Action-oriented Scene Understanding

Dissertation

zur Erlangung des Mathematisch-Naturwissenschaftlichen Doktorgrades
"Doktor rerum naturalium"
der Georg-August-Universität Göttingen

vorgelegt von

Timo Julian Lüddecke

aus Wolfsburg

Göttingen, 2019

Referent
Prof. Dr. Florentin Wörgötter
Georg-August-Universität Göttingen

Koreferentin
Prof. Dr. Caroline Sporleder
Georg-August-Universität Göttingen

Weitere Referenten
Prof. Dr. Justus Piater
Leopold-Franzens-Universität Innsbruck

Prof. Dr. Ramin Yahyapour
Georg-August-Universität Göttingen

Prof. Dr. Eckart Modrow
Georg-August-Universität Göttingen

Prof. Dr. Carsten Damm
Georg-August-Universität Göttingen

Tag der mündlichen Prüfung: 21. August 2019

Abstract

In order to allow robots to act autonomously it is crucial that they do not only describe their environment accurately but also identify how to interact with their surroundings. While we witnessed tremendous progress in descriptive computer vision, approaches that explicitly target action are scarcer. This cumulative dissertation approaches the goal of interpreting visual scenes "in the wild" with respect to actions implied by the scene. We call this approach *action-oriented scene understanding*. It involves identifying and judging opportunities for interaction with constituents of the scene (e.g. objects and their parts) as well as understanding object functions and how interactions will impact the future. All of these aspects are addressed on three levels of abstraction: elements, perception and reasoning.

On the elementary level, we investigate semantic and functional grouping of objects by analyzing annotated natural image scenes. We compare object label-based and visual context definitions with respect to their suitability for generating meaningful object class representations. Our findings suggest that representations generated from visual context are on-par in terms of semantic quality with those generated from large quantities of text.

The perceptive level concerns action identification. We propose a system to identify possible interactions for robots and humans with the environment (affordances) on a pixel level using state-of-the-art machine learning methods. Pixel-wise part annotations of images are transformed into 12 affordance maps. Using these maps, a convolutional neural network is trained to densely predict affordance maps from unknown RGB images. In contrast to previous work, this approach operates exclusively on RGB images during both, training and testing, and yet achieves state-of-the-art performance.

At the reasoning level, we extend the question from asking what actions are *possible* to what actions are *plausible*. For this, we gathered a dataset of household images associated with human ratings of the likelihoods of eight different actions. Based on the judgement provided by the human raters, we train convolutional neural networks to generate plausibility scores from unseen images. Furthermore, having considered only static scenes previously in this thesis, we propose a system that takes video input and predicts plausible future actions. Since this requires careful identification of relevant features in the video sequence, we analyze this particular aspect in detail using a synthetic dataset for several state-of-the-art video models. We identify feature learning as a major obstacle for anticipation in natural video data.

The presented projects analyze the role of action in scene understanding from various angles and in multiple settings while highlighting the advantages of assuming an action-oriented perspective. We conclude that action-oriented scene understanding can augment classic computer vision in many real-life applications, in particular robotics.

List of Acronyms

Adam Adaptive Moment Estimation, an optimizer. 21, 146

AGI Artificial General Intelligence. 1, 2

CE Binary Cross Entropy, a loss function. 78, 90, 93

CE Cross Entropy, a loss function. 118

CNN Convolutional Neural Network. 13, 26, 30, 41, 46, 47, 75, 149

GloVe GloVe: Global Vectors for Word Representation, an algorithm for language modeling. 27, 51, 59, 61, 65

GRU Gated Recurrent Unit. 140, 142

Inception Inception, a popular family of neural network architectures [88]. 46

IoU Intersection over Union. 26

LSTM Long short-term memory introduced by Hochreiter and Schmidhuber [91]. 17, 136, 140, 142, 145

mAP mean Average Precision. 97

MLP Multi Layer Perceptron. 13, 16, 17, 142

NLP Natural language processing. 22, 27, 39–41, 43, 51

RCNN Region Convolutional Neural Network. 41

ResNet Residual Network, a popular family of neural network architectures [88]. 3, 46, 89–91, 99

RMSprop Root Mean Square Propagation, an optimizer. 21, 94, 96

SGD Stochastic Gradient Descent. 21

W2V word2vec, a family of algorithms for language modeling. 23, 64, 65, 72

Acknowledgements

This dissertation would not have been possible without the continual support and valuable advice of so many people. In particular, I thank:

- my supervisor Florentin for providing a vivid research environment and giving me the necessary freedom to develop original ideas. Thank you for our discussions on thesis-related topics (and on not so related topics, too).
- my thesis committee members Minija and Christian for guiding and questioning the topic of the thesis.
- the authors and co-authors of papers I worked on for their patience in correcting me and their contributions: Florentin, Tomas, Minija, Michael, Alejandro, Alexander and Tristan.
- the former and current members of the computer vision group Markus, Simon, Florian, Tanya, Fatemeh, Aisha and Eren for advice and feedback on many of my article drafts.
- the non-computer vision group members Christian, Daniel, David, Mayte, Moritz, Jannik, Sebastian, Carlo and past members Timo, Juliane, Jan, Johannes, Sakya, Xiaofeng, Yinyun and Mina. Special thanks to Juliane and Jan with whom I shared a tiny office during my first months.
- the secretaries and technical staff of the third institute of physics: Ursel, Nikole, Sabine, Elke and Thomas for providing support in administrative issues.
- the proofreaders Oliver, Michael, Tomas, Jörn, David, Dominik and Florian for pointing out typos and incomprehensible paragraphs.

Furthermore, I am grateful to the contributors of many open source software projects that I used. Without building on these foundations, this dissertation would have been infeasible.

Lastly, I owe many thanks to my family, my parents Britta and Wilfried, and my brother Niklas for always being supportive, welcoming and providing a place to resort from the strains of academia.

Contents

CHAPTER 1

Introduction

1.1 Problem Definition and Motivation

Throughout the last years we witnessed remarkable progress in intelligent systems. Computers learned to play Atari games [147], have beaten the world champion in the game of Go [187], even without ever playing against a human before [188]. The quality of cross-language text translations improved massively [219], while speech recognition [238] has reached a level that allows it to be used in many households [1]. Images can be classified with an accuracy that is comparable to humans [95] while realistically looking natural images of fictive subjects can be synthesized [106, 26].

Although this progress clearly is striking, each achievement is constrained to a comparatively narrow domain. All these systems implement a specific interface to master a specific task: Pixels of a video game are mapped to a fixed set of controls. Text in one language is mapped to text in another language. Real-valued vectors are mapped to synthetic, yet realistically looking, images. Hence, despite impressive results, such systems are constrained to act within their individual boundaries and consequently called narrow (or weak) artificial intelligence (AI) [74]. Narrow AI is often contrasted with artificial general intelligence (AGI), which means exhibiting intelligent behavior over a diverse set of tasks and coping with unknown situations [74]. Yet it is unclear if building an AGI that is on par with humans is possible at all. Undoubtedly, if it is possible, there is a long way to go and many technical but also ethical challenges must be addressed. Arguably, a key requirement for an AGI system is to be able to move and interact autonomously in an unknown environment to enable discovery and continuous learning. Therefore, the system must not only be able to accurately describe what it sees but it must primarily know how to interact with the environment.

[1] https://www.theverge.com/2019/1/4/18168565/amazon-alexa-devices-how-many-sold-number-100-million-dave-limp

While we do not address AGI in general, in this thesis we will focus on certain indispensable technical components of an AGI system, in particular those that are concerned with the deduction of potential actions in an unknown environment. We focus on the perception of visual cues that are relevant for reasonable acting and propose to extend the classic descriptive approach to scene understanding with action-oriented scene understanding.

Organization Next, action-oriented scene understanding will be described in greater detail as the guiding framework of this thesis. It is put in contrast with classic descriptive approaches from computer vision. Subsequently, we will motivate our choice of learning from data in the wild and state why we employ distributed methods to represent knowledge. At the end of this chapter, contributions encompassed by the individual articles and manuscripts of this cumulative thesis are presented.

1.1.1 Action-oriented Scene Understanding

We define action-oriented scene understanding as the interpretation of a scene regarding opportunities of interaction, manipulation and locomotion for a (potentially robotic) agent. It is about identifying and judging potential actions, explicitly involving unnamable constituents of the scene such as parts of an object or surfaces. Action-oriented scene understanding differs from traditional scene understanding, which strives to provide an accurate description of a scene. Descriptive scene understanding asks "what is there?" while action-oriented scene understanding asks "what can be done?" and "what are the consequences?". In Fig. 1.1 we compare descriptive with action-oriented scene understanding on a hierarchy over three levels of abstraction:

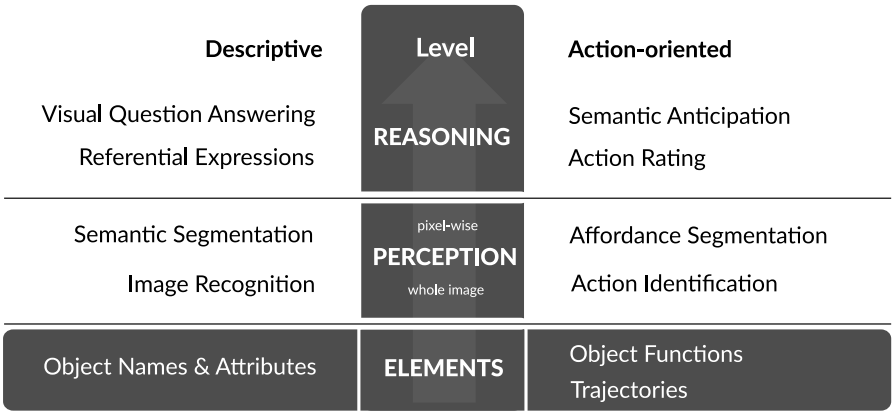

Figure 1.1: Descriptive and Action-oriented tasks organized over three levels.

elements, perception and reasoning. Elements represent the foundation of the approaches and define the entities to be predicted. While elements in descriptive computer vision involve symbols like names and labels, the action-oriented counterpart considers functions, opportunities to interact and trajectories. Often these latter elements can hardly be expressed symbolically. The perceptive level comprises components concerned with recognizing occurrences of these elements in a natural scene, which first and foremost demands pattern recognition capabilities. Building on these earlier levels, at the top level, reasoning is required. Common tasks at this level in descriptive vision are visual question answering (VQA) and resolving referential expressions. In action-oriented scene understanding reasoning is executed over actions. Accordingly, associated tasks involve appraising the presence or plausibility of certain actions and anticipating future actions.

Descriptive Computer Vision So far, research in semantic computer vision focused mostly on descriptive approaches, centered around objects, with object recognition, object detection and object segmentation being ubiquitous tasks. Other tasks include scene recognition [247], fine-grained recognition [131] or attribute recognition [160]. Recently published datasets, such as visual genome [116], contain richer information, e.g. visual relationships between objects, but still remain descriptive. In visual question answering [79], most questions assess descriptive qualities, too. For example, common question patterns are "what color is . . . " or "is there . . . ". In action-oriented scene understanding the output should tell us how to act or what to do. Action recognition from video does not match this criterion as it only seeks to provide a description of what is shown in the video ignoring what might happen next. This focus on descriptive tasks is not surprising because such tasks are easy to define and unambiguous when performance needs to be assessed. Hence, they enable the development, evaluation and comparison of novel models, for example AlexNet [117] or ResNet [88] that were developed to compete in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [176] to excel at image classification. However, the ability to conduct straightforward comparisons of descriptive methods does not imply that practical applications, for instance in robotics, should solely rely on these methods. In this dissertation, we advocate the use of action-oriented scene description as a complementary method. In fact, deriving actions would also be possible by building upon descriptive approaches. Descriptive vision systems map from images to a specification of the status quo of a scene. To add an action output, a second stage of processing, which involves reasoning on the obtained object names or attributes that describe the scene, is necessary. In this case, another mapping from a scene representation to action would need to be learned or defined. This introduces a potentially unnecessary layer of complexity and requires the explicit choice of a scene representation.

In this work, we pursue a direct approach and produce action-oriented output immediately

from images. Although this might also involve learning an implicit scene representation we do not need to define any properties of this representation a-priori (read more on this in Section 1.1.2.1). Instead of reasoning on abstract symbols like object names, action is immediately generated in an end-to-end manner, effectively implementing a shortcut from perception to action. In the most extreme form, this could involve generating trajectories and movements. However, in this thesis we will resort to predicting action labels.

Properties of Action-related Scene Understanding Action-related scene understanding tasks differ from conventional descriptive tasks in many ways. A key trait of action-oriented scene understanding is *vagueness*. Often, action-oriented statements cannot be divided sharply into wrong and right but rather associated with a plausibility. In these cases, we need to concede that there is uncertainty involved and take this into account in the design of corresponding algorithms. It is impossible to drink from an empty cup. However, the chances of drinking from a filled cup are better expressed by a plausibility depending on the cup's context instead of a predicate that can only be true or false. For example, is the environment dirty and the glass is left-over or is the glass part of a well-laid table? Related to vagueness, statements in action-oriented scene understanding are often more *subjective* compared to descriptive approaches. This means that the judgement of certain qualities differs across various persons. While people might have different opinions about whether one can eat an apple lying on the floor or whether it should be disposed, they will agree in describing the object as "cup". Another important aspect is *temporal context*: To accurately describe a scene it does not matter what has happened before since everything can be seen (apart from occluded objects). However, for anticipating what actions are likely to be conducted in the future, activities that took place in the past become relevant. For example, if a person drank coffee a moment ago, the probability for preparing another one decreases.

1.1.2 Choice of Methods

Having outlined the problem of action-oriented scene understanding and its properties, next we will motivate why we employ distributional approaches to represent knowledge and data in the wild to learn from.

1.1.2.1 Non-symbolic Approaches

This work pursues a non-symbolic approach, sometimes also called distributional or connectionist [68]. Instead of representing knowledge in discrete, atomic units (i.e. symbols), it is distributed across the weights of synaptic connections or – more general – as learned param-

eters. This allows us to employ machine learning techniques to learn this implicit knowledge directly from raw images. This decision in favor of non-symbolic approaches is motivated by two observations:

1. Some fractions of common-sense knowledge are hard or impossible to express symbolically. This applies in particular for body movements. For example, consider expressing precisely with words (i.e. symbolically) how to fetch a bottle of milk from the fridge or how to climb onto a tree. Although these activities are simple to conduct, corresponding precise descriptions easily become lengthy indicating that words or discrete symbols are not the optimal language to express these activities.

2. Symbolic approaches would require a universe of discourse, i.e. the definition of a set of symbols (such as chair, table, cup) and an enumeration of their possible states (full, dirty, etc.). For a rich scene representation this set would need to be large and its creation would require substantial efforts, while crucial constituents still might be missed. Non-symbolic approaches allow us to circumvent this explicit definition by directly processing the non-symbolic input such as an image, a pointcloud or a video without specific intermediate representations. Consequently, no space (or universe of discourse) hosting this representation must be designed which extricates us from making assumptions about the structure of such space.

1.1.2.2 Data in the Wild

In this thesis we mostly deal with data in the wild. Ang et al. [9] provide an explanation of this term, for them data in the wild means: "[...] that the datasets are not constructed and designed with research questions in mind [...]" [9]. They put this in contrast with conventional datasets that are "generally planned according to conceptual or theoretical interests with articulated research questions." [9]. This means that actual photographs of real-life scenes are used as an input to the proposed methods, which is different from many other approaches in robotics that rely on a simplified lab setup. Operating on data in the wild is necessary because autonomous robots have to cope with real-life scenes. The degree of variation is larger than in artificially set-up scenes. This increase in variation is caused by several factors: Perspectives can change, object appearances, even within a class, vary strongly and the illumination of scenes might differ. In lab setups, these factors are often kept constant. Consequently, it is more challenging to learn and evaluate on data in the wild. We make use of real-life data by using large-scale real-life datasets such as COCO [130], ADE20K [248], OpenImages [115] or EPIC Kitchens [50].

1.2 Papers and Manuscripts Contained in this Thesis

This thesis encompasses four separate articles of which two are published. The four articles are indicated by the letters A to D. For completeness, additional articles and posters that were created during the thesis are listed without being assigned a letter. For the published articles, the contents are identical to the accepted versions except for formatting and figure enumeration which we adjusted for this thesis. To enhance the readability, we maintain only a single list of references at the end of this thesis. Hence, the reference numbers differ from the published version, too. The two unpublished articles might be submitted to journals or conferences in the presented or a deviated form in the future. Below, we present all articles in conjunction with the contributions and grouped by the level of abstraction introduced above. For the articles contained in this thesis we provide a summary of their contributions. A complete graphical depiction of the contributions is provided in Fig. 1.2.

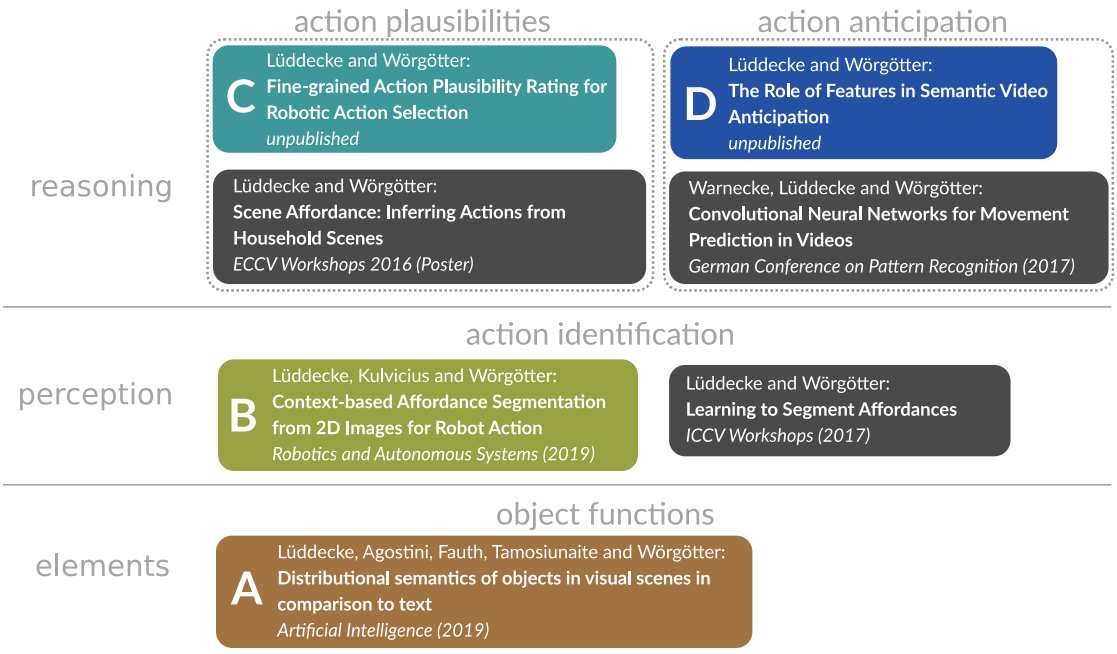

Figure 1.2: Overview of the contributions of this thesis, structured by the three levels of abstractions. Papers contained in this dissertation are denoted by the letters A to D.

1.2.1 Object Functions (Elementary Level)

- Timo Lüddecke, Alejandro Agostini, Michael Fauth, Minija Tamosiunaite and Florentin Wörgötter
  A: Distributional Semantics of Objects in Visual Scenes in Comparison to Text
  Artificial Intelligence (2019)
  Contribution: Idea, concept, implementation and most of the paper writing (ca. 85%).

  Summary: In this paper (Chapter 4) we show that visual context can be used to generate meaningful vector representations of objects. The novelty is to use a visual context instead of a textual context which is known to work well. Furthermore, we show that our vector representations of words do not only cluster semantically but also encode functional features.

1.2.2 Action Identification (Perception Level)

- Timo Lüddecke and Florentin Wörgötter
  Learning to Segment Affordances (ICCV Workshops 2017)
  Vision in Practice on Autonomous Robots (ViPAR) Workshop in conjunction with ICCV 2017
  Contribution: Idea, concept, implementation and most of the paper writing (ca. 90%).

- Timo Lüddecke, Tomas Kulvicius and Florentin Wörgötter
  B: Context-based Affordance Segmentation from 2D Images for Robot Action
  Robotics and Autonomous Systems (RAS)
  Contribution: Idea, concept, implementation and most of the article writing (ca. 85%).

  Summary: In this article (Chapter 5) we propose a system that can densely (per-pixel) label a set of 12 affordances. Compare to previous work, this system has multiple novelties: The training algorithm only requires part-level annotation but no depth or 3d representations of the scene. The number of considered affordances is larger. Lastly, we outperform the state-of-the-art approaches.

1.2.3 Action Rating and Anticipation (Reasoning Level)

- Timo Lüddecke and Florentin Wörgötter
  C: **Fine-grained Action Plausibility Rating**
  unpublished
  Contribution: Idea, concept, implementation and most of the article writing (ca. 95%).

  > **Summary:** In this article (Chapter 6) we address the problem of calculating the plausibility of conducting certain actions in a scene presented in an image. To the best of our knowledge this is a novel task. Consequently, we gather a dataset of human plausibility ratings. Then we define expressive metrics and baselines to assess the performance of various models trained on this data.

- Timo Lüddecke and Florentin Wörgötter
  Scene Affordance: Inferring Actions from Household Scenes.
  Poster at the 1st Workshop on Action and Anticipation for Visual Learning in conjunction with ECCV 2016
  Contribution: Idea, concept and poster creation (ca. 90%).

- Alexander Warnecke, Timo Lüddecke and Florentin Wörgötter
  Convolutional Neural Networks for Movement Prediction in Videos
  German Conference on Patter Recognition (GCPR) 2017
  Contribution: Idea and concept. Supervision of the master thesis (ca. 20%).

- Timo Lüddecke and Florentin Wörgötter
  D: **The Role of Features in Semantic Video Anticipation**
  unpublished
  Contribution: Idea, concept, implementation and most of the paper writing (ca. 95%).

  > **Summary:** Our work on semantic video anticipation presented in Chapter 7 is mostly empirical. We investigate how long-term temporal context can be processed by various video models involving both, recurrent and 3D-convolution-based techniques. For this analysis, we design the SymbolSeq dataset which requires keeping track of events that happen over many frames. Additionally, we conduct experiments on non-synthetic data.

CHAPTER 2

Foundations

This thesis includes diverse models and techniques with an emphasis on artificial neural networks. We do not only adapt methods from computer vision and robotics but also from related fields such as natural language processing. In this chapter, we introduce the foundations required to understand the subsequent chapters. We start with machine learning in general and focus then on neural networks as specific models. Finally, we discuss selected tasks and metrics which are relevant in context of this thesis. While we cover a broad range of topics, the explanations are kept brief. However, pointers to resources are provided that treat special aspects in greater detail. For a detailed introduction to the field of machine learning in general we refer to the books by Murphy [150] and Bishop [23]. This chapter is intended for readers that are new to machine learning and computer vision. It can be skipped by experienced readers.

2.1 MACHINE LEARNING

The key idea of machine learning is to solve problems by detecting patterns in data through parameterizing a computational model. Machine learning can be seen as a paradigm that competes with classical programming. Programming requires understanding of the problem at hand and manually designing rules and sequences of operations, i.e. an algorithm. Machine learning, on the other hand, eludes stating concrete instructions but requires the specification of a model, a learning method and data instead. The model defines the space of possible solutions: an overly simple model cannot exhibit complex behavior. In such cases, learning must necessarily fail as the correct model is not within the space of all possible solutions. The values of the model's parameters represent one solution to a machine learning problem specifically for a model. They can be discovered through learning from data. To solve a problem successfully, the correct solution must be within the model's space of possible solutions and the learning algorithm must find the correct parameters. In this chapter we will use the simple

notation $\hat{\mathbf{y}} = f(\mathbf{x})$ to refer to a model f transforming the input $\mathbf{x}$ into an output (prediction) $\hat{\mathbf{y}}$. For example, if the model is defined as a straight line, $\hat{\mathbf{y}} = \theta x$, it cannot represent data with a non-linear relationship (e.g. a parabola) between input and output.

2.1.1 FORMS OF LEARNING

First, we provide a small overview on the various settings in which machine learning can be applied. This overview is ordered by the strength of the supervisory signal.

Supervised Learning In supervised learning, N samples in form of input $\mathbf{x}_i$ and desired output (label) $\mathbf{y}_i$ pairs are available, with $i \in \{0, N\}$. Through learning, the model is parameterized to map each input $\mathbf{x}_i$ to the corresponding $\mathbf{y}_i$. An example is image recognition, where image pixels represent the input and the desired output is the class of the shown object. If an output space is discrete (e.g. for objects), we speak of a classification problem. If the output space is continuous (e.g. a temperature), we speak of regression problem. Conventionally, the data is split into a training set, a validation set and a test set. Training data is used for the actual learning while the validation set serves as a control to monitor training progress. For this, the training error, which is a measure of how bad the network performs on the training data, and the validation error, which is the same measure calculated on unseen validation data, are compared. When the model has too many parameters, it becomes susceptible to over-adaption to the specific training data rather than learning the general principles required to solve the problem. This phenomenon is called overfitting and can be detected by tracking the validation error during learning. When overfitting occurs, the validation error starts to increase while the training error continues to decrease. The test set is meant to be evaluated after concluding the parameter search on training and validation set. For standard benchmarks it is common practice to keep the labels of the test set private and restrict the number of evaluations. This prevents implicit overfitting the test set by the model design [176].

Self-Supervised Learning Recently, self-supervised learning has gained popularity where an artificial proxy task is created on an unlabeled data collection. This way, labeled training data pairs $(\mathbf{x}_i, \mathbf{y}_i)$ can be automatically generated. Self-supervised learning is primarily employed to learn meaningful feature extractors (e.g. descriptions of shapes, textures or sounds). These learned features are then re-used in different target tasks like object classification [58, 10], sound classification [10] or tracking [222]. For example, Doersch, Gupta, and Efros [58] employed self-supervised learning on an unlabeled image collection. A supervised task is created by sampling patches from an image and asking where they were originally positioned relatively to each other. Arandjelovic and Zisserman [10] trained a two-stream neural network on unlabeled

video data to decide whether a sound corresponds to an image. This way they learn feature detectors for audio as well as images that are successfully applied in downstream audio and image classification tasks.

Reinforcement Learning Contrary to supervised learning, in real life scenarios an immediate signal that tells us if a decision was right or wrong is not always present. Instead, humans commonly perform a series of actions before receiving any feedback. For example, an unexperienced chess player conducts a series of sub-optimal moves before being defeated. Reinforcement learning involves an agent that carries out an action at every time step in an environment (e.g. the chess board). The action selection model is called the policy. After acting, the agent learns how the environment has changed and sometimes receives a reward signal. This signal is not only dependent on the previous action but on the sequence of all actions conducted before [23, Chapter 1]. The goal of reinforcement learning is to find a policy that generates a high reward in the long term.

Unsupervised Learning In unsupervised learning, no labels $\mathbf{y}_i$ are available. Hence, no classification or regression can be performed. However, structure inherent in the data can be learned. This could involve finding a better data representation (e.g. in terms of size) than the one provided by the input samples. Classic examples of unsupervised learning are clustering and dimensionality reduction. For the former, a set of items (e.g. cars) is grouped based on some features (e.g. color, weight, mileage) of the items. In dimensionality reduction, high-dimensional data (e.g. an image) is reduced to low dimensional data (e.g. a compressed image) under preservation of variance [23, Chapter 1]. Mixtures between supervised and unsupervised learning are possible, too. When only a fraction of the samples is labeled, the problem is denoted as semi-supervised.

2.2 NEURAL NETWORKS

This section is meant to provide an overview of neural networks as parameterizable models for machine learning. They can be applied in any of the settings presented above. As they are central to this thesis, neural networks and associated methods will be explained in greater detail. First common types of neural networks are presented, then more sophisticated architecture and design patterns are discussed. Finally, an intuition of how the learning of weights works is provided. Elaborate explanations of the foundations of neural networks for machine learning can be found in textbooks by Goodfellow, Bengio, and Courville [76] and Murphy [150].

Neural networks are inspired by the organization of the brain: neurons are connected by synapses to form a network through which neural activity is propagated. Each neuron generates an output activation depending on accumulated activations from incoming connections. Neurons integrate incoming currents from adjacent neurons. When the charge exceeds a certain threshold, a so-called action potential is triggered and propagated along the axon until it eventually causes the release of neurotransmitters at the neuron's synapses. These neurotransmitters then travel to connected neurons, where they lead to an ion current by opening specific ion channels. For a comprehensive discussion of biological neural networks used in computational neuroscience see the book of Dayan and Abbott [52]. In contrast to these biologically more accurate models, we rely on rate-based models. These simplify the complex process of activity propagation by modelling synaptic strength by a scalar called the weight. This enables learning weights through the backpropagation algorithm [175]. To enable learning of complex functions, activations are modified by non-linear functions (non-linearities).

Each synaptic weight expresses how well activation is propagated from one neuron to the other. The weights are the learnable parameters of the model, which encode the knowledge that has been acquired during training. Not all neurons are necessarily connected, i.e. for some connections the weight is always zero and no synaptic weight can be learned. The architecture of the neural network determines which synaptic connections have zero weight, i.e. which neurons are connected. A common way to define an architecture is through layers, which are groups of neurons. Neurons of a layer can interact with neurons of adjacent layers only. By stacking multiple layers, deep neural networks can be built, with deep referring to a large number of layers. Consequently, the field dedicated to training and designing deep neural network models is called deep learning. Note that there is no necessity of organizing the network into layers. Yet the concept of layers often simplifies thinking about neural networks. In practice, neurons between any layer can be connected (skip-connections [134]). The organization into layers bears some advantages in computational efficiency since some operations (in particular matrix multiplications) can be carried out in parallel.

Another, crucial building block of neural networks are activation functions. These alter the neural activation by applying a nonlinear, normally monotonic function. Examples are the sigmoid (or logistic) function σ or rectified linear units ReLU [73] with

$$\sigma(x) = \frac{\exp(x)}{\exp(x)+1} \text{ and } \mathrm{ReLU}(x) = \max(0, x).$$

Despite their simplicity, neural networks are powerful models. Hornik [92] has shown that multi-layer perceptrons (see 2.2.1.2) involving only a single hidden layer can approximate any continuous function given enough hidden units. See Nielsen [155, Chapter 4] for a more intuitive explanation.

2.2.1 NEURAL NETWORK COMPONENTS

Subsequently, we present a selection of the most common components of neural networks. This includes both layers and networks. These two concepts are sometimes hard to differentiate because a network can occur as part of a larger network, like a layer. We start with feed-forward networks before discussing recurrent networks.

2.2.1.1 Fully-Connected Layer

Arguably, the central layer type is the fully-connected layer, which is sometimes also called linear layer or dense layer. It takes a vector of activations as input and transforms it to a vector output through a projection, which can be implemented by a matrix multiplication. The corresponding matrix encodes the synaptic weights (or parameters) between pairs of neurons. Since every input neuron interacts with every output neuron, the layer is called fully-connected.

2.2.1.2 Multi Layer Perceptron

A multi-layer perceptron (MLP) consists of a sequence of fully-connected layers. Neurons of adjacent layers are connected. After each layer, an activation function can be applied. If no activation function is used, a deep network is equivalent to using a single fully-connected layer, because a series of linear projections can be replaced by a single projection.

2.2.1.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are central to the majority of modern computer vision systems. In contrast to the MLP where activation is represented in form of vectors, a CNN transforms higher dimensional tensors. In the case of a 2-dimensional CNN (which is typical for image processing) the tensors have three dimensions: height, width and channels (feature maps). In the input, channels can represent the RGB colors while they encode more abstract features in later layers. This distinction between spatial components and channels is important for weight sharing, which is the central idea of CNNs: Instead of considering each neuron in isolation, a convolutional layer acknowledges the spatial layout of its input. A bank of filters (which are composed of weights and are also called kernels) with a fixed spatial extent (called the kernel size) is applied (multiplied) at multiple locations (e.g. the pixel grid) of the input. The distance between these locations is called the stride. In conventional CNNs, each filter generates one output channel. The filter's coefficients represent the parameters of a CNN.

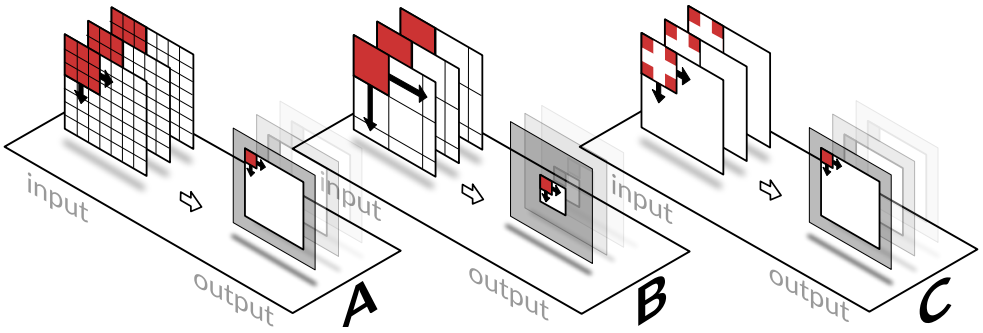

Figure 2.1: 2-dimensional convolution with strides of 1, 3 and 1 and dilation of 1, 1 and 2. The filter indicated in red and encompasses all three input channels to write one output element.

Note, a convolutional layer is a special case of a fully-connected layer. This means we can construct a fully-connected layer that computes the same function as a convolutional layer. However, the concept of weight sharing allows a convolutional layer to learn better features with less data as they encompass a smaller number of parameters (weights).

Receptive Field The receptive field of a neuron refers to the area of the input for which this neuron is sensitive. This means, the receptive field determines how well spatial context can be processed by the CNN. Consider a single convolutional layer with a kernel size of 3x3 and a stride of 1. Each output neuron after applying this layer captures interaction in a 3x3 environment and therefore has a receptive field size of 3x3. By stacking many convolutional layers, the receptive field size grows, but not very fast. For a reasonably large receptive field size, we would need to build a very deep network, which is undesirable due to its high computational demands and large number of parameters. Therefore, to enable a faster increase of the receptive field size over a fixed number of layers, different techniques were developed. Subsequently, we present the most common ones.

Pooling Pooling divides the input into a spatial grid and reduces all activations that fall into a grid cell of this grid to a single activation. The reduction operation is applied individually per-channel on all activations that fall in a grid cell, which means that the spatial extent is reduced while the number of channels remains the same. Common reduction operations are the maximum function or averaging. The compression of the spatial grid leads to a strong increase in the receptive field.

Stride In a convolutional layer, filters are multiplied with parts of the input at locations determined by a uniform grid. The distance between these locations is called stride. Similar to

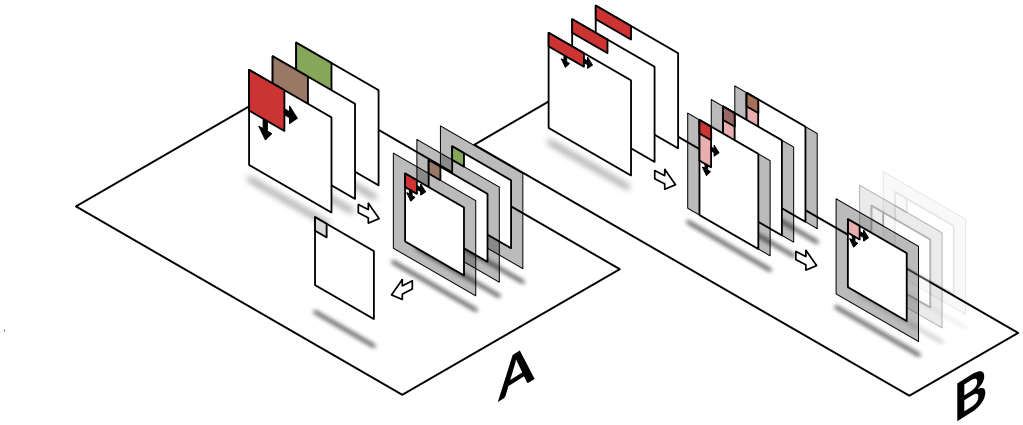

Figure 2.2: Separable convolution and channel-wise separable convolutions.

pooling, the stride determines the spatial extent of the output. Larger strides lead to a smaller output size and also a larger increase in receptive field size. In Fig. 2.1 we depicted a stride of 1 (A) and a stride of 3 (B).

Dilation In dilation (also called a-trois), the multiplication of filter and input is not carried on adjacent pixels but space is added in between. Dilation is equivalent to using a larger kernel and adding zeros between the original kernel weights. In Fig. 2.1 we show the idea of dilation. A 2x2 filter is dilated to a size of 3x3 [240].

Separable Convolutions In some cases, convolutions with an n-dimensional filter can be replaced by multiple m-dimensional convolutions, with $n > m$. Depending on the input and filter size, this can allow a much more efficient implementation. A classic example of such a separable filter is the Sobel operator, which is a 2-dimensional filter that can be implemented by two 1-dimensional convolutions.

Separability is neither limited to the spatial dimensions nor to 2-dimensional CNNs. Each filter of a convolutional layer conventionally encompasses all input channels, i.e. if there are k input channels, a 3x3 kernel would have $9k$ parameters. As opposed to this, in depthwise separable convolutions individual channels are used as input. The Xception architecture by Chollet [41] heavily relies on depthwise separable convolutions.

3D Convolutions 3D-convolutional layers allow learning kernels with an extent in four dimensions (3D + channel) and movement in three directions. A classical application of this is video, where two spatial dimensions are combined with a temporal dimension, such that

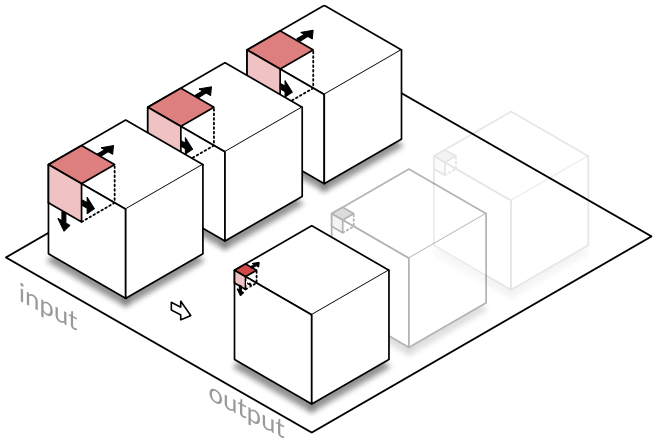

Figure 2.3: Illustration of a 3D convolution over a three-channel input generating a single output element. When generating other outputs, the filter moves synchronously in all channels in three directions.

spatio-temporal filters can be learned. Due to the high dimensionality, the concept of 3D convolutions might be hard to imagine. In Fig. 2.3 we show a sketch of a 3D convolution with one filter location and corresponding output being indicated in red.

Global Average Pooling Earlier convolutional network architectures such as AlexNet [117] and VGG16 [189] were restricted to operate on fixed size input only. As a consequence, input images need to be scaled or cropped to match the network's input size. Activations in all spatial locations of the tensor yielded by the last convolutional layer are concatenated (lined up) to obtain a single fixed sized vector. Sometimes this vector is called the feature vector. Then a fully-connected layer or an MLP (see Sec. 2.2.1.2) is used to transform the feature vector into the network's output, which, in classification, typically has as many neurons as the dataset has classes (e.g. 1000 in ImageNet). The MLP requires feature vectors of a certain size, which in turn allows only for certain input image sizes of the whole network. If the image is too large or too small, the resulting feature vector would have a different spatial extent and hence would be incompatible with the MLP.

In contrast, global average pooling allows to operate on images of (almost) arbitrary size. Instead of concatenating to generate a single feature vector, features are averaged channel-wise across all locations to yield a single feature vector (see Pooling in Sec. 2.2.1.3). This way the output size becomes insensitive to spatial extent of the input, because the resulting feature vector always has the same size. Hence, a network that uses global average pooling can work with a large range of image sizes. Note, processing of small images can still fail, e.g. when the

output size of a layer is smaller than the kernel size of the next layer.

2.2.1.4 Recurrent Neural Networks

While fully-connected layers require a fixed size input, many problems involve an input of variable length. For example, sentences of a text normally vary in length. Recurrent neural networks (RNNs) allow operating on this kind of data by maintaining a hidden state that is updated sequentially for each element of the input. This means, a sequence with n elements causes n updates of the hidden state. A more detailed explanation of RNNs can be found in Chapter 7.2.

It was found that conventional recurrent neural networks but also deep MLPs suffer from the so-called vanishing gradient problem during the learning phase. This means that the gradients computed through backpropagation (see section 2.2.2.2) become very small, essentially preventing progress in learning. Hochreiter and Schmidhuber [91] propose long short-term memory (LSTM) to overcome this problem by protecting the internal state via forget and input gates. The former decides which information of the hidden state should be removed while the latter decides which information to write. An elaborate explanation of LSTM is provided in this blog post by Olah [1]. Recently, evidence accumulated that sequence modelling can be carried out with convolutional networks [12] or using attention mechanisms [219] instead of recurrent units.

2.2.1.5 Residual Connections

Residual connections are an architectural pattern that has proven to improve performance dramatically in deep networks. He et al. [88] observed that the training error of networks at a certain depth starts increasing, which they call the degradation problem. Intuitively, this should not happen because a deeper network can trivially be transformed into a shallower one by setting some transformations to identity. They infer that training techniques tend to have problems to learn the identity transformation. To overcome this problem, they suggest to facilitate learning the identity. This is done by adding connections that bypass blocks of convolutional layers and merge bypassed activation with the altered activation that went through the block. This way only changes in activation need to be learned. Their experiments indicate that this enables training deeper architectures without being affected by the degradation problem, resulting in the well-known ResNet family [88].

[1] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

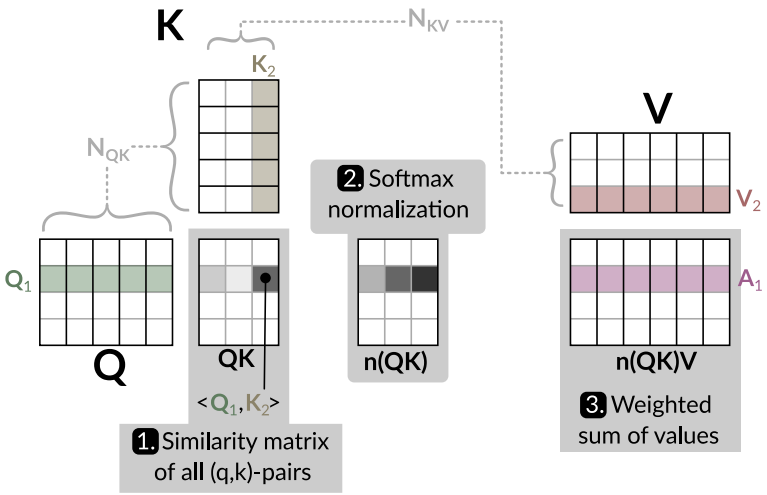

Figure 2.4: Attention Mechanism: By stacking query vectors into $\mathbf{Q}$ and using corresponding keys $\mathbf{K}$ and values $\mathbf{V}$ multiple attention vectors are computed by two projections (1,3) and a normalization (2).

2.2.1.6 Attention Mechanism

Attention mechanisms have emerged as a central component in many modern network architectures across various tasks. They allow the network to focus on certain parts of the input while ignoring others. For example, this could involve a salient image region in a computer vision model or related words in a language model. A fairly general definition of attention is provided by Vaswani et al. [219]. Subsequently, we present their definition and follow their notation. The idea is to attend over a set of values $\mathbf{v}_i$. To each value belongs a key $\mathbf{k}_i$ that might express some properties about its corresponding value $\mathbf{v}_i$. This storage can now be queried by providing a query vector $\mathbf{q}$. For this, we compute a compatibility between $\mathbf{q}$ and all $\mathbf{k}_i$ and then use this compatibility (after normalizing) to compute a weighted sum over all values. So intuitively, the values v_i contribute most to the output where the corresponding properties $\mathbf{k}_i$ align best with our query $\mathbf{q}$ properties. For multiple queries, this algorithm can be efficiently implemented through matrix multiplications. Multiple queries are stacked into a query vector $\mathbf{Q}$. Then the attention-weighted vectors $\mathbf{A}_i$ are obtained by:

$$\mathbf{A} = \text{softmax}(\mathbf{QK})\mathbf{V}$$

with softmax denoting a row-wise softmax followed by a scaling. The softmax assures that each row sums up to one and does not contain negative values (see 2.2.2.1 for details of softmax). The scaling is intended to generate more informative gradients for backpropagation.

Fig. 2.4 illustrates the collective computation of attention for multiple queries. A similarity

matrix is generated from the keys $\mathbf{K}$ and the query $\mathbf{Q}$. The matrix expresses how well the query aligns with the keys. In a second step, the matrix is normalized. Output is generated by multiplying the normalized matrix with the values $\mathbf{V}$.

The origin of $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ depends on the application, with multiple setups being possible. In text processing $\mathbf{V}$ could represent word vectors of an input sequences and $\mathbf{Q}$ and $\mathbf{K}$ could be generated by a recurrent network. Another possibility is to generate all matrices from the output of one layer which is called self-attention. Furthermore, attention can be used to fuse two inputs where values and keys are provided by one input and the query is provided by another input. An example of this is the transformer by Vaswani et al. [219].

2.2.2 LEARNING

So far, we have seen many building blocks of neural networks. But an essential question is still unanswered: Given a model and data, how do we learn the right parameters? To this end, we first define a loss function and then optimize the model's parameters to minimize this loss. This optimization is normally carried out by sequential updates of the weights while feeding samples from the training dataset to the network. If the model is differentiable, which is commonly the case in deep learning, the computation of the weight updates can be tremendously accelerated by relying on the gradients of the parameters.

