Eberhard Karls Universität Tübingen
Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik

Master Thesis Computer Science

Object Detection Using Deep Learning - Learning where to search using visual attention

Alina Kloss

May 26, 2015

Reviewers

Prof. Hendrik Lensch
Computer Graphics
Wilhelm-Schickard-Institute for
Computer Science
University of Tübingen

Prof. Martin Butz
Cognitive Modeling
Wilhelm-Schickard-Institute for
Computer Science
University of Tübingen

Supervisors

Dr. Jeannette Bohg
Autonomous Motion Department
Max-Planck-Institute for
Intelligent Systems

Dipl. Inf. Daniel Kappler
Autonomous Motion Department
Max-Planck-Institute for
Intelligent Systems

Kloss, Alina:
Object Detection Using Deep Learning -
Learning where to search using visual attention
Master Thesis Computer Science
Eberhard Karls Universität Tübingen
Thesis period: 19.11.2014 – 26.05.2015

Abstract

Detecting and identifying the different objects in an image fast and reliably is an important skill for interacting with one's environment. The main problem is that in theory, all parts of an image have to be searched for objects on many different scales to make sure that no object instance is missed. It however takes considerable time and effort to actually classify the content of a given image region and both time and computational capacities that an agent can spend on classification are limited. Humans use a process called visual attention to quickly decide which locations of an image need to be processed in detail and which can be ignored. This allows us to deal with the huge amount of visual information and to employ the capacities of our visual system efficiently.

For computer vision, researchers have to deal with exactly the same problems, so learning from the behaviour of humans provides a promising way to improve existing algorithms. In the presented master's thesis, a model is trained with eye tracking data recorded from 15 participants that were asked to search images for objects from three different categories. It uses a deep convolutional neural network to extract features from the input image that are then combined to form a saliency map. This map provides information about which image regions are interesting when searching for the given target object and can thus be used to reduce the parts of the image that have to be processed in detail. The method is based on a recent publication of Kümmerer et al., but in contrast to the original method that computes general, task independent saliency, the presented model is supposed to respond differently when searching for different target categories.

Acknowledgements

First of all, I want to thank my supervisors Jeannette and Daniel for giving me the chance to explore such an interesting field of research and providing help and advice whenever I needed it. I also want to thank my reviewers Prof. Lensch and Prof. Butz for showing so much interest in my work. Together, all my advisors did a great job in guiding me and helping me to focus on the important things whenever I was about to get lost in all the details and interesting problems of attention.

A big thanks also goes to all the people that risked considerable boredom, sore eyes and a stiff neck to provide me with data about their fixations. Without you, that work would not have been possible!

In addition, I want to thank Jeannette, Daniel, Uli and Felix for proofreading this text. Sorry that it got so long! I am also very grateful for everything my friends, my family and especially Felix did to support me, lift my spirit when it seemed like nothing would ever work and to ensure that I had all the time and food that was necessary to finish the thesis in time.

Finally, I also want to thank the whole AMD Lab for their friendliness, helpfulness and their honest interest. I learned a lot from you in the past six months and I greatly enjoyed my time with you!

Contents

1 Introduction **1**
1.1 Visual Attention and Saliency 1
1.2 Saliency Modelling 3
1.2.1 Bottom-Up Saliency 4
1.2.2 Top-Down Saliency 5
1.3 Object Detection in Computer Vision 5
1.3.1 A Short History of Object Detection 6
1.3.2 Object Localisation with CNN 6
1.4 Objective 7

2 Foundations and Related Work **9**
2.1 Convolutional Neural Networks (CNN) 9
2.2 ImageNet Classification with Deep CNN 12
2.3 Deep Gaze 14
2.4 Selective Search 15

3 Methods **17**
3.1 Eye Tracking Data 17
3.1.1 Design of the Dataset 17
3.1.2 Experimental Setup 18
3.2 Learning a Saliency Model 22
3.2.1 Reimplementing Deep Gaze 22
3.2.2 Learning Top-Down Saliency 25
3.2.3 Modifications of the Ground Truth Fixation Data 27
3.3 Region Proposals 29
3.3.1 Pruning Bounding Box Proposals with Saliency 30
3.3.2 Sampling Bounding Boxes with Saliency as Spatial Prior . . 31

4 Tuning the Networks **33**
4.1 A Word on Scores 33
4.2 Training 37
4.3 Experiments with the Reimplemented Deep Gaze Model 37
4.3.1 Comparison of Input Features and Input Image Size 37
4.3.2 Effect of Blurring 40
4.4 Experiments for the Top-Down Saliency Model 42
4.4.1 Comparison of Input Features and Input Image Size 42
4.4.2 Comparison of Input Features for Modified Fixation Maps . 44

5 Evaluation and Results **49**
5.1 Evaluation of Eye Tracking Data 49
5.1.1 Experiments 49
5.1.2 Results 51
5.2 Reimplementation of Deep Gaze 56
5.3 Saliency Map Analysis 58
5.3.1 Results with Modified Fixation Maps 59
5.4 Region Proposals 64
5.4.1 Experiments 64
5.4.2 Results 66

6 Discussion **73**
6.1 Learning Top-Down Saliency from Eye Tracking Data 73
6.2 Discussion of the Model Architecture 76
6.3 Saliency for Region Proposals 79

7 Outlook **81**
7.1 Improving the Saliency Model 81
7.2 Using Saliency for Region Proposals 85

Bibliography **86**

8 Appendix **91**

Chapter 1

Introduction

For humans and many other animals, visual perception is one of the most important senses. We heavily rely on vision whenever we interact with our environment: When we pick up an object, when we move through our environment and avoid bumping into everything on the way or when we recognise our friends by their faces. For all those tasks, object recognition and localisation is essential. In order to pick up a glass, we need to first determine which part of our visual impression corresponds to the glass before we can find out where we have to move our hands in order to grasp it. And if we want to recognise another human, we first have to find out which part of the image we see represents him or her and where in this part the face is.

Of course, we never think about these basic processing steps actively. But what seems so effortless for our brain still poses a major challenge for artificial systems like robots that need to process image content. Existing algorithms most often only tackle a small subset of the different tasks necessary for understanding an image and are very demanding in terms of computational resources and runtime. In order to reproduce at least a part of the human visual perception abilities, one would have to combine several different algorithms. Making such a combined system run in real time with today's hardware is a big challenge. A small step towards this goal is explored in this work by training a neural network model to learn which parts of an image are interesting to human observers that search for a specific object. This knowledge can then be used to speed up object search in computer vision.

In this chapter, first, the concepts of *visual attention* and *saliency* are explained. In Section 1.3, the computer vision task of *object detection* is introduced and the state-of the-art for this discipline is presented. Section 1.4 contains the objective of this thesis and the motivation for the presented approach.

1.1 Visual Attention and Saliency

Human vision and visual perception appear extremely fast and precise despite the fact that it receives constant input from about 120 million receptors and the retina transmits approximately 1.1 $\frac{\text{MB}}{\text{s}}$ to the visual system of the brain [1]. This is only possible because we have means of ignoring a big part of this information and can focus our attention only on the part of the scene that is currently relevant. We are usually not conscious of this, as the brain has its ways of filling in the missing parts from memory and expectation. A famous example for how this mechanism makes us miss other things is the experiment that was first conducted by Simons and Chabris [2] in 1999: Subjects are shown a video of two teams passing a basketball around while walking criss-cross through the room. The subjects are asked to count the passes of the team that wears white shirts. After the experiment, most subjects are very surprised when they are shown that a man dressed as a gorilla slowly crosses the scene on the video. Being completely occupied with tracking the movements of people in white clothes, the dark gorilla becomes invisible to them.

The mechanism responsible for our selective perception is called visual attention. While research is far from fully understanding how attention works, scientists agree on it being a selective process that distributes the limited computing capacity of the brain [3, p. 5]. A common notion is that groups of neurons in different locations in the visual field compete for capacities. To determine which location should be attended, a certain set of features needs to be computed in parallel at every location. This pre-attentive stage is then followed by further processing of those locations that won the competition. There exist numerous theories about which stages of processing are pre-attentive and which only happen when the stimuli is attended. Two famous examples are the *early* and *late selection* theories: Broadbent proposed the early selection model [4], where no semantic analysis occurs pre-attentively. Many other authors (e.g. Deutsch and Deutsch [5]) instead belief that a certain level of semantic processing is necessary for determining which stimuli to attend. This late selection theory seems more plausible, because it can explain why certain objects like faces draw our attention and others usually do not.

Visual attention as a means of coping with huge amounts of data under constrained computing capacities has not only been studied by biologists: As computer vision is very demanding both in terms of computational resources and time, computer vision researchers have always been interested in reproducing the abilities of humans to quickly detect which regions of an image are important to a task and which are not for their algorithms.

The quality of an image location that corresponds to how likely a human deploys attention to this location, is called *saliency*. As a simple example, a red ball on a green lawn is very salient because it stands out both in colour and in shape. However, saliency is not only governed by “low-level” features like contrast, orientation, shape and colour, but also by higher-level concepts. Faces are an example for such a highly salient higher-level concept: They might not always stand out very much from the background in terms of visual features, but attract a big amount of attention nevertheless (see for example [6]). The cause for this is presumably the huge behavioural importance of interactions among humans. The face plays an important role in these interactions, as we look to the face e.g. for identification of persons or judgement of their emotional state and intentions.

In other cases, an object or a location in an image might become more salient than usual when we look at the scene with a certain goal. If we are for example searching for our key, a small piece of metal is more likely to draw our attention than when we are searching for our wallet. This is because we have a model of the object that we are looking for in mind and automatically look for stimuli that are similar to this model.

But how to measure whether an image location is more or less salient? Here the anatomy of human eyes comes in handy: As humans only see a sharp image with a very small patch of their retina (the so called *fovea*), we need to move our eyes constantly in order to bring the regions we are currently interested in into focus. The direction of the human gaze can thus directly be used as an indicator for attention and therefore saliency. For this reason, the movements of human eyes have been studied since the 19th century. Nowadays, so called eye trackers enable researchers to register the direction of their subject's gaze with up to 2000 Hz and accuracies below $0.5°$. Several datasets with images or videos and the corresponding fixation data exist.

However, while gaze direction is a good indicator for the saliency of an image location, it does not give full information about where a human is currently focusing his or her attention. Humans are also able to deploy attention on areas which they are not currently looking at. This phenomena is called *covert attention* (while attending the point on which the gaze is directed is called *overt attention*). Covert attention is especially necessary if we are attending multiple objects simultaneously or do not want to give clues about our plans by moving our eyes. Findings in psychophysical studies imply that, for tracking tasks, humans are able to attend up to eight targets at the same time under certain conditions [7]. So despite of the fact that it contains only incomplete information, the fixation data from human subjects is frequently used as the ground truth for the saliency of an image.

1.2 Saliency Modelling

Both in neuroscience and in computer vision, scientists have been trying to find algorithms and models that are able to explain and reproduce the mechanisms of saliency and attention. Saliency models are often divided into two groups: The so called *bottom-up* saliency models are concerned with saliency under *free viewing* conditions, i.e. which image parts are salient when we look at an image without any conscious intentions, while *top-down* saliency models deal with the influence of tasks, such as searching for a certain object, on saliency.

1.2.1 Bottom-Up Saliency

One of the most influential models of saliency was published by Itti et al. in 1998 [8] and introduced the term *saliency map*: Following the Feature Integration Theory [9], they extract a set of topological feature maps from the image and combine them into a saliency map. This map contains a measure of saliency for each image location and was used as the basis for a simple attention mechanism. The extracted features are purely low-level: colour, intensity and orientation. Feature maps are computed at different scales and normalised to enhance the maps with few strong maximum peaks while suppressing those with high values almost anywhere. The maps for the three feature types are summed and normalised independently and the three resulting *conspicuity maps* are averaged to form the final saliency map.

Later works began to focus more strongly on the impact of semantic content on bottom-up saliency: As some object classes like faces, pets or text reliably draw human attention, the necessity to include such higher-level concepts in the models soon became apparent. For example, Judd et al [10] added detectors for the horizon line, faces and persons to the low-level features used for saliency prediction.

It is not consistent whether the saliency of objects such as faces are termed bottom-up or top-down saliency: For a face to be salient because of being a face, it first needs to be detected as one and one could argue that such a learned classification of image content is already a top-down influence. On the other hand, face detection is an automatic behaviour that does not need any conscious intention and can therefore be considered bottom-up. For being able to clearly assign eye-tracking experiment conditions to resulting saliency models, the latter interpretation is adopted here: Bottom-up saliency is what can be observed during free viewing experiments and therefore the saliency of an object because of its semantic content is viewed as bottom-up saliency.

As stated in [11], adding explicit, hand-crafted detectors for all object classes that might be relevant to model human free viewing behaviour is not feasible. Instead, they trained a model consisting of three layers of sparse coding and pooling units on 150×150 pixel sized patches that were extracted from image regions with local maxima of human fixations.

The features that were learned this way were then combined to a saliency map with weights optimised by a linear SVM (*support vector machine*). Visualisation of the learned features showed that the neurons of the third layer mostly responded to behaviourally relevant stimuli like text and faces. Similar results were also found when visualising the preferred stimuli of the neurons in the higher layers (3 to 5) of deep convolutional neural networks that had been trained for object classification (see the next section and Section 2.2 for details) [12]. Those findings motivated Kümmerer et al. to train a model that linearly combines features that are computed by such a convolutional network [13]. Their work, called Deep Gaze, will be explained in detail in Section 2.3.

1.2.2 Top-Down Saliency

In order to create saliency maps that differ depending on the current task under which an image is viewed, the special properties of the task (e.g. a model of the search target) need to be known first. A common approach that was first presented in the Guided Search model by Wolfe [14] is to compute multiple different feature maps based on different properties (channels) of the image. Then, one especially characteristic feature map from each channel (i.e. the map for red from the colour channel and the map for horizontal orientation from the orientation channel) is selected to be used for the construction of the overall saliency map. The chosen feature map should be the one that best distinguishes between the target and the distractors in the image.

For computing saliency under a search task, in [15], a representation of object categories is learned during a free viewing phase. For this, feature vectors (i.e. the response of the different features in use to one location) are sampled from several locations across the currently attended object, where attention is modelled using the bottom-up saliency model from [8]. The sampled feature vectors are hierarchically combined into increasingly general object representations. When searching for one of the represented objects, the combination of feature maps to form the saliency map is biased towards the features that are considered most important for the representation. The importance is determined by looking at the mean value of the feature and its variance across the different samples from the learning stage: A feature is important if the mean response is high and the variance of the response is low.

A similar approach is used by the authors of [16]: They calculate different feature maps by applying convolutional filters to the input image. In a second convolution step, the different feature maps are combined into a so called “priority map”, that should ideally have maximum response at the location of the target and minimal response elsewhere. The weights for this convolution are hereby the output of a neural network that receives the task as an input.

1.3 Object Detection in Computer Vision

The ability to identify the objects present in an image or scene is one of the most basic requirements when it comes to interacting with ones environment. While it seems completely effortless with humans and in fact most animals, trying to teach computers to see - and also “understand” what they are seeing - has proven extremely difficult.

The key to understanding visual scenes are three closely related sub-problems. The easiest one will be called *classification* in the following. For *classification*, the one dominant object in a given image should be determined and labelled. The next more demanding task is *object localisation*: In addition to labelling the dominant object, it also needs to be localised in the image, usually by determining a bounding box around the image region that is occupied by the object. The difficulty of this task again increases if not only one but all objects in an image need to be labelled and multiple objects of the same category can appear in one image. This task is called *object detection*. Numerous variations of this tasks exist and the terminology used in this thesis will not always be conform with other sources. In addition, object detection is a big and very active field of research, so the following paragraph will only give a very brief overview about recent advances.

1.3.1 A Short History of Object Detection

A good means of judging how close the computer vision community has come to solving the problems of object detection is to look at the results of challenges like the *PASCAL Visual Object Challenge* (VOC) and later the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC)[17].

Pascal VOC started out in 2005 and was held annually until 2012. Among other tasks, it always featured classification as well as localisation tasks. In 2009, the organisers published a paper that presented the challenge and the contributions of this year in more details [18]. Back then, most of the submissions employed the bag-of-visual-words technique based on hand-crafted features like SIFT [19] and HOG [20], where feature vectors are computed at keypoint locations in the image and then a kind of histogram over the feature vectors is used to classify the image content.

From 2010 on, the ILSVRC was also held annually and became the main benchmark for object detection when the PASCAL VOC ended in 2012. It uses images from the huge ImageNet dataset (more than 14 million images) that include up to 1000 different classes for the classification task and 200 classes for detection.

In 2012, the challenge saw a big improvement in performance, when Krizhevsky et al. [21] entered a deep convolutional neural network (CNN) for the first time. They were able to bring down the top-5 classification error[1] from 25.2 % (for the second best entry) to 15.3 % and the localisation error from 50 % to 34.3 %. Since then, the ILSVRC has been dominated by convolutional neural networks of increasing depth. In 2014, it seems that the classification task has become relatively easy with the winning entry [22] achieving a top-5 error of 6.7%. The localisation error has dropped to 25 % [23] and the best detection mean average precision (mAP) has risen from 22.58 % in 2013 to 43.93 % in 2014.

1.3.2 Object Localisation with CNN

For image classification, the output of the CNN is a 1000-dimensional vector that contains a probability of being the most prominent object for each of the possible classes. If only this one most prominent object should be localised, the network

[1]As it is sometimes ambiguous which of the objects in an image is most prominent, each model is allowed to give five predictions and the prediction counts as correct if any one of them is correct.

can simply be expanded to produce bounding box coordinates (either only one bounding box or one bounding box per possible object class). This was first done in [21] and is still used successfully by recent models like [23].

The big drawback of this approach is that it is not applicable if the number of bounding boxes that need to be predicted (i.e. the number of objects detectable in the scene) is not previously known. But this is precisely the case in object detection tasks. A simple cure would be to apply a network for classification and localisation to every part of the image (at different scales) successively and to accumulate the boxes with the best results along the way.

This sliding window approach is of course extremely expensive due to the huge search space. To reduce the amount of times that the classifier (a CNN) needs to run, it could be applied on a coarser grid, but this has the risk of missing the ideal bounding box in some cases. Ideally, one would like to have an initial set of somehow proposed regions that need to be evaluated. It should be minimal while still containing all ground truth bounding boxes of objects in the image. This approach was successfully introduced by Girshick et al. [24] in their work called R-CNN. To generate the region proposals, they employed the Selective Search algorithm [25] that generates regions based on an hierarchical segmentation approach and is described in more detail in Section 2.4.

Unlike most other authors in this field, the authors of R-CNN also report how much time the computations take: Using a GPU, their algorithm spent about 13 s on computing the region proposals for one image and then extracting the features from the proposed regions. On a CPU, this process took almost one minute per image. Even with improvements through optimization and better hardware, it is clear that this algorithm could not be used as is in a system that needs to operate in real time (and perform other tasks in parallel as well).

1.4 Objective

Motivated by the recent success of convolutional neural networks on many different vision tasks and the good performance of the CNN based saliency model Deep Gaze, the objective of this work is to train a top-down saliency model for the task of searching for (possibly multiple) instances of a target object category. The saliency maps that are produced by the model are meant to be used to focus classification and localisation effort to image locations where the target is likely to be found.

In order to train the model, a dataset containing images with annotated object bounding boxes and fixation data from humans is collected using an eye tracker. The subjects from which the data is recorded had the task of counting the appearances of objects of a certain category in the images. The hypothesis is that fixations made during search are sufficiently distinct across different target categories to allow for learning of task specific search strategies and heuristics. As the goal of the saliency map is not only to highlight the target object's location but also other locations where it *could be expected to be found*, human fixations promise to offer more information than the pure object locations that could have been used instead.

It is demonstrated how the saliency maps could in principle speed up region-proposal based detection algorithms like R-CNN: The task-dependent saliency is used as additional information in the process of computing region proposals for the task of searching for objects of a certain category. In the best case, this should allow a method to be faster than Selective Search while still retaining a comparable performance in terms of generating a reasonably small set of proposals. In contrast to the *object detection* scenario, that was described in Section 1.3, for object search, the target object category is given as part of the task. This means that prior knowledge about the target object can be used for the search. Therefore, the proposed regions should ideally be specific for the target category and not category independent like the ones used in R-CNN.

The appeal of the approach to saliency modelling described in Deep Gaze [13] is that the same features that are computed for calculating saliency (which in turn is used for generating region proposals) can also be used for object classification. However, depending on how far up in the processing chain the features for the saliency computation are extracted, further processing of the features will of course still be necessary for good classification results. Nevertheless, the computational overhead for generating region proposals could be greatly reduced by relying on the same features as for the classification step instead of running an algorithm like Selective Search that is completely independent from the classification framework.

This approach also nicely fits with the theory of attention as a computational bottleneck. As explained in Section 1.1, many models of attention claim that in a first, pre-attentive step, the whole input is processed and different features are calculated in parallel. At some point, the computational cost of parallel processing gets too high, so attention is used to determine which stimuli are processed further. The pre-attentively computed features are thus necessary to decide which region to attend, but they also form the input for consecutive processing stages.

To summarise, the contributions of this work are the following: First, a small dataset of eye tracking data from participants performing a search task is collected. Second, based on the results of a thorough evaluation of the Deep Gaze model, a task-specific saliency model is trained for three different tasks. The saliency maps that are computed by this model are demonstrated to be beneficial for object search as they can help to reduce the amount of regions that have to be processed for object classification by adding task-specific information. In addition, methods based on the saliency maps can save computational overhead compared to other approaches by relying only on features that have to be computed for the classification step anyway.

Chapter 2

Foundations and Related Work

This chapter contains a closer look on basic concepts and related published work that are essential for the work in this thesis. The first section introduces convolutional neural networks (CNN) and explains their relation to models of visual perception in computational neuroscience. In the following, the first CNN that was successfully employed for large scale image classification is presented. Section 2.3 presents the Deep Gaze bottom-up saliency model and finally, Section 2.4 gives details about how region proposals are used for object detection and how they can be generated in general.

2.1 Convolutional Neural Networks (CNN)

Convolutional neural networks (CNN) are named after the mathematical operation convolution: The convolution of a function f with another function g is defined as

$$(f * g)(t) = \int_{-\infty}^{\infty} f(x) \cdot g(t-x) dx \tag{2.1}$$

$$(f * g)(t) = \sum_{x=-\infty}^{\infty} f(x) \cdot g(t-x) \tag{2.2}$$

in the continuous or discrete domain.

Convolution is often encountered in the context of image processing, where f is the intensity of a given pixel and g is a 2-dimensional weighting function that is called *kernel*. g is usually non-zero only for a few values in the close neighbourhood to the central pixel and therefore the sum has to be computed only over those values instead of the whole image. The kernel g is often defined as a small square matrix whose size is called the *kernel size* k.

Depending on the values of the kernel, convolution can be used for several image manipulation operations: If the weights are all positive and for example reflect a Gaussian function with its maximum at the centre, a blurring effect is achieved by convolving the image with the kernel. If on the other hand the kernel's values are set similar to

$$\begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \text{ or } \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

it can be used to detect vertical or horizontal edges. The kernel crudely approximates the local gradients of the intensities (in x or y direction) around the centre pixel. This specific kernel is called *Sobel operator* [26] and forms the basis of many edge detection algorithms.

Very similar kernels are also used in computational neuroscience to model the receptive fields of neurons in the visual system: A simple example is lateral inhibition: Neurons receive excitatory input from the receptor at their location and weaker inhibitory input from the neighbouring receptors. This causes the neurons to respond more strongly in the vicinity of abrupt intensity changes. Figure 2.1 visualises this concept in the 1-dimensional case. The ganglion cells in the human retina perform this kind of operation, which is the reason why two neighbouring tiles of different intensities seem brighter or darker at the edge between them [27]. In 2D, a similar behaviour leads to the so called centre-surround receptive fields: A neuron either reacts to a big intensity on a less intense background or the other way around. These operations were also modelled by Itti and Koch [8] to extract features for their saliency map model in a biologically plausible way (not with convolutions though).

Adopting the principle of convolution to neural networks led to convolutional neural networks. The probably first CNN, the Neocognitron, was introduced by Fukushima in 1980 [28]. In 1998, LeCun et al. created a CNN for character recognition called LeNet-5 [29], which became one of the most famous examples for this technique. Its structure of alternating sub-sampling and convolution layers followed by fully connected layers for the classification of the input into characters is still used (although with modifications) in many of today's CNNs (compare, for example, Section 2.2).

In convolutional neural networks, the convolution is 3-dimensional. It is carried out across C input images (channels) and outputs N output channels. At the first layer, the number of input channels is usually either one (for greyscale images) or three (for RGB colour images).

