Stian Rikstad Hanssen

OptiNet: AMD Detection Network

Master's thesis in Computer Science
Supervisor: Professor Keith L. Downing (NTNU) and Professor Masashi Sugiyama (UTokyo and Riken)
September 2019

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Stian Rikstad Hanssen

OptiNet: AMD Detection Network

Master's thesis in Computer Science
Supervisor: Professor Keith L. Downing (NTNU) and Professor Masashi Sugiyama (UTokyo and Riken)
September 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

The field of medicine is exploring increasingly more automated techniques to aid professionals in their work. In recent years, a surge of artificial neural networks has found success in a wide range of fields, performing comparable or even better than humans. Because of this, neural networks are currently being tested in an increasing number of domains. Diagnosis of eye conditions such as *age-related macular degeneration* (AMD) is one such example. A highly rampant disease known for being the biggest cause of irreversible visual impairment in high-income countries. AMD is commonly diagnosed by analysis of medical images such as *optical coherence tomography* (OCT). This visual analysis is a task often suitable for *convolutional neural networks* (CNNs). Therefore, many experiments using CNNs have been conducted and they show promising results.

I propose a CNN architecture for OCT scans that perform at a state-of-the-art level when detecting AMD. The architecture is tested using two kinds of convolution: Regular 3D convolution and a decomposition of 3D convolutions known as spatiotemporal convolution. Unlike many approaches in the literature, the model requires no pre-training, while achieving an AUC of $\sim 99.7\%$ on the dataset from Duke University, and an AUC of $\sim 99.9\%$ on a new dataset from St. Olav's University Hospital. Furthermore, the results are examined using CNN Fixations, a visualization technique that can reveal what the model focused on in the OCT scans.

Sammendrag

Det medisinske fagfeltet utforsker flere og flere automatiske teknikker for å hjelpe fagpersoner i sitt arbeid. I de siste årene, har kunstige nevrale nettverk funnet suksess i et stort antall fagfelt med ytelse på likt nivå eller bedre enn mennesker. På grunn av dette, blir nevrale nettverk testet i et økende antall domener. Diagnose av øyesykdommer, som for eksempel *aldersrelatert makuladegenerasjon* (AMD), er et av disse domenene. En svært utbredt sykdom kjent som den største årsaken til irreversibel synshemming i høyinntektsland. AMD er vanligvis diagnostisert gjennom analyse av medisinske bilder slik som *optisk koherens tomografi* (OCT). Denne visuelle analysen er en oppgave som ofte er passende for *konvolusjonsbaserte nevrale nettverk* (CNN). Mange eksperimenter som bruker CNNs har blitt gjennomførst på grunn av dette, og de viser lovende resultater.

Jeg foreslår en CNN arkitektur for OCT bilder som yter på det høyeste nivå i verden på gjenkjenning av AMD. Arkitekturen er testet med to forskjellige konvolusjonstyper: Vanlig 3D konvolusjon og en dekomposisjon av 3D konvolusjon kjent som "spatiotemporal" konvolusjon. Ulikt mange fremgangsmåter i litteraturen, min modell trenger ingen pre-trening, samtidig som den oppnår AUC på $\sim 99.7\%$ på et datasett fra Duke Universitet, og en AUC på $\sim 99.9\%$ på et nytt dataset fra St. Olavs Universitetssykehus. I tillegg blir resultatene undersøkt med bruk av CNN Fixations, en visualiseringteknikk som kan avsløre hva modellen fokuserer på i OCT bildene.

Preface

This is a practical project in the field of computer science and medicine. It proposes a new architecture for detection of the illness AMD. The project is conducted primarily at The University of Tokyo (UTokyo) and in collaboration with St. Olav's University Hospital. Preliminary research was conducted at the Norwegian University of Science and Technology (NTNU) and their computer cluster IDUN was used throughout the research period. The supervisor from NTNU was Professor Keith L. Downing and Professor Masashi Sugiyama was the supervisor at UTokyo. Furthermore, Dr. Tora Sund Morken and Dr. Arnt-Ole Tvenning from St. Olav's University Hospital provided the problem setting, data, and advice related to the medical aspects of the project. Additional credit goes to fellow student Håkon Hukkelås for providing suggestions on related works such as Tran et al. [2017] that turned out to be highly relevant to the project. A final thank you goes to all the authors that gave permission to use their figures in this project.

Stian Rikstad Hanssen

Tokyo, September 19, 2019

Contents

List of Figures

List of Tables

Glossary

AMD Age-related Macular Degeneration (AMD) is a medical condition that can affect vision ranging from blurred to no vision in the center of the visual field.

ANN Artificial Neural Network (ANN) a machine learning method applying some of the concept inspired by the human brain.

AUC Area Under the Curve (AUC) is a metric that measures the area under the receiver operating characteristic (ROC) curve.

CNN Convolutional Neural Network (CNN) is a type of artificial neural network designed for handling image data and other data with similar properties.

Drusen Yellow deposits under the retina consisting of fatty proteins.

Feature Map The output of a convolution.

GPU Graphics Processing Unit (GPU) is a piece of computer hardware designed to handle computation for graphics and data similar to graphics.

IC Inclusion criteria (IC) is a criteria which a paper must fulfil in order to be further evaluated in an SLR.

ILSVRC (ImageNet) ImageNet Large Scale Visual Recognition Competition (ILSVRC) is a competition about creating the best algorithms for handling various image related tasks.

Kernel A moving grid of values that are applied to an image to perform a convolution.

OCT Optical Coherence Tomography (OCT) is a type of non-invasive cross-sectional imaging in biological systems.

Padding Values added around an image to counter the size reduction of a convolution.

PED Pigment Epithelial Detachment (PED) is the condition in which a cell layer just outside the neurosensory retina detach.

Python A programming language.

PyTorch A machine learning framework for Python.

QC Quality criteria (QC) is a criteria to evaluate the quality of a paper in an SLR.

ROC Receiver Operating Characteristic (AUC) is a curve showing the relation between true positive rate and false positive rate for some experiment attempting to make predictions.

RPE Retinal Pigment Epithelium (RPE) is a highly reflective band right under the retina that is often evaluated for signs of age-related macular degeneration (AMD).

SLR Structured Literature Review (SLR) is a method to systematically find, evaluate and filter literature in the field relevant to a set of research questions.

Stride The distance a kernel is moved for each step in a convolution.

TensorBoard A visualization tool developed by Google for monitoring machine learning methods.

VGG Visual Geometry Group (VGG) is a well known CNN architecture that had the best performance on certain tasks in the ImageNet competition 2014.

Chapter 1

Introduction

This is a computer science master thesis about detection of *age-related macular degeneration* (AMD). In the project, the words: Detection, classification, and diagnosis will be used interchangeably. They entail a system being able to tell whether an image shows a healthy eye or an eye with a variation of the disease AMD. The project will focus on using artificial neural networks to detect AMD. In the literature as well as in this project, a neural network called *convolutional neural network* (CNN) will be used. The goals of the project and all research questions will be laid out in this chapter.

1.1 Motivation

AMD is the leading cause of irreversible visual impairment in high-income countries. It globally accounts for 6% of blindness [Flaxman et al., 2017; Jonas et al., 2014]. The number of people affected by AMD rise with age [Akuffo et al., 2015]. As the number of elderly rises [Carbonaro et al., 2018], automation will be critical to handle the increasing number of people in need. This is especially important in the case of AMD, which affect such a large number of elderly people.

AMD is a severe and prevalent disease that can result in blindness. Automated tools such as trained neural networks can aid professionals in predicting a prognosis, plan the need for follow-up, and future treatment. An objective system can provide reassurance and aid patients in self-management of the disease, as well as provide information to the patient in regards to future prognosis.

Diagnosis and follow-up of AMD is a field of medicine where examination of images such as *optical coherence tomography* (OCT) is essential. These are 3D scans of the macula and fovea, a specific region on the retina which is located in the eye. The fovea is crucial for visual resolution. Evaluation is done by doctors through qualitative assessments such as presence or absence of fluid in the retina. However, recent machine learning techniques have started to show potential in the field. Lee et al. [2017] showed how a deep CNN based on the VGG model can classify a large number of OCT scans with high precision. Apostolopoulos et al. proposed their own CNN architecture with pre-training that scored well on the biggest public dataset of AMD patients.

1.2 Goals and Research Questions

In order to have a clear direction in the project, it is important to set goals, research questions and definitions to tell when a goal is achieved. In this section, these guidelines will be laid out.

Goal 1 Create a neural network capable of learning symptoms of AMD.

The most essential part of this project is the neural network. This is the fundamental system that will try to learn the symptoms of AMD in OCT scans and classify the disease. To assert whether the network has achieved this goal, an evaluation measure must be used such as *accuracy*. To verify that the model has learned to detect symptoms of AMD, it is expected that the model is able to perform better than an agent which chooses randomly. If 60 % of cases are labeled AMD, then the network should obtain accuracies higher than this percentage.

Goal 2 Achieve positive results by finding a balance between learning time, GPU capacity and size of the data.

Diagnosis of AMD is an image classification task. Many of the approaches today utilize large convolutional neural networks that require high GPU capacity and a lot of training time. For this project the GPU resources are limited. Therefore, it will be important to find a fitting architecture that can achieve results with these constraints in mind. Positive results should be predictions better than chance and preferably at the same level as works in the literature. Accuracy and *area under the receiver operating characteristic curve* (AUC) will be used to compare relevant works in the domain.

Research question 1 Will a neural network using the depth information have any advantage over approaches that only evaluate per slice information of

OCT scans?

OCT scans are like a stack of images going depth-wise through the eye. Many approaches in the literature choose a method that only evaluate each of these images individually. By using 3D convolution or some variation of it, one can find patterns in all dimensions of the scans. To find whether there is any advantage to considering the depth-wise relations, results between the thesis model and related works that disregard depth relations will be compared using the AUC metric.

Research question 2 Is there any advantage to using (2+1)D convolution over regular 3D convolution in this domain?

(2+1)D convolution is an alternative to 3D convolution that might aid in achieving research goal 2. Tran et al. had great success with (2+1)D convolution in classifying actions in video. Certain properties of this kind of convolution might also give advantages in classification of AMD. To answer this research question similar networks using both kinds of convolution will be compared on performance on several metrics: AUC, Recall, Precision and accuracy. AUC will be the metric of highest importance.

1.3 Research Method

The research goals in this project is all about creating a neural network architecture that can handle the criteria specified in section 1.2. The nature of this research is therefore experimental in that a model will be designed and then tested through experiments. The design procedure moves forward through ideas of related works, reflections on concepts in the field and incremental improvement of models through testing. I choose this type of methodology because it is the most common in the field. The focus is not on finding a new theoretical concept or architecture, but rather the performance in a particular domain on specific datasets. This might inherently lead to new models on the way and at the very least provide more insight into the field of deep learning on AMD.

1.4 Thesis Structure

The thesis is structured to gradually build up the needed theoretic background throughout the chapters up until the chapter 4. Whenever a new term comes up it will be formatted in *italic* and all acronyms will be shown in parenthesis the first time the acronymed term occurs. The current chapter, chapter 1, covers

why the research is performed, what the aim for the research is as well as how the thesis is structured. Chapter 2 covers the basic foundation of the thesis such as information about AMD, the dataset and information about the class of neural networks relevant to this research. Chapter 3 covers more advanced and specific theory related to the research goals. Each section in chapter 3 covers a related work in the field that has similar research goals, show popular concepts in the field or has concepts that could be useful in detecting AMD. The architecture developed in this thesis is explained in chapter 4 followed by experimental results in chapter 5. An evaluation and conclusion of the architecture and results is found in chapter 6. In chapter 6, I discuss the research questions and reflect on design decisions, performance compared to related works and potential future work.

Chapter 2

Background Theory

This chapter will start by covering OCT scans, the data used to detect the illness AMD. The objective in this thesis is to distinguish scans of AMD from scans of healthy cases. AMD will be explained in more detail in the subsequent section. A final part of the chapter will explain theoretical aspects of CNNs, the type of model the network will be based on. The chapter ends with a brief explanation for why CNNs are useful in the field of medicine.

2.1 Optical Coherence Tomography

Optical Coherence Tomography (OCT) is a type of non-invasive cross-sectional imaging in biological systems. OCT use a technique analogous to ultrasonic pulse-echo imaging. The technique produces two-dimensional images of optical scattering from internal tissue [Huang et al., 1991]. These scans can be done multiple times to produce a volumetric image of the tissue. This type of imaging has become a useful tool in the examination of the eye. It is used in finding signs of various conditions, one of them being AMD.

The OCT scans can vary greatly depending on the specific scanner and the operator. The operator may choose resolution, depth and to some degree how the patient is positioned. The patient can affect the scan by blinking and shifting their position. Also, the scanner can vary in degree of noise and resolution. These factors makes it challenging for *artificial neural networks* (ANNs) to generalize to unseen situations. Training on a dataset only using OCT scans from one scanner can cause problems for a network attempting to handle scans from multiple

sources.

The datasets handled in this project, as well as those in the related work, all have one channel per OCT "slice" making up the volume. That is, the images are gray-scale or more precisely a single channel as opposed to common RGB (colored) images that have 3 channels.

An OCT scan can be so large that some graphics processing units (GPUs) do not have the capacity to load a full-sized OCT scan into VRAM (memory) for training. In many cases, the OCT scan is therefore cropped and downsampled to a size more acceptable for the GPU. An additional technique, is to store the OCT scan in a suitable data type and structure. In code, this would be an array with little overhead that is stored in values of an acceptable size. For instance, 32-bit floats hold enough precision and take half the space of 64-bit floats.

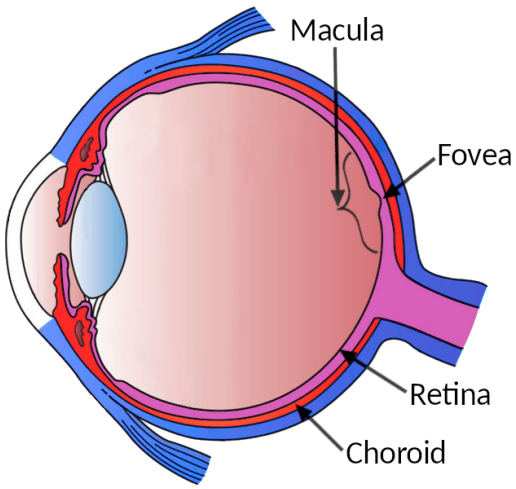


Figure 2.1: A diagram of the human eye seen as a cross section from the side. Light enters the eye from the left going through the eye and finally hitting the retina. The retina covers the backside of the eye, not just where the arrow is pointing. A certain part of the retina is known as the macula indicated by the curly bracket, while the fovea is the dip seen in the macula. The choroid is displayed as the red layer in the diagram. Diagram from Fischer [2013] with altered colors and labels.

2.1.1 Data Augmentation

A challenge with medical data is the accessibility. Due to strict regulations, it is often required for the data to be anonymized and in many cases, each patient must be asked for permission to use their data. Therefore, it takes a considerable amount of time to create a dataset, and the dataset may be significantly smaller than common image datasets used in machine learning. A way to combat a small dataset is *data augmentation*. Small operations on the scans to make them different enough to count as a new case, while accurate enough to not mislead the neural network. For medical data, it is particularly important to be delicate in this process as certain symptoms can be dependent on properties in the scan. As this project is about diagnosis of AMD, the augmentation techniques covered will be specifically tailored to this task.

