Technische Universität Berlin
Faculty IV – Electrical Engineering and Computer Science
Institute of Computer Engineering and Microelectronics
Dept. Computational Psychology

Bachelor Thesis

Do Fixational Eye Movements Facilitate Object Recognition in CNNs?

Nico Kestel
nico.kestel@campus.tu-berlin.de

B.Sc. Computer Science
Student ID. 401128

Berlin, 28.02.2022

First examiner: Prof. Dr. Marianne Maertens
marianne.maertens@tu-berlin.de
Second examiner: Prof. Dr.-Ing. Olaf Hellwich
olaf.hellwich@tu-berlin.de

Abstract

Fixational Eye Movements (FEMs) are considered relevant for encoding spatial edges as temporal modulations in the retina. Visual processing therefore is fundamentally spatiotemporal. We mimic visual processing with Convolutional Neural Networks (CNNs), which only work with purely spatial images. In this thesis, we investigate whether simulating FEM induced spatiotemporal processing on images facilitates object recognition in CNNs.

We approximate FEM induced spatiotemporal processing in the retina via difference images. We train two CNNs on the ImageNet dataset for object recognition. We train the first CNN on standard grayscale images (CNN-gray). We train the second CNN on difference images (CNN-diff). We compare accuracy scores of both CNNs on natural images. We also compare accuracy scores on the ImageNet-Sketch dataset because CNNs typically are reported to perform poorly on sketches. In both cases, natural images and sketches, the CNN-gray achieves higher accuracy scores than the CNN-diff. We discuss possible explanations for these results and potential limiting factors of our approach. We conclude that FEMs do not facilitate object recognition in CNNs.

Zusammenfassung

Augenbewegungen während visueller Fixation (FEMs) sind relevant, um räumliche Kanten als zeitliche Änderungen in der Retina zu kodieren. Visuelle Verarbeitung geschieht demnach grundlegend in Raum und Zeit. Wir imitieren visuelle Verarbeitung mit Convolutional Neural Neutworks (CNNs), die nur mit räumlichen Bildern arbeiten. In dieser Thesis untersuchen wir, ob Objekterkennung in CNNs vereinfacht werden kann, indem räumliche und zeitliche Verarbeitung, die durch FEMs induziert wird, auf Bildern simuliert wird.

Wir approximieren durch FEMs induzierte räumliche und zeitliche Verarbeitung in der Retina über Differenzbilder. Wir trainieren zwei CNNs auf dem ImageNet Datensatz zur Objekterkennung. Wir trainieren das erste CNN auf gewöhnlichen Graustufenbildern (CNN-gray). Wir trainieren das zweite CNN auf Differenzbildern (CNN-diff). Wir vergleichen die Genauigkeit beider CNNs auf natürlichen Bildern. Wir vergleichen die Genauigkeit beider CNNs auch auf Skizzenzeichnungen, weil CNNs typischerweise schlecht auf Skizzenzeichnungen abschneiden. In beiden Fällen, natürliche Bilder und Skizzenzeichnungen, erreicht das CNN-gray höhere Genauigkeitswerte als das CNN-diff. Wir diskutieren mögliche Erklärungen für unsere Ergebnisse und potentielle einschränkende Faktoren unserer Herangehensweise. Zusammenfassend schließen wir, dass FEMs die Objekterkennung in CNNs nicht vereinfachen.

Contents

1 Introduction

1.1 Luminance Edges

We perceive the world around us through different modalities. Probably the most important modality is visual perception which starts in our eyes. Light that is reflected off the environment enters through our eyes and hits the retina. The retina consists of several layers of cells. First, the incoming light is transduced to an electric signal by roughly 130 million photoreceptors per eye. The electric signal then is processed by horizontal, bipolar and amacrine cells. The final processing step in the retina is performed by ganglion cells before the signal leaves the retina and is propagated to higher structures of the visual system.

Processing the full incoming signal from phototreceptors might be metabolically too expensive (Laughlin et al., 1998). This might be the reason why humans and other mammals have evolved to efficiently encode sensory input in the early visual system, e.g. the retina. Single-cell recordings of retinal ganglion cells revealed that the visual system is most sensitive to luminance changes of the sensory input in space and time (Frishman et al., 1987). Luminance is the objective measure for the amount of light that reflects off or is emitted from an object. In the following, we will describe the relevance of spatial and temporal luminance edges for visual processing.

A spatial luminance change is the transition between bright and dark regions in space, as it exists for example in an image of black and white stripes. We also refer to spatial luminance changes as spatial luminance edges. It has been shown in physiological studies that retinal ganglion cells respond best to edges because of how their receptive fields are organized (Croner & Kaplan, 1995). The receptive field of a ganglion cell refers to the proportion of the visual field which elicits a response in the cell. A temporal luminance change occurs when the light intensity, that a fixed point on the retina experiences, changes over time. This happens, for example, when flickering light hits the retina or when an image of black and white stripes

moves across the retina (orthogonal to the stripes). Physiological investigations revealed that the receptive fields of retinal ganglion cells have specific spatial and temporal characteristics.

Spatially, they are roughly circular and show an antagonisitic center-surround relationship. That is, when light hits receptors in the center of the receptive field, this induces an excitatory effect to the ganglion cell, and when light hits receptors in the surround of the receptive field, this induces an inhibitory effect to the ganglion cell, or vice versa. Likewise, there are two types of retinal ganglion cells (Croner & Kaplan, 1995). The first type responds best if light hits the center of the receptive field and the surround is dark. The second type responds best if light hits the surround and the central portion is dark. If every receptor in the receptive field receives the same amount of light, the excitatory and inhibitory effects cancel out. If the stimulus is distributed unequally over the center and surround, e.g. when a luminance edge enters the receptive field, the ganglion cell's firing rate changes according to the ratio of excitatory to inhibitory signals (Kuffler, 1953). Likewise, spatial luminance edges are considered to be stimuli which effectively stimulate retinal ganglion cells. Experiments by Anstis (2013) even suggest that the perception of surfaces fully depend on the visibility of the surrounding edges. By intervening with edge-sensitive processes through adapting our photoreceptors to the flickering outline of a surface, the surface perceptually disappears and the perceptual "hole" is filled completely with information from the surround.

Temporally, retinal ganglion cells respond best to luminance changes in a specific range of temporal frequencies and worse to stimuli that change too quickly or not at all (Frishman et al., 1987). In fact, an image that is stabilized to the retina fades from our perception until it disappears entirely (Ditchburn & Ginsborg, 1952; Riggs & Ratliff, 1952; Yarbus, 1967). Retinal stabilization is accomplished by paralyzing the ocularmotor system or moving the distal stimulus following the eye movements such that the movements effectively cancel out and the stimulus ends up fixed (stablized) to the retina (Santini et al., 2007).

The human retina is designed to report spatial and temporal changes in the visual scene. Therefore, homogeneous surface information is neglected and edges yield strong neural responses. Similarly, the retina is only sensitive to temporal modulations around a specific temporal frequency, and does not respond to static inputs. This is an efficient code to represent

our surround and probably is crucial for us to be able to comprehend the amount of information our eyes are confronted with.

1.2 Fixational Eye Movements (FEMs)

We need to move our eyes in order to fixate objects of interest because the fovea, i.e. the spot of highest acuity in the retina, only covers a small portion of our visual field. However, even during these fixation periods our eyes are always in motion. Fixational eye movements (FEMs) are small involuntary eye movements that occur during visual fixation (Yarbus, 1967). For a long period of time, FEMs have been neglected (Steinman et al., 1973) or considered obstructive for high acuity vision because it was assumed that FEMs cause the retinal image to blur which reduces fine spatial details (Packer & Williams, 1992). Besides that, the only function that was sometimes attributed to FEMs was to prevent perceptual fading (Martinez-Conde et al., 2004). Nowadays, however, evidence has accumulated that FEMs are part of the active sampling strategy of the visual system to encode spatial information within a spatiotemporal code (Rucci & Victor, 2015). FEMs are microscopically small. For comparison, our index finger nail at arm's length roughly covers 1 degree (deg) of our visual field, whereas FEMs are as small as <1 minute of arc (60 $arcmin$ = 1 deg) and maximally as large as 1 deg (Martinez-Conde et al., 2004). Further analyses of recorded gaze trajectories revealed three subcategories of FEMs: micro-saccades, ocular drift and tremor (Martinez-Conde, 2006).

Microsaccades are the largest and fastest FEMs. They cause the retinal image to move across up to hundreds of photoreceptors (Møller et al., 2002). Due to their size, microsaccades are proposed to be the most effective FEM to prevent visual fading by "refreshing" underlying photoreceptors (Carpenter, 1988; Ditchburn et al., 1959). Microsaccades also appear to keep the gaze on the fixated object by counteracting ocular drift (Cornsweet, 1956).

Ocular drifts occur between consecutive microsaccades. Drifts show a continuous meandering motion that often is described as Brownian motion (e.g. Kuang et al., 2012). Drift causes the retinal image to be moving incessantly over photoreceptors during periods of visual fixation. It has been suggested that this constant motion of eyes is essential for the visual system

to actively encode spatial information within a spatiotemporal signal (Kuang et al., 2012). Likewise, the spatiotemporal modulations that emerge from drift motions have been proposed to reduce redundant information when viewing natural scenes by equalizing the spectral power across a wide range of spatial frequencies (Kuang et al., 2012). Moreover, Kuang et al. (2012) and Rucci and Victor (2015) propose that through ocular drift, edge extraction is already started in the retina. Due to ocular drift, the retinal image gets slightly shifted across the receptive fields of retinal ganglion cells over time. Since retinal ganglion cells respond best to temporally changing inputs which mainly occur at the edges of objects, drift further enhances information at luminance edges (Rucci & Victor, 2015). Thus, drift can be considered to play a central role in extracting edge signals from the sensory input.