For the nth output channel, the definition of the convolution is

$$(I * w)(n, x, y) = \sum_{c=0}^{C} \sum_{p=-\frac{k}{2}}^{\frac{k}{2}} \sum_{q=-\frac{k}{2}}^{\frac{k}{2}} I(c, x+p, y+q) \cdot w(c, n, p, q) \tag{2.3}$$

Here, $I(c, x, y)$ is the pixel at location (x, y) in channel c. k is the kernel size,

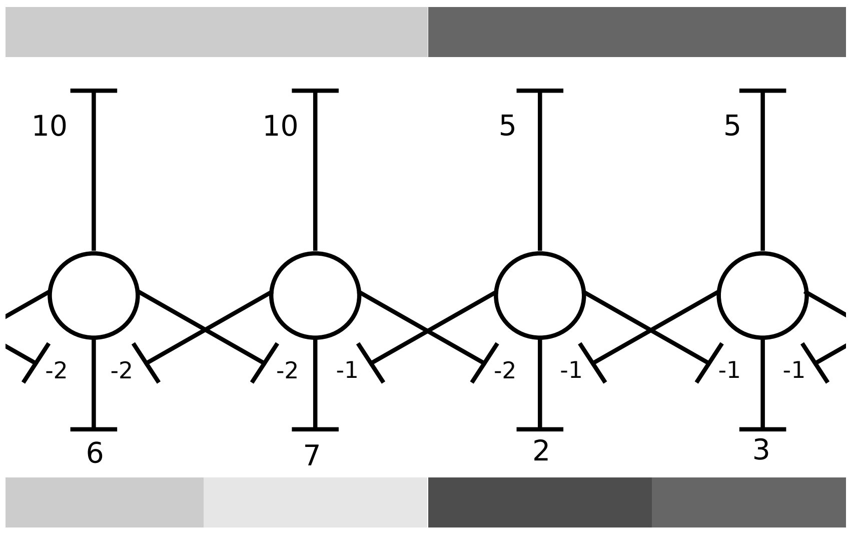


Figure 2.1: A simple model of lateral inhibition in the retina: While the bar on top shows the real intensities of the stimulus, the bottom one shows the perceived brightness. The neurons receive excitatory input relative to the intensity of the stimulus at their location. Their response is the sum of input minus the inhibitory input from their neighbours. As this inhibition is also dependent on the stimulus intensity, neurons in darker locations inhibit their neighbours less strongly. Thus, of all neurons on the brighter side, the one directly at the border to the dark region is inhibited least strongly its response is highest. The same principle makes the border-neuron on the dark side respond less strongly than its neighbour.

that determines how much of a pixel's neighbourhood is taken into account for the convolution. Finally, $w(c, n, p, q)$ gives the weights of the network.

It should be noted that w only depends on the input and output channel number (c and n) and on the pixel offsets within the kernel (p and q). It is therefore independent of the location of the "base" pixel (x, y). This means that the number of parameters (weights) of a convolution layer is independent of the size of the input images, which is a very desirable property, as it keeps the total number of weights of a CNN relatively small compared to a traditional neural network with the same number of layers.

But despite of their relatively small amount of parameters, for a long time, convolutional neural networks were too expensive to train for images with a higher resolution than the 32×32 pixels used by LeCun et al. or for deeper network architectures (i.e. networks with more than three convolution layers). Recent advances in both hardware (especially GPUs) and efficient implementations of 2-dimensional convolution brought them back into the focus of attention of the research community. Yamins et al. recently showed that a strong correlation exists between the output of deep CNN and the firing of neurons in the human visual cortex (V4 and IT) [30] and thereby again suggested that convolutional neural networks process images in a biologically plausible way.

2.2 ImageNet Classification with Deep CNN

The convolutional neural network that Krizhevsky et al. [21] published in 2012 for image classification and object localisation had a huge impact on the computer vision community. This was not only due to the big improvement in classification performance. It also soon became clear that the convolutional layers of the network learned image features that are applicable for a wide range of vision related tasks like scene recognition and domain adaptation [31]. In the remainder of this work, the network will be called *Krizhevsky network*, but by other authors, it is also frequently referred to as *AlexNet*.

The network consists of five convolutional layers (abbreviated with *conv* in the following). Each convolution layer is followed by a layer of Rectified Linear Units (*relu*-layer), that apply the function $f(x) = \max(x, 0)$ to every pixel of their input. The first and second relu layers are followed by a response normalisation step (*norm*) that encourages some small amount of competition between the different output channels of the convolution layers. After each response normalisation layer as well as after the fifth convolution layer, overlapping max pooling (*pool*) is employed. Maximum pooling means that each pixel in the output is set to the maximum value of a small square region (again called kernel) in the input image. In this case, the kernels have dimension 3×3 and are applied at every second pixel, so that the pooling regions overlap. This results in a reduction of the size of the output images by factor 2 with each pooling layer. The first convolution layer also reduces the image size because the convolution is applied with stride 4, i.e. there is one output neuron for every fourth input pixel. Table 2.1 gives an overview over the layers used for feature extraction.

While the first part of the layers of the network computes increasingly global feature representations of the input image, the last three layers are only for the classification. Their output is a distribution over the 1000 potential object categories from which the dominant object in the image could be taken. Donahue et al. also showed that for classifying one image, more than half of the computational time is spent on those three layers [31].

The training of the network was done using an optimization algorithm called *stochastic gradient descent* (SGD) with minibatches, momentum and weight decay. For each training iteration i, the image is fed through the network in a forward pass and the loss L of the prediction in comparison to the ground truth label is calculated. Then all gradients of the loss with respect to the weights w of the respective layers are calculated in the backward pass. Finally, the weights of the layers are updated according to

layer	kernel size	stride	output channels	output size	receptive field size	receptive field size in % of image area
conv1	11	4	96	56	11	0.24%
relu1						
norm1						
pool1	3	2	96	28	19	0.72%
conv2	5	1	256	28	51	5.18%
relu2						
norm2						
pool2	3	2	256	14	67	8.95%
conv3	3	1	384	14	99	19.53%
relu3						
conv4	3	1	384	14	131	34.2%
relu4						
conv5	3	1	256	14	163	52.95%
relu5						
pool5	3	2	256	7	195	75.78%

Table 2.1: An overview over the feature extraction layers of the Krizhevsky network. All values are in pixels if not stated otherwise. The convolution kernels and therefore also the receptive fields are square, so only one side length is given. The percental size of the receptive field is given for the expected size of the input images of 224×224 pixels. Note that the input images were downscaled to 256×256 pixels before the 224×224 pixel sized patches are extracted.

$$v_{i+1} = m \cdot v_i - \lambda \cdot \alpha \cdot w_i - \alpha \cdot \left\langle \left. \frac{\partial L}{\partial w} \right|_{w_i} \right\rangle_{D_i} \tag{2.4}$$

$$w_{i+1} = w_i + v_{i+1} \tag{2.5}$$

Here, $m \cdot v_i$ is the momentum term ($m = 0.9$ was used) that should smooth the updating. $\lambda \cdot \alpha \cdot w_i$ is the weight decay, which is a regularization term that encourages the algorithm to learn small weights. λ was set to 0.0005. α is the learning rate of the algorithm. It started at 0.01 and was reduced by factor 10 two times during the training. The core part of SGD is the last term $\alpha \cdot \langle \frac{\partial L}{\partial w}|_{w_i} \rangle_{D_i}$. $\frac{\partial L}{\partial w}|_{w_i}$ is the gradient of the loss with respect to the weights. In normal gradient descend, this derivative would be computed over all training examples. As this is not feasible for big training sets, the gradients are approximated using a number of samples from the training set (D_i) instead. The D_i are called minibatches or, shorter, batches.

For training the network, the authors had to use two GPUs in parallel because the network would not have fit into the 3 GB of memory of a single GPU. As their model has 60 million parameters, even with the 1.2 million training images from ImageNet, the model was still very prone to overfitting. To deal with this problem, two measures were taken: The first one was to artificially increase the amount of training data by extracting a number of slightly smaller random patches from the images and also including the horizontal reflections of all patches in the training set. In addition, a technique based on *principal component analysis* (PCA) was used to systematically change the values of the RGB channels of the images.

The second technique that was successfully employed to tackle overfitting is called dropout: At each training iteration, the output of each neuron in the first two fully connected layers was set to zero with a probability of 50 %. This means that a different architecture of the network (with roughly half of the neurons in those two layers) is sampled at every iteration. In this way, the neurons are forced to learn more robust features, as they cannot rely on the other neurons in their layer. For testing, the architectures sampled during training are approximately averaged by simply multiplying each neurons output by 0.5.

2.3 Deep Gaze

One classical approach to saliency computation is to linearly combine multiple feature maps of an image into one saliency map. While the feature maps in [8] were calculated using hand crafted features, the authors of Deep Gaze [13] use the feature representation learned by the fully trained Krizhevsky network that was presented in the previous section. They did tests on the output of all layers detailed in Table 2.1 but for their end result, they only used the output of the fifth convolution layer (*conv5*).

For training, the authors used images and fixation maps from the MIT data set [10]. They rejected all images that did not have the desired dimension of 1024×768 pixels and used roughly half of the remaining images for training.

For each image from their training set, the 256 output channels from the conv5 layer of the Krizhevsky network are used as an input for the saliency map computation. The input is linearly combined into one saliency map with weights learned from the free viewing eye tracker data. The output of this combination is then blurred with a Gaussian convolution kernel and a constant term representing the centre bias is added. The centre bias describes the tendency of humans to look at the centre of an image more frequently when free-viewing images. Finally, the softmax of the output is computed to normalise the saliency values in such a way that they form a proper distribution over the image pixels:

$$p(x, y) = \frac{\exp\left(I(x, y)\right)}{\sum_{x', y'} \exp\left(I(x', y')\right)} \tag{2.6}$$

Again, $I(x, y)$ denotes the value of the pixel at location (x, y).

The loss function that is minimised during the learning is is

$$-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i, y_i) + \lambda\frac{\| w \|_1}{\| w \|_2} \tag{2.7}$$

where N is the number of fixations on the image and $p(x_i, y_i)$ is the output of the softmax function at the pixel coordinates of fixation i. $\lambda\frac{\|w\|_1}{\|w\|_2}$ is a regularization term on the weights of the linear combination.

Using this method, the authors could outperform many other saliency models listed in the MIT saliency benchmark set [32] [33] in terms of AUC and sAUC score, as well as in terms of the Information Gain score from [34] (not included in the benchmark). An explanation of AUC and sAUC can be found in Section 4.1. They concluded that the features learned by the later convolution layers of the Krizhevsky network reflect high level concepts like faces and therefore are a good basis for learning of bottom-up saliency, which is largely driven by the behavioural relevance of stimuli.

2.4 Selective Search

As explained in Section 1.3.2, localisation and classification of multiple (different) objects in one image is not easily done with a single application of one CNN. In theory, one would instead need to classify every object in the image separately by running the CNN on a subregion of the image that contains only the current object. Then this region could either be used directly as the bounding box that localises the object or further refinement could be done to improve the bounding box prediction.

The big challenge is how to find the right regions for all objects in the image without first detecting the objects. As simply testing all possible regions is not efficient, instead a relatively small set of best guesses for the regions that should be classified is used. In the following, the Selective Search algorithm [25] for generating such *region proposals* is presented. It was used in state-of-the-art algorithms like R-CNN and GoogLeNet [22].

Selective Search is a hierarchical approach that tries to determine all possible object locations by segmenting the image in different ways. It can output either segmentation masks or bounding boxes for the segmented regions. As only bounding boxes are relevant for this work, the terms *region* and *bounding box* are used interchangeably from now on.

The algorithm starts with a set of rather small regions that should not span multiple objects. Those initial bounding boxes are repetitively merged and the new boxes are added to the set. The order in which the regions are merged is determined greedily: For all pairs of neighbouring regions, a similarity value is computed and the pair with the highest similarity is merged. All similarity values for the two regions that were merged are removed from the set and the similarities of the new bounding box and its neighbours are computed. The algorithm runs until only one region that spans the whole image is left. By not rejecting any of the bounding boxes that get generated along the way, the algorithm has a good chance to find suitable bounding boxes even for composite objects that consist of very different looking parts.

The strength of Selective Search is that this hierarchical merging is not run only once but multiple times, with different measures of similarity, different initial regions and in different colour spaces. The rational behind using different measures of similarity is the following: In order to determine whether two image patches belong to the same object, different features need to be regarded depending on the object. Texture for example will not help to tell an orange from a lemon, but colour will. In addition to colour and texture, the authors use a criterion that encourages smaller regions to merge earlier and a criterion that helps to merge two regions if one is contained in the other. At each run of hierarchical grouping, a different combination of those four criteria is used as similarity measure.

Finally, all regions are combined and loosely sorted by the order in which they were generated, starting with the region from every run of merging that was generated last. This order is thought to reflect the likelihood of the regions to contain an object. After the sorting, duplicate bounding boxes are removed. This is not done before sorting as it helps to promote bounding boxes that were generated by multiple grouping strategies.

In practice, Selective Search produces about 2000 proposals for one image and achieves a coverage of about 92 % of all ground truth object bounding boxes on the training data for the ILSVCR 2014. However, depending on how many different grouping strategies are used, the algorithm is relatively slow. The way it is currently used in algorithms like R-CNN, it seems like a waste of resources to process the image as far as Selective Search does, only to then compute completely new features for all the regions it proposed.

Chapter 3

Methods

The aim of this work was to create a saliency model for object search. As a reference, the bottom-up saliency model Deep Gaze (see Section 2.3) was used. However, in contrast to Deep Gaze, the proposed model should account for the top-down influence of different search tasks.

The presented model is a very basic neural network that uses the output from the Krizhevsky network (Section 2.2) and learns to combine these feature maps into saliency maps that reflect the search behaviour of human test subjects. As an application for the saliency data generated by the model, it is demonstrated how the saliency maps could be used in the generation of bounding box proposals for object search tasks.

Section 3.1 describes how the training data for the model was acquired. In Section 3.2, the model and its training are described. Section 3.3 explains how the saliency maps are used for either sampling region proposals or pruning the proposals from another algorithm.

3.1 Eye Tracking Data

To train a neural network for task dependent saliency prediction, a dataset of images with recorded gaze data had to be acquired first. Although there are a few datasets that include eye tracking data from search tasks freely available, none met the exact requirements described in the following paragraph. Therefore, a set of images had to be chosen and viewed by participants while their gaze direction was recorded.

3.1.1 Design of the Dataset

There were several requirements for the images that would be used as stimuli for the data collection. First of all, for the sake of easy evaluation, the images had to meet some technical requirements: The dataset should contain annotations on the images, listing the depicted objects and their positions in the image. This position information could either come in the form of bounding boxes or, more accurately, as segmentation masks. To allow for an uncomplicated comparison of fixated locations, the images should ideally have the same size or at least the same aspect ratio. Also, the images should not be too small (on average at least 400 pixels in the smaller dimension).

Those first requirements left a couple of freely available datasets of annotated images to chose from. In the end, Microsoft's COCO (common objects in context) dataset [35] was chosen, as the content of the images seemed most appropriate to the task: The COCO dataset was collected with the intention of depicting objects in their natural environments, as opposed to other image datasets (like ImageNet), that contain a lot of images that only show a single object centred on a relatively plain background. Avoiding this so called *photographer bias* and showing natural scenes with rich context is hoped to enable learning algorithms to gather contextual information about objects, like where a certain object is likely to be found and which other objects often appear in its close proximity. It should also keep learning algorithms from overfitting to simply focus on the centre of an image.

Two subsets of images were taken from the validation set of COCO that was released in 2014: one for training the model (again split into training, validation and testset) and one for evaluating the benefit of the generated saliency maps for the generation of bounding box proposals (see Section 3.3). Fixation data was only recorded for the training image set. Some statistical information about the subsets can be found in Table 3.1. The two datasets were completely disjoint. Note that the set for training contained two additional object categories that are not included in the statistics (umbrellas and mobiles) to make the images more diverse and allow for later expansion of the fixation dataset. This reduces the average number of targets and different target categories per image compared to the evaluation set as umbrellas and mobiles were not considered for the computation of this values.

3.1.2 Experimental Setup

The Hardware

For the data recording, an eye tracker by *The Eye Tribe* [36] was used at a publishing rate of 60 Hz. The device uses infra-red projections and a camera to detect corneal reflections relative to the centre of the pupil. Both eyes are tracked and the gaze coordinates are averaged between them. The producer advertises accuracy values between $0.5°$ and $1°$ at a spacial resolution of $0.1°$. The accuracy reported by the device after calibration was frequently observed to be lower (i.e. better) than the advertised value of $0.5°$.

The images were presented on a 22' LG Flatron W2261VP flat screen which was placed at 65 cm distance from the subject. Participants were required to use a chin rest for keeping their head as stable as possible during the experiment. The

	Training Set	Evaluation Set
average number of target instances	0.938	1.049
average number of different object categories	0.743	1.086
average scaling factor for presentation	1.687	1.687

(a) Image statistics

	clocks	laptops	beds
average bounding box size	5.1 %	22.27 %	52.96 %
minimum bounding box size	0.007 %	0.1 %	1.03 %
maximum bounding box size	85.1 %	93.83 %	100 %
average segmentation area	4.02 %	14 %	32.43 %
average number of instances per image	0.275	0.408	0.255
number of images containing the target	94	110	93

(b) Target statistics of the training set

	clocks	laptops	beds
average bounding box size	4.17 %	22.65 %	50.98 %
minimum bounding box size	0.012 %	0.05 %	0.2 %
maximum bounding box size	98.04 %	99.48 %	100 %
average segmentation area	3.18 %	14.39 %	32.06 %
average number of instances per image	0.345	0.391	0.313
number of images containing the target	229	243	236

(c) Target statistics of the evaluation set

Table 3.1: Statistical information about the sets of images that were used for training and evaluating the saliency model. All size information is in percent of the total image size.

eye tracking device was mounted on a tripod approximately 15 cm in front of the screen, but the distance between screen and eye tracker had to be varied between different participants to allow for stable tracking of the eyes. This was necessary because the tracking device was found to work best if the angle in which it was pointed upwards to see the participant's eyes was kept as low as possible. Figure 3.1 shows the experimental setup.

The Presentation GUI

A program was written in Java to control image presentation, calibration and eye tracking. Participants are presented with images and asked to count the number of objects of a certain category in the image.

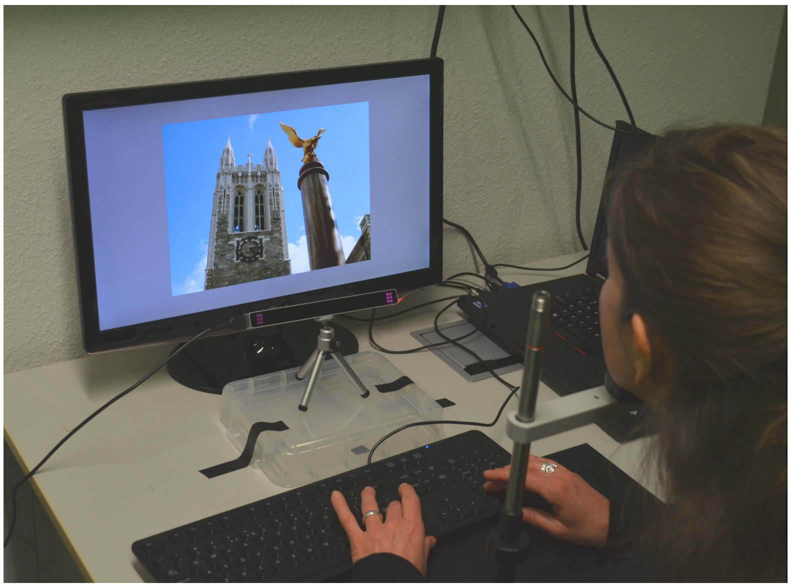


Figure 3.1: A photograph of the experimental setup used for recording the fixation data.

Upon start up, the GUI first informs participants about the task and explains the controls. The user has to press Enter to continue from each stage of the program to the next. The first stage is the calibration of the tracker: The participant is asked to fixate and track a point that appears at twelve different locations on the screen. From these locations and the recorded gaze data, the mapping from data to screen coordinates is calculated by the device (being a commercial product, the method of this calculation is not public). If the overall error as well as the maximum error for a single point are not too high, the calibration is accepted and the user can continue. Otherwise, the user is asked to repeat the calibration.

In the next stage, the error between calculated and intended gaze coordinates is assessed again, this time in terms of on pixel distance. This value is necessary for the later processing of the fixation data (see Section 3.2.2) and the procedure can be used to validate the calibration quality reported by the device.

Like for the calibration, the user is asked to fixate points on known locations. Then the measured coordinates of all fixations are clustered using k-means into as many clusters as there were points to fixate. The distances between the cluster centres (their average location) and the closest target point are calculated and the average of those distances is recorded for later use. If the average error is too high, the user is asked to calibrate the device again and the process of calibration and error assessment is repeated.

If the device is calibrated accurately enough, the images are presented. Each image is upscaled to 1024×768 pixels and displayed in the centre of the screen on a gray background. Each time a new image is shown, a new logfile for the fixation data is created, where the accuracy of the calibration (averaged and for both eyes separately), the pixel error from the accuracy check, the factor by which the image was upscaled and the gaze coordinates are logged. For each gaze data point, a timestamp, the raw and smoothed coordinates (averaged and for both eyes separately) and the classification into fixation and non-fixation are reported. The images are presented for a maximum of 10 seconds each. The user can abort the presentation by pressing Enter, but has to view the image for at least 1.5 seconds. After each image, a screen appears that asks the user to enter his or her count of the target objects in the image. To give feedback, when the user has pressed enter, the background colour of the screen changes from gray to green, yellow, orange or red, depending on how accurate the count was. For each batch of 40 images, another logfile records the users counts, the correct number of targets and the time for which the images were viewed.

Before starting a new batch, the calibration quality is assessed again. Also, the user is given the opportunity to issue a new calibration him- or herself before the accuracy test. This might be a good idea if the user changed position between the batches and therefore already knows that the calibration quality will be found too low.

Data Recording

Participants were shown a total of 410 images divided into ten regular batches of 40 images each and a smaller training batch to start with. The dataset is composed of images that contain at least one of the following object instances: Laptops, clocks, umbrellas and sunshades, mobile phones and beds (see Table 3.1 for details). In the end, three of those categories, namely clocks, laptops and beds were used as targets for the search task. 15 subjects (five female, ten male) who signed a consent form, participated in the experiment. They were aged between 24 and 57. One participant used contact lenses during the experiment and three wore glasses. Each object search task was completed by five subjects, whereby each subject only completed one task to avoid learning effects due to seeing the images multiple times and loss of interest.

The experiment was carried out in a separate room under as constant lighting conditions as possible. Also, the presented images were modified in order to keep brightness changes between the images low. Keeping the brightness changes small during the whole experiment is desirable, because changes in the subject's pupil size between calibration and stimulus presentation were found to affect the eye tracker's accuracy [37].

The adjustment of the overall image brightness was done by converting all images to the CIE L*a*b* colour space, in which the L-component denotes perceived brightness and is independent of the colour defining components a and b. L ranges from 0 to 100, where 0 is black and 100 is white. The average value of L across all images and all pixels was 44.8; the desired brightness was set to 42. Then the brightness of the images was adjusted by the following equations, where $L_{x,y}$ denotes the value of the L-component at pixel (x, y):

$$\begin{aligned}
L_I &= \frac{\sum_{x,y} L_{x,y}}{size(I)} && (3.1)\\
d &= L_I - 42 && (3.2)\\
\alpha_{x,y} &= \begin{cases} \tanh(0.075 \cdot L_{x,y}) & \text{if } L_{x,y} < 42 \\ \tanh(-0.1 \cdot L_{x,y} + 10) & \text{else} \end{cases} && (3.3)\\
L'_{x,y} &= \begin{cases} L_{x,y} - \min(d \cdot \alpha, L_{x,y}) & \text{if } d > 0 \\ L_{x,y} + \min(-d \cdot \alpha, 100 - L_{x,y}) & \text{else} \end{cases} && (3.4)
\end{aligned}$$

The factor α is basically a slightly smoothed step function with values close to 1 between 0 and 100 and 0 otherwise. It drops less steep for values close to 0 in order to avoid setting too many pixels to 0 in the process of the adjustment. All background colours in the image presentation GUI were set to have $L = 42$, too.

3.2 Learning a Saliency Model

Having acquired the dataset of human fixations for search tasks, the next step was to train a top-down saliency model on those examples. In the best case, the model should identify search strategies that are employed by human viewers and learn to apply them to previously unseen images. Again, the goal was not to only find the real target location, but to distinguish between regions that are likely to contain the target and regions that are not.

All learning and network implementation was done in *Caffe*, a framework for convolutional neural networks with a focus on large-scale visual processing tasks. It includes most common layer types for those tasks and allows to use CUDA for fast computations on GPUs [38]. The GPU implementation is especially valuable for convolutions, as their calculation takes very long on CPUs. In addition to the C++ and CUDA network implementation, Caffe also features MATLAB and Python interfaces for convenient training, deployment and inspection of the networks.

3.2.1 Reimplementing Deep Gaze

The Deep Gaze architecture (see Section 2.3) was chosen as the basis for the saliency model architecture. As Deep Gaze was implemented in Theano [39], the architecture had to be recreated in Caffe. Caffe includes a fully trained version of the Krizhevsky network, so implementing everything in Caffe makes it possible to train the whole pipeline from the input image over the feature extraction with the Krizhevsky network to the saliency map calculation in the same framework. The resulting model was then trained on the MIT dataset [10] to test whether the author's results could be reproduced.

Implementation in Caffe

As the output of the Krizhevsky network does not change during the training process for the Deep Gaze model, this data was prepared in advance in order to save computation time during experiments. For each training, validation and test image, the output filters of each layer of the Krizhevsky network were calculated and saved together with the associated fixation maps in hdf5 format.

The linear combination of the input feature maps is implemented by a standard Caffe convolution layer with kernel size 1×1 and stride 1. Likewise, the Gaussian blur, that is applied after the combination, is implemented with a convolution layer whose kernel size is dependent on the width of the Gaussian function. The necessary weights were calculated in Python and set manually via Caffe's Python interface before training. As the blurring layer is a mere image manipulation step, its weights were fixed to not take part in the learning process.

The loss function used in Deep Gaze [13] is

$$-\frac{1}{N}\sum_{i=1}^{N}\log p(x_i, y_i) + \lambda\frac{\parallel w \parallel_1}{\parallel w \parallel_2} \tag{3.5}$$

where N is the number of fixations in the image, w represents the weights of the model and $p(x_i, y_i)$ is the output of a softmax function applied to the blurred linear combination of the input feature maps at the pixel coordinates of fixation i. The regularization term λ had not to be modelled explicitly as Caffe takes care of this. Caffe's layer catalogue also already includes a layer for the softmax computation as well as an optimised "softmax loss layer" for this kind of loss functions. Those layers were only meant for 1-dimensional data though, and did not include the normalization over the number of fixations. For this reason, the softmax layer was adapted to support 2-dimensional data and the loss layer was changed to account for the normalization.