Horizontal Flipping

OCT scans are taken of both eyes, so the network is forced to learn both orientations either way. The dataset can be doubled by flipping all scans making the left eye the "right eye" and vice versa. As the OCT scans are 3D volumes, it should be clarified that the horizontal flip is done for each individual image in the volume and that all images in one volume are flipped the same way.

Cropping

While cropping can be used for data augmentation by randomly cropping the scans, that is not the typical use on OCT scans showing AMD. OCT scans are quite large and in many cases the area of interest is only in the center of the OCT scan. Therefore, in order to reduce the size of the OCT scan, one can perform a center crop. The center crop removes parts of the scan around the edges in all directions leaving only a cube of data in the middle. Alternatively, it can be viewed as cutting out a rectangle from the center of each slice in the OCT scan and finally cut away some of the first and last slices.

Rotation

As noted earlier each OCT scan does not necessarily have the exact same orientation. When comparing OCT scans one can find a difference in rotation between them. As this is already a varying factor, one could do more rotations in order to produce more data. The rotation would normally be done for each slice, where

the rotation remains the same throughout an OCT scan. It is however important that this rotation is reasonable as training the network to identify upside down OCT scans would be useless. Furthermore, rotating an image would leave empty spaces around the edges. The network could start to use the empty spaces as features in their classification. It can be avoided by cropping in on the image to the point where the empty spaces disappear. Therefore, this technique is preferred when the scans are already going to be cropped.

2.2 Age-Related Macular Degeneration

Age-related macular degeneration (AMD) is a medical condition that can affect vision ranging from blurred to no vision in the center of the visual field. Macula being a part of the retina, the innermost light-sensitive layer of the eye seen in figure 2.1. The number of people affected by AMD increases by age. According to Akuffo et al. [2015], 20% are affected above the age of 50 and approximately 47% by age 86 and higher.

2.2.1 Variations

In approximately 10% of patients, neovascularization from the choroid into the retinal tissue occur [Akuffo et al., 2015; Jonasson et al., 2011]. This is an abnormal blood vessel growth starting at a layer in the eye known as the choroid marked with red in figure 2.1. This abnormal growth causes fluid accumulation in the macula with rapid visual loss. This type of AMD is known as *neovascular AMD* or *wet AMD*.

The majority of patients have dry AMD. This type of AMD is not related to any leakage of liquids, however loss of vision may still occur. The retina deteriorates due to the formation of small yellow deposits, known as drusen, under the macula. Furthermore, drusen leads to drying and thinning of the macula to the point where it loses its function [Parmet et al., 2006]. In one subtype of AMD known as *pigment epithelial detachment* (PED), a cell layer just outside the neurosensory retina, known as the pigment epithelial layer, detach. For these patients, the visual acuity is gradually lost as the illness results in geographical atrophy (a more severe advancement of AMD), scarring or rupture of the PED [Pauleikhoff et al., 2002]. Patients with PED also have a much higher risk of developing wet AMD. Cukras et al. found that one in three patients with PED developed wet AMD during a two-year observational period. There are still many unknown factors

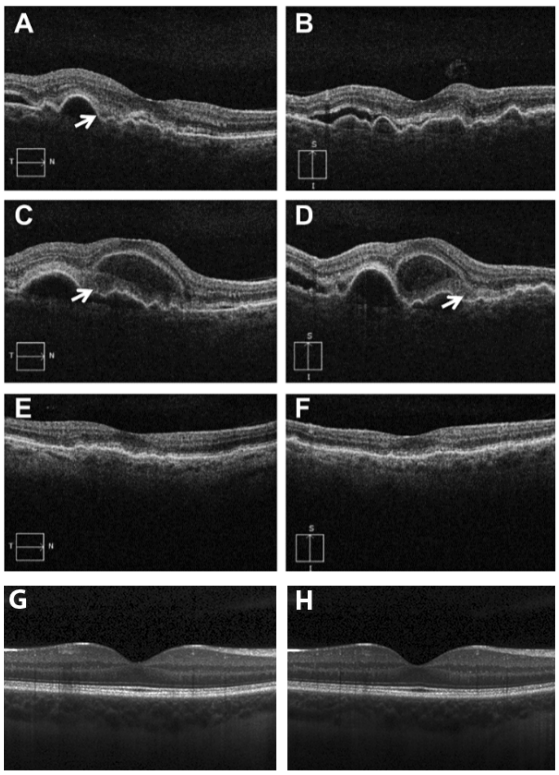


Figure 2.2: Cross sections of Optical Coherence Tomography (OCT) focused in on the fovea, a particular spot in the retina. A-D display choroidal neovascularization and pigment epithelial detachment. Images E and F shows dry AMD. Images A-F are images from Rosenfeld [2016]. G and H show two examples of healthy macula from Srinivasan et al. [2014].

related to identifying patients at risk for developing a more severe prognosis from AND and PED.

2.2.2 Symptoms in OCT Scans

Symptoms vary across types of AMD as well as the degree within one type. Figure 2.2 shows various examples of AMD as well as healthy eyes. These images are focused on the macula. One can see the location of the macula and other components in the eye from figure 2.1. The retina is composed of layers that reflect light to varying degrees based on the composition of the tissue. *Retinal pigment epithelium* (RPE) is the bottom-most highly reflective band, which is best seen in G and H in figure 2.2. Reduction in thickness and increased reflectively (brightness) of the RPE is often one of the signs of AMD. Additionally,

the RPE may show signs of detachment by the irregular shape, as seen in images A-D. Some irregularities in the RPE can also be seen in E and F; however, the steepness of the curves are much smaller. In these images, the fovea is also swollen to the point of not being visible. The fovea is the dip only seen in G and H. A-D are examples of wet AMD, where the macula has swollen substantially. Highly reflective particles under the retina can be a sign of drusen. Having drusen increases the patient's risk of having AMD [Crabb et al., 2002]. These particles may displace the RPE and be a cause for its irregular shape. Drusen can increase in size and number as the condition progresses, or disappear over time which is often associated with atrophy of the RPE and overlying retina.

For each variation and development, there are further specific symptoms that will not be covered here, however, further information can be found in Adhi and Duker [2013]; De Carlo et al. [2015]; Chiu et al. [2012].

2.3 Convolution

Convolution is the mathematical operation on two functions to produce a third function that expresses how the shape of one is modified by the other. In the case for images, an image can be seen as a discrete two-dimensional function. Thus, convolution on images is a discrete higher dimensional convolution.

2.3.1 Applying Convolution to Images

Figure 2.3 shows the overall concept of convolution on images. The input is an image that has values for each pixel. This is the first function mentioned previously. A *kernel*, sometimes called *filter*, is the second function we apply to the image, which also contains values on discrete positions.

The kernel can be viewed as a sliding window. For each position the window is placed over, each value in the kernel is multiplied to the corresponding values in the image as seen in figure 2.3. The products are added together to one final output value which is placed onto the output image. In figure 2.3, the output value is 6. This can be expressed mathematically as

$$Y(j,k) = (X * f)(j,k) = \sum_{\gamma=-K_h}^{K_h} \sum_{\delta=-K_f}^{K_w} X(j+\gamma, k+\delta) f(\gamma, \delta) \qquad (2.1)$$

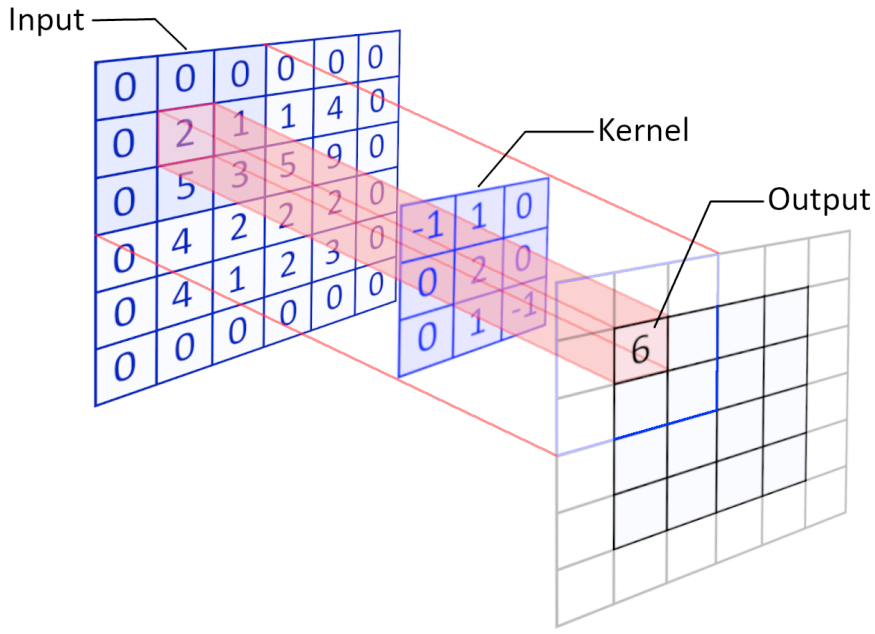


Figure 2.3: Illustration of two-dimensional convolution.

where $Y(j,k)$ is the value on the output image at position (j,k). $X(y,x)$ is the value of the input image at position (y,x) and $f(h,i)$ is the value of the kernel at position (h,i). Finally, K_h and K_w define the size of the kernel, $(2 \cdot K_h + 1)$ is the height and $(2 \cdot K_w + 1)$ is the width. In equation 2.1 it is assumed that all kernels are of an odd number size, but this doesn't have to be the case. However, for the purpose of this project, this equation is sufficient.

Two final parameters s_x and s_y is defined as the *stride*. The stride is how far the kernel is moved each time the kernel is applied, s_x and s_y for horizontal and vertical movement respectively. In many cases $s_x = s_y$, however, this does not have to be the case. To do a full convolution the following steps would be executed:

- Start in the top left corner of X and apply equation 2.1.
- Move kernel horizontally on X by a distance s_x at a time and apply equation 2.1 for each step.
- Once the end of the row is reached, move the kernel vertically down by a distance s_y and start over again at the beginning of that row.
- Continue the process until the kernel has reached the bottom right corner.

Output image Y should now have all values filled in and the convolution is com-

plete.

2.3.2 Padding

Looking at figure 2.3 you may notice that the output value is not placed all the way in the upper right corner of the output image. In fact, once a convolution is complete the size of the image will always be reduced by $2 \cdot K_h$ in height and $2 \cdot K_w$ in width. Sometimes this is not desired, so *padding* is used to adjust the output dimensions of the image.

Padding is the concept of adding additional values around the edge of the input image to counter the reduction in size from the convolution. The most typical type of padding is *zero padding*, which means to simply put values of zero around the edge of the image. If one wish to counter the reduction from convolution with stride $s_x = s_y = 1$, the thickness of the padding would be K_h and K_w.

It should be mentioned that regular convolution is not the only way an image may be reduced in size. With a stride greater than 1, the output image would only place a value every *stride* position of the input image. Therefore, the thickness of the padding may vary depending on the use.

2.4 Convolutional Neural Networks

It is assumed the reader has a basic understanding of neural networks going into this section as the focus will be on the specifics of *convolutional neural networks* (CNNs) [LeCun et al., 1999]. These networks have come to be extremely useful in the domain of big data that has spatial relations such as images.

Using a normal fully connected network would result in an impractical amount of trainable parameters. Just the input layer would require as many neurons as there are values in the image. If the image had dimensions 512×512 and three (RGB) channels there would be 786 432 neurons just in the first layer! If the second layer then had 256 neurons there would be 201 326 592 weights between these two layers. This is not the only problem: Fully connected networks learn features looking at an image as a whole. If it has been trained to detect a circle in the left bottom corner, it might fail to recognize a circle in the top right. The features it learns lack *spatial invariance*. These problems gave rise to the convolutional neural network explained in this section.

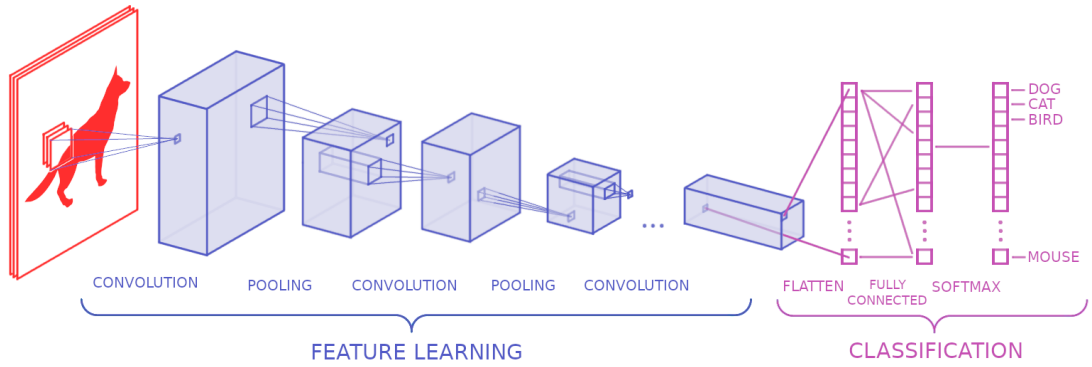


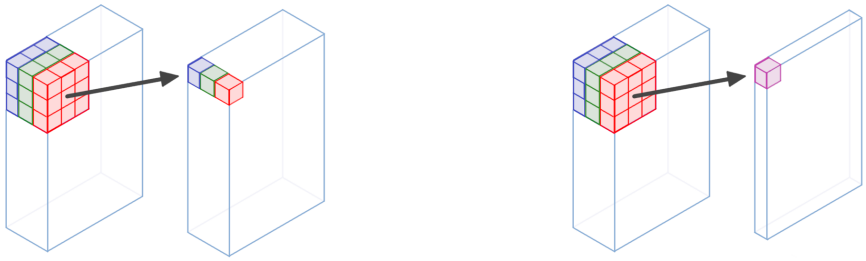
Figure 2.4: Example of a convolutional network architecture for classification of animals. Softmax is not a typical layer, it is an activation function often used on the last layer of the network for multi-class classification. It converts the output of the network to a probability distribution.

2.4.1 Dimensionality of Images

Section 2.3 explained the concept of convolution and how it is applied to images. However, a few important details were left out. A two-dimensional (2D) image with color is not really 2D for a neural network. Color images have channels, usually one for red, one for green and one for blue. Therefore, the real dimensions of such an image are $channels \times height \times width$, three dimensions. The kernel applied will therefore be 3D. Even though the image is 3D, the convolution applied is called 2D convolution. This is because when depth is considered as channels, the depth of the kernel is always equal to the depth of the image. In terms of movement, the kernel only has room to move horizontally and vertically (two dimensions). Additionally, the output image will be 2D because the kernel output one value at the center of the kernel. The difference between convolution with channels and convolution with spatial depth can be seen in figure 2.5.

2.4.2 Convolutional Layer

In CNNs, there are layers based on convolution. Instead of having layers of neurons, like in fully connected networks, one layer has multiple kernels and all the values in a kernel are trainable parameters. As seen in figure 2.6, a forward pass consists of applying convolution with each of the kernels in a layer, in addition to adding a bias for each kernel. Convolution with one kernel produces one channel in the output image. This is because the kernel covers all channels in the input image, but only output one value at a time resulting in one channel,



(a) Depth-wise convolution. (b) Regular convolution with channels.

Figure 2.5: Depth-wise convolution vs regular convolution with channels. Depth-wise convolution treats each depth-wise slice independently and therefore outputs a value for each depth-wise slice seen with each having their own color in the output. Regular convolution treats the depth as channels and thereby does the calculation with all channels at once. This will output one value for the $3 \times 3 \times 3$ kernel seen in the figure b.

as seen in figure 2.5b. The output channels are stacked depth-wise, resulting in a 2D image with a channel for each kernel; assuming the input data is a 2D image with channels.

