Computer Aided Medical Procedures
Prof. Dr. Nassir Navab

Dissertation

# Learning under Ambiguity through Multiple Predictions

Christian Ernst Joachim Rupprecht

Fakultät für Informatik
Technische Universität München

Technische Universität München
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin

# Learning under Ambiguity through Multiple Predictions

Christian Ernst Joachim Rupprecht

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

| | |
|---|---|
| *Vorsitzende(r):* | Prof. Dr. Matthias Nießner |
| *Prüfer der Dissertation:* | 1. Prof. Dr. Nassir Navab |
| | 2. Prof. Gregory D. Hager, Ph.D.<br>Johns Hopkins University, USA<br>Department of Computer Science |

Die Dissertation wurde am 28.06.2018 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 23.11.2018 angenommen.

**Christian Ernst Joachim Rupprecht**
*Learning under Ambiguity*
*through Multiple Predictions*
Dissertation, Version 1.0.1

**Technische Universität München**
Fakultät für Informatik
Lehrstuhl für Informatikanwendungen in der Medizin
Boltzmannstraße 3
85748 and Garching bei München

# Abstract

Nearly all real-world image understanding problems in computer vision are inherently ambiguous. Often, predictive systems do not model this ambiguity and do not consider the possibility that there can be more than just a single outcome for a given problem. This leads to sub-par performance on ambiguous tasks as the model has to account for all possibilities with one answer. We define three typical sources of *confusion* that render tasks not optimally solvable with a single unique prediction.

First, the fact that we are using 2D images means that the input to our models is a non-reversible projection of the 3D real world. Thus, information about sizes, distances and shapes is lost and has to be inferred by the algorithm from cues such as shading, texture or relative size. Sometimes the ambiguities cannot be resolved by the image context alone. For example, objects of interest can be occluded or too small and do not provide sufficient details to accurately solve a task. The second source of ambiguity are the annotations in our datasets. In supervised or semi-supervised learning scenarios, predictive systems are typically trained using labeled data. However, labeling large amounts of data is tedious and expensive. Thus, often not everything is fully annotated or the annotations can be wrong and inconsistent if they come from unreliable sources, such as non-experts, crowd-sourcing or automatically from the internet. Finally, and most importantly, the problem itself could be ambiguous. For example, predicting the future is inherently uncertain and, in most cases, it cannot be precisely foreseen.

In this dissertation we describe two principled and general approaches of dealing with ambiguity. First, we elaborate on a method that allows the algorithm to predict multiple instead of one single answer. This is a pragmatic way of dealing with ambiguity: instead of deciding for an exclusive outcome for a given problem, we enable the system to list several possibilities. This approach has already proven to be useful in several exciting downstream applications. We show that the general framework of multiple hypothesis prediction can be used for image classification, segmentation, future prediction, conditional image generation, reinforcement learning, object pose estimation and robotic grasp point estimation.

The second part describes an alternative way to deal with uncertain predictions. Often human perception can provide additional information about a task or application that an intelligent system might have not recognized. Building on the paradigm of human-machine interaction, we show how enabling interaction between the system and a user can improve predictions on the example of semantic segmentation. We describe a novel *guiding* mechanism that can be seamlessly integrated into the system and shows great potential beyond the demonstrated tasks for several further applications.

# Zusammenfassung

Im Bereich des Bildverstehens sind fast alle Problemstellungen inhärent mehrdeutig. Viele Methoden modellieren diese Uneindeutigkeit nicht explizit und können deshalb nur schwer mit Szenarien umgehen, in denen es mehr als ein mögliches Ergebnis gibt. Das liegt daran, dass in diesen Systemen alle Möglichkeiten in einer einzigen Antwort zusammengemischt werden müssen. Wir definieren drei Quellen von Mehrdeutigkeit.

Da Fotos eine zweidimensionale Projektion der echten Welt sind, können wir als erste Quelle die Uneindeutigkeit von Bildern bestimmen. Durch die Projektion gehen Informationen wie Größe, Distanzen und Formen verloren und müssen von Algorithmen aus Textur, Schatten und relativer Größe abgeschätzt werden. Verdeckungen oder Perspektive können es auch unmöglich machen ein Problem direkt zu lösen. Da die meisten Methoden des maschinellen Lernens mit annotierten Daten trainiert werden, ist die zweite Quelle definiert durch Fehler oder unzureichend beschriebene Goldstandards definiert. Dies liegt oft darin begründet, dass Annotationen teuer zu erstellen sind, oder automatisierte Systeme, wie zum Beispiel eine Websuche, fehlerbehaftet sind. Letztlich kann auch die Problemstellung selbst uneindeutig sein. Beispielweise die Vorhersage der Zukunft ist inhärent nicht eindeutig lösbar.

In dieser Dissertation beschreiben wir zwei Möglichkeiten mit dieser Uneindeutigkeit umzugehen. Die erste Methode erlaubt es dem Algorithmus statt einer einzigen Antwort mehrere Möglichkeiten vorzuschlagen. Dieser Ansatz hat sich in mehreren nachgelagerten Anwendungen als sehr nützlich erwiesen. Für Bilderkennung, Segmentierung, Zukunftsvorhersage, Bildersterstellung, verstärkendes Lernen, Objektorientierungserkennung und Greifpunktbestimmung in der Robotik können die Vorteile dieser Methode zeigen.

Im zweiten Teil bringen wir den Nutzer des Systems ins Spiel. Oft verfügt ein Mensch über zusätzliche Informationen und kann dem System helfen die richtige Antwort zu finden. In zwei Ansätzen können wir zeigen, wie der Nutzer ins System integriert werden kann. Auch diese Methode hat weitreichende Anwendungsmöglichkeiten und zeigt sich vielversprechend für viele wichtige Probleme.

# Acknowledgments

This dissertation and the work contained in it would not have been possible without the help of many amazing friends and collaborators at TUM and JHU. I would like to thank my advisors Greg Hager and Nassir Navab for supporting me, enabling many interesting collaborations and at the same time granting me great freedom to experiment and explore different fields. Further, I would like to thank Federico Tombari for fruitful discussions, teaching me proposal writing, and guidance. Maximilian Baust had to answer many math questions and was never tired of helping, explaining and advising. Iro Laina is not only my partner in crime for many publications (who makes things work, has the right ideas and knows the literature) but also my partner in life (who always loves and supports me). Loïc Peter, Robert DiPietro and Nicola Rieke were invaluable for their feedback, discussions, help and support during the last years. During my time at MILA in Montréal Chris Pal, Alexandre Piché and Simon Ramstedt enabled me to venture into the field of Reinforcement Learning. There are many more students, colleges and friends in all three institutions, industry and academia that made it a pleasure to do exciting research, develop new ideas, eat amazing food and travel to exciting places. One person that is guilty of all of the above is Salvatore Virga. Very special thanks go to Martina Hilla for making impossible thing possible and for always being happy to talk.

My family and Iro were my grounding on which this work became possible. I am unbelievably thankful for their enduring support, advice and love.

Thank you all.

# Contents

# Part I

Introduction

# 1 Learning in an Ambiguous World

## 1.1 Introduction

In this dissertation we investigate a group of problems that we call *ambiguous problems*, which are prominent in many image understanding tasks. The basic concept is that in many cases it is not fully clear what the correct answer to a given problem is. Humans experience this in the same way. However, when we do not know the exact answer, we often have an intuition about the range of possibilities. For example, when we see a dog playing in the field but it is too far away to identify it exactly. We could say that "*This could be a golden retriever or a labrador, but definitely not a chihuahua.*" This is a typical example where distance, perspective or just lack of expertise in dog-breeds creates an ambiguous prediction problem. We note that even when we are not sure, we have expressed some options that are more likely than others. In the above sentence, it is also implicitly clear that the speaker does not think that this could be a giraffe or an elephant.

This thesis will show three principled ways of dealing with ambiguity. The first two will follow the intuition from the aforementioned example and allow the system to express several guesses – or as we call them *hypotheses*. As many cases of ambiguity are caused by too little information the methods in the last part of the thesis will acquire additional, external knowledge to solve the task by interacting with a human user. As long as the systems that we build are not perfect humans act as excellent sources of information. We will also see that users can sometimes even be the source of ambiguity, as the algorithm needs to understand the operator's goals and expectations. In the case of several possible actions, it is often better to ask before a potentially wrong decision is made.

In the following, we will introduce three typical sources of ambiguity for many problems in computer vision and mention our contribution in these areas.

### 1.1.1 Ambiguous Images

A first source of ambiguity is of course the input image itself. A 2D image is the projection of the 3D world onto the image plane. This projection is non-invertible and thus inherently creates ambiguities when we try to reason about the 3D scene. In [99] we investigate this problem and try to infer the 3D point cloud of a scene just from just a single input image.

Often in medical imaging it is hard to reason about object boundaries and region labels. Thus in [154] we rely on user interaction to segment the left and right ventricle of the heart by learning the deformation field of an active contour algorithm with a convolutional neural network.

We redefine the task of joint localization of surgical tools as a heatmap prediction problem, to combat learning the average of two possible locations in [98]. Still in the medical domain, estimating the kidney volume is also an ambiguous task, if the kidney should actually remain within the patient – which is usually desired – during the measurement. Thus, in an extensive study [168], we can estimate the total kidney volume by using segmentation on CT volumes. For robotic tasks such as (human) activity recognition, often additional sensors such as accelerometers are used. In [160] we learn to predict the sensor readings just from an image in order to substitute the actual sensors with the learned values. This can save cost and effort for many downstream applications.

### 1.1.2 Ambiguous Labels

Even if the input to the algorithm contains all necessary information, ambiguity can arise from the annotations. Labeling datasets is a tedious and often expensive process. For this reason, often not all possible outcomes are annotated in a dataset. In the domain of robotic grasping, there are often many possible poses in which an object can be held. A pen for example can be held almost anywhere along its length and is additionally rotational symmetric, so there are infinitely many possibilities. The dataset we use in our grasping approach [45] has annotated some proposals but not all of them. When an image contains several objects, sometimes only some or only one of them are annotated [158]. Due to labeling cost and effort, it would be desired to obtain labels from the internet, but often when automatically downloading images or videos from online sources, the rate of erroneous labels is high. Thus, in [159] we proposed methods to detect outliers and filter training data from online sources.

### 1.1.3 Ambiguous Problems

Estimating the pose of a symmetric object is inherently ambiguous around the axis of symmetry. Our corresponding work [123] technically falls into all three categories. First, in almost all datasets only one ground truth pose is annotated for an object (ambiguous labels). Second, sometimes self occlusions can also cause unclear scenarios – for example when the handle of a cup is not visible, an arc of possible poses emerges (ambiguous images). Finally, symmetric objects do not have one single correct pose (ambiguous problems). Another problem, which is inherently unpredictable is estimating the future [158]. When learning a policy for behavior, it is often the case that not only one action is best at a given moment. To mitigate this issue, in [96] we learn a stochastic policy that allows predicting several equally valued options in ambiguous scenarios

We have now seen that there are countless scenarios in which ambiguity plays a major role during learning and inference. Before we introduce the main contributions and topics of this thesis, we build a foundation including the basics of neural networks and supervised learning that are common within the remainder of this thesis.

## 1.2 Learning

In this section we briefly introduce neural networks. As they have become an integral part of the computer vision community, we will explain the basics but focus on current techniques. Nonetheless, we will cover all necessary fundamentals to understand this thesis. For an excellent detailed description of deep learning we refer to the book of Goodfellow *et al.* [50].

### 1.2.1 Neural Networks

Neural Networks have a long history - in computer science terms. Theoretical models have been proposed as early as the 1940s [126]. The first notable contribution goes back to Frank Rosenblatt who invented *The Perceptron* in 1957 and is quoted to have envisioned it as "*the embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence*" [137]. The preceptron was build as a mechanical device, where motors controlled potentiometers which can be seen as the free parameters of the network. The input consisted of an array of $20 \times 20$ photo-diodes, rendering it also the first neural network for image recognition. In hindsight we know that the function it learned was a linear function $f$ of the form

$$\tilde{y} = f(x) = Wx + b \; , \tag{1.1}$$

where in this case $W \in \mathbb{R}^{400\times 1}$ denote the parameters and $b \in \mathbb{R}$ the bias. $f$ predicts a single value $f(x) \in \mathbb{R}$, given the input $x$. If $f(x) = 1$ the image contains a target object, if $f(x) = 0$ it does not.

On this simple example we will introduce the concept of supervised learning. As usual in supervised learning we are given a dataset of $N$ samples $(x_i, y_i)$ where $x_i \in \mathbb{R}^{d_x}$, $y_i \in \mathbb{R}^{d_y}$ and $i \in \{1, \dots, N\}$. We are now interested in learning a function $f_\theta(x) = \tilde{y}$ from these examples that minimizes the error we make on the training examples $(x_i, y_i)$. The function $f$ is parametrized by $\theta$ that contains all learnable parameters. Learning is then the process of finding an optimal $\theta^*$ with minimal error or loss $\mathcal{L}$:

$$\theta^* = \operatorname*{argmin}_\theta \sum_{i=1}^{N} \mathcal{L}(f(x_i), y_i) \; . \tag{1.2}$$

A typical loss for regression problems is the $\ell_2$ loss, where

$$\ell_2(\tilde{y}, y) = \frac{1}{2} \|\tilde{y} - y\|_2^2 \tag{1.3}$$

is the squared Euclidean norm of the difference between prediction and ground truth. However, a plethora of generic and task-specific loss functions has been proposed. As long as the loss is differentiable, with respect to the network's output, training with stochastic gradient descent is possible.

Due to their linear nature, it became clear in 1969 [130] that perceptrons cannot learn non-linear functions such as the XOR-function. This result was part of the cause for a sudden disinterest in neural networks.

The first renaissance of neural networks came 1986 with the invention of back-propagation [153] which is an efficient way of computing the derivative of the loss $\mathcal{L}$ with respect to the parameters. To understand back-propagation we will first introduce the basic notation of a multi-layer preceptron. The first key insight over the perceptron was the idea to include a *non-linearity* $\phi : \mathbb{R} \to \mathbb{R}$ in the network, adapting the original formulation to

$$f_\theta(x) = W_2\phi(W_1 x + b_1) + b_2 \ , \tag{1.4}$$

where again $W_1$ and $W_2$ are matrices and $b_1$ and $b_2$ are vectors with $\theta = (W_1, b_1, W_2, b_2)$. Initially, a typical choice for $\phi$ was a sigmoid function such as $\tanh$ or $\frac{e^x}{1+e^x}$. Later it was found that $\max(0, x)$ (ReLu [47, 58]) is not only faster to compute, but also partially helps to mitigate the *vanishing gradient* problem. While many other (minor) variants [27, 88, 120, 134] have been proposed since then, ReLU remains the most widely used activation function to date.

Coming back to back-propagation, we are now interested in computing the gradient of the loss with respect to the parameters $\theta$. For this simple example we will use the network from Equation (1.4) and the $\ell_2$ loss.

$$\frac{\partial}{\partial \theta}\mathcal{L} = \frac{\partial}{\partial \theta}\sum_{i=1}^{N} \|W_2\phi(W_1 x + b_1) + b_2\|_2^2 \tag{1.5}$$

We can decompose this function into the *layers* of the network by defining $f(x) = g(h(x))$ with $h(x) := \phi(W_1 x + b_1)$ and $g(x) := W_2 x + b_2$. The composite function we want to optimize is then $\ell_2(g(h(x)), y)$. The "trick" of back-propagation is the observation that some terms are shared between parameters. For example the derivative with respect to $W_1$,

$$\frac{\partial}{\partial W_1}\mathcal{L} = \frac{\partial \ell_2}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial W_1} \ , \tag{1.6}$$

shares the first two terms with the derivative with respect to $b_1$,

$$\frac{\partial}{\partial b_1}\mathcal{L} = \frac{\partial \ell_2}{\partial g}\frac{\partial g}{\partial h}\frac{\partial h}{\partial b_1} \ . \tag{1.7}$$

This means that back-propagation is simply a smart way to compute all partial derivatives of the loss with respect to the network's parameters. Due to the nature of the chain rule, all derivatives will share the derivative of the loss $\frac{\partial \ell_2}{\partial g}$ which leads to the reason why it is called *back*-propagation. A reasonable order of computation is to compute the most used term first and work backwards to the beginning of the network. Naturally, gradients of parameters of the first layers depend on the derivatives of the later layers.

Due to the number of parameters in a network often being in the millions, it is infeasible to train with second order methods such as BFGS [15, 39, 48, 167] or its variants. Storing and computing/approximating the Hessian matrix exceeds current (GPU) memory limits by far. Thus, currently all training methods are almost exclusively variants of gradient descent:

$$\theta^{(t+1)} = \theta^{(t)} + \lambda \frac{\partial \mathcal{L}}{\partial \theta} , \tag{1.8}$$

where the step-size $\lambda$ is usually called the *learning rate* and $\theta^{(0)}$ is randomly initialized. Since the training dataset usually consists of many samples $N$ and every sample is needed to compute the full gradient of $\mathcal{L}$ (1.5), even gradient descent is too slow. Thus, *stochastic* gradient descent estimates $\frac{\partial \mathcal{L}}{\partial \theta}$ from a small sub-set of randomly selected samples only. The size of this sub-set is called the *batch-size*.

Further, several improvements over vanilla stochastic gradient are used in practice. The most prominent being *momentum* [143, 162, 175] and the more sophisticated, adaptive optimizer Adam [86]. An overview can be found in [152].

Currently, evidence is surfacing that the stochasticity of the gradient estimate actually helps to find better minima during optimization [25, 197] acting as a regularizer for the surface of the loss function.

### 1.2.2 Neural Network Building Blocks

In this section we will briefly discuss typical building block of modern, deep neural networks.

#### Convolutions

Convolutional Neural Networks (CNNs) [101] have been shown to be flexible function approximators and have been used extensively for a wide variety of tasks, such as image classification [59, 94], object detection [149] and semantic segmentation [23, 117].

The main intuition for CNNs is that images are inherently structured. Thus, instead of multiplying the whole image with a full, unconstrained matrix $W$ as in the previous section, we can save parameters by exploiting the structure of the image. This is done using convolutions. A convolution is a filter (set of weights) that is spatially restricted (typically to $3 \times 3$ pixels). This filter is applied to every location in the image or output of the previous layer. This means that all parameters are shared spatially, resulting in a massive reduction in number of weights and easy parallelization on GPUs. This also allows us to recognize similar patterns in different locations, as the filter response does not depend on its location in the image.

### Pooling

In a CNN pooling layers [101] are a form of subsampling; they are usually added in-between convolutions in order to gradually reduce the spatial resolution of feature maps. Pooling layers operate on a single feature map channel at a time and downsample it using a $\max$ or mean operation within a specified window. In the backward pass, mean-pooling is straight forward. Since the $\max$ operation is not differentiable, it transfers the corresponding gradient to the input location where the maximum value was found.

### Batch Normalization

Batch normalization [71] is currently the most popular normalization technique. Normalization after convolutional layers was first used in [94] (Local Response Normalization) to combat vanishing gradient problems. [59] shows that it becomes possible to train very deep networks using batch normalization and residual connections. Batch normalization is used to normalize the activations to zero mean and unit standard deviation for every neuron. Special care has to be taken to estimate these moments during training across multiple batches. Usually a exponential moving average is continuously updated during training.

### Dropout

Dropout [65, 173] was first used in AlexNet [94]. It has been introduced in order to reduce over-fitting a model to the training data. The idea is to randomly set neuron activations to zero, usually with $0.5$ probability. These neurons are then not included in the forward or backward pass. It helps the network learn features without relying on fixed neuron arrangements, since for each input only a random set of neurons contributes to the final outcome. Dropout layers usually appear towards the end of a network, following (hidden) fully-connected layers.

### Residual Blocks

Residual connections were introduced by [59] to enable better gradient flow inside the network. The idea is to change the function each block inside a network computes from $h(x)$ to $x + h(x)$ by adding a skip connection around the block. This allows the network to learn what has to change in $x$ (the residual) indeed of the whole mapping. In their work they show that they are able to learn very deep networks ($> 1000$ layers). $h(x)$ typically consists of three convolutions where the first and the last have a filter size of $1 \times 1$ and reduce/increase the number of channels respectively. This helps to reduce the number of parameters of the $3 \times 3$ filter in the middle.

All of the networks that we use or build in the following sections of the paper contain most of these blocks. Our work on depth prediction makes special use of the residual idea while the guiding block in Section IV is inspired by batch normalization.

# Contributions 2

This thesis is build from the following contributions. In chapter II we present the core concept of multiple hypothesis prediction (MHP):

- **Rupprecht, Christian**, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D. Hager. "*Learning in an uncertain world: Representing ambiguity through multiple hypotheses.*" In International Conference on Computer Vision (ICCV). 2017.

This framework allows a system to predict multiple instead of a single prediction to model the ambiguities inherent in a task. Based on this very general idea, in the same chapter we show several extensions that have been published, are currently under review and/or were developed in a master's thesis with some excellent students:

- Kumar, Sanjeev, **Christian Rupprecht**, Federico Tombari, Gregory D. Hager, "*Predicting Multiple Actions for Stochastic Continuous Control.*", 2018 (under submission).

- Ignacio Sarasúa, "*Adversarial Multiple Hypothesis Prediction*." Master's thesis, advisors: **Christian Rupprecht**, Federico Tombari, Nassir Navab, 2017

- Ghazaei, Ghazal, Iro Laina, **Christian Rupprecht**, Federico Tombari, Nassir Navab, Kianoush Nazarpou, "*Dealing with Ambiguity in Robotic Grasping via Multiple Predictions*", Asian Conference on Computer Vision (ACCV), 2018.

- Manhardt, Fabian, Diego Martín Arroyo, **Christian Rupprecht**, Benjamin Busam, Tolga Birdal, Nassir Navab, Federico Tombari, "*Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data.*" International Conference on Computer Vision (ICCV), 2019 (*equal contribution).

In chapter III we present a novel idea that was inspired by multiple hypothesis prediction. It extends the multiple hypothesis idea to the continuous domain and thus can get rid of a hyper parameter.

In chapter IV we investigate how ambiguity can be resolved by involving a human into the loop. The chapter is encompassed by two fundamentally different concepts for human interaction:

- **Rupprecht, Christian**, Loic Peter, and Nassir Navab. "*Image segmentation in twenty questions.*" IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

- **Rupprecht, Christian***, Iro Laina*, Nassir Navab, Gregory D. Hager, and Federico Tombari. "*Guide Me: Interacting with Deep Networks.*" IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR - spotlight), 2018 (*equal contribution).

In the last part of this chapter we demonstrate an elegant link between the two methods that highlights several connections. Further ambiguous problems (and solutions) that we have worked on and that are mentioned in this thesis are:

- Laina, Iro*, **Christian Rupprecht***, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. "*Deeper depth prediction with fully convolutional residual networks.*", International Conference on 3D Vision (3DV), 2016 (*equal contribution).

- **Rupprecht, Christian***, Colin Lea*, Federico Tombari, Nassir Navab, and Gregory D. Hager. "*Sensor substitution for video-based action recognition.*" IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016 (*equal contribution).

- **Rupprecht, Christian**, Ansh Kapil, Nan Liu, Lamberto Ballan, and Federico Tombari. "*Learning without prejudice: Avoiding bias in webly-supervised action recognition.*" Computer Vision and Image Understanding (CVIU), 2017.

- **Rupprecht, Christian**, Elizabeth Huaroc, Maximilian Baust, and Nassir Navab. "*Deep active contours.*" arXiv preprint arXiv:1607.05074, 2016 (under submission).

- Laina, Iro*, Nicola Rieke*, **Christian Rupprecht**, Josué Page Vizcaíno, Abouzar Eslami, Federico Tombari, and Nassir Navab. "*Concurrent segmentation and localization for tracking of surgical instruments.*" International Conference on Medical Image Computing and Computer-Assisted Intervention, (MICCAI) 2017 (*equal contribution).

- Sharma, Kanishka, **Christian Rupprecht**, Anna Caroli, Maria Carolina Aparicio, Andrea Remuzzi, Maximilian Baust, and Nassir Navab. "**Automatic segmentation of kidneys using deep learning for total kidney volume quantification in autosomal dominant polycystic kidney disease.**" Nature Scientific reports 7. 2017.

- DiPietro, Robert, **Christian Rupprecht**, Nassir Navab, and Gregory D. Hager. "*Analyzing and Exploiting NARX Recurrent Neural Networks for Long-Term Dependencies.*" International Conference on Learning Representations - Workshop Track (ICLR-W). 2018.

- Spier, Nathalia, Rischpler Christoph, **Christian Rupprecht**, Nassir Navab, Maximilian Baust, and Stephan Nekolla. "*Defect detection in cardiac SPECT using graph-based convolutional neural networks.*" Journal of Nuclear Medicine, 2018.

Although many of the methods and ideas presented in this dissertation have been published, all of them are extended with ideas and results that were developed after the original publications. Very prominently you will find hints like this throughout this document:

**Hint: Hints**

For all presented concepts we have added hints that are based on typical questions from students or colleagues. The hints explain concepts further or make connections to other related work within this thesis or in the literature.

Usually after working with a specific method for some time, we have developed intuitions that did not find their way into the paper but that we deem valuable to reason about the capabilities of the method.

**Intuition: Intuitions**

Intuitions will give insights into the model that we deem not straightforward. We gained these intuitions while developing the method and usually through a large bandwidth of failed and successful experiments. Our hope is that they give the reader a head start when building upon the presented techniques.

We deem these hints and intuitions extremely valuable as they are usually what we base our decisions and experiments on. With this, we hope to give the reader a much richer picture than the limited space in publications allows.

# Part II

Multiple Hypothesis Prediction

# Introduction 3

In this chapter we investigate an approach that we refer to as *Multiple Hypothesis Prediction* (MHP). This method that we present is generic, as it does not depend on a specific task, but is formulated as a technique to predict multiple possibilities instead of a single answer for any given task. The core method has been published in [157] and applications and extensions will be describes in chapter 5.

## 3.1 Motivation

As we have seen before, when asked to recognize objects in a given image, humans do not only predict a single, definite answer but usually express their uncertainty. A reasonable expression could be: "*this could be an alpaca or a llama, but it is definitely not an elephant*". When predicting the behavior of other drivers on the road a reasonable assumption is "*They are in the right lane, so they might drive straight or take a right turn soon*". This prediction is justified as it is based on the observation of previous behavior of ourselves on the road.

Uncertainty can be used to model situations in which we only observe incomplete information. It is impossible to distinguish a mug from a cup if the handle is hidden on the back side. In fact, we will later demonstrate this behavior for the task of object pose estimation in Section 5.9.

In this chapter we present our *Multiple Hypothesis Prediction* framework that can easily extend typical single-loss, single-output systems to predict multiple outputs. Mathematically, this approximates the conditional output space with a piece-wise constant function instead of a single answer. Our formulation is probabilistic and we can show that its minimizer yields a Voronoi tessellation in the output space with the original task loss defining a metric in this space. Over a large variety of applications we demonstrate how deep neural networks can be modified to predict multiple hypotheses. The method introduces a *meta loss* and can be trained using standard procedures – gradient descent and backpropagation.

We will demonstrate that the MHP framework is general in the sense that it can easily retrofit any CNN architecture and loss function enabling the estimation of multiple hypotheses in a large variety of tasks. Analyzing the variance over different hypotheses provides insights into the model, predictions and task uncertainty. In the experiments we observe that it often also improves performance as the model can express possible outcomes instead of averaging all possibilities into one answer. In regression tasks, for example, a single hypothesis prediction (SHP) model tends to average distinct modes often resulting in blurred and unrealistic predictions. We will demonstrate that MHP models are capable of overcoming this issue.

## 3.2 Related Work

In the following, we discuss in detail the work related to the principle of multiple predictions.

Closely related are Mixture Density Networks (MDNs) [10] that predict a parametrized Gaussian mixture model to handle multimodal regression tasks. We will later show that MDNs differ from our approach in two major ways. Due to the nature of Gaussian mixtures, MDNs are constrained to regression tasks. MHP models, however, are agnostic to the task objective and can be applied to a much wider variety of problems. In addition, we model the output space by an implicit Voronoi tessellation rather than using a mixture of Gaussians, resulting in a vastly different parametrization and flexibility. For example, we find that due to numerical instabilities of high dimensional Gaussian distributions, MDNs can be difficult to train in large output spaces such as images (Section 5.2.1).

The idea of learning several predictors for one task has been previously investigated under the term *Multiple Choice Learning* [31, 102, 103]. [56] build upon this concept by enforcing diversity among the predictions. Closely related is the work of Lee *et al*. [103] who train an ensemble of networks with a minimum formulation that is similar to ours. The concept of multiple hypothesis prediction extends these ideas while simultaneously providing a mathematical understanding why this formulation is beneficial. Further, MHP extends to regression tasks and introduces a relaxation that helps convergence. On the technical side, instead of training separate networks for each of the multiple choices, we use a shared architecture for the hypotheses which saves a considerable amount of parameters and enables information exchange between predictions. In our experiments with MHP-augmented CycleGANs (Section 5.6) we will further study the impact on how parameter sharing can influence the diversity in predictions.

Label ambiguity in different domains is investigated by Gao *et al*. [42] for the tasks of age estimation and image classification. The work studies performance improvement caused by soft, probabilistic class assignments and Kullback-Leibler (KL) divergence, especially in the case of smaller datasets in which this soft assignment provides additional information. For age estimation multi-label approaches have become popular due to the inherent difficulty of guessing someone's age form just a single picture [44]. Also head pose estimation [43] can be modeled as a multi-label task.

Single-label image classification is a simplified recognition task, as in most natural images objects usually appear not alone but along with other objects of different categories. Label dependency can be modeled using a recurrent neural network (RNN) on top of a CNN [190] or by using deep convolutional ranking [49]. Several other work proposes pipelines of object proposals [191] or use ground truth bounding boxes [195] and/or classifiers [201] to predict multiple labels.

Many tasks such as future prediction exhibit in inherent ambiguity. When interacting with robots it is crucial to anticipate what the humans will be doing or where they will move to [90]. The transfer of motion from video to images is studied in [202]. Predicting the future of collapsing toy towers [107] is an interesting task to learn basic physics. Since the future can usually not be predicted precisely, Vondrick *et al*. [186] use estimate several possible future frames form the given current one. Fouhey and Zitnick predict spatio-temporal likelihood distributions [40] for humans in pictures and cartoons, while Walker *et al*. [188] resolve uncertainty using dense trajectories of motion with a variational autoencoder.

With the exception of [10] (and maybe [42] for classification), the aforementioned work is application-driven and specific, with no straightforward translation to other tasks.

Loosely related are methods that estimate prediction confidences together with the task. Gal *et al*. [41] analyze dropout layers as a means to extract uncertainty estimates using multiple forward passes. In Kingma *et al*. [87] a stochastic gradient variational Bayes estimator is used to understand the posterior probability.

The theoretical background of this work is based on the concept of (centroidal) Voronoi tessellations. However, the tessellation emerges implicitly and does not have to be computed. Thus, a detailed knowledge of Voronoi tessellations is not necessary for the application of this method.

**HINT: VORONOI TESSELATIONS**

For a detailed explanation of Voronoi tessellations, we refer the interested reader to the general book of Okabe *et al*. [136] or to Du *et al*. [32] for centroidal tessellations. However, all necessary background used in this work will be explained here.

In the following section, we will explain in detail the background and derivation of our multiple hypothesis prediction approach.

# Mathematical Derivation 4

Here, we describe the proposed ambiguity-aware MHP framework and investigate the relationship of such model to traditional (unambiguous) prediction models. One can deem a standard model as *unambiguous* as it deterministically maps one input to one output. More formally, we term $\mathcal{X}$ as the vector space of input variables and $\mathcal{Y}$ as the corresponding space of outputs or *labels*. As explained before, we are given a set of $N$ training tuples $(x_i, y_i)$, where $i = 1, \dots, N$. Since we are interested in problems for which several possible outcomes could exist, we want to model the joint probability density $p(x, y) = p(y|x)p(x)$ of inputs and outputs. We use $p(y|x)$ to denote the conditional probability for $y$ given $x$.

## 4.1 The Unambiguous Prediction Model

We will first investigate what learning a standard model on an ambiguous task means mathematically. In a typical supervised learning scenario, we are interested in learning a parametrized function $f_\theta : \mathcal{X} \to \mathcal{Y}$, with $\theta \in \mathbb{R}^n$, such that we minimize the expected error

$$\frac{1}{N}\sum_{i=1}^{N} \mathcal{L}(f_\theta(x_i), y_i). \tag{4.1}$$

Further, we assume that the training samples follow the distribution $p(x, y)$. $\mathcal{L}$ stands for any typical loss function, dependent on the task itself. It could for example be the classical $\ell_2$-loss (Equation 1.3).

For a better analysis it is often interesting to investigate the limit where $N \to \infty$. In other terms, for sufficiently large $N$, Equation (4.1) holds as a good approximation of the continuous formulation

$$\int_{\mathcal{X}}\int_{Y} \mathcal{L}(f_\theta(x), y)\, p(x, y)\; dy\; dx, \tag{4.2}$$

In case of $\mathcal{L} = \ell_2$, Equation (4.2) is minimized by the conditional average (see *e.g.* [89])

$$f_\theta(x) = \int_{\mathcal{Y}} y \cdot p(y|x)\; dy. \tag{4.3}$$

The conditional average can be thought of as the mean of all possible $y$ given the current $x$, weighted with the probability $p(y|x)$ of their occurrence.

**INTUITION: CONDITIONAL AVERAGE**

Intuitively this is the optimal solution, for which we expect to make the smallest error. We find the mean of all possible outcomes weighted by their probability.

However, the conditional average can be a poor representation, depending on the complexity of the conditional density $p(y|x)$. Consider a mixture model with two distinct Gaussian modes. Then the expected outcome will fall in between the two means, where the probability density is low. This means that the expected error for the prediction is minimized but the prediction itself is not a likely solution for the problem. This dilemma is tackled by our multiple hypothesis formulation.

## 4.2 The Ambiguous Prediction Model

The unambiguous model approximates the distribution $p(y|x)$ with a single constant $f_\theta(x)$ for every $x$. Naturally, replacing the single outcome with multiple ones should yield a closer approximation of the actual distribution. To better model ambiguity in our system, we present a formulation that produces $M$ predictions instead:

$$f_\theta(x) = (f_\theta^1(x), \ldots, f_\theta^M(x)). \tag{4.4}$$

However, in most tasks, only a single outcome $y^*$ and not the full distribution $p(y|x)$ is usually annotated for each $x$ in a dataset. To learn a predictor as in Equation 4.4, the intuition is that at least one of the predictions $f_\theta^i(x)$ should be close or equal to the annotated $y^*$. Thus, in simple terms, we compute the loss $\mathcal{L}$ always for the single prediction that is closest to the ground truth. Mathematically, we propose to minimize

$$\int_{\mathcal{X}} \sum_{j=1}^{M} \int_{\mathcal{Y}_j(x)} \mathcal{L}(f_\theta^j(x), y) p(x, y)\ dy\ dx, \tag{4.5}$$

instead of (4.2). Let us investigate a Voronoi tessellation of the label space $\mathcal{Y} = \cup_{i=1}^{M} \overline{\mathcal{Y}_i}$. It is induced by $M$ generators $g^j(x)$ and a loss $\mathcal{L}$:

$$\mathcal{Y}_j(x) = \left\{ y \in \mathcal{Y} : \mathcal{L}(g^j(x), y) < \mathcal{L}(g^k(x), y)\ \forall k \neq j \right\}. \tag{4.6}$$

Each Voronoi cell $\mathcal{Y}_j(x)$ contains all points that are closest to its generator, where the loss $\mathcal{L}$ defines the distance between two points as a measure of closeness. The output space is thus divided by (4.5) into $M$ cells generated by the hypotheses $f_\theta^j(x)$.

For a more in depth analysis of this formulation we will consider the typical regression case where $\mathcal{L} = \ell_2$. This view is helpful as then the loss preserves the intuitive geometric understanding of distance in the output space. We can show an interesting property while solidifying intuitions about the formulation.

**Theorem 1 (Minimizer of 4.5)** *A necessary condition for Equation* (4.5) *to be minimal is that the generators $g^j(x)$ are identical to the predictors $f_\theta^j(x)$, and both correspond to a centroidal Voronoi tesselation:*

$$g^j(x) = f_\theta^j(x) = \frac{\int_{\mathcal{Y}_j} \ell_2(f_\theta^j(x), y) p(y|x)\ dy}{\int_{\mathcal{Y}_j} p(y|x)\ dy}, \tag{4.7}$$

*i.e. $f_\theta^j$ predicts the conditional mean of the Voronoi cell it defines*[1].

[1] if the density $p(x, y)$ satisfies mild regularity conditions (*i.e.* it vanishes only on a subset of measure zero)

*Proof.* The first observation is that Equation (4.5) can be minimized point-wise with respect to $x$ since $\ell_2 \geq 0$ and $p(x, y) \geq 0$. We can simply minimize

$$\sum_{j=1}^{M} \int_{\mathcal{Y}_j(x)} \mathcal{L}(f_\theta^j(x), y) p(x, y) \, dy \tag{4.8}$$

for all $x \in \mathcal{X}$ to minimize the original expression. The second equality in Equation (4.7) follows by computing the first variation w.r.t. $f_\theta^j$ as done in [32, Proposition 3.1]:

$$f_\theta^j(x) = \frac{\int_{\mathcal{Y}_j} \mathcal{L}(f_\theta^j(x), y) p(x, y) \, dy}{\int_{\mathcal{Y}_j} p(x, y) \, dy}. \tag{4.9}$$

Using the factorization $p(x, y) = p(y|x)p(x)$ and noting that the integration does not depend on $x$, we pull $p(x)$ out of the integrals and eventually replace $p(x, y)$ by $p(y|x)$ in Equation (4.9).

Secondly, we use contradiction to show that $g^j(x) = f_\theta^j(x)$. Following [32], if we would assume that the generators $g^j(x)$ do not coincide with $f_\theta^j(x)$, it would be possible to find one or more subsets of $\mathcal{Y}$ with non-vanishing measure and thus Equation (4.8) could not be minimal.

**Intuition: What are we optimizing?**

One can think of the method consisting of two components. It finds a set of $M$ generators $f_\theta^j$ that divide the (conditional) output space into cells. Each cell's generator is its conditional average minimizing the expected error within. The overall task becomes finding good cell centers such that the total expected loss is minimal.

## 4.3 Minimization Scheme

After having justified the use of generators $f_\theta^j$ to minimize (4.5), we will now describe how to actually learn them from a set of examples $(x_i, y_i), i \in \{1, \ldots, N\}$. As usual in current computer vision research, we will employ CNNs as general function approximates not only due to their flexibility but also since in all our tasks $\mathcal{X}$ and sometimes $\mathcal{Y}$ will be images.

For a condensed overview, we propose the following algorithm for training neural networks with back-propagation to minimize Equation (4.5) in five (easy) steps.

1. Predict the set of $M$ generators/hypotheses $f_\theta^j(x_i), j \in \{1, \ldots M\}$ for each training sample $(x_i, y_i)$ by a forward pass though the network.

2. The tessellation $\mathcal{Y}_j(x_i)$ of $\mathcal{Y}$ is implicitly built using the generators $f_\theta^j(x_i)$, Equation (4.6) and a loss function $\mathcal{L}$.

3. Compute the gradients for each Voronoi cell $\frac{\partial}{\partial\theta} \frac{1}{|\mathcal{Y}_j|} \sum_{y_i \in \mathcal{Y}_j} \mathcal{L}(f_\theta^j(x_i), y_i)$, where $|\mathcal{Y}_j|$ denotes the cardinality of $\mathcal{Y}_i$.

4. Perform an optimizer step of $f_\theta^j(x_i)$ using the gradients per hypothesis $j$ from step 3.

5. If convergence is reached: terminate. Otherwise continue with step 1.

We can implement this algorithm using a *meta-loss* $\mathcal{M}$ derived from Equation (4.5).

$$\mathcal{M}(f_\theta(x_i), y_i) = \sum_{j=1}^{M} \delta(y_i \in \mathcal{Y}_j(x_i)) \mathcal{L}(f_\theta^j(x_i), y_i). \tag{4.10}$$

We deem the term meta-loss fitting, as $\mathcal{M}$ encapsulates any task loss $\mathcal{L}$ within itself. In the formulation we use the Kronecker delta $\delta$ which returns $1$ when its condition is met and $0$ otherwise, to compute the loss only for the best hypothesis $f_\theta^j(x_i)$ for the label $y_i$.

**Hint: Lloyd's Method**

There are different connections that can be drawn to other algorithms used in different fields. For example it can be seen as an extension of Lloyd's Method [116] to be trained by gradient descent via back-propagation.

Transforming an existing network into a MHP model is as simple as to replicate the output layer $M$ times (with different initializations). Each prediction $f_\theta^j$ is compared to the ground truth label using the original loss $\mathcal{L}$ but activated or deactivated by $\delta$ (Equation (4.10)).

**Hint: Expectation Maximization**

Another link can be made to Expectation Maximization (EM) methods. Analogous to an E-step, we first find which of the predictions $f_\theta^j(x_i)$ best matches the annotation $y_i$. Then, the parameters $\theta$ of the predictor are updated for the matching prediction only - this can be seen as the M-step.

In practice, we observe that training with a hard $\delta$ is only stable if the initial starting hypothesis distribution is already close to the ground truth.

To understand this instability, let us analyze what happens if all generators $f_\theta^j(x)$ are initialized far away from the ground truth labels $\mathcal{Y}$. We will consider an initialization that is shifted away in one direction from the target distribution. In a one-dimensional example, one could assume that all labels lie somewhere in $[-1, 1]$ but if the predictor (network) is initialized in the range $[-1+\tau, 1+\tau]$ and $\tau > 2$ then all labels lie to the left of the generators. That means that no matter which training sample was used, we would always update only the left most $f_\theta^j$ while all others are starved of gradient steps.

**Intuition: Hypothesis Starvation**

In higher dimensions the convergence problem can be understood as all ground truth labels $\mathcal{Y}$ lying in the same, single Voronoi cell $\mathcal{Y}_k$. This has the effect that only this cell's generator would be updated during training, since $\delta(y_i \in \mathcal{Y}_j(x_i)) = 0, \ \forall j \neq k$, degenerating the model to an unambiguous model.

However a simple solution mitigates this problem. We propose to relax the hard assignment $\delta$ using a weight $0 < \epsilon < 1$:

$$\hat{\delta}(a) = \begin{cases} 1-\epsilon & \text{if } a \text{ is true,} \\ \frac{\epsilon}{M-1} & \text{else.} \end{cases} \tag{4.11}$$

The annotation $y_i$ is now softly assigned to the closest generator $f_\theta^k(x)$ using a weight of $1-\epsilon$ and with $\frac{\epsilon}{M-1}$ to all remaining generators.

The formulation fulfills $\sum_{j=1}^{M} \hat{\delta}(y_i \in \mathcal{Y}_j(x_i)) = 1$ keeping the overall weight of one sample at $1$. In our experiments we found that dropping out [173] full predictions with some low probability (1% in our experiments) acts as an additional regularizer. Hypothesis dropout ensures, that sometimes the best generator will not be available and the second closest will receive the corresponding update, with a similar effect as (4.11). Now, even in the case of a bad initialization, all hypotheses will be updated, gradually moving them towards the range of the target distribution.

**HINT: ALTERNATIVE META-LOSS FORMULATION**

Although implementing (4.10) and (4.11) in current deep learning frameworks is possible (but potentially convoluted), we found a (implementation wise) more elegant and mathematically equivalent formulation.

$$\mathcal{M}(f_\theta(x_i), y_i) = \phi \min_k \mathcal{L}(f_\theta^k(x_i), y_i) + \psi \sum_{j=1}^{M} \mathcal{L}(f_\theta^j(x_i), y_i). \tag{4.12}$$

Where we set the constants $\phi = 1 - \frac{M\epsilon}{M-1}$ and $\psi = \frac{\epsilon}{M-1}$ to normalize the loss in accordance to (4.11). We notice that $\epsilon \to 0$ results in $\phi \to 1$ and $\psi \to 0$ which is identical to the unrelaxed formulation in (4.10).

It is important to note that our meta-loss $\mathcal{M}$ is agnostic to the choice of internal task-loss $\mathcal{L}$ and the parametrization and structure (architecture) of $f$. We demonstrate this on a wide variety of tasks and CNN architectures in Section 5.

**HINT: $M$-MEANS CLUSTERING**

There is a distinct difference to $k$-means (or in this case $M$) clustering. In this method we are approximating the *conditional* distribution $p(y|x)$ which naturally depends on the input $x$ and changes together with it. Clustering $\mathcal{Y}$ would not only be independent of $x$, but also much less powerful as the conditional representation.

The number of hypotheses $M$ is a hyper-parameter, but we do not see any deterioration of the performance when overestimating $M$ in all regression problems. In fact, almost every method modeling posterior probabilities depends on a manually tuned, additional parameter: for example $k$ in $k$-means ($k$) and the number of Gaussians $m$ in MDNs [10].

# Applications

# 5

In this section, we will first show extensive experiments to validate different properties of multiple hypothesis prediction and then move to further additional applications [45, 96, 123] that we built upon the original publication [157].

In the first part we will demonstrate the following properties.

1. With the simplicity of a 2D toy example, we show a visual representation of the implicit Voronoi tessellation (Section 5.1).