Tremors are superimposed to ocular drift and present high frequency (~$90Hz$) jitter-like movement of gaze with amplitudes that match the diameter of single retinal photoreceptors (Martinez-Conde, 2006). The function of tremors is still debated. Some suggest, because of the size of these movements, tremors might be caused by neural noise in the ocularmotor system (Carpenter, 1988) or consist of artifacts in the recording devices.

1.3 Convolutional Neural Networks (CNNs)

1.3.1 CNNs as Models for Visual Perception

CNNs are commonly referred to as state-of-the-art computational model for visual perception (Nandhini Abirami et al., 2021). For simple recognition tasks, e.g. handwritten digit recognition, they perform very well and achieve human-like performance (Cireşan, Meier, & Schmidhuber, 2012). A reason for their success in visual recognition might be the mathematical operation that separates CNNs from other artificial neural networks: the convolution.

A convolution can be thought of as applying a filter to an input image at every location in the image analogous to cells in the visual system that process the incoming signal based on their receptive field properties. To better describe how convolutions work, let's imagine a single convolutional filter that slides over a signal. In our case the signal is an image in which we want to recognize objects. The convolutional filter scans a neighbourhood of

pixels in the image instead of the whole image, before sliding to the next neighbourhood of pixels. By scanning only a limited area of an image at once, small features of objects might be detectable more easily because the filter won't be misled by the information in the remaining image, which might contain other object specific features. This is similar to how we scan an image for small details; we systematically investigate small regions to check for details and move our searchlight to another small region if we haven't found what we were looking for. A convolutional filter wouldn't stop if it had found the feature but would always slide over the entire image and respond whether or not the feature is present[1]. In further steps, detected features can be aggregated to identify the entire displayed object. For example, when convolutional filters detect a lot of straight lines and rectangles, then the image might show a building of some kind. A convolutional filter can only detect one single feature best. In order to recognize objects faster, we want a CNN to detect multiple different features simultaneously. For that reason, multiple convolutional filters are organized as a convolutional layer in the network. Which filters a CNN should apply to images to recognize certain objects best is the result of training a CNN (see **Section 1.3.2**).

A convolutional filter responds to a fixed size portion of the signal, depending on its filter size. Similarly, a retinal ganglion cell responds to the light that hits receptors in its receptive field, which is only a limited section of the entire retinal image (Frishman et al., 1987). Nevertheless, retinal ganglion cells also respond to temporal modulations of the signal (Frishman et al., 1987). Whereas convolutional filters, and most CNNs in general, do not consider signal changes over time but only work with purely spatial information. This is a potentially fundamental misconception about CNNs and how they are used to model the human visual system because the human retina is only sensitive to temporal changes.

1.3.2 Training a CNN

To understand how Convolutional Neural Networks (CNNs) work, we go into standard Artificial Neural Networks (ANNs) first.

1 This is a simplification. Usually the responses of convolutional filters are not binary but continuous.

ANNs originate from Rosenblatt's Perceptron (Rosenblatt, 1958). The Perceptron describes a mathematical model of a neuron's basic functionality. The Perceptron receives multiple inputs that get weighted individually. The weighted sum of inputs is fed to the Perceptron's activation function that decides whether or not the Perceptron fires. Depending on the specific activation function, it can also decide on how strongly the Perceptron fires according to the input intensity, a scenario that is not possible with real neurons because, due to the uniformity of an action potential, they encode intensity with firing frequency, not with amplitude.

To expand the idea of an artificial neuron to an artificial neural network, many neurons are organized in layers that are connected to previous and succeeding layers (Multilayer Perceptron; Gardner & Dorling, 1998). The first layer of neurons (input layer) receives the raw net input, therefore there must be one neuron for each input feature. The neurons in the last layer (output layer) represent the net predictions, i.e. there is one neuron per class that indicates that the net input belongs to the neuron's dedicated class. Layers that are neither the input nor output layer are referred to as hidden layers.

Each neuron in the hidden layers and output layer receives the output of all neurons in the previous layer. We call layers, that are connected to their previous layer in this way, fully-connected or dense layers. In addition to the individual weights for different inputs, a neuron also receives a bias input. The bias is independent from the net input and represents a base activation of the neuron. This bias usually is realized as an additional weight to a constant artificial 1 input. We include the bias term when referring to weights.

Training an ANN. The knowledge of an ANN about a certain domain is represented in the entirety of its weights. Adjusting these weights is how ANNs learn, therefore we briefly cover the general training procedure of ANNs.

In order for an ANN to learn the difference between several objects, it has to observe many objects. This kind of learning is also called "supervised learning" because the observations come with ground truth labels stating what the object is, e.g. if there is an image of an apple (the object) there is also a label stating "this is an apple". During training, the ANN can compare its predicted label of an object with the ground truth label to evaluate its learning progress.

Aim of learning is not to perfectly discriminate objects in the available training data but to learn discriminants that also apply to data not contained in the training data. For that purpose the available data is divided into smaller datasets. The train set is used to actually train the ANN. The validation set is used to monitor the generalization capability of the ANN. This is accomplished by checking if the learned discriminants also apply to the validation set which itself is not used by the ANN to extract these discriminants. The test set is used to finally report the generalization performance of the ANN. It is larger than the validation set and therefore provides a more reliable evaluation. The three subsets of the available data are disjoint.

The ANN learns the discriminants in an iterative process, during which it receives the whole train set as input once per iteration (epoch). In each epoch the network predicts each one label for each input stimulus. In our case, it would therefore predict the object category for each input image. The prediction error is determined by an error function (loss function). The loss function calculates the difference between prediction and ground truth and returns a real valued measure of the mismatch. The prediction error is calculated based on the network's output and is propagated back through the network. At each step in backpropagation the ANN's weights are adjusted according to the learning schedule provided by the chosen optimizer. A popular choice for such an optimizer is a variant of Gradient Descent (e.g. Bottou, 1991; Kingma & Ba, 2017) that minimizes the loss function by adjusting a weight in the direction of the loss function's gradient with respect to the weight. The gradient shows how each individual weight should be changed to minimize the overall loss function. The size of the weight adjustment steps is influenced by the optimizer's learning rate, a small positive scaling factor.

Adjusting the weights of an ANN often does not happen with the prediction error on the whole train set at once but in batches, i.e. smaller portions of the training data. Here, the train set is divided into multiple disjoint batches. The prediction error on a batch is used to adjust weights as described before. With a large batch size few but large adjustments are done. With smaller batches small adjustments happen more frequently.

Smaller recognition and classification tasks with small datsets and few classes, such as digit recognition (Lecun et al., 1998) can be tackled effectively with large Multilayer Perceptrons (Cireşan, Meier, Gambardella, et al., 2012). ANNs liks this have enormous amounts of weights which makes them hard to train if the problem to solve gets more complex, e.g. higher dimensional input and more classes. Convolutional Neural Networks (CNNs) come with way less weights which makes them easier to train. The smaller weight count also means that a CNN cannot hold as much information as a standard ANN of comparable size. Despite the theoretically smaller knowledge capacity, CNNs perform remarkably well in large scale object recognition (Krizhevsky et al., 2012) and have since been handled as state-of-the-art visual recognition method.

1.4 Aim of this Thesis

Visual processing is in its core spatiotemporal and the visual system is most sensitive to information that is changing over time. Temporal modulations that are caused by ocular drift are supposed to start the extraction of edges in the retina (Rucci & Victor, 2015). CNNs are often used to model the visual system. However, CNNs typically are trained on static images, and thus do not consider the relevance of temporal modulations for visual processing. in this thesis, we want to investigate to which extent a classical CNN benefits from a visual input that reflects that the visual system is only sensitive to temporally changing inputs.

For this, we want to investigate whether a CNN performs better if it is trained on images that are actively derived from ocular drift and the sensitivity of the visual system to temporally changing inputs (here called "difference images"), compared to a CNN that is trained on standard achromatic images. We are expecting to achieve better performance of the first CNN because we assume that drift reduces redundant information of low-contrast surfaces (Rucci & Victor, 2015) and thus will emphasize the most relevant regions with higher variance in luminance, i.e. luminance edges.

Second, Singer et al. (2020) reported that CNNs, that were trained on natural images, perform poorly when applied to more abstract data like line-drawings and sketches. Since ocular drift might be involved in extracting edges (Rucci & Victor, 2015) and thus accentuates contour over surfaces similar to sketches, we want to investigate whether CNNs trained on difference images will recognize sketches more accurately than CNNs trained on natural grayscale images.

We assume that drift accentuates contour over surfaces similar to sketches, which leads to the assumption, a CNN that is trained on drift images will predict sketches more accurately than a CNN that is trained on natural images.

