz U. Introduction to focus issue: complex cardiac dynamics. *Chaos* (2017) **27**:093701. doi: 10.1063/1.5003940
- Clayton RH, Bernus O, Cherry EM, Dierckx H, Fenton FH, Mirabella L, et al. Models of cardiac tissue electrophysiology: progress, challenges and open questions. *Prog Biophys Mol Biol.* (2011) **104**:22–48. doi: 10.1016/j.pbiomolbio.2010.05.008
- Uzelac I, Ji YC, Hornung D, Schröder-Scheteling J, Luther S, Gray RA, et al. Simultaneous quantification of spatially discordant alternans in voltage and intracellular calcium in langendorff-perfused rabbit hearts and inconsistencies with models of cardiac action potentials and Ca transients. *Front Physiol.* (2017) **8**:819. doi: 10.3389/fphys.2017.00819
- Evensen G, van Leeuwen P. An ensemble Kalman smoother for nonlinear dynamics. *Mon Weather Rev.* (2000) **128**:1852–67. doi: 10.1175/1520-0493(2000)128<1852:AEKSFN>2.0.CO;2
- Abarbanel HDI. *Predicting the Future - Completing Models of Observed Complex Systems.* New York, NY: Springer-Verlag (2013).
- Law K, Stuart A, Zygalakis K. *Data Assimilation - A Mathematical Introduction.* Cham: Springer International Publishers (2015).
- Hoffman MJ, LaVigne NS, Scorse ST, Fenton FH, Cherry EM. Reconstructing three-dimensional reentrant cardiac electrical wave dynamics using data assimilation. *Chaos* (2016) **26**:013107. doi: 10.1063/1.4940238
- Berg S, Luther S, Parlitz U. Synchronization based system identification of an extended excitable system. *Chaos* (2011) **21**:033104. doi: 10.1063/1.3613921
- Bueno-Orovio A, Cherry EM, Fenton FH. Minimal model for human ventricular action potentials in tissue. *J Theor Biol.* (2008) **253**:544–60. doi: 10.1016/j.jtbi.2008.03.029
- Box GE, Jenkins GM, Reinsel GC, Ljung GM. *Time Series Analysis: Forecasting and Control.* San Francisco, CA: John Wiley & Sons (2015).
- Solomatine DP, Ostfeld A. Data-driven modelling: some past experiences and new approaches. *J Hydroinformatics* (2008) **10**:3–22. doi: 10.2166/hydro.2008.015
- Parlitz U, Merkwirth C. Prediction of spatiotemporal time series based on reconstructed local states. *Phys Rev Lett.* (2000) **84**:1890. doi: 10.1103/PhysRevLett.84.1890
- Jaeger H. *The 'Echo State' Approach to Analysing and Training Recurrent Neural Networks - With an Erratum Note.* GMD Report (2001).
- Jäger H, Haas H. Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science* (2004) **304**:78–80. doi: 10.1126/science.1091277
- Verstraeten D, Schrauwen B, D'Haene M, Stroobandt D. An experimental unification of reservoir computing methods. *Neural Netw.* (2007) **20**:391–403. doi: 10.1016/j.neunet.2007.04.003
- Pathak J, Hunt B, Girvan M, Lu Z, Ott E. Model-free prediction of large spatiotemporally chaotic systems from data: a reservoir computing approach. *Phys Rev Lett.* (2018) **120**:024102. doi: 10.1103/PhysRevLett.120.024102
- Zimmermann RS, Parlitz U. Observing spatio-temporal dynamics of excitable media using reservoir computing. *Chaos* (2018) **28**:043118. doi: 10.1063/1.5022276
- Quade M, Abel M, Nathan Kutz J, Brunton SL. Sparse identification of nonlinear dynamics for rapid model recovery. *Chaos* (2018) **28**:063116. doi: 10.1063/1.5027470
- Abebe A, Solomatine D, Venneker R. Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events. *Hydrol Sci J.* (2000) **45**:425–36. doi: 10.1080/02626660009492339
- Abebe AJ, Guinot V, Solomatine DP. Fuzzy alpha-cut vs. Monte Carlo techniques in assessing uncertainty in model parameters. In: *Proc. 4-th International Conference on Hydroinformatics.* Iowa City, IA (2000). p. 1–8.
- Krizhevsky A, Sutskever I, Hinton GE. Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems.* Curran Associates Inc. (2012). p. 1097–105.
- Lafferty JD, McCallum A, Pereira FCN. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In: *Proceedings of the Eighteenth International Conference on Machine Learning. ICML '01.* San Francisco, CA: Morgan Kaufmann Publishers Inc. (2001). p. 282–9.
- Vemuri V. *Artificial Neural Networks.* Rockville, MD: Computer Science Press Inc. (1988).
- LeCun Y, Bengio Y, Hinton G. Deep learning. *Nature* (2015) **521**:436–44. doi: 10.1038/nature14539
- Schmidhuber J. Deep learning in neural networks: an overview. *Neural Netw.* (2015) **61**:85–117. doi: 10.1016/j.neunet.2014.09.003
- Hinton GE, Salakhutdinov RR. Reducing the dimensionality of data with neural networks. *Science* (2006) **313**:504–7. doi: 10.1126/science.1129198
- He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. *arXiv:1512.03385* (2015).
- Noh H, Hong S, Han B. Learning deconvolution network for semantic segmentation. *arXiv:1505.04366* (2015).
- Chatzis SP, Demiris Y. The infinite-order conditional random field model for sequential data modeling. *IEEE Trans Pattern Anal Mach Intell.* (2013) **35**:1523–34. doi: 10.1109/TPAMI.2012.208
- Ten Tusscher K, Noble D, Noble P, Panfilov AV. A model for human ventricular tissue. *Am J Physiol Heart Circ Physiol.* (2004) **286**:H1573–89. doi: 10.1152/ajpheart.00794.2003
- Strain MC, Greenside HS. Size-dependent transition to high-dimensional chaotic dynamics in a two-dimensional excitable medium. *Phys Rev Lett.* (1998) **80**:2306–9. doi: 10.1103/PhysRevLett.80.2306
- Lilienkamp T, Christoph J, Parlitz U. Features of chaotic transients in excitable media governed by spiral and scroll waves. *Phys Rev Lett.* (2017) **119**:054101. doi: 10.1103/PhysRevLett.119.054101
- Kingma DP, Ba J. Adam: a method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).
- Lin J. Divergence measures based on the Shannon entropy. *IEEE Trans Inform Theor.* (1991) **37**:145–51.
- Kullback S, Leibler RA. On information and sufficiency. *Ann Math Stat.* (1951) **22**:79–86.
- Gültas M, Düzgün G, Herzog S, Jäger SJ, Meckbach C, Wingender E, et al. Quantum coupled mutation finder: predicting functionally or structurally important sites in proteins using quantum Jensen-Shannon divergence and CUDA programming. *BMC Bioinformatics* (2014) **15**:96. doi: 10.1186/1471-2105-15-96
- Roberts N. Assessing the spatial and temporal variation in the skill of precipitation forecasts from an NWP model. *Meteorol Appl.* (2008) **15**:163–9. doi: 10.1002/met.57
- Parlitz U. Estimating Lyapunov Exponents from Time Series. In: Skokos C, Gottwald G, Laskar J, editors. *Chaos Detection and Predictability. Lecture Notes in Physics.* Vol 915. Berlin; Heidelberg: Springer (2016). p. 1–34.

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Copyright © 2018 Herzog, Wörgötter and Parlitz. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner(s) are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.

3.3.2.1 Conclusions from (Herzog et al. 2018)

The last work somewhat completes to the studies performed in *(Herzog et al. 2021c). It can be seen that even if the system is only partially observed, the full state can be reconstructed and predicted. As *(Herzog et al. 2021c) had already shown, the CAE alone has very good reconstructive properties and these properties are not interfered by the integration of the CRF in the latent space, the opposite is the case, the combination of the CAE and CRF can even deal with the case of where a incomplete set of observations is used as an input for prediction, although the prediction with all variables allows a longer prediction horizon.

3.3.3 Publication: (Herzog et al. 2020b)

The last publication to be presented in this chapter is Herzog, S. and Wagner, C. (2020b). “Development of Artificial Neural Networks with Integrated Conditional Random Fields Capable of Predicting Non-linear Dynamics of the Flow Around Cylinders”. In: *New Results in Numerical and Experimental Fluid Mechanics XII*. Cham: Springer International Publishing, pp. 71–79. It demonstrates the use of the approach from *(Herzog et al. 2019) on a computational fluid dynamics (CFD) case, but also shows some generalising properties of the approach. To study the generalising properties one network was trained with data containing different parameters, more specifically the initial velocity was 0.02 and 0.05 u_{LB} , where u_{LB} is the velocity in lattice units per time step, as well different Reynolds numbers (Re) of 100, 500, 1000, leading to six different cases. However, the validation of the network was done on cases where the $Re = 250$ and $Re = 750$ and $u_{LB} = 0.04$. Furthermore, it was investigated what happens when the position and number of cylinders are changed without retraining the network.



Development of Artificial Neural Networks with Integrated Conditional Random Fields Capable of Predicting Non-linear Dynamics of the Flow Around Cylinders

Sebastian Herzog^{1,2,3(✉)} and Claus Wagner^{1,4}

¹ German Aerospace Center (DLR), Institute of Aerodynamics and Flow Technology, Bunsenstr. 10, 37073 Göttingen, Germany

sebastian.herzog@dlr.de

² Max Planck Institute for Dynamics and Self-Organization, Am Faßberg 17, 37077 Göttingen, Germany

³ Third Institute of Physics, University of Göttingen, Friedrich-Hund-Platz 1, 37077 Göttingen, Germany

⁴ Institute of Thermodynamics and Fluid Mechanics, Technische Universität Ilmenau, Helmholtzring 1, 98693 Ilmenau, Germany

Abstract. This paper presents a new approach intended to predict flow dynamics based on observed data. The approach uses artificial neural networks extended by an adapted conditional random field. This artificial neural network is trained end-to-end and the embedded conditional random field memorizes previous events and uses this memory for flow predictions. The prediction capability of the proposed method is demonstrated for flows around cylinders which are computed with a Lattice Boltzmann method in order to train the artificial neural network.

Keywords: System modeling · Machine learning · ANNs

1 Introduction

To reliably and unambiguously describe physical phenomena, the use of mathematical models is almost inevitable. However, the development of these models relies on new data obtained from difficult experiments or simulations. This new data can be either fitted by adapting parameters of existing mathematical models or new mathematical models are required to describe the physical phenomena of interest as precisely as possible. Either way, a high level of expert knowledge is mandatory and even then, it is still challenging. In the field of fluid dynamics, artificial neural networks (ANN) [15] were used early on. [17] introduced a non-linear unsteady reduced-order model (ROM), which was developed using radial basis function neural networks to analyze the limit cycle oscillation for two linear

structural models with large shock motions in transonic flow. [12] introduced a different approach where the ROM is structured as a continuous time recurrent neural network and verified on a typical section of the “benchmark active control technology” wing. This approach is useful when the investigated dynamic system cannot be reconstructed from a sequence of state observations (Takens’s theorem [13]). Further, [16] presented a similar approach. It is, however, based on recurrent local linear neurofuzzy models for the prediction of generalized aerodynamic forces in the time domain. In [4] several recommendations are formulated on how and where ANNs can be used in the field of fluid dynamics. Besides these approaches, there are also solutions for generating models from (training) data including autoregressive models [2] or adaptive fuzzy rule-based models [1]. Furthermore, Monte Carlo techniques may be used for assessing uncertainty in model parameters [1]. In the presented approach the dynamics of the considered dynamic system learned using only the recorded data. After learning the model should describe the experiment precisely and with a high level of generalizability.

2 Methods

In data-driven modeling, mathematical models are not based on first principles (e.g., Newton’s laws, Maxwell’s equations, ...) but directly derived from experimental data. In this work, we present a modeling ansatz combining deep convolutional neural networks (CNNs) [8] for feature extraction and dimension reduction with an adapted version of conditional random fields (CRFs) [9] in order to model the properties of temporal sequences. We start by recapping basic ANNs: ANNs are parameterizable models for approximating an (unknown) function F implicitly given by the data. The actual function provided by the ANN,

$$f : \mathbb{R}^O \mapsto \mathbb{R}^P, \quad (1)$$

should be an approximation of F , i.e. $f \simeq F$. Here, $O \in \mathbb{N}$ and $P \in \mathbb{N}$ denote the dimension of the input and the output of f , respectively. A widely used type of ANN are feed-forward neural networks (FNN) where, in general, f is given by

$$f(X) = \psi(WX + b) \quad (2)$$

with a nonlinear function ψ (a possible choice would be: Eq. (5)), a weight matrix $W \in \mathbb{R}^{P \times O}$, some input $X \in \mathbb{R}^{O \times R}$, and a bias $b \in \mathbb{R}^P$. Equation (2) is called a one-layer FNN. By recursively applying the output of one layer as input to the next layer, a multi-layer FNN can be constructed:

$$\mathbf{f}(X) = f^{(L)}(\dots f^{(2)}(f^{(1)}(X; W^{(1)}, b^{(1)}); W^{(2)}, b^{(2)}) \dots; W^{(L)}, b^{(L)}), \quad (3)$$

describing a multi-layer FNN with $L \in \mathbb{N}$ layers. In the following, an input with several variables is taken into account. The input is denoted as $\mathbf{X}_\alpha \in \mathbb{R}^{h^{(0)} \times w^{(0)}}$, with $h^{(0)} \in \mathbb{N}$ rows, $w^{(0)} \in \mathbb{N}$ columns, and $\alpha = 1, \dots, d$ as the number of variables. To improve the approximation properties of the network Eq. (3),

FNNs may contain additional convolutional layers leading to state-of-the-art models for data classification, so-called CNNs [8]. CNNs receive a training data set $\mathcal{X} = \{\mathbf{X}_1, \mathbf{X}_2, \dots, \mathbf{X}_m\}$, where $\mathbf{X}_\alpha \in \mathbb{R}^{h \times w}$ is a field with $h \in \mathbb{N}$ rows, $w \in \mathbb{N}$ columns, and $\alpha = 1, \dots, d$ number of variables/fields. In this case, the input for the network has the dimensions $h \times w \times d$. Data processing through the network is described layer-wise: In the l -th convolutional layer the input $\mathbf{X}^{(l)}$ is transformed into the *raw output* $\mathbf{o}^{(l)}$, which is in turn the input into the next layer $l + 1$, where the dimensions change depending on the number and size of convolutions, padding and stride of the layers as illustrated in Fig. 1. Our proposed network consists of four main components: **1. convolutional layers:** [11] to extract relevant features from the input, **2. downsampling with maxpooling layers:** [11] to further reduce the data dimensions, **3. adapted CRF:** Eq. (6) is used as a kind of “working memory”, it remembers the past features and predicts them. **4. upsampling with deconvolutional layers:** [6] inverse functions of the downsampling and convolutional layers. They transform the features back into the original mathematical space. This entire system is trained end-to-end. To prevent gradient vanishing during training, **residual blocks** [5] are used as helper components. The convolutional layers are always followed by rectified linear unit activation (Eq. (5)) and batch normalization [7] (these blocks are omitted in Fig. 1 for reasons of simplicity). Each residual layer consists of three convolutional blocks and a residual skip connection. A maxpooling layer is located between the levels in the encoding path to perform downsampling for feature compression. The dimensions of the layers are based on the architecture of an *Autoencoder* [6] consisting of an encoding path (from 512×512 to 64×64) and a symmetric decoding path (from 64×64 back to 512×512), as illustrated in Fig. 1. For the cases presented in this work two channels were used: one for

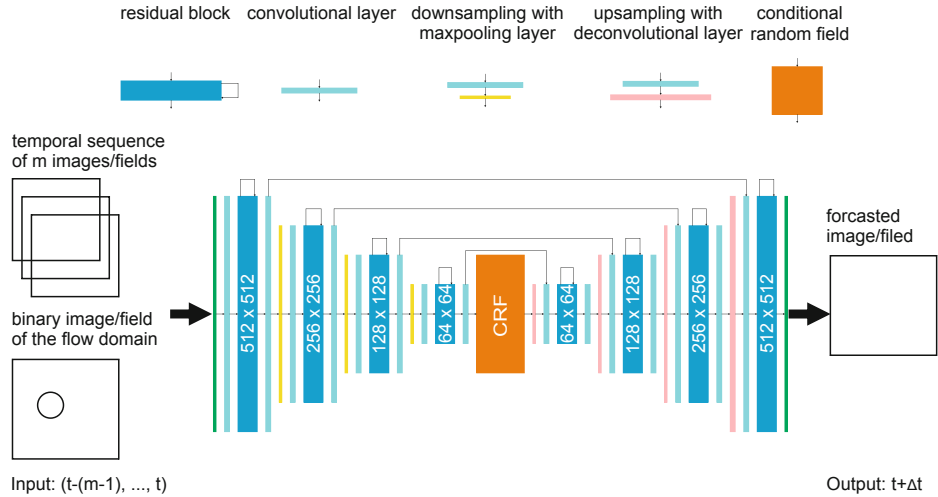


Fig. 1. Proposed architecture for the prediction consisting of a splitted autoencoder and the adapted CRF (orange block) with residual layers (cyan blocks), convolutional layers (turquoise blocks), maxpooling and downsampling layers (yellow blocks), and deconvolutional layers (pink blocks). As input, $m = 10$ images/fields are used.

the velocity field u and another one for the size and shape of the flow domain. The padding parameter $P^{(l)} \in \mathbb{N}$ for layer l describes the number of zeros at the edges of a field by which the field is extended. This is necessary since every convolution larger than 1×1 will decrease the output size. The stride parameter $S^{(l)} \in \mathbb{N}$ determines how much the kernel is shifted in each step to compute the next spatial position x, y . This specifies the overlap between the individual output pixels, and is set to 1 in this case. Each layer l is specified by its number of kernels $\mathbf{K}^{(l)} = \{K^{(l,1)}, K^{(l,2)}, \dots, K^{(l,d^{(l)})}\}$, where $d^{(l)} \in \mathbb{N}$ is the number of kernels in layer l , and its additive bias terms $\mathbf{b}^{(l)} = \{b^{(l,1)}, b^{(l,2)}, \dots, b^{(l,d^{(l)})}\}$ with $b^{(l,d)} \in \mathbb{R}$. Note that the input $\mathbf{X}^{(l)} \in \mathbb{R}^{h^{(l)} \times w^{(l)}}$ in the l -th layer with size $h^{(l)} \times w^{(l)}$ and depth $d^{(l)}$ is processed by a set of kernels $\mathbf{K}^{(l)}$. For each kernel $K^{(l,d)} \in \mathbb{R}^{h_K^{(l)} \times w_K^{(l)}}$ with size $h_K^{(l)} \times w_K^{(l)}$ and $d \in \{1, \dots, d_K\}$, the *raw output* $o^{(l)} \in \mathbb{R}^{\frac{h^{(l)} - h_K^{(l)} - 1 + P^{(l)}}{S^{(l)}} \times \frac{w^{(l)} - w_K^{(l)} - 1 + P^{(l)}}{S^{(l)}}}$ is computed element by element as:

$$o_{x,y}^{(l,d)} = (\mathbf{X}^{(l)} * K^{(l,d)})_{x,y} = b^{(l,d)} + \sum_{k=1}^{d^{(l)}} \sum_{i=1}^{h_K^{(l)}} \sum_{j=1}^{w_K^{(l)}} K_{i,j}^{(l,d)} \cdot X_{x+i-1,y+j-1}^{(l,k)}. \quad (4)$$

Applying a kernel to an input is called a convolution operation. In every convolutional layer different kernels are applied. Kernel parameters are determined during the training by a gradient descent method. The result is modulated by an *activation function* ψ to obtain the *activation* $\psi(o_{x,y}^{(l,d)})$ of each unit:

$$\psi(o_{x,y}^{(l,d)}) = \max\{0, o_{x,y}^{(l,d)}\}. \quad (5)$$

To obtain $\mathbf{o}^{(l)} = \{o^{(l,1)}, \dots, o^{(l,d_K)}\}$, Eq. (5) needs to be solved $\forall d = 1, \dots, d_K$ and $\forall x, y$. Each spatial element of $o_{x,y}^{(l,d)}$ is considered as a *unit* and $\psi(o_{x,y}^{(l,d)})$ as the feedforward *activation* of the unit. The value of a kernel element ($K_{i,j}^{(l,d)}$) between two units is the weight of the feedforward connection. Such systems are well-suited for feature extraction [6], but their linear structure does not allow direct modeling of temporal changes or the possibility to process a sequence of data. To enable temporal modeling, we employ adapted CRF which will be introduced in the following: We assumed the output of the convolutional layer to be \mathbf{o} and the corresponding prediction q to be the random variables \mathbf{O} and \mathbf{Q} , respectively. \mathbf{O} and \mathbf{Q} are jointly distributed. In a predictive framework, we aim at developing a conditional model $P(\mathbf{Q}|\mathbf{O})$ based on paired observation and predicted sequences. Let $G = (V, E)$ to be an undirected graph such that $\mathbf{Q} = (\mathbf{Q}_v)_{v \in V}$, where \mathbf{Q} is indexed by the vertices of G . Each vertex in G represents a state, a history or a prediction. Consequently, (\mathbf{O}, \mathbf{Q}) is a CRF [9]. An adapted CRF, where \mathbf{o} is a sequence of historical extracted features up to time t and q represents a corresponding predicted feature at $t + \Delta t$, with Δt being the time step between two consecutive training dates, is given by:

$$P(q | \mathbf{o}, \theta) = \sum_{\mathbf{h} \in \mathcal{H}} P(q, \mathbf{h} | \mathbf{o}, \theta) = \frac{\sum_{\mathbf{h} \in \mathcal{H}} \exp(\Psi(q, \mathbf{h}, \mathbf{o}; \theta))}{\sum_{q' \in \mathcal{Q}} \sum_{\mathbf{h} \in \mathcal{H}} \exp(\Psi(q', \mathbf{h}, \mathbf{o}; \theta))}. \quad (6)$$

In Eq. (6) $q \in \mathcal{Q}$, \mathcal{Q} is a set of future events, $\sum_{q' \in \mathcal{Q}}$ describes the sum of all possible future events based on the trained data, $\mathbf{h} \in \mathcal{H}$, \mathcal{H} is the set of layers of the CRF and each $h_i \in \mathbf{h}$ is a historical state of an event at time t . θ is the set of parameters. $\Psi(q, \mathbf{h}, \mathbf{o}; \theta)$ is a so-called *potential function* (also called *local* or *compatibility function*):

$$\begin{aligned} \Psi(q, \mathbf{h}, \mathbf{o}; \theta) = & \sum_{j=1}^n \phi_j(\mathbf{o}, \omega) \cdot \theta_{\mathbf{h}}[h_j] + \sum_{j=1}^n \theta_y[y, h_j] \\ & + \sum_{(j,k) \in \epsilon} \theta_{\epsilon}[q, h_j, h_k] + \frac{\phi(\mathbf{o}, \omega) \cdot \theta_p[q]}{k}, \end{aligned} \quad (7)$$

which measures the probability between a prediction, observed features, and a configuration of historical states. Here, n is the number of historical states and $\phi_j(\mathbf{o}, \omega)$ is a vector that can include any feature of the observation specific to a certain time window ω , and $\theta = [\theta_{\mathbf{h}}, \theta_q, \theta_{\epsilon}, \theta_p]$ are model parameters. To restrict the search space for possible parametrizations, only sine, cosine and linear interpolation functions are allowed to be used as feature functions. $\theta_{\mathbf{h}}[h_j]$ is the parameter that corresponds to the state $h_j \in \mathbf{h}$. The function $\theta_q[q, h_j]$ indicates the parameters which correspond to the prediction q and the state h_j . $\theta_{\epsilon}[q, h_j, h_k]$ refers to parameters describing the dependency relation between the nodes h_i and h_k . $\theta_p[q]$ defines the parameters for q . The dot product $\phi(\mathbf{o}, j, \omega) \cdot \theta_{\mathbf{h}}[h_j]$ measures the compatibility between the observed features and the state at time j . In contrast to this, $\phi(\mathbf{o}, \omega) \cdot \theta_p[q]$ measures the compatibility between the observation and the prediction. \mathbf{h} consists of k elements (in this work $k = 1024$) and the last term in Eq. (7) captures the influence of the past features on the prediction. For training, the likelihood function is defined:

$$L(\theta) = \sum_{i=1}^n P(q_i | \mathbf{o}_i, \theta) - \frac{1}{2\sigma^2} \|\theta\|^2, \quad (8)$$

where n is the number of training examples. By maximizing the likelihood for the true prediction based on the training data, the optimal parameter set θ^* is determined. To determine θ^* , Eq. (8) can be evaluated by means of the same gradient descent method as used for the autoencoder. To predict the input sequence with the introduced CRF, the q sequence maximizing $\hat{q} = \arg \max_q P(q | \mathbf{o}; \theta^*)$ must be computed. The maximizing sequence is then used by the deconvolutional part of the network to map the features back to the desired system variables at $t + \Delta t$. The CNN part of the network was implemented with Keras/Tensorflow 1.10 [3]. The CRF was implemented with Python 3.6 [14], from scratch.

3 Results

The showcase of the flow around a cylinder provided by the Palabos LBM library [10] is used for demonstration. To generate training data for the ANN, six cases

are initially computed with the LBM for an initial velocity of $u_{LB} = 0.02$ and $u_{LB} = 0.05$, where u_{LB} is the velocity given in lattice units per time step. Varying the Reynolds number (based on u_{LB} and the radius of the cylinder in lattice units) between three different values (100, 500 and 1000) results in six cases. The lattice dimensions are $n_x = 520$ and $n_y = 180$, and the cylinder position is $c_x = n_x/4$ $c_y = n_y/2$ with the radius $r = n_y/9$, all indicated in lattice units. The three snapshots of the velocity fields obtained for the three Reynolds numbers and $u_{LB} = 0.05$ shown in Fig. 2 reflect vortex streets whose nonlinearity increases with the Reynolds number. The network is trained with the velocity fields computed for these three Reynolds numbers and $u_{LB} = 0.05$. Two additional simulations are performed for the Reynolds numbers $Re = 250$ and $Re = 750$ for $u_{LB} = 0.04$ which are not used for the training process. Their solutions are compared to the prediction of the network and are used to determine the accuracy of the latter. For all cases, the input sequence of the network is $m = 10$. In this respect, the prediction of the network for $Re = 750$ is compared to the LBM solution, see Fig. 3. Starting the $m = 10$ initial field in the first iteration, the prediction of the network is used for the second iteration in an open loop until the maximal number of iterations is reached. Using one Nvidia GTX 1080 Ti, 997 GPU min. are required for training the network and 3 GPU min. for the prediction of one case on average. For comparison, processing 200000 time steps with the LBM on an AMD Ryzen 1800X takes 491 CPU min. Additionally, the network trained with the flow around one cylinder is used to predict the flow around three cylinders in a flow domain twice as long and wide. The position and the size of the two additional cylinders illustrated in Fig. 4 are chosen arbitrarily; in this case 3 GPU min. and 23 GPU sec. The network prediction must be compared to the 1325 CPU min. of the LBM simulation for this case. To analyze the deviation of the network prediction from the LBM solutions in more detail,

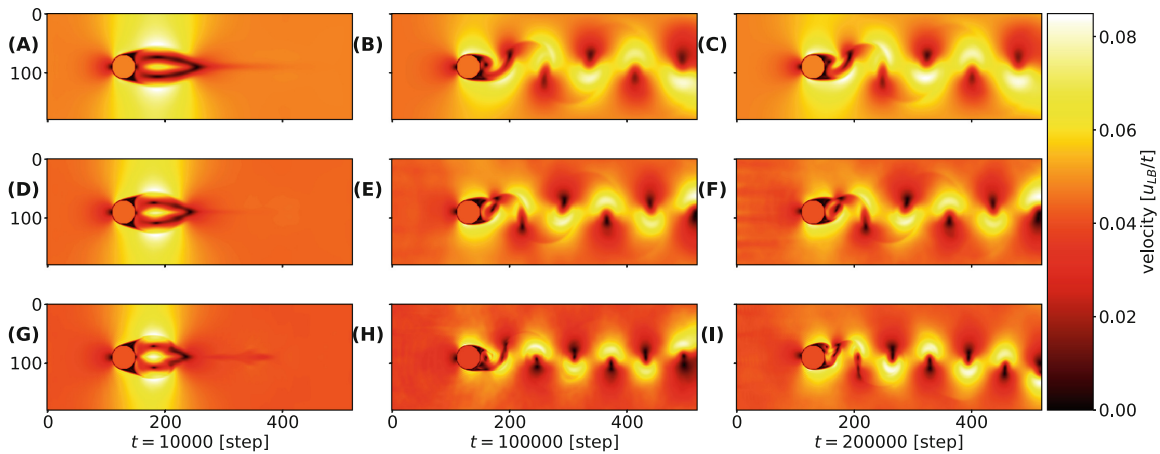


Fig. 2. Color-coded contours of the velocity distribution obtained from the LBM for three different Re values with $u_{LB} = 0.05$. (A)–(C) correspond to $Re = 100$, (D)–(F) correspond to $Re = 500$ and (G)–(I) correspond to $Re = 1000$. The individual columns (from left to right) correspond to the time steps $t \in [10000, 100000, 200000]$ and the velocity is given in lattice units.