2. On the example of human pose estimation *as a proxy for real-world low-dimensional regression problems* we highlight the underlying information obtained by analyzing the variance across hypotheses (Section 5.2).

3. We demonstrate the generalization to high-dimensional problems for the task of future frame-prediction and observe that the predicted images become sharper with more predictions (Section 5.4).

4. The applicability to discrete problems such as multi-label image classification and segmentation is demonstrated (Section 5.3 and 5.5).

**Hint: SHP vs. MHP**

In most of our experiments we follow the same approach to build a multiple hypothesis (MHP) model. We start from an existing method/network that uses the traditional unambiguous (single hypothesis prediction – SHP) approach and modify it for multiple hypothesis prediction. This means that in all cases a direct baseline is always the original SHP model.

We mainly chose to use straightforward, single-stage models to study the behavior and evaluate multiple hypothesis prediction. Although, complex multi-stage pipelines could benefit both SHP and MHP models and likely improve their performance, their often long chain of modules would obscure the analysis of the raw MHP framework. Thus, for every task we build upon previously proposed CNN architectures [9, 59, 99, 117] by modifying them to predict multiple hypotheses. A single NVIDIA TitanX with 12GB GPU memory has been used for all experiments. Since in most cases the main body of the network is unmodified, the influence of the number of predictions $M$ on training and inference time is negligible. It affects only the last layer of the network where we separate the individual predictions and thus impacts the overall execution time only marginally. In all primary experiments we set the association relaxation to $\epsilon = 0.05$, in later experiments we found that decaying $\epsilon \to 0$ can further increase the performance (Section 5.9). When we denote our model as $M$-MHP, we refer to an architecture

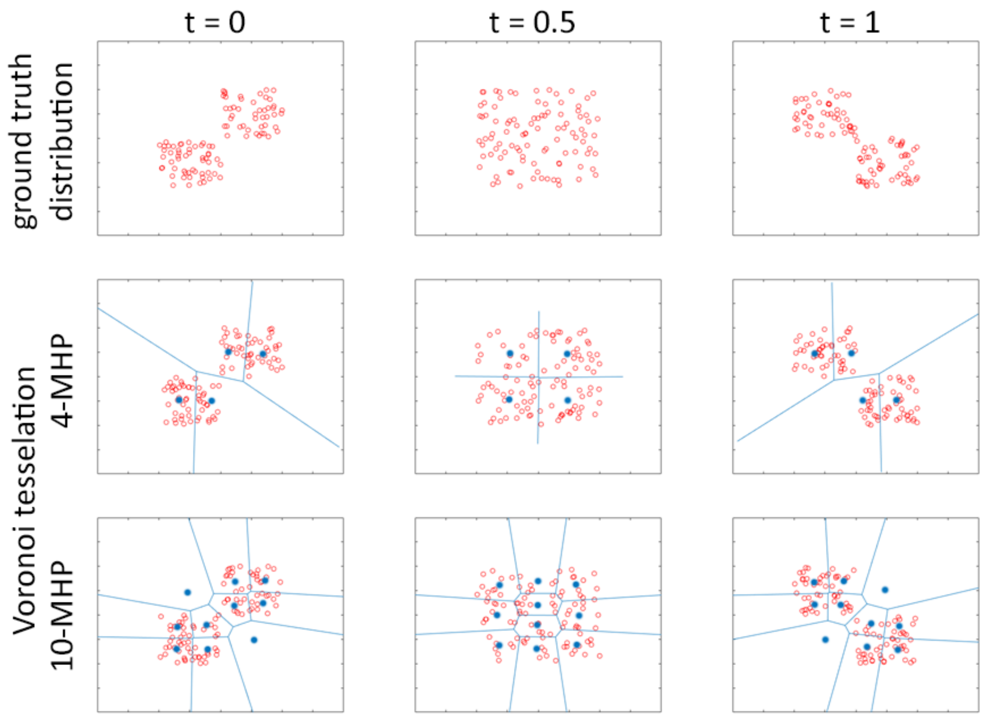


Figure 5.1. **Temporal 2D Distribution Illustration.** Red points are drawn from the true underlying distribution, blue points show predictions and blue lines highlight the resulting Voronoi regions.

trained to predict $M$ hypotheses. As previously stated the corresponding, unmodified original single prediction (unambiguous) model is termed SHP.

## 5.1 Toy Examples

To visually demonstrate the type of representation that is build by an MHP model, we utilize a toy example of a two-dimensional distribution that depends on a time variable $t \in [0, 1]$.

We divide the region of a square centered around $0$ in a 2D-plane into four equal regions, and smoothly transition from having high probability mass in the lower-left and top-right quadrants to high probability mass in the upper-left and lower-right quadrants. At time step $t = \frac{1}{2}$ all points in the whole square have equal probability. More precisely, we divide the 2D plane around $0$ into five regions $S_i$:

$$S_1 = [-1, 0) \times [-1, 0) \subset \mathbb{R}^2, \tag{5.1}$$

$$S_2 = [-1, 0) \times [0, 1] \subset \mathbb{R}^2, \tag{5.2}$$

$$S_3 = [0, 1] \times [-1, 0) \subset \mathbb{R}^2, \tag{5.3}$$

$$S_4 = [0, 1] \times [0, 1] \subset \mathbb{R}^2, \tag{5.4}$$

$$S_5 = \mathbb{R}^2 \backslash \{S_1 \cup S_2 \cup S_3 \cup S_4\}. \tag{5.5}$$

We then define a time-dependent distribution, by first conditioning the probability for a region $S_i$ to be selected onto the time $t$.

$$p(S_1) = p(S_4) = \frac{1-t}{2}, \tag{5.6}$$
$$p(S_2) = p(S_3) = \frac{t}{2} \tag{5.7}$$

The outside of the square will never be selected, thus $p(S_5) = 0$. When a region is chosen, a point is uniformly sampled from within its area. This distribution can be seen in the first row of Fig. 5.1 as it smoothly transitions between all three states.

The task is now to predict the 2D coordinate of a sample dependent on the time $t$. Since the distribution is always symmetric, an unambiguous model would learn to predict the center of the probability density, which remains at $(0, 0)$ at all times. This is an unsatisfying result as the model does not predict useful information approximating a dynamic, time-dependent distribution with a single constant. The MHP network is a two layer MLP with 50 neuron in both hidden layers and is trained with $\ell_2$-loss as objective. The Voronoi tessellation for 4-MHP and 10-MHP is shown in the bottom two rows of Figure 5.1. Contrary to the SHP model, the multiple hypotheses of the MHP model allow to divide the space into Voronoi cells that match the regions and vary with $t$. When increasing the number of hypotheses the tessellation becomes more dense.

In the following sections we demonstrate further properties of MHP on real-world tasks.

## 5.2 Human Pose Estimation

In this experiment we increase the complexity of the task to image input and 24-dimensional human skeleton joint potions as output. 2D human pose estimation requires regressing the pixel locations of the joints of a human body and usually assumes detection, i.e. the input frame is already cropped around the person.

The goal of this experiment is to demonstrate that MHP generalizes to a robust loss function, but also how additional insights can be obtained from an MHP model. We measure the variation between the predictions estimate the confidence of the model. Here, we adapt our human pose estimation model from [9], that uses Tukey's bi-weight function as a robust estimator for the loss. We evaluate the strict PCP score using an oracle to select the best hypothesis which results in SHP: 59.7%, 2-MHP: 60.0%, 5-MHP: 61.2%, 10-MHP: 62.8% to understand the gain of increasing $M$. When the number of predictions is increased the method is able to model the output space more and more precisely, resulting in a better oracle performance with an increased number of predictions. Very recently [110] show that it is possible to design secondary approaches to select which hypothesis is best to further improve the performance.

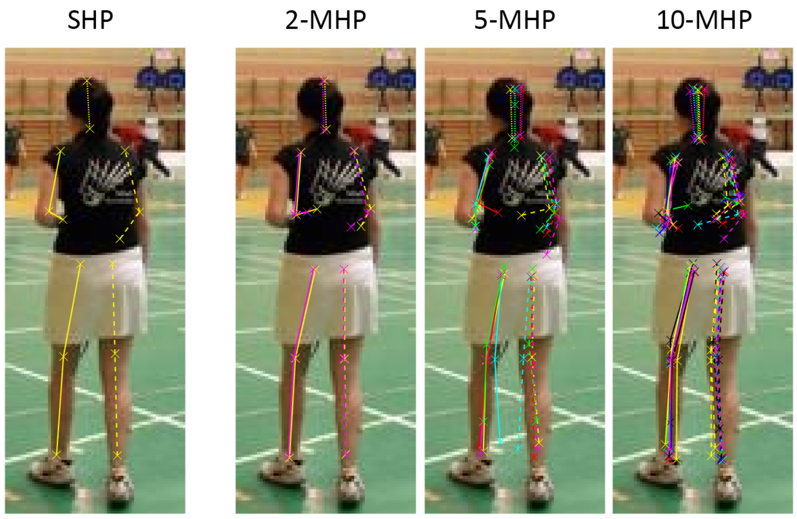


Figure 5.2. **Human Pose Estimation on the LSP dataset.** We show the predicted human pose for an image with SHP and with two, five, and ten MHPs. We observe the uncertainty of the hand positions in the high variance with multiple predictions. Joints like shoulders and hips are easy to detect and also vary much less.

| **body part** | ankle | knee | hip | wrist | elbow | shoulder |
|---|---|---|---|---|---|---|
| dist. visible | 4.8 | 3.0 | 1.9 | 5.0 | 3.1 | 2.3 |
| dist. occl. | 5.9 | 3.7 | 2.4 | 5.1 | 3.3 | 2.6 |

Table 5.1. **Mean joint position variance:** For each joint we compute the mean distance from every hypothesis to the mean prediction. In all cases the mean distance of the predictions for occluded joints is higher than the one for visible joints. This can be used as a confidence measure. The head and neck joint were not regarded since less than 10 samples were occluded.

**Intuition: Learning an Oracle**

In theory it should not be possible to learn an oracle to select the best hypothesis. Since the oracle only has access to the hypotheses and the original input, it does not have more information as the original model $f$ – both operate on $\mathcal{X}$ or direct derivatives $f(\mathcal{X})$ of it. Thus, if some hypotheses were less likely than others the MHP model should learn this directly and not depend on an oracle to decide. In practice however, it is possible to imagine that the performance improvement comes from the increased capacity of training an additional model and from non-perfect generalization to unseen data.

In Figure 5.2 we show qualitative results for human pose estimation for different $M$. It becomes clear that the variance per joint of the predictions of the occluded joints (both wrists) is higher than the variance of directly visible joints like the shoulder or the hips.

In the Leeds Sports Pose dataset [79] it is annotated whether a joint is visible or occluded. This allows us to compute mean distances of joint positions to the mean predicted skeleton for occluded and visible joints which resembles a variance estimation. In Table 5.1 we observe that this variation is correlated with the occlusion of joints (higher for occluded joints), hinting a connection between variance and model uncertainty. A further insight is that the variance

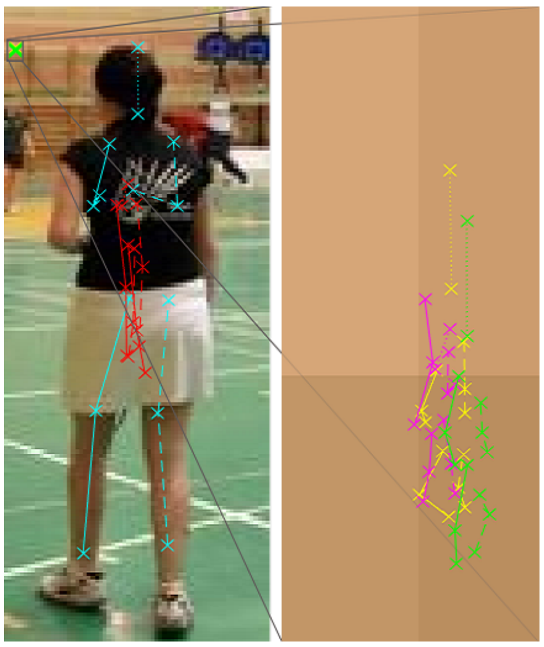


Figure 5.3. **MDNs for human pose estimation.** Mixture Density Networks become numerically unstable in higher dimensions, while at the same time suffer from degenerate predictions. The mixing coefficients for the degenerate predictions in the top left are almost 0, which lets all of their gradients vanish.

for the extremities (hands, feet) is higher than for more stable points like hips and shoulders, thus reflecting the relative difficulty of predicting these joints.

### 5.2.1 Comparison to Mixture Density Networks

As previously mentioned, in contrast to our point-wise constant approximation of the posterior density, one can explicitly estimate it using a parametrized model such as mixture density networks (MDNs) [10]. Two main points render MDNs fundamentally different than our MHP method. From the perspective of the output of the model, MDNs estimate the full density $p(y|x)$, while our MHP model predicts multiple hypotheses instead. Second, MDNs —and Gaussian mixture models— are only well-defined for regression problems. The MHP meta loss is task agnostic and can be applied to any problem.

For comparison we modified our architecture to be a MDN for human pose estimation. Although, $14\times2$D joints can be considered a low dimensional problem, we found it challenging to train a stable MDN. One reason for this is that the Gaussians contain exponents with the number of dimensions ($c$ in Eq. 23 in [10]), which causes severe numerical problems. In fact, we had to resort to RMSProp as optimizer as training the MDN with SGD with momentum resulted in exploding gradients or stagnation. Bishop trains with BGFS in the original paper, which is a second order optimization technique and thus infeasible for deep networks where it is impossible to store a Hessian matrix due to the very large number of parameters.

In Figure 5.3 we show qualitative results of a MDN trained with five Gaussian mixtures for the same image as the MHP cases in Figure 5.2.

| | VOC07 | VOC12 | COCO | COCO |
|---|---|---|---|---|
| **Method/Dataset** | mAP | mAP | mAP | mAP@10 |
| WARP [49] | - | - | - | 49.2 |
| HCP-1000 [195] | 81.5 | - | - | - |
| CNN-RNN [190] | 84.0 | - | - | 61.2 |
| SHP (baseline) | 83.8 | 86.9 | 65.2 | 81.0 |
| 3-MHP (ours) | 84.1 | 87.3 | 66.1 | 82.2 |
| 5-MHP (ours) | 84.7 | 87.5 | **67.8** | **83.3** |
| 9-MHP (ours) | **85.1** | **87.6** | 67.4 | 82.8 |
| 13-MHP (ours) | 84.7 | 87.0 | 67.7 | 83.1 |

Table 5.2. **Results on Pascal VOC 2007, 2012 and MS-COCO:** Classification results improve with more predictions over the single prediction baseline. At 9- and 13-MHP the performance decreases slightly due to false positives in some of the hypotheses as there are often much less true labels. (Results for [49] from [190])

The MDN predicts probability density of 98% for the cyan skeleton, $1.9\%$ for red and almost $0\%$ for the remaining three (degenerated in top left corner). This sample is representative in the sense that also in any other frame MDNs do not recover more than one reasonable hypothesis.

**Hint: Gradients in Mixture Density Networks**

One reason for degenerate modes in mixture density networks is that all gradients for MDNs contain a multiplicative factor ($\alpha_i$ in [10]) for each mixture $i$ that prevents the model from learning a mean and variance for this component once its $\alpha_i$ is close to 0.

Conceptionally, mixture density networks have a clear advantage as they are predicting a full density, probabilities and variances. However, in our experiments they are significantly more difficult to train and suffer from severe numerical instabilities caused by high dimensional multivariate Gaussian distributions. MHPs are able to handle high dimensional problems without any stability issues due to the simplistic formulation of the meta-loss that does not require exponential functions.

## 5.3 Multi-label Image Classification

For multi-label classification it is often argued that single-label CNN models are not suitable and multi-stage methods such as R-CNN [46] and it extensions are proposed. We can show that extending a standard image classification CNN architecture with MHP can achieve competitive performance, without the complexity of a multi-stage pipeline. Here we pre-train a ResNet-101 [59] on ImageNet data, replace the final layer with one that predicts a set of $C$ class confidences for $M$ hypotheses ($C \cdot M$ values/neurons in total).

The idea is to reformulate the multi-label image classification task as an ambiguous problem where it is not clear which of the objects in the image give it its label. Every image is assigned a probabilistic label, uniformly sampled from all annotated objects that exist in the image not taking into account multiple instances. For example, an image that contains a plant and two birds will be labeled *plant* 50% of the time and *bird* $50\%$ of the time.

**HINT: COUNTING INSTANCES WITH MHP**

The choice of not taking the number of objects per class into account is due to the evaluation metric which also does not consider multiple instances of the same object. However, it would be possible to select the label based on the relative number of objects per class in the image. However, since an MHP model always predicts $M$ hypotheses, one would obtain the relative amount of objects from the network, i.e. *33% of all objects are birds* and not absolute results.

Our evaluation uses the 2007 and 2012 renditions of the Pascal Visual Object Classes (VOC) [37] dataset with twenty different classes ($C = 20$). Similar to other methods, we train the network(s) on the *train*-set of VOC2012 and evaluate performance using VOC2012 *val*- and VOC2007 *test*-splits. A much more diverse dataset is the Microsoft Common Objects in Context (COCO) [115] dataset, which contains $C = 80$ classes, 82,783 training images and 40,504 validation images. Since MSCOCO also contains annotations for small objects such as spoons and fruit, the variance in number of objects per image is much higher than for PascalVOC.

Table 5.2 lists a condensed version of the multi-label recognition results and compares them to three other methods. The typical metric here are the mean average precision (mAP) and mAP@10.

**HINT: MEAN AVERAGE PRECISION AT $K$**

The mAP@$K$ metric computes the mAP for the $K$ classes that were detected with the highest confidence and can be seen as the multi-class analogous of the Top-$N$ performance for image classification.

Here we do not make use of an oracle selecting the best prediction since this would almost trivialize the problem. Still, in every experiment the MHP model outperforms its SHP baseline.

In discrete problems, we observe a decreased performance at high $M$ (in this case $\geq 9$). In that case, there are often more hypotheses than possible discrete outcomes and the additional hypotheses contribute noise that reduces the scores slightly.

**INTUITION: $M$ AND DISCRETE PROBLEMS**

If $\epsilon$ was $0$ and the number of hypotheses $M$ was equal or greater than the number of classes in the dataset, there exists a trivial solution with perfect meta-loss $\mathcal{M} = 0$. When every class in the dataset is predicted by (at least) one hypothesis, and the minimum formulation is not relaxed ($\epsilon = 0$) then no matter what label the image actually has, the model always predicts a perfect hypothesis. This is a further motivation for the relaxed formulation with $\epsilon > 0$ and hypothesis dropout.

| | | | | | | |
|---|---|---|---|---|---|---|
| SHP | person | person | bicycle | plant | person | train |
| 2-MHP | motorbike, person | dog, person | bicycle, ~~bottle~~ | plant (2x) | person (2x) | train (2x) |
| 3-MHP | car, motorbike, person | dog (2x), person | bicycle, bird, ~~person~~ | plant (3x) | ~~chair~~, person (2x) | train (3x) |
| 5-MHP | car, motorbike, person (3x) | ~~chair~~, dog (2x), person, sofa | bicycle, bird (2x), ~~mbike~~, plant | plant (5x) | ~~chair~~, person (4x) | train (5x) |
| 9-MHP | car, motorbike, person (7x) | ~~chair~~, dog (6x), person, sofa | bicycle (2x), bird (3x), ~~bottle~~, ~~motorbike~~, ~~person~~, plant | plant (9x) | ~~chair~~, ~~table~~, person (5x), ~~sofa~~, ~~tv~~ | person, train (8x) |

Figure 5.4. **Multiple Predictions on VOC 2012.** We show qualitative examples of multiple predictions. For each prediction we select the class with the maximum confidence. Networks with multiple predictions are able to identify several different classes in the images. The last image the ground truth annotation contains the *person* label for the conductor in the train. Incorrect predictions are crossed out.

We can accumulate hypotheses for classification by computing the class with the highest confidence for every prediction. In Figure 5.4 we show qualitative results for varying $M$. Naturally the single hypothesis network predicts only one class per image. We observe that when increasing $M$, the model successfully identifies more and more different objects. In the case where only a single class exists in the image – for example for the plant in Figure 5.4, all hypotheses predict the same answer. Another interesting observation is, that for high $M$ we sometimes obtain false positives – such as the chair and table in the fifth example – however, even erroneous predictions seem reasonable in the sense that they usually come from object categories that fit in the depicted scene.

Additionally we provide detailed results on the MS COCO dataset in Table 5.3 and 5.4. Since often in classification tasks improvements can be marginal, for each row we highlight results which are statistically equivalent to the best performing (bold) model with a green background. In this challenging dataset, which contains on average 3.5 unique classes and 7.7 instances per image, the models with up to three hypotheses are very rarely equivalent to any of the other (higher-$M$) models. Performance usually increases with more hypotheses. This further justifies the relationship between the number of hypotheses and the number of instances in an image and demonstrates the effect of MHP on more cluttered images.

Furthermore, in Fig. 5.5 we provide the precision-recall plots for a representative class of each one of the 12 super-categories in COCO. They are on par with the average precision results of Tables 5.3 & 5.4.

## 5.4 Future Frame Prediction

Interestingly, predicting the future is a task that inherently possesses the properties of a typical multiple hypothesis task. Often the captured scenario is not repeatable with exactly the same starting conditions. This means that only one outcome label can be obtained and not the full distribution of possibilities. This renders future prediction an ideal showcase of MHP. In future

| $M$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **mAP** | 59.10 | 60.69 | 61.92 | 61.64 | 62.89 | 63.30 | 62.74 | 62.68 | 63.32 | 63.07 | 62.39 | 63.06 | **63.60** | 63.46 | 60.47 |
| **person** | 91.22 | 93.90 | 92.47 | 93.57 | 93.97 | 93.91 | 93.31 | 94.13 | 92.97 | 94.18 | 94.05 | 94.08 | 93.45 | **94.80** | 93.69 |
| **bicycle** | 40.62 | 43.34 | 43.53 | 46.14 | 49.56 | 51.32 | 48.50 | 48.67 | 49.92 | 47.82 | 47.80 | **52.32** | 50.61 | 47.20 | 42.59 |
| **car** | 61.34 | 61.90 | 62.38 | 65.90 | 64.60 | 64.80 | 65.26 | 65.52 | **68.12** | 65.40 | 66.96 | 64.42 | 67.45 | 65.79 | 66.02 |
| **motorcycle** | 75.90 | 79.98 | 80.12 | 80.15 | 80.48 | 81.34 | 81.47 | 80.93 | 80.12 | 80.92 | 81.79 | **81.99** | 81.99 | 80.31 | 80.28 |
| **airplane** | 86.41 | 91.60 | 89.93 | 90.78 | 90.53 | 88.31 | 88.02 | 90.58 | **92.64** | 91.31 | 92.14 | 88.99 | 92.33 | 90.59 | 89.78 |
| **bus** | 68.12 | 72.40 | 72.09 | 75.09 | 74.67 | 75.82 | 73.71 | 74.83 | 74.58 | 75.92 | 75.50 | 74.67 | **76.30** | 74.98 | 73.24 |
| **train** | 88.76 | 88.02 | 90.21 | 90.13 | 90.84 | 90.34 | 89.43 | 88.98 | 89.47 | 90.30 | **90.88** | 89.94 | 90.11 | 90.41 | 90.23 |
| **truck** | 51.78 | 56.43 | 50.63 | 55.79 | 54.87 | 55.09 | 55.43 | 57.31 | 55.80 | 55.42 | **58.57** | 57.87 | 56.45 | 56.07 | 54.69 |
| **boat** | 69.92 | 74.07 | 74.12 | 73.93 | 73.59 | 72.58 | 74.10 | 73.39 | 74.47 | 73.66 | 73.02 | 74.32 | **74.92** | 73.90 | 72.82 |
| **traffic light** | 62.72 | 62.91 | 63.87 | 66.43 | **67.75** | 65.45 | 62.80 | 64.21 | 65.48 | 63.93 | 66.60 | 67.60 | 64.83 | 66.16 | 62.11 |
| **fire hydrant** | 64.86 | 64.99 | 64.84 | 65.68 | 68.05 | 67.02 | 70.24 | 66.45 | 69.56 | **70.62** | 66.71 | 68.61 | 69.09 | 68.87 | 67.58 |
| **stop sign** | 67.30 | 67.18 | 67.74 | 65.39 | 67.07 | 68.15 | 68.89 | 65.98 | 70.02 | 66.78 | 68.10 | 68.22 | **70.06** | 69.45 | 67.03 |
| **parking meter** | 42.29 | 41.71 | 44.12 | 46.60 | 50.12 | 49.39 | **51.05** | 48.39 | 48.43 | 47.92 | 48.72 | 45.39 | 50.89 | 47.25 | 42.48 |
| **bench** | 42.07 | 45.69 | 47.11 | 45.35 | 45.67 | 47.56 | 47.24 | 49.09 | 49.03 | 47.96 | 47.99 | 47.99 | 48.85 | **49.72** | 45.34 |
| **bird** | 54.26 | 59.67 | 59.07 | 59.07 | 58.21 | 59.44 | 60.79 | 59.27 | 60.02 | 58.89 | **61.00** | 60.11 | 59.75 | 59.87 | 60.28 |
| **cat** | 77.04 | 81.64 | 80.62 | **84.50** | 81.66 | 83.98 | 81.29 | 79.08 | 81.91 | 80.42 | 82.44 | 81.81 | 81.07 | 82.89 | 83.54 |
| **dog** | 58.34 | 66.22 | 62.93 | 63.49 | 65.31 | 67.15 | 63.85 | 64.66 | 66.09 | 67.02 | 65.80 | **67.46** | 65.07 | 65.96 | 64.99 |
| **horse** | 67.67 | 77.70 | 77.55 | 74.38 | **80.50** | 79.41 | 72.16 | 73.60 | 77.42 | 75.94 | 76.21 | 73.45 | 75.60 | 72.35 | 67.25 |
| **sheep** | 79.03 | 80.35 | 81.71 | 80.82 | **83.19** | 82.83 | 81.83 | 80.54 | 81.60 | 80.53 | 81.59 | 81.41 | 81.93 | 78.14 | 78.72 |
| **cow** | 68.91 | 74.41 | 73.37 | 73.97 | 74.87 | 75.58 | 73.31 | 74.88 | 76.41 | **77.69** | 74.26 | 74.65 | 75.46 | 73.46 | 71.85 |
| **elephant** | 92.16 | 92.55 | 92.22 | 93.19 | 93.40 | 94.29 | 93.85 | 93.51 | 92.58 | **94.30** | 92.31 | 92.78 | 92.87 | 92.79 | 90.86 |
| **bear** | 88.71 | 89.66 | 90.69 | 90.70 | 90.61 | 87.99 | 89.11 | 89.52 | 90.52 | 89.69 | 89.53 | **91.18** | 88.20 | 90.01 | 90.98 |
| **zebra** | 96.12 | 96.56 | 96.99 | 97.15 | 97.54 | 97.44 | 97.29 | 96.86 | 96.74 | 97.44 | 97.42 | 96.80 | 97.47 | 97.26 | **97.55** |
| **giraffe** | 97.41 | 97.67 | 97.45 | 98.12 | 98.19 | 98.07 | 98.00 | 97.71 | 97.87 | 98.16 | **98.65** | 97.64 | 98.25 | 97.72 | 97.93 |
| **backpack** | 24.89 | 26.08 | 25.41 | 25.02 | 26.75 | 27.29 | 26.58 | **28.08** | 26.49 | 26.38 | 27.91 | 27.82 | 26.50 | 26.43 | 24.68 |
| **umbrella** | 60.05 | 61.03 | 62.21 | 64.02 | 64.47 | **66.07** | 64.48 | 64.14 | 63.59 | 64.47 | 65.55 | 65.50 | 65.90 | 62.68 | 61.80 |
| **handbag** | 29.38 | 33.24 | 34.75 | 30.21 | 33.13 | 35.98 | 35.73 | **36.77** | 35.82 | 35.38 | 35.65 | 35.42 | 35.97 | 36.34 | 33.42 |
| **tie** | 60.61 | 62.76 | 64.43 | 65.31 | 65.84 | 67.33 | 65.79 | **68.59** | 67.02 | 68.51 | 67.67 | 63.43 | 66.87 | 66.98 | 65.50 |
| **suitcase** | 43.51 | 41.95 | 47.78 | 45.00 | 48.48 | 48.05 | 49.85 | 50.97 | 48.22 | 46.58 | 49.42 | **51.48** | 49.54 | 50.31 | 45.82 |
| **frisbee** | 66.61 | 71.44 | 72.61 | 70.43 | 71.91 | **74.24** | 70.83 | 72.42 | 71.35 | 73.23 | 69.80 | 70.08 | 71.19 | 70.83 | 62.98 |
| **skis** | 85.06 | 85.35 | 86.04 | 85.90 | **87.16** | 87.02 | 85.78 | 85.08 | 84.10 | 85.29 | 85.24 | 86.93 | 86.40 | 83.76 | 84.63 |
| **snowboard** | 59.18 | 62.84 | 62.20 | 62.36 | 60.41 | **66.79** | 64.63 | 63.33 | 65.83 | 64.09 | 59.83 | 59.21 | 65.33 | 66.35 | 52.72 |
| **sports ball** | 61.63 | 65.60 | 62.04 | 63.21 | 65.85 | 65.42 | 65.01 | 63.82 | 63.95 | **66.01** | 60.37 | 62.17 | 65.42 | 63.84 | 60.40 |
| **kite** | 77.10 | 83.99 | 86.00 | 81.06 | 81.88 | 84.62 | **86.81** | 82.83 | 82.05 | 84.91 | 82.85 | 82.51 | 86.43 | 86.74 | 84.61 |
| **baseball bat** | 74.23 | 78.73 | **81.64** | 79.78 | 79.92 | 81.04 | 79.45 | 78.13 | 77.14 | 77.35 | 73.97 | 80.60 | 78.90 | 77.55 | 71.78 |
| **baseball glove** | 81.38 | 86.92 | 81.13 | 86.04 | 81.53 | 86.16 | **86.97** | 83.75 | 83.55 | 84.03 | 82.70 | 84.21 | 83.37 | 85.99 | 82.95 |
| **skateboard** | 76.71 | 83.20 | 84.75 | 83.45 | 80.98 | 83.34 | **85.59** | 84.93 | 84.67 | 82.48 | 82.72 | 81.02 | 85.23 | 84.87 | 78.89 |
| **surfboard** | 85.38 | 85.94 | **89.05** | 88.09 | 87.13 | 88.40 | 88.22 | 88.08 | 88.20 | 88.29 | 88.48 | 87.30 | 88.62 | 88.07 | 85.23 |

Table 5.3. **Average precision on COCO (Part 1/2).** We report the per class average precision (AP%) and the mean average precision (mAP%) for all MHP models.

frame prediction we are interested in the pixel-wise estimation of a future frame in a video, given one or more previous frames. Naturally, this task is inherently ambiguous.

With this experiment we can demonstrate that MHP models extend to high dimensional problems, since we predict images with a resolution of $128 \times 128 \times 3$ and $256 \times 256 \times 3$. We use our previous fully convolutional residual architecture [99], which we know works well for pixel-wise regression tasks, achieving state-of-the-art results on depth estimation without the need for additional refinement steps (at the time of publication). Adapting the model for MHP is as simple as modifying the number of convolutions in the last layer to $3M$ (for RGB predictions). All encoder weights are initialized with ResNet-50 weights (pre-trained on ImageNet [161] data) and the decoder with random zero-mean Gaussian distributions with 0.01 standard deviation.

| $M$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **tennis racket** | 92.39 | 94.72 | 95.28 | 95.01 | 95.22 | 94.91 | **96.02** | 94.76 | 94.75 | 95.00 | 93.93 | 94.48 | 95.06 | 95.62 | 94.25 |
| **bottle** | 40.48 | 43.37 | 40.86 | 42.03 | 42.41 | 39.70 | 41.91 | 41.36 | 43.61 | 43.10 | 43.70 | **45.83** | 42.79 | 42.69 | 43.61 |
| **wine glass** | 39.05 | 38.05 | 41.84 | 40.04 | 43.78 | **45.38** | 42.32 | 42.43 | 42.24 | 43.02 | 40.60 | 44.18 | 44.11 | 44.34 | 43.48 |
| **cup** | 46.30 | 49.66 | 48.57 | 43.28 | 49.32 | 48.83 | 48.17 | 48.42 | 50.66 | **51.65** | 44.85 | 50.24 | 49.12 | 50.08 | 50.87 |
| **fork** | 42.70 | 42.54 | 40.42 | 40.13 | 43.11 | 43.91 | 41.19 | 39.98 | 42.69 | **46.02** | 42.54 | 41.69 | 42.71 | 43.50 | 39.35 |
| **knife** | 40.09 | 34.78 | 40.21 | 33.81 | 38.38 | 39.72 | 38.12 | 38.22 | 34.55 | **41.26** | 34.21 | 39.65 | 38.04 | 40.12 | 37.42 |
| **spoon** | 30.56 | 29.98 | 34.20 | 32.32 | 30.50 | **36.18** | 31.97 | 32.99 | 34.75 | 34.20 | 34.58 | 33.85 | 35.07 | 34.28 | 35.37 |
| **bowl** | 48.39 | 42.70 | 47.22 | 48.36 | 50.36 | 50.01 | 48.24 | 44.51 | 50.90 | 49.58 | 47.12 | 52.40 | **52.88** | 47.62 | 46.47 |
| **banana** | 56.91 | 61.41 | 61.35 | 61.43 | 64.28 | 61.57 | 65.04 | 60.10 | 63.66 | **65.43** | 59.50 | 65.02 | 62.12 | 63.29 | 59.80 |
| **apple** | 39.41 | 39.95 | 42.44 | 39.62 | 40.68 | 43.00 | 41.94 | 39.38 | 40.51 | 39.34 | 39.33 | **45.06** | 41.46 | 42.61 | 32.89 |
| **sandwich** | 50.50 | 50.11 | 53.58 | 50.73 | 54.46 | 55.36 | 54.22 | 55.75 | 53.04 | 52.74 | 51.76 | 54.25 | 54.69 | **56.66** | 47.23 |
| **orange** | 53.10 | 55.34 | 54.47 | 57.81 | 58.50 | 56.09 | **59.40** | 56.82 | 57.94 | 56.45 | 53.72 | 58.59 | 57.34 | 57.74 | 51.52 |
| **broccoli** | 78.58 | 80.89 | 81.50 | 80.42 | 82.19 | **85.16** | 82.77 | 83.12 | 83.84 | 83.58 | 81.50 | 82.52 | 82.53 | 82.35 | 83.57 |
| **carrot** | 50.77 | 48.95 | 52.69 | 45.85 | 50.50 | **56.19** | 54.78 | 49.28 | 55.26 | 55.41 | 53.63 | 55.19 | 50.47 | 55.02 | 47.64 |
| **hot dog** | 47.72 | 48.53 | 51.65 | 51.54 | 51.34 | 51.80 | 53.12 | 51.09 | 55.77 | **56.36** | 52.05 | 54.89 | 50.23 | 53.32 | 43.48 |
| **pizza** | 83.00 | 78.98 | 81.59 | 79.97 | 83.33 | 84.09 | 82.87 | 83.84 | **85.50** | 83.45 | 82.85 | 84.39 | 85.48 | 84.40 | 81.50 |
| **donut** | 56.94 | 58.52 | 58.10 | 60.02 | **63.75** | 62.61 | 62.05 | 62.10 | 63.50 | 60.01 | 62.82 | 61.96 | 61.22 | 63.53 | 59.59 |
| **cake** | 49.71 | 54.29 | 58.94 | 55.50 | 59.70 | 55.21 | 57.15 | 56.13 | 60.48 | **62.70** | 57.20 | 53.60 | 62.48 | 62.32 | 53.58 |
| **chair** | 49.46 | 47.38 | 52.52 | 47.87 | 49.41 | 51.63 | 53.88 | 52.04 | 54.70 | 51.95 | **55.47** | 54.28 | 54.19 | 55.09 | 52.11 |
| **couch** | 53.71 | 56.20 | 60.76 | 59.26 | **63.80** | 55.54 | 59.25 | 58.17 | 57.27 | 59.19 | 57.50 | 56.46 | 58.34 | 58.96 | 55.59 |
| **potted plant** | 35.32 | 36.05 | 36.29 | 32.83 | **40.83** | 40.09 | 38.96 | 39.42 | 38.45 | 39.70 | 34.25 | 36.80 | 37.44 | 38.66 | 36.07 |
| **bed** | 67.25 | 69.43 | 72.01 | **74.43** | 74.38 | 73.20 | 74.25 | 71.91 | 73.64 | 71.60 | 71.72 | 68.31 | 71.71 | 70.32 | 69.80 |
| **dining table** | 58.74 | 57.16 | 54.88 | 53.60 | 56.34 | 58.62 | 55.28 | 56.52 | 60.39 | 59.02 | 58.50 | 59.93 | 61.26 | **62.72** | 58.82 |
| **toilet** | 87.82 | 90.11 | 90.45 | 90.80 | 91.18 | **91.70** | 90.19 | 90.76 | 90.58 | 90.76 | 89.99 | 89.95 | 90.56 | 91.14 | 89.75 |
| **tv** | 65.61 | 57.62 | 62.33 | 67.28 | 68.75 | 67.11 | 66.82 | 69.18 | 68.98 | 66.24 | 64.22 | 68.27 | 69.19 | **70.42** | 68.34 |
| **laptop** | 60.52 | 56.66 | 66.47 | 65.48 | 65.04 | **69.77** | 66.26 | 66.47 | 68.28 | 65.23 | 66.00 | 63.28 | 65.97 | 67.55 | 65.00 |
| **mouse** | 61.62 | 59.45 | 65.49 | 61.87 | 62.34 | 64.83 | 62.66 | 66.99 | 65.88 | 56.96 | 66.11 | 65.39 | 66.11 | **68.55** | 56.87 |
| **remote** | 43.51 | 47.95 | 49.96 | 49.23 | **53.80** | 48.30 | 45.56 | 53.20 | 46.20 | 51.50 | 49.31 | 48.79 | 52.14 | 53.18 | 45.91 |
| **keyboard** | 64.94 | 64.82 | 66.69 | 69.84 | 67.89 | 66.85 | 68.39 | 71.15 | 69.15 | 65.12 | 68.11 | 70.08 | 70.90 | **71.53** | 67.00 |
| **cell phone** | 37.57 | 38.23 | 40.53 | 39.80 | 43.10 | 43.83 | 43.13 | 43.53 | 43.48 | **45.30** | 42.23 | 43.06 | 43.97 | 43.44 | 41.29 |
| **microwave** | 41.06 | 38.27 | 44.47 | 40.31 | 43.97 | 41.87 | 42.76 | 44.40 | 38.87 | 42.57 | 37.88 | 43.08 | **44.69** | 43.01 | 40.49 |
| **oven** | 61.73 | 61.62 | 64.15 | 64.94 | 62.76 | 65.84 | 63.85 | 65.67 | 63.32 | 64.88 | 62.37 | **68.85** | 65.72 | 67.39 | 61.56 |
| **toaster** | 7.46 | 7.59 | 7.68 | 8.65 | 9.72 | 8.64 | 5.71 | 7.19 | **10.68** | 10.50 | 9.23 | 10.47 | 9.97 | 6.76 | 8.28 |
| **sink** | 71.37 | 68.06 | 71.40 | 70.94 | 73.00 | 72.26 | 72.06 | 71.79 | 72.48 | 72.17 | **76.61** | 74.91 | 74.01 | 74.63 | 73.24 |
| **refrigerator** | 51.58 | 50.82 | 54.99 | 54.39 | 54.30 | 55.03 | 55.92 | 52.16 | 56.13 | 55.32 | 53.60 | 55.11 | 55.81 | **57.07** | 50.12 |
| **book** | 36.60 | 37.31 | 38.48 | 41.54 | 40.73 | 37.43 | 41.67 | **46.28** | 45.41 | 39.87 | 44.60 | 42.00 | 45.50 | 42.77 | 38.77 |
| **clock** | 68.56 | 67.62 | 68.92 | 69.65 | 67.58 | 69.02 | 68.49 | 69.02 | 69.65 | 69.92 | 69.55 | **70.99** | 70.89 | 69.31 | 67.03 |
| **vase** | 57.20 | 58.98 | 57.66 | 58.76 | 61.04 | 59.26 | 60.50 | 61.28 | 60.16 | 60.71 | 60.75 | 61.00 | **61.67** | 58.46 | 56.54 |
| **scissors** | 30.69 | 32.36 | 34.01 | 32.92 | 34.83 | **35.62** | 34.12 | 33.03 | 33.39 | 32.97 | 32.53 | 32.97 | 35.21 | 33.70 | 29.62 |
| **teddy bear** | 62.48 | 67.51 | 68.68 | 68.80 | 69.96 | 67.69 | 67.16 | 66.53 | **73.12** | 69.34 | 69.04 | 67.29 | 69.29 | 69.37 | 67.89 |
| **hair drier** | 2.37 | 3.46 | 3.14 | 1.77 | 2.23 | **6.30** | 3.49 | 5.84 | 2.26 | 4.32 | 2.84 | 3.54 | 2.74 | 4.20 | 2.42 |
| **toothbrush** | 25.69 | 37.92 | 40.93 | 41.83 | 41.51 | 44.49 | 40.89 | 42.83 | **45.42** | 41.53 | 42.60 | 40.56 | 44.16 | 42.42 | 36.28 |

Table 5.4. **Average precision on COCO (Part 2/2).** We report the per class average precision (AP%) and the mean average precision (mAP%) for all MHP models.

### 5.4.1 Intersection

We begin the experimental section for future frame prediction with a dataset created by a simulation where it is easy to control the ambiguity and parameters. Here, we model a top-down view of a four-way intersection and try to predict which way a car is going to take. To this end, we render sequences where a simplified model car approaches the intersection from a random two-way road and slows down at the crossing. The car then picks one of the three possible routes to leave the crossing with equal probability. The task is to predict – from a current frame – the last frame of the sequence, where the car is about to exit the field of view. This dataset combines two types of uncertainty. A discrete uncertainty regarding which exit the car will choose and a continuous uncertainty in the exact pose of the car in the last frame which depends on the speed, acceleration and steering capabilities chosen at random. A sample sequence is shown in Figure 5.6, where the first row shows the single input frame and the target frame is depicted in the second row. The columns represent time stamps

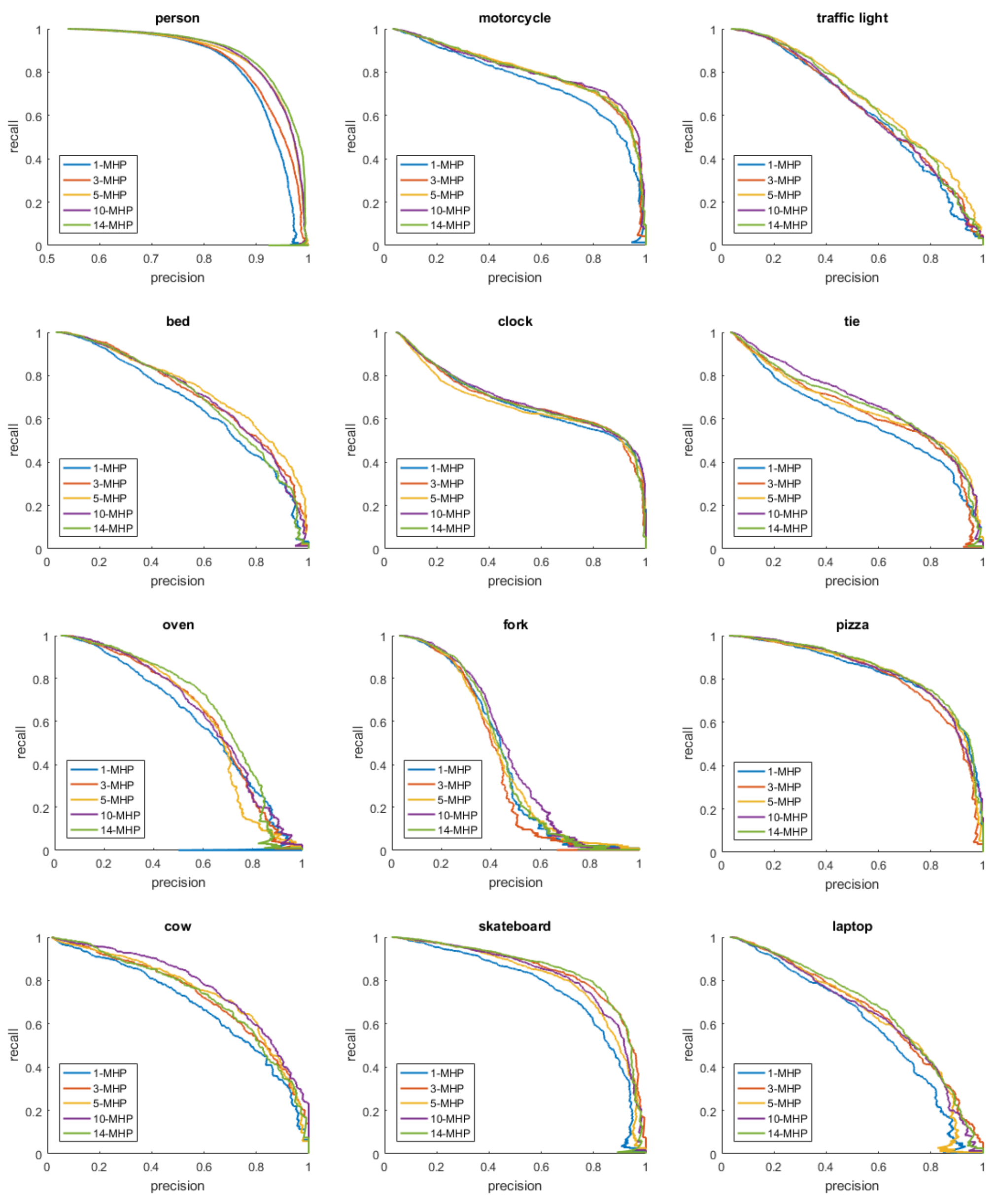


**Figure 5.5. Precision-Recall (PR) curves on COCO.** We show the PR plots for different numbers of hypotheses (1, 3, 5, 10 and 14) for 12 COCO classes. Each class is taken out of a different super-category in MS COCO (e.g. vehicles, furniture, food, etc).