1.5 Summary

Our visual system is most sensitive to temporal modulations in the sensory input and therefore is in its core spatiotemporal. CNNs are often used to model the visual system, although CNNs work with purely spatial images and neglect signal changes over time. We wanted to test if a CNN benefits from including temporal information, that result from ocular drift, in the training images. To accomplish this, we generated a drift trajectory that simulates the moving eye during fixation. We created two nearly identical but slightly offset images from one training image. The offset is determined by the generated drift trajectory. The pixel-wise difference of both slightly offset images can be interpreted as the temporal change of the retinal image as it drifts across the photoreceptors. We call the resulting image difference image. A difference image holds information about temporal changes but is still purely spatial, hence CNNs can be trained on difference images.

To test our hypotheses, we trained two CNNs on the same dataset. We trained the first CNN on grayscale images (CNN-gray) and the second on difference images that we derived from ocular drift (CNN-diff). The accuracy scores on the test set of both CNNs revealed no support for our assumption that considering visual modulations as would occur from ocular drift is beneficial for object recognition in CNNs. Furthermore, our results indicate that considering visual transients as would result from ocular drift do also not facilitate object recognition on

images with sketches of objects. Finally, we discuss our results and potential limiting factors of our study.

2 Methods

2.1 Overview

In this thesis, we want to investigate whether CNNs benefit from considering the temporal modulations caused by ocular drift. For this, we compare object recognition performance of two CNNs. The first CNN (CNN-gray) we train on natural grayscale images. The second CNN (CNN-diff) we train on images that also take the sensitivity of the visual system to temporal modulations into account. In our simulation the temporal modulations result from ocular drift. To simulate the effect of ocular drift on the retinal image, we generate a trajectory that represents the moving eye during fixation. We apply the generated drift trajectory to the input images to approximate the temporal changes of the sensory input between two consequent drift motions. This approximation we call difference image.

In the following, we describe how we approximate the effect of ocular drift on images. We describe the dataset that contains the training images for both CNNs. We also describe the dataset of images that we use to test object recognition performance of both CNNs on sketches of objects. Afterwards, we describe the architecture of both CNNs and the general training procedure that both networks share. Next, we go into details of how we prepare the images individually for each of the CNNs. Last, we describe both of our experiments that we set up to test our hypotheses.

2.2 Modelling the Effect of Drift on Images

2.2.1 Computational Model of Drift

Ocular drift creates a continuously moving retinal image. We simulate the shift of the retinal image caused by drift as a two-dimensional Brownian motion process (Kuang et al., 2012). To be accurate, we need to choose fitting parameters for the Brownian motion which samples from a two-dimensional zero-mean normal distribution. The standard deviation controls

the amplitude of single steps in the drift trajectory and is calculated from the number of dimensions n, a diffusion coefficient D and the sampling frequency f. We choose $n = 2$ and $D = 20 arcmin^2 s^{-1} = \frac{20}{3600} deg^2 s^{-1}$ (Kuang et al., 2012). The visual system is most sensitive for temporal frequencies in the range of 5 to $30 Hz$ (Frishman et al., 1987). In order to account for that, we need to sample with at least double the frequency of this range (Shannon, 1949). We decided on a sampling frequency of $f = 100 Hz$. This results in a standard deviation of $SD = \sqrt{n * dt * D}$, where $dt = f^{-1}$ is the time between two consecutive steps. The number of steps to simulate as a Brownian motion process is calculated by f times an observation time span T. We choose $T = 200 ms$ which approximately matches the duration of inter-saccadic intervals (Martinez-Conde et al., 2004).

The generated drift trajectory is defined in degree (deg). Referring to the angle that an object covers in our visual field is convenient because, by doing so, we directly measure the size of the proximal stimulus, i.e. the object in the retinal image. Visual angle includes the size and distance of the object, i.e. distal stimulus. To be able to apply a trajectory to images, which are defined in pixel (px), we need to convert the trajectory from deg to px by multiplying with a conversion factor (PPD = "pixel per degree") that defines how many pixels correspond to one degree of the retinal image. We decided on $PPD = 40$ which is large enough to detect the effect of drift but small enough to have sufficiently large input images. Later, we will train our networks on $224 \times 224 px$ images. They will represent a retinal image of $5.6 \times 5.6\ deg$ which is about the size of the fovea with about 5 deg in diameter (Kolb, 1995). Additionally, the trajectory is quantized to whole pixel values by rounding to the nearest integer in both dimensions, as visualized by the green (sampled) and red (quantized) trajectories in **Figure 2.1**.

2.2.2 Approximation Model via Difference Images

Simulating ocular drift and its effect on visual processing is a crucial step in our experiments. Here, we are focusing on the assumption that drift highlights luminance edges in the temporal domain (Kuang et al., 2012). Further challenge exists because CNNs only process spatial

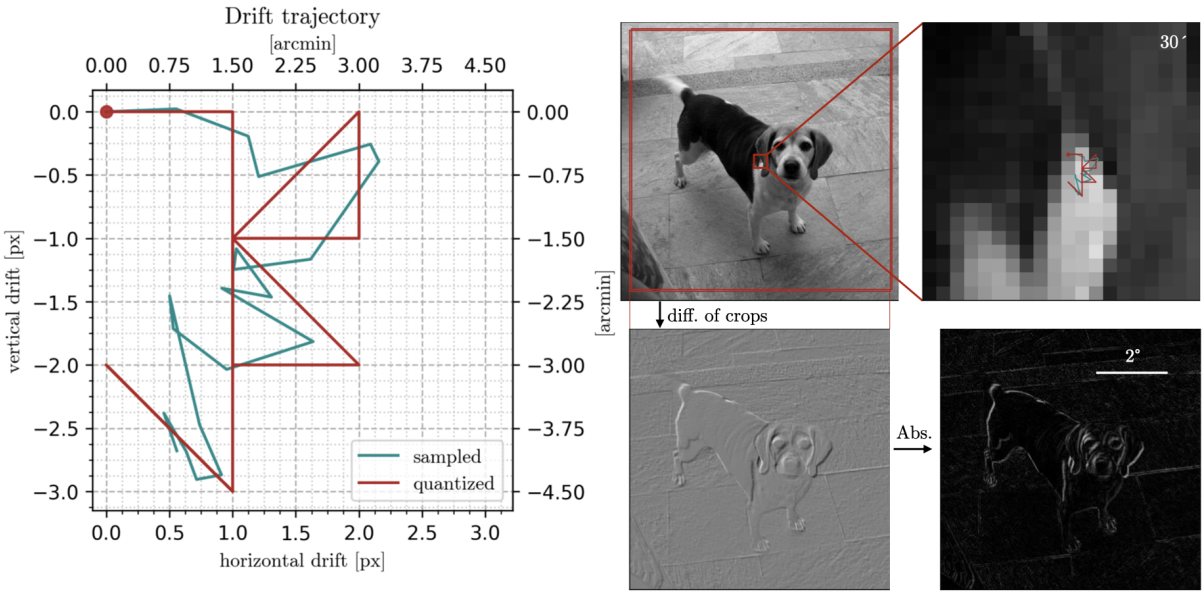

Figure 2.1: *Difference image creation.* (**Left**) One example of a sampled and quantized drift trajectory. A new trajectory is generated for each training sample individually. (**Right**) Our difference images are the result of subtracting a slightly offset crop from the center crop of the original image. The offset is determined by the first quantized trajectory step that leaves the coordinates origin. Notice that after computing the absolute values of our difference images, areas with high variance in luminance are highlighted, i.e. luminance edges.

information. This limitation urges us to encode luminance edges, that we assume to be highlighted in the temporal domain, in a spatial format. We came up with a computationally inexpensive approximation that we call difference image.

First, we generate a quantized drift trajectory (see **Section 2.2.1**). We perform two crops from the original image. The first is a center crop. The second crop is slightly offset from the center. The offset is determined by the first quantized drift step that leaves the coordinates origin, which makes it highly unlikely that both crops are identical. The sequence of the two cropped images represent a time series of the retinal image. By subtracting both images, we get the temporal change of the moving retinal image in a spatial format that can be processed by a CNN (see **Figure 2.1** "diff. of crops"). Finally, we calculate the pixel-wise absolute values (see **Figure 2.1** "Abs."). In the resulting difference image, luminance edges are encoded as non-zero pixel values because, at a luminance edge, the difference of two nearby pixels is larger than in an equiluminant area. Areas of homogeneous luminance will likewise have zero-values.

Notice, that this is independent from the area's original luminance, as can be seen in the final difference image; the dog's rear body and the floor tiles share (near) zero-values, although in the original grayscale image they hold very different luminance values. Luminance edges are detected best if they are oriented orthogonally to the drift direction.

Creating the difference images includes performing crops from a source image resolution to a target image resolution. This is the reason, why the image gets smaller in **Figure 2.1** "diff. of crops". We wanted the source image resolution of the difference image creation process to be as high as possible and such that as little as possible of the images get upsampled. Therefore, we chose the source resolution to be $320 \times 320px$. Beyond, the train set resolution drops rapidly (**Figure 2.2**). We chose $300 \times 300px$ as target image resolution, which gives us a $10px$ buffer in each direction for the offset crops when creating the difference image. We never exceeded this buffer in the entire training process.

