INVESTIGATING THE GENERALIZATION ABILITIES OF BOTTOM-UP VISUAL SALIENCY MODELS IN TASK-SPECIFIC SCENARIOS USING TRANSFER LEARNING

By

Konstantinos Chaldaiopoulos

2024

A Thesis submitted in partial fulfillment of the requirements for the degree of

Master of Artificial Intelligence

University of Piraeus and NCSR Democritus

Piraeus, Athens

INVESTIGATING THE GENERALIZATION ABILITIES OF BOTTOM-UP VISUAL SALIENCY MODELS IN TASK-SPECIFIC SCENARIOS USING TRANSFER LEARNING

by

Konstantinos Chaldaiopoulos

2024

A Thesis submitted in partial fulfillment of the requirements for the degree of
Master of Artificial Intelligence
University of Piraeus and NCSR Democritus
Piraeus, Athens

Approved by:

Dr.Stasinos Konstantopoulos

Committee Chair

Dr.Aikaterini Pastra

Committee Member

Prof.Georgios Vouros

Committee Member

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Stasinos Konstantopoulos, for his consistent support, patience, and trust. This endeavor would not have been possible without his thought-provoking suggestions and comments.

I would also like to extend my sincere appreciation to Dr. Aikaterini Pastra. Her expertise and knowledge were invaluable throughout this journey. I could not have undertaken this challenge without her detailed and always constructive feedback.

Last but not least, I would like to thank Prof. Georgios Vouros for providing me with stimulating ideas and an exceptional environment that has allowed me to grow as a scientist and individual.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In this thesis, we combine knowledge from two different topics: the first is computational models of attention and free-viewing fixation prediction, and the second is transfer learning and the generalization ability of deep learning models.

Inspired by human transfer learning abilities, we hypothesized that a fixation model trained on fixation maps representing human free-viewing behavior would have increased generalization abilities compared to a task-optimized model.

To test this hypothesis, we created different experimental scenarios that aimed to extensively compare the generalization abilities of these two models or agents. In these scenarios, we introduced two animal image segmentation tasks between the agents: one similar and one dissimilar to that of the task-optimized agent.

We concluded that, for our experimental conditions, the fixation agent did exhibit increased generalization abilities, both when direct training was applied to each task separately and when retraining was performed successively.

CHAPTER 1

INTRODUCTION

1.1 Overview

Each environment we find ourselves in is abundant with information. Our brain has to undergo the strenuous task of controlling this information stream. The process of the brain that "controls and tunes information processing" [88] can be arguably called *attention*. A subset of this process that deals with visual information is *visual attention*.

Visual attention researchers use different categorizations for attention. One of them is the distinction between *bottom-up* and *top-down* attention. For example, when viewing a scene, attention may be involuntarily guided by the most stimulating parts (bottom-up mechanism) or be voluntarily guided according to a specific goal of a viewer (top-down mechanism). Intuitively it may seem that "everyone knows what attention is" [29], however, how these mechanisms function and interact is not always apparent. Are these mechanisms competitive, supplementary, or a mixture? How can researchers isolate and study them? These questions have been extensively researched by cognitive and psychology scientists who created well-thought experiments to address them.

A model of visual attention is a theory that aims to explain experimental observations or make predictions by addressing possible patterns. Based on these models, researchers create computational models of visual attention. Computational models are not simply a theory. They are "instances of models of visual attention" [88]. Given an input image, they can produce output images reflecting the most prominent parts of an image. They are divided

into many different categories [87] based on their initial hypothesis with one of the most researched being the saliency map [36] branch.

The saliency map branch originates in the *Feature Integration Theory* (FIT). Feature Integration Theory proposes that "attention must be directed serially to each stimulus in a display whenever conjunctions of more than one separable features are needed to characterize or distinguish the possible objects presented" [86]. It was a hypothesis that aimed to explain the results of experimental observations. The hypothesis suggested a serial scan of attention to distinguish objects, especially when the objects have a similar "conjunction of features" [86].

This initial hypothesis led to two significant advancements. The first advancement was the proposal of a computational mechanism based on FIT by Koch and Ulman [36] in which the term "saliency map" was first mentioned. The second advancement was the implementation of this proposal by Itti et al. [27]. These advancements aimed to "explain covert shifts of attention" [41]. However, after the sufficiently established connection between model predictions and eye gaze [37, 71], computational models of attention employing saliency maps have begun to represent mostly overt attention bottom-up processes [6, 7, 69, 79].

The focus of these computational models was to make fixation predictions as accurate as possible. However, due to small datasets, models trained only on fixation maps could not achieve their full potential. A significant improvement in model accuracy was achieved when the models began to utilize the generalization ability from models with deep architectures trained on task-specific scenarios, "loosely related" [41] to fixation prediction like object recognition.

For example, Kümmerer et al. [43] used ImageNet [14], a pioneering model for object recognition, as a base model and then retrained it on the MIT300 dataset [9]. Similarly, Pan et al. utilized ImageNet and retrained on the MIT300 and Salicon datasets [32] while the Salicon model [32] and the model by Kruthiventi et al. [40] employed the more advanced VGG-19 [82] as their source before applying the retrain.

This improvement is due to *transfer learning*, a phenomenon that arises when knowledge from one domain is applied to another related domain. In the above cases saliency models, exploit the knowledge gained from models trained with extensive datasets and very deep architectures [14, 82] in a task loosely related to fixation prediction like object recognition.

However, according to our understanding, the opposite —meaning the generalization ability of saliency computational models to task-specific scenarios— has never been studied utilizing transfer learning. This is followed logically by the fact that until recently, the fixation datasets [5, 9] did not contain a significant amount of images, and therefore the focus was on improving accuracy with limited resources. Subsequently, testing the transfer learning abilities of such models on new tasks was even more challenging.

To tackle the problem of small datasets, scientists introduced different techniques to capture fixation-like data like mouse-clicking, which had a very high correlation with fixations from observers [32] aiming to create low-cost techniques for extensive dataset creation. These datasets are sufficient for training an agent on saliency map generation without pre-training. This agent will possess the generic skill of creating bottom-up saliency maps for different visual scenes.

Inspired by human transfer learning abilities, we hypothesize that a fixation agent trained on general purpose human fixation data is more adaptable when trained in new tasks compared to a task-specific one. To test this hypothesis, we will compare the performance of this general-purpose agent with that of an agent specifically trained for an image segmentation task. The general-purpose agent will be pre-trained to generate fixation predictions and retrained on the same task as the segmentation agent. The segmentation agent will be optimized on this task while the general-purpose agent won't be.

Then, we will introduce common tasks between the agents that become more and more dissimilar. By observing their adaptability and accuracy in common tasks, we expect valuable insights, especially for the general-purpose agent. In particular, we expect that the general-purpose agent will prove to be more adaptable across tasks, while the segmenta-

tion's agent ability will be greatly influenced by the similarity between tasks.

This hypothesis will be extensively tested across different transitive [85] scenarios, assuming that there could be a difference in the acceptance or rejection of the hypothesis when there is direct retraining, where the agents adapt to only one task at a time, compared to successive retraining, where the agents adapt to each task one after the other.

CHAPTER 2

BACKGROUND

2.1 Cognitive Science Background

2.1.1 The Visual Pathway

To understand the properties of attention, a basic background in the anatomy of the visual pathway is required. For this reason, we will present a short description of its structure. For a more detailed explanation that includes structural measurements, orientations, blood flow details, and aging information, our main source of information was [76].

The visual pathway consists of a series of *cells* that are connected with *synapses*. A cell is the basic building block of a living being and a synapse is a connection mechanism through which an electrical or chemical flow passes. Scientists usually term this electrical or chemical flow a ”signal”, ”message”, or ”information”.

The first type of cells that are encountered in the visual pathway are the *photoreceptors*. They are located inside the *retina*. There are two types of photoreceptors, *rods* and *cones*. The rods and cones convert the light entering the eye into a *neuronal signal* which is then passed to the *bipolar* cells and then the *amacrine* cells. The bipolar cell and the amacrine cell are *interneuron* cells connected with the *ganglion* cell.

All, the mechanisms described above are located inside the retina. The axons of the ganglion cells transmit the electrical impulses outside of the retina via the *optic nerve*. The optic nerve consists of the *retinal nerve fibers*, which are the extension of the ganglion cells, and the *glia* cells. The glial cells hold the structure of the nerve fibers together. One optic

nerve structure consisting of the retinal nerve fibers and the glial cells arises from each retina. These two structures cross each other in the *optic chiasm* and then extend toward the *Lateral Geniculate Nucleus* (LGN). The part of the nerve fiber-glia cell structure that connects the optic chiasm with the LGN is called the *optic tract*.

The LGN is located inside the thalamus. All sensory systems, except the olfactory the sensory system related with smell, pass through the thalamus. LGN is asymmetrical and cone-like. This is the part where the structure originating from the ganglion cells terminates. In the LGN complex processes occur. One of them is the regulation of information, ensuring the most important are sent to the *visual cortex*. The fibers originating from the LGN project to the primary visual cortex or *striate cortex*. These fibers are called *optic radiations*.

In the striate cortex (V1), the regulated information is processed and then transmitted to *higher visual association areas* called the *extrastriate cortex*. These areas, which were formerly called the Broadman areas 18 and 19, are located around the striate cortex and contain the distinct sub-areas V2, V3, V4 and V5.

2.1.2 Defining Visual Attention

The brain, through the visual pathway, is not able to utilize all available information in its environment. However, the brain can control and tune the information processing [88]. As mentioned in the introduction, this process can be arguably called attention and a sub-process of attention is visual attention. The prioritization of certain stimuli against others is called selective attention [74]. By studying the attention mechanism, scientists created different categorizations based on the viewpoint and focus of the experiments. The two most common categorizations are based on the existence of eye gaze focus, and the intentions or prior knowledge [35] of an individual.

Overt vs Covert Attention: Attention that is connected with the physical direction of the eye gaze is called *overt attention*. However, the focus of the eye gaze is not always the area of interest in an attentive process. Although eye gaze and attention are highly correlated, we

can focus our attention on areas of our peripheral vision without directly looking at them. One can easily observe this phenomenon in the deep calculations of professional chess players who are mentally focused on the position of the chess board without directly looking at it. This phenomenon is described as *covert attention* [17]. Due to the creation and optimization of eye trackers, mechanisms for recording overt attention are easier to obtain and use. For that reason, overt attention is studied significantly more than covert attention.

Bottom-Up versus Top-Down Attention: Another categorization of attention is based on the intentions or lack of in an individual. It is a categorization of the selective attention processes. Bottom-up attention occurs when an individual is naturally attracted to a stimulus in a scenery due to its properties. For example, the eye gaze is attracted to the moon in an empty night sky. There is no intention behind this attraction. It just naturally occurs.

However, when a human has an intention, the intention biases the attention mechanism to detect areas relevant to the goal. This process is called top-down attention. For example, when an individual seeks his black dog in a field, his gaze will gravitate towards the black areas.

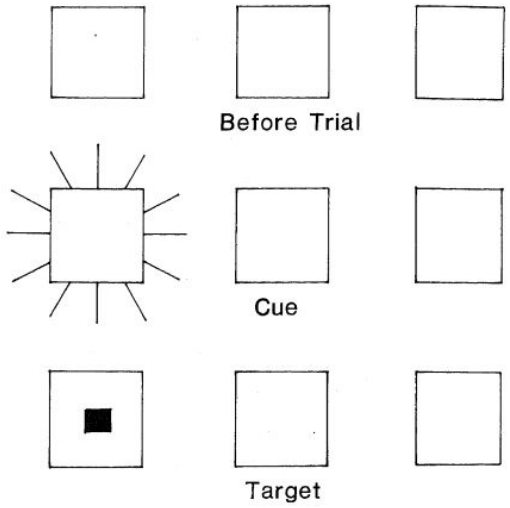
Bottom-up attention data are much easier to produce than top-down data. For this reason, scientists study bottom-up attention much more frequently than top-down attention. How these two processes interact is an area of emerging scientific interest.

Comment on the term "Attention": There has been sufficiently established criticism of the term. The main argument addresses the cyclical definitions used to describe what attention is. In particular, the synopsis of the arguments is that the term is used to "refer to both the explanandum (the set of phenomena in need of explanation) and the explanans (the set of processes doing the explaining)" [24]. Even early publications address the ambiguity of the term, considering it a "vague conceptualization" that is "difficult to falsify empirically" [33]. However, addressing these arguments and the questions that arise is outside the scope of this thesis. Although criticism exists and is established, we still chose to use the term due

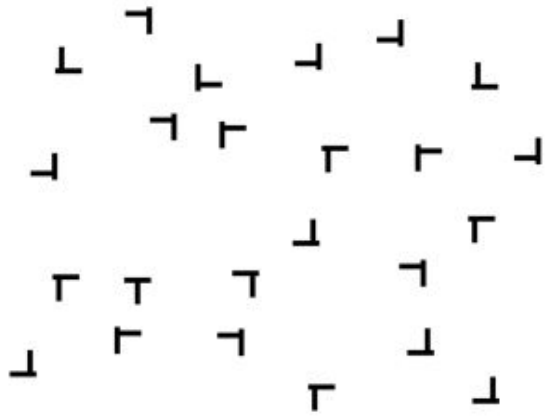
to its deep background and categorizations.