2.2.2.1 Loss Function

The loss function $\mathcal{L}$ tells us how good the current parameterization of the network performs on a given task. It takes a model's prediction $\hat{\mathbf{y}}$ and the ground truth $\mathbf{y}$ and computes a scalar that expresses the misalignment of both. Note, in this section the loss of a single sample is considered. In practice, it is common to compute an average loss for multiple samples (batch).

L1/L2-Loss Straightforward choices for the loss function are the L1-norm and L2-norm of the difference between prediction and ground truth. While both are related, an important difference is that the L1-norm is more tolerant to outliers while the punishment for extreme differences is stronger in the L2-norm due to the exponent of two.

Cross Entropy A frequently used loss function is cross entropy which is based on the concept of entropy from information theory. It requires predictions to be probability distributions. This can be done by applying a softmax function on the network's output (prediction) vector $\hat{\mathbf{y}}'$ of

length N, given by

$$\hat{\mathbf{y}}_i = \text{softmax}(\hat{\mathbf{y}}')_i = \frac{\exp \hat{\mathbf{y}}'_i}{\sum_j^N \exp \hat{\mathbf{y}}'_j}.$$

Alternatively, the sigmoid function $\sigma(\mathbf{y}_i) = \frac{1}{1+\exp \mathbf{y}_i}$ maps each element to the interval [0,1], which is useful for binary classification. Let C be the number of classes or categories, then we can define the cross entropy loss $\mathcal{L}_{\text{CE}}$ by

$$\mathcal{L}_{\text{CE}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} \mathbf{y}_i \log \hat{\mathbf{y}}_i.$$

Sometimes predictions are interpreted as individual binary (e.g. presence) probabilities rather than a probability distribution over many cases. An example is predicting attributes which are non-exclusive, i.e. more than one attribute can occur at the same time. In this binary case, a probability distribution can be expressed with a scalar. Hence, here the cross entropy can be simplified to

$$\mathcal{L}_{\text{BCE}}(y, \hat{y}) = -y \log \hat{y} - (1 - y) \log (1 - \hat{y}).$$

Weighted Cross Entropy Often categories are imbalanced. This involves situations where one category is more frequently represented in the training data than other categories. A possible solution to this is to assign each category an individual weighting factor $\mathbf{w}_i$:

$$\mathcal{L}_{\text{CEW}}(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{C} \mathbf{w}_i \mathbf{y}_i \log \hat{\mathbf{y}}_i.$$

Note, this will not differentiate between samples within a category as $\mathbf{w}_i$ only depends on the category. However, even within a category there might be easier or more difficult samples.

2.2.2.2 Backpropagation

Many optimization methods require the derivative of the loss with respect to each layer's parameters, which is a measure of how much this parameter has contributed to the loss. An efficient way to compute these gradients is through the backpropagation algorithm [175].

A neural network is composed of many functions with at least some having learnable parameters (especially fully-connected and convolutional layers). The goal of backpropagation is to determine the partial derivative of the loss with respect to all parameters, which is often called the gradient. Let $\mathcal{L}$ be the loss and $\boldsymbol{\theta}$ a parameter vector encompassing N elements. Then we can write

$$\frac{\delta \mathcal{L}}{\delta \boldsymbol{\theta}} = \left(\frac{\delta \mathcal{L}}{\delta \boldsymbol{\theta}_0}, \ldots, \frac{\delta \mathcal{L}}{\delta \boldsymbol{\theta}_N} \right)^T = \nabla \boldsymbol{\theta}.$$

Intuitively, this derivative tells us how the loss would change when we change the parameter. For this, we start at the scalar loss and then traverse the network in reverse to calculate gradients using the sum-rule and the product-rule. Backpropagation is also called reverse-mode differentiation [2]. For an exhaustive explanation of how backpropagation works, we refer the reader to this blog post by Olah [2] and Chapter 3 of the book by Nielsen [155].

2.2.2.3 Weight Updating

In the context of the neural networks we discuss here, learning means updating the network's synaptic weights, i.e. the network's parameters, such that the loss $\mathcal{L}$ is reduced. Before training starts, the network's weights are randomly initialized (here the underlying probability distributions is crucial) or initialized with weights obtained from a previous training round (transfer learning). Learning can be considered a search for a local minimum of the loss function. This search is very challenging because it takes place in a high-dimensional space. Luckily, the previously computed gradients help to guide this search. This is done by multiplying the gradients obtained through backpropagation with a learning rate λ. In gradient descent the update is applied individually for each parameter by

$$\boldsymbol{\theta}' = \boldsymbol{\theta} - \lambda \nabla \boldsymbol{\theta}.$$

Since the learning rate is not a synaptic weight but still a parameter relevant for learning it is denoted as a hyperparameter. Common choices for the learning rate range between 0.1 and 0.0001.

In many cases it is not possible to compute the gradient for all samples of the training dataset at once due to memory constraints. In order to elude these limitations, random subsets of the dataset are drawn, and the network is updated based on the gradient of these mini-batches. The mini-batch size is another hyperparameter that influences the success of training. Gradient descent on such randomly sampled mini-batches is called stochastic gradient descent (SGD). It can be extended with momentum by using gradients from previous iterations. This introduces robustness to noise to the optimizer while it travels across the parameter space. For a detailed discussion of momentum, we refer to the interactive article of Goh [75].

SGD uses the same learning rate for every parameter. However, this is not necessarily a good idea, because some parts of a model could deal with larger updates leading to faster learning while other might be more sensitive, requiring small learning rates. The Adam [110] and RMSprop [203] optimizers address this problem by allowing parameter-wise learning rates. They differ in how they make use of past gradients to adapt the learning rate for each parameter.

[2] http://colah.github.io/posts/2015-08-Backprop/

A comprehensive summary including a large set of optimization methods is provided by Ruder [174].

2.2.2.4 Normalization

In practice, a neural network's input and activations between layers are often normalized. The well-known batch normalization has proven to accelerate convergence and improve on generalization while being more robust to initializations of the network's parameters [98]. A straightforward way of normalizing can be applied on the input. By aggregating statistics over the training dataset, images (or generic features) can be transformed such that the expected value of groups of neurons is zero with a certain standard deviation. More recent normalization methods [98, 125, 235] are adaptive, i.e. they involve parameters that are learned in addition to the actual network weights with the sole purpose of normalizing the input. The normalization can be carried out over different dimensions, or ranges of features along a dimension. Batch normalization [98] has proven to be an essential component in many networks providing strong boosts in training speed and accuracy. However, for small batch sizes the calculated statistics become inaccurate, potentially disturbing the training process. This is particularly important for training large networks or large input data, where only few samples fit in the memory. Therefore, more recently, normalization methods encompassing layers [125], instances [216] and groups [235] were proposed. Layer normalization and instance normalization have shown to improve sequence modelling tasks while group normalization enhanced visual recognition. The latter positions it as an alternative to batch normalization when only small batch sizes [235] are possible.

2.3 Tasks and Metrics

In addition to the methods discussed above, it is important for understanding the remainder of this thesis to introduce selected tasks (or problems). In addition, we present metrics that are employed to assess the quality of models working on these tasks.

2.3.1 Language Modelling

While most of the foundation section deals with topics from computer vision, language modelling originates from the field of natural language processing (NLP). It involves modelling the probability of word sequences. This means predicting how likely one or more words are, given a window of contextual words. A simple example is finding a plausible next word for

the sequence ”tennis is . . . “. Your internal language model will probably tell you that in this case the words ”fun“ and ”exhausting“ are more likely than "apple“. Making useful predictions requires understanding of both grammar and semantics. Hence, language modelling is a challenging task. A favorable property is that training data can easily be obtained by removing words from existing texts.

While neural language models are not new [20], the success of the word2vec method [144] has revived interest in the topic. The word2vec algorithm, or more precisely skip-gram negative sampling, proposed by Mikolov et al. [144] learns word representations in form of vectors that are optimized to predict contextual words. The model is trained on a large-scale text dataset by differentiating the correct contextual words from randomly sampled “noise” words. The learned word vectors turn out to capture semantics well, i.e. words with similar meaning are assigned vectors close to each other. Additionally, analogies are captured in form of basic arithmetics.

Language models are typically assessed in terms of the perplexity measure. Perplexity [101, Chapter 3] tells us how well a model predicts missing words from the context. A low perplexity indicates good predictions of the model. Let C_i denote the contextual words of word w_i in the evaluation text W (having N words). Then we can define perplexity as

$$PP(W) = \Big(\prod_{i}^{N} P(w_i \mid C_i) \Big)^{-1/N}.$$

While perplexity gives a good estimate of how well the language model works, commonly the performance for an actual downstream task is measured [101, Chapter 3]. Recently, extended neural language models, which use attention, were able to generate paragraphs of consistent text when conditioned on a sentence (e.g. the first sentence of the paragraph) [219, 56]. These models use the attention mechanism described in Sec. 2.2.1.6 to incorporate information over already generated text and the conditional text input for the prediction of each new word.

2.3.2 IMAGE AND VIDEO CLASSIFICATION

Classifying an image or a video is a key task in computer vision and requires understanding contents of the media. Given an image or a video, usually a probability distribution over a fixed set of classes is generated from which the most likely class is taken. In the context of this thesis they are important because we borrow many methods that were originally designed to work on these tasks.

Traditionally, manually-designed features [49, 135] were extracted from images and processed [48, 190, 124] to carry out classification. The seminal paper by Krizhevsky, Sutskever, and Hinton [117] and its successors [189, 88, 201] revolutionized this field by learning features

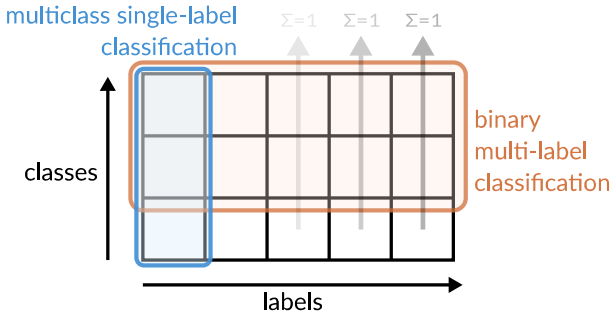

Figure 2.5: Illustration of different classification settings. The output can involve this scheme once (e.g. image classification) or multiple times (e.g. per-pixel in segmentation).

in a supervised learning setting and outperforming previous approaches significantly. In the well-known ILSVCR challenge [176] the top-performing systems have matched and exceeded even some human performance baselines. More recently, video models have undergone a similar development as image models with learned feature extractors becoming the standard, too [30].

Multiclass and Multi-label In the context of classification, it is important to distinguish the orthogonal terms multiclass and multi-label. Here, each label can assume one class out of the set of all classes as a value. A classification problem can be binary or multiclass in one dimension and simultaneosly single-label or multi-label in the other dimension. Multiclass single-label classification means that, out of all possible classes for a given sample, only one class is correct. Multi-label classification allows for multiple entities being present in one sample at the same time. In these cases, often, the number of classes is two, i.e. a label can either be present or absent. This is called a binary multi-label classification problem. A graphical explanation of both terms is provided in Fig. 2.5. The number of classes and the number of labels must be considered in the loss function used for training as well as in the metrics used for evaluation. Multiclass classification requires a categorical loss function (e.g. cross entropy) while binary multi-label classification requires a binary loss function that only considers presence of labels (e.g. binary cross entropy).

2.3.2.1 Metrics

In this section we follow the distinction between multiclass and multi-label explained above. Binary single-label classification metrics introduced here can be expressed in terms of true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). These concepts are best explained graphically. In Fig. 2.6 a binary ground truth (purple) and binary prediction (green) are depicted. Predictions that do not overlap with ground truth are false positives. Ground truth elements (e.g. pixels or bounding boxes) that does not overlap with predictions

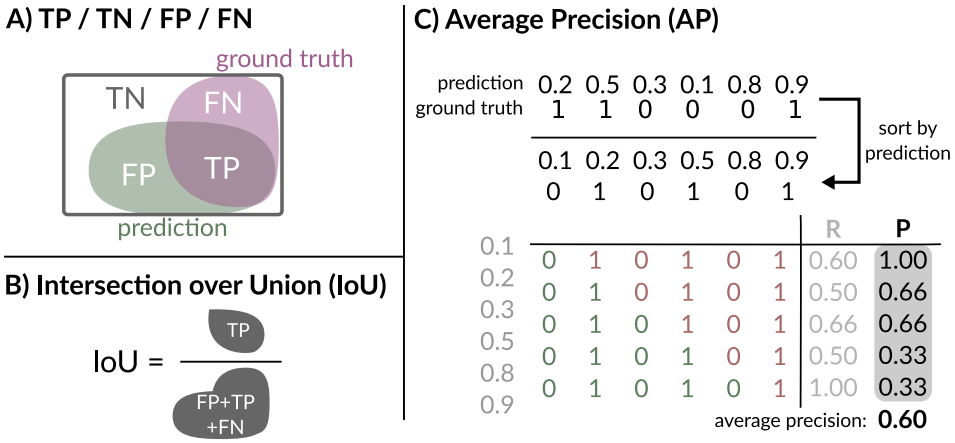

Figure 2.6: Depiction of the calculation of TP/TN/FP/FN (A), intersection over union (B) and mean average precision (C). R and P denote recall and precision.

are false negatives. At the intersection of ground truth and prediction lie the desired true positives. The white space indicates absence of both, which means true negatives.

Accuracy In multiclass single-label classification, accuracy is defined as the fraction of correct classifications within N samples, i.e. where prediction $\hat{\mathbf{y}}_i$ is equal to the ground truth $\mathbf{y}_i$.

$$\text{Acc}_{\text{cat}} = \frac{|\{i \mid i \in \{0, \dots N\} \mid \mathbf{y}_i = \hat{\mathbf{y}}_i\}|}{N}$$

In binary multi-label classification, accuracy can be determined by counting the cases in which prediction and ground truth agree. When the obtained number is divided by the number of all considered samples, we obtain the accuracy.

$$\text{Acc}_{\text{bin}} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

In case of imbalanced data, i.e. when there are much more samples for one class than another, accuracy can be misleading as always predicting the most frequent class (the mode of the labels) can yield high scores without doing any computations.

Precision and Recall A better way to deal with imbalanced data is to consider precision and recall. Precision expresses how good the samples accepted by the classifier are, recall tells us how good the space of all correct samples is covered by the classifier.

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad R = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

To combine both metrics in a single score, often the harmonic mean of precision and recall, denoted as F1 score, is used:

$$\text{F1} = \frac{2PR}{P+R}$$

Class-wise metrics Metrics can be computed over all available samples or specifically over all samples of a class. The latter is often more insightful as it allows to distinguish between classes and speculate about the reasons of deviation in the performance. Often, the average over all class-specific scores is used as a metric because it gives more weight to rare classes.

2.3.3 Semantic Segmentation

While the output in image classification tasks refers to the whole image, in semantic segmentation a prediction is made densely for each pixel. Hence, the output has the same spatial extent as the input. Hence, semantic segmentation requires other metrics than image classification. Recently, multiple CNN-based approaches to semantic segmentation were proposed: While some approaches scale-up the output in a single step [134, 38], others use multiple up-scaling steps [172, 11]. Dilated convolutions are occasionally used due to their ability to rapidly increase the receptive field [38, 240].

2.3.3.1 Metrics

Intersection over Union A very common metric in semantic segmentation is intersection over union (IoU). It has the advantage of punishing over- as well as under-segmentations, i.e. segmentations that capture more than the target or less than the target. For two classes, (binary) IoU is calculated by the intersection of predicted segmentation and ground truth segmentation divided by the union of both. As illustrated in Fig 2.6, intersection over union is calculated by

$$\text{IoU} = \frac{\text{TP}}{\text{TP}+\text{FP}+\text{FN}}.$$

For computing an IoU for $C > 2$ classes, we can construct C one-vs.-rest problems. This is done by transforming all pixels of a class to 1 and pixels of other classes to 0. Now we can compute IoU as defined above. The mean IoU is calculated by averaging all C class-specific IoU scores.

Mean Average Precision Commonly in neural networks, predictions are expressed in terms of probabilities over classes rather than crisp true or false decisions. Intersection over union requires us to binarize these predictions. However, if we binarize them, i.e. pick a threshold value t and define everything below t as absent and above t as present, we add another parameter to the metric. Mean average precision allows to use probabilistic predictions directly. The

calculation is sketched in Fig 2.6. First, precision is calculated over multiple different threshold levels. One way to obtain these threshold levels is by considering all occurring prediction scores as shown in Fig 2.6. The resulting precisions are then averaged to obtain the average precision. This is roughly equivalent to the area below the precision-recall curve. By averaging multiple class-specific average precisions the mean average precision (mAP) is obtained. In semantic segmentation, each pixel is typically considered to be an individual prediction.

2.3.4 TRANSFER LEARNING

Transfer learning refers to the concept of using knowledge acquired in one task to improve the performance in a second target task. Transfer learning is particularly useful in conjunction with learned feature extractors. A model that must operate on a small dataset benefits from pre-training on a large-scale dataset because the learned features are to some degree universal when the large dataset offers a lot of variety. Currently, it is very common to initialize feature extractors for visual tasks with features learned on the ImageNet dataset [53].

Zamir et al. [243] investigate practical transfer learning in detail. Concretely, they study the relationship between 26 tasks concerning transfer learning using a computational approach. They build a taxonomy that indicates which tasks benefit from transfer learning from which other tasks. This taxonomy highlights the utility of transfer learning in real world applications.

A method of making use of transfer learning in natural language processing (NLP) are word vectors. These vectors are generated by training a language model like word2vec [144] or Glove [161] and are then used in other tasks. Since such vectors encode knowledge from the text, they facilitate the conduct of other tasks. The effectiveness of this approach has been shown for tasks such as sentiment analysis of statements [192] or question answering [237]. Recently, a new generation of approaches that uses conditional word vectors has been proposed [93, 162, 56]. Here the idea is to generate word vectors considering the context, which particularly addresses polysemes, i.e. words with multiple meanings. E.g. the word "bank" has a different meaning, depending on context. Consequently, the corresponding word vectors should differ, too. This conditioning on the context, along with other enhancements such as the use of attention, has further improved the performance on language modelling and downstream tasks, e.g. translation.

CHAPTER 3

Related Work

Since this thesis is composed of multiple manuscripts, each having a specific discussion on related work, this section is meant to provide a broad perspective on related work across multiple fields.

3.1 COMPUTER VISION AND ROBOTICS

Related work within computer vision and robotics is discussed thoroughly in the individual articles A to D. For a review of methods concerned with object semantics we refer to Section 4.2, for affordance generation we refer to Section 5.2, for rating action plausibility to Section 6.2 and for semantic video anticipation to Section 7.2 We constrain this review to work within computer vision and robotics that is relevant to this thesis from a broader perspective.

There is a large body of research in action recognition [228, 249, 30, 218]. However, as stated in the introduction, this thesis is about making action-oriented predictions (e.g. what happens next) and action recognition is only relevant as far as the model design is concerned. For both anticipation and recognition, we need to process image sequences and carry out a classification. Moreover, there exist approaches that use action to predict other qualities. For example, Fouhey et al. [65] used human actions to estimate scene geometry. Gupta and Davis [83] combined video understanding and object perception in probabilistic framework and showed that object recognition benefits from information about the ongoing action (and vice versa). There is some work that falls into the category of action-oriented scene understanding but does not employ the semantic setting we use in this thesis. Instead, they predicted future dynamics of objects [149, 230], pursuing the goal of physically accurate predictions.

Reinforcement Learning Inferring actions is the goal of the policy in reinforcement learning and indeed some approaches rely on images as an input for the policy. Levine et al. [127] used a CNN as a policy to generate motor torque. By formulating policy search as a supervised problem, the perception module and control policy which are implemented by the CNN can be trained jointly in an end-to-end fashion. The authors demonstrated the success of their method by multiple robotic experiments. More recently, Pathak et al. [159] introduced a curiosity-driven approach to learn a policy which does not depend on dense rewards. While both of these reinforcement learning-based approaches share the goal of deriving actions from images, they operate in a different scenario and use different methods. They involve low-level motor actions while we are primarily interested in higher-level actions such as drinking or cutting. In contrast to reinforcement learning used here, we exclusively make use of supervised learning.

3.2 Psychology and Cognitive Science

Next, we review selected work from the fields of psychology and cognitive science. Naturally, this is only related on a coarse scale as it tends to involve more theoretical considerations compared to the computational methods contained in this thesis.

3.2.1 Ecological Perception

J.J. Gibson's work on affordances [70, 69] is a central inspiration of this thesis. Summed up, affordances are possible interactions an animal encounters in its environment. They are specific to the animal and the environment, i.e. the set of present affordances in a given environment is different for each animal.

Affordances were introduced by Gibson [70] and later refined by Gibson [69, Chapter 8] as part of an extensive theory of ecological perception, which we briefly present here: Basically, Gibson argues that animal vision is interconnected with the environment, which consists of media (transmits light and allows motion, e.g. air), substances (no light transmission and movement, e.g. rock) and surface (the border between substances and media). The environment is not measured in physical units but in units that are meaningful to a specific animal. In his view, perception is direct and all necessary information is provided by ambient light emitted from the environment. An observer perceives this ambient light through the optic array, which changes when the observer moves. During the motion some properties of the optic array remain constant, which are consequently called invariants of structure. Examples of invariants are the horizon and the increase in texture density from close to the horizon. Changes in the environment are caused by ecological events involving changes in the surface structure which

are meaningful to animal.

Embedded in this ecological framework, Gibson proposes the term *affordance* mentioned above. Any constituent of the environment can afford something to an animal: A substance, medium, surface, other animal or object. The latter is differentiated between attached and detached. A detached object is entirely surrounded by a medium (e.g. air) and can be moved while an attached object cannot be moved without causing damage. Affordances must not necessarily imply positive opportunities but also negative ones. For example, a knife affords cutting a finger. Gibson defines the existing term ecological niches of an animal as a set of affordances highlighting the complementary relation between animal and environment. Further ideas of Gibson involve information pickup and visual awareness but are not of central relevance to this work and are therefore skipped.

In the context of this thesis it is important to note that Gibson's theory of ecological perception rejects the existence of intermediate 3D representations of the world and argues in favor of direct perception. This is to some extent in accordance with our approaches where we infer actions from 2d sensory input. However, within the networks we employ there exists intermediate representations of the input which might not be consistent with Gibson's radical rejections of any mental or neural pictures. Gibson rejects static images and argues that information pickup is a continuous process. While most of our work considers static images only, this problem is addressed by us in the article on semantic video anticipation in Chapter 7. Moreover, in Chapter 5 we will show an approach to affordance segmentation. For a review of follow-up work building on the seminal theory of affordances by Gibson we refer to Chapter 5.2.1. A more elaborate discussion on the affordance term and a review on the use of affordances in the context of robotics was conducted by Zech et al. [245]. They discuss different notions of the term affordance that have emerged over the years and structure existing work on affordances in the field of robotics according to a novel taxonomy.

While Gibson is a proponent of anti-representationalism (naïve realism), many scholars argue in favor of the existence of internal representations. A notable example is the action-oriented representation theory by Mandik [140]. For a more elaborate discussion on these philosophical and psychological questions we refer to Zech et al. [244, Section 2.1 and 2.2].

3.2.2 Active Vision and Active Perception

The active vision and active perception paradigms acknowledge the fact that human (and animal) vision does not operate on static images but is active. The key idea is that the image acquisition process can be actively controlled, potentially using information gathered from previous observations. This requires an agent that is capable of controlling its image acquisition

parameters through processing previous inputs. It is important to note that active vision is goal-driven. This means, there exists a target which drives the data acquisition process.

Active Vision While earlier systems implemented similar ideas to active vision, it was formalized by Bajcsy [13] and by Aloimonos, Weiss, and Bandyopadhyay [8]. In both articles similar concepts are proposed which both fall under the description provided above. The active perception of Bajcsy [13] involves a controlled and intelligent acquisition of data. Aloimonos, Weiss, and Bandyopadhyay [8] took a more practical approach and highlighted the benefits of active vision for a set of low- and intermediate-level problems, such as shape from texture and structure from motion. An extensive survey on the development of active vision since its inception in the late 1980s until today was recently conducted by Bajcsy, Aloimonos, and Tsotsos [14]. They compare various approaches to active perception and related topics according to a five components scheme. Recent implementations of active vision involve answering questions in simulated 3D environments [51], gradually explore scenes and objects [100] or for feature learning [166].

Interactive Perception Interactive perception [25] is a paradigm closely related to active vision. The underlying idea of interactive perception is that physical (forceful) interactions with the environment are crucial for the interpretation of sensory signals. Bohg et al. [25] argued that making use of structure in the shared space of multiple sensory inputs and action parameters over time facilitates many robotic problems. This includes classic tasks like object segmentation, pose estimation and grasp planning. The difference to active vision is the explicit emphasis on the usage of the modality of force.

The work contained in this dissertation differs from active vision and related ideas in not controlling the data acquisition process. Instead of setting up dedicated lab experiments where control over the acquisition process is possible, we rely on large quantities of pre-recorded data in the wild. Due to the variety of this data, we expect many diverse situations to be covered. However, we agree that the action selection mechanism must be driven by perception.

3.3 KNOWLEDGE REPRESENTATION

> *"Common Sense: The basic level of practical knowledge and judgment that we all need to help us live in a reasonable and safe way."*
>
> Cambridge English Dictionary [29]

Various aspects of this thesis are related to representing common-sense knowledge: Follow-

ing the dictionary quotation from the above, the knowledge about functions of objects can be considered common-sense. Additionally, the ability to judge actions with respect to their plausibility and to anticipate possible futures of a scene is another form of common-sense knowledge.

Common-sense is an essential ingredient in building technical systems that interact with humans through natural interfaces (e.g. speech and gestures) to master everyday problems. One way to put this is: Often instructions uttered by humans are vague and as such ill-posed problems. In these cases, common-sense can act as a regularizer that enables conducting a specific task. E.g. the instruction "put the knife into the dishwasher" is ambiguous if two or more knifes are present in a scene. The knowledge that a dirty knife should be cleaned, and a dishwasher conducts this cleaning is common-sense and allows the correct execution of the ambiguous instruction.

However, injecting this kind of information to computers is extremely difficult as it is mostly learned through experience while growing up and rarely written down. Obviously, a technical system like a robot lacks this experience and must obtain common-sense knowledge through different means. On the other hand, once such knowledge has been acquired, it can, other than in humans, simply be copied from robot to robot, given the robot have an identical embodiment. Acquiring and representing action-oriented common-sense knowledge poses one of the goals of this thesis. Here we review different means to generate and represent common-sense knowledge in general, distinguishing between declarative and functional knowledge.

3.3.1 DECLARATIVE KNOWLEDGE

The term declarative knowledge refers to information in form of facts (sometimes called "knowing that") [46, descriptive knowledge]. In the following, we discuss several approaches to gathering and representing this form of knowledge. Cyc [126] is an ontology designed to capture common-sense facts, which are rarely explicitly mentioned in written text or encyclopedias. It was started in 1984 and successively gathered more knowledge. As a commercial product, Cyc is (at this writing) not freely available. WordNet [145] is organized as a hierarchy of synsets. A synset corresponds to one meaning but can include one or more words, which share this meaning (synonyms). Synsets are linked through relations hypernym (more general) or hyponym (more specific) relations such that hierarchy emerges. WordNet is particularly used in computational linguistics. The newer ConceptNet [132] is a knowledge database that seeks to combine the simple use of WordNet with the richness of Cyc, while being openly accessible. For this, crowd-sourced statements were gathered, automatically processed and inserted into the ontology.

The field of statistical relation learning (SRL) encompasses many methods that combine

symbolic knowledge representation with probabilistic approaches. A popular method is the Markov logic network (MLN) proposed by Richardson and Domingos [170]. It augments first-order logic with a sense of uncertainty and consists of first-order formulae that are associated with an importance weight. A larger weight causes a larger increase in the plausibility of a certain interpretation of the world, if the formula is fulfilled. By combining the MLN with a set of constraints, a ground Markov network can be constructed with each state corresponding to one interpretation of the world. Zhu, Fathi, and Fei-Fei [250] learned an MLN-based knowledge base that involves both, object attributes obtained through a visual feature extractor and affordances. Here, affordances are represented by a name, pose and human-object relative location. During inference this knowledge base is used to infer affordances from unseen objects. As the field of SRL is too large to be covered in its entirety we refer to Koller et al. [112] for a comprehensive review of existing SRL methods.

In contrast to the presented methods, we do not use symbolic but distributed knowledge representations. Actions are directly inferred from the image input without relying on intermediate representations.

Distributed Representations Knowledge can not only be expressed symbolically but also in a distributed way. This means the knowledge is not stored at a distinct location but scattered over the entire model. For factual knowledge, Socher et al. [191] proposed the Neural Tensor Network, that stores facts implicitly in projection parameters and can generate new relations. Nickel et al. [154] conducted a study comparing different techniques to generate new facts from existing knowledge bases through machine learning.

In contrast to these methods, our primary focus is inferring action rather than representing factual knowledge. Unlike our methods, the presented approaches do not use visual data to learn from. However, a commonality is that we store knowledge implicitly in the network's weights, e.g. in the filters of a CNN.

3.3.2 Procedural Knowledge

Procedural knowledge is different from the factual common-sense knowledge reviewed above. It encodes how procedures are conducted instead of being a collection of plain facts about the world (see [46, procedural knowledge]). For an extensive review of action representations specifically in robotics we refer to Zech et al. [244]. They introduce a taxonomy involving many criteria of action representations including perspective, level of abstraction, prediction type. This taxonomy is then used to structure an extensive set of 152 approaches that address this topic. In addition, they contribute to the discussion of what an action constitutes.

Wörgötter et al. [234] and Krüger et al. [119] proposed Object-Action-Complexes (OACs) as a mean to represent how objects and actions interact. Each OAC consists of a description of how the action changes the state of the world (called attribute space) associated with a success probability. See Section 6.2 for a more detailed explanation. Aksoy et al. [5] represented action sequences by a dynamic scene graph, capturing the relations between segments that were obtained using an arbitrary segmentation method. These relations are touching, non-touching, overlapping and no relation. Topological transitions of the graph are tracked in a matrix called the semantic event chain. Recently, Ziaeetabar et al. [251] extended the semantic event chain with static and dynamic spatial relations. Here relations are defined between hand, main object, primary object and secondary object. Note, except for "hand", the method is agnostic to the classes of the objects it operates on. They showed favorable performance in early detection of activities compared to conventional SECs and a hidden Markov model baseline. Following the assumption of touching being a critical cue in the description of actions, Wörgötter et al. [233] proposed an ontology for single handed object interactions. They identified a set of only 30 actions to explain (at least) most one-handed manipulation actions.

Feature-based Representations Kjellström, Romero, and Kragic [111] used histograms of oriented gradient-based features [49] to represent objects as well as motion and pose of the hand as action features to infer object affordances from video. Pirk et al. [167] proposed to represent objects in terms of interaction landscapes. These capture potential trajectories when interacting with a second object. They show the utility of their descriptor on various tasks such as shape classification and saliency estimation.

Our work differs as we mostly deal with static images and do not use touching as an explicit cue for classification. In the work on anticipation in Chapter 7 we learn relations between frames of a sequence are learned instead of explicitly designing them.

CHAPTER 4

Object Semantics in Visual Scenes

The first article in this thesis is positioned at the elementary level by mapping names of object classes to vector representations. To this end, we introduce a system that uses annotated (segmented) images to generate meaningful vector representations of objects. Objects are central ingredients of household scenes and many actions can be defined based on certain objects or aspects of objects. In particular, we show that the generated vector representations do not only exhibit semantic but also functional properties. While grouping of objects based on their semantic qualities is not action-oriented, grouping according to function is. This is because functions imply potential interactions with a specific object in the future. Hence object functions give hints about which actions might be carried out next. Consequently, the article is located at the intersection between descriptive and action-oriented scene understanding comprising aspects of both. It represents the lowest step in our abstraction hierarchy by answering fairly generic questions like: What can I do with a certain object?

Summary Research in linguistics and natural language processing has shown that context can be used to extract the meaning of words. In the following article, we transfer this idea to the visual domain and show that the visual context similarly to the textual context can be used to extract the meaning of objects. Here we define meaningful analogue to recent work in computational linguistics: Similar objects should be represented by vectors that are close to each other. The vector representation is obtained by averaging CNN-based context representations of objects in a scene. We show that this way not only object group according to their super-categories, but also functional classes emerge. Functional classes can be action-related attributes like "can fly" or "has wheels". The following article is published as:

Timo Lüddecke, Alejandro Agostini, Michael Fauth, Minija Tamosiunaite and Florentin Wörgötter

Distributional Semantics of Objects in Visual Scenes in Comparison to Text

Artificial Intelligence (2019)

https://doi.org/10.1016/j.artint.2018.12.009

The following article's presentation was adapted to match the format of this thesis, references are shown at the end of the thesis. The content is identical to the published version.

Timo Lüddecke, Alejandro Agostini, Michael Fauth, Minija Tamosiunaite and Florentin Wörgötter

Distributional Semantics of Objects in Visual Scenes in Comparison to Text

(published in Artificial Intelligence (2019) Volume 274)

Abstract

The *distributional hypothesis* states that the meaning of a concept is defined through the contexts it occurs in. In practice, often word co-occurrence and proximity are analyzed in text corpora for a given word to obtain a real-valued semantic word vector, which is taken to (at least partially) encode the meaning of this word. Here we transfer this idea from text to images, where pre-assigned labels of other objects or activations of convolutional neural networks serve as context. We propose a simple algorithm that extracts and processes object contexts from an image database and yields semantic vectors for objects. We show empirically that these representations exhibit on par performance with state-of-the-art distributional models over a set of conventional objects. For this we employ well-known word benchmarks in addition to a newly proposed object-centric benchmark.

4.1 Introduction

It remains a matter of debate, which aspects constitute the *meaning* of a word or of an object in a scene and the term meaning is heavily discussed in different fields. Here we are specifically concerned with the distributional representation hypothesis by Harris [87], which states that the company of a word determines its meaning (*distributional semantics*). This study sets out to test this hypothesis on images.

For the definition of meaning in the above sense, natural language processing (NLP) uses semantic vectors, which represent word-neighborhoods in a sentence. This approach has proven to be useful in many different applications, e.g. for text translation between different languages [143], for determining the sentiment of a sentence [192] and for question answering [237]. Thesaurus generation, spelling correction, and query expansion count among further applications discussed by Turney and Pantel [208].

Analogously, context in visual scenes is also important for defining the meaning of objects. It might serve as a basis for a variety of approaches in image understanding that involve interactions across multiple objects, e.g. determining useful robotic actions or finding task-relevant

objects in a scene. Thus, in this work, inspired by NLP approaches, we develop methods to obtain semantic vector representations of objects by considering their respective contexts in real world scenes that are composed of multiple objects.

Texts and scenes are akin structures, both being composed of many individual but interrelated constituents: words or objects. The location of a word in a sentence but also that of an object in a scene is subject to some constraints. On the one hand, these constraints can be *fundamental*, e.g. imposed by grammar or physics. The violation of these constrains will render a sentence wrong or make a scene impossible or nonsensical. E.g., grammar forbids two subsequent articles as much as a chair can not stand on the ceiling due to gravity.

On the other hand, obeying only grammatical constraints does not guarantee that a sentence will make sense and the laws of physics will also not guarantee that a room has a useful structure. For making sense also the respective neighborhoods need to be appropriate, i.e. the context in which words are used or the arrangement of items in a room.

Furthermore, context can help to disentangle multiple meanings of words or objects. For example in natural language, the meaning of a polyseme depends on its context and, similarly, multi-functional objects are employed differently depending on the situation. In the same way that e.g. the word "board" acquires a particular sense (out of many) by contextual words, objects surrounding a coffee cup in a scene constrain the set of useful actions involving the cup (e.g. at a coffee klatsch v.s. when the same cup is in the sink with dirty dishes).

However, it is only text analysis where there has been a long history of approaches that address the question of distributional semantics and that derive the meaning of a word, at least to some degree, from context. Interest in these approaches recently revived by the success of large-scale methods [144, 161] exhibiting remarkable performance in judging similarity or analogy of concepts. By contrast, little work has been done on obtaining meanings of objects by considering their scene contexts. This should, however, be possible, in particular due to the large-scale data-sets that have recently been published in the computer vision community, which allow now for the investigation of distributed semantics in the domain of objects. Therefore, it seems justified to investigate the learning of semantic vectors not only of words but also of objects.

In this work we study the hypothesis that the spatial context contributes to the meaning of an object in a scene, analogously to surrounding words defining the meaning of a word. To gather evidence we design an algorithm that extracts semantic vectors from scenes as visualized in Figure 4.1. We aim at analyzing a big enough set of images to arrive at a representation, where semantic vectors for similar objects would group together. Figure 4.1 schematically shows this for $cake$, $spoon$, $fork$ and $knife$, which group together (see schema with object names) but remain separate from $bicycle$, $motorcycle$ and car, which form a different group. Just from common sense one would hope to obtain such clustering results, but there are obvious differences

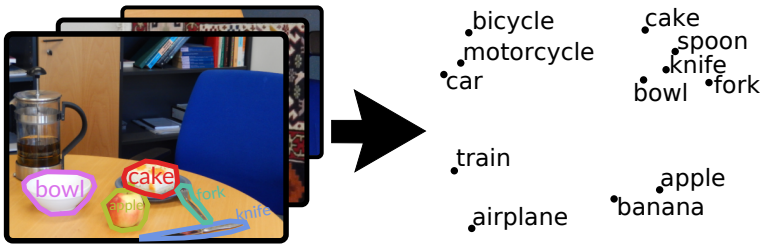

Figure 4.1: By assessing object co-occurrences and visual features we obtain semantic vectors for objects (right, actual output).

between sentences and scenes. In contrast to a scene, a text is a linear structure, i.e. each word has a predecessor and a successor. Thus, while there is the straightforward assumption that the distributional hypothesis should also hold for images, it remains quite a question whether or not the more complex 3D layout of the visual world (or its 2D image projections) might not render context relations too spread out? Hence, we ask: Will scenes provide equally strong context relations than text? In this paper we address this question comparing a large set of different NLP as well as image-based methods.

4.1.1 Overview of the Approach

A schematic introduction to our approach in comparison to linguistic approaches is presented in Figure 4.2. Common distributional models (top) take natural language text corpora, determine word sequences, and generate context vectors by word co-occurrence. A context vector describes the surrounding of individual entities (such as words) by an array of real numbers. Different methods are used to combine such vectors so that finally semantic vector representations emerge for different concepts that can be compared. In essence, our approach (middle) is similar, but it only considers concepts that are objects. We take scene datasets and extract different image descriptors, which are (see numbering in the figure): 1) Human-assigned object labels, 2) object labels automatically obtained by applying an RCNN [71] to detect objects, and 3) CNN activations, which are features generated using pre-trained convolutional neural networks. From all three approaches we extract context vectors for each object in each scene. Context vectors are then merged together to create the unique semantic vector for a specific object class. This allows directly comparing not only the three types of descriptors but also benchmarking our results against automatic NLP-based methods (top) as well as against human rater-based methods (bottom). This is done by measuring semantic distances between objects (the inverse of semantic similarity) in the respective semantic vector spaces.

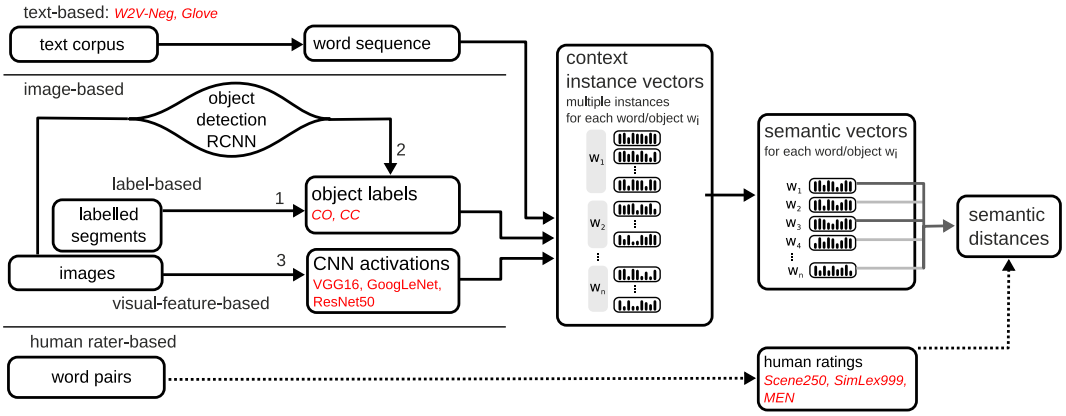

Figure 4.2: Flow diagrams of the analyzes performed here with standard natural language processing, our new approach, as well as based on human labeling. In red, we show the abbreviation of the different analysis methods used (for descriptions see Methods).

4.1.2 Contributions

We propose a simple algorithm to compute semantic vectors for object classes in segmented images and extensively compare this with existing (text-based) methods. This paper is to our knowledge the first to specifically focus on objects in scenes. In addition to the analysis of existing data, we also collected a dataset of similarity and relatedness judgments for 250 object pairs from 20 raters specifically slanted towards everyday objects.

Our analysis shows that the obtained image-based representations are on par with existing text-based representation methods. Quality can be further improved by concatenating different context models. These findings indicate that not only text might serve as a basis for distributional semantics but also visual scenes, which are fundamentally different from text.

The remainder of this paper is structured in the following way. Literature, mostly originating in the natural language processing community, is reviewed in Section 2. Section 3 introduces the theoretical background of our approach and section 4 the experimental methods. In order to assess the validity of the approach, experiments investigating the semantic similarity are carried out in Section 5. Discussion and outlook are provided in Section 6.

4.2 Related Work

There are many approaches originating from diverse fields that can be related to our work. They are discussed in this section and ordered according to the field they come from. Most similar to the idea of this paper are approaches that link distributed semantics from natural language

processing with images. However, the field of natural language processing arguably has the largest impact on this work and is consequently discussed next in more detail. Image labeling is discussed, too, as it links visual with textual data.

Natural Language Processing The idea of representing entities through distributional representations has its origins in linguistics. The distributional hypothesis introduced by Harris [87] established the field of distributional semantics. The idea in distributional semantics is to statistically analyze the distribution of words or other linguistic entities in order to derive a meaning or simply put: "You shall know a word by the company it keeps." [63, 208].

Several methods represent contextual concepts as real numbered vectors, either as an intermediate representation to be used in various NLP tasks [20, 45, 16] or, as we do in this work, as the final output to be assessed with respect to semantic similarity [207].

An elaborate survey has been conducted by Turney and Pantel [208]. The work by Lund and Burgess [138] is particularly relevant as their co-occurrence-based method is closely related to our approach but differs with respect to the derivation as well as the data underlying the methods. A direction of research, which is only laterally addressed by Turney and Pantel [208], is based on parameterizing more complex statistical models by optimizing from random initializations. Bengio et al. [20] proposed to condition the probability of a word on the context of co-occurring words by a neural architecture. This system learns to represent words as vectors and to employ these representations in order to infer the word probability. Recently, starting with the *word2vec* algorithms [144], large scale models have gained a lot of attention due to their ability to extract synonyms and analogies with remarkable quality for numerous words in an unsupervised way from text collections. Subsequent papers discussed further ways of obtaining word vectors exhibiting these properties [161] and related them to traditional approaches [128].

Image Labelling Vector representations are not only employed in natural language processing but also in computer vision. Some image labelling methods learn to map images and labels (i.e. text) into a joint space. One of these approaches is the WARP model of Weston, Bengio, and Usunier [232]. Images are mapped to this space by multiplying a bag-of-visual-word representation of the image, which can be obtained by quantization of features [48]. Learning is carried out by an optimization procedure starting from random initializations for the mappings and then minimizing the hinge rank loss, given ground truth data. Having learned the mappings, unknown images can be mapped into the vector space and, by determining the nearest neighbors, labels can be predicted.

Instead of training from scratch, a more recent approach [67] makes use of transfer learn-

ing. They learn a mapping for images into the word vector space obtained using the skip-gram model proposed by Mikolov et al. [144]. The mapping is carried out by a pre-trained convolutional neural network that is fine-tuned to adapt to the task using a hinge rank loss. At test time, the image is mapped into the joint vector space and its nearest neighbors are evaluated in order to predict a category.

Both approaches differ from our methods because they aim at attributing labels to images based on their visual content rather than obtaining general vector representations of objects. They focus on images presenting a single salient object (which is common in ImageNet) whereas our work is interested in constellations of multiple objects.

Distributed Semantics and Images Linking distributed semantics from natural language processing with images is not a new idea. There has been some work employing labels or tags of images in order to link textual meaning with visual features extracted from the images.

Feng and Lapata [61] propose a method that works on documents with an associated image and experiments are conducted on news articles. For both modalities, a bag-of-word representation is used with the image's representation being obtained through extracting visual words. SIFT descriptors [135], which capture local edge orientations, are extracted from images and matched to visual words, which have been learned previously by k-means clustering of all feature descriptors from all images. In this way, visual words can be treated like actual words in a text, i.e. semantic vectors are assigned to words. A topic model, based on Latent Dirichlet Allocation [24], is trained on documents involving both, textual and visual words. In their experiments, the model is used to measure similarity between words.

Bruni, Tran, and Baroni [27] also employ visual words descriptions of images. However, visual features are computed differently: SIFT is extracted densely (without keypoints) with spatial binning from each channel of a HSV encoded image, making the algorithm sensitive to color. Also, a simple count model is employed rather than a topic model. Visual words are used in conjunction with multiple textual labels (tags) to obtain co-occurrence counts for certain terms. Finally, they are transformed into Local Mutual Information association scores. Multiple models of fusing representations from the modalities are discussed.