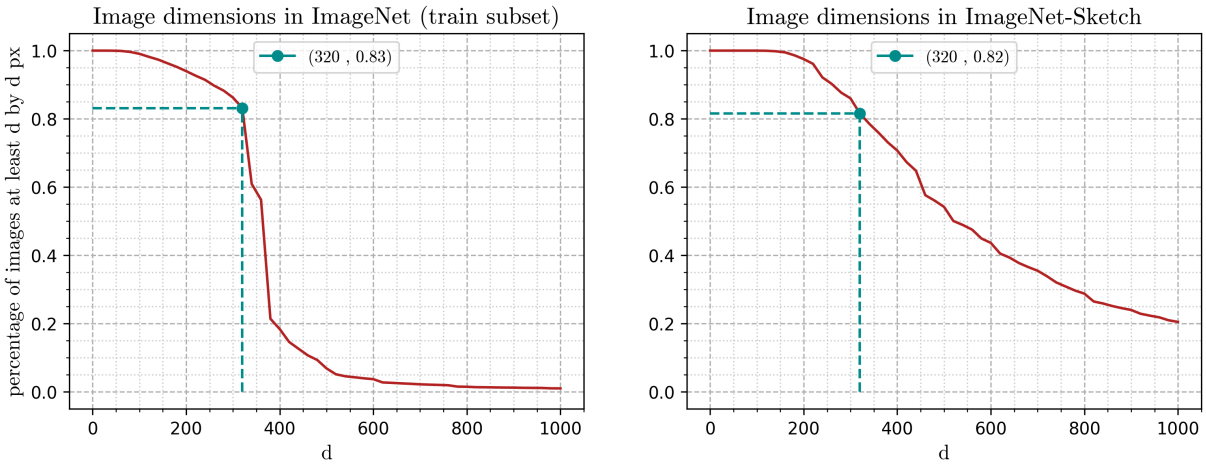

Figure 2.2: *Dataset resolutions.* (**Left**) Roughly 83% of our ImageNet training images are at least $320 \times 320px$ large. (**Right**) Coincidentally, the ImageNet-Sketch dataset shows similar percentage for this specific resolution.

2.3 Datasets

We use the same dataset to train the CNN, which is trained on natural grayscale images (CNN-gray), and the CNN, which is trained on the difference images (CNN-diff), to make both networks more comparable. We decided on using the well known ImageNet dataset (Deng

et al., 2009). Additionally, we use the ImageNet-compatible ImageNet-Sketch dataset (Wang et al., 2019) to see how well both networks perform on recognizing objects in sketch images.

2.3.1 ImageNet

We need a lot of labeled images that provide a variety of relevant class-specific features for the CNNs to learn how to differentiate several classes from each other. In computer vision, several benchmark datasets for object recognition exist. Here we use ImageNet (Deng et al., 2009). The ImageNet dataset consists of roughly 15 million high-resolution images in more than 21,000 different categories and varying aspect ratios. Because of limited computational resources, we only use a subset of ImageNet; the same subset that is also used in the ImageNet Large-Scale Visual Recognition Challenge 2012 (Russakovsky et al., 2015). The ILSVRC-2012 subset consists of 1000 different classes with roughly 1.28 million images in the train set, 50.000 images in the validation set and 100.000 images in the test set.

The labels of the test set are not publicly available, so we extracted a custom test set from the train set. Our test set consists of 100.000 images and shows the same label ratios as the train set, that way classes are as much represented in the test set as they are in the train set. Our train set, therefore, remains with roughly 1.18 million images. For simplicity, we refer to our dataset as ImageNet. Some samples can be seen in **Figure 2.3**. Note that the dataset consists of photographs. Images therefore do not display the isolated objects but also include visual background and surrounds of the object.

2.3.2 ImageNet-Sketch

Since we hypothesize that the CNN-diff, which we will train on images we derive from ocular drift, recognizes sketches of objects better than the CNN-gray, we want to evaluate the generalization performance of both CNNs on sketches. To receive reliable evaluations, we need a fairly large amount of sketch images. ImageNet-Sketch is a dataset consisting of roughly 50.000 sketches labeled with the 1000 ImageNet labels, which makes the dataset compatible to our ImageNet-trained CNNs (Wang et al., 2019). The provided high-resolution

Figure 2.3: *ImageNet samples.* The displayed images are quadratic center crops resized to 224 × 224*px*. Original images show various aspect ratios and resolutions.

sketches are grayscale images with dark drawing on bright background and come in varying aspect ratios. Some samples can be seen in **Figure 2.4**. Note that the sketches show different degrees of abstraction, i.e. some images show clean line-drawings and some show more details like shadows and texture.

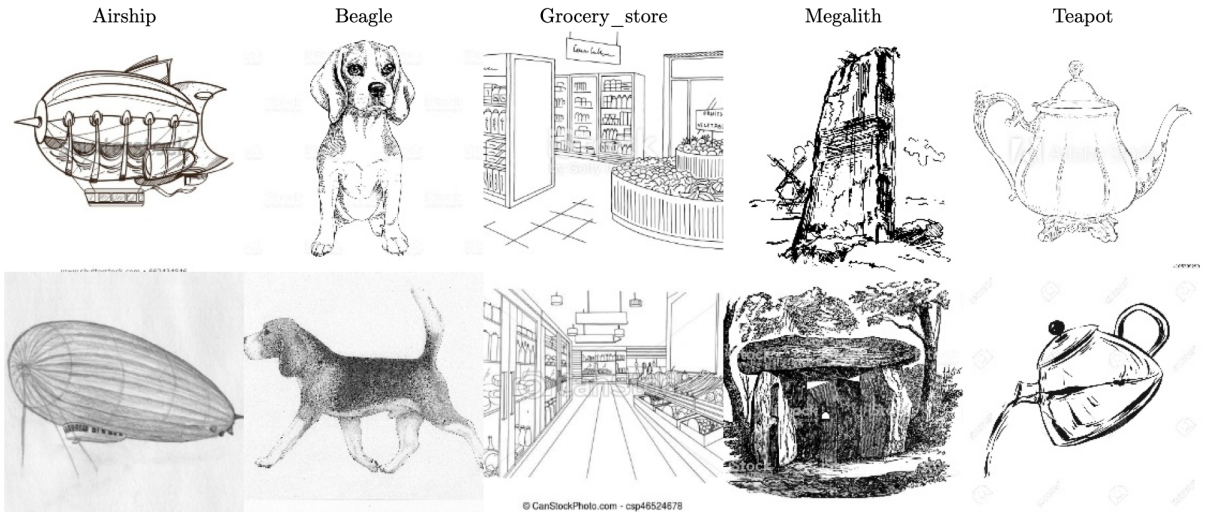

Figure 2.4: *ImageNet-Sketch samples.* The displayed images are quadratic center crops resized to 224 × 224*px*. Original images show various aspect ratios and resolutions.

2.4 CNN Architecture and Training Procedure

The CNN-gray, which we train on standard grayscale images, and the CNN-diff, which we train on difference images derived from ocular drift, share the same network architecture (**Figure 2.5A**), which is highly inspired by the well known AlexNet architecture (Krizhevsky et al., 2012). In total, each of the networks consist of five convolutional layers followed by three fully-connected layers. Both CNNs minimize the sparse categorical crossentropy[1] measure.

We use batch normalization after each convolutional layer to speed up learning (Santurkar et al., 2018). Batch normalization estimates mean and variance from a batch during training and transforms each sample to a zero-mean input with unit standard deviation. We use 3×3 max-pooling layers with strides of 2 in vertical and horizontal direction. We use the Rectified Linear Unit (ReLU) as activation function on the output of the five convolutional layers and the two fully-connected hidden layers. We apply the softmax activation to the output of the last layer, which represents the probability distribution of how likely each class label is, given an input.

The first convolutional layer applies 96 kernels of size $11 \times 11 \times 1$ with strides of 4 to the $224 \times 224 \times 1$ (achromatic) input image. The output of the first convolutional layer is batch normalized and max-pooled. The second convolutional layer applies 256 kernels of size $5 \times 5 \times 96$ with strides of 1. The output of the second convolutional layer is batch normalized and max-pooled. The third convolutional layer applies 384 kernels of size $3 \times 3 \times 256$ with strides of 1. The output of the third convolutional layer is batch normalized. The fourth convolutional layer applies 384 filters of size $3 \times 3 \times 384$ with strides of 1. The output of the fourth convolutional layer is batch normalized. The fifth convolutional layer applies 256 filters of size $3 \times 3 \times 384$ with strides of 1. The output of the fifth convolutional layer is batch normalized and max-pooled.

We flatten the output of the last convolutional layer with a size of $5 \times 5 \times 256$ into a 6400-vector which is passed into two consecutive fully-connected layers that consist of 4096 neurons. Both fully-connected layers are followed by 50%-dropout which randomly omits half of the neurons

1 We do not use OneHot-encoded class labels, therefore we cannot use the "normal" categorical crossentropy

during training. This has been shown to help CNNs to learn more robust features (Hinton et al., 2012). During testing, dropout does not show any effect. Finally, the last fully-connected hidden layer is fully-connected to the 1000-vector output layer.

Overall, we prepared the two CNNs for training as described in Krizhevsky et al. (2012). We initialized the biases of the first and third convolutional layer and the output layer with zeros. The biases of the three remaining (second, fourth and fifth) convolutional layers, as well as of the two fully-connected hidden layers, are initialized with ones. At inference time, i.e. while predicting labels, a neuron's bias represents its baseline activation because it provides a constant signal to the neuron's activation function. By setting the biases of some neurons to one, we provide positive inputs to the neurons' ReLUs, which accelerates the training process in the first few epochs. As loss function, we used the sparse categorical cross entropy loss between the predicted labels and the true labels. We minimized this loss function with a stochastic gradient descent optimizer with momentum of 0.9 and, in contrast to Krizhevsky et al. (2012), no weight decay. We shuffled the training data prior to each epoch and then fed it to the networks in batches of 128 samples.