($t = 0, 1, 2, 3$) of the simulation. As the car is approaching the intersection ($t = 0, 1$) and its destination is still unclear, the hypotheses are distributed over the plausible outcomes. Each hypothesis predicts a different one of the three possible exit locations *i.e.* north, east or west for the car coming from the south. From this experiment we obtain another intuitive explanation of the conditional average (see Equation 4.3). The SHP model averages all possibilities in its single output, creating an unrealistic frame where each exit shows a one-third transparent car. At $t = 2$ when the car starts turning to the right, the three predictions collapse into a single decision (eastern exit) with small pose variations modeling the much smaller variance in exit

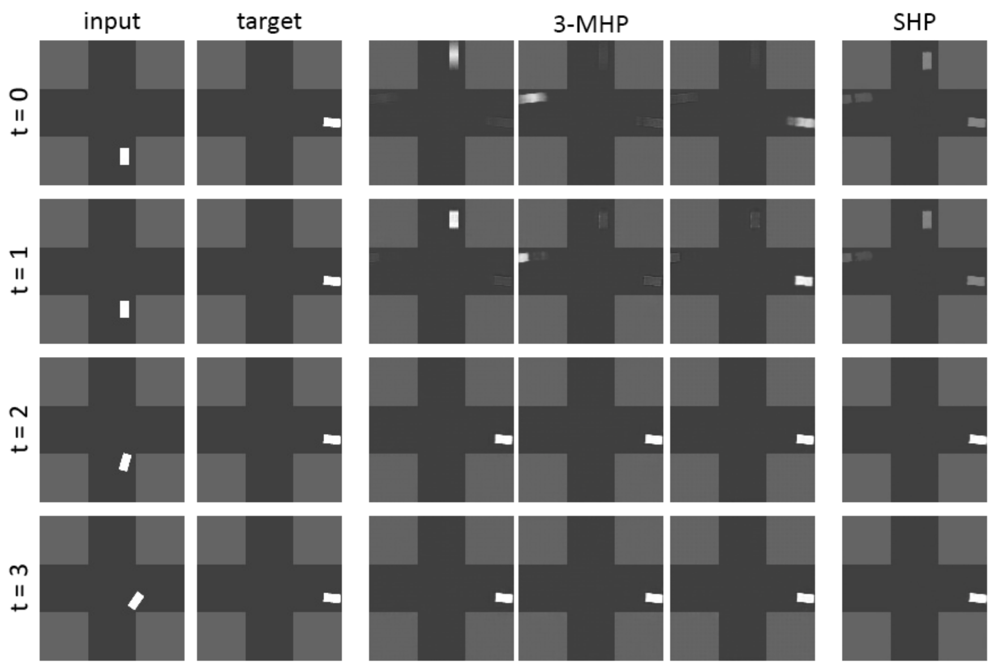


**Figure 5.6.** **Predicting the next frame on the synthetic Intersection dataset.** A SHP model is compared to a 3-MHP model, trained to predict the last frame of a sequence in which a car drives through an intersection. For $t = 0, 1$, three outcomes are possible; SHP blurs them into one unrealistic frame with three ghost cars, whereas MHP predicts all three possible frames distinctly.

positions. Here, even the SHP model predicts a reasonable image, since almost all uncertainty vanished.

From the training data the network has learned to recognize whether a decision about the exit has been already made, and is then able to predict a meaningful set of hypotheses in each of the two cases.

**Intuition: Clustering in Output Space**

As previously mentioned, clustering in output space is very different to MHP. Another example is this experiment. $K = 4$-means clustering in the target domain would yield the four distict exit scenarios whereas MHP can adapt the distribution based on the input – here this means whether the car can be seen turning into its chosen exit or not.

In this experiment we set $\epsilon = 0.05$ which results in faint ghost-cars for the non-selected exits. The loss sharing pulls the predictions slightly towards the conditional average (as seen in the single prediction). Decaying $\epsilon$ towards the end of the training should resolve this issue.

### 5.4.2 NTU Action Recognition Dataset

The NTU RGB-D Action Recognition dataset [166] provides many takes of the same (simple) action with different actors and outfits. We discard the depth streams and use only the RGB videos for training and testing. Since a considerable part of each frame is background, we focus on the region of interest by automatically cropping each sequence around the moving

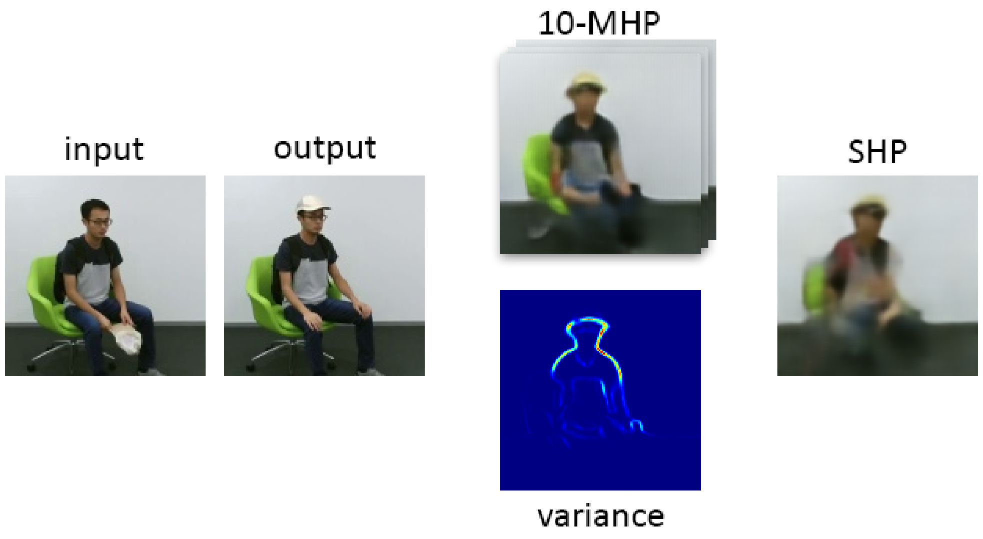


Figure 5.7. **Last-Frame Prediction.** Qualitative results for predicting the last frame of the *put on a hat/cap* action. We show one randomly selected hypothesis. Again, SHP is very blurry, whereas MHP yields a sharper, distinct result. An additional benefit is the ability to compute per pixel variances over the predictions.

| Model | Sharpness | Min. MSE |
|---|---|---|
| SHP | 319.5 | 960.6 |
| 5-MHP | 359.2 | 808.2 |
| 10-MHP | **419.7** | **728.5** |

Table 5.5. **Sharpness and Error Analysis:** We measure the image sharpness (Eq. 5.8, higher is better) for different numbers of hypotheses on the NTU dataset for the *put on a hat/cap* action. Additionally, we report the average mean squared error (MSE) between the best prediction and the ground truth (lower is better).

parts by thresholding the per pixel change between frames. Training to predict one of the last images at the end of the sequence, the network learns to model the visual outcome of an action. We use the mean gradient magnitude of a prediction, as a measure of sharpness:

$$\mathcal{S}(f_\theta(x)) = \frac{1}{3whM} \sum_{c,p,j} ||G_c^j(p)||_2^2, \text{ where } G^j = \nabla f_\theta^j(x), \tag{5.8}$$

and thus as a proxy for image quality. In (5.8) pixel locations are represented by $p$, whereas $w$ and $h$ stand for the image dimensions and $c$ indexes the color channel.

This sharpness measure is compared for the *put on a hat/cap* action for different $M$ in Table 5.5. The intuition that with more hypotheses the model produces sharper images and a lower error is confirmed quantitatively in the table and qualitatively in Figure 5.7. Here the increased sharpness between 1 and 10 predictions becomes evident. Similar to human pose estimation, the variance over predictions becomes and additional output source. The map clearly identifies head and shoulder regions with higher estimated per-pixel uncertainty. We provide further qualitative results in Figure 5.8. Additionally to the *put on a hat/cap* action, another model was trained to predict the last frame in sequences of the *falling down* action.

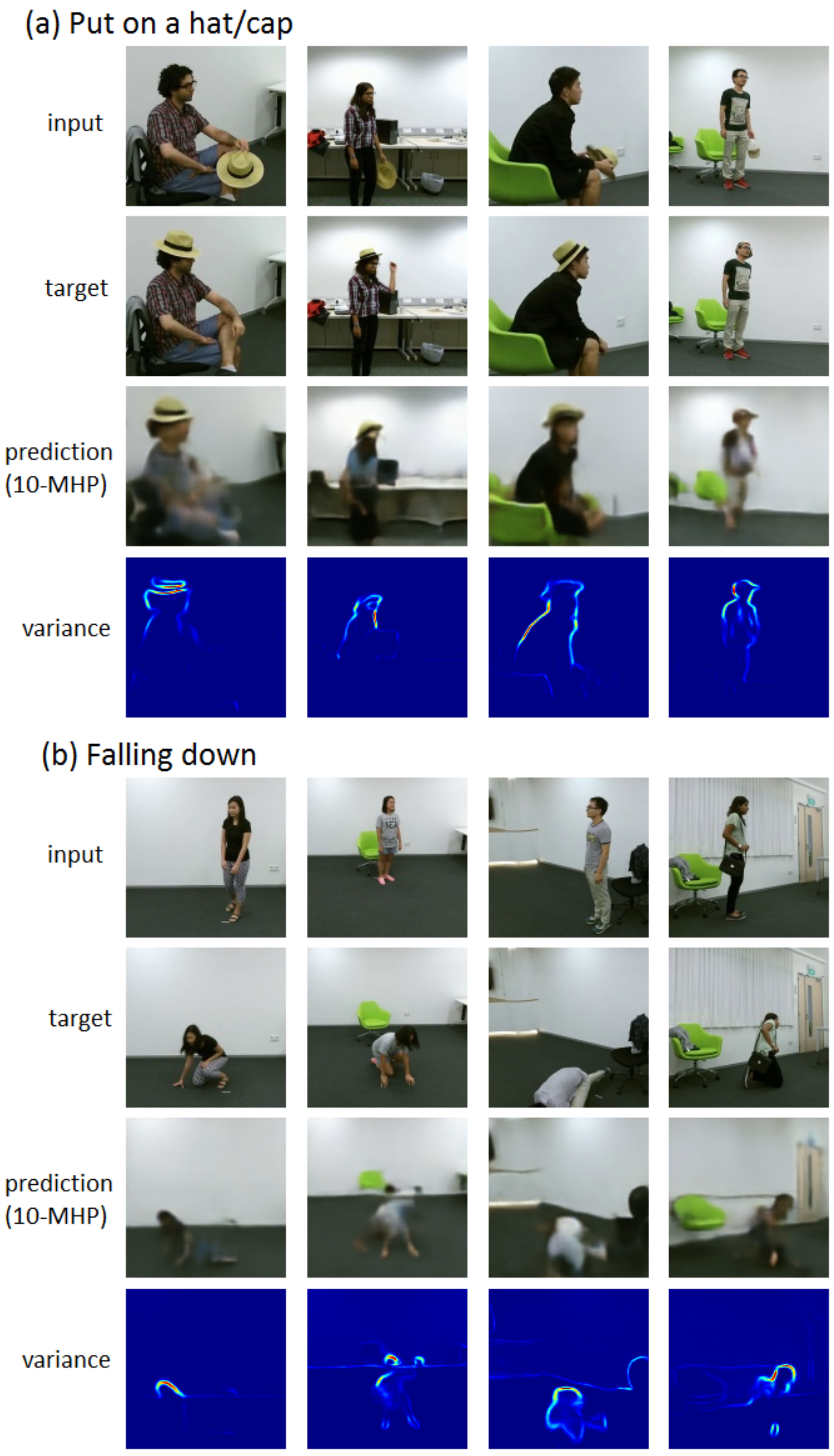


Figure 5.8. **Future frame prediction on the NTU dataset.** We predict the last frame of each sequence for *put on hat* (a) and *falling down* (b) and show one of the ten hypotheses. The variance map shows where the hypotheses vary the most.

## 5.5 Image Segmentation

To be able to directly compare to multiple choice learning (MCL) [103] we train a 4-MHP FCN8s [117] for semantic segmentation on VOC2012. MCL trains separate networks making

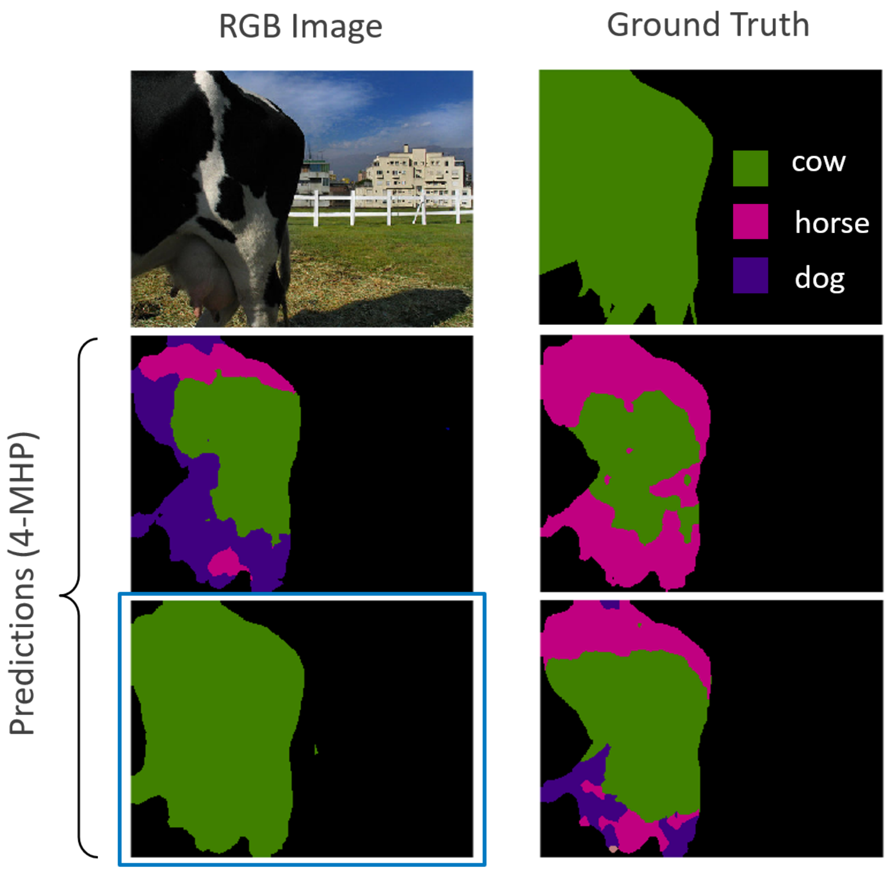


**Figure 5.9.** **Segmentation on PascalVOC.** For the input image we predict four hypotheses and observe varying classes for ambiguous objects.

information exchange between ensemble members harder as each model needs to learn its specialization individually without sharing information or parameters. Additionally, a full CNN needs to be trained for every single output of the ensemble, whereas adding more hypotheses does not add much overhead in our approach. Our model achieves a mean IoU of 70.3%, compared to MCL's 69.1% and uses $\frac{1}{4}$ of the parameters (134.9M [ours] compared to 539.6M [103]).

## 5.6 Generative Adversarial Networks

Together with Ignacio Sarasúa (Master's thesis) we applied the idea of multiple hypothesis prediction in the domain of conditional image generation methods. We use the idea of cycle consistent generative adversarial networks from Zhu *et al*. [206] and extend it for use with MHP.

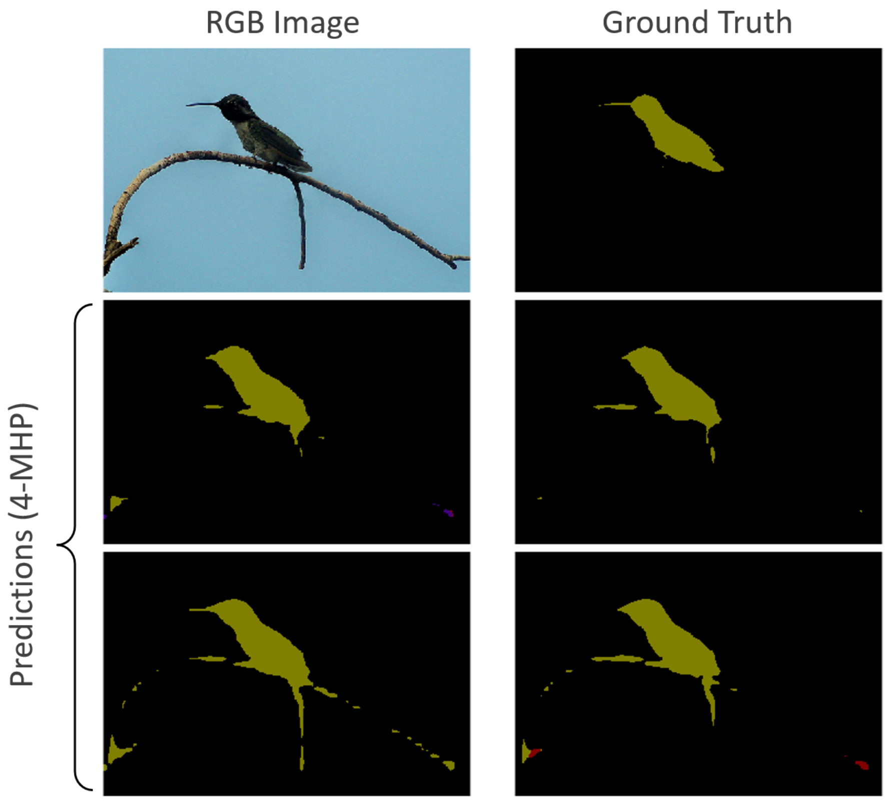


**Figure 5.10. Segmentation on PascalVOC.** For the input image we predict four hypotheses and observe varying detail in the different segmentations.

### 5.6.1 Image-to-Image Translation

Image-to-image translation is the task of transforming an image from one domain to another. Typical examples include day↔night, photograph↔painting, cat↔dog, image↔sketch or summer↔winter. The literature divides this task into two sub-categories based on the data. Paired translation requires corresponding training pairs from both domains. Day↔night is a typical example where it is easy to collect images from the exact same scene in both modalities. Unpaired image-to-image translation does not need paired samples and is thus a harder task. Photograph↔painting is a typical example, as it is usually impossible to acquire the corresponding photograph to a painting, even when the painter originally tried to keep as close to the real world as possible.

We apply the MHP framework to both paired [72] and un-paired [206] image-to-image translation techniques. For implementation details we refer the reader to the orginal methods, as they are relatively unchanged, we will not repeat their details here. However, the image generation CNN is adapted to predict multiple hypotheses.

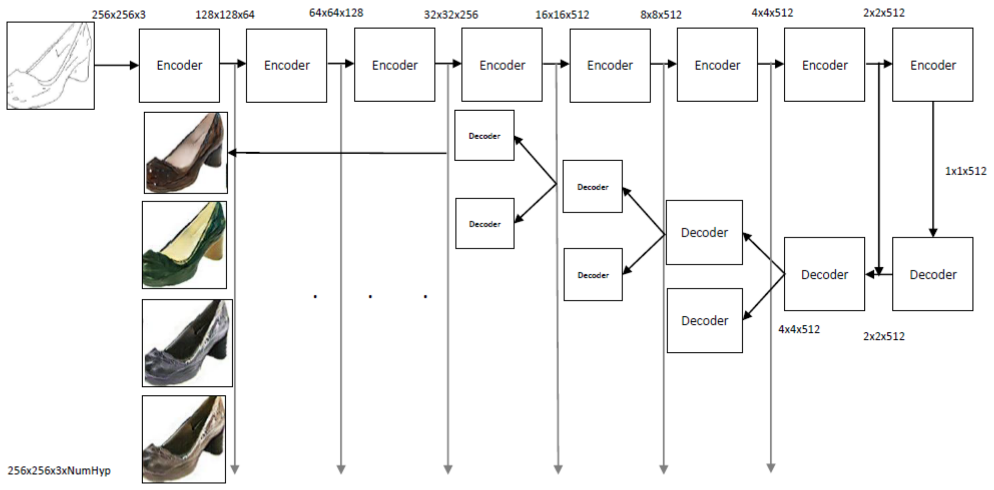
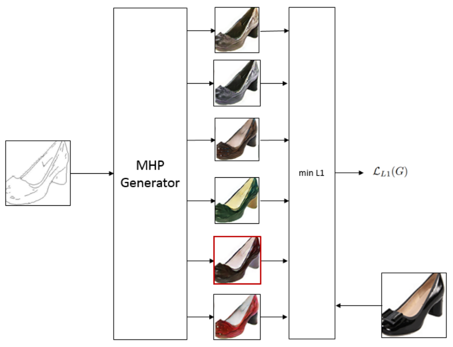
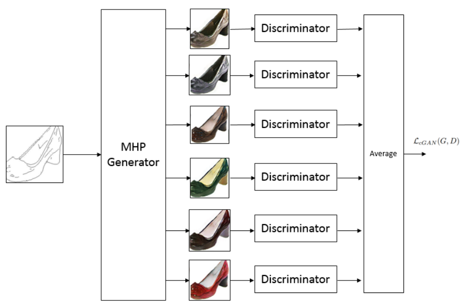


**Figure 5.11.** The generator architecture for image to image translation splits the decoder at several depths to increase the variety in prediction. Thus, at every stage of the decoder the number of hypotheses doubles.



**(a)** The MHP loss is applied as usual on all predictions.



**(b)** A discriminator is applied on every single prediction to encourage them all to look as real as possible.

**Figure 5.12.** The two loss terms used for the MHP-modified image generator architecture. The typical min loss is applied and additionally, the discriminator encourages all hypotheses to look like the target distribution.

**INTUITION: ARCHITECTURE**

Here, instead of predicting all hypotheses from the last layer of the network, we found that splitting the decoder in a tree structure generated more diverse predictions.

Figure 5.11 shows the generator architecture. We split the decoder in several upsampling steps to increase the variety in predictions. The loss is composed of two terms.

In Figure 5.12a we show the typical min formulation loss of MHP with the $\ell_1$ loss. Additionally, a discriminator is applied to *all* hypotheses (Figure 5.12b) to make them as real as possible even if they are not closest (in the $\ell_1$ sense) to the ground truth annotated image.

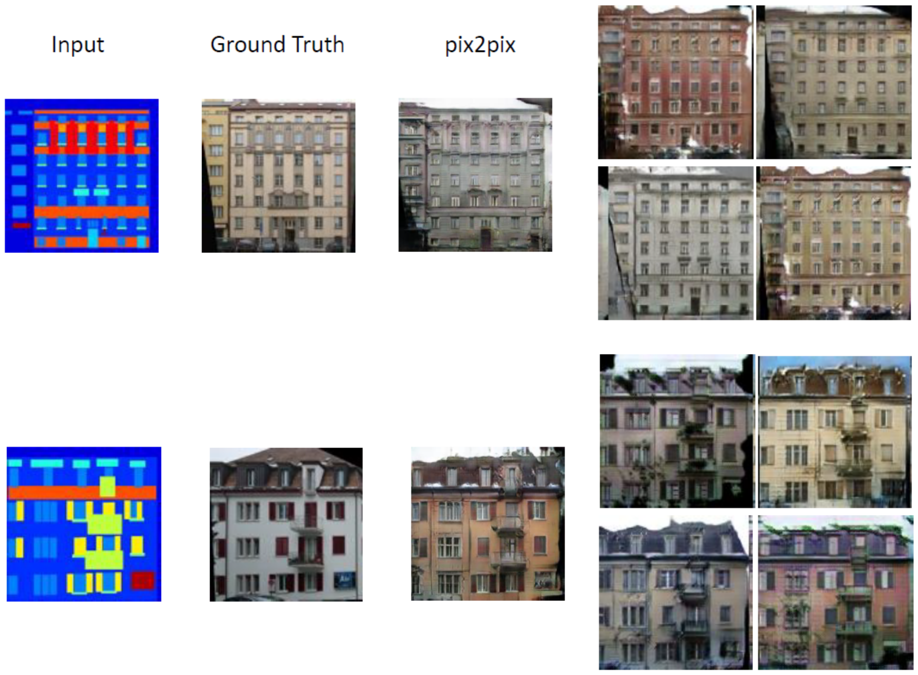


**Figure 5.13.** Comparing MHP results to pix2pix [72]. Due to the multiple predictions we are able to display a much greater variety of possible images for the given segmentation

**HINT: EVALUATION**

Typically, with image-to-image translation techniques it is very hard to obtain quantitative evaluations. Without a large scale study where humans are asked which of the two presented images looks better to them it is difficult to assess quality. We leave this evaluation to future work. However, here we could think of a task such as segmentation↔image where the segmentation performance can indeed be measured.

### 5.6.2 Experiments

In Figure 5.13 we compare our results with pix2pix [72] on paired image-to-image translation, specifically on the task of segmentation→image for building facades. In our experiments we find that typically the results from pix2pix are mostly gray or brown colored. The intuition is that these colors are closest (see *conditional average*) to all ground truth possibilities. The MHP-augmented version is able to predict more colorful variations of the facades as it does not have to average all possible outcomes.

For unpaired image-to-image translation, we demonstrate our approach on the task sketch↔cat. We are given a set of hand-drawn sketches and download many cat images from an automated Google image search. Then we augment the CycleGAN approach [206] with the MHP-modified decoder and compare results after training. Figure 5.14 shows qualitative results. Again we observe a wider variety in predictions coming from our MHP formulation.

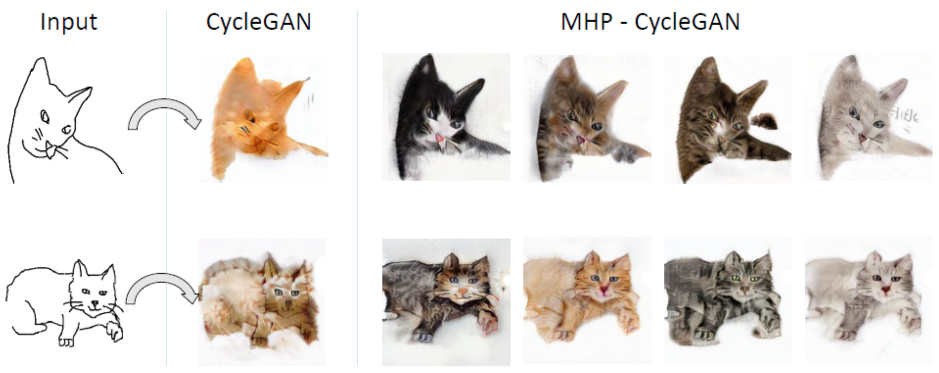


**Figure 5.14.** Comparing MHP results to CycleGAN [206] on the task of translating sketches to cats and vice versa. Due to the multiple predictions we are able to display a much greater variety of possible images for the given sketch

**INTUITION: MHP-CYCLEGANS**

We think that MHP-CycleGANs are a powerful idea, which we would like to further explore in the future. In the original CycleGAN paper it is already mentioned that the cycle consistency that is enforced during training only works well if both domains contain roughly equal information. This means that the projection between the two domains is bijective. However, often one domain has significantly less information (sketch) than the other (image). We strongly believe that in these cases the MHP framework can help resolve the ambiguity when projecting from the diminished representation to the richer one.

## 5.7 Reinforcement Learning

In this section we will have a deeper look into the domain of reinforcement learning and how multiple hypothesis prediction can help to learn a robust stochastic policy. The method presented in this section was developed together with Sanjeev Kumar, whom I supervised for his master's thesis.

### 5.7.1 Motivation

Reinforcement learning is a traditional branch of machine learning which focuses on learning complex tasks by assigning rewards to agents that interact with their environment. It has recently gained momentum thanks to the combination of novel algorithms for continuous control with deep learning models, sometimes even matching human performance in tasks such as playing video games and manipulating objects [133, 170]. Recent methods for continuous control problems like Deep Deterministic Policy Gradient (DDPG) [111], Asynchronous Advantage Actor Critic (A3C) [132] use actor-critic architectures, where an action function is learned by mapping states to actions. DDPG works well on many tasks, but it does not model the uncertainty in actions as it produces a point estimate of the action distribution over states. The actor is forced to deterministically choose one action for every state. A3C and other stochastic policy gradient algorithms output distribution parameters (e.g. Gaussian distributions) instead of point estimates, which can be sampled for action values.

As a simple example where this is sub-optimal, consider the inverted pendulum task, where a pendulum is attached to a cart and the agent needs to control the one dimensional movement of the cart to balance the pendulum upside down. A deterministic agent chooses a single action for every state. This breaks the inherent symmetry of the task. When the cart is not moving and the pendulum is hanging down, two actions are equally promising: either moving left or right. The distribution parameter estimation (e.g. A3C) might work better in this case as there are only two good options, but in cases when there are more than two good actions to select, this will not be optimal. In our approach we allow the agent to suggest multiple actions (using MHP), which enables it to resolve cases like this easily.

Further, we observe that a deterministic behavior of DDPG can lead to sub-optimal convergence during training. The main limitation is that, especially in the beginning of the learning procedure, the actor favors actions that lead to a good immediate reward but might end up being far from the globally optimal choice.

**INTUITION: MULIPLE ACTION PREDICTION**

If the actor is allowed to suggest, at each time step, multiple actions rather than a single one, the resulting policy is non-deterministic, leading to a better exploration of the entire solution space as well as a final solution of potentially higher quality. This can also eliminate the external exploration mechanisms required during training e.g. Ornstein-Uhlenbeck process noise [182], parameter noise [141] or differential entropy of normal distribution.

Here, we introduce an algorithm, which we refer to as Multiple Action Policy Gradients (MAPG), that models a stochastic policy with several point estimates and allows to predict a pre-defined number $M$ of actions at each time step, extending any policy gradient algorithm with little overhead. We will demonstrate it by adapting DDPG [111] to use MAPG.

Another benefit of the proposed method is that the variance of the predicted actions can give additional insights into the decision process during runtime. A low variance usually implies that the model only sees one way to act in a certain situation. A wider or even multi-modal distribution suggests that there exist several possibilities given the current state.

We evaluate the proposed method on six continuous control problems of the OpenAI Gym [14] as well as a *deep driving* scenario using the TORCS car simulator [196]. For a fair evaluation we directly compare DDPG to our MAPG without changing hyper-parameters or modifying the training scheme. In all experiments, we show an improved performance using MAPG over DDPG. To verify if MAPG helps in better exploration during training, we also analyze MAPG under no external exploration policy.

### 5.7.2 Related Work

There is currently a wide adoption of deep neural networks for reinforcement learning. Deep Q Networks (DQN) [133] directly learn the action-value function with a deep neural network. Although this method can handle very high dimensional inputs, such as images, it can only deal well with discrete and low dimensional action spaces. Guided Policy Search [108] can exploit high and low dimensional state descriptions by concatenating the low dimensional state to a fully connected layer inside the network.

Recent methods for continuous control problems come in two flavours, *vanilla policy gradient methods* which directly optimize the policy and *actor-critic methods* which also approximate state-value function in addition to policy optimization. Trust Region Policy Optimization (TRPO) [165] and Proximal Policy Optimization Algorithms [164] can be used as vanilla policy gradient as well as actor-critic methods. Whereas, Deep Deterministic Policy Gradient (DDPG) [111] and Asynchronous Advantage Actor Critic (A3C) [132] use actor-critic architectures, where state-action function is learned to calculate policy gradients.

Stochastic Value Gradients (SVG) [60], Generalized Advantage Estimation (GAE) [165], A3C, TRPO all use *stochastic policy gradients* and predict action probability distribution parameters. The action values are then sampled from the predicted distribution. A parametrized normal distribution is most commonly used as action distribution. This means that this formulation models “action noise” instead of the true action distribution. For example a distribution with two modes cannot be modeled with a Gaussian.

DDPG [111] which extends DPG [169] uses *deterministic policy gradients* and achieves stability when using neural networks to learn the actor-critic functions. The limitation of DDPG is that it always gives a points which may not be desired in stochastic action problems.

[100] estimate stochastic action values using a sequential Monte Carlo method (SMC). SMC has actor and critic models where the actor is represented by Monte Carlo sampling weights instead of a general function approximator like a neural network. SMC learning works well in small state space problems, but cannot be extended directly to high dimensional non-linear action space problems.

### 5.7.3 The Multiple Action Policy Gradient Algorithm

In this section we will describe in detail how multiple action policy gradients can be derived and compare it to DDPG. We will then analyze the differences to understand the performance gain.

#### Background

We investigate a typical reinforcement learning setup [176] where an agent interacts with an environment $E$. At discrete time steps $t$, the agent observes the full state $s_t \in \mathcal{S} \subset \mathbb{R}^c$, and after taking action $a_t \in \mathcal{A} \subset \mathbb{R}^d$, it receives the reward $r_t \in \mathbb{R}$. We are interested in learning a policy $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$, that produces a probability distribution over actions for each state.

Similar to other algorithms, we model the environment as a Markov Decision Process (MDP) with a probabilistic transition between states $p(s_{t+1}|s_t, a_t)$ and the rewards $r(s_t, a_t)$.

We associate a state with its current and (discounted with $\gamma \in [0, 1]$) future rewards by using

$$R_t = \sum_{i=1}^{T} \gamma^{i-t} r(s_i, a_i). \tag{5.9}$$

Since $\pi$ and $E$ are stochastic, it is more meaningful to investigate the expected reward instead. Thus, the agent tries to find a policy that maximizes the expected discounted reward from the starting state distribution $p(s_1)$.

$$J = \mathbb{E}_{r_i, s_i \sim E, a_i \sim \pi}(R_1) \tag{5.10}$$

Here, it is useful to investigate the recursive Bellman equation that associates a value to a state-action pair:

$$Q^\pi(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi}[Q^\pi(s_{t+1}, a_{t+1})]]$$

Methods such as (D)DPG use a deterministic policy where each state is deterministically mapped to an action using a function $\mu : \mathcal{S} \to \mathcal{A}$ which simplifies Equation 5.11 to

$$Q^\mu(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}[r(s_t, a_t) + \gamma Q^\mu(s_{t+1}, \mu(s_{t+1}))]. \tag{5.11}$$

The $Q$ value of an action is approximated by a critic network which estimates $Q^\mu(s_t, a_t)$ for the action chosen by the actor network.

### Algorithm

The key idea behind predicting multiple actions is that it is possible to learn a stochastic policy as long as the inner expectation remains tractable. Multiple action prediction achieves this by predicting a fixed number $M$ of actions $\rho : \mathcal{S} \to \mathcal{A}^M$ and uniformly sampling from them. The expected value is then the mean over all $M$ state-action pairs. The state-action value can then be defined as

$$Q^\rho(s_t, a_t) = \mathbb{E}_{r_t, s_{t+1} \sim E}\left[ r(s_t, a_t) + \gamma \frac{1}{M} \sum_{m=1}^{M} Q^\rho(s_{t+1}, \rho_m(s_{t+1})) \right]. \tag{5.12}$$

This is beneficial since we not only enable the agent to employ a stochastic policy when necessary, but we also approximate the action distribution of the policy with multiple samples instead of one.

There exists an intuitive proof that the outer expectation in Equation 5.12 will be maximal if and only if the inner $Q^\rho$ are all equal. The idea is based on the following argument: let us assume $\rho$ as an optimal policy maximizing Equation 5.10. Further, one of the $M$ actions $\rho_j(s_{t+1}))$ for a state $s_{t+1}$ has a lower expected return than another action $k$.

$$Q^{\rho}(s_{t+1}, \rho_j(s_{t+1})) < Q^{\rho}(s_{t+1}, \rho_k(s_{t+1})) \tag{5.13}$$

Then there exists a policy $\rho^*$ that would score higher than $\rho$ that is exactly the same as $\rho$ except that it predicts action $k$ instead of $j$: $\rho^*_j(s_{t+1}) := \rho_k(s_{t+1})$. However, this contradicts the assumption that we had learned an optimal policy beforehand. Thus in an optimal policy all $M$ action proposals will have the same expected return.

**Intuition: Expected Return**

More informally, this can also be seen as a derivation from the training procedure. If we always select a random action from the $M$ proposals, they should all be equally good since the actor cannot decide which action should be executed.

This result has several interesting implications. From the proof, it directly follows that it is possible – and sometimes necessary – that all proposed actions are identical. This is the case in situations where there is just one single right action to take. When the action proposals do not collapse into one, there are two possibilities: either it does not matter what action is currently performed, or all proposed actions lead to a desired outcome.

Naturally, the set of stochastic policies includes all deterministic policies, since a deterministic policy is a stochastic policy with a single action having probability density equal to one. This means that in theory we expect the multiple action version of a deterministic algorithm to perform better or equally well, since it could always learn a deterministic policy by predicting $M$ identical actions for every state.

In the experimental section we will investigate if the predicted value for each of the $M$ actions is indeed similar during an episode.

Algorithm 1 outlines the MAPG technique. The actor is modified to produce $M$ instead of one output. We achieve this in practice by modifying the number of outputs of the last layer of the network to be $M$ times larger before. For every timestep one action $j$ is then selected. When updating the actor network, a gradient is only applied to the action (head) that was selected during sampling. Over time each head will be selected equally often, thus every head will be updated and learned during training.

**Hint: MAPG vs. MHP**

Multiple Action Policy Gradients differ from the MHP idea in the optimization procedure. While MHP updated only the best prediction with a $\min$-formulation, MAPG samples uniformly from the predicted action hypotheses. This change is necessary, as in reinforcement learning, we do not know a-priori which action will be the best. However, MAPG has similar properties as MHP. For example – as proven above – the expected reward for each action is equal after training. This corresponds to all hypotheses in MHP having equal probability.

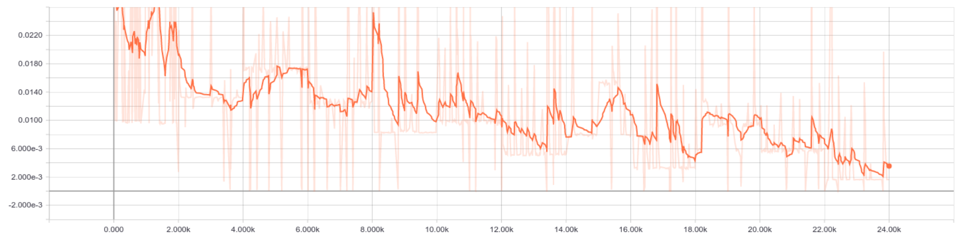
```
Modify actor network $\mu(s|\theta_\mu)$ to output $M$ actions,
        $A_t = \{\rho_1(s_t), \ldots, \rho_M(s_t)\}$.
Randomly initialize actor $\mu(s|\theta_\mu)$ and critic $Q(s|\theta_Q)$ network weights.
Initialize target actor $\mu'$ and critic $Q'$ networks, $\theta'_\mu \leftarrow \theta_\mu$ and $\theta'_Q \leftarrow \theta_Q$.
for episode $= 1$ to $N$ do
   Initialize random process $\mathcal{N}$ for exploration.
   Receive initial observation/state $s_1$.
   for $t = 1$ to $T$ do
      Predict $M$ action proposals
            $A_t = \{\rho_1(s_t), \ldots, \rho_M(s_t)\} = \mu'(s_t|\theta_\mu)$.
      Uniformly sample an action $j$ from $A_t$:
            $a_t^j = \rho_j(s_t) + \mathcal{N}_t$.
      Execute action $a_t^j$ and observe reward $r_t$ and state $s_{t+1}$.
      Store transition $(s_t, a_t^j, r_t, s_{t+1})$ to replay buffer $R$.
      Sample a random batch of size $B$ from $R$.
      Set $y_i = r_i + Q'(s_{i+1}, \mu'(s_{i+1}|\theta'_\mu)|\theta'_Q)$.
      Update critic by minimizing the loss,
            $L = \frac{1}{B}\sum_i (y_i - Q(s_i, a_i^j|\theta_Q))^2$
      Update all actor weights connected to $a_t^j$.

      $$\nabla_{\theta^\mu} J \approx \frac{1}{B}\sum_i \nabla_{a_i^j} Q(s, a|\theta^Q)|_{s=s_i, a=a_i^j} \nabla_{\theta_\mu - \theta_\mu^{\{1\ldots M\}} + \theta_\mu^j} \mu(s|\theta^\mu)|_{s_i}$$

      Update the target networks:
            $\theta'_\mu \leftarrow \tau\theta'_\mu + (1-\tau)\theta_\mu$
            $\theta'_Q \leftarrow \tau\theta'_Q + (1-\tau)\theta_Q$
   end for
end for
```

**Algorithm 1:** MAPG algorithm

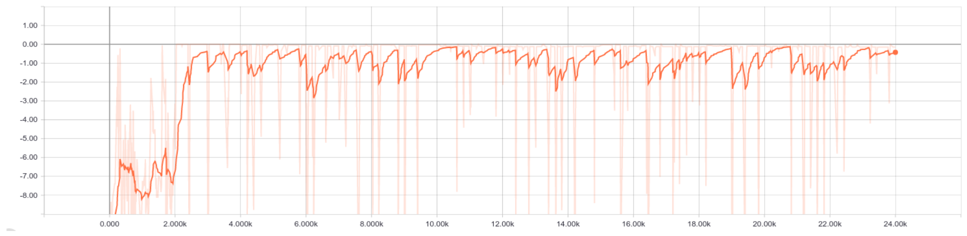
## 5.7.4 Experiments

In this section we will investigate and analyze the performance of MAPG in different aspects. First, we evaluate the performance of MAPG on six different Mujoco tasks and compare the scores with DDPG, A3C and SVG(0). Second, we analyze the influence of the number of actions on the performance by training agents with different values of $M$ on five tasks. Further, to understand the benefit of multiple action prediction, we observe the variance over actions of a trained agent: the goal is to analyze for which states the predicted actions greatly differ from each other and for which ones they collapse into a single choice instead. Finally, we compare the performance of DDPG and MAPG without any external noise for exploration during training.

In all our experiments, we use five continuous control tasks from the Mujoco Simulator [178] and a driving task for The Open Racing Car Simulator (TORCS). A detailed description about the tasks is given in Table 5.6. We use the OpenAI Gym [14] and OpenAI baselines [62] for evaluating our experiments.



(a) Standard deviation in action values.



(b) Reward.

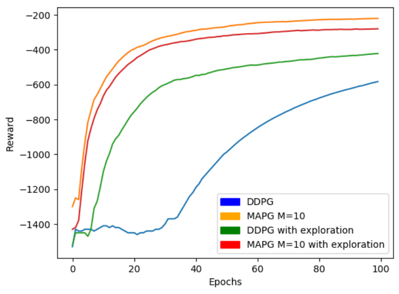
**Figure 5.15.** Standard deviation and reward with $M = 10$ for the `Pendulum` task during training.

**Table 5.6.** Tasks used for evaluation

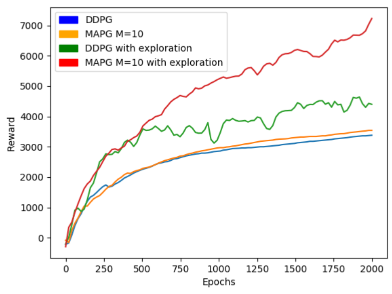
| Task | Action Dimension | State Dimension | Description |
|---|---|---|---|
| Pendulum | 1 | 3 | Pendulum on a cart. |
| Hopper | 3 | 11 | One legged robot. |
| Walker2d | 6 | 17 | Two dimensional bipedal robot. |
| Humanoid | 17 | 376 | Three dimensional bipedal robot. |
| HalfCheetah | 6 | 17 | Two leg robot. |
| Swimmer | 2 | 6 | Three joint swimming robot. |
| Torcs | 3 | 29 | Control car in 3D simulation. |

The base actor and critic network architectures are fixed in all experiments. Each network has two fully connected hidden layers with 64 units each. Each fully-connected layer is followed by a ReLU non-linearity. The actor network takes the current observed state $s_t$ as input and produces $M$ normalized actions $a_t^{(m)} \in [-1, 1]^d$ by applying $\tanh$. We then scale the predicted actions to the actual range specific to the current environment. This allows us to have the same learning rate even when dealing with different action ranges. From the $M$ actions $a_t^{(m)}$, a single action $a_t$ is randomly chosen with equal probability. The critic uses the current state $s_t$ and action $a_t$ as input and outputs a scalar value ($Q$-value). In the critic network, the action value is concatenated with the output of the first layer followed by one hidden layer and an output layer with one unit.