Kádár, Alishahi, and Chrupała [102] propose a model to learn meanings of words using images and their caption text, relating words to visual features. Similarly to our work, they make use of pre-trained Deep Convolutional Neural Networks (DCNNs) to extract visual features. A similar approach is pursued by Kiela and Bottou [108]. Rather than relying on captions, they concatenate visual features of multiple object classes of ImageNet [53], which have been extracted using pre-trained DCNNs with the well-known word semantic vectors from Mikolov et al. [144]. Both assess their models on common word similarity benchmarks. Another example

of a DCNN-based approach is by Peterson, Abbott, and Griffiths [163]. They extract features from 120 photographs of animals and observe that the pairwise dot similarity between the features only correlate weakly with human similarity judgments. Consequently, a transformation of the visual features is learned which strongly increases the correlation with human judgments.

All the discussed approaches differ from our work in the aspect that labels or text refer to the entire image rather than only to regions of it. In our setting, due to availability of segmentations, we are able to directly access both, object location in the image and class of the object, which enables us to explicitly focus on the context of objects rather than the features of the object itself and with this we can evaluate the validity of the distributional hypothesis in images.

4.3 Determining Image-Based Context

One of the central assumptions of the approaches discussed above is that context strongly determines meaning. In this section we will, therefore, first describe how we define object context in an image and then how we integrate contexts extracted from separate images to obtain context representations over an image database. We call context obtained for one object in one image *context instance* or *context instance vector* and the context representation of an object integrated over many instances we call *semantic vector*. As our main approach for integration is context averaging, we will also provide a mathematical justification for that. For comparison, we also use the standard skip-gram method and the median for integration. Finally, we explain how we concatenate different features to create fused context representations.

4.3.1 Context Instance Definition

Given a dataset of scenes containing multiple objects, context extraction is applied on each object in each scene to obtain the set of context instances for each object. In this study we used two main approaches to define object instance context in an image: *label-based* context, described in subsection 4.3.1.1 and *visual feature-based* context, described in subsection 4.3.1.2.

4.3.1.1 Label-based

Label-based contexts fully rely on image annotations, i.e. image pixels are not taken into consideration. An illustration of a context description for this case is provided in Fig. 4.3. Three schematic images are shown (left), where objects are represented by abstract shapes. Two ways of determining context are here defined: For each object in the scene all neighboring objects for each of the other object classes are counted. The resulting counts are used as the components

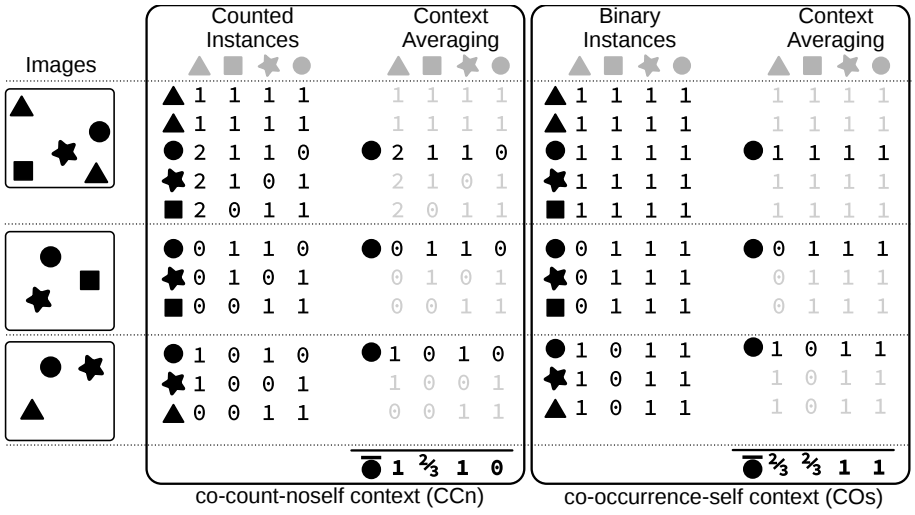

Figure 4.3: The context averaging method considers each object (illustrated by shapes) in each image and determines its context. Subsequently, the averaged context is calculated across all objects and used as a semantic vector. Contextual instances can either be counted (co-count) or just indexed (co-occurrence) and the reference object can be counted among the context (self) or not (noself).

of a vector, which is then taken as the instantiation of this object's context. We call this model *co-count* (CC, see column called "Counted Instances" in the left box in Fig. 4.3). Alternatively, we can neglect the number of objects in the scene and just tick-mark every existing object creating a binary (there vs. not-there) vector. This model will be referred to as *co-occurrence* (CO, see column called "Binary Instances" in the right box in Fig. 4.3). The reference object (the object we currently consider) can be counted among the context or not, which we call s or n (*self* or *noself*, Fig. 4.3, right vs. left). Hence for a label-based vector v holds: $v \in \mathbb{N}^N$ with N being the number of object classes in the dataset, in case of co-occurrence even $v \in \{0,1\}^N$.

4.3.1.2 Visual feature-based

An alternative, yet also image-based, way of describing object context is through visual features. While multiple approaches to extract visual features from an image are possible, e.g. bag-of-visual words [48], recent years have been dominated by the success of convolutional neural networks (CNNs), where activations in specific layers can be considered as an abstract representation ("features") of the image that was fed to the CNN. Here we constrained our analysis to four CNNs which were trained on ImageNet [53]: *InceptionV1 (GoogLeNet)* [201], *InceptionV4* [202], *VGG16* [189] and *ResNet50* [88]. In contrast to label-based vectors, visual feature-based vectors are more general. If o is an object with an associated image then the visual feature-based

context $\mathbf{f}$ of o is defined as the result of the sequence of the interleaved matrix multiplications, non-linearities, and pooling operations performed in the CNN up to the layer from which $\mathbf{f}$ is extracted. Visual-feature-based context is described as a vector of real numbers as opposed to a vector of natural numbers for the label-based context, above. Technical details of the employed CNNs are shown in Table 4.1, where *Feature Size* indicates the number of features we were actually using.

Network architecture	Feature Vector Layer	Feature Vector Size	reported top5-accuracy
InceptionV1	AvgPool_1a_7x7	1,024	89.6 %
InceptionV4	global_pool	1,536	95.2 %
ResNet50	global_pool	4,096	89.8 %
VGG16	fc7	2,048	92.8 %

Table 4.1: Comparison of the employed pre-trained CNNs. The layer from which features are extracted, the number of features as well as top-5 accuracy achieved in the ImageNet classification challenge are provided.

Following the distributional hypothesis [87], we constrained our analysis to the image region that pertains to the context of the object but exclude the object itself. In the visual feature-based case this means that we extract CNN features after covering the reference object with black box mask, such that not even the shape of the object is visible. In case there are multiple objects considered all this is done for every one of them.

From a theoretical perspective, we expect a black box in the image to inhibit the features of the masked area and this way emphasizing the features extracted from the unmasked area – the context. If the convolution window contains both, mask and contents, there might be inference. However, this is a bias that affects the “context” of every object and even if "wrong" features are now introduced, they are canceled out when analyzing the differences (distances) between two object representations. Actually, we experimented with an object-shaped (instead of rectangular) black mask, which yielded similar results.

4.3.2 Context Integration

Context integration is used to obtain the semantic vectors. Three methods are compared: context averaging, skip gram based context integration, and the use of the median instead of the average.

4.3.2.1 Context Averaging

One way to obtain the semantic vector of an object is to calculate the element-wise average of the context vectors obtained for individual object instances. This seems overly simple but it is in fact the maximum likelihood estimator of the underlying distribution(s) of objects in images, as shown below.

Let the set $\mathcal{O}(k)$ denote all object instances of class k in the dataset. $\mathbf{f}$ is any context extraction function that considers an object's context and returns a real-valued vector representing this context. Examples of such functions are described in 4.3.1.1 and 4.3.1.2. The vector representation $\hat{\mathbf{\Lambda}}_k$ for concept k is then obtained by this formula:

$$\hat{\mathbf{\Lambda}}_k = \frac{1}{|\mathcal{O}(k)|} \sum_{o \in \mathcal{O}(k)} f(o) \tag{4.1}$$

The averaging method is illustrated in Fig. 4.3 for the "black-disk-object" $\bullet$. Each highlighted row corresponds to the context vector of the black disk for the corresponding schematic image on the left side, hence to a vector $\mathbf{f}(o)$ in equation 1. In the depicted case $k = \bullet$ the semantic vector $\hat{\mathbf{\Lambda}}_\bullet$ is obtained by averaging the individual context vectors.

For obtaining semantic vectors when using visual-feature-based analysis the real-numbered context instance vectors are averaged in the same way to obtain object class representations.

Justification for context averaging The averaging of context can be justified by imagining a scenario where the scene is scanned from the perspective of an object o and considering each time another object is seen as an event, which is dependent on the object class of o. In the co-count model (CC), the number of such events (per object class) is described by integer random variables, while in the co-occurrence model (CO) the random variables refer to the probability of the event. Consequently, the underlying distributions are for co-count *Poisson* and for co-occurrence *Bernoulli*. For both distributions, averaging all observations is equivalent to the maximum likelihood estimation of the distribution's parameters. Hence, we implicitly define the meaning of objects to be optimal parameters of random distributions over visual features or object labels of given observations in a large scale dataset.

The theoretical motivation for averaging rests on the following arguments first discussed for the co-count model. As stated above, scanning a scene under the co-count model renders that the number of contextual objects of type i given the object class k is Poisson distributed, i.e. $\mathbf{c}_i \mid k \sim Poisson(\mathbf{\Lambda}_{ki})$ with each $\mathbf{c}_i \in \mathbb{N}_0$. Hence, the corresponding likelihood function is

$$\mathcal{L}(\lambda_{ki} \mid \tilde{c}_{ki}^{(0)}, ..., \tilde{c}_{ki}^{(n)}) = \prod_{j=1}^{N} \frac{\lambda^{\tilde{c}_{ki}^{(j)}} e^{-\lambda}}{\tilde{c}_{ki}^{(j)}!}$$

with $\{\tilde{c}_{ki}^{(j)} \mid j = 1, ..., N_k\}$ being the observed counts of object i in the context of object class k. Then the maximum likelihood estimator for each λ_{ki} is the sample mean.

An alternative model is the co-occurrence model. Here, we only mark object existence. Hence, we assume that encountering one or more objects of class i in the context of class k has a certain success probability. This leads to $P(\mathbf{c}_i \mid o = k) \sim Bernoulli(p_{ik})$ with each $\mathbf{c}_i \in \{0, 1\}$ and the likelihood function

$$\mathcal{L}(p_{ki} \mid \tilde{c}_{ki}^{(0)}, ..., \tilde{c}_{ki}^{(n)}) = \prod_{j=1}^{N} p^{\tilde{c}_{ki}^{(j)}} (1-p)^{(1-\tilde{c}_{ki}^{(j)})}$$

Also here a maximum likelihood estimation of the parameters p_{ki} is represented by the sample mean. In the Appendix we provide additional derivations concerning these aspects.

4.3.2.2 Skip-Gram-based Context Integration

In addition to the counting-based averaging described above we implemented a skip-gram-based context integration method, which originates from natural language processing and is considered to be state-of-the-art therein. Its goal is to generate representations of words that, given one word from a corpus, allow for the prediction of contextual (surrounding) words. This algorithm serves as the basis for the word2vec [144] programs, which have been shown to yield very good semantic word representations.

We implemented the skip-gram negative sampling algorithm [144], which approximates the skip-gram target in a computationally efficient way, by using a fixed number of negative-sample-pairs. This method is only compatible with co-occurrence (CO) context. In our case, positive samples are obtained from pairs of objects within an image. Training is carried out for 50 epochs with a learning rate of 1. For every positive sample, 25 negative samples are presented and the dimension of the concept vectors is 64. A detailed exploration of hyperparameters leading to values reported above is provided in section 4.10.6 in the Appendix.

4.3.2.3 Median-based Context Integration

In general, there would be many more integration methods possible and we had to limit ourselves to some of the most common ones, where using the median appears another possible "natural" choice. Results are not impressive and therefore we show this only in the Appendix to not overburden the main text.

4.3.3 FUSION OF CONTEXTS

Label-based and visual feature-based contexts encode different information. The former tells us something about co-occurring object classes using a high-level (symbolic) description, while the latter addresses visual appearance (features) of the context-objects. Hence, it appears natural to combine both and see whether the semantic quality improves. While there exist various possible methods, we will emphasize on concatenation. The advantage of this method is that it does not require strong assumptions on the structure of the context vectors, i.e. it can be used to combine any number of real-valued vectors, also in case they have different length.

Concatenation can be applied at two levels: Given a scene and one object in that scene, we can apply several context extraction methods, e.g. label-based context and visual-feature-based context and concatenate both context vectors. This procedure we call *early fusion*. Alternatively, we can integrate contexts into semantic vectors using the two methods independently and then concatenate the obtained vectors; e.g., we first average all visual feature-based contexts of "car" and average all label-based contexts of "car" and then concatenate the resulting two vectors. This we call *late fusion*.

Let $\mathbf{f}$ and $\mathbf{g}$ be two context extraction methods, $O(k)$ the set of all object instances of class k in the dataset, and ϕ denote the context integration function (e.g. averaging).

Then we can formally define early fusion as

$$\Lambda_k^{early} = \phi(\{\mathbf{f}(o) \mid\mid \mathbf{g}(o) \mid o \in O(k)\})$$

with $\mid\mid$ being the concatenation operation which involves normalization using mean and standard deviation of each vector:

$$a \mid\mid b = (\frac{a - \mu_a}{\sigma_a}, \frac{b - \mu_b}{\sigma_b}).$$

Late fusion is defined by:

$$\Lambda_k^{late} = \phi(\{\mathbf{f}(o) \mid o \in O(k)\}) \mid\mid\mid \phi(\{\mathbf{g}(o) \mid o \in O(k)\}).$$

with $\mid\mid\mid$ also being a concatenation operation but here normalization is made in a different way. Let a be the semantic vectors obtained by the first of the two to-be-fused context extraction methods for each object included in the study. Then we arrange those semantic vectors into matrix A row-wise (each vector is one row). Analogously, into matrix B semantic vectors obtained using the second context extraction method are arranged. From those matrices we obtain the vectors of the means: $\mu(A)$ and $\mu(B)$, and their standard deviations: $\sigma(A)$ and

Image Training datasets			
Dataset	#Classes	#Scenes	#Instances
COCO	80	82,081	604,907
LM	3,288	28,072	325,288
RCNN	79	111,974	286,835

Text Training datasets		
Method	Dataset	#Words
W2V-Neg	GoogleNews	100B
Glove	Wikipedia + Gigaword 5	6B

Human-rater datasets		
Name	#Words	#Pairs
Scene250	80	250
Simlex999	1028	999
MEN	751	3000

Table 4.2: Overview of image datasets used to obtain vector representations. LabelMe dataset abbreviated as LM.

$\sigma(B)$ now column-wise, so that each semantic vector is zero-centered and normalized. Then the concatenation operation uses this normalization:

$$a \,|||\, b = (\frac{a - \mu(A)}{\sigma(A)}, \frac{b - \mu(B)}{\sigma(B)})$$

where subtraction and division operations are performed element-wise.

In section 4.10.2 in the Appendix we introduce and analyze more fusion techniques.

4.4 Standard Methods from NLP

One major goal of this study is to compare our new image-based methods against state-of-the-art text-based methods for the extraction of semantic vectors from NLP (see top in Fig. 4.2). For this we used Glove [161] (50 dimensions, 6 billion tokens) and the skip-gram negative-sampling method (part of Word2Vec toolkit, 300 dimensions, 100 billion words, Google News dataset) [144]. In both cases, we downloaded pre-computed semantic vectors provided by the authors from the Internet. As these are standard methods, we do not extend explanations here and instead refer the reader to the respective references [144, 161].

4.5 Datasets

4.5.1 Image Datasets

Since our method requires annotated image data, we started our experiments using datasets offering object bounding boxes as well as object labels for each box. However, to show that our

method can also be used in case where only raw images are available, we took an additional dataset that is initially not annotated and we used an advanced off-the-shelf object detection-and-labeling algorithm to perform automatic labeling. Specifically, we used COCO [130] and LabelMe [177] as annotated source datasets. We created the additional, automatically labeled dataset, which we will call RCNN through-out this paper, from a combination of images from the COCO and ImageNet [53] datasets. Dataset statistics are presented in Table 4.2.

4.5.1.1 Manually annotated datasets

Both, COCO and LabelMe provide annotations in form of object segments with corresponding object names. The COCO dataset has 7.4 object labels per scene on average and the LabelMe has 11.6 object labels. The object labels in COCO are very reliable, but the dataset contains only 80 object classes, while LabelMe provides a more diverse set of classes but has a tendency to contain wrong or nonsensical labels due to being crowd-sourced. Because of this, we only consider object labels that occur at least three times in LableMe, which yields 3288 object classes.

4.5.1.2 Automatically annotated dataset, RCNN

We have created the RCNN dataset by applying the object detector *Mask R-CNN* which is a pre-trained neural network taken from [89]. Mask R-CNN takes a raw image as input and automatically generates labeled bounding boxes for objects detected in the scene. This way an annotated dataset is created, which is analogous to the manually labeled datasets introduced in the subsection 4.5.1.1 above. We used on purpose the same initial dataset, namely COCO, to create RCNN, to allow for a direct comparison with the results from the human-labeled data. We were, however, forced to use the smaller *testing set* of COCO, because He et al. [89] had trained Mask R-CNN on the COCO-training set. Thus, to increase the resulting RCNN dataset, we added a subset of ImageNet images. With this, we processed 160,000 images in total and Mask R-CNN found objects in 111,974 images therein, because of a high threshold we applied for reliable object detection. By this procedure, 79 object classes were discovered from the possible 80 COCO objects classes for which the Mask R-CNN was pre-trained. The average number of discovered objects per scene is 2.5.

4.5.2 Human rater-based Datasets (Ground Truth Benchmarks)

To compare our scene-based analysis with the corresponding results obtained in natural language analysis, we conducted extensive experiments on common human rater-based bench-

marks from NLP including a newly introduced benchmark specifically targeting everyday objects.

All these benchmarks rely on word-pair comparisons assessed by human raters. In general, two relations are measured: *similarity* and *relatedness* [90]. *Similarity* refers to shared properties between the two words (concepts) in a pair; e.g. a train and a car both have wheels, move on their own and host passengers. On the other hand, the more general aspect *relatedness* incorporates additional relationships [28], like function or meronymy; e.g. a remote control is used to turn on the TV, hence, both are related but not very similar.

4.5.2.1 Existing word pair-based benchmarks

Here we used the MEN ([27], 3000 word pairs) and SimLex999 ([90], 999 word pairs) benchmarks. Naturally, we had to restrict all analyses to the intersection of words between benchmark- and image datasets, which is rather small (for quantification of intersection see section 4.10.3 in the appendix).

4.5.2.2 Novel object name-based benchmark

Due to the small intersection found for the common benchmarks, we decided to create our own benchmark dataset, called *Scene250*, which consists of a randomly sampled subset of 250 pairs from the possible 3160 pairs made by the 80 COCO objects. This way, *Scene250* fully intersects with COCO and also to a large degree with LabelMe (see section 4.10.3). We asked 20 adult raters (10 male, 10 female) to indicate the degree of object similarity and relatedness in those pairs using a scale from zero to ten. The concepts of similarity and relatedness were explained to the participants based on definitions and examples. For obtaining the final score we averaged the scores from all 20 participants. The average pair-wise Spearman correlation coefficient across raters was 0.544 for similarity and 0.659 for relatedness.

The *Scene250* dataset is available for download at:
`http://www.dpi.physik.uni-goettingen.de/cns/index.php?page=coco-wordsim`.

4.6 MEANS FOR QUANTIFICATION

We introduced two quality measures to evaluate obtained semantic vectors: (1) clustering consistency and (2) system- to-human correlation.

4.6.1 CLUSTERING CONSISTENCY

This measure is based on the organization of the 80 object classes contained in the COCO dataset into eleven super-categories, like animal, vehicle, kitchenware, etc [130]. In an ideal case, the closest neighbors of a given semantic vector would all belong to the same super-category. We measure to which proportion this is satisfied. Specifically, for every object o in a super-category, we consider its semantic vector and then find the k nearest neighbor semantic vectors. Of those k vectors we count the number of objects that fall into the same super-category as o and normalize the count by k. Then we average results for all o in that super-category, for all super categories and, finally, we average scores for $k = 1$ to $k = 5$ (5 is the size of the smallest super-category in COCO) to obtain a single scalar score. For COCO we have 80 object classes in an 80-dim semantic vector space. To remain comparable, we are also evaluating LabelMe using a compatible subset of 60 classes occurring in LabelMe from the 80 COCO classes, but now the semantic vectors have 3288 instead of 80 dimensions. The clustering consistency measure is based on similar grounds as the most common standard classification accuracy metrics for a k-nn classifier, like e.g. precision and recall. To keep the main text concise, we discuss this relation in more detail in subsection 4.10.4 in the Appendix.

4.6.2 SYSTEM-TO-HUMAN CORRELATION

For this we compared similarity (relatedness) scores of word-pairs determined by human raters, using the benchmarks MEN, Simlex, and Scene250, with ratings for the same pairs calculated by the different automatic procedures. This can be best understood by a simple example.

Let us assume that our benchmark dataset consists of five word-pairs, which are here listed in descending order according to their human-determined similarity (rank-order list):

motorcycle-bicycle, train-motorcycle, car-train, train-orange, orange-airplane.

From the vector representations of our concepts, calculated by – say – *VGG16*, we can compute the semantic vector *distance* between word pairs. Semantic distances could be: car-train: $d = 0.2$, orange-airplane: $d = 0.8$, motorcycle-bicycle: $d = 0.1$, train-orange: $d = 0.9$ and train-motorcycle: $d = 0.3$. For comparison, we – thus – use *inverse distance* and we get the following rank-ordered list:

motorcycle-bicycle, car-train, train-motorcycle, orange-airplane, train-orange.

Both lists are not identical but clearly correlated, which we can quantify using the Spearman

rank correlation coefficient, for which we get in this example a value of 0.8.

This Spearman rank correlation calculation has been performed for *COn*, *CCn*, *COs*, *CCs*, *InceptionV1*, *InceptionV4*, *VGG16*, *ResNet50*, *Skip-Gram* and *Random* vectors against MEN, Simlex, and Scene250, whenever there were enough word-pairs existing in *both* datasets (see section 4.10.3 for quantification of word-pair intersections).

Note that all automatic procedures will calculate only one distance value for any given pair. Thus, we compared the resulting inverse distance rank list to *both*, similarity as well as relatedness, rank lists from the human raters.

4.7 EXPERIMENTS

This section reports the quantitative results of our experiments. Regarding the measure of semantic distance, we found that there are no consistent differences between results obtained with Euclidean versus cosine distance. Both measures render the same results with usually less than $\pm 10\%$ differences. Therefore, we show only results for the cosine distance, which is the one commonly used in the relevant literature.

In the following we show that our visual-feature based semantic vectors match the quality of state-of-the-art text-based methods for everyday objects despite being extracted from different data with a much smaller number of samples.

4.7.1 COMPARISON OF CONTEXT MODELS

The first experiment (see Fig. 4.4) is used to compare different context models and here first and foremost label-based context vs. visual feature-based context. Performance is analyzed using semantic vectors that we had we obtained from the three scene datasets (from top to bottom): COCO, LabelMe and RCNN. Number of classes, scenes and contexts in those datasets are provided in Table 4.2 in section 4.5.1. Label-based contexts are annotated as *CO* or *CC* (corresponding to Co-Occurrence and Co-Count as discussed in section 4.3.1). With subscripts *s* or *n* we indicate whether the reference objects is excluded from the context description (s means self-included, n means non-self-included). Visual-feature-based contexts are obtained by the CNNs *VGG16*, *InceptionV1*, *InceptionV4* and *ResNet50*. All the mentioned contexts are obtained using averaging for context integration. Skip-gram performance for the context COn is provided as a baseline, for comparison with the averaging-based methods. Chance performance is indicated for each case, too (see label "Random"). For the skip-gram and the chance columns in Fig. 4.4 the ten samples, from which the average is obtained, are shown by horizontal lines on

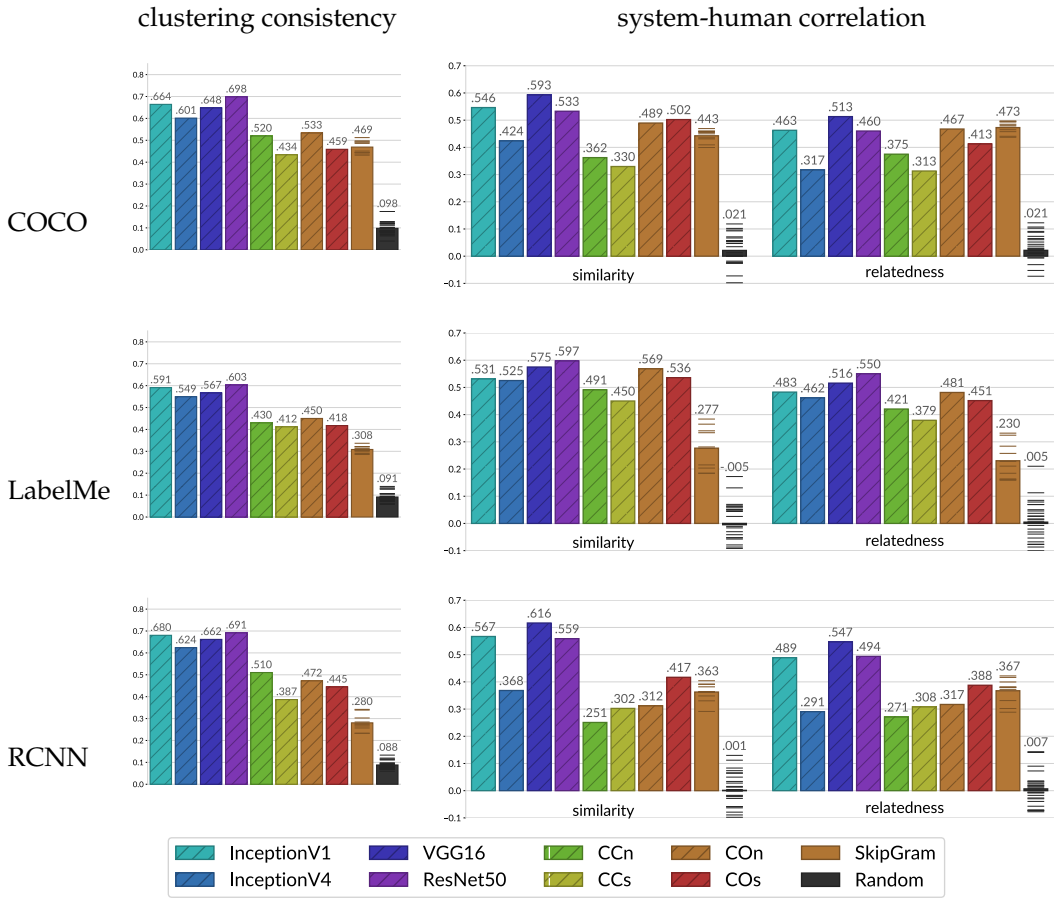

Figure 4.4: Experimental evaluation of different context models (given by different colors, see legend at the bottom of the figure) on COCO (top), LabelMe (middle) and RCNN (bottom). On the left side clustering consistency is shown. On the right system-to-human correlation for Scene250 data measuring similarity (sim) and relatedness (rel) are shown.

the corresponding column. The remaining columns do not rely on any randomized sampling.

All plots on the left show the *clustering consistency* measure for the different methods and the ones in the center and to the right show the *system-to-human correlation* as described above. For *clustering consistency*, neighbors in the semantic vector space are analyzed based on the existing division of the 80 COCO object classes into 11 super-categories as given by the developers of this dataset. System-to-human correlation is based on our newly introduced Scene250 benchmark. One can observe that all measures for all contexts show clear above-chance performance. Different measures show similar rank-order for the analyzed contexts, which also holds between the COCO and LabelMe datasets. RCNN shows a bit lower scores when measuring system-to-human correlation for label-based contexts. However, RCNN is the one dataset with the

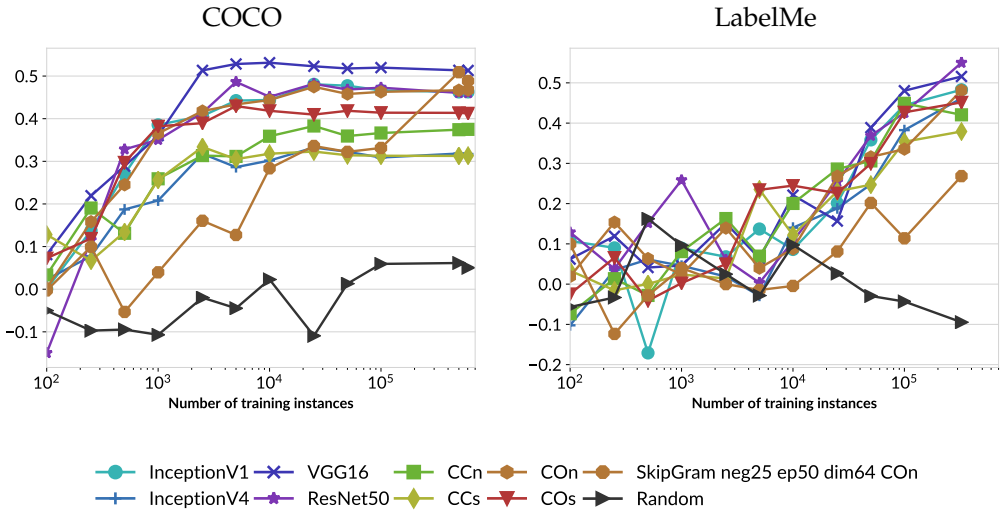

Figure 4.5: Plot of the Spearman correlation coefficient with similarity ratings in relation to the number of instances used for training.

sparsest labels: only 2.5 labels per image on average as reported in the method section above. Apparently, object density is still enough to support visual-feature-based context analysis, but is too low for reliably analyzing label co-occurrences.

Furthermore, we find that the skip-gram negative sampling baseline (based on COn, see 9-th column in each plot) is performing almost always worse than the corresponding COn context obtained through simple averaging (7th column). The exception is the RCNN dataset where sparsity of labels seem to make evaluation of label-based contexts less stable, as discussed in the paragraph above. Relatively low skip-gram performance might have several reasons: In contrast to text corpora where skip-grams are employed on very big datasets, our dataset have a fairly small amount of labels. Also, negative sampling requires many hyper-parameter choices (number of negative samples, sampling strategy of negatives) and there is some chance that there are better configurations which we did not find despite a fairly extensive search (see subsection 4.10.6 in the Appendix).

Surprisingly we found that visual feature-based contexts (specifically, look at *ResNet50*) exhibit better performance than label-based ones. Hence, it appears that CNN-generated visual features have a stronger association with the meaning of objects than manually assigned class labels, where — in addition — the latter is also more expensive to obtain. In general, best performance is found for context model *CO*(*s* and *n*) and for feature-based model *ResNet50* and *VGG16*. Thus, for further evaluations we put an emphasis on these methods.

Concerning *system-to-human correlation* (central and right column) we find that our system

	Clustering	Scene250	
	consistency	sim	rel
MSCOCO (80 concepts)			
COn	0.533	0.467	0.489
VGG16	0.648	0.513	0.593
ResNet50	0.698	0.460	0.533
LateFusionNorm: VGG16+COn	0.680	0.643	0.685
EarlyFusion(norm): VGG16+COn	0.656	0.579	0.645
LateFusionNorm: ResNet50+COn	0.738	0.663	0.696
EarlyFusion(norm): ResNet50+COn	0.702	0.522	0.587
RCNN (79 concepts)			
COn	0.472	0.317	0.312
VGG16	0.662	0.547	0.616
ResNet50	0.691	0.494	0.559
LateFusionNorm: VGG16+COn	0.717	0.653	0.704
EarlyFusion(norm): VGG16+COn	0.690	0.614	0.681
LateFusionNorm: ResNet50+COn	0.726	0.686	0.715
EarlyFusion(norm): ResNet50+COn	0.713	0.554	0.614

Table 4.3: Fusion of contexts extracted from COCO and from RCNN.

most of the times produces slightly bigger system-to-human correlations for similarity than relatedness. We get a maximal correlation of 0.616 for similarity and of 0.550 for relatedness. Correlation *between* humans is 0.544 for similarity and 0.659 for relatedness. This is to some degree remarkable because it shows that this system performs very close to the human level.

An additional, important observation is that the RCCN dataset, which is automatically labeled, together with neural net methods (visual-feature-based approaches (*VGG16*, *InceptionV1*, *ResNet50*, left bars) will lead to scores that are even higher than those from the COCO dataset. This is quite remarkable as automatic labeling using Mask R-CNN is less reliable than human-labeling. These findings demonstrate the impressive progress in object detection in recent years and suggest that applying an object detector as a pre-processing step is a viable solution towards a system that handles a huge number of scenes, beyond a quantity that can annotated by humans. In the future, performance could be improved simply by plugging-in a more advanced object detection algorithm. In particular it would be interesting to see a larger set of objects being covered.

4.7.2 SCALE: DEPENDENCY ON THE SIZE OF THE DATASET

Under the assumption that quality increases with number of data points, a valid question is to ask whether quality has already reached ceiling with the here-used number of scenes or whether we can expect it to further increase given a larger dataset. Since the number of considered object classes might be a crucial factor, we determine the scaling behavior for the COCO dataset as well as for the more diverse LabelMe dataset.

Figure 4.5 suggests a different answer for both datasets: the results that have been obtained from the COCO dataset (80 different objects and 604,907 context instances (see 4.3.1)) reach ceiling already at 1000 analyzed contexts (not scenes) while, when using LabelMe (3288 different objects), there is no clear saturation visible even at 325,288 analyzed instances, which are all the instances available for this dataset. This can be expected since the number of context vector dimensions for LabelMe is much larger (due to a higher number of possible contextual objects) and this allows for more configurations of the context, requiring more examples for convergence.

4.7.3 FUSION OF CONTEXTS

Table 4.3 reports fusion results on the supervised COCO dataset as well as the automatically annotated RCNN dataset. We apply fusion of the context COn (which is among the best label based contexts as shown in Figure 4.4) with visual-feature based contexts VGG16 and ResNet50. It can be seen that fusing of the two types of contexts improves both clustering consistency as well as system-to-human correlation both for similarity and relatedness, in both datasets. Hence, fusion turns out to be an inexpensive yet powerful method to increase quality. Late fusion provides consistently better scores than early fusion. Thus, we will be using late fusion further, in comparing our methods to the state-of-the-art.

4.7.4 COMPARISON TO STATE-OF-THE-ART

How does our image-based method compare to automatic text-based methods from NLP such as Word2Vec, Glove, etc? This comparison is made in two ways: 1) We determine and compare clustering consistency for the different methods and 2) we ask how do all the methods behave relative to the human-rater based ground-truth data from the MEN, Simlex999, and Scene250 datasets. Hence, for this we ask specifically, whether the Spearman correlation coefficient between image-based vs. human rater based is bigger or smaller than the one between NLP-based vs. human rater-based.

	Clustering Consistency	Scene250 sim	Scene250 rel	SimLex999	MEN
COCO	64 concepts	165 pairs		9 pairs	3 pairs
ResNet50	0.682	0.459	0.548	-	-
LateFusionNorm: ResNet50+COn	0.670	0.702	0.744	-	-
LateFusionNorm: ResNet50+COn+W2V-Neg	0.693	0.726	0.766	-	-
VGG16	0.631	0.537	0.626	-	-
LateFusionNorm: VGG16+COn	0.654	0.689	0.740	-	-
LateFusionNorm: VGG16+COn+W2V-Neg	0.678	0.703	0.754	-	-
Glove	0.590	0.666	0.744	-	-
W2V-Neg	0.716	0.707	0.742	-	-
KielaMax	0.580	0.435	0.494	-	-
KielaMean	0.592	0.520	0.574	-	-
LabelMe	60 concepts	151 pairs		166 pairs	626 pairs
ResNet50	0.641	0.588	0.617	0.271	0.343
LateFusionNorm: ResNet50+COn	0.647	0.681	0.727	0.230	0.431
LateFusionNorm: ResNet50+COn+W2V-Neg	0.663	0.711	0.755	0.290	0.545
VGG16	0.606	0.567	0.606	0.297	0.354
LateFusionNorm: VGG16+COn	0.614	0.666	0.716	0.241	0.404
LateFusionNorm: VGG16+COn+W2V-Neg	0.622	0.687	0.737	0.268	0.461
Glove	0.587	0.704	0.752	0.289	0.679
W2V-Neg	0.715	0.727	0.751	0.463	0.776
KielaMax	0.589	0.462	0.506	0.430	0.573
KielaMean	0.599	0.549	0.598	0.366	0.608
RCNN	64 concepts	165 pairs		9 pairs	3 pairs
ResNet50	0.631	0.521	0.590	-	-
LateFusionNorm: ResNet50+COn	0.677	0.724	0.754	-	-
LateFusionNorm: ResNet50+COn+W2V-Neg	0.719	0.743	0.772	-	-
VGG16	0.605	0.591	0.661	-	-
LateFusionNorm: VGG16+COn	0.677	0.708	0.751	-	-
LateFusionNorm: VGG16+COn+W2V-Neg	0.703	0.718	0.763	-	-
Glove	0.590	0.666	0.744	-	-
W2V-Neg	0.716	0.707	0.742	-	-
KielaMax	0.580	0.435	0.494	-	-
KielaMean	0.592	0.520	0.574	-	-

Table 4.4: Comparison of our representation with state-of-the-art methods on various benchmarks, from left to right: Clustering consistency, Spearman correlation for relatedness (rel) and similarity (sim) ratings using the Scene250 benchmark, Spearman correlation for similarity using SimLex999 and for relatedness using MEN.

For comparison we use the purely text-based approaches Word2Vec skip-gram negative sampling trained on Google news [144] (W2V-Neg) and Glove 6B [161] trained on Wikipedia and Gigaword 5 (*Glove*). We also use multi-modal text- and image-based methods proposed by Kiela and Bottou [108] (*KielaMax* and *KielaMean*).

This is compared to our methods where we use as a basis *ResNet50* and *VGG16*, which are the best performing methods according to previous experiments and extend the comparisons to different combinations of these base-methods by using Late Fusion with co-occurrence context without self-counting (*COn*) as well as in addition Late Fusion with the NLP-method skip-gram negative sampling whose vectors were computed on Google News (*W2V-Neg*).

Note that any considered comparison could only be made for the smallest common subset of concepts (or word pairs), which corresponds to the respective intersection of the datasets used in that comparison. This is always an intersection between three sets, for example COCO with Scene250 and the data used in Glove, etc. In section 4.10.3 in the Appendix we explain, how the intersection subsets look like. Due to the fact that RCCN is a descendent of COCO, their intersections are always the same.

Table 4.4 shows the results.

4.7.4.1 Clustering consistency

We evaluate clustering consistency on the intersection of categories between scene datasets (COCO, LabelMe, or RCNN) and state-of-the-art implementations (*Glove, W2V-Neg* and *Kiela*). As before, we use the eleven super-categories defined in COCO. Under these assumptions, we find that there is an intersection of 60 and 64 concepts, respectively, between the datasets for these cases.

Fusing label-based with feature-based contexts (*VGG16+COn*) improves results. Adding a text-based context (*VGG16+COn+W2V-Neg*) further increases performance, although this improvement is sometimes rather small. This confirms our findings about the usefulness of context fusion from above, now also including fusion with text-based contexts.

Concerning pure NLP-based methods we find that *W2V-Neg* shows a very strong performance, which our methods alone do not achieve. But we are not far off and still beat any of the other NLP-based methods.

4.7.4.2 System-to-human correlation

System-to-human-correlation is measured relative to the Scene250, SimLex999 and MEN datasets. The table shows that—when considering COCO (or RCNN)—there is essentially no intersection between the data (9 and 3 pairs, respectively). Scene250 had been purposefully created so that there was a useful overlap (165 and 151 pairs).

Findings on the Scene250 benchmark are closely following the results obtained using clustering consistency. However here the fusion of visual contexts with the *W2V-neg* was twice outperforming pure *W2V-neg*, which happened for the scene datasets with strictly controlled object labels (COCO and RCNN).

The overlap between LabelMe and SimLex999 (166 pairs) as well as MEN (625 pairs) was big enough. This intersection, however, contains a big proportion of very general objects, which do not have straightforward visual representation (examples of those are *art, bedroom, construction,*

game, image, reflection, shoulder, smile, subway, shoulder, sunlight, water). Also, it includes super-categories, e.g., *animal, cloth, container, food, furniture, vehicle* the usage of which introduces ambiguities in image annotations (e.g., is it a bus or a vehicle?). Some pairs in the benchmarks include words with multiple meanings (homonyms) where specifically the non-object meaning makes the words related, like (guitar, rock) or (card, bridge). As a consequence, here our image-based methods are worse than the NLP-based ones.

Another reason for this is based on the statistics. LableMe has only 28,072 scenes were—for robustness reasons—we had considered only objects that were labeled in the scenes three times or more. Thus, some objects we included into our analysis were fairly infrequent (or appeared in contexts infrequently). This likely led to too thin data to gain statistically solid scene-based knowledge about those more abstract objects mentioned above. Furthermore, SimLex999 and MEN were collected to asses general concepts are therefore not the best to judge object concepts.

Scene250, instead, had been created specifically to judge everyday object pairs. Pairs were rated by informed subjects who treated word-pairs very thoughtfully. It seems that under this more rigorous condition differences between image-based and word pair-based assessments vanish.

Summary Summed up, our methods compare very well to text-based semantic vectors and often even outperform multi-modal semantic algorithms as long as the evaluation is run over a set of proper objects. W2V-Neg turns out to exhibit very good scores, but note that the text corpora that W2V-Neg was trained on are magnitudes larger both in terms of vocabulary size (which would be similar to our context) and number of words (object instances) than the datasets we employ.

4.7.5 Qualitative Results

Results can be visualized by reducing their dimensions to two while preserving distances using multi-dimensional scaling. This way, qualitative insights can be obtained. Here we present three different types of visualization:

- **Super classes (Fig. 4.6 Top)**: A plot of all vectors corresponding to COCO classes visualized as dots with super-categories (e.g. animal) determining their color. Ideally, object vectors pertaining to the same super-category should form groups.
- **Local space (Fig. 4.6 Bottom)**: On a more fine-grained scale, object vectors from a preselected set of names are plotted. Instead of a dot, here the respective object name is plotted.

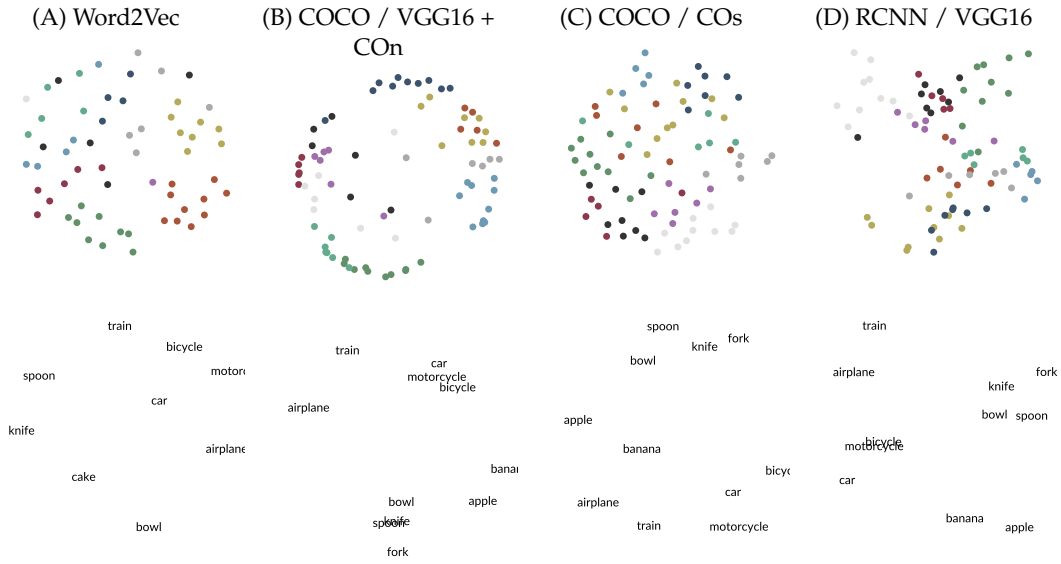

Figure 4.6: Qualitative results, top: Plot of 80 object vectors of labels in COCO projected to 2D using multi-dimensional scaling with color corresponding to super-category. Bottom: Plot of some belonging word labels.

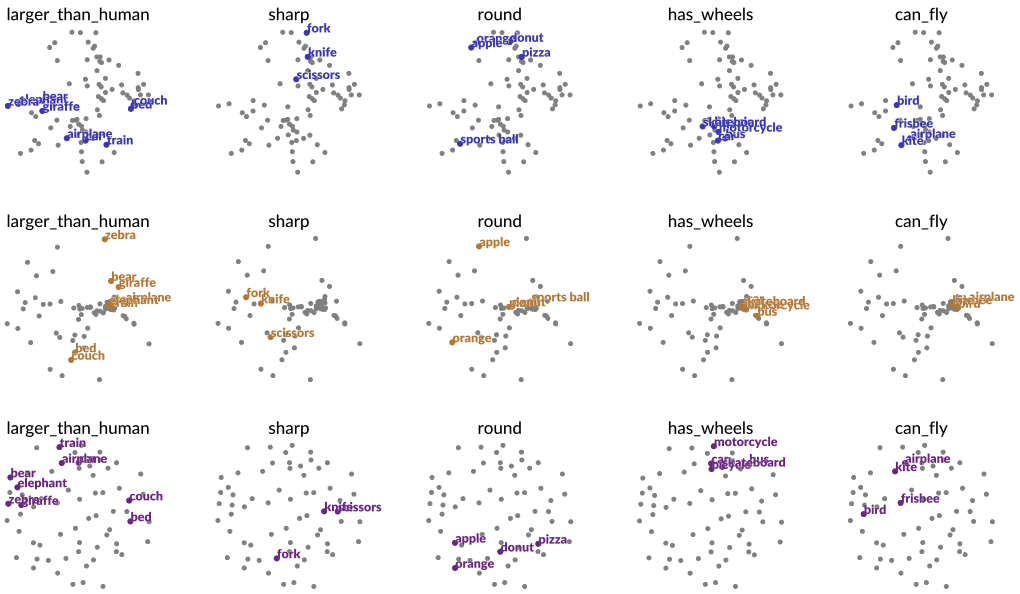

Figure 4.7: Qualitative analysis of how well functional concepts are represented. Top: VGG16, middle: COn, bottom: W2V-Neg. Positive examples for the categories are highlighted in red while other concepts are gray points.