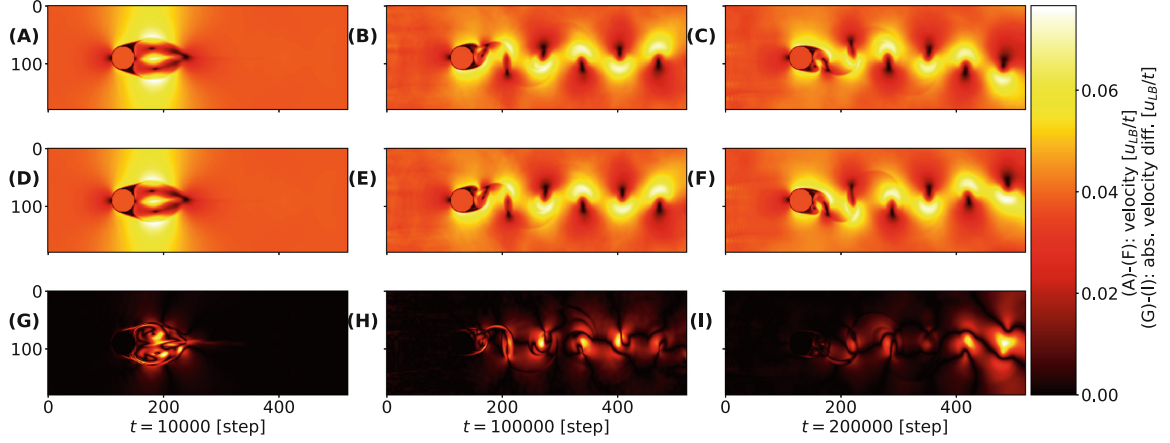


Fig. 3. Color-coded contours of the velocity distribution obtained from the LBM (A)–(C) and the network (D)–(F) for $Re = 750$, $u_{LB} = 0.04$. (G)–(I) present the absolute error between the LBM simulation and the network prediction.

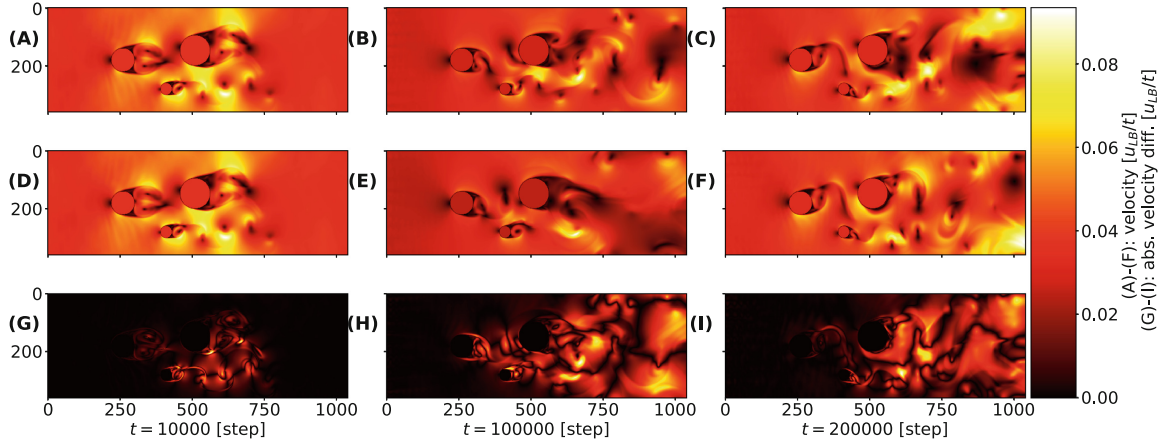


Fig. 4. Comparison of the LBM (A)–(C) and the network prediction (D)–(F), with a doubled domain size and two additional cylinders with randomly chosen positions, for $Re = 750$, $u_{LB} = 0.04$ and $t \in [10000, 100000, 200000]$. The network was not retrained. (G)–(I) show the absolute error between the LBM simulation and their reconstruction by the network. The velocity is given in lattice units.

the root-mean-square deviation $\text{RMSD} = \frac{1}{n_x n_y} \sqrt{\sum_{i=1}^{n_x} \sum_{j=1}^{n_y} (u_{i,j}^{\text{LBM}} - u_{i,j}^{\text{Net}})^2}$ is determined, where u^{LBM} denotes the LBM solution and u^{Net} the output of the network. Figure 5 illustrates the temporal development of the RMSD obtained for the single cylinder cases ($Re = 250; u_{LB} = 0.04$), ($Re = 750; u_{LB} = 0.04$) and ($Re = 750; u_{LB} = 0.04$) and the case with three cylinders which is similar for all four cases. With increasing Re the predicted vortex streets become more chaotic and thus, more susceptible to perturbations. This results in a higher RMSD over time for the case of $Re = 750$ in comparison to the case of $Re = 250$. The exponentially increasing RMSD and progression, however, are comparable in both cases. On the other hand, it must be noted that the RMSD of $Re = 750$ is smaller compared to the RMSD for the case $Re = 250$ in the beginning,

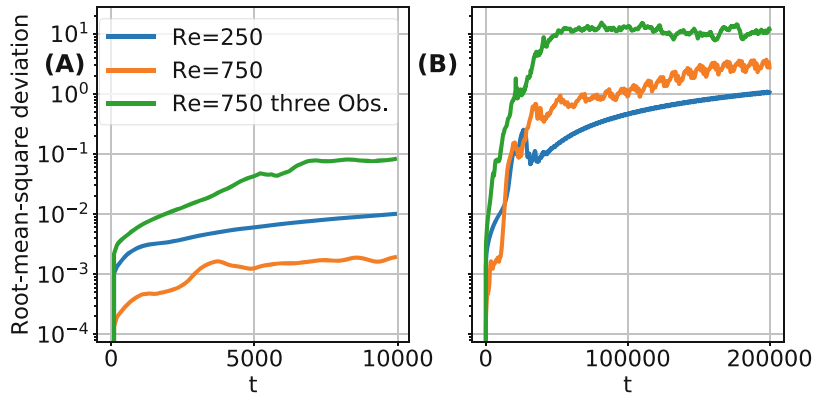


Fig. 5. Temporal development of the RMSD between the LBM and the network prediction, with different Re values. The blue and orange line are based on the two cases where only the parameters of the flow have been changed, so that $Re = 250$ or $Re = 750$ was reached. The green line shows the case with three cylinders and doubled domain size from Fig. 4. (A) is the development for $t \in [0, 10000]$ and (B) for $t \in [0, 200000]$

$t \in [0, 20000]$, while for $t > 20000$ the RMSD of $Re = 750$ exceeds the RMSD of $Re = 250$. Although the errors of the network prediction increase as expected, it is remarkable that the RMSD is still lower than 0.1 for $t \in [0, 10000]$ in spite of the fact that the network was not trained on data with more than one cylinder.

4 Conclusion

Spatio-temporal nonlinear dynamical systems like the Kármán vortex street may exhibit dynamics which are hard to learn for a data-driven modeling system. The presented modeling approach is based on the extension of ANNs with an adapted CDR, introducing a “working memory”, to remember past features and use this information for prediction. We used this approach to develop a data-driven model of nonlinear dynamical systems from data, which is able to predict the repeating pattern of swirling vortices downstream of cylinders. The latter are caused by vortex shedding, occurring due to the unsteady separation of the flow around blunt bodies. Running the trained network in a closed-loop (feedback) configuration resulted in a long-term prediction of the dynamics which follow the solution in a complete LBM simulation, when suitably trained.

Acknowledgment. We thank Annika Köhne for the valuable corrections of this manuscript and Axel Dannhauer for fruitful discussions and continuous support.

References

1. Abebe, A.J., Solomatine, D.P., Venneker, R.G.W.: Application of adaptive fuzzy rule-based models for reconstruction of missing precipitation events. *Hydrol. Sci. J.* **45**(3), 425–436 (2000)
2. Box, G.E.P., Jenkins, G.: *Time Series Analysis, Forecasting and Control*. Holden-Day Inc., San Francisco (1990)
3. Abadi, M., et al.: TensorFlow: large-scale machine learning on heterogeneous systems. Software available from [tensorflow.org](https://www.tensorflow.org) (2015)
4. Faller, W.E., Schreck, S.J.: Neural networks: applications and opportunities in aeronautics. *Prog. Aerosp. Sci.* **32**(5), 433–456 (1996)
5. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778 (2016)
6. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. *Science* **313**(5786), 504–507 (2006)
7. Ioffe, S., Szegedy, C.: Batch normalization: accelerating deep network training by reducing internal covariate shift. *arXiv preprint [arXiv:1502.03167](https://arxiv.org/abs/1502.03167)* (2015)
8. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: *Advances in Neural Information Processing Systems*, vol. 25, pp. 1097–1105. Curran Associates, Inc. (2012)
9. Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: *Proceedings of 18th International Conference on Machine Learning*, pp. 282–289 (2001)
10. Latt, J.: Palabos, parallel Lattice Boltzmann solver (2009)
11. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 609–616. ACM (2009)
12. Mannarino, A., Mantegazza, P.: Nonlinear aeroelastic reduced order modeling by recurrent neural networks. *J. Fluid Struct.* **48**, 103–121 (2014)
13. Takens, F.: Detecting strange attractors in turbulence. In: *Dynamical Systems and Turbulence, Warwick 1980*, pp. 366–381. Springer (1981)
14. van Rossum, G., Drake, F.L.: *The Python Language Reference Manual*. Network Theory Ltd., Bristol (2011)
15. Vemuri, V.: Artificial neural networks: theoretical concepts. In: *Computer Society Press Technology, Neural Networks*. Computer Society Press of the IEEE (1988)
16. Winter, M., Breitsamter, C.: Neurofuzzy-model-based unsteady aerodynamic computations across varying freestream conditions. *AIAA J.* **54**, 2705–2720 (2016)
17. Zhang, W., Wang, B., Ye, Z., Quan, J.: Efficient method for limit cycle flutter analysis based on nonlinear aerodynamic reduced-order models. *AIAA J.* **50**(5), 1019–1028 (2012)

3.3.3.1 Conclusions from (Herzog et al. 2020b)

As can be seen in *(Herzog et al. 2020b, figure 3), the hybrid approach predicts dynamics that fit well to the ground truth data, even for the case where the initial velocity and Reynolds number was not part of the training data, which is quite an interesting result. Adding more cylinders has led to predictions that are consistent, although the error has increased more than in the other cases, as one can recognise from *(Herzog et al. 2020b, figure 5). This can be seen as an indication that the presented approach is able to generalise across different boundary values and geometries to some degree. However, further investigations need to be carried out in this context. My hypothesis based on the findings from this work would be that the network can interpolate between the training data. If, instead, parameters were used that were not within the range of the training data, e.g. cases where the Reynolds number was set to $Re = 2000$ resulted in artefacts or sometimes to cases where the dynamics slowly converge to that of $Re = 1000$. Anyhow, it was all very sensitive and strongly parameter dependent.

3.4 Summary and outlook

In general, publications *(Herzog et al. 2019; Herzog et al. 2018; Herzog et al. 2020b), which presented the hybrid model, show that the combination approach CAE+CRF has interesting properties and seems to be very powerful. Unfortunately, the generalisation behaviour is difficult to assess, *(Herzog et al. 2020b) shows that generalisation between the cases from the training data is indeed possible. However, extrapolation does not work well. Another question that remains and needs further investigation is *why* the combination of CAE and CRF works so well. The current hypothesis is that the CAE transforms the data into a coordinate system that is particularly easy for the CRF to predict. Which could mean that a finite Koopmann operator (Koopman 1931) is approximated by the CAE in the latent space. In general, the Koopmann operator is an infinite-dimensional linear operator defining a infinite-dimensional linear dynamical system which transfers the state vector \mathbf{x}_t from

the original dynamical system to the next state $\mathbf{x}_{t+\Delta t}$, where Δt is a suitable step size. Since the latent space in CAE cannot be infinite the finite Koopmann operator is assumed, which can be considered as an approximation of the infinite Koopmann operator. A first simple examination of this suggestion can be made as follows.

If one considers the CAE simplified as

$$\phi^{-1}(\phi(\mathbf{x}_t)) = \mathbf{x}_t$$

for any input \mathbf{x} at some time t then $\phi(\mathbf{x}_t)$ would be the encoder part of the CAE and \mathbf{x}_t^* the latent space representation $\mathbf{x}_t^* = \phi(\mathbf{x}_t)$. And $\phi^{-1}(\cdot)$ shall then be the decoder part of the CAE. Then the framework from *(Herzog et al. 2019) can be considered as

$$\phi^{-1}(\text{CRF}(\phi(\mathbf{x}_t))) = \mathbf{x}_{t+\Delta t},$$

where ΔT is the time step between tow data points. Without loss of generality, $\Delta t \in \mathbb{N}_+$, it is possible to write

$$\phi^{-1}(\text{CRF}(\phi(\mathbf{x}_t))) = \mathbf{x}_{t+1}.$$

Assuming that a transfer to a simpler linear coordinate space takes place in the latent space, then there must exist a $\mathbf{K} \in \mathbb{R}^{n \times n}$, where n is the size of the input, such that

$$\phi^{-1}(\mathbf{K}\phi(\mathbf{x}_t)) = \mathbf{x}_{t+1}.$$

In the best of all cases even a \mathbf{K} would exit, such that

$$\phi^{-1}(\mathbf{K}^m \phi(\mathbf{x}_t)) = \mathbf{x}_{t+m+1}, \quad (3.6)$$

where $m \in \mathbb{N}$ and $m \gg 0$.

To investigate this, the network from *(Herzog et al. 2019) was applied to data from *Lorenz's model from 1996* with $N = 32$. In the first step, the already trained network was taken to obtain \mathbf{x}_t^* and $\mathbf{x}_{t+1}^* = \text{CRF}(\mathbf{x}_t^*)$ over all training data.

Remark. The approach of *(Herzog et al. 2019) does not work with $\mathbf{x} \in \mathbb{R}^n$ but with $\mathbf{x} \in \mathbb{R}^{m \times n}$ so the input is a chunk that looks at the spatial positions (m) and their temporal evolution (n). In the following consideration, \mathbf{x} is therefore broken down into its columns.

With this data, a \mathbf{K} should then be found that satisfies equation 3.6. To determine \mathbf{K} , the same gradient procedure was used as in *(Herzog et al. 2019). If one tries to find a \mathbf{K}^m that is valid over the entire prediction horizon that the CAE+CRF approach could reach, the optimisation fails and a \mathbf{K} is not found. But if we take an m corresponding to the latent dimension, a set of \mathcal{K} can be found which reaches an error according to the used float precision. Using \mathcal{K} to set up the CAE without CRF allows a predictive horizon of $\Lambda_{\max t} = 13.03$ or $7.945t$ (For comparison, the CAE+CRF approach achieves $\Lambda_{\max t} = 15.66$ or $9.55t$). Which is significantly worse but can certainly be seen as an indication that a linear system \mathbf{K} exists which transfers \mathbf{x}^*_t to \mathbf{x}^*_{t+1} .

Furthermore, an attempt was made to integrate the \mathbf{K} directly into the training process in order to prevent the approximation of \mathbf{K} , i.e. the error of $\|\mathbf{K}\mathbf{x}^*_t - \mathbf{x}^*_{t+1}\|$ was added reciprocally into the loss function of the CAE+CRF training. In short, however, this has not led to any convergence of training. But this approach was probably not a good idea either since for each input during the training a second gradient procedure has to be started that optimises \mathbf{K} , which resulted in an extremely long computation time.

These two attempts should by no means be understood as rigorous results. Much more, they should form the motivation for further investigations. With this thought, I would like to conclude this chapter. In summary, one can say that hybrids of neural networks are able to predict even complex non-linear systems over a certain time horizon, and their application to systems that are not describable by first principals could prove a very useful tool.

Philosophy is written in that great book which ever lies before our eyes — I mean the universe — but we cannot understand it if we do not first learn the language and grasp the symbols, in which it is written. This book is written in the mathematical language, and the symbols are triangles, circles and other geometrical figures, without whose help it is impossible to comprehend a single word of it; without which one wanders in vain through a dark labyrinth.

— Galileo Galilee (Galilei 1623), translation by Thomas Salusbury (1661) found in (Burt 2003)

4

From data to symbols

Contents

4.1	Introduction data to symbols	103
4.1.1	Learning as a search	104
4.2	Learning symbolic representations	105
4.2.1	Publication: (Herzog et al. 2021a)	106

4.1 Introduction data to symbols

In the last two chapters, approaches were shown that can support the modelling of non-linear systems. In particular, we were able to demonstrate that the combination of an CAE with a CRF makes it possible to very well approximate even rather complex non-linear dynamics. Such a hybrid network allows predicting the evolution of systems that are otherwise difficult or impossible to model. However, there is also a serious drawback because these systems are *black-box* systems, they process input in a way that is difficult for us humans to comprehend and thus do not allow any physical insight into the system under investigation. Furthermore, it does also not give us symbolic representations with which we, humans, could then formulate - for example by ways of an explicit mathematical model - the underlying

mechanisms behind these systems. This, however, would be very desirable, especially with regard to theory building of systems that are not covered by first principles. Previously ANNs were considered approximators of functions, and strictly speaking they will remain so, but these approximator methods can now be used to also engage in a search for symbolic representations.

4.1.1 Learning as a search

Tom Mitchell in 1982 had described learning from examples as search (Mitchell 1982). Examples are statements assigned to a category (classified) in the learning literature. By contrast, observations are not classified. Thus, in learning from examples, the assignment (aggregation) has already been done by the user or another system. In learning from observations, aggregation is part of the learning task. The task of learning from examples, according to (Mitchell 1982) can be considered as:

1. Description language LE for examples.
2. Definition language LH for the expression.
3. Set P of positive examples (examples for the expression).
4. Set N of negative examples (non-examples for the expression).
5. Matching predicate that classifies examples (*covers*).

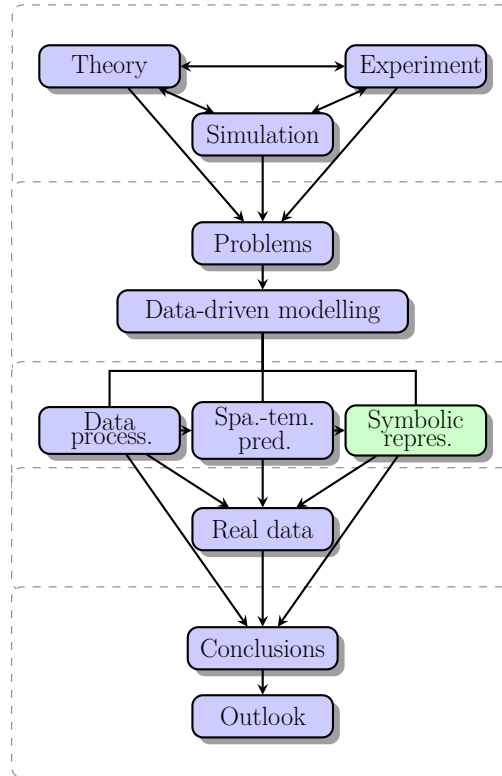


Figure 4.1: Structure overview: From data to symbols chapter 4

The goal is to find $c \in LH$ with $\forall p \in P$, $covers(c, p)$ is true and $\forall n \in N$, $covers(n, p)$ is false. The search or hypothesis space for expressions is the set of all expressions that can be formed with the help of LH . These are all possible characterisations for which it must then be determined whether they cover all positive examples and none of the negative ones. The matching predicate can be realised, for example, by the logical conclusion: $covers(c; e)$ exactly when $c \models e$. The simplest learning algorithm is, thus, the enumeration algorithm: it enumerates all expressions that can be formed in LH (hypothesis generation) and checks for each which examples (and non-examples) are covered (hypothesis test). Once the target condition holds, the algorithm stops.

4.2 Learning symbolic representations

Finding system equations from data is an old problem, perhaps the simplest case being linear regression. Where a linear equation is adjusted by optimising some parameter variables until the error to the data is minimal. The concept of *system identification* (SID) goes further than the simple case of linear regression, since for linear regression the general structure of the function is already given as $f(x) = mx + b$. Such approaches only make sense if one already has a good understanding of the system to be modelled (as a straight line in this case). In the case of SID, however, the goal is more ambitious since one wants to provide as little information as possible about the structure of the equations to find the symbolic representation of the underlying dynamics. A very interesting and modern approach has been demonstrated by AlMomani et al. (2020), where the *Shannon entropy* was utilised for a sparse regression to find equations for several non-linear systems. The approach of (AlMomani et al. 2020) tries to find a linear representation of non-linear basis functions, which reproduce the data with a minimum error and where as many coefficients as possible are zero. A similar parameterisation approach is also attempted by so-called *neural ordinary differential equations* (Chen et al. 2018). These authors describe a neural network as a sequence of complicated transformations that are achieved by a composition of transformations into a hidden

state. In analogy to equation A.3, the authors of (Chen et al. 2018) define the sequence of transformations as

$$\mathbf{h}_{t+1} = f(\mathbf{h}_t, \theta_t)$$

with $t \in \{0 \dots T\}$ and some $\mathbf{h}_t \in \mathbb{R}^D$. These iterative updates are seen as *Euler discretization* of a continuous transformation by Chen et al. (2018). If the ANN has residual connections, equation 4.2 becomes

$$\mathbf{h}_{t+1} = \mathbf{h}_t + f(\mathbf{h}_t, \theta_t).$$

Based on equation 4.2, the authors motivate the following formulation

Definition 4.2.1 (Neural ordinary differential equations from (Chen et al. 2018)).

$$\frac{d\mathbf{h}(t)}{dt} = f(\mathbf{h}(t), t, \theta_t), \quad (4.1)$$

where the input layer is given by $\mathbf{h}(0)$ and the output layer by $\mathbf{h}(T)$. Starting from the input layer, the output can be calculated by solving the initial value problem.

Considering *Eulers method* as a discretisation of the continuous relationship between the input and output domains of the data. ANNs can be considered in a similar fashion, since ANNs also discretise the continuous relationship using their hidden states in their latent space.

4.2.1 Publication: (Herzog et al. 2021a)

Motivated by the work in (Chen et al. 2018), the idea behind the next publication was to see if exactly this approach is suitable to train an ANN according to equation 4.1 in such a way that it describes the data by a *Hamiltonian* system. For this, the approach of Chen et al. (2018) was slightly modified as described in Herzog, S. and Wörgötter, F. (2021a). “Application of neural ordinary differential equations to the prediction of multi-agent systems”. accepted for SWARM 2021 (to be considered for full publication in Artificial Life and Robotics). With the goal to model a vector field that describes the swarm motion of interacting agents.

Application of neural ordinary differential equations to the prediction of multi-agent systems

Sebastian Herzog¹ and Florentin Wörgötter¹

Third Institute of Physics - Biophysics, Department for Computational Neuroscience,
University of Göttingen, Göttingen, Germany
sherzog3@gwdg.de

Abstract. Dynamic systems are usually described by differential equations, but formulating these equations requires a high level of expertise and a detailed understanding of the observed system to be modelled. In this work we present a data driven approach, which tries to find a parameterization of neural differential equations system to describe the underlying dynamic of the observed data. The presented method is applied to a multi-agent system with thousand agents.

Keywords: Artificial neural networks, neural ordinary differential equations, data-driven modelling, hamiltonian

1 Introduction

We present an approach to learn the dynamics of hundreds to thousands of agents in a system, where the underlying assumption is that the agents follow a collective dynamic, which is not obviously recognisable, but present. Examples for such dynamics systems are flocks of birds or the movement of cell components. The underlying dynamics allow many conclusions, e.g. about the energy distribution in the system or whether there are symmetries or not.

To visualise the motion of many particles or of a swarm with many agents several approaches exist, like the group of optical flow methods [8, 9, 12], cross correlation [1] and Kalman filters [19], these methods can be translated into more complex approaches like particle tracking velocimetry (PTV) [5, 14, 15, 20]. If a sufficient amount of data is available and agent/particle density is high enough, a vector field can be obtained by mapping the trajectories on a Cartesian grid. Thus, it is possible to visualise the dynamics of the underlying system. However, these approaches often result in an algorithmic frameworks of high complexity. The approach presented here is based on neural ordinary differential equations (nODEs) [3]. An artificial neural network (ANN) is used to approximate the Hamiltonian of canonical classical mechanic equations. This allows describing the underlying particle- or agent-swarm directly as a dynamic system without detours. To train the here-presented approach only a list of positions with their momentum is necessary. In this study we derive the method for this and apply it to simulated data based on the boids [18] algorithm.

2 Methods

Considering a set of N coordinates (\mathbf{q}, \mathbf{p}) , where $\mathbf{q}, \in \mathbb{R}^N$ represents the positions and $\mathbf{p} \in \mathbb{R}^N$ their momentum. A scalar function $\mathcal{H}(\mathbf{q}, \mathbf{p})$ is called the Hamiltonian if:

$$\dot{\mathbf{q}} = \frac{d\mathbf{q}}{dt} = \frac{\partial \mathcal{H}}{\partial \mathbf{p}} \text{ and } \dot{\mathbf{p}} = \frac{d\mathbf{p}}{dt} = -\frac{\partial \mathcal{H}}{\partial \mathbf{q}}. \quad (1)$$

this system can be rephrased without restricting generality to

$$\dot{\mathbf{q}} = \nabla_{\mathbf{p}} \mathcal{H}(\mathbf{q}, \mathbf{p}) \text{ and } \dot{\mathbf{p}} = -\nabla_{\mathbf{q}} \mathcal{H}(\mathbf{q}, \mathbf{p}), \quad (2)$$

where ∇ is the gradient operator. For nODEs it is common to approximate the right-hand side of (Eq. 2) by ANNs, denoted by $\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})$, where θ is a vector with all parameters of the ANN. A (feedforward) artificial neural network, also known as a multi-layer perceptron (MLP), is a series of logistic regression models stacked on top of each other. MLPs can be used for classification problems as well as for regression problems, depending on the final layer. In this work we are solving regression problems defined by:

$$p(y|\mathbf{x}, \theta) = \mathcal{N}(y|\mathbf{w}^T \mathbf{z}(\mathbf{x}), \sigma^2) \quad (3)$$

$$\mathbf{z}(\mathbf{x}) = q(\mathbf{V}\mathbf{x}) = [g(\mathbf{v}_1^T \mathbf{x}), \dots, g(\mathbf{v}_H^T \mathbf{x})], \quad (4)$$

where y is the desired regression output, \mathbf{x} the input vector, \mathbf{w} the weight matrix, $\mathbf{v}_j = \mathbf{V}_{j,:}$ is the j th row of \mathbf{V} , hidden layers $\mathbf{z}(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{V})$ of size H , where g is the so called activation function. $\mathcal{N}(\cdot, \sigma^2)$ is the symbol for the normal distribution with variance σ^2 . The hyperbolic tangent (tanh) is used as the activation function. To use (Eq. 3) for $\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})$ the framework of nODEs is used, which are a family of ANNs where the hidden layers parameterize the derivative of the hidden state using a neural network and the output of the network is computed using a black-box differential equation solver, where backpropagation [2, 4, 10] through the ODE solver is applied [3]. A special feature of nODEs is that the gradient is calculated using the adjointed sensitivity method [17], allowing a linear relation between problem size and computational complexity and a low memory consumption. After obtaining the gradient the parameters of $\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})$ are trained by a gradient method with adaptive moment estimation, called ADAM [11]. The input data are all pairs of (\mathbf{q}, \mathbf{p}) in the training-set. And the loss function which is supposed to be optimised is

$$l_{\theta} = \|\dot{\mathbf{q}} - \nabla_{\mathbf{p}} \mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})\|_2^2 + \|\dot{\mathbf{p}} - \nabla_{\mathbf{q}} \mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})\|_2^2 + \alpha \|\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})\|_1, \quad (5)$$

where the first two terms simply describe the quadratic deviation, $\|\cdot\|_2^2$ denotes the l_2 -norm and the third term $\alpha \|\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})\|_1$, with $\|\cdot\|_1$ denotes the l_1 -norm and forces \mathcal{H} to be sparse with $\alpha \in \mathbb{R}$ a weighting parameter (in this work $\alpha = 0.3$). The implementation of $\mathcal{H}_{\theta}(\mathbf{q}, \mathbf{p})$ was done with pytorch [16] and is based on [3]. The network architecture consists of four dense layers, each layer has an activation layer with tanh. The first layer expects 2 inputs and has 128 nodes, followed by two more layers with 128 nodes and finally a layer with 128

nodes and one output node. In this work we have deliberately avoided dividing the data into a training, validation and test set because we are interested in the vector field in the end, which is not directly evident from the data anyhow. In the following results part two points are examined. Firstly, the learned vector field is validated if it can reproduce the ground truth trajectories from the boid algorithm, when using the initial conditions from the boid algorithm. Secondly, the trained system is used on new data, but with the same parameters for the random sources.