The concepts explained here can be applied to any higher dimension. For example, medical images can be 3D color images. This means images with dimensions $channels \times depth \times height \times width$. In the same way as before the kernel is now 4D, with the channel dimension equal to the channel dimension of the input volume. The kernel will move in the three dimensions $depth \times height \times width$. Thus, the output image will be a 4D cube or a 3D image, with the number of channels equal to the number of kernels applied.

In further detail, how can this be perceived in comparison to a normal fully connected networks? The nodes are not in a line anymore, but rather arranged in more dimensions. As figure 2.6 show, the nodes make up the output "volume" of a layer. The nodes are still connected with weights to the previous layer, however, one node in the current layer is not connected to every node in the previous layer. Additionally, all nodes in one channel share the same set of weights, the set of weights from the kernel that made that output "slice" or channel.

This convolutional operation is differentiable, thus when doing backpropagation, one would need to calculate the derivative of the convolutional operation with respect to each of the trainable parameters. For a more detailed explanation, one can read LeCun et al. [1999]; Lecun [1988].

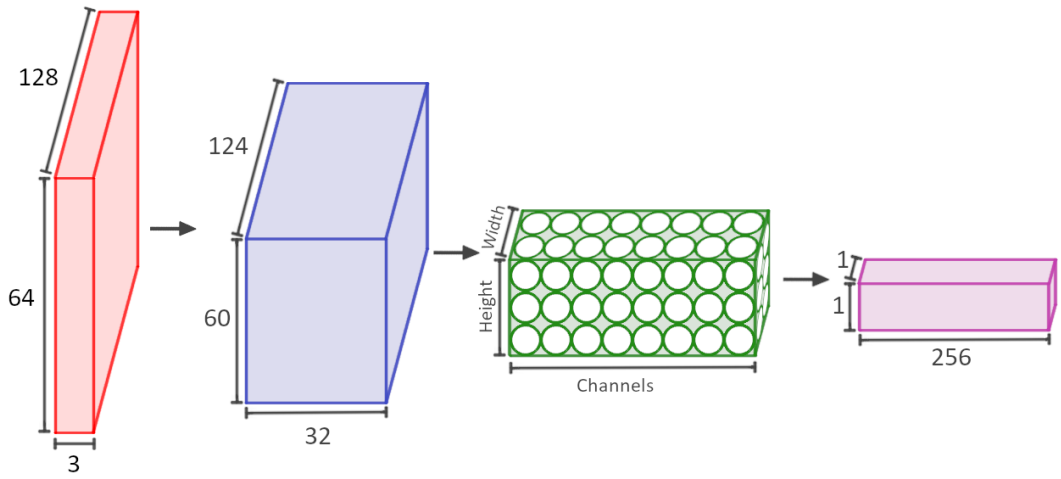


Figure 2.6: Red volume is the $3 \times 64 \times 128$ (*channels* $\times$ *height* $\times$ *width*) input image. 32 different kernels of size $3 \times 5 \times 5$ with stride of 1 are applied to the input image. The output from each kernel becomes a channel "slice" stacked depth-wise in the blue volume. The blue volume can be seen as the activations after the first layer. As no padding was applied, the height and width were reduced. The green volume shows how these shapes can be viewed as neurons stacked in a 3D formation. Each neuron in the green volume are only connected to a select group of the neurons in the blue volume. The kernels are not shown in this figure.

2.4.3 Downsampling

A convolutional neural network does not necessarily only consist of convolutional layers. Even though using kernels has reduced the number of parameters substantially, it is often useful to reduce the parameter count even further. Thus *downsampling* is used.

Downsampling can be done in many different ways. As mentioned in section 2.3, one can reduce the output image simply by making the stride bigger than 1. A natural downsample happens if padding is not utilized and also techniques such as *dilation* can reduce the output image. Dilation is a way to spread the kernel as seen in figure 2.7.

Another technique that has become very common is *pooling* proposed in Scherer et al. [2010] for CNNs. It should be mentioned that pooling has been a well-known concept long before 2010. Scherer et al. proposed the variation called *max pooling*. It works similarly to a convolutional layer in that it utilizes the sliding window method, and applies a function to a local selection of the input at a time. However, max pooling operates on a per channel basis. Figure 2.8

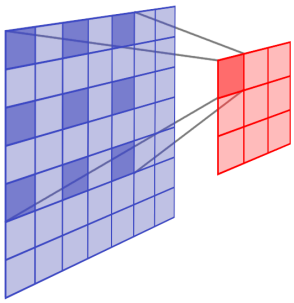


Figure 2.7: Illustration of dilation. Blue plane is the input image and red plane is the output image. Dark blue is the input values used to compute the dark red output value. Dilation is the concept of spacing the dark blue values apart.

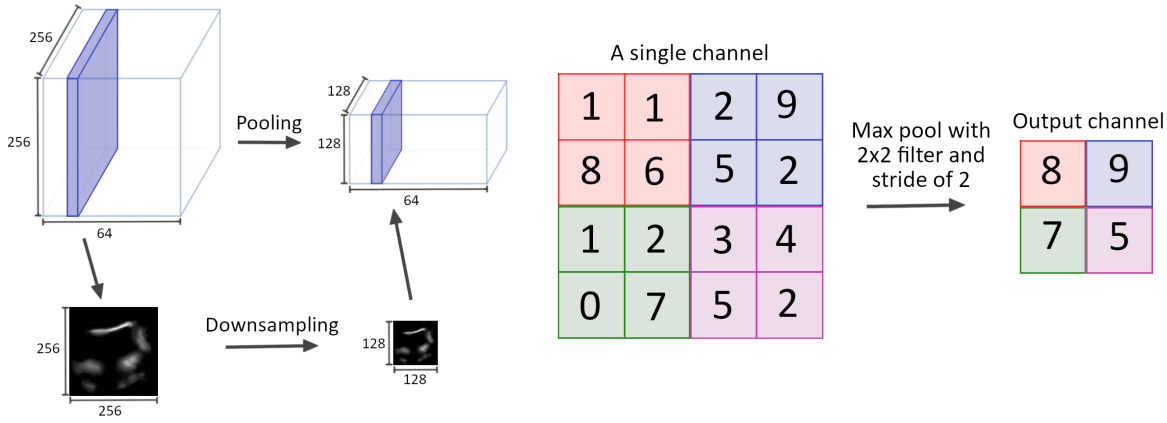
illustrates how the operation is done. Mathematically max pooling on 2D images for one kernel operation is expressed as:

$$M(j,k) = \max_{\gamma=0}^{K_h} \max_{\delta=0}^{K_w} X(j+\gamma, k+\delta) \tag{2.2}$$

In this case, $M(j,k)$ is the output max value of the window defined by positions (j,k) and $(j+K_h, k+K_w)$ on the input slice X. These two points define the diagonal of the kernel. It can be thought of as doing the exact same thing as a convolution except one takes the max instead of calculating a sum of products. This operation also uses strides in the same way as convolution. It is very common to have the strides be $s_x = K_w$ and $s_y = K_h$ (notation from section 2.3.1) as seen in figure 2.8b.

Backpropagation through a max pool layer is done by setting the gradient for all values that are not local maximums to 0. Then only compute the gradient for the max values M going from the layer after the max pool to the layer before the max pool. In a way, computing the gradient as if the max pool layer was not there on selected connections. If kernels overlap one would need to accumulate several error signals in one unit [Scherer et al., 2010].

There are several pooling techniques other than max pooling. In general, they work the same way, each using a different reduction function, for example average. The goal is to increase spatial invariance by reducing the representation to the most important features. If the input has been transformed in some way or been exposed to a lot of noise, this can confuse the CNN. Max pooling attempts to fix this by reducing the representation enough to keep the most important spatial relations while removing unnecessary information [Boureau et al., 2010].



(a) Overview of pooling. (b) Max pool for one channel.

Figure 2.8: Illustration of max pooling. The blue slice in figure (a) and the many colored grid in (b) both represent a single channel in a volume consisting of dimensions *channels* × *height* × *width*. The black image in figure (a) is an image representation of the channel. Figure (b) shows the operations happening on the "Downsampling" arrow in figure (a). Each color in figure (b) represents a max pool kernel position and its output position.

It should be noted, however, that there is no guarantee that max pooling will not remove data that could be useful for the model.

2.4.4 Batch Normalization

Normalizing the input of convolutional layer improves the total training time significantly [LeCun et al., 1998]. The normalization is usually done by setting the mean and variance of the input features to 0 and 1 respectively. Due to the adjusted distribution of the input, the network only needs to handle numbers in one range for all input neurons. *Batch normalization* takes this a step further and applies this concept also to the hidden layers of the network.

Among the reasons why batch normalization work is that it reduces *covariant shift*. One way to imagine what this means for a neural network goes as follows: You are making a network to say whether there is a dog in an image or not. The dataset consists of many images of brown dogs and many images of other random objects that are not dogs. The network needs to train the mapping $X \to Y$ where X is the input images and Y is the yes or no answer to whether there is a dog in the picture. If X is changed to also include dogs of other colors there will be a covariant shift. The boundary function the network has learned

has only seen one part of the problem, brown dogs. It lacks the information to know how the function should look for dogs of more colors. As the input has not been normalized, the new images of colored dogs are too far away from the original distribution and so the network is unable to generalize for the new dog images.

By normalizing and thus reducing the range of values of the input, the covariant shift is reduced. As batch normalization can be utilized between hidden layers, it also reduces the need for the current layer to adjust to the previous one. Each layer is made slightly more independent, since a hidden layer no longer needs to handle varying ranges coming from the previous layer. It can be viewed as an input layer, where the input features are the output of the previous layer. It would gain the same advantages of normalization as the input layer. This has shown to speed up learning even further [Ioffe and Szegedy, 2015].

Batch normalization is an algorithm made up of a few definitions where input values x over a mini-batch B is given as $B = \{x_{1...m}\}$. In these equations, layer index k in $x^{(k)}$ is omitted for clarity. All equations work on an individual activation. The mean of the mini-batch is defined by:

$$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{2.3}$$

The variance of the mini-batch is defined by:

$$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2 \tag{2.4}$$

Then one can normalize the mini-batch using the obtained mean and variance:

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} \tag{2.5}$$

Epsilon (ϵ) is a small contstant for computational stability. The normalized outputs $\widehat{x}_i$ are scaled and shifted by trainable parameters γ and β to produce the final output, y_i, of the batch normalization layer:

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv BN_{\gamma,\beta}(x_i) \tag{2.6}$$

The parameters γ and β are necessary to avoid reducing the freedom of the network. As $\widehat{x}_i$ will always of have mean of 0 and variance of 1, only outputting

$\widehat{x}_i$ restricts the network to always have a mean and variance of exactly 0 and 1, respectively. Therefore, two parameters are introduced to scale and shift the distribution to have any mean and variance desired. Each layer dictates their preferred mean and variance for their input. There is a set of γ and β for each batch normalization layer.

An important detail to notice about batch normalization, is that it does normalization and scaling over a *mini batch*, a selection of input-target pairs sent through the network as one to compute an average gradient. When calculating the mean and variance, it only takes the mini batch into account, not the whole dataset. This has a slight regularizing effect on the network. The normalization will be slightly noisy, due to the mean and variance being a little different between each mini-batch. Bigger mini-batches have mean and variance closer to the mean and variance of the dataset, thus smaller mini-batches have a greater regularizing effect.

When the network performs inference, as in testing without learning, it can be problematic to run batch normalization the same way as in training. The inferred results depend on the mini-batch size. During inference it is desirable for the output to only depend on the input in a deterministic fashion.

Algorithm 1: Training of network using batch normalization and setup for inference

Input: Network N with trainable parameter Θ; subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, N_{BN}^{inf}

$N_{BN}^{tr} \leftarrow N$ `// Training BN network`
for $k = 1...K$ **do**
  Add transformation $y^{(k)} = BN_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to N_{BN}^{tr} `// Equation 2.6`
  Modify each layer in N_{BN}^{tr} with input $x^{(k)}$ to take $y^{(k)}$ instead
Train N_{BN}^{tr} to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
$N_{BN}^{inf} \leftarrow N_{BN}^{tr}$ `// Inference BN network with frozen parameters`
for $k = 1...K$ **do**
  `/* For clarity:` $\mathbf{x} \equiv \mathbf{x^{(k)}}, \gamma \equiv \gamma^{(\mathbf{k})}, \mu_{\mathbf{B}} \equiv \mu_{\mathbf{B}}^{(\mathbf{k})}$, `etc. */`
  Process multiple training mini-batches B of size m, and average over them:

$$E[x] \leftarrow E_B[\mu_B]$$
$$Var[x] \leftarrow \frac{m}{m-1} E_B[\sigma_B^2]$$

  In N_{BN}^{inf}, replace the transform $y = BN_{\gamma,\beta}(x)$ with:

$$y = \frac{\gamma}{\sqrt{Var[x]+\epsilon}} \cdot x + (\beta - \frac{\gamma E[x]}{\sqrt{Var[x]+\epsilon}})$$

Ioffe and Szegedy suggest algorithm 1 for training and testing a network using

batch normalization. The first part of the algorithm, line 1 to 4, adds batch normalization to each layer, such that each layer has the input $y^{(k)}$ instead of $x^{(k)}$. As mentioned previously, (k) represents the index for a layer in the network and K is the final layer to implement batch normalization. N_{BN}^{tr} stands for network using batch normalization during training, while N_{BN}^{inf} is the network using batch normalization during inference. From line 5 starts the training of the network, which involves optimizing all the original weights and biases as well as all the new trainable parameters $\gamma^{(k)}$ and $\beta^{(k)}$.

From line 7 in algorithm 1, a setup for inference is applied to the batch normalized network. The inference mean in line 8 is defined as:

$$E_B[\mu_B] = \frac{1}{m}\sum_{j=1}^{J}\mu_B^{(j)} \tag{2.7}$$

One batch is indexed by j and there are J batches. The inference variance is defined as:

$$E_B[\sigma_B] = \left(\frac{m}{1-m}\right)\frac{1}{m}\sum_{j=1}^{J}\sigma_B^{2\,(j)} \tag{2.8}$$

During inference the means and variances are fixed. They are estimated using the previously calculated mean and variance of each training batch. Ioffe and Szegedy proposes moving averages of the mean and variance during training to obtain the population statistics $\mu_B^{(j)}$ and $\sigma_B^{2\,(j)}$, shown in equation 2.7 and 2.8. This way, the mean and variance is not affected by mini-batch size during inference.

For convolutional neural networks batch normalization works slightly differently. The normalization will rather be computed with the output of each kernel's convolution, also known as feature maps, across the mini-batch. If a feature map has dimensions $q \times p$ and B is defined as all values in a feature map across the mini-batch and spatial locations, then m in the previous equations has the new definition $m' = |B| = m \cdot pq$. This implies that the mean and variance of each feature map across the mini-batch becomes 0 and 1, respectively. Therefore, there will be a $\gamma^{(k)}$, $\beta^{(k)}$ pair for each feature map. Furthermore, running training on mini-batches of size 1 leads to $m = 1 \Rightarrow m' = pq$. In other words, batch normalization can be useful even for $m = 1$, as this turns batch normalization into instance normalization [Ulyanov et al., 2016] for convolutional layers.

2.4.5 Classification Layer

The convolutional layers of a CNN can be viewed as feature extractors. The network learns what parts of the image to focus on and makes its own representation of the data to pass on to the classifier. A traditional CNN will use one or more fully connected layers at the end to take in all the refined features and make the classification decision.