However, it was not possible to adapt the gradient function of the loss layer in such a way that it passed all unit tests that compare the calculated gradient against one obtained by finite differencing. This is probably due to numerical inaccuracies that arise in the softmax layer and the fact that the logarithm is not defined at zero (where the minimum float number is used for computations instead). To be on the safe side, a different loss function was used for training instead: The softmax output of the network was compared pixelwise to the softmax of the label data (the fixation maps) by computing the Euclidean Loss.

This loss also has the advantage of explicitly penalizing false positives (i.e. high values of the predicted saliency maps at non-fixated locations) in comparison to the function defined in Equation 3.5 which only looks at fixated pixels. So in the original loss function, false positives only matter in so far as they result in smaller values of the softmax function at other locations of the image. While the calculation of the softmax would not be necessary for the Euclidean Loss, it was kept because it establishes competition between different locations. This is desirable because saliency itself is a competitive quantity, describing how much a region stands out compared to its surroundings.

As the purpose of the network was to capture task dependent fixations, the term for the centre bias, employed by Kuemmerer et al., was not used.

Preprocessing

The Krizhevsky network had been trained on 224×224 pixel sized image patches that are extracted from the downscaled original images from ImageNet. The average size of the original images is about 482×415 pixels for the subset that was used in the 2013 ILSVRC. Values for the 2012 challenge could not be found but are expected to be similar. Using the full resolution of the 1024×768 images from the MIT dataset and the gathered task fixation dataset was therefore not possible: The receptive field of a neuron in the topmost layer (pool 5), that covers about 76% of a 224×224 input image would only cover about 1% of the bigger sized images. This means that the network would only be able to capture very local features of the image. The authors of Deep Gaze downscaled the input images by factor two, which is roughly the same downscaling factor as was used for the images from ImageNet. In this configuration, the maximum receptive field size is about 14% of the total (downscaled) image size. This was also found to work best in experiments described in 4.3.1 and therefore adopted.

Fixations on the MIT dataset are given in form of raw data logfiles, separately for each participant. They were preprocessed in order to exclude non-fixations and invalid points with MATLAB code provided by the authors of the dataset. For each image, the fixations from all participants were combined and saved in image form, where each pixel's value is the the number of fixations on this pixel for all participants. The fixation maps were not blurred as the loss function used by the authors of Deep Gaze assumes that pixels were either fixated or not and the fixation map therefore has discrete integer values.

For calculating the loss, the saliency map produced by the model and the ground truth fixation map have to be of the same size. While Kuemmerer et al. upscaled the input feature maps prior to the computations, in this work, the fixation maps were downscaled to the network's output size instead using nearest neighbour interpolation. Upscaling before doing the computations was no option because the GPU that was used[1] "only" had 6 GB RAM: The data format used by Caffe is float (32 Bit). Therefore, the input data with 256 input channels (as produced by the conv5 layer) alone needs $32\,Bit \cdot 256 \cdot 768 \cdot 1024 = 6,442,450,944\,Bit \approx 805,3\,MB$ per image. Network training was carried out using stochastic gradient descend with minibatches in order to improve the approximation of the gradients used for calculating the weight updates (see Section 4.2 and Section 2.2 for details). This means that a couple of images are loaded into memory and processed at the same time

[1]Nvidia GeForce GTX TITAN Black Edition

and the updating of the weights is only done once per batch. The more images are used for one batch, the better the approximation of the gradient gets. However, the GPU's RAM would not even have sufficed for eight images per batch if full resolution had been used.

After calculating the feature maps for all images in the dataset, the images were divided into training, validation and test set and were normalised to have zero mean and unit standard deviation (pixelwise) across the respective set. The data was saved in hdf5 format together with the corresponding fixation maps and some meta information like source image names, source directory and dataset size. Chapter 4.3 contains detailed information about all experiments done with the Deep Gaze model and their results.

Experiments with Deep Gaze

The Deep Gaze architecture has two main parts that can be varied without changing the overall architecture: The first is which layers of the Krizhevsky Network are used as the input feature maps for the model. The second is how much the original input images are downscaled before the feature maps are calculated. While the authors of Deep Gaze report experiments for comparison of the performance with different feature maps, no results for different input image sizes are included in the paper. So for finding the best possible configuration for the reimplementation of Deep Gaze and for better understanding which factors are important for the success of the model, a range of experiments with different configurations was done. Details on this experiments and their results can be found in Section 4.3.

3.2.2 Learning Top-Down Saliency

In order to change the reimplementation of Deep Gaze from a bottom-up into a top-down model, not much had to be changed: The most important difference is in the training data: Instead of fixations from free viewing, the top-down model gets search task dependent fixations as ground truth data. In addition, the model needs a specification of the object category for which it should search as additional input. Depending on this target category, the model should then learn to use different weights in the linear combination of the feature maps. This specific weights can be viewed as the model's internal representation of the object.

One way to implement such a model would be to represent the target category with a binary code. A sub-part of the model gets this code as input and outputs the corresponding weights that are then used to compute the saliency map. This approach is very close to the one described in [16]. However, implementing such a model in Caffe is not straight forward, as the framework offers no way of setting the weights of one layer with the output of another.

So in order to save time and to also be more flexible with the training schedule, I instead trained a separate model for each target object category. Without the

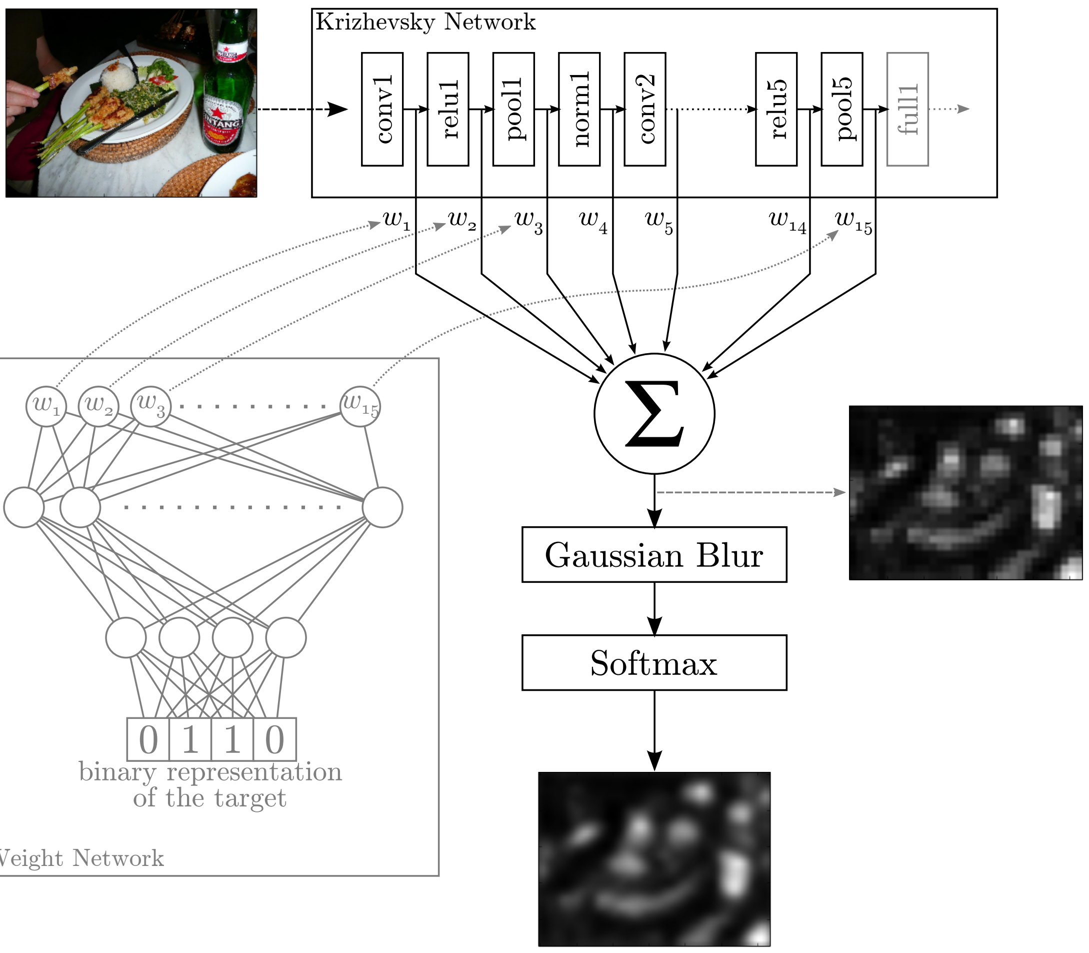


Figure 3.2: Overview over the model's architecture. The separate weight network that gets the task as input and sets the weights of the linear combination of the feature maps accordingly was not actually implemented. Instead, separate models were trained for the three tasks. The only difference to the original Deep Gaze model is that no centre bias is added after the blurring step.

necessity for a task representation as input, the architecture of the separate top-down models is therefore exactly the same as for the bottom-up case. Figure 3.2 visualises the model's architecture and how the network for setting the weights according to the task representation could be integrated.

Preprocessing of Fixation Data

To learn from the humans that participated in the experiment described in Section 3.1, the fixations from the individual participants had to be combined into fixation maps for the different images and tasks first. For every image and participant separately, the fixations were first roughly clustered over time to exclude the first cluster of fixations. This was done because the first fixations on an image usually reflect the centred text and text field from the previous screen of the GUI and therefore do not contain useful information. Afterwards, each fixation was marked in an empty image with a Gaussian with σ of half of the average pixel error measured for the current participant and batch (see Section 3.1.2). This results in a Gaussian function whose width at half of its maximum value is approximately equal to the pixel error. The function is normalised to unit integral to account for different calibration qualities among participants: A high pixel error that was measured before the recording makes the Gaussian around a fixated point wider but at the same time decreases its maximum value. For each image and task, the fixation maps of all users who completed this task were averaged for the final fixation maps.

Experiments with the Model

As done before with the reimplementation of Deep Gaze, again different configurations of input feature maps and input image sizes were evaluated. The previous experiments had already reduced the number of input feature maps to a few promising layers. However, especially concerning the size of the input images, it was not clear whether the best configuration for the MIT free viewing data would also be the best for the new dataset with task dependent fixations. Section 4.4 contains all experiments that were done to find the best combination of input image size and feature maps for the top-down saliency model.

3.2.3 Modifications of the Ground Truth Fixation Data

When analysing the fixation data that was collected during the experiment (see Section 5.1), it turned out that bottom-up effects had a big influence on the fixations: Independent of the task, the participants fixated for example faces and animals quite reliably.

As a result, the fixation data includes a bias towards objects that are behaviourally relevant for the human observers. To make things worse, this bias is independent from the search task, so the fixation maps for the same image do not differ much for different target categories. It seems like the remaining differences between the fixations for the three tasks did not suffice for the model to learn task-specific weights. Visual inspection revealed almost no difference between the saliency maps produced by the models that should search for clocks, laptops or beds. Instead, all three models essentially behaved like bottom-up saliency models (see Section 5.3).

In order to improve the task-specificity of the fixation data and confirm whether the bias in the fixation data really is the cause of the problem, two different methods of modifying the fixation data were evaluated.

Removing the Bottom-up Bias

The first option is to try to remove task-unrelated fixations from the fixation maps. It is of course not possible to precisely determine which fixations are task-related and which are not. A good heuristic is probably to reduce the fixations in areas that were fixated by participants from all three tasks.

Modifying the fixations in this way could be considered bad scientific style: The reason why learning algorithms are used in the first place is that the underlying processes are not known. Removing fixations that are thought to not be relevant for the task has always the risk of excluding too many fixations and thereby losing information that the model could have used otherwise. Or in other words, by assuming that regions that were fixated under all tasks were not relevant for the search task, one introduces one's own assumptions about the process of human search into the data. In this way, the model is constrained to learning a representation of the process that fits the researcher's expectations.

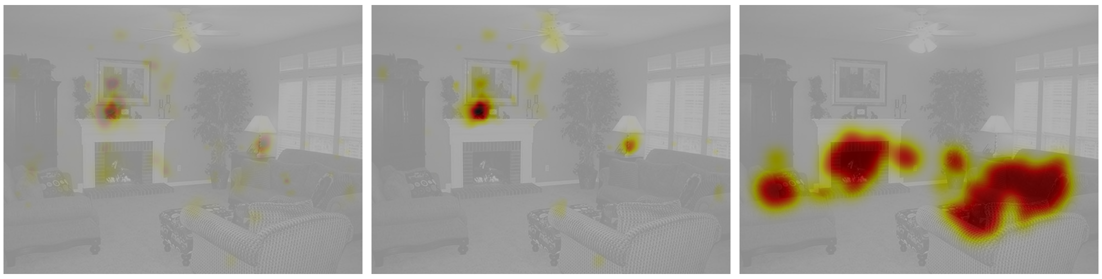
Despite those concerns, a dataset with such modified fixation maps was generated for each tasks and evaluated, as it might also help to confirm whether the many common fixations in the training data really are the main problem. This fixation maps will be called *purified* fixation maps in the remainder of this work.

To determine which fixations have to be removed in each task-specific fixation map (target map), first the fixation maps from the other two tasks are loaded. In a new empty image, the pixels that have values above a certain threshold in both fixation maps are marked. This threshold is currently set to 110 % of the minimum non-zero value in the fixation map, with the idea to avoid excluding regions that were only fixated once by a single participant. The resulting image is blurred with $\sigma = 20$ and scaled to have a maximum value equal to 65 % of the maximum value of the current target map. In the last step, the image is subtracted from the target map and possibly occurring negative values are set to zero again. Figure 3.3 shows the visualisation of original and remaining fixations, as well as the areas where fixations were removed, exemplary for one image and fixation data from the clock search task.

Adding Target Location Information

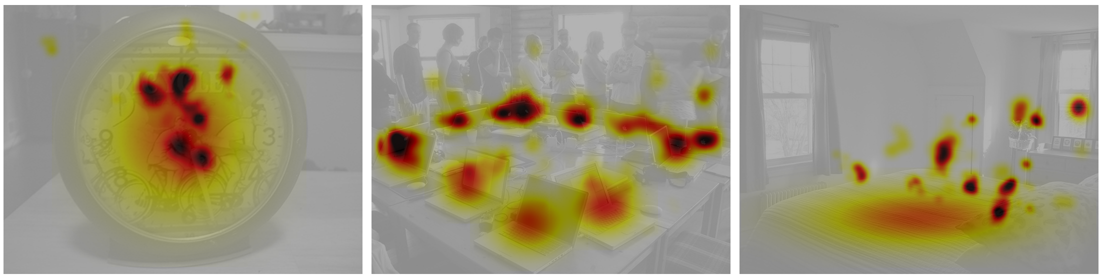
The second possibility to make the ground truth data more task-specific is to add the knowledge about position and form of the target objects. The analysis of the fixation data showed that the targets usually drew no especially high amount of attention, so the actual location of the targets is not emphasised by the fixation data. In addition, fixations are usually only on one point of the target, so they contain even less information about the form and size of the objects.

To add this information, for each target bounding box, a Gaussian function with $\sigma = 8$ is drawn in a 20×20 pixel sized image. This image is then reshaped to fit the bounding box and the values are rescaled such that the maximum of the Gaussian has half of the maximum value of the fixations for the image. Then, the



(a) Original fixation map **(b)** Remaining fixations **(c)** Removed fixations

Figure 3.3: Example visualisation of a fixation map for the clock search task, from which fixations were removed in the locations that were also fixated under both other tasks. As some fixations might be difficult to see, it might help to view this image in the electronic version of the text.



(a) Task: clocks **(b)** Task: laptops **(c)** Task: beds

Figure 3.4: Example visualisations of fixation maps after the locations of the target objects were marked.

image with the Gaussian is added to the fixation map at the target location. Figure 3.4 shows three example fixation maps. The markers for the target location are kept relatively small both in value and in area, to prevent them from dominating the fixation map and thereby rendering the fixations useless.

3.3 Region Proposals

A possible scenario for using the generated saliency maps is for creating region proposals for an object search or detection task. The hypothesis is that the saliency information could help to either reduce the effort for creating the bounding box proposals or to reduce the amount of created proposals. The latter would help as the classification network, that determines if an object is present in the region and to which category it belongs, would need to run fewer times.

Two approaches for this are explored: The saliency maps could be used as an heuristic score to determine which of the bounding boxes proposed by some other method lie on interesting parts of the image, and to filter out the ones that do not. Alternatively, the saliency could be used as a prior over the spatial distribution of the bounding boxes when generating the proposals. The hope is that by using the saliency information, fast but inaccurate methods for generating bounding box proposals could be improved to perform similar to slower methods like Selective Search.

3.3.1 Pruning Bounding Box Proposals with Saliency

For evaluating how useful saliency is for pruning bounding box proposals, first of all, proposals need to be generated. This is done using three different methods: Random sampling, statistical sampling and Selective Search (see Section 2.4). Generating region proposals by random sampling is the easiest method and has least computational overhead. However, it will produce many bounding boxes with sizes and forms that are very unlikely to fit any object (for example very long and thin bounding boxes) and has no guaranty to cover the image evenly.

Statistical sampling aims at generating more reasonable bounding box forms by using statistical information about the target object category. To do so, one first needs to compute the distributions of the aspect ratio and size of the ground truth bounding boxes for the target category. While the location is still sampled randomly, the form and size of the generated boxes are drawn from this distributions. Region proposals that are created in this way are of course not object category independent. As a result, this method is not applicable, if the objects present in the image are not known in advance. On the other hand, if the task is not to detect all objects but to search for instances of a specific object category, the use of statistical information about the target object is a good way to include top-down task information.

To make the two sampling methods comparable, in all cases, the centre of the bounding box as well as its width and height are sampled. For random sampling, if bounding boxes exceed the image's dimensions, they are simply clipped to fit. This of course cannot be done for statistical sampling: If one first determines a scale for the bounding box by using prior knowledge, it does not make much sense to cut of the parts that do not fit in the image afterwards. Instead, the location and aspect ratio are sampled first. Then the maximum possible scale of the box is determined from those two values and the scale is sampled from the distribution of the remaining scales that fit into the image.

Before applying the saliency maps for bounding box pruning, they need to be upscaled to the original image size and normalised to a defined value range. There are different ways to decide whether or not to reject a proposed bounding box based on the saliency map for the image. The easiest options are to calculate sum, maximum or average of the saliency values in the proposed bounding box and apply a threshold.

Summing is difficult, as a threshold would need to be found, that does not always reject small bounding boxes and at the same time is high enough to not let all bounding boxes of a certain minimum size pass. Using the average value on the other hand could put very large bounding boxes at a disadvantage, as the saliency cannot be expected to be spread evenly over an object. For example, for a person, the face will usually be more salient than the feet, but both should be contained in the bounding box. The maximum is agnostic to the bounding box size and therefore probably the best choice. In addition, the low resolution of the saliency maps guarantees that there are no large bounding boxes with only a tiny but high saliency peak. A high maximum value thus usually comes with a reasonably big area of high saliency. Experiments were nevertheless done with all methods (see Section 5.4).

While the softmax computation is good for training the network, using the "softmaxed" version of the saliency maps for pruning bounding boxes is inconvenient: If there are for example multiple target objects in an image, even if the saliency map was perfect (i.e. had maximum values at the targets and zero elsewhere), the absolute saliency values in the bounding boxes would still be lower than for an image containing only one target, as the total saliency has to sum up to one. The maximum of the saliency values is thus dependent on the proportion of the image that is salient. Accordingly, thresholds on the maximum or average saliency in a bounding box would have to be set relative to the maximum saliency value in the image and no single absolute value could be used as a threshold.

3.3.2 Sampling Bounding Boxes with Saliency as Spatial Prior

Instead of first sampling bounding boxes and then rejecting the ones in uninteresting regions, one could also already prefer high saliency regions during the sampling process. As the resolution of the saliency maps is many times lower than the resolution of the images, it gives little information about bounding box sizes. Also, especially big objects might have regions that are more salient than others, so just fitting bounding boxes around highly salient regions will not suffice. Instead, the pixels of the ("softmaxed") saliency map can be used as a distribution from which the centre of the bounding box is sampled. In contrast to the pruning scenario explained above, for this application, the softmax is useful, as it turns the saliency values into a distribution. A distribution generated by the softmax function cannot have zero values. This allows the sampling process to still generate region proposals in non-salient image regions, although with very low probability.

Due to the low resolution of the saliency map, it can only be used to sample a region for the location of the centre pixel (in the current implementation, one pixel of the saliency map corresponds to a 16×16 pixel region in the original image). Within this region, the final centre coordinates are thus chosen randomly. From this point on, one of the sampling methods explained in the above section can be used to determine aspect ratio and scale of the bounding box.

Chapter 4

Tuning the Networks

This chapter describes the experiments that were done to find the best network architecture and parameters for the saliency model as well as their results. Section 4.2 contains general information about the method used for training the models. In Section 4.3, the experiments done on the reimplementation of Deep Gaze are described: First, different variants of the Deep Gaze bottom-up saliency model are evaluated. The best architectures and parameter configurations are then used as a starting point for the experiments on the top-down model, that are presented in Section 4.4. For comparing the performance of the different models, a measure is necessary and thus, this chapter starts with a section about different scores and their advantages and disadvantages.

4.1 A Word on Scores

There exist a wide range of scores for the evaluation of saliency models. The MIT benchmark [32, 33] uses seven different scores, including two different implementations of AUC (area under curve for ROC curves), sAUC (shuffled area under curve), SIM (similarity), EMD (earth mover's distance) and CC (correlation coefficient). A MATLAB implementation of the scores is provided at `https://github.com/cvzoya/saliency/tree/master/code_forMetrics`.

This big variety of different scores suggest a great uncertainty about the correct score. And indeed, a closer look at the scores reveals flaws in most of them: While almost all scores behave good for a good model, some scores also favour models that display a specific kind of error. Other scores give different results for different types of saliency and fixation data (e.g. whether the data is given as a distribution or not) or their score depends on the ratio between fixations and non-fixations in the ground truth data and therefore do not allow for a fair comparison across different datasets and tasks.

Area Under Curve (AUC)

The AUC score expects a saliency model to account for the centre bias. It favours models that predict a high saliency almost everywhere near the centre of the image, while punishing models that give sparser predictions if they are only the tiniest bit inaccurate. A simple implementation of the centre bias alone (a Gaussian distribution with maximum value at the centre of the image and distorted to fit the image's aspect ratio) achieves an AUC score of about 78% on the MIT benchmark dataset [33] and thereby outperforms a good portion of the real saliency models.

This problem is addressed by the sAUC score, which in turn punishes models who include the centre bias, by sampling negative examples (non-fixated points) not randomly but from the distribution of fixations of the other images in the training set. In [33], the authors also found that blurring the saliency maps increased the AUC score about 10 % for most models and up to 16 % for models that produce predictions with extremely sharp borders between salient and non-salient regions.

For this work, AUC was implemented as follows: First, all pixels from the target fixation map are divided into fixations and non-fixations. Since the number of positive and negative examples should be the same for the calculation of AUC, all points from the smaller class are used and an equal number of samples from the other class is chosen randomly. Note that this introduces a small amount of randomness into the score and will result in slightly different scores for the same input every time the score is calculated.

Then, for 100 different thresholds between the minimum and maximum value of the test fixation map, the number of false positive (FP) and true positive (TP) predictions is counted. The AUC is calculated form the true positive and false positive rates ($\frac{TP}{P}$, $\frac{FP}{F}$ with P being the total number of positive examples) at the different thresholds. The negative and positive examples are chosen anew for every value of the threshold.

Similarity (SIM)

The similarity score is a measure for the similarity of two distributions P and Q and is defined as

$$SIM = \sum_{x,y} \min(p(x,y), q(x,y)) \tag{4.1}$$

The SIM score rewards models that output saliency maps that have either very low or very high values in almost all locations. As it is the sum of the minima of the predicted and the target distribution at each pixel, it generates almost perfect scores if one of the compared distributions has lower values than the other in a big proportion of pixels. The implementation used here follows the implementation used for the MIT benchmark set: First, the fixation map is blurred. Then the blurred fixation map and the saliency map are turned into distributions by first rescaling the values to lie in $[0, 1]$ and then dividing by the sum over the whole map to make them sum to one. This approach for turning the data into distributions seems to be more robust against the above mentioned problem than turning them into distributions by taking the softmax[1]. A reason for this could be that it leaves zero values at zero instead of assigning a very small value to them.

Earth mover's distance (EMD)

Earth mover's distance intuitively seems the best score to judge the performance of a saliency model, as it not only captures if the predicted saliency values are correct pixelwise, but also, how close predicted salient locations are to the next fixated location. It is defined as

$$\mathrm{EMD}(P,Q) = \left(\min_{f_{ij}} \sum_{i,j} f_{ij} d_{ij}\right) + \left|\sum_i P_i - \sum_j Q_j\right| \cdot \max_{i,j} d_{ij} \tag{4.2}$$

$$\begin{aligned} \text{s.t. } f_{ij} &\geq 0 \\ \sum_j f_{ij} &\leq P_i \\ \sum_i f_{ij} &\leq Q_j \\ \sum_{i,j} f_{ij} &= \min\left(\sum_i P_i, \sum_j Q_j\right) \end{aligned}$$

where the distributions P and Q can be interpreted as piles of soil on the image. The EMD is then the minimum cost of moving soil around such that one distribution is turned into the other. The cost of moving soil is defined by the amount that is moved from location i to j (f_{ij}) times the distance between the locations (d_{ij}). Unfortunately, EMD is also very hard to calculate for large images because of the global optimisation that cannot be computed in closed form (EMD is usually only used for histogram comparison, where the number of bins is significantly below 100, compared to a full resolution image with a total of 786432 pixels).