3 Results

To investigate the presented approach empirically a simulation of the flocking behaviour of birds is used. For this simulation 1000 agents with separation, alignment and cohesion rules were modelled by the boids algorithm [18]. The boids algorithm with reflecting boundaries was implemented using python [21] and the numpy package [7]. The 1000 agents were placed randomly centred around $(0, 0)$ (Fig. 1) but without overlap and with random initial velocities from $\mathcal{U}[0.2, 5]$ and random accelerations of $\mathcal{U}[0, 0.03]$, where $\mathcal{U}[a, b]$ is the uniform distribution between a and b . The system was simulated $t_{max} = 5000$ time steps. These data were then used to train the presented approach with early

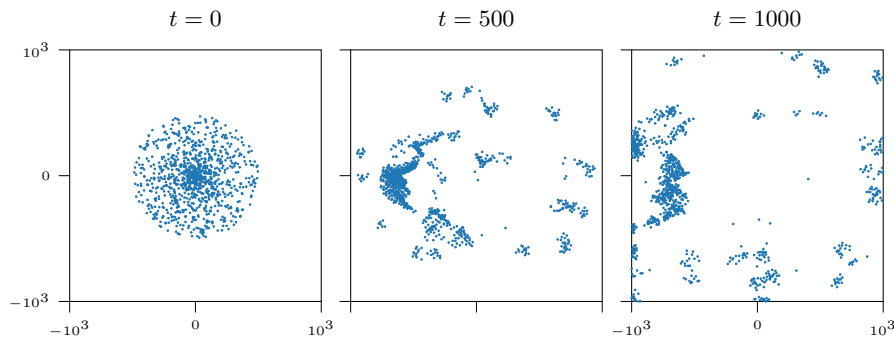


Fig. 1. Snapshots of 1000 agent system simulated with the boids algorithm [18] at different times (from left to right: $t = 0$, $t = 500$, $t = 1000$). Each dot represents an agent and the borders are reflective.

stopping [6] and a patience of 15. In (Fig 2) the values of the loss function over the epochs are plotted. Early stopping ends the training if the loss function does not improve over the last 15 epochs. The system was trained on a Nvidia GTX 1080, an epoch calculates on average 61 ± 3 seconds. In total 54 minutes 54 seconds, but probably half of the time would also be sufficient. After training the vector field of the nODE can be visualised, by generating an equidistant

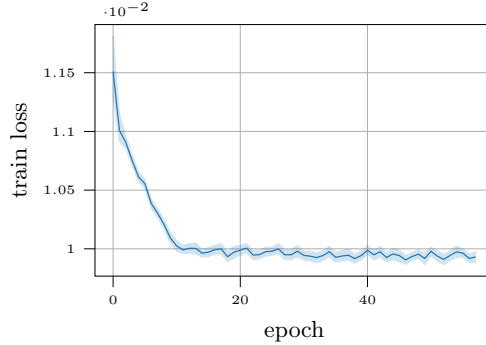


Fig. 2. Development of the train loss. Early stopping, with a patience of 15 stopped the training after 57 epochs. It can be seen that the training started to oscillate from around epoch 30.

grid with $n = 50$ points and applying the learned $\mathcal{H}_\theta(\cdot, \mathbf{0})$ to each point. The resulting vector field is illustrated in (Fig. 3). Based on the learned vector field

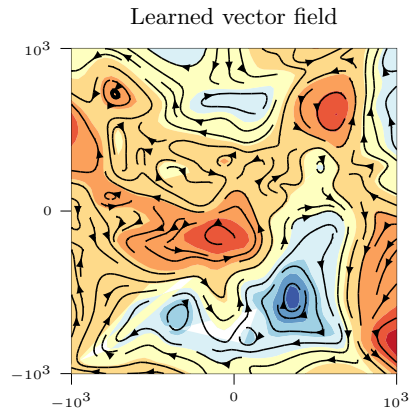


Fig. 3. Visualisation of the learned vector field. The colours (blue to red) are an indication of the energy in the system.

it's possible to predict trajectories, by using some values for \mathbf{q} and \mathbf{p} as initial conditions and applying this to the field. To check if the presented approach learned the underlying dynamic from the boid algorithm, all \mathbf{q} and \mathbf{p} from the training data were selected. A few exemplary trajectories are shown in (Fig 4). The corresponding statistics are available in blue in (Fig 5). It can be seen (Fig. 5) that the trajectories deviate about 7 units of length in the median, but there are also cases where the deviate exceeds 80 units, with a domain size of

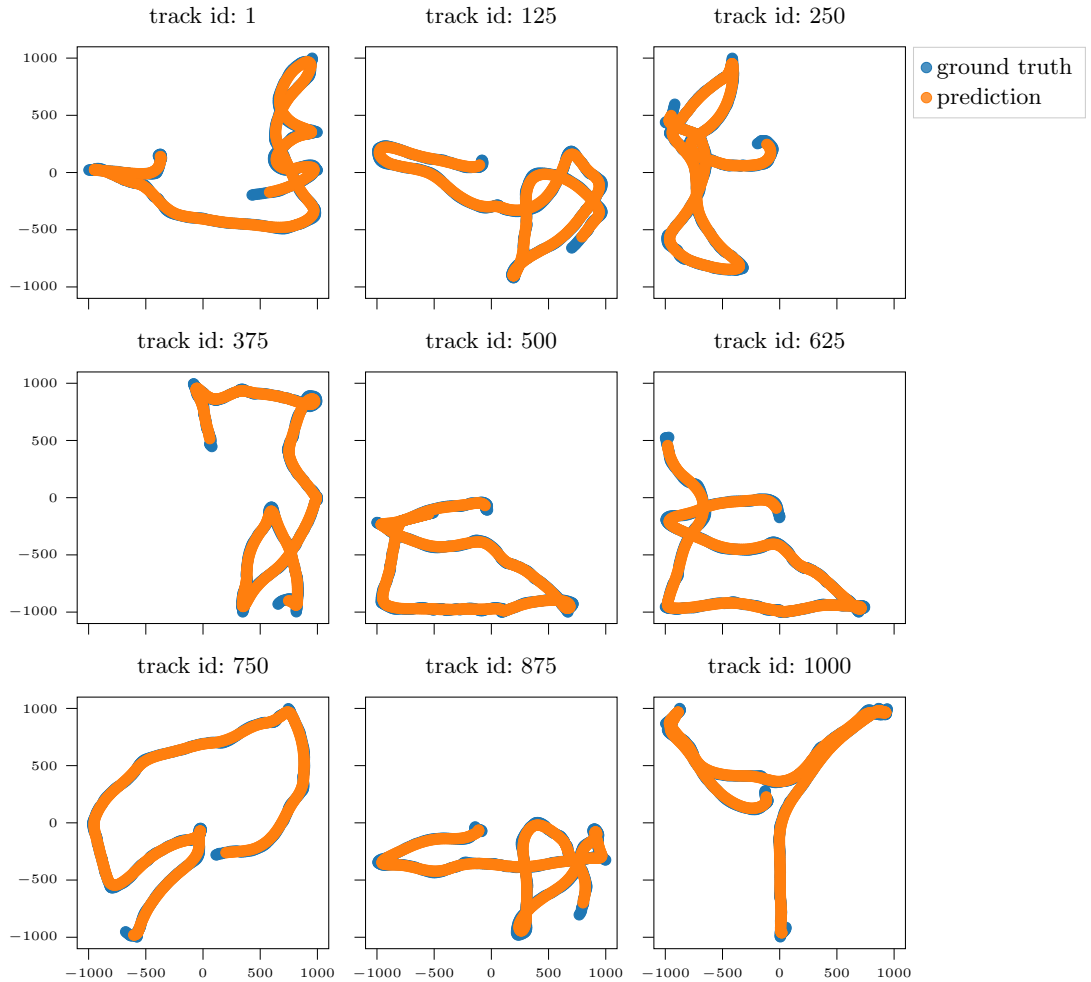


Fig. 4. Selection of trajectories for visualization purposes. The ground truth trajectories are coloured in blue, while the trajectories generate from the nODE are coloured in orange.

$10^3 \times 10^3$ this is a comparably small error. To validate our approach further, we took the predicted nODE and randomly initialised 1000 agents with the same scheme as above and calculated the trajectories based on nODE over 5000 time steps. For each trajectory the nearest neighbour of the original trajectories was searched and the l_2 -deviation was calculated. This deviations can be seen in Fig. 5 in orange. The expectation of the distribution is about 80 length units, which in relation to the total domain size again is not much.

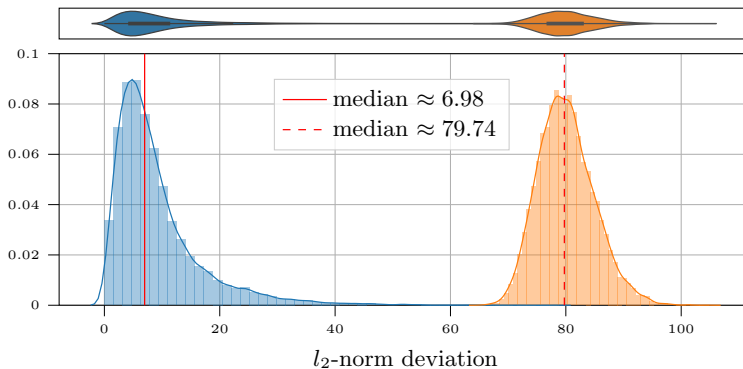


Fig. 5. In blue: Statistical error distribution of the trained nODE, based on the l_2 -deviation to the original trajectories from the boid algorithm. In orange: Statistical error distribution of the trained nODE with 1000 newly initiated agents. The deviation was calculated to the nearest neighbour to the original trajectories from the training data.

4 Discussion

The approach presented here is an extension of the classic nODE approach. By adding a sparse regularisation term to (Eq. 5), it was possible to train the nODE network to learn the underlying dynamic of the agent motions from the boids algorithm. Surprisingly, even the resulting trajectories from the learned system only have a maximum deviation of 7 length units (divided by the domain width this equals 0.7%) compared to the ground truth data, which means that the underlying dynamic of the training data can be reconstructed. Considering randomly initialised agents, from the same source distribution, the deviation increases in the median to up to 80 length units (or 8%) which is an increase of a factor of 10, but given the large domain this is still quite a good result. It also should be noted that no hyperparameter optimisation was considered for the approach presented at this stage. It is therefore to be expected that, if the loss function or the architecture of the network is further adapted, even better results could be achieved. It is also worth mentioning that we have chosen ADAM [11] because of its fast convergence, other algorithms such as SGD [13] may achieve better results, too. Methods like these need further investigation to understand in detail in which cases they generalise and in which not. However, we hope that approaches like the presented one can be helpful to gain new insights into systems where models based on first principles are hard to obtain.

References

1. Ronald Newbold Bracewell and Ronald N Bracewell. *The Fourier transform and its applications*, volume 31999. McGraw-Hill New York, 1986.

2. A. E. Bryson. A gradient method for optimizing multi-stage allocation processes. In *Proc. Harvard Univ. Symposium on digital computers and their applications*, 1961.
3. Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems, NIPS'18*, page 6572–6583, Red Hook, NY, USA, 2018. Curran Associates Inc.
4. S.E. Dreyfus. The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*, 5:30–45, 08 1962.
5. T. Fuchs, R. Hain, and C. J. Kähler. Non-iterative double-frame 2D/3D particle tracking velocimetry. *Exp. Fluids*, 58:119, 2017.
6. Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, MA, USA, 2016. <http://www.deeplearningbook.org>.
7. Charles R. Harris, K. Jarrod Millman, St'efan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fern'andez del R'io, Mark Wiebe, Pearu Peterson, Pierre G'erard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.
8. Berthold K.P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1):185 – 203, 1981.
9. J Yu Jason, Adam W Harley, and Konstantinos G Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conference on Computer Vision*, pages 3–10. Springer, 2016.
10. Henry J. Kelley. Gradient theory of optimal flight paths. *ARS Journal*, 30(10):947–954, 1960.
11. Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.
12. Till Kroeger, Radu Timofte, Dengxin Dai, and Luc Van Gool. Fast optical flow using dense inverse search. In *European Conference on Computer Vision*, pages 471–488. Springer, 2016.
13. Yann A. LeCun, Léon Bottou, Genevieve B. Orr, and Klaus-Robert Müller. *Efficient BackProp*, pages 9–48. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
14. Gruen A. Maas H.G. and Papantoniou D. Particle tracking velocimetry in three-dimensional flows. *Exp. Fluids*, 15(2):133–146, 1993.
15. Koichi Nishino, Nobuhide Kasagi, and Masaru Hirata. Three-dimensional particle tracking velocimetry based on automated digital image processing. 1989.
16. Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.
17. Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. Routledge, 2018.
18. Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive*

- Techniques*, SIGGRAPH '87, page 25–34, New York, NY, USA, 1987. Association for Computing Machinery.
19. Jinya Su, Baibing Li, and Wen-Hua Chen. On existence, optimality and asymptotic stability of the kalman filter with partially observed inputs. *Automatica*, 53:149 – 154, 2015.
 20. Y. Tanida and H. Miyashiro. *Flow Visualization VI, Chapter: 3D Particle Tracking Velocimetry-Its Possibilities and Limitations*. Springer-Verlag, 1992.
 21. Guido Van Rossum and Fred L. Drake. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA, 2009.

4.2.1.1 Conclusion from (Herzog et al. 2021a)

Considering *(Herzog et al. 2021a, figure 4), one can see that the predicted vector field itself reflects the individual movement of the agents well, which is interesting since the data for the training is not from a system of equations but from an algorithm describing a set of rules. This can be taken as an indication of how descriptive ODEs can be if they are parameterised appropriately. However, this was only a toy example with now physical meaning, but with the purpose to show that even for a case where the best description of the agent dynamics is an algorithm a simpler symbolic representation can approximate the dynamics as well. More important, as the two works (AlMomani et al. 2020; Chen et al. 2018) and our own work *(Herzog et al. 2021a) exemplify, there are several approaches to find a symbolic representation for the data and, depending on what kind of symbols to use, both approaches have their advantages. These methods differ from each other around their core, but they have a similar idea. Based on some user specifications, a rough structure which symbols should be used to represent the data is given. In the case of (Champion et al. 2019) it can be e.g. trigonometric functions, polynomials, or monomials, in the case described by us arbitrary ODEs. The process of learning is then supposed to find possible parameters so that the data is represented as accurately as possible by the user defined structure.

In the next chapter, the approach from *(Herzog et al. 2021a) will be applied to real data, in contrast to the data used in *(Herzog et al. 2021a), the experimental data have a more stochastic nature and do not follow smooth trajectories.

Insofern sich die Sätze der Mathematik auf die Wirklichkeit beziehen, sind sie nicht sicher, und insofern sie sicher sind, beziehen sie sich nicht auf die Wirklichkeit.

— Albert Einstein, found (Einstein 1921)

5

Application to experimental data

Contents

5.1	Introduction to experimental data application	117
5.2	Publication: (Herzog et al. 2021b)	118
5.3	Conclusions from (Herzog et al. 2021b)	137
	5.3.1 Application of the presented methods	137
5.4	Summary	142

5.1 Introduction to experimental data application

In the last three chapters, methods were presented that support the modelling of systems. We had covered the aspect of data processing (chapter 2), demonstrating the potential of CAEs to complete and enhance data. Followed by the first hybrid model (chapter 3), a combination of CAE and CRF resulting in a method capable of learning the spatio-temporal non-linear dynamics from observed data and had ended so far with partially addressing the problem of finding of symbolic representations (chapter 4). Up to this point, these methods had been used on simulated data, which has the advantage that a ground truth exists, which is good for validation purposes, but on the other hand the data

are of a quality that is often not matched by real experimental data. In this penultimate chapter, the application of the presented methods from chapters 3 and 4 on real experimental data will be discussed and illustrated. The presented methods are supposed to be used in the field of particle tracking, which is of great relevance in many areas of physics but also in biology. The goal of particle tracking is that, based on data mostly in the form of images (e.g. from cameras), particles are extracted per time step and these particles are then assembled into trajectories in order to record the movement of the particles.

5.2 Publication: (Herzog et al. 2021b)

The framework from the following publication is a prerequisite to allow training of the methods (CAE+CRF and nODE) presented in *(Herzog et al. 2021a; Herzog et al. 2019) and later for comparing the results. In this sense this paper is a bit "outside" the core of this thesis but - clearly - it is needed. In the next publication, which will be presented in Herzog, S., Schiepel, D., Guido, I., and Wagner, C. (2021b). "A probabilistic particle tracking framework for guided and Brownian motion systems with high particle densities". submitted to Int J Comput Vis, a particle tracking method will be presented that will be able to reconstruct two-dimensional trajectories as well as three-dimensional trajectories from particle images. One of the main goals in the development of the method was that it should be able to handle even very high particle densities.

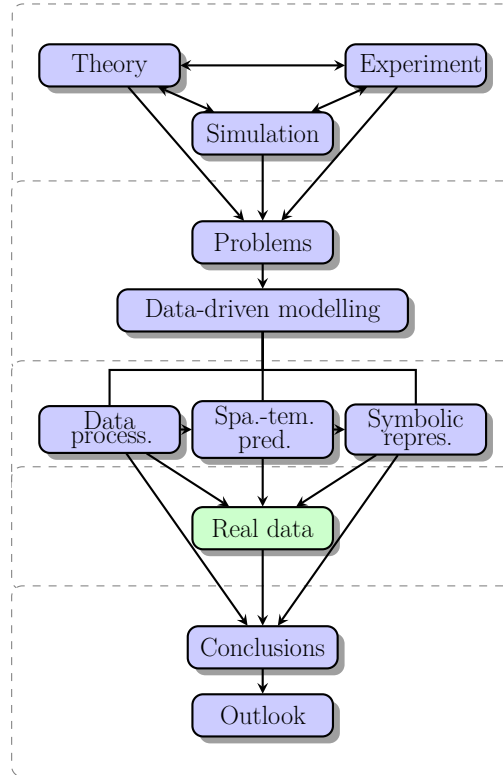


Figure 5.1: Structure overview: Application to experimental data chapter 5

A probabilistic particle tracking framework for guided and Brownian motion systems with high particle densities

Sebastian Herzog^{1,2,3} · Daniel Schiepel¹ · Isabella Guido² · Claus Wagner^{1,4}

Received: date / Accepted: date

Abstract This paper presents new framework for particle tracking based on a Gaussian Mixture Model (GMM). It is an extension of the state of the art iterative reconstruction of individual particles by a continuous modeling of the particle trajectories considering the position and velocity as coupled quantities. The proposed approach includes with an initialization and a processing step. In the first step, the velocities at the initial points are determined after iterative reconstruction of individual particles of the first four images in order to be able to generate the tracks between these initial points. From there on, the tracks are extended in the processing step by searching for and including new points obtained from consecutive images based on continuous modeling of the particle trajectories with a Gaussian mixture model.

The presented tracking procedure allows to extend existing trajectories interactively with low computing effort and to store them in a compact representation using little memory space. To demonstrate the performance and the functionality of this new particle tracking approach, it is successfully applied to a synthetic turbulent pipe flow, to the problem of observing particles correspond to a Brownian motion (e.g.: motion of cells), as well as to problems where the motion is guided by boundary forces, e.g., in the case of particle tracking velocimetry of turbulent Rayleigh-Bénard convection.

Keywords Brownian motion · Turbulent flow · Particle tracking · Gaussian mixture model

1 Introduction

In many areas of science, theories and experiments are in constant interaction. Theories are based on experimental designs and the insights gained with experiments can change theories or create new ones. However, quantitative key indicators cannot be directly derived from any experiment. For the quantitative analysis of different dynamic processes from sequential image data particle tracking is important. To detect and follow large numbers of individual particles, many automated computational methods have been developed [23,9,13,17,19,24,30,26]. Generally speaking, these methods rely on two steps: particle detection where spatial information is extracted based on spots that protrude from the background, using certain criteria to identify the particle and its position in every frame of the image sequence. The second step is so-called particle linking by where the temporal relation between the particles from frame to frame is determined using another set of criteria to form tracks. These two components result in different approaches to visualise the movements, like optical flow methods [4], Particle Image Velocimetry[3] (PIV) and Particle Tracking Velocimetry (PTV) [24,30]. While particle detection is highly task specific - the linking procedure has a more general character.. This work presents an approach that can be applied to many different cases, as it describes the movement of a particle using a probabilistic model based on a modified Gaussian Mixture Model (GMM) [27]. The main focus of this method is: (i) to model trajectories with coupled dimensions (i.e.: position and velocity) where changing one dimension leads to changes in the other dimensions, resulting in a high prediction accuracy, (ii) the ability to change the start/end point without a total recalculation of the whole track allowing high computational efficiency, (iii) creating a compact representation of the determined tracks requiring less memory space and (iv) to be able to model periodic

¹German Aerospace Center, Institute for Aerodynamics and Flow Technology, Bunsenstr. 10, 37073 Göttingen · ²Max Planck Institute for Dynamics and Self-Organization, Am Fassberg 17, 37077 Göttingen · ³University of Göttingen, Third Institute of Physics - Biophysics, Department for Computational Neuroscience, Friedrich-Hund-Platz 1, 37077 Göttingen · ⁴Technische Universität Ilmenau, Institute for Thermodynamics and Fluid Mechanics, Helmholtzring 1, 98693 Ilmenau

trends in the particle motion. After validating the approach in a synthetic case providing a ground truth for the quantitative comparison, the functionality is demonstrated dealing by means of two different applications. The first application is the tracking of cell migration. The capacity of cells to migrate can be considered as a fundamental mechanism that is physiologically essential for biological processes, which include embryonic development, immune response, wound healing and spread of pathological conditions like cancer [34]. More general cells are able to sense external stimuli of different nature, such as chemical gradients [36], electric fields [20, 10], substrate stiffness [14] and to respond with a directed movement toward or away from the stimulus source depending on the reaction transduced by the cells. Analyzing the dynamics of cells during migration by following them over time provides information about trajectories and velocities that allow the characterization of fundamental cellular mechanisms. Cell-imaging is the most appropriate approach to get access to cell behaviour and the automation of microscopes has enabled the acquisition of time-lapse images with fluorescence or bright-field microscopy of big samples of cells at single-cell resolution [7]. However, image analysis is a bottleneck for biological advances as there is a gap between the advanced techniques to collect data and the ability to analyze it. Manual analysis of big data sets is time-consuming. Additionally, it introduces bias of the operator and results that cannot be reproduced. This applies especially to the linking of single cells to tracks. Therefore, automated methods for cell tracking represent a vivid field of research as there is considerable interest in an efficient and accurate cell-tracking method that can overcome problems such as low signal-to noise ratio, poor staining, variable fluorescence in cells, low contrast, stain-free cell images, high cell density, deformable cell shapes [26], sudden changes in motion direction and speed and temporary drop out of focal plane [35]. The second application belongs to the domain of experimental fluid mechanics, where methods like PIV and PTV are the most prominent flow field measurement techniques yielding velocity vectors within observation planes or volumes. They have in common that the velocity vectors in a moving fluid are determined from the displacement of seeding particles transported by the flow during a prescribed time interval. For many years the main difference was that PIV [1] yielded two-dimensional (2D) velocity vectors in planes at particle image densities of $\approx 0.03 - 0.05$ particles per pixel (ppp), whereas PTV provided three-dimensional (3D) velocity vector fields in volumes based on particle triangulation and nearest-neighbor searches [17] for one magnitude lower tracer particles densities. The game changer for PIV was the introduction of tomographic reconstruction of particle distributions followed by three-dimensional cross-correlation [8] which were prerequisite for the time-resolved 3D tomographic PIV technique. In the latter, the particle po-

sitions are iteratively reconstructed as intensity peaks in a 3D Eulerian voxel space using algorithms like the multiplicative algebraic reconstruction technique (MART) or simultaneous MART (SMART) [12, 3]. This made tomographic PIV a reliable and robust 3D flow field measurement technique. Additionally, negative effects introduced by ghost particles are less significant than in PTV since the 3D correlation is performed after the reconstruction process. However, both reconstruction and cross-correlation are computationally expensive and the latter reduces the spatial resolution by relying on the mean displacement of particles within interrogation volumes. This is especially significant for measurements of turbulent flows since the cross-correlation filters out small-scale structures and high velocity gradients due to the inherent spatial averaging over interrogation volumes. Unlike in tomographic PIV, in 3D PTV the particle positions are first identified in a number (more than two) of 2D camera images and subsequently matched - typically by triangulation [11] - to obtain the 3D positions in the measurement volume for each time step in a Lagrangian reference systems. Subsequently, trajectories of individual particles are determined by matching particle positions of the successive time step [17, 23, 18]. Based on these trajectories the velocity and acceleration fields can be determined more precisely than in PIV since spatial averaging is not involved.

The presented tracking method shall be considered as an extension to existing frameworks. For the cell tracking case the input for the presented approach are segmented images from microscope images, where the segmentation can be achieved with the open source software ImageJ [32]. In the domain of Particle Tracking Velocimetry (PTV) the input images are achieved by the particle displacement per time interval typically prescribed by the pulse illumination frequency. This frequency can be increased using high-speed lasers and high-speed cameras for measurements of high-speed flows. However, until a few years ago the downside of the PTV approach used to be the limitation of the triangulation and matching process restricting the technique to low particle image densities, i.e. in the order 0.005 particles per pixel (ppp). To match particles to tracks, nearest-neighbor methods were used which could not cope with such dense (and possibly false) particle distributions in volumes. In order to relax the density limitations, approaches for tracking single particles like Enhanced Particle Tracking Velocimetry (EPTV) [21, 6] are multi-parametric as they consider the particle size or local particle concentration. [21] propose to improve the tracking procedure by proper pre-conditioning of the particle displacement which is realized in the time-resolved 3D tracking method of the commercially distributed "Shake-The-Box" (STB) software. With the latter, particle positions in subsequent images are predicted by extrapolating trajectories generated using former images with a third-order polynomial [30]. A prerequisite for the latter is the

combination of calibration methods like volume self-calibration [37] developed for tomographic PIV with iterative particle reconstruction (IPR) and image matching by shaking proposed in [38]. In contrast to the STB software, the here presented probabilistic particle tracking approach does not rely on a tomographic PIV evaluation for initialization or a Particle-Space Correlation as used for multi-pulse applications. [24].

2 Methods

The approach proposed below includes a triangulation step and a modeling step with a new method for tracking and predicting the motion of 3D particles. The triangulation is based on an iterative approach inspired by [38] to triangulate points in a three-dimensional volume, based on a number of two-dimensional images. The modeling step tracks and predicts the motion of a reconstructed 3D particle. With the proposed model any particle trajectory can be approximated by a GMM [27]. The outstanding feature of this method is that no initial 3D particle distribution is required. The presented method uses the *Soloff model* [33] as an optical transfer function (OTF), whose parameters are determined during a volume self-calibration [37]. In contrast to [38], the particle trajectories of any reconstructed particle are predicted using a probabilistic model based on GMMs [27] allowing to determine the local velocity and acceleration vectors as derivatives of basis functions. The procedure of the resulting framework is summarized below together with references to sections and equations which provide more details of the respective steps.

Algorithm 1: procedure of the framework

Data: Images
Result: Tracks
 initialization;
for $t = 1, 2$ **do**
 Image processing (sec. 2.1);
 Initial points \leftarrow Triangulate 3D points for t (sec. 2.2)
end
 Calculate initial velocities based on initial points (sec. 2.3);
 All tracks \leftarrow Link initial points based on initial velocities ;
 Predict all tracks to update all velocities (sec. 2.4);
for $i = 3, \dots, T_{max}$ **do**
 Image processing (sec. 2.1);
 Extend all tracks (Eq. 2.5);
 Remaining points \leftarrow Back projection of all extended
 Tracks and remove 2D points when back projection
 matches;
 New 3D points \leftarrow Triangulate 3D points for t (sec. 2.2);
 Calculate new velocities based on new 3D points and non
 extend 3D points (sec. 2.3);
 All tracks \leftarrow Link new points based on the new
 velocities;
end

2.1 Image processing

To reconstruct the three-dimensional particle distribution \mathcal{P}_t , where $t \in \mathbb{N}$ is the time, from the individual perspectives $c \in \mathbb{N}$, it is necessary to remove possible noise sources from the camera images. A camera image is considered as a set of pixels with intensities $I_{t,c} \in \mathbb{R}^{r \times c}$, where $r \in \mathbb{N}$ is the number of rows and $c \in \mathbb{N}$ the number of columns of the image. After processing the image $I_{t,c}$ the subpixel localization of imaged particles is calculated. By doing so, the individual pictures are reduced to a set of coordinates $\mathcal{C}_{t,c}$ and a set of intensities $\mathcal{I}_{t,c}$ in the following four steps:

1. **Masking:** For the masking step a predefined mask M_c is placed over the image $I_{t,c}$, such that $I'_{t,c} = M_c \cap I_{t,c}$ removing all irrelevant or physically meaningless parts of the images.
2. **Richardson–Lucy deconvolution:** To reverse blurring from the input images and to amplify the noise, the Richardson–Lucy deconvolution [28, 15] is used.
3. **Background subtraction:** In order to remove all parts of $I'_{t,c}$ which do not change in time a the minimum image $\min\{I'_{t,c}, I'_{t+1,c}, I'_{t+2,c}\}$ is calculated and subtracted pixel-wise such that: $I''_{t,c} = I'_{t,c} - \min\{I'_{t,c}, I'_{t+1,c}, I'_{t+2,c}\}$.
4. **Thresholding:** In the next step, all remaining intensities in the image below a certain threshold are removed. This is done by simply checking whether the intensity of a pixel on an image is above a threshold ϵ_{thres} , such that $I'''_{t,c} = \text{thres}(I''_{t,c}, \epsilon_{thres})$.
5. **Calculating sub-pixel particle localization:** In the last step, the clustered pixels which belong to one real particle on the camera images are reduced to coordinates by applying the analytic, non-iterative radial symmetry center method $\text{RSC}(\cdot)$ presented in [25], such that $\mathcal{C}_{t,c} = \text{RSC}(I'''_{t,c})$. The RSC method tries to fit a radial symmetry function on a cluster of pixels. The radial symmetry center is then the sub-pixel particle localization for the clustered pixels, where the corresponding intensity values $\mathcal{I}_{t,c}$ are integrated over the radial symmetry.

2.2 Triangulation

To reconstruct 3D points from $\mathcal{C}_{t,c}$ and $\mathcal{I}_{t,c}$ a camera calibration is required to provide intrinsic and extrinsic parameters in order to set up mapping function $F_c : \mathbb{R}^3 \rightarrow \mathbb{R}^2$, $F_c(X) = x_c$, where $X \in \mathbb{R}^3$ is the position in the 3D space and $x \in \mathbb{R}^2$ the position on some camera images with index c . The general case of triangulation deals with the problem of reconstructing 3D objects from a series of perspective shifted images in the real world. To compensate for different distortions occurring in experiments, we make use of the *Soloff model* [33] for volume self-calibration [37] to obtain the parameters for the mapping function F_c . Figure 1 illustrates the relations of a 3D point $X \in \mathbb{R}^3$ and its projection

on to the images from $\mathbb{R}^3 \rightarrow \mathbb{R}^2$, such that $L_1 : F_1(X) = x_1$ and $L_2 : F_2(X) = x_2$.