2.1.3 Observing Visual Attention in Humans

Forming a basic understanding of how visual attention is studied in an experimental setting can help us comprehend the reason visual models of attention are designed in a specific way. Usually, the insights are derived from controlled experiments of visual search tasks. During these experiments, scientists, using techniques like *Magnetic Resonance Imaging* that can detect changes in blood flow, oxygenation, and neuronal activity [76], attempt to identify the areas that are activated during the experimental process and spot possible patterns.



(a) A spatial cueing paradigm task. [75]



(b) A visual search task. The goal is to find the T among distractors. [73]

Figure 2.1: The two most common attention experiments.

The two most common types of experiments are the *spatial cueing paradigm* [74] in which a human is instructed to react when a stimulus is present in an image, and the *visual search task*, in which a human is instructed to detect a stimulus among distractors [58]. Below we present one of the most researched phenomena in visual attention to further expand the understanding of the reader on conclusions made from observations in an experimental setting.

Efficient Search versus Inefficient Search: After conducting experiments on visual search tasks, scientists observed that when a stimulus has a feature that is unique among other elements, for example, a red square among yellow squares, the search is relatively quick. The number of elements in an array of stimuli does not meaningfully influence the search time. This is called the *pop out* phenomenon or *efficient search*. However, when the target shares features with the distractors, search time is proportional to the number of distractors in the array. This phenomenon is called *inefficient search*. However, it would be better to imagine the above phenomena in a spectrum. Search time efficiency is directly related to the number and shared features with distractors. So a search can be "more efficient" and "less efficient" [17].

Visual Pathway related observations: In this part, we present some of the most common observations through the combination of visual search tasks and fMRI imaging. In particular, scientists observed that on bottom-up attention tasks, the V1 area was the major activated area [49, 59, 65, 99], and top-down attention is mainly reflected in the V2 area of the extrastriate cortex while other areas may contribute [18, 59].

Evidence for a hierarchical relationship has been presented by Melloni et al.[59]. In particular, the author states that hV4 "qualifies as the locus of convergence between saliency processing and top-down guidance". Before that, it was also considered an area that "harbors an overall saliency map" [59]. The term *saliency map* is a fundamental concept in our endeavor and will be explained in detail in following sections.

2.1.4 Modeling Visual Attention

We have briefly presented the way scientists study attention in an experimental setting the main categorizations and some insights. We did not get into extensive details, mainly because our purpose was to paint a picture to the reader of how observations are made in humans and primates and how they are construed and not to extensively explain the experimental observations and interpretations.

A *Model of Visual Attention* "addresses the observed and/or predicted behavior of human and non-human primate visual attention"[87]. So a model of visual attention aims to address the underlying biological and psychological phenomena observed in an experimental setting or make predictions based on the observed patterns.

However, it is not only the biology and psychology communities that are interested in studying the attention processes. A separate branch of research in attentive processes was created in the computer vision field before even the appearance of the biological branch [87]. Its main interest was to reduce the computational load of visual information to create more efficient machines.

The biological and computer vision branches intersect on computational models of attention. Computational models do not only describe observed or predicted behaviors. They also provide a testing mechanism by accepting image inputs so their performance can be compared with the performance of humans [87]. One of the first and most influential theories for the creation of computational models that emerged from the field of psychology was the *Feature Integration Theory (FIT)* [86].

The theory addressed the experimental observations of the time and was iteratively adapted and revised when new information were observed or old beliefs were proven false [17]. The synopsis of the theory was that "different features are registered early, automatically and in parallel across the visual field, while objects are identified separately and only at a later stage, which requires focused attention" [86].

Another pioneering model of attention for computational models was the Guided Search Model [90]. It aimed to improve and build upon the criticism made to FIT while explaining the observed phenomena in experiments [17] posing as "an alternative to the feature integration model of visual search" [90]. This led to a "competition" between the FIT and the Guided Search model which created more efficient explanations in subsequent publications of the authors [17]. It is worth mentioning that both of these models proposed a parallel computation of features and a combination in a fused representation which at a later time was

termed as a *saliency map* [17]. Saliency-based models are a main interest in our endeavor.

2.1.5 What is Visual Saliency?

Before attempting to present a formal definition of visual saliency, we will discuss some thought experiments. A human is placed in front of a black screen without given instructions on what to expect. Suddenly, a white circle pops out on the screen. His eye gaze naturally shifts toward the white circle almost instinctively. In another scenario, a human overwhelmed by thirst seeks water under the hot sun with no possible place to relieve his craving. His gaze starts to shift toward the hands of the bystanders holding water bottles who had the foresight not to be so inconsiderate of their needs.

In an empty scenery without apparent stimulants, it is almost certain that our gaze will naturally gravitate to the white circle. Similarly, when we have a task in mind -which in our example is to find water- our gaze will shift toward the water bottles much more frequently than usual. The white circle and the bottles are called the *salient* parts in our scenery and it is obvious by the descriptions we have already presented that saliency is affected by bottom-up and top-down mechanisms.

Let's now present a formal definition of visual saliency. "Visual salience (or visual saliency) is the distinct subjective perceptual quality which makes some items in the world stand out from their neighbors and immediately grab our attention." [26] The items that stand out in our examples are the white circle and the water bottle. A representation of the visual saliency of a scene is the *Saliency Map*.

To present a formal definition: "The Saliency Map is a topographically arranged map that represents visual saliency of a corresponding visual scene [62]." The saliency map is strongly connected with selective attention and especially bottom-up processes.

Saliency Map as a Covert attentional selection mechanism: The earliest use of the term was by Koch and Ulman [36]. They proposed that basic features like color, orientation, etc. are represented in the brain as different "elementary feature maps" [36] which have

higher and lower activity depending on the visual scene. The information from all these different maps was assumed to be merged in another topographical map termed the saliency map whose purpose was to represent a combined representation of the elementary feature maps. This representation was considered to be covert and computed at a pre-attentive stage. Although the term "saliency map" was proposed by Koch and Ulman the term "master map" proposed five years earlier by Treisman and Gelade [86] in FIT was very similar.

One of the first applied models using the term "saliency map", based on the paper of Koch and Ulman was by Niebur and Koch [63]. This model added the different activities of each feature map considering the same feature to have the same weight as the others except for temporal change. This was computed as five times more important claiming that changing stimuli are of greater significance to attention. The maximum of this map was considered the most salient region and was computed by a Winner-Take-All (WTA) mechanism. To represent the saccadic eye movements, a representation of sequential selections was created starting from the maximum salient point in the saliency map and after suppressing it, continuing to the second, third, etc. highest value. However, the model considered a breakthrough was a refinement of the base model [62] based on FIT by Itti et al.[27].

Saliency Map for Overt attentional prediction: As mentioned, a saliency map is the output of computational model of attention. The term saliency map was primarily used to describe a covert attentional process [41] meaning attention that was expressed through an "inner 'spotlight'" [28] and not through eye movements.

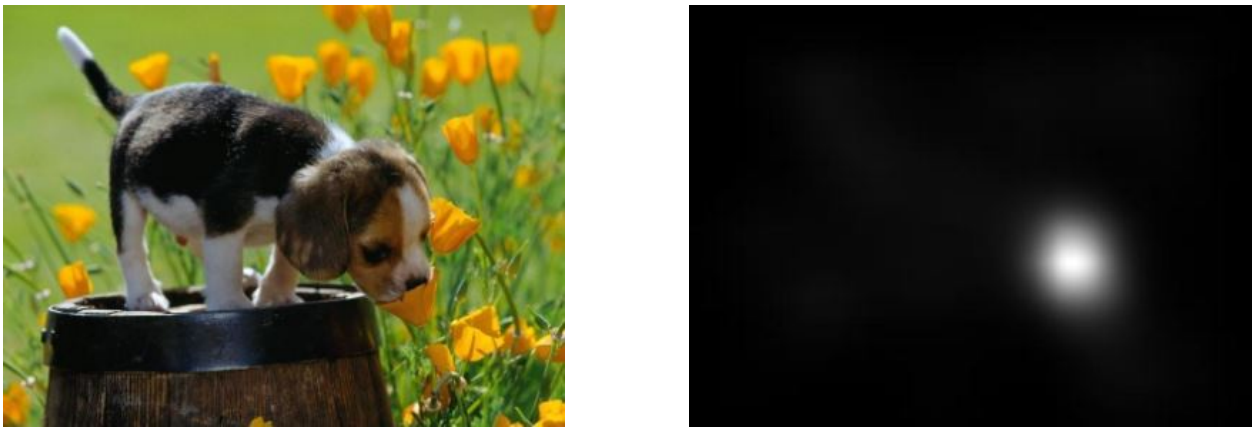


Figure 2.2: An image of a puppy (left) and the corresponding saliency map (right)[32].

This shift in the meaning of the term occurred after a connection with covert attention and eye-movement was established both through fMRI imaging [64], and by comparing covert model predictions with eye movements [71]. One of the driving factors was propositions like that of Itti and Koch [28] whose contributions in covert saliency computational models laid significant groundwork in the previous years [27, 36]. Therefore it is important to have an understanding of the different uses of the term. In computer vision, saliency is mostly used as a term when referring to fixation prediction while in vision science saliency is connected with FIT and the original computational models of covert attention prediction [41].

2.1.6 Saliency-Based Computational Models

Early models of visual attention used handcrafted features to predict the saliency levels in an image. Using the pre-established categorization of psychology, they were categorized based on whether they processed bottom-up or top-down information.

Early bottom-up models

Early bottom-up models usually extracted low-level features such as color, contrast, and texture [92]. One of the first and most influential bottom-up models was based on the *feature integration theory* (FIT) and was proposed by Itti et al. [27]. After this, several other feature-based models were proposed [1, 4, 12, 22, 53, 55, 57, 77, 94, 95]. For example,

Liu T and al. [53] extracted several features and then used a *conditional random field* (CRF) to combine them and make predictions. Zang and Sclaroff [95] proposed a *boolean map* approach. Achanta et al. [1] proposed a *frequency-tuning* model based on features of color and luminance. Besides the feature extraction methods, a notable mention is Bruce and Tsotsos [8] who proposed an *Attentional Information Maximization* (AIM) model based on information theory ideas.

Early top-down models

Early top-down models were more difficult to model due to their task-specific nature. The first proposed mechanism[67] used a *Bayesian rule-based* system to combine information from bottom-up and top-down saliency maps. After this, several Bayesian models were proposed [38, 91, 96]. Early machine-learning models, like Jud et al.[34] using *Support-Vector-Machines* are also notable because they served an introductory role to the subsequent *Deep Learning models*. [92]

The Deep Learning models

Advancements in Deep Learning [46] have enabled the creation of bottom-up and top-down models using Deep Learning technology. The first Deep Learning model [89] was created by Vig et al.. It was construed as an *ensemble of Deep-Networks* (eDN) and outperformed all the existing models on the MIT300 challenge [9].

Researchers then began to utilize the exceptional abilities of *pre-trained* models and *transfer learning* that had emerged the previous years. The first model was DeepGaze I [43] utilizing the pre-trained weights of AlexNet [39] a Deep Learning model trained on ImageNet [14]. Other models followed like *DeepFix* [40] utilizing *VGG-16* [83], and *DeepGaze II* [44] utilizing *VGG-19* [83].

Jetley et al. [30] followed a different approach by regarding saliency as a *generalized Bernoulli distribution*. By combining the Softmax activation function with "measures de-

signed to compute distances between probability distributions" [30], they created an effective loss function that demonstrated better results than typical loss functions. Other notable approaches combined *Convolutional Neural Networks* (CNN) with *Long Short-Term Memory* (LSTM) networks [13, 52], or even approaches such as *SalGan* [69] that utilized *Generative Adversarial Network* (GAN) [21] models.

In recent years, other approaches have been proposed based on the publications of the previous years. For example, *EML-net* [31] used an encoder-decoder architecture. The encoder contained more than one CNN models, which were separately optimized, then compressed, concatenated, and connected to a decoder for further optimization. Dodge et al. [15] used a mixture of experts. In their model, an image is passed through a pre-trained network and then the last layers are concatenated and fed into a category-specific network of experts. A gated network determines the weights for each of them and the final saliency map is a weighted sum of the expert saliency maps. Maldi et al. [56] described bottom-up and top-down features of convolutional networks as saliency maps and combined them to produce the resulting saliency map.

Last but not least, expanding on their previous work of DeepGaze I and DeepGaze II, Kummerer et al. proposed *DeepGaze III* [42]. A model with an architecture consisting of three parts. A spatial priority network, a scanpath network, and a fixation selection network.