Correlation Coefficient (CC)

The correlation coefficient is defined as

$$CC = \frac{\sum_{x,y}(p(x,y) - \bar{p})(q(x,y) - \bar{q}}{\sqrt{(\sum_{x,y}(p(x,y) - \bar{p})^2)(\sum_{x,y}(q(x,y) - \bar{q})^2)}} \tag{4.3}$$

It produces values between -1 and 1, where 0 indicates no correlation, 1 indicates a perfect positive and -1 a perfect negative correlation. It is symmetric and invariant to value ranges and magnitudes. Again, for the implementation, the fixation map was blurred previous to the computation to achieve comparable results with the MIT benchmark.

[1]When using the softmax approach, similarity was constantly at 99 %.

Kullback-Leibler Divergence (KL)

Another score that is sometimes used if ground truth and saliency map are viewed as distributions is the Kullback-Leibler divergence:

$$D_{KL}(P \parallel Q) = \sum_x p(x) \cdot \log_2 \left(\frac{p(x)}{q(x)} \right) \tag{4.4}$$

It measures how much information is lost when the distribution P is approximated by Q. However, for the very sparse ground truth fixation data, the result is largely dominated by the non-fixated locations and results for the comparison of two sparse distributions will be almost perfect, even if the peaks are at completely different locations. Also, it is largely sensitive to the value range of the distributions that are to be compared: a larger value range makes for larger possible values of $\frac{p(x)}{q(x)}$ and can easily change the magnitude of the KL divergence by a factor of 10000. This is especially a problem because the KL divergence is not defined for $q(x) = 0$ if $p(x) \neq 0$ and a common practise is to replace zero values with something close to zero, thereby introducing extremely small values and taking quotients up to very high values. This can for example be done by calculating the softmax of the data in order to turn it into a distribution: as it exponentiates all values, a zero value is turned into $\frac{exp(0)}{\sum_{x,y} \exp(I(x,y))} = \frac{1}{\sum_{x,y} \exp(I(x,y))}$.

Summary

Using the entries on the MIT Saliency Benchmark as samples, the correlation between the different scores can be calculated (using Pearson's correlation coefficient). This shows that all the AUC scores are not correlated to EMD very much. Especially sAUC is not highly correlated to any other score. This again underlines that different scores judge different aspects of the saliency maps and therefore, models should always be compared in terms of several different scores.

For this work, the results of the network training were evaluated using AUC, CC, KL and SIM scores. The KL divergence was later excluded again, because it was found to not be comparable with results by other authors and not add any further information. The InfoGain score developed by the authors of Deep Gaze [34] also could not be used, because not all information necessary to recreate their gold-standard model could be found in the paper at that time. The loss function that was used for training (Euclidean Loss, see 3.2.1) is also a score and was reported, too. However, it includes a regularization term on the weights of the network and is therefore not really comparable between models with a different number of weights.

4.2 Training

All models for the experiments in the following two sections were trained using SGD (*stochastic gradient descend*, see Section 2.2) with momentum 0.9 and a batch size of at least 50 (the exact value depends on the size of the training and validation set). Every 100 iterations, the loss on the validation set was computed. As in [13], the weight λ of the regularization term in the loss function was set to 0.001. The learning rate for the optimisation algorithm can be decreased after a fixed amount of steps, by multiplying it with a constant factor γ. A range of different values for the initial learning rate, the step size for decreasing the learning rate and for γ were tried out for each model. During training, the ordering of input data is constantly shuffled to avoid getting stuck in a cycle of contradicting weight updates.

4.3 Experiments with the Reimplemented Deep Gaze Model

The implementation of the Deep Gaze architecture was studied thoroughly to determine the best possible setup for the later learning task. Experiments were performed with different input feature maps and input image sizes. In addition, the effect of the blurring that is applied to the generated feature maps was quantified. I expect that the results from training on the MIT dataset [10] with free viewing gaze data will also apply to the training of a model for task dependent top-down attention, as the parameters in question (such as input image size) mainly affect the feature extraction and not so much the saliency learning itself. Nevertheless, the two datasets in use (the MIT dataset for bottom-up saliency and part of the COCO dataset [35] for top-down saliency) might be different enough in terms of e.g. average object size, that this assumption needs to be validated.

Every 100 iterations of the training process, a snapshot of the model was saved. The best model in terms of the validation loss as well as every thousandth iteration's model were saved in order to plot the development of scores over the learning process. To better monitor the process, the development of the weights was also plotted. In addition, the output of the network for some random example images was saved as an image for all iteration snapshots.

4.3.1 Comparison of Input Features and Input Image Size

The first experiment was carried out on the most basic setup to determine which layers of the Krizhevsky network provide the best input for the saliency model and which size the input images should have for the feature extraction. For this evaluation, three datasets (each split into training, validation and test set) had to be constructed, that included all output feature maps of the Krizhevsky network [21] for all its layers. For the set *big*, input images were used at full resolution

(1024 × 768 pixels), for *medium* the resolution was halved (512 × 384 pixels) and for *small* halved again (256 × 192 pixels). As the feature maps of the different layers have different sizes, all were rescaled to 64 × 48 pixels, which is the size of the biggest output feature map for the smallest input image size.

The network that is trained only consisted of the input layer that loads the data form the hdf5-files, the linear combination layer, a softmax calculation and the loss layer. The blurring employed in the Deep Gaze paper is expected to bring equal benefit to all layers and input sizes and was therefore left out for this comparison. The effect of the blurring step is evaluated in Section 4.3.2. In the following, a separate model was trained for each combination of input size and input layer. All models were trained with a range of different solver parameters (e.g. learning rate and interval of learning rate adaptation) and the best model in terms of validation loss was chosen for evaluation.

SGD Parameters

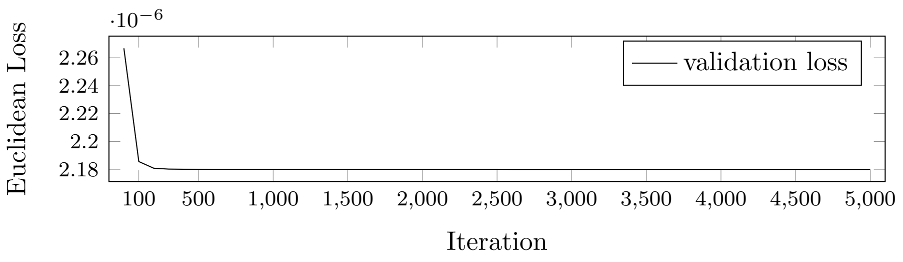
Learning turned out to be very “quick”: In most cases, the model had already converged when the snapshot at iteration 1000 was taken. Learning rates were mostly 1 and learning rate adjustment did not seem to play a big role, as it mostly happened only after the model had already converged. Only for the layers of the first two stages, the selected learning rates were smaller and the convergence took more time. Figure 4.1 shows an example of the plots of the weights, the validation loss and scores over time.

Results

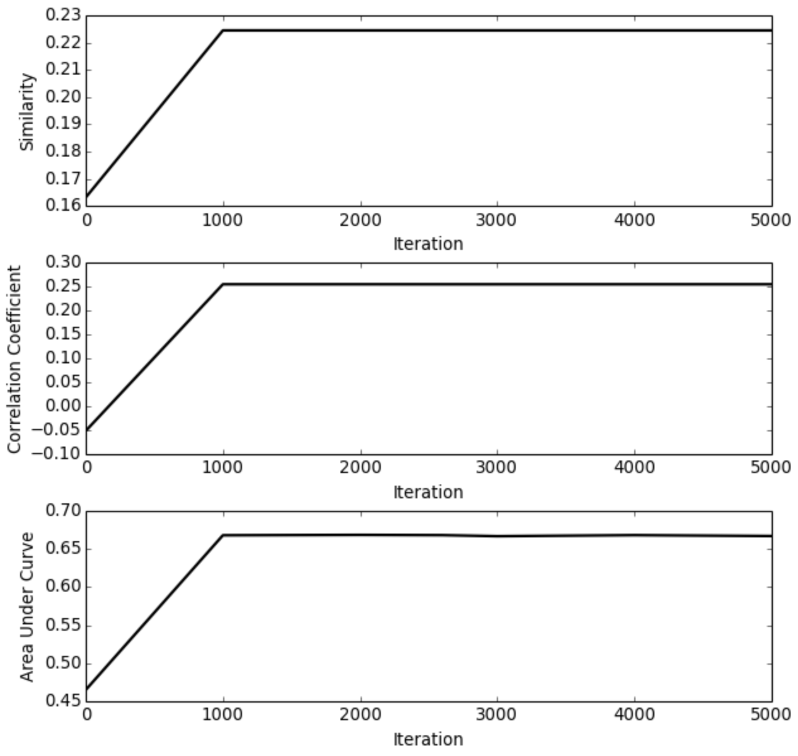
The comparison between the input layers showed that the saliency models that are trained on the features from the first processing stage of the Krizhevsky network (conv1, relu1, norm1 and pool1) perform considerably worse than those who got layers from later stages as input. Figure 4.2 shows plots of the AUC and CC scores achieved by the different layers and input image sizes. The SIM score is not included, but behaves very similarly to the AUC score. A table with all scores can be found in the appendix (8.1).

An interesting behaviour can be observed when looking at the results for the maximum pooling layers, especially for the CC score: At the first pooling layer (where the input image is subsampled by a factor of two), the scores increase for all tested sizes of the input images. At the second pooling stage, the score for the small input images suddenly drops below the other two’s score. And at the last pooling layer, both the medium images’ and the small images’ score drop by a very big amount. The only score that still improves is the one for the biggest input images.

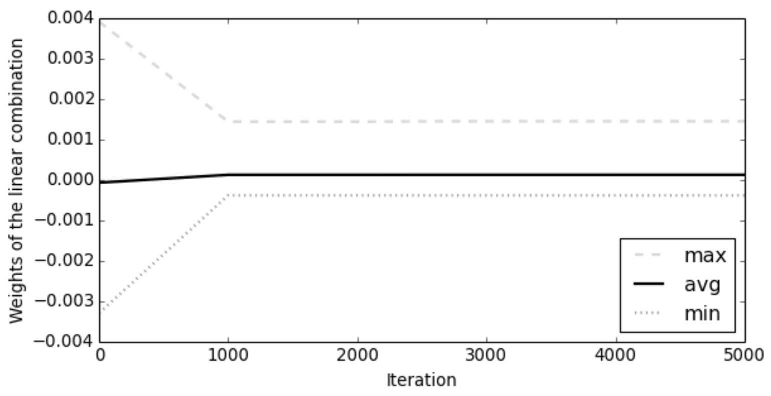
So on the one hand, the quality of the extracted features is increased by the subsampling steps. But on the other hand, the relation between a neuron’s response and the region of the image that caused this response gets weaker with every layer,



(a) Development of the loss on the validation set.



(b) Development of the scores of the model during training.



(c) Development of the weights of the linear combination layer during training.

Figure 4.1: Example of plots that were used to monitor the training process. The plots are from the training on the relu5 feature maps and medium sized images. It can be seen that the model converges very fast.

as the size of the receptive fields of the neurons (when projected to the input image) increases. For pooling layers, this increase in size is even stronger. So at some processing stage, the point where the increase in feature quality cannot compensate for the decrease in spatial information anymore is reached.

Interestingly enough, the results of the layer type comparison differ from the results the authors of Deep Gaze reported: They found an overall better performance of the convolution layers in comparison to the corresponding further processing layers, especially the normalisation and pooling layers. At a closer look on their evaluation, it seems like they averaged over the results of all layers of the different types instead of regarding the layers on the different processing stages separately: This method gives worse results for the normalisation and pooling layers, as they appear only on the first two (and the last, for pooling) processing stages which show an overall worse performance independent from the layer type.

In the end, the layers norm1, norm2, relu3, relu4, relu5 and pool5 were chosen for further experiments. While the layers from the first two parts of the network (norm1 and norm2) do not give very good results in comparison to the other layers, they were kept as representatives of those processing stages for later comparisons nonetheless. The same is true for the pool5 layer, which only gives really good results for full resolution input images.

Concerning the size of the input images, the medium input image size was chosen, as the overall best scores could be achieved with this configuration and the relu5 layer. All in all, it seems to be the best trade-off between increasing the receptive fields of the neurons of the Krizhevsky network in relation to the image size on the one hand and keeping an acceptable resolution for the output feature maps on the other.

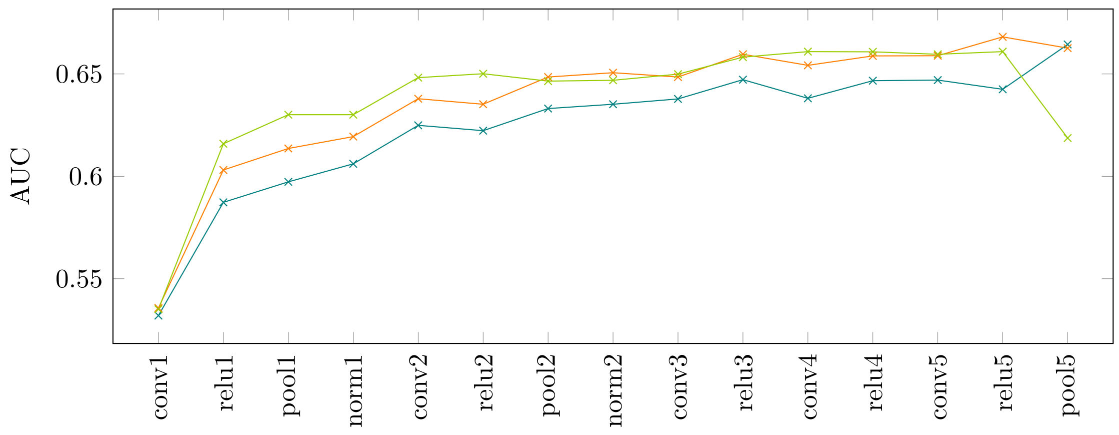
4.3.2 Effect of Blurring

The benefit of applying a Gaussian blur to the linear combination of the input feature maps was evaluated by testing its effect with the selected input layers on the medium image size testset. After initializing a new test network, the weights were adapted according to a Gaussian function with a specific σ. Two different values for σ were evaluated: $\sigma = 0.7$ was found to fit best into a convolution kernel of size 5×5 and $\sigma = 1.1$ was used for a 7×7 kernel.

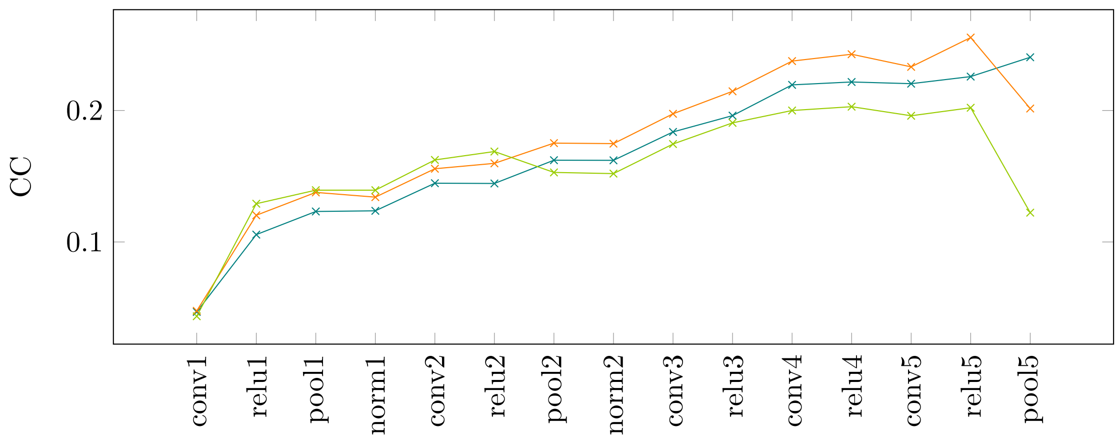
While a stronger blurring (i.e. a bigger σ) might have been even more beneficial, this kernel size was found to be the biggest size for which the backpropagation step of the network training could still be computed without running out of memory (supposedly, this problem is no longer present in more recent versions of caffe).

Results

The blurring was found to increase all scores, but by a smaller amount than expected. The average percental increase can be found in Table 4.1a. Blurring had



(a) Comparison of the AUC scores



(b) Comparison of the CC scores

Figure 4.2: Comparison between the model performance for different layers of the Krizhevsky network as input and different input image sizes.

	AUC	CC	SIM
$\sigma = 0.7$	1.09 %	4.73 %	1.03 %
$\sigma = 1.1$	1.76 %	6.29 %	1.56 %

(a)

	AUC	CC	SIM
no blur	66.81 %	0.256	0.225
$\sigma = 0.7$	67.67 %	0.269	0.228
$\sigma = 1.1$	68.01 %	0.271	0.229

(b)

Table 4.1: The effect of Gaussian blurring with different values for σ. All scores increase with stronger blurring, but only by an overall small amount. Table 4.1a shows the average of the percental increase in scores over the models getting input from the norm1, norm2, relu3, relu4, relu5 and pool5 layers of the Krizhevsky network. In Table 4.1b, the absolute values of the scores achieved by the best model (relu5 input) for the different amounts of blurring are shown.

the strongest effect on the CC score while the AUC and SIM scores only improved by less then 2 %. The absolute values for the scores for the model based on the relu5 layer (which performed best in all scores) can be found in Table 4.1b.

4.4 Experiments for the Top-Down Saliency Model

For the training of a task dependent top-down saliency model, a collection of the best performing configurations found in the previous tests on the Deep Gaze reimplementation was evaluated. As explained in Section 3.2.2, a separate model was trained for each target object category and thus, there are no architectural differences between the networks for the top-down saliency model and the reimplementation of Deep Gaze. However, the images on which the model is trained come from another source (the COCO [35] dataset instead of the MIT data set [10]) and so the experiments with the different input image sizes had to be repeated. To some extend, the different images might also influence the performance of the model with different feature maps from the Krizhevsky network, but it is expected that layers that did perform best for the reimplementation of Deep Gaze will also perform well for the task dependent models.

4.4.1 Comparison of Input Features and Input Image Size

Again, the performance of the model was evaluated with three scales for the input images: big (1024×768 pixel), medium (512×384 pixel) and small (256×192 pixel). While the performance with the different layers from the Krizhevsky network as feature maps also depends on the initial rescaling of the input image, not all layers had to be evaluated again: The results from 4.3.1 show, that relu layers generally perform slightly better than the corresponding convolution layers. As there is also no difference in receptive field size between the two layer types, it suffices to evaluate the relu layers. The layers from the first two processing stages are also unlikely to perform better than the layers from later stages. So again, only the layers norm1, norm2, relu3, relu4, relu5 and pool5 were chosen for the evaluation. In contrast to the initial tests on the Deep Gaze reimplementation, the blurring step with the 7×7 kernel (see Section 4.3.2) was already included in the model architecture for this evaluation.

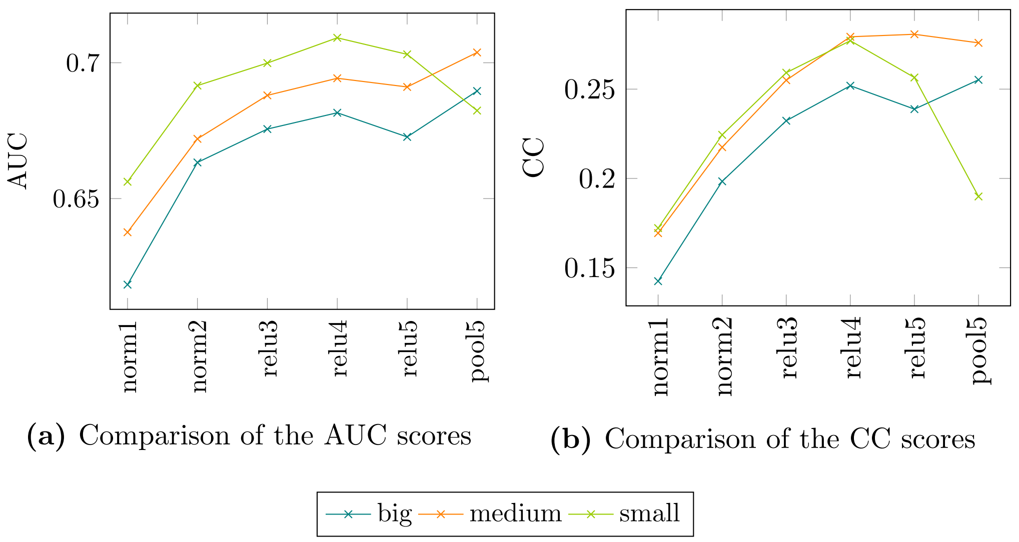
Results

The first result of this comparison is that it was indeed necessary to try out all input sizes again: The overall scores achieved when using the smallest input image scale were in many cases better than those for the medium sized images. Also, the performance with the pool5 layer for the medium sized images improved in comparison to the tests on the MIT dataset. Figure 4.3 shows the AUC and CC scores that were achieved with the different input image sizes exemplary for the model that was trained on the fixations from the laptop search task. For both scores, the performance is best when using small input images and the feature maps from the relu4 layer of the Krizhevsky network.

The observed results are plausible, as the average image size in the COCO dataset is smaller than the 1024×768 pixels of the images from the MIT dataset. For the eye tracking experiment, all used images from COCO were however rescaled to 1024×768 pixels. Because of this, the 512×384 pixel size is actually closest to the original image size, the big images have about double and the small images have about half of the original resolution. As a result, the critical point, where the receptive fields of the neurons in the Krizhevsky network become too big to still give enough spatial information, is only reached after a few more processing steps than for the images from the MIT dataset.

Figure 4.4 shows a comparison of the AUC and CC scores that were achieved by the models trained on fixation data from the different tasks on the smallest input image scale. For comparison, the results of the reimplementation of Deep Gaze (with the same amount of blurring, but on the medium sized images from the MIT dataset) are also reported. It can be seen that the results for the clocks and laptops tasks are much better than for the beds task. Both models achieve higher scores than the reimplementation of Deep Gaze, while the model trained on fixations from the bed task performs worse.

This big difference in the performance of the models is most probably due to the fact that human observers do not need to actively search an image for determining whether there is a bed in it or not. Beds are usually big enough to be noticed on first glance. Because of this, the average recording time was much shorter (between 2.2 seconds and 1.9 seconds, see 5.1.2) than for the other tasks and also than the 3 seconds per image that were used to record the fixation data for the MIT data set. As a result, the ground truth fixation data for the bed search task is sparsest



(a) Comparison of the AUC scores **(b)** Comparison of the CC scores

Figure 4.3: Comparison between the performance of the model trained with fixations from the laptop search task on the different input image sizes.

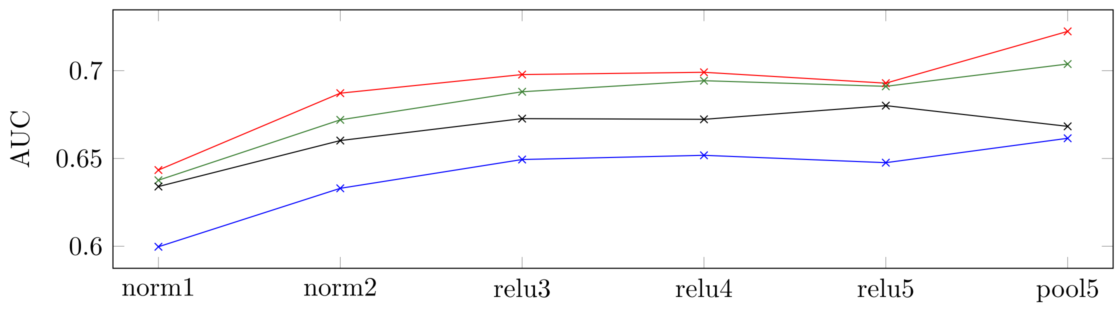
and probably also contains the least usable information.

In the end, one input image size and one set of input feature maps had to be chosen for the model. As in principle, the models for the different tasks should be integrated into one single network that only adjusts the weights of the linear combination depending on the task, it was preferable to use the same setup for all tasks. Table 4.2 shows which combinations of input size and feature maps from the Krizhevsky network achieved the best scores for the different target objects. Only three combinations are present: The AUC score was best for small input images and the relu4 layer for all tasks. For beds, the same holds true for the CC score, while the best CC for clocks was achieved by the medium + pool5 combination. The SIM score was best for the combination of medium sized images and the pool5 layer for both clocks and beds. Only the results from the laptops task diverged from this, with the medium + relu5 combination working best for both SIM and CC.

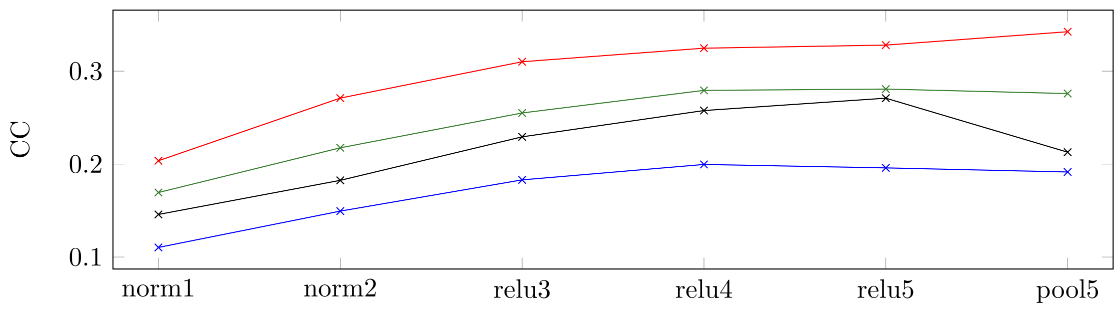
Eventually, the combination of small input images and features from the relu4 layer was chosen as the final model configuration, as it performed best for the most target objects and scores. Using the relu4 instead of the pool5 layer also saves three processing steps during the feature extraction (the last stage of convolution, relu and maximum pooling layers).