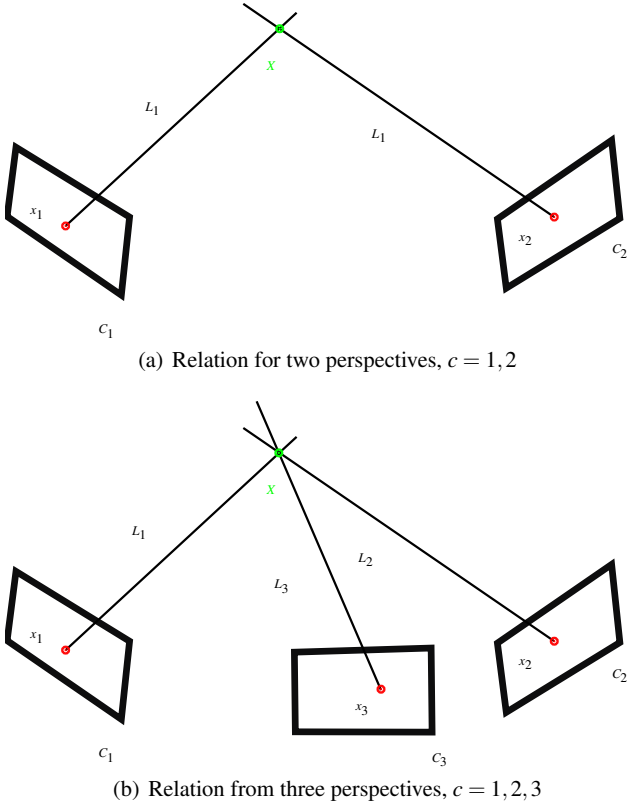


Fig. 1 Visualization of the projection lines for two different cases. The first one with two perspectives and the second one with three perspectives. The point X in 3D projected onto x_c lying in two and three images.

For the triangulation of each $x \in \mathcal{C}_{t,c}$ two points $Y, Y' \in \mathbb{R}^3$ are required, such that $F_c(Y) = F_c(Y') = x_c$. The straight line through Y and Y' defines the line L_c . However, to triangulate the desired point X , at least two projection lines $L_c, L_{c'}$ where $c \neq c'$ are needed. To construct these lines the correspondence between at least two $x_c \in \mathcal{C}_{t,c}$ is necessary. In order to determine this correspondence, the two points Y and Y' are projected onto another image, such that $F_{c'}(Y) = x_{c'}$ and $F_{c'}(Y') = x'_{c'}$. The line through $x_{c'}$ and $x'_{c'}$ on $I_{t,c'}$ is called the epipolar line, $E_{c \rightarrow c'}$. By calculating the Euclidean distance for all possible coordinate $x \in \mathcal{C}_{t,c'}$ to the line $E_{c \rightarrow c'}$, given by $d(E, x)$, all coordinates, where $d(E, x) < \epsilon_E$, are considered as possible candidates and ϵ_E is a control parameter. If the number of candidates is higher than 1, new epipolar lines are calculated for all possible candidates and are then projected onto the next image. In the next image, only the intersections of epipolar lines define as possible areas for corresponding coordinates. If more lines intersect in an area, the correspondence for a point is considered higher. Finally, the points with the minimal distance and highest number of

intersections are used for triangulation, as illustrated in figure 2. Remark: If several points have the same number of intersections and the same distance, all points are triangulated.

After the corresponding points on different images are found, it is possible to use these points for triangulation of the 3D point. Considering the associated projection lines L_c the analytical solution of this triangulation problem is the point of intersection of all L_c 's. If the lines are not intersecting $X_{est} \sim X$ is used where X_{est} is the point which minimizes the the sum of the distances $\sum_{\forall c} d(L_c, X)$ to approximate X .

With the above discussed approach, it is possible to triangulate any number of points X_{est_t} for any time step t . Further, the intensities I of X_{est} is obtained by using $F_c(X_{est}) = x_{est_c}$ for back projection into the images $I_{t,c}$. The nearest point in a ϵ_E space on $I_{t,c}$ is then considered as the intensity value for X_{est} , denoted by $I(X_{est})$.

2.2.1 Local optical transfer function

The local optical transfer function by Soloff [33] is a non-linear function $F_a : \mathbb{R}^3 \mapsto \mathbb{R}$:

$$F_a(X, Y, Z) := a_0 + \quad (1)$$

$$\begin{aligned} & X(X(a_9X + a_{11}Y + a_{14}Z + a_4) + a_{13}YZ + a_6Y + a_7Z + a_1) + \\ & Y(Y(a_{12}X + a_{10}Y + a_{15}Z + a_5) + a_8Z + a_2) + \\ & Z(Z(a_{17}X + a_{18}Y + a_{16}) + a_3), \quad (2) \end{aligned}$$

that maps a 3D point to one coordinate where (X, Y, Z) is the 3D position of the point, and a are the parameters of the Soloff model. By using two different parameter sets, i.e. a, b it is possible to match the 3D point to a 2D image position. Further, this function is used to generate the lines L_c . L_c is defined as the line by passing through two 3D points. To find this two points, the partial derivatives of F , which are:

$$\frac{\partial}{\partial X} F_a(X, Y, Z) = 3a_9x^2 + 2a_{11}xy + 2a_{14}xz + 2a_4x \quad (3)$$

$$+ a_{12}y^2 + a_{13}yz + a_6y + a_{17}z^2 + a_7z + a_1,$$

$$\frac{\partial}{\partial Y} F_a(X, Y, Z) = a_{11}x^2 + 2a_{12}xy + a_{13}xz + a_6x + 3a_{10}y^2$$

$$+ 2a_{15}yz + 2a_5y + a_{18}z^2 + a_8z + a_2,$$

$$\frac{\partial}{\partial Z} F_a(X, Y, Z) = 2z(a_{17}x + a_{18}y + a_{16})$$

$$+ x(a_{14}x + a_{13}y + a_7) + y(a_{15}y + a_8) + a_3$$

are used to solve the system:

$$\begin{pmatrix} \frac{\partial F_a}{\partial X} & \frac{\partial F_a}{\partial Y} & \frac{\partial F_a}{\partial Z} \\ \frac{\partial F_b}{\partial X} & \frac{\partial F_b}{\partial Y} & \frac{\partial F_b}{\partial Z} \end{pmatrix} \begin{pmatrix} X^* \\ Y^* \\ Z^* \end{pmatrix} = \begin{pmatrix} (F_a(X, Y, Z) - x) \\ (F_b(X, Y, Z) - y) \end{pmatrix}, \quad (4)$$

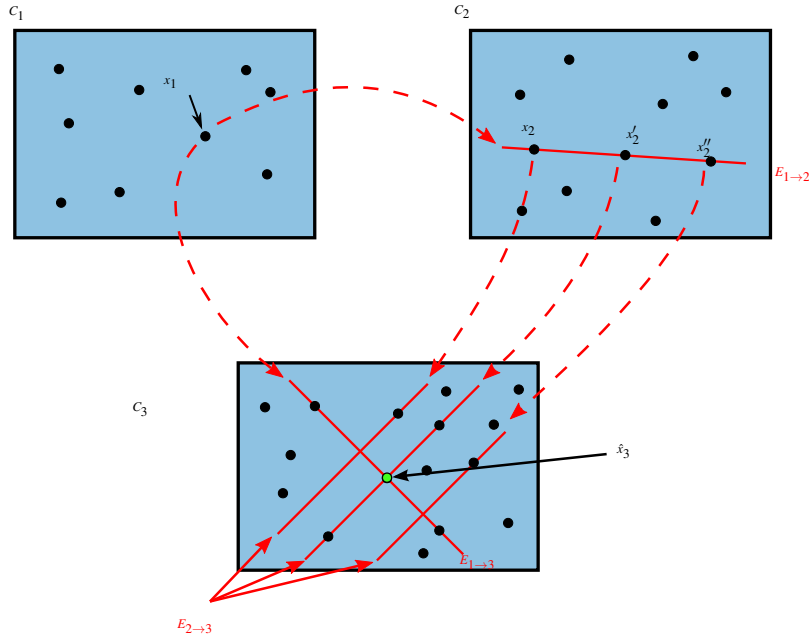


Fig. 2 Starting from C_1 , a point x_1 is selected for which the corresponding points must be found on other images. By calculating two 3D points Y, Y' , the epipolar line $E_{1 \rightarrow 2}$ in C_2 is obtained. Based on $E_{1 \rightarrow 2}$, three possible candidates need to be considered: x_2, x_2', x_2'' . For each of these points new epipolar lines are calculated and projected onto the next image C_3 together with the epipolar line from C_1 . This leads to three intersection points in C_3 , where the nearest point to one of this intersections is \hat{x}_3 . Going the way back the corresponding points for x_1 from C_1 are x_2' on C_2 and \hat{x}_3 on C_3 .

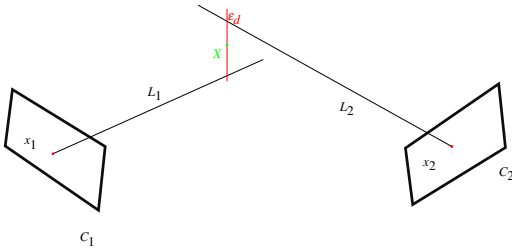


Fig. 3 Illustration of a misalignment of the projection lines L_c , where the distance ϵ_d is the maximum distance that is still accepted as the intersection point

where (x, y) is a selected point on an image and (X, Y, Z) are arbitrary start values for the algorithm. To determine the line L_c , the algorithm 2 must be executed twice for the same 2D point but with different initial values (X, Y, Z) . We suggest selecting the values for (X, Y, Z) so that they are on the opposite side of the object to be examined. L_c is the line between the two new 3D points.

2.3 Generation of initial tracks

The previously introduced triangulation is used to triangulate all possible points from the first two time steps t_0, t_1 . This serves as an initialization for the tracking approach

Algorithm 2: Determining two points on a line of sight

Data: $a, b, (x, y), (X, Y, Z)$

Result: $(X + h_x, Y + h_y, Z)$

$(X_i, Y_i, Z_i) = (X, Y, Z)$;

$\mathbf{f} = (f_1, f_2) = (\max \text{value}, \max \text{value})$;

while $\|\mathbf{f}\|_2 > \text{float precision}$ **do**

$$\mathbf{A} = \begin{pmatrix} \frac{\partial}{\partial X} F_a(X_i, Y_i, Z) & \frac{\partial}{\partial Y} F_a(X_i, Y_i, Z) & \frac{\partial}{\partial Z} F_a(X_i, Y_i, Z) \\ \frac{\partial}{\partial X} F_b(X_i, Y_i, Z) & \frac{\partial}{\partial Y} F_b(X_i, Y_i, Z) & \frac{\partial}{\partial Z} F_b(X_i, Y_i, Z) \end{pmatrix};$$

$$\mathbf{b} = \begin{pmatrix} (x - F_a(X_i, Y_i, Z)) \\ (y - F_b(X_i, Y_i, Z)) \end{pmatrix};$$

Solve $\mathbf{A}\mathbf{h} - \mathbf{b} = \mathbf{f}$;

Update $X_i = X_i + h_x$;

Update $Y_i = Y_i + h_y$;

end

described in section 2.4. In order to form tracks from the two point sets at t_0 and t_1 , it is necessary to first estimate velocities for the points at t_0 . To calculate the initial displacement, first the sets $\forall \omega_{t_0} \in \Omega_{t_0}$ and $\forall \omega_{t_1} \in \Omega_{t_1}$, where $\Omega_{t_0} = \{p_{t_0} \in \mathcal{P}_{t_0} : \|X_{est_{t_0}} - p_{t_0}\|_2 < \epsilon_{\Delta}/2\}$ and $\Omega_{t_1} = \{p \in \mathcal{P}_{t_1} : \|X_{est_{t_1}} - p_{t_1}\|_2 < \epsilon_{\Delta}/2\}$, are determined. Second, the Cartesian product $\Omega_{t_0} \times \Omega_{t_1}$ is used to calculate histograms for each displacement direction. The most frequent values in each histogram form $\Delta X_{est_{t_0}}$. Both steps are illustrated in figure 4. Two points are forming a track if the $\|(X_{est_{t_0}} + \Delta X_{est_{t_0}}) - X_{est_{t_1}}\|_2 \leq \epsilon_s$, if there are several candidates for

X_{est_t} that fulfill this requirement, the X_{est_t} with the smallest deviation is taken. ε_S and ε_Δ are user-defined parameters.

2.4 Probabilistic motion prediction

To track individual particles X in time t , three (one for each dimension) Gaussian mixture models (GMMs) [27] are used to approximate the trajectory of X . The aim of this approximation is to determine a function mapping based on the currently observed history for predicting the most likely particle motion. However, since the trajectories may perform a chaotic movement in a turbulent flow, we suggest using a probabilistic prediction based on the GMM instead of the extrapolation used in [38]. More precisely, we consider a GMM which can model both the position q_0, q_1, \dots, q_{n-1} and velocity $\dot{q}_0, \dot{q}_1, \dots, \dot{q}_{n-1}$ of the particle. Considering the density of a Gaussian mixture model [27] given by

$$p(x) = \sum_{k=1}^K \pi_k \mathcal{N}(x | \mu_k, \Sigma_k), \quad (5)$$

where x is the d -dimensional random variable, $\mathcal{N}(x | \mu_k, \Sigma_k)$ is a multivariate normal distribution, with mean μ_k and a covariance matrix Σ_k . The coefficients of Σ_k termed by π_k are the components of the density distribution $p(x)$, which have to satisfy $0 \leq \pi_k \leq 1$ and $\sum_{k=1}^K \pi_k = 1$. To derive the parameters for Eq. (5), the Expectation-Maximization (EM) algorithm [22] is used. It is a two-step iterative algorithm which identifies the maximum likelihood solution by computing the expectation step (E-step) of the log-likelihood evaluated with the current parameter estimation followed by the maximization step (M-step). This step estimates parameters that maximize the expected log-likelihood obtained with the E-step. Applying the above to the prediction leads to

$$p(x_f, x_h) = \sum_{k=1}^K \pi_k \mathcal{N}(x_h, x_f | \mu_k, \Sigma_k), \quad (6)$$

by inferring a joint Gaussian mixture distribution, where x_h is the history of the trajectory and x_f the future. The prediction is then performed by calculating the conditional mixture density

$$p(x_f, x_h) = \frac{p(x_h, x_f)}{\int p(x_h, x_f) dx_f} = \sum_{k=1}^k \hat{\pi}_k \mathcal{N}(x_f | x_h, \hat{\mu}_k, \hat{\Sigma}_k), \quad (7)$$

which is again a GMM, with the parameters

$$\hat{\pi}_k = \frac{\pi_k p(x_h | \mu_{k,x_h}, \Sigma_{k,x_h x_h})}{\sum_{j=1}^K \pi_j p(x_h | \mu_{j,x_h}, \Sigma_{j,x_h x_h})} \quad (8)$$

$$\hat{\mu}_k = \mu_{k,x_f} + \Sigma_{k,x_f x_h} \Sigma_{k,x_h x_h}^{-1} (x_h - \mu_{k,x_h}) \quad (9)$$

$$\hat{\Sigma}_k = \Sigma_{k,x_f x_f} - \Sigma_{k,x_f x_h} \Sigma_{k,x_h x_h}^{-1} \Sigma_{k,x_h x_f}, \quad (10)$$

where

$$\mu_k = \begin{bmatrix} \mu_{k,x_h} \\ \mu_{k,x_f} \end{bmatrix} \quad (11)$$

$$\Sigma_k = \begin{bmatrix} \Sigma_{k,x_h x_h} & \Sigma_{k,x_h x_f} \\ \Sigma_{k,x_f x_h} & \Sigma_{k,x_f x_f} \end{bmatrix} \quad (12)$$

is the partitioning of the means and covariance matrices of the GMM. Eq. (7) defines the future trajectories for which expectations can be evaluated, where the mean and covariance of a GMM is given by

$$\mu = \sum_{k=1}^K \pi_k \mu_k \quad (13)$$

$$\Sigma = \sum_{k=1}^K \pi_k (\Sigma_k + (\mu_k - \mu)(\mu_k - \mu)^T) \quad (14)$$

and the probabilistic prediction can be applied. However, to obtain a parametric representation of each trajectory a compact representation of a single trajectory is desired. In addition, the dimensions of the position and the velocity should be coupled. By defining $\phi_t = [\phi_t, \dot{\phi}_t] \in \mathbb{R}^{K \times 2}$ as the time-dependent basis matrix for q_t and \dot{q}_t and some noise $\varepsilon_{Noise} \sim \mathcal{N}(0, \Sigma_k)$, a trajectory τ can be defined as

$$\tau = \begin{bmatrix} q_t \\ \dot{q}_t \end{bmatrix} = \phi_t^T w + \varepsilon_{Noise}, \quad p(\tau | w) = \prod_k \mathcal{N}(x | \mu_k, \Sigma_k). \quad (15)$$

Eq.(15) represents a linear basis function model with weights $w \in \mathbb{R}^K$, where the basis functions for the position are defined by the product of normal distributions and the basis functions for the velocity are the time derivatives of the basis functions for the position. Using this representation only the weights w need to be stored for each trajectory. This process is illustrated in figure 5.

2.5 Extending trajectories and finding new trajectories

After the generation of the initial trajectories, these can be used to predict the position of the traced particles at the next time step $t+1$, denoted by X_{pred} , by adding the velocity at time t to the position at time t . The predicted position at $t+1$ is then projected back onto the images $I_{c,t+1}$ and evaluated in such way that all points $x_{I_{c,t+1}}$, but at least two points $x_{I_{c',t+1}}$ and $x_{I_{c'',t+1}}$, where $c \neq c'$, are found such that:

$$\left(\sum_c \|x_{I_{c,t+1}} - F_c(X_{pred})\|_2 * \left| 1 - \frac{I(x_{I_{c,t+1}})}{I(F_c(X_{pred}))} \right| \right) < \varepsilon_{back}, \quad (16)$$

where ε_{opti} is a user-defined parameter. In the future, other metrics shall also be investigated, like a higher order metric for evaluating multi-object tracking (HOTA)[16]. However, if such a point is found, the trajectory is extended to this

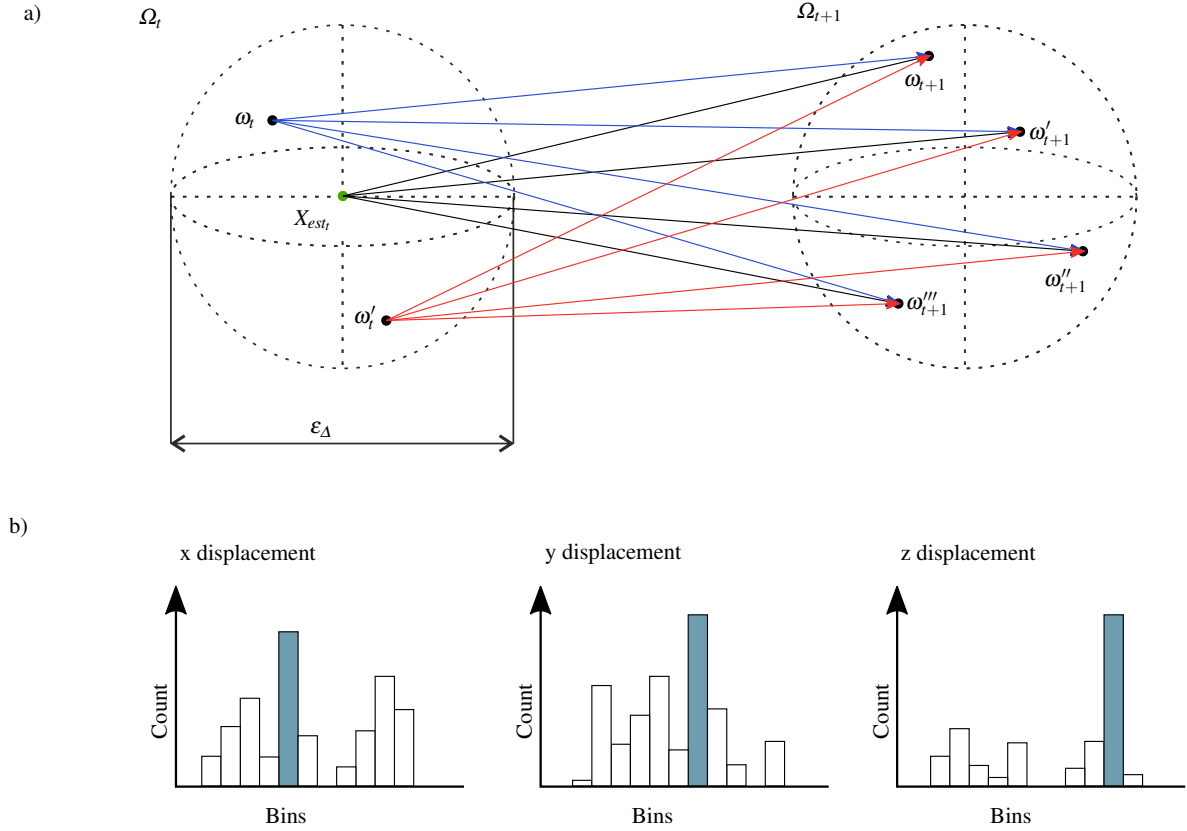


Fig. 4 a) Visualization of the Cartesian product $\Omega_t \times \Omega_{t+1}$, where $\Omega_t = \{\omega_t, X_{est_t}, \omega'_t\}$ and $\Omega_{t+1} = \{\omega_{t+1}, \omega'_{t+1}, \omega''_{t+1}, \omega'''_{t+1}\}$. The blue arrows starting from ω_t exemplify all possible displacements for ω_t , the same applies to for the black arrow starting from X_{est_t} and red arrow starting at ω'_t . Based on all this displacements the histograms in b) are set up. The bins with the highest counts (colored in cyan blue) in each histogram are the estimation for ΔX_{est_t} .

point. If this is not the case, time step $t + 1$ is skipped for the trajectories and the process is repeated for time step $t + 2, t + 3, \dots, t + gap$, where gap is a user-defined parameter. If still no matching is found the trajectory is considered to be terminated. To further minimize Eq. (16), X_{pred} , is varied by componentwise moving X_{pred} such that the projection of X_{pred} only changes by one pixel on $I_{c,t+1}$. The variation ends when Eq. (16) does not change anymore or a maximum number of variation steps is reached. Remaining points which are not part of any trajectory, are considered as positions of not yet identified or new trajectories and the above-described steps are repeated, until no more points remain or until a maximum number of iterations is reached. Remark: In general, it would be possible to formulate Eq. (16) as a multi-objective optimization so that the distance for the deviation in distance as well as the deviation of intensities are optimized independently, building a pareto front. This leads to the problem that the dimensions for the deviation in distance and the deviation of intensities need to be normalized. Instead of finding a meaningful and generally valid normalization for these independent physical deviations, we decided to use a formulation for a scalariza-

tion for the intensity deviation and to scale the deviation in distance by this scalar resulting in to a single-objective optimization problem Eq. (16).

3 Synthetic pipe flow

In order to validate and benchmark our framework, we set up a synthetic case of a generalized pipe flow. The simulated flow is used as a ground truth case so that we can directly compare our reconstruction results. First, in subsection 3.1 the generation of the synthetic particles is described, whereas in subsection 3.2 the validity of the results obtained from our framework is proven.

3.1 Particle generation

A synthetic case is defined with the aim to provide data sets describing the ground truth, based on which the number and accuracy of the generated trajectories can be evaluated. The case is designed to evaluate the capability of the

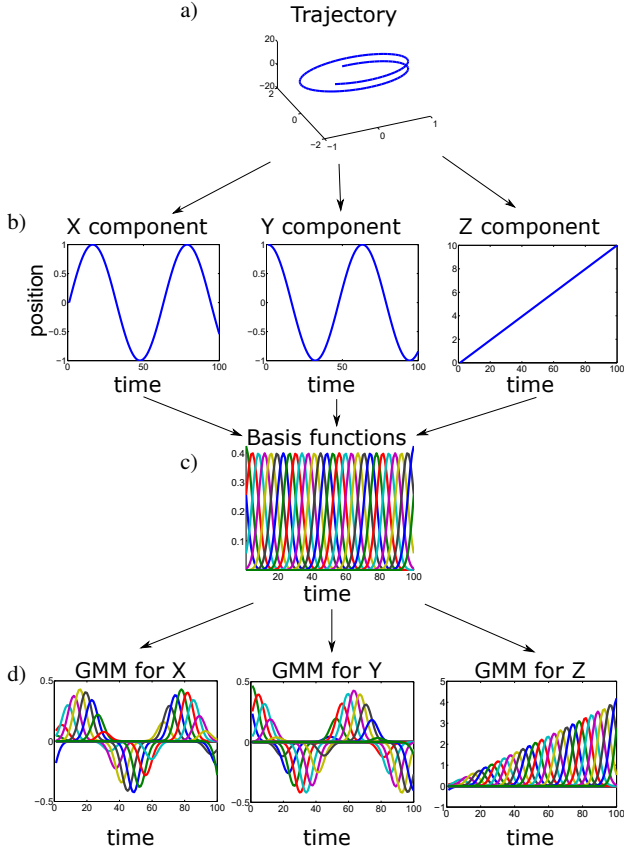


Fig. 5 Representation of a three-dimensional trajectory with three GMMs. The first line a) shows an example of a trajectory and the second line b) the x , y and z components of the trajectory. Line c) represents the set of basis functions from Eq. (5). By using the basis functions and the components from line b), w is calculated by Eq. (15). By scaling the basis functions with the obtained w the basis functions preserve the shape of the desired component. This is illustrated in line d).

new framework to fulfill five requirements related to: three-dimensional particle movement, temporal resolution, particles leaving the domain, contradicting movement of the particle images between the time steps and varying ppp densities. The case represents a generalized pressure-driven turbulent flow through a pipe.

For the synthetic case, N_p particles are initially distributed within a cubic volume with dimensions of $W \times H \times D = 500 \times 500 \times 500 \text{ mm}^3$. Each particle i has the attributes

$$P_i = \begin{pmatrix} X_i \\ I_i \end{pmatrix} \quad (17)$$

with the position $X_i \in \mathbb{R}^3$ in 3D space and the intensity $I_i \in \mathbb{R}^{n_c}$ for the individual intensities on the n_c camera images. All I_i are initialized with random uniform values between $I_{\min} = 800$ and $I_{\max} = 1200$ to prescribe a certain signal-to-noise ratio. In the considered cases $n_c = 4$

virtual cameras observe the particle distributions within the 3D cubic volume centered around $(0, 0, 0)$ corresponding to boundaries $X_{\min/\max} = \pm(250, 250, 250)$.

In a real experiment, the illuminated particles are imaged with cameras. Here, this is simulated using the transfer functions defined in equation (2) to project the particles onto the camera images. The particle positions are generally represented by floating point values and are accordingly blurred on the projected image in a way that they cover 2×2 pixels. Further, intensity fluctuations of the particle images are taken into account. In addition, camera noise is simulated by adding a random intensity amplitude for every pixel in order to simulate different signal-to-noise ratios. For the cases considered below, a moderately high signal-to-noise ratio of $SNR = 5 : 1$ is prescribed, see e.g. [39].

The generated particles are positioned in clusters forming superordinate structures A_O and advected for a prescribed number of time steps by solving analytical functions specific to A_O . For a particle i belonging to A_O at position $X_{A_O,i}^t$ at time step t , the position in the next time step $X_{A_O,i}^{t+1}$ is calculated by determining the movement direction $D_i \in \mathbb{R}^3$ defined later in equation 22. For two particle clusters, $O \in \{1, 2\}$, the particle velocities are computed solving

$$V_i = \delta_p \cdot S_i \circ D_i, \quad (18)$$

where $S_i \in \mathbb{R}^3$ is a random, uniformly distributed scaling factor with the operator \circ denoting component-wise multiplication. S_i in Eq. (18) are distributed between 0.95 and 1.05. This distribution is required to generate non-uniform particle motions with different particle velocities. The factor $\delta_p \in \mathbb{R}$ is an input parameter and controls the step width per time step of the particles in the artificial measurement volume:

$$\delta_p \propto \left| \frac{\Delta X}{\Delta t} \right| \quad (19)$$

Yet, a limiting factor for the framework is the step width per time step $\delta_S \in \mathbb{R}$ on the camera image:

$$\delta_S = \left| \frac{\Delta x}{\Delta t} \right| \quad (20)$$

with the traveled distance of a particle $\Delta x \in \mathbb{R}^2$ measured in pixels. For a particle moving parallel to the plane of the camera sensor, both quantities are proportional $w \propto \delta_S$. A large step width δ_S may reduce the accuracy of the prediction, since a particle is more likely to deviate from the path predicted by the framework. The position in the next time step $X_{O,i}^{t+1}$ is then given by:

$$X_{A_O,i}^{t+1} = X_{A_O,i}^t + V_i \quad (21)$$

The synthetic case mimics a pipe flow by forming two superordinate aligned annular structures with particle numbers N_{A_1} and N_{A_2} moving predominantly in X direction. They are Gaussianly distributed with a spread of $\mu_A = 25$ around the center line of the respective structures to simulate a generalized turbulent pipe flow. The two structures are located around the center line of the cube normal to the X axis with different radii $r_{A_1} = 175$ mm and $r_{A_2} = 105$ mm. The direction of movement $D_i \in \mathbb{R}^3$ of a particle i is computed solving

$$D_i = (2, \cos(\alpha), \sin(\alpha + \pi)) \quad (22)$$

with

$$\alpha = \arctan((X_{A_{O,i}}/r_A, (X_{A_{O,i}}/r_A) \quad (23)$$

with the angle α . By solving Eq. (18), (22) and (23) with opposite signs for δ_p in Eq. (18), two particle clusters A_O with opposite mean flow directions and rotation senses are generated. The subsequent particle position at time $t + 1$ is obtained from Eq. 21 which depends on V_i defined in Eq. (18) with the above-given D_i . In order to keep the particle number and the resulting seeding density constant, particles leaving the domain in the outflow cross-sections are reset to their starting position in the corresponding inflow cross-section mimicking a periodic boundary condition.