- **Functional (Fig. 4.7)**: Instead of grouping based on super-categories, we split the set of concepts according to a functional trait. Concepts that have this trait are visualized in color together with their name while concepts that do not have a trait are visualized as gray dots. We defined the traits `larger_than_human`, `sharp`, `round`, `has_wheels` and `can_fly`.

We observe that concepts indeed cluster according to their super-category and see reasonable groups of objects in the local space. Furthermore, we find concepts being grouped together within functional groups, even though they do not share the same super-category. Thus, these figures provide a visual confirmation of the above reported clustering results.

4.8 Discussion

4.8.1 Main Contribution

In this study we were asking whether the distributional hypothesis of linguistics [87] would also hold for images. Text is a linear structure, while images are 2D-projections of 3D scenes and this higher dimensionallity might lead to a dilution of information. Thus, we specifically wanted to quantify how powerful a semantic vector representation obtained from scene information still is. Here, we presented a simple but effective method to approach the problem of understanding objects in their scene context by extracting semantic vectors. The main result of this study is that across many experiments, methods and data, text and images appear equally powerful as sources for context analysis. This is to some degree astonishing, because text-based methods, like W2V-Neg and GloVe, use more than 10^{11} words, while we have used on the order of 10^5 images with 10^6 labels only.

Taken together this shows that visual context contributes to our cognitive concepts of objects and should be considered when building semantic vectors. Our evaluations also suggests that simple integration methods like the one presented in this paper suffice to capture these kind of semantics. The amount of data is more critical than the (text- or image-based) context method used.

4.8.2 Pros and Cons

Why should one analyze images instead of (or together with) text? One aspect seems to be the above mentioned density of information. Different from our own expectations, it seems that even a small set of images can already carry substantial amounts of (certain) context informa-

tion. This holds specifically for context information on aspects, which texts don't talk about, like the layout of the utensils in a kitchen drawer or the way a workshop might be arranged. Arguably, not many texts provide this type of information and in these cases context extraction should, thus, be easier in images than in text. Action understanding and execution in robotics has encountered this situation, too. While it is easy to obtain massive higher level action information from text sources, some types of information, which can be vital for a household robot, are missing. For example: Hardly ever one finds a text on how to handle a knife or a spatula, when cooking. Videos and images, on the other hand, are abundant.

In addition, to this another important beneficial aspect when using image-based context extraction is that our results on the performance of RCNN-labeled as well as CNN activation-determined object context indicate that it is also possible to perform automatic semantic assessment of objects in scenes without human labeling, where accuracy is similar to that of the human-supervised methods.

Furthermore, we found in the small pilot experiment shown in Figure 4.7 that even functional groupings can be extracted. Other visual feature-based grouping might be possibly, too, but a more detailed analysis is currently not possible mainly because object categories are too broad (from the category "knife" we cannot tell if it is metallic or not). For a quantitative analysis it would be interesting to train a classifier or regressor to capture specific aspects of the semantic vectors. However, this is currently not possible since the set of objects is still too small and the number of features too large. E.g. training something like "eatable" on maybe tens of positive samples and hundreds of negative samples while having thousands of features will likely overfit even with simple classifiers. At the moment, this prevents rigorous quantitative analyses of this kind, but the results from Figure 4.7 look promising and future work (after addressing the above mentioned problems) should be possible in this direction.

Image-based context analysis, however, has clear disadvantages, too. For example, abstract concepts cannot be extracted. Also, longer text-narratives often contain aspects of reasoning. Images do not. Hence, artificial narrative generation from images, at least so far, remains relatively shallow as compared to the creativity with which – for example – a child describes an image. This is achieved by humans being able to extract in a generative way complex context from an image. Modern image analysis methods, including the ones presented here, stop still short of this.

However, from a practical perspective we are confident that existing methods to extract word vectors, such as Glove and Word2Vec, that serve as a basis for numerous methods in NLP, could benefit from the addition of visual, scenic context. This would require either a large dataset or an object detector capable of handling thousands of classes, but considering the recent progress in computer vision it is only a matter of time until these become available.

4.8.3 OTHER ASPECTS AND FUTURE WORK

It may be interesting to discuss these findings also from a different perspective. "Meaning" cannot be obtained by considering nothing but text. This issue is commonly known as the symbol grounding problem [86] in artificial intelligence: Glenberg and Robertson [72] argue that it is not possible to assign meaning to an abstract symbol like a word as long as it is expressed only with respect to other abstract symbols. This is the case for text but also for labels of image annotations. However, these authors suggest overcoming this problem by *linking* different modalities, where here we had considered text and images.

Evidently humans often perform co-context analyses of text and image information, too; for example, where text remains too abstract and opaque without an image. A good example of this are industrial instruction sheets for small product assembly by human workers. Text-only instructions are here often quite nontransparent. One example we had worked on in the context of the European funded ACAT (Action Categories) project had been: "Place the rotor cap on top of the magnet holder". This instruction becomes only meaningful when seeing the actual objects (before and after this assembly step).

Object class vectors, in conjunction with text, might afford this to some degree as they have been obtained from two complementary modalities. This way, an object's spatial context becomes linked with its textual context. Hence, the process of acquiring meaning would now stand on two legs. While we cannot directly show this, our results support at least the notion that context is strengthened by such combinations.

Several additional aspects could be addressed in future work. Currently, the knowledge captured by the model is expressed by linear translations in a high dimensional feature space. It might be useful to investigate more complex representations, e.g. to reduce the size of the feature space. As a starting point hyperbolic geometry could be considered. Additionally, we have indeed performed some early experiments concerning the combination of vectors spaces obtained from different modalities. The performance gain in these experiments indicates that this direction might be promising as it is a fairly cheap way for improving results relative to single modal semantic vectors.

All this could, for example, lead to an improved identification of task-relevant objects to suggest tasks and to generate planning problem definitions automatically depending on the objects in the scene. This is of great use in robotic executions of human-like tasks [3], where, for instance, the automatic generation of task descriptions can significantly ease the human-robot interaction.

ACKNOWLEDGMENTS

This research has received funding from the European Commission Horizon 2020 Program H2020-ICT-2016-1 under grant agreement 731761 "IMAGINE".

4.9 REFERENCES

We maintain a single list of references at the end of the thesis.

4.10 APPENDIX

4.10.1 DERIVATION OF CO-COUNT CONTEXT

This derivation explains the co-count context. Co-occurrence context can be seen as a special case of this with the frequency of any object class being limited to one. Hence, the argument following now can be modified accordingly and transferred to these other cases, too.

4.10.1.1 Probabilistic Derivation

A common way to describe events without knowing the number of trials is the Poisson distribution [18]. The choice of the Poisson distribution is justified because the occurrence of objects can be considered as events in the spatial domain. Furthermore, a Poisson distributed random variable takes only positive integer values, which is true for co-occurring objects. Hence we will assume that the frequency of an object m occurring in the context of object k follows a Poisson probability distribution with rate λ_k^m. This means, to generate a context vector for an object, we would draw samples from a Poisson distribution with specific parameters for each component. Following the idea of the distributional hypothesis, as these parameters describe the context, they will be used to represent the object classes. Subsequently, we show how to obtain these parameters.

Having observed N_k instances of each object class k, the variable $i \in [1, N_k]$ refers to the context vector $\hat{\mathbf{c}}_{ki}$ of the i-th instance of class k. We assume a probabilistic model $P(\mathbf{c} \mid o)$ in which the object class determines the context. The goal is to find parameters $\mathbf{\Lambda} = \{\lambda_k^m : k, m \in [1, K]\}$ that maximize the likelihood L of these observations.

$$L(\mathbf{\Lambda}) = P(\mathbf{c} \mid o; \mathbf{\Lambda}) = \prod_{k=1}^{K} \prod_{i=1}^{N_k} P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}). \tag{4.2}$$

Due to monotonicity, a logarithm can be applied. This facilitates further steps and retains the same maximum.

$$\log L(\mathbf{\Lambda}) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \log P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}). \tag{4.3}$$

The Poisson distribution's mass function is inserted. Due to the assumption of co-occurring objects being independent, the occurrence probabilities of context objects follow a multiplication:

$$P(\mathbf{c} = \hat{\mathbf{c}}_{ki} \mid o = k; \mathbf{\Lambda}) = \prod_{m=1}^{K} \frac{(\lambda_k^m)^{\hat{c}_{ki}^m}}{\hat{c}_{ki}^m!} e^{-\lambda_k^m}. \tag{4.4}$$

In order to find the maximum likelihood, the derivative with respect to $\boldsymbol{\lambda}_k$ is calculated,

$$\log L(\mathbf{\Lambda}) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \sum_{m=1}^{K} \hat{c}_{ki}^m \log \lambda_k^m - \lambda_k^m - \log \hat{c}_{ki}^m!, \tag{4.5}$$

$$\frac{\partial \log L(\mathbf{\Lambda})}{\partial \lambda_k^m} = \sum_{i=1}^{N_k} \frac{\hat{c}_{ki}^m}{\lambda_k^m} - 1. \tag{4.6}$$

By setting the derivative zero, we obtain the optimal parameterization for $\boldsymbol{\lambda}_k$ which is simply the average of all context vectors.

$$\boldsymbol{\lambda}_k^* = \left(\arg\max_{\mathbf{\Lambda}} L(\mathbf{\Lambda}) \right)_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \hat{\mathbf{c}}_{ki}. \tag{4.7}$$

4.10.1.2 Geometric Perspective

Rather than considering observations of objects as events, the problem also permits a geometric perspective, with observations of context vectors involving co-occurrence frequencies being spread in a high-dimensional (K dimensions) vector space. Every observed context has an associated class and the goal is to find a representative context for each class. We can interpret this problem as optimization of a cost function. If the cost is defined as the sum of the mean quadratic euclidean distances of each class, the minimum again can be analytically derived to be the mean context:

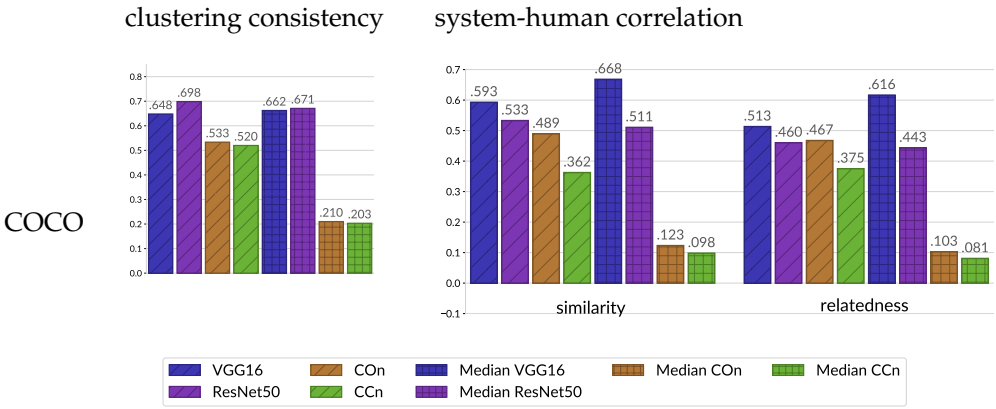

Figure 4.8: Comparison between averaging (diagonal hatching) and median (square hatching) for context integration.

$$\boldsymbol{\lambda}_k^* = \arg\min_{\mathbf{c}_k} \frac{1}{N_k} \sum_{i=1}^{N_k} (\mathbf{c}_k - \hat{\mathbf{c}}_i)^2 = \frac{1}{N_k} \sum_{i=1}^{N_k} \hat{\mathbf{c}}_i \tag{4.8}$$

4.10.2 Alternative Context Integration Methods

In the main text we had compared average context with Skip Gram context integration. Many more integration methods would be possible, too. For example, using the median appears a possible natural choice. Thus, here we take a look at this based on the COCO dataset (Fig. 4.8) and compare its performance with averaging (see Fig. 4.8). In each plot the first four columns are obtained by averaging, while the columns 5 to 8 are obtained with the median. The results show that the median sometimes leads to an improvement and sometimes it decreases performance. In particular in case of VGG16 the median yielded a fairly strong improvement. However, as the median does not show stable performance (decrease in performance may be substantial too, e.g. for COn and CCn context methods in the figure), averaging remains a better method.

4.10.3 Comparison of Fusion methods

There are multiple ways for combining different types of context vectors for a concept. In the main text we analyzed one type of early fusion (concatenating context instances) and one type of late fusion (concatenating semantic vectors). We applied one specific type of normalization for early fusion and one type for late fusion. Here we will analyze more such methods.

	Clustering	Scene250	
		sim	rel
CCn	0.520	0.375	0.362
COn	0.533	0.467	0.489
EarlyFusion(concat): CCn+COn	0.529	0.385	0.377
EarlyFusion(sum): CCn+COn	0.530	0.395	0.391
EarlyFusion(average): CCn+COn	0.530	0.395	0.391
EarlyFusion(concat): InceptionV1+COn	0.670	0.466	0.550
EarlyFusion(equal): InceptionV1+COn	0.627	0.573	0.611
EarlyFusion(norm): InceptionV1+COn	0.682	0.573	0.637
EarlyFusion(concat): VGG16+COn	0.649	0.513	0.593
EarlyFusion(equal): VGG16+COn	0.639	0.563	0.615
EarlyFusion(norm): VGG16+COn	0.656	0.579	0.645
LateFusionNorm: InceptionV1+COn	0.693	0.643	0.674
LateFusionEqual: InceptionV1+COn	0.627	0.573	0.611
LateFusionNormToOne: InceptionV1+COn	0.535	0.467	0.490
LateFusionNormStdOnly: InceptionV1+COn	0.677	0.419	0.499
LateFusionNorm: VGG16+COn	0.680	0.643	0.685
LateFusionEqual: VGG16+COn	0.639	0.563	0.615
LateFusionNormToOne: VGG16+COn	0.535	0.468	0.490
LateFusionNormStdOnly: VGG16+COn	0.645	0.473	0.556

Table 4.5: Comparison of various fusion methods on COCO.

The investigated methods are concatenation-based (except for sum) and vary in the way how normalization is carried out prior to concatenation. Let us say $\mathbf{a}$ and $\mathbf{b}$ stand for two vectors and $||$ the concatenation operation. We define the following fusion methods of those vectors:

1) Simple concatenation:

$$cat(a, b) = \mathbf{a} \,||\, \mathbf{b}, \tag{4.9}$$

2) Simple averaging:

$$\text{average}(a, b) = \frac{\mathbf{a} + \mathbf{b}}{2}.$$

Note, however, that this method is fairly limited as it requires the context vectors to have the same length.

3) Concatenation with length-based normalization:

$$\text{eq}(a, b) = \frac{\mathbf{a}}{dim(\mathbf{a})} \,||\, \frac{\mathbf{b}}{dim(\mathbf{b})},$$

where dim denotes the length of the vectors.

4) Early fusion with weight normalization (same as in the main text):

$$\text{norm}_{early}(a,b) = \frac{\mathbf{a}-\mu_{\mathbf{a}}}{\sigma_{\mathbf{a}}} \,||\, \frac{\mathbf{b}-\mu_{\mathbf{b}}}{\sigma_{\mathbf{b}}},$$

where μ_a, μ_b and σ_a, σ_b are scalar mean and standard deviation of vectors a and b respectively.

5) Late fusion with weight normalization (same as in the main text):

$$\text{norm}_{late}(a,b) = \frac{\mathbf{a}-\mu(A)}{\sigma(A)}||\frac{\mathbf{b}-\mu(B)}{\sigma(B)},$$

considers vectors in their respective embedding (A and B) and computes mean and standard deviation independently for every dimension, i.e. here μ and σ are vectors

6) A simplified version of weight normalization only considers the standard deviation:

$$\text{norm}_{std}(a,b) = \frac{\mathbf{a}}{\sigma(A)}||\frac{\mathbf{b}}{\sigma(B)},$$

7) Late fusion norm to one concatenates vectors that were normalized to length one.

$$\text{norm}_{one}(a,b) = \frac{\mathbf{a}}{|\mathbf{a}|}||\frac{\mathbf{b}}{|\mathbf{b}|},$$

Note, when using context averaging as integration method, early fusion and late fusion are equivalent for concatenation, average and length normalization, because the order of the operations are commutative.

In Table 4.5 results obtained using different fusion methods are shown. First we compare results for simple concatenation and averaging between contexts CCn and COn. Note, we cannot use averaging for other, potentially more interesting context combinations, as for that vectors shall be of equal length. Here we see that none of the two indicated context fusion methods is of substantial advantage.

We experimented also with quite a few other fusion methods for different types of context vectors: label-based and visual-feature-based. In summary this shows that fusion will usually improve the result, where here the "Late Norm" method is the best. Note however that also for fusion other methods would be possible and an exhaustive analysis cannot be performed.

4.10.4 Intersections of Different Datasets

When evaluating our method against the state-of-the-art methods on given ground-truth benchmarks, we needed to consider a tripartite intersection of different datasets (1,2,3, numbering as in Fig. 4.9, left side). Here we will explain the reasoning behind this and specify coverage for these intersections.

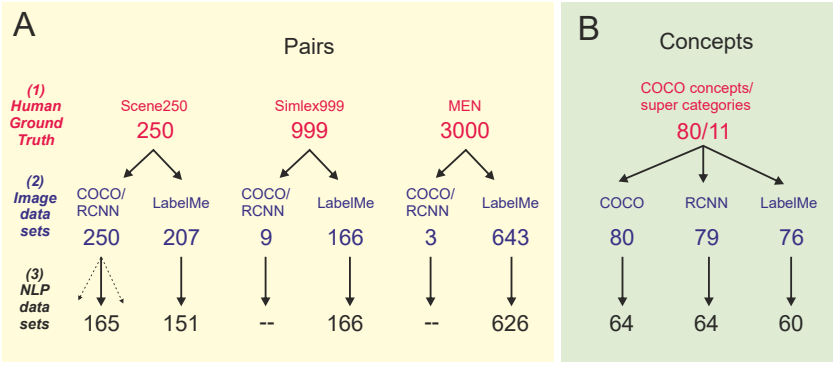

Figure 4.9: Intersections between the different data sets.

Concerning similarity ratings (Fig. 4.9 A), we compared (3) conventional NLP text-based methods with (2) our image-based methods, grounding these comparisons on (1) three human-rated datasets: Scene250, Simplex999 and MEN. These datasets are named in the top row in Fig. 4.9 (red). The human rated datasets are a collections of pairs of concepts with similarity (and relatedness) for each pair indicated. The number of pairs included in each dataset is given in the figure beneath the datasets' names. In order to evaluate performance of our methods against NLP text-based methods, we can only consider pairs that are conjointly contained in the datasets across all levels 1-3. The next row in the figure (blue) shows the number of pairs conjointly contained in the human-rater as well the image datasets and the bottom row (black) shows the final minimal resulting intersection with the NLP datasets. These are the ones used by W2V-Neg and Glove. Clearly, only large enough intersections, all of which are given in the bottom row, could be used for our study. Intersection with only 3 or 9 pairs had to be ruled out.

The same procedure is performed on the right side of Fig. 4.9 B, but this time for concepts and super-categories on which we evaluate clustering consistency. As ground truth in this evaluation we take the division of 80 COCO concepts into 11 super-categories as done by humans. RCNN (2nd row) contains 79 of those, because the Mask R-CNN was not able to find the hair dryer. LabelMe overlaps with 76 concepts. The final overlap with the NLP datasets is given in the bottom row of panel B, which are those concepts with which we could perform the comparison.

4.10.5 Precision and Recall

As an alternative to clustering consistency, the quality of the clustering can be assessed through a k nearest neighbor classifier. First, we explain how to create such a classifier. Consider the semantic vectors in Fig. 4.10, involving 12 concepts and three super-categories: Square, circle

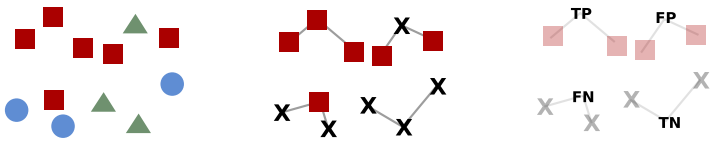

Figure 4.10: Sketch of the precision/recall evaluation: Left concepts with super-category, middle: square centered view with two-nearest-neighbors of four concepts being indicated, right: error categories based on the true labels and the classification based on two nearest neighbors.

and triangle. We iterate over all super-categories (in Fig. 4.10 mid, square) and differentiate between element (square) and non-element (X). Using this binarization, we can decide between true-positive, false-positive, false-negative and true-negative from which precision and recall are computed:

$$\text{precision} = \frac{TP}{TP + FP}$$

$$\text{recall} = \frac{TP}{TP + FN}$$

Since this only involved one super-category, the procedure is repeated for every super-category and precision and recall scores are averaged, yielding the numbers reported in Figure 4.11.

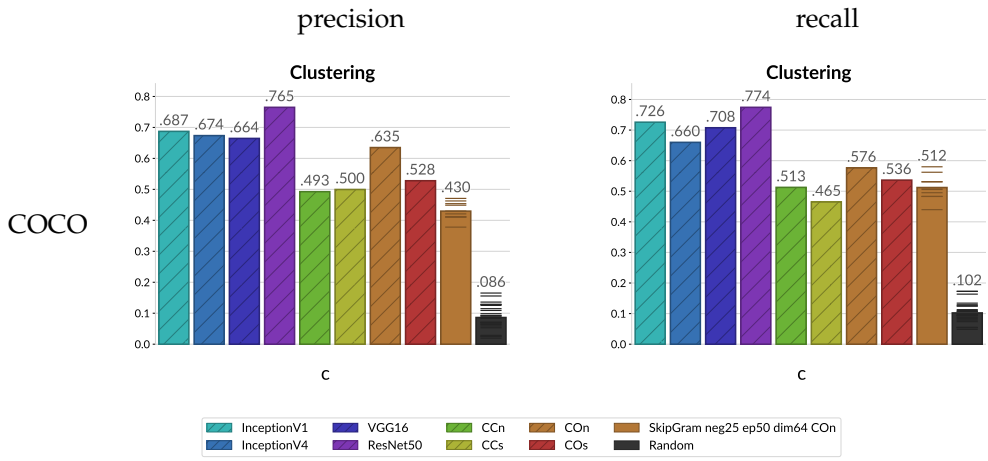

Figure 4.11: kNN scores for various context extraction methods.

Results remain consistent in comparison to the ones obtained when using clustering consistency (compare to Figure 4.4 from the main text).

4.10.6 SKIP-GRAM HYPER-PARAMETER SEARCH

The Skip-Gram negative sampling training has multiple hyper-parameters. To enable a fair comparison with other methods, we investigate different choices for the number of negative samples being presented for every positive sample, the number of training epochs and the dimension of the concept vectors. Regarding the number of negative samples, we find the algorithm to be fairly stable and choose 25. The scores further suggest that ten epochs are too few while after 50 epochs the training seems to have converged in all cases and that even 64 dimensions already exhibit a good performance. Hence, we decided to run the training for 50 epochs with 64-dimensional concept vectors.

negative samples	epochs	dimension	clustering consistency	similarity	relatedness
10	50	128	0.423 (0.015)	0.438 (0.031)	0.397 (0.030)
25	50	128	0.413 (0.016)	0.455 (0.026)	0.433 (0.019)
50	50	128	0.404 (0.013)	0.452 (0.017)	0.441 (0.014)
25	10	128	0.228 (0.044)	0.265 (0.029)	0.258 (0.030)
25	100	128	0.468 (0.015)	0.474 (0.027)	0.441 (0.016)
25	50	64	0.478 (0.020)	0.472 (0.021)	0.451 (0.019)
25	50	256	0.372 (0.018)	0.403 (0.021)	0.386 (0.016)

Table 4.6: Scores for the hyper-parameter search of skip-gram. The number in bracket indicates the standard deviation over ten runs.

CHAPTER 5

Affordance Segmentation

In Chapter 4 we have presented an approach that, among other properties, can assign functions to object classes. However, this method still relies on object names, which are not always precise concerning possible interactions. Not every part of a chair is sit-able but only its base, the handle is the only part of a door that is pull-able and not every ground is walk-able. Additionally, the layout of the scene might alter the presence of some interactions. For instance, it is not possible sit on chairs that were placed upside down on a table to allow for cleaning the floor. In general, an important trait of autonomy is to identify possible means of interaction with the world. This involves questions like these: Which doors can be opened? Where can I pull to open a container? Where can I walk? When these questions can be reasonably answered, a robotic system can explore its surroundings without human intervention. In the following article, we propose a system that is able to answer these questions by identifying potential interactions with the environment. This means it directly transforms perceptive cues into possible actions.

Summary In the following article, we develop an algorithm that segments a set of 12 affordances from RGB images. For this, we employ a CNN that is trained to directly predict these affordances. For the training we transform pixel-wise labels from the ADE20K dataset [248] to affordances. This is carried out using a manually defined transfer table that describes a mapping from object names or object part names to affordances. When the object/part name does clearly indicate the presence or absence of an affordance we ignore the respective pixels in the affordance map by using a masked loss function. Our experiments show state-of-the-art results and good generalization capabilities of the algorithm. Additional to the quantitative experiments, we demonstrate that the method can be applied in a robotic context. This paper is published as:

Timo Lüdecke, Tomas Kulvicius and Florentin Wörgötter
Context-based Affordance Segmentation from 2D Images for Robot Action
Robotics and Autonomous Systems (RAS)
https://doi.org/10.1016/j.robot.2019.05.005

The following article's presentation was adapted to match the format of this thesis, references are shown at the end of the thesis. The content is identical to the published version.

Timo Lüdecke, Tomas Kulvicius and Florentin Wörgötter

Context-based Affordance Segmentation from 2D Images for Robot Action

(published in Robotics and Autonomous Systems Volume 119)

Abstract

Affordances play a crucial role in robotics since they allow developing truly autonomous robots, which can freely explore and interact with the environment. Most of the existing approaches for analyzing affordances in a scene consider only one or few types of affordance, e.g., grasping points, object manipulation or locomotion. In many cases only whole objects are considered. In our study we include in total 12 affordances of object-related, manipulation and locomotion affordances, considering affordances of both objects and/or their parts. We design a system that can densely predict affordances given only a single 2D RGB image. For this, we propose a method that transfers object class labels to affordances. This enables us to train convolutional neural networks, a PSPNet-based network and a U-Net-style network, to directly predict affordances from an image using a selective binary cross entropy loss function. The method is able to handle (potentially multiple) affordances of objects and their parts in a pixel-wise manner even in the case of incomplete data. We perform qualitative as well as quantitative evaluations with simulated and real data including robot experiments. In general, we find that frequent affordances are recognized with a substantial fraction of correctly assigned pixels, while this is harder for infrequent affordances and small objects. In addition, we demonstrate that our method performs better than a recent competitive approach. As the proposed method operates on 2D images, it is easier to implement than competing 3D methods and it could therefore more easily provide useful affordance estimates for robotic actions as demonstrated experimentally.

5.1 Introduction

To express opportunities for action of an animal in its environment, the perceptual psychologist J.J.Gibson [69] coined the term *affordances*. He defines affordances as opportunities for action between an animal and the environment. Examples for affordances from the perspective of a cat are: Shrubbery affords shelter and a mouse affords nutrition. Later, the term was adopted by the robotics community and extended from animals to robots. Essentially in robotics this term very often takes the meaning of: "Which actions could a robot perform in a given situation (with some given objects)?". This perspective on affordances is adopted in our work, too. We assume a human-like embodiement of the robot, leading to a set of affordances similar to those

that humans would encounter. Task-specific refinements of the affordances can be carried out depending on actual embodiement and application. But even for other systems the capability to understand affordances can be useful, in particular if interaction with humans is required such as in a smart home. In this work, we address the problem of segmenting affordances for (but not limited to) robotic applications.

Segmentation means that the output of the system we propose in this paper consists of areas (segments) that represent different (sometimes overlapping) affordances. This is done in a pixel-wise manner by assigning presence-probabilities for all different affordances to every pixel in the image. Affordance segmentation is more challenging compared to object class segmentation due to three aspects. 1) Affordances are not disjoint, i.e., the presence of an affordance does not exclude the presence of other affordances. This requires us to make use of multi-class segmentation. 2) Affordances can refer to very small structures, which are only parts of objects. 3) There are no large-scale datasets available yet. This means that the problem cannot simply be addressed by training a semantic segmentation model on new data. These challenges in conjunction with the practical applicability makes affordance segmentation an interesting and challenging research topic.

There are multiple approaches to identify affordances, each having their own pros and cons. For example, affordance assignment can be done by considering object geometry [80]. If performed purely in this way, this leaves out all semantic object knowledge which we–humans–have access to when assigning affordances. Therefore, in this work we pursue a semantic, context dependent approach and show that this is very powerful even when only considering single 2D-RGB images. This method leverages knowledge about the relation between affordances and object parts and employs a convolutional neural network (CNN) to assign action affordances probabilistically to the pixels of a new image. The advantage of this is that it combines a knowledge-based approach with 2D images making it useful for a wide range of robotic applications. Naturally, limitations of all single image-based approaches apply to this work as well, e.g., in the case of 3D to 2D projection inconsistencies. However, such problems can often be solved by incorporating other methods, e.g., fusing the segmentation with a structured light sensor output and doing some post-processing.

In summary, the contributions of this work are as follows:

- A method to generate a large set of action affordances from semantic segmentations using a selective binary cross entropy loss function.
- A method to fuse simulated data with real data to improve generalization and their empiric comparison.
- An extensive evaluation of UNet- [172] and PSPNet-like [246] convolutional neural net-

works for affordance segmentation taking both runtime speed and prediction quality into consideration.

- Validation of the method in a robotic scenario that consists of images that had *not* been part of the training set.

For reproducibility of our proposed approach and as a starting point for using our method in practice we published the source code and pre-trained models online. [1]

5.2 Related Work

In the past affordances have been addressed in multiple studies, both in computer vision and in robotics. In the following, we will review existing work from psychology before discussing different approaches for affordance segmentation and relate those to our approach.

5.2.1 Affordances in Psychology

The term affordance originates from the field of perceptual psychology. It was originally coined by J.J. Gibson as part of his direct perception theory. He defined affordances as potential actions between an animal and its environment, given the capabilities and state of the animal and the structure of the environment [69]. The set of all affordances of an animal is called its ecological niche. Based on the seminal work of Gibson [69], later research sought to refine the question of what constitutes an affordance also addressing philosophical implications. Early theories considered affordances to be properties of the environment which are used by an animal [209]. Chemero [36] argues against this and proposes to consider affordances not as properties at all but as relations between particular aspects of animals and situations. Recently, Rietveld and Kiverstein [171] suggested applying the notion of affordances in a much broader context and on a higher level: They argue that any skill, not only motoric abilities, that a form of life possesses establishes a set of affordances. They use form of life instead of animal to emphasize cultural differences within a species of animals. A review on the historical development of the term affordance and a discussion of competing models was carried out by [60] and more recently by [133].

In addition to these theoretical and philosophical aspects, several studies have assessed the perception of affordances in humans and animals experimentally. Warren [231] suggests differentiating the transition between affordances on the basis of critical and optimal points and finds experimental evidence that the perception is affected by the capabilities of the agents. A

[1] https://gitlab.gwdg.de/cns-group-public/aff-seg

mathematical formulation of affordances is obtained by expressing critical points as functions of ratios of body parts (leg length) and environmental quantities (riser height). Cole et al. [43] studied differences in perception of the three types of affordances leap, arm-swing and crawl in humans. They found participants to systematically underestimate their abilities in launching actions (leap and arm-swing) but improved their judgement after actually performing the actions.

5.2.1.1 Relation of our work to affordances in psychology

Our approach can be seen as a partial implementation of the direct perception paradigm by Gibson. Light represented by an RGB image is directly transformed into actions without explicit intermediate representations (the activations in the layers form implicit representations, though) in a pure feed-forward fashion. Instead of binary categories, affordances are represented probabilistically as suggested in [66]. One of the affordances analyzed by our system is *illumination*, which does not imply an immediate motor action. This can be seen as an example of the richer set of affordances enabled by animal skills as suggested by [171]. Wagman, Caputo, and Stoffregen [224] studied a multi-affordance environment and found evidence that humans sense means-end relations between affordances. Our work shares the trait of predicting multiple affordances simultaneously but does not encompass hierarchies.

5.2.2 Affordances in Computer Vision

A common approach for affordance segmentation is to predict affordances of whole objects [197, 250]. Akin to these is the work of Ye et al. [239] who detect bounding boxes of affordances using a two-stage approach consisting of region proposal and CNN-feature-based affordance recognition. Sawatzky, Srikantha, and Gall [182] learn to segment affordances on weakly supervised data. More recently, Sawatzky, Garbade, and Gall [181] address a few-example-setting using label transfer on images of objects. Given a query image, a similar example is retrieved from a database. Then the segmentation is transferred to the query and refined by a CNN. It seems unlikely that this approach generalizes beyond object level as whole scenes vary much stronger and finding a similar example scene would require an extremely large database.

Affordance datasets have been proposed before. The UMD RGB-D part affordance dataset [151] focuses on objects only, was captured in a controlled lab environment and is intended for approaches that rely less on context but on depth information. The IIT-AFF dataset [153] might seem more applicable for us. However, it involves only 10 object classes which is too limited for studying whole scene affordances. Furthermore, it is sparse as affordances are only assigned to

objects but not other parts of the environment (e.g. the floor). Due to these shortcomings we decided to construct our own dataset that is more suitable for our analysis.

Affordances can also be predicted in form of (human) poses. This scheme is adopted by Gupta et al. [84] making use of scene geometry and in Grabner, Gall, and Van Gool [80] specifically for chairs and by Fouhey et al. [65] using video. Similarly, Kjellström, Romero, and Kragic [111] learn affordances by observing interactions with objects in videos. More recently, Wang, Girdhar, and Gupta [229] analyze popular sitcoms to learn pose-based affordance prediction.

The idea of "action maps" is closely related to affordance segmentation. However, the former tends to be more specific, e.g., by referring to very concrete objects and the set of considered actions is fairly small. Examples of these approaches are Savva et al. [180] who generate seven different "action maps" by tracking people in RGB-D video footage and Rhinehart and Kitani [169] who learn 6 action maps through analyzing egocentric video recordings.

The method proposed by Roy and Todorovic [173] is similar to ours as it also generates pixel-wise maps given an RGB image. Their model learns intermediate representations for depth, surface normals and object classes, which are then employed to carry out the affordance map prediction. The learning of these representations is actively enforced during training, i.e., the method requires additional data during training, while our method only needs RGB images and affordance map ground truth. Another difference to our work is the set of considered affordances.

Concurrently to this work, a similar method is proposed by Do, Nguyen, and Reid [57]. It deviates from our method as it conducts affordance and object detection jointly and is trained on the sparse IIT-Aff dataset, while we are interested in entire scenes "in the wild".

A common trait of many approaches is that they are constrained to a specific domain or the set of considered affordances is small (see Table 5.1). Our dataset consists of 22,000 images, which is significantly more than IIT-AFF (8,835 images) and UMD RGB-D (more than 10,000 annotated images but only 105 object instances). As stated in the introduction, our approach is context-based and we do not explicitly consider object geometry (for such approaches see [80] or [173]).

5.2.3 Affordances in Robotics

Affordances play a major role in cognitive and developmental robotics since they are crucial for a robot allowing it to explore and interact with the environment fully autonomously. Affordance research in robotics has received a lot of attention during the last decade and led to many contributions. For detailed recent surveys see Min et al. [146] and Zech et al. [245]. In

Table 5.1: Comparison of related algorithms with # denoting the number of used affordances.

Approach	#	input	output
Grabner, Gall, and Van Gool [80]	1	RGB-D	per voxel
Gupta et al. [84]	4	RGB	per pixel
Savva et al. [180]	7	Video	per voxel
Rhinehart and Kitani [169]	6	Video	per grid-cell
Roy and Todorovic [173]	5	RGB	per pixel
Our approach	**12**	**RGB**	**per pixel**

the following, we will only briefly review main concepts and approaches in affordance-related research in robotics and relate them to our approach. Also, we will be only concerned with object/tool related affordances and will not talk about social affordances (e.g., [217]).

There are mainly three *categories* of object-related affordances [245, 146]: 1) grasping affordances, 2) manipulation affordances, and 3) traversability and locomotion affordances. Grasping affordances are mainly concerned with how to find a grasp-able point of the object for a particular object manipulation [183, 55, 54, 210]. Manipulation affordances relate to manipulations of single objects, e.g., push, pull, turn, lift, etc. [82, 81, 204, 119, 242] or to the interaction with multiple objects and/or tools, e.g., stack, sort, tool use, etc. [198, 148, 212, 62]. Traversability and locomotion affordances relate to motion affordances for mobile robots, e.g., cross, climb, select foot placement, etc. [214, 215, 200, 129]. While most of the studies focus on one type of affordance or consider only few affordances (mostly in the range of two to four [245]), in our study we deal with a total of 12 affordances within three *categories* of affordances: manipulation-related affordances (break, grasp, pull, tip-push, place-on), traversability- and locomotion-related affordances (sit, roll, walk, obstruct, support) and two object-related affordances, which do not belong to the three main types of affordances stated above (illuminate and observe). Also, most of the studies assign affordances to a whole object [146] and only few studies consider parts of objects [198, 148, 212], whereas in our study we investigate affordances of both objects and their parts.

Affordances can be acquired or learned in several ways [245]. The most common strategy to learn affordances is exploration, which is inspired by the cognitive development of children [17, 21, 15, 31]. Other strategies include supervised learning approaches such as programming by demonstration [197, 193, 194] or by providing ground truth data to an agent [6, 34, 37]. Some other approaches do not use learning at all and use hard-coded affordances [118, 103, 104]. In our study we use a supervised learning approach, which is much easier to implement and is less time-consuming as compared to exploration.

Different learning strategies have been used to implement affordance learning ranging from

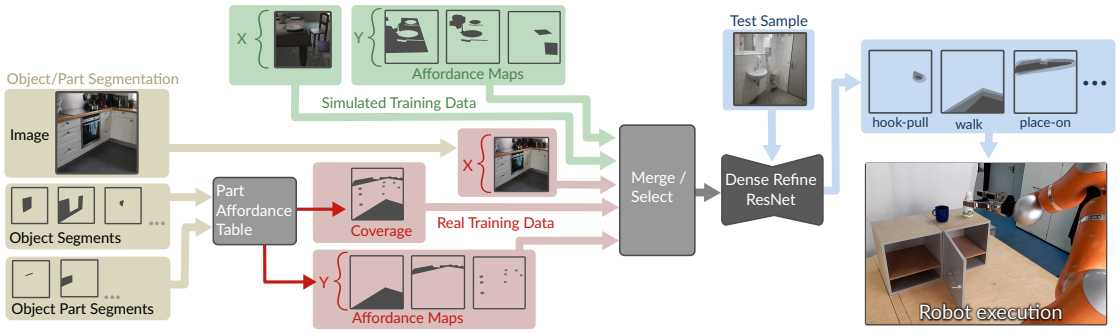

Figure 5.1: Our approach: We train a neural network to predict a set of affordance maps (Y) from a single RGB image (X) using a loss function that allows for incomplete data by incorporating a coverage map. Real (red) and simulated training data is mixed (green), with the real data being generated from object part segmentations using a manually specified part-affordance table. The output of the algorithm can serve as an input to many robotic tasks.

unsupervised learning methods such as self organizing maps [4] and K-Means clustering [33, 109], and reinforcement learning techniques [198, 199, 227, 47] to supervised learning techniques such as support vector machines and multi layer perceptrons [32, 6, 213, 211] and recent approaches using CNNs [173, 37, 152, 153]. In our study we also employ CNNs, however, as already discussed above, we only rely on single 2D images. In [173, 152, 153] RGB-D images are used while in [37], in addition to RGB images, motion data were employed being specific to autonomous-driving applications.

5.2.4 Simulated Training Data

Previous works have studied the effect of training on simulated data. For the related tasks of stereo matching and optical flow, Mayer et al. [141] discuss several data generation schemes. However, these tasks are quite different from affordance segmentation as they require alignment of image regions instead of semantic understanding. Saleh et al. [178] address semantic segmentation, but they only consider a scenario where all training data is synthesized while we explicitly focus on how to merge simulated and real data.

5.3 Methods

In this section we describe the sub-modules of our method. The general pipeline of our approach is outlined in Figure 5.1. First, we explain how we transfer object (part) labels to affordance labels. Then the model for generating simulated training data is defined. In section

5.3.3 we provide an overview of the CNN architectures and the loss function we employ in this work. Finally it is explained how the system can be used in a robotic framework.

5.3.1 Part Labels for Affordance Definition

We use real as well as simulated data for training (and testing) our system. In the following, we will describe how to assign affordances to real scenes, which is the more complicated case. Generation of simulated scenes is described afterwards, where the same principles for affordance assignment are employed.

Our method requires access to object and part segmentations, with as fine-grained labels as possible. From these annotations we derive affordances using a transfer table while obeying these principles:

1. Affordances should be *meaningful* (in some sense) for robots or humans.

2. We require that affordance names are specific. For example, *open* is a very unspecific multi-action. *Tip-push* implies a very well defined motion, of approaching a surface (e.g. a button) with a finger (mostly the index finger). Therefore we consider *tip-push* to be specific enough.

3. Actions can have a hierarchy, but lead to the same final outcome: E.g. a house can be entered, a door, which is a part of the house, can be opened and the door's handle, as a part of the door can be pulled. All of this will be done to enter the house, where the pulling of the door handle is here the action at the lowest semantic hierarchical level. Only this level will be considered to label affordances in this study.

Considering these guiding principles as well as the underlying dataset (ADE20K), we define a set of 12 affordances: obstruct, break, sit, grasp, pull, tip-push, illumination, observe, support, place-on, roll and walk. They are presented along with short descriptions in Table 5.2.

5.3.1.1 Object Parts

An affordance most often refers to only a part of an object. For example, it is the surface of the table that affords placing an object there, but not the table legs. Thus, we define affordances part-wise. ADE20K [248] is currently the only sufficiently large dataset that resolves objects into their parts, hence it is used in this work. Although MsCOCO [130] has many more images, this dataset is not suitable for our approach as it only provides 80 object classes.

Table 5.2: Description of the set of affordances used.

Affordance	Description
obstruct	vertical surface that prevents locomotion. *e.g. wall*
break	detachable objects that can easily be damaged or destroyed *e.g. vase*
sit	surface a human can sit on while having the feet on the ground *e.g. seat cushion*
grasp	detachable objects that can be encompassed with one hand or only few fingers and be moved with one arm.*e.g. vase)*
pull	surfaces that can be pulled through a hook or pinch movement of the fingers (all directions). *e.g. knob, handle*
tip-push	surfaces that trigger some action when being pushed. *e.g. button-panel*
illumination	surfaces that emit visible light.*e.g. bulb*
observe	surfaces that present information or art, i.e. that can be read or watched. *e.g. display*
support	stable surfaces that provide support for standing (for the agent) except ground. *e.g. wall*
place-on	raised surfaces where objects can be placed on (this excludes the ground). *e.g. tabletop*
roll	surfaces that can be used with wheels. *e.g. road*
walk	surfaces a human can walk on. *e.g. grass*

5.3.1.2 Transfer Table

We manually define a mapping from object and object-part labels to 12-dimensional affordance vectors. Each dimension in this vector corresponds to one affordance. Each vector element can have a value between 0 and 1 or can be undefined. The latter is useful if the presence of an affordance cannot be reliably inferred from the object or part name. Clearly, multiple affordances can be present simultaneously, so, in contrast to semantic segmentation, the vector does not have to sum to one. This mapping from objects and parts to affordance we call the transfer table. It consists of round 250 rows that have been manually defined by us. This table serves as the basis for turning segmentation ground truth data from ADE20K into affordance ground truth data (which will be later fed to the CNN). Adding a new affordance would require updating the transfer table with an additional column. For this, compatibility (yes or no) between the new affordance and all approximately 250 object needs to be defined.

The transformation from an object-part segmentation $\mathbf{O}$ into an affordance segmentation $\mathbf{A}$ is carried out pixel-wise. The original label of a pixel is searched in the transfer table T and replaced with the associated affordance vector from the table, if there is a matching entry in the table, i.e. $\mathbf{A_{ij}} = T(\mathbf{O_{ij}})$. Otherwise we acknowledge that no affordance can be assigned. To make the latter accessible later on we store a binary mask tensor $\mathbf{M}$, that encodes the validity of

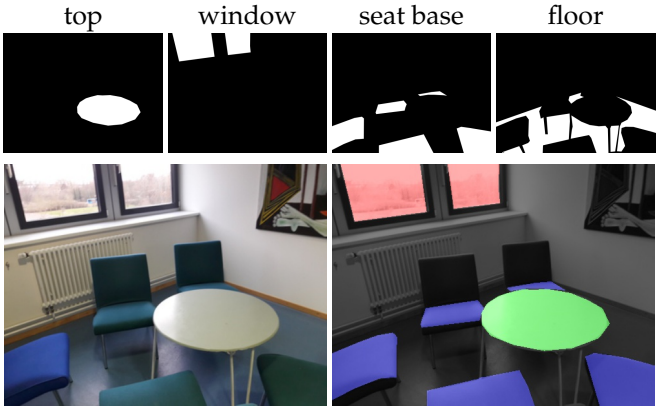

Figure 5.2: Example usage of the transfer table: Given a set of object-part segmentations (e.g. from ADE20K), the transfer table is queried to generate 12 affordance maps of which three are shown (sit (blue), place (green) and illumination (red)). In general, affordances can overlap, although this is not the case here.

object	obstruct	pinch-pull	break	sit	grasp	illumination	support	place-on	...
swivel chair	1	0	0			0	0	0	...
door	1	0		0	0			0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱

Table 5.3: Excerpt from the transfer table

the assigned affordances:

$$\mathbf{M}_{ij} = \begin{cases} 1 \text{ if } O_{ij} \in T \wedge T(O_{ij}) \neq \text{undef.} \\ 0 \text{ otherwise} \end{cases}$$

This mask is subsequently leveraged in the cost function to select valid pixels (see section 5.3.4). A sketch of how the transfer table works is shown in Figure 5.2.