2.2 Deep Learning Background

2.2.1 Artificial Neural Networks

An Artificial Neural Network (ANN) is a biologically inspired [46] machine learning algorithm and the most basic form of a deep-learning model. ANN's success lies in detecting non-linear relationships in the input data. Early linear classifiers were only able to divide their input space into "very simple regions" [46]. Thus, they approximated solutions to problems with data sets that were only linearly separable or became separable after hand-crafted feature extraction. Usually, creating hand-crafted features requires knowledge from

a domain expert [46]. ANNs can extract features automatically in tasks that require the representation of high-dimensional relationships creating a distinctive advantage over linear learners.

An ANN is a multilayered network consisting of an input layer, one or more hidden layers, and an output layer. The input layer receives the data based on which we want to make a prediction. The hidden layers are a series of adjustable values called weights or parameters that can be modified to make the prediction more accurate, and the output layer contains the final prediction of the model, usually "in the form of a vector of scores"[46]. The weights or parameters are inspired by "neurons" hence the term Artificial Neural Networks. They are responsible for modifying the input values to their respective output values.

However, we have not yet explained one of the most important aspects of an ANN. How are these weights adjusted? Before addressing this question in ANNs, we will first address it in linear learners because they serve as a simple introduction to the more complex nature of ANNs.

In a linear learner, an objective function measures the difference between the real and predicted outputs in an instance of training data or an average of multiple instances. Each weight in a weight vector is then adjusted by a very small amount and the new error is computed. This new resulting vector is called the gradient vector. If the error increases, the weights are adjusted in the opposite direction or else in the same.

One of the earliest methods that were used based on the above is called the *Stochastic Gradient Descent* which computes the average error over a few examples of the training data and then repeats the process to the rest reducing or "descending" the error bit by bit. During this process, subsets of the training dataset are used and not the whole dataset making the process "noisy" [46].

Linear learners performed their computations in parallel. The learning procedure between a shallow learner and a deep learner is really similar. However, the deep learner computes the gradients backward and in steps - from output to input - using a technique called back-

propagation. This technique is a "practical application of the chain rule" [46]. To understand this concept, one could imagine a neural network as a "chain combination" [20]. For example, the combination:

$$f(x) = f^{(3)}\left(f^{(2)}\left(f^{(1)}(x)\right)\right). \tag{2.1}$$

Would be a neural network and $f^{(1)}$, $f^{(2)}$, $f^{(3)}$ would be each layer of the network [20].

When an input value is given in this chain combination, an output is produced through a process which is called *forward propagation*. Then, using the output produced by the network and the expected value we calculate the loss of the network using a predefined *loss function* that suits our needs. A loss function is the way we compute the error between the predicted and expected values and can take different forms based on the problem we are studying. After the error is computed a process of adjustment of the chain combination by taking advantage of the chain rule. This process is called *back propagation*.

To dive even deeper in our explanation, the functions in our chain can be represented as different multi-dimensional vectors representing the weights. These vectors are mapped with one another as a chain combination with the vector closer to the input as the inner function. Each value in the vector can be though of as a neuron. The output of a neuron could be represented as [46]:

$$z_j = \sum_{i=1}^{n} w_{ji} x_i \tag{2.2}$$

- j is the dimension index of our neuron in a layer.
- i is the index of the different weights that are multiplied with the input value of the previous layer and then summed to get the final value.

So the output of our first layer consisting of m neurons would be represented as:

$$f^{(1)}(x) = \begin{bmatrix} z_1 \\ z_2 \\ \vdots \\ z_m \end{bmatrix} \tag{2.3}$$

Now the neurons of the second layers would be represented as:

$$q_l = \sum_{i=1}^{m} w_{li} z_i \tag{2.4}$$

Notice that the input value of the neurons are the sum of the weights $\times$ neuron values of the previous layer. And the new layer would be represented as:

$$f^{(2)}\left(f^{(1)}(x)\right)) = \begin{bmatrix} q_1 \\ q_2 \\ \vdots \\ q_k \end{bmatrix} \tag{2.5}$$

Now the pattern becomes obvious.

These however does not explain how the ANN is able to capture non-linear relationships as what has been described are all linear transformations.

This is achieved through the use of activation functions. An activation function is a non-linear function. The most commonly used activation function is Rectified Linear Unit (ReLU) which is defined as $a(x) = \max(0, x)$. Now the new approximator $f(x)$ takes the form:

$$f(x) = a\left(f^{(3)}\left(a\left(f^{(2)}\left(a\left(f^{(1)}(x)\right)\right)\right)\right)\right). \tag{2.6}$$

The addition of the activation functions forces the network to adjust its weights after passing them through the activation functions which effectively adds non-linearity.

2.2.2 Convolutional Neural Networks

Convolutional neural networks(CNNs) [45, 47] "is a specialized kind of neural network for processing data that has a known grid-like topology" [20]. They have proven to be especially useful for image-processing tasks due to the grid-like structure of images. The name of these networks is derived from a mathematical operation known as *convolution*, which is used to replace general matrix multiplication within a layer. In a strict mathematical sense, however, the operation performed by most code libraries is not convolution but *cross-correlation* [20].

The difference between convolution and cross-correlation is that in convolution a flip of one of the functions involved is needed making the order in which you apply the operations irrelevant (*associativity*) [20]. However, in CNNs, we don't need the associative property and thus it makes the flipping irrelevant. Hence, conventionally, rather than being called cross-correlational neural networks, they were named convolutional.

Convolutional Neural Networks Architecture

Convolutional neural networks have three main layers: *convolution layers*, *pooling layers*, and *fully-connected layers* [68]. The first layer is usually the typical input layer which can be also found in ANNs. Then a convolution layer is introduced:

Convolution Layer: This is the main difference from a typical ANN. They introduce the idea of computing the convolution operation between the input layer and *learnable kernels* [68]. A learnable kernel in our context is a matrix that slides across the input grid, computing a dot product with the local regions that are learned by the training algorithm. The result of the sliding is a smaller representation combining information from each local region. Essential parameters for this procedure are *stride* and *padding* which control the size of the output. Stride refers to the step size of the kernel when sliding across the input. A bigger step results in a smaller representation. Padding refers to the addition of pixels with zero value in the edges. This ensures that the image can be thoroughly covered by a two-dimensional filter.

Each convolution layer uses multiple filters to create different feature maps. The formula for convolution, as presented below [21], expresses in mathematical terms how these kernels are applied to local regions of the input to generate a new grid.

$$Y[i,j] = \sum_m \sum_n X[i-m, j-n] \cdot K[m,n] \tag{2.7}$$

In this formula:

- $Y[i,j]$ is the output value at position (i,j) of the grid called the feature map.
- $X[i-m, j-n]$ is the input value which is based on the position (m,n) of the kernel.
- $K[m,n]$ is the kernel's value at position (m,n). The kernel is not flipped.

What are the advantages of Convolution: Three advantages of using a CNN are: "sparse interactions, parameter sharing, and equivariant representations" [20].

Sparse Interactions: In an ANN each input in a layer is connected with every output. The way this is represented is a matrix multiplication between the input and output values. However, by using the convolution operation, the number of parameters is drastically reduced.

For example, if we have 200 inputs connected to 200 outputs the resulting matrix needs 200x200 parameters to perform the multiplication in a typical ANN. However, the resulting matrix is smaller by making the interaction sparse, meaning, reducing the number of connections with an output. If we reduce the number of connections to an output to 5, the number of parameters needed is 5x100. This is achieved due to the difference in size between the input and the learnable kernels.

The loss of information resulting from the fact that each input is no longer connected with each output is partially compensated by faster running time, resistance against overfitting, and the CNNs' receptive field. The receptive field is closely tied to the depth of a CNN. A

higher depth allows for indirect communication of the sparse connections that were previously disconnected.

Parameter Sharing: In a typical ANN, the elements of a weight matrix are used only once. With the use of a sliding, learnable kernel, each value of the kernel is applied across the entire input. This allows the CNN to capture multi-dimensional relationships. For two-dimensional inputs specifically, the way the CNN captures information is much closer to the human visual system than the typical ANN. In fact, "the overall architecture is reminiscent of the LGN–V1–V2–V4–IT hierarchy in the visual cortex ventral pathway" [68]. The result is a significant improvement in the ability of neural networks in human-like interpretations of grid-like inputs such as images.

Equivariant Representations: Parameter sharing allows the network to be equivariant to translation, meaning that when a transformation is applied to an input, a corresponding transformation is also applied to the output. Practically this means that the network can recognize features irrespective of their location in the grid.

Calculating the Output Size for $N \times N$ Inputs in Convolution Layers For better understanding, we also present the formula [68] that can calculate the output size for an input of size $N \times N \times D_{\text{in}}$ as :

$$N_{\text{out}} = \frac{N_{\text{in}} + 2P - F}{S} + 1 \tag{2.8}$$

In this equation N_{out} is the size of the $N \times N$ output grid. N_{in} is the size of the $N \times N$ input grid. P is the padding applied to the input. F is the size of the filter or kernel. S is the stride step of the convolution operation. D_{in} and D_{out} is the number of input kernels and output kernels respectively.

Pooling Layer: CNN layers have multiple filters. Each filter produces an activation map. The more filters a layer has the more dimensions are in the depth of the resulting

activation map. Pooling layers can reduce the width and height of the resulting activation maps to reduce the number of parameters and improve running time. In a similar way to the kernels in convolution, the pooling filters are applied by sliding the filter across the activation map. The data-reducing operations that are mostly used in this process are a max (Figure 2.3) or an average.

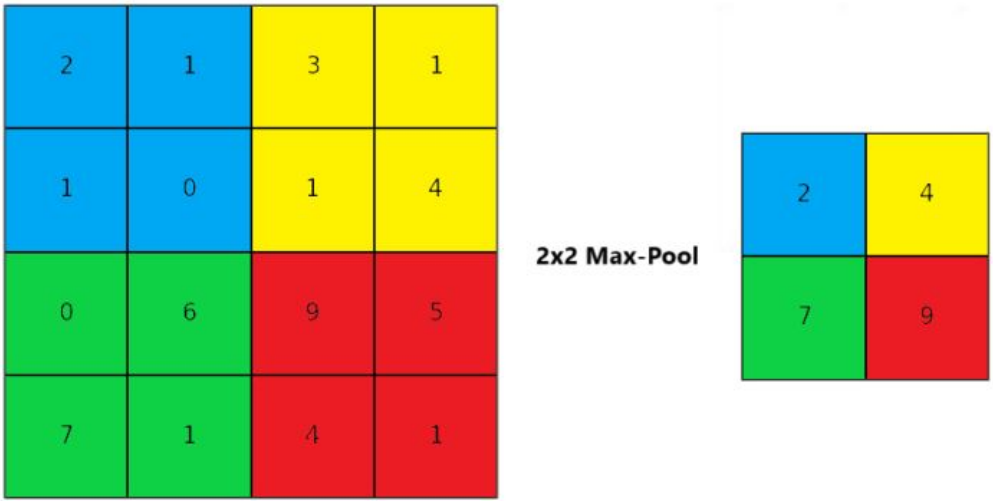


Figure 2.3: Before and after max-pooling with a stride of 2.

Fully Connected Layer: The outputs from the successive convolutional and pooling layers terminate in a fully connected layer, the typical layer of an ANN. This layer is usually combined with a Sigmoid or a Softmax function. These two functions can translate the outputs of the fully connected layer in an interpretable format. In detail, the Sigmoid translates all the outputs into values between 0 and 1, so it is mostly used for binary classification. On the other hand, the Softmax function can translate the output vector into a probability distribution and can be used for multi-class classification.

2.2.3 Autoencoders

An autoencoder is a neural network structure that can employ both fully connected layers and convolution layers. It was inspired to solve problems in an unsupervised way [3]. Its main idea lies in distorting the input data and then recreating them in an altered form. This new representation can provide us with insights into the original data. So the original goals

of the autoencoder as a neural network structure were dimensionality reduction and feature extraction [20].

The autoencoder structure consists of two main parts, the encoder and the decoder. The encoder part is responsible for encoding the input information into a lower-dimensional representation and the decoder is responsible for reconstructing the lower-dimensional representation into a new output. However, the goal is to create a representation in an altered form from the input. The reason is that simply reconstructing the input would not provide us with meaningful information [20].

The structure of an autoencoder looks like an hourglass. Again it consists of an input layer, one or multiple hidden layers, and an output layer. The layers become smaller in size towards the end of the encoder with the last layer named the bottleneck. The bottleneck is the layer that marks the end of the encoder and the start of the decoder. After the bottleneck, the layers start to increase in size. This gradual increase enables the decoder to reconstruct the latent representation of the bottleneck.

The reason for this decrease in dimensionality and the subsequent increase is to achieve *regularization* [60]. This allows the network to not simply output the input image but to also learn the most "salient" [20] properties based on the task. If this decrease didn't occur the network structure would be "overcomplete" [20] and would overfit the input data without learning anything useful.

Image Denoising: One of the most well-known problems that use autoencoders as a possible solution is image denoising. In this problem, the goal is to teach a learner to recreate the original image given an image that has noise added to it. To tackle this problem the noisy image is given as an input to the autoencoder. The autoencoder then performs the forward pass and outputs the prediction of the real image. Then the loss is calculated between the output image and the real image with no noise. The most common loss function

used for this problem is usually *Mean Squared Error (MSE)*. The formula of MSE is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

For the denoising task:

- n is the number of pixels in the image.
- y_i is the real pixel value.
- $\hat{y}_i$ is the predicted pixel value.