Both CNNs, the CNN-gray and CNN-diff, were trained on one GTX 1660S 6GB GPU for a total of 80 epochs each with an initial learning rate of 0.01. We decided on shrinking the learning rate by dividing it by 10 as soon as the validation accuracies stagnated. This way, the networks' parameters get adjusted by smaller amounts which enables them to learn finer-grained discriminants of the classes. We reduced the learning rate after 45 and 65 epochs because previous training runs have indicated that this was the time when validation accuracies stagnated.

To evaluate object recognition performance of the two CNNs, we calculated and monitored top-1 accuracy (accuracy, for short) and top-5 accuracy. Top-1 accuracy refers to the percentage of images a network correctly classifies. Top-5 accuracy considers an image correctly classified if its true label is among the five labels with the highest probabilities predicted by the networks.

In order to monitor how well training performance generalized to unseen images, we calculated the accuracy and top-5 accuracy on the validation set.

2.5 Data Preprocessing and Augmentation

Artificial neural networks come with many adjustable parameters that get adjusted to minimize the prediction error during training. This does not imply that a CNN performs well on previously unseen data that was not included in the training data. A network that performs well on the training data but does not generalize well to new data is considered to be overfitted (Hawkins, 2004). A lot of research is done to find methods on how to prevent networks from overfitting, e.g. constraining parameters (Moody et al., 1992) and deciding which parameters to adjust in each step of the learning process (Hinton et al., 2012; Wan et al., 2013).

Foundation of our training procedures is the fairly large amount of over 1.18 million images from our train set. Nevertheless, training the network on even larger datasets further reduces the risk of overfitting. One option to circumvent data shortage is to gather more manually-labeled images. This would be a very time-consuming undertaking, which urges us to artificially enlarge our dataset. Data augmentation summarizes such methods that mainly consist of label-preserving transformations (Wong et al., 2016), e.g. flipping, rotating or adjusting pixel intensities of images. Shorten and Khoshgoftaar (2019) report the effectiveness of data augmentation techniques in counteracting overfitting and thus increasing the generalization performance of CNNs. Further, we benefit from this group of transformations because they can be applied to a batch of images parallel to the network still training on a previous batch, thus reducing the overall latency of our pipelines. Finally, the transformations are computed on demand and do not have to be stored on disk (Krizhevsky et al., 2012).

Another beneficial effect of data augmentation is that the object recognition of the CNN will be more robust to changes that naturally occur. An object might be oriented differently relative to the observer each time they see it. The lighting situation of the object might change over time, e.g. many objects occur in day light and at night. Despite the variety of possibilities objects are presented to us, the observer, we are able to recognize objects. This tolerance of our visual system in detecting and recognizing objects makes the above mentioned data augmentation techniques an intuitively plausible method to enrich image datasets for object

recognition tasks. Here, we have used multiple augmentations which we will describe in detail for the CNN-gray and CNN-diff in the following (**Figure 2.5 B-C**).

2.5.1 CNN-GRAY

The training pipeline for the CNN-gray network is nearly identical to Krizhevsky et al. (2012). We are interested in the luminance information of the images. For that purpose, we convert the colour images to grayscale by averaging the pixel intensities over the RGB channels. Both datasets consist of images with varying aspect ratios and resolutions whereas the architecture of both networks only allows for same shaped inputs. In order to solve this mismatch, we resize the images so that the shorter side is $256px$ long and extract the center square of size $256 \times 256px$ ($= 6.4 \times 6.4deg$). That way, we can resize images without distorting the displayed objects. Also, we assume the object of interest to be located near the center of the image, therefore little information about the object is lost by performing a squared center crop. When reading image files, raw pixel values come in range from 0 to 255[2] by default. A neural network, that is trained on a large value range, is likely to learn high biases which might impair the generalization performance of the network. In order to prevent the CNN from getting strongly biased, in contrast to Krizhevsky et al. (2012), we perform a Min-Max normalization of the pixel values to $[0, 1]$ (Jayalakshmi & Santhakumaran, 2011).

Humans are able to recognize objects, even if only a subsection of the object is presented (e.g. We can identify trees by only looking at their leaves). To simulate this behaviour, we extract a random $224 \times 224px$ ($= 5.6 \times 5.6deg$) crop from the image. This allows us to convert the image to the network's expected input shape while simultaneously introducing more variation to the training set. Also, this reduces the location bias in the network, i.e. the network does not expect the objects to be located exactly at the center of the image (Shorten & Khoshgoftaar, 2019). Next, we flip the image along its horizontal axis by a 50% chance. Finally, we adjust the contrast of our images by subtracting the mean pixel intensity from an image, multiplying the zero-mean image with a random factor $0.5 \leq \alpha \leq 1.5$ and adding the (previously subtracted)

2 Each pixel in each channel is encoded as 8-bit unsigned integer, which results in 256 different values a pixel can hold per channel

mean again. Note, that, after adjusting the contrast, pixel values can exceed the range of $[0, 1]$. Especially, this is the case with images that provide high variance in pixel intensities.

At validation and test time, we check how well the network has learned general class-specific features from the train set. During validation and testing, we use a similar preprocessing of the input images but without data augmentation. As Krizhevsky et al. (2012) has shown, performing data augmentation during validation and testing can result in higher accuracy scores. Here, we do not aim for the highest possible accuracy score but we rather want to compare the performance of two CNNs. Hence, we discard data augmenting operations during validation and testing. The training pipeline preprocesses images in a specific scheme at train time. Therefore, the network expects images to arrive in that specific scheme at validation and test time. Nevertheless, we replace data augmenting operations with deterministic operations. Similar to the training pipeline, we thus replace the random $224 \times 224px$ ($= 5.6 \times 5.6deg$) crop by a center crop of same size. Further, we did not perform any horizontal flips and did not adjust the image contrast. The remaining preprocessing operations are identical to the CNN-gray's training pipeline.

2.5.2 CNN-DIFF

To make the prediction performances of both CNNs as comparable as possible, we constructed the training pipeline of the CNN-diff similar to the training pipeline of the CNN-gray.

We first convert the colour images to grayscale (luminance information). We resize the image so that the shorter side is $320px$ long and extract the center $320 \times 320px$ ($= 8.0 \times 8.0deg$) patch. Next, we normalize the pixel values to $[0, 1]$. We create $300 \times 300px$ ($= 7.5 \times 7.5deg$) difference images which only contain information about luminance edges of the original image (see **Section 2.2.2** for details). Since the underlying drift trajectory is random, the creation of the difference images can be considered an additional data augmenting operation. The pixel values of the difference image are the absolute differences of two slightly shifted versions of the original image, and thus have a range between zero and one.

We resize the difference image to $256 \times 256px$ (= $6.4 \times 6.4deg$) to match the training pipeline of the CNN-gray. From here on, both training pipelines are identical. We extract a random $224 \times 224px$ (= $5.6 \times 5.6deg$) crop of the difference image. Next, we flip the image by a 50% chance and adjust the contrast of the (zero-mean) difference image by a factor of $0.5 \leq \alpha \leq 1.5$ (see CNN-gray training pipeline for details).

At validation and test time, the CNN-diff expects difference images because it is trained on difference images. For that reason we create a random $300 \times 300px$ (= $7.5 \times 7.5deg$) difference image of the squared and normalized $320 \times 320px$ (= $8.0 \times 8.0deg$) grayscale image. We resize the resulting difference image to $256 \times 256px$ (= $6.4 \times 6.4deg$) to further match the validation and test pipeline of the CNN-gray network. Finally, we extract the center $224 \times 224px$ (= $5.6 \times 5.6deg$) patch of the image.