4.4.2 Comparison of Input Features for Modified Fixation Maps

As described in Section 3.2.3, two different methods of modifying the ground truth fixation maps (as well as their combination) were tried out: excluding fixations that



(a) Comparison of the AUC scores on small sized images.



(b) Comparison of the CC scores on small sized images.

Figure 4.4: Comparison between the performance of the top-down saliency models for the different tasks. All scores were calculated on 256×192 pixel sized images. For comparison, the scores achieved by the reimplementation of Deep Gaze are also reported. Note that they were calculated on a different dataset (the MIT free viewing data set) and on a different input image size (512×384 pixels), that, however, corresponds to the same average downscaling factor of the images.

	Best AUC			Best CC			Best SIM		
	value	size	layer	value	size	layer	value	size	layer
clocks	73.2 %	small	relu4	0.342	medium	pool5	0.324	medium	pool5
laptops	70.9 %	small	relu4	0.281	medium	relu5	0.258	medium	relu5
beds	66.5 %	small	relu4	0.222	small	relu4	0.22	medium	pool5

Table 4.2: For each task and score, this table shows the best performing combination of input image size and the layer of the Krizhevsky network that was used as input feature map.

are common to the fixation maps for all three tasks (which is termed "*purifying* the fixations" in the remainder of this work) and adding markers for the target objects. While the best size for the input images should not be affected by this, the altered ground truth data could well result in other features from the Krizhevsky network being better suited to model the fixations. To test this, different models were trained for all combinations of task, modification method and input feature maps. Concerning the input image size, the smallest scale was used. The feature maps from the pool5 layer as well as from the norm1 and norm2 layers were excluded as they did not promise good results and only the three remaining layers relu3, relu4 and relu5 were used in the evaluation.

Results

A first look at the results showed that the AUC and CC scores achieved by the models trained on all variants of modified fixation maps were generally worse than the scores with the unmodified fixation maps. An exception are the SIM scores for the models that were trained on the original fixation maps with added target markers.

This decrease in score does not come as a surprise, as both modification methods make it slightly more difficult to recreate the exact pattern of the ground truth: The responses on the different feature maps most often roughly follow the structure of the image. So generating a saliency map where for example one part of a sofa is salient and the rest is not, is hard if all feature maps either respond to the whole sofa or do not respond to it at all. The Gaussian function that is used as a marker for the target objects is also a structure that is very unlikely to be represented in the feature maps and likewise, it is difficult to exactly recreate this structure accurately from them. Nevertheless, the markers pose only a small problem because of their rather small values in comparison to the fixations. Therefore, the scores achieved by the variant with purified fixations were lower than those of the variant with markers. The models that were trained on the combination of both variants performed even worse in terms of AUC and CC. Only the SIM scores of the combination increased in comparison to the variant with purified fixations.

In Figure 4.5, the results of the models trained on the purified fixation maps are shown for the AUC, CC and SIM scores. For the variant with markers on the target locations, all scores but the SIM score of the model for the clock task were highest when using the feature maps from the relu4 layer of the Krizhevsky network. Results with the purified fixations or the combination of both were less consistent: For clocks and laptops as target, the relu4 layer performed best in terms of AUC and CC in both cases. The performance in terms of the SIM score was mostly best with the relu5 layer. Only for the model that was trained for the bed task, the AUC and CC scores achieved with the feature maps from the relu3 layer were highest.

In the end, the relu4 layer was again found to perform best in most cases, so

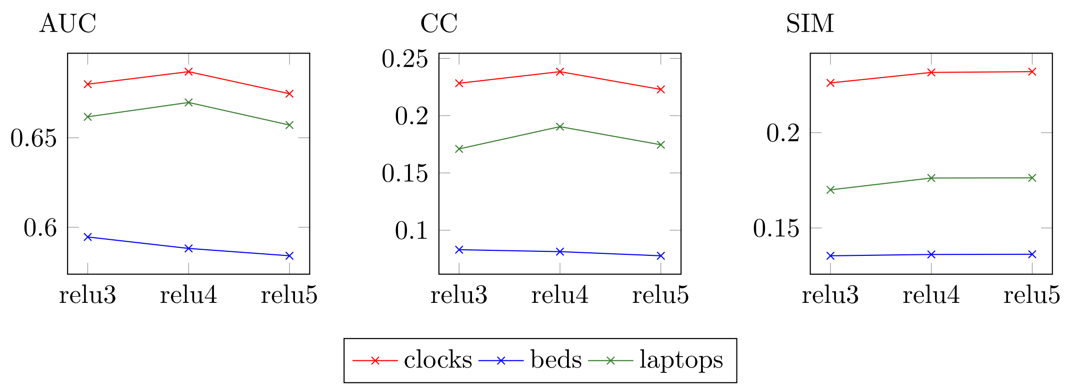


Figure 4.5: Comparison between the performance of the top-down saliency models for the different tasks when trained on the fixation maps from which the fixations in regions that were fixated by participants from all were deleted.

	AUC			CC			SIM		
	clocks	laptops	beds	clocks	laptops	beds	clocks	laptops	beds
original	73.2 %	70.9 %	66.5 %	0.333	0.277	0.222	0.301	0.253	0.211
marker	71.4 %	71.9 %	63.8 %	0.326	0.309	0.202	0.311	0.269	0.267
purified	68.7 %	67 %	58.8 %	0.238	0.19	0.081	0.232	0.176	0.136
both	66.8 %	65.7 %	55.5 %	0.231	0.188	0.076	0.24	0.206	0.205

Table 4.3: The scores achieved by all models that were used for the further evaluations described in Section 5.3 and Section 5.4.

those models were used in the following experiments on region proposals, that are described in Section 5.4. Table 4.3 summarises and compares the scores of all final models.

Chapter 5

Evaluation and Results

This chapter starts with descriptions of the experiments done to evaluate the quality and characteristics of the gathered fixation data and their results. In Section 5.2, the success of the reimplementation of the Deep Gaze bottom-up saliency model is discussed. The saliency maps generated by the different versions of the top-down saliency model are analysed in Section 5.3, before the results of the experiments on generation region proposals with saliency maps are shown in Section 5.4.

5.1 Evaluation of Eye Tracking Data

Multiple evaluations were carried out on the fixation data acquired through the experiment described in Section 3.1.2. They mostly aim at validating the assumption that fixation locations are roughly consistent between subjects for the same task and differ for subjects that view the same image under different tasks. While these findings were documented by Yarbus [40] and were later validated by Mathe and Sminchisescu in [41], this was still necessary to validate the experimental setup (especially given the rather cheap eye tracking device) and for having a kind of benchmark on which to measure the produced saliency maps.

5.1.1 Experiments

All fixation maps used in this evaluation were created in the way described in Section 3.2.2. For visual inspection of the fixation patterns, the fixations were overlaid over a greyscale version of the presented images in two different ways: First, the combined fixation maps of all participants for each task are overlaid over the image separately as a heatmap. In the second variant, the fixation maps for the different tasks are thresholded and then overlaid on one image together, where every task is displayed in another colour channel. This highlights image areas that were viewed by observers from different tasks. Figure 5.1 in the results part of this Section shows two examples for the visualisations.

Agreement of Fixation Patterns Across Subjects

The agreement of fixation patterns from different participants that viewed the same image under the same task was calculated in the following manner: For each participant and each image, the participant's fixation map is compared to the averaged fixation maps from the other participants on this image. For comparison, AUC and CC are used (see Section 4.1 for the definitions). For AUC, the single fixation map is the target, so that the score describes how well the other participants' combined fixations can be used to predict the target fixation map.

Agreement of Fixation Patterns Across Images

As a comparison to the consistency of fixations among subjects, the agreement between fixations of the same participant on different images was measured: For each image and participant, the participant's fixation map is compared to the averaged fixation maps from the same participant on four other images. The number of other images was chosen this small to ensure comparability with the comparison between subjects described above: In [33], the authors found that comparing two fixation maps for the same image from different observers results in increasingly better scores, the more observers are used to create the fixation maps. In their case, the AUC score increases from 86.5 % for two observers to 89.9 % for 39 observers. The other scores they tested (EMD ans SIM) also increased, but more slowly. As every task was completed by five subjects in the above described experiment, the fixations from one participant are always compared to an average of the remaining four other participants.

The hypothesis is that the agreement between fixations for different images should be small. This is because the pure location of a fixation in one image (without any image features) should give no information about the fixated locations in another image. If the images in the COCO dataset [35] are as diverse and free of the photographer bias as the authors suggest, there should be no common geometric structure to the images and therefore fixated locations in one image should not be able to predict the fixations in other images well.

Agreement of Fixation Patterns Across Tasks

To evaluate how different the search behaviour of humans is across different tasks, the combined fixation maps for the different tasks were compared in the same manner as described above. To make results comparable to the results from the cross-subject and cross-image comparison, not all participants' fixation maps were used for the combined fixation maps of the different tasks. Instead, for each task, each participant's fixations were separately compared to the fixations of two randomly drawn participants from each of the two other tasks and then averaged. Like this, the agreement scores are again computed from the comparison of a fixation map from one participant with the combination of four other fixation maps, which is in line with the other evaluations.

It can be expected that the fixation patterns of participants that perform different tasks are less similar than for participants with the same task. However, the agreement should be higher than for the comparison across different images, as the image is the same and some details might draw attention independent from the tasks.

Percentage of Fixations on Targets

To measure the proportion of fixations on the actual targets, the raw fixation points were used and the number of fixations lying in target bounding boxes was counted. As the bounding boxes usually cover more than the actual target, the proportion is slightly overestimated. This should compensate for the underestimation resulting from not blurring the fixation points. In a second analysis, it was evaluated how well the target locations predict the participants fixations. For this, the regions enclosed by the bounding boxes are approximated by a Gaussian function with maximum at the centre of the box, that is resized such that its values that are noticeably different from zero fit the boxes dimensions (thereby slightly underestimating the boxes area).

Influence of Reaction Time and Target Size

The percentage of fixations on the targets as well as the agreement of fixation patterns between subjects were also plotted against the time the subjects viewed a certain image in the presentation, to determine if the reaction time influences this values. The same was done for the proportion of the whole image's area that the target took up (although only for images that include a single instance of the target object). Finally, reaction time was plotted against target area, to validate the assumption that smaller targets make for a more difficult search task. [1]

5.1.2 Results

Visual Inspection

Visual inspection of the overlaid fixation maps from the different tasks showed that for example faces, hands, food, cats and dogs do reliably draw attention irrespective of the task under which the image is viewed.

While the fixations differ for different tasks in most cases and different search patterns and strategies can be observed, the differences are not as pronounced as expected. There are many potential reasons for this: First, as mentioned above, certain objects will draw attention no matter the task. Also, some subjects reported that they were not always able to resist the temptation to look at certain

[1]The plots turned out to not contain much interesting information. They are therefore not included in this document, but can be found on the enclosed CD.

pictures a little bit longer than necessary. Second, the tracker might not always separate fixations and saccades correctly and therefore wrong fixations might show up between two salient objects. And finally, the target objects often show up in similar locations: Laptops tend to be left on beds and clocks will often be found on desks next to laptops or on bedside tables. Therefore, task dependent search behaviour might often result in similar fixations despite of different tasks.

It can also already be observed that the targets of the search tasks do not always draw a peak of fixations. Participants are able to notice very prominent objects without directly fixating them (*covert attention*, see Section 1.1). Noticeably large peaks of fixations on the target are most often only encountered if participants are uncertain about the identity of the object they are looking at or about whether or not it is included in their target category. So instead of sparse peaks on search targets, the fixations are usually spread out widely across regions that are relevant for the search task and regions that are just interesting anyway. Or, to put it the other way around, fixations omit "boring" regions like the sky and plain walls pretty reliably.

Figure 5.1 shows two examples for the visualisations. The first image (Figures 5.1a to 5.1d) is a case where fixation patterns differ perceivably for the different tasks: While subjects looked at the faces of people in the image in all tasks, the wrists were only interesting when searching for clocks. When looking for laptops, the table in the foreground (with a laptop on it) becomes interesting, as well as the screen in the background (which could belong to a laptop). Fixations for the bed task are less specific, mostly because there is clearly no bed in the image.

The second image is an example where the task does not influence the fixations. In all cases, viewers fixated the prominent dog in the centre of the image while mostly not noticing the laptop on the left edge of the image. In fact, all of the five participants that searched for laptops counted zero laptops in this scene.

Counting Task Results

The first quantity that was evaluated was the counting performance of the subjects: If one subject had had noticeably worse scores than the others, the data would have had to be excluded from the evaluation. This was not the case and performance was relatively constant among subjects, with slightly better results in the case that no target was present in the image. The most common source of mistake that was observed during the experiment and also reported by the participants afterwards, were disagreement with the classification of objects in the dataset and incomplete labelling. For example, in some cases, sofas are labelled as beds and often, watches are not labelled at all (presumably because only the wristband is visible). Results are shown in Table 5.1.

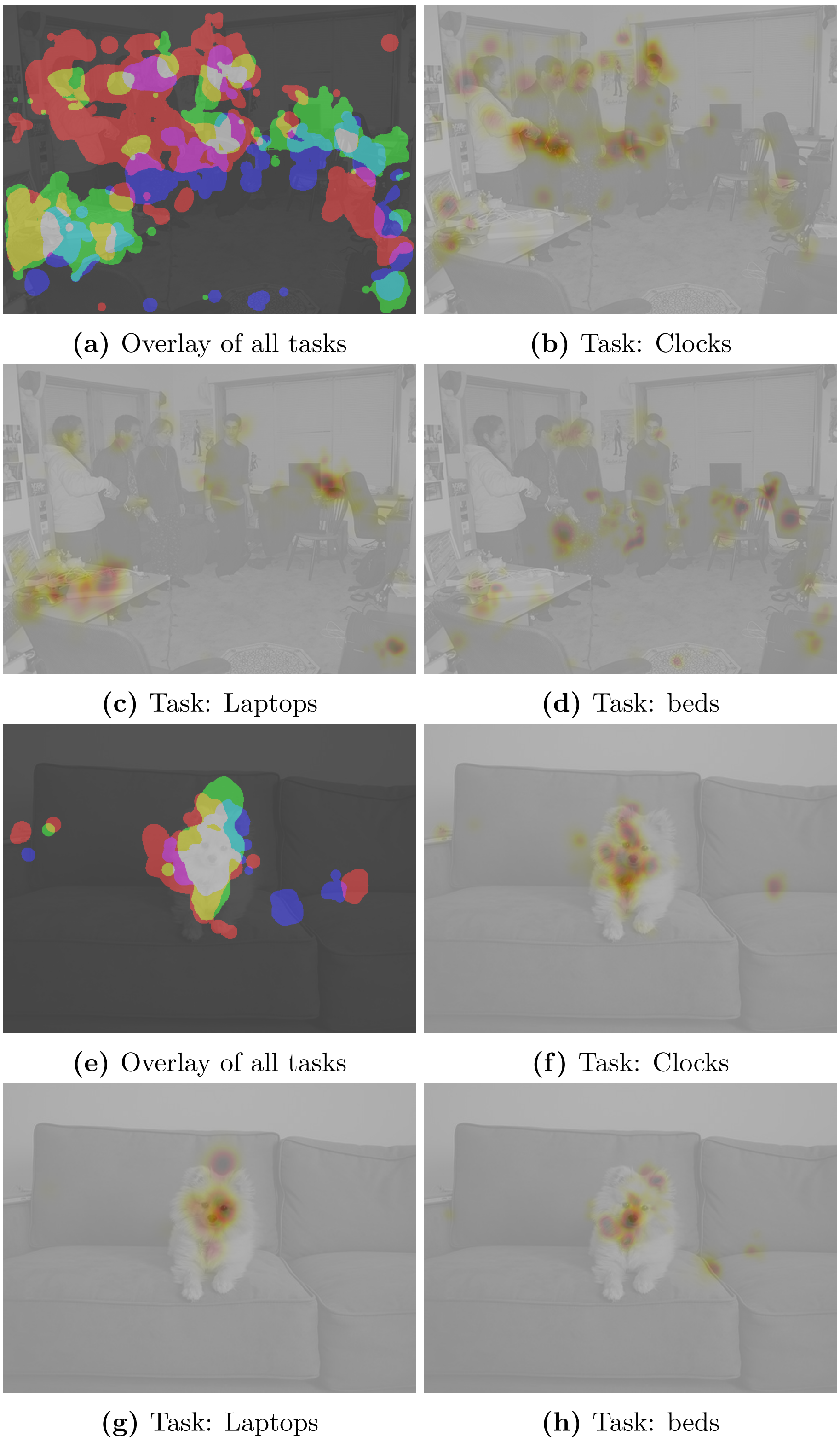


Figure 5.1: Visualisation of fixations on the same image for the different tasks. In (a) and (e), the colour codes for the task that fixations are from: Red is for clocks, blue for beds and green for laptops. Mixed colours indicate areas fixated by viewers from multiple tasks, white areas were fixated under all tasks. In the other images, redder colour indicates a higher amount of fixations at this point.

	average	clocks	laptops	beds
target present	76.08 %	72.94 %	77.68 %	77.6 %
target absent	88.02 %	84.62 %	90.16 %	89.29 %

Table 5.1: Percentage of correct answers averaged over all participants as well as averaged separately for the different tasks.

	average	clocks	laptops	beds
target present	3.089 s	3.884 s	3.199 s	2.183 s
target absent	2.667 s	3.707 s	2.411 s	1.882 s

Table 5.2: Times for which the images were viewed averaged over all participants as well as averaged separately for the different tasks.

Image viewing times

During the eye tracking experiment, the time for which the image was visible was recorded (remember that participants had the possibility to abort the image presentation whenever they felt ready). The average time that participants viewed the images is slightly higher if targets were present in the images. Averaging the times for the different tasks separately shows that participants spent the longest time when searching for clocks, while participants that searched for beds regarded the image not significantly longer than the minimum presentation time of 1.5 seconds. Times are shown in Table 5.2.

Plotting the time for which the images were viewed against the percentage of the image area the target took up showed that images with very small targets were on average viewed slightly longer than images where the target takes up almost the whole image. This was to be expected as small targets are harder to spot and also more difficult to classify.

Agreement of Fixation Patterns Across Participants for the Same Task

Results of the experiment are depicted in Table 5.3. The results are fairly constant between tasks, with slightly worse scores for the bed search task. This could be due to this task being the easiest and having the lowest average reaction times: If the search task could be solved in less than the minimum viewing time of 1.5 seconds, participants had to spent the rest of the 1.5 seconds essentially free viewing the image.

Plotting the agreement of the fixation patterns against the time the images were viewed revealed no significant or systematic differences between different viewing times. The same is true for the influence of the target size on the agreement.

	overall		clocks		laptops		beds	
	AUC	CC	AUC	CC	AUC	CC	AUC	CC
target present	67.18 %	0.21	69.23 %	0.28	69.07 %	0.24	63.23 %	0.12
target absent	67.7 %	0.2	68.82 %	0.23	70.11 %	0.24	64.15 %	0.14

Table 5.3: Agreement of fixation patterns between participants on the same task and image.

overall		clocks		laptops		beds	
AUC	CC	AUC	CC	AUC	CC	AUC	CC
55.7 %	0.039	55.16 %	0.033	56.28 %	0.043	55.87 %	0.041

Table 5.4: Agreement of fixation patterns on different images for the same participant.

Agreement of Fixation Patterns Across Images

As a comparison, the agreement between the fixation patterns of the same participant on different images was calculated (*cross-image control*). Results can be found in Table 5.4. As expected, agreement scores were lower than for the across participant agreement, with the AUC score only slightly above chance (50 %).

Agreement of Fixation Patterns Across Tasks

The agreement values for the fixation patterns on the same image and different tasks can be seen in Table 5.5. It is lower than for the comparison among participants that did the same task, but still higher than the cross-image control. This is in line with the hypothesis formulated above: Fixations on the same image are more similar than those on different images regardless of the task, as certain image elements draw attention from all participants in a bottom-up manner. Still, the agreement between two participants that perform the same task is higher, as the task also guides attention in a top-down manner. However, the difference between the agreement of the fixation patterns with different tasks and with the same task is not very significant, especially not if the target object of the search is not present in the image. The consequences of this for the top-down saliency model and possible ways for improving the situation are presented in Section 5.3.

Fixations on Target Objects

The average percentage of fixations that fell into the bounding boxes of the target objects was surprisingly low with 16.88 % for beds, 11.33 % for laptops and 3.77 % for clocks. The differences of values can be explained by the different object sizes and the inaccuracy of the tracker, which plays a much bigger role for small bounding

	overall		clocks		laptops		beds	
	AUC	CC	AUC	CC	AUC	CC	AUC	CC
target present	62.34 %	0.123	62.84 %	0.151	63.15 %	0.124	61.03 %	0.095
target absent	65.31 %	0.155	62.91 %	0.157	67.6 %	0.18	65.42 %	0.127

Table 5.5: Agreement of fixation patterns between participants viewing the same image under different tasks.

overall		clocks		laptops		beds	
AUC	CC	AUC	CC	AUC	CC	AUC	CC
61.17 %	0.12	55.04 %	0.12	61.55 %	0.14	66.93 %	0.09

Table 5.6: Agreement of fixation patterns of participants with predictions based on the target bounding boxes.

boxes. When using the bounding boxes as a prediction of fixations, the AUC and CC scores were mostly higher than the cross-image control but lower then the agreement between participants on the same image (both for participants doing the same or different tasks). Only for beds, the AUC score is higher, which is probably due to the fact that beds are usually very big and the AUC score tends to reward a bigger amount of positive predictions. When looking at the CC score, the results are however worse, reflecting that the spatial distribution of the fixations is mostly different from the bed locations.

The agreement of the fixations with the region defined by the target objects was of course higher for bigger objects. As observed before for the agreement between participants, the plot of the agreement versus the time that the image was viewed revealed no systematic effect of the presentation time. It can thus be noted that neither the size of the target objects nor the reaction time of the participants influence the fixation patterns in a consistent way.

5.2 Reimplementation of Deep Gaze

Reimplementing Deep Gaze was an important step for this work, as the model forms the basis for the top-down saliency model that was implemented. In order to confirm that the reimplementation was successful, one needs to compare the scores achieved by the two models. There are, however, some problems in doing so:

The first is that the authors of Deep Gaze added a model of the centre bias to their saliency map predictions. The MIT saliency benchmark contains a very simple centre bias model (a Gaussian function that is resized to fit the image's aspect ratio) for comparison, which alone achieves an AUC score of 78 %, a similarity score of 0.39 and a correlation coefficient of 0.38. The Deep Gaze implementation of the centre bias is learned from the combined fixations on the training set and can therefore be expected to perform even better.

At least for the comparison of the AUC score, this problem could be solved by comparing the AUC score of the presented model with the sAUC score of Deep Gaze: This score was designed to punish models that implement the centre bias. Therefore, the authors of Deep Gaze used no centre bias for the computation of the saliency maps that were evaluated using sAUC. The 71.69 % that Deep Gaze achieves for this score are much closer to the AUC score of 68 % that was found for the best configuration of the reimplementation.

However, as sAUC was not implemented for this work, this comparison is not quite fair either. So instead, for the best performing reimplementation (based on input from the relu5-layer of the Krizhevsky network), a simple Gaussian centre bias model was combined with the predicted saliency maps and the scores on the test set were calculated again. The bias was calculated on a 512×384 pixel sized grid according to

$$c(x, y) = \exp\left(-\frac{(0.425x)^2 + (0.575y)^2}{2\sigma^2}\right) \tag{5.1}$$

with $\sigma = 35$. After computation, it was downscaled to 64×48 pixel. For combination with the predicted saliency maps, it was rescaled to have a maximum value of 7.5 % of the maximum value of the saliency map. Then, the softmax of the combination was calculated.

A comparison of the scores can be found in Table 5.7. The first result is that the combination of the presented model with the simple centre bias achieves much higher scores than the model without the bias or the bias without the model. However, the scores, especially the CC and SIM scores, are still worse than those achieved by Deep Gaze.

As a second result, it should be noted that my own model of the centre bias has the same AUC score as the one present in the MIT benchmark set, but its SIM and CC scores are also worse than those of the MIT bias. The amount of difference is hereby almost the same as the difference between the scores of Deep Gaze and the presented model (with centre bias). This observation implies that the difference in scores does not come from a difference in the models but from a difference in how the scores are computed.

A careful reader might have already noticed the second problem for the comparison of the scores: While the results reported for Deep Gaze are for the 300 test images of the MIT saliency benchmark dataset, all results in this work are based on the test split (107 images) of the MIT dataset. Both sets come from the same authors and the fixations were recorded with the same eye tracker. It seems reasonable to assume that the two data sets are not substantially different in terms of content, especially because the authors recommend to use the MIT dataset for training models that should be evaluated on the benchmark set. However, while the MIT

model	AUC	CC	SIM
Deep Gaze	84 %	0.48	0.39
centre bias MIT	78 %	0.39	0.38
centre bias own	78 %	0.29	0.3
relu 5	68 %	0.27	0.23
relu 5 with centre bias	82 %	0.38	0.3

Table 5.7: Scores achieved by Deep Gaze (as reported on the MIT saliency benchmark), the centre bias model included in the MIT saliency benchmark, an own centre bias implementation on the test set and the best reimplementation of Deep Gaze done in this work with and without centre bias.

dataset contains fixations from 15 observers, the benchmark dataset was viewed by 39 observers. As was already discussed in Section 5.1, better prediction scores can generally be achieved by combining fixation data from more observers for the labels. This leads to the assumption that the model presented in this work would have achieved slightly better scores on the MIT benchmark dataset, but it could of course also be the other way around.