The computed loss is then back-propagated in the network and the process continues to the next image until the network can denoise the images effectively.

Image Segmentation - U-Net: One of the most prominent and successful uses of the autoencoder is in the task of image segmentation. Specifically, an autoencoder is utilized with skip connections between the encoder and the decoder. A skip connection is a concatenation of every step in the encoder with the corresponding layer in the decoder [80]. This approach significantly increased the accuracy of the models that were used. Until then, the most successful approaches utilized fully convolution layers with typical CNN architectures such as Alex-Net, VGG Net, or Google Net [54].

By adding to the decreasing-in-size, fully convolution layers, an up-scaling counterpart with skip connections, they created an architecture [80] that can make a per-pixel classification output. This *U-Net* architecture significantly outperformed previous attempts in several tasks of medical image segmentation [80].

Improving the U-Net architecture - Adding Residual Connections One of the major problems that scientists encounter when training very deep neural networks is the problem of vanishing or exploding gradients [19]. This is a degradation problem that occurs much more frequently when the layers of a network increase significantly. This problem can

be addressed by an architectural modification called *residual connections* [23]. The original input is feed-forwarded in the stacked layers. The mapping learned from these layers is denoted as $\mathcal{F}(x)$. Then utilizing a skip connection that performs an identity mapping, in which the original input values x are simply returned as is, we add the original input to $\mathcal{F}(x)$. So the final output of a residual layer is $\mathcal{F}(x) + x$. He et al. [23] showed that a network that utilized stacks of residual layers was easier to optimize than a network with stacked convolution layers especially when depth increased.

Zhang et al. utilized the idea of residual layers and modified the U-Net architecture to incorporate it. This allowed them to outperform "U-Net with only 1/4 of its parameters" [98] in a road extraction task.

Further Improving the U-Net architecture - Incorporating Attention Gates: CNNs can capture relationships of a grid by a down-sampling procedure that enables a model to understand global connections in an image. However, it is difficult for such a model to capture "small objects that show large shape variability" [66] resulting in an increased false positive rate in a segmentation task. To address this problem, *attention gates* (AG) were proposed [66]. These attention gates enable a network to suppress unnecessary background noise and significantly improve localization.

The incorporation of attention gates was inspired by *sentence embedding learning* [81]. In particular, Oktay et al. [66] use attention coefficients $\alpha_i \in [0, 1]$ multiplied with the values of an input feature map in an element-wise fashion. This computes a single scalar value for each pixel and all the scalar values constitute the output of an attention gate. This AG enables the model to focus on smaller and more complex objects and areas.

In detail, as we can see in 2.4 the input tensors are linearly transformed using a 1x1x1 channel-wise convolution. Then, the addition operation between the two outputs is performed which then passes through a ReLU activation function to add non-linearity. Then, the output of the activation function is again linearly transformed and given as an input to

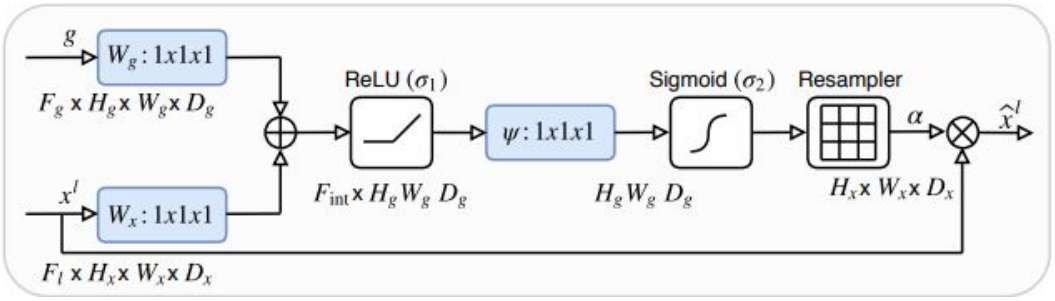


Figure 2.4: The internal process of an attention gate taken from [66]. Licensed under Creative Commons Attribution 4.0.

a sigmoid function. Finally, the output of this process is multiplied by the original input. In the U-Net context, the gating g is applied to the decoder and the original input x is from the skip connections between the encoder layers and the decoder.

2.2.4 Generative Adversarial Networks

A *Generative Adversarial Network* [21] is a combination of neural networks that are "simultaneously trained" [21] in a competitive or "adversarial" [21] way which revolutionized the image generation domain. In the original paper [21], Goodfellow et al. compare this process with an interaction between counterfeiters and policemen. The generative model is the counterfeiter trying to produce images that will fool the discriminator which has the policing role. Through this rivalry, the generator and discriminator are driven to improve their outputs reaching a point where their output may look realistic to a human.

In the original work, the Generator and Discriminator are represented by two neural networks. The interaction between the two is described as a minmax game. This means that one player aims to minimize a value while another maximizes it. In our context, the generator aims to minimize the times the discriminator makes a mistake while the discriminator aims to minimize their mistakes between fake and real outputs.

In detail, the goal of the Discriminator is to maximize the correctness of the value $\log(D(x))$ which is the logarithm probability that an input x is real rather than an output of the Generator. The Generator receives noise z as input from a noise distribution $p_z(z)$ and aims to rearrange the inputs in such a way that it fools the discriminator. This

means that we train the Generator to "minimize $\log\left(1 - D\left(G\left(z\right)\right)\right)$" [21]. The minmax game can be summarized in an equation taken from [21]:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}\left[\log D(x)\right] + \mathbb{E}_{z\sim p_z(z)}\left[\log(1 - D(G(z)))\right]. \tag{2.10}$$

However, in practice, the generator performs much better if instead of trying to minimize $\log\left(1 - D\left(G\left(z\right)\right)\right)$ it aims to maximize $\log D(G(z))$. This is because in the original definition, the Discriminator can distinguish the images with great accuracy in the first iterations which leads to saturation [21]. The new definition provides the Generator with an attainable goal, especially in the early training of the model while keeping intact the dynamic interplay between the agents.

Conditional Generative Adversarial Network In the original paper on GANs, Goodfellow et al. propose a different definition that would make the GAN conditional. They state that "A *conditional generative* model $p(x|c)$ can be obtained by adding c as input to both G and D". The actualization of this idea was created by Mizra et al.. In their implementation, an extra input layer is used for the Generator and the Discriminator that represents the addition of c. They state that c (y using their notation) "could be any kind of auxiliary information, such as class labels or data from other modalities" [61]. The combination of input noise with this extra information such as the class label, allows the Generator to output a diversity of images that can be easily conditioned by the user.

Image-to-image translation with GANs Early implementations [48, 72] of image-to-image translation utilizing GANs were not conditional. They utilized the structure of regular GANs relying on a loss to ensure the translation of the image from one domain to another. This loss is usually the mean squared error loss (called L2 loss) that would be slightly adapted based on the particular task. In this process, it is very common to utilize an autoencoder as

the Generator. The use of the mean squared error loss in the image-to-image task reminds the use of the aforementioned in the denoising task of autoencoders.

The GAN in an image-to-image translation task does not need the addition of noise, as the goal now is not to generate new, different images from scratch that look like the originals. Using noise would mean that the aim is to achieve different outputs which doesn't serve a purpose in an image-to-image task.

The first conditional implementation of image-to-image translation utilizing GANs was by Isola et al. [25]. The condition was the addition of the real image in the input of the generator and discriminator. They utilized a U-Net [80] architecture for the generator part and a PatchGan architecture for the discriminator part.

A typical discriminator should be able to discern what image is counterfeit and which is not. However, until then, discriminators produced only one classification judgment for the entire image. Contrary to that, the PatchGan architecture classifies patches of the image and produces the final decision after producing a decision for each patch separately and then combining these decisions for a final result. This allows the discriminator to focus on local areas and forces the generator to produce finer details.

Conditional GAN architectures similar to the implementation of Isola et al. have been utilized to unravel relationships between visual tasks [93]. Intuitively, we know that some visual tasks are more similar than others. However, the first systematic way to measure this similarity was proposed by Zamir et al. [93]. By utilizing transfer learning scenarios and conditional GANs, they created the most extensive taxonomy among visual tasks until then.

2.2.5 Transfer Learning

Transfer learning in the real world is the process in which a person's general experience can be applied to many different situations. For example, if one has taken advanced mathematical lectures at the university, he is more likely to understand theoretical concepts in physics. Similarly, if one can play the violin at a professional level, he is also more likely to learn the piano faster than a beginner [100] even if there is no difference in time spent study-

ing. As we can deduce, there needs to be a connection between the two learning procedures and common learning domains or tasks for a person to gain an easily observable advantage in the learning process. We shouldn't expect that learning to swim will give us a significant advantage in understanding advanced theoretical mathematical concepts.

In deep learning, transfer learning is a process that allows the training of one model to be used as a starting point for another model. Usually, it is used when there are not enough labeled data for a new task while there are plenty in a similar one. This allows the model to require much less task-specific training data for the new task. However, the two tasks must be significantly similar or even the same, else there could be no positive gain from the knowledge transfer, and even in some cases, the new model could be influenced negatively [97]. Below we present a definition taken from [70]:

Transfer Learning: *Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target predictive function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$.*

This definition implies that transfer learning can occur even if only the tasks or the domains are different without needing both to occur simultaneously even though it is possible. However, in most practical uses in deep learning tasks, usually $D_S \neq D_T$ [70]. The term domain in this definition consists of a feature space and the marginal probability distribution for that feature space, and the term task refers to the possible labels a learning sample can take in combination with the objective function. This is the function that we want to approximate.

Measuring the success of transfer learning through fine-tuning:

Advances in deep learning and particularly research in the abilities of transfer learning have reached a point where the success or failure of a transfer learning task can be measured

experimentally. The generalization ability of the base model to the new domain or the "transferability" measure that has been developed in the past few years, "can be categorized into three groups: feature statistics based, test performance based, and finetuning based " [97].

The feature statistics-based measurement relies on metrics like *maximum mean discrepancy* (MMD) and *KL-divergence*, and the performance-based measurement measures the success of a classifier trained in one domain when applied to another [97]. However, these two measurements would compute an expected transferability measure for our case. To accurately capture the transferability of a source model to another domain, the best way is to utilize the last technique which evaluates actual fine-tuning performance.

This technique involves training or using a pre-trained base model and fine-tuning the base model to a different task. This technique uses performance metrics like the model's accuracy in the new task to measure the similarity between the two domains, called "transferability" from one domain to another.

The pioneering work that used fine-tuning to measure transferability was from Zamir et al. [93]. They created a taxonomy of vision models based on their pre-training and success in subsequent fine-tuning. Their goal was to unveil an underlying structure [93] that connected basic vision tasks to reduce the need for labeled data. Subsequent publications like Achille et al. [2] tried to create a more efficient measurement of taxonomic distance named *Task2Vec* which we will not explain in detail. However, based on the work of Zamir et al, in their work, they provide us with a metric of transfer distance between two tasks which is a great measure to showcase the transferability from one task to another. We present the definition below verbatim [2]:

Transfer distance: *We define the transfer (or fine-tuning) gain from a task t_a to a task t_b (which we improperly call distance, but is not necessarily symmetric or positive) as the difference in expected performance between a model trained for task t_b from a fixed initialization (random or pretrained), and the performance of a model fine-tuned for task t_b starting from*

a solution of task t_a:

$$D_{ft}(t_a \to t_b) = \frac{E[\epsilon_{a\to b}] - E[\epsilon_b]}{E[\epsilon_b]},$$

where the expectations are taken over all training with the selected architecture, training procedure, and network initialization, ϵ_b *is the final test error obtained by training on task b from the chosen initialization, and* $\epsilon_{a\to b}$ *is the error obtained instead when starting from a solution to task a and then fine-tuning (with the selected procedure) on task* t_b.

In simple words, this definition provides a measure of the benefit of training in a source task t_a and then retraining on another target task t_b, against training in the target task t_b from scratch. A positive number indicates a benefit while a negative indicates that there is a negative transfer between the two tasks.

CHAPTER 3

EXPERIMENT DEFINITION AND METHODOLOGY

In our definition of the experiment, two agents are trained. The first agent is trained on fixation maps under free-viewing conditions representing the attention mechanism of a human and the second task-specific agent is trained to become optimized in a segmentation task. The first fixation agent is then retrained in the same task as the second agent. Then, common tasks are introduced between the two agents.

The fixation maps are expected to make the first agent behave in a human-like way and the second agent as a task-optimized mechanism. This means:

- The first agent will adapt quickly to the common tasks using a small or moderate-sized dataset regardless of their similarity or dissimilarity.
- The first agent, trained on human fixation maps, is expected to be more adaptable across tasks than the task-specific showcasing generalization ability, a human characteristic.
- The second agent will be adaptable to similar tasks and could be equal or better in performance than the first agent. However, as the tasks get more dissimilar his performance is expected to decline in contrast to the first agent.