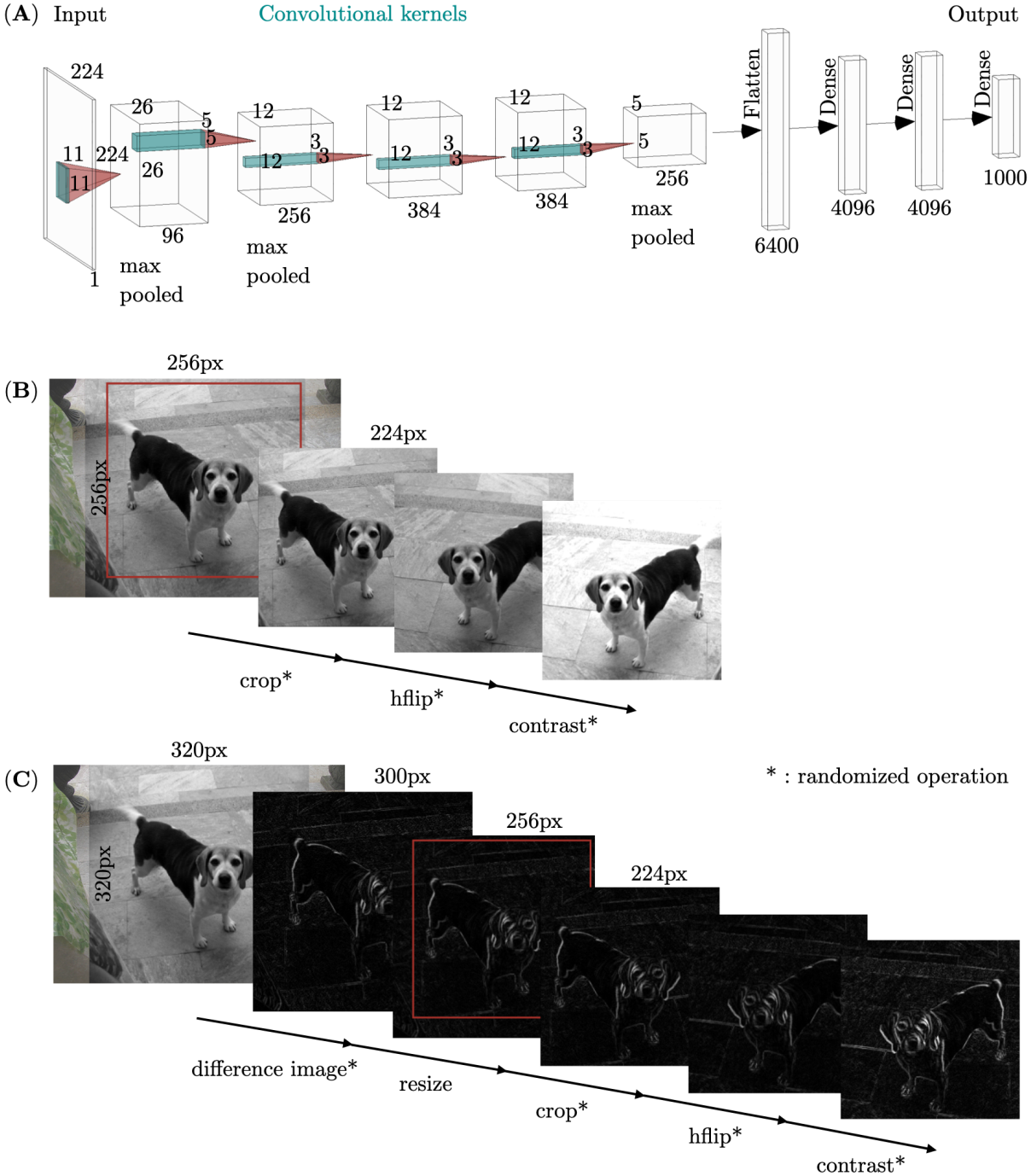

Figure 2.5: *CNN Architecture and Training Pipelines.* (**A**) The CNN architecture scheme is created with LeNail (2019). Both networks (CNN-gray and CNN-diff) share the same architecture. The output of each convolutional layer is batch-normalized. The output of the first, second and fifth convolutional layers is max-pooled, additionally. (**B - C**) Training pipelines from raw RGB-valued images to final net inputs of CNN-gray (**B**) and CNN-diff (**C**).

2.6 Experiment 1

In the first experiment, we test our hypothesis that the CNN-diff learns more robust features from the train set compared to the CNN-gray. Hence, we expect the CNN-diff to generalize better to previously unseen images. While we use the validation set to monitor the generalization performance of our networks after each training epoch, we use the test set to evaluate the generalization capability of both CNNs after training. The test set is larger than the validation set which provides more reliable generalization scores. We trained both CNNs on the exact same train set taken from ImageNet. We prepared the training datasets according to the CNN-specific training pipelines. We prepared the test images according to the CNN-specific validation and test pipelines. To determine if the CNN-diff generalizes better than the CNN-gray, we compare top-1 accuracy and top-5 accuracy scores of both CNNs on the test set. High top-5 accuracy scores would indicate that a CNN recognizes the superordinate category of the objects (e.g. "that's a dog") well. Whereas a CNN with high top-1 accuracy scores could distinguish between true classes (e.g. "that's a beagle").

2.7 Experiment 2

In the second experiment, we investigated whether training a CNN on images derived from ocular drift (CNN-diff) improves its object recognition performance on sketch images. We expect the CNN-diff to predict class labels of sketch images more accurately than the CNN-gray because the CNN-diff is trained on difference images, in which edges are highlighted, and sketches mainly contain information about the shape of objects rather than their achromatic surface colour. To test this hypothesis, we reused the trained networks taken from experiment 1 and calculated top-1 accuracy and top-5 accuracy scores for both CNNs on the ImageNet-Sketch dataset. We prepared the sketch images according to the CNN-specific validation and test pipelines.

3 Results

3.1 Object Recognition Performance

Our results of both experiments and the validation curves of the loss and accuracies throughout training are summarized in **Figure 3.1**. We have found that the object recognition performance of the CNN-gray is better than the performance of the CNN-diff after training it on ImageNet (top-1 accuracy: 54.0% vs. 37.8%; top-5 accuracy: 76.8% vs. 61.5%, see **Figure 3.1C**). This gap in accuracies is relatively constant throughout training as the validation accuracy curves indicate (see **Figure 3.1B**) and the inverse can be observed for the validation losses (see **Figure 3.1A**).

The CNN-gray achieves top-1 and top-5 accuracy scores on the ImageNet-Sketch dataset of 13.8% and 26.9%. The CNN-diff's accuracy scores on the sketch set are 10.3% and 21.9% (see **Figure 3.1C**).

Further, we observe a large gap between the training and validation loss scores with both CNNs (see **Figure 3.1A**). These gaps indicate that both CNNs began overfitting in early stages of the learning process. Both, validation and test accuracy rates, measure how well a network generalizes to previously unseen data. The test accuracy scores are more reliable because test sets usually contain more samples than validation sets. Nevertheless, test accuracy rates on our ImageNet test set of both CNNs surpass the corresponding validation accuracy rates by $1 - 2\%$.

3.2 Smoothness of Validation Curves

After training both networks, it seemed to us that the validation accuracy curve of the CNN-diff was smoother than the validation accuracy curve of the CNN-gray (see **Figure 3.1B**). Therefore, we wanted to test whether the validation accuracy curve of the CNN-diff is statistically smoother than the validation accuracy curve of the CNN-gray. Since both curves have a very similar overall shape, we defined the smoothness based on the absolute accuracy difference between consecutive epochs of the normalized validation accuracy curves. The

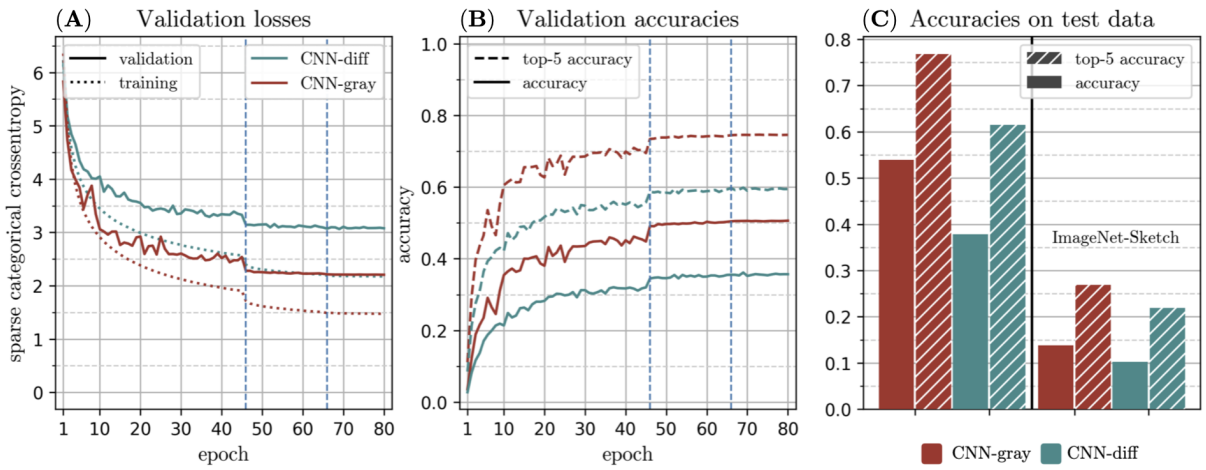

Figure 3.1: *Training results.* CNN-gray scores in red, CNN-diff in green. (**A**) Both networks started to overfit from early epochs on. (**B**) Validation accuracies after 80 epochs (top to bottom): 0.746, 0.594, 0.506, 0.357. (**C**) Accuracies on test data (left to right): 0.540, 0.768, 0.378, 0.615, 0.138, 0.269, 0.103, 0.219. Vertical dashed lines in (**A**) and (**B**) mark epochs, from which on we reduced the learning rate.

smaller this absolute difference, the smoother the curve. To compute this smoothness, we first added a leading zero to both validation accuracy curves (both validation accuracy curves start with non-zero values because the first validation score was acquired after the first training epoch). We than normalized each curve individually to a range of $[0, 1]$. We did so because we are interested in the variance of both curves and the overall higher accuracy scores of the CNN-gray would effect the step size between two epochs. Finally, we extracted the absolute differences between two consecutive epochs.

Since the absolute differences of both curves are not normally distributed, we performed the non-parametric Mann-Whitney U test to investigate whether the absolute differences of the CNN-diff curve are statistically smaller than the differences of the CNN-gray curve (McKnight & Najab, 2010). However, we have found no significant difference in the smoothness of both curves ($U = 2805.0$, $p = .178$).