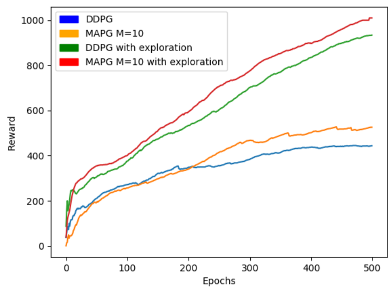
The critic network is trained by minimizing the mean square loss between the calculated discounted reward and the computed $Q$ value. The actor network is trained by computing the policy gradient from the $Q$-value of the chosen action. The network weights of the last layer are only updated for the selected action. This is achieved by setting a zero loss/gradient for the remaining $M - 1$ actions. Ornstein-Uhlenbeck process noise is added to the action



(a) `Pendulum`.



(b) `HalfCheetah`.



(c) `Walker2d`.

**Figure 5.16.** Performance curves for two environments with and without external exploration noise on DDPG and MAPG: original DDPG with OU process noise (green), DDPG without any exploration noise (blue), MAPG (M=10) with OU process noise (red) and MAPG (M=10) without OU process noise (orange).

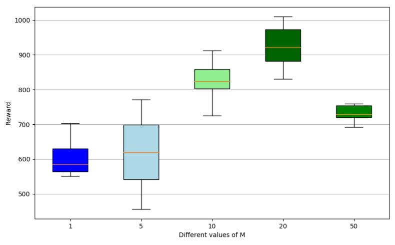
values from the actor for exploration. The parameters for the OU process are fixed for all experiments as $\mu = 0$, $\sigma = 0.2$ and $\theta = 0.15$. The training is carried out for a total of two million steps in all tasks.

For A3C training, we use same actor-critic networks as for the earlier experiment. The output of actor network is a mean vector ($\mu_a$) (one for each action value) and a scalar standard deviation ($\sigma^2$, shared for all actions). The actions values are sampled from the normal distribution ($\mathcal{N}(\mu_a, \sigma^2)$). We used differential entropy of normal distribution to encourage exploration with weight $10^{-4}$. In our experiments, A3C performed poorly than DDPG in all tasks and was not able to learn a good policy for the `Humanoid` task.

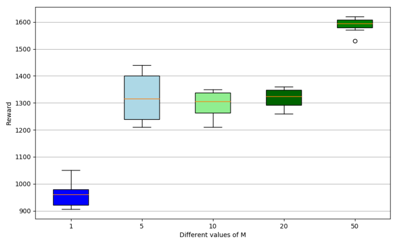
We also run experiments with SVG(0) which is similar to DDPG, but the policy is made stochastic by parameterizing the actor with a random variable sampled from $\mathcal{N}(0, 1)$. We used off-policy variant of SVG(0) for training. It performed rather poorly on almost all tasks.

## Mujoco Experiments

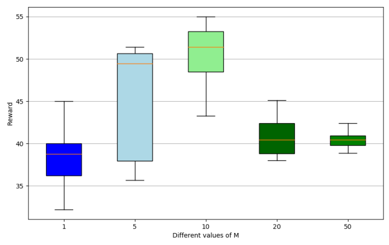
For more meaningful quantitative results, we report the average reward over 100 episodes with different values of $M$ for various tasks in 5.7. Additional the the scores of MAPG we report the scores of DDPG, A3C and SVG(0) which are the closest related methods from the literature. For almost all environments we score higher with $M = 5$. The lower performance in the `Humanoid` task for $M = 5$ might be explained by the drastically higher dimensionality of the world state in this task which makes it more difficult to observe.



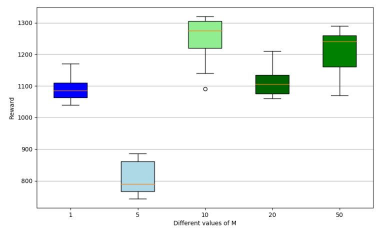
(a) `Hopper`.



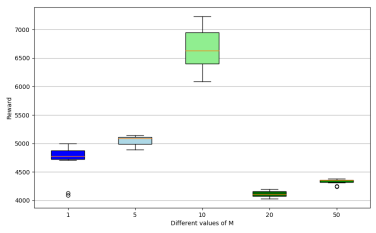
(b) `Walker2d`.



(c) `Swimmer`.



(d) `Humanoid`.



(e) `HalfCheetah`.

**Figure 5.17.** Variation in score of (from top left) `Hopper`, `Walker2d`, `Swimmer`, `Humanoid` and `HalfCheetah` with different values of $M$.

The scores of policy-based reinforcement learning algorithms can vary a lot depending on network hyper-parameters, reward function and codebase/framework as outlined in [61]. To minimize the variation in score due to these factors, we fixed all parameters of different algorithms and only studied changes on score by varying $M$. Our metric for performance in each task is average reward over 100 episodes by an agent trained for 2 million steps. This evaluation hinders actors with high $M$ since in every training step only a single out of the $M$ actions will be updated per state. Thus, in general actors with higher number of action proposals, will need a longer time to learn a meaningful distribution of action. This also means that an actor with higher $M$ will take longer to train all heads sufficiently long. This is partially remedied by the fact that we only split the last layer of the actor into the different actions. With this, most of the actor is updated at every training step and the additional number of training iterations scales slower than linear with increased $M$.

We show the box plots for the scores of Mujoco environments in Figure 5.17, where we can see that the overall score usually increases with $M$. If we set $M = 1$ our algorithm is equal to DDPG, so its performance can also be seen in Figure 5.17. `HalfCheetah` has the most unstable training procedure for all algorithms, often a good policy is lost and the algorithms degenerate back to a zero reward policy where they begin to learn again during training.

**Table 5.7.** Average score $\pm 3\sigma$ over 100 episodes for Mujoco tasks with different $M$. For better readability we denote the highest mean score for each task in bold. Corresponding boxplots can be found in Figure 5.17.

| Environment | DDPG | A3C | $M = 10$ | $M = 20$ | $M = 50$ |
|---|---|---|---|---|---|
| Hopper-v1 | $603 \pm 76$ | $532 \pm 105$ | $824 \pm 94$ | **923** $\pm 90$ | $732 \pm 34$ |
| Walker2d-v1 | $960 \pm 72$ | $764 \pm 112$ | $1297 \pm 70$ | $1319 \pm 50$ | **1589** $\pm 45$ |
| Humanoid-v1 | $1091 \pm 65$ | $281 \pm 40$ | **1248** $\pm 115$ | $1112 \pm 75$ | $1212 \pm 110$ |
| HalfCheetah-v1 | $4687 \pm 455$ | $3803 \pm 125$ | **6659** $\pm 570$ | $4116 \pm 85$ | $4333 \pm 70$ |
| Swimmer-v1 | $38 \pm 7$ | $33 \pm 10$ | **51** $\pm 6$ | $41 \pm 4$ | $40 \pm 2$ |

In Figure 5.15, we studied the variance in action values for $M = 10$ during training together with the achieved reward. The standard deviation of actions generated by MAPG decreases with time. As the network converges to a good policy (increase in expected reward) the variation in action values is reduced. However there are some spikes in standard deviation even when network has converged to a better policy. It shows that there are situations in which the policy sees multiple good actions (with high $Q$-value) which can exploited using MAPG.

This is an interesting aspect of the algorithm. In many tasks there exist a good amount of states where only one single action is optimal. In these cases we expect all actions of MAPG to collapse into a single (or very similar) action. However, in certain situations it might be beneficial to act non-deterministically. We will analyze this observation further in the next section.

## Action Variance Analysis

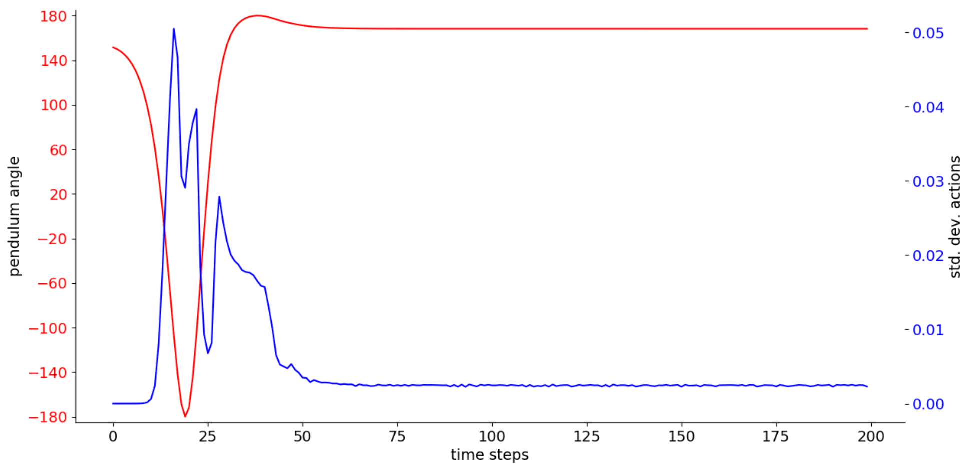


**Figure 5.18.** Standard deviation and angle during one episode of the `Pendulum` environment. An angle of $\pm 180$ is the target inverted pose. $0$ is hanging downwards.

We use the simple `Pendulum` environment to analyze the variance during one episode. The task is the typical inverted pendulum task, where a cart has to be moved such that it balances a pendulum in an inverted position.

Figure 5.18 plots standard deviation and the angle of the pendulum. Some interesting relationships can be observed. The variance exhibits two strong spikes that coincide with an angle of $0$ degrees. This indicates that the agent has learned that there are two ways it can swing up the pole: either by swinging it clockwise or counter clockwise. A deterministic agent would need to pick one over the other instead of deciding randomly. Further, once the target inverted pose (at 180 degrees) is reached the variance does not go down to 0. This means that for the agent a slight jitter seems to be the best way to keep the pendulum from gaining momentum in one or the other direction.

Further, it seems to have learned that slowing the pendulum down close to the upright position, is possible with some noisy torque which explains the second peak in Figure 5.18 at timesteps 30-40.

With this analysis we could show that a MAPG agent can learn meaningful policies. The variance over predicted actions can give additional insight into the learned policy and results in a more diverse agent that can for example swing up the pole in two different directions instead of picking one.

## 5.8 Robotic Manipulation

Inspired by the research directions in human pose estimation [16, 18, 135, 194] and [98] that reformulate keypoint localization as regression of belief maps, we extend this idea within the MHP framework for the detection of robotic grasp points. This work has been published in [45].

### 5.8.1 Motivation

The ability to grasp is important for many applications in the field of personal robotics as well as advanced industrial manufacturing. Currently, robots cannot match human performance. Detecting and precisely localizing reliable grasping points on the surface of an object is required to increase the chance of a successful grasp through a robotic hand or a gripper. This task has gained great attention in recent years, with a plethora of methods being proposed [6, 55, 75, 84, 97, 105, 109, 129, 148, 163, 183, 184, 193, 203], and the creation of benchmarks, such as the Cornell grasp detection dataset [105] for evaluating the performance of approaches designed for this specific task.

Traditional approaches rely on estimating object geometry for localizing grasp points [129, 203]. This leads to slow runtime and failure cases when dealing with complex or unseen object shapes. The success of deep learning in various computer vision applications has led to recent approaches [55, 97, 105, 121, 148, 184, 193] employing CNNs to predict grasp points as 5-DoF bounding boxes directly from images or depth maps. Some prior work specifically

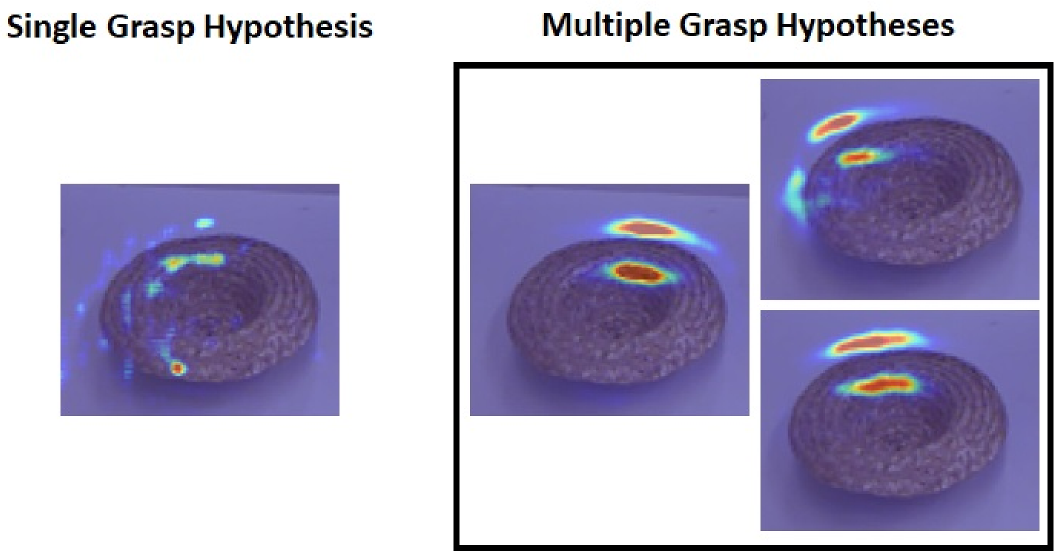


**Figure 5.19.** We propose a model for regressing multiple grasp hypotheses as 2D belief maps, which tackles the ambiguity of grasp detection more effectively than a single grasp detection, in particular for completely unseen shapes, as the one depicted here.

targets grasp estimation for *unseen* objects from RGB-D images [6, 55, 75, 97, 105, 148, 163, 193, 203]. However, this problem is far from being solved [203].

We start from the observation that grasp detection is an inherently ambiguous task, since most objects can be grasped in different ways, and although some ways may be more reliable than others, there does not necessarily exist an optimal grasp. For this reason, each object in benchmark datasets is annotated with a variety of possible grasping positions. To better model the problem's inherent ambiguity, we employ the MHP framework to estimate multiple grasping points for each object. Finally, we investigate a task-specific way to select one grasp location out of these hypotheses. To achieve this, we rank the hypotheses by fitting Gaussian Mixture Models (GMMs) [127] to the predicted belief maps.

## 5.8.2 Multiple Grasp Hypotheses

### Grasp Belief Maps

Formally, a grasp can be denoted as a 5D representation $(x, y, \theta, h, w)$, where $(x, y)$ is the center of a bounding box of width $w$ and height $h$. $\theta$ is the orientation with respect to the horizontal axis. This representation has been often used in related work [6, 55, 97, 105, 148, 193]. We propose an alternative approach to model the detection of a robotic grasp and, instead of using bounding boxes as ground truth, we use belief maps. A belief map corresponding to an $N$-finger gripper can be thought as a mixture of $N$ multivariate normal distributions around the finger tips. Take for example a parallel gripper. The means $\mu^{(n)} = (\mu_x^{(n)}, \mu_y^{(n)})^T$, $n \in \{1, 2\}$, of the normal distributions represent the gripper plate centers and $\|\mu^{(1)} - \mu^{(2)}\|_2 = w$ represents the width of the grasp. $\Sigma = \mathrm{diag}(\sigma_x^{(n)}, \sigma_y^{(n)})^2$ renders the

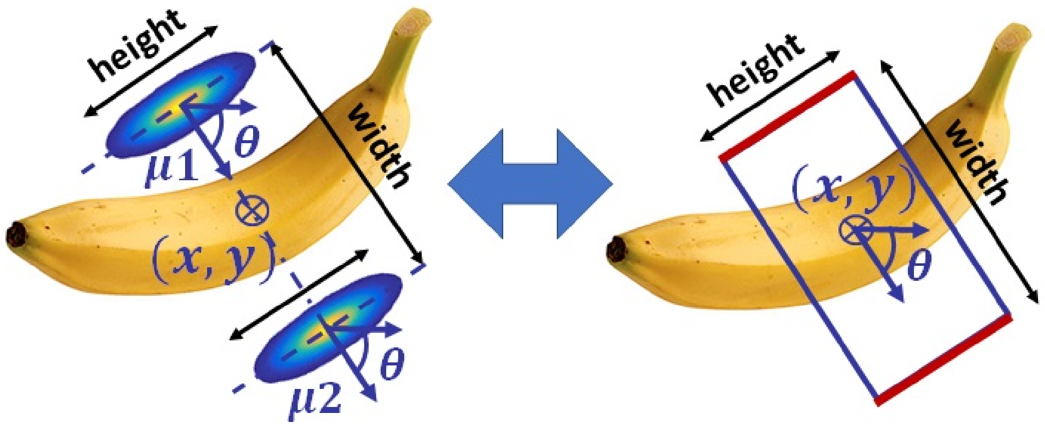


Figure 5.20. An illustration of the adaptation of grasp rectangles to their associated grasp belief maps.

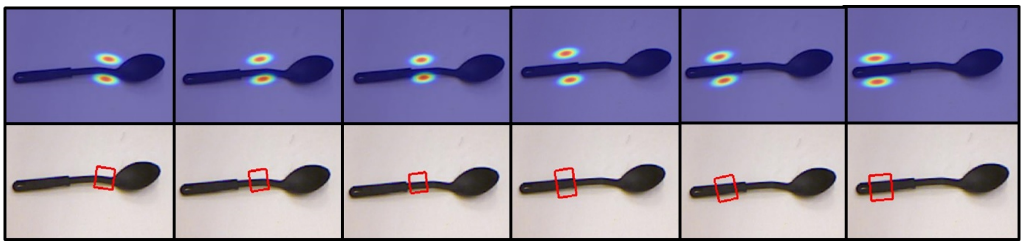


Figure 5.21. Samples of rectangle grasps and grasp belief maps shown for the same item.

distributions elliptical with the primary axis representing $h$. The orientation is controlled by rotation matrix $R(\theta)$ and results in the correct grasp pose in relation to the object.

$$G(p) = \sum_{n} \frac{\exp\left(-\frac{1}{2}(\mathbf{p} - \mu^{(n)})^T R(\theta)\Sigma^{-1} R(\theta)^T (\mathbf{p} - \mu^{(n)})\right)}{\sqrt{2n\pi}\,\sigma_x^{(n)}\sigma_y^{(n)}}, \tag{5.14}$$

where $\mathbf{p} = (x, y)$. Figure 5.20 shows an illustration for the conversion of grasp rectangles to belief maps. In Figure 5.21, we show all annotated grasp configurations for a spoon using the traditional bounding box representation and our adapted belief maps.

### Multiple grasp options

Given the above definition for the ground truth grasps, a CNN is trained to map an RGB input to the heatmap representing a Gaussian mixture. To regress the belief map we use our fully convolutional residual architecture [99]. We train our baseline under a single-grasp setup (SHP) by choosing the most stable available grasp – the one covering the maximum area – as ground truth The loss function to be minimized is the $\ell_2$ norm between the predicted heatmap $\tilde{G}$ and the ground truth map $\mathcal{L}_m = ||\tilde{G} - G||_2^2$.

In the MHP setup, we can produce multiple simultaneous grasp maps, instead of requiring the model to produce just one. Similar to before, we can minimize the meta-loss

$$\mathcal{M}(\tilde{G}, G) = (1-\epsilon) \min_{m=1,\ldots,M} \mathcal{L}_m + \frac{\epsilon}{M-1} \sum_{m' \neq \operatorname{argmin} m \mathcal{L}_m} \mathcal{L}_{m'} \ . \tag{5.15}$$

A ground truth grasp map is randomly chosen among all available possibilities for an image at each training step, so that overall during training, the entire grasp distribution can be seen.

### Grasp option ranking

In a practical setup, we would need a way to judge which of the $M$ hypotheses should be selected. Thus, we could rank all hypotheses and select the one with the highest probability of success. As the network is trained to produce two 2D Gaussians, a natural way of ranking could be based on Gaussian mixture models. Thus we fit two-component GMMs to each hypothesis using finite mixture model estimation [127].

The main parameters of a GMM are the mixture component weights $\phi_k$, means $\mu_k$ and variances $\sigma_k$.

$$p(\vec{x}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\vec{x} \mid \vec{\mu}_i, \Sigma_i), \quad \sum_{i=1}^{K} \phi_i = 1 \tag{5.16}$$

We fit the same parametric model that was used to create the ground truth belief maps (Equation 5.14) and use the highest likelihood fit to select the best hypothesis.

## 5.8.3 Experiments

We validate our method on the Cornell grasp detection dataset [105] and compare to the state of the art.

### Dataset

The Cornell grasp detection dataset [105] which provides 885 RGB-D images with a size of $640 \times 480$ pixels. The images come form 240 graspable objects including several grasp labels per object. There are between 2 to 25 grasp options per object, representing a variety of scales, orientations and locations and making this task an ideal application of the MHP framework. Still, the annotated labels are not exhaustive. In Figure 5.22 we show some RGB samples from the dataset.

### Cross-validation sets

Previous work [6, 55, 97, 105, 148, 193]. has used two types of splits: image-wise and object-wise for cross-validation. Splitting image-wise means training the network while leaving out some views of each object. This evaluates the within-object generalization capability of the network. Object-wise splits use all views of some objects for training, while testing happens on unseen objects.

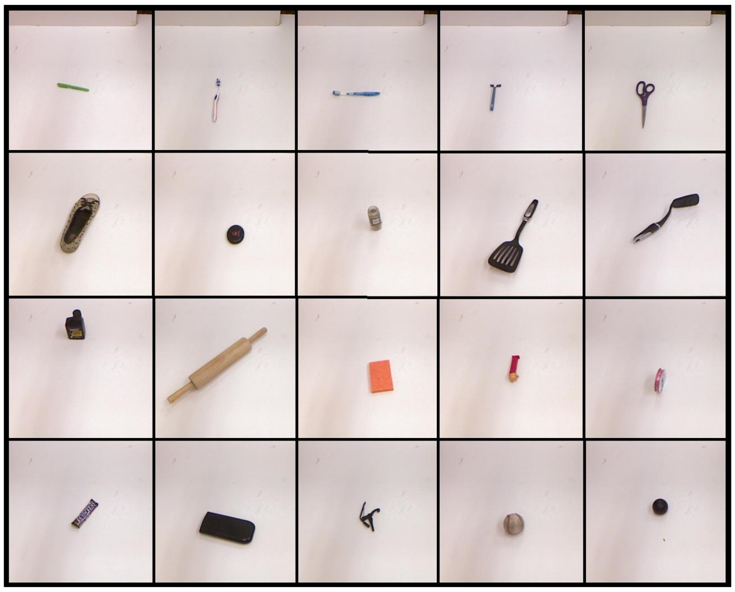


Figure 5.22. A representation of a subset of the objects of the Cornell grasp detection dataset [105].

As there are several objects with different colors but the same shape, we find that the object-wise split is not a good measure for generalization to novel shapes. To investigate our framework's performance on novel shapes, we create an additional shape-wise split, in which we include more novel shapes included in the test set. Image-wise and object-wise splits are validated in five folds and the shape-wise split is done in two folds training on the first 80% of objects and testing on the last 20% and vice versa.

### Grasp Detection Metric

The evaluation metric is the rectangle metric suggested by [75]. A grasp is considered to be valid when it fulfills the following two conditions.

- The intersection over union (IoU) score between the ground truth bounding box ($B$) and the predicted bounding box ($B^*$) is greater than $25\%$, where IoU $= \dfrac{B \cap B^*}{B \cup B^*}$.
- The difference in orientation between the ground truth and the predicted grasp rectangles is less than $30°$.

This metric requires the prediction to be in the form of a grasp rectangle, while our network predicts heatmaps. We therefore fit a rectangle around the Gaussians. We can then compute the percentage of valid grasps as the *Grasp Estimation Accuracy*.

### Evaluation and Comparisons

We compare MHP grasp estimation with the SHP baseline and state-of-the-art methods. We compare the chosen hypothesis (after GMM-ranking) to all annotated ground truth grasp rectangles to find the best match. A full comparison is reported in Table 5.8 for various $M$; $M = 1$ equals to SHP.

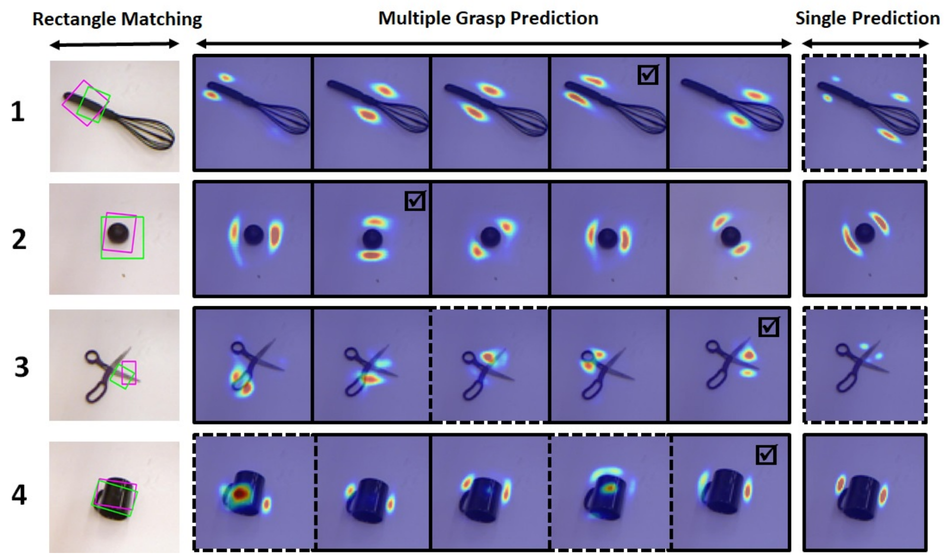


Figure 5.23. Five and single grasp map predictions of sample objects in the dataset. A solid frame around an image is an indicator of grasp detection success, while a dashed line shows an incorrect detection. The images with the ✓ are the top-ranked predictions picked by our GMM likelihood estimation module. These predictions are converted back to grasp rectangles (shown in Magenta) and compared with Green rectangles indicating ground truth grasps.

Although our model relies on RGB input only, it outperforms state-of-the-art approaches that use additional depth information, except for Guo *et al.* [55] that use tactile data. The difference between MHP and SHP is also significant. We observe similar performance between image- and object-wise splits (also for other methods), which shows that the task difficulty has not changed for these two setups, meaning that the new objects that appear in the object-wise split are not substantially different than the ones used for training. With the shape-wise split, we provide a better measure in dealing with novel shapes. The performance gap between the MHP and SHP models is in fact the largest for shape-wise split, which is the most challenging setup. 10-MHP shows a significant boost in performance ($> 10\%$) over SHP and good generalization to unseen shapes/objects.

Figure 5.23 shows qualitative examples from the 5-MHP model and comparing to SHP. Object 3 is undoubtedly a challenging object with many different grasping poses, which are estimated correctly with multiple predictions. Although the grasp rotation of the best ranked prediction did not match with any ground truths, its other predictions are also valid options.

### Evaluating Multiple Grasps

In addition to evaluating the top-ranked prediction, we also want to evaluate the quality of all hypotheses. We report the average grasp detection accuracy over all predictions, which gives the lower performance bound in Table 5.9.

Comparing to the results in Table 5.8, we observe that most hypotheses do correspond to valid grasps. This lower bound decreases with increasing $M$, which suggests that it is more likely to have noisy hypotheses for large $M$. However, GMM-based hypothesis selection counteracts this performance drop.

Table 5.8. Comparison of the proposed method with the state of the art.

| | | Grasp Estimation Accuracy (%) | | |
|---|---|---|---|---|
| Method | Input | Image-wise | Object-wise | Shape-wise |
| Lenz [105] | RGB-D | 73.9 | 75.6 | - |
| Wang [97] | RGB-D | 85.3 | - | - |
| Redmon [148] | RGB-D | 88.0 | 87.1 | - |
| Kumra [97] | RGB-D | 89.2 | 89.0 | - |
| Asif [97] | RGB-D | 89.2 | 89.0 | - |
| Guo [55] | RGB-D | **93.2** | 89.1 | - |
| Kumra [97] | RGB | 88.8 | 87.7 | - |
| ours ($M = 1$) | RGB | 83.3 | 81.0 | 73.7 |
| ours ($M = 5$) | RGB | 91.1 | **90.6** | 85.3 |
| ours ($M = 10$) | RGB | 91.5 | 90.1 | **86.2** |

Table 5.9. Average grasp estimation accuracy of all hypotheses (lower limit) and average grasp success (upper limit).

| Method | Image-wise | Object-wise | Shape-wise |
|---|---|---|---|
| ours - lower limit ($M = 5$) | 80.0 | 77.4 | 75.0 |
| ours - lower limit ($M = 10$) | 76.5 | 73.3 | 72.1 |
| ours - upper limit ($M = 5$) | 98.0 | 98.5 | 96.3 |
| ours - upper limit ($M = 10$) | 99.2 | 98.4 | 99.1 |

On th other hand, the highest-ranked hypothesis might not be the one that leads to best grasping performance. We evaluate this with an oracle that gives the upper limit in Tabletab:grasp:ablation, *i.e.* if at least one valid grasp rectangle is found among the hypotheses, the prediction counts as successful. We observe that in almost all cases MHP gives at least one valid hypothesis, however, GMM-based ranking does not always select a valid hypothesis. Still, the evaluation based on the ranked hypotheses in Table 5.8 is closer to the upper limit.

**Intuition: Dealing with Grasp Hypotheses**

In an actual robotic grasping scenario, one can also imagine the robot making multiple attempts to pick up an object. In this case, it is not necessary to select only a single hypothesis. The robot can instead be given a list of ranked hypotheses which likely lead to successful grasping.

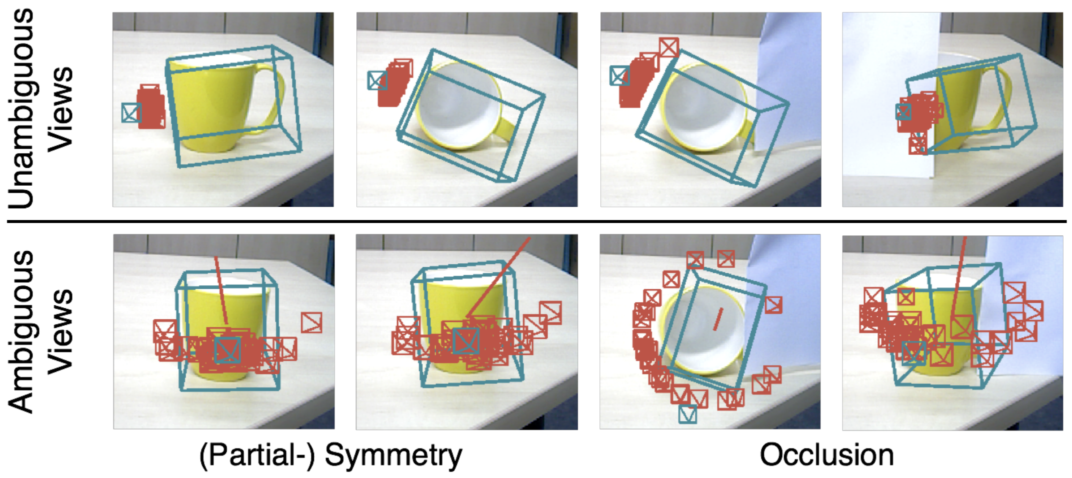


**Figure 5.24.** Pose Estimation from a single RGB or RGB-D image can easily become ambiguous, due to symmetries, occlusion, or even repetitive patterns. The pose of a cup is well defined as long as the handle is visible (upper row). The red frustums visualize a collection of pose hypotheses, regressed by our method. The blue frustum constitutes the median, which is employed to render the 3D bounding box. The red line shows the axis of ambiguity for the given view. The collapse of all hypotheses towards one signalizes that the task is unambiguous. However, partial symmetries and occlusion lead to multiple possible outcomes for the 6D pose (lower row). Our method is capable of determining the ambiguity of the pose by analyzing the distribution of hypotheses (red lines).

## 5.9 6DOF Object Pose Estimation

In [123], we extend the MHP idea to handle symmetries and other ambiguities during object pose estimation from a single RGB image.

### 5.9.1 Motivation

Object detection and pose estimation in 3D from images are ambiguous problems. Objects can have similar appearance from different viewpoints, due to shape symmetries, occlusion and texture. We show typical cases for ambiguity in Figure 5.24. We explicitly deal with this problem by modeling it as a multiple hypothesis prediction problem. The benefits of our approach are better explanations for pose ambiguity and also a higher accuracy in terms of pose estimation. We do so by predicting $M$ pose hypotheses from the network.

### 5.9.2 Experiments

We analyze the distribution of rotations to estimate ambiguities and demonstrate how we can use the variance over hypotheses to model the uncertainty in pose estimation. In this experiment, we utilize the Hinterstoisser [63] dataset. Secondly, we show how multiple hypotheses can be exploited to improve the accuracy for pose estimation when dealing with ambiguous objects. Since the T-LESS dataset [66] mostly consists of symmetric and texture-less industrial objects it poses an excellent benchmark for this problem. Finally, we find that our

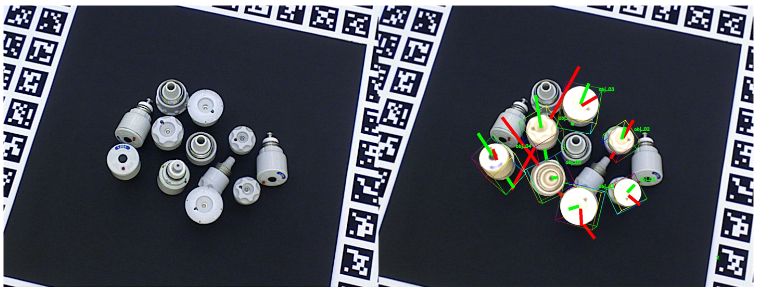


Figure 5.25. Pose and axis of ambiguity estimation on the T-LESS dataset. The green line represents the annotated ground truth axis, the red one is the estimated value.

| | Mean deviation [°] | $\leq 25°$[%] | $\leq 15°$[%] | $\leq 10°$[%] | $\leq 5°$[%] |
|---|---|---|---|---|---|
| Mean | 35.09 | 40.78 | 21.83 | 12.36 | 4.50 |

Table 5.10. Symmetry axis estimation: Mean deviations from the ground truth and fraction of deviations bigger than several offsets over data in Table 5.13.

formulation can also be used for simultaneous object detection and pose estimation, where we show that even different objects can look similar from certain viewpoints.

### Evaluation metrics

For a proper evaluation of our method in terms of 6D pose estimation performance, we distinguish between non-symmetric and symmetric objects. For non-symmetric objects we report the absolute error in degrees for the 3D rotation and 3D translation error in millimeters. The Average Distance of Distinguishable Model Points (ADD) metric [64] measures if the mean average deviation of the transformed points is less than 10% of the object's diameter. For symmetric objects we mostly rely on the Average Distance of Indistinguishable Model Points (ADI) metric. It is similar to ADD, but always considers the closest corresponding points for the distance to be symmetry agnostic.

We also report the Visual Surface Similarity (VSS) metric [83]. VSS is similar to the Visual Surface Discrepancy (VSD) [67], however, we set $\tau = \infty$. Hence, we measure the pixel-wise overlap of the rendered ground truth pose and the rendered prediction. This metric can be applied to both, symmetric and non-symmetric, objects.

### Shape Ambiguity Analysis

In this section we demonstrate the robustness of detecting pose ambiguities. To this end, we manually annotate the symmetry axes for each object for the same subset of T-LESS as before. It is important to note that we do not perform direct object symmetry detection. Instead, we describe the perceived pose ambiguity in terms of a symmetry axis. For instance, a cup itself is not rotationally symmetric due to the handle, yet, we can define an axis along which there exists an arc that describes the same visual output as seen in Figure 5.24. For each correctly detected object we compute the angular difference in degrees, if ambiguity was detected. In Table 5.10 we analyze the accuracy of our estimator. We show an estimated ambiguity axis in

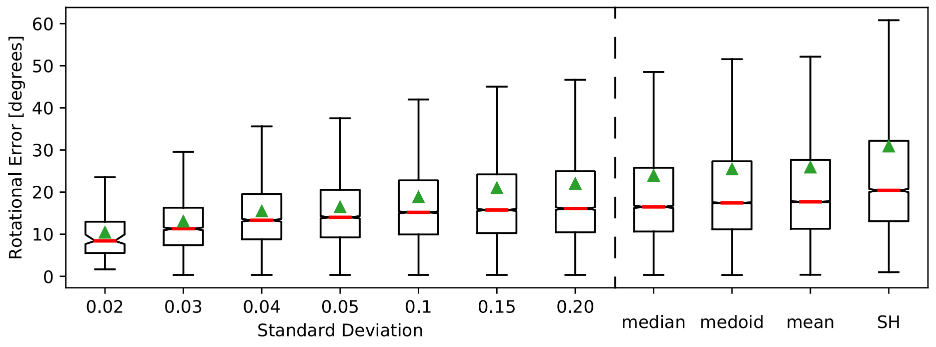


**Figure 5.26.** Behavior of our method with respect to increasing degrees of accepted variances in our hypotheses. The red line depicts the median while the green point shows the mean of the error. For comparison, we show the box plot also for the median of all hypotheses as well as for the medoid, the mean and for a single hypothesis (SH) network. We conducted our experiment on the real scenes over all objects of [64].

Figure 5.25. For each detection, we draw the estimated axis in red and the green line denotes the ground truth axis.

### Employing Multiple Hypothesis as Measurement For Reliability

To the best of our knowledge, there is no prior work that models both, the confidence in class and in continuous pose. We believe that this knowledge strongly affects the overall pipeline. In particular, in a multitude of use-cases, a 3D detector provides a first rough estimate for the pose, nevertheless, thereafter some robust tracker is operating. The subsequent tracking benefits heavily from a confidence for the pose estimation. When we detect the object but cannot reliably predict the pose, we can give feedback to the system to try again, or change the viewpoint.

Besides the additional information on the viewpoint ambiguity, we can utilize the different hypotheses as a measurement for the confidence for the unambiguous 6D pose. To quantify this effect, we report our test results on the 'Hinterstoisser' dataset, which consists of only unambiguous objects. We compute a confidence measure via standard deviation to the Karcher mean [80]. In Figure 5.26, we show this uncertainty measure and its relation to the pose accuracy. We plot the error statistics for pose estimates where all hypotheses are in a certain increasing range around the mean (object pose via Weiszfeld). We compare this with three estimates. First is the geometric median of all hypotheses (median). Second the oracle pose hypothesis closest to it (medoid) and third the mean. We note that an allowed variance in the hypotheses introduces uncertainty in the pose estimation and that the Weiszfeld algorithm gives the most reliable pose estimates. Moreover, MHP predicts more reliable results than the single hypothesis network (SH).

### 2D Object Detection 6D Pose Estimation

Although detection is not the focus of this work, we report similar results to [83]. In Table 5.11 we find that larger objects are detected very robustly, smaller ones are sometimes missed such as the 'ape' and the 'cat' object. As previously mentioned, we remove the very first prediction branch in our architecture, which is mostly responsible for small objects.

| | | ape | bvise | cam | can | cat | driller | duck | holep | iron | lamp | phone | mean |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| *RGB* | ours | 61 | **100** | 73 | **94** | 73 | 97 | 92 | 77 | 97 | – | 91 | 86 |
| | [83] | 76 | 97 | 92 | 93 | 89 | **98** | 80 | 72 | **98** | 93 | **92** | 89 |
| *RGBD* | [82] | **98** | 95 | **93** | 83 | **98** | 97 | **98** | **98** | 91 | **98** | 85 | 94 |
| | [64] | 53 | 85 | 64 | 51 | 66 | 69 | 58 | 52 | 68 | 68 | 56 | 63 |

**Table 5.11.** F1-scores for each sequence of [64]. Note that the 'Hinterstoisser' scores are supplied from [177] with their evaluation since [64] does not provide them.

| | Rot. Err. [°] | Trans. Err. [mm] | VSS [%] | ADD [%] | Rejection [%] |
|---|---|---|---|---|---|
| Ours ($\sigma < 0.05$) | 16.42 | 6.54, 6.20, 46.72 | 76.4 | 29.3 | 38.8 |
| Ours ($\sigma < 0.10$) | 18.82 | 7.22, 6.79, 52.04 | 74.5 | 24.8 | 16.3 |
| Ours ($\sigma < 0.15$) | 20.94 | 7.53, 7.27, 55.14 | 73.7 | 23.0 | 7.1 |
| Ours ($\sigma < 0.20$) | 22.00 | 7.70, 7.48, 56.39 | 73.3 | 22.4 | 3.5 |
| Ours ($median$) | 23.90 | 7.89, 7.69, 57.52 | 72.8 | 23.9 | 0.0 |
| Ours ($mean$) | 23.47 | 7.84, 7.62, 57.78 | 72.9 | 22.0 | 0.0 |
| Ours ($medoid$) | 24.08 | 7.91, 7.68, 57.80 | 72.6 | 21.5 | 0.0 |
| Ours (SH) | 30.87 | 9.66, 9.51, 71.44 | 68.8 | 15.5 | – |
| SSD-6D [83] | 27.96 | 9.75, 9.33, 71.09 | 67.4 | 9.4 | – |
| BB-8 [145] | – | – | – | 43.6 | – |

**Table 5.12.** Pose Errors on 'Hinterstoisser' dataset [64]. We also report our performance with respect to different typs of averaging, i.e. the mean, the geometric median, and the medoid retrieved from the closest hypothesis to the median. In addition, we also report the results for different bins for the standard deviation over all hypotheses for the poses.

In Table 5.12, we show significant improvements over [83]. Our robust averaging on all hypotheses helps in all metrics. Additionally, by only allowing poses with a standard deviation $\sigma < 0.1$, we can further improve the scores, while only discarding $16.3\%$ of all pose estimates. The rotational error decreases by almost 10°(i.e. from 27.96°to 18.82°), while the translation error improves by approximately 25% for each axis. In general, Table 5.12 shows that the standard deviation of the pose hypotheses can be utilized as a robust measurement of reliability. Overall, the multiple hypothesis formulation yields a robust pose estimation method in case of an unambiguous pose detection with multiple outputs. In essence, the median filter of the quaternion hypothesis strengthens the pose estimates significantly compared to the Single Hypothesis (SH) predictions shown in Figure 5.26. However, similar to [83], [145] achieve a better ADD score through training on real data. Contrary to these approaches, we train on purely synthetic data, which allows us to train for any object, provided the CAD model, without having to conduct time-consuming annotations.

Table 5.13 reports results for the T-LESS dataset. The dataset contains multiple very similar objects. For our experiments we choose a subset that covers most cases: complete rotational symmetry along an axis (object 4), similarly-looking objects (7, 9) and objects with more than

| | | Multiple Hypotheses | | | | Single Hypothesis | |
|---|---|---|---|---|---|---|---|
| | Scene | VSS [%] | ADI [%] | Cluster VSS [%] | Cluster ADI [%] | VSS [%] | ADI [%] |
| obj_04 | 5, 9 | 73.36 | 50.58 | **73.56** | **52.18** | 60.88 | 43.46 |
| obj_05 | 2, 3, 4 | 79.78 | 58.44 | **81.83** | **62.53** | 72.63 | 49.51 |
| obj_07 | 2 | 76.5 | 75.93 | **78.38** | **84.36** | 53.71 | 43.49 |
| obj_09 | 5, 11 | **66.78** | 46.95 | 66.28 | **47.40** | 48.87 | 23.02 |
| obj_10 | 5, 11 | 67.65 | 45.33 | **69.80** | **46.58** | 60.87 | 43.38 |

Table 5.13. Evaluation Scores on a subset of T-LESS. Thereby, we compare our results with and without clustering with the same predictor trained to output a single hypothesis.

one rotational symmetry (5, 7, 9, 10). Our clustering approach performs better in most cases than the average rotation. Again, the single hypothesis baseline is always outperformed by its multiple hypotheses counterpart.

### Beyond Single Object Ambiguities

Distinct objects perceived from different viewpoints can have a very similar visual appearance. In Figure 5.27 (left) we show that certain poses for objects 01 and 02, and 05 and 06 result in near-similar images. We extend our architecture to also model uncertainty in the object detection branch of our network. In particular, we adapted the classification loss to only distinguish between foreground and background $L_{fg}$, rather than directly classifying the associated object inside the anchor box. The classification loss is moved inside the multiple hypotheses formulation:

$$L(Pos, Neg) := \sum_{b \in Neg} L_{fg} + \sum_{b \in Pos} (L_{fg} + \alpha L_{fit} + \mathcal{M}(\beta L_{class} + \gamma L_{rotation}) \tag{5.17}$$

For each positive anchor box and threshold $t_1$, we get a set of $m$ rotations paired with the associated class distribution. We employ a second threshold $t_2$ to filter out weak hypotheses before clustering. Note, that we conduct this clustering only for hypotheses which voted for the same class.

Figure 5.27 show a prediction of the method. After the detector correctly detects the object of interest (mid left), it casts $m = 30$ votes for object and rotation (right), which are all very plausible. We only show five hypotheses for clarity. All hypotheses visually match the object and are only differ rotation along the object's symmetry axis. This shows that our method is capable of conducting proper reasoning based on visual appearance.