Table 5.3 below shows an excerpt from the transfer table. Each cell can have three values: Affordance present (1), absent (0) or "unknown" ( ). A door and a swivel chair are always obstruct-able. However, only some parts of the swivel chair are sit-able and grasp-able so the corresponding fields are left blank, indicating uncertainty. The same holds for door with break, illumination and support, because it might be a glass door.

By applying the transfer table to the original segmentation labels of ADE20K we obtain our affordance maps. The distribution of the different classes is depicted in Figure 5.3.

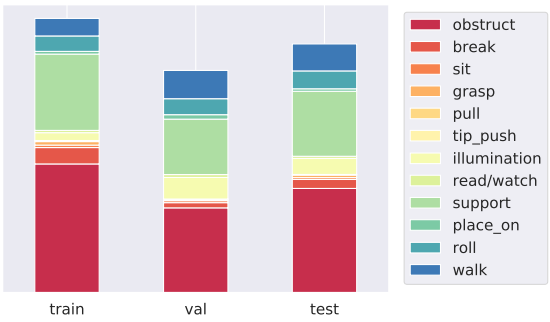

Figure 5.3: Occurrence distribution of the 12 affordances on the different splits. The classes are highly imbalanced, which poses a challenge for training the network. Some classes are so rare they cannot be seen in the diagram.

5.3.1.3 Data Augmentation

Scene quality in ADE20K substantially varies. This leads to the situation that only a rather small number of good-quality training samples can directly be generated from ADE20K. Therefore, we augment the dataset by cropping out image patches from an original image where we then vary color and contrast within such a patch. For large images, this can lead to multiple non-overlapping crops, which can be considered individual samples. The augmentation is carried out online, i.e. therefore in each training epoch completely new samples (new crop, new color, new contrast) are fed into the network.

5.3.2 Simulation Model

Transferring labels from real-image object parts has some disadvantages: Maps are incomplete and some affordances occur rarely. We overcome this problem by generating a new dataset of simulated images. It relies on a probabilistic scene model of a living room and a kitchen with several constituents of the scene being randomized. Hence, we can generate strongly varying images of the scene. More precisely, the randomized variables in our model are object material, -position, -shape, scene illumination, and perspective.

Object material Objects can have different materials. A table surface, for instance, can be composed of plastic, wood or glass. Glass can be transparent or opaque. During scene generation, every object in the scene gets a randomly assigned material, with possibilities being constrained based on the object name.

Object positions Several objects are randomly positioned in the scene and relative to other objects. Examples are a plate on a table, which is dependent on the table's height and a fork and knife, which are positioned relative to the plate.

Object shape For some objects we define key model shapes and interpolate between these key shapes when a scene is generated. E.g. we interpolate between a chair with rounded edges and a chair with sharp edges.

Scene illumination The world during day-time looks entirely different than at night. We account for this by varying light from the outside as well the intensity of indoor and outdoor illumination.

Perspective Having obtained a variable scene model, we still need to simulate the process of photography by projecting the 3D scene onto a 2D plane from many possible viewpoints. For this, it is desirable to use viewpoints that sample mostly interesting aspects of the scene (e.g. multiple objects and sufficient distance), while avoiding irrelevant projections (e.g. view of the ground only) or invalid perspectives (e.g. taking an image from behind a wall). We address this challenge by sampling the camera's position randomly along a fixed heuristically assumed trajectory and introducing slight variances with respect to the position.

For each object or object part we manually define corresponding affordances and render the corresponding affordance maps in a second pass by changing the objects' materials.

This procedure allows us to generate an arbitrary number of training samples each providing consistent, fully covered affordance maps. This way, we can extend the training set by many additional images.

The simulation model is implemented in the open source 3D modeling and simulation software blender[2] using its scripting API and the unbiased, physics-based renderer cycles. Figure 5.4 gives an impression of the variability of the simulated samples. The dataset obtained using this method involving 2280 scenes is subsequently denoted by Sim^T. While we can draw an infinite number of samples from the simulation, the critical task is to introduce variability. Hence, the number of artificially generated scenes represents a trade-off between performance gain and effort we put into designing the simulation environment.

[2]https://blender.org

Figure 5.4: Three simulated samples where the perspective is fixed while other variables (e.g. floor) were randomly sampled.

5.3.3 CNN

Affordances are context dependent. An example makes this clear. We could ask whether a surface is walk-able or suitable to place things? If we now compare the ground with a table surface, we find that, locally, both are flat and uniform. Only context may resolve the difference between them. Walk-able surfaces, for example, may be accompanied by cars and trees, a table surface, on which we would put things — on the other hand — is often flanked by e.g. chairs. This leads to the requirement that the receptive field of a pixel should, ideally, cover the whole image because even distant pixels might be decisive for a local affordance.

This could well be in conflict with the second essential requirement, which demands that image details must not get lost during the forward pass of the network. Hence, object- and part-boundaries should be preserved. For example, many affordances concern rather smaller image aspects (e.g. a knob for pulling) and these aspects should not be lost by the network's operation.

With these requirements in mind, we propose and compare multiple neural network architectures. As it might be advisable to choose the architecture depending on the application we conduct an experiment to guide this decision in section 5.5.4. Our choice of models can be divided into two branches: PSPNet-based [246] and U-Net-based [172].

Both are deep convolutional neural networks that predict densely, i.e. per pixel. The former relies on the work of Zhao et al. [246], which proposed the pyramid pooling module (PPM). Evolving from the fully-convolutional network (FCN) [134], this model extracts features using a conventional encoder, applies the PPM on the obtained feature maps and upscales to the output tensor. In our case, feature maps are extracted using a 102-layer dilated ResNet encoder[3], hence the model will be called *P-102*. Our implementation of PSPNet focuses on the architecture and avoids training tricks like an auxiliary loss (see Figure 5.5). The PPM ensures that contextual cues can be processed, which we believe to be an important trait. Due to the small spatial

[3]We use this implementation https://github.com/fyu/drn

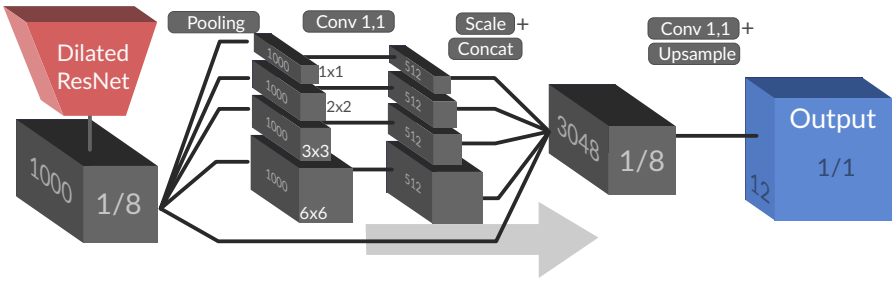

Figure 5.5: Architecture of the PPM module in the PSPNet model.

resolution output of the PPM, predictions tend to be blurry, if no further processing (such as an conditional random field [38]) is applied. Also, the PSPNet-based model is fairly complex, requires a lot of memory for training, resulting at smaller batch sizes, and is slow at inference.

Therefore we designed an alternative network following the U-Net [172] paradigm, which has the advantage of being faster to train and to run, compared to [246]. This model is depicted in Figure 5.6. The encoder is based on ResNet [88] and the architecture adopts the idea of refinement modules [165]. It had been shown by these authors that ResNet50 together with refinement modules successfully generates sharp object proposals, because refinement modules offer an elegant way for merging local with scene-level information. Thus, here we use a modified version of the architecture from [165].

This model integrates abstract information from deep layers with the spatially more accurate representations still present in less deep layers. Here both input layers will deliver maps of the same image size where they are then first stacked on top of each other (concatenated along depth) and subsequently convolved with the learned filters to obtain feature maps. While the base model is fixed to be a ResNet, we experiment with several configurations: The encoder size is varied from 18 to 152 layers, which heavily influences the execution (and training) speed of the model. In the decoder, we only vary the number feature maps in the last two refinement modules: In addition to the normal setting with 32 feature maps each we introduce a "small" setting involving 16 feature maps each (see DS in Table 5.4). We initialize the ResNet encoder with features obtained from ImageNet [53] pre-training, but do not freeze any weights.

5.3.4 Cost function

We propose a novel cost function we call selective binary cross entropy. This cost function deals with two aspects: 1) affordances are often not unique and for a given pixel multiple affordances may exist simultaneously. Hence, we imply a binary (present vs not present) probability dis-

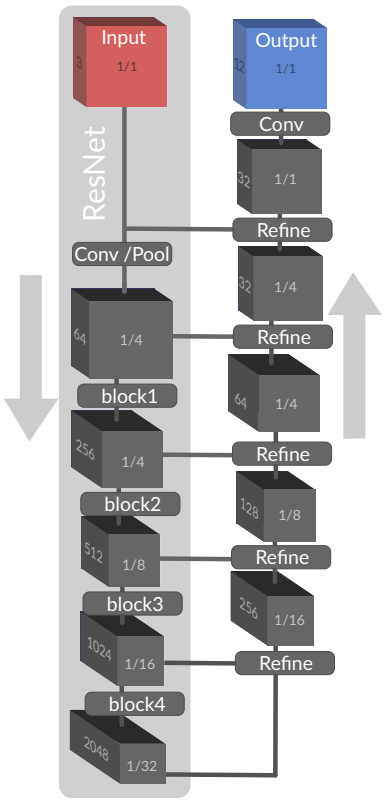

Figure 5.6: Architecture of the U-Net-based model. The ResNet encoder on the left involves four blocks. The boxes indicate the corresponding intermediate tensor sizes (number of channels and fraction of the image size).

tribution for each pixel and each affordance. 2) For some parts of the image no affordances may be defined. For those, we cannot tell whether an affordance is present or not, because the corresponding object or part is not found in the transfer table. However, since we also generate the corresponding validity mask, we know the location of the invalid regions. The idea is to incorporate also this information into the cost function.

This means during optimization we search for a model that agrees with the ground truth, but only where the latter is defined. For some regions no ground truth is defined. Here the model is free to predict whatever it considers to fit best and is not falsely punished due to over-generalizing annotations.

This concerns objects and object parts where no decision about the presence of an affordance can be made based on the object or part name alone. For example consider a bench. Does it afford placing-on? This depends on whether the bench's sitting surface is even. Does a coffee table afford support? Larger tables might but smaller ones do not. In these cases, we

prefer to ignore those uncertain fractions of the data in order to avoid false annotations. This is implemented by masking in the loss function.

Both aspects from above lead to the fact that commonly-used cost functions for semantic segmentation cannot be employed here. Subsequently, we will formally derive the here used *selective binary cross entropy cost function*.

We annotate the ground truth matrix of an image for affordance $a \in \mathcal{A}$ and pixel $i \in \mathcal{I}$ with $\mathbf{Y}_{ai}$ and the associated model prediction is given by $\hat{\mathbf{Y}}_{ai}$. Then the binary cross entropy BCE is defined by: $\text{BCE}(p, q) = -p \log(q) - (1-p) \log(1-q)$. This is summed up to render a scalar loss (cost), which captures the average binary entropy over all affordances and the image.

$$\mathcal{L}(\mathbf{Y}, \hat{\mathbf{Y}}) = (|\mathcal{A}||\mathcal{I}|)^{-1} \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{I}} \text{BCE}(\mathbf{Y}_{ai}, \hat{\mathbf{Y}}_{ai})$$

So far this definition is compatible with non-exclusive classes, but it does not yet account for incomplete data. To achieve this, we mask the cross entropy matrix, excluding all regions where no (or indecisive) affordances are present, before averaging. Masking is a very efficient and simple way for removing the incompleteness ambiguities and we get the following loss:

$$\mathcal{L}^{\text{m}}(\mathbf{Y}, \hat{\mathbf{Y}}) = (|\mathcal{A}| \sum_{i \in \mathcal{I}} \mathbf{M}_{ai})^{-1} \sum_{a \in \mathcal{A}} \sum_{i \in \mathcal{I}} \mathbf{M}_{ai} \text{BCE}(\mathbf{Y}_{ai}, \hat{\mathbf{Y}}_{ai})$$

with $\mathbf{M}_{ai} \in \{0, 1\}$ indicating if pixel i is valid, i.e. if a corresponding entry is found in the transfer table.

Contrary to [137], the mask M (of each image) is defined for every pixel *and* affordance. This allows us to specify uncertainties in the mapping from object/part names to affordances in a more fine-grained way during the learning phase. In test mode, the mask is no longer required.

In Figure 5.7 we highlight how the mask loss leads to a different gradient. It can be seen that the gradient is zero where the mask is zero. Hence, relevant weights are changed with higher magnitude while irrelevant weights are ignored.

5.4 Experimental Setup

5.4.1 Evaluation Datasets

The training and validation samples of ADE^T are generated from the ADE20K *training* dataset [4]. The ADE20K validation dataset is used for testing. It contains 2000 scenes and will be called

[4] Here you can use the interactive dataset browser to get an impression of how the samples look like: http://groups.csail.mit.edu/vision/datasets/ADE20K/dataset_browser/

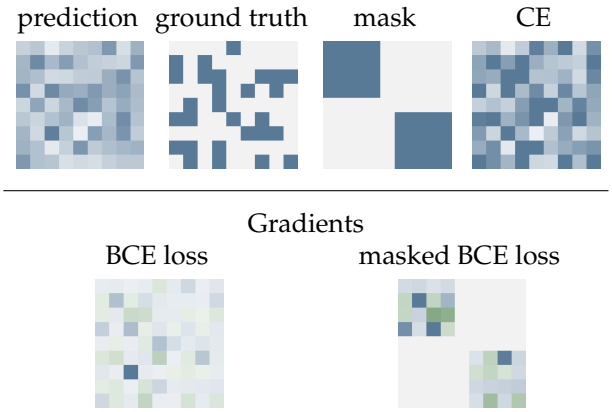

Figure 5.7: Illustration of how masking changes the gradient. The top row depicts network prediction, ground truth, a toy mask and the binary cross entropy computed from prediction and ground truth. In the bottom row the gradients resulting from conventional CE and masked CE loss are shown (green indicates negative values). Gradients are used to change the preceding layer's weights. We can see that masking leads to stronger changes in relevant regions while irrelevant weights are ignored (ie. not changed).

ADE^E. From this we manually pick 50 high quality (sharp, multi-object) images and transfer the annotations to affordances using the table. Then we let a person (expert) manually correct this according to the definitions provided by Table 5.2 by editing each affordance map individually with an image editor. Due to this manual correction, systematic errors of the part-to-affordance conversion procedure are punished during evaluation and we obtain a more realistic estimate of the error. Hence, this metric assesses a stronger form of generalization. This dataset will be called "Expert50" subsequently. While a number of 50 samples might seem small, note that the networks outputs probabilities for every pixel and affordance resulting in a few million predictions even in the expert dataset. Hence we consider this dataset reliable enough to be used for evaluation. Scores are determined by comparing predictions of our network on the test set with corresponding ground truth data according to the metrics that will be discussed next.

5.4.2 METRICS

To quantify the performance, we use the intersection over union (IoU) metric (sometimes referred to as Jaccard Index), which is defined as follows:

$$\text{mean IoU}(\mathbf{Y}, \hat{\mathbf{Y}}) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} \frac{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \wedge \hat{\mathbf{Y}}_{ai} = 1]}{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \vee \hat{\mathbf{Y}}_{ai} = 1]},$$

following the notation introduced in section 5.3.4 with $\mathbf{Y}$ denoting ground truth and $\hat{\mathbf{Y}}$ a model's prediction. $1[\cdot]$ is the indicator function defined as:

$$1[c] = \begin{cases} 1 & \text{if } c \\ 0 & \text{else} \end{cases}$$

Maximal IoU would be 1.0. It is important to note that IoU is measuring the overlap with the ground truth image segment area and punishes both, lack of overlap in the labeling as well as false positive outside-of-segment labeling. To compute this, the probabilistic predictions of the network must be binarized. In our experiments we use two fixed threshold values: 0.1 and 0.5 for this.

Additionally, we report the mean average precision. This metric has the advantage of not relying on a single threshold level but averages over multiple levels. It is based on the precision P defined as follows

$$\mathrm{P}(\mathbf{Y}_a, \hat{\mathbf{Y}}_a) = \frac{\sum_{i \in \mathcal{I}} 1[\mathbf{Y}_{ai} = 1 \wedge \hat{\mathbf{Y}}_{ai} = 1]}{\sum_{i \in \mathcal{I}} 1[\hat{\mathbf{Y}}_{ai} = 1]},$$

which is evaluated at several recall levels to obtain an average precision. This is equivalent to the area below the precision-recall curve. The mean average precision is obtained by computing an average score over all (affordance) classes.

5.4.3 Implementation Details

The models are trained on a single Geforce 1080 Ti GPU or Titan V with a pytorch backend [158]. Weights are updated using RMSprop [203] with mini batch sizes of 16 (or 8 for large models) and a learning rate of 0.0001. The training is stopped after 25 epochs.

5.4.4 Configuration

We assess various components of our model with respect to their impact on the performance.

Encoder Since a large fraction of the computational budget is dedicated to extracting features using the encoder, it seems justified to investigate different choices in more detail. For our U-Net based model, we constrain the analysis to three common ResNets [88] having 18, 50 and 152 layers, since they provide a trade-off between accuracy and speed, which is suitable for our task. While a larger encoder tends to exhibit a better performance it might also be more sensitive to overfitting due to its larger number of parameters. Depending on the number of layers these models will be referred to as R-18, R-50 and R-152. The PSPNet is only evaluated in one configuration with a 105-layer encoder involving dilated convolutions, it is called P-105.

Table 5.4: Ablation study. This experiment has been conducted exclusively on R-50 and ADE^T as training dataset. DS: decoder size, IMP: importance sampling, SEL: scene selection, PRE: ImageNet pre-training, M: masked loss, CM: class mean, CW: class weighting. For IoU, 0.1 and 0.5 are binarization thresholds. The expert columns refer to manually corrected samples as explained in Section 5.4.1.

							ADE^E			Expert50		
							IoU			IoU		
DS	IMP	SEL	PRE	M	CM	CW	@0.1	@0.5	mAP	@0.1	@0.5	mAP
			✓	✓			41.5	**45.6**	**61.8**	**36.5**	**37.2**	**53.4**
	✓	✓	✓	✓			35.6	38.8	51.5	35.2	35.1	51.1
	✓		✓	✓			41.8	44.9	54.0	34.5	34.2	52.9
			✓				42.4	42.3	58.7	33.4	32.9	48.9
				✓			34.7	36.3	51.4	33.0	28.7	50.9
			✓	✓	✓		35.3	39.1	54.5	31.1	30.0	48.9
			✓	✓		✓	37.7	39.6	53.8	31.0	29.4	48.0
s			✓				**43.0**	41.1	58.4	33.4	31.0	48.4
s			✓	✓			41.1	44.7	50.5	35.3	34.7	51.3

Decoder size Analogously to the encoder, also the decoder can be configured in different ways. While we keep the number of 5 skip connections constant, we vary the number of feature maps each decoder uses.

Masked Loss Above we described how to take care of undefined values in our loss function. In our experiments we empirically evaluate whether these changes actually lead to an increase in performance.

Pre-training: Knowledge acquired for recognizing images can be leveraged in affordance segmentation. Does the model benefit from pre-trained features or is the architecture (ResNet) good enough as a prior?

5.5 Results

5.5.1 Ablation and Additions

First we conduct an ablation study to identify the best performing configurations and validate the impact of various modifications (see Table 5.4). The encoder is fixed to be R-50, i.e. a ResNet with 50 layers.

In Table 5.4 we present our analysis on various modifications regarding sampling, loss and architecture on the base model R-50. Abbreviations in the text refer to the respective table columns. The columns of this table indicate different configurations of the respective model as discussed in Section 5.4.4 and the associated scores on: ADE^E and the expert dataset.

First, we empirically verify the utility of masked loss. With mask, scores tend to be higher by around 2 to 4 percentage points. Further additions address the problem of infrequent classes, i.e. those affordances that are rare and cover small areas. One natural way to tackle the unequal distribution of classes in the dataset is giving less frequent classes more weight. We experiment with two mechanism to achieve this. Class-specific weights (CW) in the loss and Class-Mean loss (CM). In the former case we multiply with manually specified, class-specific factors in the loss tensor before summing it up to a scalar. The factors roughly express how rare a class is and the idea is to compensate for rare classes. In the latter case, instead of computing the loss individually for every pixel, we compute a mean over class-specific losses, resulting in each class being weighted equally. Since both metrics also average over class-specific scores we would expect them to increase, as rare classes should perform better. However, this is not the case. We even observe a substantial drop in performance if CM or CW is enabled. An alternative to manipulating the loss function is to change the data that is fed to the network during training. Instead of showing a uniformly sampled cropped image exactly once per epoch, we changed the sampling to include on rare classes in the crop with higher probability (IMP) and even excluded images that do not contain specific classes (SEL). Also, these modification did not lead to a better performance but are kept in the paper for completeness. We conclude that data quantity outweighs data quality at least for this task.

During experimentation we found the choice of the optimizer and its learning rate to be crucial for the performance of rare classes. High learning rates drive the optimizer into a minimum where rare classes are never predicted. Hence, in the presence of highly imbalanced classes it seems advisable to use dynamic learning rates as conducted by RMSprop and other more sophisticated gradient descent algorithms.

5.5.2 Simulated Data

With around 20,000 training samples the ADE dataset is fairly small compared to other datasets which are common in deep learning such as COCO [130] or OpenImages [115]. To compensate for the small number of training samples we generated simulated scenes. We propose and assess two methods of integrating simulated data with real world data:

- *Joint Training* In addition to models trained on individual datasets, we also train models on both datasets conjointly: The datasets are first concatenated and then randomly mixed. This way, each mini-batch for training can encompass samples from both datasets and each update of the network weights through the gradient will reflect this. We will refer to this training method by $+$, i.e. $A + B$ means joint training mixing samples from A and B.
- *Simulation Pre-Training* The network is first trained exclusively on simulated data. Sub-

Table 5.5: Comparison of combined training strategies. The '+' denotes joint training while $\rightarrow$ indicates multiple subsequent training procedures. The expert columns refer to manually corrected samples as explained in Section 5.4.1

			ADE^E			Expert50		
			IoU			IoU		
model	train data	M	@0.1	@0.5	mAP	@0.1	@0.5	mAP
R-50	ADE^T	✓	41.5	45.6	61.8	36.5	37.2	53.4
R-50	ADE^T + Sim^T	✓	43.9	46.5	58.9	37.4	37.4	54.3
R-50	Sim^T	✓	16.1	14.6	20.9	18.1	17.7	24.0
R-50	$Sim^T \rightarrow ADE^T$	✓	42.8	46.2	60.2	37.5	38.8	53.9
R-50	COCO $\rightarrow ADE^T$		**45.6**	41.7	59.4	34.7	32.1	48.9
R-50	COCO $\rightarrow ADE^T$	✓	43.4	47.1	54.3	37.1	38.0	53.6
P-105	Sim^T	✓	13.6	11.8	16.6	18.6	18.0	24.9
P-105	ADE^T	✓	44.3	**48.5**	**62.3**	37.9	38.2	52.7
P-105	ADE^T + Sim^T	✓	44.7	47.1	59.8	**38.2**	**39.4**	54.1
P-105	$Sim^T \rightarrow ADE^T$	✓	42.7	46.1	59.2	37.7	37.9	**54.5**

sequently, in a second training stage, the network is fine-tuned to the target dataset. To reflect the sequential nature of this training we denote it with an arrow ($\rightarrow$).

As an alternative to simulated data, we pre-train networks on the COCO dataset [130], too. Our results are presented in Table 5.5.

The R-50 model seems to take advantage of additional data and improves most scores (with mAP on ADE^Tbeing the exception). The increase is subtle, though. On P-105, performance on some scores even decreases when simulated data is used. Regarding the comparison between joint and pre-training we note that both methods perform on-par. Possibly, this could be explained by the limited variability in the simulated scenes we employ here. In general, simulating data seems a promising option if data is scarce but a more sophisticated simulation model, which is beyond the scope of this paper, would possibly be required.

We also find that pre-training on the COCO dataset yields a similar performance boost as the simulated dataset. However, considering the enormous annotation efforts of COCO we still think that simulation is a better option for pre-training. Comparing both scores involving COCO also confirms the usefulness of the loss masking, which was discussed earlier.

5.5.3 Comparison to State-of-the-Art Model

Although some previous work is akin to the idea of affordance segmentation we only find the work of Roy and Todorovic [173] to be suitable for a direct comparison. They evaluate

Table 5.6: Performance on the Oregon dataset. NYU indicates the original roughly 50-50 train-test split, while OWN uses more samples for training.

		IoU		
model	train dataset	@0.1	@0.5	mAP
R-50	$\text{Oregon}_{\text{NYU}}$	26.0	16.8	34.5
R-50	$\text{Oregon}_{\text{OWN}}$	29.3	20.5	39.4
P-105	$\text{Oregon}_{\text{NYU}}$	30.1	22.1	42.6
P-105	$\text{Oregon}_{\text{OWN}}$	34.2	26.9	50.3
R-50	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{NYU}}$	53.0	56.4	72.4
R-50	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{OWN}}$	58.1	**65.5**	**83.5**
P-105	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{NYU}}$	**59.0**	63.0	82.4
P-105	$ADE^T + Sim^T \rightarrow \text{Oregon}_{\text{OWN}}$	58.8	63.0	81.9
Roy and Todorovic [173]		49.6		n/a
Roy and Todorovic [173] using GT cues		53.2		n/a

their system on images of the NYUv2 [186] dataset, which provides depth maps in addition to RGB images as they rely on 3d scans for training their network. They collected pixel-wise annotations for a set of five affordances for all images of the NYUv2 dataset. This set does not directly correspond to our affordances thus we cannot directly run a network, which was trained with our method on their data. Instead, we make use of transfer learning: We take a trained model and replace the last layer responsible for classifying with a new one involving only five classes. The weights of all other layers are maintained. During training all weights are changed, i.e. the transfer learning only affects initializations.

The experiment involves different models and two different train/validation/test splits. One is the original NYUv2 split used in [173], which divides the 1,449 training samples in almost equally sized halves. The disadvantage of this split it that the training set is very small. Therefore we incorporate a more training-heavy split into the analysis, involving 1,100 samples for training and validation leaving the remaining samples for test.

The results reported in Table 5.6 reveal a strong improvement over state-of-the-art by models that were pre-trained using our method. Baselines, that were trained on the NYUv2 affordance dataset only but with encoders being pre-trained on ImageNet, performed much worse and don't even come close to state-of-the-art. Pre-training, in this case by join-training, on real and simulated data yields a large performance improvement. All of our models that were trained using this method outperformed the network of [173]. By switching to a more training-heavy split, we are able to obtain IoU scores up to 64.5, which is more than 12 percentage points above state-of-the-art results which used ground truth cues.

Also the qualitative results shown in Figure 5.8 are remarkable. In the bottom row, the P-102

Figure 5.8: Probabilistic segmentations generated using our method on the NYUv2 dataset [186]. Columns (left to right): original image, predictions of P105, and ground truth.

network even discovers a place-able surface that was not annotated in the ground truth. Apart from that, the depicted predictions of walk-able, grasp-able and place-able are close to ground truth.

5.5.4 Model Comparison

Following the conclusions we can draw from the previous experiments, we evaluate a set of specifically tuned models for different purposes and discuss the specific trade-offs. This is necessary as affordance segmentation can be used in a variety of diverse environments, such as a mobile robot or a fixed installation. Each of these environments has its own requirements with respect to inference speed and CPU/memory demands. As the encoder carries out a large share of the computations we compare different encoder sizes, all based on the ResNet architecture. For comparison, we also report PSPNet scores. Following our nomenclature from above, the number after R denotes the number of layers. Results are shown in Table 5.7. We find that a larger encoder does not improve performance, while a smaller one only has a slight impact on performance.

5.5.5 Affordance-wise Evaluation

Table 5.8 reports individual scores for selected configurations. We can observe a strong variation in performance across the affordances. For example, obstruct, walk and support are learned

Table 5.7: Comparison of models. Best performance is achieved by the most complex models, but even very simple models (R18) can perform well.

		ADE^E			Expert		
		IoU			IoU		
model	train data	@0.1	@0.5	mAP	@0.1	@0.5	mAP
R-152	Sim^T	16.0	13.9	20.8	18.6	17.8	25.5
R-152	$Sim^T \rightarrow ADE^T$	39.3	43.2	56.9	36.9	34.9	54.2
R-152	$ADE^T + Sim^T$	43.1	44.6	56.8	35.7	35.0	53.1
R-50	Sim^T	16.1	14.6	20.9	18.1	17.7	24.0
R-50	$Sim^T \rightarrow ADE^T$	42.8	46.2	60.2	37.5	38.8	53.9
R-50	$ADE^T + Sim^T$	43.9	46.5	58.9	37.4	37.4	**54.3**
R-18	Sim^T	16.9	14.9	23.0	20.0	17.6	27.5
R-18	$Sim^T \rightarrow ADE^T$	41.7	44.1	56.8	35.6	35.6	51.2
R-18	$ADE^T + Sim^T$	43.3	44.8	57.6	36.9	35.6	53.2
P-105	$Sim^T \rightarrow ADE^T$	42.7	46.1	59.2	37.7	37.9	54.5
P-105	$ADE^T + Sim^T$	44.7	47.1	59.8	38.2	**39.4**	54.1
P-105	$ADE^T + Sim^T$	44.1	46.8	58.7	**38.5**	38.2	53.9
P-105	$ADE^T + Sim^T$	**45.3**	**47.4**	**63.1**	38.4	36.8	53.8

Table 5.8: Performance for individual affordances. The IoU threshold is 0.1. * indicates that class mean loss is used.

		obstruct		break		sit		grasp		pull		tip-push		illum.		observe		support		place-on		roll		walk		mean	
model	dataset	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP	IoU	mAP
P-102	$Sim^T \rightarrow ADE^T$	87.5	94.1	41.6	71.8	40.4	53.7	19.3	29.5	3.3	1.6	3.1	11.2	7.4	24.9	23.2	40.1	54.4	89.7	29.4	44.6	72.3	96.3	70.8	96.0	37.7	54.5
R-152	$Sim^T \rightarrow ADE^T$	86.4	94.1	38.6	69.2	30.6	41.3	19.0	34.2	1.2	0.8	8.6	5.4	5.5	30.4	28.0	47.5	53.5	89.1	24.6	45.9	75.0	96.6	72.1	96.2	36.9	54.2
R-50	$Sim^T \rightarrow ADE^T$	87.6	93.5	38.1	73.2	41.4	60.2	17.7	25.0	1.2	1.0	5.3	3.0	12.4	23.7	27.4	40.7	53.3	90.2	27.0	42.7	69.4	96.9	69.2	96.7	37.5	53.9
R-50	Sim^T	79.4	87.7	14.5	16.9	8.8	11.9	8.0	7.7	0.0	0.4	0.0	0.2	16.0	23.3	3.5	8.4	27.8	50.0	14.4	21.1	22.5	30.5	22.7	29.8	18.1	24.0
R-18	$Sim^T \rightarrow ADE^T$	86.6	93.7	36.2	67.1	35.3	46.8	19.3	30.2	2.3	1.0	2.7	1.3	11.1	19.8	24.2	42.4	51.9	88.5	21.0	32.1	69.1	96.1	67.5	95.8	35.6	51.2
R-50*	ADE^T	85.7	93.9	27.1	60.6	27.4	42.0	16.8	23.0	0.0	0.6	0.0	0.5	11.6	30.6	21.6	36.3	49.7	85.3	18.4	27.8	59.8	93.0	54.6	92.9	31.1	48.9

well while grasp, place-on and observe turn out to be more challenging. In particular the small structures of pull and tip-push seem to be hard to predict as their scores are close to zero. This is probably due to these structures not only being small but also rare. This means they need to be learned from less samples than other classes, which is more challenging. Additionally, their geometry might be harder to learn, in particular as features are more difficult to be recognized due to their small size. The more complex encoders of PSPNet and R-152 exhibit the best performance on these small classes, which might be due to their higher capacity allowing them to preserve details of smaller structures while encoding. At the same time, larger networks are more prone to overfitting, because of their larger number of parameters, which limits their overall performance.

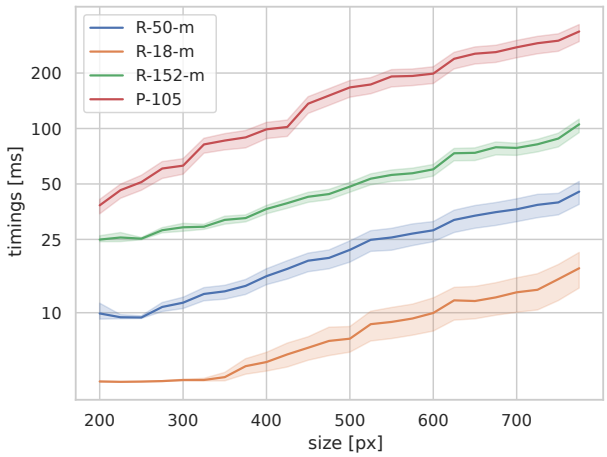

Figure 5.9: Speed vs. accuracy trade-off between the models. Note, we use the log scale of timings for better readability, the execution speed grows exponential with image size.

Note, the scores of zero do not mean that rare affordances are never predicted as we can see in the qualitative evaluation. Also, due to the cross entropy loss encouraging cautiousness, the predictions are fairly weak such that false negatives are common if the same threshold value is used for all affordances. A possible way to overcome this in practice would be to use a very low threshold (<0.1) to generate candidates and then apply an application-specific post-processing to remove false detections. Alternatively, false detections could be filtered by rule-based approaches using heuristics.

As we can see in the last row, modifying the loss to be a mean over class-specific losses instead of a pixel-wise loss does not improve in rare classes. It seems that, for now, the most straightforward way to get better performance would be to use more training samples.

5.5.6 Speed Trade-off

In practice, networks cannot be arbitrarily large and the availability of memory is limited. This holds in particular if a robotic platform is used, which runs on batteries. To obtain an intuition on the respective trade-offs of the models we report inference time and memory footprint of selected models in Figure 5.9. Here, we constrain the batch size to one as we expect that responses of the system are required immediately in realtime systems.

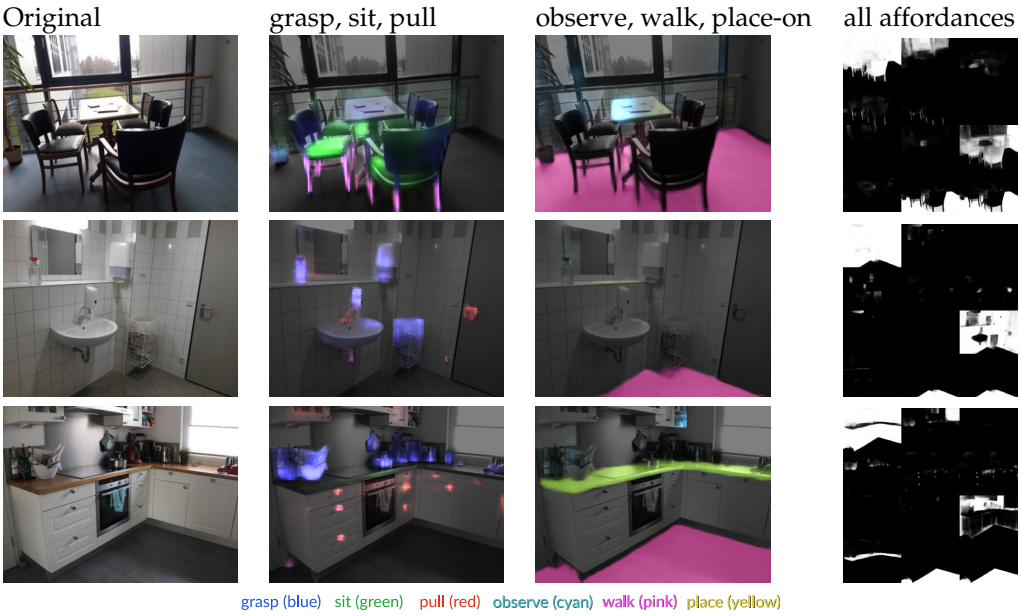

Figure 5.10: Probabilistic segmentations generated by the network P-102 jointly trained with simulated data. From left to right: Original Image; predictions grasp (blue), sit (green), pull (red); predictions of observe (cyan), walk (pink), place (yellow); all 12 affordance map predictions in their original form with the intensity corresponding to the presence probability of the affordance. Note, these images are not from the ADE dataset and therefore no ground truth is shown.

5.5.7 Qualitative Evaluation

In addition to these quantitative findings we now discuss qualitative output of the models on various different scenes. In Figure 5.10 we show predictions of network P-102. For these results we do not use binarization but visualize the probabilistic predictions directly. All colored pixels encode the probability for the corresponding affordance by color intensity. This renders an assessment of the degree of confidence the model attains for any given pixel's affordance. Mixed colors indicate the presence of multiple affordances. The here used images challenge the network with difficult situations like front-lighting and transparent materials. We observe that the network generalizes well to these new situations, even small structures are mostly correctly predicted, for instance the drawer knobs in the bottom row. There are some false positive cases: The tissue and soap dispenser and the trash bin in the second row are marked as grasp-able. However, in order to know that these items are actually fixed one would need to physically inspect the scene (the trash bin could well be detachable), so, given only an image, these predictions are plausible. Another observation is that the prediction intensity is sometimes weak, as in the second column for place-on. However, depending on the application, this can overcome

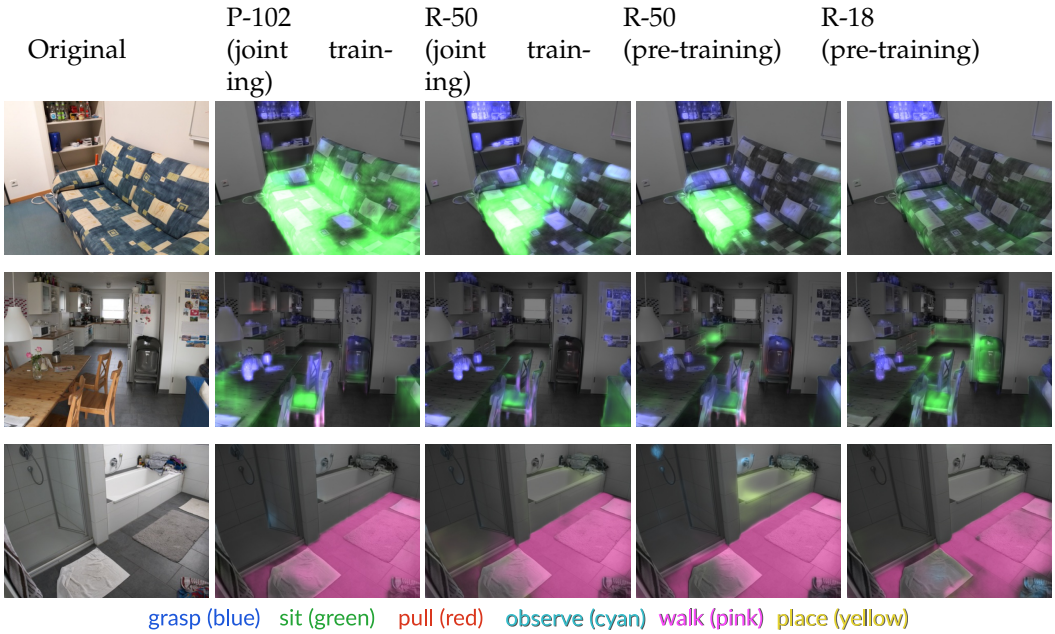

Figure 5.11: Probabilistic segmentations generated by different networks. Row 1 and 2: grasp, sit, pull, Row 3: observe, walk, place. Note, image sizes vary as the networks are fully convolutional and there is no need to rescale to a common resolution.

either by scaling the intensities or By some post-processing.

Figure 5.11 features a qualitative comparison showing the same scenes but predictions of different models. Here we can see that the predictions of the networks look fairly similar, which could be expected since both were trained on the same data. Furthermore, the subtle differences between the models are in accordance with the quantitative findings above.

5.5.8 Robot Experiment

In order to demonstrate the potential of the proposed system we performed a robotic experiment using a KUKA LWR robot-arm [121] with a 3-finger Schunk SDH hand [184]. The task for the robot was to conduct simple manipulation actions based on an affordance segmentation of the scene.

Execution of robotic actions is based on our previous work and was implemented using a library of manipulation actions [1, 2], which utilizes modified dynamic movement primitive (DMP) framework [97, 122], for trajectory generation. DMPs are formalized as stable attractor dynamic system and can generalize to new start and end points while being robust against perturbations. Specifically, here we used "pick-and-place", "take down" and "push" (to perform

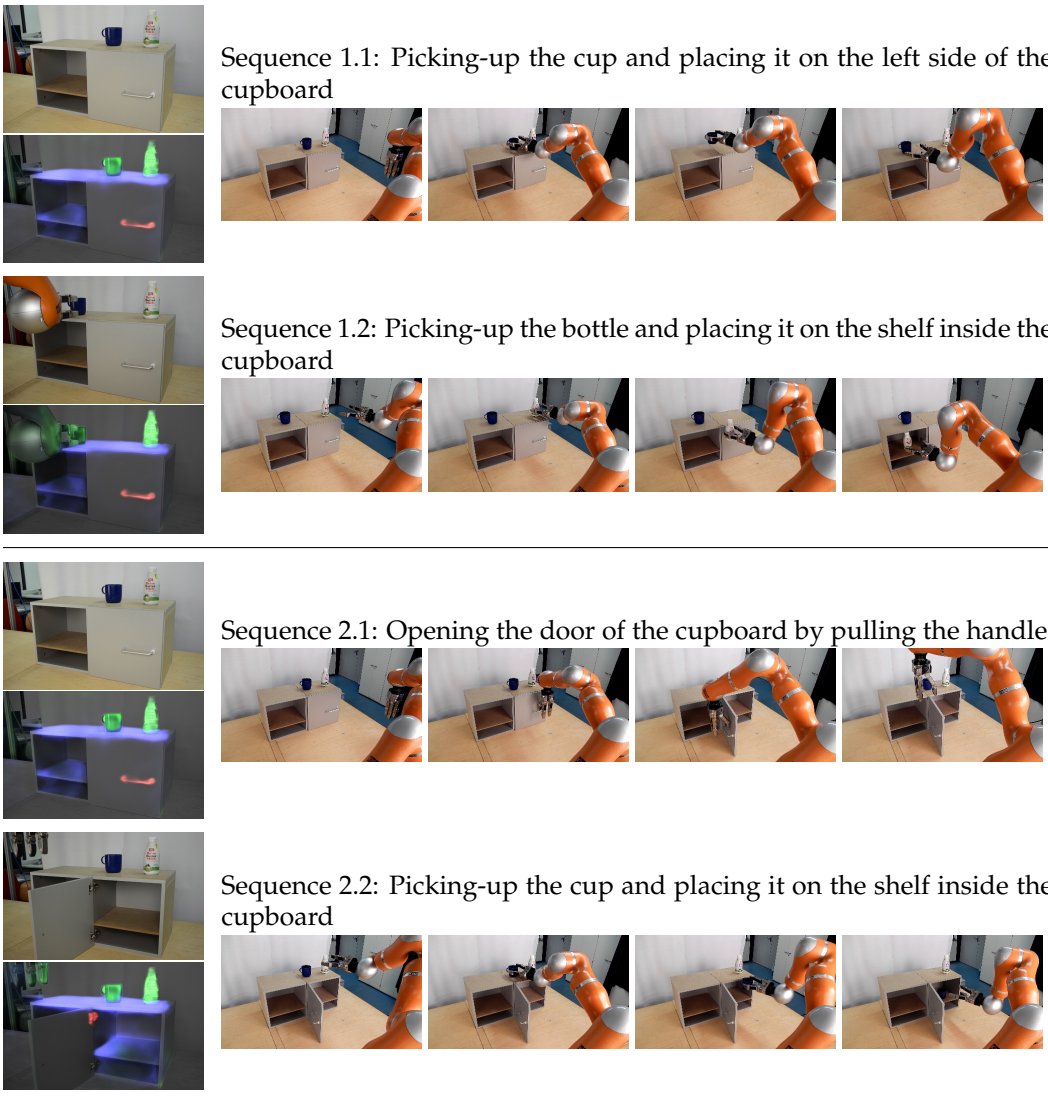

Figure 5.12: Results of execution of two action sequences based on affordance maps obtained using R-50. Affordance maps (left) and selected frames of robot action executions (right) are shown. For the complete experiment see supplementary video. Colors denote grasp (green), place (blue) and pull (red).