Finally, the last problem for comparison is the blurring of the ground truth fixation maps: For the scores reported on the MIT saliency benchmark, the binary fixation maps were blurred with a Gaussian filter with $\sigma = \frac{8}{\sqrt{\log(2)}} \approx 9.6$ in the frequency domain. As the fixation maps used in this work are by factor 16 smaller than the original image size, $\sigma = \frac{9.6}{16} = 0.6$ was used for blurring instead. A quick test showed that doubling σ to 1.2 improved both CC and SIM scores of the model with added centre bias by 0.09 to 0.46 and 0.39 respectively. As AUC is the only score that is not affected by blurring (because the unblurred fixation maps are used for its calculation) and the AUC scores of the original model and the reimplementation are already close, this suggest that differences in the blurring of the fixation maps could be responsible for the worse CC and SIM scores.

5.3 Saliency Map Analysis

For analysing the saliency predictions of the different models, the 859 images of the evaluation set described in Section 3.1.1 were used. It should again be noted that no fixation data was collected for these images and none of them was used in the training or the quantitative evaluation of the models.

A first visual comparison of the saliency maps that were produced by the saliency models for the three different search tasks was sobering: The saliency predictions for the same image are almost identical, independent of the task for which the model was trained. Figure 5.2 shows an example of the saliency maps for one image from the evaluation dataset.

However, while no top-down effects of the task are noticeable, all models show the behaviour of a bottom-up saliency model that is driven mostly by the semantic content of the depicted objects: Humans for example are predicted to be very salient, with highest values of saliency on the faces. The same holds true for animals like cats and dogs.

This suggests that the reason for the missing task specificity lies not in the model or a mistake during the training procedure but in the collected fixation data: The object categories mentioned above are very strong drivers of attention and participants are usually not able to avoid fixating them. The influence of those bottom-up distractors on the data could be so big that the remaining task specific fixations are not plenty enough to keep the model from learning bottom-up saliency. This would also explain the small difference in the agreement of fixation patterns of different participants depending on whether or not they had the same task (see Section 5.1). For testing this hypothesis, different methods of modifying the fixation data were tried out and results are reported in the following section.

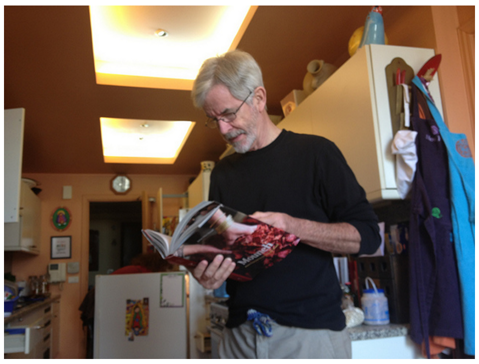
5.3.1 Results with Modified Fixation Maps

As described in Section 3.2.3, two different ways of modifying the fixation maps for the training data were tried out: Purifying the fixation maps and adding markers for the target locations. They reflect two possible causes for the missing task specificity of the model: For once, the fixations on objects that are not related to the task are thought to dominate the fixation data. As a result, the model is not forced to learn anything task dependent, because it gets good enough scores by simply learning bottom-up saliency.

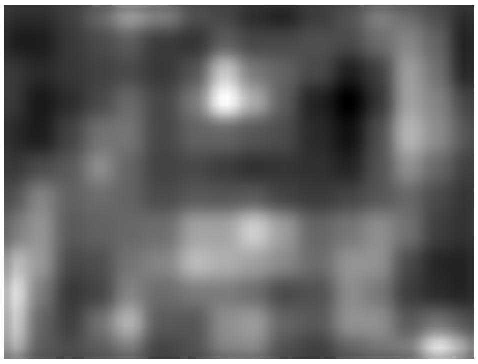
The second possible problem is that the target objects might not be emphasised enough by the fixations. There usually are fixations on the search targets, however, they are not more plentiful than on other interesting parts of the scene and do not reflect the objects shape and size. Thus, the model could probably benefit from more direct information about how the target of the search actually looks like. Such knowledge is of course less important for contextual information like that people usually look to tables to find laptops. On the other hand, search is not only guided by such higher-level considerations but also by simply matching the stimuli with the internal representation of the target object.

For evaluating the impact of the modified training data, the average agreement between the saliency maps that two different models produce for the same task and image was computed in terms of CC and SIM score. The results of this comparison can be found in Table 5.8. It can be seen that the saliency maps for the bed task are the most dissimilar between the models and those for the clock task have a higher agreement than those for laptops.

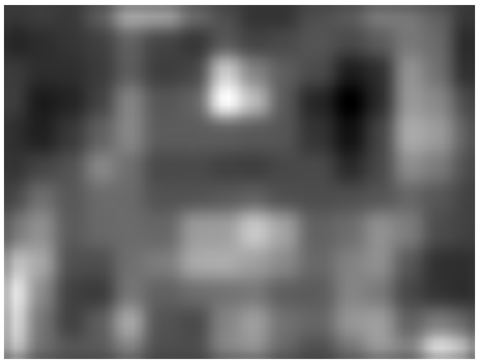
Saliency maps produced by the model trained on fixation maps with markers on the target objects are most similar to those produced by the original version. The



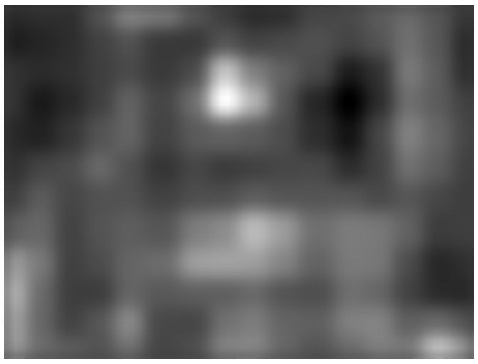
(a) Original image



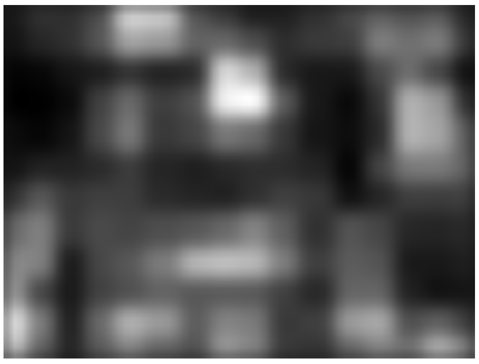
(b) Task: clocks



(c) Task: laptops



(d) Task: beds



(e) Trained on MIT free viewing data

Figure 5.2: Example of the saliency maps produced by the models trained for the different task as well as by the reimplementation of Deep Gaze. The saliency maps are almost identical and mainly emphasise the face and the hands of the person depicted in the original image. In comparison, the clock on the wall in the background has low saliency values, even for the model trained to find possible clock locations. The predictions of the top-down models also do not differ much from the output of the reimplementation of Deep Gaze, despite the fact that the models were trained on different datasets and use different feature maps as input.

	original - marker			original - purified			original - both		
	clocks	laptops	beds	clocks	laptops	beds	clocks	laptops	beds
SIM	0.96	0.94	0.92	0.95	0.95	0.87	0.94	0.9	0.81
CC	0.99	0.97	0.91	0.97	0.97	0.75	0.96	0.88	0.46

	marker - purified			marker - both			purified - both		
	clocks	laptops	beds	clocks	laptops	beds	clocks	laptops	beds
SIM	0.94	0.93	0.88	0.95	0.93	0.86	0.95	0.91	0.85
CC	0.97	0.95	0.74	0.98	0.94	0.67	0.98	0.91	0.55

Table 5.8: This table shows how similar the saliency predictions of the models trained on different variants of modified fixation data are for the three tasks.

	clocks & laptops		clocks & beds		laptops & beds	
	CC	SIM	CC	SIM	CC	SIM
original	0.967	0.942	0.944	0.927	0.957	0.932
markers	0.901	0.9	0.832	0.873	0.847	0.887
purified	0.903	0.907	0.545	0.82	0.648	0.837
both	0.777	0.862	0.301	0.776	0.393	0.807

Table 5.9: Comparison of the similarity of the saliency predictions on the same image for different tasks.

combination of both modification methods results in the biggest differences to the original model. Finally, the resulting saliency maps when using both modifications are slightly more similar to the predictions of the model based on the fixation maps with markers. This seems plausible as for big target objects like beds, the markers are also very big and therefore have a bigger impact on the model than the removal of a few fixations.

In addition, for each variant of fixation map modification, the average agreement between the saliency maps produced by the models that were trained for the different tasks was computed to measure the task specificity of the models. Finally, as baseline for the maximum amount of dissimilarity that can be expected, the average similarity between the saliency maps of different images was approximated by comparing each saliency map to the maps for the following three images. This was done separately for each model and task.

Unmodified Fixation Data

The high similarity of the saliency maps for the different tasks that was found by visual inspection was confirmed by the agreement data: The lowest CC was 0.944

for the comparison between the saliency maps for clocks and those for beds. The SIM score was also lowest for this comparison with a value of 0.927. All results about the agreement between the saliency predictions for the different tasks can be found in Table 5.9.

The average agreement of the saliency maps for different images was judged differently by the CC and SIM scores: While the average similarity score lies between 0.688 for the laptop saliency maps and 0.723 for those from the bed task, the average correlation coefficient ranges from 0.024 for clocks to 0.031 for beds. So the SIM score thus still finds a rather strong similarity between the saliency maps while the correlation coefficient indicates that the maps are not correlated at all. This extreme difference in the scores can at least partially be explained by the higher value range of the CC score: it can take values from -1 to 1 while the SIM score is limited to values between 0 and 1.

Markers for Target Locations

Adding markers for the target locations to the fixation maps slightly decreased the similarity between the saliency predictions for the different tasks, as can be seen in Table 5.9. The baseline agreement between saliency maps for different images on the other hand was a bit higher than for the unmodified fixation data, with a maximum SIM of 0.749 and maximum CC of 0.047 for the bed task.

Purified Fixation Maps

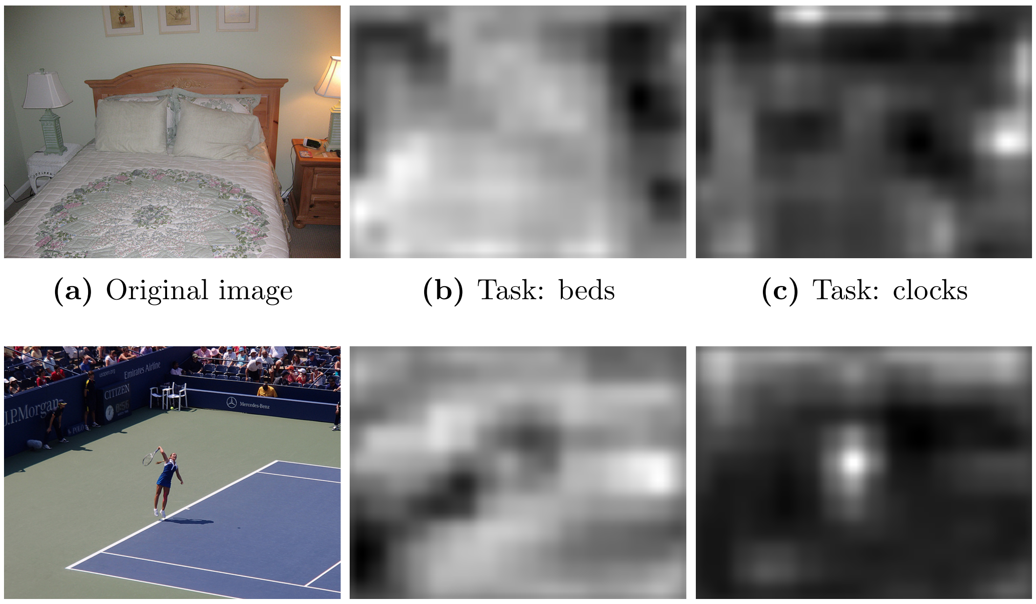
In comparison to the variant with markers on the target object locations, for the purified variant, the agreement between the fixation maps for clocks and laptops increased slightly. At the same time, the agreement between the saliency maps for the bed task and those for the other two tasks showed a strong decrease in CC. It fell to 0.545 for the comparison of the saliency maps for clocks with those for beds. This is probably a sign for an increase in task specificity, as intuitively, the locations viewed when searching for clocks or laptops should be quite similar (as for example both frequently stand on desks), while locations should differ strongly between searching for clocks and searching for beds. The agreement between the saliency maps for different images again slightly increased for the bed task in comparison to the variant with markers, but decreased for the laptop task.

Purified Fixation Maps with Markers for the Targets

The combination of the two modification methods turned out to be most promising. A visual inspection of the saliency maps showed that while the models trained for searching laptops and clocks still produce quite similar saliency predictions, the model for beds has learned to highlight large areas of uniform colour. Therefore, the saliency maps produced by this model differ perceivably from the predictions of the models for the other two tasks. The observed behaviour of the bed model makes sense, as a bed is mostly a large area of uniform texture and colour, especially if it is made. Figure 5.3 shows two example images and the corresponding saliency maps produced by the model for the clock and the model for the bed task. While the bed model performs well in the first example, in the second image it seemingly takes the tennis court for a bed.

A look at the results of the the comparison between the saliency maps of the same image for different tasks confirms the impression that the saliency predictions for the bed task are very different from those for the other two tasks: The average CC between saliency maps for the bed task and those for the clock task dropped to 0.301 and the agreement between the models for beds and laptops is not much higher (0.393). Even the difference between the predictions for clocks and laptops increased; the CC of 0.777 is relatively low in comparison to the results found for the models trained on fixation maps without or with only one modification.

At least for the bed task, the agreement between the saliency maps for different images again increased strongly to a CC of 0.109. This is probably due to the model's tendency to predict very large patches of saliency, which slightly increases the chance that the output for two different images is similar. On the other hand, a correlation coefficient of 0.1 is still very low and usually interpreted as no correlation.



(a) Original image **(b)** Task: beds **(c)** Task: clocks

(d) Original image **(e)** Task: beds **(f)** Task: clocks

Figure 5.3: Two examples for the saliency maps produced by the models for beds and clocks that were trained on the purified fixation maps with markers. It can be seen that the model for beds responds to large areas of roughly uniform colour and texture. The model for clocks performs very well in the first example where it highlights the alarm clock on the bedside table and the pictures on the wall, but it still reacts strongly to the tennis player in the second example.

All in all, both, agreement scores and visual inspection of the saliency maps, show that the task specificity of the predicted saliency was improved by the modifications of the fixation maps. However, while there are cases in which the predicted saliency matches the image and the task very well, there are also examples for which the prediction makes no sense at all. This is most often the case for images that clearly do not contain an instance of the target objects (for example street scenes or the tennis court in Figure 5.3 for beds or a picture of a building for laptops). So it seems like the model is missing a mechanism that suppresses the response to certain features depending on the overall scene content. For example, if the image is classified as showing a room, the bed model should react to the roughly uniform area that could be a bed, but it should not do so if the scene is found to show a tennis court.

5.4 Region Proposals

In this section, the images from the evaluation dataset are used to find out how beneficial the saliency information is for generating or pruning region proposals for an object search task. To do so, first, a list of all images in the set that contain at least one instance of the target object had to be created for every task. The experiments described in this section were then all done on the set of positive examples for the corresponding task.

5.4.1 Experiments

Bounding box predictions in object detection tasks are usually evaluated using the *intersection over union* (IoU) score (also known as *Jaccard index*). For two bounding boxes with areas A and B, it is simply defined by

$$IoU = \frac{A \cap B}{A \cup B} \tag{5.2}$$

It is 1 if the bounding boxes are identical and smaller otherwise, as the intersection of the bounding boxes is always smaller than or equal to the union of both areas. In the two object detection challenges PASCAL VOC and ILSVRC, a bounding box prediction for an object is counted as correct if the IoU score is at least 0.5. This may seem like a very lenient score, but at least for small ground truth bounding boxes, it is very likely that a predicted bounding box is at the right location but still does not count as a hit, because it is more than twice as big as the target bounding box.

Bounding Box Sampling

In a first set of experiments, it was evaluated how many bounding box proposals per image need to be created by the presented methods for the different target objects to achieve a coverage of at least 95 %. A coverage of 95 % means having at least one bounding box proposal that has an IoU score of over 0.5 for each ground truth bounding box in 95 % of all images. The evaluated methods are *random sampling, statistical sampling, random sampling with prior, statistical sampling with prior* and *Selective Search* (see Section 3.3 for details).

The necessary numbers soon turned out to be extremely high for tasks other than beds. Therefore, a maximum number for how many samples are considered was set and the best score that could be achieved with this number was reported if the 95 % coverage could not be achieved with fewer samples.

This experiment was done to both evaluate the different sampling methods (and mainly the sampling with the saliency map as prior over the spatial distribution of bounding boxes) and to ensure meaningful results for the following experiments on pruning the proposals with saliency: For comparing the numbers of bounding boxes that could be rejected, unnecessary high amounts of samples should be avoided.

After the necessary number of samples was determined, for each saliency model, a set of bounding box proposals was generated with each of the methods. To keep the time that was necessary for the experiments on pruning within reasonable bounds, only 25000 samples instead of 50000 were used in cases where 95 % coverage had not been achieved in the previous experiment.

When using Selective Search, the number of generated samples cannot be controlled directly, but it can be influenced by decreasing the minimum size of the initial segments (*minSize* in the MATLAB implementation provided by the authors). The default value used for Selective Search in Caffe was 200. This was reduced to 100 in order to allow the algorithm to generate more bounding boxes and to better reflect the very small bounding boxes from the clock task.

Bounding Box Pruning

The different pruning methods were evaluated on samples generated by all sampling methods: For a range of different thresholds of the value that is used for pruning (average, maximum or total saliency in the bounding box), all bounding boxes below the threshold are rejected and the remaining boxes are scored using two different scores: The average IoU score of all proposals and the coverage of the ground truth bounding boxes. All scores were averaged over all evaluated images. The value of interest is then the number of proposed bounding boxes that can be pruned away before the regarded score drops for the first time or, less strict, before it drops below 95 % of its maximum value.

5.4.2 Results

Saliency in Target Bounding Boxes

For each manner of scoring bounding boxes with saliency (maximum saliency, average saliency or summed saliency), the three images with the best and worst saliency scores in the ground truth bounding boxes were saved for inspection. As could be expected, the results for the summed saliency were rather uninteresting: The best results could be achieved for bounding boxes that spanned the whole image and the worst results for very small bounding boxes. For images with ground truth bounding boxes that had the highest maximum and average saliency, the corresponding saliency maps generally also look plausible. For the maximum saliency, this also works the other way around and thus it can be concluded that given a good saliency map, a low maximum saliency score should not occur in the region of a target object. When on the other hand the bounding boxes with the lowest average saliency score are regarded, it can be seen that some of them in fact contained a small but high peak of saliency on the target object. This confirms the concern that the average saliency value in the bounding boxes is not a good measure for the quality of bigger bounding boxes. Some examples images of objects with high or low saliency scores for their bounding box can be found in the appendix (see Figure 8.1).

Bounding Box Sampling

The number of samples necessary to get a coverage of 95 % in three consecutive trials or the coverage achieved with 50000 samples is shown in Table 5.10. The coverage that Selective Search achieves is reported for comparison along with the average number of proposals generated by the algorithm. For the sampling with prior, the used saliency maps were generated with the model trained on the fixation maps that were purified and had markers for the target objects.

Figure 5.4 shows a plot of the coverage versus the number of generated samples per image for the different tasks, exemplary for the random sampling method. The shape of the curves is almost the same for all tasks and sampling methods: An initial strong increase in coverage is followed by a very slow increase of coverage with further samples. The slope is so small that it almost seems as if the coverage saturates and going from 10000 to 20000 samples does not reliably improve the results at all. The only difference between the three tasks is the maximum coverage that can be reached with a certain number of samples. While sampling for the bed task easily reaches the 95 % mark with fewer than 2000 samples, sampling for clocks does not even get close to 50 % coverage with 10000 samples.

It turned out that for the clock task, none of the proposed sampling methods could cover more than 61 % of the ground truth bounding boxes. This might partly be due to the very small size of many of the clock bounding boxes, which makes them hard to hit and makes it even more unlikely that a bounding box in the

	clocks		laptops		beds	
	#	coverage	#	coverage	#	coverage
random sampling	50000	36 %	50000	90.8 %	1500	96.5 %
random sampling with spatial prior	50000	41.2 %	50000	90.9 %	900	96.4 %
statistical sampling	50000	58.9 %	9000	96.3 %	1400	95.5 %
statistical sampling with spatial prior	50000	61.1 %	7000	95.4 %	1100	96.1 %
Selective Search	2187.7	90.5 %	2281.4	96.3 %	1960.7	98.2 %

Table 5.10: The number of bounding box samples per image (#) that are necessary to get at least one sample with an intersection over union score of 0.5 for 95 % of the ground truth bounding boxes or the coverage achieved with 50000 samples if 95 % could not be reached beforehand. For Selective Search, the number of generated proposals can not be chosen directly, so the results are reported simply as is for comparison.

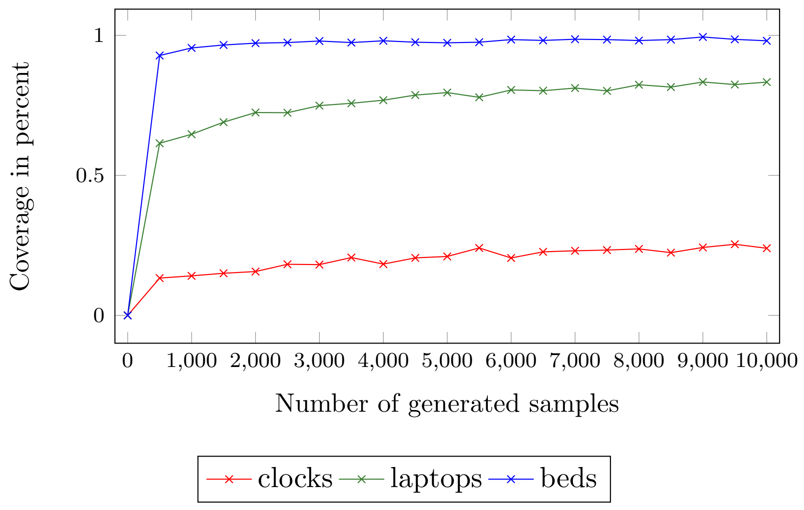


Figure 5.4: Development of the average coverage of ground truth bounding boxes (in percent) versus the number of bounding box proposals generated using random sampling. It can be seen that the coverage quickly stops to increase much with more samples for all tasks and only the amount of the initial increase in coverage differs quite strongly.

right location also has the right size to count as correct. The same probably holds true for bounding boxes of closed laptops that are shown from the side, as those result in very long and thin bounding boxes. It is however not quite clear why the coverage of ground truth bounding boxes only increases so slowly with more samples and almost seems to saturate. This could probably be dependent on the random number generator that is used for sampling (for this work, the functions from the Python library NumPy were used), but no experiments were done to check this hypothesis.

Using statistical knowledge about the target objects improves the achieved coverage strongly for laptops, where the 95 % coverage could be achieved with fewer than 10000 samples. For the clock task, the average coverage also increased, but is still far away from the 95 % mark.

Only for beds no big advantage of the statistical over the random sampling could be found, but this can probably be explained by the already very low number of samples necessary to cover most of the ground truth bounding boxes.

However, the results with the statistical sampling could be further improved by increasing the number of bins of the histograms that were used to approximate the distributions of aspect ratio and scale of the ground truth bounding boxes. The results presented are based on 100 bins for both aspect ratio and scale. In a small test with higher values (250 bins for aspect ratio and 500 bins for scale), over 80 % coverage could be reached for the clock task with fewer than 20000 samples using statistical sampling.

Sampling with the saliency maps as prior for the spatial distribution of the centre of the bounding boxes did not bring the big improvement that was expected. The coverage achieved for clocks could be improved by about 2 % when sampling with statistic and up to 6 % for random sampling. For laptops, the number of necessary samples when using statistical sampling decreased by about 2000 with the use of the prior. This number decreased also for the bed task, but only by 200 to 600 samples, which can also be a matter of luck during the sampling. Table 8.2 in the appendix contains the comparison of the effectiveness of sampling with prior using the saliency maps produced by the models based on the different variants of fixation data. It shows that the results do not vary much between the models. This suggests that the information encoded in the saliency maps is not well used by the proposed sampling method, because using the different modification methods for pruning had a much more noticeable effect on the performance.

A possible problem with the proposed method of sampling the centre of the bounding boxes can be identified by considering the bed search task: In many images, the beds take up the whole scene or at least a big part of it. So if the saliency map has high values for the whole bed (and thereby points it out correctly), the sampling algorithm has nevertheless only a relatively small chance to pick a pixel from the centre region of the bed. This is however necessary for the bounding box to be positioned correctly and thus a correct saliency map is no guaranty for good sampling results.

For smaller objects like clocks this should not be such a big problem, as ideally, the salient regions that really do mark clocks are small. In their case, the reason for the modest effect of sampling with prior could also be much simpler: Many of the proposed saliency maps have only few regions of very low saliency. Thus the effect of sampling fewer bounding boxes in those regions is probably simply not big enough to make a difference when the overall chance of hitting a target bounding box is as low as for the clock task.

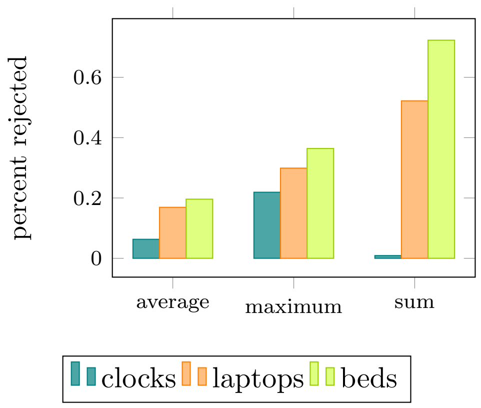
Pruning Bounding Box Proposals

For comparing the results of of the different pruning methods and to find out which saliency model produces the most useful saliency maps for pruning, it is most interesting to look at the results with bounding box proposals produced by Selective Search. The reason for this is simply that Selective Search is the only method that achieves a very good coverage for all three tasks with roughly the same number of bounding boxes per image. Figure 5.7 shows the number of rejected bounding boxes and the coverage achieved with the remaining proposals for different thresholds on the maximum saliency of the proposals.