A transition from a convolutional layer to a fully connected layer would be needed. Looking back to figure 2.6, it was shown that the output volume of a convolutional layer is a set of neurons arranged in more dimensions. A fully connected layer has connections from every neuron in the previous layer to every neuron in the current layer. To make the transition, connect every neuron from the output volume to every neuron in the next flat layer. Alternatively, one can flatten the volume into one line of neurons and then make the connections as usual, as seen in figure 2.4.

2.4.6 Use of Artificial Neural Networks in Medicine

In recent years, ANNs such as CNNs have become popular when attempting to solve various image related problems. The recent surge of CNNs is likely attributed to the many advances in the field. In 2012, AlexNet beat all current methods in the ImageNet competition having a top-5 error 10.8 percentage points lower than the runner up [Krizhevsky et al., 2012]. After this, a new architecture of CNN has won the competition every year since. Through countless publications and competitions, CNNs has been shown to excel at image related tasks, especially classification. These models are becoming deeper thanks to concepts such as the ReLU, batch normalization, skip connections and more. More depth allows for more complex internal representations. Therefore, the advancement of GPUs has also been important in order to handle the increasing load of training larger and larger models.

In medicine, tasks such as diagnosis of AMD relies on classification from a health professional. The work may include manually segmenting parts of the image or attempting to find specific patterns in the image that indicate symptoms. These are tasks CNNs have been shown to handle quite well. Unlike many existing methods including traditional ANNs, CNNs have very few design decisions or assumptions made by humans. As CNNs are their own feature extractors, one avoids the need for most human assumptions, which in complex cases might be the hardest part in designing a model. The model will create its own feature extractors and classifier based on the dataset. This means it is important to make

the dataset representative of the domain it should handle. Large datasets with variation covering the whole domain is ideal. In the medial field, there is often large amounts of data, image evaluation can be quite complex and objectivity is desired. These are aspects suited for CNNs. However, it should be noted that though a large amount of data exist, it can be hard to gain permission to use it. Chapter 3 will show that even with lower amounts of data, some tasks can still be handled by CNNs.

2.5 Summary

This chapter has covered the foundation for this thesis. Optical Coherence Tomography (OCT) is a type of non-invasive cross-sectional imaging in biological systems. These scans can be viewed as a stack of images that together make up a 3D image. OCT scans are used by doctors to evaluate illnesses such as age-related macular degeneration (AMD) through image analysis.

AMD is a medical condition that can affect vision ranging from blurred to no vision in the center of the visual field. One of the most common symptoms is an irregular shape of the RPE, a white band seen in the OCT scans.

In order to detect patterns in images such as symptoms in OCT scans, it is common to use CNNs. Convolution can be viewed as a moving window over the image. The window is commonly called a kernel and can be described as a grid of values. A convolutional layer is a set of neurons stacked in more than one dimension, and each neuron may share weights in the form of a kernel. In other words, the values in the kernel are the weights for a convolutional layer. The downsampling technique known as max pooling runs a kernel over a channel only picking out the biggest value for each stride. This reduces the number of input values for the next layer.

Batch normalization is a method to reduce internal co-variate shift. In essence, the technique normalizes the input to a layer by using the batch as the population to find the mean and variance. The input distribution is set to have mean of 0 and variance of 1. Furthermore, the same distribution is then scaled and shifted by a set of trainable parameters so that each layer can pick their desired mean and variance for their input distribution.

CNNs have improved greatly over the last years. This has been shown through countless publications and competitions such as ImageNet. CNNs are their own feature extractors and are therefor prone to fewer human assumptions and errors, something that is ideal for medical use.

Chapter 3

Related Work

In this chapter, we will have a look at recent methods in the field. The first part of the chapter will cover the procedure used for handling the literature called a *structured literature review* (SLR). In the following section, RetiNet [Apostolopoulos et al., 2016] proposes a semi-supervised training approach to tackle problems with small datasets, and achieves state-of-the-art results. In section 3.3, Lee et al. [2017] found great results classifying AMD with 2D convolution with a VGG architecture. Tran et al. suggests a new way to do 3D convolution that outperformed other traditional methods in their domain. Finally, at section 3.5, a visualization technique by Mopuri et al. for CNNs is covered.

In all of these works, interesting aspects that can be of use to the project will be highlighted. This includes methods, architectural choices, performance studies, and tools.

3.1 Structured Literature Review Protocol

SLR [Kofod-Petersen, 2012] is a method to systematically find, evaluate and filter literature that is relevant to a set of research questions. This can in turn ensure a proper process has been conducted in the development of the research. The SLR was used to quality check the work already done and find any additional relevant literature.

The procedure is initialized by making a set of research question that should not be confused with the research questions of the master thesis. These research ques-

tion are tailored for the search of related works and for guiding the development of the SLR. The following research questions were used:

RQ1 What existing deep learning methods are used in classification of AMD in OCT scans?

RQ2 What unique contributions does the methods addressed in **RQ1** provide?

RQ3 What is the evidence in support of the methods found addressing **RQ1**?

3.1.1 Planning

Identification of the Need For a Review

For the master project it is a requirement to have a structured literature review. The primary goal of the SLR will be to understand the present landscape in the domain of detecting AMD from OCT scans.

Commissioning a Review

It can be assumed the review is commissioned for this project. No commission report was produced.

Specification of Research Question

It should be clarified that this is not the same as the research questions in section 1.2. These questions are specifically used when searching and evaluating the literature, as well as to guide the making of the SLR. The questions were developed incrementally. Initially, there were more questions, but it imposed too many constraints which lead to a very limited number of papers.

Development of a Review Protocol

The SLR was not in place at the start of the project. This is because there was no requirement to have the protocol in place during the preliminary studies. Thus, focus was placed on looking for variations of approaches to the problem and background theory. Parts of the related work is therefore not found through the protocol. The protocol is instead a quality check for the existing work as well as a protocol for later additions in the literature. All work found before the SLR was also found in the procedure with one exception. This exception was one

section in the related work from the preliminary study that was removed as it did not fit the requirements of the SLR.

Evaluation of the Review Protocol

The protocol was mostly evaluated by the author of the thesis and quality checked by Prof. Downing.

3.1.2 Conducting the Review

Identification of Research

A big focus in the review was finding papers in the domain of detecting AMD with deep learning methods. Furthermore, in order to compare approaches, the search also focused on papers using the biggest public dataset on AMD from Duke University [Farsiu et al., 2014]. Therefore, two search strings were used: One for relevant works conducted on the Duke Dataset, and one for classifying healthy versus AMD cases in more general terms.

The search domain was Google Scholar. This is a search engine used for searching scientific papers from multiple sources. Google Scholar also has features such as finding papers that have cited a specific paper.

In the preliminary studies, a similar approach was used, though outside of any formal protocol. In the early stages of the project, the objective was to understand what works when detecting AMD by exploring approaches in the domain. Google Scholar was used, but also suggestions from fellow students and professors.

To come up with search strings, two tables containing groups of terms were assembled. Each group contains words that are synonymous or have a similar meaning semantically. Table 3.1 lists terms for search of papers that can be compared to the thesis' results, while table 3.2 is for general domain search.

Group 1	Group 2	Group 3	Group 4	Group 5
AMD	OCT	Detection	Dataset	Duke
Age-related Macular Degeneration	Optical Coherence Tomography	Identification		
		Classification		

Table 3.1: Terms for search of papers that can be compared to the model in this thesis.

Group 1	Group 2	Group 3	Group 4	Group 5
AMD	Healthy	OCT	Detection	Deep Learning
Age-related Macular Degeneration	Normal	Optical Coherence Tomography	Identification	Machine Learning
			Classification	

Table 3.2: Terms for search of papers that operate in the domain of distinguishing between AMD and healthy cases in OCT scans using learning methods.

The following search strings were developed:

Search string 1, made from table 3.1:
("AMD" OR "Age-related Macular Degeneration") AND ("OCT" OR "Optical Coherence Tomography") AND ("Detection" OR "Identification" OR "Classification") AND "Dataset" AND "Duke"

Search string 2, made from table 3.2:
("AMD" OR "Age-related Macular Degeneration") AND ("Healthy" OR "Normal") AND ("OCT" OR "Optical Coherence Tomography") AND ("Detection" OR "Identification" OR "Classification") AND ("Deep Learning" OR "Machine Learning")

Search string 1 gave a total of 375 results. Initially, it was required to have cited the study related to the dataset, however, Google Scholar missed some papers in such a search. Instead the terms "duke" and "dataset" was added in the search string, which improved the results.

Search string 2 returned 4330 results, which was further reduced to 1960 by only showing results published in 2015 or later. After looking at 25 pages of results, it was concluded that no relevant results after page 10 could be found. Thus,

only the first 10 pages (100 papers) of results went on for further evaluation. In addition to search results, papers suggested by students and professors were added as well.

Inclusion Screening

A set of additional constraints were added to further decrease the results. The following *inclusion criteria* (IC) was used in the next selection:

IC 1 Study is about detection of AMD.

IC 2 The proposed model takes OCT scans as input.

IC 3 The method is machine learning related.

IC 4 The paper is in English.

IC 5 The study presents empirical results.

IC 6 The study is about classification between AMD and Healthy cases.

***IC 7** The method is tested on the dataset from Duke University [Farsiu et al., 2014].

IC 1 to 4 were utilized for primary inclusion screening, while IC 5 to 7 were used for secondary inclusion screening. IC 7 is only applied to papers with the same problem setting as the thesis, but the method is not related to supervised deep learning. The primary selection looked at the title as well as the abstract, while secondary screening looked through the whole text. Two papers were accepted despite not fitting the criteria. The paper from Tran et al. proposes an interesting architecture that I believed could be used in a the domain of this thesis. Mopuri et al. [2019] proposes a visualization technique that could supplement the results of the study. These papers were found through suggestions of students and professors. Both of these papers are not directly related to the research questions of the SLR, but could still go through a quality assessment. After both screenings, 7 papers remained. These papers were put through the final quality screening.

Quality Screening

The quality screening evaluated the quality of each paper. For each *quality criteria* (QC) the score 0, 0.5 or 1 is given based on how well the paper fit the criteria. Each paper will receive a total score based on the evaluation from each of the QC. A high score indicates good quality. The QCs are given below:

QC 1 Is there a clear aim in the research?

QC 2 Is the proposed method reproducible?

QC 3 Are design decisions explained and justified?

QC 4 Is the study put in context of other studies?

QC 5 If results are presented, are they compared to other studies?

QC 6 Are the test results thoroughly analyzed?

QC 7 Does the evidence support the conclusion made in the study?

The total QC score as well as extraction of selected data from the papers is found in table 3.3.

3.1.3 Result

Of the four top scoring papers, only two was deep learning based methods for distinguishing between AMD and healthy cases from OCT scan. The three bottommost papers are still relevant as they have the same task and dataset, which means their performance can be compared to the method of this thesis. The four top scoring papers were synthesised and added to the related works.

Authors	Title	Year Published	Method	Dataset	Findings / Conclusion	QC Score
S. Apostolopoulos, C. Ciller, S. De Zanet, S. Wolf and R. Sznitman	RetiNet: Automatic AMD identification in OCT volumetric data	2016	Convolutional Neural Network, Transfer Learning, Extreme Learning	Duke University Dataset [Farsiu et al., 2014]	State-of-the-art performance at the time. AUC: 99.7%	7/7
Lee, Cecilia S and Baughman, Doug M and Lee, Aaron Y	Deep learning is effective for classifying normal versus age-related macular degeneration OCT images	2017	Convolutional Neural Network	Private dataset collected in this particular study.	The deep learning technique achieves high accuracy and is effective as a new image classification technique. Macula level AUC: 93.83%.	6.5/7
Tran, Du and Wang, Heng and Torresani, Lorenzo and Ray, Jamie and LeCun, Yann and Paluri, Manohar	A closer look at spatiotemporal convolutions for action recognition	2018	Convolutional Neural Networks - ResNet Architecture	Sports-1M, Kinetics, UCF101 and HMDB51	Results comparable or superior to the state-of-the-art on Sports-1M, Kinetics, UCF101, and HMDB51 with their proposed R(2+1)D architecture.	7/7
Mopuri, Konda Reddy and Garg, Utsav and Babu, R Venkatesh	CNN fixations: An unraveling approach to visualize the discriminative image regions	2018	CNN Fixations	ILSVRC (ImageNet) and PASCAL	Demonstrated that the approach enables an interesting set of applications. In cases of erroneous predictions, the proposed approach offers further insight into why a CNN makes a particular decision.	7/7
Farsiu, Sina and Chiu, Stephanie J and O'Connell, Rachelle V and Folgar, Francisco A and Yuan, Eric and Izatt, Joseph A and Toth, Cynthia A and others	Quantitative Classification of Eyes with and without Intermediate Age-related Macular Degeneration Using Optical Coherence Tomography	2014	Semi-automatic method using 5 extracted features. Generalized linear model regression framework.	Duke University Dataset (published with this study) [Farsiu et al., 2014]	Identified and validated efficient biometrics to distinguish AMD from normal eyes byanalyzing the topographic distribution of normal and abnormal RPEDC thicknesses across a large atlas of eyes. AUC: 99.17%	5.5/7
Sun, Weiwei and Liu, Xiaoming and Yang, Zhou	Automated Detection of Age-related Macular Degeneration in OCT Images using Multiple Instance Learning	2017	Multiple Instance Learning	Duke University Dataset [Farsiu et al., 2014]	Method gets higher classification accuracy comparing with other methods shown in cited in their paper. Accuracy: 94.4%	5/7
Venhuizen, Freerk G and van Ginneken, Bram and Bloemen, Bart and van Grinsven, Mark JJP and Philipsen, Rick and Hoyng, Carel and Theelen, Thomas and Sanchez, Clara I	Automated age-related macular degeneration classification in OCT using unsupervised feature learning	2015	Unsupervised Feature Learning	Duke University Dataset [Farsiu et al., 2014]	Paper claimed to achieve excellent results in distinguishing AMD patients from healthy control subjects. AUC: 98.4%	4/7

Table 3.3: Selected papers from the SLR procedure.

3.2 RetiNet

The goal of RetiNet [Apostolopoulos et al., 2016] is to detect AMD in OCT volumes in order to aid in the time-consuming diagnosis process. It does so in a semi-supervised fashion using a two-step approach. The model first trains what they call the feature extractor of the model on 2D slices of the OCT scans, and then in a second step, train the classification part of the model using the already learned features on full OCT scans.

Before taking a closer look at the architecture, some of the concepts it uses will be explained.

3.2.1 Transfer Learning

The way RetiNet is able to train different aspects of the model in two steps is by using *transfer learning*. Transfer learning is the concept of training parameters in a model and then transfer the trained parameters over to a new model. The transferred parameters can then be frozen, which means these weights will not be adjusted during training. The amount of parameters that need training is reduced because the transferred parameters have already been adjusted. Alternatively, the transferred parameters will not be frozen, a very low learning rate is used instead to fine-tune the transferred parameters to the new model. Although this means training more parameters, it is still useful as the model can potentially converge faster.

3.2.2 Extreme Learning

Extreme Learning [Huang et al., 2006] is another technique which has the goal of reducing training time. The idea is to initialize the model and then immediately freeze a subset of the trainable parameters from being changed during training. RetiNet utilize this technique by performing training only on the convolutional layers in the model. This part of the model is called the *feature extractor*, though it is better described as a section of convolutional layers. As the remaining part of the model can not change, the feature extractor has to work around the frozen parameters, such that it can still perform the classification task well. It reduces training time, because the model avoids adjusting all the parameters in the last fully connected layers, known as the *classification layer*.