An exemplary particle distribution is illustrated in figure 6. For the particle numbers $N_{A_1} = 18000$ and $N_{A_2} = 10000$ and a step width of $\delta_s = 7$ px, structure A_1 is reflected by the green particle distribution and A_2 by the yellow one. The front view in figure 6 a) shows the annular structures with different radii and the rotational movement around the axis by accordingly colored arrow. The particle motion in longitudinal direction is illustrated by the side view in figure 6 b) and the dominant movement direction is indicated by arrows.

Additionally, in figure 7 the discussed flow structures are shown and the position as well as the orientation of the cameras are highlighted in blue. For this test case, three cameras are installed half-height on the horizontal plane with the same distance R_c to the cube's center and an angular displacement of 120° . One of the cameras is positioned at the front of the volume, normal to the X - Y -plane and the other two cameras are arranged accordingly. The fourth camera is located above the synthetic experiment with the central line of sight in Y -direction, also with the distance R_c to the cube's center.

A representative value of the ppp density is determined counting the number of particles in an area of 100×100 pixels. Note that the ppp densities determined with the images of the other cameras lead to nearly the same ppp values. Starting from 0.01 ppp for the lowest number of particles (7500), a particle density of 0.106 ppp is reached for the highest number of particles (45000).

In this synthetic experiment the mean velocity magnitude is $\bar{V} = 3.8$ mm/time step. Considering, a typical PIV experiment of a turbulent pipe flow relying on a PCO Dimax.HS4 camera operating at 7000Hz in rolling shutter, the above-mentioned mean velocity can be specified by $\bar{V} = 21$ m/s. For a turbulent flow of air in a pipe with a diameter of 0.35 m this leads to a bulk Reynolds number, see e.g. [5], of $Re = 440493$, which is in the range of state of the art turbulent pipe flow experiments.

3.2 Particle tracking

The proposed particle tracking approach is used to track the particle motion in comparison to the ground truth of the particle trajectories generated in the above-defined synthetic test case. To evaluate the performance of the method, the percentage of matched particles (pmp) is introduced and analyzed for particle densities between 0.015 ppp and 0.107 ppp. Two pixel step widths per time step, $\delta_s = 7$ and 14 px, which represent the mean particle velocities and thus the temporal resolution, are investigated. Each reconstructed and true particle is identified by a unique ID. The earlier is classified as matched, if it is within 1.5 mm of a true particle and both particle IDs persist over time. The pmp is then defined by the number of matched particles compared to the true total particle number.

The employed input parameters used for the framework are summarized in table 1 including links to their definition in section 2.

Parameter	Definition	Value
ϵ_d	fig. 3	1
ϵ_s	sec. 2.3	1
ϵ_Δ	sec. 2.3	3
ϵ_{back}	Eq. 16	1.5
gap	sec. 2.3	1

Table 1 Summary of the parameter settings for evaluation with the framework.

A basic requirement for the method is the reconstruction of large flow structures. Figure 8 shows the reconstructed structures for the synthetic data set at time step 50, where parts of the reconstructed particle trajectories extending over 10 time steps are depicted. They are color-coded with starting time step of the particle trajectory. The comparison with the synthetically generated structures shown in figure 6 reveals the qualitative agreement. Further, most of the trajectories are colored in blue and started accordingly at a time step close to $t = 0$. This proves the capability of the track initialization and maintaining long trajectories. Only the two regions for the newly entering particles in figure 6b) (left for

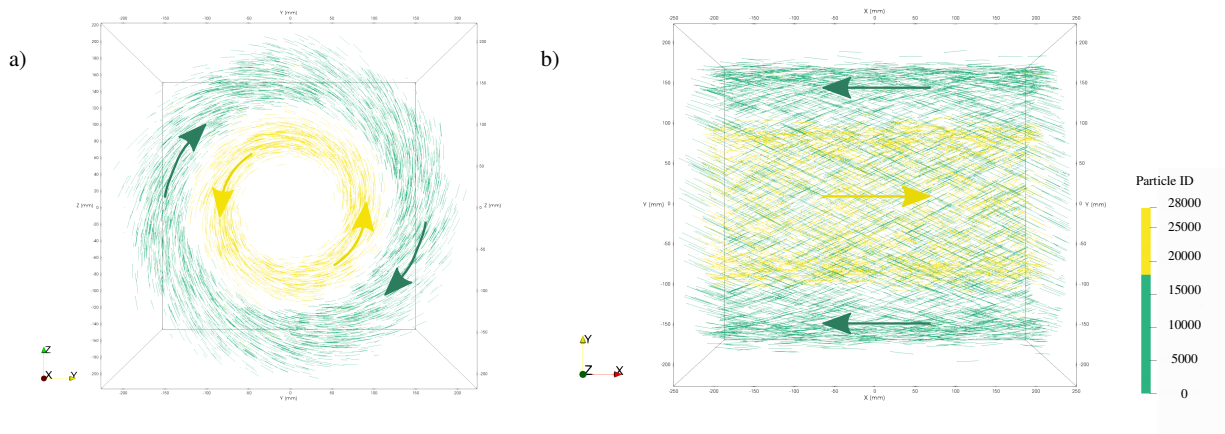


Fig. 6 Particle distribution organized in two counter-rotating aligned annular structures with opposite mean flow directions. Particles belonging to structure A_1 are colored in green and those forming A_2 are highlighted in yellow. In a) the perspective is chosen along the X axis to illustrate the rotational movement around the axis which is also indicated by accordingly colored arrows. In b) the view along the Z -axis reflects the movement in X direction with the dominant movement direction indicated by arrows.

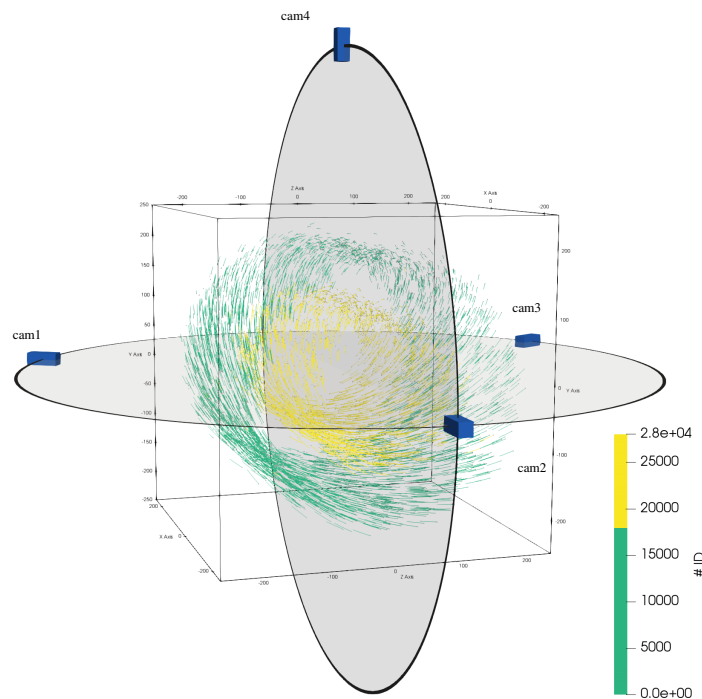


Fig. 7 Four cameras shown in blue located on planes indicated by large circles surrounding the artificial flow structures of the synthetic case. Particles belonging to structures A_1 and A_2 are colored in green and yellow, respectively.

outer structure and right for the inner one) have higher starting time steps up to $t = 50$ reflected by the white to red colors.. This highlights the ability of adding new particles and trajectories and at the same time maintaining long trajectories. The quantitative differences are discussed in the following.

It is well-known for PTV methods that a certain number of time steps is required to accumulate all particles for the tracking system in the first processing step. Thus, an important quality criterion for those methods is the number of required time steps to acquire the maximum possible pmp.

Figure 9a) shows the pmp and the number of matched particles obtained for 28000 true particles as a function of the time step. A close-up view of the first 20 time steps in the lower right corner is presented. After 9 time steps a plateau of about 90% matched particles corresponding to a total number of 25200 particles is reached. It should be noted that the method does not reconstruct ghost particles.. Additionally, a significant length of the reconstructed tracks is necessary to obtain a sufficient amount of connected information for the paths of the particles. Figure 9b) shows the frequency distribution of the track length measured in time steps. For

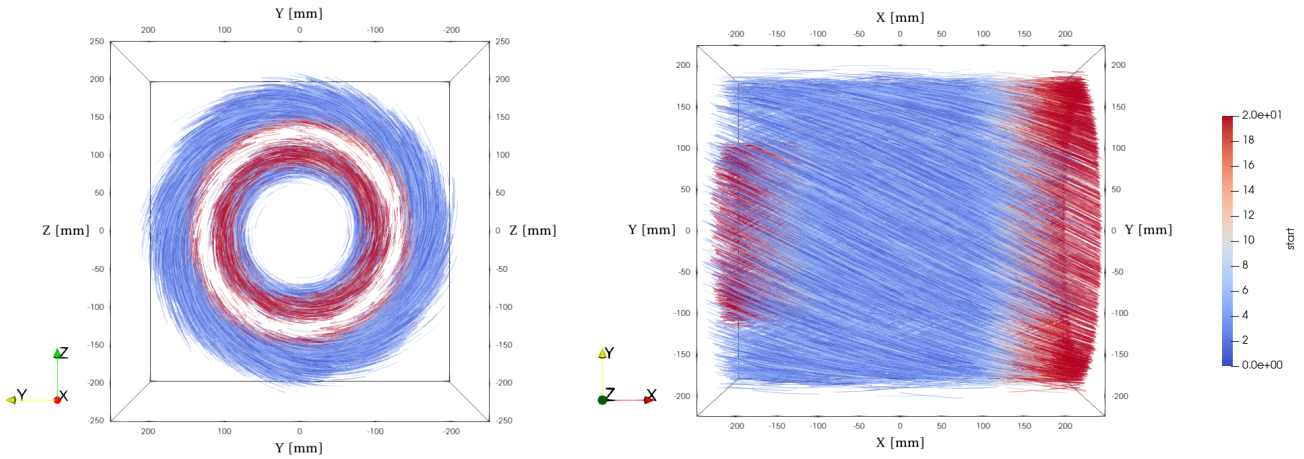


Fig. 8 Reconstruction of the aligned annular structures from the synthetic data set at time step 50. Particle trajectories reflecting 20 time steps are depicted and color-coded with the starting time step of the particle trajectory.

short track lengths of 0 to 70 time steps a low frequency of $0.2 \cdot 10^8$ is reported with a large increase to $1.6 \cdot 10^8$ and $1.2 \cdot 10^8$ for longer tracks of 70 and 90 time steps respectively. Considering an average velocity of 4.1 mm/s and a sample length of 500, a maximum of 120 time steps could be reached before the particles leave the domain indicating that some time steps are necessary to add the particles to stable tracks and some tracks break while the particle is advected through the volume. Figure 9c) shows a violin plot of the three velocity components. The colored area is normalized thus showing the relative distribution per component. The X component exhibits two separated agglomerations between ± 3.5 and 4 mm/s. The contribution to the negative values is larger compared to the positive values. This results from the higher particle number in the outer structure with negative movement direction in X . The distributions for the Y and Z velocity components are similar with two agglomerations at the higher velocities. This is due to the circular motion coupling both components: When the velocity, is high in Y , it is low in Z and vice versa. Both the qualitative behaviour and quantitative distribution comply with the ground truth.

In order to demonstrate the performance of the framework, the matched particles are evaluated in terms of the ppp. In this respect, figure 10 reflects the matched particles and pmp as a function of the true particle number. The curve starts close to 100 % pmp for the lowest particle number. For an increasing ppp, a steady decrease of the pmp is observed. For ppp=0.05, the pmp is about 92.5%. For a high ppp of 0.09, a fluctuation from the otherwise steady decrease is reported. Finally, for the highest ppp of 0.11, a pmp of 80% is achieved. An increased particle velocity is investigated represented by the step width per time step of $\delta_S = 14$ px reflected by the orange line in figure 10. The general behavior of this reconstruction efficiency is similar to the previous case. For the lowest particle density of 0.005 ppp an

efficiency of 97 % pmp is reported with a steady decrease to 77 % pmp for a high ppp of 0.1. While for the lowest particle number, the difference between the two particle velocities is about 1 % pmp it increases to about 2.5 % for the highest ppp. The general decrease of the pmp is not surprising, since the association of particles with stable tracks becomes more ambiguous for increasing ppp. Still, while pmp = 99% for ppp = 0.007 corresponds to stable information from 7000 particle tracks, a significant increase in information density is achieved with 34000 stable tracks for the highest ppp.

Considering the above-discussed, the proposed method is applicable for the investigated parameter range of $\text{ppp} \leq 0.100$ and $\delta_S \leq 14$ px. The following section presents the applicability to experimental data.

4 Applications

In the following subsections the applicability of the framework to experimental data is demonstrated and the respective results are presented. In subsection 4.1 the tracking of the motion of *Dictyostelium discoideum* cells in a 2D micro-scale application is investigated. In subsection 4.2 the results of the flow measurement in 3D confined thermal convection are discussed.

4.1 *Dictyostelium discoideum* motion

As one of the experimental cases we investigate the amoeba cells *Dictyostelium discoideum* (*Dd*), a key model organism for the study of eukaryotic migration. Besides chemical gradients, *Dd* cells have the ability to detect electric fields of continuous current and respond to them with a directed migratory movement toward the cathode.

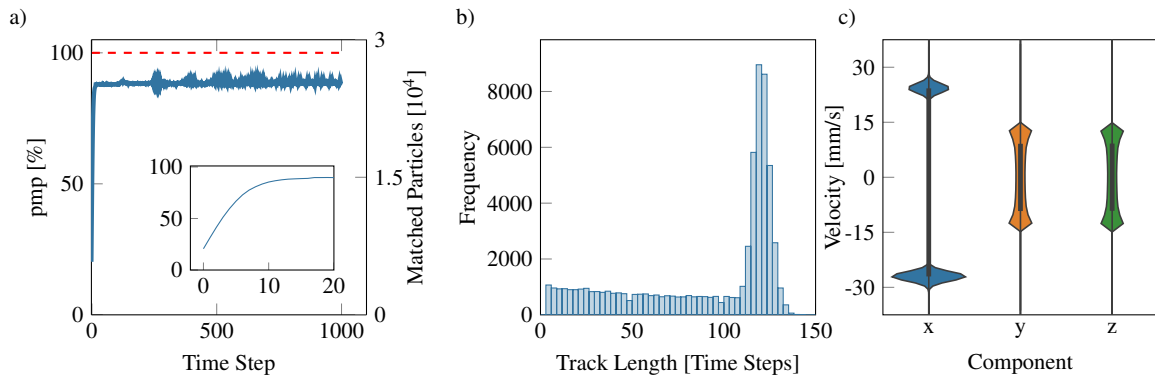


Fig. 9 Statistics for the synthetic data set containing 28000 true particles. a) shows the number of matched tracks and pmp as a function of the time step in blue. The true particles number is indicated as a red line. b) shows the frequency distribution of the track length. c) shows the violin plot of the velocities of the tracked particles, where the estimated probability density is given by the colored shape, the interquartile range by the thicker black line and the whole data range by the thinner line.

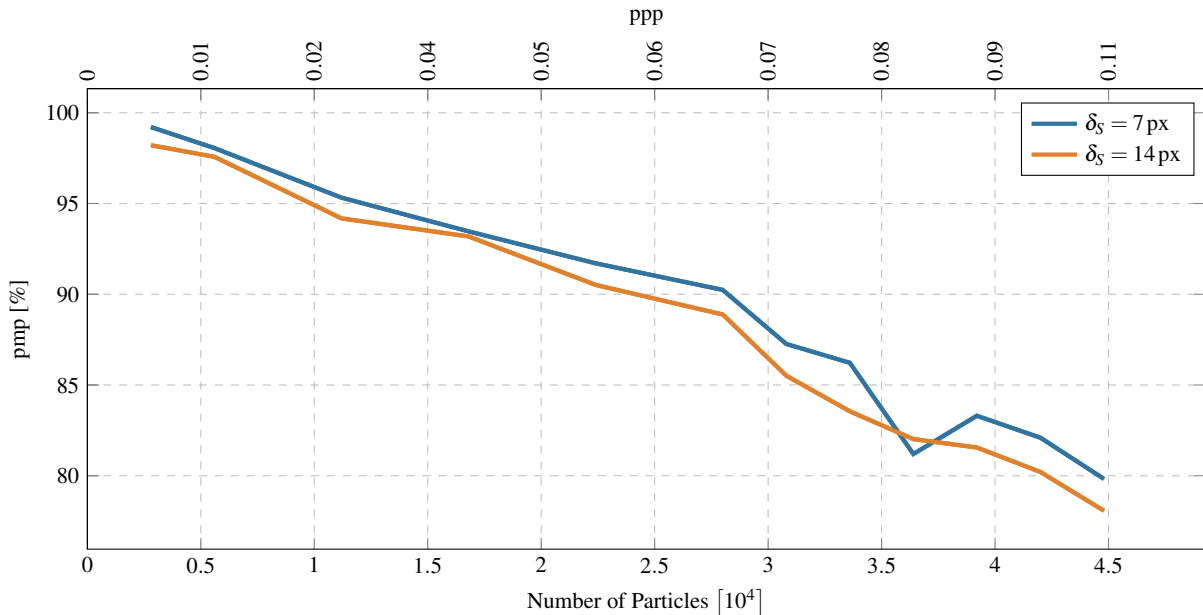


Fig. 10 Number of matched particles and pmp as a function of the true particle number for two $\delta_S = 7$ and 14 px.

4.1.1 Experiment

In a previous study we applied the electric field to the cells seeded into a microfluidic channel and reversed the polarity of the electric field every 30 minutes. We observed a strong cellular response to the electric stimulation [10]. Indeed, the cells show a directed migration toward the cathode and inverted their trajectory when the polarity of the electric field was reversed.

This experiment is an example to illustrate how our framework deals with cells that promptly change their shape and partly overlap. Especially challenging for a particle tracking algorithm is the abrupt reversal of the movement direction of the cells in addition to the stochastic component of the directed walk. The movement of the *Dd* cells is observed in

a total field of view of $660 \times 660 \mu\text{m}^2$ using a bright field inverted microscope with one mounted camera. The recording frequency is 0.05 Hz, acquiring one image frame every 20 seconds.

4.1.2 *Dictyostelium discoideum* tracking

The movement of the cells is recorded by a single camera, thus reducing the employed input parameters for the framework as summarized in table 2 including links to their definition in section 2. The current position of the cells for the particle tracking is deduced from the particle images with the open source software ImageJ [29], illustrated in figure 11. For the background subtraction the *Trainable Weka Segmentation* [32] was used on the raw data, where the *Dd* cells

Parameter	Definition	Value
ϵ_Δ	sec. 2.3	1.5
n_p	sec. 2.1	5
ϵ_{back}	Eq. 16	1.5
gap	sec. 2.5	3

Table 2 Summary of the parameter settings for evaluation with the framework.

were segmented to one class and everything else (like the background) to another. like this, frame was segmented individually. The segmentation was then applied as a mask to the raw images followed by the RSC method to determine the centers of the cell clusters.

The experimental applicability of the framework is demonstrated by tracking the Dd movement presented in figure 12. The results of the particle tracking for three time instants, $t = 2000$ s, $t = 6000$ s and $t = 10000$ s, are shown.

In order to make quantitative statements on the Dd motion, the trajectories resulting from the tracking are further analyzed. As cells are no solid particles they tend to aggregate with neighbor cells when they come close to each other. Subsequently, they are likely to split again and separately continue migrating and they may move out of the field of view. For this reason, the number of simultaneously tracked cells is not constant over time, see figure 13 a). Starting at about 90 cells for $t = 0$ min an increase in simultaneously tracked cells is reported up to 113 cells at frame $t = 65$. Afterwards, the events described above yield fluctuations in the number and on average a decrease in simultaneously tracked cells down to 76 for frame $t = 474$ is reported. A total of 1059 tracks were generated and their distribution is shown in figure 13 b). 566 tracks has a length of ≤ 20 time steps, the average track length is ≈ 48.3 with a standard deviation of ≈ 84 . The median track length is 17. 160 tracks have a length of more than 80 time steps.. As mentioned above, Dd cells have the ability to detect continuous current electric fields. Interestingly, this characteristic becomes evident in the velocity distribution of the pursued amoebae. The violin plot in figure 13 c) shows that the density distributions exhibit two extreme values. From this, it can be interpreted that the cells do not have one expected value in terms of velocity, but two. This is probably due to the reversed polarity of the electric field during the experiment. At this point, however, it should be noted that this work is not about generating insight into Dd dynamics and that the interpretation and validation of the two expected values is not part of the present study. We only want to demonstrate that our approach is suitable for evaluating this kind of data.

4.2 Turbulent Rayleigh-Bénard convection

The other experimental case investigates PTV data acquired in a cubic Rayleigh-Bénard convection (RBC) sample with side length of $l = 500$ mm filled with water.

4.2.1 Experimental setup

A sketch of the sample is shown in figure 14. The top and bottom of the experiment are embedded in an insulation mantle indicated by (a). The temperature difference ΔT is generated by the cooling and heating elements, (b) and (f), whose temperatures, T_C and T_H , are controlled by two water circuits. The developing fluid flow is measured between two anodised black aluminum plates, (c) and (e), enclosing the sample. The side walls (d) are made of 10 mm thick glass yielding high optical accessibility of the measurement volume. As light source, a high-power white-light LED array is used to illuminate TiO_2 -coated latex particles as flow tracers imaged by four cameras with a recording rate of 5 Hz.

For the here considered case the experimental settings are $T_C = 18^\circ\text{C}$, $T_H = 24^\circ\text{C}$, with an average sample temperature of $\bar{T} = 21^\circ\text{C}$ and $\Delta T = 6$ K. The corresponding characteristic numbers are a Prandtl number of 6.9 and a Rayleigh number of $1.0 \cdot 10^{10}$. A full description of the experimental apparatus and setup can be found in [31].

4.2.2 Particle tracking

The employed input parameters for the evaluation of this turbulent 3D flow with the HD-PTV method are summarized in table 3 including links to their definition in section 2. Statistics are acquired by evaluating 250 time steps.

Parameter	Definition	Value
ϵ_d	fig. 3	2
ϵ_S	sec. 2.3	1.7
ϵ_Δ	sec. 2.3	3
ϵ_{back}	Eq. 16	2.5
gap	sec. 2.5	1

Table 3 Summary of the parameter settings for the evaluation with the framework.

Here, the applicability of the framework is proven by the obtained particle trajectories presented in figure 15. For time step $t = 100$, the particles are presented as dots with the particle path for the previous 50 time steps attached as a tail. The latter is color-coded by the velocity magnitude. In figure 15a) a view along the Z -axis is chosen to demonstrate the reconstruction of particle tracks in the entire volume. In addition, a large-scale flow structure filling the convection sample is visible. This was also found for the same data set

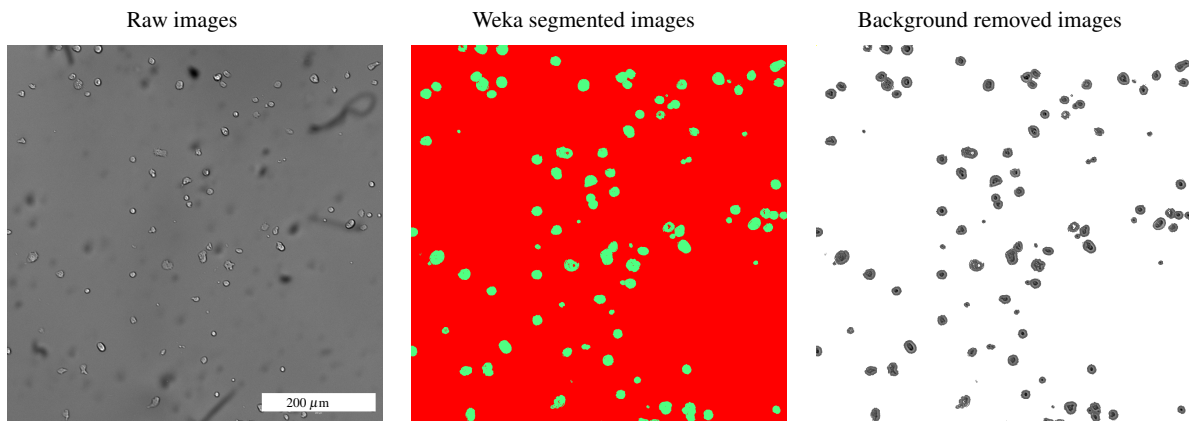


Fig. 11 Image preprocessing steps for the amoeba case. The raw data are shown on the left. To remove the background, the Weka segmentation [2] was applied to the raw data with default parameters. The green segments in the image are the amoeba, while the red segment is considered as the background or other irrelevant parts. The image on the right shows the background removed images, achieved by using the segmented images as masks on the raw images.

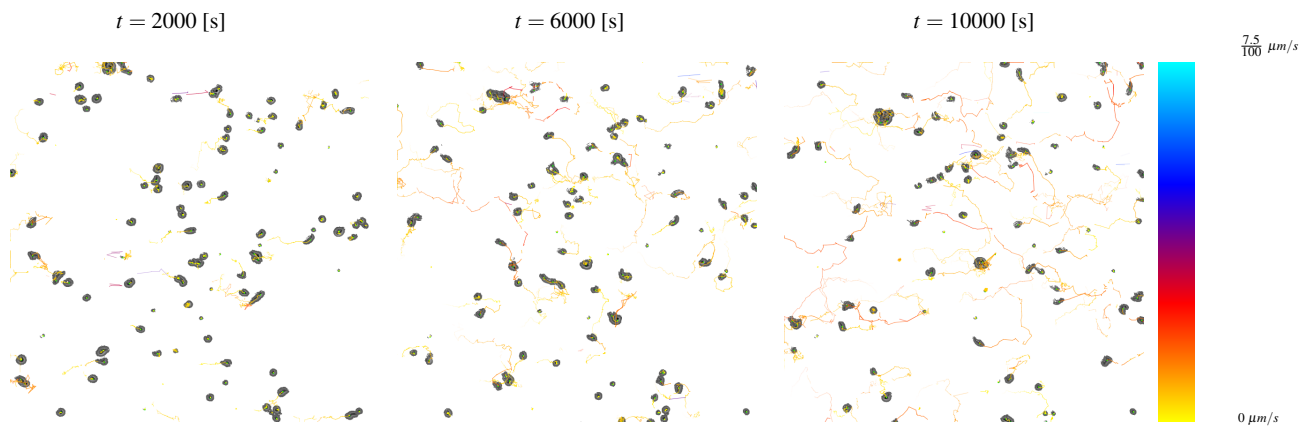


Fig. 12 Snapshots of the reconstructed tracks. The dots are the centers of the segments used for tracking. The segments of the tracks are color-coded in accordance with to mean velocity of the track.

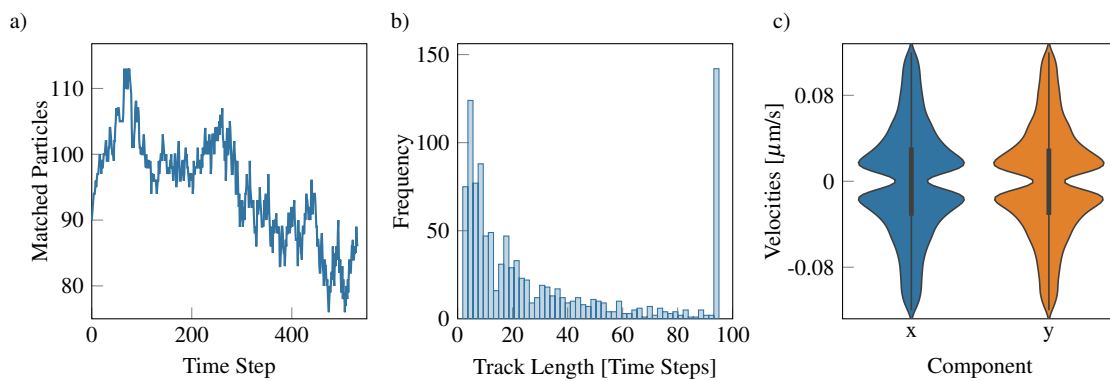


Fig. 13 a) shows the number of simultaneously tracked amoeba depending on the time step. b) reflects the frequency distribution of the track length, whereas c) shows a violin plot of the velocities.

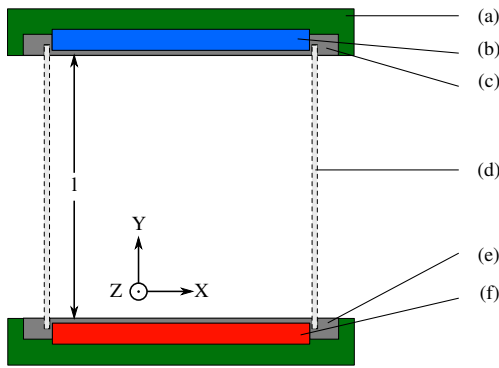


Fig. 14 Schematic of the cubic RBC sample along Z-direction. (a) marks the insulation layer, (b) and (f) indicate the cooling and heating elements and (c) and (e) mark the anodised aluminum plates enclosing the sample. The glass cube (d) and the characteristic length l of the sample are indicated.

by [31] using tomographic PIV. New insight is gained by the reconstruction of a previously unresolved smaller corner flow reported at X and $Y \approx 50$ mm. In figure 15b) the perspective is rotated to a diagonal plane of the sample in order to reflect the orientation of the large-scale circulation (LSC) more clearly. This perspective can also be seen in figure 16, where the velocity components are projected onto a Cartesian grid. For this projection the mean velocities in a radius of 50 mm are projected onto an equidistantly distributed grid over the whole volume. The grid points are illustrated as glyphs, where the orientation is given by the velocity and the size corresponds to the magnitude. The instantaneous velocity magnitudes range up to 20 mm/s with the majority being significantly slower as reflected by the dominantly blue trajectories.