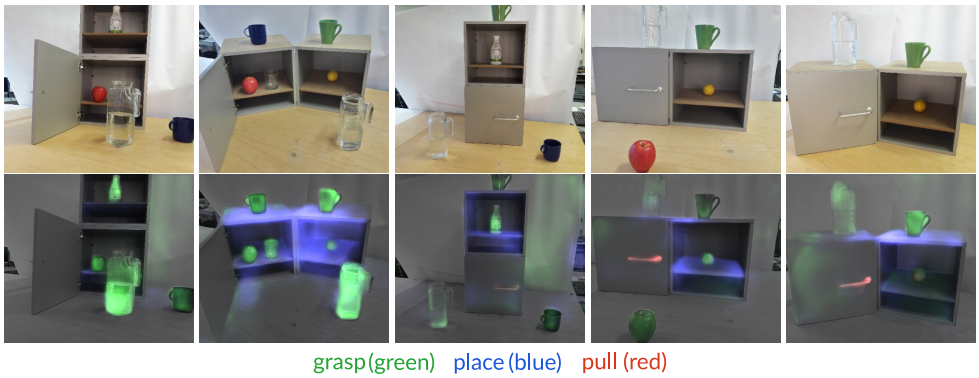

Figure 5.13: Qualitative Evaluation in a lab setting using R-50. Left: Frames from the experiment with corresponding affordances, right: the same objects from a different perspective. Colors denote grasp (green), place (blue) and pull (red).

pull) actions to manipulate an object or an object part in the scene. For simplicity, in our study we used predefined object poses. In general, this can be done using existing state-of-the-art methods for pose estimation [157, 44], however, this is out of the scope of our study.

We show how affordance maps enable the robot to select and perform actions that are at that moment available in a scene. As we are not interested in any kind of planning problem, we let the robot "decide what to do" on its own, implementing some kind of playful mode. The robot had to select and perform two actions in a sequence. Hence, affordances need to be re-analyzed after the first action. However, this does not perturb execution of the second action due to the speed of affordance computation. Figure 5.9 has already shown that new affordances can be provided in realtime.

As in all quantification experiments above, affordances are assigned in a pixel-wise manner, where the color-intensity in the visualization (Figure 5.12) indicates the existence probability of the affordance. Hence, thresholding the affordance map enables determination of potential target locations, such that unlikely places, like "placing on the bottom shelf" (faint blue intensity), are ruled out. No explicit object knowledge is required for this task and – as mentioned – no planner was used. The general setting is illustrated in Figure 5.1.

We show two such action sequences resulting from this setup given the same initial conditions: 1) pick-and-place a cup and take a bottle down; and 2) pull a handle and take a cup down. Note that pick-and-place and take down actions consist of a sequence of grasp, place and release actions. Results of the robot experiment are shown in Figure 5.12, where we show affordance maps obtained using R-50 and the key frames of the robot action execution (please see supplementary video for the full experiment). It is important to stress that the network has not been trained on this kind of the scenes. We can see that the affordances of objects (the bot-

tle and the cup) and object parts (the door handle and the shelf) were correctly identified and action sequences successfully executed.

In order to show robustness and generalization abilities of our approach we have also generated affordance maps for five other scenes with different object configurations and different perspectives. The scenes and their corresponding affordance maps are presented in Figure 5.13. Again, note that the training set (same as for Figure 5.12) for these experiments consists of scenes, which are quite different from the ones shown here. It can be seen that most of the here-considered affordances are correctly assigned and not many errors are found. Challenging (e.g. transparent) objects are also correctly identified. The ground plane is usually not considered for a placing affordance as the system would recognize it rather as "walk-able" (coloring not shown). Surfaces, which are nearly horizontal, appear in the 2D image with only a few pixels and the system also considers them not for placing. Furthermore, some other placing-surfaces are only partially detected. But note, that these results are all based only on single 2D views. Methods for accumulating knowledge (e.g. based on a voting scheme across a sequence of images) can without problems be added to further improve on this. However, already these results demonstrate that our approach can generalize very well to different scenes with variable object configurations even though the network has not been specifically trained on such scenes.

5.6 CONCLUSION

In this paper we have described a method that labels a comparatively large set of 12 affordances pixel-wise given only single 2D RGB images. We have shown how to construct an affordance training dataset from object parts segmentation and apply recent semantic segmentation methods to learn affordances effectively. An extensive analysis on the impact of modifications to the loss, sampling and architecture has been carried out. The state-of-the-art method of Roy and Todorovic [173] is substantially outperformed by adopting features that were obtained using our method.

A strength of our method is that it is fast (less than 10ms) while operating on 2D images of arbitrary size. It is applicable on all kinds of scenes, even in presence of light-absorbing, transparent and reflecting materials where structured light cannot be used. Hence, it can easily be adapted to multiple scenarios. The transfer to a scene that had not been part of any training set was not a problem in the here-shown robotic test. When our method is to used in practical applications it can easily be modified: The set of affordances can be adopted according to the capabilities of a robot or detection thresholds of individual affordances can be modified. The field of autonomous robotics, for example considering service robots, relies heavily on semantic scene analysis methods. We think that the here-presented fast and simple 2D-affordance detec-

tion can be used to provide a machine with good estimates of what to do in a scene using few computational resources. Given a task ("clean up the room") and pairing this type of affordance analysis with planning algorithms would make such a system applicable in different domains that require autonomous action decisions.

ACKNOWLEDGMENT

This research was funded by the European Union H2020-ICT-2016-2017/H2020-ICT-2016-1 / 731761 (IMAGINE)

5.7 REFERENCES

We maintain a single list of references at the end of the thesis.

CHAPTER 6

Action Plausibility Rating

Previously, in Chapter 5, we have seen a system that detects opportunities of interaction with the environment. However, depending on the situation, these interactions may not be equally appropriate. Drinking a glass of wine on a laid table is more plausible than drinking a glass of wine that stands next to a pile of dirty dishes to be cleaned up. Food is normally served on clean dishes while dirty dishes are put to the dishwasher. Such knowledge is widely applicable. Not only for robotic systems but all systems that interact with humans, it is crucial to understand these differences in plausibilties to be helpful in everyday use or make reasonable suggestions. However, when a scene is represented on the object level (e.g. labels and positions of objects) such differences cannot be considered. In order to make informative guesses about plausible actions the appearance and context of individual objects must be considered. In the following article, we implement this idea by directly mapping image input to action plausibilities. Following the motivation from the introduction (Section 1.1.2.1), we explicitly evade object-based intermediate representations and use a distributed knowledge representation in form of synaptic weights.

Summary The following article addresses the problem of inferring the plausibility of actions from object images. For this, we define a set of 10 actions and ask human raters to judge how likely they consider these actions in a large set of images. We collect annotations from eight raters using a web-based interface. We train a CNN to predict the distribution of these gathered ratings, which encode the likelihood of actions. For proper tracking of performance we introduce three metrics as well as meaningful baselines. Furthermore, we investigate various forms of integrating object context. The results indicate that the system works well and differentiation of plausibilities is generally possible. Incorporating the context yields only minor performance gains.

Timo Lüddecke and Florentin Wörgötter

Fine-grained Action Plausibility Rating for Robotic Action Selection

(unpublished)

Abstract

An essential capability of humans is the effortless identification of useful tasks based on visual cues in everyday situations. Object appearances and contexts are integrated and processed to differentiate plausible from implausible actions. In this work, we study how to teach this ability to robots. In contrast to many tasks in computer vision where the goal is an accurate scene description (object labels, caption) of the present scene here the challenge is to make reasonable guesses about the future outcome of an action. To this end, we collect a dataset that associates images with probabilities over a set of actions. A convolutional neural network is trained to match these ground truth plausibility scores using this dataset. We compare the performance of state-of-the-art encoder architectures and specifically analyze the role of contextual cues quantitatively. While the object recognition capabilities of the encoder have a strong impact on performance, using context did not lead to substantial improvements. We show qualitatively the utility of such a system for robotic action selection in a household setting.

6.1 Introduction

In a given situation humans often have plenty of action possibilities, but commonly only a tiny fraction is appropriate. Making such action decisions in everyday life feels effortless, which is partly due to our common-sense knowledge. The sense of appropriateness that guides the decision is probably not innate but learned, while growing up. Robotic systems, however, naturally lack this skill and therefore can exhibit a behavior that is surprising and unexpected for humans due to the robot's misinterpretation of a situation. Thus, transferring this kind of common-sense knowledge to machines would have a great impact on their usability, in particular for situations where interaction with humans is required.

The problem of representing common-sense knowledge itself is not new and has been addressed for decades. Most of these past approaches to represent common-sense facts are symbolic, i.e. they assume that knowledge can be expressed in terms of a finite set of discrete symbols and their relations. While this comes with the advantage of interpretability, it is unlikely that all knowledge can be expressed in this form, especially not inherently continuous facts

Figure 6.1: Sketch of the idea presented in this paper: Small changes in the image can have a vast impact on the plausibility of actions.

as *the likelihood of eating from this dish decreases with its level of dirtiness*. Examples of this kind of knowledge representation can be found in the psychological literature where action possibilities (affordances) are modeled as probabilistic functions instead of binary attributes [66]. Symbolic approaches fail if they only take class labels into account as contextual and appearance details are crucial for such a task. Of course, with quite some effort these aspects can be incorporated into symbolic systems, too. However, for this all relevant details need to be known and specified a-priori. If the fill-level of a cup is critical for drinkability, a logical variable "fill-level" needs to be added to the system along with an image recognition component that can detect it. The same is true for each and every such object and situation-dependent aspect leading to a massive effort in pre-defining all of this in the right way. Different from that, the advantage of our approach is that there is no need to explicitly model the space of all variables influencing an action as the system learns these relationships end-to-end.

In this article we address the novel problem of rating how plausible certain actions are. An illustration of this idea is shown in Fig. 6.1. For this purpose, we develop a hybrid system that represents common-sense knowledge in a distributed, implicit way but also relies on a hard coded action compatibility table that defines if actions can in-principle be conducted on different object classes. We take images from the OpenImages dataset [115] and then ask humans to rate how plausible they consider certain actions. Having obtained a such-labeled dataset, we train a neural network to predict action plausibilities, which relies on the implicit encoding into the network of the human common-sense knowledge during training. This way rules like *if dirty dishes are close to each other stack them* or *if room is empty use remote control to turn off TV* could be learned from data. Importantly, after training, the system is able to directly map from pixels to action plausibility probabilities and a symbolic representation is not any longer needed.

Simple symbolic mappings (like object labels mapped to actions) would be doomed anyhow, because commonly object classes are very broad (high variance). For example the class "cup" contains images of full and empty cups as well as cups with and without handle. However, actions often depend on the state of the object, which cannot be inferred from the label only. Often, the state can be much more informative concerning an action than the object label, since an action can be compatible with a broad range of object classes in a certain state.

6.2 Related Work

We are not aware of any approach that explicitly deals with the problem of rating actions with respect to their plausibility from observed scenes. However, several related tasks have been addressed before. Here we present an overview, differentiated by the input data the methods use.

Video-based Methods A large body of work in anticipation operates on videos, which seems natural since movies provide a large temporal context to base predictions on. In the work of Lan, Chen, and Savarese [123], the next action in a TV show is predicted based on previous frames and object bounding boxes. For this a hierarchical video representation called movemes is proposed. The anticipation of human activities that is addressed in Koppula and Saxena [114] can be considered a closely related task. They model human pose, object affordances, object locations and sub-activities in a graph that changes over time through a temporal conditional random field. By sampling from this model, prospective activities can be predicted. These possible futures could also involve actions we are interested in. While their dataset only comprises 120 scenes, we prefer a larger number of scenes to allow for more detail within scenes. Vondrick, Pirsiavash, and Torralba [220] model the development of visual feature representations (obtained from a CNN) over time in a self-supervised setting. Some video recognition approaches have been evaluated in an early recognition setting [252, 249]. Given only a certain fraction (e.g. 20%) of the first frames of an action, the goal is to determine the action, which can also be seen as a weak form of anticipation.

Our task differs from the tasks addressed in these paper in using only a single RGB image as input. This implies that models cannot rely on patterns that occur in sequences of actions to generate predictions but have to identify cues only from the provided single image.

Still Image-based Methods Besides relying on video, anticipations can be made from static images. For example, Walker et al. [225] predict pixel-wise trajectories. For each pixel a prediction of how it will evolve in the future is conducted using an autoencoder. A similar idea

is pursued by Chao et al. [35]. Instead of dense pixel trajectories, they specialize entirely on anticipating pose dynamics. Similar to us, Vu et al. [223] predict distributions over plausible actions from images for which they collected the SUN Action dataset. While they predict general actions for whole scenes, we focus on more specific actions considering only individual objects. Fouhey and Zitnick [64] follow a single image setting, too, but they use abstract scene representations to learn what might happen next. Instead of predicting specific actions they consider the dynamics of objects. In the work of Qi et al. [168], interactions between humans and objects are studied in images as well as in videos. Scenes are parsed into a graph that indicates relations between objects. In one experiment, this graph is used to anticipate future activities on the CAD-120 dataset [113].

Psychology and the Concept of Affordances Action plausibility scoring is related to the concept of affordances coined by Gibson [70] and later refined by Gibson [69, Chapter 8]. While affordances indicate what interactions with the environment are possible for an agent, they do not come with any notion of preference. No differentiation about what action is more likely to happen takes place, physical compatibility is the only aspect that matters. Hence, affordances can be considered to be less-abstract than the plausibilities we propose in this paper. Affordances have been studied in various forms: for whole images [250], as poses [80], bounding boxes [239, 57], densely for every pixel [151, 173, 169, 137, 136] or from video [113, 229]. However, existing research is not limited to discovering action possibilities: Mechanisms that drive the selection of actions have been investigated in neuroscience [7] including the creation of computational models [42, 185].

Note that the concept of affordances centers strongly on the objects, essentially asking: which actions are suggested by different objects? Agents, humans or robots, however many times are rather plan-driven and they ask this question the other way round: which object can I use for a planned action? To better accommodate both types of queries, recently the concept of Object-Action Complexes (OACs) had been introduced [234, 119] that assumes that objects and (planned) actions are inseparably intertwined. Our current study takes this one step further stating that objects *with certain properties* and actions are intertwined. For example *full* cups are for drinking, *dirty* cups for cleaning, etc.

6.3 THE PLAUSIBLACT DATASET

In this section, we introduce the PlausiblAct dataset which associates images with a probability distribution over a set of ten actions. We explain the design of the dataset from the selection of actions via collecting data to generating probability distributions from the gathered anno-

tations. The images of PlausiblAct come from the OpenImages dataset [115], which contains scenes (images) showing multiple objects with corresponding bounding boxes. For our dataset, we extract individual objects and denote them as instances.

6.3.1 Choice of Actions and Ratings

In contrast to object names, it is more challenging to assign actions. Actions are to some degree subjective, depend on a state (e.g. hungry, tired) or on past actions. Therefore, a key challenge in this work is to constrain the setting in such a way that actions become less subjective. To this end, we focus on actions that tend to be *unconditional*. This involves actions the utility of which immediately pops up when a scene is perceived without depending on the state of the observer. We say "tend to" because even under these considerations the here-chosen actions remain *somewhat* conditioned on the state but to a smaller extent than many others. Specifically actions which are either plan-driven (e.g. to hammer a nail to fix something) or mood-driven (e.g. watch TV, read a book) are excluded. In such cases we would not expect the actions to be reliably rate-able as raters might assume different states leading to inconsistent ratings. We identify a set of ten actions $\mathcal{A}$ that is compatible with these principles. They are presented in Fig. 6.2. In addition to actions, we need to define possible ratings for an action instance. In order to reduce the cognitive load for the raters we follow a simple approach and use only three possible ratings $\mathcal{R} = \{\text{impossible}, \text{implausible}, \text{plausible}\}$. While impossible refers to the physical layout of a scene, plausibility decisions often depend on the context within an image.

For each action of these ten actions, we manually enumerate the complete subset of compatible object classes from all 600 object classes in OpenImages [115] (see appendix). Compatible means that, based on the object class name, it is potentially possible to conduct the action on an object of this class. E.g. a glass is potentially compatible with the action drinking (but not always, as it can be empty). We will implicitly assume that incompatible object-action pairs (as specified by the table in the appendix) are implicitly rated as impossible. For instance, let us assume there were only the actions *eat* and *sit on* and the object *cake*. Then defining the set of eat-able objects to be {*cake*} implies that the cake is never sit-able.

6.3.2 Scene and Instance Selection

Having defined compatibility between actions and objects, the next step is to select *good* scenes from the set of remaining scenes. Note that people do not take photos randomly. They rather focus on beautiful and tasty things. E.g. food is most often photographed before and not during eating. This leads to the fact that image databases are really representative illustrations of reality but collections of cherry-picked moments. However, to generate reasonable action plausibilities

we need a good coverage of all situations. In the following we introduce mechanisms that counteract these biases.

First, scenes are excluded when one of these criteria is met:

- Small coverage (less than 2% of all pixels), as the crop would not be recognizable.
- Large coverage (more than 70% of all pixels), as there would be little room for context.
- Image with humans as this would often require the rater (and later-on also the system) to infer intention, which we consider beyond the scope of this paper.

Furthermore, we maintain only one bounding box if two bounding boxes overlap with an intersection over union of over 0.5. Then we use the one for the less frequent class. Lastly, we manually remove scenes showing humans that were not considered by the labels and hence slipped through our previous filtering mechanism. Additionally, product photos and images having poor quality are removed.

Lastly, we put an upper limit on the number of occurrences of each object class. To prefer larger objects we sort all instances descending by size and then select the first 1000 instances of each object class which increases the variety of the included object classes.

6.3.3 Collection of Annotations

Annotations are gathered using a web-based interface. After receiving instructions and being shown example ratings, raters could explore a large number of instances for each action. The order of instances is shuffled individually for each rater. Instances to be rated (with impossible, implausible or plausible) can be freely chosen by the users.

Rater instructions All raters received explicit instructions. Pilot experiments suggest that these are critical for obtaining a reasonable inter-rater reliability as the annotation of actions can be highly ambiguous. Following our observations from the pilot experiments, we instructed raters to follow three principles. These are the original instructions presented to the raters:

- **Optimism about the Unseen**: If you are uncertain about some unseen aspects of the scene, please assume the most favorable situation for the given action.
- **Immediate Acting**: Consider the plausibility of conducting the action without delay. Do not assume that the action execution could wait.

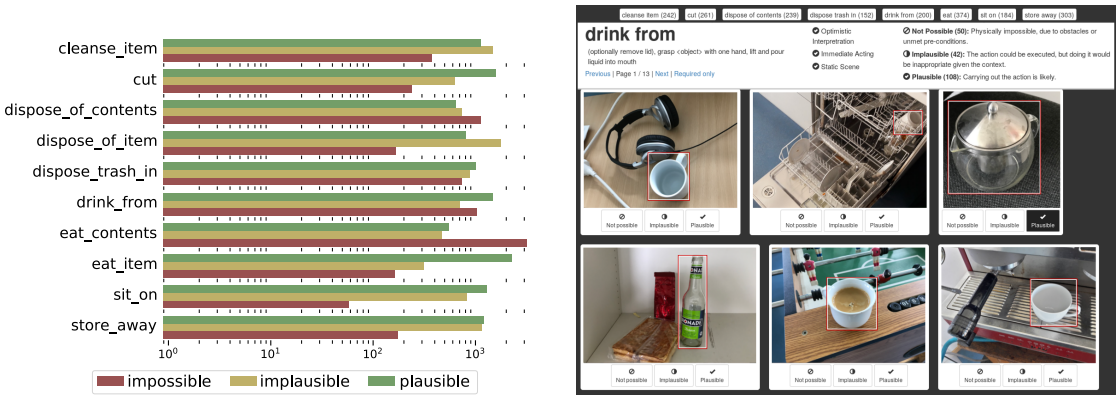

Figure 6.2: Left: Frequencies of the ratings for each action (note the logarithmic scale). Right: Screen-shot of the annotation tool.

- **Static Scene**: Do not assume changes to the scene that make the action possible that go beyond the definition of the action. Only consider the presented situation and pay attention to the action definition.

In addition, we showed to the raters eleven examples of how these principles are supposed to be interpreted.

Consider rating a scene involving an opaque bottle on a table regarding the action *drink*. The principles above mean that the action should be rated by assuming that the bottle contains drinkable liquid (optimism), the table layout cannot be changed (static scene) and we cannot conduct other actions before drinking (like filling the bottle first).

While we first experimented with a sequential design, where only one image at a time is presented to the rater, we finally decided to employ a multi-image paradigm. For a given action, multiple scenes are presented and the user can freely select, which instances to annotate. This allows for faster and more reliable annotations as hard, unclear samples can be skipped. Furthermore, this paradigm allows us to ask the raters to provide a minimal amount of ratings for the categories implausible and plausible, which results in a more balanced dataset. The web-based tool is shown in Fig. 6.2 (right). We discuss inter-rater reliability in Section 6.4.2.4, after the explanation of the metrics used in this work. We use the split in training, validation and test data defined by OpenImages [115]. For the training data, we allow choosing annotations freely as described above. As a consequence, the training procedure has to deal incompletely annotated instances. For creating the ground truth of the test data, we requested the raters to label instances completely (i.e. all compatible actions must be rated), which enables computing meaningful metrics on the test set. For this, indicators of missing instances are shown in the web-based interface to prompt the rater to complete it.

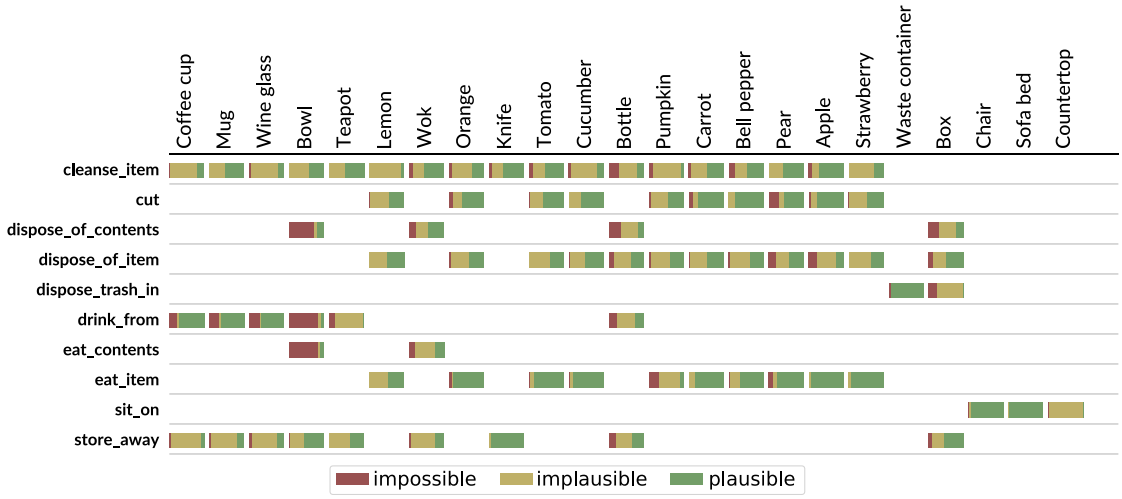

Figure 6.3: Rating distributions for all actions and frequent objects. Note, ratings are often not uniformly distributed for objects. Incompatible combinations of action and object are left blank.

Annotation statistics In Fig. 6.2 (left) and Fig. 6.3 we present distributions of the user-provided ratings for all actions and selected objects. In total, eight raters provided 28,046 ratings on 18,837 instances. Impossible was chosen 7,219, implausible 8,922 and possible 11,905 times.

6.3.4 From Annotations to Plausibilities

Having collected a set of annotations, we need to transform it to trainable data. Each instance may have received ratings for some actions from one or more raters.

The key idea is to train the network to match the plausibility distribution of the raters for each instance. Not every instance suggests clear actions and often multiple ratings seem plausible. By modeling the ground truth as a distribution over ratings we can incorporate a notion of uncertainty. This approach is different from image classification, where the ground truth distribution accumulates all mass on a single label. In our case this happens only if all raters agree. Moreover, we predict 10 actions per image simultaneously.

Formally, for every instance $i \in \mathcal{I}$ (i.e. an object in an image) we aggregate all associated ratings into a matrix $\mathbf{R}^{(i)} \in \mathbb{N}^{|\mathcal{A}| \times 3}$. Each element $\mathbf{R}^{(i)}_{a,r}$ denotes the count of ratings r for action a. In addition, a mask $\mathbf{v}^{(i)} \in \{0,1\}^{|\mathcal{A}|}$ is computed that indicates which rows (actions) of $\mathbf{R}^{(i)}$ are valid for an instance. This is necessary because in the training set annotations can be incomplete. Since raters can freely choose which instances to annotate, there is no guarantee that for a given instance all possible actions are actually rated. The values of unrated yet compatible actions in $\mathbf{R}^{(i)}$ are not informative and therefore must be excluded from the computation of the loss. Thus,

later, we will use $\mathbf{v}^{(i)}$ to exclude undefined actions from being considered in the loss. Next, the ground truth plausibility matrix $\mathbf{P}^{(i)}$ is generated from $\mathbf{R}^{(i)}$.

$$\mathbf{P}_a^{(i)} = \begin{cases} \dfrac{\mathbf{R}_a^{(i)}}{\sum_r \mathbf{R}_{a,r}^{(i)}} & a \text{ is compatible with instance } i \\ [1,0,0] & \text{otherwise} \end{cases} \tag{6.1}$$

Here the vector $[1,0,0]$ is used to assign the rating *impossible* to all incompatible actions (as described above).

6.3.5 Loss

Given an image $\mathbf{I}$, the network f predicts a matrix that assigns a probability to each rating for all actions. The rating probabilities for an action must sum to one. The loss is calculated by the cross entropy CE between each action's predicted rating distribution and the actual distribution obtained from the raters, denoted by $\mathbf{P}^{(i)}$.

$$\mathcal{L}^{(i)} = \frac{1}{\sum_{a \in \mathcal{A}} \mathbf{v}_a^{(i)}} \sum_{a \in \mathcal{A}} \text{CE}(f(\mathbf{I}^{(i)})_a, \mathbf{P}_a^{(i)}) \mathbf{v}_a^{(i)}$$

In case an action is required but not provided the value of $\mathbf{P}$ is invalid and should not contribute to the error expressed by the loss. This is realized by using the validity mask $\mathbf{v}^{(i)}$.

Data Augmentation Since we have to cope with limited training data, we apply different forms of data augmentation. This involves random cropping, adding Gaussian blur, changing gamma and colors of the image. We control the strength of these operations with a single integer value. The optimal value of this is determined experimentally (see Table 6.2).

Implementation We employ batch normalization [98] and early stopping after 7 epochs without improvement of the validation loss. Weight updates are carried out with ADAM [110]. The code is implemented based on the PyTorch [158] framework.

6.3.6 Models

We use state-of-the-art convolutional neural networks architectures that have proven to work well for image recognition tasks. These include different variations of ResNet [88] and InceptionV4 [202]. Instead of training from scratch, we initialize the networks weights from pre-training on ImageNet [53] unless otherwise stated.

setting	1st input	2nd input
ignore	black image	-
img+mask	instance image	context image with instance being masked
img+full	instance image	context image (no masking)
only-masked	context image with instance being masked	-
only-full	context image (no masking)	-

Table 6.1: Input data for the different context settings

6.3.7 BASELINES

We start our analysis by introducing two baselines:

- The **mode baseline** always predicts the most common rating for the depicted object. This is somewhat unfair since the baseline uses object labels other models do not have. However, it provides us with insights about how strongly the prediction of an action is tied to the underlying object class.
- The **ignore image baseline** is identical to a normal model but does not receive any image as input. Hence, the only way it can minimize loss is to learn the dataset distribution. This baseline provides us with a reference to relate other scores with. If a model does not perform better than this baseline it has not learned anything but the biases present in the dataset.

6.3.8 CONTEXT REPRESENTATIONS

As stated above, instances are objects that are part of larger scenes. Hence, it might be useful to make the entire scene accessible to the model. For incorporation of this kind of context, we differentiate between multiple ways, which we describe in the following. Context representations that involve a "+" imply two image inputs (instance image + some context) to the model and thus require two separate image encoder networks.

- The trivial case *ignore* means ignoring the context entirely and considering only the instance's object.
- In the *img+masked* setting, we mask the object bounding box with a black rectangle. Hence, the network has no access to the object's visual features but has to rely only on contextual cues. Additionally, as a second input, the instance image is shown, too.
- In the *img+full* setting, the entire context is shown (without masking the instance) and the instance image is provided as a second input.

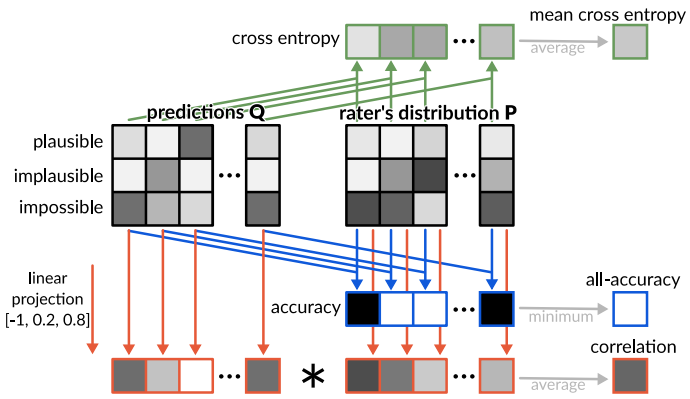

Figure 6.4: Illustration of how metrics are computed for one instance. Predictions of the network and ratings of the annotators are collected in two matrices $\mathbf{Q}$ and $\mathbf{P}$. By comparing the most likely ratings for each, the accuracy is obtained.

- In the *only-masked* setting, the entire context with the instance being masked is shown.
- In the *only-full* setting, the entire context is shown.

In Fig. 6.1 we provide an overview of the different input data types in the context settings.

6.3.9 METRICS

Obtaining quantitative scores for performance is a challenging task because the model's predictions and the ground truth are proper probability distributions. This is different from image classification where the ground truth distribution has only one non-zero element. Furthermore, for each instance, all ten actions are predicted simultaneously. For calculating performance metrics, we compare the ground truth $\mathbf{P}^{(i)}$ with $\mathbf{Q}^{(i)}_{a,r}$, which represents the network's predictions for action a and plausibility rating r of an instance i. The rating distribution sums to one, i.e. $\sum_{r \in \mathcal{R}} \mathbf{Q}^{(i)}_{a,r} = 1$.

All-action Accuracy (Acc) A straightforward choice to assess how well the predictions of a model are aligned with user annotated ratings is accuracy. If the highest mass rating is identical for prediction and ground truth an instance is considered to be classified correctly. We consider accuracy in two settings: Independently for each action as described above and for all actions of an instance. In the latter case, successful classification requires the correct prediction of all actions. The disadvantage of accuracy is its sensitive to the maximum. The actual distribution

of the ratings apart is ignored.

$$\text{Acc}_a(\mathbf{P}^{(i)}, \mathbf{Q}^{(i)}) = \begin{cases} 1 & \text{if } \arg\max(\mathbf{P}_a^{(i)}) = \arg\max(\mathbf{Q}_a^{(i)}) \\ 0 & \text{otherwise} \end{cases}$$

Since we require all actions to be correctly annotated, we apply a $\min$ function on all action-wise accuracies. By averaging over all actions we obtain the single Acc score:

$$\text{Acc} = \frac{1}{|\mathcal{I}|} \sum_{i \in \mathcal{I}} \min_{a \in \mathcal{A}} \text{Acc}_a(\mathbf{P}^{(i)}, \mathbf{Q}^{(i)})$$

Accuracy is easy to interpret but fails to represent the whole plausibility distribution. This means, a close to uniform distribution with most mass on x is treated equally as a low-entropy distribution that accumulates all mass on a single x.

Cross Entropy (CE) Since we need to compare probability distributions, we can make use of divergence measures, which express how similar probability distributions are. While many of such measures exist, a natural choice is to use cross entropy that is also used to train the network. We compute cross entropy for each action by:

$$\text{CE}_a(\mathbf{P}^{(i)}, \mathbf{Q}^{(i)}) = -\sum_{r \in \mathcal{R}} \mathbf{P}_{a,r}^{(i)} \log \mathbf{Q}_{a,r}^{(i)}$$

Then a single score is obtained by averaging individual cross entropies CE_ay over all actions:

$$\text{CE} = \frac{1}{|\mathcal{I}|} \frac{1}{|\mathcal{A}|} \sum_{i \in \mathcal{I}} \sum_{a \in \mathcal{A}} \text{CE}_a(\mathbf{P}^{(i)}, \mathbf{Q}^{(i)})$$

A small cross entropy indicates high similarity between prediction and ground truth and is therefore desirable. In contrast to accuracy, CE is not intuitively interpretable (how good is a CE of say 0.2?) but it captures differences in the non-maximum parts of the distributions. Comparison is enabled by considering the CE of one setting relative to others.

Correlation (Corr) The annotated data is ordinal, i.e. there exists an order from impossible over implausible to plausible. By defining a distance between the three ratings we can transform a plausibility distribution to a continuous, scalar value. This is done by a linear projection with a fixed vector $\mathbf{l} = [-1, 0.2, 0.8]$ that expresses the distances between the ordinal values. For correlation, we do not compute action-wise scores but consider instances and actions jointly. Let the index j iterate over instances as well as actions (hence the mappings $i(j)$ and $a(j)$), then scores for an action can be computed by: $\mathbf{r}^{(j)} = \max(0, \langle \mathbf{l}, \mathbf{P}_{a(j)}^{i(j)} \rangle)$ and $\mathbf{q}^{(j)} = \max(0, \langle \mathbf{l}, \mathbf{Q}_{a(j)}^{i(j)} \rangle)$

Now that predictions and ground truths are mapped to a sequence of scalars, we can access the quality of the model's predictions by employing Pearson's correlation coefficient. The resulting score indicates to which degree predicted and ground truth scores are linearly related.

We consider this a good measure as it is normalized between -1 and 1 and the top score of 1 or 100% is only attained if scores are identical, except for a scaling factor. In practice, if scores are normalized, the scaling factor becomes irrelevant. The correlation coefficient is defined as follows:

$$\text{Corr} = \frac{\sum_j (\mathbf{q}^{(j)} - \bar{\mathbf{q}}) * (\mathbf{r}^{(j)} - \bar{\mathbf{r}})}{\sqrt{\sum_j (\mathbf{q}^{(j)} - \bar{\mathbf{q}})^2} \sqrt{\sum_i (\mathbf{r}^{(j)} - \bar{\mathbf{r}})^2}}$$

A problem of the correlation score is that it requires variance to be computable. If all predictions (or all ground truth scores) are identical, the term $(\mathbf{r}^{(j)} - \bar{\mathbf{r}})$ is zero and causes division by zero. In fact, this case rarely occurs in our experiments, we indicate it by "-". The correlation coefficient is both easy to interpret and captures differences across distributions. However, one might argue that the projection vector is somewhat arbitrary.

6.4 Experiments

Next we conduct a series of experiments assessing the quality of the trained networks and relating them to meaningful baselines. First, we show some qualitative results, involving both instance only and context. Quantitatively, we analyze performance concerning context, architecture and training settings using the metrics defined above.

6.4.1 Qualitative Evaluation

In Fig. 6.5 we present a set of images with their associated action plausibilities computed using the single-image InceptionV4-based model as well as the 2xRN50 model which uses the instance image in conjunction with full context. Note the variety of sample images, ranging from an outdoor cherry tree to different cup close-ups having vastly different illuminations.

The presented samples indicate that the trained model generates useful predictions of the plausibilities of the actions on these unseen samples. We observe that the plausibilities are strongly dependent on the object class. However, this is not true in all cases. For example the plausibility for drinking is zero for the empty cup while it is the most likely action fpr the filled cup. Additionally, while the object class often seems to determine the presence of plausible actions, there are fine-grained differences in the individual plausibilities. These differences represent a crucial aspect of the visual common-sense knowledge about household scenes that has been learned. In a robotic context, such differences could be used to compare plausibilities of a given action across multiple objects and then pick the most suitable object.

The qualitative samples that involve context suggest that the context has an inhibitory effect on action plausibilities. The predicted plausibilities tend to be smaller. Especially in the

Training settings / Augmentation

mask	MR	SR	PT	aug	A-Acc	CE	Corr
✓	-	-	✓	2	47.4	0.200	74.2
✓	2	-	✓	2	39.0	0.255	65.0
✓	2	✓	✓	2	37.9	0.275	59.8
✓	-	-	-	2	29.0	0.295	53.7
-	-	-	✓	2	36.9	0.301	42.9
✓	-	-	✓	0	45.3	0.212	73.4
✓	-	-	✓	4	43.9	0.213	71.1
✓	-	-	✓	6	46.3	0.224	71.6

Table 6.2: Ablation of different training settings (top) and augmentation strengths (bottom). PT: pre-trained, SR: same ratings only, MR: minimal number of ratings

Encoders

model	A-Acc	CE	Corr
SqueezeNet	40.0	0.212	66.8
RN18	40.4	0.193	70.7
RN50	42.1	0.191	71.6
RN101	44.4	0.216	68.5
RN152	44.4	0.219	70.0
Xception	44.9	0.208	70.1
Inc3	40.9	0.244	63.9
Inc4	47.4	0.200	74.2

Table 6.3: Comparison of different encoders.

bottom row that involves the same object on different backgrounds we observe much higher plausibilities in case of a uniform white background compared to the real world background.

6.4.2 Quantitative Evaluation

Based on the previously defined baselines and metrics, we begin our analysis by comparing various training settings, augmentation strengths, and encoder architectures. In subsequent experiments we address special questions investigating how many samples are sufficient, the role of context, the impact of the encoder architecture and several design choices as part of an ablation. Additionally, human performance using the same metrics is assessed and related to the computational models.

6.4.2.1 Ablation

Training Setting and Augmentation First we assess the impact of several training parameters, introduced above, on the performance. The corresponding results are reported in Table 6.2. MR refers to the minimal number of ratings required for a sample. While this is per default 1, in case of MR $= 2$ the dataset size is reduced but samples are more reliable. Same rating (SR) means that samples are only accepted when the raters agree (which only makes sense for MR > 1). Moreover, we find that both, pre-training on ImageNet and masking the loss are crucial for performance. In both cases, performance decreases compared to single rater samples. This suggest that the increased variance introduced by a large dataset weighs more than the increased reliability of multiple ratings per instance. In augmentation we find a moderate strength of 2 to perform best.

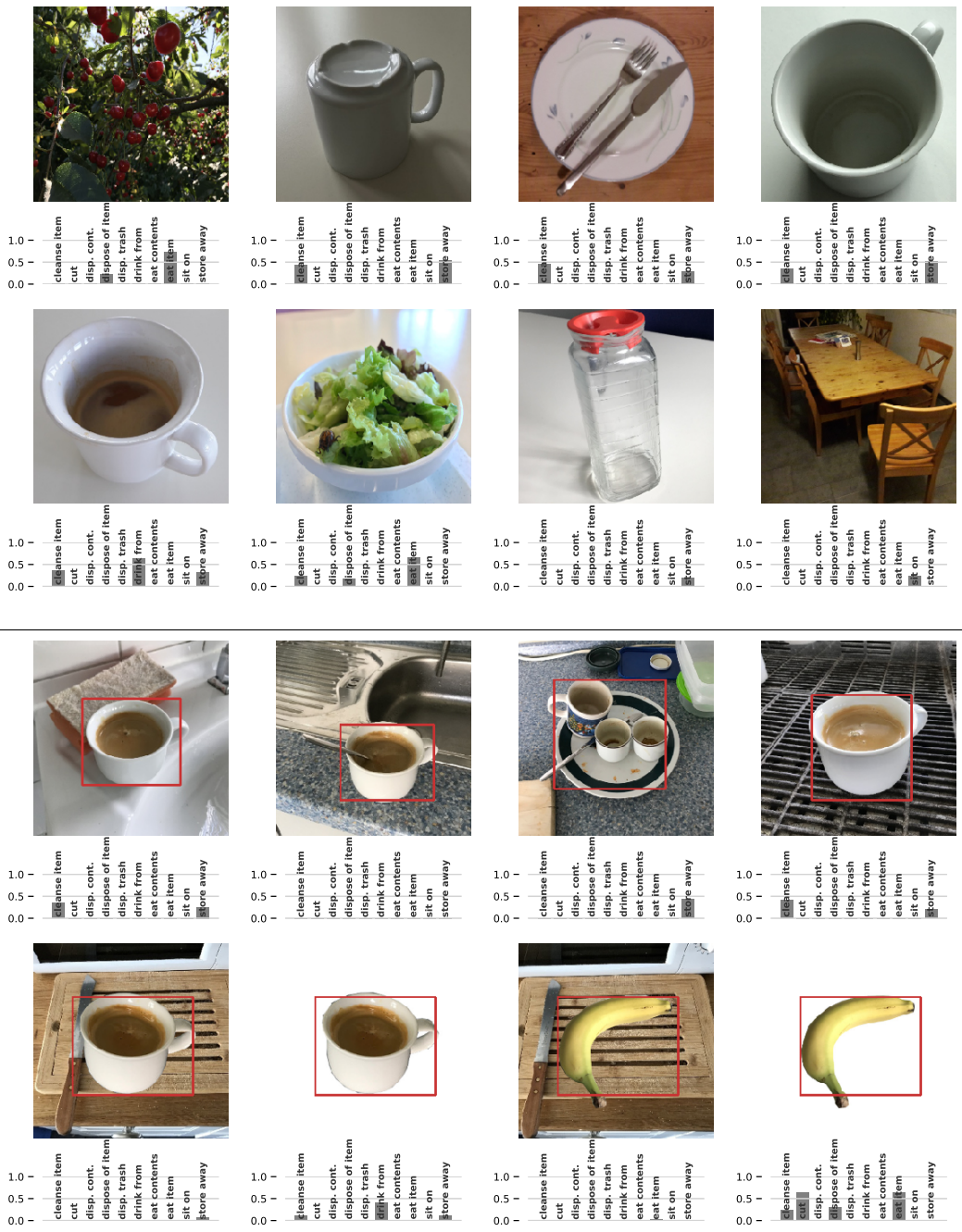

Figure 6.5: Top two rows: Qualitative samples generated using the InceptionV4-based network. Bottom row: Samples generated using full context of the 2xRN50 network (the instance image is indicated in red)

Table 6.4: Evaluation per action for selected models (right) as well as comparison to baseline performances (left). The mode baseline does not receive the image as an input but has access to the name of the object shown. "No inp." refers to a RN50 network where all input information is removed by multiplying with zero.

model	context	A-Acc	σ (A-Acc)	CE	σ (CE)	Corr	σ (Corr)	cleanse-item	cut	dispose-of-contents	dispose-of-item	dispose-trash-in	drink-from	eat-contents	eat-item	sit-on	store-away
No Inp.	ignore	7.0	0.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Mode	ignore	50.7	0.0	0.350	0.000	75.9	0.0	59.9	88.3	16.8	71.6	86.8	65.5	-	96.7	94.9	73.0
Inc4	ignore	45.9	1.6	0.202	0.007	73.6	1.3	53.2	84.8	18.6	70.6	84.3	77.0	7.9	94.4	91.1	62.6
RN50	ignore	43.5	2.6	0.214	0.005	70.6	1.7	50.9	77.8	19.1	68.8	80.1	71.2	9.3	91.0	88.2	62.8
2xRN50	full	44.6	1.5	0.208	0.007	70.4	2.1	52.0	78.7	20.1	68.3	82.4	65.0	-	91.6	88.0	63.1

Encoder The comparison of different encoder architectures, presented in Table 6.3, indicates that larger models tend to perform better. We attribute this to two reasons: First, they can capture more complex features. Second, their object detection performance is better. Given reliable object detection, it is easier to exploit dataset biases. For a more detailed discussion of this we refer to Sec. 6.4.2.3.

Besides the shown experiments, we found the batch size to play a critical role for performance and thus suggest to keep the batch size as large as possible. Additionally, we tried to use larger images to improve performance without success. We hypothesize that the reason for this is that models strongly benefit from the pre-trained ImageNet weights. This pre-training was done for a fixed image size and the ImageNet dataset is fairly consistent with respect to scale. Hence, the features encoded by the weights are optimized for this specific size. Possibly, our dataset is too small to cause substantial changes in the features and hence it benefits from objects being provided at the original scale.

6.4.2.2 Action-wise Evaluation and Comparison to Baselines

The results shown Table 6.4 show improvement over the ignore-image baseline. This means the models indeed use information from the image to improve predictions. In fact, the ignore-image baseline considers all actions implausible, which is the best guess without knowing the image. However, the mode baseline outperforms our methods in Acc and Corr while our method achieves better CE. This means that our method has advantages at predicting fine-grained differences in the rating distribution, while for coarse accuracy that neglects details in

the distribution, the mode baseline is good enough.

The class-wise scores give more insights. For most action classes, our methods yield a worse accuracy (Acc) than the mode baseline. However, drink-from is a notable exception as it performs much better than mode. This suggests that for drink-from, the image content is crucial and must be considered to make a decision.

Considering the good performance of the mode baseline it should be noted that it benefits from several factors: First, it knows the object class being depicted, an information that other models have no access to. Evidence for the importance of this is found in the gap between ImageNet pre-trained and untrained models in the ablation (Table 6.2). Second, it knows the modes of the rating distributions. Since these distribution are far from being uniform, the mode alone often is a powerful predictor for the most likely rating. This means that the mode baseline has an unfair advantage over our method: In practice the information about the object class being shown is obviously not available as it would require a perfect object recognizer. Plugging in a sub-optimal object recognizer would diminish the performance. Nonetheless, the mode baseline serves as a useful anchor to relate scores to.

When we consider all ratings in the CE metric, the mode baseline does not perform as good anymore. To some extent this is not surprising because the mode baseline always generates one-hot distributions. Still these fine-grained differences in plausibilities are crucial for many applications in robotics since they enable the comparison and selection across different potential actions.