3.2.3 Architecture

In essence, RetiNet's architecture is two regular 2D convolutional models. The first model, RetiNet B, takes one slice of the 3D volume as input at a time. It is trained to classify the slice using the label of the volume it belongs to. RetiNet attempts to classify two classes with the labels: Control and AMD. The paper calls these labels *weak labels* as they are meant for the volume and not the slice. Not all slices in the OCT volume show AMD, thus these labels are "weak" in that they can give up to 50% of an OCT volume the wrong label.

Extreme learning is used by freezing the classification layer of RetiNet B. This is the shaded section at the top seen in figure 3.1. Only training the remaining part of RetiNet B, the feature extractor, is conducted.

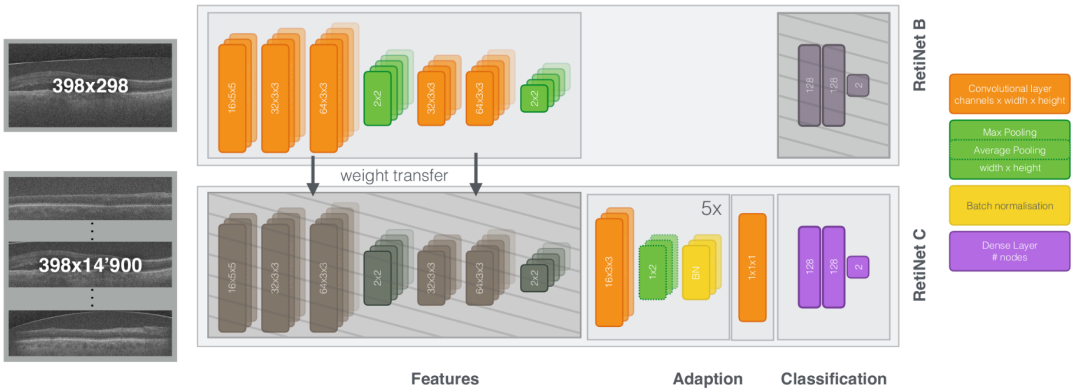


Figure 3.1: RetiNet Architecture from Apostolopoulos et al. [2016] modified with shaded areas. Sections of the model that has frozen parameters that will not be adjusted during training are shaded.

In the second stage, transfer learning is used to transfer the feature extractor over to the new model called RetiNet C. The feature extractor will be frozen and the remaining part is trained on a transformed version of the OCT volumes. Each slice in the volume is stacked such that a single 2D image remains with the same width as one slice, but with the height of all slices stacked on top of each other, as seen in figure 3.1.

As the feature extractor is frozen, an adaption layer is added to adjust for the differences between classifying slices and flattened OCT volumes. After the adaption layer comes the classification layer which is not frozen in RetiNet C. This will finally give the prediction to whether the volume shows signs of AMD or not. RetiNet C will also use the volume labels.

The architecture allows more training on a small data set by making each slice in the model an input for training. It promises fast training through extreme learning and claims to outperform recent models from the computer vision literature trained from scratch, at its time. It does, however, have some difficulties with edge cases where the differences between AMD and control are small.

3.3 2D Convolution by VGG16

Lee et al. [2017] propose a pure 2D convolution approach to diagnosis of AMD. In this study, they obtained 2.6 million OCT images linked to clinical data points from the electronic medical records. Out of these, a selection of 52 690 control (normal) and 48 312 AMD OCT images were selected. OCT images refer to each 2D slice of the 3D volume, where they selected 11 of the central 2D slices from each OCT volume. Lee et al. utilize parts of an already trained model, VGG16, to create a new model for classification of control and AMD cases.

The performance was measured by the *area under the receiver operating characteristic curve* (AUC), accuracy, and an occlusion test. The AUC was measured on three levels: Image level, macula level (OCT volume) and patient level. The measure for macula and patient level was done by averaging the probabilities for all images belonging to the same macula or patient. At the image level, they achieved an AUC of 92.78% with an accuracy of 87.63%. At the macula level, they achieved an AUC of 93.83% with an accuracy of 88.98%. At a patient level, they achieved an AUC of 97.45% with an accuracy of 93.45%. Peak sensitivity and specificity with optimal cutoffs were 92.64% and 93.69%, respectively. Lee et al. conclude that the findings have important implications in utilizing OCT in automated screening and the development of computer-aided diagnosis tools in the future. It shows that diagnosis on the image level obtains viable results.

3.3.1 ImageNet Models

Many pre-trained models are available today. Among the most popular ones are competitors in the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) [Russakovsky et al., 2015]. VGG [Simonyan and Zisserman, 2014] is one of these, an image classification CNN invented by *Visual Geometry Group* (VGG) from The University of Oxford. ImageNet requires the competitors to classify 1000 different classes from a subset of the whole ImageNet dataset consisting of 14 million hand-annotated images. VGG was the first runner-up in 2014 right behind

GoogLeNet, however, VGG performed better on the localization task. There are variations of the network depending on the depth. VGG16 is the VGG variation using 16 layers, only counting convolutional and fully connected layers.

As explained in subsection 3.2.1, transfer learning can be used to repurpose a neural network for a different task. A model, such as VGG16, has shown excellent performance on image classification tasks, which on an abstract level is the same domain as in Lee et al. [2017]. The idea is that earlier layers in a convolutional network detect more fundamental features. The first layers may detect simple shapes such as edges, then the next layers raise the complexity slightly, finding geometric shapes, and for each layer, more complex features are represented. The most fundamental features a network finds are similar for most convolutional networks, therefore one should be able to take the first layers from a well-trained model and reduce the parameters that require training.

3.3.2 Architecture

It was not clearly stated how the VGG16 was utilized as they only mention using a "modified version of the VGG16 convolutional neural network". For instance, they fail to tell how many layers from VGG16 were used, or whether these were frozen from training or trained further. They do mention using Xavier Initialization [Glorot and Bengio, 2010] implying one or more layers where trained from scratch. An illustration of the full architecture can be seen in figure 3.2.

The network uses 13 convolutional layers, and 3 fully connected layers. The classification of AMD in OCT images is quite different from classification in ImageNet. ImageNet requires classification of everyday objects such as animals, household items, etc., while OCT images are medical data where the objective is detecting symptoms of AMD. Therefore it is safe to assume the fully connected classification layers were trained from scratch and it is highly likely the deeper convolutional layers were replaced as well.

Similar to RetiNet, this approach is directed towards AMD diagnosis with limited resources. Transfer learning ensures fewer training parameters as well as a good starting point for basic feature detection. The depth of the VGG16 network provides the ability to represent complex features which may be needed in the domain of AMD diagnosis. A large dataset is ensured by training on image slices instead of volumes, which is favorable for large deep networks that require a large amount of data for learning. It should be noted that Lee et al. is unable to use the VGG16 model to evaluate volumes. This is because VGG16 only consist of 2D convolutions.

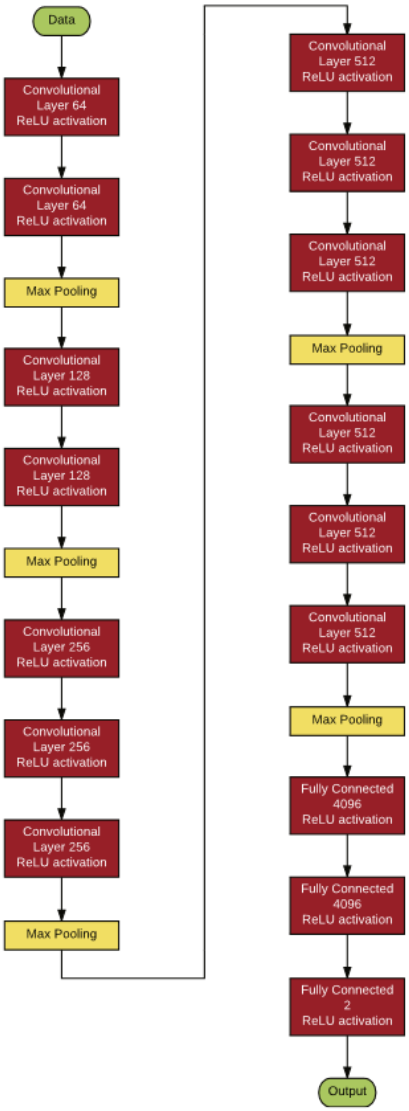


Figure 3.2: Illustration of the modified VGG16 model from Lee et al. [2017]

3.3.3 Occlusion Test

Figure 3.3 show some results from the occlusion test. These show the focus points from the network in the input image. Visualizations of how the network performs give great insight into what exactly the network has learned. The visualization shows a *heat map* layered on top of the image. A heat map has a scale in colors from low interest in black and dark blue to high interest in orange up to almost white. One can compare the heat maps to the evaluation of professionals and see which symptoms the network has learned. Lee et al. [2017] do exactly this in their discussion on the network's performance. Among other things, they discovered the network did not utilize certain symptoms in their evaluation. For instance, it seems the model lacked focus on drusen, sub-retinal fluid, pigment epithelial detachment, among other things. A second valuable aspect of occlusion testing is the aid it gives to the user of the system. If a professional were to use this network for diagnosis and come to disagree with its evaluation, the professional can better understand why the network made that diagnosis and confidently disregard or accept the assessment. The occlusion test is based upon the method proposed in Zeiler and Fergus [2014].

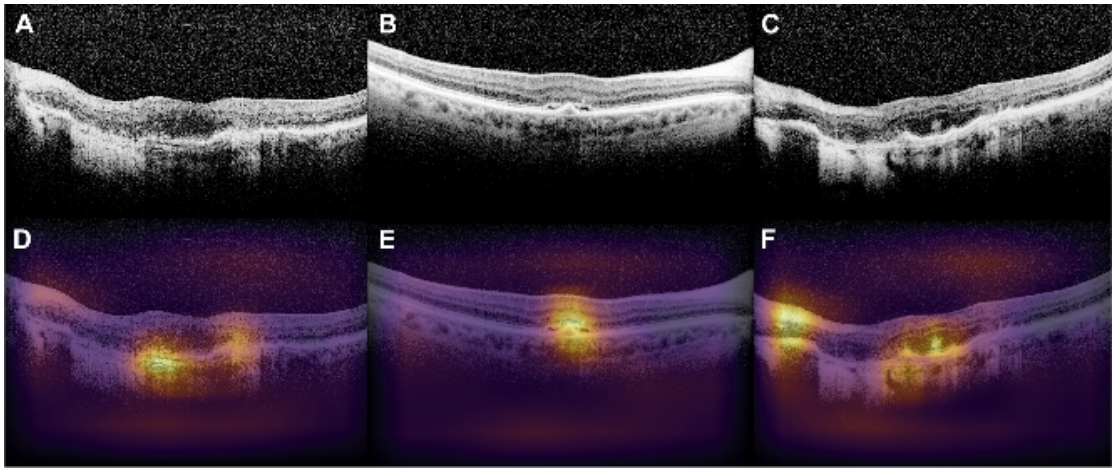


Figure 3.3: Occlusion results from Lee et al. [2017]. A, B, C are input images to the network, and D, E, F are the corresponding occlusion test resultsing.

3.4 Spatiotemporal Convolutions

The paper *A Closer Look at Spatiotemporal convolutions for Action Recognition* [Tran et al., 2017] does not tackle the problem of diagnosing any sort of illness,

but rather to recognize actions in video. However, these two problems are not too far apart. A video is merely a 3D volume in which the depth is temporal and each slice is one frame in the video. A video is analogous to a medical image and classifying the action would correspond to diagnosing the illness.

The paper dives into several interesting approaches using 2D convolutions, 3D convolution, and combinations of the two. Through testing the different approaches a new interesting architecture emerged called (2+1)D convolution. The comparisons bring useful insight on the many approaches used in handling classification of 3D volumes. It is important to note that all tests were done using the ResNet architecture [He et al., 2015].

3.4.1 Convolutional Residual Blocks for Video

This subsection will briefly go into the many spatiotemporal convolution variants Tran et al. [2017] tested using residual learning. The variants can be seen in figure 3.4. The residual blocks discussed here are the typical "vanilla" residual blocks explained in He et al. [2015].

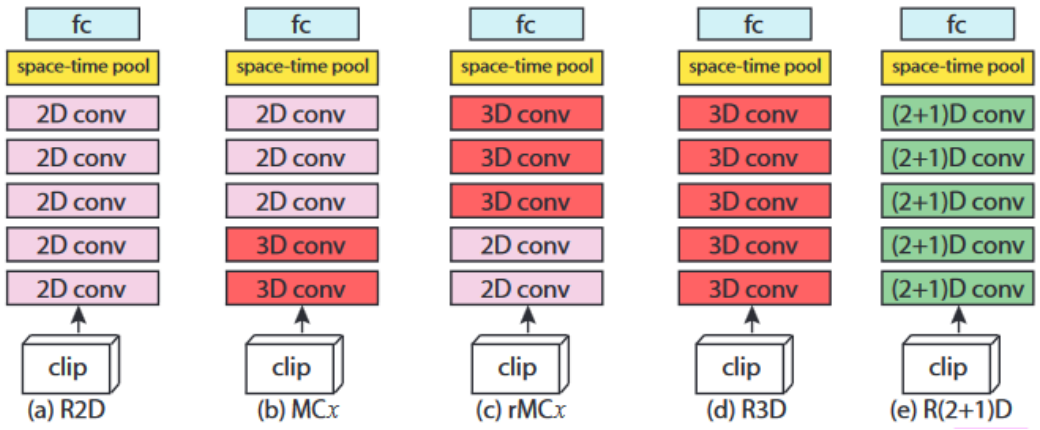


Figure 3.4: Residual network architectures from Tran et al. [2017]. (a) shows 2D ResNets (R2D). (b) displays ResNets with mixed convolutions (MCx, in this case MC3). (c) shows reversed mixed convolutions (rMCx, in this case rMC3). (d) illustrates 3D ResNets (R3D), while (e) shows ResNets with (2+1)D convolutions (R(2+1)D). For clairty, residual connections are omitted.

A residual block is defined as follows: Let z_i be the computed tensor by the i-th convolutional block in the residual network. Each block consists of two convolutional layers with a ReLU activation function after each layer. The output

of the i-th residual block is given by:

$$z_i = z_{i-1} + F(z_{i-1}; \theta_i) \tag{3.1}$$

$F(;\theta_i)$ is the full operation of the block parameterized by weights θ_i, which consist of two convolutions as well as the ReLU activation functions. It should be further mentioned that Tran et al. [2017] uses networks where the sequence of convolutional residual blocks culminates into a top layer performing global average pooling over the entire spatiotemporal volume and a fully-connected layer responsible for the final classification prediction.

R2D and f-R2D Architecture:

This architecture does 2D convolution over the entire video ignoring temporal ordering [Feichtenhofer et al., 2016]. In the case of f-R2D, a series of 2D residual blocks are applied to each frame independently, with the same filters applied to all frames. No temporal information is considered in the convolutional layers and a global spatiotemporal pooling layer collects the information from all the independent frames to make a classification. Such a mechanism can be viewed as taking a vote from each frame instead of looking at the relationship between them. This is in many ways similar the approach used in Apostolopoulos et al. [2016] where one only learns features based on each slice of a volume.

For R2D the input is reshaped from $C\times L\times H\times W$ to $CL\times H\times W$ where C is the number of channels, L is the number of frames in the video or depth, H is height and W is width. The filters are 3D where the depth (channel dimension) of the filter cover the whole depth of the clip (CL), thus collapsing the temporal information into a 2D plane. Therefore, any temporal information cannot be extracted in the layers after.