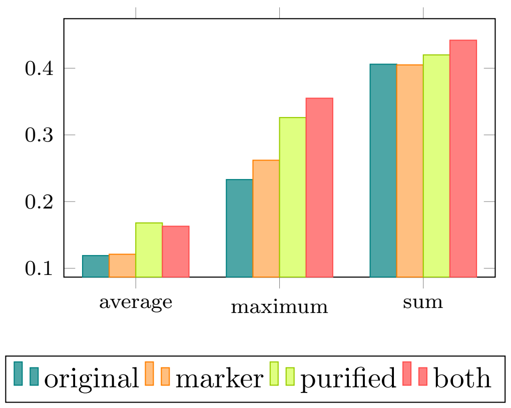
Looking at the plots in the bottom row of Figure 5.7 it can be seen that pruning works best for the bed task and has the lowest effect for the clock task. If the average coverage of the ground truth bounding boxes is not allowed to change at all, less than 10 % of the proposed bounding boxes can be rejected when searching for clocks, slightly more then 10 % for the laptops task and up to 20 % for beds. However, by allowing a decrease of the maximum achieved coverage by 5 %, the proportion of bounding boxes that can be excluded from further processing rises to about 30 % for clocks and laptops and 40 % for beds. With the roughly 2000 regions per image proposed by Selective Search, this means a total of about 600 to 800 fewer times that a classification network would need to run to process all bounding boxes.

For the proposals generated using Selective Search or the statistical sampling with the saliency maps as spatial prior, more than 75 % of the proposed bounding boxes can be rejected before the average over the IoU scores of all remaining proposals drops for the first time. This however is only true if the maximum or the sum of the saliency maps is used for pruning. So in general, the pruning with maximum and summed saliency has a high chance to remove low quality proposals while keeping the proposals with high IoU scores most of the times. The fact that this is not true if the average saliency value is used confirms the impression that the average saliency is not a good choice for judging the quality of a bounding box proposal.

If not the average IoU score of the proposals but the coverage of target bounding



(a) Averaged over all models



(b) Averaged over all tasks

Figure 5.5: Comparison of the amount of bounding boxes that could be rejected by using a threshold on the average, maximum or summed saliency in the bounding boxes. In the first plot, the results for each task are averaged over the different models types that produced the saliency maps for the respective task. The second plot shows the performance of the different model types averaged over the different search targets. All bounding box proposals were generated using Selective Search and the overall coverage was allowed to decrease by 5 %.

boxes is used as score, the highest amount of proposals could be rejected when thresholding the summed saliency values. However, this only worked well for laptops and beds. As was already expected, the ground truth bounding boxes of the clock task are generally too small to possibly contain a higher amount of saliency than any random proposal that is considerably bigger. Using a threshold on the average saliency in the bounding boxes again did not work well for any target object. In theory, this method should work best for the small bounding boxes of the clock task, as with small bounding boxes a high average saliency is much easier to achieve than with the large bounding boxes that are typical for the bed task. However, this was not the case and in the end, as expected, the maximum score again proved to be the best choice, as it treats all bounding box sizes the same. The results of the different pruning methods on the regions proposed by Selective Search can be seen in Figure 5.5a.

Figure 5.5b shows the performance of the proposed pruning methods when using the saliency maps from models trained on the different variants of the fixation data. In general, it can be seen that the models trained on the unmodified fixation maps or the fixation maps with markers are less useful for pruning than the models trained on the other two variants. The overall best performance for pruning by maximum and summed saliency could be achieved with the fixation maps that were both purified and had markers for the target objects. Only for the method based on the average saliency the purified fixation maps alone achieved a better result.

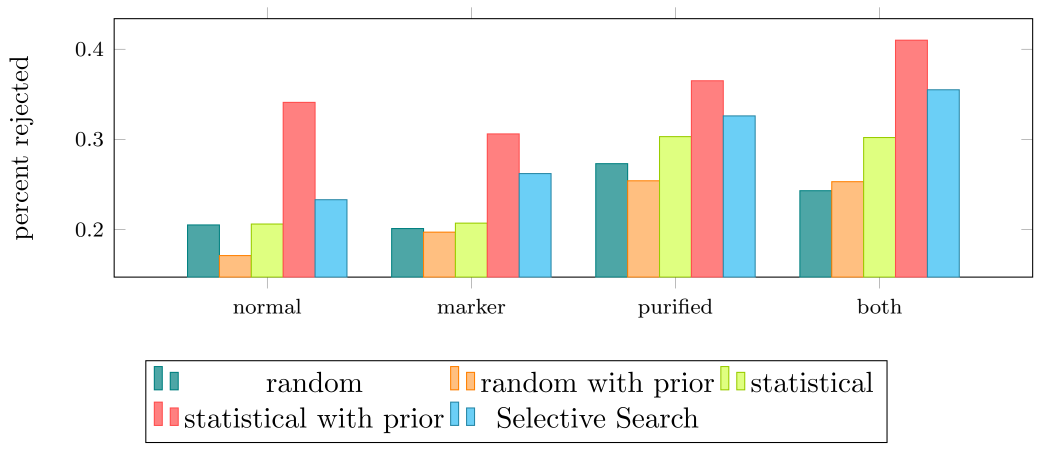
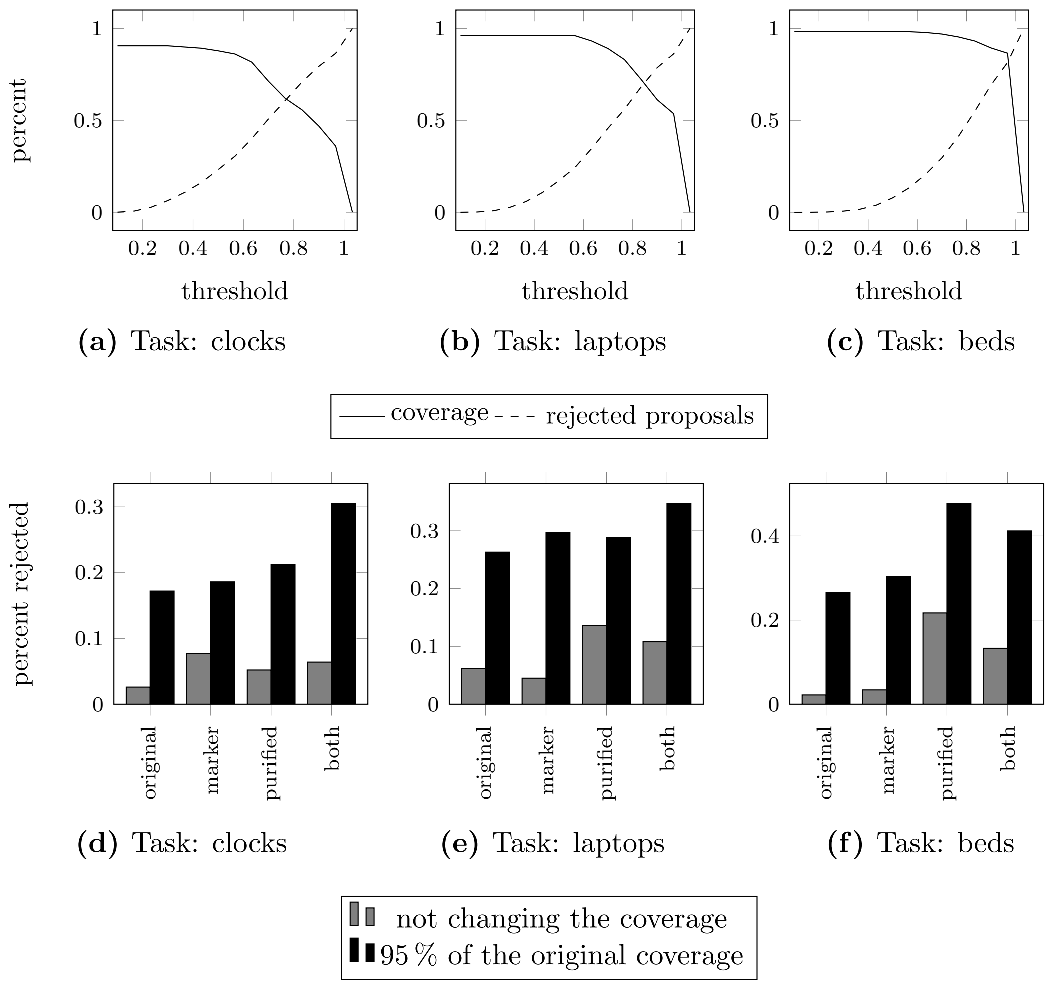


Figure 5.6: Comparison of the rejection rates of pruning with the maximum saliency value on the proposals generated by the different methods that were explored. The values are averaged over all tasks and the coverage was allowed to decrease by 5 %.

In the plots (d) to (f) of Figure 5.7, the performance of the different models with the maximum pruning method can be compared for the different tasks. There it can be seen that the ordering in performance of the different modification methods is not the same for all tasks. For example for laptops, the fixation maps with only markers work slightly better than the purified fixation maps while the combination of both is still best. The model trained to search for beds on the other hand performs best for the purified fixation maps alone. This could be due to the fact that the markers for beds have the highest influence on the fixation maps, because the average bed bounding box is very big. As a result, they encourage the model to output very large patches of saliency and thus pruning with the produced saliency maps can reject fewer bounding boxes than with sparser maps.

Finally, in Figure 5.6 the average performance of the pruning with saliency maps from the models trained on the differently modified fixation maps is compared for the different methods of generating the region proposals. The highest amount of proposals could be rejected (using the threshold on the maximum saliency) when generating them by sampling statistically with the saliency maps as prior.

This is kind of surprising, as the proposals generated when sampling with prior should have a higher quality than the ones generated without and there should therefore be fewer candidates for rejection. On the other hand, for Selective Search the number of rejections is also higher than for the random sampling methods (with and without prior) and Selective Search is thought to propose the bounding boxes with the overall highest quality. A possible explanation is that both methods produce a high number of proposals that cover the same target bounding box. So if some of them are accidentally removed by the pruning, the coverage does not drop immediately and thus the thresholds can be set higher.



(a) Task: clocks **(b)** Task: laptops **(c)** Task: beds

(d) Task: clocks **(e)** Task: laptops **(f)** Task: beds

Figure 5.7: The plots in the upper row show the coverage of ground truth bounding boxes as well as the percentage of proposals that was rejected for different thresholds on the maximum saliency value in the bounding boxes. All proposals were generated using Selective Search, so the initial coverage is close to 100 % for all three tasks. The saliency maps used for pruning come from the model that was trained on the purified fixation maps with markers. The bottom row shows the percentage of rejected bounding boxes with saliency maps from the models trained on the different variants of the fixation maps.

Chapter 6

Discussion

The results of this work have shown that a saliency model can be trained from human fixations and that the saliency predictions of the model can be used to improve region proposals for object search or detection tasks. However, strong modifications of the fixation data are necessary for the model to learn task-specific behaviour instead of task unrelated bottom-up saliency. Even with the help of those modifications, the model does not perform well in all cases. Possible reasons for this, as well as ways of improving the model, are discussed in the following chapter.

6.1 Learning Top-Down Saliency from Eye Tracking Data

Consistency of Fixation Patterns

Since the classic experiments by Yarbus [40], it has been widely believed that human gaze behaviour is influenced by top-down tasks in a way that is largely consistent among individuals. This has only recently been verified on a large scale dataset by Mathe and Sminchisescu [41]. Ehinger et al. [42] studied fixations of 19 subjects that had to search for pedestrians in images. They found an agreement of the fixations of 93 % in the target absent case and 95 % in the target present case, when predicting the fixations of one participant with the combined fixations of the others. For the cross-image control, the AUC scores were still high, with 68 % if pedestrians were present and 62 % otherwise.

Such high values of agreement were not found for the eye tracking data collected for this work. The average of the AUC score over all three tasks is only 67 % and the cross-image control lies at 55.7 %. A good part of this difference in scores can probably be explained by the rather small number of subjects that participated in the experiment. Combining only few other fixation maps to predict the fixations of one participant has been demonstrated to reduce the agreement scores (see Section 5.1). The absolute difference between the agreement of fixation patterns of participants with the same task on the same image, and the agreement of fixations on different images is still high enough to confirm the expected consistency of human gaze behaviour during search.

However, what is more important for this work is the difference of fixation patterns of participants that searched for different target objects in the same image. For the saliency model to learn to predict task-specific saliency, the fixations that are used as training data need to differ noticeably between the different tasks. While the overall agreement between the fixations from different tasks was found to be lower than the agreement of fixations of participants with the same task, the difference in scores was rather small (between 3 % in the target absent case and 5 % if the search target was present). A visual inspection of the fixation maps showed that the reason for this lies mainly in distracting image elements like faces, that are very salient in a bottom-up manner and reliably draw human attention, independent from the task.

Unfortunately, the two above mentioned studies did not include a comparison of fixation patterns for different search tasks, so there are no other scores that could be used to compare the values found in this work to. Despite the fact that the heuristic approach of removing fixations from image regions that were fixated by participants of all three tasks seems to improve the task specificity of the data and the model that is trained on it, it remains unclear to what extend human search behaviour really differs for different search targets.

Information Content of Fixation Data

Even if the eye tracking data can be modified to only contain task specific fixations, the question remains if the data contains enough information about the search process for the model to learn a good representation. This was found to be largely dependent on properties of the object that is the target of the search. It quickly became apparent that no real search (i.e. eye movement) is necessary if the targets are too obvious: Especially when searching for beds, the number of fixations per image was usually very low (because of very short viewing times) and the few fixations made could often be explained by the centre bias and task unrelated bottom-up saliency.

The information content of the recorded fixations is thought to increases with the difficulty of the search task. This in turn is dependent on the average size of the target object, the variety in the appearance of its instances and also on the variety of locations where it can be found: The clock search task was not only the hardest because many clocks are so tiny, but also because the clock category includes many different types like watches, alarm clocks and built-in digital clocks in different kitchen devices. In addition, clocks can appear in many locations, for example in almost all rooms of a house, on buildings, in train stations and on people's wrist. A bed in comparison will only very seldom appear outside of a sleeping room.

Apparently, humans process a scene not only in terms of image features, but also on a semantic level, before overt attention is employed. A recent study by Potter et al. found that humans are able to judge the semantic content of an image, like that it shows a *smiling couple*, in as few as 13 ms [43]. This time span is thought to not be long enough to allow for any feedback loops like deploying attention. Therefore, measuring overt attention will not give much information about the human search strategy if the semantic content of the scene already sufficed for solving the search task. For example, if an image is recognised to show a highway at first glance, it is very unlikely that a bed will be present in the image and thus no further detailed processing of different image locations will be done to validate this assumption.

Another difficulty in learning saliency for search from human fixation data is that the fixations are very punctual and therefore do not give any information about the extend of the object that was fixated. This makes it hard for the model to learn to associate the fixations with the underlying objects, especially if the objects are relatively big. Thus it was not surprising to see that adding markers for the target objects and their shape and size had the most noticeable effect for the bed search task.

Summary

In summary, the analysis of the eye tracking data and the results of the saliency model that was trained with it showed that the unmodified fixation data is too heavily biased towards bottom-up saliency to use it for training a top-down saliency model for search tasks. If this bias is removed, the results look much more promising. However, because of pre-attentive processing and covert attention, the fixation data does not contain complete information about the search process. It is most useful for targets and scenes where global processing of the scene content is not very helpful for the search task.

In contrast to bottom-up saliency, top-down saliency is a much more difficult process that needs a higher level of semantic processing. It is possible that the model would eventually uncover the hidden regularities behind the fixations, if it only had enough examples. This is of course based on the assumption that the feature maps provided by convolutional neural networks contain all information that is necessary for doing so, but the good results that could be achieved with those features on various different vision tasks suggest that this might be the case.

The 400 images that were used for training in this work are a very small sample of the real world and it cannot be expected that a model would learn more than the most obvious characteristics of the data when trained on them. Under this consideration, the performance of the model is already quite impressive. However, as the amount of training data that can be provided is limited (mostly by the time that eye tracking experiments take and the number of pictures that any person will willingly search), it seems like a good alternative to provide additional information to the model, like it was demonstrated with the markers for the target objects in this work.

6.2 Discussion of the Model Architecture

First of all, it can be stated that the model and the training on fixation data work in principle: Despite the slight differences in implementation, the reimplementation of Deep Gaze presented in this work achieves similar results to those reported by the authors of the original model. As described in Section 5.2, most of the differences in the scores between the two models can be explained by different evaluation conditions and the missing centre bias.

The results on the evaluation set do not differ much between the model trained on the MIT dataset and the one trained with the fixation data collected during this work. So on the unmodified fixation data, the model still learns a representation of bottom-up saliency that is plausible. In addition, despite of the low number of training images, the model does a surprisingly good job in learning task-specific saliency if the fixation maps are preprocessed to emphasise the target objects and the differences of the search behaviour.

Advantages of the Proposed model

One advantage of the presented architecture is that it only makes use of biologically plausible features that generalize well to other vision tasks like classification and does not need any complicated or expensive calculations itself. So if the computed features could be reused in other parts of the application, saliency comes almost for free.

The uncomplicated architecture also makes it easy to analyse the model and keeps the amount of parameters that have to be trained low. While a small number of parameters limits the model in its possible responses, this is still an advantage under the constraint of extremely few training data. As saliency itself is only an heuristic, it seems preferable to train a model that can only learn a simplified version of the process over having a model that could potentially reflect the whole process correctly but is bound to overfitting as it cannot get enough examples for training.

If more different target object categories are used, it could turn out that the current number of parameters is too small for the model to respond adequately different for different categories. In this case, the degrees of freedom that the model has, could be increased by using feature maps from multiple layers of the feature extraction network. By doing so, one could not only capture a broader range of different features (higher-level and low-level features). In addition, the different layers work on different scales of the input image, so the lower-level features can capture objects of smaller scales and also provide more fine-grained information about the locations of objects.

The scalability of the model to more tasks thus mostly depends on how the weights are set for the different tasks. Training separate models for every object category is of course not practical and with a higher number of target categories, a separate neural network for setting the weights of the linear combination according to the task specification would be necessary (see Section 3.2.2 for a more detailed explanation). This network would be the only part of the proposed architecture that grows if the number of feature maps or the number of target objects is increased. The network needs at least as many input units as the code for representing a target object is long and has one output unit for every weight that needs to be set in addition to its hidden units. With tens of thousands of object categories this could probably become quite big (and its calculations quite slow) but for a more modest amount of different objects the model should still work sufficiently fast without consuming too much additional memory.

Things to Improve

The proposed model of course also has some disadvantages. The first one is the rather low resolution of the produced saliency maps. This could to some extend be improved by reducing the downscaling during the feature extraction process. Especially the first subsampling by factor 4 in the conv1 layer[1] seems unnecessary strong. Presumably, the authors of [21] did this mostly to reduce the amount of memory necessary for the following calculations and similar results could be achieved with less subsampling and a deeper architecture. This line of thought was addressed by Zeiler et al. in [12], with the result that a decrease in the stride of the first convolution layer (as well as in its kernel size) led to an increase of the overall performance of the Krizhevsky network for object classification.

However, a certain amount of downscaling is also necessary to ensure high quality features. This can be seen for example in Figure 4.2 in Section 4.3, where the performance of the bottom-up saliency model increases with each pooling layer (that reduces the scale of the image by factor two), at least until a critical amount of subsampling is reached. In addition, the memory that the feature extraction network consumes is still a limiting factor for the network depth and the size of its layers. As convolution is much faster when carried out on a GPU, it is preferable to limit the network to a size that fits in the comparably small RAM of a GPU[2]. Thus the resolution of the saliency maps cannot be increased endlessly by simply avoiding the downscaling during the feature extraction process. But, as described above, the resolution could also be improved by combining low-level feature maps that have a comparably high resolution with the upscaled higher-level features.

This leads to a second, rather small problem: Because convolutions take the neighbourhood of a pixel into account, they cannot be applied at the edge of the image, where the kernel would exceed the image's dimensions. So with every convolution, the feature maps get a bit smaller. To avoid this change in size, for Deep Gaze and also in this work, the input image and intermediate feature maps are

[1]The downscaling is achieved by applying the convolution kernel with stride 4, i.e. only at every fourth pixel of the input.

[2]To the author's knowledge, the maximum amount of RAM available in a GPU is currently 24 GB and more affordable models have between 2 GB and 6 GB RAM

padded with zeroes so that the convolution can be computed at every image pixel. This of course results in inaccurate values at pixels on the edge region, and the boundary area for which results are not precise gets bigger with every convolution. The resulting artefacts can be observed in the results of the proposed saliency model, although they are not as strong as expected. Nevertheless, when combining lower-level features and high-level features, it should be taken into account that the lower-level features have a smaller boundary region and should therefore have more weight in the areas for which the higher-level features cannot provide accurate results.

The probably biggest drawback of the proposed model architecture is that it is not flexible enough: The example saliency maps for the bed search task shown in Figure 5.3 illustrate this problem quite well: In terms of image features, the tennis court in the one picture looks quite similar to the bed in the other, and thus, the model responds to both with high saliency values. A human can in this case use the overall scene context to quickly rule out the possibility that the tennis court is a bed.

But even if the model could figure out the overall semantic content of the scene, it has no possibility to react to it, because the weights of the different features are fixed for each task. Enabling such behaviour would require either feedback loops or a global scene representation that can be used to suppress the response of the saliency map in some regions of the image. Such an approach can be found in [44]. The authors combine a saliency map that is based on local features with a low-dimensional global scene representation in a Bayesian framework. The global features are used to predict the vertical position of search targets in the image and the authors demonstrate that the combination of this global features with saliency has a higher rate of detecting the search targets than the saliency model or the global features alone.

Another minor problem that might also impair the model's performance, especially for images that are very "uninteresting" under a certain task (for example a wide landscape when searching for laptops) is the softmax computation during the training. Normalising the saliency as well as the fixation values before computing the loss has the effect that the model does not get any feedback about the absolute amount of fixations in the image. It can therefore not distinguish between an image where only few fixations are for example in the centre, because the subjects had to look somewhere while classifying the scene (and finding that it is very unlikely to contain the search target) and an image where all participants fixated a certain region for a longer time because it was very interesting under their given task. However, in the proposed architecture, the softmax function is the only part that enforces competition between different regions of the saliency map. As this is an important aspect of saliency, another way of doing so would need to be found if the softmax function was going to be replaced.

Summary

To conclude, the proposed top-down saliency model architecture demonstrates a promising performance given the extremely small set of training examples. With more (and better) training data and some small changes like the combination of feature maps from multiple layers, the performance can be expected to increase even further. A lot of interesting research could still be done with the proposed model and an overview of promising options for improving it will be given in the next chapter.

However, the results also demonstrated that the model's performance is limited by its inability to react to features flexibly depending on the overall context of the scene. So if further research on the model is done, this should probably be the first point to be addressed.

6.3 Saliency for Region Proposals

The experiments with pruning region proposals demonstrated that the information encoded in the produced saliency maps is beneficial for reducing the search space for an object search task. Even for a sophisticated algorithm like Selective Search, the number of proposed bounding boxes could be reduced by up to 10 % without impairing the coverage of the ground truth bounding boxes and up to 35.5 % without losing more than 5 % of the previous coverage.

The comparison between the pruning performance with the saliency maps produced by models that were trained on the original or the modified fixation maps showed that performance was better with the modified fixation maps. As the models trained on the unmodified fixation data are de facto bottom-up saliency models, this confirms that preprocessing the fixation data allows the model to learn useful task-specific behaviour and that the task-specific saliency information is indeed more helpful for the search for a specific object than general bottom-up saliency.

Despite its slightly worse pruning performance, bottom-up saliency is of course useful as well. If the task is not to search for a certain target category but general object detection, bottom-up saliency can still help to reduce the number of regions that have to be classified by pointing out whether or not the proposed regions lie in an interesting part of the image.

In comparison to applying the saliency maps for pruning, using them as a spatial prior for sampling bounding box proposals did not bring as much benefit. This is probably due to the way in which the information was used, namely to sample the centre of the bounding boxes. For small peaks of saliency, this assumes that the most salient part of an object is located in the middle of its bounding box, which is for example not true for persons, where the face is usually most salient but lies at the upper end of the bounding box. Bigger patches of saliency on the other hand could indicate a large object of more or less constant saliency (like beds, that usually do not contain any point that is more or less salient than the rest). For them, the algorithm would need to place the centre of the bounding box in the centre of this saliency patch. Instead, in the current implementation, the location is drawn with more or less uniform probability from all pixels of the patch, what can result in large misplacements of the bounding boxes.

The only real contribution of the sampling with saliency as prior is that it reduces the number of proposals in clearly uninteresting regions of the image, such as the sky. This would probably be different if the spatial resolution of the produced saliency maps was higher and the saliency maps therefore contained more accurate information about the location of highly salient regions. In addition, both the proposed sampling methods as well as pruning with the saliency maps would benefit if the saliency prediction was sparser.

Summary

All in all, other than for beds, none of the proposed alternative methods could actually match the performance of Selective Search in terms of coverage of the ground truth bounding boxes or the number of necessary samples per image. This is partly due to the quality of the saliency maps that could still be improved, but also due to the rather unsophisticated methods that were proposed.

Developing those methods was however also not the main goal of this thesis. Instead, the experiments with pruning and sampling are intended as an in principle demonstration of how saliency could be used to improve the process of generating region proposals. Under this conditions, the experiments can be considered a success: Not only did they show that even with very simple methods, the number of proposed bounding boxes that have to be processed further can be reduced by a considerable amount without impairing the overall coverage of the target bounding boxes much. They also again confirmed that the proposed model is able to learn task-specific saliency if provided with appropriately preprocessed training data and that this top-down information is more beneficial for the search task than bottom-up saliency.