## 5.10 Summary

This extensive evaluation of the multiple hypothesis idea shows that it is a powerful concept that can be applied to a wide variety of tasks. Often the necessary modifications are small and the adaption in straight forward. In many of the described applications MHP does not

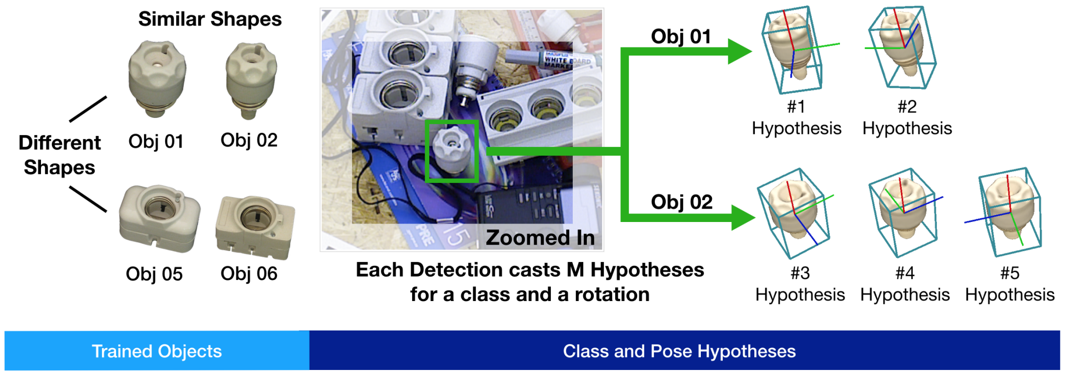


**Figure 5.27.** Simultaneous pose and class ambiguity. The pairs 1, 2 and 5, 6 are completely different to each other, but their members are very similar. A network trained to generate multiple hypothesis for both class and pose learns the ambiguity and only proposes feasible hypotheses based on this similarity.

only improve the performance of the model but also provides additional information about the prediction. For example we can use it to estimate confidences or detect symmetries in the data.

However, the number of hypotheses $M$ remains a hyper-parameter that needs to be chosen manually. In the next part we propose an algorithm that does not need a fix set of predictions by can sample continuously from the target distribution.

# Part III

Learning to Sample

# 6 Introduction

In the previous chapter we presented the MHP framework, which enables the network to predict $M$ guesses in order to model the distribution of the output. Naturally, a static $M$ might not be optimal. This thought has motivated the following work, in which we replace the fixed number of output hypotheses with stochastic sampling inside the network, theoretically allowing the model to learn any distribution arbitrarily well.

## 6.1 Motivation

As we discussed previously, uncertainty is fundamental to many inference problems. While most recent deep learning methods are designed to produce a single statistic (e.g the conditional mean), in many practical scenarios knowing the quality of the answer is as important as the answer itself. For example, when making a medical diagnosis, it is essential to evaluate alternative diagnoses and the consequences of an error.

In this chapter, we investigate ambiguous prediction problems by formulating them as conditional distribution problems. Instead of learning a mapping from an input to one output, we design an algorithm to produce samples that emulate the conditional distribution $p(y|x)$, where $y$ is a label or value to be predicted and $x$ is a datum from which it is inferred. By sampling from what is now effectively a simulated random variable – that is sampling many values of $y$ for a given $x$ – it is possible to compute any number of statistics about the answer: the mean, the variance, a confidence interval, and so forth.

As a result, we do not need to make any explicit distributional assumptions. Instead, the goal is to simply learn to predict samples that follow a certain distribution by maximizing the similarity between the predicted samples and the ground truth samples. The advantage then is generalizability, *i.e.* a non-parametric model is not constrained by the specific choice of the distribution or mixture model, but can theoretically learn any distribution function.

First, we derive a loss function, based on an approximation of the Kullback-Leibler (KL) divergence, that can be optimized to learn a sampling function. Second, we show that in the special case of a single ground truth label for each input, the proposed loss converges to a heuristic that has been previously employed for modeling learning problems with ambiguous labels. As a result, we now know these methods also empirically minimize KL divergence and are a mechanism for learning to sample from the empirical distribution. Finally, we can derive a measure of density which enables us to select samples that belong to high density regions. As a result, in cases where a single output or few high probability modes are desired, these can be computed from the predicted samples.

**Intuition: Fixed and Continuous Sampling**

The goal is learning to sample from a distribution, whereas MHP learned to approximate a distribution with $M$ point estimates. The sampling idea is mathematically more satisfying, however, we find that it requires several trade-offs. With the increased complexity of learning a full distribution comes the disadvantage that we now (mostly) require more than one outcome to be annotated – this is almost never the case in current public datasets. Since we are estimating the sample density using nearest neighbors, we are restricted to low and medium dimensional problems, as ($\ell_2$) distances lose meaning in very high dimensions. Similar to mixture density networks, the new loss comes with numerical difficulties and makes training noticeably slower. However, we are able to show some interesting properties and good results in our experiments.

In what follows, we briefly review related work, derive the loss function, and provide an empirical evaluation of neural networks trained with this loss function on synthetic and real problems. We note that although we use neural networks, in principle our approach is agnostic to the learning method.

## 6.2 Related Work

Again, a good portion of the work related to multiple hypothesis prediction is applicable here. The topic of learning multiple outputs has been investigated from several different directions. One approach is to explicitly learn a representation of a distribution. For example, Bishop [10] predicts the parameters of (Gaussian) mixture models using a maximum likelihood formulation. However, Gaussian mixture models quickly become unstable or intractable in higher dimensions.

An important, related line of work models uncertainty by approximating Bayesian inference in Gaussian Processes [147] within neural networks. In deep learning, dropout [173] has been used as a standard regularization technique by randomly dropping (hidden) units while training. It prevents the co-adaptation of units and provides an average of many different weight combinations inside the network. Gal and Ghahramani [41] develop a probabilistic representation for dropout (applied before each weight layer) in order to estimate model uncertainty. Gathering the outcomes of several stochastic forward passes of a sample through the network, the mean and variance of the prediction can be obtained.

Density estimation from samples has been studied in [34, 106] which form the theoretical basis for the KL divergence [95] estimators in [140, 192] based on the $k$ nearest neighbor formulation that we use in our optimization.

Multiple Choice Learning (MCL) is an approach to address problems in which, due to ambiguities or mislabeling, a conclusive prediction is not possible or not optimal. For example, Guzman [57] uses MCL, based on a Structured SVM [180], to solve image segmentation and protein prediction problems. The MCL method is later extended to deal with the problem of enforcing the diversity of the predictions [56], with an additional loss function term that penalizes similar outputs. While this achieves a higher variety in the predictions, it also introduces a bias which means the predictions might not represent the true distribution of the data. Lee et al. [103] apply these principles to neural network ensembles (Stochastic Multiple Choice Learning - sMCL) by minimizing an oracle loss. In MHP we express ambiguity through a generic model-agnostic and loss-agnostic multiple hypotheses model that learns an implicit Voronoi tessellation of the output space. Our methods can be used to address MCL problems, but are more general as they are not limited to a fixed number of output values and are agnostic as to the source of the uncertainty in the data.

On the application side, various specific approaches have been proposed. For example, Bayesian SegNet [85] predicts semantic segmentation together with uncertainty by creating a posterior distribution of the output via Monte Carlo sampling with dropout. Gao et al. [42] improve the accuracy of multi-label image classification by training with KL-divergence and soft assignments. In this case, directly optimizing KL-divergence is relatively straightforward since their output is a discrete probability distribution. Variational auto encoders (VAEs) [87, 150] also minimize the KL divergence of the encoding inside the network. This is another example, however, where the distribution is parametrized by a multivariate Gaussian which is then used to generate samples from the decoder.

Directly relevant to our work, there exists a variety of papers [57, 103, 157, 186], that have used a *heuristic* minimum loss formulation $\min_i ||f_i(x) - y||_2^2$ for learning to produce multiple predictions $f_i$. In what follows, we show that there is theoretical basis for learning to sample using KL divergence and that this formulation is a special case where only a single ground truth annotation is available for each sample (Section 7.3). Thus, we show that this prior work can also be viewed as minimizing the KL divergence.

# Mathematical Derivation 7

We start by defining our goal and introducing the notation. The notation is kept as closely as possible to the previous sections to be able to compare to multiple hypothesis prediction. However, since we are now modeling distributions we will have a probabilistic view on the problem. We denote the vector space of input variables by $\mathcal{X}$ and the vector space of output variables or *labels* by $\mathcal{Y}$. It is important to note the assumption that for a given $x$, there does not exist a single, unique label $y$ but several ones following an unknown probability distribution, i.e. we are given a set of $N$ training tuples $(x_n, y_n)$ with

$$\mathcal{Y}_n = \{y_{1,n}, \dots, y_{|\mathcal{Y}_n|,n}\} \subset \mathcal{Y}. \tag{7.1}$$

The joint probability density over input variables and labels is denoted by $p(x, y) = p(y|x)p(x)$, where $p(y|x)$ describes the conditional probability for the label $y$ given the input $x$.

## 7.1 Unconditional Distributions

To build intuition, we start with the simpler case of an unconditional distribution. Let us assume that we are given a set of $N$ samples $y_n$, drawn from an unknown distribution $\mathbf{Y}$. Our task is then to learn a model $f_\theta$ that creates samples $\hat{y}$ that follow $\mathbf{Y}$. To this end we will introduce a random variable $z \sim \mathbf{Z}$ that is easy to sample from, e.g. a uniform distribution ($\mathbf{Z} = \mathbf{U}^d(0, 1)$), and use it as a stochastic input to drive our model $f_\theta(z) = \hat{y}$. How well $f_\theta(z)$ achieves this goal, can be measured by the Kullback-Leibler divergence [95]:

$$D(\mathbf{P}||\mathbf{Q}) = \int_{\mathbb{R}^d} p(x) \log \frac{p(x)}{q(x)}\, dx, \tag{7.2}$$

where $\mathbf{P}$ and $\mathbf{Q}$ denote the two distributions to be compared – in this case $\mathbf{Y}$ and the distribution implicit in $f_\theta$. The KL divergence vanishes for $\mathbf{P} = \mathbf{Q}$ and can thus be used as a loss function for optimization.

Equation (7.2) assumes that the densities $p$ and $q$ are known; in the context of this work, however, we are only given samples $\{y_1, \dots, y_{|\mathcal{Y}|}\} \subset \mathcal{Y}$ of the distribution $\mathbf{Y}$ and predictions $\hat{y}$. Thus, we need to approximate (7.2) using the provided samples. It has been shown in [140, 192] that this can be done using $k$-nearest-neighbor density estimation:

$$\hat{D}(\mathcal{P}||\mathcal{Q}) = \frac{1}{|\mathcal{P}|} \sum_{i=1}^{|\mathcal{P}|} \log \frac{d_k(p_i, \mathcal{P} \setminus \{p_i\})}{d_k(p_i, \mathcal{Q})}, \tag{7.3}$$

where the Euclidean distance of a point $p_i$ to its $k$-th nearest neighbor in a set $\mathcal{Q}$ is denoted by $d_k(p_i, \mathcal{Q})$. We note that we have dropped an additive normalization constant which depends on $|\mathcal{P}|$ and $|\mathcal{Q}|$ in (7.3). As this constant does not depend on the parameters of $f$ it can be neglected for optimization purposes.

It has been shown [192] that for the special case of $k = 1$ (nearest neighbor) the bias as well as the variance of this estimator still vanish as $n \to \infty$. In this case, Equation (7.3) is simplified to

$$\hat{D}(\mathcal{P}||\mathcal{Q}) = \frac{1}{|\mathcal{P}|}\sum_{i=1}^{|\mathcal{P}|}\log\frac{\min_{j\neq i}||p_i - p_j||_2}{\min_l ||p_i - q_l||_2}. \tag{7.4}$$

Intuitively, the numerator estimates the density in $\mathcal{P}$ around $p_i$ while the denominator estimates the density of $\mathcal{Q}$ around $p_i$. If the two densities are equal, the ratio becomes 1 and the loss vanishes.

When using $\hat{D}$ as the objective function for training, we replace $\mathcal{P}$ and $\mathcal{Q}$ by the set of empirical samples $\mathcal{Y}$ and the set of predicted samples $f(\mathcal{Z}) = \hat{\mathcal{Y}}$, respectively. Since the KL divergence is not symmetric, i.e. $D(\mathcal{P}||\mathcal{Q}) \neq D(\mathcal{Q}||\mathcal{P})$, we symmetrize the loss (similar to the original formulation by Kullback and Leibler) by minimizing the sum:

$$\hat{D}(\mathcal{Y}||\hat{\mathcal{Y}}) + \hat{D}(\hat{\mathcal{Y}}||\mathcal{Y}). \tag{7.5}$$

**HINT: OPTIMIZATION IN PRACTICE**

In practice, we sample $m = |\mathcal{Z}| = |\hat{\mathcal{Y}}|$ times from $f$ using different random $z$ to obtain an estimated distribution set $\hat{\mathcal{Y}} = \{\hat{y}_1, \ldots, \hat{y}_m\}$ and minimize (7.5) comparing to the desired distribution set $\mathcal{Y}$.

## 7.2 Conditional Distributions

In the conditional case, the distribution of $\mathbf{Y}$ depends on $x$. Here, we return to the original formulation where $f(x, z) = \hat{y}$ is conditioned on $x$. We recall that the training pairs take on the form of $(x_n, \mathcal{Y}_n)$, where for every $x_n$ we are given a set $\mathcal{Y}_n$ of corresponding samples. During training, we then sample a set $\hat{\mathcal{Y}_n} = f(x_n, \mathcal{Z})$ of $m$ predictions for every $x_n$ and minimize

$$\sum_{n=1}^{N}\hat{D}(\mathcal{Y}_n||f(x_n, \mathcal{Z})) + \hat{D}(f(x_n, \mathcal{Z})||\mathcal{Y}_n), \tag{7.6}$$

where $f(x_n, \mathcal{Z})$ yields the set $\hat{\mathcal{Y}_n}$ of predictions obtained by feeding a constant input $x_n$ into $f$ but varying the random variable $z$.

## 7.3 Dealing with a Limited Number of Labels

For some applications it might be the case that $|\mathcal{Y}_n|$ is very small or even equal to one. For example, in future frame prediction only a single future is usually observed and that limits the ground truth labels for this problem, although in reality exponentially many futures could

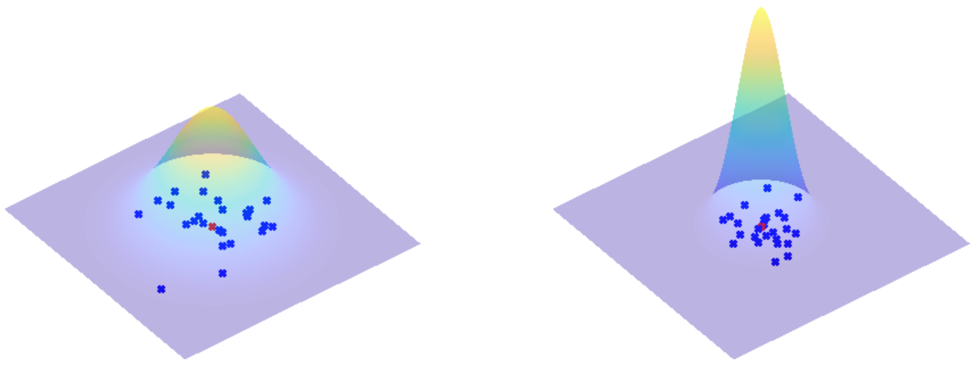


Figure 7.1. **Density estimation with nearest neighbors.** We show two distributions with different densities and samples (blue) drawn from them. The wider distribution (left image) creates on average larger distances between samples compared to the sharper one on the right.

have been possible. Furthermore, as labeling is a tedious, time consuming and sometimes expensive, many datasets provide only one annotation per $x_n$. This reduces the available training samples from $(x_n, \mathcal{Y}_n)$ to $(x_n, y_n)$.

In case of only one label being available, $\hat{D}(f(x_n, \mathcal{Z})||\mathcal{Y}_n)$ reduces to:

$$\hat{D}(f(x_n, \mathcal{Z})||y_n) = \hat{D}(\hat{\mathcal{Y}}_n||y_n) = \frac{1}{|\hat{\mathcal{Y}}_n|}\sum_{i=1}^{|\hat{\mathcal{Y}}_n|} \log \frac{\min_{j\neq i}||\hat{y}_{n,i} - \hat{y}_{n,j}||_2}{||\hat{y}_{n,i} - y_n||_2}. \tag{7.7}$$

Minimizing this condition alone does not, however, yield meaningful solutions. It would, for instance, allow that all $\hat{y}_{n,i}$ and $y_n$ form the vertices of a regular, multi-dimensional simplex:

$$||\hat{y}_{n,i} - \hat{y}_{n,j}||_2 = ||\hat{y}_{n,i} - y_n||_2 \geq 0 \;\forall i, j \neq i. \tag{7.8}$$

Let us investigate the other term, $\hat{D}(\mathcal{Y}_n||f(x_n, \mathcal{Z}))$: In case of only $(x_n, y_n)$ being available, computing $\min_{j\neq i}||y_{n,i} - y_{n,j}||_2$ is not possible as only one label $y_n$ is available. If we assume, however, that this one label is very likely, then if any other labels $y_{n,1}, \dots, y_{n,|\mathcal{Y}_n|}$ exist(ed), they would be very close to it and each other. (see Figure 7.1). As a consequence, all distances $d^*(y_{n,j}) = \min_{j\neq i}||y_{n,i} - y_{n,j}||_2$ would be very small relative to the denominator, and they would be nearly constant with respect to the optimization parameters. In other words, the more concentrated the distribution around $y_n$, the more reasonable it is to minimize just the denominator of $\hat{D}(\mathcal{Y}_n||f(x_n, \mathcal{Z}))$, i.e.

$$\min_l ||y_n - \hat{y}_{n,l}||_2^2, \tag{7.9}$$

The oracle loss in Equation 7.9 has been used in many other works, such as [56, 57, 103, 110, 157]. The intuitive explanation has been that it is desired to have at least one correct output from the network, so the output "closest to correct" is backpropagated.

**Hint: Link to Multiple Hypothesis Prediction**

This result shows that there is a clear connection to the previous chapter, where we assume that only one output sample is annotated in the data. We have gained the insight that the oracle loss is an approximation of the KL divergence using samples instead of distributions when only a single ground truth label is available. The discovery of this link shows that an MHP model does indeed also minimize the KL-divergence between its piecewise constant approximation and the target distribution even with its simplistic meta-loss formulation.

## 7.4 Random Variables in Neural Networks

After having discussed the details of minimizing KL divergence to learn a function $f(x, z) = \hat{y}$, we will now present different possibilities to represent $f$ as a neural network which can be trained using first order gradient descent methods. The key insight is that we need to introduce a random variable $z$ into the network to be able to sample from $f$ instead of computing a fixed response. This can be done in different ways.

The most straightforward approach is to concatenate a random noise image to the input of the network. This has two benefits. First, the architecture remains mostly identical except for the first layer of the network which needs to be adapted to the differently sized input. The disadvantage of this approach is that the whole network needs to be evaluated fully for every different $z$, which is inefficient and even computationally infeasible for large networks. To mitigate this problem, one can concatenate a random noise vector or feature map to an intermediate layer of the network. This has the benefit that the computation graph of the network can be executed once until the point where the randomness is introduced. Only the remaining part needs to be recomputed for changes in $z$. However, in order to be able to transform the simple (uniform or normal) distribution of $z$ into the desired output distribution, a sufficiently deep or wide network is needed after introducing $z$.

Another way to introduce randomness into the network is via dropout [173]. This has been analyzed in the context of obtaining uncertainty measures from the network in [41]. Here we can use it to create different outputs by sampling the dropout configuration. This shares the benefit that every layer up to the dropout layer can be precomputed for a specific sample, and only the layers after dropout need to be evaluated for every sample individually. Furthermore, compared to the concatenation of a randomly sampled feature map, no additional parameter is introduced into the network.

**Intuition: Noise Inputs and Multiple Outputs**

In [157] multiple outputs are generated by adding a fixed number of additional output layers. This has the benefit that a single pass through the network already yields multiple samples. While this does indeed create several samples for each input, their number is fixed. Thus, by combining the multiple output idea with one of the aforementioned techniques for introducing randomness, we can get different samples on each forward pass while retaining the computation efficiency of predicting multiple samples at the same time.

## 7.5 Density Estimation for Selecting Samples

In methods that predict multiple samples for a single input, we have observed the problem of choosing the “best” or most likely prediction. Here, we have seen that we can instead estimate the density of the distribution set $\hat{\mathcal{Y}}$ by using the distance to the $k$-th nearest neighbor (see Equation 7.3). This enables us to choose the sample with the highest estimated density in tasks for which a single answer is required.

In low dimensional output spaces (for example, in age estimation) we can also use binning to find the highest density region instead of $k$-NN. Binning becomes more difficult as the number of dimensions increases since the samples can be sparse and the number of bins grows exponentially with the number of dimensions.

# Applications

# 8

In this section we will first validate the proposed KL divergence estimation loss on a synthetic two-dimensional distribution and then evaluate the performance on the real-world task of age estimation.

## 8.1 Toy Examples

In a first experiment, we study the influence of our proposed KL divergence formulation on a simple two-dimensional example to analyze the influence of the two terms $D(P||Q)$ and $D(Q||P)$.

The example is adopted from the MHP experiments in Section 5.1, which enables us to directly compare the representation of the distributions. The said distribution changes over time $t \in [0, 1]$. This dependence on $t$ directly represents the conditional distribution $p(t, y)$ where the sampling in $y$ depends on $t$.

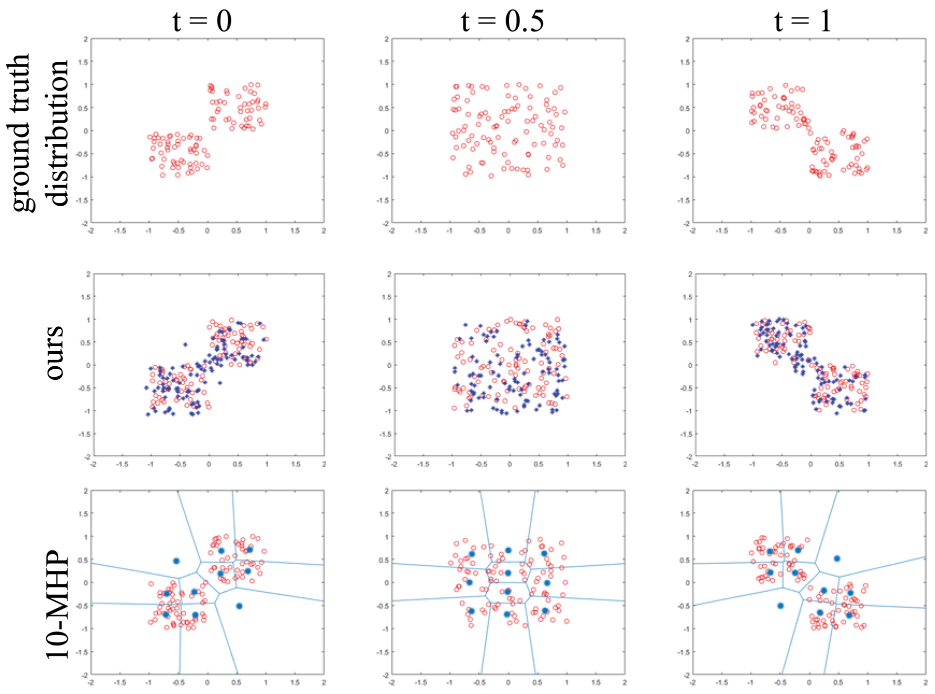


**Figure 8.1.** **2D Example.** Red points are drawn from the true underlying distribution, blue points show predictions. Additionally, we show the results of the MHP model predicting a Voronoi tessellation instead of sampled points.

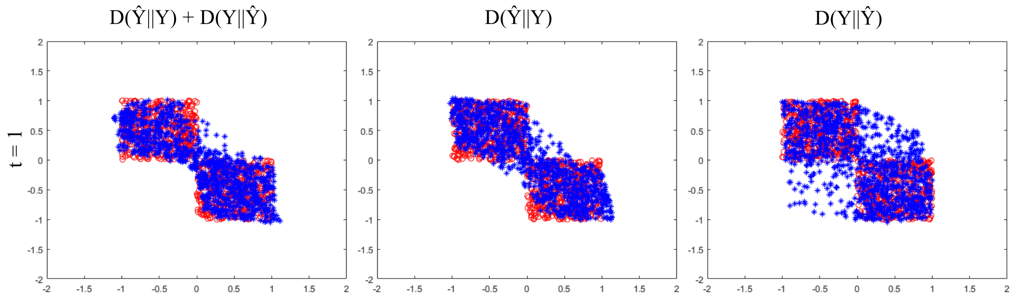


**Figure 8.2.** **Loss Analysis.** We visualize the quality of the sampling obtained from toggling both loss terms. $D(\hat{Y}||Y)$ optimizes samples to lie inside of the ground truth distribution and $D(Y||\hat{Y})$ optimizes overlap.

A zero-centered square is split into 4 equal regions. With $t$ the probability mass changes over the regions. At $t = 0$ high probability mass lies in the lower-left and top-right quadrants which transitions to the upper-left and lower-right quadrants at $t = 1$. At $t = \frac{1}{2}$ the whole square has uniform probability. The 2D space $\mathbb{R}^2$ is subdivided into five sections $S_i$

$$\begin{aligned} S_1 &= [-1,0) \times [-1,0) \subset \mathbb{R}^2 \\ S_2 &= [-1,0) \times [0,1] \subset \mathbb{R}^2 \\ S_3 &= [0,1] \times [-1,0) \subset \mathbb{R}^2 \\ S_4 &= [0,1] \times [0,1] \subset \mathbb{R}^2 \\ S_5 &= \mathbb{R}^2 \backslash \{S_1 \cup S_2 \cup S_3 \cup S_4\} \end{aligned}$$

We select the sections with probability $p(S_1) = p(S_4) = \frac{1-t}{2}$, $p(S_2) = p(S_3) = \frac{t}{2}$ and $p(S_5) = 0$. Then a point is uniformly chosen from the selected section. The distribution is shown in Figure 8.1 for the time steps $t = 0, t = \frac{1}{2}$ and $t = 1$.

We train a simple neural network with two hidden layers and 50 neurons each, using ReLU as activation function. As input to the network we feed the time $t$, concatenated with a uniformly sampled random vector $z \sim U^2(0,1)$ which introduces the necessary randomness into the network. The output $\hat{y} \in \mathbb{R}^2$ is a 2D location of the sample. The network is trained using Adam optimizer [86] and initial learning rate of $0.001$.

In Figure 8.2, we analyze the influence of the two components $D(\hat{Y}||Y)$ and $D(Y||\hat{Y})$ on the sampling. Observing the ratio inside the first term $D(\hat{Y}||Y)$

$$\log \frac{\min_j ||\hat{y}_i - \hat{y}_j||_2}{\min_l ||\hat{y}_i - y_l||_2}, \tag{8.1}$$

the numerator only operates on predictions, while the denominator selects the closest ground truth sample $y_l$ for each predictions. Thus, $D(\hat{Y}||Y)$ has the property of pulling the predictions to the closest ground truth samples. This can be seen in the middle plot of Figure 8.2 where there is almost no sample far away from the ground truth, but this term alone does not enforce the full ground truth distribution to be covered. The size of the covered region mostly depends on the distribution coverage right after initialization. If the output of the randomly initialized (and untrained) network covers the whole space of the target distribution, then $D(\hat{Y}||Y)$

Table 8.1. Estimated KL divergence at after training with different loss terms.

| | | Loss | | |
|---|---|---|---|---|
| time | estimator | $D(\hat{Y}\|\|Y) + D(Y\|\|\hat{Y})$ | $D(\hat{Y}\|\|Y)$ | $D(Y\|\|\hat{Y})$ |
| $t = 0$ | $D(\hat{Y}\|\|Y)$ | $0.34 \pm 0.02$ | $0.79 \pm 0.03$ | $0.91 \pm 0.02$ |
| | $D(Y\|\|\hat{Y})$ | $0.29 \pm 0.02$ | $0.34 \pm 0.03$ | $0.37 \pm 0.02$ |
| $t = \frac{1}{2}$ | $D(\hat{Y}\|\|Y)$ | $0.09 \pm 0.03$ | $0.13 \pm 0.01$ | $0.06 \pm 0.02$ |
| | $D(Y\|\|\hat{Y})$ | $0.09 \pm 0.03$ | $0.10 \pm 0.01$ | $0.07 \pm 0.02$ |
| $t = 1$ | $D(\hat{Y}\|\|Y)$ | $0.43 \pm 0.03$ | $0.46 \pm 0.02$ | $0.93 \pm 0.02$ |
| | $D(Y\|\|\hat{Y})$ | $0.47 \pm 0.02$ | $0.39 \pm 0.02$ | $0.31 \pm 0.03$ |

arrives at a good solution, otherwise the predicted distribution will converge with parts of the real distribution left uncovered.

Conversely, the other term, $D(Y||\hat{Y})$ accounts for selecting the closest prediction to each ground truth sample during optimization. This ensures that the whole ground truth distribution is covered but predictions that lie far away from the ground truth will never be selected and thus will not be optimized and *starve*.

The impact of both terms is crucial as they synergistically help to get a better coverage ($D(Y||\hat{Y})$) and fewer outliers ($D(\hat{Y}||Y)$) in the final learned sampler. This can also be observed in Table 8.1, where we use the described estimator $\hat{D}$ to measure KL divergence on differently trained networks at different time steps $t$. For the results reported in this table, we have sample significantly more for both the network and ground truth ($10^5$ samples each) and average over 10 experiments to arrive at a better estimate. Comparing the table to Figure 8.2, we can see that $D(Y||\hat{Y})$ stays relatively unchanged while $D(\hat{Y}||Y)$ is able to detect the failure in $t = 1$ (right most image $D(\hat{Y}||Y) = 0.93$).

## 8.2 Age Estimation

For evaluating the sampling method on a real problem, we choose the task of age estimation from single facial images. For this purpose, we use the publicly available Cross-Age Celebrity Dataset (CACD) [20], which contains 163,446 images from 2,000 celebrities, gathered via web search engines. The data is also pre-processed and released as the cropped face images. The age distribution ranges between 16 and 62 years. Each estimated age label is given by the difference between the year of birth and the year that the photo was taken. Naturally, the collection process causes inaccuracies because it assumes that the photo's time stamp is correct. The dataset is split into training (1,800 celebrities), validation (80) and testing (120) subsets. For the validation and test set, the labels are manually cleaned; while some outliers still remain, as it is sometimes also hard for humans to guess the age of a person (apparent versus real age).

Table 8.2. Quantitative results on the CACD dataset

| | DEX [151] | baseline | mean | max density | max density+augm |
|---|---|---|---|---|---|
| MAE (years) | 4.785 | 10.221 | 5.268 | 5.216 | 5.084 |

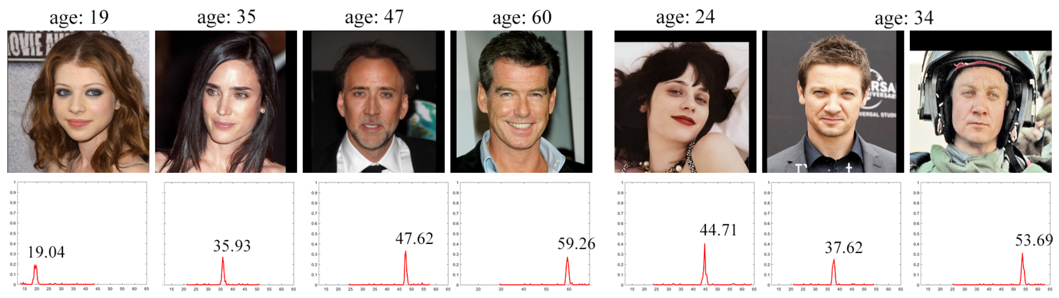


Figure 8.3. **Qualitative Age Estimation Results.** We show qualitative example of the age estimation task. In addition to the prediction we are able to obtain a histogram of samples. The last three examples depict failure cases. The last two images contain the same person.

Rothe et al. [151] observed that when using the "*clean*" validation set for training (7,600 samples) the mean absolute error (MAE) on the test set becomes significantly worse, as opposed to using the "*noisy*" 145,275 training samples. This suggests that the increased amount of training data helps, despite the inaccurate labels. Hence to facilitate comparisons we train a VGG-16 architecture [171], as in [151], on the full training set. This choice is also encouraged by the fact that the proposed method is capable of developing an inherent robustness to noisy training labels and therefore allows us to better showcase its applicability on this specific problem.

In our experiments, we replace the classification layer of VGG with a fully-connected layer with 100 outputs (similarly, DEX [151] performs classification by discretizing the age space into 100 bins). Unlike DEX [151], however, we do not perform classification but regress 100 samples simultaneously. Stochasticity is introduced by appending a small noise vector of 10 elements to the 4096-dimensional fully connected layer `fc7`, i.e. before the prediction layer as well as the usual dropout layers from VGG with dropout rate 0.5. We train the network with a batch size of 32 images for approximately 35 epochs with stochastic gradient descent, learning rate $10^{-5}$ and momentum 0.9.

During inference, the multiple outputs of the network will form distributions such as those depicted in the age histograms of Figure 8.3 (bottom row). The predicted age is derived as the mean over all samples that fall into the maximum density bin of the histogram. In the figure, we show both success and failure cases. The two right-most images of Figure 8.3 represent a special case in which one can observe the impact that facial characteristics have on the outcome. The same person, at the same age can receive a very different estimation based on changes in their expression, hair, etc. Interestingly, we also found several inaccurately labeled images in the test set, fact which sometimes leads to predictions that are more reasonable – according to our own guesses – than the corresponding ground truth.

In Table 8.2 we report our experiments. We first construct a single-output regression baseline, which scores 10.2 years in MAE. The proposed sampler with 100 outputs remarkably outperforms this baseline and results in an error of $5.2$ years. Notably, when we use the proposed density estimation to select the sample with the highest density (max density) we gain an improvement in comparison to merely computing the mean of the predicted distribution ($5.3$ years). When additionally performing a series of small random augmentations on the input and aggregating all network outputs to the age histogram, the performance is further improved to $5.08$, due to the increased number of samples drawn from the model. We finally report the state of the art method DEX [151], to which we show comparable performance despite the fact that they use a large amount of additional data for pre-training (over 200k image from the IMDB-WIKI dataset).

# 9 Conclusions

We have presented a method that pushed the multiple hypothesis idea from a fixed number of discrete samples (hypotheses) to learning to sample from a target distribution. Naturally, this richer output requires a more complex training procedure, extensive annotations – having labeled *distributions* instead of single outcomes is preferable here – and is more difficult to train than MHP. Nonetheless, we are able to show good performance for a 2D toy task and age estimation, where the data is given in the right format for this method to learn.

We would like to further investigate the link to (conditional) variational autoencoders (VAEs). A clear difference, for example, is the way a conditional VAE learns a manifold representing the data and samples from it. However, an understanding on what effects these differences have in practice could be useful.

# Part IV

Asking Humans for Help

# Introduction 10

In this chapter we will have a look at another powerful way to deal with ambiguity. The insight is, that often a human user can resolve an ambiguity because either she has additional outside knowledge, or he is the source of the confusion, as the method tries to understand her goals. This chapter is composed of two publications, one from the beginning of my Ph.D. [156] before I came in contact with deep learning, the other at the very end fully powered with CNNs, RNN and lots of data [155], both published at CVPR.

## 10.1 Related Work

Since this chapter covers two methods, we structure the related work into different categories, related to individual and common parts between the two approaches. The related work spans a large variety of work before and after the deep learning paradigm shift.

### 10.1.1 Twenty Questions

The Twenty Questions setting has been explored for computer vision applications by Branson *et al*. [13] and its extensions [187] and [12] for interactive fine-grained image classification. Their work sets up collaborative task between human and algorithm, where the system knows important questions but needs human supervision to find the correct answer – especially since during that time image understanding technology was not good enough to detect and describe the color and shape of a bird's beak for example. The method combines the expertise of the computer with the visual abilities of the human to collaboratively discover the object's class. A Bayesian view of the twenty questions problem can be found in [73]. We have extended the method presented here in a follow-up work on medical segmentation, where we integrate a learned shape prior [33].

### 10.1.2 Interaction with neural networks

The problem of dynamic user interaction with deep learning agents such as neural networks, is not yet extensively studied in literature. One problem is that integrating a human into a training loop that incorporates thousands of images is difficult. Nonetheless, some methods regarding interactive (deep) learning exist. For example weakly-supervised semantic segmentation via human scribbles [112], user-provided clicks and Euclidean distance transforms [200] or initialization with bounding boxes [29, 146]. For the task of image or sketch colorization, more and more methods allow for human interaction [204]. [2] propose to interactively improve medical image segmentation with user-specified seeds and also in the medical domain [189] uses a second network to process the original prediction and the user's feedback.

The system presented in the second part, can substitute the human input with an algorithm that dynamically generates textual hints based on previous predictions, to be able to integrate large scale feedback into the training. The original CNN is already trained and its weights remain unchanged. We only learn a hint comprehension network that modifies the activations of the original CNN on the fly. This is a major difference between our approach and most of the previous systems, as they depend on the user's input to produce a prediction, therefore they do not function actively.

### 10.1.3 The intersection of vision and language

To enable user interaction in a natural and intuitive way, we propose a novel idea that lies in the joint domain of natural language and vision. Visual Question Answering (VQA) is a task involving problems at the crossroad of natural language and vision. VQA systems receive and input image together with a question and are tasked to provide a usually unrestricted and open-ended answer [1, 205]. A common approach is to ground the question in the input image to be able to focus on the relevant regions which are necessary to reason about the answer [4, 5, 68, 78, 118, 122]. The CLEVR dataset [76] further facilitates unveiling the decision making process through a set of visual reasoning tasks related to the relationship and attributes of simple objects in an image. Other examples in the joint text-vision domain include image captioning [28, 77, 185], visual dialog [30] and segmentation or object retrieval from referring expressions [69, 70]. Most other work combines CNNs and RNNs using visual and textual attention mechanisms [3, 119, 198, 199]. Closely related to our second work [155] is the conditioning layer of Perez *et al*. [139] for visual reasoning.

### 10.1.4 Semantic Segmentation (pre deep learning)

Before CNNs drastically changed the field, interactive segmentation techniques usually relied on seeds [11, 52, 144] or bounding boxes [53, 104] placed by a human in or around the object of interest. Closer to our first work, are a few approaches [8, 38, 91, 92, 174] that keep the human user in the loop and suggest the most informative areas to label next. This is similar to active learning where the system learns online, during runtime, from a user. An important aspect of the twenty questions game is the intrinsic ambiguity of the task. One cannot anticipate which region and at what semantic level was picked by the oracle. Tu and Zhu [181] augment the active contours idea with a data-driven MCMC framework to generate several parsings of the same image. Object proposal methods identify a set of candidate regions in a scene [19, 35, 54, 93] by learning typical visual properties of objects.

### 10.1.5 Semantic Segmentation (post deep learning)

Both methods are demonstrated on the application of semantic image segmentation which was significantly boosted by the success of deep learning methods [7, 21, 74, 113, 117, 142]. Since our contributions do not encompass an improved image segmentation method but rather build on top of existing architectures, we deploy out-of-the-box, state-of-the-art models [21, 117].

# 11 Twenty Questions

Our first method [156] can be summarized as follows. The goal for the system is, after twenty (or a fixed number of) questions, to propose a segmentation that is as close as possible to what the human/oracle had in mind. The user is only allowed to answer yes or no, which translates to one bit of information per question. This renders the task extremely under-constraint and ambiguous as $2^{20} \ll 2^{640\times480}$ of possible segmentations for a typical image. Here, we choose to have a fixed type of question: *"Is this pixel inside the region you want to segment?"*.

**INTUITION: QUESTION TYPES**

It is possible to extend the range of questions. In fact, while developing this system we experimented with all sorts of questions such as "*Is your object to the left of this line?*" or "*Is your object mostly blue?*". However, for this work we settled for asking for pixels, as it has several advantages. It has the property of converging to the ground truth segmentation after asking as many questions as there are pixels. Questions about color and shape can be imprecise and depend on the interpretation of the user. Asking for lines or regions always leaves the question of what to do if the object intersects the boundary.

With the type of question being fixed, the system needs to find the optimal pixel location to maximize the information brought by the answer. Formally, we define a probability distribution over all possible segmentations $\mathcal{S}$, measuring how likely it was picked by the oracle. The probability consists of two components: first the intrinsic segmentation quality in terms of image partitioning (*i.e.* the homogeneity of the segmented object), and second a dynamic term based on the feedback of the user. An ideal question choice would try to divide $\mathcal{S}$ into two parts of equal probability sum. However, since $\mathcal{S}$ contains an excessively large amount of items, the probability distribution is intractable for explicit computation. Instead, we compute a Monte Carlo approximate by sampling segmentations proportionally to their (estimated) probability. The Markov Chain Monte Carlo (MCMC) framework is ideal for sampling from a complex probability density function. It builds upon the Metropolis-Hastings algorithm [128] and explores $\mathcal{S}$, spending more time in areas of high probability. This makes computing the most informative question more tractable and allows for the knowledge obtained from the answer to be directly integrated into the next round. To assess the performance of the method, we evaluate it on three public datasets and compare it to several baselines.

## 11.1 Mathematical Derivation

The algorithm is divided into two alternating parts. Using the Metropolis hastings algorithm, we first sample a set of likely segmentations. Then, the most informative question is computed and asked to the user. Based on their answer, we update the fore- and background model of the Markov chain. Both steps are repeated until the maximum number of questions is reached or the user is satisfied with the current segmentation outcome.

### 11.1.1 Problem Statement and Notations

We define an image $\mathcal{I}$ over the lattice $\Omega = \{1, \ldots, h\} \times \{1, \ldots, w\}$ where $h$ and $w$ denote height and width of $\mathcal{I}$. A binary segmentation of the image $\mathcal{I}$ is then a function $\mathbf{s} : \Omega \to \{0, 1\}$ indicating whether the pixel $(x, y)$ of the image is inside ($\mathbf{s}(x, y) = 1$) or outside ($\mathbf{s}(x, y) = 0$) the segmented area. Intuitively, as $wh$ is finite, $\mathbf{s}$ can also be modeled as a vector $\mathbf{s} \in \mathcal{S} = \{0, 1\}^{|\Omega|}$, explicitly listing the inside-status for each pixel.

We formalize the twenty questions problem as follows. For a given image $\mathcal{I}$, the oracle (*i.e.* the human user) decides on a segmentation $\hat{\mathbf{s}} \in \mathcal{S}$. This segmentation is hidden from the algorithm as it only exists in the user's head. To discover the segmentation, the computer is allowed to ask a series of binary questions of the form $Q(\mathbf{p})$: "*Is the displayed location* $\mathbf{p}$ *inside the object you want to segment?*" to the oracle. With the question text fixed the task boils down to finding the most informative location $\mathbf{p}$. The oracle's answer provides the true label $l(\mathbf{p}) \in \{0, 1\}$ at this location $\mathbf{p}$.

Depending if the answer was yes or no, we collect them in two sets $\Sigma_-^k$ and $\Sigma_+^k$ of background and foreground seeds respectively, with $|\Sigma_-^k| + |\Sigma_+^k| = k$. These we call these sets *reliable* sets of seeds as the state for these pixels has been directly specified by the oracle.

**INTUITION: RELIABLE SEEDS**

During the rest of the mathematical derivation, we assume that the answer given by the oracle is always correct. However, in the experiments, we will validate what happens if the oracle was not fully reliable and would sometimes respond with the wrong answer.