In detail, to achieve the training of the agents would be to create a model that given an input or a pair of inputs can output a segmentation map that represents the target of our task. In the above definition, this target would be a segmentation map of the goal. This model will be a pixel-to-pixel [93] classification else called a segmentation model that will

divide the input image into two regions. The first region represented with black pixels would be the area that the model would have no interest in and the region represented with white pixels would be the area that represents the understanding of the model in the goal.

For the retraining of the model, transfer learning is utilized. In our context, we have a case of *inductive transfer learning* [70], which refers to a scenario where the source and target tasks are different, but there is labeled data available for the target task. The process we are going to follow will be based on the influential paper for transfer learning of Zamir et al. [93].

In their approach, models are trained from scratch for several visual tasks with a moderate amount of pictures. Then retraining is applied from the base models to all other tasks. They also compare their results with a task-optimized mechanism. In their results, we can observe that there can be significant improvements in the models using a source task and then retraining when contrasted to models that do not.

There are also clear levels of improvement and different efficiency using different source tasks which indicate a better level of transferability between tasks and a clear difference in taxonomy. This indicates that their approach of comparing accuracy with and without pre-training, might yield interpretable results to our problem as well.

3.1 Datasets

All of the datasets used for the different agents are based on the COCO dataset. This is one of the most extensive datasets labeled using "per-instance segmentation" [50] with more than 91 object categories and 328k images.

Segmentation Datasets: The segmentation datasets are constructed from the different categories of the Microsoft COCO dataset [50]. The COCO dataset provides several categories that may become increasingly dissimilar. In our case, we chose three progressively dissimilar animals—dogs, giraffes, and birds—based on the animal taxonomy established by

Carl Linnaeus. This classification system, described in his tenth edition of Systema Naturae [51] in 1758, constitutes a golden standard generally accepted until today, albeit with new scientific discoveries incorporated.

To further explain, this animal taxonomy consists of seven basic ranks created based on similar physical characteristics. Therefore, the visual appearance of the animals is analogous to the rank they belong to. These ranks are Kingdom, Phylum, Class, Order, Family, Genus, and Species [11]. Dogs and giraffes are mammals and belong to the same Kingdom, Phylum, and Class. However, birds belong to the same Kingdom and Phylum as dogs and giraffes but in a different Class [84]. Therefore, based on this taxonomy, it is expected that the visual appearance of birds has a higher distance from giraffes than the visual appearance of dogs. Furthermore, it should be noted that the bird category has more branches in the sub-categories when contrasted to the other two tasks, meaning that their Order, Family and Genus varies depending on the bird Species. This implies that this task is not only more dissimilar, but could also prove to be harder due to higher diversity.

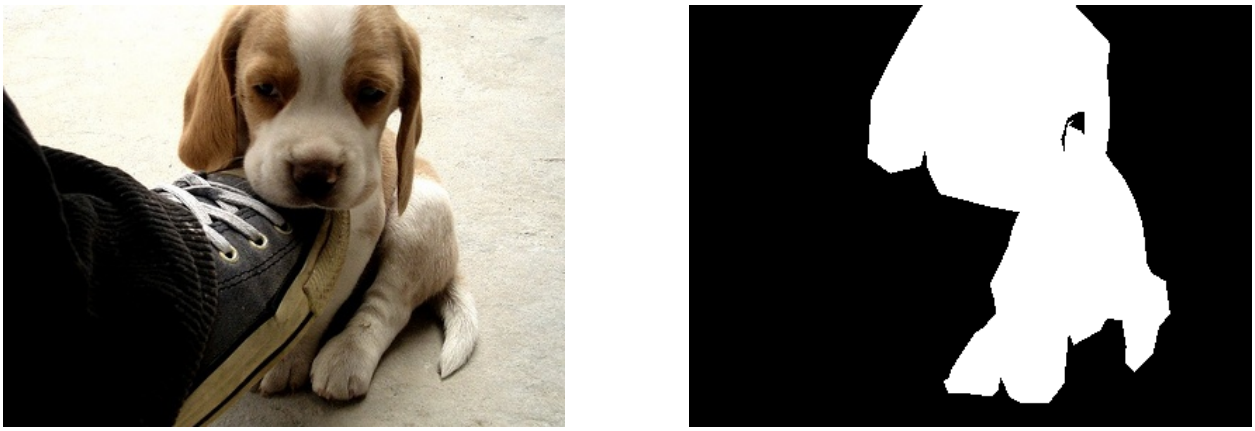
Fixation Dataset: For the bottom-up fixation model, we utilized Salicon's[32] training dataset. Salicon's training dataset is constructed from a selection (10k) of images from the Microsoft COCO dataset. This selection is made from 80 out of the 91 categories of the original COCO dataset, including the categories we will utilize in our training, as well as other very similar categories to ours, like cats and zebras, and very different categories like clocks and chairs. This creates an extensive training dataset that aims to be representative of the general free-viewing behaviors humans exhibit in everyday life.

The Salicon dataset consists of pairs of images from real-world images and fixation maps. However, the fixation maps are not constructed with an eye-tracker, which was the typical way to construct such datasets until then, but by mouse clicks of humans observing the real-world images. Scientists showcased [32] that there is a sufficient connection between eye fixation and mouse clicks so they replaced the eye tracker with a procedure that was much

easier to perform. This allowed them one of the most extensive eye-fixation datasets.

For our training needs, we utilized the entire Salicon dataset. This ensures that our fixation agent will be presented with a diverse set of images, including those relevant to our experiment. Since the Salicon dataset contains many different animal categories, including ours, we expect a well-crafted model to exhibit high accuracy in these categories and therefore to successfully mirror human free-viewing behaviour in them.

Preprocessing the datasets: There were several other category combinations that could follow the criteria of dissimilarity. However, there are not many categories with at least 2k images. The three categories we picked had enough images to set up our experiment. After cropping the images to be of similar size, we picked the 2k images with the highest percentage of pixels that belong to the goal. This helps us to partially avoid the cases where the conditional GAN would overfit the negative class resulting in a generated image with only black pixels. After modification, we can achieve the below representation 3.1 for each category.



(a) The original image (b) The segmentation map of the goal

Figure 3.1: Representation of the original and segmented image of a dog.

Then we split the resulting datasets to create a testing dataset that contains 100 images. Due to the nature of our experiment, we won't be focusing on hyper-parameter searching or model-tuning, so a validation dataset is not required.

3.2 Model

Combining ideas from SalGan [69] that used a GAN to predict visual fixations and [93] that used a conditional GAN to predict visual relationships through transfer learning we chose to perform the segmentation with a conditional GAN. As mentioned, the original idea was proposed in the paper of Mirza et al.[61] with one of the most famous implementations being the pix2pix GAN [25]. However, we did not use this exact architecture. We created a new architecture similar to the architecture by Zamir et al. [93] that uses residual connections.

For the generator, we employed a U-Net architecture incorporating residual connections and an attention mechanism. For the discriminator, we employed a typical Patch-GAN architecture. In detail:

Generator: The main building part of the generator that replaces the simple convolution layers is a structure that applies a series of residual blocks repeated n times. We found that this repetition of residual blocks increased the model accuracy, especially in the harder tasks like bird segmentation. Our residual block contains two convolution layers and the subsequent normalization and activation function before adding input and output. Before the residual blocks, in our structure, a 1x1 convolution layer is applied that increases the number of filters. Between, this convolutional and the residual layers, no normalization or activation function is applied to decrease training time. This structure along with max pooling layers is progressively applied, reducing the spatial dimensions while increasing the filters. The filters produced are:

Encoder Filters: 64 - 128 - 256 - 512 - 1024

The output of the encoder is fed to a transposed convolution layer reducing the filters by half while increasing the spatial dimension. With this upsampled feature map, the concatenation with the layers having the same number of filters from the encoder begins producing

the first skip connection. After this concatenation of the feature map (512 filters) with the encoder layer (512 filters), the resulting map has 1024 filters. The attention mechanism is applied in the skip connection, meaning between the encoder layer and the upsampled feature map before the concatenation. Also, instances of custom pixel padding are applied to make sure that the concatenated layers are of the same size. The pattern continues producing the below filters and concatenations:

Decoder Filters: 512+512 - 256+256 - 128+128 - 64+64

This produces a layer with 128 filters. This layer is fed into a last transposed convolution layer which reduces it to 64 filters. Then a final convolution is applied with 3 filters representing the RGB values utilizing a tanh activation function and the final output is created.

Discriminator: The discriminator architecture is simpler than the generator. The real image with the output of the generator is concatenated and fed as input to the discriminator. Then, a series of five convolution layers ensue resulting in a single channel output with each value in the output representing a patch in the concatenated image grid. In detail, the filters produced are:

Discriminator filters: 64 - 128 - 256 - 512 - 1024

As mentioned, the PatchGAN discriminator is essential to achieve a good representation due to its localization abilities achieved by examining several different patches separately instead of an image as a whole.

3.3 Observability of the experiment

3.3.1 Measuring the success of our saliency model

To understand if the training of our model is a success, first, we need to establish our needs. As showcased, the most successful fixation prediction models utilized some form of

transfer learning from task-specific scenarios. However, this cannot be our case since this will introduce bias to the comparing process of the agents. Our agent should be as pure as possible, trained only with fixation maps. This means that we should not expect our fixation model to achieve top-notch accuracy in its predictions. Nevertheless, we need a model that is accurate enough and has a significant fixation prediction ability.

Many metrics can showcase this capability. However, since our purpose is not to extensively compare the model's accuracy with that of the top-performing models, we will use a single metric that is both simple and efficient. This metric is called *Judd-Area Under the Curve* (AUC) [34] and is based on the Area Under the ROC Curve metric [16]. It measures the "tradeoff between true and false positives at various discrimination thresholds" [10].

To use this metric, the saliency map is converted to a binary map for different threshold values. The pixels with values above this threshold are classified as positive while the others as negative. For each threshold, the true positive rate and false positive rate are computed. A true positive pixel is a positive pixel that agrees with the pixel of the fixation map while a false positive pixel is a positive pixel that does not. To calculate the rate, we divide these numbers by the total amount of positive or negative pixels respectively [10]. Then a curve is created representing the contrast of these rates. Specifically, the true positive rate against the false positive rate at various threshold levels.

3.3.2 Measuring the success of our segmentation models

Many metrics could be considered when measuring the success of a segmentation model. The first thing that comes to mind would be to measure how many pixels from the generated image match the ground truth and then divide them by the total number of pixels. However, this metric comes with many limitations, especially if imbalanced classes exist which is also our case. For example, when we are trying to segment birds, the model can overfit the data and generate only black images. This is because the majority of the image is black. However, the score mentioned above would be very high in this case as the white pixels are much less than the black pixels. Yet our model would have completely failed to achieve our goal which

is to label the white part of the image where birds exist.

To overcome this problem another metric is used called *Intersection over Union* (IoU). This metric not only takes into account the true positive (TP) pixels but also the false positives (FP) and false negatives (FP). Although this technique is usually used for bounding boxes [78], it is common and valid to use this metric for pixel-to-pixel segmentation tasks [25, 54]. In this measurement, a logical AND is computed between the true mask and the predicted mask representing the true positives, and also a logical OR representing the sum of true positives, false positives, and false negatives. In detail:

$$\text{IoU} = \frac{\text{Intersection (TP)}}{\text{Union (TP + FP + FN)}} \tag{3.1}$$

If there were multiple classes, we would compute the IoU for all of them and then calculate an average. However, in our case, we are dealing with a binary task of black-and-white images. The black part of the image does not convey meaningful information, while also dominating the image in many cases. Adding it to our calculation would lead to a misrepresentation of the result. So we choose to discard it and use the IoU metric for the white class only.

3.3.3 Measuring the success of agents

As mentioned, to measure transfer learning success scientists have utilized transfer distance. However, based on our experiment structure, there is no one-to-one correspondence with transfer distance in all comparable scenarios of our agents. Nevertheless, by utilizing a similar logic, in most scenarios, we are going to compare the maximum accuracy achieved in the test set of models with different pre-trainings and the same goal task. Transfer distance though will still be useful for us in some scenarios where we compare models with and without pre-trainings. The metric that we are going to compare the models on is the IoU metric.

Moreover, to make the comparison even more meaningful, we will compare the models

with different pre-trainings across different dataset modifications. Meaning, we will observe the behavior of the agents when utilizing different dataset sizes. We consider this crucial, as for an agent to adapt to a task with a smaller amount of images than another, is significant evidence for the adaptability comparison that we will perform.

CHAPTER 4

RESULTS

4.1 Experimental evaluation of the hypothesis

Establishing a baseline: To meaningfully interpret our results and ensure the experiment proceeds as expected, we must establish a baseline against which our agents will be compared. In this baseline, we utilized the three animal datasets that we prepared. We used the hyperparameters shown in Table 4.1. These are typical hyperparameters for the domain translation task utilizing conditional GANs. For uniformity, we applied them in all of our trainings. The base models were trained for 20 epochs for different dataset sizes. When retraining to another task, all the encoder layers of the U-Net architecture were frozen. The values reported are the maximum values achieved in the test dataset during these 20 epochs.