In order to make quantitative statements on the flow occurring within RBC, the trajectories resulting from the tracking are further analyzed. Contrary to the amoeba example from the previous case, the particle number for the enclosed RBC case is expected to be constant. This is confirmed in figure 17 a). After only six time steps the number of simultaneously tracked particles is 27472 with a stable plateau of about 28500 particles reached at time step 10.

For PTV methods, typically long trajectories are desired. Figure 17b) reflects the frequency distribution of the track length. A distribution with a track length of up to 250 time steps and with the strongest contribution to smaller track length (≤ 50 time steps) is obtained. The evaluated data set contained 250 time steps thus determining the limit which indicates that the longest tracks reached the potential distance. While the synthetic case showed a clear indication at a maximum track length determined by the system, the experimental case is expected to show less pronounced dominant track length then found here. The mean track length is 10.04 ± 10.03 and the median length is 7. Turbulent RBC

is characterized by velocity distribution and can therefore be used as a validation case for the presented PTV framework. A violin plot of the velocities is presented in figure 17c). Each velocity component shows a symmetric distribution. This is in agreement with the expectations in terms of a flow within an enclosed system driven by a dominant circular structure. The fact that the LSC is not exactly aligned with the diagonal plane explains the different spread in the frequency distribution of the velocity components.

The results justify the applicability of the developed framework for turbulent flows as it allows to reliably measure particle position, velocities and their associated trajectories.

5 Conclusion

A new framework approach for particle tracking based on a Gaussian Mixture Model (GMM) is presented. It is divided into an initialization step and a processing step. In the first step the velocities at the initial points are determined after iterative reconstruction of individual particles of the first four images in order to generate the tracks between these initial points without using tomographic particle image velocimetry. Subsequently, the tracks are extended by searching and including new points obtained from consecutive images using iterative reconstruction of individual particles in combination with continuous modeling of the particle trajectories with a Gaussian mixture model considering the position and velocity as interdependent quantities.

The presented approach is validated and benchmarked by tracking particles generated in a synthetic generalized turbulent pipe flow which defines the ground truth. Considering this synthetic case, the approach returns about 90% matched particles after 9 time steps without generating any ghost particles. Further, starting with 100% of matched particles for the lowest particle density considered the percentage of matched particles (pmp) decrease from 92.5%. For a particle density in terms of particle per pixels (ppp) of 0.05 to a pmp of 80% at the highest ppp of 0.11 for a step width per time step of $\delta_S = 7px$. Increasing the step width per time step from $\delta_S = 7px$ to $\delta_S = 14px$ displacement results in a similar declining curve and pmp values that are generally 5% lower.

Finally, the approach is successfully applied to two well-known tracking problems. The first one is tracking the eukaryotic migration of amoeba cells for which 1059 tracks were generated with tracks lengths of up to 100 time steps. The second one is turbulent Rayleigh-Bénard convection for which the motion of about 28500 particle is visualized using tracks of lengths up to 250 times steps.

Acknowledgements The authors like to thank Annika Köhne for proof-reading this work.

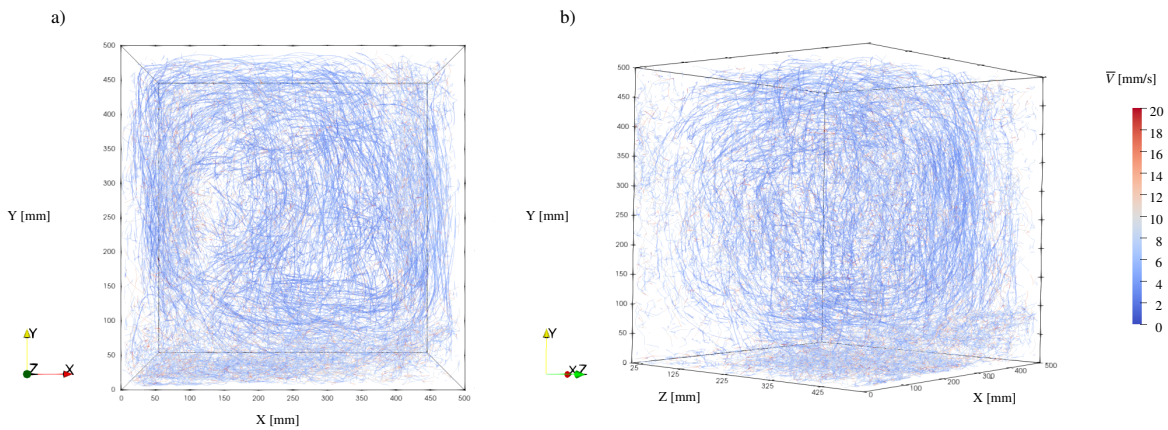


Fig. 15 Reconstruction of the RBC data at time step 100. Particle trajectories reflecting 50 time steps are depicted and color-coded with the velocity magnitude.

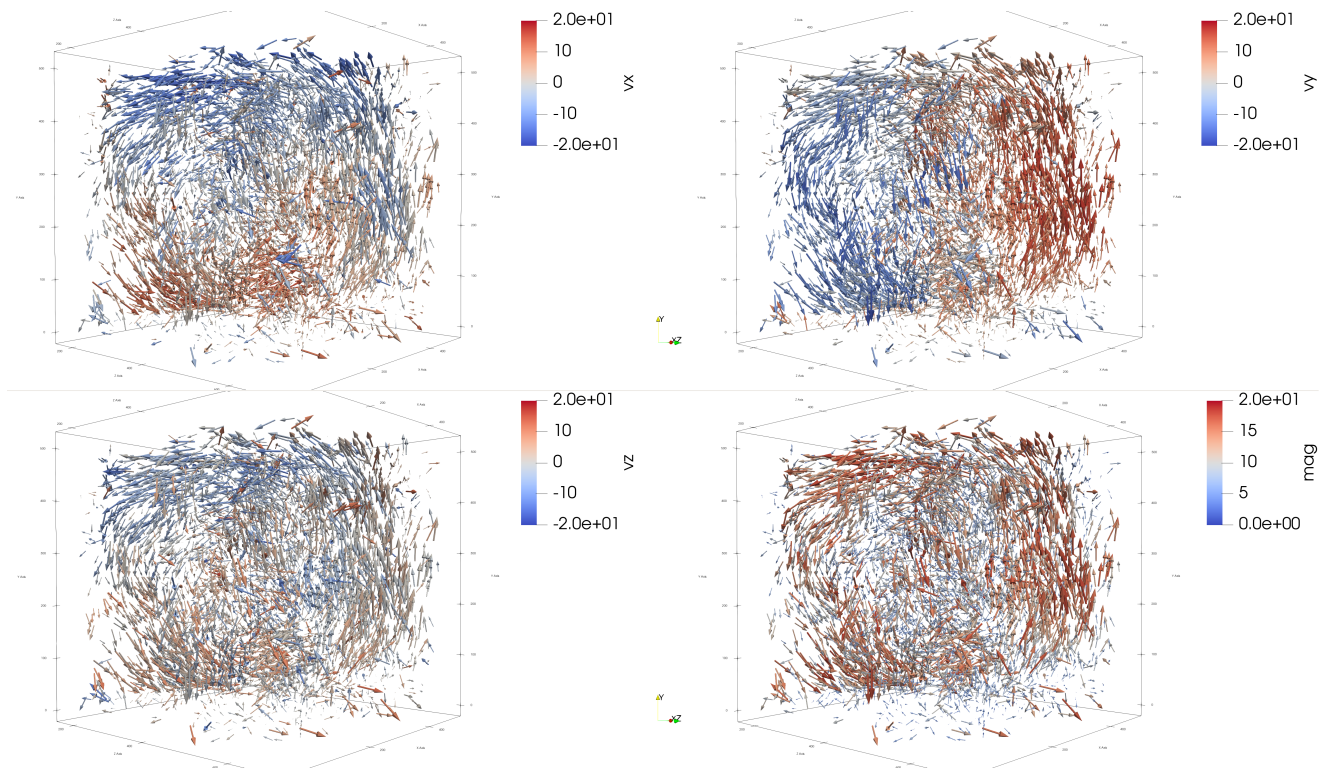


Fig. 16 Projection of the velocity components and the velocity magnitude onto a Cartesian grid, where the volume is clipped along the LSC.

Conflict of interest

The authors declare that they have no conflict of interest.

References

1. Adrian, R.: Multi-point optical measurements of simultaneous vectors in "Insteady flow" - a review. *Int. J. Heat and Fluid Flow* **7**, 127–145 (1986)
2. Arganda-Carreras, I., Kaynig, V., Rueden, C., Eliceiri, K.W., Schindelin, J., Cardona, A., Seung, S.H.: Trainable weka segmentation: a machine learning tool for microscopy pixel classification. *Bioinformatics* **33**(15), 2424–2426 (2017)
3. Atkinson, C., Soria, J.: An efficient simultaneous reconstruction technique for tomographic particle image velocimetry. *Exp. Fluids* **47**, 563–578 (2009)
4. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. *International Journal of Computer Vision* **12**(1), 43–77 (1994). DOI 10.1007/BF01420984. URL <https://doi.org/10.1007/BF01420984>
5. Bauer, C., Feldmann, D., Wagner, C.: On the convergence and scaling of high-order statistical moments in turbulent pipe flow using direct numerical simulations. *Phys. Fluids* **29**(12), 125105 (2017). DOI 10.1063/1.4996882. URL <https://doi.org/10.1063/1.4996882>

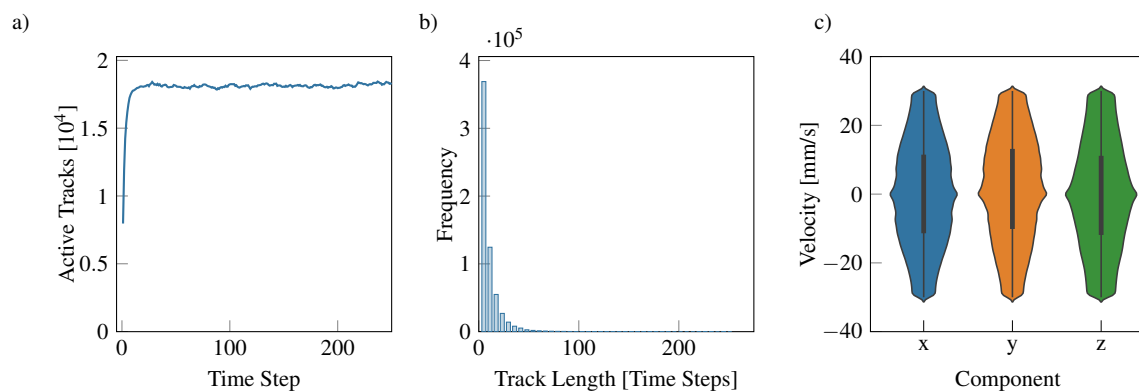


Fig. 17 Same figure structure as for figure 13 but for the RBC case. a) shows the number of simultaneously tracked tracer particles in each time step. b) reflects the frequency distribution of the track length whereas c) shows a violin plot of the velocities.

6. Cardwell, N., Vlachos, P., Thole, K.: A multi-parametric particle-pairing algorithm for particle tracking in single and multiphase flows. *Meas. Sci. Technol.* **22**, 105406 (2011)
7. Eliceiri, K.W., Berthold, M.R., Goldberg, I.G., Ibáñez, L., Manjunath, B.S., Martone, M.E., Murphy, R.F., Peng, H., Plant, A.L., Roysam, B., Stuurman, N., Swedlow, J.R., Tomancak, P., Carpenter, A.E.: Biological imaging software tools. *Nature Methods* **9**(7), 697–710 (2012). DOI 10.1038/nmeth.2084. URL <https://doi.org/10.1038/nmeth.2084>
8. Elsinga, G., Scarano, F., Wieneke, B., van Oudheusden, B.: Tomographic particle image velocimetry. *Exp. Fluids* **41**, 933–947 (2006)
9. Fuchs, T., Hain, R., Kähler, C.J.: Non-iterative double-frame 2D/3D particle tracking velocimetry. *Exp. Fluids* **58**, 119 (2017). DOI 10.1007/s00348-017-2404-0
10. Guido, I., Diehl, D., Olszok, N.A., Bodenschatz, E.: Cellular velocity, electrical persistence and sensing in developed and vegetative cells during electrotaxis. *PLOS ONE* **15**, 1–16 (2020). DOI 10.1371/journal.pone.0239379
11. Hartley, R.I., Strum, P.: Triangulation. In: *Computer Analysis of Images and Patterns: Proc. of 6th International Conference CAIP '95*, pp. 190–197. Springer, Prague, Czech Republic (1995)
12. Herman, G., Lent, A.: Iterative reconstruction algorithms. *Comput Biol Med* **6**, 273–294 (1976)
13. Kreizer, M., Ratner, D., Liberzon, A.: Real-time image processing for particle tracking velocimetry. *Exp. Fluids* **48**(1), 105–110 (2010). DOI 10.1007/s00348-009-0715-5. URL <http://dx.doi.org/10.1007/s00348-009-0715-5>
14. Lo, C.M., Wang, H.B., Dembo, M., Wang, Y.I.: Cell movement is guided by the rigidity of the substrate. *Biophysical Journal* **79**(1), 144–152 (2000). DOI [https://doi.org/10.1016/S0006-3495\(00\)76279-5](https://doi.org/10.1016/S0006-3495(00)76279-5). URL <http://www.sciencedirect.com/science/article/pii/S0006349500762795>
15. Lucy, L.: An iterative technique for the rectification of observed distributions. *Astron. J.* **79**, 745–754 (1974). DOI 10.1086/111605
16. Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixe, L., B., L.: Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision* (2020)
17. Maas, H., Gruen, A., Papantoniou, D.: Particle tracking velocimetry in three-dimensional flows. *Exp. Fluids* **15**(2), 133–146 (1993)
18. Malik, N., Dracos, T., Papantoniou, D.: Particle tracking in three dimensional turbulent flows—part ii: particle tracking. *Exp. Fluids* **15**, 279–294 (1993)
19. Malik, N.A., Dracos, T., Papantoniou, D.: Particle tracking velocimetry in three-dimensional flows, Part II: Particle tracking. *Exp. Fluids* (1993)
20. McCaig, C.D., Rajnicek, A.M., Song, B., Zhao, M.: Controlling cell behavior electrically: Current views and future potential. *Physiological Reviews* **85**(3), 943–978 (2005). DOI 10.1152/physrev.00020.2004. URL <https://doi.org/10.1152/physrev.00020.2004>. PMID: 15987799
21. Mikheev, A., Zubtsov, V.: Enhanced particle-tracking velocimetry (eptv) with a combined two-component pairmatching algorithm. *Meas. Sci. Technol.* **19**, 085401 (2008)
22. Moon, T.K.: The expectation-maximization algorithm. *IEEE Signal processing magazine* **13**(6), 47–60 (1996)
23. Nishino, K., Kasagi, N., Hirata, M.: Three-dimensional particle tracking velocimetry based on automated digital image processing. *Trans. ASME J. Fluid Eng.* **111**, 384–390 (1989)
24. Novara, M., Schanz, D., Gesemann, S., Lynch, K., Schröder, A.: Lagrangian 3d particle tracking for multi-pulse systems: performance assessment and application of shake-the-box. In: *18th International Symposium on the Application of Laser and Imaging Techniques to Fluid Mechanics*, pp. 2638–2663 (2016)
25. Parthasarathy, R.: Rapid, accurate particle tracking by calculation of radial symmetry centers. *Nature Methods* **9**(7), 724–726 (2012)
26. Rathi, Y., Vaswani, N., Tannenbaum, H., Yezzi, A.: Tracking deforming objects using particle filtering for geometric active contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **29**(8), 1470–1475 (2007). DOI 10.1109/TPAMI.2007.1081
27. Reynolds, D.: Gaussian mixture models. *Encyclopedia of biometrics* pp. 827–832 (2015)
28. Richardson, W.H.: Bayesian-based iterative method of image restoration*. *J. Opt. Soc. Am.* **62**(1), 55–59 (1972). DOI 10.1364/JOSA.62.000055. URL <http://www.osapublishing.org/abstract.cfm?URI=josa-62-1-55>
29. Rueden, C.T., Schindelin, J., Hiner, M.C., DeZonia, B.E., Walter, A.E., Arena, E.T., Eliceiri, K.W.: ImageJ2: ImageJ for the next generation of scientific image data. *BMC Bioinformatics* **18**(1), 529 (2017). DOI 10.1186/s12859-017-1934-z. URL <https://doi.org/10.1186/s12859-017-1934-z>
30. Schanz, D., Gesemann, S., Schröder, A.: Shake-the-box: Lagrangian particle tracking at high particle image densities. *Exp. Fluids* **57**(5), 70 (2016)
31. Schiepel, D., Bosbach, J., Wagner, C.: Tomographic Particle Image Velocimetry of Turbulent Rayleigh-Bénard Convection in a Cubic Sample. *Journal of Flow Visualization and Image Processing* **20**(1-2), 3–23 (2013)
32. Schindelin, J., Arganda-Carreras, I., Frise, E., Kaynig, V., Longair, M., Pietzsch, T., et al.: Fiji: an open-source platform for biological-image analysis. *Nature methods* **9**(7), 676–682 (2012)
33. Soloff, S.M., Adrian, R.J., Liu, Z.C.: Distortion compensation for generalized stereoscopic particle image velocimetry. *Meas. Sci. Technol.* **8**(12), 1441 (1997)

34. Trepap, X., Chen, Z., Jacobson, K.: Cell Migration, pp. 2369–2392. American Cancer Society (2012). DOI 10.1002/cphy.c110012. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/cphy.c110012>
35. Ulman, V., et al.: An objective comparison of cell-tracking algorithms. *Nature Methods* **14**(12), 1141–1152 (2017). DOI 10.1038/nmeth.4473. URL <https://doi.org/10.1038/nmeth.4473>
36. Van Haastert, P.J.M., Devreotes, P.N.: Chemotaxis: signalling the way forward. *Nature Reviews Molecular Cell Biology* **5**(8), 626–634 (2004). DOI 10.1038/nrm1435. URL <https://doi.org/10.1038/nrm1435>
37. Wieneke, B.: Volume self-calibration for 3D particle image velocimetry. *Exp. Fluids* **45**(4), 549–556 (2008). DOI 10.1007/s00348-008-0521-5. URL <http://dx.doi.org/10.1007/s00348-008-0521-5>
38. Wieneke, B.: Iterative reconstruction of volumetric particle distribution. *Meas. Sci. Technol.* **24**, 024008 (2013)
39. Xue, Z., Charonko, J., Vlachos, P.: Particle image velocimetry correlation signal-to-noise ratio metrics and measurement uncertainty quantification. *Measurement Science and Technology* **25**(11), 115301 (2014). URL <http://stacks.iop.org/0957-0233/25/i=11/a=115301>

5.3 Conclusions from (Herzog et al. 2021b)

The presented publication shows the framework that was developed to extract particles from images and assemble them into trajectories. The approach works for two dimensional as well as for three dimensional trajectories, depending on the images used as input. If the images are multi-perspective images, the individual images, at every time-step, are used to triangulate the particles, in which case the extracted particles are three dimensionally represented, resulting in three-dimensional trajectories. The approach had first been validated on synthetic cases, where a performance was achieved which is equally good to the state of the art procedures, which, however, are only available commercially. Later two experimental cases had also been evaluated as examples.

The generated trajectories will be used in the following to train the presented two approaches from *(Herzog et al. 2021a) (nODE) and *(Herzog et al. 2019) (CAE+CRF network). In the case of the method from *(Herzog et al. 2021a) the goal is to investigate what the predicted vector field looks like and the goal of the second case is to validate if it is possible to predict the reconstructed trajectories from *(Herzog et al. 2021b) using the hybrid model from *(Herzog et al. 2019).

5.3.1 Application of the presented methods

We will start with the case of the motion of *Dictyostelium discoideum* introduced in *(Herzog et al. 2021b, section 4.1). As described in *(Herzog et al. 2021b, section 4.1), these amoebae react to electric fields of continuous current and move towards them. For this purpose, two sets of trajectories were generated, one for the case where the amoebae move predominantly to the left (see figure 5.2) and one for the case where they move to the right. Looking at figure 5.2, it is difficult to see whether the generated trajectories really follow a trend, the trajectories are not particularly smooth and the amoebae do not move in any straight pattern.

The hybrid network (CAE+CRF) from *(Herzog et al. 2019) was then applied on these two data sets. The above extracted trajectories were used as training

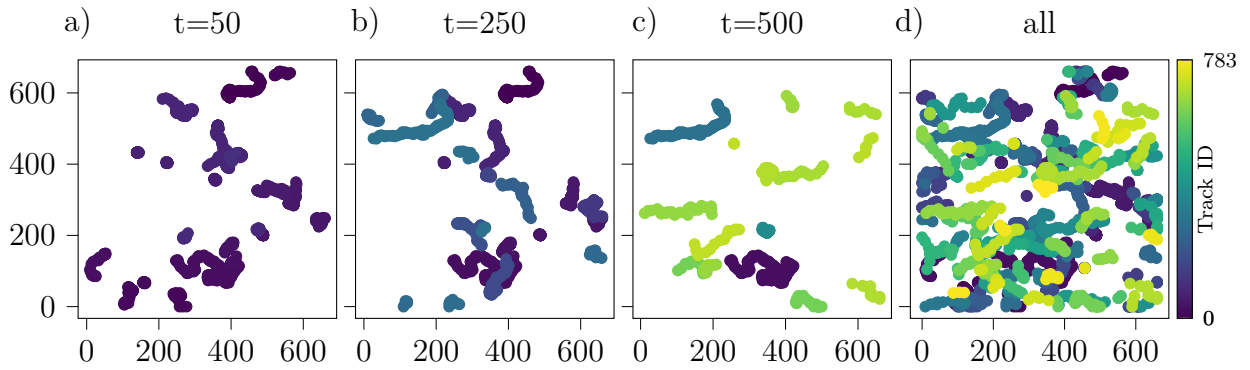


Figure 5.2: Trajectories generated based on the method *(Herzog et al. 2021b) for dictyostelium discoideum motion. To avoid clutter in the graphics, only every fourth trajectory is shown, with a maximum trajectory length of 250 time steps. In a) one sees the trajectories at time $t = 50$, b) shows the trajectories at time $t = 250$ and c) for $t = 500$. The trajectories can be identified unambiguously by their colour, the colour of the trajectories is consistent over all time steps. Figure d) shows all generated trajectories for all time steps.

components for position and velocity. Figure 5.3 shows the two vector fields that were created for each data set. Each vector field was cropped to the size of the image sections.

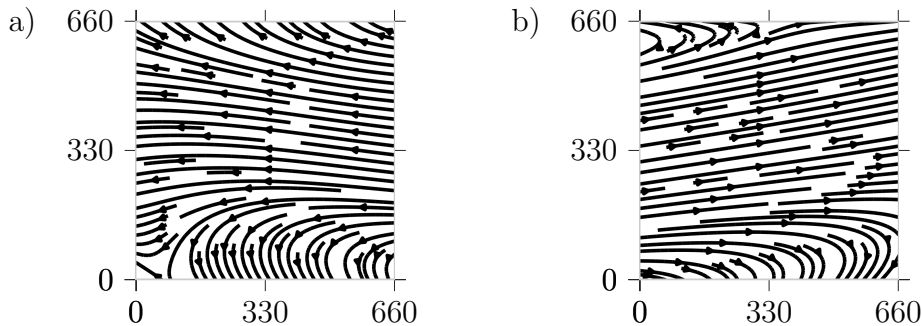


Figure 5.3: Reconstructed vector fields, where in a) one sees the case where the amoebae were supposed to move to the left and in b) the reversed case. The size of the vector fields was limited to the size of the reconstructed data.

As can be seen in figure 5.3 a), there is a dominant movement to the left, the same applies to figure 5.3 b) where the dominant direction is to the right. This observation is consistent with the intention behind the experimental setup. If, on the other hand, one looks only at the reconstructed trajectories (figure 5.2), this trend is much more difficult to recognise, if at all.

The second case to be considered is the turbulent Rayleigh-Bénard convection (RBC) cell also from *(Herzog et al. 2021b, section 4.2). Classically, the liquid starts rising at the heated bottom plate, cools down at the surface plate and sinks to the bottom again. Depending on the physical conditions, different structures are created as investigated in (Pallares et al. 2001). In the case presented in *(Herzog et al. 2021b, section 4.2), a large-scale circulation (LSC) is formed across the diagonal of the cell. The figures *(Herzog et al. 2021b, p. 15) and *(Herzog et al. 2021b, p. 16) attempt to illustrate this. As before, the reconstructed trajectories from *(Herzog et al. 2021b) serve as training data for the *(Herzog et al. 2019) (CAE+CRF) approach. The architecture of the network was adapted in such a way that it takes input $\{((X(t), X_v(t)), (Y(t), Y_v(t)), (Z(t), Z_v(t)))\}_{t=0}^{t=9} \in \mathbb{R}^{2 \times 3 \times 10}$, where $X(t), Y(t), Z(t)$ are the position components and $X_v(t), Y_v(t), Z_v(t)$ the velocity components of a reconstructed trajectory, at some time step t . To generate the training data set each reconstructed trajectory from *(Herzog et al. 2021b) was extended to the left with 9 times the start position of the trajectory but a velocity of 0 and 9 times the end position of the trajectory, but again 0 velocity. That is, if a trajectory consists of t^*, \dots, t_N^* elements, it was extended to $t - 9, \dots, t^*, \dots, t_N^*, \dots, t_{N+9}$. A sliding window approach over $t - 9$ to t_N^* then divided the reconstructed trajectories into pieces of size $\mathbb{R}^{2 \times 3 \times 10}$. The desired output from the network should be $\{((X(t), X_v(t)), (Y(t), Y_v(t)), (Z(t), Z_v(t)))\}_{t=10}^{t=20}$. Hence, based on 10 consecutive time steps, the next 10 time steps are predicted.

After training, it should be seen whether the network predicts a dynamic that is consistent in respect to the observed data. For this purpose, 50 starting points were randomly initialised in a sphere at (100, 100, 50) having a radius of 50. The process was then run for 1000 time steps. Based on these 50 points, the network should predict 50 trajectories with a length of 1000. The result of this prediction can be seen in figure 5.5. It is well visible how the trajectories start in the lower corner and move into the large-scale circulation (LSC) and getting faster towards it. It is also noticeable that not all trajectories immediately transfer into the LSC, which is an indication that the network predicts more than only a transfer to the LSC. The

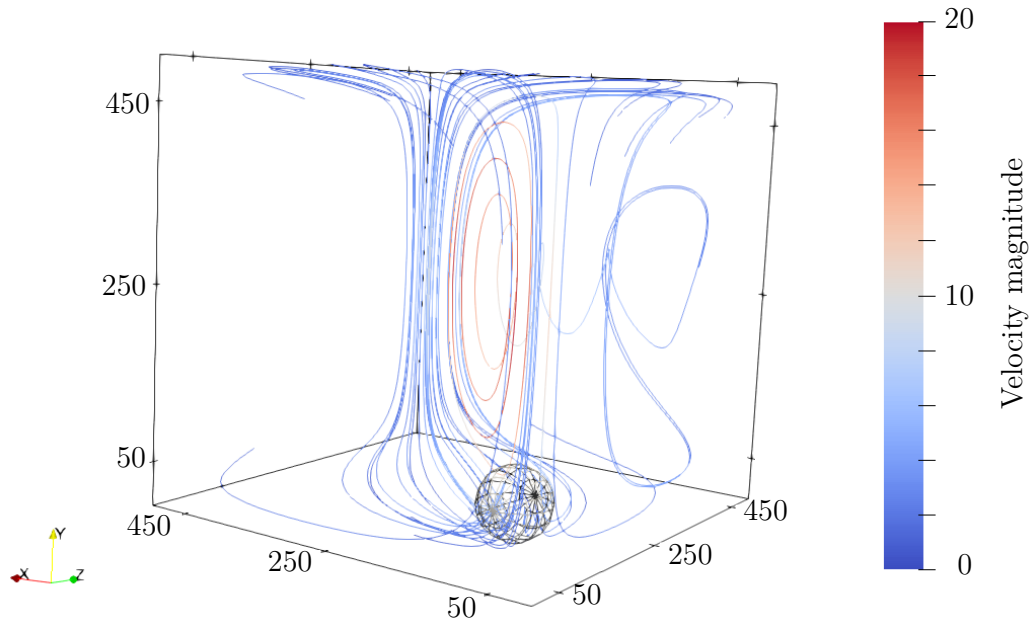


Figure 5.4: Prediction of the RBC data based on 50 starting points, randomly chosen in a sphere at (100, 100, 50) and with a radius of 50 (the sphere is drawn in), the colour of the trajectories corresponds to the velocity magnitude in a range of 0 in blue up to 20 mm/s.

next step was to see if the entire dynamics of the cell could be reconstructed. For this purpose, 5000 equidistant points were initialised over the entire volume and predicted for 2000 time steps. Furthermore, these results can be compared with *(Herzog et al. 2021b, figure 15), for which the same viewpoints were chosen.