With $\Sigma^k = \Sigma_-^k \cup \Sigma_+^k$ we denote the set containing all reliable seeds. We encode this knowledge through the Bayesian posterior probability $p(\mathbf{s} = \hat{\mathbf{s}} | \Sigma^k)$ over the set of segmentations $\mathcal{S}$. It states how likely it is that $\mathbf{s}$ is the hidden segmentation. It is informed by the known seeds revealed by the oracle. For brevity, we will write $p(\mathbf{s}|\Sigma^k)$ for this probability in order to clarify that it is seen as a function of $\mathbf{s}$ and as a probability distribution over $\mathcal{S}$. As stated before, if we could compute this posterior for every possible segmentation in $\mathcal{S}$ (which is finite for finitely large images), an optimal divide and conquer strategy would at each turn find a question location $\mathbf{p}$ that halves the set of possible segmentations into two subsets of equal probability $0.5$. However, with $2^{|\Omega|}$ possibilities, the set $\mathcal{S}$ is far beyond a size for exhaustive computation of $p(\mathbf{s}|\Sigma^k)$ over the whole set $\mathcal{S}$. For comparison a $10 \times 10$ image already has far more possible segmentations ($2^{100}$) than the estimated number of atoms in the universe $2^{83}$. However, we can assume that the segmentation picked by the user is corresponding to some semantic object in the image and has some visual support, so that $p(\mathbf{s}|\Sigma_-^*, \Sigma_+^*)$ is very unbalanced (*i.e.* higher from several orders of magnitude for a few good candidates that are likely to be picked). This means that the distribution $p(\mathbf{s})$ does not have equal probability even without any knowledge of the user. To overcome the astronomically large size of $\mathcal{S}$, we propose to approximate the posterior $p(\mathbf{s}|\Sigma^k)$ at each iteration $k$ by a series of samples $\mathbf{s}_1^k, \dots, \mathbf{s}_N^k \in \mathcal{S}$. These samples are drawn from a Markov Chain Monte Carlo (MCMC) scheme, which we describe in detail in Sec. 11.1.2. We use $N$ samples to compute an approximation of the most informative question by following the question selection method exposed in Sec. 11.1.3. Sampling and question selection are then alternated until a predefined amount of allowed questions is reached or the oracle deems the current estimate correct.

```
Input   : Image $\mathcal{I}$, number of allowed questions $K$
$\Sigma_-^0 \leftarrow \emptyset$; $\Sigma_+^0 \leftarrow \emptyset$;
for $k \leftarrow 0$ to $K - 1$ do
    Sample $\mathbf{s}_1^k, \dots, \mathbf{s}_N^k$ from $p(.|\Sigma^k)$;
    Find most informative location $\mathbf{p}_k$ w.r.t $\mathbf{s}_1^k, \dots, \mathbf{s}_N^k$;
    Ask question $Q(\mathbf{p}_k)$ and receive true label $l(\mathbf{p}_k)$;
    if $l(\mathbf{p}_k) = 0$ then
        $\Sigma_-^{k+1} \leftarrow \Sigma_-^k \cup \{\mathbf{p}_k\}$;
        $\Sigma_+^{k+1} \leftarrow \Sigma_+^k$;
    else
        $\Sigma_-^{k+1} \leftarrow \Sigma_-^k$;
        $\Sigma_+^{k+1} \leftarrow \Sigma_+^k \cup \{\mathbf{p}_k\}$;
    end
end
$\mathcal{G} \leftarrow$ GDT-segmentation with seeds $\Sigma^K$;
Output : Guess $\mathcal{G}$ of the oracle segmentation
```

**Algorithm 2: Overview of our method.** At each step, the question selection strategy consists in sampling segmentations according to their probability of having been picked by the oracle, and asking the question related to the most informative location with respect to these samples. This location, together with the provided label form a seed is added to either $\Sigma_-^0$ or $\Sigma_+^0$ depending on its label.

### 11.1.2 Sampling Likely Segmentations with MCMC

In this section, we show how to sample representative segmentations from the posterior probability distribution $p(.|\Sigma^k)$ conditioned on the current knowledge. We follow a MCMC-based approach and explore the set of segmentations, spending more time on segmentations that are likely to have been picked by the oracle. We first review the Metropolis-Hastings algorithm and discuss the adaptation to our method. We build upon the classical Metropolis-Hastings algorithm [128] and an original idea from Tu and Zhu [181] introduced in the context of image parsing into multiple regions. Our sampling procedure is defined as a Markov chain over the space of segmentations with the following transition probabilities. Given a current state (which is a segmentation $\mathbf{s}$), a new segmentation candidate $\mathbf{s}'$ will be suggested according to a *proposal distribution* $q(\mathbf{s}'|\Sigma^k, \mathbf{s})$ to be the next state of the Markov chain. This *move* is accepted or rejected with probability $\min(1, \alpha)$ and

$$\alpha = \frac{p(\mathbf{s}'|\Sigma^k)q(\mathbf{s}|\Sigma^k, \mathbf{s}')}{p(\mathbf{s}|\Sigma^k)q(\mathbf{s}'|\Sigma^k, \mathbf{s})}, \tag{11.1}$$

where $p(.|\Sigma^k)$ denotes the *posterior probability distribution* according to which we want to draw samples. This is the probability for a segmentation to have been picked by the oracle given the already known seeds $\Sigma^k$. The algorithm starts from an initial segmentation $\mathbf{s}_0^k$ and generates a succession of segmentations. To achieve a greater diversity of segmentations, the $\mathbf{s}_i^k, 1 \leq i \leq N$ are selected as samples at a fixed rate during this exploration process. Additionally, as typical in MCMC methods, a burn-in step is performed before collecting samples so that they lose their dependency on the initial state $\mathbf{s}_0^k$. We will now define the posterior distribution and the proposal distribution which are critical for the MCMC process. They respectively define how to rate the likelihood of a segmentation to have been picked by the user and how to move from a segmentation to another in the Markov chain process.

### State-Space Parametrization

To keep the system responsive to the user, the processing time between two questions needs to be kept low. For this reason, we introduce an alternative parametrization of the segmentation space based on geodesic distance transforms (GDT). GDTs have been shown to be computationally efficient for example in the context of object proposals [93].

**INTUITION: STATE-SPACE PARAMETRIZATION**

The reason that makes a parametrization of the state space attractive is that we need to suggest moves of the Markov chain. If we keep the original, pixel-wise formulation, moves are either very complex, or they can only change the segmentation very locally, *i.e.* flipping a pixel. With the following parametrization, we can not only allow larger changes in the segmentation but also are able to integrate the previous answers $\Sigma^k$ in a very natural and intuitive way to adjust the distribution based on the oracle's hints.

We define $\mathcal{X} = \{1, \ldots, C\} \times [0, B] \times \mathcal{P}(\Omega)^2$ as our state space and consider the function $\phi^{GDT} : \mathcal{X} \to \mathcal{S}$ that associates to a vector $\mathbf{x} = (c, \beta, \Sigma^+, \Sigma^-) \in \mathcal{X}$ to the segmentation $\phi^{GDT}(\mathbf{x}) \in \mathcal{S}$. The segmentation is obtained by computing two geodesic distance transform on the $c^{th}$ image channel that was blurred with a Gaussian of standard deviation $\beta$, one with the sets of positive $\Sigma^+$ and one using the negative seeds $\Sigma^-$. We denote the number of color channels in the image with $C$, and the maximally allowed standard deviation for Gaussian blurring with $B$. We obtain the GDT by computing the shortest distance between every pixel and the set of seed pixels. Typical GDTs define the distance between two neighboring pixels as a convex combination of the Euclidean distance and the intensity gradient. However, since the location of our seeds is chosen by an algorithm and since the segmentation is unknown during seed placement, they do not always fall perfectly inside the center of regions. In typical applications, the user places the seeds, knowing the segmentation, thus, they will not be placed unnecessarily close to boundaries. Here however, the segmentation is unknown to the algorithm that places seeds. We thus only use the squared intensity difference to remove the dependency to the distance between seed and object boundary. Each pixel is labeled as inside or outside depending whether it is geodesically closer to an inside or outside seed. We introduce the coefficient $\beta \in [0, B]$ to control the sensitivity to the scale of edges in the image. Implicitly, it generates an image pyramid during the MCMC process (see Sec. 11.1.2) allowing for coarse and fine segmentations. This GDT-based segmentation algorithm is further illustrated in Fig. 11.1.

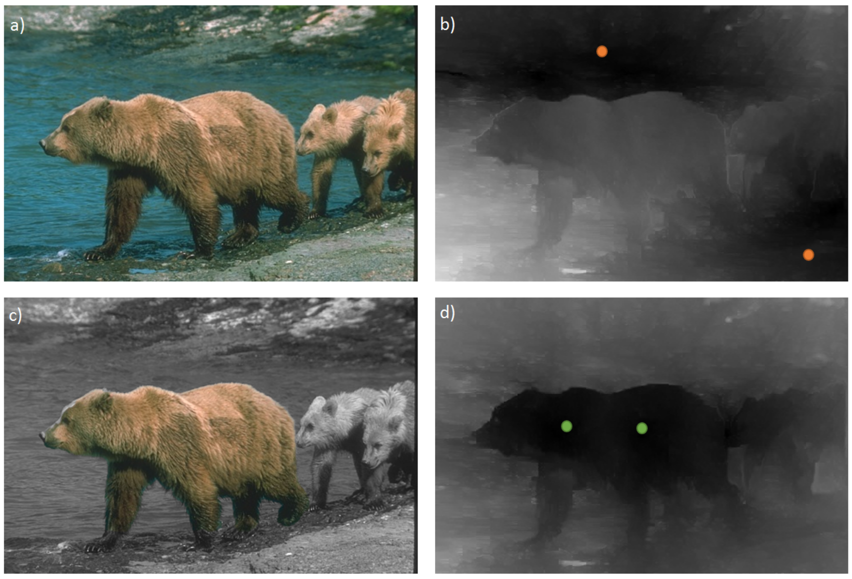


**Figure 11.1.** **Segmentation example using geodesic distance transforms.** a) Image from the Berkeley 300 dataset [125]. b) + d) Approximated geodesic distance to outside (orange) and inside (green) seeds respectively. c) Final segmentation.

**HINT: SEGMENTATION ALGORITHM**

Note that any other seed-based interactive segmentation algorithm could be included at this stage of the framework. Our main motivation behind the choice of geodesic distance transform was the fact that it can be approximated in linear time [179] and were thus crucial to keep the segmentation time low between individual questions. In fact, in [155] we show that we can modify a pre-trained segmentation CNN to be able to work with seeds. Using deep networks for segmentation became popular with [117] which was – just as our original twenty questions segmentation idea – published at CVPR 2015.

With the above definitions, we build the Markov chain for segmentation sampling to act at the state space level $\mathcal{X}$. Thus, it modifies the GDT-based segmentation parameters and seeds and Eq. 11.1 becomes

$$\alpha = \frac{p(\phi^{GDT}(\mathbf{x}')|\Sigma^k)q(\mathbf{x}|\Sigma^k, \mathbf{x}')}{p(\phi^{GDT}(\mathbf{x})|\Sigma^k)q(\mathbf{x}'|\Sigma^k, \mathbf{x})}. \tag{11.2}$$

In the following, we will expose our design for $p(\phi^{GDT}(\mathbf{x})|\Sigma^k)$ and $q(\mathbf{x}'|\Sigma^k, \mathbf{x})$ respectively.

## Posterior Probability

As described above, $p(\mathbf{s}|\Sigma^k)$ is the probability of the segmentation to have been picked by the user, given the set of $k$ seeds $\Sigma^k$ already revealed by the first $k$ questions. Interestingly, in the Metropolis-Hastings acceptance probability (Eq. 11.2) uses only the *ratio* of the probabilities $p(\phi^{GDT}(\mathbf{x}')|\Sigma^k)$ and $p(\phi^{GDT}(\mathbf{x})|\Sigma^k)$. Hence, constant factors such as normalization are canceled out and do not play any role. We can thus design this distribution without taking normalization into consideration. We propose to distinguish two cases depending on whether there is at least one background seed and one foreground seed in $\Sigma^k$. The intuition is that before we know at least one seed in fore- and background, we will use a simple prior to guess what a good segmentation is. After we found one seed each, we can use them to build better fore- and background models.

**Case** $1$**:** $\Sigma^k_+ = \emptyset$ **or** $\Sigma^k_- = \emptyset$ This case occurs during the first questions, corresponds to the absence of seeds for at least one of the two labels. We define the segmentation probability in that case as

$$p(\mathbf{s}|\Sigma^k) \propto \frac{1}{1 + Var(\{\mathcal{I}(\mathbf{p}), \mathbf{p} \in \mathbf{s}^{-1}(1)\})}, \tag{11.3}$$

where $Var(\{\mathcal{I}(\mathbf{p}), \mathbf{p} \in \mathbf{s}^{-1}(1)\})$ denotes the intensity variance inside the segmentation $\mathbf{s}$. This variance is averaged over all color channels and intuitively encourages segmentations which delineate homogeneous regions as foreground. This is a naive model that will be replaced by a better approximation once at lease one seed for each, fore- and background is known.

**Case** $2$**:** $\Sigma^k_+ \neq \emptyset$ **and** $\Sigma^k_- \neq \emptyset$ We can now use the GDT segmentation algorithm to build a more accurate model of the visual properties of the background and foreground regions. Intensity histograms are a good proxy to model image region properties and have been used extensively in active contour methods for segmentation. We compute the two GDT segmentations using the known seeds on each color channel and compute background and foreground intensity histograms $H^k_-$ and $H^k_+$ aggregated over the color channels. Each intensity value is weighted by its inverse geodesic distance to model that the confidence decreases with increasing geodesic distance to the seeds. A new segmentation can then be scored by computing the similarity between the background and foreground histograms of $\mathbf{s}$ denoted $H_-(\mathbf{s})$ and $H_+(\mathbf{s})$ and the current estimates $H^k_-$ and $H^k_+$ via a chi-squared distance:

$$p(\mathbf{s}|\Sigma^k) \propto \frac{1}{1 + \frac{1}{2}\sum_{\delta \in \{-,+\}} \chi(H^k_\delta, H_\delta(\mathbf{s}))}. \tag{11.4}$$

## Proposal Distribution

A set of parameters $\mathbf{x} = (c, \beta, \Sigma^+, \Sigma^-) \in \mathcal{X}$ is the input to the GDT segmentation algorithm. This parametrization of the state space $\mathcal{X}$ gives a more natural way to move from one state to another and facilitates the design of the proposal distribution $q(.|\Sigma^k, \mathbf{x})$. We maintain two sets of seeds $\Sigma^+$ and $\Sigma^-$ which contain both the already known, reliable seeds $\Sigma^k_+$ and $\Sigma^k_-$ (*fixed* seeds) and a new set of seeds created by the MCMC process which we call *mobile* seeds. We use five types of moves to traverse the Markov chain from a given state $\mathbf{x}$ to a new state $\mathbf{x}'$:

1. *Changing image channel*: The image channel $c$ is redrawn uniformly.

2. *Changing* $\beta$: The smoothing parameter $\beta$ is redrawn uniformly.

3. *Adding a seed*: A mobile seed is added at a random (non seed) location and randomly added to $\Sigma^+$ or $\Sigma^-$.

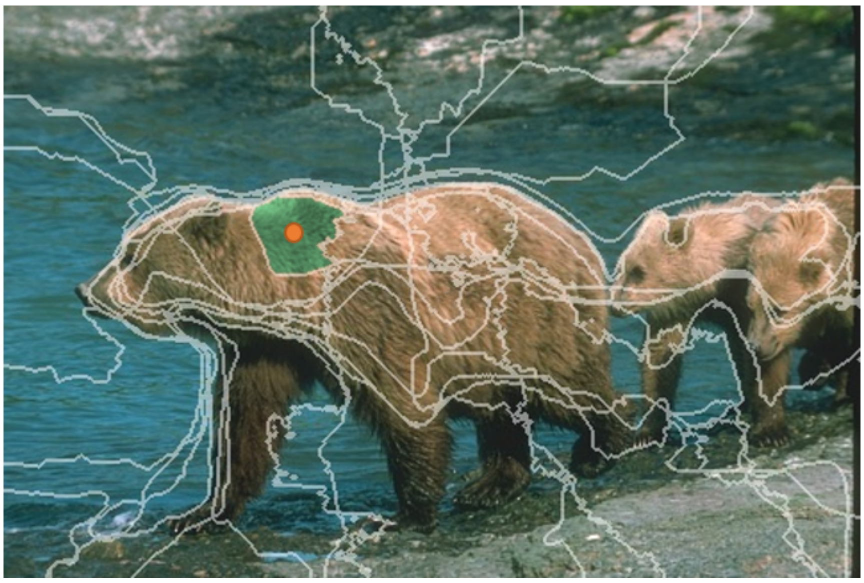


Figure 11.2. **Over-segmentation of the image for the question selection step.** The over-segmentation is obtained by intersecting all $\mathbf{s}_i^k, 1 \leq i \leq N$. The most uncertain region $R_j^*$ is shown in green and the question pixel $\mathbf{p}_k$ (shown in orange) is chosen as the most interior point of $R_j^*$.

4. *Removing a seed*: A mobile seed is removed from either $\Sigma^+$ or $\Sigma^-$ (if this does not leave this set empty).

5. *Moving a seed*: A mobile seed is moved spatially to a (non seed) pixel according to a normal distribution.

We draw moves again with uniform probability. This completes the definition of the proposal distribution $q(.|\Sigma^k, \mathbf{x})$.

**INTUITION: NUMBER OF SEEDS**

To avoid that the number of mobile seeds explodes, we balance the probabilities of picking the moves 3. and 4. such that forward and backward moves are equally likely. This means that once a seed was added, removing seeds becomes more likely and vice versa.

### 11.1.3 Question Selection

We draw $N$ segmentations $\mathbf{s}_i^k, 1 \leq i \leq N$ to approximate the probability distribution $p(\mathbf{s}|\Sigma^k)$ after $k$ questions have been asked and answered ($k \geq 0$) using the method previously described in Section 11.1.2. Given these $N$ samples, we have to select on the optimal question (location $\mathbf{p}$) for the oracle.

As shown in Figure 11.2, we first perform an over-segmentation of the image by intersecting all sampled segmentations $\mathbf{s}_i^k$. This partitions the image into regions $(R_j)_{1 \leq j \leq \rho}$ based on the samples $\mathbf{s}_i^k$.

**INTUITION: OVER-SEGMENTATION**

The intuition behind the over-segmentation is as follows. Since the segmentation samples $\mathbf{s}_i^k$ are the only available information, the information carried by each pixel is constant over an individual segment. Thus we can simplify the problem by reasoning about region $R_j$ instead of pixels.

A region $R_j$ belongs to the object of interest with probability $p(R_j \subset \hat{\mathbf{s}})$, which can be approximated from the samples as

$$p(R_j \subset \hat{\mathbf{s}}) = \frac{1}{\sum_{i=1}^{N} p(\mathbf{s}_i^k|\Sigma^k)} \sum_{i|R_j \in \mathcal{S}_i} p(\mathbf{s}_i^k|\Sigma^k). \tag{11.5}$$

The most valuable region is the one for which this probability is the closest to $0.5$, as it is the most uncertain between fore- and background. However, we weigh each segment by its size to account for the fact that – as previously mentioned – each pixel will equally benefit from the information carried by the segment it belongs to. To find the most informative region, we select the segment $R_j^*$ that maximizes $|R_j|(1 - |1 - 2p(R_j \subset \hat{\mathbf{s}})|)$. Now, we could choose any location $\mathbf{p}_k \in R_j^*$ to define the selected question $\mathcal{Q}(\mathbf{p}_k)$ as in theory they carry equal information. In practice, we choose for $\mathbf{p}_k$ the most interior point of $R_j^*$ (Fig. 11.2), which is usually the centroid of the region. Special care has to be taken when the centroid does not fall within the segment. The closest point to the centroid is chosen in these cases instead. Interestingly, in our experiments we found that this results in questions that are in general easier to answer visually as the most interior point typically lies far from the edges of objects and the image boundaries.

## 11.2 Experiments

Since we can replace the actual human with an oracle that has access to the ground truth segmentation, we can perform extensive experiments quickly and fully automatic. In the following section we will demonstrate the effectiveness and robustness of the approach.

### 11.2.1 Experimental Setup

All hyper parameters are constant for all experiments. We represent images in the CIELab color space and the standard deviation of the Gaussian smoothing applied during the MCMC process (Section 11.1.2) is set to $\beta = 6$ pixels. We *burn in* the Markov chain for $1000$ steps to gain independence from the starting state $\mathbf{s}_0^k$. Between each sample, we move $250$ steps along the Markov chain to ensure a strong decorrelation between two consecutive samples. For each question we sample $N = 32$ Segmentations which was empirically found to be sufficient to produce enough variety between the segmentations such that the over-segmentation does not become too fine grained without exceeding a limit of one second between two questions. Despite the linear-time approximation, the main bottleneck remain the computation of the

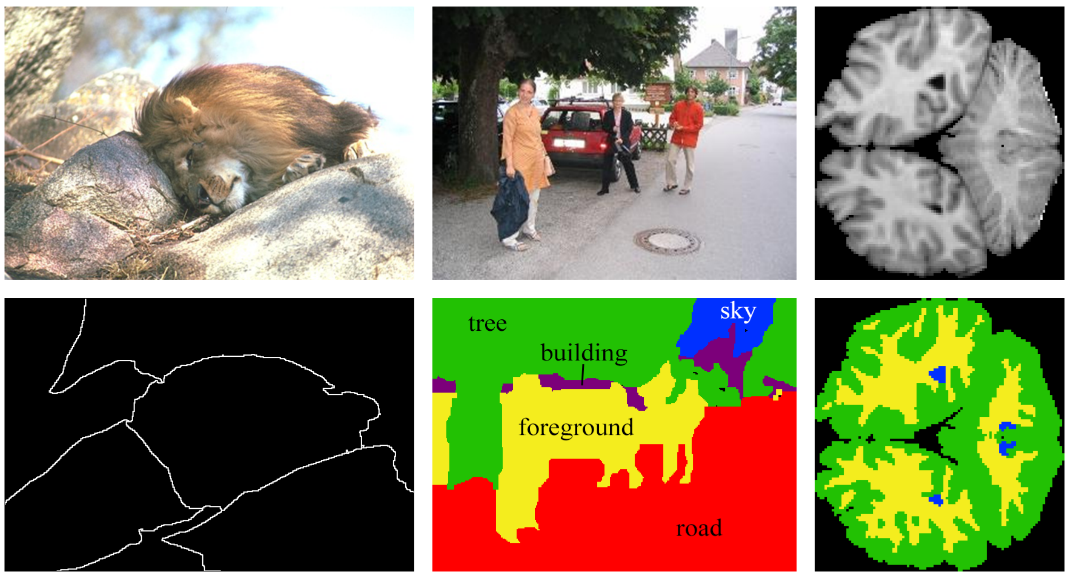


Figure 11.3. **Representative images from the datasets.** From left to right: (1) Berkeley Segmentation Dataset, (2) Stanford Background Dataset, (3) IBSR Brain Dataset. Note the diversity in terms of visual content (natural scenes (1,2) *vs* medical data (3)) and labels (unsupervised parsing (1) *vs* semantic labeling (2,3)).

GDT, since it is required at every MCMC step. The total complexity of our algorithm is $\mathcal{O}(NKwh)$. For further performance, we run four Markov chain in separate threads in parallel and each chain using down-sampled images between the segmentations samples. The drawn segmentations steps are however computed on the full image to ensure fine details in the segmentation. In our experiments, we evaluated for up to $K = 30$ questions. A great advantage of choosing our particular question type is that the human user can be simulated by the ground truth segmentation, which allows an extensive evaluation over a large set of images.

## 11.2.2 Results

Since foreground background segmentation is a two class problem, we can use the Sørenson index or Dice score [172], which measures the overlap between the segmentation and the ground truth, to measure the quality of a segmentation. Since the twenty questions segmentation task is novel in its self, we compare our method against two intuitive and strong baselines. The first evaluates the impact of the proposed question selection approach, by choosing each question location randomly. Our second baseline is an extension of the first. During questioning, when seeds have been found inside and outside the hidden segmentation, we use our GDT-based segmentation to compute a pixelwise confidence to select the question within the area where the label uncertainty is maximal. In other words, we ask questions on the border between inside and outside which can be seen as an iterative refinement of the segmentation border. To demonstrate generalization, we evaluate using three different datasets. In the following, we will introduce all three datasets in detail.

**Berkeley Segmentation Dataset 300** The Berkeley Segmentation Dataset 300 [125] consists of $300$ natural images. Several ground truth human annotations are available for each image

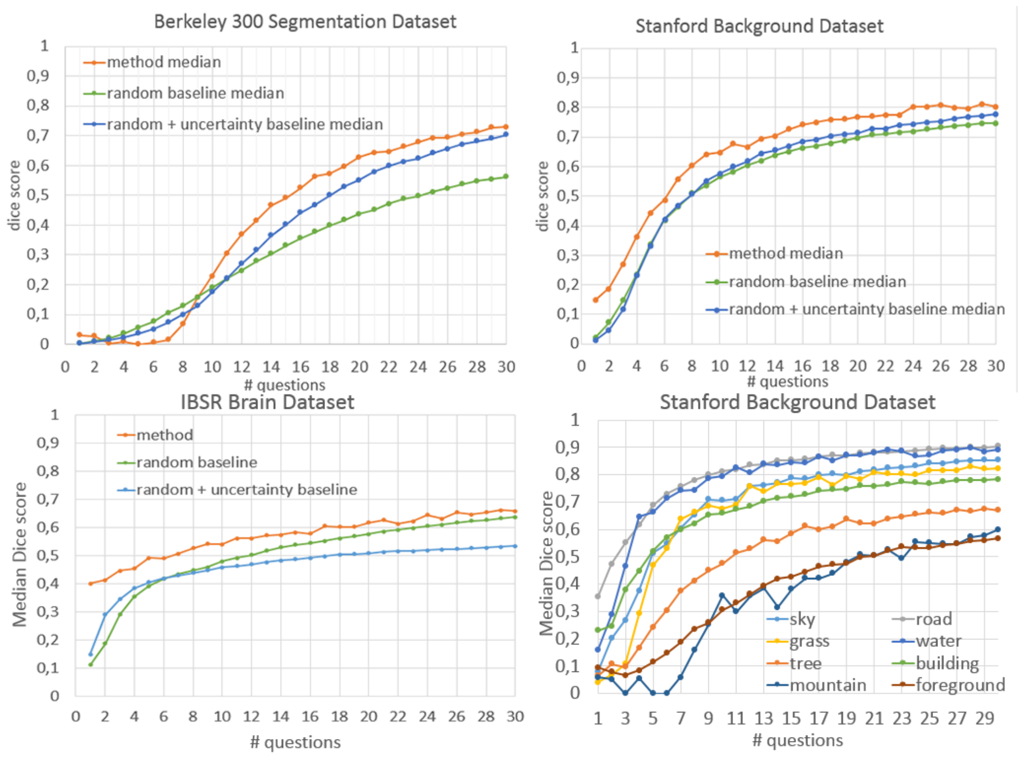


**Figure 11.4.** **Evolution of the median Dice scores with the number of questions.** The performance is shown on the three datasets: Berkeley Segmentation Dataset 300 [125], Stanford Background Dataset [51] and IBSR Brain Dataset. We also show, on the Stanford background dataset, the individual performance for each class label. The small amount of mountain labels in the dataset explains the noise on the corresponding curve.

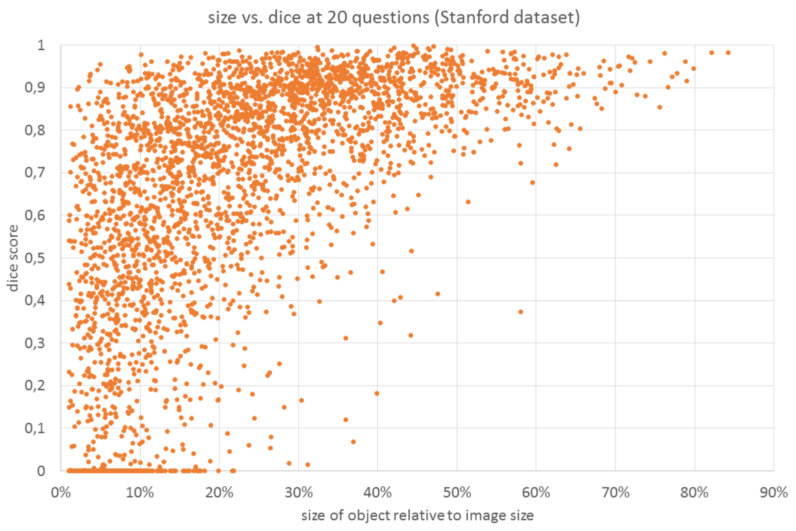


**Figure 11.5.** **Relationship between size of the hidden segmentation and performance.** Shown is a scatter plot of relative object size (x-axis) *vs* dice score (y-axis) after 20 questions. Large areas are easier to guess than small ones.

defining a parsing of the image into several regions. This nicely illustrates the complexity of the task, since the segmentations differ from one annotator to another. T This shows that the definition of *what is an object* is highly user dependent and thus must heavily rely on user feedback. On average, there are $20.3$ regions per image each of which will be selected as ground truth during the automatic evaluation. The graph in Figure 11.4 displays the median Dice score over all runs with all possible segments when the number of allowed questions increases, for both our method and the aforementioned baselines. The curves show low performance for the first 6-7 questions. This corresponds to the initial stage where no reliable foreground seed has been found yet. During that stage the sampled segmentations are very noisy since they are only based on the mobile seeds from the Markov chain. The median number of questions required to find a reliable seed is approximately $7$. The dataset contains $20$ non-overlapping regions per image in average. Hence, even if this pool of $20$ possible ground truth segmentations was known in advance (which it is not), a naive algorithm would still take $10$ questions to find a foreground seed. This demonstrates that our question selection procedure identifies regions more efficiently and is able to utilize the information given by previous answers well.

**Stanford Background Dataset** Our second dataset is the Stanford Background Dataset [51] – a diverse collection of $715$ natural images sampled from different other datasets. The images have been labeled into regions of eight different semantic categories: sky, road, grass, mountain, water, tree, building and foreground object. We chose this dataset as this is very different to the Berkeley Segmentation Dataset where images were parsed into arbitrary user-defined regions. Figure 11.4 shows quantitative results over the whole dataset but also separately per class. Here we notice that homogeneous and larger regions such as *road* are easier to find than complicated foreground objects. Figure 11.5 shows the relationship between Dice score after $20$ questions and size of the target segmentation and further links size to ease of discovery. This point is discussed in more detail in the discussion in Section 11.3.

**IBSR MR Brain Data** Finally, to assess the generality of our method, we evaluate it on the medical IBSR MR brain dataset. It is composed of $18$ magnetic resonance brain scans labeled pixel-wise with $4$ different brain structures. We decomposed the scans into $2D$ slices to have a larger number of runs for our evaluation. (Our later extension to 3D volumes can be found in [33]) At `http://www.cma.mgh.harvard.edu/ibsr/` the data and their manual segmentations are provided by the Center for Morphometric Analysis at Massachusetts General Hospital. Typically, labeling brain structures is a very complex task and requires a high expertise. Again, Figure 11.4 contains the results of our method. Due to the difficulty of the task, the absolute performance is lower than on the two other datasets which also have had a richer visual content and might be more suited for our prior and color histogram model. However, still a significant improvement after $20$ questions can be achieved.

**Comparison with geodesic object proposals** Finally, we compare our MCMC-based sampling of segmentations described in Sec. 11.1.2 with – at the time – recent geodesic object proposals (GOP) introduced by Krähenbühl and Koltun [93], whose goal is to suggest possible object regions in an image. We use the available (pre-trained) code for GOP, and for each image sample both, $N$ object proposals with GOP and $N$ segmentations with our sampling method. $N$ is automatically inferred by the GOP algorithm which we use then as the target during

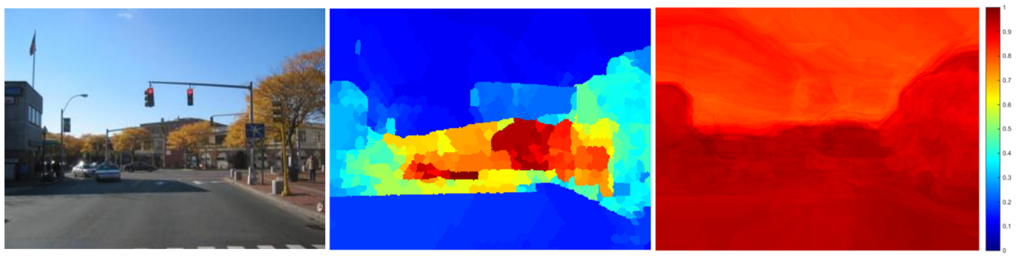


Figure 11.6. **Comparison between our MCMC-based proposals and geodesic object proposals.** The heat maps correspond to the frequency of occurrence of each pixel in the set of sampled segmentations, for the geodesic object proposal method [93] (middle) and our MCMC sampling scheme (right). While the former focuses more specifically on some areas, ours explores more the segmentation space.

| ABO score | Stanford | Brain |
|---|---|---|
| ours | 0.596 | **0.449** |
| GOP [93] | **0.660** | 0.431 |

Table 11.1. Comparing ABO scores (as in [93] sec. 6). We used the available MATLAB code for the Geodesic Object Proposals (GOP) and generated an equal amount of segmentations by sampling from our Markov chain.

sampling in our method. We report the Jaccard index of the sample matching the ground truth best (ABO score [93]) in Table 11.1. GOP have an advantage on the Stanford Background Dataset, as their seed placement technique is learned precisely on outdoor scenes. Since our method does not rely on any offline training step, it does not make assumption on the image content. This becomes apparent on brain images which illustrates the flexibility of our framework. In Figure 11.6 we demonstrate another interesting difference between the proposed framework and GOP. The figure shows a heatmap of regions that a generated by each approach. As GOP is learned to predict objects its segmentations are more redundant than ours, which is a good property to propose the likeliest object but suits less the context of an interactive guessing scenario where the user might not want to segment a specific object. Additionally, segmentations from GOP are constrained to a grid of super-pixels and thus cannot produce any possible segmentation from $\mathcal{S}$. Our method however runs directly at pixel level and therefore can generate any possible segmentation.

**Inaccurate Answers** In the theoretical derivation, we assumed the answers of the oracle to be always correct. However, sometimes a user can be confused, especially when a question is posed directly at an object boundary. To explore the robustness to inaccurate answers, we let the oracle answer randomly (50% yes – 50% no) when a question is posed within 3 pixels of the border of the desired object. Over the whole dataset this happens about 20% of the time, mainly on the later questions when the rough outline is known and the questions tend to refine object boundaries. On average, $1.2$ additional questions are needed to achieve the same Dice score as with perfect answers. Hence, the proposed method is reasonably robust to wrong answers but needs more questions. This is intuitively clear, as with uncertainty more questions are needed to transfer the same amount of information from the oracle to the algorithm.

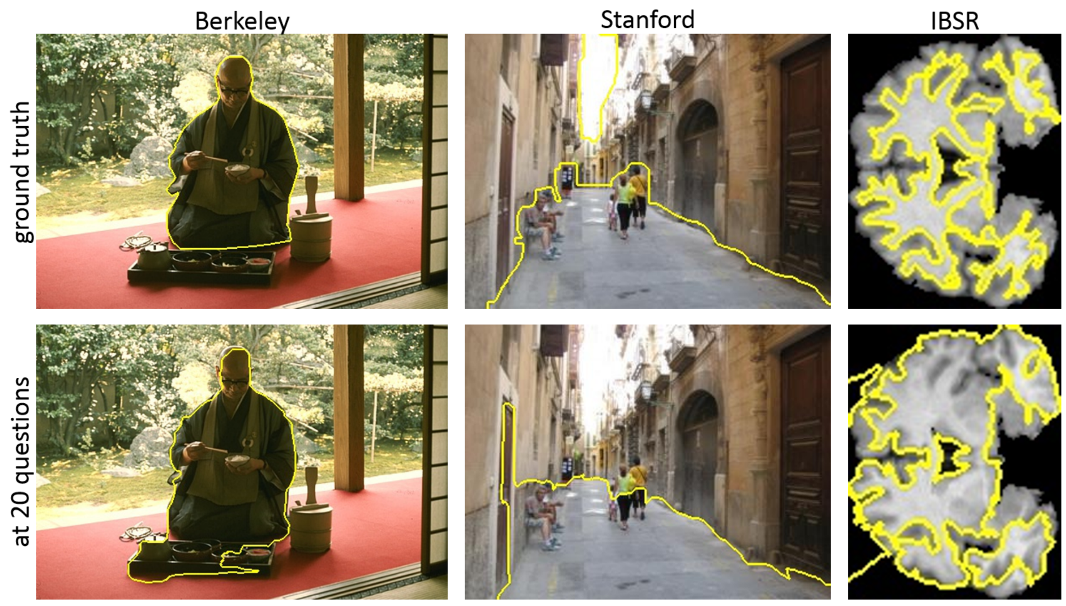


**Figure 11.7.** **Qualitative results.** Examples of oracle segmentation and the corresponding segmentation guess after 20 questions. From left to right: Berkeley Segmentation Dataset, Stanford Background Dataset, IBSR Brain Dataset.

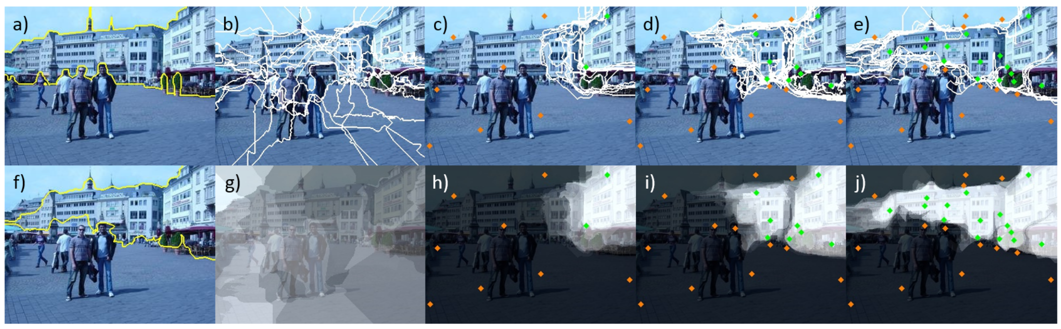


**Figure 11.8.** **Evolution of the segmentation belief with the number of answers collected.** a) shows the original image and the segmentation chosen by the oracle. Below, f) shows the final segmentation proposed by the computer after 30 questions. The images on the right show the evolution of the segmentation belief with the number of seeds collected (b: 0, c: 10, d: 20, e: 30). Both the intersection of all sampled segmentations (*i.e.* the over-segmentation described in 11.1.3 and shown in Fig. 11.2) and a heat map of the pixel likelihood are shown, respectively on the top and bottom row. The heat maps are generated by computing, for each pixel, the proportion of sampled segmentations that contained it (similarly to Eq. 11.5). The positive (resp. negative) seeds collected from the answers are shown in green (resp. orange). Note how the questions are progressively asked towards the left where the uncertainty is the highest, so that the whole series of buildings can be eventually segmented.

| Dataset | Method | 10Q | 20Q | 30Q |
|---|---|---|---|---|
| **Berkeley** | Random | 30.9 / 19.0 | 41.7 / 43.7 | 48.9 / 56.2 |
| | Uncertainty | 32.5 / 17.5 | 46.9 / 55.0 | 55.9 / 70.3 |
| | Ours | **34.7** / **23.8** | **48.8** / **62.0** | **56.1** / **73.2** |
| **Stanford** | Random | 49.8 / 56.5 | 60.2 / 69.8 | 65.6 / 74.7 |
| | Uncertainty | 50.6 / 57.6 | 61.9 / 71.4 | **67.9** / 77.7 |
| | Ours | **52.6** / **63.9** | **63.9** / **75.8** | **67.9** / **79.8** |
| **IBSR** | Random | 43.1 / 47.8 | 52.0 / 57.6 | 58.1 / 63.6 |
| | Uncertainty | 41.7 / 45.8 | 46.2 / 50.7 | 49.2 / 53.3 |
| | Ours | **51.5** / **53.7** | **58.4** / **60.1** | **62.9** / **64.7** |

Table 11.2. **Quantitative Results.** Mean and median Dice score after 10, 20 and 30 questions on the three datasets for our method and the two baselines described in Sec. 11.2.2

## 11.3 Discussion

Figure 11.8 shows a typical run of the proposed game. The oracle thinks of a desired, hidden segmentation – in this case the row of buildings in the background. It is a challenging target for a (non CNN) segmentation algorithm due to the visual complexity of its structure coming fact that it corresponds to a high-level semantic. Our method generates a first set of possible segmentations as seen in b) and g). Since no seeds are available yet, uniform regions like sky or pavement are favored at this stage and are sampled more often than others as seen in g). As questions are posed and answers from the oracle are progressively collected, we see that the space of sampled segmentations adapts to the target. Towards the end, the algorithm questions the oracle at boundary locations expanding step by step the initial guess towards the left. A good estimate of the oracle segmentation after 30 questions is seen in f).

Overall, our results suggest that our method has more difficulties to segment small objects. In particular, finding a first foreground seed inside a small object is very challenging. It is intuitive that it takes longer to find small areas than big ones – even with today's CNNs small objects are much more difficult than large ones. Additionally, the nature of the questions we ask is not optimal for finding small regions.

With this we conclude the chapter on the twenty question game but will later come back to it, equipped with the power of deep learning, to understand the improvements we can make by training a CNN for segmentation on a large dataset to learn a prior.

# Guiding a Network 12

In the previous section we have seen how human interaction can be used to segment an image. Since the original conception of that twenty questions idea in 2014, a lot has changed in the computer vision community. We will now explore how we can integrate human interaction to neural networks that dominate almost all vision benchmarks. Due to the amount of data that is used to train a deep network, we adapt the idea of simulating a human by using the ground truth. That allows us to train fast and without additional annotation effort. The work from this section has been published in [155].

## 12.1 Interacting With A Deep Network

Convolutional neural networks (CNNs) play a prominent role in a large amount of computer vision tasks, such as image classification, semantic segmentation, object detection, depth prediction and human pose estimation. The vast majority of the techniques proposed for these applications train specific network architectures once, then deploy them as static components inside an algorithm. However, it is unlikely that any static network will be perfect at the task it was designed for. Nevertheless, real world applications could often benefit if the predictions of the CNN were to adapt to feedback or specifications provided by a human user, this holding the potential to improve the network's performance.

We can train a CNN to segment an image and deploy it in a photo editing software. However, it is clear that sometimes the network will fail to produce a perfect segmentation, as it is almost impossible to anticipate every possible scenario that the application will be used in.

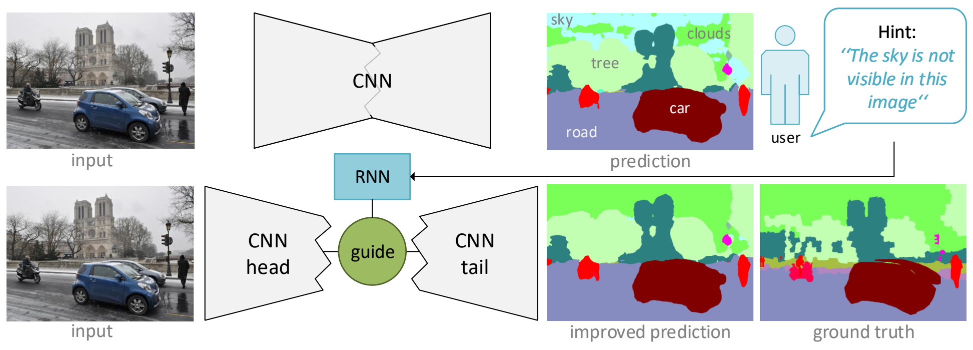


Figure 12.1. **Overview.** We introduce a system that is able to refine predictions of a CNN by injecting a guiding block into the network. The guiding can be performed using natural language through an RNN to process the text. In this example, the original network had difficulties to differentiate between the `sky` and the `cloud` classes. The user hints that there is no sky and the prediction is updated. The CNN remains fixed and does not need additional training.

The user would need to manually repair the segmentation output or the developer could add a post-processing algorithm based on some heuristics. A simpler and more effective way would be to allow the user to interact directly with the network, by for example, pointing out that "*the child on the right of the image is in the background, not in the foreground*". The user who was previously presented with a fixed, black-box prediction would now be able to influence and alter the outcome according to their needs. This allows a network and the user to collaborate by using language to tackle a problem. In high-risk domains such as computer-assisted diagnosis and medical image analysis, where fully automated segmentation is not always robust, the possibility of interaction is particularly useful. It is also useful to ease and speed up labor-intensive and repetitive labeling tasks, such as those needed to create datasets for semantic segmentation.

We propose a novel approach for network interaction focused on improving the performance of a pre-trained CNN at test time. A guiding mechanism translates the user's feedback into changes in the internal activations of the network, thus modulating the inference process. Interaction can take different forms, and common in literature have been scribbles, click-based or contour-based inputs, or conditioning on prior information. Here, we focus on a language-based approach, as it is the most natural way of interacting. However, our guiding system is not specific to language input and can be used with any kind of hint or modality. Language based guiding enables interaction with a trained model in the form of a dialog. The user receives the initial (black-box) prediction and then has the option to input a text query that updates the network's prediction.

We showcase this interactive module on the task of semantic image segmentation. Our approach does not depend on any explicit annotation for text-region correspondences, but results indicate that the module can successfully transfer semantic information from the natural language domain to the visual domain, so that the network eventually produces more accurate segmentations. Even without explicit annotation for text-region correspondences, our experiments show that the guiding module can successfully transfer semantic information from the natural language domain to the visual domain, resulting in a more accurate segmentation. This can also be seen as a translation task, where the guide learns to translate human hints to network activations of the fixed network.

The intersection of vision ad language is currently heavily explored. However, most of the related literature outputs text, *i.e.* either an answer or an image caption. Our system produces updated segmentation masks in our experiments, in general the output of the interactive CNN lies in the same domain as the initial one.

**INTUITION: THE GUIDE**

We are now thinking of the guide a universal module to expose trainable *control-knobs* in an network. This makes the idea of guiding very powerful, as many architectures could benefit from additional data to improve predictions. One can also think of two different modalities or tasks guiding each other. This could potentially mitigate the typical problem of a multi-task setting where one task loses performance due to balancing of the loss terms.

## 12.2 Guiding a CNN

He, we detail how our interaction module – the *guide* – is injected into a *fixed* CNN. Figure 12.1 shows a high level overview of the system.

We will first specify a nomenclature to be able to precisely refer to the different components of the system. We insert a module into the CNN, which we call the *guide*. This guide uses a *guiding block* to interact with the *guided CNN*. The guiding block adjusts the activation map of the CNN to improve the prediction for the current input. The injection of the guide splits the guided CNN into two parts: the *head*, that processes the input upto the guiding block, and the *tail*, from guiding block to the final prediction. The guide uses a *hint* – typically from a human user – to modify the guided network.