Hyperparameter	**Value**
Learning Rate	0.0002
Beta	0.9
Loss Function	Binary Crossentropy

Table 4.1: Hyperparameters of our experiments

After examining the results, some key observations can be made:

- The easiest segmentation task to perform, is the giraffe task, while the hardest is the bird task. Both by a large margin.

Dataset Size	Dog	Giraffe	Bird
250	0,49	0,63	0,22
500	0,47	0,62	0,24
1000	0,52	0,65	0,22
1900	0,53	0,66	0,23

Table 4.2: IoU values for the base model segmentation performance with different dataset sizes.

- The ease of the task correlates with the size of the animal. This could be attributed to a higher presence of black pixels in the images or to the fact that the bird task is more diverse.
- The model of the bird segmentation consistently achieves the lowest scores indicating significant difficulties in training.
- For the dog and giraffe categories, there is an obvious correlation with performance and dataset size increase. For the bird category however dataset size is less significant. We can observe that utilizing the dataset size of 1900 increases performance when contrasted to the 1000 dataset size. Surprisingly though, the best performance can be observed when utilizing the 500 dataset size.

Evaluating our fixation model: After having established these baseline models, we are going to make sure that the fixation model is performing as well as expected. We trained the model for 50 epochs utilizing the training dataset of Salicon. As mentioned, the metric that we are going to use is the AUC-Judd metric. After getting an average for all the different thresholds and for 1000 test images, the score achieved was **0.79** (Table 4.3).

Epochs	AUC - Judd
50	**0.79**

Table 4.3: AUC-Judd score for the fixation model trained for 50 epochs

We deem this score significant enough for a fixation model. This is because the baseline for such a model is 0.5 while the maximum value to be expected is 0.92 [10]. To further showcase the efficiency of our model, we also provide some qualitative results from the test set in Figure 4.1.

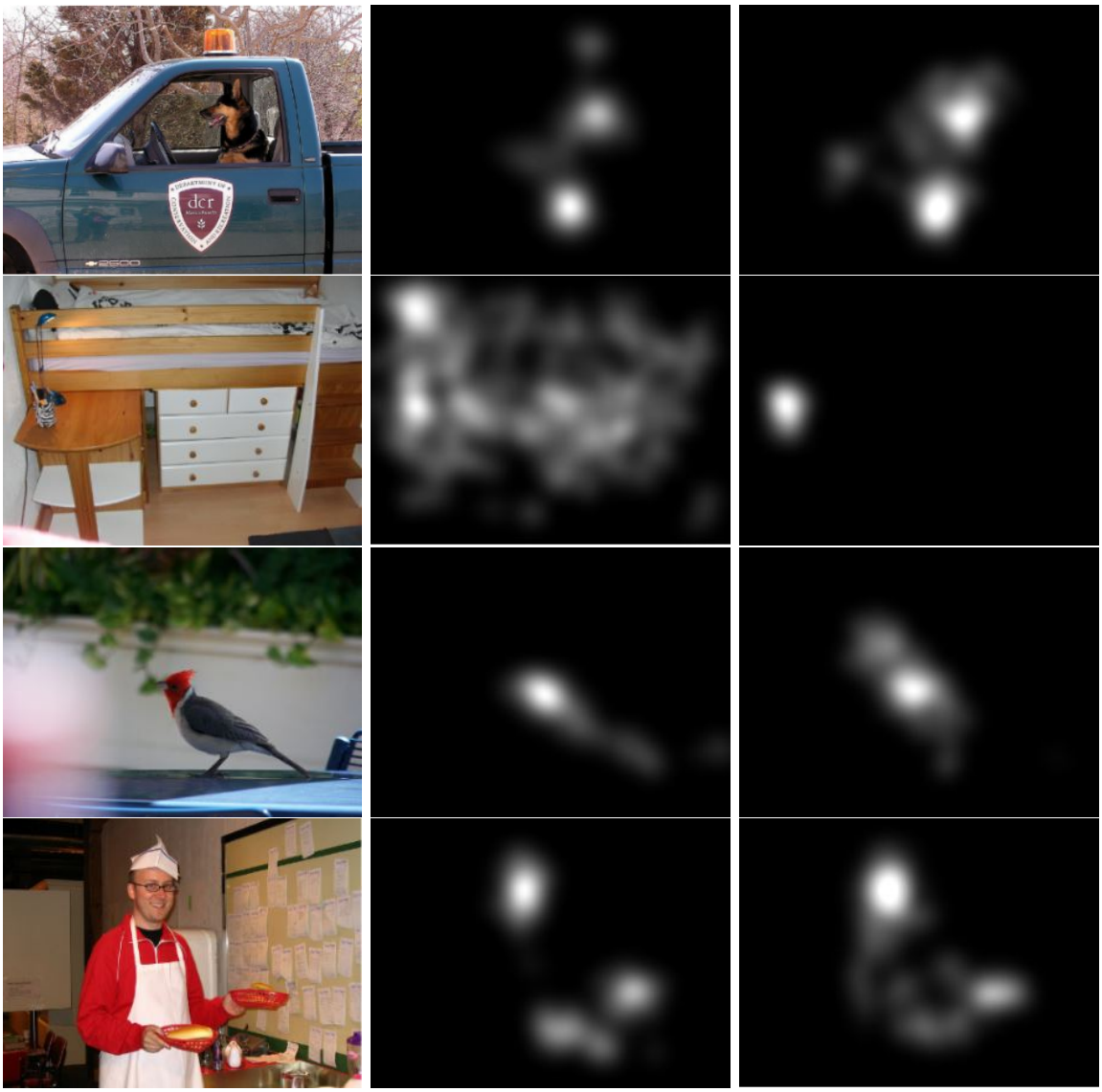


Figure 4.1: Real-world images, ground truths and our generated saliency maps.

Evaluating the ability of the agents: To test the abilities of the agents, we executed various experiments. We focused on creating different scenarios that will accurately capture the accuracy and adaptability of the agents while extensively highlighting their transfer

learning abilities. The difference between these scenarios is the number of successive trainings for the fixation agent.

In the first scenario we directly train the fixation agent to the new segmentation tasks therefore $n = 1$. In the second scenario we re-train the fixation agent to the dog-segmentation task and then separately to the giraffe and bird tasks, therefore $n = 2$. and in the third scenario we apply successive training for all segmentation tasks therefore $n = 3$.

First scenario: n = 1 For our first experiment, the agents are trained on one task at a time. Before diving into the results of the comparison, we will present the results of the fixation agent when we retrain to the dog dataset and compare them to the base model.

Dataset Size	**Base Agent (Dog)**	**Fixation to Dog**
250	0.49	**0.57**
500	0.47	**0.59**
1000	0.52	**0.60**
1900	0.53	**0.60**

Table 4.4: Comparison between the base dog agent and fixation agent when trained on the dog task. Note that the fixation agent wins in all training scenarios.

As we can observe from the results in Table 4.4 when we utilize the fixation agent the new model achieves an increase in accuracy for all dataset sizes.

Next, we will employ the results of the task-optimized agent. This agent was trained for 100 epochs utilizing all 1900 images on the task of dog segmentation and achieves an accuracy of 0.58 for this particular dataset size as can be seen in Table 4.5. Surprisingly, the fixation to dog agent outperforms the task-optimized agent with all dataset sizes except the 250.

The significant improvement in the fixation agent's accuracy in the dog task, along with its superior performance compared to the task-optimized agent, provides initial evidence that training with fixation maps can enhance performance.

Base Agent (Dog)	**Fixation to Dog**	**Task-Optimized Agent**
0.53	**0.60**	0.58

Table 4.5: Comparison between the base dog agent the fixation agent and the task-optimized agent when trained on the dog task with a dataset size of 1900.

Now we will compare the performance in the giraffe and bird segmentation tasks when utilizing the fixation and task-optimized models as a base, respectively. The processes can be seen in Figures 4.2 and 4.3. We will focus solely on the giraffe and bird segmentation tasks since we have already presented a comparison between the base dog segmentation model and the dog segmentation results assisted by the fixation agent.

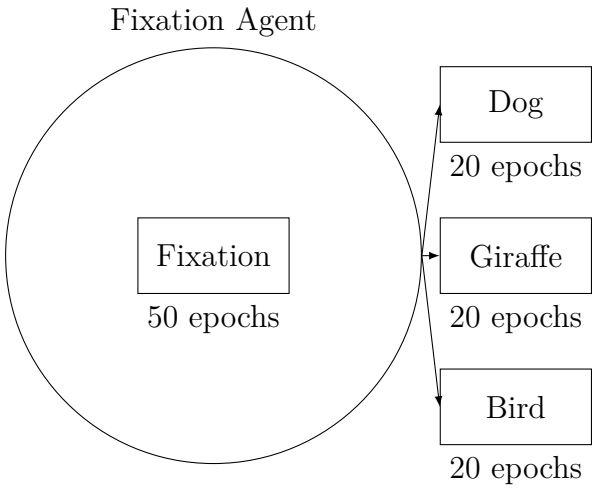


Figure 4.2: Setup for the fixation agent in the first scenario

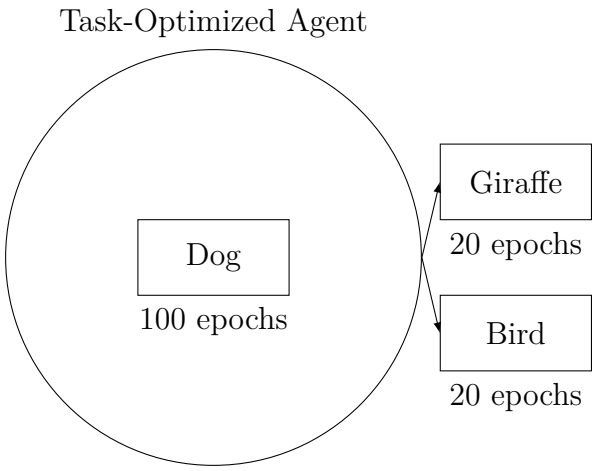


Figure 4.3: Setup for the task-optimized agent in the first scenario

The comparison of the task-optimized and the fixation agent for the giraffe and bird task can be seen in Table 4.6 and 4.7 respectively. Since we didn't apply successive retraining between the giraffe and bird tasks, we chose to present the results in separate tables.

Dataset Size	**Task-Optimized to Giraffe**	**Fixation to Giraffe**
250	0.60	**0.68**
500	0.60	**0.69**
1000	0.61	**0.70**
1900	0.62	**0.70**

Table 4.6: Comparison between the task-optimized and fixation agent when trained on the giraffe task. Note that the fixation agent wins in every training scenario.

Dataset Size	Task-Optimized to Bird	Fixation to Bird
250	0.22	**0.25**
500	0.24	**0.32**
1000	0.19	**0.30**
1900	0.18	**0.36**

Table 4.7: Comparison between the task-optimized and fixation agent when trained on the bird task. Note that the fixation agent wins in every training scenario.

From these results we can make the following observations:

- When we apply the retraining from the task-optimized agent to the giraffe and bird task, we see a small decrease from the base model for the giraffe task and an even smaller decrease for the bird task on average, especially for the largest dataset sized of 1000 and 1900.
- When we apply the retraining from the fixation agent to the giraffe and bird task, we see a substantial increase in both tasks from the base models.
- When the two agents are compared, the superiority of the fixation agent is clear.

In figure 4.4 we provide a qualitative comparison between the fixation and task-optimized agent for the hardest task of bird segmentation when utilizing the model with the highest accuracy for both agents - the model trained with 1900 images for the fixation agent and the model trained with 500 images for the task-optimized agent.