Perhaps most interesting is the results this achieved. The network in Feichtenhofer et al. [2016] was capable of obtaining accuracies comparable with the state-of-the-art on action recognition at the time. This pulls relations to RetiNet [Apostolopoulos et al., 2016] that also managed to achieve superb results despite the lack of consideration for relations between slices. Tran et al. do not mention any results for f-R2D.

MCx and rMCx Architecture:

These architectures will only be briefly mentioned as they mostly serve as a foundation to the R(2+1)D architecture. MCx stands for *Mixed Convolutions* where x denotes how many layers will be 2D convolutions while the remaining are 3D convolutional layers starting from the end of the network. rMCx is simply the reverse of this. Figure 3.4 may aid in clarifying the concept.

According to Tran et al. [2017], these networks got results comparable to R3D, but with the advantage of substantially reducing the number of trainable parameters. A reduction in parameters is desired as it speeds up learning time considerably, and can times enable training with smaller datasets.

R3D Architecture:

A network only using 3D convolutions [Karpathy et al., 2014] preserve the full information of the input through the network. Each filter has a shape of $N_{i-1} \times t \times d \times d$ where N_i is the number of filters used in the i-th block, t is the depth of the filter, and d is the width and height. They are convolved in 3D and thereby outputs the same dimensionality as the inputs.

Through tests performed by Tran et al. it was found that even R3D performs better than R2D, indicating that the temporal information does indeed aid in obtaining higher accuracies. However, R3D is the most costly architecture having almost three times as many parameters as R2D.

3.4.2 The R(2+1)D Architecture

The R(2+1)D architecture or even the (2+1)D convolutional block is of the biggest interest due to its favorable properties that may suit AMD diagnosis. It is for this reason R(2+1)D gets its own subsection, where a more detailed explanation is given.

The R(2+1)D convolution is a decomposition of 3D convolution. Instead of moving a 4D kernel in three dimensions, a 2D convolution with a 3D kernel is applied on each spatial frame and then finally a 1D convolution is done with a 2D kernel only in the temporal dimension. There are a few advantages in this approach compared to 3D convolution. The first one being the number of non-linearities in one convolutional block. A regular 3D layer does only one convolution with one activation function, while (2+1)D convolution does two convolutions with an activation function after each, doubling the number of non-linearities in one block. Secondly, M_i the hyper-parameter determining number of output channels going from the 2D convolution to 1D convolution can be adjusted. This allows the freedom to test variations of M_i adjusting how many parameters are in the block. If there is a need for a model with fewer parameters, M_i can be reduced so that a (2+1)D convolutional block has fewer parameters than a normal 3D convolutional layer. In Tran et al. [2017] M_i is set as:

$$M_i = \left\lfloor \frac{ts^2 N_{i-1} N_i}{s^2 N_{i-1} + tN_i} \right\rfloor \tag{3.2}$$

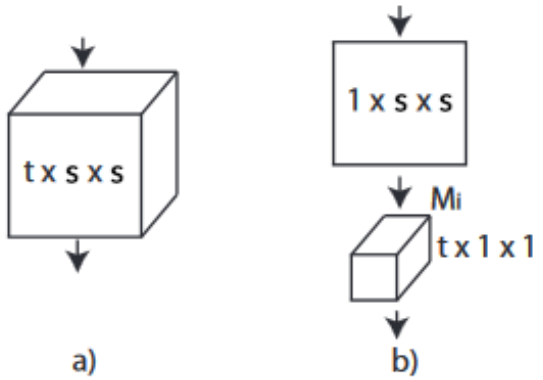


Figure 3.5: 3D and (2+1)D convolution from Tran et al. [2017] with modified labels. (a) A single 3D convolution filter with dimensions $t \times s \times s$ where t is temporal extent (depth) and s is spatial width and height. (b) A single (2+1)D convolutional block, which is a 2D convolution followed by a 1D depth-wise convolution. M_i represents the number of 2D filters, this will affect the number of trainable parameters in the block.

Where t is the size of the temporal dimension, s represents both the height and the width as the paper assumes quadratic kernels and N_i is the number of filters in layer i. This value for M_i makes the number of parameters in the (2+1)D block approximately equal to the number of parameters in a 3D layer, however maintaining the advantage of having twice the number of non-linearities, which allows for more complexity in the network according to Tran et al. [2017].

(2+1)D convolution can be explained as the following steps:

1. Arrange the 3D input so that each frame is considered a case in the mini-batch. In perspective of the 2D convolutional layer, it is doing a 2D convolution on a large batch of individual video frames.
2. Perform 2D convolution with M kernels.
3. Arrange the output back into 3D volumes.
4. Make each line of pixels depth-wise in the volume be an individual case in a mini-batch.
5. Perform 1D convolution with the wanted number of kernels.
6. Arrange output back into 3D volumes.

In the end, the size and dimensionality of the output are the same for a 3D convolutional layer and a (2+1)D convolutional block. Tran et al. showed empirically

that (2+1)D obtained better results than regular 3D convolution, for residual nets on various datasets such as Sports-1M, Kinetics, UCF101, and HMDB51. It was additionally shown that the effect is even more prominent in deeper networks, meaning it is ideal for video classification tasks with today's trend of making rather deep networks. Results from one of the tests are seen in table 3.4.

Net	**# params**	**Clip@1**	**Video@1**	**Clip@1**	**Video@1**
Input		$8 \times 112 \times 112$		$16 \times 112 \times 112$	
R2D	11.4M	46.7	59.5	47.0	58.9
f-R2D	11.4M	48.1	59.4	50.3	60.5
R3D	33.4M	49.4	61.8	52.5	64.2
MC2	11.4M	50.2	62.5	53.1	64.2
MC3	11.7M	50.7	62.9	53.7	64.7
MC4	12.7M	50.5	62.5	53.7	65.1
MC5	16.9M	50.3	62.5	53.7	65.1
rMC2	33.3M	49.8	62.1	53.1	64.9
rMC3	33.0M	49.8	62.3	53.2	65.0
rMC4	32.0M	49.9	62.3	53.4	65.1
rMC5	27.9M	49.4	61.2	52.1	63.1
R(2+1)D	33.3M	**52.8**	**64.8**	**56.8**	**68.0**

Table 3.4: Action recognition accuracy for different forms of convolution on the Kinetics validation set from Tran et al. [2017].

3.5 CNN Fixations

CNN Fixations [Mopuri et al., 2019] is a visualization technique for CNNs. Visualization of ANNs has the goal of displaying the inner working of the network in a way that brings insight for humans. ANNs are often viewed as black boxes that give little "reasoning" for the output. Therefore, it is often desired to find ways to visualize what part of the input made the greatest impact for generating the output. Mopuri et al. [2019] has shown great results, one such example is seen in figure 3.6. This technique could be a key in displaying the symptoms in an AMD detection model. Not only does the technique perform well on a wide range of image-related tasks, CNN Fixations requires no architectural changes, additional training or gradient computation [Mopuri et al., 2019].

Figure 3.6: Illustration of CNN Fixations on images from ImageNet validation set Mopuri et al. [2019]. From left to right: Input images, CNN fixations, heat map calculated through Gaussian blurring of CNN fixations.

3.5.1 Calculation of CNN Fixations

Mopuri et al. proposes a simple, but robust concept in making their visualization. The method works iteratively backwards from the output layer of the model to the input image. For each layer the algorithm determines the set of positively correlated activations from the previous layer. These positively correlated activations are called CNN fixations or discriminative locations and can be found for each layer moving backwards all the way to the input. The method only need a forward pass in the model to calculate the discriminative locations.

Rather than being one algorithm, CNN Fixations is a set of modular algorithms that each work on a particular part of the CNN. For instance, one algorithm works in convolutional layers, while another one handles fully connected layers. The concept is generic enough that it can be used even for advanced architectures such as ResNet [He et al., 2015] and GoogLeNet [Szegedy et al., 2015].

Algorithm 2: Discriminative Localization at Fully Connected Layers

Input: X_l, incoming discriminative locations from higher layer:
$\{X_l[1], ..., X_l[m]\}$
W_l, weights of higher layer l
A_{l-1}, output activations at current layer $l-1$

Output: X_{l-1}, outgoing discriminative locations from the current layer

$X_{l-1} = \emptyset$
for $i = 1...m$ **do**
  $W_l^{X_l[i]} \leftarrow$ weights of neuron $n_l^{X_l[i]}$
  $C \leftarrow A_{l-1} \odot W_l^{X_l[i]}$ `// Point-wise multiplication`
  $X_{l-1} \leftarrow$ append (X_{l-1}, $args(C > 0)$)

Algorithm 2 shows how to compute discriminative locations for fully connected layers. m is the number of discriminative locations calculated from layer l in the network. The discriminative locations X_l are indices for neuron locations. $W_l^{X_l[i]}$ is the weights from all neurons at layer $l-1$ to neuron $n_l^{X_l[i]}$ at location $X_l[i]$ in layer l. C is the input activations going into neuron $n_l^{X_l[i]}$, while A_{l-1} is the output activations coming out of neurons in layer $l-1$.

The first set of fixations found in the output layer is usually the prediction of the model. For instance, if the last layer is a softmax layer, the discriminative location will be the single output activation with the highest value. As seen in figure 3.7, the algorithm works by looking at the input activations going into each of the neurons at discriminative locations. The discriminative locations for layer $l-1$ are the positions of neurons that contributed a positive input activation to a discriminative location in layer l.

Algorithm 3: Discriminative Localization at Convolution Layers

Input: X_l, incoming discriminative locations from higher layer:
$\{X_l[1], ..., X_l[m]\}$
W_l, weights of higher layer l
A_{l-1}, output activations at current layer $l-1$

Output: X_{l-1}, outgoing discriminative locations from the current layer

$S(.)$: A function that sums a tensor along xy axes $X_{l-1} = \emptyset$
for $i = 1...m$ **do**
  $W_l^{X_l[i]} \leftarrow$ weights of neuron $n_l^{X_l[i]}$
  $A_{l-1}^{X_l[i]} \leftarrow$ receptive activations for neuron $n_l^{X_l[i]}$
  $C \leftarrow S(A_{l-1} \odot W_l^{X_l[i]})$ `// Per channel contribution`
  $ch \leftarrow argmax(C)$ `// Discriminative channel`
  $(P_x, P_y) \leftarrow argmax(A_{l-1} \odot W_l^{X_l[i]}(:,:,ch))$ `// Discriminative location in channel 'ch'`
  $X_{l-1} \leftarrow$ append (X_{l-1}, $ch \cdot k_{l-1}^2 + P_x \cdot k_{l-1} + P_y$)
$X_{l-1} \leftarrow unique(X_{l-1})$

Algorithm 3 describes how one finds discriminative locations for convolutional layers. In a convolutional layer, all neurons in the current layer is not connected to the previous layer. Therefore, algorithm 3 uses $A_{l-1}^{X_l[i]}$ which is all the activations linked by weights to neuron $n_l^{X_l[i]}$. Furthermore, C is the input activations going into $n_l^{X_l[i]}$ where the values in each channel is summed together, leaving an array of total input activations from each channel going into neuron $n_l^{X_l[i]}$. ch is the channel index for the channel with the highest total input activation. Once ch is

found, algorithm 3 finds the coordinates of the highest input activation in channel ch, as seen in line 7. The higher dimensional discriminative location is flattened in line 8 and stored as a fixation for layer $l-1$, where k_{l-1} is the kernel size in layer $l-1$. A fixation in layer $l-1$ may be linked to more than one fixation in layer l, which means one fixation can be stored multiple times. It is unnecessary to compute the next fixations twice, therefore X_{l-1} is reduced to only unique discriminative locations.

Algorithm 3 can be generalized to any n-dimensional convolution with few differences. Functions S would generalize to a function reducing channels to one total input activation value, and line 7 may have n-dimensional coordinates. Additionally, line 8 would need to flatten the n-dimensional discriminative location. The general rule followed by Mopuri et al., is to always find the highest contributor on a large scale and move smaller and smaller. For instance, when computing CNN fixations in ResNet, one first determines which path in the residual block contributes most, and then computes fixations only within that path.

Mopuri et al. also shows how fixations can be computed for other architectures such as recurrent neural networks, however, such architectures are irrelevant in this thesis.

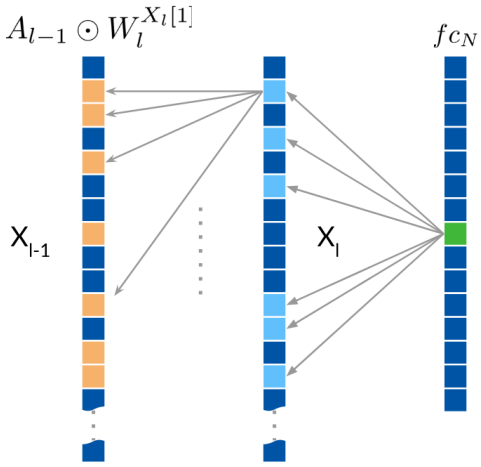


Figure 3.7: Illustration of finding discriminative locations in fully connected layers from Mopuri et al. [2019]. fc_N is the Nth fully connected layer, X_l is the discriminative locations at layer l and $A_{l-1} \odot W_l^{X_l[1]}$ is the input activations going into neuron $X_l[1]$ in layer l.

3.5.2 Display of CNN Fixations

Mopuri et al. mostly focus on displaying CNN fixations on the input image, as it is hard to understand the output images of layers inside a CNN. The discriminative locations in the input image are distributed on each of the channels of the image. When displaying these fixations, their channel position is ignored and they are all displayed on one plane on top of the input image itself, as seen in figure 3.6. However, before fixations are displayed, a filter is applied to all the fixations to remove outliers. Mopuri et al. defines an outliers as a fixation that is not supported by sufficient neighboring fixations. In more detail, a fixation needs to have more than a certain percentage of total fixations within a given circle around it to remain. Mopuri et al. selected the circle radius to be 10% of the diagonal of the image and required 5% of total fixations to be within the circle.

A localization map is computed by taking the Gaussian blur of the fixations. The result is something resembling a heat map where high concentrations of CNN fixations is highlighted, as seen in the rightmost images in figure 3.6. The range goes from low interest with blue to high interest in red. The representation brings an additional perspective on what the model focuses on. Higher concentration of fixations is shown as red and indicates higher interest in that region.

3.6 Summary

This chapter has covered the related works to this thesis and the methodology in finding and selecting related works. The first paper presented was about RetiNet, a network using two stages of training. First, a pre-training stage where a network was trained using the slices in the OCT scans as individual cases for training. Some parts of this network was transferred over to a second network. This second network flattened the OCT scans by stacking the slices height-wise, and only trained the fully connected layers. This yielded state-of-the-art results on classification of AMD.

Lee et al. proposed a pure 2D convolutional approach to classification of AMD by using parts of the pre-trained network VGG16. They also used a visualization technique called an occlusion test to evaluate performance of the network.

Spatiotemporal convolution was examined for classification of actions in video. This type of convolution is a decomposition of 3D convolution where one first executes 2D convolution for each slice in the 3D volume and then execute 1D convolution for each string of depth-wise pixels. It was shown that this outperformed regular 3D convolution for action recognition obtaining state-of-the-art

results. This may be useful even in the domain of AMD detection.

CNN Fixations is a visualization technique for CNNs. It works by going backwards through the network finding positively correlated activations called CNN Fixations or discriminative locations.