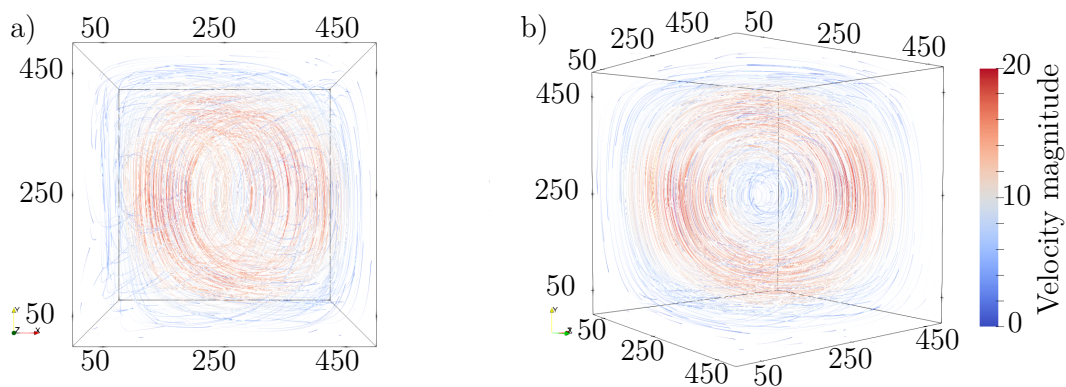


Figure 5.5: Prediction of the RBC data based on 5000 equidistant starting points, same colour bar as in figure 5.4. This plot represents the direct comparison to *(Herzog et al. 2021b, figure 15).

As can be seen, the LSC is easier to identify based on the predicted trajectories than on the reconstructed data. Furthermore, the predicted trajectories are always

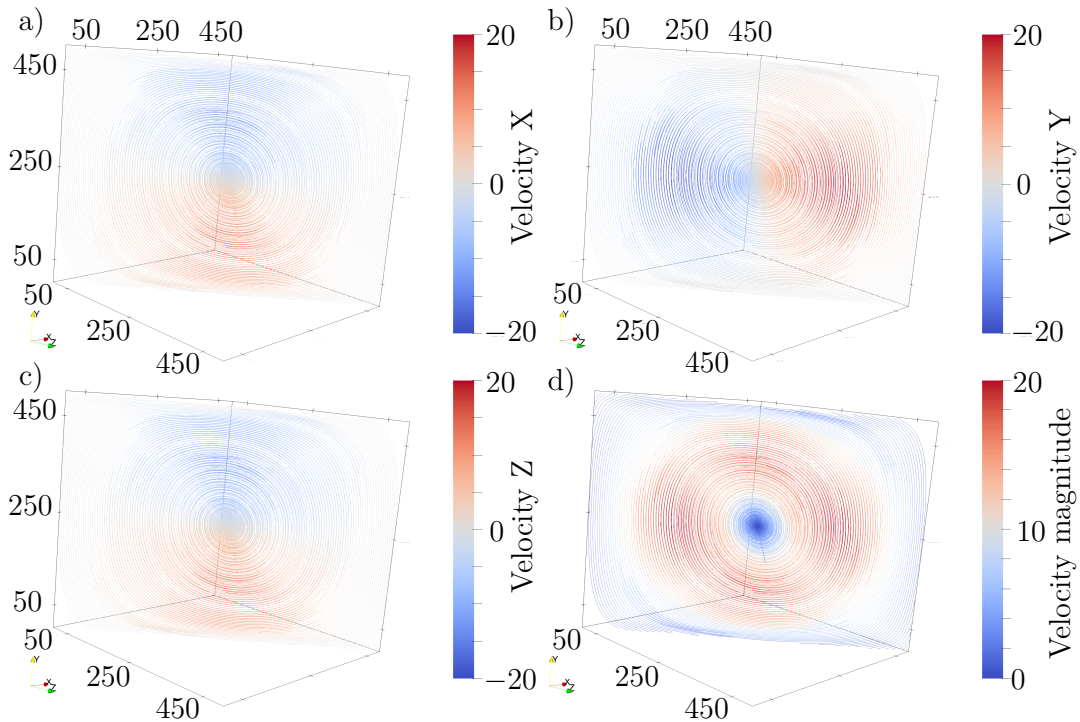


Figure 5.6: The views shows a slice along the LSC in the cell. The frames a)-d) always show the same trajectories but with different colouring. This plot represents the direct comparison to *(Herzog et al. 2021b, figure 16). In a) it is easy to see that the X component of the speed is to the right in the lower half and to the left in the upper half. b) Shows the same for the movement upwards, i.e. in the y direction. The particles float upwards on the right side of the cell and downwards on the left side of the cell. Since the LSC lies along the diagonal of the cell, it is also easy to see in c) that the particles near the bottom tend to move towards the camera and at the top they tend to move back. d) shows again the magnitude of the velocities and illustrates the LSC.

of full length, unlike the reconstructed trajectories. Next, the last comparison to be made corresponds to *(Herzog et al. 2021b, figure 16). For this purpose, the number of initial points was ten times higher (50000) than before and a slice along the diagonal of the LSC was made to see how the velocities are distributed inside the LSC. If one compares the two figures, one finds that both approaches deliver similar results, but the results shown here are clearer to recognise and contain less disturbances. The transitions of the velocity distributions are easier to recognise and interpret. It should also be pointed out that the prediction did not generate any effects that are not present in the data. Rather, the noise in the reconstructed trajectories seems to have been strongly reduced.

Looking at figure 5.6 and comparing it with *(Herzog et al. 2021b, figure 16),

one can qualitatively determine that the velocities in the X and Y directions are consistent to the reconstructed trajectories. The same applies to the Z direction, although the difference is greater. The results of the prediction can thus be considered consistent with the results of the reconstruction. In detail, however, there are major differences. The LSC is much clearer in the predicted trajectories than in the reconstructed data, as are the transitions of the velocities. In particular, when looking at the velocity magnitude, one sees in the predicted images the classic textbook picture of the LSC, particularly fast in the inner ring, slower at the edges towards the cell wall and comparatively slow in the very centre. Such formations have also been seen in other experimental set-ups like presented by Pallares et al. (2001), where, by contrast, a much smaller cell and silicon oil was used as the fluid, which is generally characterised by other parameters ($Pr = 130$ and $5 \cdot 10^3 \leq Ra \leq 8 \cdot 10^4$), i.e. the ones we have. More important, the work of Pallares et al. (2001) shows that variation of the Rayleigh number results in different LSCs. And the generated prediction of the hybrid approach produces exactly one of the possible configurations, only using the reconstructed data.

5.4 Summary

This chapter showed what happens when data from real experiments are reconstructed by the particle tracking framework from *(Herzog et al. 2021b) and this reconstructed data is then used as a baseline for training the *(Herzog et al. 2021a) (nODE) and *(Herzog et al. 2019) (CAE+CRF) approach. On the one hand, in both cases, the presented methods provide predictions that are easier to interpret than the interpretation of the raw reconstructed trajectories. On the other hand, the predicted results from the networks do not reflect the underlying data by 100%. This is easy to see when the predicted vector fields in figure 5.3 are compared with real motions from figure 5.2. Nevertheless, the trend in the movement of the amoebae can be recognised well, but their fine, random movements cannot be seen, at all. One point that could be examined in the future is to use the predicted vector fields in figure 5.3 and simulate the motion of the particles directly,

by simply allowing the vector field to act as an external force on the particle and adding some stochastic disturbance for each time step to see if the resulting trajectories are comparable to the real data.

In the case of the RBC data, a similar idealisation effect can be observed, but to a much lesser extent. The predicted trajectories reflect the underlying dynamics very well, which makes it very easy to recognise the LSC.

The presented approach shows great flexibility in terms of possible investigations. One can start trajectories at arbitrary points or increase the density of the trajectories at will and, thus, investigate the behaviour of the system more deeply. It should also be noted that the prediction with the approach of *(Herzog et al. 2019) is very fast, even on my notebook with an Intel Xeon E3-1505M v5 at 2.8 GHz the prediction of the 50000 initial conditions for 2000 time steps took only a few seconds. For these two applications, these hybrid approaches are a good complement to investigate the data and get better ideas about the underlying dynamics. However, in my opinion, the true potential of these methods lies in the combination of the reconstruction method with the hybrid prediction networks. In the last chapter, this idea is taken up again and a conclusion is also presented regarding the use of ANNs and hybrid methods for modelling dynamical systems. This is then followed by an outlook concerning further investigations. Some of these aspects had been discussed in the specific chapters and the outlook provides then an additional overview concerning potential future work.

The trick is the idealizations. [...] This system is quite unlike the case of mathematics, in which everything can be defined, and then we do not know what we are talking about. In fact, the glory of mathematics is that we do not have to say what we are talking about. The glory is that the laws, the arguments, and the logic are independent of what 'it' is.

— Richard P. Feynman (1965)

6

Conclusion and outlook

Contents

6.1 Conclusion	145
6.2 Outlook	147

6.1 Conclusion

As described in the introduction, theory building can begin with the acquisition of experimental data. However, in many cases the data collected do not have a sufficient quality to draw conclusions in order to build a model or even a theory. In the second chapter, the convolutional autoencoder (CAE) has been introduced and examined to see how well it can represent data from dynamical systems. It was shown that the CAE is a suitable architecture for different reconstruction tasks supporting both pre- and post-processing of data. It can compensate measurement errors and limitations and one can obtain this way data that is easier to analyse, with the potential to complete and enhance experimental observations, providing a possible solution to some data acquisition problems. Furthermore, a possibility to increase the performance of the CAEs was presented by adding feedback connections to the CAE.

Another problem that can arise during theory building is the absence of first principles, which makes it hard or impossible to formulate (numerical) models to investigate the spatio-temporal dynamics of systems that cannot be observed experimentally over sufficiently long times periods. In the third chapter, the investigation provided in *(Herzog et al. 2019; Herzog et al. 2018; Herzog et al. 2020b) showed that the introduced hybrid model consisting of a CAE and a CRF is very powerful in learning spatio-temporal dynamics of non-linear systems from data, which has been demonstrated on several cases. It is even so powerful that it outperformed the echo state network (ESN) approach of Pathak et al. (2018), which, to the best of my knowledge, had been the best performing approach given the current state of the art. Thus, this here-introduced hybrid model provides a possible way to predict the dynamics of systems which are difficult or impossible to be modelled by first principles, the ability to predict such systems can allow users to identify patterns and other correlations in their data.

As also described in the introduction, idealisations and abstractions are necessary steps to formulate models leading to theories that are generally valid. But these idealisations can require a high degree of expert knowledge, which can be a problem especially when having very complex data that shall be represented symbolically. In the fourth chapter, the so-called neural ordinary differential equations (nODE) approach introduced by Chen et al. (2018), was slightly extended together with the loss function including a sparse regularisation. This has then been finally applied to predict vector fields of a multi-agent system. We could this way show that even

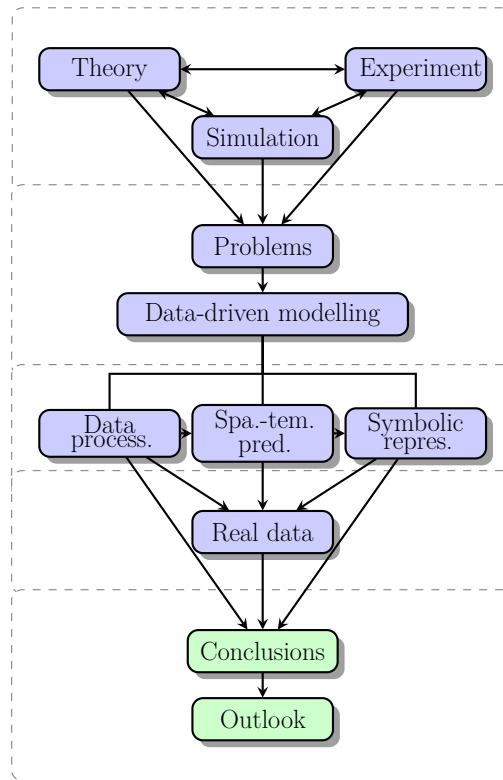


Figure 6.1: Structure overview: Conclusion and outlook chapter 6

data coming from a system that is not described by ordinary differential equations can be well represented by vector fields from the nODE approach. In other words, this approach is suitable for reducing the complexity in data such that the data can be expressed by mathematical symbols.

In short, the methods discussed in chapters 2-4 present methods that can be used at different points in the interplay between theory, experiment and simulation. For reasons of validation and data quality they all have been applied to simulated data. As stated in the introduction, though, these methods are motivated to bring the fields of theory, experiment and simulation closer together. And so, at least by way of examples, it should be demonstrated how these methods can be used on real experimental data. Therefore, Chapter 5 demonstrates how both, the combination of CAE+CRF and the nODE, approaches can be applied to real experimental data. In particular, chapter 5 shows that a synergy of classical with modern evaluation and reconstruction methods works well and leads to results that simplify model and theory building. The hybrid approach presented can itself take the role of a simulation *(Herzog et al. 2019; Herzog et al. 2018; Herzog et al. 2020b) or, alternatively, it can be used within a numerical solver to parameterize the derivative of the hidden states from the assumed equations *(Herzog et al. 2021a). The second case is particularly useful if assumptions are already made about the system under investigation and one wants to check how well these assumptions can describe the data. As shown, application of the presented methods arrive at enriched data, the possibility of predicting non-linear dynamics for longer periods of time and also the possibility of representing complex data using mathematical expressions.

6.2 Outlook

In the second chapter, the convolutional autoencoder (CAE) has been introduced and examined on different reconstruction tasks. A necessity for successful reconstructions is that the CAE is sufficiently well trained. For successful training, there needs to be a sufficient amount of data. Unfortunately, the word sufficient is not quantifiable, the only metric one currently has is given by the behaviour of the loss function

during training, e.g.: *(Herzog et al. 2021c, Figures 9, 13, 16). When the loss function on the training data and validation data converges to a small value and further the evaluation on the test data shows also small errors, it can be assumed that the training was successful. This justifies the assumption that the neural network is a sufficiently good approximator for the function to be approximated. If one of the points mentioned does not apply, this could be understood as an indication that not enough data was available during the training. However, this statement is very weak and only can be understood as a "could" and not a "must". Even if this measure would be a unique indicator (which it is not), it is still a very unsatisfactory measure, because it is not possible to estimate the amount of data needed before training, which can lead to several data acquisition and training phases until satisfactory performance is obtained.

But what if there is not enough data available? Would it be possible to use a mixture of simulated and experimental data? And if so, how should these be regularised? The field of "data augmentation" deals with such questions (Shorten et al. 2019; Cubuk et al. 2019). However, how well this method can be transferred to dynamical systems still needs to be investigated, as there is little or no experience existing about this so far.

Looking next at chapter 3: Here a hybrid model, which is an extension of the CAE with a CRF was introduced for predicting spatio-temporal dynamics of non-linear systems from data. Exceeding the former state of the art echo state network (ESN) approach from Pathak et al. (2018). However, the question why this hybrid model is so powerful can only be answered by speculation at the moment. As stated at the end of chapter 3, it could be that the combination of the CAE with the CRF works so well because the encoding part of the CAE encodes the input data into a coordinate system where the dynamic is more predictable by the CRF than in the original coordinate system. Initial investigations support this assumption. These investigations, however, must be greatly expanded and also performed on more different cases and different parameterizations of the CAEs. In particular, variation of parameters in the latent space layers could have a substantial

influence and should be investigated in the future. Provided that the transfer to a coordinate system, where the dynamics can be described as a linear system, is the decisive point, the question arises whether, for example, an ESN-based system, as in the former state of the art (Pathak et al. 2018), would work better than a CRF. To address this question, an ESN is currently being implemented as part of a master's thesis that can be integrated directly into the auto-encoder framework to allow end-to-end optimisation of all parameters. Along the same lines it will be important to compare CRF- (or ESN)-based performance with a simple regression in the latent space, where the coordinate system is simplified, to contrast the advanced approach with this kind of baseline. What can already be said with a high degree of certainty is that the loss function for calculating the error during training plays a major role. Smarter functions, which possibly represent parts of the physics of the system that is to be modelled, could lead to improved performance. An additional aspect to be considered is the uncertainty estimation of the hybrid method during prediction. A promising method for this would be the approach called *deep evidential regression* by Amini et al. (2020). The idea of *deep evidential regression* is to train additional parameters for distributions expressing a measure of uncertainty in the network. Depending on the complexity to integrate these additional parameters, this could lead to an uncertainty estimation for each time step during the prediction by the hybrid model. If the uncertainty was then found to correlate with the prediction error, one could use this as an indicator of how long one trusts the prediction by the hybrid model. A final point concerns studies on generalisability, where more thought needs to be given on how to proceed here in a structured way. Testing different parameters and trying out when they work and when not is not a sensible way to proceed, as there is little insight to be gained by such a process. My hypothesis based on the initial research in *(Herzog et al. 2020b) would be that the networks do well with initial conditions that are a linear combination of the training data. I would like to investigate this question in a further study and restructure the investigations from *(Herzog et al. 2020b) in more detail, based on the case with multiple cylinders from *(Herzog et al. 2020b).

One could then train the hybrid network with data where, for example, the initial velocity comes from a Gaussian distribution, with expected value and variance set. After successful training, one would generate test data, but then vary the variance and examine the prediction errors as a function of variance. In the second step, one would draw the training data from two Gaussian distributions, so that one has two expected values and variances. The test data would be drawn from the composite distribution of both distributions and the prediction quality would be examined. In the third step, one would generate test data that can no longer be regarded as a linear combination of both distributions and further investigate how the prediction errors develop. In the last step, one would draw test data from a distribution that no longer has any similarity with the distributions of the training data and examine the prediction error as a function of a measure of the difference between two (train and test) probability distributions. Through these four steps, it would be possible to investigate how the prediction performance depends on the distributions from which the training and test dates come, and to what point my hypothesis that the prediction works well for initial conditions that are a linear combination of the training data holds. The first two steps would cover the point of linear combination, while the last two steps would represent the counterhypothesis.

In chapter 4 an extension of the so-called neural ordinary differential equations (nODE) introduced by Chen et al. (2018), was introduced. For this approach to work, it is necessary that the user specifies the structure of the ODEs, e.g., here to represent Hamiltonian mechanics. At the moment, only one user defined structure can be used to train nODEs. However, this severely limits the scope of possible questions that could be addressed by this approach. Wouldn't it also be possible to use a set of ODEs and divide this set into subsets where each ODE receives the data in a different coordinate system (e.g., Cartesian and polar coordinates) and spaces (e.g., linear and polynomial vector spaces), in the hope that even more complex systems can be described in this way? One could try to extend this approach also with a genetic algorithm that searches for the coordinate system that allows for the best approximation of the data. If this works, the next step

would be to combine two such structures and parameterise the combination. All this is only a sketch of a possible extension of this approach, which needs to be made more precise in the future.

The methods discussed in chapters 2-4 present approaches that can be used at different points during the interplay between theory, experiment, and simulation. Chapter 5 was supposed to further support this statement through demonstrations with real data. As a necessary prerequisite, a reconstruction approach *(Herzog et al. 2021b) was presented to reconstruct trajectories from image sequences. These trajectories provide the data required to train both approaches *(Herzog et al. 2019) (CAE+CRF) and *(Herzog et al. 2021a) (nODE), which were then applied to the two experimental cases presented in *(Herzog et al. 2021b). The use of the modified neural ordinary differential equation approach *(Herzog et al. 2021a) led to the prediction of a comparatively simple vector field describing the main motion of some amoebae. In the second case, the hybrid of CAE and CRF, the approach from *(Herzog et al. 2019), was used to predict the particle motion in Rayleigh-Bénard convection. The use of the hybrid approach resulted in the possibility to generate trajectories with much higher densities and lengths than by reconstruction of experimental data. As written at the end of chapter 5: An integration of the hybrid model with the reconstruction framework from *(Herzog et al. 2021b) would be desirable to replace the regression with the Gaussian mixture model in *(Herzog et al. 2021b, section 2.4), with the goal to get a better prediction of the trajectories during reconstruction. This should, hopefully, lead to longer reconstructed tracks and much shorter reconstruction times.

Due to the promising results obtained by applying the presented methods on experimental data, it seems desirable to disseminate more widely these methods into research fields and communities that have less experience with machine learning. This means, on the one hand, that the application of the methods presented should be simplified to be accessible also for researchers who have less experience with the implementation of such methods. Which also means that more efforts and demonstrations are probably needed on how to use these methods in fields such

as biology or chemistry. On the other hand, one needs to think about possible integrations into established frameworks. An example of one of these frameworks is *OpenFoam* (see (Jasak et al. 2007)), an open source framework for computational fluid dynamics, for which there are also preliminary works that can simplify the integration of the methods presented, like (Maulik et al. 2021).

Súmma summárum: This thesis had focused on what we called *hybrid approaches* and we tried to demonstrate that they can act as a link between theory and experiment. One can - as shown - arrive at enriched simulations, improved predictive power and also at model estimates by integrating assumptions about the physics of the underlying system into these approaches. The potential of these approaches, thus, appears high, but - as discussed above - much remains to be done.

Appendices

There must be a trick to the train of thought, a recursive formula. A group of neurons starts working automatically, sometimes without external impulse. It is a kind of iterative process with a growing pattern. It wanders about in the brain, and the way it happens must depend on the memory of similar patterns.

— Stanisław M. Ulam (1991)



Formal definitions for artificial neural networks

Contents

A.1 Artificial neural networks	155
A.1.1 Network structures	156
A.1.2 Multilayer feed-forward network	157
A.1.3 Recursive definition	157
A.1.4 Network state	158
A.1.5 Topological network dynamics	158

This is the appendix for artificial neural networks, it contains formal definitions of the terms, which for the sake of readability are not elaborated in the main text.

A.1 Artificial neural networks

Definition A.1.1 (Artificial neural network). A artificial neural network (ANN) is a tuple $\mathcal{N} = (N, \mathcal{A}, \mathbf{w}, \theta, \sigma, I, O)$ consisting of:

- $N = \{1, \dots, n\}$ nodes
- $\mathcal{A} \subset N \times N$ the network structure
- $\mathbf{w} = (w_{ij})_{i \rightarrow j} (w_{ij} \in \mathbb{R})$ weights

- $\theta = (\theta_i)_{i \in N}$ ($\theta_i \in \mathbb{R}$) biases
- $\sigma = (f_i : \mathbb{R} \rightarrow \mathbb{R})_{i \in N}$ Activation functions. (There are a variety of activation functions, they can be radial function, or a composition of a univariate function with an affine transformation or even higher-order functions. Some examples would be linear activation, identity, relu, sigmoid, tangenshyperbolicus and many more. The choice of the activation function often has an influence on the performance of the model Zaheer et al. 2018.)
- $I \subset N$ Input nodes
- $O \subset N$ Output nodes

A neuronal architecture is a tuple $\mathcal{N}' = (N, \mathcal{A}, \mathbf{w}', \boldsymbol{\theta}', \sigma, I, O)$ as above, but \mathbf{w}' is only defined for some connections $i \rightarrow j$ and $\boldsymbol{\theta}'$ only for some neurons.

A.1.1 Network structures

Definition A.1.2 (Network structures). Feed-forward neural networks (N, \mathcal{A}) are an acyclic graph, I are the neurons without predecessors and O are the neurons without successors. Assuming $I \cap O = \emptyset$, then there always exists a decomposition of the form $N = N_0 \cup \dots \cup N_h$ with $N_i \cap N_j = \emptyset$ for $i \neq j$, $I = N_0$, $O = N_h$

$$\mathcal{A} \subset \bigcup_{i=0}^{h-1} \bigcup_{j=i+1}^h N_i \times N_j$$

Related to such a representation, N_i is called i -th *hidden layer* for $0 < i < h$, N_0 is called *input layer*, N_h is called *output layer*, h denotes the *depth* of the network. Connections in $N_i \times N_j$ with $i < j - 1$ are called *shortcut connections* or *residual connections* He et al. 2016. A (fully connected) *multilayer feed-forward network* is present if additionally

$$\mathcal{A} = \bigcup_{i=1}^h N_{i-1} \times N_i$$

holds. The network structure is represented by the expression (n_0, \dots, n_h) with $n_i = |N_i|$.

A.1.2 Multilayer feed-forward network

As an extension to this a (fully connected) *multilayer feed-forward network* with *shortcut connections* or *residual connections* He et al. 2016 is a feed-forward network with

$$\mathcal{A} = \bigcup_{i=0}^{h-1} \bigcup_{j=i+1}^h N_i \times N_j. \quad (\text{A.1})$$

The network structure is given by the expression (n_0, \dots, n_h) with $n_i = |N_i|$. A network is called (fully) recurrent network if

$$\mathcal{A} = N \times N.$$

For recurrent networks often $I = O = N$ or $I = O$. This can be attenuated to partially recurrent networks where:

$$\mathcal{A} \subset N \times N.$$

A.1.3 Recursive definition

For the sake of completeness, a recursive definition of a *multilayer feed-forward network* with h hidden layers shall also be given, as it is often found in the literature.

If each layer has some input $z^{m+1} \in \mathbb{R}^p$, the the network is defined by

$$y_k^{(m+1)} = \sigma_k^{(m+1)} \left(z^{(m+1)} \right), \quad k = 1, \dots, k_{m+1} \quad (\text{A.2})$$

$$z_k^{(i)} = \sum_{j=1}^{k_{i+1}} w_{jk}^{(i)} y_j^{(i+1)} + b_k^{(i)}, \quad i = 0, \dots, m, k = 1, \dots, k_i \quad (\text{A.3})$$

$$y_k^{(i)} = \sigma_k^{(i)} \left(z_k^{(i)} \right), \quad i = 0, \dots, m, k = 1, \dots, k_i, \quad (\text{A.4})$$

where $w_{jk}^{(i)}$ are the weights, $b_k^{(i)}$ biases and σ the activation functions, for $i = 0, \dots, m$. Equation A.2 defines the transformation of the input vector per layer $z^{(m+1)}$ to some real values $y_k^{(m+1)}$. In equation A.3 the input values are calculated as a weighted linear combination of the previous output values, where ultimately the activation functions in equation A.4 map the input to the output values for every layer and every neuron.

A.1.4 Network state

Definition A.1.3 (Network state). The network state of a network \mathcal{N} is a tuple $\mathbf{o} = (o_i)_{i \in N}$ with $o_i \in \mathbb{R}$. An activation of \mathcal{N} is a sequence of states $\mathbf{o}(t)_{t \in \mathbb{N}}$. The activation of node i at time t is

$$\text{net}_i(t) = \sum_{j \rightarrow i} w_{ji} o_j(t) - \theta_i.$$

In general, the dynamics of a network can be divided into four classes. Starting with the initial state $\mathbf{o}(0)$ and some inputs, it is possible to calculate an activation which defines the functionality of the network.

A.1.5 Topological network dynamics

1. **Topological dynamics in feed-forward networks:** Let $N = N_0 \cup \dots \cup N_h$ be a decomposition for $\mathbf{o}(0) \in \mathbb{R}^{|N|}$, then set

$$o_i(t+1) = \begin{cases} o_i(t) & \text{if } i \in N_j \text{ with } j \leq t \\ f_i(\text{net}_i(t)) & \text{else.} \end{cases}$$

The network \mathcal{N} calculates the function $f: \mathbb{R}^{|I|} \rightarrow \mathbb{R}^{|O|}$, $f(\mathbf{x}) = (o_{i_j}(h))_{i_j \in O}$, where $o_{i_j}(0) = x_j$ for $i_j \in I$ and $o_{i_j}(0) = 0$ else. The function f is independent of the decomposition.

2. **Synchronous dynamics in recurrent networks:**

$$o_i(t+1) = f_i(\text{net}_i(t))$$

$\forall i \in N, t \in \mathbb{N}$.

3. **Asynchronous dynamics in recurrent networks:**

$$o_i(t+1) = \begin{cases} f_i(\text{net}_i(t)) & i = i(t) \\ o_i(t) & \text{else} \end{cases}$$

where $i(t)$ is an arbitrary node. In this and the previous case one starts with a given vector $\mathbf{o}(0) \in \mathbb{R}^{|N|}$.

4. **Recurrent switching dynamics in partially recurrent networks:** $(\mathbb{R}^{|I|})^*$ is the set of finite sequences with elements in $\mathbb{R}^{|I|}$, denoted as $[\mathbf{x}^0, \dots, \mathbf{x}^T]$ (where $T+1$ is the sequence length). For some given $\mathbf{y} \in \mathbb{R}^{|N|-|I|}$, the so-called *starting context* and a sequence of length T define

$$o_i(t) = \begin{cases} x_j^t & i = i_j \in I, t \leq T-1 \\ y_j & i = i_j \notin I, t = 0 \\ f_i(\text{net}_i(t-1)) & i \notin I, 0 < t \leq T \\ o_i(t-1) & \text{else.} \end{cases}$$

One can define a function

$$f_{\mathbf{y}} : (\mathbb{R}^{|I|})^* \rightarrow \mathbb{R}^{|O|} \text{ as } f_{\mathbf{y}}([\mathbf{x}^0, \dots, \mathbf{x}^T]) = (o_{i_j}(T+1))_{i_j \in O}.$$

The most important questions of life are, for the most part, really only problems of probability.

— Pierre S. de Laplace, found in (Laplace 1820)

B

Theoretical background for graphical models

Contents

B.1 Random vector	161
B.1.1 Joint distribution	162
B.1.2 Density	162
B.1.3 Multivariate random variable	162
B.2 Stochastic independence	164
B.3 Graph	164
B.3.1 (Un)directed graph	165
B.4 Gaussian process	165

This is the appendix to theoretical foundations for graphical models, the definitions given here correspond to the classical foundations of stochastics, which are not elaborated in the main text for the sake of readability.