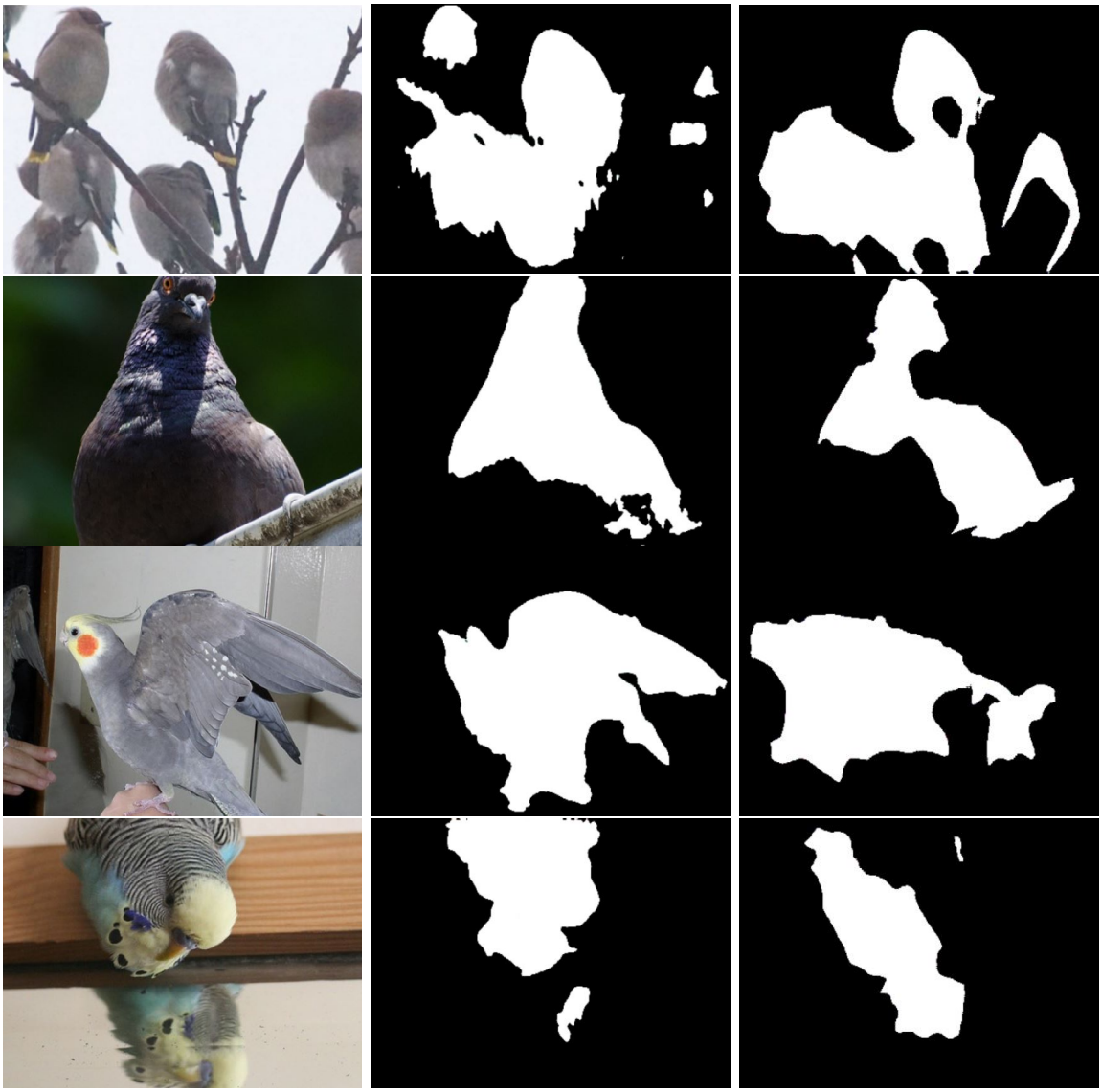


Figure 4.4: Comparison between images generated by the fixation agent and the task-optimized agent for the bird task.

Second scenario: n = 2 Now we will present the results when applying our first successive training. The fixation agent is first trained for 50 epochs, then retrained for 20 epochs on the dog dataset and finally, we apply separate training for the giraffe and the bird datasets as seen in figures 4.5 and 4.6. We present the accuracy of the agents in Table 4.8.

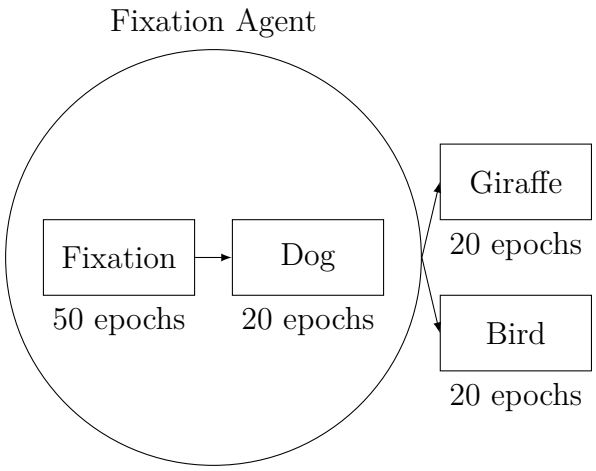


Figure 4.5: Setup for the fixation agent in the second scenario

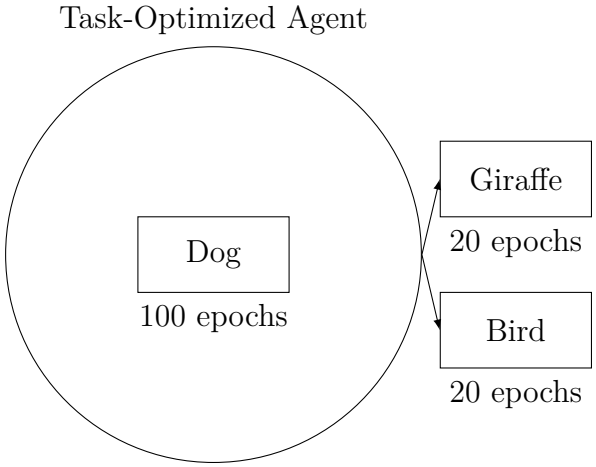


Figure 4.6: Setup for the task-optimized agent in the second scenario

Dataset Size	Task-Optimized Agent		Fixation Agent	
	Giraffe	**Bird**	**Giraffe**	**Bird**
250	0.60	0.22	**0.66**	**0.36**
500	0.60	0.24	**0.66**	**0.35**
1000	0.61	0.19	**0.67**	**0.32**
1900	0.62	0.18	**0.68**	**0.32**

Table 4.8: Comparison between the fixation agent and the task-optimized agent when trained on the dog dataset and then to giraffe and bird segmentation tasks directly.

- The setup for this particular scenario for the task-optimized agent is the same with the previous scenario, therefore the observations for the task-optimized agent hold true to this as well.
- Again, when we apply the retraining from the fixation agent to the giraffe and bird task, we see a substantial increase in both tasks from the base models.
- Concluding, when contrasted with the base and task-optimized agents the superiority of the fixation agent is clear.

Third scenario: n = 3 In our last scenario, we utilized three successive retrains for the fixation agent as can be seen in figures 4.7 and 4.8, and then we compared his accuracy with the accuracy of the task-optimized agent.

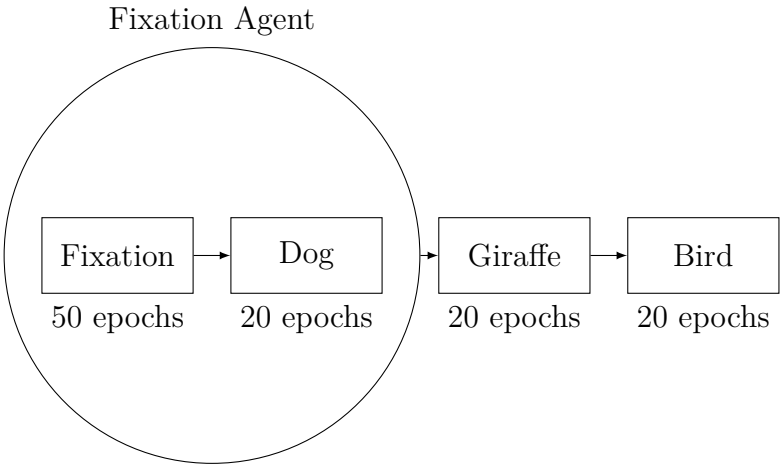


Figure 4.7: Setup for the fixation agent in the third scenario

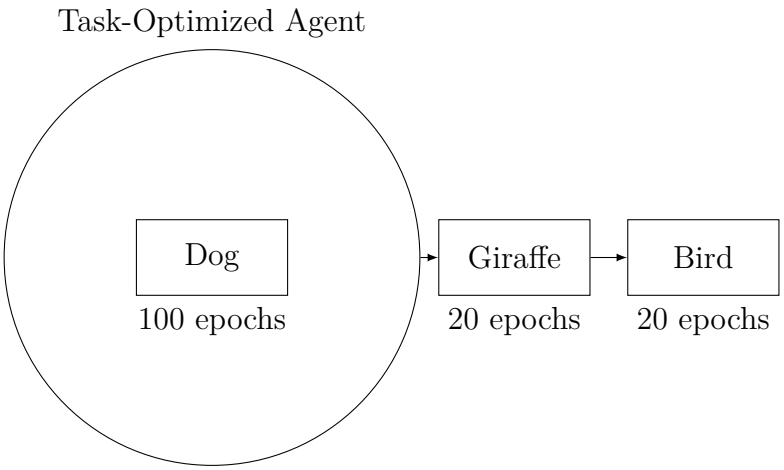


Figure 4.8: Setup for the task-optimized agent in the third scenario

The results can be seen in Table 4.9. For these results we made the following observations:

- The task-optimized agent is outperformed by the fixation agent on both tasks.
- The task-optimized agents ability seems to diminish on dataset size increase for the bird task.
- Again, when contrasted with the base and task-optimized agents the fixation agent is superior.

Dataset Size	Task-Optimized Agent		Fixation Agent	
	Giraffe	**Bird**	**Giraffe**	**Bird**
250	0.60	0.26	**0.66**	**0.32**
500	0.60	0.25	**0.66**	**0.29**
1000	0.61	0.17	**0.67**	**0.30**
1900	0.62	0.15	**0.68**	**0.33**

Table 4.9: Comparison between the fixation agent and the task-optimized agent when trained on each task succesively.

Interpretation of the experimental results for each scenario: Before diving into the experimental interpretations and verdicts, let's repeat our hypothesis in a slightly modified way. This time we will take into account the fact that we don't have a range of tasks but only two, a similar and a dissimilar:

- **First part:** The fixation agent will adapt quickly to the common tasks using a small or moderate-sized dataset regardless of the similarity or dissimilarity.
- **Second part:** The fixation agent, is expected to be more adaptable across tasks than the task-specific agent showcasing generalization ability, a human characteristic.
- **Third part:** The task-optimized agent could be adaptable to the similar task with equal or better performance than the fixation agent. However, in the dissimilar task, his performance is expected to decline in contrast to the first agent.

Now, we will judge the hypothesis separately for each scenario. If all parts of the hypothesis are undeniably true then we will accept the hypothesis for the respective scenario.

We will reject the hypothesis if there is at least one undeniably false part. Finally, it could also be the case that the results are fuzzy:

Verdict for n = 1: We observed that the fixation agent outperforms the task-optimized agent and the base agents in both the giraffe and the bird task for all dataset sizes therefore all three parts hold undeniably true, subsequently the hypothesis is **accepted** for this particular scenario.

Verdict for n = 2: Again the fixation agent outperformed the task-optimized agent and the base agents in both the giraffe and the bird task for all dataset sizes, therefore all three parts hold undeniably true. For $n = 2$, the hypothesis is **accepted**.

Verdict for n = 3: Finally, the same pattern holds true for the third scenario. The task-optimized agent and the base agents are outperformed by the fixation agent an all dataset sizes. Again, for $n = 3$, the hypothesis is **accepted**.

Retrains	First part	Second part	Third part
n=1	True	True	True
n=2	True	True	True
n=3	True	True	True

Table 4.10: Summary of hypotheses evaluation across different number of retrains

Based on the above table, we can claim that for our experimental setup, the hypothesis is **accepted**.

4.2 Limitations

The main experiment definition has a significant limitation:

- The time and cost to train our model architecture was significantly high. For a base model of 1900 images, training for 20 epochs utilizing an A100 graphics card, required 25 minutes of training time. This computational complexity is a well-known problem when training conditional GANs and significantly limits the number of experiments that could be executed. Consequently, more computational resources would allow us to perform multiple experiments with different types of animals, which would allow us to establish an extensive computational taxonomy and derive further insights.

CHAPTER 5

CONCLUSIONS

In this thesis, we mainly focused on two topics. The first was computational models of attention and bottom-up fixation prediction. The second was transfer learning and the ability to generalize across tasks. By combining knowledge from both domains and inspired by human transfer learning abilities, we created a hypothesis for the generalization ability of fixation models in task-specific scenarios. To test this hypothesis, we crafted and executed different experimental scenarios.

The experimental scenarios revolved around the comparison of a general-purpose agent trained on fixation maps and a task-optimized segmentation agent. The central hypothesis was that if common tasks are introduced between them, which become increasingly dissimilar, the general-purpose agent will showcase greater adaptability and generalization.

To test this hypothesis, we introduced two common tasks between the agents, a similar task and a dissimilar task to that of the task-optimized agents. The tasks were simple animal segmentation tasks, and the similarity of the animals was determined based on the generally accepted animal taxonomy created by Carl Linnaeus [51], which takes into account the similar physical characteristics of animals.

When creating the different experimental scenarios, we focused on creating experiments that would be able to capture the generalization abilities of the fixation and the task-optimized agents while providing us with a full view of their transfer learning abilities on both direct and successive training. Consequently, we crafted three different scenarios with both direct retrains on each task separately and sequential retrains applying the tasks one

after the other.

For our experimental conditions, we concluded that the fixation agent did indeed have increased generalization abilities when compared to the task-optimized for all the different scenarios we have created.

REFERENCES

[1] Radhakrishna Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1597–1604, 2009.

[2] Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C. Fowlkes, Stefano Soatto, and Pietro Perona. Task2Vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *CoRR*, abs/2003.05991, 2020.

[4] Ali Borji and Laurent Itti. Exploiting local and global patch rarities for saliency detection. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 478–485, 2012.