In addition, the results especially with statistical sampling already look promising and can without doubt be improved further if more time is invested.

Chapter 7

Outlook

Modelling human visual search behaviour and saliency is a challenging area of research. The saliency model proposed in this work already performs well given the small amount of training examples it could use to learn from. However, there are a lot of possibilities to still improve its performance, with and without changing the proposed architecture. Some of these possibilities will be discussed in the first part of this chapter, before some thoughts on how saliency maps could be better integrated in the process of generating region proposals for object search are presented.

7.1 Improving the Saliency Model

Feature Extraction

To further increase the quality of the presented saliency model, two possibilities come to mind quickly: The first was already suggested by the authors of Deep Gaze, namely to use even more sophisticated convolutional networks such as GoogLeNet [22] or VGG [23] for feature extraction. The depth and layer size of the used network is however limited by the amount of memory that is available. In addition, simply increasing the number of processing stages will only work to a certain point where either the resolution of the saliency maps gets too low or the boundary area where convolutions do not give exact results gets too big (remember that at the edge of the image, the convolution cannot be computed exactly because the window over which it is computed exceeds the image's dimensions).

As already explained in Section 6.2, it would be desirable to use a feature extraction network that does not reduce the resolution of the input images as fast as the Krizhevsky network does. GoogLeNet is an interesting candidate, as in this model, multiple different kernel sizes (as well as maximum pooling) are used in parallel on each of the later processing stages and their output feature maps are concatenated to form the input of the next stage. Intuitively, this should enable the model to capture image structures on a bigger range of scales, as in effect, a larger number of different receptive field sizes is used. The good results of the model in object detection prove that the generated features not only work better for classification but that they also preserve the spatial information in the image better.

It also remains to validate whether the features produced by convolutional networks that were trained for object detection or classification really generalise well for saliency modelling for visual search. Intuitively, this is the case, as the features by which one recognises an object should be the same as those one uses to search for an object. The results achieved with the proposed model also indicate that the features work well for modelling saliency, especially bottom-up effects. In order to test the pure feature generalisability, a simple way would be to include the feature extraction network in the training process and to check whether the features change dramatically if the network is allowed to learn. Training the network from scratch is not possible without much more data than currently available, but also finetuning it could prove difficult with so few examples.

Training Data

The second obvious way for improving learned saliency models is to increase the size and quality of the training data. For once, it is unclear to which extend the accuracy of the eye tracking device used for creating the fixation maps influences the models. It can be expected that a higher accuracy of fixations and a bigger amount of subjects will have positive effect on the model quality.

The number of images that are used for training and the number of participants per image are however much more important: Ideally, datasets should contain examples from a huge variety of locations and scenes, sampling many different variants of all relevant object classes. At the same time, all those images should be viewed by as many subjects as possible for every task. The five subjects per task in this work are clearly too few to get a really good approximation of the ground truth distribution of fixations (see Section 6.1), as are the 400 training images.

However, most of today's eye tracking datasets contain at most 1000 images and the number of viewers per image is seldom higher than 20, which is mainly due to the big time consumption of eye tracking experiments and the still high price of high accuracy eye tracking devices. An alternative for eye tracking data could be mouse tracking: In [45], the authors demonstrated that the data gathered from a large scale mouse tracking experiment conducted via Amazon Mechanical Turk could be used to learn a bottom-up saliency model that beat all other models in the MIT saliency benchmark by a large margin (notably in all scores). In the mouse tracking experiment, the participants are basically shown a blurred version of the image, that is only sharp at a small region around the current mouse cursor position. So by moving the mouse, participants can explore the parts of the image that they find interesting. The authors already implied that their method of generating data could easily be employed to other viewing tasks such as search.

What is most interesting about this approach is that it might provide a cleaner way to get rid of the influence of behaviourally driven bottom-up saliency, that has to otherwise be reduced by further processing of the fixation data. As gaze direction is to a large extend controlled subconsciously, participants in an eye tracking experiment can hardly avoid looking at for example faces. If in contrast they have to move the mouse for exploring an image, they will presumably act more conscious and can therefore omit task unrelated "fixations".

While the method for removing task unrelated fixations that was explored in this work might have a similar effect, it comes with the risk of cancelling out task related fixations as well, if they coincided with location of something that is salient in a bottom-up way: For example, the dataset used in this work contains a lot of images that show cats lying on either beds or laptops. Thus, deciding whether a fixation near or on a cat was task-related or not is not really possible.

Combining Features From Multiple Processing Stages

As already explained in the Discussion Chapter, using feature maps from multiple layers of the feature extraction network can be expected to improve the performance of the model, at least if enough training data is available to train the additional parameters.

The idea is that feature maps that are produced by different stages of the Krizhevsky network encode different features that might be beneficial for different aspects of saliency: The lower layers produce low-level features that are best fit to model pop-out effects or the basic geometric shape of small objects, whereas the later layers will react to objects and patterns that are salient because of there semantic content. In addition, the earlier layers cover objects of smaller scales and the feature maps have a higher spatial resolution. The bigger amount of parameters can also be expected to help the model to better adapt to different tasks, as with more features, it can form a richer representation of the search task.

As was also done for Deep Gaze, all feature maps could simply be linearly combined in one step, if they are normalised before to avoid that some features dominate the others because they have overall higher values. An alternative approach would be to first combine the feature maps of each layer separately and to then combine the resulting intermediate saliency maps in a separate step. This would not add too many parameters (as many as different feature layers are used) and has the advantage that the overall weight the model assigns to the different layers can be easily observed from the weights of the final combination. In addition, it might be helpful that in this way the weights of the separate linear combinations could be initialised with the weights learned when only the respective layer of the feature extraction network was used.

Providing Additional Information

The results with training the model on modified fixation maps have shown that the model can benefit from getting additional information along with the fixations. Removing the fixations that were made by participants from all tasks clarified which of the fixations were specific to the current task. As discussed before, this step could perhaps be made unnecessary by using mouse tracking instead of eye tracking.

Adding markers for the target locations included information about the size and shape of the target objects. Instead of approximating them with a Gaussian that fits the shape of the bounding box, it would of course be more accurate to use the segmentation of the objects as marker. Segmentation information is already included in the COCO dataset, so there would be not much extra effort necessary to include it.

Global information about the scene layout and the overall scene classification could be very valuable for the model. However, providing such information is not straight forward and including it into the saliency computation would probably require big changes in the model architecture.

A comparably easy approach could be to include a coarse segmentation of the scene as layout information. The model could then learn which parts of a scene are often fixated for a search and which parts do not have to be searched further (for example, when searching for beds, street, lawn or sky do not have to be considered, but surfaces in rooms should be). This information could be used to suppress or enhance the response of features dependent on the region they lie in. With this mechanism, mistakes like predicting a tennis court as a bed, because they look similar in terms of visual features, could be avoided and the model would gain flexibility.

However, for this approach to work, the model also needs a possibility to generate a scene layout representation. This is a completely different field of research that has not been explored during this work, so no statements can be made about how easy or difficult this problem is and whether it could be integrated in the current feature extraction scheme, but it would certainly be an interesting direction for further research.

In the line of providing more information to the model, it would also be interesting to somehow include 3D information: Object classification and localisation performance heavily depend on segmenting the object from the background and it seems reasonable to assume that depth information would be very beneficial for this task. This is also supported by Finlayson's 2013 PhD thesis [46], where the author comes to the conclusion that depth information from stereo vision is used at multiple processing steps in the human visual system, including search and object classification. However, she suggests that depth should not be used as an additional feature but rather as a third dimensions for feature response localisation. This seems rather difficult as visual information becomes sparse in the third dimension. As stated in [22], our current hardware is highly optimised for dense calculations, so including a third dimension would increase the computation time even more than one would first expect for the extra data.

7.2 Using Saliency for Region Proposals

The simple methods for creating or pruning bounding box proposals that were presented in this work can certainly be improved in many aspects. For example, the thresholds for rejecting proposed bounding boxes could be changed depending on the size of the bounding box. This can be expected to improve results because the bigger an object (and therefore its bounding box) gets, the higher the chance is that the object has different sub-parts that are more or less salient. Thus, the saliency inside a box should have more importance for smaller bounding boxes. For bounding boxes that span almost the whole image, saliency is of no big use anyway, as their position can not be varied much in order to include more or less saliency.

If saliency should be used directly during the generation of bounding box proposals, it would be interesting to somehow integrate it in the Selective Search algorithm. This could be done either by using the saliency map like an additional channel of the image and defining a similarity measure that takes saliency into account or by using it as a spatial prior for the distribution of the initial bounding boxes. By doing so, the problem that bounding boxes are not necessarily centred on saliency peaks could be solved as the initial bounding boxes are merged with their neighbours depending on the image features and thus in the best case, the full segmentation of the salient object is approximated by one of the resulting bounding boxes.

However, it probably makes more sense to not use the proposed regions directly as bounding boxes. Most state of the art algorithms perform an additional regression step after the classification anyway, so that the form of the proposed region does not matter much for the final bounding box prediction. In addition, if regions should be classified that are not quadratic, additional warping is necessary to bring the proposed region into a form that can be used as input for the feature extraction network. If instead patches of fixed aspect ratio would be used as proposals, sampling would get much easier as only the scale and position of the sample have to match the ground truth bounding box. Also, without the warping, the features computed for the saliency map could be easier reused for classification. Of course, all this will only work if a good method for predicting the bounding box of the object in the given image patch can be found. But given the already decent performance of CNNs for localisation of the single most dominant object in an image, this does not appear to be an unsolvable problem.

To take this another step further, in the end, the goal is of course to combine all steps, i.e. feature extraction, the task dependent saliency model, region proposals and classification, into one framework. For this, the saliency models for the different tasks need to be unified into one network that receives the task as additional input, as was proposed in Section 3.2.2. Also, when the feature extraction is part of the model, the feature maps either have to be used on their original scale or the rescaling step has to be included into the model. This becomes inevitable if feature maps from more than one layer of the feature extraction network should be used, as they have different scales.

Doing so would make it possible to train the whole model from the input image to the output object annotations with bounding box predictions. This would probably help especially in the saliency prediction step, as in the current implementation, the model has no knowledge about what the saliency data should be used for and gets no feedback about how useful its predictions were for the region proposal step and if the target of the search was found or not. Such a joint optimisation is not easy, as the amount of parameters increases extremely in comparison to the separate parts. It would require a large training dataset and a good initialization of the parameters of each sub-part. This could be achieved by first training them for their specific task independently and then finetuning the complete model jointly.

And finally, instead of using the region proposals as presented in this work, a real attention system could be implemented based on the computed saliency map. Such a system would not only point out which regions are interesting given the current search task, but also determine the order in which the regions should be processed. It could probably even respond dynamically depending on the results of the classification of the last processed region. However, attention is a very complicated process that has not yet been completely understood. And although computational models of attention exist, including them into a system for object search, which should in the best case run in real time, poses a challenging task for future work.

Bibliography

[1] K. Koch, J. McLean, R. Segev, M. A. Freed, M. J. B. II, V. Balasubramanian, and P. Sterling, “How much the eye tells the brain,” *Current Biology*, vol. 16, no. 14, pp. 1428 – 1434, 2006.

[2] D. J. Simons and C. Chabris, “The invisible gorilla.” `http://www.theinvisiblegorilla.com/gorilla_experiment.html`, 1999.

[3] M. Carrasco, “Visual attention: The past 25 years,” *Vision Research*, vol. 51, no. 13, pp. 1484 – 1525, 2011. Vision Research 50th Anniversary Issue: Part 2.

[4] D. Broadbent, “Perception and communication,” 1958.

[5] J. A. Deutsch and D. Deutsch, “Attention: some theoretical considerations.,” *Psychological review*, vol. 70, no. 1, p. 80, 1963.

[6] M. Cerf, J. Harel, W. Einhäuser, and C. Koch, “Predicting human gaze using low-level saliency combined with face detection,” in *Advances in neural information processing systems*, pp. 241–248, 2008.

[7] G. A. Alvarez and S. L. Franconeri, “How many objects can you track?: Evidence for a resource-limited attentive tracking mechanism,” *Journal of Vision*, vol. 7, no. 13, 2007.

[8] L. Itti, C. Koch, and E. Niebur, “A model of saliency-based visual attention for rapid scene analysis,” *IEEE Transactions on pattern analysis and machine intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[9] A. M. Treisman and G. Gelade, “A feature-integration theory of attention,” *Cognitive psychology*, vol. 12, no. 1, pp. 97–136, 1980.

[10] T. Judd, K. Ehinger, F. Durand, and A. Torralba, “Learning to predict where humans look,” in *Computer Vision, 2009 IEEE 12th international conference on*, pp. 2106–2113, IEEE, 2009.

[11] C. Shen, M. Song, and Q. Zhao, “Learning high-level concepts by training a deep network on eye fixations,” in *NIPS Deep Learning and Unsup Feat Learn Workshop*, vol. 2, 2012.

[12] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *Computer Vision–ECCV 2014*, pp. 818–833, Springer, 2014.

[13] M. Kümmerer, L. Theis, and M. Bethge, "Deep gaze I: boosting saliency prediction with feature maps trained on imagenet," *CoRR*, vol. abs/1411.1045, 2014.

[14] J. M. Wolfe, "Guided search 2.0 a revised model of visual search," *Psychonomic bulletin & review*, vol. 1, no. 2, pp. 202–238, 1994.

[15] V. Navalpakkam and L. Itti, "Modeling the influence of task on attention," *Vision research*, vol. 45, no. 2, pp. 205–231, 2005.

[16] P. van de Laar, T. Heskes, and S. Gielen, "Task-dependent learning of attention," *Neural Networks*, vol. 10, no. 6, pp. 981 – 992, 1997.

[17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *CoRR*, vol. abs/1409.0575, 2014.

[18] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vision*, vol. 88, pp. 303–338, June 2010.

[19] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[20] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1, pp. 886–893, IEEE, 2005.

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. Burges, L. Bottou, and K. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

[25] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.

[26] I. Sobel and G. Feldman, "A 3x3 isotropic gradient operator for image processing," 1968.

[27] E. Mach, *Über die Wirkung der räumlichen Vertheilung des Lichtreizes auf die Netzhaut: vorgelegt in der Sitzung am 3. October 1865.* K. k. Hof-und Staatsdruckerei, 1865.

[28] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[29] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[30] D. L. K. Yamins, H. Hong, C. F. Cadieu, E. A. Solomon, D. Seibert, and J. J. DiCarlo, "Performance-optimized hierarchical models predict neural responses in higher visual cortex," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8619–8624, 2014.

[31] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," *arXiv preprint arXiv:1310.1531*, 2013.

[32] Z. Bylinskii, T. Judd, A. Borji, L. Itti, F. Durand, A. Oliva, and A. Torralba, "Mit saliency benchmark." `http://saliency.mit.edu/`.

[33] T. Judd, F. Durand, and A. Torralba, "A benchmark of computational models of saliency to predict human fixations," 2012.

[34] M. Kümmerer, T. Wallis, and M. Bethge, "How close are we to understanding image-based saliency?," *CoRR*, vol. abs/1409.7686, 2014.

[35] T. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[36] T. E. Tribe, "Eye tracking hardware." `https://theeyetribe.com/products/`.

[37] M. Tall, K. R. Choudhury, S. Napel, J. E. Roos, and G. D. Rubin, "Accuracy of a remote eye tracker for radiologic observer studies: Effects of calibration and recording environment," *Academic Radiology*, vol. 19, no. 2, pp. 196 – 202, 2012.

[38] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[39] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio, "Theano: a CPU and GPU math expression compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[40] A. L. Yarbus, "Eye movements during perception of complex objects," in *Eye movements and vision*, pp. 171–211, Springer, 1967.

[41] S. Mathe and C. Sminchisescu, "Action from still image dataset and inverse optimal control to learn task specific visual scanpaths," in *Advances in Neural Information Processing Systems 26* (C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger, eds.), pp. 1923–1931, Curran Associates, Inc., 2013.

[42] K. A. Ehinger, B. Hidalgo-sotelo, A. Torralba, and A. Oliva, "Modeling search for people in 900 scenes: A combined source model of eye guidance," *Visual Cognition*, 2009.

[43] M. C. Potter, B. Wyble, C. E. Hagmann, and E. S. McCourt, "Detecting meaning in rsvp at 13 ms per picture," *Attention, Perception, & Psychophysics*, vol. 76, no. 2, pp. 270–279, 2014.

[44] A. Torralba, A. Oliva, M. S. Castelhano, and J. M. Henderson, "Contextual guidance of eye movements and attention in real-world scenes: the role of global features in object search.," *Psychological Review*, vol. 113, pp. 766–786, October 2006.

[45] M. Jiang, S. Huang, J. Duan, and Q. Zhao, "SALICON: Saliency in context," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[46] N. Finlayson, "Visual search in 3d space: Deploying attention in depth," 2013.

Chapter 8

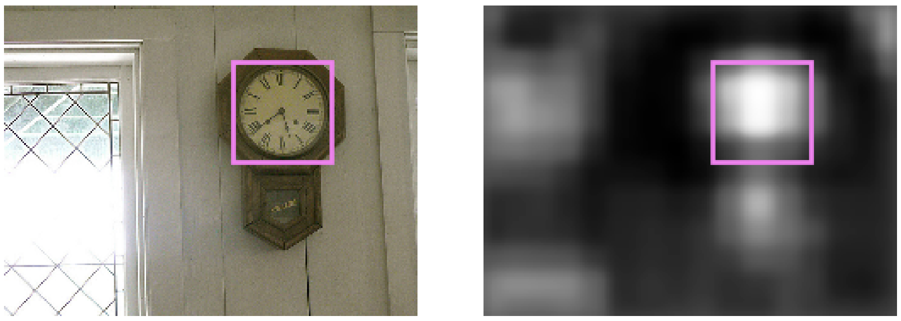
Appendix

This chapter contains some additional data and example images that were too big to fit in the actual text.

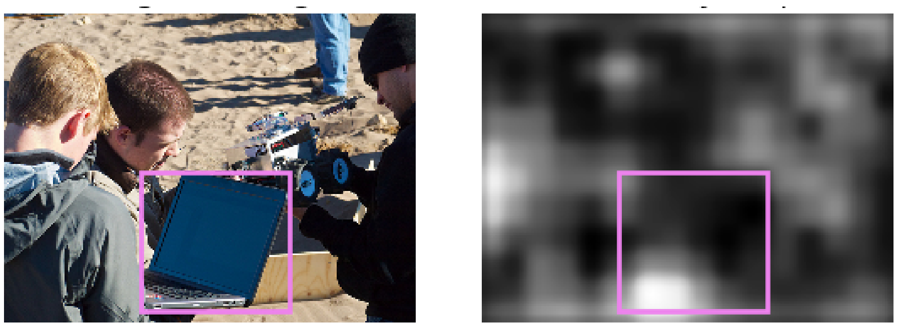
On the Enclosed CD

The enclosed CD contains a PDF version of this document as well as the source code that was written for this thesis and additional data from the evaluations.

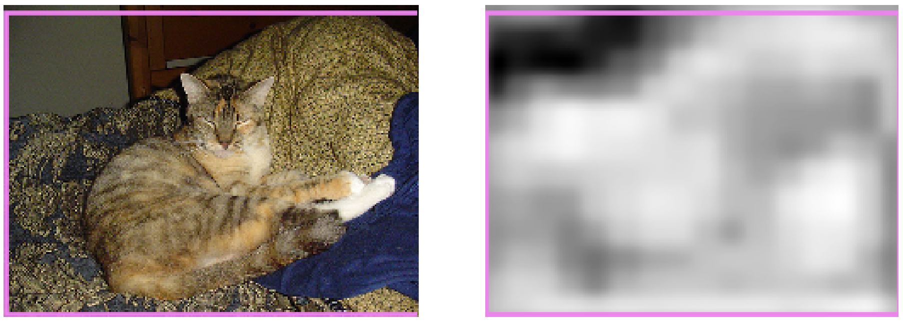
- **eyeTracking:** The software for recording fixation data (requires OpenCV).
- **fixationAnalysis:** Scripts used to analyse the fixation data.
- **preprocessing:** Scripts for choosing the images for the dataset and adjusting their brightness.
- **caffe:** Contains the Caffe framework with modified and additional code.
- **learning:** Convenience scripts for automatically running the model training with different parameters.
- **evaluation:** Scripts for bounding box sampling and pruning and saliency map comparison. Includes the Selective Search implementation provided by the authors at `http://koen.me/research/selectivesearch/`.
- **evaluation_data:** Additional data from the evaluations that were done.



(a) Clock ground truth bounding box with the highest maximum saliency value.



(b) Laptop ground truth bounding box with one of the lowest average saliency values.



(c) Bed ground truth bounding box with the highest total saliency value.

Figure 8.1: Examples for the saliency maps produced by the model trained on fixation maps that were purified and had markers on the target locations. The ground truth bounding boxes are annotated in purple in the original image and the corresponding saliency map. All examples were either amongst the three examples with the highest or lowest saliency for the regarded value. It should be noted that the saliency map for the laptop search task actually contains a high peak of saliency on the target object and has a bad score anyway.

	Euclidean Loss			AUC			CC			SIM		
	big	medium	small	big	medium	small	big	medium	small	big	medium	small
conv1	$1.234 \cdot 10^{-6}$	$1.234 \cdot 10^{-6}$	$1.234 \cdot 10^{-6}$	0.532	0.536	0.535	0.047	0.048	0.043	0.172	0.172	0.172
relu1	$1.231 \cdot 10^{-6}$	$1.230 \cdot 10^{-6}$	$1.229 \cdot 10^{-6}$	0.587	0.603	0.616	0.106	0.120	0.129	0.188	0.192	0.195
pool1	$1.230 \cdot 10^{-6}$	$1.228 \cdot 10^{-6}$	$1.227 \cdot 10^{-6}$	0.597	0.614	0.626	0.123	0.138	0.145	0.194	0.198	0.200
norm1	$1.229 \cdot 10^{-6}$	$1.227 \cdot 10^{-6}$	$1.226 \cdot 10^{-6}$	0.606	0.619	0.630	0.124	0.134	0.139	0.192	0.195	0.195
conv2	$1.226 \cdot 10^{-6}$	$1.224 \cdot 10^{-6}$	$1.223 \cdot 10^{-6}$	0.625	0.638	0.648	0.145	0.156	0.162	0.196	0.198	0.198
relu2	$1.226 \cdot 10^{-6}$	$1.224 \cdot 10^{-6}$	$1.223 \cdot 10^{-6}$	0.622	0.635	0.650	0.145	0.160	0.169	0.194	0.192	0.193
pool2	$1.223 \cdot 10^{-6}$	$1.221 \cdot 10^{-6}$	$1.225 \cdot 10^{-6}$	0.633	0.648	0.646	0.162	0.175	0.153	0.202	0.201	0.194
norm2	$1.223 \cdot 10^{-6}$	$1.221 \cdot 10^{-6}$	$1.225 \cdot 10^{-6}$	0.635	0.651	0.647	0.162	0.175	0.152	0.201	0.201	0.194
conv3	$1.219 \cdot 10^{-6}$	$1.215 \cdot 10^{-6}$	$1.219 \cdot 10^{-6}$	0.638	0.649	0.650	0.184	0.197	0.174	0.201	0.201	0.198
relu3	$1.215 \cdot 10^{-6}$	$1.210 \cdot 10^{-6}$	$1.215 \cdot 10^{-6}$	0.647	0.660	0.658	0.196	0.215	0.191	0.204	0.207	0.204
conv4	$1.209 \cdot 10^{-6}$	$1.202 \cdot 10^{-6}$	$1.212 \cdot 10^{-6}$	0.638	0.654	0.661	0.219	0.238	0.200	0.201	0.205	0.203
relu4	$1.208 \cdot 10^{-6}$	$1.200 \cdot 10^{-6}$	$1.212 \cdot 10^{-6}$	0.647	0.659	0.661	0.222	0.243	0.203	0.208	0.214	0.209
conv5	$1.209 \cdot 10^{-6}$	$1.203 \cdot 10^{-6}$	$1.215 \cdot 10^{-6}$	0.647	0.659	0.660	0.220	0.233	0.196	0.206	0.207	0.203
relu5	$1.213 \cdot 10^{-6}$	$1.205 \cdot 10^{-6}$	$1.215 \cdot 10^{-6}$	0.643	0.668	0.661	0.226	0.255	0.202	0.210	0.224	0.214
pool5	$1.210 \cdot 10^{-6}$	$1.216 \cdot 10^{-6}$	$1.227 \cdot 10^{-6}$	0.664	0.663	0.619	0.240	0.201	0.122	0.222	0.215	0.192

Table 8.1: Scores and Euclidean Loss achieved with feature maps from the different layers of the Krizhevsky network and different sizes of the input images.

	clocks				laptops				beds			
	random		statistical		random		statistical		random		statistical	
	#	coverage	#	coverage	#	coverage	#	coverage	#	coverage	#	coverage
original	50000	42.5 %	50000	61.3 %	50000	91.3 %	8000	96.2 %	900	96 %	1100	96.9 %
marker	50000	36.6 %	50000	60.7 %	50000	91.4 %	800	96 %	700	95.4 %	1000	95.7 %
purified	50000	40.7 %	50000	60.9 %	50000	92.6 %	10000	96.7 %	800	96.3 %	8000	95.6 %
both	50000	41.2 %	50000	61.1 %	50000	90.8 %	7000	95.4 %	900	96.4 %	1100	96.9 %

Table 8.2: The number of samples necessary for achieving 95 % coverage of ground truth bounding boxes in three consecutive runs or the coverage achieved with 50000 samples if 95 % could not be achieved with fewer samples for the two sampling methods that use the saliency map as a spatial prior.

Selbständigkeitserklärung