Chapter 4

Architecture

Some interesting approaches on classification of medical data has been mentioned in the previous chapter. These approaches disregard the depth-wise relations in the input volumes, but still obtaining impressive results. Tran et al. found most 3D models to be superior to 2D, and proposed a (2+1)D convolution, which achieves state-of-the-art results by reducing the computational complexity while maintaining performance. I draw a parallel between Tran et al. [2017] and my own task and argue there might exist a similar advantage in using 3D convolution for AMD diagnosis. In this chapter, the architecture of my model will be covered. The architecture is implemented using Python with the framework PyTorch.

Full implementation and datasets are found at:
https://github.com/StianHanssen/OptiNet

A final note: Throughout this chapter, the dimensions of the input is given as $(channels \times depth \times height \times width)$. Dimensions of kernels will be $(depth \times height \times width)$.

4.1 Base Block

With experimentation and inspiration from Tran et al. [2017] and He et al. [2016], an architecture was assembled. A *base block* consisting of multiple layers is used as the basis for most of the network. Just like in ResNet [He et al., 2016], a certain pattern of sequential layers occurred that can be made into its own "super

layer" called a block. Through experimentation I discovered that a base block with two convolutions was ideal. Validation accuracy went down when using one convolution in the base block. Using base blocks with 3 convolutions made the model so large it was difficult to train with the given resources. An overview of the block is given in table 4.1.

Layers / Methods	Channels In	Channels Out
(2+1)D Convolution	*in_channels*	*out_channels*
Batch Normalization	-	-
ReLU Activation Function	-	-
(2+1)D Convolution	*out_channels*	*out_channels*
Batch Normalization	-	-
ReLU Activation Function	-	-
Downsampling	-	-

Table 4.1: The base block is specified by arguments: *in_channels*, *out_channels*, *down_sample* and *stride*, which means number of channels going into the block, number of channels going out of the block, the downsampling method to use (specified with its own arguments before added to block) and stride for the (2+1)D convolutional layer respectively. For batch normalization, ReLU activation, and downsampling it is assumed they will not change the number of channels. The methods are always applied to the whole input.

The block consists of 4 components: (2+1)D convolution, batch normalization, ReLU activation and downsampling. The (2+1)D convolutional layers use a $3 \times 3 \times 3$ kernel with stride of $1 \times 1 \times 1$. A padding is also added with thickness 1 in all directions. The ReLU activation function is applied point-wise to the whole output of the batch normalization layer. The downsampling technique used is 3D max pooling with varying kernels and strides, as described in table 4.2. All the convolutional layers use bias initialized to 0. The batch normalization is applied to all values coming from the convolutional layer. It was used for the properties explained in section 2.4.4.

4.2 OptiNet

For the sake of simplicity, the model developed in this thesis will be called OptiNet. The full architecture of OptiNet is seen in table 2.4 as well as in figure 4.1. It follows a pattern of repeated base blocks that contains two convolutions and a downsample layer. They all use 3D max pooling as their downsampling

Layers	Parameters	Activation	Output Shape
Input			$1 \times 32 \times 256 \times 256$
Base Block	In: 1, Out: 64, DS1	ReLU	$64 \times 16 \times 64 \times 64$
Base Block	In: 64, Out: 128, DS1	ReLU	$128 \times 8 \times 16 \times 16$
Base Block	In: 128, Out: 256, DS1	ReLU	$256 \times 4 \times 4 \times 4$
Base Block	In: 256, Out: 512, DS2	ReLU	$512 \times 2 \times 2 \times 2$
(2+1) Convolution	In: 512, Out: 512	ReLU	$512 \times 2 \times 2 \times 2$
3D Max Pool	Kernel=Stride: $2 \times 2 \times 2$		$512 \times 1 \times 1 \times 1$
Flattening Dimensions			512
Fully Connected	Size: 256	ReLU	256
Fully Connected	Size: 1	Sigmoid	1

Table 4.2: The full architecture of OptiNet. In and out parameters specify the number of channels going in and out of a layer. DS1 stands for $Down\_Sampling1$ which is 3D max pooling with $2 \times 4 \times 4$ kernel and stride. DS2 stands for $Down\_Sampling2$ which is 3D max pooling with a $2 \times 2 \times 2$ kernel and stride. Size specifies the number of neurons in the fully connected layers. The (2+1)D convolutional layer used a $3 \times 3 \times 3$ kernel with stride $1 \times 1 \times 1$ and padding of 1 in all directions.

layer with two variations of kernels and strides. The first one called $DS1$ uses kernel and stride of dimensions $2 \times 4 \times 4$. The second variation called $DS2$ use kernel and stride of dimensions $2 \times 2 \times 2$. The reason for the usage of the two variations is due to the OCT scans having the shape $1 \times 32 \times 256 \times 256$. The height and width are eight times larger than the depth. It is assumed the earlier features in the network are noisier and can afford to be reduced by a factor of four, while the later layers use $DS2$ because it is assumed their features are higher level and more important.

After the base blocks, one additional convolution and max pool is added. This is to add additional complexity to the network while reducing the number of parameters. The final (2+1)D convolutional layer has the same number of input and output channels for this reason. The extra max pooling layer allows the number of neurons to be 512 instead of $512 \times 2 \times 2 \times 2 = 4096$. In terms of parameters between the last convolutional layer and the first fully connected layer: $512 \times 256 = 131\ 072$ parameters instead of $4096 \times 256 = 1\ 048\ 576$, excluding bias. A substantial reduction in parameters was necessary due to the limited capacity of the GPU in terms of memory.

The number of fully connected layers was chosen on a similar basis. Having one layer lead to the network never converging while two layers performed quite well. As space was already limited, three layers was not tested.

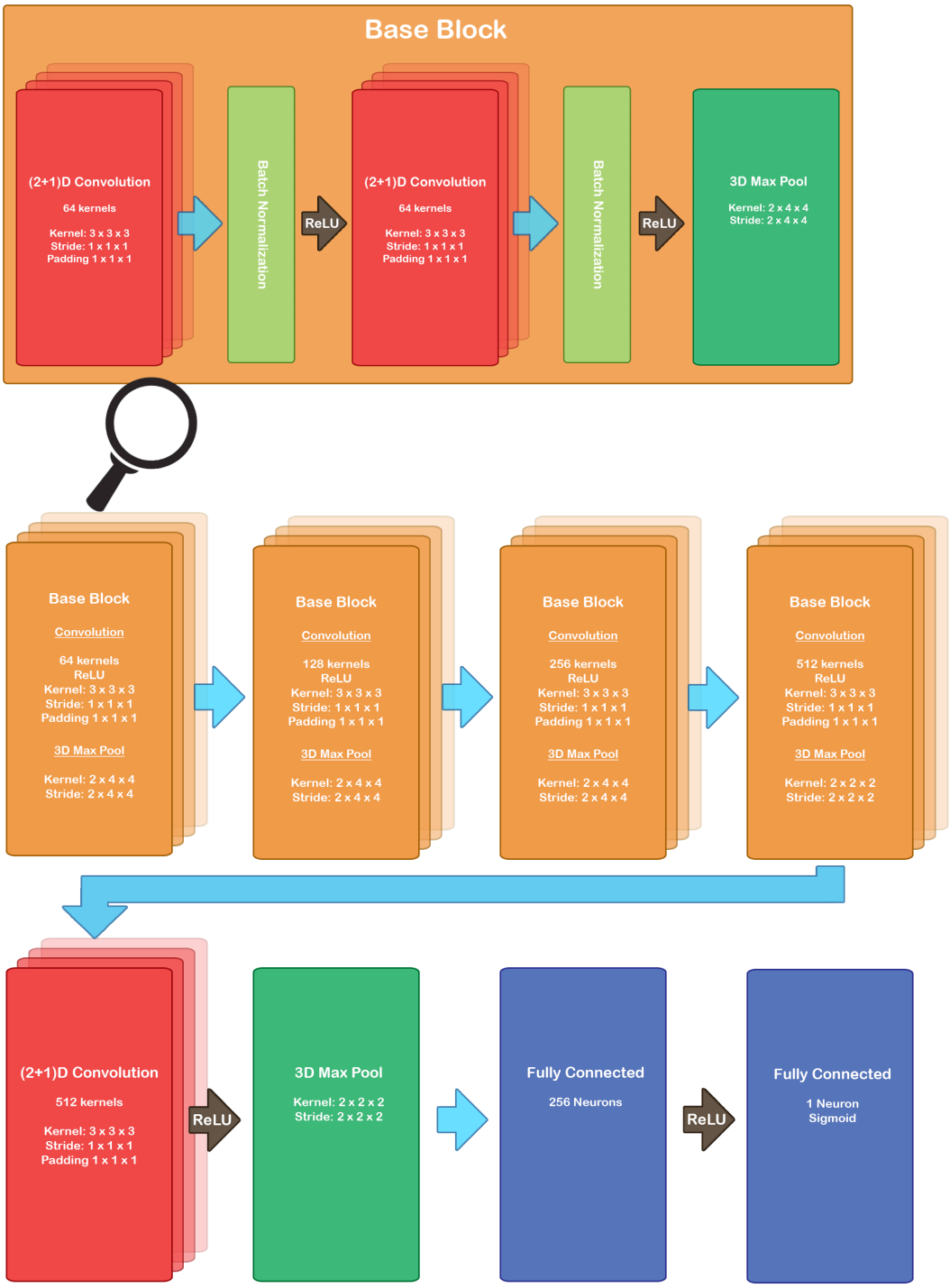


Figure 4.1: Illustration of OptiNet

Using four base blocks was a result of needing to reduce the number of neurons before reaching the fully connected layers. Adding any more base blocks would once again cause problems for the GPU. The small kernels of $3 \times 3 \times 3$ with strides $1 \times 1 \times 1$ was chosen for its popularity in modern networks with its roots in VGG from Simonyan and Zisserman [2014]. ReLU deals with vanishing gradients better than many other activation functions, in addition to overwhelmingly be the choice for activation in the literature. As covered in section 2.4.3, max pooling makes the network more robust against noise and variation in rotation present in the data.

4.3 Summary

OptiNet's architecture is based on many popular choices in the field of convolutional neural networks. The use of kernels of size 3 is inspired by the famous CNN model VGG16. The use of batch normalization is inspired from ResNet as well as other models that have performed well in the ImageNet competition. The activation function ReLU and max pooling for downsamipling was used for the same reasons.

The number of convolutional layers was derived by trial and adjustment as well as to fit in the restrictions in resources available. A repeating pattern in the network was abstracted to its own layer called a base block. Reducing a block to only have one convolution lead to significant decrease in validation accuracy. Furthermore, there was no capacity for a network much bigger than OptiNet in the experimental setup.

Each block contained two convolutional layers, which used (2+1)D convolution. Inspired by Tran et al.'s description of the action recognition field, I saw similarities to the field of detecting AMD. (2+1)D convolution seemed to be ideal for training where high complexity is wanted, but fewer parameters are required because the dataset is small. The architecture can however easily switch to using 3D convolution, which is explored further in the next chapter. The overview of the architecture can be seen in figure 4.1.

Chapter 5

Experiments and Results

This chapter will cover the experimental aspects of the project. It will go into detail about the handling of the dataset and how training was conducted. The reasoning for choices such as hyper-parameters will be given, as well as data augmentation choices and metrics used. At the end of the chapter, the evaluation method and results will explain how the network was tested and how it performed.

5.1 Experimental Plan

This section covers the experiments planned for the thesis and what these experiments should answer. The following experiments are planned:

- Train multiple models with the same datasets and evaluate them on the validation set. Do this while testing the OptiNet architecture with (2+1)D convolution and regular 3D convolution. Find the mean and variance for the set of collected results. These result allow for a more realistic evaluation of the networks performance by accounting for random initialization. This will aid in evaluating whether both research goals are reached and give metrics needed to answer research question 2.
- Train multiple models of OptiNet using (2+1)D convolution and 3D convolution on a shuffled Duke dataset. For each training session of a model, shuffle the dataset and then split it into the training and validation partition. This experiments tests the models to see how sensitive they are

to arrangements of training/validation splits, and shed light on research question 1.

- Run CNN Fixations on q selected model that seem to perform well. CNN Fixations will highlight what the model focus on, which helps in goal 1. The visualizations will be evaluated by a health professional to see how the model performs from a human perspective. The model will be evaluated on focus points rather than overall classification.

5.2 Experimental Setup

This section will highlight how the datasets are used, how training was done as well as the evaluation process. The datasets comes from Duke University and St. Olav's University Hospital, and was altered to fit the specific task of the project. The training process includes the hyper-parameters and certain design decisions made for OptiNet. The evaluation process covers which metrics were used to evaluate the results of the network and how the evaluation was done.

5.2.1 Dataset from Duke University

In order to compare performance of OptiNet to other works, the the biggest public dataset on AMD was used. This dataset is from Duke University [Farsiu et al., 2014] and is referred to as the Duke dataset in this thesis. It contains OCT scans from 269 AMD cases and 115 healthy cases. Each scan is of shape $1 \times 100 \times 512 \times 1000$ ($channels \times depth \times height \times width$).

The OCT scans were too big for the hardware setup I had available. They were, therefore, cropped down to shape $1 \times 32 \times 512 \times 512$. Center crop was used for all dimensions except for in height where only the bottom was cropped off. This is because the area of interest seemed to be mostly located at the top in the height dimension. Furthermore, the height and width were resized so the final shape of the OCT scans became $1 \times 32 \times 256 \times 256$. Finally, all slices in the scans had their mean set to 0 and variance to 1. The reasoning for this can be found in section 2.4.4. The adjusted OCT scans are stored in batches of 24 in the HDF5 format for its compactness and fast loading time.

The dataset was partitioned into a training set and a validation set. The training set contains 221 AMD scans and 91 healthy scans. This leaves the validation set to have 48 AMD scans and 24 healthy scans. In terms of percentage of AMD in

each partition: Training has $\sim 71\%$ and validation has $\sim 67\%$. Validation was only used to evaluate the network and was never trained on.

A fairly small amount of data augmentation was used. To ensure as few implementation flaws as possible only horizontal flipping on each slice was applied. All slices in a volume were flipped the same way. This doubled the size of the dataset. The training partition is also shuffled during training.

5.2.2 Dataset from St. Olav's University Hospital

For simplicity, the dataset from St. Olav's University Hospital will be called the St. Olav's dataset. The dataset consists of data collected through the NorPED study conducted by Dr. Arnt-Ole Tvenning. The healthy cases are collected from volunteer students at St. Olav's University Hospital.

There are 343 cases of AMD and 46 healthy cases in the original dataset. One OCT scan is stored as a folder of 2D images that each represent a slice of the OCT scan. The folders containing AMD cases also contain some additional information such as visual acuity and a 2D photograph of the rear of the eye, however, this information is never used. Additionally, there is an excel sheet mapping patient ids to OCT scan ids. Almost all patients in the dataset contributed more than one OCT scan from different time periods. The scans' dimensions are not consistent across the dataset. To make the dataset easier to handle, it was processed in a similar fashion to the Duke dataset.

The OCT scans were first pruned by only accepting scans with dimensions $1 \times 32 \times 384 \times 384$ or larger. Any OCT scan larger than $1 \times 32 \times 384 \times 384$ was center cropped to fit these dimensions and then resized slice by slice to the final dimensions $1 \times 32 \times 256 \times 256$. The final number of AMD cases is 337 while the number of healthy cases remained unchanged.