The decomposition into a head function $h$ and a tail function $t$ can be used to specify the output prediction $\tilde{y}$ from input $x$ as

$$t(h(x)) = \tilde{y} \; . \tag{12.1}$$

Where the original network is split is chosen manually. A reasonable choice is featuremap with the lowest dimensional encoding, as this layer contains high-level information in a compact representation, ideal for manipulation by the guide.

**Hint: Guiding Location**

The intuition is that guiding late towards the output of the network produces more local changes, while early guiding is able to make bigger changes. It easily possible to guide the network in multiple layers, in the following however, we will assume one guiding location not to clutter the notation.

We validate different choices of splits in Section 12.5.

## 12.3 Guiding Block

The integral piece of this approach is the guiding block which enables the guide to modify activations inside the guided network. Element-wise control is volatile to over-fitting, since activation maps usually consist of a large number of elements (*e.g.* $32 \times 32 \times 1024 \approx 1$ million). Our guiding block avoids learning a full control tensor by making use of the following intuition. A CNN typically encodes specific features in each channel/neuron of a given layer. Thus, strengthening or dampening the activations per channel, can be enough to emphasize or suppress aspects of the image representation and thus adapt the prediction in a semantically meaningful way.

A feature representation $h(x) = A \in \mathbb{R}^{H \times W \times C}$ with width $W$, height $H$ and number of channels $C$ is predicted by the head of the network. To adjust that representation per channel, we can multiply it with a guiding vector $\gamma^{(s)} \in \mathbb{R}^C$ and a bias $\gamma^{(b)} \in \mathbb{R}^C$,

$$A'_c = (1 + \gamma_c^{(s)}) A_c + \gamma_c^{(b)} \ , \tag{12.2}$$

where $c \in [1, \ldots, C]$ selects channels of the feature tensor $A$ and the corresponding elements of the guiding vector $\gamma = (\gamma^{(s)}, \gamma^{(b)})$.

**Hint: Connection to ResNets**

When (12.2) is expanded

$$A'_c = A_c + (\gamma_c^{(s)} A_c + \gamma_c^{(b)} \ ), \tag{12.3}$$

one can interpret it as the guide predicting a residual update (similar to ResNets [59]).

The formulation is structured such that for $\gamma = 0$, our guiding block reproduces the input feature map or in other words, does not guide the network. The influence factor $\gamma$ plays the role of a filter on the feature maps. For example, if for a specific channel $c$ the influence would be $\gamma_c^s = -1$, all information in that channel would be suppressed and set to 0. Conversely, for $\gamma_c^s > 0$, the activation strength of that feature channel is increased.

**Intuition: Linear Guide**

The fact that a linear change in activations works well can be linked to sever other observations of neural networks. Batch normalization [71] is linear in the activations and helps to train very deep architectures. Another interesting link are cross stich networks [131]. This idea of adapting the activations has also been used in the conditioning layer in [139].

Per-channel updates and feature re-weighting via $\gamma$ modifies the whole feature map with the same weight. To be able to handle hints with location information, a spatial component needs to be introduced, as it is impossible for this module to encourage spatially localized changes in each feature map. To overcome this limitation, we extend the same idea to the spatial dimensions of the feature map and thus guide each direction (width, height, and channels) with a linear component. Two additional guiding vectors $\alpha \in \mathbb{R}^H$ and $\beta \in \mathbb{R}^W$ will be used to modify the feature map $A$ with spatial attention.

$$A'_{h,w,c} = (1 + \alpha_h + \beta_w + \gamma_c^{(s)}) A_{h,w,c} + \gamma_c^b \ , \tag{12.4}$$

where we will index $A$ with $h$, $w$ and $c$ to uniquely identify a single element $A_{h,w,c} \in \mathbb{R}$ of $A$.

Putting it all together, the guided network now computes the following function.

$$y^* = t\left[(1 \oplus \alpha \oplus \beta \oplus \gamma^{(s)}) \odot h(x) \oplus \gamma^{(b)}\right] , \tag{12.5}$$

where the vectors $\alpha$, $\beta$, $\gamma$ are tiled along their appropriate dimensions with $\oplus$ and the Hadamard product is denoted with $\odot$. Spatial influence is controlled by $\alpha$ and $\beta$, while and $\gamma$ controls semantic meaning. Although we multiplying the components $\alpha$, $\beta$ and $\gamma$ is possible we found its gradients numerically unstable.

Using $\alpha$, $\beta$ and $\gamma$ instead of element-wise control, reduces the number of parameters from $H \times W \times C$ to $H + W + C$. Which is $32 + 32 + 1024 = 1088$ – much less parameters than the previous example and thus less prone to over-fitting

**Hint: Fully Convolutional Architectures**

Since fully convolutional architectures are a common choice for image prediction tasks such as semantic segmentation, we linearly interpolation of $\alpha$ and $\beta$ when the feature map spatial resolution varies. This makes $\alpha$ and $\beta$ independent of $H$ and $W$ and allows us to select the granularity of the spatial resolution by changing the dimensionality of $\alpha$ and $\beta$ to match the spatial complexity of the hints the guide follows.

We now show the guide in practice for online interaction with neural networks via user feedback.

## 12.4 Learning to Guide with Text

In the example of user interaction for semantic segmentation hints can be provided in the form of clicks that carry information about the class of the chosen pixels. However, here we explore a more natural way of human-machine interaction. The user is allowed to type hints in natural language to the network. The guiding mechanism will be trained to update its parameters based on the user's hint. To the best of our knowledge, this idea of interacting with a pre-trained network is novel and has not been studied before.

### 12.4.1 Processing Queries

As usual for processing text with a neural network, we use a recurrent neural network (RNN). The raw text is encoded using a pre-trained word embedding matrix [138], to acquire a fixed-length embedding vector per word. Gated Recurrent Unit (GRU) [24, 26] is then fed with the embedded word sequence.

**Hint: Word Embedding**

The word embedding is frozen and not further trained together with our guiding module. This gives us the hope that the model can later deal with words outside the training set, if they have similar embeddings as a known word. In some further

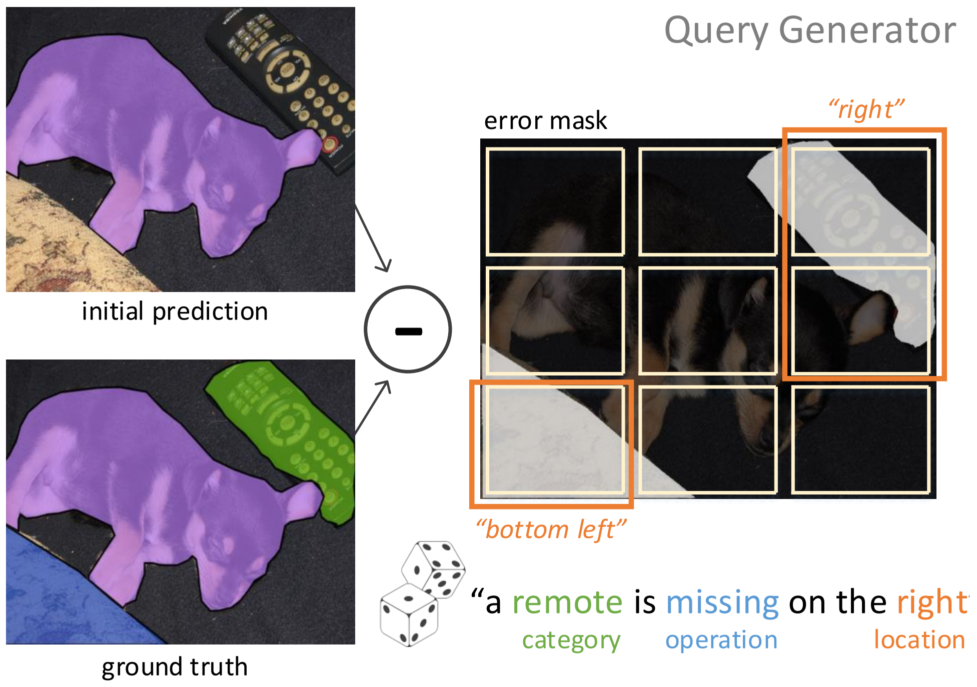


Figure 12.2. **Query Generator**. We illustrate the process to automatically generate queries to substitute the user during training.

experiments, where we used a similar architecture to generate text, it was crucial to actually train the embedding together with the GRU. Otherwise the network was not learning much.

The GRU predicts the guiding parameters $\alpha$, $\beta$ and $\gamma$ from its final state using a simple linear transform. During training the following steps are repeated for every sample.

1. Generate an initial prediction with the fixed, unguided CNN.

2. Prediction and ground truth are fed into our hint generator (next paragraph) that creates a query (e.g. "*the sky is not visible in this image*").

3. The hint is tokenized and embedded into the GLoVe vector space.

4. The GRU reads the hint and predicts $\alpha, \beta$ and $\gamma$.

5. Using (12.5) we update the activations of the guided feature map which changes the original prediction.

6. The standard loss for the given task (pixel-wise cross entropy loss for semantic segmentation) is re-weighted giving positive weight (1) to the class(es) mentioned in the query and negative weight (-1) to all other pixels. This encourages changes that coincide with the query.

The above steps are repeated batch-wise as usual during training. However, we only update the parameters of the GRU (and the linear mapping to the guiding parameters), the CNN remains unchanged.

### 12.4.2 Generating Queries

We again simulate the human using the ground truth segmentation annotation. This concept has two benefits. Naturally, expensive annotation of human interaction can be avoided. Additionally, dynamic interactions can also be modeled whereas this would require two additional labeling sessions with humans. We automatically generate textual expressions from visual categories. We use standard segmentation datasets and do not require visual/textual annotations, such as captions [114], referring expressions [70, 81] and region-description correspondences [124]. As our method is built to correct prediction errors on the fly, we synthesize dynamic natural language queries by comparing the initial prediction and the ground truth segmentation.

To automatically generate queries we use a combination of functionality, semantic categories and spatial layouts (see Figure 12.2). The semantic categories are defined by the set of classes in the prediction and ground truth. They encompass operations such as discovering missing classes, suppressing noisy outliers or replacing wrongly predicted pixels with another class. In our experiments we use two complementary operations: `find` to hint classes missing in the prediction and `remove` to specify wrongly predicted labels.

Queries are built by their function and two placeholders. Usually for each prediction several combinations of functions and placeholders are possible. Whenever there are multiple options, we select the entries randomly at each training step from the set of plausible combinations. For the spatial component, we divide prediction and groundtruth into a $N \times N$ grid. We search for all the erroneous classes, either missing or mistaken in each grid cell. Tiny regions comprised of only several pixels are thereby ignored.

Gathering all possible classes, we randomly pick one and use its semantic name for the textual expression (*e.g.* "*find the person*"). Using the cells we then track its position and define different spatial attention areas which are then converted into text phrases such as "*on the top left*", "*on the bottom*", "*in the middle*". With this method we can generate textual phrases automatically and dynamically.

**INTUITION: QUERY GENERATOR**

The query generator is the most fragile part of the proposed method. During the conception of this method, most time was spent fine-tuning its parts so that it generates reasonable and useful hints. There are many small corner cases where a naive generator would produce a hint that a human would never utter. Because of this we are currently working on a learned hint generator, which however comes with its own downsides such as drift in semantic meaning of the language when all parts are learned.

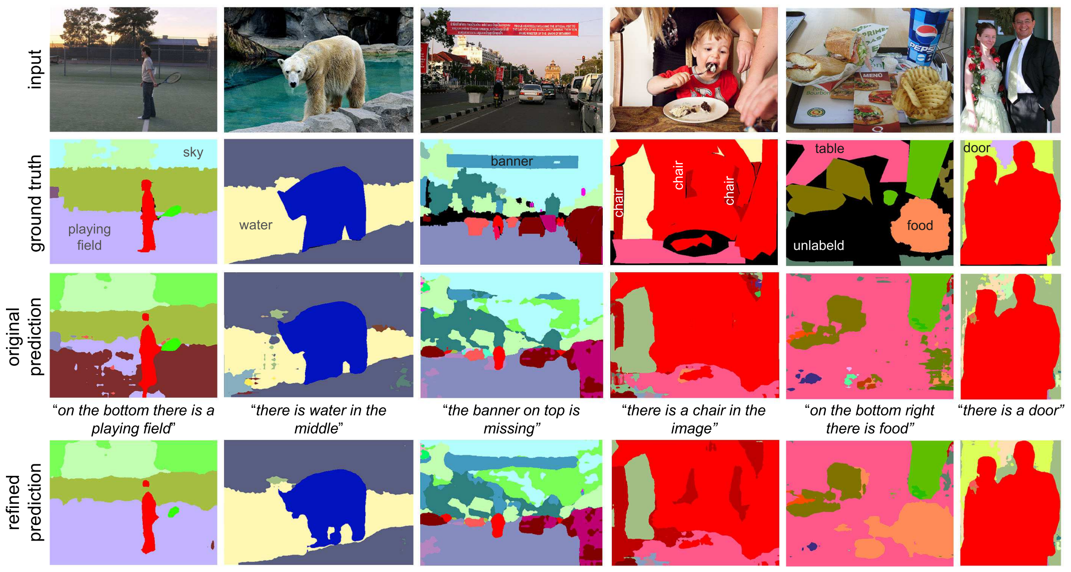

input
ground truth
original prediction
refined prediction
sky
playing field
water
banner
chair
chair
chair
table
unlabeld
food
door
“on the bottom there is a playing field”
“there is water in the middle”
“the banner on top is missing”
“there is a chair in the image”
“on the bottom right there is food”
“there is a door”

Figure 12.3. **Qualitative Results**. We show qualitative results using `find` hints for missing classes. In the first example, we resolve a confusion between `ground` and `playing field`. In the second example, we show that the often occurring spurious predictions can also be handled. The third column shows that the network get the hint to find the `banner`, although it bleeds slightly into the `building` below. In the fourth and the last column, classes that are heavily occluded can be discovered too after guiding. The black ground truth label stands for `unlabeled` thus any prediction is allowed there.

## 12.5 Experiments

The performance of state-of-the-art CNNs on the typical (dated) benchmarks such as PascalVOC is close to perfect, only suffering from small errors such as boundary imperfections or tiny objects. Bringing a human into the loop to request improvements is not meaningful for almost perfect initial performance. We thus evaluate our guiding idea under a much more challenging setting, where the performance of a state-of-the-art model is far from perfect and interaction with a user is actually desired.

COCO-Stuff [17] is a subset of 10k images of the 2014 MS Common Objects in Context (COCO) dataset [114], further split into 9k training and 1k testing images. The images are annotated per pixel-level with 91 “things” and 91 “stuff” classes. The dataset is limited in number of images but rich in categorical context, rendering it very challenging for current data hungry methods.

### 12.5.1 Implementation Details

To keep a fixed pre-trained segmentation CNN we split the training set into two halves and train a DeepLab model [21] with a ResNet-101 [59] as back-end on the first half which scores only 30.5% mIoU (which is comparable to the state of the are even when training on only half of the data). The weights of this CNN are now fixed for all further experiments. Now only the guiding mechanism is trained on the remaining and unseen 4,500 images. As explained above, the guide is trained to understand text queries via a pre-trained, 50-dimensional GloVe word

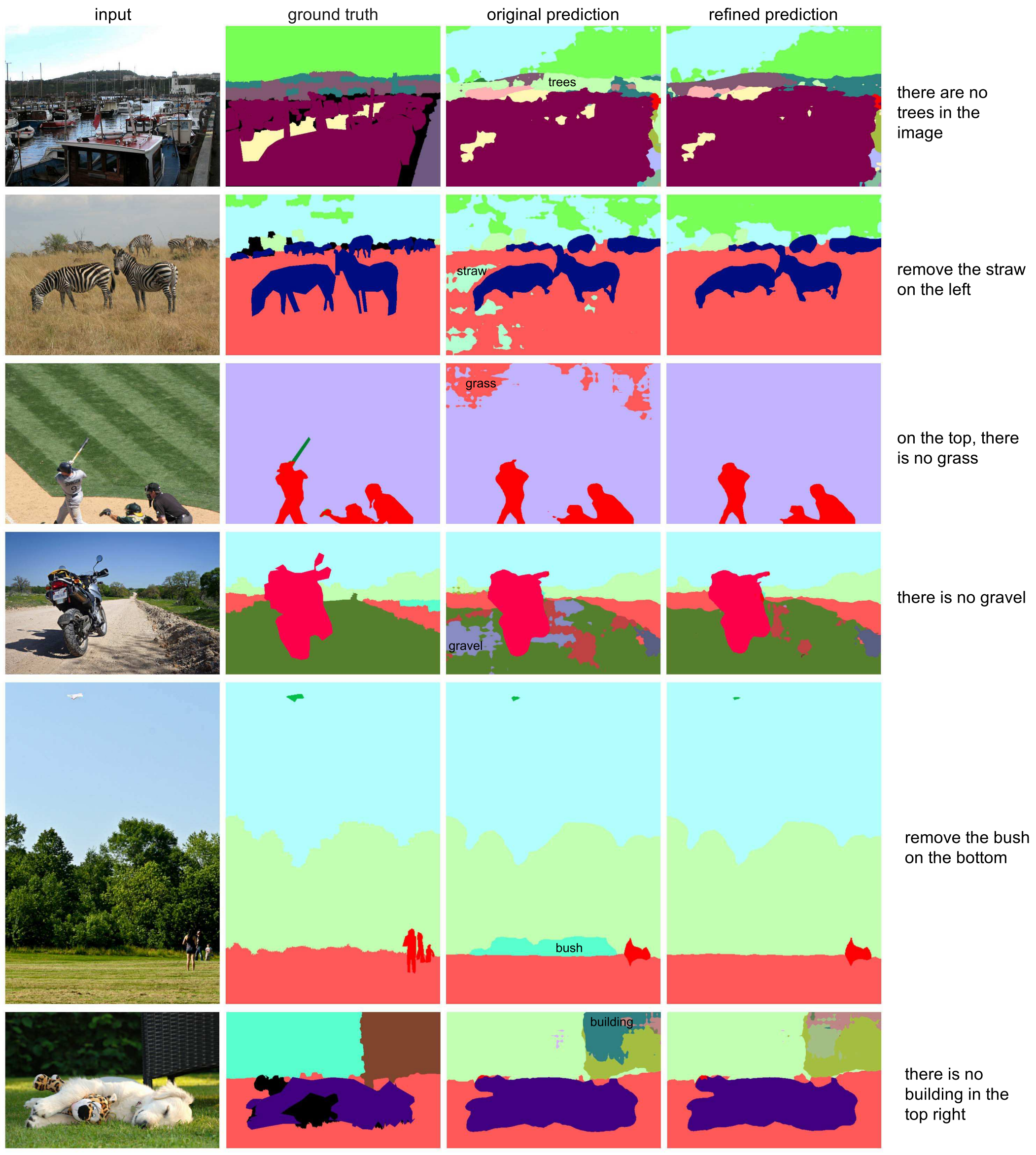

input
ground truth
original prediction
refined prediction
trees
there are no trees in the image
straw
remove the straw on the left
grass
on the top, there is no grass
gravel
there is no gravel
bush
remove the bush on the bottom
building
there is no building in the top right

**Figure 12.4.** **Qualitative Results**. We show qualitative results using `remove` hints for correcting noisy or wrongly predicted classes. The last row shows a failure case, where the surrounding pixels of the queried class are also wrong, therefore in the updated prediction the noise cannot be replaced by the correct class.

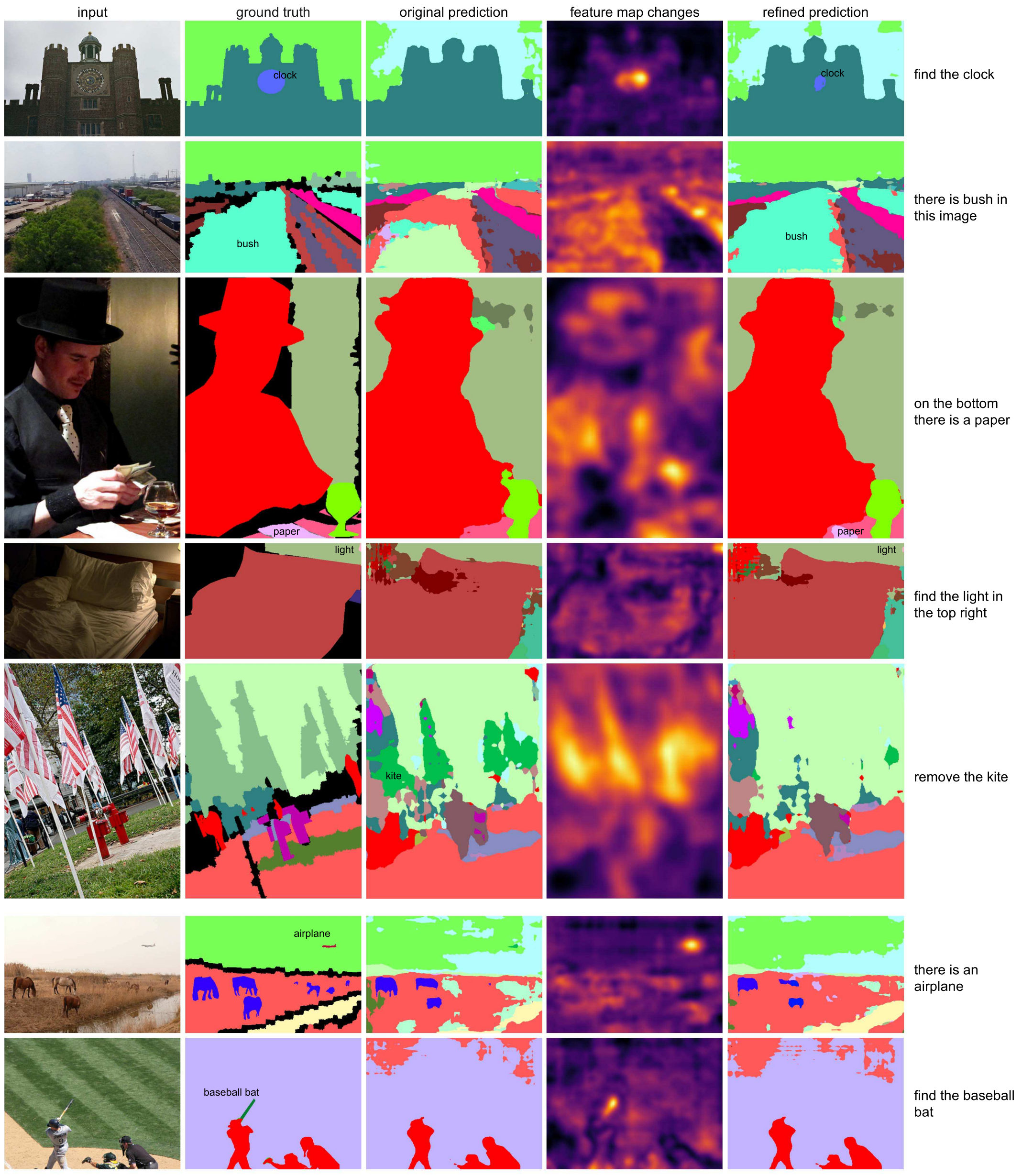


Figure 12.5. **Visualizing the guiding vectors**. We show qualitative results and visualize the heatmaps that the corresponding guiding vectors produce. We observe that the guide has learned to attend to relevant regions, despite not seeing the image context. The last two rows show failure cases, in which the guide's influence is still not enough to alter the output.

embedding matrix and an RNN to produce relevant guiding parameters. The GRU contains 1024 hidden units. $\alpha$, $\beta$ and $\gamma$ are predicted with a linear layer from the last hidden state of the GRU.

There are two ways of injecting the guide. The first one alters the CNN's activations directly, therefore the guiding vector size depends on the CNN layer that is being guided. The second wraps the predicted weights inside a trained, additional residual block with 256 channels. The input dimensions are $320 \times 320 \times 3$. To further vary the queries, we randomly select from a set of variations with similar meaning such as "*the . . . is missing*", or "*there is a . . . in the image*". The grid size $N$ for automatic query generation is set to 3, resulting in 9 cells belonging to the locations (and their combinations) *top, bottom, left, right, middle*. To mitigate this randomness during evaluation, we report averages over five runs per image.

To summarize the upcoming results, our best guided model improves the overall score from 30.5% to 34.4% with a single hint and to 35.6% with two consecutive hints. As detailed above, training the original CNN on the full train set is only marginally better than on the half, reaching 30.8% mIoU instead of 30.5%. We show qualitative CNN predictions before and after guiding in Figure 12.3. In our experiments we observe that our method resolves typical problems with the initial predictions, such as confusions between classes (columns 1, 2), partially missing objects (column 3, 4) and only partially visible objects in the background (column 6). In Figure 12.4 we show how guiding helps to remove noisy classes from the model's prediction. In Figure 12.5 we also visualize the attention maps that the guiding vectors produce conditioned on the user's hint. We observe that the guide focuses on relevant regions of the input image, despite not seeing the image context as input.

Since the conditioning layer of [139] is closely related (although for a different task) we first compare our guiding block in the following section. Then, we evaluate the influence of the guiding location, by inserting the guide at different layers of the CNN. Hint complexity is evaluated by successively increasing the diversity of the queries. Finally, we show that repeated guiding further improves the result without the need for additional training. To understand failure cases and the inner mechanism of the guiding block we visualize heat maps and embeddings of the guiding vectors in the last part of this evaluation.

## 12.5.2 Guiding Block Evaluation

In this section, we investigate four variants of the guiding block. The raw segmentation performance is shown in Table 12.1. [139] wraps a res-block around their FiLM layer, thus we compare to the FiLM layer with and without res-block. The main difference (apart from [139] training the whole network) is that our guiding block also contains the guiding components $\alpha$ and $\beta$, that translate spatial information from the text to localized attention in the feature space.

We find that our spatial attention helps performance, while an additional res-block does not change the score. This indicates that the guiding layer alone has enough capacity to change the activations and does not need a residual block to perform well.

| guiding module | mIoU w/ res-blk | mIoU w/o res-blk |
|---|---|---|
| FiLM [139] | 31.3 | 31.3 |
| ours | **32.2** | **32.3** |

Table 12.1. **Guiding Block Variants.** We evaluate mIoU performance when guiding res4a using `find` queries, in comparison to the conditioning layer of [139].

| | res3a | res4a | res5a | res5c |
|---|---|---|---|---|
| mIoU | 31.2 | 32.3 | **34.3** | 34.2 |

Table 12.2. **Location of the guiding block.** We evaluate mIoU performance when guiding different layers inside the CNN using `find` queries. Please note that there are 22 res-blocks between res4a and res5a which explains the large performance difference.

### 12.5.3 Guiding Location

As stated before, the guiding block is very flexible and can be injected into the network at any location. Our intuition is that a location that is very late in the decoder - close to the prediction - often results in small, local changes of the output. Earlier guiding results in more global updates that affect a bigger region of the prediction and occasionally multiple classes. This corresponds to the understanding that higher level concepts are learned deeper in the encoder. When the guide is placed too early in the encoder the feature maps are too simple (edges and object parts) to guide appropriately. Quantitative results supporting this observation are shown in Table 12.2.

### 12.5.4 Complexity of Hints

We analyze how the complexity of hints affects the performance improvements achieved by the guide. Here, we differentiate between two hints with different functionality: `find` and `remove`. A `find` hint indicates that the network had missed a class: "*There is a person in the top right*". We use `remove` when the network had predicted a class that is not in the ground truth or in an incorrect location.

Table 12.3 summarizes the performance for the different hint types. The first observation is that `remove` generally yields a lower performance gain than `find`. Inherently, `remove` is more ambiguous than `find`. When we tell the network to remove a class, it still does not know what should be in its place instead. However, find directly delivers the right class to the network. Naturally, training with both queries (`find` or `remove`) simultaneously, achieves average performance between the two types as the query is selected randomly each time. This also indicates that the guide has enough capacity to handle both, `find` and `remove`, simultaneously.

| hint complexity | guiding loc. res4a (res) | res4a | res5a |
|---|---|---|---|
| `remove` | 30.6 | 30.9 | 30.9 |
| `find or rmv` | 30.8 | 31.0 | 33.2 |
| `find` | 32.2 | 32.3 | **34.3** |

Table 12.3. **Complexity of Hints.** We show performance of the method using two different types of hints.

| # hints | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| mIoU | 30.5 | 34.4 | **35.6** | 33.9 | 32.0 |

Table 12.4. **Guiding multiple times.** We guide iteratively with multiple hints. After three hints performance decreases due to the guide over-amplifying certain features, causing noise in the predictions.

### 12.5.5 Guiding Multiple Times

Since the guide is trained to modify an activation map in such a way that the tail of the CNN can still use it, it is possible to guide this activation map serially, with a second or third hint without additional training. The key insight is that the guiding block will still result in a valid feature map that can then be guided again. The results for this experiment can be seen in Table 12.4 and Figure 12.6. We observe a performance increase until two hints. With three or more the guide seems to over-amplify some channels, which causes noise in the predictions which reduces the score. Nonetheless, even with four hints we observe a good gain over the original network. We expect that training with multiple hints would make this procedure mode table although increase training time as several guidings would need to be done per sample.

## 12.6 Insights into the Model

In this section we will look into the guide and perform two experiments to shed light onto its inner workings.

### 12.6.1 Failure Cases

First we will analyze typical failure cases. If the initial prediction was particularly bad, the guide has difficulties to repair the mistakes because of two reasons. On one hand, fully wrong predictions indicate that also the guided feature map will be very noise and thus unable to be guided properly. On the other hand, our automatic query generation module will have problems selecting a useful hit as there are many possibilities, but each one only improving a small part of the prediction. In Figure 12.7 we showcase one of these scenarios. The hint

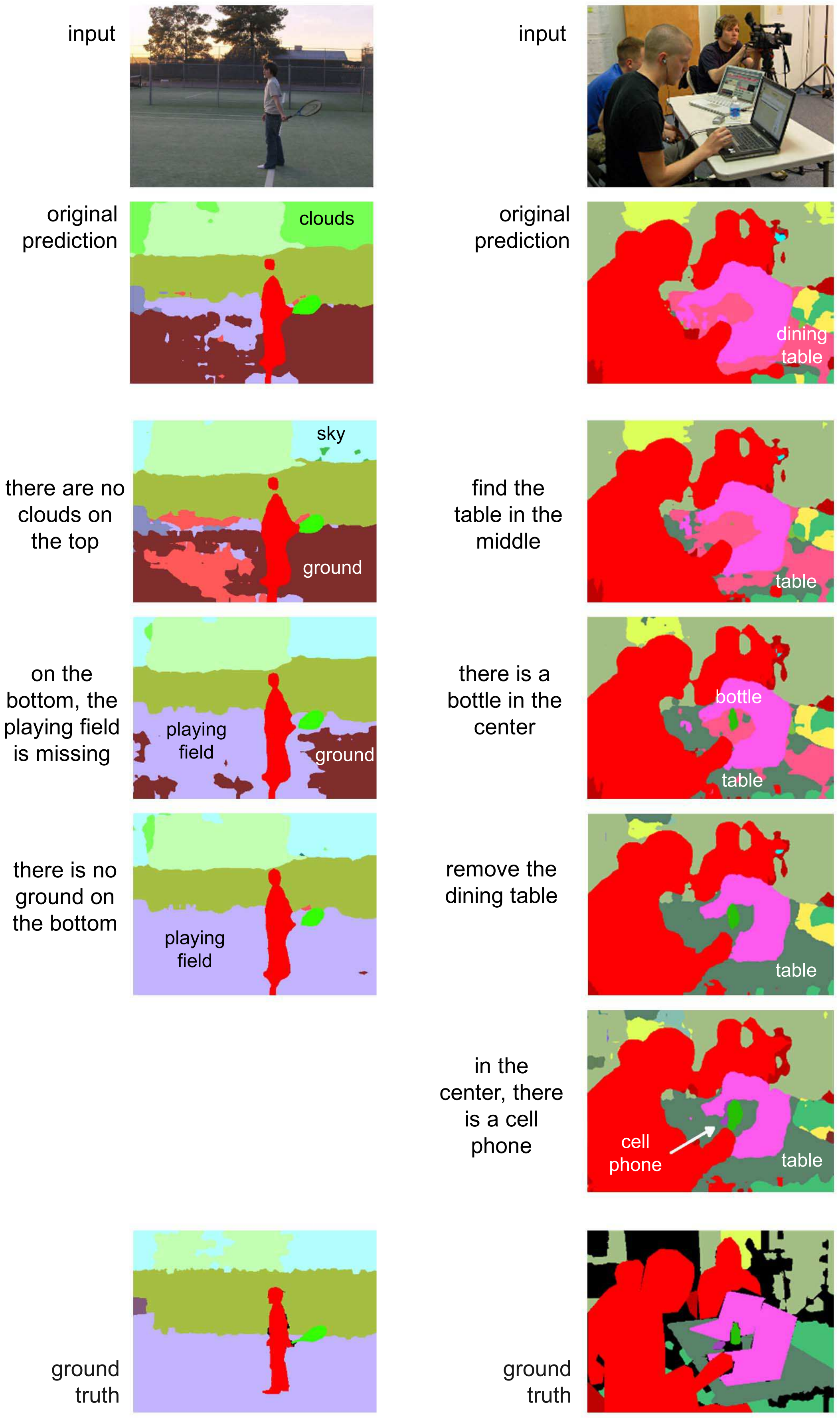

input
original prediction
clouds
there are no clouds on the top
sky
ground
on the bottom, the playing field is missing
playing field
ground
there is no ground on the bottom
playing field
ground truth
input
original prediction
dining table
find the table in the middle
table
there is a bottle in the center
bottle
table
remove the dining table
table
in the center, there is a cell phone
cell phone
table
ground truth

Figure 12.6. **Guiding Multiple Times.** In this experiment we guide an input image multiple times. We can show that this is possible even when training with only one hint.

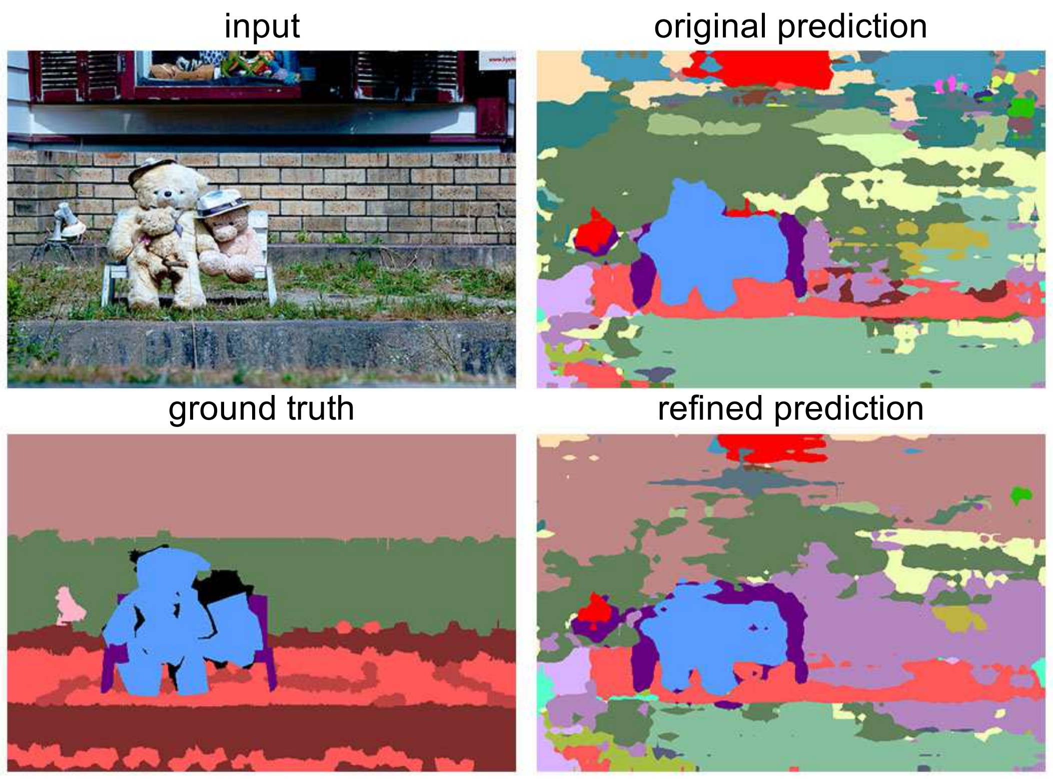


Figure 12.7. **Failure case.**. Hint: "*there is a building in the top*" When the initial prediction fails, our method has difficulties recovering the mistakes. The refinement includes the `building` only partially and it bleeds into `stone-wall` below.

indicates to the network that a `building` is missing on the top. The guide can recover some parts of it but not fully and most of the remaining prediction stays faulty. In these and some more sever cases we can assume that the relevant features necessary for a successful guiding, cannot be recovered in the guided activation map or are not present at all.

In a second experiment, aimed at understanding how the activation map is influenced by the guide, in Figure 12.8 we visualize a heatmap for different queries in the same image. From this visualization we can conclude that the guide understands the hint about the `sky` (a). However, when asking for the `surfboard` in (b), no change reaches the prediction. Without visualization, we would assume that the query was not understood correctly. However, our heatmap visualization indicates that the guide does indeed change the right regions of the image, but apparently not strong enough to overpower the `sea` label in the center.

We postulate that more precise queries could potentially mitigate this problem. "*There is a surfboard where you predicted sea*" would not only let the guide emphasize `surfboard` but simultaneously let it know to dampen the `sea` class.

### 12.6.2 Semantic Analysis of the Learned $\gamma$-vectors

Finally, we perform a last experiment to understand the mapping between text and guiding vectors. Using simple `find` queries, we predict a $\gamma$ vector for each semantic class. We then

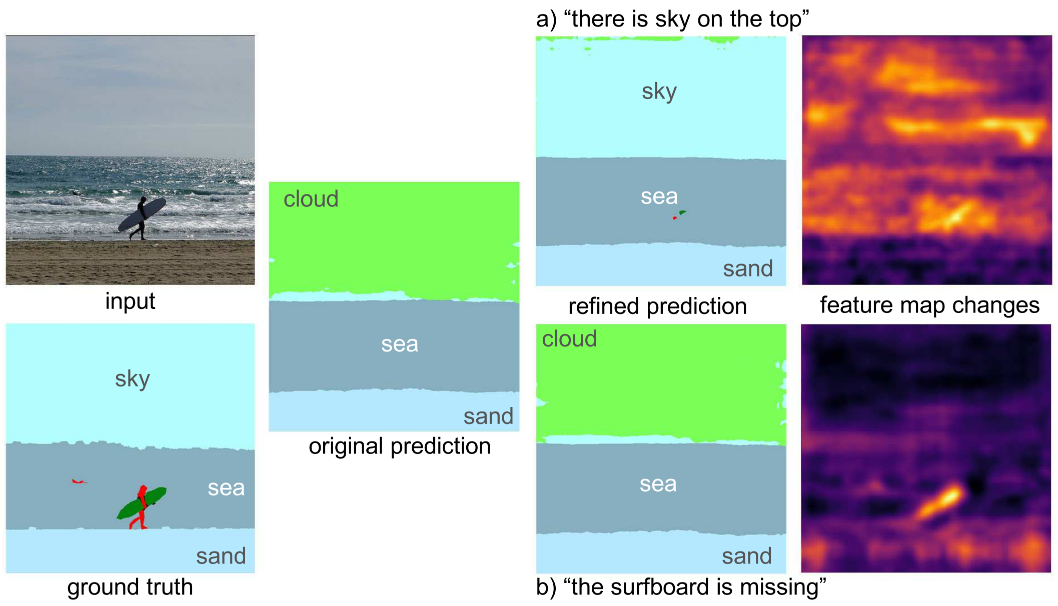


**Figure 12.8.** **Failure Case Visualization**. In the first example a) the refined prediction is correct. In b) the heatmap indicates that the guide activates the right feature maps but it is not enough to change the output.

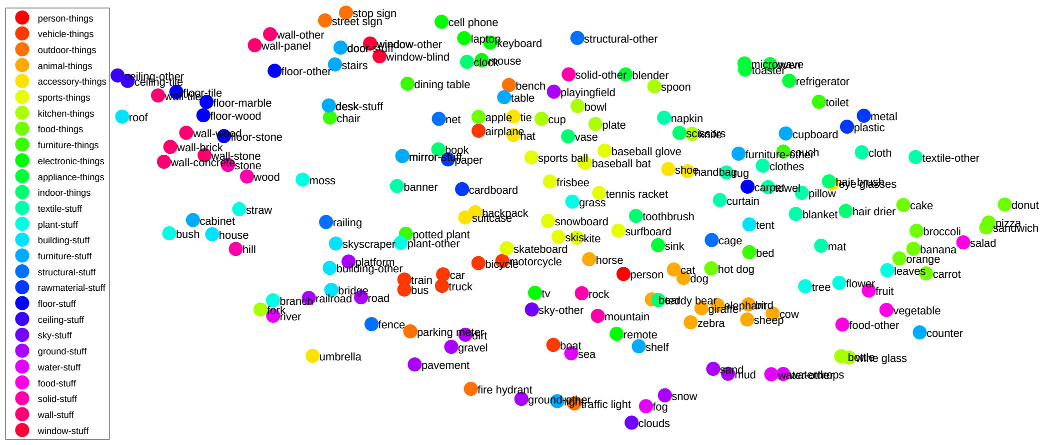


**Figure 12.9.** **Visualizing $\gamma$**. We visualize the learned $\gamma$ vector for every class using t-SNE. The colors correspond to the higher level categories which are present in the dataset but not used in training. Best viewed in digital version.

use a t-SNE projection of these 256 dimensional vectors together with higher-level semantic group labels. The color categories in Figure 12.9, that the 182 classes are grouped into, correspond to these higher level categories. This grouping was never used during training or testing of the method. The purpose of this experiment is to visualize the intersection of features learned from the CNN for segmentation and text representation learned by the GRU. We see semantically similar words cluster, indicating that the joint embedding successfully correlated text and image features. Visually similar classes such as `sports ball`, `basket ball` and `frisbee` are grouped.

# 13 Twenty Deep Questions

With this chapter we close the loop between our recent *guiding* idea for neural networks and the *twenty questions* game proposed earlier. The idea is to use the guiding block in a CNN and guide it with the answer to the questions. This can be accomplished without the need of learning a guide, by simply formulating it as an optimization problem.

## 13.1 Guiding by Backpropagation

We will again visit the task of semantic segmentation, now with a CNN. A guiding block can be equipped to any network, as it does not modify the feature maps when $\alpha = \beta = \gamma = 0$. In the following, we consider the case of a user giving feedback to the network and propose an effective method on how to enable fixed, pre-trained CNNs for interaction at *test* time.

We place the guiding block between head and tail, and initialize guiding parameters to $0$. For a given input $x$, we define an energy function to optimize $\alpha$, $\beta$ and $\gamma$. The questions we ask to the user are not binary in this case but will be of the from "*What class does this pixel have?*". The accumulated of hints will from a sparse input $\hat{y}$ associated to a mask $\hat{m}$, where $\hat{y}$ and $\hat{m}$ have the same dimensions as the prediction $\tilde{y}$ and the ground truth $y$. $\hat{m}$ is a binary mask that indicates the locations where a hint (*i.e.* prior knowledge) was given with one and is zero everywhere else.

Initially the segmentation CNN was trained with a loss $\mathcal{L}(t(h(x)), y)$ – typically softmax cross entropy. For guiding with the previous answers we now use the same objective, but focus on the given answers using the mask $\hat{m}$. Instead of updating the network's parameters, we minimize the objective for the guiding parameters

$$\alpha^*, \beta^*, \gamma^* = \operatorname*{argmin}_{\alpha,\beta,\gamma} \left[\hat{m} \odot \mathcal{L}(y^*, \hat{y})\right] \ . \tag{13.1}$$

Everything in this formulation is differentiable, thus we can minimize (13.1) using backpropagation. The minimization optimizes for the best parameters $\alpha^*, \beta^*, \gamma^*$ according to the hints and mask. In our experiments we use gradient descent with momentum to minimize (13.1). Momentum helps to find a suitable minimum after fewer iterations, but in this case does not affect performance.

As before, modulating the feature map with the guide, results in an overall adjusted prediction.

| #questions | 0 | 1 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| **FCN-8s** | | | | | | |
| mIoU | 62.6 | 65.3 | 73.1 | 76.9 | 77.3 | **81.0** |
| p.accuracy | 91.1 | 91.8 | 94.1 | 95.3 | 96.0 | **96.3** |

Table 13.1. **Performance after number of questions** We guide a pre-trained FCN-8s [117] on PascalVOC 2012 *val* set [36] directly, using back-propagation. We report the mean intersection over union (mIoU) score and pixel accuracy. Every interaction with the user improves the result.

**INTUITION: LOCAL MINIMA**

The tendency of gradient descent to fall into local optima close to their initialization is desirable here. We are looking for the smallest possible change in $\alpha$, $\beta$ and $\gamma$ that brings the guided prediction closer to the hint while avoiding degenerate solutions. For example such as letting the given hint propagate to predict the whole image as the single, hinted class.