6.4.2.3 Selected Objects and Raters

In many cases the rating distribution is highly dependent on the object class, i.e. given the object class we can make the correct prediction without having looking at the image. While this is just a natural phenomenon, it interferes with our analysis since we are particularly interested in cases where the image content matters. Hence, we conduct an analysis with a subset of objects whose plausibility rating distribution has a higher entropy. Concretely, these object classes are: bottle, bowl, wok and box. The corresponding results are shown in Table 6.5. We see that the mode baseline is strongly outperformed in terms of Acc and slightly outperformed on CE. This indicates that the good performance of the mode baseline is an artifact of unbalanced rating distributions.

Similar to picking specific object classes, we can also limit the training data to specific raters. For this we select a subset of 3 raters having an average pairwise agreement of 73.4. When we use this set for training and test we obtain the scores reported in Table 6.6. Here we see substantially better performance in terms of CE. In addition, the gap between mode baseline

Table 6.5: Performance on four selected objects: bottle, bowl, wok and box. PT means pre-training.

Selected Objects						
model	ign.	context	PT	A-Acc	CE	Corr
Mode	-	ignore	-	3.0	0.400	62.2
RN50	✓	ignore	✓	0.0	-	43.6
Inc4	-	ignore	✓	12.1	0.352	62.2
RN50	-	ignore	✓	15.2	0.412	57.0
2xRN50	-	img+full	✓	18.2	0.356	57.7

Table 6.6: Performance on three selected raters having high agreement.

Selected Raters						
model	context	ign.	PT	A-Acc	CE	Corr
Mode	ignore	-	-	67.1	0.337	83.0
RN50	ignore	✓	✓	1.8	1.931	-
Inc4	ignore	-	✓	55.7	0.147	83.2
RN50	ignore	-	✓	58.7	0.179	73.8
2xRN50	img+full	-	✓	56.3	0.160	77.6

and our methods is larger. From these results we conclude that consistency of training and test data is a crucial property.

6.4.2.4 Rater Reliability

Having only compared scores obtained from different computational methods so far, a natural question is: How consistent are the ratings provided by humans? For this, we apply the metrics introduced above on pairs of human raters. By averaging all pairwise scores we obtain the following: Acc of 42.0, CE of 0.347 and a Corr of 44.8. While Acc is comparable to some models, in terms of CE and Corr the raters perform significantly worse than the computational methods. If we require a minimal intersection of 100 instances to compensate for statistically unreliable data points, we obtain slightly better scores.

We also tracked the self-consistency of the raters by presenting selected instances twice within the collection of all instances. Since the raters were free to select which samples they annotate, not all of them annotated these instances. However, across those who did, the self-consistency varies between 0.77 and 1.0 with an average of 0.90. The number of samples that were annotated twice ranges from 1 to 26 with an average of 13.1.

6.4.2.5 Scalability

The number of training samples is a quantity that normally has a strong impact on the performance. Since we are collecting the data, it is crucial to understand the effect of the training sample size to avoid an insufficiently small dataset. Fig. 6.6 provides an overview on the relationship between training samples and performance. It suggests that the dataset is large enough and no major improvements could be expected from gathering more data. Surprisingly, we find that already a fairly small amount of annotated scenes allows models to attain a high correlation with the ground truth probabilities.

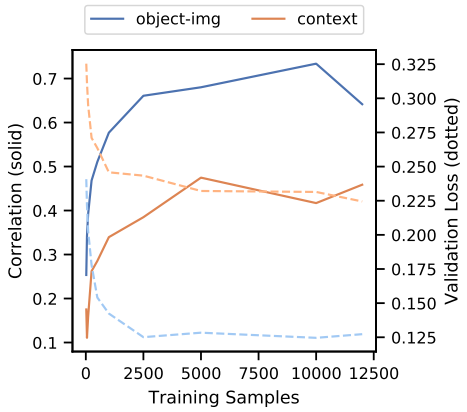

Figure 6.6: Performance for different numbers of training samples.

Table 6.7: Comparison of different context representations. BS means batch size.

model	context	A-Acc	CE	Corr
RN50	ignore	42.8	0.212	68.3
Inc4	ignore	43.2	0.212	70.5
RN50	only	31.3	0.302	48.3
RN50	only-full	35.5	0.252	65.1
2xRN50	full	42.1	0.207	71.1
2xRN50	masked	44.4	0.208	72.3

6.4.2.6 Context

Not only the appearance of an object is relevant for actions, potentially also the context can give hints about the status of an object. Having introduced context representations in Sec. 6.3.8, here we run an explicit comparison of the representations.

From Tab. 6.7 we observe that context with the instance object being masked (only-masked) helps to predict actions but does not achieve the performance of showing the object itself (ignore context). When the instance image is combined with a context representation, the Corr scores slightly improve compared with RN50 ignoring context. However, this improvement is fairly small. This is probably due to small parts of the context being included in the image itself. The information that can be extracted from a bigger context is therefore negligible and does not outweigh the problems of having more parameters. Relying exclusively on the context does not seem to be a good idea. This is not surprising because the object appearance clearly gives hints about possible actions.

6.5 Conclusion

In this paper, we established a framework of how to gather action plausibility ratings, creating a dataset called "PlausiblAct", transform them to train neural networks and evaluate the corresponding results. After defining a set of ten actions and three ratings, we presented our sparse data collection method relying on web techniques allowing for a fast and comparatively effortless data annotation. Next these ratings of object instances are transformed into distributions on which a neural network can be trained to make action-oriented predictions. To assess

the quality of the predictions we proposed three metrics capturing complementary aspects of the predictions. In our comparison of state-of-the-art feature encoders, we find the InceptionV4 network to be suited best for the task. The experiments suggest that object-classification performance is still a crucial factor for scoring action plausibilities. Combinations with context seem to improve the performance slightly, while context alone ignoring the actual objects' appearances performs quite badly.

We believe such systems are useful in robotics because they allow the comparison and selection of actions in various settings. An advantage of the proposed method is that it can be combined with other robotic algorithms. For example, assume a scene involving a dirty and a clean cup and the instruction "put cup to dishwasher". Although it is obvious to humans that the instruction is refers to the dirty cup, this common-sense knowledge is not available to the robot. By using our method, the system can evaluate images of both cups and then pick the one for which the action "cleanse" is more plausible. Thus, potential applications, where we expect action plausibilities to be helpful, concern robotic action planning, where our method allows better disentangling action preconditions needed in the planning operators.

The presented approach has some limitations. So far, the models we employed are simple image classification models that are not specifically designed for reasoning. Furthermore, the inter-rater reliability is far from being optimal. Hence, future work might involve reasoning-oriented models, e.g. the relation network [179] and a more constrained annotation setting to obtain more consistent and reliable action ratings. So far, we excluded images depicting humans from the data as far as possible. As a potential next step, showing humans could increase the complexity of this or similar approaches as intentions would need to be estimated.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Community's H2020 Programme ICT-2016-2017/H2020-ICT-2016-1 under grant agreement no. 73761, IMAGINE.

6.6 REFERENCES

We maintain a single list of references at the end of the thesis.

APPENDIX

	drink from	cut	sit on	eat item	eat contents	cleanse item	store away	dispose of item	dispose of contents	dispose trash in
Toothbrush	-	-	-	-	-	✓	-	✓	-	-
Apple	-	✓	-	✓	-	✓	-	✓	-	-
Chopsticks	-	-	-	-	-	✓	✓	-	-	-
Croissant	-	✓	-	✓	-	-	-	✓	-	-
Cucumber	-	✓	-	✓	-	✓	-	✓	-	-
Radish	-	✓	-	✓	-	✓	-	✓	-	-
Hot dog	-	✓	-	✓	-	-	-	✓	-	-
Waffle	-	✓	-	✓	-	-	-	✓	-	-
Pancake	-	✓	-	✓	-	-	-	✓	-	-
Pretzel	-	✓	-	✓	-	-	-	✓	-	-
Bagel	-	✓	-	✓	-	-	-	✓	-	-
Teapot	✓	-	-	-	-	✓	✓	-	-	-
Popcorn	-	-	-	✓	-	-	-	✓	-	-
Burrito	-	✓	-	✓	-	-	-	✓	-	-
Scissors	-	-	-	-	-	-	✓	-	-	-
Chair	-	-	✓	-	-	-	-	-	-	-
Muffin	-	✓	-	✓	-	-	-	✓	-	-
Cookie	-	✓	-	✓	-	-	-	✓	-	-
Calculator	-	-	-	-	-	-	✓	-	-	-
Box	-	-	-	-	-	-	✓	✓	✓	✓
Stapler	-	-	-	-	-	-	✓	-	-	-
Studio couch	-	-	✓	-	-	-	-	-	-	-
Zucchini	-	✓	-	✓	-	✓	-	✓	-	-
Ladle	✓	-	-	-	-	✓	✓	-	-	-
Winter melon	-	✓	-	✓	-	✓	-	✓	-	-
Spatula	-	-	-	-	-	✓	✓	-	-	-
Pencil sharpener	-	-	-	-	-	-	✓	-	-	-
Eraser	-	-	-	-	-	-	✓	-	-	-
Tin can	✓	-	-	-	-	-	✓	✓	✓	-
Mug	✓	-	-	-	-	✓	✓	-	-	-
Can opener	-	-	-	-	-	✓	✓	-	-	-
Coffee cup	✓	-	-	-	-	✓	✓	-	-	-
Cutting board	-	-	-	-	-	✓	✓	-	-	-
Vase	-	-	-	-	-	✓	✓	-	-	-
Slow cooker	-	-	-	-	-	✓	✓	-	✓	-
Whisk	-	-	-	-	-	✓	✓	-	-	-
Salt and pepper shakers	-	-	-	-	-	✓	✓	-	-	-

French fries	-	✓	-	✓	-	-	-	✓	-	-
Tart	-	✓	-	✓	-	-	-	✓	-	-
Egg	-	-	-	✓	-	-	-	✓	-	-
Grape	-	✓	-	✓	-	✓	-	✓	-	-
Mixing bowl	✓	-	-	-	-	✓	✓	-	-	-
Hammer	-	-	-	-	-	-	✓	-	-	-
Sofa bed	-	-	✓	-	-	-	-	-	-	-
Adhesive tape	-	-	-	-	-	-	✓	-	-	-
Saucer	-	-	-	-	✓	✓	✓	-	✓	-
Drinking straw	-	-	-	-	-	✓	✓	-	-	-
Common fig	-	✓	-	✓	-	✓	-	✓	-	-
Cocktail shaker	✓	-	-	-	-	✓	✓	-	-	-
Artichoke	-	✓	-	✓	-	✓	-	✓	-	-
Knife	-	-	-	-	-	✓	✓	-	-	-
Bottle	✓	-	-	-	-	✓	✓	✓	✓	-
Bottle opener	-	-	-	-	-	✓	✓	-	-	-
Bowl	✓	-	-	-	✓	✓	✓	-	✓	-
Frying pan	-	-	-	-	✓	✓	✓	-	✓	-
Ring binder	-	-	-	-	-	-	✓	-	-	-
Plate	-	-	-	-	✓	✓	✓	-	✓	-
Pitcher	✓	-	-	-	-	✓	✓	-	-	-
Pencil case	-	-	-	-	-	-	✓	-	-	-
Kitchen knife	-	-	-	-	-	✓	✓	-	-	-
Plastic bag	-	-	-	-	-	-	✓	✓	✓	✓
Potato	-	✓	-	✓	-	✓	-	✓	-	-
Pasta	-	-	-	✓	-	-	-	✓	-	-
Pumpkin	-	✓	-	✓	-	✓	-	✓	-	-
Pear	-	✓	-	✓	-	✓	-	✓	-	-
Infant bed	-	-	✓	-	-	-	-	-	-	-
Pizza	-	✓	-	✓	-	-	-	✓	-	-
Submarine sandwich	-	-	-	✓	-	-	-	✓	-	-
Loveseat	-	-	✓	-	-	-	-	-	-	-
Coffee table	-	-	✓	-	-	-	-	-	-	-
Taco	-	-	-	✓	-	-	-	✓	-	-
Strawberry	-	✓	-	✓	-	✓	-	✓	-	-
Tomato	-	✓	-	✓	-	✓	-	✓	-	-
Measuring cup	-	-	-	-	-	✓	✓	-	-	-
Paper cutter	-	-	-	-	-	-	✓	-	-	-
Wok	-	-	-	-	✓	✓	✓	-	✓	-
Jug	-	-	-	-	-	✓	✓	-	-	-
Pizza cutter	-	-	-	-	-	✓	✓	-	-	-
Bread	-	✓	-	✓	-	-	-	✓	-	-
Platter	-	-	-	-	-	✓	✓	-	-	-
Toilet paper	-	-	-	-	-	-	✓	-	-	-
Lemon	-	✓	-	✓	-	✓	-	✓	-	-
Banana	-	✓	-	✓	-	✓	-	✓	-	-
Wine glass	✓	-	-	-	-	✓	✓	-	-	-

Countertop	-	-	✓	-	-	-	-	-	-	-
Waste container	-	-	-	-	-	-	-	-	-	✓
Book	-	-	-	-	-	-	✓	-	-	-
Hamburger	-	-	-	✓	-	-	-	✓	-	-
Asparagus	-	✓	-	✓	-	✓	-	✓	-	-
Spoon	-	-	-	-	✓	✓	✓	-	✓	-
Oyster	-	-	-	✓	-	-	-	✓	-	-
Ice cream	-	-	-	✓	-	-	-	✓	-	-
Orange	-	✓	-	✓	-	✓	-	✓	-	-
Beaker	✓	-	-	-	-	-	-	✓	-	-
Peach	-	✓	-	✓	-	✓	-	✓	-	-
Fork	-	-	-	-	✓	✓	✓	-	✓	-
Cabbage	-	✓	-	✓	-	✓	-	✓	-	-
Carrot	-	✓	-	✓	-	✓	-	✓	-	-
Mango	-	✓	-	✓	-	✓	-	✓	-	-
Pineapple	-	✓	-	✓	-	✓	-	✓	-	-
Stool	-	-	✓	-	-	-	-	-	-	-
Envelope	-	-	-	-	-	-	✓	✓	-	-
Cake	-	-	-	✓	-	-	-	✓	-	-
Candy	-	-	-	✓	-	-	-	✓	-	-
Salad	-	✓	-	✓	-	✓	-	✓	-	-
Serving tray	-	-	-	-	-	✓	✓	-	-	-
Kitchen and dining room table	-	-	✓	-	-	-	-	-	-	-
Cake stand	-	-	-	-	-	✓	✓	-	-	-
Broccoli	-	✓	-	✓	-	✓	-	✓	-	-
Grapefruit	-	✓	-	✓	-	✓	-	✓	-	-
Bell pepper	-	✓	-	✓	-	✓	-	✓	-	-
Pomegranate	-	✓	-	✓	-	✓	-	✓	-	-
Doughnut	-	✓	-	✓	-	-	-	✓	-	-
Pen	-	-	-	-	-	-	✓	-	-	-
Watermelon	-	✓	-	✓	-	✓	-	✓	-	-
Cantaloupe	-	✓	-	✓	-	✓	-	✓	-	-

CHAPTER 7

Semantic Action Anticipation

In the previous chapter we have presented a method to tell plausible from implausible actions by considering an image. In a way, this can be seen as a form of anticipation. We reason about what might happen next. Which actions are more plausible to happen than others? This question often can be answered by analyzing static images for the specific set of actions we choose in the article above. However, in general a static image is not enough because it completely ignores which events have led to the depicted situation. Such a temporal context often is important to interpret scenes correctly. For example, from a static image it is hard or even impossible to tell whether a person puts an object into the fridge or takes out an object. Hence, we count temporal context among the essential components of action-oriented scene understanding (see Introduction). By considering the temporal context of a scene, we expect to be able to make more informative guesses about what might happen next.

Summary In the following article, we specifically address the temporal context. We study the task of semantic action anticipation which involves making a prediction about which actions will happen next, given a sequence of images. This requires carefully tracking events over some time to generate a reasonable prediction about the future. We dedicate a large analysis on understanding the processing of long sequences and the relation to feature extraction. In this case, features can be spatio-temporal and encode events. Specifically, for this investigation we design the synthetic SymbolSeq dataset which can be adapted to different settings answering different questions. Our experiments suggest that feature extraction in fact is the limiting factor for better performance. Current datasets do not seem to be large enough to allow for learning meaningful semantic features. This even holds if models are pre-trained on ImageNet.

Timo Lüddecke and Florentin Wörgötter

The Role of Features in Semantic Video Anticipation

(unpublished)

Abstract

Anticipation of upcoming actions in an image sequence (e.g. video) is of importance in a wide variety of applications, for example in surveillance or during human-robot interaction. However, action anticipation is difficult because the temporal context of events in the past must be considered to make reasonable predictions. This paper studies how well state-of-the-art video models capture such context and how temporal context is related to visual feature extraction. First, we introduce the parameterizable synthetic SymbolSeq dataset, which consists of sequences of simple symbols that transform – as simulated "actions" – into each other. Based on this dataset, we investigate the impact of distraction, sequence length, feature quality and pre-training on the performance in predicting these "actions" by state-of-the-art models. The latter involves both, two stage feature and extraction models as well as one-stage 3d-convolution-based models. Additional experiments on anticipation from natural images sequences are conducted using the SomethingSomething and EPIC datasets. For EPIC, visual input is also compared to a symbolic representation of the data. We find that performance crucially depends on the feature representation, in particular when the dataset contains few or noisy samples. In these cases, no rich features are learned, and pre-training becomes indispensable, which is the case for the natural image datasets included in our work.

7.1 Introduction

Deep neural networks that operate on video have gained popularity throughout the last years [105, 59]. However, most work addresses the recognition and description of actions, either globally for a video as a whole or for temporal segments within a video. In this work, we go a step further and study the problem of semantic video anticipation. Given a long video sequence, the goal is to generate a reasonable probability distribution over events that might happen next. Unlike in action recognition and video forecasting, a wider temporal window must be considered to make an anticipation. Potentially, all events in the past can influence what will happen next. Therefore, we carefully analyze how video models behave in the presence of long-term dependencies under consideration of the interplay between feature extraction and temporal sequence modeling. The latter is an aspect that is under-explored as common datasets provide pre-computed features, which evade backpropagating through the feature extractor.

Note, anticipation is a challenging, often ambiguous problem, so the accuracy of predictions is commonly smaller than in action recognition.

Beyond semantic video anticipation, aggregation of information from long video sequences and reasoning on it is a crucial ability for a wide range of applications: The plot of a movie can be understood, instructional video can be interpreted, and surveillance can be automatized. In each of these tasks single frames are not necessarily informative, while combinations of frames are. For instance, the order of the events is relevant for understanding a movie plot. However, this property is absent from many video datasets where single frames can be expressive enough to base classification on them [105, 195].

For our analysis, we introduce a synthetic dataset that helps us to study sequence models in conjunction with feature extraction. It involves sequences of images of moving symbols that transform into other symbols or stay the same. The visual quality of symbol presentation and presence of distractive cues can be controlled (among other factors). This allows for an inspection of the impact of individual factors that would be infeasible with natural image datasets. Due to the temporal dimension, video modeling is more resource demanding than image recognition, both in terms of compute and memory. Particularly for long sequences, the amount of data needed to be processed is large. By relying on a synthetic dataset, we can keep images small and yet contain all relevant aspects. The central idea is that the synthetic dataset is sufficient to analyze models with respect to their capacity to keep track of events in the past. In our SymbolSeq dataset, events involve symbol-to-symbol transitions while in the real-world events are more complex.

In our experiments, we compare various state-of-the-art models (see Sec. 7.4.1) on this new dataset. Additionally, experiments on anticipation in the natural image datasets EPIC Kitchens and SomethingSomething V1 are conducted. As stated above, anticipation is a natural application for models dealing with long sequences since it requires to keeping track of what has happened before to make an informative guess of what will happen next. For example, in the kitchen setting of EPIC, the preparation of a breakfast takes several walks to the refrigerator. To make a guess of what will be retrieved next, we need to keep score of the items that were taken out before. This is different from action recognition where all relevant information is contained in a short sequence.

7.2 Related Work

7.2.1 Long-term Dependencies in RNNs

Capturing long-term dependencies (without visual input) is an old problem [19] in machine learning. Notable approaches addressing the problem are long short-term memory (LSTM) [91] and gated recurrent units (GRU) [40], which build upon basic recurrent neural networks. Echo-state networks [99] make use of randomly connected recurrent networks for which readouts are trained in a supervised manner. More recently, the statistical recurrent unit was proposed by Oliva, Póczos, and Schneider [156] and Trinh et al. [206] argue in favor of auxiliary losses to tackle the long-term dependency problem.

7.2.2 CNN-based Video architectures

Recently, multiple neural network-based models to process videos were proposed. One of the first approaches to apply a CNN for video processing was Karpathy et al. [105] who explored various architectural designs to fuse frame features. Furthermore, they introduced the Video1M dataset, which contains videos of a length of more than five minutes on average. The annotated clips refer to much smaller time frames, though. Bilen et al. [22] address video modeling by mapping videos to 2d intermediate representations from which allows using conventional 2d-CNN for classification. Besides the ubiquitous 2d-convolutions, 3d-convolutions, simultaneously operating over space and time, are frequently used, e.g. by Carreira and Zisserman [30], Hara, Kataoka, and Satoh [85] and Varol, Laptev, and Schmid [218]. The ECO network by Zolfaghari, Singh, and Brox [252] combines 2d convolution-based feature extraction with temporal processing using 3d-convolutions. The temporal relation network proposed by Zhou et al. [249] compares multiple tuples containing two or more frame features to classify a video. Xie et al. [236] explore various architectures involving different positions of 2d- and 3d-convolutions. A common approach to incorporate the temporal domain of video is to use a recurrent neural network on top of features extracted using a shared 2d-CNN [196].

The recent work of Varol, Laptev, and Schmid [218] focuses on long-term dependencies, involving videos up to 100 frames. A model using five layers of 3d-convolutions is proposed. For a comparison of work involving predictions from long sequences see Tab. 7.1. We differentiate between the number of frames and the actual duration of clips because videos can be sampled at different frame-rates.

In addition to RGB images, frequently optical flow is used as a second input [228, 30, 218] and has shown been to improve performance. However, we decided not to use optical flow

Table 7.1: Comparison of State-of-the-art datasets regarding sequence length in terms of number of frames and average clip duration.

Paper	Frames	Time
Donahue et al. [59]	16	6.2s (UCF101)
Yue-Hei Ng et al. [241]	120	2 mins (Sports 1M)
Srivastava, Mansimov, and Salakhudinov [196]	16	3.2 (HMDB), 6.2s (UCF101)
Varol, Laptev, and Schmid [218]	100	3.2 (HMDB), 6.2s (UCF101)
Pigou et al. [164]	60	1 - 2 mins (Montalbano Gesture)
Carreira and Zisserman [30]	64 [a]	10s (Kinetics)

[a] at training time

because we are mainly interested in semantic predictions, where we expect complex object movements not to play a crucial role. Furthermore, optical flow would add another layer of complexity.

Pre-training In general, pre-training plays an important role for the performance of video models. Carreira and Zisserman [30] found that pre-training on a large video dataset was beneficial for other classification tasks. Mettes, Koelma, and Snoek [142] pre-train their video model on the complete ImageNet hierarchy consisting of 21,814 classes and observe an improvement in video tasks compared to pre-training with only 1000 classes. Hara, Kataoka, and Satoh [85] found only the Kinetics dataset [30] to be sufficiently large to train a video model. They found that Kinectics pre-training improved performance on the UCF101 [195] and HMDB-51 [120]. All of these datasets involve actions rather than long-term activities, though.

7.2.3 ANTICIPATION

Vondrick et al. [221] propose a system that represents the future in terms of feature extractor representations. This has the advantage of allowing training in a self-supervised fashion. Instead of predicting future labels, a natural approach is to predict images as a whole. This has the advantage that data is available in large quantities, as any video can be leveraged for this task. Walker et al. [226] generate future frames by using a generative adversarial network [77] that is conditioned on predicted future poses.

7.3 SymbolSeq Dataset

SymbolSeq is a synthetic video dataset that we specifically designed for the purpose of investigating long sequences. Its images are of size 36x36 pixels and contain randomly moving symbols starting from a random location at a random time. After a while, these symbols transform to other symbols or stay the same. The goal is to count how often certain symbol-to-symbol transitions occur. In Fig. 7.1 we present samples from SymbolSeq with different parameters. The dataset is motivated by the need to simulate real life videos where events might occur at any time and must be aggregated and processed to infer a meaning. For example, in a household setting, multiple, subsequent human interactions with various objects must be detected in order to interpret the conducted activity or to predict what activity might follow.

The dataset is configurable with respect to multiple parameters, allowing the investigation of different aspects of a model in detail. Specifically, these parameters are:

- **Sequence Length** The number of frames the sequence involves, denoted by F.
- **Number of Symbols** The number of symbols to be processes by the network, denoted by S.
- **Number of Distractors** The number of distractive cues, denoted by D.
- **Symbol Noise** The presentation quality of each symbol can be reduced by omitting a certain amount of pixels making the recognition of this symbol more challenging

The scale range of the symbols is parameterizable, too. However, in all experiments of this work we randomly sample the point size from $\{6, 7\}$.

An important feature is that the recognition of symbols is fairly easy and will not require a complex feature extractor. This, in conjunction with the small size of the images, enables a fast conduction of experiments without having to rely on pre-training.

7.3.0.1 Symbolic SymbolSeq

Besides the visual representation described above, we can represent a sequence symbolically in form of a matrix $\mathbf{X}_{\text{sym}} \in \mathbb{R}^{F \times (S+D)}$. An element at index (i, j) in this matrix indicates whether a symbol or distractor i is present in certain frame j. Using this matrix, no feature extraction is required. Through comparison with models trained on this matrix, we can analyze the feature extraction in isolation.

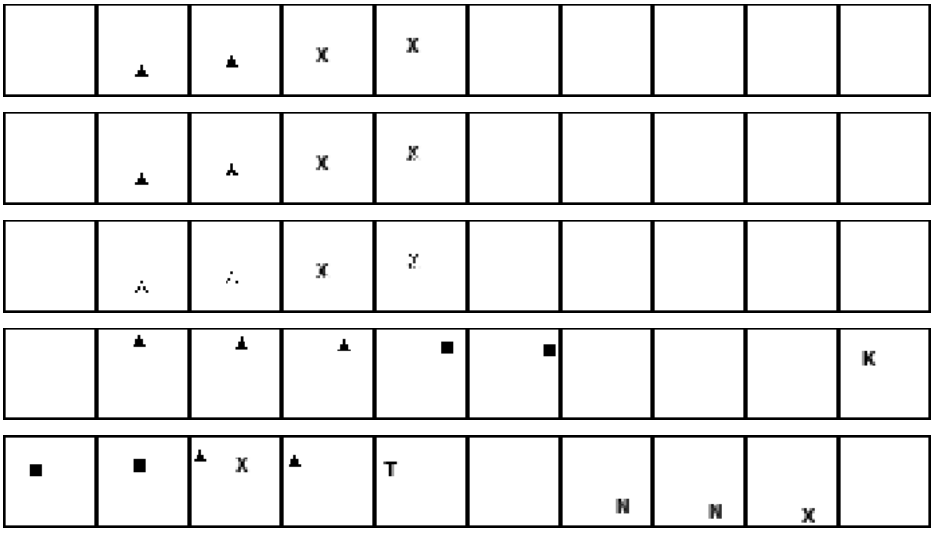

Figure 7.1: Short sequence samples from the SymbolSeq dataset involving different parameterizations. By rows: Base setting with one symbol (1), little symbol noise of 0.1 (2), symbol noise of 0.4 (3), with distractor (4), simultaneous symbols (5). There are four symbols: X, T, triangle and square, other letters are distractive cues. The goal is to count symbol transitions (i.e. 4^2 numbers). To properly identify the right transition the past trajectory of a symbol must be considered: In the fifth row, the cube transforms to X and not triangle.

7.3.0.2 Labels

There are two different questions that could be asked for each sequence: Did a certain symbol transition occur? And how often did each symbol transition occur? For generating labels, we consider the latter option and use counts of transitions as targets to be predicted because we consider counting transitions a suitable proxy task for detecting events in natural video. The dataset is based on four symbols. This means there are 16 possible symbol transitions, including not changing the symbol. Transitions can occur multiple times, so for each sample 16 transition counts must be predicted. These counts are discretized into five bins corresponding to counts of 0, 1, 2, 3 and more than 3. This allows us to regard the problem as a classification. The number of actually present transitions is a parameter of the dataset and can be set individually for each experiment.

7.3.0.3 Metrics

For each transition, our models output a probability distribution over all five bins, with the bin having most mass being considered to be the count prediction. We consider a sample to be correctly classified if all 16 symbol transitions were predicted to be in the right bin. The fraction

of correctly classified samples within all samples is our accuracy score.

7.4 MODEL

We differentiate between one and two-stage models. The latter consist of two components: A feature extractor f and a temporal sequence model s: Given a sequence $\mathbf{x}$ with each $\mathbf{x}_i$ being an image, a feature encoder f converts images into feature vectors. Then, a sequence model uses these features to make a prediction about the sequence: $\hat{y} = s(\mathbf{x}) = s(f(\mathbf{x}_0), ..., f(\mathbf{x}_N))$. An advantage of two-stage models is that the feature extractor f can be re-used from other tasks such as image classification.

One stage models unify both steps into a single function g by processing spatial and temporal information simultaneously: $\hat{y} = s(\mathbf{x}) = g(\mathbf{x}_0), ...)$. While this design is more general, it is harder to transfer knowledge, as pre-training must be carried out on the full model rather than the feature extractor only.

7.4.1 IMPLEMENTED MODELS

We implement several models covering a broad range of the state-of-the-art techniques available for long sequence modeling of videos.

7.4.1.1 2d-CNN + RNN (two-stage)

A straightforward way to implement the sequence model s is through a recurrent neural network (RNN). At each time step, an RNN cell receives two inputs, an external input (i.e. a feature of a frame) and the previous hidden state, and returns an output vector. Let $\mathbf{x}_t$ be a frame at time t. For the first frame the previous hidden state is often defined as the zero vector: $\mathbf{y}_0 = g_0(\mathbf{0}, f(\mathbf{x}_0))$. The following outputs can then be generated recursively by $\mathbf{y}_t = g_t(g_{t-1}, f(\mathbf{x}_t))$. In a multi layer RNN, the later layers are provided with outputs of previous layers, i.e. $\mathbf{y}_t^{(2)} = g_t^{(2)}(g_{t-1}^{(2)}, g_t^{(1)})$. The classification is obtained by average pooling all outputs $\mathbf{y}_t$ of the latest layer.

In our experiments we consider the classical RNN in addition to bidirectional LSTM [91] and bidirectional GRU [40] cells, which are designed to capture long-term dependencies. In all cases we use two hidden layers and an internal state of 512 neurons (unless stated otherwise).

7.4.1.2 2d-CNN + TCN (two-stage)

Recent research by Bai, Kolter, and Koltun [12] suggests that temporal convolutions can achieve a similar performance as recurrent networks. For the temporal model s, we follow the approach of [12] and stack five layers. Each layer has a residual connection and involves two convolutions with a kernel size of two. Dilation successively increases with each layer, enabling a large (temporal) receptive field.

7.4.1.3 ECO Lite (two-stage)

ECO Lite [252] is another two-stage approach, particularly designed for long sequences. Here, the feature extractor f obtains features over various spatial locations (represented by a tensor instead of a vector). These features are then stacked and processed by a 3d-CNN, i.e. s is a 3d-CNN. We use our own implementation of ECO Lite, which uses the squeeze-and-excitation ResNeXt 50 model as feature extractor [94]. For further details we refer to the paper by Zolfaghari, Singh, and Brox [252].

7.4.1.4 FFPool Baseline (two-stage)

Frame feature Pooling (FFPool) is a baseline that has a strongly simplified form of sequence modeling: s is implemented by pooling over the temporal domain. It applies a feature extractor, e.g. ResNet50, on each frame. On the resulting tensor, for each dimension, statistics are calculated and concatenated. Specifically, we calculate maximum, minimum and mean across all frames. Consequently, the output of FFPool is invariant to permutations of the input sequence.

7.4.1.5 3d ResNet (one-stage)

Based on the highly successful ResNet architecture [88], Hara, Kataoka, and Satoh [85] develop deep video models. Their idea is to take these image processing architectures and extend them to process video through the use of 3d-convolutions. Here we use their implementation [1] as well as their weights pre-trained on Kinectics [107].

7.4.1.6 LTC-CNN (one-stage)

We re-implement the long-term temporal convolution network proposed by Varol, Laptev, and Schmid [218] that involves simultaneous filtering over time as well space with 3d-convolutions.

[1] https://github.com/kenshohara/video-classification-3d-cnn-pytorch

Table 7.2: Comparison of different models for transition counting on a sequence length of 100. SqueezeNet [96] (SN) is used for feature extraction in the two-stage methods LSTM, GRU and FFPool. Accuracy is measured by counting samples where all 16 transitions were predicted correctly.

model	source	Accuracy
SN + FFPool	image	0.0
ECO	image	89.4
LTC-CNN	image	88.4
LTC-CNN (modified)	image	**93.9**
SN + RNN	image	87.3
SN + GRU	image	87.8
SN + LSTM	image	87.1
SN + TCN	image	87.4
MLP	symbol	0.0
TCN	symbol	76.6
LSTM	symbol	82.8

Two configurations are compared: The original one proposed by [218] and a modified version. In the latter case, temporal dimension filter sizes are 1, 1, 3, 3 and 2 and we apply a pooling of 2x2x1 in first two layers instead of only the first one while other pooling sizes are 2x2x2. This means that this network yields twice as many temporal features. The number of feature maps of the modified version is 32, 64, 128, 256 and 256.

7.4.1.7 Multi-Layer-Perceptron (MLP) Baseline (only symbolic input)

In the case of symbolic input data, i.e. when no visual feature extraction needs to be carried out, we can feed the matrix $\mathbf{X}_{\text{sym}}$ describing a sequence directly into a multi layer perception. Here, an MLP is used with three hidden layers composed of 1024 neurons each. After each layer except for input and output a ReLU non-linearity is applied.

7.5 EXPERIMENTS

We begin our experiments on the synthetic SymbolSeq dataset by investigation relations between distraction, sequence length, symbol quality and size of the training set.

7.5.1 Distraction and Sequence Length

We conduct a more detailed analysis on distraction, starting by determining how strong distractive cues impact the performance. We configure the dataset to show only a single symbol at a time while distractors can occur in any frame. Simultaneously, we compare with symbolic versions of the dataset. Results are shown in Tab. 7.2. We do not expect that images in SymbolSeq require complex feature extraction. Therefore, for two-stage approaches, we rely on the small and computational efficient SqueezeNet [96].

We observe that FFPool and MLP do not work. The former is expected as the dataset requires the consideration of order while the FFPool output is invariant to permutations of the input sequence. In contrast to TCN and LSTM, MLP does not assume the input to be sequential, which we hypothesize is the reason for MLP's bad performance. Without such an inductive bias, learning in MLP might be less efficient and require more samples. Despite their individual differences, other RNN- or 3d-convolution-based architectures work well but are outperformed by models involving 3d-CNNs with a modified version of LTC-CNN working best.

Next, we investigate the impact of varying the sequence length and the level of distraction. While the number of frames varies, the number of symbols scattered over the frames is kept constant. This means, the expected values of the counts remain the same in each setting while the sequence length varies. This allows us to study the effects of longer sequences in isolation. We vary between four different modes of distraction by enabling distractors in the training and test datasets respectively. The corresponding scores for different models including a baseline which does not receive visual input (no inp.) are shown in Fig. 7.2. The number of distractors D_F at F frames is sampled from uniform distributions U such that the distraction densitity remains constant, i.e. $D_{50} \sim U(0,1)$, $D_{150} \sim U(0,3)$ and $D_{300} \sim U(0,6)$.

We find a consistent pattern for all models despite their technical differences. Without distraction or when distraction is present during training, close to perfect performance is obtained. If distraction only occurs at test time, the performance drops with increasing sequence length or increasing number of distractors.

7.5.2 The Role of Features

This section focuses on the feature extractor, which is a crucial component of the model. A desirable feature extractor detects the symbols at any position while being resilient to distractors and noise.

Table 7.3: The effect of pre-training and freezing layers for transition counting on SymbolSeq for two-stage models.

	GRU	GRU	GRU	LSTM	LSTM	LSTM	TCN	TCN	TCN
Freeze Features	✓	-	-	✓	-	-	✓	-	-
Pre-training	✓	✓	-	✓	✓	-	✓	✓	-
Accuracy	91.8	99.3	98.9	85.8	99.1	99.3	87.3	97.7	88.4

7.5.2.1 Pretraining

Conventionally, in two-stage approaches, feature extractor and sequence model are trained jointly. However, datasets often provide frame-wise features instead of images, essentially freezing the feature extraction (i.e. features are not modified during training). Here we investigate the effects of freezing the feature extractor and pre-training on two-stage models. Pre-training is carried out in a single image setting: The feature extractor is trained to recognize up to two symbol simultaneously. This is intended to be a reasonable equivalent to the single image pre-training on ImageNet which is common for many video tasks. We do not pre-train on the large scale video dataset Kinetics [107] because we expect semantic qualitities learned from differentiating object classes in ImageNet to be more important than recognizing the motions prevalent in the actions of Kinetics.

From the scores reported in Tab. 7.3 we learn that fixing weights consistently leads to a diminished performance. Of course, one might argue that this effect depends on the concrete pre-training. However, a dependency on the pre-training task exists for natural images, too, where ImageNet pre-training may not necessarily be the best choice. Pre-training in conjunction

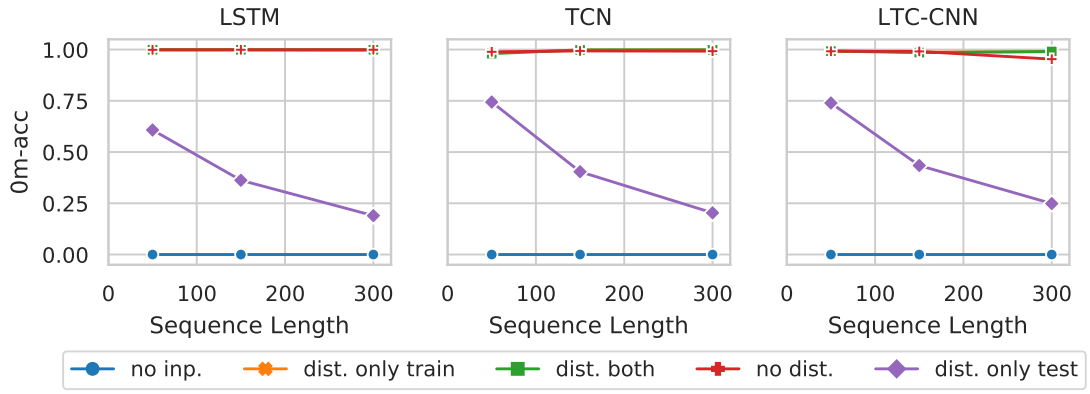

Figure 7.2: Accuracies of different distraction settings over varying frame-lengths on SymbolSeq. Dist. means distraction. The curves for red, orange and green curves overlap.

with trainable feature encoder weights only seems useful for TCN. This seems to suggest that the underlying sequence model has an impact of how well (or how fast in terms of iterations) features are learned. While we used a large training dataset here, next we focus on a setting where training data is limited.

7.5.2.2 Learning from few samples

Often, training data is scarce. When can feature extractors still be learned without relying on pre-training? We try to answer this question by training from differently sized training sets while varying the difficulty of feature extraction. The latter is implemented by omitting random pixels from each symbol (symbol noise in Sec. 7.3).

In Fig. 7.3, we see that learning meaningful features poses a problem in particular if training is carried out from few noisy samples. While LTC-CNN fails for smaller dataset sizes, it exhibits better performance if enough data is provided, especially if the symbol noise is strong. This might be due to the spatio-temporal kernels allowing for a more robust symbol detection but requiring more training data.

7.5.3 Early Recognition on Something

SomethingSomething V1 [78] is a dataset for learning visual common sense from video. It includes 174 activities such as "pour something into something" or "turning something upside down". For each sample only one class is correct. In Tab. 7.4 we report accuracies for early recognition of actions on this dataset. This means the model has to classify each clip based on the first 20% of all frames only. This can be seen as a precursor of actual anticipation, because it requires some intuition of how certain cues are correlated with future actions. We use sequences

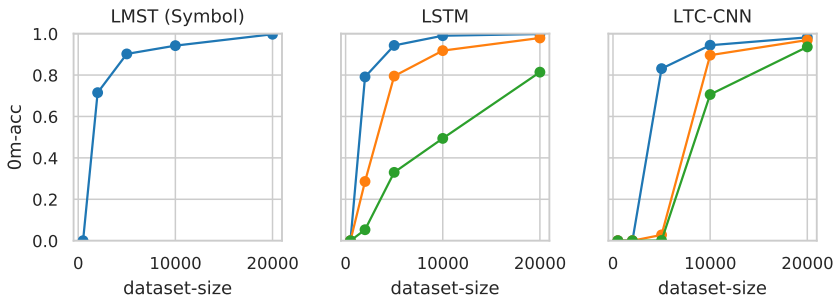

Figure 7.3: Sample efficiency of LSTM, with symbol input on the left and visual input in the middle, and LTC-CNN with visual input. Blue means no symbol noise, orange and green corresponds to symbol noise of 0.2 and 0.4. Dataset-size refers to the number of samples.

Table 7.4: Early detection (first 20% of frames) on SomethingSomething V1 dataset. Accuracy@N measure how often the ground truth is within the top N predictions of the model.

model	pretraining	Accuracy@1	Accuracy@10
No input	-	0.9	5.8
RN50 + FFPool	✓	**10.6**	**43.2**
ECO Lite	✓	7.5	34.6
RN50 + LSTM	-	4.3	24.1
RN50 + LSTM	✓	9.8	41.5
LTC-CNN	-	3.8	22.4
LTC-CNN	✓	4.9	25.2

of ten frames, images are scaled to 110x202px and we train using Adam [110] for 25 epochs with early stopping after 3 epochs without improvement. The batch size is 24.

7.5.3.1 Pretraining of LTC-CNN

While pre-trained weights are readily usable for the frame-wise image models used in two-stage models, for one-stage models we must implement our own pre-training. We pre-train LTC-CNN on ImageNet [53] for 20 epochs. Static images are converted into movie sequences by moving the image with random speed into a random direction.

7.5.3.2 Results

We find that the sequential processing can be neglected. The order insensitive FFPool model performs on-par with the LSTM-RN50 model. If not pre-trained on ImageNet, models consistently achieve lower accuracies. This suggests that the crucial features are too complex or noisy to be learned from the dataset itself, akin to high-noise setting in Fig. 7.3. There might be only few relevant cues that indicate the video type, otherwise we would expect the order of the cues to have a positive impact on performance. LTC-CNN performs worse than two-stage methods, even when pre-trained on ImageNet. We attribute this to the LTC-CNN architecture being not as deep as the ResNet encoders it competes with and hence not being able to capture as complex features.

7.5.4 Anticipation in EPIC Kitchens

The anticipation ability of the models is evaluated on Epic Kitchens [50]. Actions are divided into a noun and a verb and can be predicted independently. Hence, for each sample, we obtain two accuracies, one for verb and one for noun. While most current video datasets contain only

Table 7.5: Comparison of different models on Epic Kitchens. Batch size varies due to different memory demand.

model	input	pretrained	batch-size	Accuracy verb	Accuracy noun
RN50 + FFPool	images	✓	16	21.1	6.8
ECO Lite	images	✓	16	23.0	4.5
ResNet3d	images	✓	16	20.7	5.1
LTC-CNN	images	-	16	21.5	4.1
LTC-CNN	images	✓	16	22.6	4.5
RN18 + LSTM	images	-	24	20.5	3.5
RN18 + LSTM	images	✓	24	23.6	7.2
FFPool	symbols	-	24	27.0	39.2
LSTM	symbols	-	24	**31.1**	**59.7**
TCN	symbols	-	24	30.3	55.4

fairly short clips, EPIC Kitchens encompasses long sequences with multiple actions being annotated. This enables us to specifically investigate the long-term context setting we are interested in. Ten frames of a temporal window, ranging from five seconds to one second before the anticipated action takes place, are fed to the network. Images have a size of 140x245px except for ResNet3d which works with 160x160px input. Note, this is a challenging setting and we would not expect humans to achieve a 100% accuracy since there often is some level of uncertainty. Performance for object and verb classification is shown in Tab. 7.5. For LTC-CNN, ImageNet pre-training using the method described above in Sec. 7.5.3.1 is applied. For the symbolic input, we use annotations (objects and verbs) from the temporal window as input features. Thus, anticipation is still required but no feature extraction.

7.5.4.1 Results

The conclusion on the visual input is similar to the one of the previous section: Features matter. This is confirmed when we consider the cases where verb and noun annotations were used as features (symbolic input), i.e. emulating an extremely good feature extractor, which resulted in a dramatic increase of performance. Consequently, it seems, in order to carry out sophisticated anticipation, we need better feature extractors that work well in household scenes, capturing multiple small objects at the same time and relations to the person acting.

7.6 CONCLUSION

We presented an empirical evaluation of long-term temporal context integration in state-of-the-art video models with the objective of conducting anticipation. Specifically, we quantified the interplay between sample-efficiency and signal quality, sequence length and performance under distraction and analyzed various forms of pre-training.

Our findings suggest that feature extraction represents a bottleneck in natural video tasks. While this bottleneck can be reduced using transfer-learning, performance cannot (yet) compete with purely symbolic representations that bypass feature extraction. The latter we have shown for both, synthetic and natural video. Furthermore, in anticipation from natural video, we found the frame order to be irrelevant for classification. This appears to be an artifact of bad features, too, because when provided with symbolic input order mattered. This motivates more research on approaches for learning meaningful feature extractors particularly for video. Datasets are required that involve long sequences in diverse environments with an emphasis on common sense, unlike the focus on action and sports in Sports1M and Kinetics.

We believe that synthetic datasets are a useful tool for conducting future research in video modeling due to their ability to keep the visual complexity limited and control parameters of interest.