[5] Ali Borji and Laurent Itti. CAT2000: A large scale fixation dataset for boosting saliency research. *CoRR*, abs/1505.03581, 2015.

[6] Ali Borji, Dicky N. Sihite, and Laurent Itti. Quantitative analysis of human-model agreement in visual saliency modeling: A comparative study. *IEEE Transactions on Image Processing*, 22(1):55–69, 2013.

[7] Ali Borji, Hamed R. Tavakoli, Dicky N. Sihite, and Laurent Itti. Analysis of scores, datasets, and models in visual saliency prediction. In *2013 IEEE International Conference on Computer Vision*, pages 921–928, 2013.

[8] Neil Bruce and John Tsotsos. Saliency based on information maximization. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005.

[9] Zoya Bylinskii, Tilke Judd, Ali Borji, Laurent Itti, Frédo Durand, Aude Oliva, and Antonio Torralba. MIT Saliency Benchmark.

[10] Zoya Bylinskii, Tilke Judd, Aude Oliva, Antonio Torralba, and Frédo Durand. What do different evaluation metrics tell us about saliency models? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41:740–757, 2016.

[11] A. Cain. Taxonomy, 2024. Encyclopedia Britannica, June 20, 2024.

[12] Ming-Ming Cheng, Niloy J. Mitra, Xiaolei Huang, Philip H. S. Torr, and Shi-Min Hu. Global contrast based salient region detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):569–582, 2015.

[13] Marcella Cornia, Lorenzo Baraldi, Giuseppe Serra, and Rita Cucchiara. Predicting human eye fixations via an lstm-based saliency attentive model. *IEEE Transactions on Image Processing*, 27(10):5142–5154, 2018.

[14] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[15] Samuel F. Dodge and Lina J. Karam. Visual saliency prediction using a mixture of deep neural networks. *IEEE Transactions on Image Processing*, 27(8):4080–4090, 2018.

[16] Tom Fawcett. An introduction to ROC analysis. *Pattern Recognit. Lett.*, 27:861–874, 2006.

[17] Simone Frintrop, Erich Rome, and Henrik Christensen. Computational visual attention systems and their cognitive foundations: A survey. *TAP*, 7, 01 2010.

[18] Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350–363, 2013.

[19] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256. PMLR, 13–15 May 2010.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[21] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[22] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems*, volume 19. MIT Press, 2006.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[24] Bernhard Hommel, Craig S Chapman, Paul Cisek, et al. No one knows what attention is. *Attention, Perception, & Psychophysics*, 81(7):2288–2303, 2019.

[25] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.

[26] L. Itti. Visual salience. *Scholarpedia*, 2(9):3327, 2007. revision #72776.

[27] 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*, 20(11):1254–1259, 1998.

[28] Laurent Itti and Christof Koch. Computational modelling of visual attention. *Nature Reviews Neuroscience*, 2:194–203, 2001.

[29] William James. *The Principles of Psychology*. Henry Holt, New York, 1890.

[30] Saumya Jetley, Naila Murray, and Eleonora Vig. End-to-end saliency mapping via probability distribution prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[31] Sen Jia and Neil D. B. Bruce. Eml-net:an expandable multi-layer network for saliency prediction, 2019.

[32] Ming Jiang, Shengsheng Huang, Juanyong Duan, and Qi Zhao. Salicon: Saliency in context. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[33] W A Johnston and V J Dark. Selective attention. *Annual Review of Psychology*, 37(1):43–75, 1986.

[34] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *2009 IEEE 12th International Conference on Computer Vision*, pages 2106–2113, 2009.

[35] Fumi Katsuki and Christos Constantinidis. Bottom-up and top-down attention: different processes and overlapping neural systems. *The Neuroscientist*, 20(5):509–521, 2014.

[36] Christof Koch and Shimon Ullman. Shifts in selective visual attention: towards the underlying neural circuitry. *Human neurobiology*, 4(4):219–227, 1985.

[37] Katrin Koehler, Fangshi Guo, Shu Zhang, and Miguel P. Eckstein. What do saliency models predict? *Journal of Vision*, 14(3):14, 2014.

[38] Antonio Torralba Krista A. Ehinger, Barbara Hidalgo-Sotelo and Aude Oliva. Modelling search for people in 900 scenes: A combined source model of eye guidance. *Visual Cognition*, 17(6-7):945–978, 2009.

[39] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.

[40] Srinivas S. S. Kruthiventi, Kumar Ayush, and R. Venkatesh Babu. Deepfix: A fully convolutional neural network for predicting human eye fixations. *IEEE Transactions on Image Processing*, 26(9):4446–4456, 2017.

[41] Matthias Kümmerer and Matthias Bethge. Predicting visual fixations. *Annual Review of Vision Science*, 9(Volume 9, 2023):269–291, 2023.

[42] Matthias Kümmerer, Matthias Bethge, and Thomas S. A. Wallis. DeepGaze III: Modeling free-viewing human scanpaths with deep learning. *Journal of Vision*, 22(5):7–7, 04 2022.

[43] Matthias Kümmerer, Lucas Theis, and Matthias Bethge. Deep Gaze I: Boosting saliency prediction with feature maps trained on imagenet, 2015.

[44] Matthias Kümmerer, Thomas S. A. Wallis, and Matthias Bethge. DeepGaze II: Reading fixations from deep features trained on object recognition, 2016.

[45] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, 1989.

[46] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[47] Yann Lecun, Leon Bottou, Y. Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278 – 2324, 12 1998.

[48] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-realistic single image super-resolution using a generative adversarial network, 2017.

[49] J. B. Levitt and J. S. Lund. Contrast dependence of contextual effects in primate visual cortex. *Nature*, 387(6628):73–76, May 1997.

[50] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context, 2015.

[51] Carl Linnaeus. *Systema Naturae*. Laurentius Salvius, 10 edition, 1758.

[52] Nian Liu and Junwei Han. A deep spatial contextual long-term recurrent convolutional network for saliency detection. *IEEE Transactions on Image Processing*, 27(7):3264–3274, 2018.

[53] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2):353–367, 2011.

[54] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[55] Yu-Fei Ma and Hong-Jiang Zhang. Contrast-based image attention analysis by using fuzzy growing. In *Proceedings of the Eleventh ACM International Conference on Multimedia*, MULTIMEDIA '03, page 374–381, New York, NY, USA, 2003. Association for Computing Machinery.

[56] Ali Mahdi, Jun Qin, and Garth Crosby. Deepfeat: A bottom-up and top-down saliency model based on deep features of convolutional neural networks. *IEEE Transactions on Cognitive and Developmental Systems*, 12(1):54–63, 2020.

[57] Ran Margolin, Ayellet Tal, and Lihi Zelnik-Manor. What makes a patch distinct? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[58] Stephanie A. McMains and Sabine Kastner. *Visual Attention*, pages 4296–4302. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[59] Lucia Melloni, Sander van Leeuwen, Arjen Alink, and Nils G Müller. Interaction between bottom-up saliency and top-down control: how saliency maps are created in the human brain. *Cerebral Cortex*, 22(12):2943–2952, Dec 2012.

[60] Umberto Michelucci. An introduction to autoencoders. *CoRR*, abs/2201.03898, 2022.

[61] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[62] E. Niebur. Saliency map. *Scholarpedia*, 2(8):2675, 2007. revision #147400.

[63] Ernst Niebur and Christof Koch. Control of selective visual attention: Modeling the "where" pathway. In D. Touretzky, M.C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8. MIT Press, 1995.

[64] A.C. Nobre, D.R. Gitelman, E.C. Dias, and M.M. Mesulam. Covert visual spatial orienting and saccades: Overlapping neural systems. *NeuroImage*, 11(3):210–216, 2000.

[65] Hans-Christoph Nothdurft, Jack L Gallant, and David C Van Essen. Response modulation by texture surround in primate area v1: correlates of "popout" under anesthesia. *Visual neuroscience*, 16(1):15–34, 1999.

[66] Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Mattias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard

Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018.

[67] A. Oliva, A. Torralba, M.S. Castelhano, and J.M. Henderson. Top-down control of visual attention in object detection. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 1, pages I–253, 2003.

[68] Keiron O'Shea and Ryan Nash. An introduction to convolutional neural networks. *ArXiv e-prints*, 11 2015.

[69] Junting Pan, Cristian Canton Ferrer, Kevin McGuinness, Noel E. O'Connor, Jordi Torres, Elisa Sayrol, and Xavier Giro i Nieto. Salgan: Visual saliency prediction with generative adversarial networks, 2018.

[70] Sinno Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22:1345 – 1359, 11 2010.

[71] Derek Parkhurst, Kinjiro Law, and Ernst Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision Research*, 42(1):107–123, 2002.

[72] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting, 2016.

[73] Chad Peltier and Mark Becker. Individual differences predict low prevalence visual search performance. *Cognitive Research: Principles and Implications*, 2, 12 2017.

[74] M. I. Posner. Orienting of attention. *Quarterly Journal of Experimental Psychology*, 32(1):3–25, 1980.

[75] Michael I Posner, Yoav Cohen, et al. Components of visual orienting. *Attention and performance X: Control of language processes*, 32:531–556, 1984.

[76] Lee Ann Remington. Chapter 13 - Visual Pathway. In Lee Ann Remington, editor, *Clinical Anatomy and Physiology of the Visual System (Third Edition)*, pages 233–252. Butterworth-Heinemann, Saint Louis, third edition edition, 2012.

[77] Zhixiang Ren, Yiqun Hu, Liang-Tien Chia, and Deepu Rajan. Improved saliency detection based on superpixel clustering and saliency propagation. In *Proceedings of the 18th ACM International Conference on Multimedia*, MM '10, page 1099–1102, New York, NY, USA, 2010. Association for Computing Machinery.

[78] Seyed Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian D. Reid, and Silvio Savarese. Generalized Intersection over Union: A metric and A loss for bounding box regression. *CoRR*, abs/1902.09630, 2019.

[79] Nicolas Riche, Matthieu Duvinage, Matei Mancas, Bernard Gosselin, and Thierry Dutoit. Saliency and human fixations: State-of-the-art and study of comparison metrics. 12 2013.

[80] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation, 2015.

[81] Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. Disan: Directional self-attention network for RNN/CNN-free language understanding. *CoRR*, abs/1709.04696, 2017.

[82] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[83] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[84] R. W. Storer, Austin L. Rand, and Frank Gill. Bird, 2024. Encyclopedia Britannica, May 14, 2024.

[85] Ben Tan, Yangqiu Song, Erheng Zhong, and Qiang Yang. Transitive transfer learning. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1155–1164, 2015.

[86] Anne M. Treisman and Garry Gelade. A feature-integration theory of attention. *Cognitive Psychology*, 12(1):97–136, 1980.

[87] J. K. Tsotsos and A. Rothenstein. Computational models of visual attention. *Scholarpedia*, 6(1):6201, 2011. revision #171311.

[88] John Tsotsos. *A Computational Perspective on Visual Attention*. 01 2011.

[89] Eleonora Vig, Michael Dorr, and David Cox. Large-scale optimization of hierarchical features for saliency prediction in natural images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[90] Jeremy Wolfe, Kyle Cave, and Susan Franzel. Guided search: An alternative to the feature integration model for visual search. *Journal of experimental psychology. Human perception and performance*, 15:419–33, 08 1989.

[91] Yulin Xie, Huchuan Lu, and Ming-Hsuan Yang. Bayesian saliency via low and mid level cues. *IEEE Transactions on Image Processing*, 22(5):1689–1698, 2013.

[92] Fei Yan, Cheng Chen, Peng Xiao, Siyu Qi, Zhiliang Wang, and Ruoxiu Xiao. Review of visual saliency prediction: Development process from neurobiological basis to deep models. *Applied Sciences*, 12(1), 2022.

[93] Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3712–3722, 2018.

[94] Yun Zhai and Mubarak Shah. Visual attention detection in video sequences using spatiotemporal cues. In *Proceedings of the 14th ACM International Conference on*

Multimedia, MM '06, page 815–824, New York, NY, USA, 2006. Association for Computing Machinery.

[95] Jianming Zhang and Stan Sclaroff. Saliency detection: A boolean map approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2013.

[96] Lingyun Zhang, Matthew H. Tong, Tim K. Marks, Honghao Shan, and Garrison W. Cottrell. SUN: A Bayesian framework for saliency using natural statistics. *Journal of Vision*, 8(7):32–32, 12 2008.

[97] Wen Zhang, Lingfei Deng, Lei Zhang, and Dongrui Wu. A survey on negative transfer. *IEEE/CAA Journal of Automatica Sinica*, 10(2):305–329, 2023.

[98] Zhengxin Zhang, Qingjie Liu, and Yunhong Wang. Road extraction by deep residual u-net. *IEEE Geoscience and Remote Sensing Letters*, 15(5):749–753, May 2018.

[99] Li Zhaoping and Keith A May. Psychophysical tests of the hypothesis of a bottom-up saliency map in primary visual cortex. *PLoS Computational Biology*, 3(4):e62, 2007.

[100] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2021.