4 Discussion

Fixational Eye Movements are well known for carrying the retinal image across photoreceptors and thus maintaining vision by preventing photoreceptors to adapt to constant stimulation (Martinez-Conde et al., 2004). Recent studies suggest that FEMs, in particular ocular drift, serve the purpose to extract features, such as luminance edges, from the retinal image before the signal reaches the cortex (Kuang et al., 2012; Rucci & Victor, 2015). Our visual system is most sensitive to temporal modulations in the sensory input and therefore is in its core spatiotemporal (Rucci & Victor, 2015). CNNs are often used to model the visual system (e.g. Krizhevsky et al., 2012), although CNNs work with purely spatial images and neglect signal changes over time. We wanted to test on natural images and sketches if the object recognition performance of a CNN can be improved by including temporal information, that result from ocular drift, in the training images.

We have come up with an approximation that represents temporal modulations of the retinal image in a purely spatial format, which makes it possible for CNNs to train on. We call this approximation difference image. To test if object recognition performance of a CNN could be improved by considering the effect of ocular drift in the training images, we have trained two identical CNNs on ImageNet and compared their generalization performance on natural images and on sketches. We have trained the first CNN on standard grayscale images (CNN-gray). We have trained the second CNN on difference images (CNN-diff). The CNN-gray achieved higher accuracy scores than the CNN-diff on natural images of the test set. The CNN-gray also achieved higher accuracy scores than the CNN-diff on sketches of objects.

4.1 Drift Based Edge Extraction Does Not Facilitate Object Recognition in CNNs

In our experiment 1, the CNN-gray, which we have trained on standard grayscale images, performed better on our test set of natural images than the CNN-diff, which we have trained

on difference images. These results do not support our assumption that including temporal information, that result from ocular drift, in the training images facilitates object recognition for natural images in CNNs. Our visual system is not sensitive to static input but only to input that changes over time (Ditchburn & Ginsborg, 1952; Riggs & Ratliff, 1952; Yarbus, 1967). Kuang et al. (2012) and Rucci and Victor (2015) suggest that temporal modulations caused by FEMs, in particular ocular drift, are involved in extracting edges in the retina. We had assumed that the CNN-diff would perform better than the CNN-gray because we have trained the CNN-diff on difference images, which we have created to include temporal modulations caused by ocular drift in the training images. Extracting edges while simultaneously reducing redundant information in the retina (Kuang et al., 2012) might be a neccessary stage in order to reduce the metabolic costs of processing visual information (Laughlin et al., 1998).

The CNN-diff has been trained on difference images that we have created to highlight luminance edges and reduce information in homogeneous areas in grayscale images. Therefore, we have trained the CNN-gray on images that contain more information than difference images. Biological systems might depend on reducing the information from the sensory input to lower the metabolic cost that comes with neural information processing (Laughlin et al., 1998). Since CNNs do not depend on being energy-efficient, CNNs might not benefit from retina-like edge extraction based on ocular drift like we would expect from biological systems. Instead, by reducing information in the input images, the input images could be reformatted to lower dimensional inputs. To learn the structure of lower dimensional data, a smaller network might be sufficient. The size of an artificial neural network still is decisive if the network is deployable in systems with limited resources, such as embedded systems (Roth et al., 2020).

In the process of creating a difference image, we perform two crops of an original image. The original image is of size $320 \times 320px$, the crops and thereby the difference image are of size $300 \times 300px$. This means, that roughly 12% of the image is discarded to create a difference image. The percentage is slightly off because it depends on the random offset between both crops. This only applies to the CNN-diff. The CNN-gray does not depend on difference images. Creating a difference image is performing a trade-off between losing spatial information (~12%) in favour of the gained temporal information. Retrospectively, we could have chosen a

smaller buffer for cropping because the offset hardly hit $3px$ in one direction. If we decided to crop $314 \times 314px$ (buffer of $3px$ in each direction) instead, we would have lost less than 4% of available spatial information while not losing any temporal information.

We have approximated the effect of ocular drift on sensory input as difference image. This approximation is greatly simplified because we have neglected precise response behaviour of retinal ganglion cells. Casile et al. (2019) have proposed a more sophisticated and biologically plausible model of spatial and temporal response behaviour of retinal ganglion cells. Using such a model to simulate how edge extraction is initiated by ocular drift in the retina might result in more comparable accuracy scores between a CNN, that is trained on standard grayscale images, and a CNN, that is trained on images derived from ocular drift.

We have chosen a CNN architecture that only processes single images. A more plausible approach would have been to use a network architecture that classifies sequences of images, i.e. videos, because drift adds a temporal dimension to even completely still images (Kuang et al., 2012). A possible architecture is described by Joe Yue-Hei Ng et al. (2015). Input data consists of an ordered sequence of images. The network might benefit from the additional temporal variability when applying drift to an image, in comparison to static grayscale images which are time-invariant.

The datasets contain images of varying resolutions. The architecture of both networks, CNN-gray and CNN-diff, only allow images of one resolution, $224 \times 224px$. When resizing images to this specific resolution we change the original data. This is of particular interest when we are resizing to a larger image size because the additional space (pixels) needs to be filled depending on the original image. This often happens by interpolation with a gaussian kernel, which smooths edges and creates a blurrier image. When we create a difference image from a blurry image, the highlighted luminance edges appear weaker because the luminance differences between nearby pixels are smaller than with non-blurry images.

We posthoc assumed, that the validation accuracy curve of the CNN-diff was significantly smoother than the validation accuracy curve of the CNN-gray. We disproved our assumption by a Mann-Whitney U-Test. A smoother validation curve could have indicated that the CNN-diff had learned more durable features from the training data, that the network did not "forget"

during training. Following this interpretation of the validation curve landscape, a smooth validation accuracy curve indicates that a network builds upon the already learned features because discarding already learned and learning new features would most possibly result in varying accuracy rates.

4.2 Drift Based Edge Extraction Does Not Facilitate Object Recognition for Sketches in CNNs

In our Experiment 2, we have compared the object recognition performance on sketches of the CNN-gray and CNN-diff. The CNN-gray, which we have trained on standard grayscale images, performed better in classifying sketches of objects than the CNN-diff, which we have trained on difference images. These results do not support our assumption that including temporal modulations, that result from ocular drift, in the training images would facilitate object recognition for sketches of objects in CNNs. Singer et al. (2020) reported that CNNs, which are trained on natural images, perform poorly in recognizing objects in more abstract data, such as sketches. We have assumed, that the CNN-diff would perform better than the CNN-gray because we have trained the CNN-diff on difference images which, similar to sketches, accentuate edges over surfaces and textures.

Geirhos et al. (2019) have reported that ImageNet-trained CNNs, such as the CNN-gray, make classification decisions based on an object's texture rather than its shape. By training a CNN on images that display objects with class-specific shapes and conflicting textures within the classes, the CNN can be forced to learn object representations based on their shape rather than their texture, which increases both, the CNN's accuracy and robustness against perturbation. Singer et al. (2020) have reported that fine-tuning ImageNet-trained CNNs on ImageNet-Sketch improves object recognition performance on sketches and line-drawings of objects. Fine-Tuning refers to the procedure that some layers of a pretrained (e.g. trained on ImageNet) network get trained on another dataset (e.g. ImageNet-Sketch) to learn more versatile features. Note, that, for testing, Singer et al. (2020) have used sketches that are not contained in ImageNet-Sketch.

To test the object recognition performance of both CNNs (CNN-gray and CNN-diff) on sketches of objects, we have calculated accuracy scores of both CNNs on the ImageNet-Sketch dataset. Geirhos et al. (2019) have reported that ImageNet-trained CNNs, such as the CNN-gray, focus on texture information to distinguish between objects. Additionally, We have trained the CNN-diff on difference images that highlight edges similar to sketches. Sketches mainly contain information about the shape of objects and neglect their texture, thus, we have expected the CNN-diff to perform better on sketches than the CNN-gray. We couldn't support this assumption. The ImageNet-Sketch might not have been the ideal benchmark dataset because images contained in ImageNet-Sketch show different levels of details. As can be seen in **Figure 2.4**, image styles vary from cartoon-like to elaborate drawings with shadows and texture. We might have received more expressive results if we had used a smaller selection of "clean" sketches like Singer et al. (2020) have done.

To further investigate the idea that CNNs could benefit from retina-like edge extraction based on fixational eye movements, one could fine-tune a ImageNet-trained CNN on difference images and compare object recognition performance of the fine-tuned CNN and the ImageNet-trained CNN on sketches similar to Singer et al. (2020). It might also be possible to substitute difference images with a biologically more plausible model such as Casile et al. (2019).

4.3 Conclusion

Spatiotemporal processing of visual sensory input is essential for us to perceive our surrounds. Nevertheless, our results suggest that temporal processing only plays a minor role for CNNs and does not facilitate object recognition in CNNs. Spatiotemporal processing might be neccessary for biological systems because, at the cost of losing information, it might be metabolically more efficient. There are many points for improvement. Further research is needed to investigate the relevance of FEMs and spatiotemporal processing for object recognition in CNNs.

Bibliography

Anstis, S. (2013): Contour adaptation. *Journal of Vision*, *13*(2), 25–25. https://doi.org/10.1167/13.2.25

Bottou, L. (1991): Stochastic Gradient Learning in Neural Networks, 12.

Carpenter, R. H. S. (1988): *Movements of the eyes, 2nd rev. & enlarged ed.* [Pages: 593]. Pion Limited.