For further performance, the optimization process can be continued for the next hint by initializing with the previous state of the guiding parameters. Guiding by back-propagation is a flexible and interactive technique that does not require further training and can easily be added to any network. However, there are two caveats: First, the optimization procedure requires several gradient steps (in our experiments around 200), which make this approach about 200 times slower than a learned guide. Second, the hints need to be in the same format and modality as the original loss in order to be able to back-propagate to the guide.

## 13.2 Evaluation

Since the task is semantic segmentation on the PascalVOC 2012 dataset [36], we use a pre-trained FCN-8s network [117] and insert a guiding block in the smallest encoding layer. The choice of the slightly "outdated" FCN-8s architecture was deliberate since we believe that user interaction is mostly necessary and useful when the performance is not close to optimal.

We set up a user interaction scheme similar to the 20-question game. As stated before, after an initial prediction, the network is allowed to ask the user for the class of a single pixel and the guiding layer updates the feature representation using (13.1). We choose the pixel with the smallest posterior probability difference between the two most confident classes. The intuition behind this choice is that this pixel has the highest interclass uncertainty and is most likely to be flipped in a prediction. Again we compute the mean intersection over union (mIoU) after every question.

In Table 13.1, we list the performance after 0, 1, 5, 10, 15 and 20 questions. Naturally, 0 questions stands for the initial raw performance of FCN-8s without guiding. Over the course of 20 interactions with the user, a significant improvement of the performance from $62.6\%$

to $81.0\%$ is recorded. This is not directly comparable to our original twenty questions result, since we receive about $4.3$ bit ($\log_2(20\text{classes}) \approx 4.3$) of information for this task, while in the 20 questions game 1 bit is communicated. The current top entry on the PascalVOC leaderboard (DeepLab-v3 [22]) scores $86.9\%$ mIoU, and is trained on much more additional data. Guiding by back-propagation can be directly incorporated into a pre-trained CNN and using human interaction boosts a comparably low performance to reach the state of the art without additional training.

# Part V

## Conclusions

In this dissertation, we have presented two general concepts for dealing with ambiguity. First, multiple hypothesis prediction is a generic framework to learn ambiguous tasks with very little overhead and without additional annotations. The original MHP idea has already grown into several applications such as conditional image generation with adversarial neural networks, multiple action prediction for reinforcement learning, grasp point estimation for robotic applications and 6 DoF pose estimation of objects. Further, in several current projects we are exploring the applicability of MHP for trajectory estimation, depth prediction, feature visualization and visual question answering. We have extended the MHP idea to allow continuous sampling in the second chapter; we find that additional benefits of modeling ambiguity based on the output distributions come at the cost of more complex annotations and numerical stability.

In the final chapter, we show an alternative possibility for resolving ambiguities. We present a method that enables user–network interaction. By allowing a user into the evaluation procedure, we can leverage additional information to deal with unclear inputs or erroneous outputs. We show how user hints can be used to sort out confused predictions retrospectively.

Multiple hypothesis prediction and user interaction can also be well combined as they seem to have a synergistic effect. We can use the help of the user to resolve one of the main difficulties of MHP. Per definition, it is not possible to choose which hypothesis is best without auxiliary information. This is how suggestions from the user can be useful in deciding which hypothesis fits best.

Comparing MHP with Bayesian methods, such as variational auto-encoders or deep belief networks, would be interesting to outline the fundamental differences between these probabilistic frameworks and the pragmatic MHP approach.

The direction of network guiding can also be useful for learning from the hints provided by the user. We describe this idea through an example related to home robots. One can think of a scenario in which a robot, trained on generic indoor data is brought home. In this new environment, it would need to adapt to the specific objects in the household. “*This is my broom.*”, “*My brother likes to sit on this side of the table, so please put his cup there.*” and other such expressions could be provided to calibrate the pre-trained models of the robot to the specific circumstances of the new environment.

Finally, over the last years we have also become interested in feature visualization and the *explainability* of predictions. This topic is also coupled to the direction of interacting with neural networks. Naturally, for a meaningful interaction each partner must be able to explain their reasoning to the other to create mutual benefit and to arrive at a common decision.

# Part VI

Appendix

# A Authored and Co-authored Publications

## Authored

1. Laina, Iro*, **Christian Rupprecht***, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. *"Deeper depth prediction with fully convolutional residual networks."*, International Conference on 3D Vision (3DV), 2016 (*equal contribution).

2. **Rupprecht, Christian**, Iro Laina, Robert DiPietro, Maximilian Baust, Federico Tombari, Nassir Navab, and Gregory D. Hager. *"Learning in an uncertain world: Representing ambiguity through multiple hypotheses."* In International Conference on Computer Vision (ICCV). 2017.

3. **Rupprecht, Christian**, Loic Peter, and Nassir Navab. *"Image segmentation in twenty questions."* IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.

4. **Rupprecht, Christian***, Iro Laina*, Nassir Navab, Gregory D. Hager, and Federico Tombari. *"Guide Me: Interacting with Deep Networks."* IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR - spotlight), 2018 (*equal contribution).

5. **Rupprecht, Christian***, Colin Lea*, Federico Tombari, Nassir Navab, and Gregory D. Hager. *"Sensor substitution for video-based action recognition."* IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2016 (*equal contribution).

6. **Rupprecht, Christian**, Ansh Kapil, Nan Liu, Lamberto Ballan, and Federico Tombari. *"Learning without prejudice: Avoiding bias in webly-supervised action recognition."* Computer Vision and Image Understanding (CVIU), 2017.

7. **Rupprecht, Christian**, Elizabeth Huaroc, Maximilian Baust, and Nassir Navab. *"Deep active contours."* arXiv preprint arXiv:1607.05074, 2016 (under submission).

## Co-Authored

1. Laina, Iro*, Nicola Rieke*, **Christian Rupprecht**, Josué Page Vizcaíno, Abouzar Eslami, Federico Tombari, and Nassir Navab. "*Concurrent segmentation and localization for tracking of surgical instruments.*" International Conference on Medical Image Computing and Computer-Assisted Intervention, (MICCAI) 2017 (*equal contribution).

2. Sharma, Kanishka, **Christian Rupprecht**, Anna Caroli, Maria Carolina Aparicio, Andrea Remuzzi, Maximilian Baust, and Nassir Navab. "**Automatic segmentation of kidneys using deep learning for total kidney volume quantification in autosomal dominant polycystic kidney disease.**" Nature Scientific reports 7. 2017.

3. DiPietro, Robert, **Christian Rupprecht**, Nassir Navab, and Gregory D. Hager. "*Analyzing and Exploiting NARX Recurrent Neural Networks for Long-Term Dependencies.*" International Conference on Learning Representations - Workshop Track (ICLR-W). 2018.

4. Spier, Nathalia, Rischpler Christoph, **Christian Rupprecht**, Nassir Navab, Maximilian Baust, and Stephan Nekolla. "*Defect detection in cardiac SPECT using graph-based convolutional neural networks.*" Journal of Nuclear Medicine, 2018.

5. Kumar, Sanjeev, **Christian Rupprecht**, Federico Tombari, Gregory D. Hager, "*Predicting Multiple Actions for Stochastic Continuous Control.*", 2018 (under submission).

6. Ghazaei, Ghazal, Iro Laina, **Christian Rupprecht**, Federico Tombari, Nassir Navab, Kianoush Nazarpou, "*Dealing with Ambiguity in Robotic Grasping via Multiple Predictions*", Asian Conference on Computer Vision (ACCV), 2018.

7. Manhardt, Fabian, Diego Martín Arroyo, **Christian Rupprecht**, Benjamin Busam, Tolga Birdal, Nassir Navab, Federico Tombari, "*Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data.*" International Conference on Computer Vision, 2019.

# Bibliography

[1] A. Agrawal, J. Lu, S. Antol, et al. “Vqa: Visual question answering”. In: *International Journal of Computer Vision* 123.1 (2017), pp. 4–31 (cit. on p. 90).

[2] M. Amrehn, S. Gaube, M. Unberath, et al. “UI-Net: Interactive Artificial Neural Networks for Iterative Image Segmentation Based on a User Model”. In: *arXiv preprint arXiv:1709.03450* (2017) (cit. on p. 89).

[3] P. Anderson, X. He, C. Buehler, et al. “Bottom-up and top-down attention for image captioning and vqa”. In: *arXiv preprint arXiv:1707.07998* (2017) (cit. on p. 90).

[4] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. “Learning to compose neural networks for question answering”. In: *arXiv preprint arXiv:1601.01705* (2016) (cit. on p. 90).

[5] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. “Neural module networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 39–48 (cit. on p. 90).

[6] U. Asif, M. Bennamoun, and F. A. Sohel. “Rgb-d object recognition and grasp detection using hierarchical cascaded forests”. In: *IEEE Transactions on Robotics* 33.3 (2017), pp. 547–564 (cit. on pp. 53, 54, 56).

[7] V. Badrinarayanan, A. Kendall, and R. Cipolla. “Segnet: A deep convolutional encoder-decoder architecture for image segmentation”. In: *arXiv preprint arXiv:1511.00561* (2015) (cit. on p. 90).

[8] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. “icoseg: Interactive co-segmentation with intelligent scribble guidance”. In: *CVPR 2010*. IEEE. 2010, pp. 3169–3176 (cit. on p. 90).

[9] V. Belagiannis, C. Rupprecht, G. Carneiro, and N. Navab. “Robust optimization for deep regression”. In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2015, pp. 2830–2838 (cit. on pp. 25, 27).

[10] C. M. Bishop. “Mixture density networks”. In: (1994) (cit. on pp. 16, 17, 23, 29, 30, 70).

[11] Y. Boykov and M.-P. Jolly. “Interactive graph cuts for optimal boundary amp; region segmentation of objects in N-D images”. In: *ICCV 2001*. Vol. 1. 2001, 105–112 vol.1 (cit. on p. 90).

[12] S. Branson, G. Van Horn, C. Wah, P. Perona, and S. Belongie. “The ignorant led by the blind: A hybrid human–machine vision system for fine-grained categorization”. In: *International Journal of Computer Vision* 108.1-2 (2014), pp. 3–29 (cit. on p. 89).

[13] S. Branson, C. Wah, F. Schroff, et al. “Visual Recognition with Humans in the Loop”. English. In: *Computer Vision â€“ ECCV 2010*. Ed. by K. Daniilidis, P. Maragos, and N. Paragios. Vol. 6314. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, pp. 438–451 (cit. on p. 89).

[14] G. Brockman, V. Cheung, L. Pettersson, et al. *OpenAI Gym*. 2016. eprint: `arXiv:1606.01540` (cit. on pp. 44, 48).

[15] C. Broyden. “A new double-rank minimisation algorithm. Preliminary report”. In: *Notices of the American Mathematical Society*. Vol. 16. 4. AMER MATHEMATICAL SOC 201 CHARLES ST, PROVIDENCE, RI 02940-2213. 1969, p. 670 (cit. on p. 7).

[16] A. Bulat and G. Tzimiropoulos. “Human pose estimation via convolutional part heatmap regression”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 717–732 (cit. on p. 53).

[17] H. Caesar, J. Uijlings, and V. Ferrari. “COCO-Stuff: Thing and Stuff Classes in Context”. In: *arXiv preprint arXiv:1612.03716* (2016) (cit. on p. 114).

[18] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. “Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 7291–7299 (cit. on p. 53).

[19] J. Carreira and C. Sminchisescu. “CPMC: Automatic Object Segmentation Using Constrained Parametric Min-Cuts”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012) (cit. on p. 90).

[20] B.-C. Chen, C.-S. Chen, and W. H. Hsu. “Cross-age reference coding for age-invariant face recognition and retrieval”. In: *European Conference on Computer Vision*. Springer. 2014, pp. 768–783 (cit. on p. 81).

[21] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs”. In: *arXiv preprint arXiv:1606.00915* (2016) (cit. on pp. 90, 91, 114).

[22] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. “Rethinking atrous convolution for semantic image segmentation”. In: *arXiv preprint arXiv:1706.05587* (2017) (cit. on p. 127).

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “Semantic Image Segmentation with Deep Convolutional Nets and Fully Connected CRFs”. In: *ICLR*. 2015 (cit. on p. 7).

[24] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. “On the properties of neural machine translation: Encoder-decoder approaches”. In: *arXiv preprint arXiv:1409.1259* (2014) (cit. on p. 111).

[25] A. Choromanska, M. Henaff, M. Mathieu, G. B. Arous, and Y. LeCun. “The loss surfaces of multilayer networks”. In: *Artificial Intelligence and Statistics*. 2015, pp. 192–204 (cit. on p. 7).

[26] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014) (cit. on p. 111).

[27] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. “Fast and accurate deep network learning by exponential linear units (elus)”. In: *arXiv preprint arXiv:1511.07289* (2015) (cit. on p. 6).

[28] B. Dai, D. Lin, R. Urtasun, and S. Fidler. “Towards Diverse and Natural Image Descriptions via a Conditional GAN”. In: *arXiv preprint arXiv:1703.06029* (2017) (cit. on p. 90).

[29] J. Dai, K. He, and J. Sun. “Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1635–1643 (cit. on p. 89).

[30] A. Das, S. Kottur, K. Gupta, et al. “Visual dialog”. In: *arXiv preprint arXiv:1611.08669* (2016) (cit. on p. 90).

[31] D. Dey, V. Ramakrishna, M. Hebert, and J Andrew Bagnell. “Predicting multiple structured visual interpretations”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2947–2955 (cit. on p. 16).

[32] Q. Du, V. Faber, and M. Gunzburger. “Centroidal Voronoi tessellations: applications and algorithms”. In: *SIAM review* 41.4 (1999), pp. 637–676 (cit. on pp. 17, 21).

[33] F. Dubost, L. Peter, C. Rupprecht, B. G. Becker, and N. Navab. “Hands-Free Segmentation of Medical Volumes via Binary Inputs”. In: *Deep Learning and Data Labeling for Medical Applications*. Springer International Publishing, 2016 (cit. on pp. 89, 103).

[34] S. Efromovich. “Conditional density estimation in a regression setting”. In: *The Annals of Statistics* (2007), pp. 2504–2535 (cit. on p. 70).

[35] I. Endres and D. Hoiem. “Category-Independent Object Proposals with Diverse Ranking”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 36.2 (2014), pp. 222–234 (cit. on p. 90).

[36] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html (cit. on p. 126).

[37] M Everingham, L Van˜Gool, C. K. I. Williams, J Winn, and A Zisserman. “The Pascal Visual Object Classes (VOC) Challenge”. In: *International Journal of Computer Vision* 88.2 (2010), pp. 303–338 (cit. on p. 31).

[38] A. Fathi, M. F. Balcan, X. Ren, and J. M. Rehg. “Combining self training and active learning for video segmentation”. In: *Proceedings of the British Machine Vision Conference (BMVC 2011)*. Vol. 29. 2011, pp. 78–1 (cit. on p. 90).

[39] R. Fletcher. “A new approach to variable metric algorithms”. In: *The computer journal* 13.3 (1970), pp. 317–322 (cit. on p. 7).

[40] D. F. Fouhey and C. L. Zitnick. “Predicting object dynamics in scenes”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 2019–2026 (cit. on p. 17).

[41] Y. Gal and Z. Ghahramani. “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning”. In: *International Conference on Machine Learning*. Vol. 2. 2015, pp. 1050–1059 (cit. on pp. 17, 70, 76).

[42] B.-B. Gao, C. Xing, C.-W. Xie, J. Wu, and X. Geng. “Deep Label Distribution Learning with Label Ambiguity”. In: *arXiv preprint arXiv:1611.01731* (2016) (cit. on pp. 16, 17, 71).

[43] X. Geng and Y. Xia. “Head pose estimation based on multivariate label distribution”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1837–1842 (cit. on p. 16).

[44] X. Geng, C. Yin, and Z.-H. Zhou. “Facial age estimation by learning from label distributions”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.10 (2013), pp. 2401–2412 (cit. on p. 16).

[45] G. Ghazaei, I. Laina, C. Rupprecht, F. Tombari, N. Navab, and K. Nazarpou. “Dealing with Ambiguity in Robotic Grasping via Multiple Predictions”. In: 2018 (cit. on pp. 4, 25, 53).

[46] R. Girshick. “Fast R-CNN”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 1440–1448 (cit. on p. 30).

[47] 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 (cit. on p. 6).

[48] D. Goldfarb. “A family of variable-metric methods derived by variational means”. In: *Mathematics of computation* 24.109 (1970), pp. 23–26 (cit. on p. 7).

[49] Y. Gong, Y. Jia, T. Leung, A. Toshev, and S. Ioffe. “Deep convolutional ranking for multilabel image annotation”. In: *arXiv preprint arXiv:1312.4894* (2013) (cit. on pp. 16, 30).

[50] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*. Vol. 1. MIT press Cambridge, 2016 (cit. on p. 5).

[51] S. Gould, R. Fulton, and D. Koller. “Decomposing a scene into geometric and semantically consistent regions”. In: *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE. 2009, pp. 1–8 (cit. on pp. 102, 103).

[52] L. Grady. “Random Walks for Image Segmentation”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 28.11 (2006), pp. 1768–1783 (cit. on p. 90).

[53] L. Grady, M.-P. Jolly, and A. Seitz. “Segmentation from a box”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 367–374 (cit. on p. 90).

[54] C. Gu, J. Lim, P. Arbelaez, and J. Malik. “Recognition using regions”. In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 2009, pp. 1030–1037 (cit. on p. 90).

[55] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi. “A hybrid deep architecture for robotic grasp detection”. In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. IEEE. 2017 (cit. on pp. 53, 54, 56, 58, 59).

[56] A. Guzman-Rivera, P. Kohli, D. Batra, and R. Rutenbar. “Efficiently enforcing diversity in multi-output structured prediction”. In: *Artificial Intelligence and Statistics*. 2014, pp. 284–292 (cit. on pp. 16, 70, 75).

[57] A. Guzman-Rivera, D. Batra, and P. Kohli. “Multiple choice learning: Learning to produce multiple structured outputs”. In: *Advances in Neural Information Processing Systems*. 2012, pp. 1799–1807 (cit. on pp. 70, 71, 75).

[58] R. H. Hahnloser, R. Sarpeshkar, M. A. Mahowald, R. J. Douglas, and H. S. Seung. “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit”. In: *Nature* 405.6789 (2000), p. 947 (cit. on p. 6).

[59] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778 (cit. on pp. 7, 8, 25, 30, 110, 114).

[60] N. Heess, G. Wayne, D. Silver, T. P. Lillicrap, T. Erez, and Y. Tassa. “Learning Continuous Control Policies by Stochastic Value Gradients”. In: *Advances in Neural Information Processing Systems*. 2015, pp. 2944–2952 (cit. on p. 45).

[61] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. “Deep Reinforcement Learning that Matters”. In: *arXiv preprint arXiv:1709.06560* (2017) (cit. on p. 51).

[62] C. Hesse, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. *OpenAI Baselines*. `https://github.com/openai/baselines`. 2017 (cit. on p. 48).

[63] S. Hinterstoisser, C. Cagniart, S. Ilic, et al. “Gradient Response Maps for Real-Time Detection of Textureless Objects”. In: *TPAMI* (2012) (cit. on p. 60).

[64] S. Hinterstoisser, V. Lepetit, S. Ilic, et al. “Model based training, detection and pose estimation of texture-less 3D objects in heavily cluttered scenes”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Vol. 7724 LNCS. 2013, pp. 548–562 (cit. on pp. 61–63).

[65] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. “Improving neural networks by preventing co-adaptation of feature detectors”. In: *arXiv preprint arXiv:1207.0580* (2012) (cit. on p. 8).

[66] T. Hodaň, P. Haluza, Š. Obdrzalek, J. Matas, M. Lourakis, and X. Zabulis. “T-LESS: An RGB-D dataset for 6D pose estimation of texture-less objects”. In: *Proceedings - 2017 IEEE Winter Conference on Applications of Computer Vision, WACV 2017* (2017), pp. 880–888 (cit. on p. 60).

[67] T. Hodan, J. Matas, and S. Obdrzalek. “On Evaluation of 6D Object Pose Estimation”. In: *ECCV Workshop*. 2016 (cit. on p. 61).

[68] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. “Learning to reason: End-to-end module networks for visual question answering”. In: *arXiv preprint arXiv:1704.05526* (2017) (cit. on p. 90).

[69] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. “Natural language object retrieval”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4555–4564 (cit. on p. 90).

[70] R. Hu, M. Rohrbach, and T. Darrell. “Segmentation from natural language expressions”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 108–124 (cit. on pp. 90, 113).

[71] S. Ioffe and C. Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015) (cit. on pp. 8, 110).

[72] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. “Image-To-Image Translation With Conditional Adversarial Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 1125–1134 (cit. on pp. 40, 42).

[73] B. Jedynak, P. I. Frazier, and R. Sznitman. “Twenty questions with noise: Bayes optimal policies for entropy loss”. In: *Journal of Applied Probability* 49.1 (2012), pp. 114–136 (cit. on p. 89).

[74] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. “The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation”. In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*. IEEE. 2017, pp. 1175–1183 (cit. on p. 90).

[75] Y. Jiang, S. Moseson, and A. Saxena. “Efficient grasping from rgbd images: Learning using a new rectangle representation”. In: *Int. Conf. on Robotics and Automation (ICRA)*. IEEE. 2011, pp. 3304–3311 (cit. on pp. 53, 54, 57).

[76] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. “CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning”. In: *arXiv preprint arXiv:1612.06890* (2016) (cit. on p. 90).

[77] J. Johnson, A. Karpathy, and L. Fei-Fei. “DenseCap: Fully Convolutional Localization Networks for Dense Captioning”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on p. 90).

[78] J. Johnson, B. Hariharan, L. van der Maaten, et al. “Inferring and Executing Programs for Visual Reasoning”. In: *arXiv preprint arXiv:1705.03633* (2017) (cit. on p. 90).

[79] S. Johnson and M. Everingham. “Clustered Pose and Nonlinear Appearance Models for Human Pose Estimation”. In: *Proceedings of the British Machine Vision Conference*. 2010 (cit. on p. 28).

[80] H. Karcher. “Riemannian center of mass and mollifier smoothing”. In: *Communications on pure and applied mathematics* 30.5 (1977), pp. 509–541 (cit. on p. 62).

[81] S. Kazemzadeh, V. Ordonez, M. Matten, and T. L. Berg. “ReferItGame: Referring to Objects in Photographs of Natural Scenes.” In: (cit. on p. 113).

[82] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab. “Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation”. In: *IEEE European Conference on Computer Vision (ECCV)*. 2016 (cit. on p. 63).

[83] W. Kehl, F. Manhardt, S. Ilic, F. Tombari, and N. Navab. “SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again”. In: *ICCV*. 2017 (cit. on pp. 61–63).

[84] B. Kehoe, S. Patil, P. Abbeel, and K. Goldberg. “A survey of research on cloud robotics and automation”. In: *IEEE Transactions on automation science and engineering* 12.2 (2015), pp. 398–409 (cit. on p. 53).

[85] A. Kendall, V. Badrinarayanan, and R. Cipolla. “Bayesian segnet: Model uncertainty in deep convolutional encoder-decoder architectures for scene understanding”. In: *arXiv preprint arXiv:1511.02680* (2015) (cit. on p. 71).

[86] D. P. Kingma and J. Ba. “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980* (2014) (cit. on pp. 7, 80).

[87] D. P. Kingma and M. Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2014) (cit. on pp. 17, 71).

[88] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. “Self-normalizing neural networks”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 972–981 (cit. on p. 6).

[89] A. N. Kolmogorov. “Foundations of the Theory of Probability.” In: (1950), pp. 47–64 (cit. on p. 19).

[90] H. S. Koppula and A. Saxena. “Anticipating human activities using object affordances for reactive robotic response”. In: *IEEE transactions on pattern analysis and machine intelligence* 38.1 (2016), pp. 14–29 (cit. on p. 16).

[91] A. Kowdle, Y.-J. Chang, A. Gallagher, and T. Chen. “Active learning for piecewise planar 3D reconstruction”. In: *CVPR, 2011*. IEEE. 2011, pp. 929–936 (cit. on p. 90).

[92] A. Kowdle, D. Batra, W.-C. Chen, and T. Chen. “imodel: interactive co-segmentation for object of interest 3d modeling”. In: *Trends and Topics in Computer Vision*. Springer, 2012, pp. 211–224 (cit. on p. 90).

[93] P. Krähenbühl and V. Koltun. “Geodesic Object Proposals”. English. In: *Computer Vision â€“ ECCV 2014*. Ed. by D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars. Vol. 8693. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 725–739 (cit. on pp. 90, 96, 103, 104).

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks”. In: *Advances in neural information processing systems*. 2012, pp. 1097–1105 (cit. on pp. 7, 8).

[95] S Kullback and R. A. Leibler. “On Information and Sufficiency”. In: *Ann. Math. Statist.* 22.1 (1951), pp. 79–86 (cit. on pp. 70, 73).

[96] S. Kumar, C. Rupprecht, F. Tombari, and G. D. Hager. “Predicting Multiple Actions for Stochastic Continuous Control”. In: *(under submission)* (2018) (cit. on pp. 4, 25).

[97] S. Kumra and C. Kanan. “Robotic grasp detection using deep convolutional neural networks”. In: *arXiv preprint arXiv:1611.08036* (2016) (cit. on pp. 53, 54, 56, 59).

[98] I. Laina, N. Rieke, C. Rupprecht, et al. “Concurrent Segmentation and Localization for Tracking of Surgical Instruments”. In: *arXiv preprint arXiv:1703.10701* (2017) (cit. on pp. 4, 53).

[99] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. “Deeper depth prediction with fully convolutional residual networks”. In: *3D Vision (3DV), 2016 Fourth International Conference on*. IEEE. 2016, pp. 239–248 (cit. on pp. 3, 25, 33, 55).

[100] A. Lazaric, M. Restelli, and A. Bonarini. “Reinforcement learning in continuous action spaces through sequential monte carlo methods”. In: *Advances in Neural Information Processing Systems*. 2007 (cit. on p. 45).

[101] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-based learning applied to document recognition”. In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324 (cit. on pp. 7, 8).

[102] K. Lee, C. Hwang, K. Park, and J. Shin. “Confident Multiple Choice Learning”. In: *arXiv preprint arXiv:1706.03475* (2017) (cit. on p. 16).

[103] S. Lee, S. P. S. Prakash, M. Cogswell, V. Ranjan, D. Crandall, and D. Batra. “Stochastic Multiple Choice Learning for Training Diverse Deep Ensembles”. In: *Advances in Neural Information Processing Systems*. 2016, pp. 2119–2127 (cit. on pp. 16, 38, 39, 71, 75).

[104] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. “Image Segmentation with A Bounding Box Prior”. In: *ICCV*. MSR-TR-2009-85. 2009 (cit. on p. 90).

[105] I. Lenz, H. Lee, and A. Saxena. “Deep learning for detecting robotic grasps”. In: *The International Journal of Robotics Research* (2015) (cit. on pp. 53, 54, 56, 57, 59).

[106] N. Leonenko, L. Pronzato, V. Savani, and Others. “A class of Rényi information estimators for multidimensional densities”. In: *The Annals of Statistics* 36.5 (2008), pp. 2153–2182 (cit. on p. 70).

[107] A. Lerer, S. Gross, and R. Fergus. “Learning Physical Intuition of Block Towers by Example”. In: *arXiv preprint arXiv:1603.01312* (2016) (cit. on p. 16).

[108] S. Levine and V. Koltun. “Guided policy search”. In: *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*. 2013, pp. 1–9 (cit. on p. 45).

[109] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. “Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection”. In: *arXiv preprint arXiv:1603.02199* (2016) (cit. on p. 53).

[110] Z. Li, Q. Chen, and V. Koltun. “Interactive Image Segmentation with Latent Diversity”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 577–585 (cit. on pp. 27, 75).

[111] T. P. Lillicrap, J. J. Hunt, A. Pritzel, et al. “Continuous Control with Deep Reinforcement Learning”. In: *Proc. International Conference on Learning Representations (ICLR)*. 2016 (cit. on pp. 43–45).

[112] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. “Scribblesup: Scribble-supervised convolutional networks for semantic segmentation”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3159–3167 (cit. on p. 89).

[113] G. Lin, A. Milan, C. Shen, and I. Reid. “Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation”. In: *arXiv preprint arXiv:1611.06612* (2016) (cit. on p. 90).

[114] T.-Y. Lin, M. Maire, S. Belongie, et al. “Microsoft COCO: Common Objects in Context”. In: *arXiv preprint arXiv:1405.0312* (2014) (cit. on pp. 113, 114).

[115] T.-Y. Lin, M. Maire, S. J. Belongie, et al. “Microsoft {COCO:} Common Objects in Context”. In: *CoRR* abs/1405.0312 (2014) (cit. on p. 31).

[116] S. Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pp. 129–137 (cit. on p. 22).

[117] 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, pp. 3431–3440 (cit. on pp. 7, 25, 38, 90, 91, 97, 126).

[118] J. Lu, J. Yang, D. Batra, and D. Parikh. “Hierarchical question-image co-attention for visual question answering”. In: *Advances In Neural Information Processing Systems*. 2016, pp. 289–297 (cit. on p. 90).

[119] J. Lu, C. Xiong, D. Parikh, and R. Socher. “Knowing when to look: Adaptive attention via A visual sentinel for image captioning”. In: *arXiv preprint arXiv:1612.01887* (2016) (cit. on p. 90).

[120] A. L. Maas, A. Y. Hannun, and A. Y. Ng. “Rectifier nonlinearities improve neural network acoustic models”. In: *Proc. icml*. Vol. 30. 1. 2013, p. 3 (cit. on p. 6).

[121] J. Mahler, J. Liang, S. Niyaz, et al. “Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics”. In: *arXiv preprint arXiv:1703.09312* (2017) (cit. on p. 53).

[122] M. Malinowski, M. Rohrbach, and M. Fritz. “Ask your neurons: A deep learning approach to visual question answering”. In: *International Journal of Computer Vision* 125.1-3 (2017), pp. 110–135 (cit. on p. 90).

[123] F. Manhardt, D. M. Arroyo, C. Rupprecht, et al. “Explaining the Ambiguity of Object Detection and 6D Pose from Visual Data”. In: 2019 (cit. on pp. 4, 25, 60).

[124] J. Mao, J. Huang, A. Toshev, O. Camburu, A. Yuille, and K. Murphy. “Generation and Comprehension of Unambiguous Object Descriptions”. In: *CVPR*. 2016 (cit. on p. 113).

[125] D. Martin, C. Fowlkes, D. Tal, and J. Malik. “A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics”. In: *Proc. 8th Int’l Conf. Computer Vision*. Vol. 2. 2001, pp. 416–423 (cit. on pp. 97, 101, 102).

[126] W. S. McCulloch and W. Pitts. “A logical calculus of the ideas immanent in nervous activity”. In: *The bulletin of mathematical biophysics* 5.4 (1943), pp. 115–133 (cit. on p. 5).

[127] G. McLachlan and D. Peel. *Finite mixture models*. John Wiley & Sons, 2004 (cit. on pp. 54, 56).

[128] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092 (cit. on pp. 93, 95).

[129] A. T. Miller and P. K. Allen. “Graspit! a versatile simulator for robotic grasping”. In: *IEEE Robotics & Automation Magazine* (2004) (cit. on p. 53).

[130] M. Minsky and S. A. Papert. *Perceptrons: an introduction to computational geometry*. 1969 (cit. on p. 6).

[131] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. “Cross-stitch networks for multi-task learning”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3994–4003 (cit. on p. 110).

[132] V. Mnih, A. P. Badia, M. Mirza, et al. “Asynchronous Methods for Deep Reinforcement Learning”. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by M. F. Balcan and K. Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. PMLR, 2016, pp. 1928–1937 (cit. on pp. 43, 45).

[133] V. Mnih, K. Kavukcuoglu, D. Silver, et al. “Human-level control through deep reinforcement learning”. In: *Nature* 518 (2015), pp. 529–533 (cit. on pp. 43, 45).

[134] V. Nair and G. E. Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814 (cit. on p. 6).

[135] A. Newell, K. Yang, and J. Deng. “Stacked hourglass networks for human pose estimation”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 483–499 (cit. on p. 53).

[136] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu. *Spatial tessellations: concepts and applications of Voronoi diagrams*. Vol. 501. John Wiley & Sons, 2009 (cit. on p. 17).

[137] M. Olazaran. “A sociological study of the official history of the perceptrons controversy”. In: *Social Studies of Science* 26.3 (1996), pp. 611–659 (cit. on p. 5).

[138] J. Pennington, R. Socher, and C. Manning. “Glove: Global vectors for word representation”. In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543 (cit. on p. 111).

[139] E. Perez, F. Strub, H. de Vries, V. Dumoulin, and A. Courville. “FiLM: Visual Reasoning with a General Conditioning Layer”. In: *arXiv preprint arXiv:1709.07871* (2017) (cit. on pp. 90, 110, 117, 118).

[140] F. Pérez-Cruz. “Kullback-Leibler divergence estimation of continuous distributions”. In: *Information Theory, 2008. ISIT 2008. IEEE International Symposium on*. IEEE. 2008, pp. 1666–1670 (cit. on pp. 70, 73).

[141] M. Plappert, R. Houthooft, P. Dhariwal, et al. “Parameter Space Noise for Exploration”. In: *CoRR* abs/1706.01905 (2017) (cit. on p. 44).

[142] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe. “Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes”. In: *arXiv preprint arXiv:1611.08323* (2016) (cit. on p. 90).

[143] B. T. Polyak. “Some methods of speeding up the convergence of iteration methods”. In: *USSR Computational Mathematics and Mathematical Physics* 4.5 (1964), pp. 1–17 (cit. on p. 7).

[144] B. Price, B. Morse, and S. Cohen. “Geodesic graph cut for interactive image segmentation”. In: *CVPR 2010*. 2010, pp. 3161–3168 (cit. on p. 90).

[145] M. Rad and V. Lepetit. “BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 3848–3856 (cit. on p. 63).

[146] M. Rajchl, M. C. H. Lee, O. Oktay, et al. “Deepcut: Object segmentation from bounding box annotations using convolutional neural networks”. In: *IEEE transactions on medical imaging* 36.2 (2017), pp. 674–683 (cit. on p. 89).

[147] C. E. Rasmussen. “Gaussian processes in machine learning”. In: *Advanced lectures on machine learning*. Springer, 2004, pp. 63–71 (cit. on p. 70).

[148] J. Redmon and A. Angelova. “Real-time grasp detection using convolutional neural networks”. In: *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE. 2015, pp. 1316–1322 (cit. on pp. 53, 54, 56, 59).

[149] S. Ren, K. He, R. Girshick, and J. Sun. “Faster {R-CNN}: Towards Real-Time Object Detection with Region Proposal Networks”. In: *Advances in Neural Information Processing Systems ({NIPS})*. 2015 (cit. on p. 7).

[150] D. J. Rezende, S. Mohamed, and D. Wierstra. “Stochastic backpropagation and approximate inference in deep generative models”. In: *International Conference on Machine Learning (ICML)* (2015) (cit. on p. 71).

[151] R. Rothe, R. Timofte, and L. Van Gool. “Deep expectation of real and apparent age from a single image without facial landmarks”. In: *International Journal of Computer Vision* (), pp. 1–14 (cit. on pp. 82, 83).

[152] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *arXiv preprint arXiv:1609.04747* (2016) (cit. on p. 7).

[153] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), p. 533 (cit. on p. 6).

[154] C. Rupprecht, E. Huaroc, M. Baust, and N. Navab. “Deep active contours”. In: *arXiv preprint arXiv:1607.05074* (2016) (cit. on p. 3).

[155] C. Rupprecht, I. Laina, N. Navab, G. D. Harger, and F. Tombari. “Guide Me: Interacting with Deep Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018) (cit. on pp. 89, 90, 97, 107).

[156] C. Rupprecht, L. Peter, and N. Navab. “Image segmentation in twenty questions”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3314–3322 (cit. on pp. 89, 93).

[157] C. Rupprecht, I. Laina, R. DiPietro, et al. “Learning in an Uncertain World: Representing Ambiguity Through Multiple Hypotheses”. In: *arXiv preprint arXiv:1612.00197* (2016) (cit. on pp. 15, 25, 71, 75, 76).

[158] C. Rupprecht, I. Laina, R. DiPietro, et al. “Learning in an uncertain world: Representing ambiguity through multiple hypotheses”. In: *International Conference on Computer Vision (ICCV)*. 2017 (cit. on p. 4).

[159] C. Rupprecht, A. Kapil, N. Liu, L. Ballan, and F. Tombari. “Learning without prejudice: Avoiding bias in webly-supervised action recognition”. In: *Computer Vision and Image Understanding* (2017) (cit. on p. 4).

[160] C. Rupprecht, C. Lea, F. Tombari, N. Navab, and G. D. Hager. “Sensor substitution for video-based action recognition”. In: *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE. 2016, pp. 5230–5237 (cit. on p. 4).

[161] O. Russakovsky, J. Deng, H. Su, et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252 (cit. on p. 33).

[162] H. Rutishauser. “Theory of gradient methods”. In: *Refined iterative methods for computation of the solution and the eigenvalues of self-adjoint boundary value problems*. Springer, 1959, pp. 24–49 (cit. on p. 7).

[163] A. Saxena, J. Driemeyer, and A. Y. Ng. “Robotic grasping of novel objects using vision”. In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173 (cit. on pp. 53, 54).

[164] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: *arXiv* abs/1707.06347 (2017) (cit. on p. 45).

[165] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel. “Trust Region Policy Optimization”. In: *Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37*. ICML’15. JMLR.org, 2015, pp. 1889–1897 (cit. on p. 45).

[166] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. “NTU RGB+D: A Large Scale Dataset for 3D Human Activity Analysis”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on p. 36).

[167] D. F. Shanno. “Conditioning of quasi-Newton methods for function minimization”. In: *Mathematics of computation* 24.111 (1970), pp. 647–656 (cit. on p. 7).

[168] K. Sharma, C. Rupprecht, A. Caroli, et al. “Automatic segmentation of kidneys using deep learning for total kidney volume quantification in autosomal dominant polycystic kidney disease”. In: *Scientific reports* 7.1 (2017), p. 2049 (cit. on p. 4).

[169] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. “Deterministic policy gradient algorithms”. In: *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*. 2014, pp. 387–395 (cit. on p. 45).

[170] D. Silver, A. Huang, C. J. Maddison, et al. “Mastering the Game of Go with Deep Neural Networks and Tree Search”. In: *Nature* 529.7587 (2016), pp. 484–489 (cit. on p. 43).

[171] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *CoRR* abs/1409.1 (2014) (cit. on p. 82).

[172] T. Sørensen. “A method of establishing groups of equal amplitude in plant sociology based on similarity of species and its application to analyses of the vegetation on Danish commons”. In: *Biol. skr.* 5 (1948), pp. 1–34 (cit. on p. 101).

[173] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. “Dropout: a simple way to prevent neural networks from overfitting.” In: *Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958 (cit. on pp. 8, 23, 70, 76).

[174] C. Straehle, U. Koethe, G. Knott, K. Briggman, W. Denk, and F. Hamprecht. “Seeded watershed cut uncertainty estimators for guided interactive segmentation”. In: *CVPR, 2012*. 2012, pp. 765–772 (cit. on p. 90).

[175] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. “On the importance of initialization and momentum in deep learning”. In: *International conference on machine learning*. 2013, pp. 1139–1147 (cit. on p. 7).

[176] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. Vol. 1. MIT press Cambridge, 1998 (cit. on p. 45).

[177] A. Tejani, D. Tang, R. Kouskouridas, and T.-k. Kim. “Latent-class hough forests for 3D object detection and pose estimation”. In: *IEEE European Conference on Computer Vision (ECCV)*. 2014 (cit. on p. 63).

[178] E. Todorov, T. Erez, and Y. Tassa. “MuJoCo: A physics engine for model-based control”. In: *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE. 2012, pp. 5026–5033 (cit. on p. 48).

[179] P. J. Toivanen. “New geodosic distance transforms for gray-scale images”. In: *Pattern Recognition Letters* 17.5 (1996), pp. 437–450 (cit. on p. 97).

[180] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. “Large margin methods for structured and interdependent output variables”. In: *Journal of machine learning research* 6.Sep (2005), pp. 1453–1484 (cit. on p. 70).

[181] Z. Tu and S.-C. Zhu. “Image segmentation by data-driven Markov chain Monte Carlo”. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 24.5 (2002), pp. 657–673 (cit. on pp. 90, 95).

[182] G. E. Uhlenbeck and L. S. Ornstein. “On the theory of the Brownian motion”. In: *Physical review*. Vol. 36.5. 1930, p. 823 (cit. on p. 44).

[183] J. Varley, C. DeChant, A. Richardson, A. Nair, J. Ruales, and P. Allen. “Shape completion enabled robotic grasping”. In: *arXiv preprint arXiv:1609.08546* (2016) (cit. on p. 53).

[184] U. Viereck, A. Pas, K. Saenko, and R. Platt. “Learning a visuomotor controller for real world robotic grasping using simulated depth images”. In: *Conference on Robot Learning*. 2017, pp. 291–300 (cit. on p. 53).

[185] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. “Show and tell: A neural image caption generator”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164 (cit. on p. 90).

[186] C. Vondrick, H. Pirsiavash, and A. Torralba. “Anticipating the future by watching unlabeled video”. In: *International Coference on Computer Vision and Pattern Recognition (CVPR)* (2016) (cit. on pp. 17, 71).

[187] C. Wah, S. Branson, P. Perona, and S. Belongie. “Multiclass recognition and part localization with humans in the loop”. In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. 2011, pp. 2524–2531 (cit. on p. 89).

[188] J. Walker, C. Doersch, A. Gupta, and M. Hebert. “An uncertain future: Forecasting from static images using variational autoencoders”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 835–851 (cit. on p. 17).

[189] G. Wang, M. A. Zuluaga, W. Li, et al. “Deepigeos: A deep interactive geodesic framework for medical image segmentation”. In: *arXiv preprint arXiv:1707.00652* (2017) (cit. on p. 90).

[190] J. Wang, Y. Yang, J. Mao, Z. Huang, C. Huang, and W. Xu. “CNN-RNN: A Unified Framework for Multi-label Image Classification”. In: *Proceedings of the Int. Conf. on Computer Vision and Pattern Recognition (CVPR)* (2016) (cit. on pp. 16, 30).

[191] M. Wang, C. Luo, R. Hong, J. Tang, and J. Feng. “Beyond Object Proposals: Random Crop Pooling for Multi-label Image Recognition”. In: *IEEE Transactions on Image Processing* 25.12 (2016) (cit. on p. 16).

[192] Q. Wang, S. R. Kulkarni, and S. Verdú. “A nearest-neighbor approach to estimating divergence between continuous random vectors”. In: *Information Theory, 2006 IEEE International Symposium on*. IEEE. 2006, pp. 242–246 (cit. on pp. 70, 73, 74).

[193] Z. Wang, Z. Li, B. Wang, and H. Liu. “Robot grasp detection using multimodal deep convolutional neural networks”. In: *Advances in Mechanical Engineering* 8.9 (2016), p. 1687814016668077 (cit. on pp. 53, 54, 56).

[194] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. “Convolutional pose machines”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4724–4732 (cit. on p. 53).

[195] Y. Wei, W. Xia, J. Huang, et al. "CNN: Single-label to multi-label". In: *arXiv preprint arXiv:1406.5726* (2014) (cit. on pp. 16, 30).

[196] B. Wymann, E. Espié, C. Guionneau, C. Dimitrakakis, R. Coulom, and A. Sumner. *TORCS, The Open Racing Car Simulator*. http://www.torcs.org. 2014 (cit. on p. 44).

[197] C. Xing, D. Arpit, C. Tsirigotis, and Y. Bengio. "A Walk with SGD". In: *arXiv preprint arXiv:1802.08770* (2018) (cit. on p. 7).

[198] H. Xu and K. Saenko. "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering". In: *European Conference on Computer Vision*. Springer. 2016, pp. 451–466 (cit. on p. 90).

[199] K. Xu, J. Ba, R. Kiros, et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2015, pp. 2048–2057 (cit. on p. 90).

[200] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang. "Deep interactive object selection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 373–381 (cit. on p. 89).

[201] H. Yang, J. T. Zhou, Y. Zhang, B.-B. Gao, J. Wu, and J. Cai. "Can Partial Strong Labels Boost Multi-label Object Recognition?" In: *arXiv preprint arXiv:1504.05843* (2015) (cit. on p. 16).

[202] J. Yuen and A. Torralba. "A data-driven approach for event prediction". In: *European Conference on Computer Vision*. Springer. 2010, pp. 707–720 (cit. on p. 16).

[203] B. S. Zapata-Impata. "Using Geometry to Detect Grasping Points on 3D Unknown Point Cloud". In: *International Conference on Informatics in Control, Automation and Robotics*. 2017, pp. 154–161 (cit. on pp. 53, 54).

[204] R. Zhang, J.-Y. Zhu, P. Isola, et al. "Real-time user-guided image colorization with learned deep priors". In: *arXiv preprint arXiv:1705.02999* (2017) (cit. on p. 89).

[205] B. Zhou, Y. Tian, S. Sukhbaatar, A. Szlam, and R. Fergus. "Simple baseline for visual question answering". In: *arXiv preprint arXiv:1512.02167* (2015) (cit. on p. 90).

[206] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017 (cit. on pp. 39, 40, 42, 43).

# List of Figures

# List of Tables