B.1 Random vector

Definition B.1.1 (Random vector). Let $(\Omega, \mathfrak{A}, P)$ be a probability space. A mapping $\mathbf{X} = (X_1, \dots, X_n)^T : \Omega \rightarrow \mathbb{R}^n$ is called a *random vector* if X_1, \dots, X_n are random variables.

A *random vector* is composed of n random variables on the same probability tree.

For example, considering a random signal $Re^{i\phi}$, where R denotes the amplitude and ϕ the phase. Then $\mathbf{X} = (R, \phi)^T$ is a two-dimensional *random vector*. A assignment to \mathbf{X} is denoted by $\mathbf{x} = (x_1, \dots, x_n)$. The next important definition is the *joint distribution*.

B.1.1 Joint distribution

Definition B.1.2 (Joint distribution). Let $\mathbf{X} = (X_1, \dots, X_n)^T$ be a *random vector*. The function

$$F_{\mathbf{X}}(\mathbf{x}) = P(X_1 \leq x_1, \dots, X_n \leq x_n)$$

is called joint distribution function of $(X_1, \dots, X_n)^T$ or distribution function of \mathbf{X} , denoted as $\mathbf{X} \sim F_{\mathbf{X}}$.

The joint distribution function uniquely describes the distribution of the *random vector* \mathbf{X} , i.e., the probabilities $P(\mathbf{X} \in A)$ for all measurable events $A \subseteq \mathbb{R}^n$.

B.1.2 Density

Definition B.1.3 (Density). A (improperly Riemannian) integrable function $f_{\mathbf{X}} : \mathbb{R}^n \mapsto \mathbb{R}_+$ is called the *density* of \mathbf{X} (or $F_{\mathbf{X}}$, the distribution function of \mathbf{X}) if

$$F_{\mathbf{X}}(\mathbf{x}) = \int_{\mathbf{x}} f_{\mathbf{X}}(\mathbf{t}) \, d\mathbf{t}$$

$\forall x_1, \dots, x_n \in \mathbb{R}$, where \mathbf{X} (or $F_{\mathbf{X}}$) is then called absolutely-continuous with density $f_{\mathbf{X}}$, in short: $\mathbf{X} \sim f_{\mathbf{X}}$.

This allows the individual probabilities to be combined into complex distributions and multivariate random variables.

B.1.3 Multivariate random variable

Definition B.1.4 (Multivariate random variable). For a *random vector* $\mathbf{X} = (X_1, \dots, X_n)^T$ means

(a) $\mathbb{E}(\mathbf{X}) = (\mathbb{E}(X_1), \dots, \mathbb{E}(X_n))^T \in \mathbb{R}^n$ of \mathbf{X} .

- (b) The symmetric matrix $C(\mathbf{X}) = (\text{Cov}(X_i, X_j))_{1 \leq i, j \leq n} \in \mathbb{R}^{n \times n}$ covariance matrix of \mathbf{X} (matrix of pairwise covariances of X_i and X_j).

Proposition B.1.1 (Expected value and covariance under linear transformations).

Let $\mathbf{X} = (X_1, \dots, X_n)^T$ be a *random vector* with $\mathbb{E}(\mathbf{X}) = \boldsymbol{\mu} \in \mathbb{R}^n$, $\text{Cov}(\mathbf{X}) \in \mathbb{R}^{n \times n}$ and $\mathbf{A} \in \mathbb{R}^{m \times n}$, $\mathbf{b} \in \mathbb{R}^m$ arbitrarily. Then

- (a) $\mathbb{E}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A}\mathbb{E}(\mathbf{X}) + \mathbf{b} \in \mathbb{R}^m$
- (b) $\text{Cov}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A} \text{Cov}(\mathbf{X}) \mathbf{A}^T \in \mathbb{R}^{m \times m}$

Proof. (a) $\mathbf{Y} = \mathbf{A}\mathbf{X} + \mathbf{b}$ means component wise $Y_i = \sum_{k=1}^n a_{ik}X_k + b_i$ for $i = 1, \dots, m$. Because of the linearity of the expected value

$$\mathbb{E}(Y_i) = \sum_{k=1}^n a_{ik} \mathbb{E}(X_k) + b_i, \quad \forall i = 1, \dots, m.$$

So

$$\mathbb{E}(\mathbf{Y}) = \mathbf{A}\mathbb{E}(\mathbf{X}) + \mathbf{b}.$$

(b)

$$\begin{aligned} \text{Cov}(Y_i, Y_j) &= \text{Cov}\left(\sum_{k=1}^n a_{ik}X_k + b_i, \sum_{l=1}^n a_{jl}X_l + b_j\right) \\ &= \sum_{k=1}^n \sum_{l=1}^n a_{ik}a_{jl} \text{Cov}(X_k, X_l) \\ &= \left(\mathbf{A} \text{Cov}(\mathbf{X}) \mathbf{A}^T\right)_{i,j} \end{aligned}$$

$\forall 1 \leq i, j \leq m$. In total

$$\text{Cov}(\mathbf{A}\mathbf{X} + \mathbf{b}) = \mathbf{A} \text{Cov}(\mathbf{X}) \mathbf{A}^T.$$

□

For many approaches from stochastic modelling, the assumption of stochastic independence is made.

B.2 Stochastic independence

Definition B.2.1 (Stochastic independence). Let $\mathbf{X} = (X_1, \dots, X_n)^T$ be a *random vector*. X_1, \dots, X_n are called stochastically independent (s.i.) if

$$F_{\mathbf{X}}(\mathbf{x}) = F_{X_1}(x_1) \cdot \dots \cdot F_{X_n}(x_n) \quad \forall x_1, \dots, x_n \in \mathbb{R}.$$

This property is elementary, especially for the formation of the conditional distributions. However, it should also be noted that the independence assumptions are often very inaccurate, which does not necessarily go hand in hand with poor performance of the methods.

In the following, transformations $T(\mathbf{X}) \in \mathbb{R}^n$ of an absolutely continuous *random vector* $\mathbf{X} = (X_1, \dots, X_n)^T$ with density $f_{\mathbf{X}}(\mathbf{x})$ are considered. It is assumed that

$$M = \{\mathbf{x} \in \mathbb{R}^n \mid f_{\mathbf{X}}(\mathbf{x}) > 0\} \subseteq \mathbb{R}^n$$

is an open set and that $T : M \rightarrow \mathbb{R}^n$ is an injective mapping with

$$\left| \left(\frac{\partial T_i(x_1, \dots, x_n)}{\partial x_j} \right)_{1 \leq i, j \leq n} \right| > 0, \quad \forall (x_1, \dots, x_n) \in M$$

i.e. the absolute value of the determinant of the Jacobi matrix is positive.

B.3 Graph

Definition B.3.1 (Graph). A graph $G = (\mathcal{V}, \mathcal{E})$ is a tuple of a set of nodes or vertices, $\mathcal{V} = \{1, \dots, V\}$, and a set of edges, $\mathcal{E} = \{(s, t) : s, t \in \mathcal{V}\}$.

Graphs can be represented by adjacency matrices, in which $G(s, t) = 1$ denote $(s, t) \in \mathcal{E}$, that is, if $s \rightarrow t$ is an edge in the graph. Based on this a graphical model can be defined as follows.

B.3.1 (Un)directed graph

GMs can be distinguished by nature of the underlying *graph*.

Definition B.3.2 ((Un)directed graph). A graph is called undirected if $G(a, b) = 1$ and $G(b, a) = 1$, where a, b are some nodes, otherwise it is called directed.

This transfers identically to the GMs, an illustration is provided in figure B.1.



Figure B.1: Example of: a) an undirected and b) directed graph, based on definition B.3.2.

B.4 Gaussian process

Gaussian process are, like ANNs, very powerful tools to represent temporal and spatial functions. They are also very well suited for modelling functions that involve uncertainties due to incomplete information.

Definition B.4.1 (Gaussian process). A stochastic process $X(t) \in \mathbb{R}, t \in I$ is called a *Gaussian process* if all linear combinations in the form

$$Y_{(\alpha_1, \dots, \alpha_k)}^{(t_1, \dots, t_k)} = \sum_{i=1}^k \alpha_i X(t_i),$$

$k \in \mathbb{N}, t_1, \dots, t_k \in I, \alpha_1, \dots, \alpha_k \in \mathbb{R}$, are normally distributed. In the case $|I| < \infty$, $X(t), t \in I$ is also called a Gaussian vector.

Definition B.4.2 (Covariance function). Let $X(t), t \in I$ be a stochastic process with $\mathbb{E}X^2(t) < \infty$ for all $t \in I$. Let $\mu(t) = \mathbb{E}X(t)$. Then the function is called

$$r : I \times I \rightarrow \mathbb{R}$$

defined by

$$r(s, t) = \mathbb{E}((X(s) - \mu(s))(X(t) - \mu(t))), \quad s, t \in I$$

Covariance function of $X(t), t \in I$.

Theorem B.4.1. Let r be the covariance function of the process $X(t), t \in I$. Then r is positive semidefinite, such that:

$$\sum_{i,j=1}^k z_i r(t_i, t_j) z_j$$

is real and non-negative for all $k \in \mathbb{N}, z = (z_1, \dots, z_k) \in \mathbb{R}^k, t_1, \dots, t_k \in I$.

Proof.

$$\begin{aligned} \sum_{i,j=1}^k z_i r(t_i, t_j) z_j &= \sum_{i,j=1}^k z_i \mathbb{E}((X(t_i) - \mu(t_i))(X(t_j) - \mu(t_j))) z_j \\ &= \mathbb{E} \left(\left(\sum_{i=1}^k z_i (X(t_i) - \mu(t_i)) \right) \left(\sum_{j=1}^k z_j (X(t_j) - \mu(t_j)) \right) \right) \\ &= \mathbb{E} \left(\left| \sum_{i=1}^k z_i (X(t_i) - \mu(t_i)) \right|^2 \right) \geq 0 \end{aligned}$$

□

Theorem B.4.2. (a) A *Gaussian process* has finite second moments, i.e. $\mathbb{E}X^2(t) < \infty$ for all $t \in I$

(b) If $X(t), t \in I$ is a *Gaussian process*, $t_1, t_2, \dots, t_k \in I$,

$$\mu(t) = \mathbb{E}X(t), t \in I, X - \mu = (X(t_1) - \mu(t_1), \dots, X(t_k) - \mu(t_k))^T$$

and

$$\Gamma = \text{cov}(X(t_1), \dots, X(t_k)) = \mathbb{E}((X - \mu)(X - \mu)^T)$$

is invertible, then the distribution of $(X(t_1), \dots, X(t_k))$ has a density given by

$$f_{t_1, \dots, t_k}(x) = (2\pi)^{-k/2} (\det \Gamma)^{-1/2} \exp \left(-\frac{1}{2} (x - \mu)^T \Gamma^{-1} (x - \mu) \right)$$

where $x = (x_1, \dots, x_k)^T, \mu = (\mu(t_1), \dots, \mu(t_k))^T$.

- (c) The distribution of a *Gaussian process* is uniquely determined by its expected value function $\mathbb{E}X(t) = \mu(t)$ and covariance function.

Theorem B.4.3. Let $m \in \mathbb{N}$, $\mu \in \mathbb{R}^m$ and $\Sigma \in \mathbb{R}^{m \times m}$ be positive semidefinite. Let $\text{rg}(\Sigma) = k$ and $n \geq \max(k, 1)$. Then there exists an $A \in \mathbb{R}^{m \times n}$, such that $\Sigma = AA^T$. If Y_1, \dots, Y_n are independently $\mathcal{N}(0, 1)$ -distributed ZVn , then

$$X = AY + \mu$$

$\mathcal{N}(\mu, \Sigma)$ -distributed.

Proof. 1. X is a Gaussian vector, since all linear combinations of the components of X are simultaneously linear combinations of the components of Y plus a constant and thus normally distributed.

2.

$$\mathbb{E}X = \mathbb{E}(AY + \mu) = A\mathbb{E}Y + \mu = \mu.$$

3.

$$\begin{aligned} \text{cov } X &= \mathbb{E} \left((X - \mu)(X - \mu)^T \right) = \mathbb{E} \left((AY)(AY)^T \right) = \mathbb{E} \left(AYY^T A^T \right) \\ &= A\mathbb{E} \left(YY^T \right) A^T = AA^T = \Sigma \end{aligned}$$

□

Theorem B.4.4. Let I be a non-empty set, $\mu : I \rightarrow \mathbb{R}$ arbitrary and $r : I \times I \rightarrow \mathbb{R}$ positive semidefinite. Then there exists (with respect to distribution) exactly one *Gaussian process* with expected value function μ and covariance function r

Proof. Define the distributions

$$P_{t_1, \dots, t_k} = \mathcal{N} \left(\begin{pmatrix} \mu(t_1) \\ \vdots \\ \mu(t_k) \end{pmatrix}, r(t_i, t_j)_{i, j=1, \dots, k} \right)$$

for $k \in \mathbb{N}$, $t_1, \dots, t_k \in I$. Since the family of these measures P_{t_1, \dots, t_k} is consistent. The unique distribution of a *Gaussian process* with the prescribed expected values and covariances can be derived. □

Theorem B.4.5. Let $X(t), t \in I$ be a *Gaussian process*. Then equivalently:

- (a) $X(t), t \in I$ are independent (i.e. $\{X(t_1), \dots, X(t_k)\}$ are independent for all $k \in \mathbb{N}, t_1, t_2, \dots, t_k \in I$ pairwise different).
- (b) $X(t), t \in I$ are pairwise orthogonal, i.e. $\mathbb{E}((X(t) - \mu(t))(X(u) - \mu(u))) = 0$ for all $t, u \in I, t \neq u$

Proof. (a) to (b) Generally applies to processes with finite second moments.

- (b) to (a) Let $t_1, \dots, t_k \in I$ be pairwise different. If $X(t_1), \dots, X(t_k)$ are pairwise uncorrelated, holds

$$\begin{pmatrix} X(t_1) \\ \vdots \\ X(t_k) \end{pmatrix} \sim \mathcal{N} \left(\begin{pmatrix} \mu(t_1) \\ \vdots \\ \mu(t_k) \end{pmatrix}, \begin{pmatrix} \sigma^2(t_1) & & 0 \\ & \ddots & \\ 0 & & \sigma^2(t_k) \end{pmatrix} \right)$$

where $\sigma^2(t_i) = \text{var}(X(t_i))$. By theorem B.4.3, it holds that for independent $\mathcal{N}(0, 1)$ -distributed Y_1, \dots, Y_k

$$\begin{pmatrix} \tilde{X}(t_1) \\ \vdots \\ \tilde{X}(t_k) \end{pmatrix} = \begin{pmatrix} \mu(t_1) \\ \vdots \\ \mu(t_k) \end{pmatrix} + \begin{pmatrix} \sigma(t_1) & & 0 \\ & \ddots & \\ 0 & & \sigma(t_k) \end{pmatrix} \begin{pmatrix} Y_1 \\ \vdots \\ Y_k \end{pmatrix}$$

has the same distribution as $(X(t_1), \dots, X(t_k))^T$, so $(\tilde{X}(t_1), \dots, \tilde{X}(t_k))$ are pairwise uncorrelated. But since $\tilde{X}(t_i) = \mu(t_i) + \sigma(t_i) Y_i$ for $i = 1, \dots, k$ as functions of independent random variables are also independent, the same is true for $X(t_1), \dots, X(t_k)$

□

Da steh' ich nun, ich armer Tor,
Und bin so klug als wie zuvor! . . .

— Johann Wolfgang von Goethe, found in
(von Goethe 1808)

References

- Abadi, M. et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org.
- Aigner, M. and Ziegler, G. M. (2004). “Buffon’s needle problem”. In: *Proofs from THE BOOK*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 133–136.
- Alex, J., ed. (2000). *Konrad Zuse: der Vater des Computers*. Fulda: Parzeller.
- AlMomani, A. A. R., Sun, J., and Bollt, E. (2020). “How entropic regression beats the outliers problem in nonlinear system identification”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.1, p. 013107.
- Amini, A. et al. (2020). “Deep evidential regression”. In: *Advances in Neural Information Processing Systems* 33.
- Argyris, J. and Scharpf, D. (1969). “Finite elements in time and space”. In: *Nuclear Engineering and Design* 10.4, pp. 456–464.
- Barth, A., Schwab, C., and Zollinger, N. (2011). “Multi-level Monte Carlo Finite Element Method for Elliptic PDEs with Stochastic Coefficients.” In: *Numerische Mathematik* 119, pp. 123–161.
- Box, G. E. P. (1976). “Science and Statistics”. In: *Journal of the American Statistical Association* 71.356, pp. 791–799.
- Burt, E. (2003). *The metaphysical foundations of modern science*. Mineola, New York: Dover Publications.
- Champion, K. et al. (2019). “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116.45, pp. 22445–22451.
- Chen, R. T. Q. et al. (2018). “Neural Ordinary Differential Equations”. In: *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. NIPS’18. Montréal, Canada: Curran Associates Inc., pp. 6572–6583.
- Cubuk, E. D. et al. (2019). “Autoaugment: Learning augmentation strategies from data”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 113–123.
- Eckhardt, R. (1987). “Stan ulam, john von neumann, and the monte carlo method”. In: *Los Alamos Science* 15, pp. 131–136.
- Einstein, A. (1921). *Geometrie und Erfahrung Erweiterte Fassung des Festvortrages Gehalten an der Preussischen Akademie der Wissenschaften zu Berlin am 27. Januar 1921*. Berlin, Heidelberg: Springer Berlin Heidelberg.
- Feynman, R. (1965). *The Feynman lectures on physics*. New York: Basic Books, a member of the Perseus Books Group.
- Friedman, J. (1963). “Alonzo Church. Application of recursive arithmetic to the problem of circuit synthesis Summaries of talks presented at the Summer Institute for Symbolic Logic Cornell University, 1957, 2nd edn., Communications Research Division, Institute for Defense Analyses, Princeton, N. J., 1960, pp. 3–50. 3a-45a.” In: *Journal of Symbolic Logic* 28.4, pp. 289–290.

- Galilei, G. (1623). *The Assayer (Italian: Il Saggiatore)*.
- Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press.
- Grefenstette, E. et al. (June 2014). “A Deep Architecture for Semantic Parsing”. In: *Proceedings of the ACL 2014 Workshop on Semantic Parsing*. Baltimore, MD: Association for Computational Linguistics, pp. 22–27.
- Guo, X., Li, W., and Iorio, F. (2016). “Convolutional Neural Networks for Steady Flow Approximation”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '16. San Francisco, California, USA: Association for Computing Machinery, pp. 481–490.
- Haigh, T., Priestley, M., and Rope, C. (2016). *ENIAC in Action: Making and Remaking the Modern Computer*. The MIT Press.
- He, K. et al. (2016). “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778.
- Herzog, S., Tetzlaff, C., and Wörgötter, F. (2020a). “Evolving artificial neural networks with feedback”. In: *Neural Networks : the official journal of the International Neural Network Society* 123, pp. 153–162.
- Herzog, S. and Wagner, C. (2020b). “Development of Artificial Neural Networks with Integrated Conditional Random Fields Capable of Predicting Non-linear Dynamics of the Flow Around Cylinders”. In: *New Results in Numerical and Experimental Fluid Mechanics XII*. Cham: Springer International Publishing, pp. 71–79.
- Herzog, S. and Wörgötter, F. (2021a). “Application of neural ordinary differential equations to the prediction of multi-agent systems”. accepted for SWARM 2021 (to be considered for full publication in Artificial Life and Robotics).
- Herzog, S., Wörgötter, F., and Parlitz, U. (2018). “Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media”. In: *Frontiers in Applied Mathematics and Statistics* 4.
- Herzog, S., Wörgötter, F., and Parlitz, U. (2019). “Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos”. In: *Chaos (Woodbury, N.Y.)* 29.12.
- Herzog, S. et al. (2021b). “A probabilistic particle tracking framework for guided and Brownian motion systems with high particle densities”. submitted to Int J Comput Vis.
- Herzog, S. et al. (2021c). “Reconstructing Complex Cardiac Excitation Waves From Incomplete Data Using Echo State Networks and Convolutional Autoencoders”. In: *Frontiers in Applied Mathematics and Statistics*. accepted version 2020/12/07 - shown in this work, published version 2021/03/18 - available online: <https://www.frontiersin.org/articles/10.3389/fams.2020.616584/full>.
- Hinton, G. E., Krizhevsky, A., and Wang, S. D. (2011). “Transforming Auto-Encoders”. In: *Artificial Neural Networks and Machine Learning - ICANN 2011*. Ed. by T. Honkela et al. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 44–51.
- Iqbal, H. (Dec. 2018). *HarisIqbal88/PlotNeuralNet v1.0.0*. Version v1.0.0.
- Ivanovska, T. et al. (Oct. 2019). “A deep learning framework for efficient analysis of breast volume and fibroglandular tissue using MR data with strong artifacts”. In: *International Journal of Computer Assisted Radiology and Surgery* 14.10, pp. 1627–1633.

- Jasak, H., Ar Jemcov, A., et al. (2007). “OpenFOAM: A C++ Library for Complex Physics Simulations”. In: *International Workshop on Coupled Methods in Numerical Dynamics, IUC*, pp. 1–20.
- Kingma, D. P. and Welling, M. (2019). “An Introduction to Variational Autoencoders”. In: *Foundations and Trends in Machine Learning* 12.4, pp. 307–392.
- Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: Principles and techniques*. Adaptive computation and machine learning. Cambridge, Mass.: MIT Press.
- Koopman, B. O. (1931). “Hamiltonian Systems and Transformation in Hilbert Space”. In: *Proceedings of the National Academy of Sciences* 17.5, pp. 315–318.
- Kramer, M. A. (1991). “Nonlinear principal component analysis using autoassociative neural networks”. In: *AIChE Journal* 37.2, pp. 233–243.
- Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). “ImageNet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., pp. 1097–1105.
- Laplace, P. S. (1820). *Théorie analytique des probabilités*. Courcier.
- LeCun, Y. et al. (Dec. 1989). “Backpropagation Applied to Handwritten Zip Code Recognition”. In: *Neural Comput.* 1.4, pp. 541–551.
- LeCun, Y. et al. (1998). “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.
- Maulik, R. et al. (2021). “Deploying deep learning in OpenFOAM with TensorFlow”. In: *AIAA Scitech 2021 Forum*, p. 1485.
- Michalski, R. (Aug. 2002). “Understanding The Nature Of Learning: Issues And Research Directions”. In: *Machine Learning: An Artificial Intelligence Approach* Vol. II.
- Mitchell, T. M. (1982). “Generalization as search”. In: *Artificial intelligence* 18.2, pp. 203–226.
- Moore, G. (2006). “Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff.” In: *IEEE Solid-State Circuits Society Newsletter* 11.3, pp. 33–35.
- Musil, J. et al. (2019). *Towards Sustainable Architecture: 3D Convolutional Neural Networks for Computational Fluid Dynamics Simulation and Reverse Design Workflow*.
- Pallares, J. et al. (Aug. 2001). “Experimental laminar Rayleigh-Bénard convection in a cubical cavity at moderate Rayleigh and Prandtl numbers”. In: *Experiments in Fluids* 31.2, pp. 208–218.
- Paszke, A. et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems* 32. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035.
- Pathak, J. et al. (Jan. 2018). “Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach”. In: *Phys. Rev. Lett.* 120 (2), p. 024102.
- Pillepich, A. et al. (2019). “First results from the TNG50 simulation: the evolution of stellar and gaseous discs across cosmic time”. In: *Monthly Notices of the Royal Astronomical Society* 490.3, pp. 3196–3233.
- Ribeiro, M., Rehman A. and Ahmed, S., and Dengel, A. (2020). *DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks*.
- Shalin, V. et al. (1988). “A formal analysis of machine learning systems for knowledge acquisition”. In: *International Journal of Man-Machine Studies* 29.4, pp. 429–446.

- Shorten, C. and Khoshgoftaar, T. M. (2019). “A survey on image data augmentation for deep learning”. In: *Journal of Big Data* 6.1, p. 60.
- Shuman, F. G. (1989). “History of numerical weather prediction at the National Meteorological Center”. In: *Weather and Forecasting* 4.3, pp. 286–296.
- Simon, H. A. (1983). “2 - Why Should Machines Learn?” In: *Machine Learning*. Ed. by R. S. Michalski, J. G. Carbonell, and T. M. Mitchell. San Francisco (CA): Morgan Kaufmann, pp. 25–37.
- Sutton, C. and McCallum, A. (Apr. 2012). “An Introduction to Conditional Random Fields”. In: *Found. Trends Mach. Learn.* 4.4, pp. 267–373.
- Tribus, M. and McIrvine, E. (1971a). “Energy and Information”. In: *Scientific American* 225.3, pp. 179–190.
- Tribus, M. and McIrvine, E. (1971b). “Energy and Information”. In: *Scientific American* 225.3, pp. 179–190.
- Turing, A., Dotzler, B., and Kittler, F. (1997). *Intelligence Service: Schriften*. ger. Nachdr. OCLC: 246060405. Berlin: Brinkmann & Bose.
- Ulam, S. (1991). *Adventures of a Mathematician*. University of California Press.
- Vincent, P. et al. (Dec. 2010). “Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion”. In: *J. Mach. Learn. Res.* 11, pp. 3371–3408.
- Von Goethe, J. W. (1808). *Johann Wolfgang Goethe 'Faust', Der Tragödie Erster Teil*.
- Wan, Z. Y. and Sapsis, T. P. (Apr. 2017). “Reduced-space Gaussian Process Regression for data-driven probabilistic forecast of chaotic dynamical systems”. In: *Physica D: Nonlinear Phenomena* 345, pp. 40–55.
- Yamashita, R. et al. (Aug. 2018). “Convolutional neural networks: an overview and application in radiology”. In: *Insights into Imaging* 9.4, pp. 611–629.
- Zaheer, R. and Shaziya, H. (2018). “GPU-based empirical evaluation of activation functions in convolutional neural networks”. In: *2018 2nd International Conference on Inventive Systems and Control (ICISC)*, pp. 769–773.



Statement of individual contributions

In the following, a list of my contributions according to each publication is given:

1. Herzog, S., Zimmermann, R. S., Abele, J., and Parlitz, U. (2021c). “Reconstructing Complex Cardiac Excitation Waves From Incomplete Data Using Echo State Networks and Convolutional Autoencoders”. In: *Frontiers in Applied Mathematics and Statistics*. accepted version 2020/12/07 - shown in this work, published version 2021/03/18 - available online: <https://www.frontiersin.org/articles/10.3389/fams.2020.616584/full>

Ideas, implementation of the CAE, simulation of the BOCF model and generation of the BOCF cases, analytic, part of figures, part of writing and part of revision.

about **60%** contribution

2. Herzog, S., Tetzlaff, C., and Wörgötter, F. (2020a). “Evolving artificial neural networks with feedback”. In: *Neural Networks : the official journal of the International Neural Network Society* 123, pp. 153–162

Ideas, implementations, analytic, part of figures, part of writing, part of revision and part publication procedure

about **80%** contribution

3. Herzog, S., Wörgötter, F., and Parlitz, U. (2019). “Convolutional autoencoder and conditional random fields hybrid for predicting spatial-temporal chaos”. In: *Chaos (Woodbury, N.Y.)* 29.12

Ideas, implementations, analytic, figures, part of writing and part of revision

about **80%** contribution

4. Herzog, S., Wörgötter, F., and Parlitz, U. (2018). “Data-Driven Modeling and Prediction of Complex Spatio-Temporal Dynamics in Excitable Media”. In: *Frontiers in Applied Mathematics and Statistics* 4

Ideas, implementations, analytic, figures, part of writing and part of revision

about **85%** contribution

5. Herzog, S. and Wagner, C. (2020b). “Development of Artificial Neural Networks with Integrated Conditional Random Fields Capable of Predicting Non-linear Dynamics of the Flow Around Cylinders”. In: *New Results in Numerical and Experimental Fluid Mechanics XII*. Cham: Springer International Publishing, pp. 71–79

Ideas, implementations, analytic, figures, part of writing, revision and publication procedure

about **90%** contribution

6. Herzog, S., Schiepel, D., Guido, I., and Wagner, C. (2021b). “A probabilistic particle tracking framework for guided and Brownian motion systems with high particle densities”. submitted to *Int J Comput Vis*

Ideas, implementations, analytic, part of figures, part of writing and publication procedure

about **80%** contribution

7. Herzog, S. and Wörgötter, F. (2021a). “Application of neural ordinary differential equations to the prediction of multi-agent systems”. accepted for SWARM 2021 (to be considered for full publication in *Artificial Life and Robotics*)

Ideas, implementations, analytic, figures, part of writing and publication procedure

about **90%** contribution

The other parts of this thesis are subjected to the official doctoral student's declaration of the Georg-August University Göttingen (copy in the following):

"[...] I hereby declare that:

1. the opportunity to work on the aforementioned doctoral thesis project was not arranged commercially. Especially, I did not engage any organization which searches for doctoral thesis supervisors or which will entirely or partly carry out my examination duties against payment;
2. I have only accepted and will only accept the assistance of third parties in so far as it is scientifically justifiable and acceptable in regards to the examination regulations. Especially, all parts of the dissertation will be written by myself; I have not accepted and will not accept impermissible help from other parties neither for money nor for free;
3. I will observe the Statute of the Georg-August-University Göttingen for ensuring good scientific practice;
4. I have not applied for corresponding doctoral degree procedures at any other university in Germany or abroad; the submitted doctoral thesis or parts thereof were not used in another doctoral degree procedure.

I am aware that incorrect information precludes the admission to doctoral studies or will later on lead to the discontinuation of the doctoral degree procedures or to the revocation of the doctoral degree."

Göttingen, 29.01.2021

Place, Date

Sebastian Herzog