Casile, A., Victor, J. D., & Rucci, M. (2019): Contrast sensitivity reveals an oculomotor strategy for temporally encoding space. *eLife*, *8*, e40924. https://doi.org/10.7554/eLife.40924

Cireşan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2012): Deep Big Multilayer Perceptrons for Digit Recognition [Series Title: Lecture Notes in Computer Science]. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural Networks: Tricks of the Trade* (pp. 581–598). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_31

Cireşan, D. C., Meier, U., & Schmidhuber, J. (2012): Multi-column Deep Neural Networks for Image Classification [arXiv: 1202.2745]. *arXiv:1202.2745 [cs]*. Retrieved February 13, 2022, from http://arxiv.org/abs/1202.2745

Cornsweet, T. N. (1956): Determination of the Stimuli for Involuntary Drifts and Saccadic Eye Movements*. *Journal of the Optical Society of America*, *46*(11), 987. https://doi.org/10.1364/JOSA.46.000987

Croner, L. J., & Kaplan, E. (1995): Receptive fields of P and M ganglion cells across the primate retina. *Vision Research*, *35*(1), 7–24. https://doi.org/10.1016/0042-6989(94)E0066-T

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009): ImageNet: A Large-Scale Hierarchical Image Database, 8.

Ditchburn, R. W., Fender, D. H., & Mayne, S. (1959): Vision with controlled movements of the retinal image. *The Journal of Physiology*, *145*(1), 98–107. https://doi.org/10.1113/jphysiol.1959.sp006130

Ditchburn, R. W., & Ginsborg, B. L. (1952): Vision with a Stabilized Retinal Image. *Nature*, *170*(4314), 36–37. https://doi.org/10.1038/170036a0

Frishman, L. J., Freeman, A. W., Troy, J. B., Schweitzer-Tong, D. E., & Enroth-Cugell, C. (1987): Spatiotemporal frequency responses of cat retinal ganglion cells. *Journal of General Physiology*, *89*(4), 599–628. https://doi.org/10.1085/jgp.89.4.599

Gardner, M., & Dorling, S. (1998): Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric Environment*, *32*(14-15), 2627–2636. https://doi.org/10.1016/S1352-2310(97)00447-0

Geirhos, R., Rubisch, P., Michaelis, C., Bethge, M., Wichmann, F. A., & Brendel, W. (2019): ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness [arXiv: 1811.12231]. *arXiv:1811.12231 [cs, q-bio, stat]*. Retrieved December 30, 2021, from http://arxiv.org/abs/1811.12231

Hawkins, D. M. (2004): The Problem of Overfitting. *Journal of Chemical Information and Computer Sciences*, *44*(1), 1–12. https://doi.org/10.1021/ci0342472

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012): Improving neural networks by preventing co-adaptation of feature detectors [arXiv: 1207.0580]. *arXiv:1207.0580 [cs]*. Retrieved November 21, 2021, from http://arxiv.org/abs/1207.0580

Jayalakshmi, T., & Santhakumaran, A. (2011): Statistical Normalization and Back Propagation for Classification. *International Journal of Computer Theory and Engineering*, 89–93. https://doi.org/10.7763/IJCTE.2011.V3.288

Joe Yue-Hei Ng, Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., & Toderici, G. (2015): Beyond short snippets: Deep networks for video classification. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4694–4702. https://doi.org/10.1109/CVPR.2015.7299101

Kingma, D. P., & Ba, J. (2017): Adam: A Method for Stochastic Optimization [arXiv: 1412.6980]. *arXiv:1412.6980 [cs]*. Retrieved December 19, 2021, from http://arxiv.org/abs/1412.6980

Kolb, H. (1995): Simple Anatomy of the Retina. http://europepmc.org/books/NBK11533

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012): ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90. https://doi.org/10.1145/3065386

Kuang, X., Poletti, M., Victor, J. D., & Rucci, M. (2012): Temporal Encoding of Spatial Information during Active Visual Fixation. *Current Biology*, *22*(6), 510–514. https://doi.org/10.1016/j.cub.2012.01.050

Kuffler, S. W. (1953): DISCHARGE PATTERNS AND FUNCTIONAL ORGANIZATION OF MAMMALIAN RETINA. *Journal of Neurophysiology*, *16*(1), 37–68. https://doi.org/10.1152/jn.1953.16.1.37

Laughlin, S. B., de Ruyter van Steveninck, R. R., & Anderson, J. C. (1998): The metabolic cost of neural information. *Nature Neuroscience*, *1*(1), 36–41. https://doi.org/10.1038/236

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998): Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324. https://doi.org/10.1109/5.726791

LeNail, A. (2019): NN-SVG: Publication-Ready Neural Network Architecture Schematics. *Journal of Open Source Software*, *4*(33), 747. https://doi.org/10.21105/joss.00747

Martinez-Conde, S. (2006): Fixational eye movements in normal and pathological vision. *Progress in Brain Research* (pp. 151–176). Elsevier. https://doi.org/10.1016/S0079-6123(06)54008-7

Martinez-Conde, S., Macknik, S. L., & Hubel, D. H. (2004): The role of fixational eye movements in visual perception. *Nature Reviews Neuroscience*, *5*(3), 229–240. https://doi.org/10.1038/nrn1348

McKnight, P. E., & Najab, J. (2010): Mann-Whitney U Test. In I. B. Weiner & W. E. Craighead (Eds.), *The Corsini Encyclopedia of Psychology* (corpsy0524). John Wiley & Sons, Inc. https://doi.org/10.1002/9780470479216.corpsy0524

Møller, F., Laursen, M., Tygesen, J., & Sjølie, A. (2002): Binocular quantification and characterization of microsaccades. *Graefe's Archive for Clinical and Experimental Ophthalmology*, *240*(9), 765–770. https://doi.org/10.1007/s00417-002-0519-2

Moody, J. E., Hanson, S., & Lippmann, R. (1992): The Effective Number of Parameters: An Analysis of Generalization and Regularization in Nonlinear Learning Systems. *Advances in Neural Information Processing Systems*, *4*, 847–854.

Nandhini Abirami, R., Durai Raj Vincent, P. M., Srinivasan, K., Tariq, U., & Chang, C.-Y. (2021): Deep CNN and Deep GAN in Computational Visual Perception-Driven Image Analysis (D. S. Sarfraz, Ed.). *Complexity*, *2021*, 1–30. https://doi.org/10.1155/2021/5541134

Packer, O., & Williams, D. R. (1992): Blurring by fixational eye movements. *Vision Research*, *32*(10), 1931–1939. https://doi.org/10.1016/0042-6989(92)90052-K

Riggs, L., & Ratliff, F. (1952): The effects of counteracting the normal movements of the eye. *Journal of the Optical Society of America*, *42*(11), 872–873.

Rosenblatt, F. (1958): The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Roth, W., Schindler, G., Zöhrer, M., Pfeifenberger, L., Peharz, R., Tschiatschek, S., Fröning, H., Pernkopf, F., & Ghahramani, Z. (2020): Resource-Efficient Neural Networks for Embedded Systems [arXiv: 2001.03048]. *arXiv:2001.03048 [cs, stat]*. Retrieved February 21, 2022, from http://arxiv.org/abs/2001.03048

Rucci, M., & Victor, J. D. (2015): The unsteady eye: an information-processing stage, not a bug. *Trends in Neurosciences*, *38*(4), 195–206. https://doi.org/10.1016/j.tins.2015.01.005

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., & Fei-Fei, L. (2015): ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, *115*(3), 211–252. https://doi.org/10.1007/s11263-015-0816-y

Santini, F., Redner, G., Iovin, R., & Rucci, M. (2007): EyeRIS: A general-purpose system for eye-movement-contingent display control. *Behavior Research Methods*, *39*(3), 350–364. https://doi.org/10.3758/BF03193003

Santurkar, S., Tsipras, D., Ilyas, A., & Ma, A. (2018): How Does Batch Normalization Help Optimization?, 11.

Shannon, C. (1949): Communication in the Presence of Noise. *Proceedings of the IRE*, *37*(1), 10–21. https://doi.org/10.1109/JRPROC.1949.232969

Shorten, C., & Khoshgoftaar, T. M. (2019): A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, *6*(1), 60. https://doi.org/10.1186/s40537-019-0197-0

Singer, J., Seeliger, K., & Hebart, M. N. (2020): The representation of object drawings and sketches in deep convolutional neural networks, 7.

Steinman, R. M., Haddad, G. M., Skavenski, A. A., & Wyman, D. (1973): Miniature Eye Movement: The pattern of saccades made by man during maintained fixation may be a refined but useless motor habit. *Science*, *181*(4102), 810–819. https://doi.org/10.1126/science.181.4102.810

Wan, L., Zeiler, M., Zhang, S., LeCun, Y., & Fergus, R. (2013): Regularization of Neural Networks using DropConnect, 9.

Wang, H., Ge, S., Xing, E. P., & Lipton, Z. C. (2019): Learning Robust Global Representations by Penalizing Local Predictive Power [arXiv: 1905.13549]. *arXiv:1905.13549 [cs]*. Retrieved June 14, 2021, from http://arxiv.org/abs/1905.13549

Wong, S. C., Gatt, A., Stamatescu, V., & McDonnell, M. D. (2016): Understanding Data Augmentation for Classification: When to Warp? *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 1–6. https://doi.org/10.1109/DICTA.2016.7797091

Yarbus, A. L. (1967): *Eye movements and vision.* [OCLC: 925109437]. Plenum.