7.7 REFERENCES

We maintain a single list of references at the end of the thesis.

Chapter 8

Conclusion

In this thesis, we argue in favor of an action-oriented interpretation of visual scenes. Action is a central target of perception and should therefore be given substantial weight when scenes are analyzed. We show various ways of designing systems that implement action-oriented scene understanding on three different levels of abstraction: elementary, perception and reasoning. All levels share the goal, that functional or action-related aspects of the scene are identified and predicted. Subsequently, we present the individual contributions of articles contained in this thesis, structured by the three levels of abstraction.

8.1 Contribution Summary

8.1.1 Object Functions (Elementary Level)

Our work on object semantics is, to the best of our knowledge, the first approach to use the visual context of objects for generating vectors of meaning. We show that visual context, in form of features extracted by a CNN, allows to generate meaningful vectors which can be combined with text-based vectors to obtain even better vectors. Quality is assessed in terms of clustering consistency and correlation with human ratings using the Scene250 dataset. The latter we collected for this purpose because no dataset existed specifically focusing on object similarity. Moreover, our findings suggest the emergence of functional groups in semantic vectors. For example, we found that vector representations of things that can fly form groups.

8.1.2 Action Identification (Perception Level)

At the perceptive level, we developed a system that can densely predict affordances. The number of affordances we incorporate is larger than in previous approaches. Additionally, the training algorithm which relies on object part annotations and does not need 3D scans of the scene is novel. After training, the system can predict pixel-wise affordance probabilities from an RGB image. In the experiments, we showed that our networks compare favorably to state-of-the-art approaches.

8.1.3 Action Rating and Anticipation (Reasoning Level)

At the most abstract level, we made two contributions: Action rating from static images and semantic anticipation from movies. We designed the novel task of action plausibility rating, where plausible actions must be determined based on a static image of a scene. For this, we manually gathered a dataset by asking human raters to judge how likely they consider certain actions in given situations. Then we trained a neural network to predict the distribution of the human-based ratings. This associates distinct visual cues with actions such that the plausibility of actions in unknown images can be calculated. In our experiments, we find a strong connection between action plausibility and object recognition performance. However, contextual cues seem to be negligible.

A related task is anticipation on videos. As a typical trait of action-oriented scene understanding, both have in common that we must work with plausibilities rather than definite true or false predicates. However, in video anticipation, the input is a sequence of images, which makes dealing with data more challenging. To this end, we designed the synthetic SymbolSeq dataset where symbol transitions need to be counted. This dataset allows us to conduct an extensive evaluation of various models with respect to their ability to capture long-term temporal context. Additionally, we evaluated anticipation in natural image datasets. A special trait of our experiments is that we compare symbolic baselines that bypass feature extraction. Our findings suggest that features are utterly important for anticipation, and for learning features often a large proxy dataset (such as ImageNet [53] for image classification) is necessary.

8.2 Discussion and Future Work

On the long path towards truly autonomous robots, the contributions made in this thesis make up only a small fraction. Many ideas presented here constitute novel approaches towards solving their corresponding problems and more research will be needed to advance these methods

to the level of practical applicability. In general, we show that the space of actions represents an attractive alternative to classic objects. In the following, we discuss potential avenues for future work involving general ideas as well as individual improvements on the three levels of abstraction. A graphical overview is provided by Fig. 8.1.

Since all presented projects rely on machine learning methods, we expect improvements from larger training datasets. This would enable us to generate semantic vectors for a larger set of object classes. In fact, it would be interesting to use the Open Images dataset [115] for this project, which was not available when at the time when we conducted the experiments. On the perceptive level, a more robust affordance segmentation model could be trained with larger datasets. In this case, we would require part-level segmentation ground truth, which requires more annotations efforts than usual segmentation data. In action plausibility rating, we rely on pre-trained ImageNet features. While this is technically inevitable because our datasets are too small to learn meaningful features, we expect such feature to introduce a bias. The available of larger training datasets would enable feature extractors specifically tuned for actions, which might show different properties than the well-known features extractors for objects. Using our web-based annotation tool, we gathered around 130 annotations per Euro spent, so data collection on a large scale is possible but might be expensive. For semantic anticipation, a large dataset could enable learning strong spatio-temporal features without ImageNet pre-training. Following the conclusions from the article, we expect this to improve anticipation quality.

A promising solution to evade the cost of labeling a large-scale dataset could be through simulation. In affordance segmentation we have shown that simulated data can improve training. Once the simulation model is set up, an unlimited number of samples can be drawn from it. The key challenge is to design a simulation that has enough variability while generating samples close enough to the distribution of real data. Recently, the development of simulation environments has become a lot easier due to the availability of modern game engines like

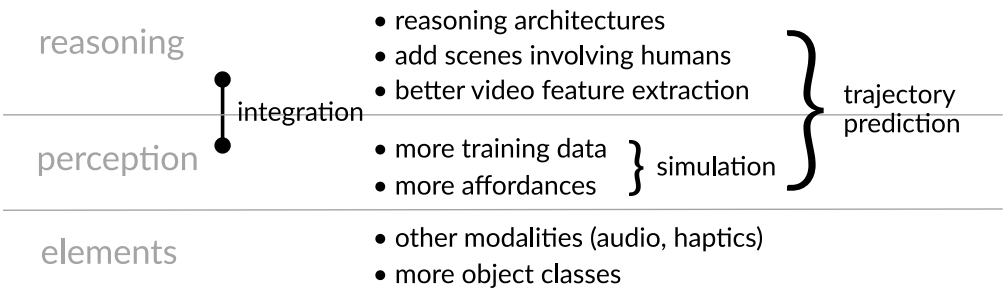

Figure 8.1: Potential future work grouped by the three levels abstraction.

Unreal [1] and Unity [2] as well as improvements in 3D modelling tools like Blender [3]. In fact, Warnecke, Lüddecke, and Wörgötter [230] made use of the latter to simulate physics and render image sequences. Simulated models do not necessarily need to generate photo-realistic images. Tobin et al. [205] trained a CNN to detect objects in simulated low-fidelity images and showed that the model generalizes well to real images. They found variability in the simulation (e.g. texture randomization) to be a crucial factor for such good performance.

The reinforcement learning paradigm is mostly evaded in this work. This is because household environments are diverse and expensive to simulate. However, given more sophisticated interactive environments, it might become feasible to employ reinforcement learning. This is interesting because it could avoid the dependency on annotated training data which is a consequence of supervised learning.

In many current computer vision systems, transfer learning (Section 2.3.4) plays a pivotal role. Performance substantially benefits from pre-trained weight initializations that were obtained by classifying ImageNet [53] images. The variety of this dataset allows for learning fairly generic features. However, recent research suggests that other forms of pre-training can perform better [139]. For semantic anticipation, we believe that pre-training on multi-object detection could be helpful. Potentially, this pre-training could be formulated as a self-supervised task which would reduce the costs of gathering labels.

8.2.1 Specific Future Work

8.2.1.1 Elementary Level

To represent objects semantically (Chapter 4 / A), a natural next step is to extend the method to modalities other than images. Potentially, this could involve sound, haptic or smell. We expect in particular haptics to be relevant for in robotic contexts as the structure determines to a large extent how an object can be grabbed and manipulated. Motivation for merging different sensual percepts can be found in the successful work on combining image and sound carried out by Arandjelovic and Zisserman [10].

8.2.1.2 Perceptive and Reasoning Level

On the perceptive and reasoning level (Chapters 5, 6 and 7 / B, C and D), our output still involves probability distributions over labels. A more natural way to express actions is

[1] https://www.unrealengine.com
[2] https://unity3d.com
[3] https://blender.org

through trajectories and forces, for example of the hand as a whole or individual fingers. Such a representation would acknowledge the fact that transitions between actions are continuous. This means that there exist actions which lie between two classes and yet are proper actions. Hence, a useful next step could involve directly generating the trajectories that carry out these actions by training an end-to-end system. However, an obvious problem for this is missing training data. Hence, this would probably require a more sophisticated training procedure than supervised training, possibly involving self-supervision and reinforcement learning.

The segmentation of affordances and the rating of plausibilities (Chapters 5 and 6 / B and C) could be joined into a single model. This could be implemented by a two-stage process where the pixel-wise probabilities for affordances are multiplied with the pixel-wise plausibilities for actions. However, this would require compatibility between the classes of the two approaches, which is not the case in the current form.

In action plausibility rating (Chapter 6 / C), a natural extension would be to incorporate humans into scenes, allowing a more realistic setting. We did not consider this setting because it requires to anticipate the intentions of humans. Furthermore, models which are explicitly designed for reasoning on relations, such as the Relation network [179], could be used to rate action plausibilities. Such models could learn relations between pairs of objects that are critical for actions. For example, the plausibility of cutting bread increases if a knife lies next to it. The success of semantic action chains [5] which are based on touch relations between objects and hands suggests that relational features could be useful, too. This refers to features that specifically encode the interactions between two or more elements of the scene rather than individual traits. Recent development in natural language processing indicates that attention mechanisms are powerful tools not only for language-related tasks [219] but also for vision [39]. Often specific actions have a strong impact on what might happen next. A natural way to incorporate this into models would be an attention mechanism over the input sequence.

8.2.2 Outlook

All things considered, we believe that computer vision and robotics must grow even closer to enable a tighter coupling between perception and action. Clearly, first steps have been taken, such as those papers discussed in Section 3.2.2. However, in general, the topic is under-explored and many questions remain unanswered. This involves both, the representation and acquisition of procedural knowledge. The non-symbolic approaches presented in this thesis contribute to this discussion but cannot give an ultimate answer.

In general, a promising approach for future work is to develop modular neural architectures that enable end-to-end training from perception to action while being decomposable.

Each module is a separate unit which can receive specific pre-training and encode specific assumptions. Based on such modules, we deem it of paramount importance to introduce transfer learning to robotics. Transfer learning has caused a massive increase in performance in many computer vision and natural language processing tasks. In robotics it could evade costly retraining and systems could be built upon existing knowledge. We are confident that these enhancements could pave the way to higher levels of autonomy and cooperation (with humans) in robotic systems, ultimately relieving us from several unwanted and stressful tasks.

Bibliography

[1] M. J. Aein et al. Toward a Library of Manipulation Actions Based on Semantic Object-action Relations. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Nov. 2013, pp. 4555–4562.

[2] M. J. Aein, E. E. Aksoy, and F. Wörgötter. Library of Actions: Implementing a Generic Robot Execution Framework by Using Manipulation Action Semantics. In: *The International Journal of Robotics Research (IJRR)* (2019).

[3] A. Agostini, C. Torras, and F. Wörgötter. Efficient Interactive Decision-making Framework for Robotic Applications. In: *Artificial Intelligence* 247 (2017).

[4] B. Akgun et al. Unsupervised Learning of Affordance Relations on a Humanoid Robot. In: *Computer and Information Sciences, 2009. ISCIS 2009. 24th International Symposium on*. IEEE. 2009, pp. 254–259.

[5] E. E. Aksoy et al. Learning the Semantics of Object–action Relations by Observation. In: *The International Journal of Robotics Research (IJRR)* 30.10 (2011), pp. 1229–1249.

[6] A. Aldoma, F. Tombari, and M. Vincze. Supervised Learning of Hidden and Non-hidden 0-order Affordances and Detection in Real Scenes. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2012, pp. 1732–1739.

[7] D. A. Allport. Selection for Action: Some Behavioral and Neurophysiological Considerations of Attention and Action. In: *Perspectives on perception and action* 15 (1987), pp. 395–419.

[8] J. Aloimonos, I. Weiss, and A. Bandyopadhyay. Active Vision. In: *International Journal of Computer Vision (IJCV)* 1.4 (Jan. 1988), pp. 333–356.

[9] C. S. Ang et al. Data in the Wild: Some Reflections. In: *ACM Interactions* 20.2 (2013), pp. 39–43.

[10] R. Arandjelovic and A. Zisserman. Look, Listen and Learn. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 609–617.

[11] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Scene Segmentation. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2017).

[12] S. Bai, J. Z. Kolter, and V. Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. In: *arXiv:1803.01271* (2018).

[13] R. Bajcsy. Active Perception. In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[14] R. Bajcsy, Y. Aloimonos, and J. K. Tsotsos. Revisiting Active Perception. In: *Autonomous Robots* 42.2 (Feb. 2018), pp. 177–196.

[15] J. Baleia, P. Santana, and J. Barata. On Exploiting Haptic Cues for Self-supervised Learning of Depth-based Robot Navigation Affordances. In: *Journal of Intelligent & Robotic Systems* 80.3-4 (2015), pp. 455–474.

[16] M. Bansal, K. Gimpel, and K. Livescu. Tailoring Continuous Word Representations for Dependency Parsing. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. 2014.

[17] C. Barck-Holst et al. Learning Grasping Affordance using Probabilistic and Ontological Approaches. In: *International Conference on Advanced Robotics (ICAR)*. IEEE. 2009, pp. 1–6.

[18] R. J. Barlow. Statistics: A Guide to the Use of Statistical Methods in the Physical Sciences. Vol. 29. John Wiley & Sons, 1989.

[19] Y. Bengio, P. Y. Simard, and P. Frasconi. Learning Long-term Dependencies with Gradient Descent Is Difficult. In: *IEEE transactions on neural networks* 5 2 (1994), pp. 157–66.

[20] Y. Bengio et al. A Neural Probabilistic Language Model. In: *The Journal of Machine Learning Research* 3 (2003).

[21] A. Bierbaum et al. Grasp Affordances from Multi-fingered Tactile Exploration using Dynamic Potential Fields. In: *IEEE RAS International Conference on Humanoid Robots (Humanois)*. IEEE. 2009, pp. 168–174.

[22] H. Bilen et al. Action Recognition with Dynamic Image Networks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40.12 (2018), pp. 2799–2813.

[23] C. M. Bishop. Pattern Recognition and Machine Learning. springer, 2006.

[24] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. In: *the Journal of Machine Learning Research* 3 (2003).

[25] J. Bohg et al. Interactive Perception: Leveraging Action in Perception and Perception in Action. In: *IEEE Transactions on Robotics* 33.6 (Dec. 2017), pp. 1273–1291.

[26] A. Brock, J. Donahue, and K. Simonyan. Large Scale Gan Training for High Fidelity Natural Image Synthesis. In: *International Conference on Learning Representations (ICLR)* (2019).

[27] E. Bruni, N.-K. Tran, and M. Baroni. Multimodal Distributional Semantics. In: *J. Artif. Intell. Res.(JAIR)* 49 (2014).

[28] A. Budanitsky and G. Hirst. Evaluating Wordnet-based Measures of Lexical Semantic Relatedness. In: *Computational Linguistics* 32.1 (2006).

[29] Cambridge Dictionary. Common Sense.

[30] J. Carreira and A. Zisserman. Quo Vadis, Action Recognition? a New Model and the Kinetics Dataset. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[31] J. T. Carvalho and S. Nolfi. Behavioural Plasticity in Evolving Robots. In: *Theory in Biosciences* 135.4 (2016), pp. 201–216.

[32] C. Castellini et al. Using Object Affordances to Improve Object Recognition. In: *IEEE Transactions on Autonomous Mental Development* 3.3 (2011), pp. 207–215.

[33] W. P. Chan et al. Determining Proper Grasp Configurations for Handovers Through Observation of Object Movement Patterns and Inter-object Interactions During Usage. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2014, pp. 1355–1360.

[34] A. X. Chang et al. Shapenet: An Information-rich 3d Model Repository. In: *arXiv preprint arXiv:1512.03012* (2015).

[35] Y.-W. Chao et al. Forecasting Human Dynamics from Static Images. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 548–556.

[36] A. Chemero. An Outline of a Theory of Affordances. In: *Ecological Psychology* 15.2 (2003), pp. 181–195.

[37] C. Chen et al. DeepDriving: Learning Affordance for Direct Perception in Autonomous Driving. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015.

[38] L.-C. Chen et al. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40 (2018).

[39] R. Child et al. Generating Long Sequences with Sparse Transformers. In: *arXiv preprint arXiv:1904.10509* (2019).

[40] K. Cho et al. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In: *EMNLP*. 2014.

[41] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1251–1258.

[42] P. Cisek. Cortical Mechanisms of Action Selection: The Affordance Competition Hypothesis. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 362.1485 (2007), pp. 1585–1599.

[43] W. G. Cole et al. Perceiving Affordances for Different Motor Skills. In: *Experimental brain research* 225.3 (2013), pp. 309–319.

[44] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED Framework: Object Recognition and Pose Estimation for Manipulation. In: *The International Journal of Robotics Research (IJRR)* 30.10 (2011), pp. 1284–1306.

[45] R. Collobert et al. Natural Language Processing (almost) from Scratch. In: *Journal of Machine Learning Research (JMLR)* 12.Aug (2011).

[46] A. M. Colman. A dictionary of psychology. Oxford Quick Reference, 2015.

[47] F. Cruz et al. Training Agents with Interactive Reinforcement Learning and Contextual Affordances. In: *IEEE Transactions on Cognitive and Developmental Systems* 8.4 (2016), pp. 271–284.

[48] G. Csurka et al. Visual Categorization with Bags of Keypoints. In: *Workshop on Statistical Learning in Computer Vision, ECCV*. Prague. 2004.

[49] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In: *International Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 1. IEEE Computer Society. 2005, pp. 886–893.

[50] D. Damen et al. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In: *European Conference on Computer Vision (ECCV)*. 2018.

[51] A. Das et al. Embodied Question Answering. In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 2054–2063.

[52] P. Dayan and L. F. Abbott. Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems. MIT press, 2001.

[53] J. Deng et al. Imagenet: A Large-scale Hierarchical Image Database. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, pp. 248–255.

[54] R. Detry et al. Learning Grasp Affordance Densities. In: *Paladyn, Journal of Behavioral Robotics* 2.1 (2011), pp. 1–17.

[55] R. Detry et al. Refining Grasp Affordance Models by Experience. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2010, pp. 2287–2293.

[56] J. Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In: *arXiv preprint arXiv:1810.04805* (2018).

[57] T.-T. Do, A. Nguyen, and I. Reid. AffordanceNet: An End-to-End Deep Learning Approach for Object Affordance Detection. In: *International Conference on Robotics and Automation (ICRA)*. 2018.

[58] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised Visual Representation Learning by Context Prediction. In: *IEEE International Conference on Computer Vision (ICCV)*. Dec. 2015.

[59] J. Donahue et al. Long-term Recurrent Convolutional Networks for Visual Recognition and Description. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2625–2634.

[60] D. G. Dotov, L. Nie, and M. M. De Wit. Understanding Affordances: History and Contemporary Development of Gibson's Central Concept. In: *Avant: the Journal of the Philosophical-Interdisciplinary Vanguard* (2012).

[61] Y. Feng and M. Lapata. Visual Information in Semantic Representation. In: *Conference of the North American Chapter of the ACL (NAACL)*. Association for Computational Linguistics, 2010, pp. 91–99.

[62] S. Fichtl et al. Bootstrapping Relational Affordances of Object Pairs Using Transfer. In: *IEEE Transactions on Cognitive and Developmental Systems* 10.1 (Mar. 2018), pp. 56–71.

[63] J. R. Firth. A Synopsis of Linguistic Theory, 1930-1955. In: *In Studies in Linguistic Analysis* (1957).

[64] D. F. Fouhey and C. L. Zitnick. Predicting Object Dynamics in Scenes. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2014.

[65] D. F. Fouhey et al. People Watching: Human Actions as a Cue for Single View Geometry. In: *International Journal of Computer Vision (IJCV)* 110.3 (2014).

[66] J. Franchak and K. Adolph. Affordances as Probabilistic Functions: Implications for Development, Perception, and Decisions for Action. In: *Ecological Psychology* 26.1-2 (2014).

[67] A. Frome et al. Devise: A Deep Visual-semantic Embedding Model. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 2121–2129.

[68] J. Garson. Connectionism. In: *The Stanford Encyclopedia of Philosophy*. Ed. by E. N. Zalta. Fall 2018. Metaphysics Research Lab, Stanford University, 2018.

[69] J. J. Gibson. The Ecological Approach to Visual Perception. Houghton Mifflin, 1979.

[70] J. J. Gibson. The Senses Considered as Perceptual Systems. Houghton Mifflin, 1966.

[71] R. Girshick et al. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 580–587.

[72] A. M. Glenberg and D. A. Robertson. Symbol Grounding and Meaning: A Comparison of High-Dimensional and Embodied Theories of Meaning. In: *Journal of Memory and Language* 43.3 (Oct. 2000).

[73] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*. 2011, pp. 315–323.

[74] B. Goertzel and C. Pennachin. Artificial General Intelligence. Vol. 2. Springer, 2007.

[75] G. Goh. Why Momentum Really Works. In: *Distill* (2017).

[76] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT press, 2016.

[77] I. Goodfellow et al. Generative Adversarial Nets. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2672–2680.

[78] R. Goyal et al. The" Something Something" Video Database for Learning and Evaluating Visual Common Sense. In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 1. 2. 2017, p. 3.

[79] Y. Goyal et al. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[80] H. Grabner, J. Gall, and L. Van Gool. What Makes a Chair a Chair? In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2011.

[81] S. Griffith et al. Object Categorization in the Sink: Learning Behavior–grounded Object Categories with Water. In: *Proceedings of the 2012 ICRA Workshop on Semantic Perception, Mapping and Exploration*. Citeseer. 2012.

[82] H. Guo and Y. Meng. Distributed Reinforcement Learning for Coordinate Multi-robot Foraging. In: *Journal of intelligent & robotic systems* 60.3-4 (2010), pp. 531–551.

[83] A. Gupta and L. S. Davis. Objects in Action: An Approach for Combining Action Understanding and Object Perception. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2007.

[84] A. Gupta et al. From 3D Scene Geometry to Human Workspace. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2011.

[85] K. Hara, H. Kataoka, and Y. Satoh. Can Spatiotemporal 3d Cnns Retrace the History of 2d Cnns and Imagenet? In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6546–6555.

[86] S. Harnad. The Symbol Grounding Problem. In: *Physica D: Nonlinear Phenomena* 42.1-3 (1990).

[87] Z. S. Harris. Distributional Structure. In: *Word* 10.2-3 (1954).

[88] K. He et al. Deep Residual Learning for Image Recognition. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[89] K. He et al. Mask R-cnn. In: *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[90] F. Hill, R. Reichart, and A. Korhonen. Simlex-999: Evaluating Semantic Models with (genuine) Similarity Estimation. In: *Computational Linguistics* 41.4 (2015).

[91] S. Hochreiter and J. Schmidhuber. Long Short-term Memory. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[92] K. Hornik. Approximation Capabilities of Multilayer Feedforward Networks. In: *Neural networks* 4.2 (1991), pp. 251–257.

[93] J. Howard and S. Ruder. Universal Language Model Fine-tuning for Text Classification. In: *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 328–339.

[94] J. Hu et al. Squeeze-and-Excitation Networks. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* (2019), pp. 1–1.

[95] J. Hu, L. Shen, and G. Sun. Squeeze-and-excitation Networks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7132–7141.

[96] F. N. Iandola et al. SqueezeNet: AlexNet-level Accuracy with 50x Fewer Parameters And< 0.5 MB Model Size. In: *arXiv preprint arXiv:1602.07360* (2016).

[97] A. J. Ijspeert, J. Nakanishi, and S. Schaal. Movement Imitation with Nonlinear Dynamical Systems in Humanoid Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Vol. 2. May 2002, pp. 1398–1403.

[98] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. International Conference on Machine Learning (ICML). Lille, France: JMLR.org, 2015, pp. 448–456.

[99] H. Jaeger. Long Short-term Memory in Echo State Networks: Details of a Simulation Study. Tech. rep. Jacobs University Bremen, 2012.

[100] D. Jayaraman and K. Grauman. Learning to Look Around: Intelligently Exploring Unseen Environments for Unknown Tasks. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 1238–1247.

[101] D. Jurafsky and J. H. Martin. Speech and Language Processing. Vol. 3. Pearson London, 2014.

[102] Á. Kádár, A. Alishahi, and G. Chrupała. Learning Word Meanings from Images of Natural Scenes. In: *Traitement automatique du langage naturel et sciences cognitives / Natural Language Processing and Cognitive Sciences (TAL)* 55.3 (2014).

[103] P. Kaiser et al. Extracting Whole-body Affordances from Multimodal Exploration. In: *IEEE-RAS International Conference on Humanoid Robots*. 2014.

[104] P. Kaiser et al. Validation of Whole-body Loco-manipulation Affordances for Pushability and Liftability. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2015, pp. 920–927.

[105] A. Karpathy et al. Large-scale Video Classification with Convolutional Neural Networks. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 1725–1732.

[106] T. Karras, S. Laine, and T. Aila. A Style-based Generator Architecture for Generative Adversarial Networks. In: *arXiv preprint arXiv:1812.04948* (2018).

[107] W. Kay et al. The Kinetics Human Action Video Dataset. In: *arXiv preprint arXiv:1705.06950* (2017).

[108] D. Kiela and L. Bottou. Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2015.

[109] D. I. Kim and G. S. Sukhatme. Semantic Labeling of 3d Point Clouds with Object Affordance for Robot Manipulation. In: *IEEE International Conference on Robotics and Automation (ICRA)*. Citeseer. 2014, pp. 5578–5584.

[110] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In: *CoRR* abs/1412.6980 (2014).

[111] H. Kjellström, J. Romero, and D. Kragic. Visual Object-action Recognition: Inferring Object Affordances from Human Demonstration. In: *Computer Vision and Image Understanding* 115.1 (Jan. 2011).

[112] D. Koller et al. Introduction to Statistical Relational Learning. MIT press, 2007.

[113] H. S. Koppula, R. Gupta, and A. Saxena. Learning Human Activities and Object Affordances from RGB-D Videos. In: *The International Journal of Robotics Research (IJRR)* 32.8 (July 1, 2013). CAD120 dataset, CAD 120, pp. 951–970.

[114] H. S. Koppula and A. Saxena. Anticipating Human Activities using Object Affordances for Reactive Robotic Response. In: *Robotics: Science and Systems*. 2013.

[115] I. Krasin et al. OpenImages: A Public Dataset for Large-scale Multi-label and Multi-class Image Classification. In: *Dataset available from https://storage.googleapis.com/openimages/web/index.html* (2017).

[116] R. Krishna et al. Visual Genome: Connecting Language and Vision using Crowdsourced Dense Image Annotations. In: *International Journal of Computer Vision (IJCV)* 123.1 (2017), pp. 32–73.

[117] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In: *Advances in Neural Information Processing Systems (NIPS)*. 2012, pp. 1097–1105.

[118] O. Kroemer and J. Peters. A Flexible Hybrid Framework for Modeling Complex Manipulation Tasks. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2011, pp. 1856–1861.

[119] N. Krüger et al. Object–Action Complexes: Grounded Abstractions of Sensory–motor Processes. In: *Robotics and Autonomous Systems (RAS)* 59.10 (2011), pp. 740–757.

[120] H. Kuehne et al. HMDB: a Large Video Database for Human Motion Recognition. In: *IEEE International Conference on Computer Vision (ICCV)*. 2011.

[121] KUKA AG. LWR. accessed 08.04.2019.

[122] T. Kulvicius et al. Joining Movement Sequences: Modified Dynamic Movement Primitives for Robotics Applications Exemplified on Handwriting. In: *IEEE Transactions on Robotics* 28.1 (Feb. 2012), pp. 145–157.

[123] T. Lan, T.-C. Chen, and S. Savarese. A Hierarchical Representation for Future Action Prediction. In: *European Conference on Computer Vision*. Springer. 2014, pp. 689–704.

[124] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[125] J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. In: *arXiv preprint arXiv:1607.06450* (2016).

[126] D. B. Lenat. CYC: A Large-scale Investment in Knowledge Infrastructure. In: *Communications of the ACM* 38.11 (1995), pp. 33–38.

[127] S. Levine et al. End-to-end Training of Deep Visuomotor Policies. In: *Journal of Machine Learning Research (JMLR)* 17.1 (2016), pp. 1334–1373.

[128] O. Levy and Y. Goldberg. Neural Word Embedding as Implicit Matrix Factorization. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014.

[129] M. A. Lewis, H.-K. Lee, and A. Patla. Foot Placement Selection using Non-geometric Visual Properties. In: *The International Journal of Robotics Research (IJRR)* 24.7 (2005), pp. 553–561.

[130] T.-Y. Lin et al. Microsoft COCO: Common Objects in Context. English. In: *European Conference on Computer Vision (ECCV)*. Ed. by D. Fleet et al. Vol. 8693. Springer International Publishing, 2014.

[131] T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN Models for Fine-grained Visual Recognition. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1449–1457.

[132] H. Liu and P. Singh. ConceptNet—a Practical Commonsense Reasoning Tool-kit. In: *BT technology journal* 22.4 (2004), pp. 211–226.

[133] L. Lobo, M. Heras-Escribano, and D. Travieso. The History and Philosophy of Ecological Psychology. In: *Frontiers in Psychology* 9 (2018), p. 2228.

[134] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015.

[135] D. G. Lowe. Object Recognition from Local Scale-invariant Features. In: *IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. Ieee, 1999, pp. 1150–1157.

[136] T. Lüddecke, T. Kulvicius, and F. Wörgötter. Context-based Affordance Segmentation from 2D Images for Robot Action. In: *Robotics and Autonomous Systems (RAS)* (2019).

[137] T. Lüddecke and F. Wörgötter. Learning to Segment Affordances. In: *IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 769–776.

[138] K. Lund and C. Burgess. Producing High-dimensional Semantic Spaces from Lexical Co-occurrence. In: *Behavior Research Methods, Instruments and Computers* 28.2 (1996), pp. 203–208.

[139] D. Mahajan et al. Exploring the Limits of Weakly Supervised Pretraining. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 181–196.

[140] P. Mandik. Action-oriented representation. In: *Cognition and the brain: The philosophy and neuroscience movement* (2005), pp. 284–305.

[141] N. Mayer et al. What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation? In: *International Journal of Computer Vision (IJCV)* 126.9 (Sept. 2018), pp. 942–960.

[142] P. Mettes, D. C. Koelma, and C. G. Snoek. The Imagenet Shuffle: Reorganized Pre-training for Video Event Detection. In: *ACM International Conference on Multimedia Retrieval*. ACM. 2016, pp. 175–182.

[143] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting Similarities Among Languages for Machine Translation. In: *ArXiv Preprint ArXiv:1309.4168* (2013).

[144] T. Mikolov et al. Distributed Representations of Words and Phrases and Their Compositionality. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013, pp. 3111–3119.

[145] G. A. Miller. WordNet: a Lexical Database for English. In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[146] H. Min et al. Affordance Research in Developmental Robotics: A Survey. In: *IEEE Transactions on Cognitive and Developmental Systems* 8 (2016).

[147] V. Mnih et al. Human-level Control Through Deep Reinforcement Learning. In: *Nature* 518.7540 (2015), p. 529.

[148] B. Moldovan et al. Learning Relational Affordance Models for Robots in Multi-object Manipulation Tasks. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2012, pp. 4373–4378.

[149] R. Mottaghi et al. Newtonian Scene Understanding: Unfolding the Dynamics of Objects in Static Images. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 3521–3529.

[150] K. P. Murphy. Machine Learning: A Probabilistic Perspective. The MIT Press, 2012.

[151] A. Myers et al. Affordance Detection of Tool Parts from Geometric Features. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015.

[152] A. T. L. Nguyen et al. Detecting Object Affordances with Convolutional Neural Networks. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 2765–2770.

[153] A. Nguyen et al. Object-based Affordances Detection with Convolutional Neural Networks and Dense Conditional Random Fields. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 5908–5915.

[154] M. Nickel et al. A Review of Relational Machine Learning for Knowledge Graphs. In: *Proceedings of the IEEE* 104.1 (Jan. 2016), pp. 11–33.

[155] M. A. Nielsen. Neural Networks and Deep Learning. Determination Press, 2015.

[156] J. B. Oliva, B. Póczos, and J. G. Schneider. The Statistical Recurrent Unit. In: *International Conference on Machine Learning (ICML)*. 2017.

[157] J. Papon and M. Schoeler. Semantic Pose using Deep Networks Trained on Synthetic RGB-D. In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 774–782.

[158] A. Paszke et al. Automatic Differentiation in PyTorch. In: *Advances in Neural Information Processing Systems Workshops*. 2017.

[159] D. Pathak et al. Curiosity-driven Exploration by Self-supervised Prediction. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2017, pp. 16–17.

[160] G. Patterson and J. Hays. COCO Attributes: Attributes for People, Animals, and Objects. In: *Computer Vision – ECCV 2016*. Ed. by B. Leibe et al. Cham: Springer International Publishing, 2016, pp. 85–100.

[161] J. Pennington, R. Socher, and C. D. Manning. Glove: Global Vectors for Word Representation. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2014).

[162] M. E. Peters et al. Deep Contextualized Word Representations. In: *Conference of the North American Chapter of the ACL (NAACL)*. 2018.

[163] J. C. Peterson, J. T. Abbott, and T. L. Griffiths. Adapting Deep Network Features to Capture Psychological Representations. In: *Proceedings of the 38th Annual Conference of the Cognitive Science Society* (2016).

[164] L. Pigou et al. Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video. In: *International Journal of Computer Vision (IJCV)* 126.2-4 (2018), pp. 430–439.

[165] P. O. Pinheiro et al. Learning to Refine Object Segments. In: *European Conference on Computer Vision (ECCV)*. 2016.

[166] L. Pinto et al. The Curious Robot: Learning Visual Representations via Physical Interactions. In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 3–18.

[167] S. Pirk et al. Understanding and Exploiting Object Interaction Landscapes. In: *ACM Transactions on Graphics (TOG)* 36.3 (2017), p. 31.

[168] S. Qi et al. Learning Human-Object Interactions by Graph Parsing Neural Networks. In: *European Conference on Computer Vision (ECCV)*. 2018.

[169] N. Rhinehart and K. M. Kitani. Learning Action Maps of Large Environments via First-person Vision. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 580–588.

[170] M. Richardson and P. Domingos. Markov Logic Networks. In: *Machine learning* 62.1-2 (2006), pp. 107–136.

[171] E. Rietveld and J. Kiverstein. A Rich Landscape of Affordances. In: *Ecological Psychology* 26.4 (2014), pp. 325–352.

[172] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015.

[173] A. Roy and S. Todorovic. A Multi-scale CNN for Affordance Segmentation in RGB Images. In: *European Conference on Computer Vision (ECCV)*. Springer. 2016.

[174] S. Ruder. An Overview of Gradient Descent Optimization Algorithms. In: *arXiv preprint arXiv:1609.04747* (2016).

[175] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Representations by Back-propagating Errors. In: *Nature* (1986).

[176] O. Russakovsky et al. Imagenet Large Scale Visual Recognition Challenge. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[177] B. C. Russell et al. LabelMe: A Database and Web-based Tool for Image Annotation. In: *International Journal of Computer Vision (IJCV)* 77.1-3 (2008).

[178] F. S. Saleh et al. Effective Use of Synthetic Data for Urban Scene Semantic Segmentation. In: *European Conference on Computer Vision (ECCV)*. 2018.

[179] A. Santoro et al. A Simple Neural Network Module for Relational Reasoning. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 4967–4976.

[180] M. Savva et al. SceneGrok: Inferring Action Maps in 3D Environments. In: *ACM Transactions on Graphics (TOG)* 33.6 (2014).

[181] J. Sawatzky, M. Garbade, and J. Gall. Ex Paucis Plura: Learning Affordance Segmentation from Very Few Examples. In: *German Conference on Pattern Recognition (GCPR)*. 2018.

[182] J. Sawatzky, A. Srikantha, and J. Gall. Weakly Supervised Affordance Detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017.

[183] A. Saxena, J. Driemeyer, and A. Y. Ng. Robotic Grasping of Novel Objects using Vision. In: *The International Journal of Robotics Research (IJRR)* 27.2 (2008), pp. 157–173.

[184] Schunk. SDH Hand. accessed 08.04.2019.

[185] A. K. Seth. The Ecology of Action Selection: Insights from Artificial Life. In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 362.1485 (2007), pp. 1545–1558.

[186] N. Silberman et al. Indoor Segmentation and Support Inference from RGBD Images. In: *European Conference on Computer Vision*. Springer. 2012, pp. 746–760.

[187] D. Silver et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. In: *Nature* 529.7587 (2016), p. 484.

[188] D. Silver et al. Mastering the Game of Go Without Human Knowledge. In: *Nature* 550.7676 (2017), p. 354.

[189] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: *arXiv preprint arXiv:1409.1556* (2014).

[190] J. Sivic and A. Zisserman. Video Google: A Text Retrieval Approach to Object Matching in Videos. In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE. 2003, pp. 1470–1477.

[191] R. Socher et al. Reasoning With Neural Tensor Networks For Knowledge Base Completion. In: *Advances in Neural Information Processing Systems (NIPS)*. 2013.

[192] R. Socher et al. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2013, pp. 1631–1642.

[193] D. Song et al. Predicting Human Intention in Visual Observations of Hand/object Interactions. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2013.

[194] D. Song et al. Task-based Robot Grasp Planning using Probabilistic Inference. In: *IEEE transactions on robotics* 31.3 (2015), pp. 546–561.

[195] K. Soomro, A. R. Zamir, and M. Shah. UCF101: A Dataset of 101 Human Actions Classes from Videos in the Wild. In: *arXiv preprint arXiv:1212.0402* (2012).

[196] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised Learning of Video Representations using Lstms. In: *International Conference on Machine Learning (ICML)*. 2015, pp. 843–852.

[197] M. Stark et al. Functional Object Class Detection Based on Learned Affordance Cues. In: *International Conference on Computer Vision Systems (ICVS)*. Springer. 2008.

[198] A. Stoytchev. Behavior-grounded Representation of Tool Affordances. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2005, pp. 3060–3065.

[199] A. Stoytchev. Robot Tool Behavior: A Developmental Approach to Autonomous Tool Use. PhD thesis. Georgia Institute of Technology, 2007.

[200] J. Sun et al. Learning Visual Object Categories for Robot Affordance Prediction. In: *The International Journal of Robotics Research (IJRR)* 29.2-3 (2010), pp. 174–197.

[201] C. Szegedy et al. Going Deeper with Convolutions. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 1–9.

[202] C. Szegedy et al. Inception-v4, Inception-resnet and the Impact of Residual Connections on Learning. In: *AAAI Conference on Artificial Intelligence (AAAI)*. Vol. 4. 2017.

[203] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the Gradient by a Running Average of Its Recent Magnitude. In: *COURSERA: Neural Networks for Machine Learning* 4.2 (2012).

[204] V. Tikhanoff et al. Exploring Affordances and Tool Use on the ICub. In: *IEEE-RAS International Conference on Humanoid Robots (Humanoids)*. IEEE. 2013, pp. 130–137.

[205] J. Tobin et al. Domain randomization for transferring deep neural networks from simulation to the real world. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[206] T. H. Trinh et al. Learning Longer-term Dependencies in RNNs with Auxiliary Losses. In: *International Conference on Machine Learning (ICML)*. 2018.

[207] J. Turian, L. Ratinov, and Y. Bengio. Word Representations: a Simple and General Method for Semi-supervised Learning. In: *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2010.

[208] P. D. Turney and P. Pantel. From Frequency to Meaning: Vector Space Models of Semantics. In: *Journal of Artificial Intelligence Research* 37.1 (2010).

[209] M. T. Turvey. Affordances and Prospective Control: An Outline of the Ontology. In: *Ecological psychology* 4.3 (1992), pp. 173–187.

[210] A. Ückermann et al. 3D Scene Segmentation for Autonomous Robot Grasping. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Oct. 2012, pp. 1734–1740.

[211] E. Ugur, E. Oztop, and E. Sahin. Goal Emulation and Planning in Perceptual Space using Learned Affordances. In: *Robotics and Autonomous Systems (RAS)* 59.7-8 (2011), pp. 580–595.

[212] E. Ugur and J. Piater. Bottom-up Learning of Object Categories, Action Effects and Logical Rules: From Continuous Manipulative Exploration to Symbolic Planning. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2015, pp. 2627–2633.

[213] E. Ugur and J. Piater. Emergent Structuring of Interdependent Affordance Learning Tasks. In: *IEEE International Conferences on Development and Learning and Epigenetic Robotics*. IEEE. 2014, pp. 489–494.

[214] E. Uğur and E. Şahin. Traversability: A Case Study for Learning and Perceiving Affordances in Robots. In: *Adaptive Behavior* 18.3-4 (2010), pp. 258–284.

[215] E. Ugur et al. The Learning and Use of Traversability Affordance using Range Images on a Mobile Robot. In: *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2007, pp. 1721–1726.

[216] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep Image Prior. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 9446–9454.

[217] K. F. Uyanik et al. Learning Social Affordances and using Them for Planning. In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 35. 35. 2013.

[218] G. Varol, I. Laptev, and C. Schmid. Long-Term Temporal Convolutions for Action Recognition. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 40 (2018), pp. 1510–1517.

[219] A. Vaswani et al. Attention Is All You Need. In: *Advances in Neural Information Processing Systems (NIPS)*. 2017, pp. 5998–6008.

[220] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating Visual Representations from Unlabeled Video. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[221] C. Vondrick et al. Predicting Motivations of Actions by Leveraging Text. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[222] C. Vondrick et al. Tracking Emerges by Colorizing Videos. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 391–408.

[223] T.-H. Vu et al. Predicting Actions from Static Scenes. In: *European Conference on Computer Vision (ECCV)*. Springer. 2014.

[224] J. B. Wagman, S. E. Caputo, and T. A. Stoffregen. Hierarchical Nesting of Affordances in a Tool Use Task. In: *Journal of Experimental Psychology: Human Perception and Performance* 42.10 (2016), p. 1627.

[225] J. Walker et al. An Uncertain Future: Forecasting from Static Images using Variational Autoencoders. In: *European Conference Computer Vision (ECCV)*. 2016.

[226] J. Walker et al. The Pose Knows: Video Forecasting by Generating Pose Futures. In: *IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 3352–3361.

[227] C. Wang, K. V. Hindriks, and R. Babuska. Robot Learning and Use of Affordances in Goal-directed Tasks. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2013, pp. 2288–2294.

[228] L. Wang et al. Temporal Segment Networks: Towards Good Practices for Deep Action Recognition. In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 20–36.

[229] X. Wang, R. Girdhar, and A. Gupta. Binge Watching: Scaling Affordance Learning from Sitcoms. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[230] A. Warnecke, T. Lüddecke, and F. Wörgötter. Convolutional Neural Networks for Movement Prediction in Videos. In: *German Conference on Pattern Recognition (GCPR)*. Springer. 2017, pp. 215–225.

[231] W. H. Warren. Perceiving Affordances: Visual Guidance of Stair Climbing. In: *Journal of Experimental Psychology: Human Perception and Performance* 10.5 (1984), p. 683.

[232] J. Weston, S. Bengio, and N. Usunier. Large Scale Image Annotation: Learning to rank With joint Word-image Embeddings. In: *Machine Learning* 81.1 (Oct. 2010).

[233] F. Wörgötter et al. A Simple Ontology of Manipulation Actions Based on Hand-Object Relations. In: *IEEE Transactions on Autonomous Mental Development* 5.2 (June 2013), pp. 117–134.

[234] F. Wörgötter et al. Cognitive Agents—a Procedural Perspective Relying on the Predictability of Object-Action-Complexes (OACs). In: *Robotics and Autonomous Systems (RAS)* 57.4 (2009), pp. 420–432.

[235] Y. Wu and K. He. Group Normalization. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 3–19.

[236] S. Xie et al. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. In: *European Conference on Computer Vision (ECCV)*. Sept. 2018.

[237] C. Xiong, V. Zhong, and R. Socher. Dynamic Coattention Networks for Question Answering. In: *International Conference on Learning Representations (ICLR)* (2017).

[238] W. Xiong et al. The Microsoft 2017 Conversational Speech Recognition System. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Apr. 2018, pp. 5934–5938.

[239] C. Ye et al. What Can I Do Around Here? Deep Functional Scene Understanding for Cognitive Robots. In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2017.

[240] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In: *International Conference on Learning Representations (ICLR)*. 2016.

[241] J. Yue-Hei Ng et al. Beyond Short Snippets: Deep Networks for Video Classification. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 4694–4702.

[242] O. Yürüten, E. Şahin, and S. Kalkan. The Learning of Adjectives and Nouns from Affordance and Appearance Features. In: *Adaptive Behavior* 21.6 (2013), pp. 437–451.

[243] A. R. Zamir et al. Taskonomy: Disentangling Task Transfer Learning. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2018.

[244] P. Zech et al. Action Representations in Robotics: A Taxonomy and Systematic Classification. In: *The International Journal of Robotics Research (IJRR)* (2019).

[245] P. Zech et al. Computational Models of Affordance in Robotics: A Taxonomy and Systematic Classification. In: *Adaptive Behavior* 25.5 (Sept. 2017), pp. 235–271.

[246] H. Zhao et al. Pyramid Scene Parsing Network. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[247] B. Zhou et al. Learning Deep Features for Scene Recognition using Places Database. In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 487–495.

[248] B. Zhou et al. Semantic Understanding of Scenes Through the Ade20k Dataset. In: *arXiv preprint arXiv:1608.05442* (2016).

[249] B. Zhou et al. Temporal Relational Reasoning in Videos. In: *European Conference on Computer Vision (ECCV)*. 2018, pp. 803–818.

[250] Y. Zhu, A. Fathi, and L. Fei-Fei. Reasoning About Object Affordances in a Knowledge Base Representation. In: *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 408–424.

[251] F. Ziaeetabar et al. Recognition and Prediction of Manipulation Actions using Enriched Semantic Event Chains. In: *Robotics and Autonomous Systems (RAS)* 110 (2018), pp. 173–188.

[252] M. Zolfaghari, K. Singh, and T. Brox. ECO: Efficient Convolutional Network for Online Video Understanding. In: *European Conference on Computer Vision (ECCV)*. 2018.