The St. Olav's dataset was partitioned into a training and a validation partition where the training partition holds $\sim 70\%$ of the cases. When making the partitions it was enforced that OCT scans from one patient remained in only one of the partitions. This is to ensure no training case is close to identical to a validation case. A greedy algorithm was used to distribute the patients into two partitions. The two partitions are even in number of OCT scans. To achieve a 70/30 split, a fake patient was initially placed in the validation partition. Once the algorithm finished distributing patients 50/50 (by number of OCT scans), the fake patient was removed shifting the case ratio to the correct 70/30 split, where the training partition holds 70% of the cases. The procedure was done separately for AMD cases and healthy cases so that both partitions would have approximately

the same distribution of AMD and healthy cases. The training and validation partition ended up with $\sim 87\%$ and $\sim 89\%$ AMD, respectively.

The final processed dataset was stored in the HDF5 format with a meta file giving information about the dataset. This meta file gives information such as number of cases in both partitions, the training/validation ratio, which patients are in which partition and more.

5.2.3 Training Process

OptiNet was trained on an NVIDIA Tesla V100 PCI-E 16 GB GPU with the hyper-parameters and design decisions found in table 5.1.

Parameter/Decision	Choice
Training Mini-Batch Size	1
Validation Mini-Batch Size	1
Learning Rate	0.00001
Loss function	Binary Cross Entropy
Bias Initialization	All values set to 0
Weight Initialization	Xavier [Glorot and Bengio, 2010]
Optimizer	Adam [Kingma and Ba, 2014]

Table 5.1: OptiNet's hyper-parameters and design decisions for training.

The mini-batch size was limited to the storage capacity of the GPU, and as it turned out, it gave quite good performance. Therefore, little effort was put into multi-GPU training which would allow bigger mini-batches. The training process stored the model parameters every time it found a better validation accuracy. This method was used instead of early stopping. When training multiple models sequentially on one GPU, storing models and using time limits was more ideal because a total time limit on the GPU resource is required by server guidelines. The training period had a set time of 3 hours per model. The training time was decided after finding that a model would usually reach peak validation loss and accuracy during this period. If a new design decision was tested, a much greater time frame was set and the network was periodically manually monitored. Training loss was logged every step to aid in the evaluation of how the network performed.

Validation loss and accuracy was calculated every 10 steps. Both loss and accuracy were calculated for the whole validation set and averaged. A prediction for accuracy was made by thresholding the network output on 0.5. To monitor

if OptiNet was stuck only selecting one class, an additional measure called AMD ratio was added. AMD Ratio is calculated by $\frac{AMD\ Preditions}{Total\ Preditions}$ for a validation set. These metrics could be live monitored using *TensorBoard*.

One could select between datasets and whether to use (2+1)D convolution or regular 3D convolution through the training parameters, making it easy to switch between experiments.

In order to train multiple models with the same setup, one could also select to run multiple training sessions in sequence where each model get a set time frame to train in.

5.2.4 Visualization

A general implementation of CNN Fixations [Mopuri et al., 2019] was made for PyTorch. The CNN Fixation framework followed the algorithms described in section 3.5.1 with a few deviations. Rather than storing the discriminative locations as flattened 1D indices, their dimensionality was preserved and transformed to the right dimensionality for each layer.

Some algorithms in the framework had to be created based on descriptions in Mopuri et al. [2019]. Max pooling already finds the highest contributing values, thus one only need to go back through the max pool layer from a discriminative location to find the next discriminative location. Batch normalization normalizes the values, which means the highest contributing values remain the same. In this case, the discriminative locations got passed on to the next layer.

A new fixation algorithm was made for (2+1)D convolution by working through the components of (2+1)D convolution backwards. This was done by first transforming the input discriminative locations to the form (batch size $\times$ height $\times$ width, channels, depth), which allows for the computation of fixations in 1D convolution. "$\times$" represents multiplication in this case. The discriminative locations calculated from 1D convolution is further transformed to be of form (batch size $\times$ depth, channels, height, width) to fit 2D convolution. Once fixations for 2D convolution is calculated, the fixations are transformed back to the regular form (batch size, channels, depth, height, width) and passed to the next layer outside of the (2+1)D convolutional block.

The fixations are visualized by showing 3 pictures side by side, the exact same way as in figure 3.6 in section 3.5. Scrolling with the mouse-wheel moves the view depth-wise in the OCT scan. Only one slice is shown for each scroll step.

The localization map was computed as described in section 3.5.2. A 3D Gaussian

blur was applied over all of the discriminative locations in the OCT scan. The map was sliced up, set opacity to 0.6 (where 1 is opaque) and layered on top of the corresponding OCT slices.

To obtain activations and weights from PyTorch, hooks were added to each layer. The hooks would place data into a global dictionary linking a layer or module object to activation tensors. All fixation algorithms had access to the dictionary and could therefore obtain all the necessary data.

The use of visualization was inspired from Lee et al. [2017] in section 3.3. By having health professionals evaluate the focus points of OptiNet, one can gain further insight into how the model views AMD and how it may differ from human evaluation. It can also aid in assessing whether OptiNet is a good model for detection of AMD.

5.2.5 Evaluation Process

Making a training, validation and testing partition would leave the network with an unreasonably small amount of cases to train on. Therefore, the validation set was also used for the final evaluation of OptiNet. Data augmentation was turned off, so only real data was evaluated. Each OCT scan was run through the network without training. The following metrics were calculated:

- **Average Validation Loss**: Calculated using Binary Cross Entropy
- **Accuracy**: Fraction of correct predictions among all predictions. Making predictions by thresholding on 0.5.
- **Precision**: Fraction of true instances among the predicted true instances.
- **Recall**: Fraction of true instances that have been predicted over all true instances.
- **Receiver Operating Characteristic (ROC) Curve**: Relation between true positive rate and false positive rate. The rate is found by adjusting the threshold for making a prediction.
- **Area under the curve of ROC (AUC)**: The final measure of correctness showing how close the network is to ground truth over predictions of all thresholds.

Accuracy was selected because it was used when storing the "best" model during training. Precision and recall was selected for details on how the network evaluates data. Finally, AUC is chosen as the overall performance measure as it works

similar to accuracy, but goes further to consider the degree of certainty for the predictions as well.

To find sensitivity to random factors in the model, the same model would be initialized and trained 10 times. Mean and variance was calculated for AUC, validation accuracy, precision and recall.

Furthermore for the Duke dataset, 10 models were trained with a randomly initialized train/validation split in the ratio approximately 70/30. This was done by collecting all cases into one array, shuffle them, and finally split the array so that both partitions have the same number of cases as they originally had. Finally, to ensure the evaluation process would not use any training cases, the shuffled indices were stored with the model. That way one can make the exact same shuffle and split again to obtain the validation partition used on a particular model.

CNN fixations for a performing model was calculated and sent to Dr. Arnt-Ole Tvenning for evaluation. Dr. Tvenning went through the OCT scans evaluating how similar the focus points are to what he would focus on, whether there is anything OptiNet puts more weight on than he would, and whether there are times OptiNet's assessment seems questionable.

5.3 Experimental Results

One experiment would usually take about 30 hours, with 3 hours of training per model. In the early stages of the project, it was more common to train a single model for 12 hours time, and monitor how the training went live.

5.3.1 Mean performance of OptiNet

OptiNet was trained with the same setup 10 times and evaluated for both datasets and both convolutions. The architecture is identical with the exception of which type convolution is used. The results on the Duke dataset can be seen in table 5.2 and the results for the St. Olav's dataset is seen in table 5.3.

The results reveal that (2+1)D convolution perform better than 3D convolution on all metrics when trained on the Duke dataset. Of all the metrics measured, recall makes up the biggest difference with ~ 0.01 difference in mean recall. This is reflected in their AUC values, where (2+1)D convolution performs significantly better.

Metric	OptiNet with (2+1)D Convolution	OptiNet with 3D Convolution
Accuracy	0.97917 ± 0.025039	0.91667 ± 0.028464
Precision	0.98361 ± 0.019689	0.98554 ± 0.024118
Recall	$\mathbf{0.98541 \pm 0.026435}$	0.88958 ± 0.055941
AUC	$\mathbf{0.99765 \pm 0.0036011}$	0.98993 ± 0.0063338

Table 5.2: OptiNet's results on the Duke dataset. Accuracy, precision, recall, and AUC are shown in fractions. The results are of the format "mean ± standard deviation", over the results of 10 trained models for each convolution type.

Metric	OptiNet with (2+1)D Convolution	OptiNet with 3D Convolution
Accuracy	0.98860 ± 0.0078947	0.99561 ± 0.0058844
Precision	0.99518 ± 0.0088118	0.99805 ± 0.0039026
Recall	0.99215 ± 0.0058824	0.99705 ± 0.0044927
AUC	0.99910 ± 0.0010621	$\mathbf{0.99984 \pm 0.00049020}$

Table 5.3: OptiNet's results on the St. Olav's dataset. Accuracy, precision, recall, and AUC are shown in fractions. The results are of the format "mean ± standard deviation", over the results of 10 trained models for each convolution type.

Results shown in table 5.3 show the opposite of 5.2. In this case, OptiNet with 3D convolutions performs better on all metrics, though the results are very close. OptiNet with 3D convolutions obtain an AUC close to 1.

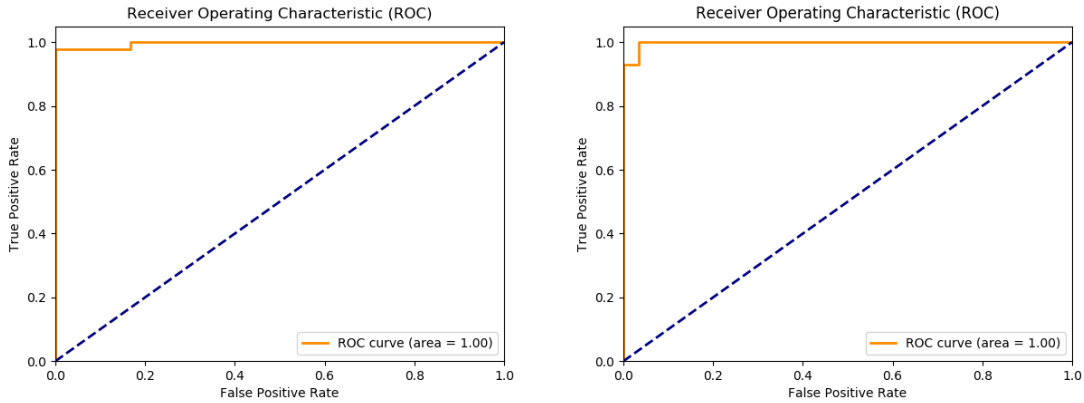
5.3.2 OptiNet's Results on Shuffled Partitions

To ensure the models are not sensitive to change in training and validation partitions, 10 additional models were trained with the total dataset shuffled before splitting into training and validation sets for each of the models. This was only performed on the Duke dataset because it has no dependencies between OCT scans. In order to perform such a shuffle on St. Olav's dataset one would need to keep the constraint of all OCT scans belonging to one patient remains in only one partition, while at the same time ensure the training and validation partitions remain approximately the same size for each model. It was decided that too many factors could affect the result of such an experiment. Thus, the experiment was only performed on the Duke dataset.

Table 5.4 shows how the two architecture variations performed on shuffled par-

Metric	OptiNet with (2+1)D Convolution	OptiNet with 3D Convolution
Accuracy	0.96528 ± 0.020833	0.93611 ± 0.031793
Precision	0.97610 ± 0.032112	0.95161 ± 0.049979
Recall	0.97598 ± 0.023258	0.95782 ± 0.046925
AUC	$\mathbf{0.997009 \pm 0.0034974}$	0.98477 ± 0.013458

Table 5.4: OptiNet's results on the Duke dataset where cases are randomly shuffled between partitions before each of the 10 training session. Accuracy, precision, recall, and AUC are shown in fractions. The results are of the format "mean ± standard deviation", over the results of 10 trained models for each convolution type.



(a) ROC curve of a random model of OptiNet using (2+1)D convolution.

(b) ROC curve of a random model of OptiNet using 3D convolution.

Figure 5.1: The ROC curve result of an OptiNet model trained on the Duke dataset with shuffled partitions. AUC is so close to 1 that it gets rounded to 1 when presented with two decimal places.

titions. Just like in table 5.2, (2+1)D convolution outperforms 3D convolution on every metric. The AUC of 99.7% is of significance because it gives the best metric for comparison to related works. This is because it accounts for all random factors and resembles the testing approach of other works the most.

Many of the models both using 3D convolution and (2+1)D convolution obtain AUCs of 1 or close to 1 on both datasets. Figure 5.1a and figure 5.1b show the ROC curve of two random models of OptiNet. The models were trained on the shuffled Duke dataset.

5.3.3 Visualization

The visualization was focused on the St. Olav's dataset because the health professional assessing the visualization created the St. Olav's dataset. The health professional can quickly find differences between his own evaluation and the model's. Figure 5.4 shows how the system displays results to the user on an AMD case. Three images are given, displaying the regular OCT scan, CNN fixations, and a localization map made from the fixations. A color bar is added to display the meaning of each color, where a high value, displayed in red, implies high interest from the model. The range is 0-255 because this is the intensity range of a channel. Figure 5.2 illustrates visualization of a control case. A few visualizations gave odd focus points that could not be justified as a sound assessment. One such example can be seen in figure 5.3.

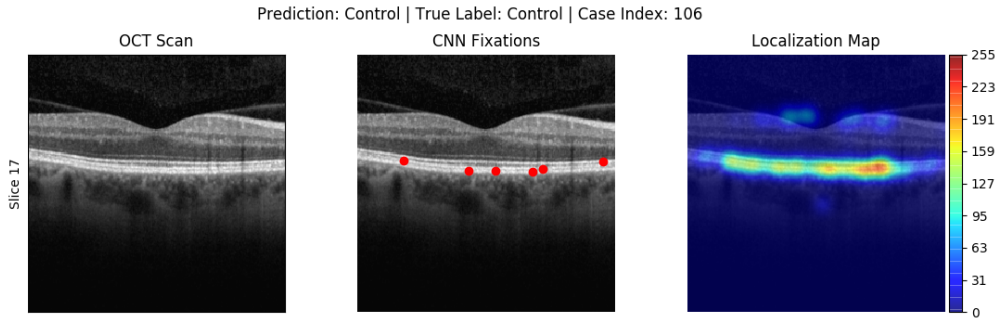


Figure 5.2: Visualization of focus points made from OptiNet using (2+1)D convolution on the St. Olav's dataset for one slice. The OCT scan displays a control case.

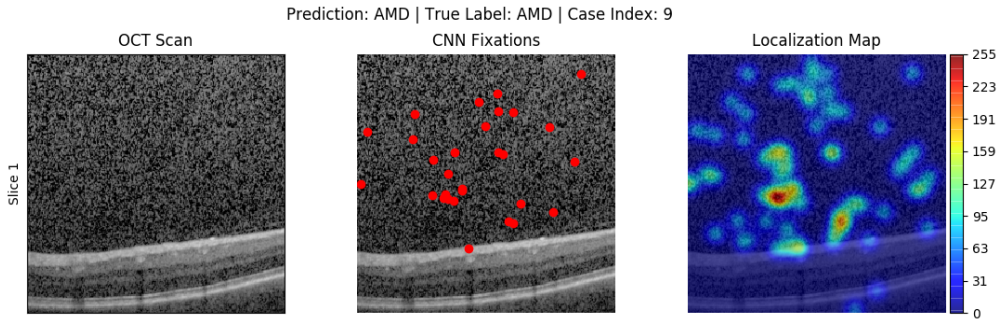


Figure 5.3: Visualization of focus points made from OptiNet using (2+1)D convolution on the St. Olav's dataset for one slice. The OCT scan displays an AMD case. The visualization displays questionable assessment according to Dr. Tvenning.

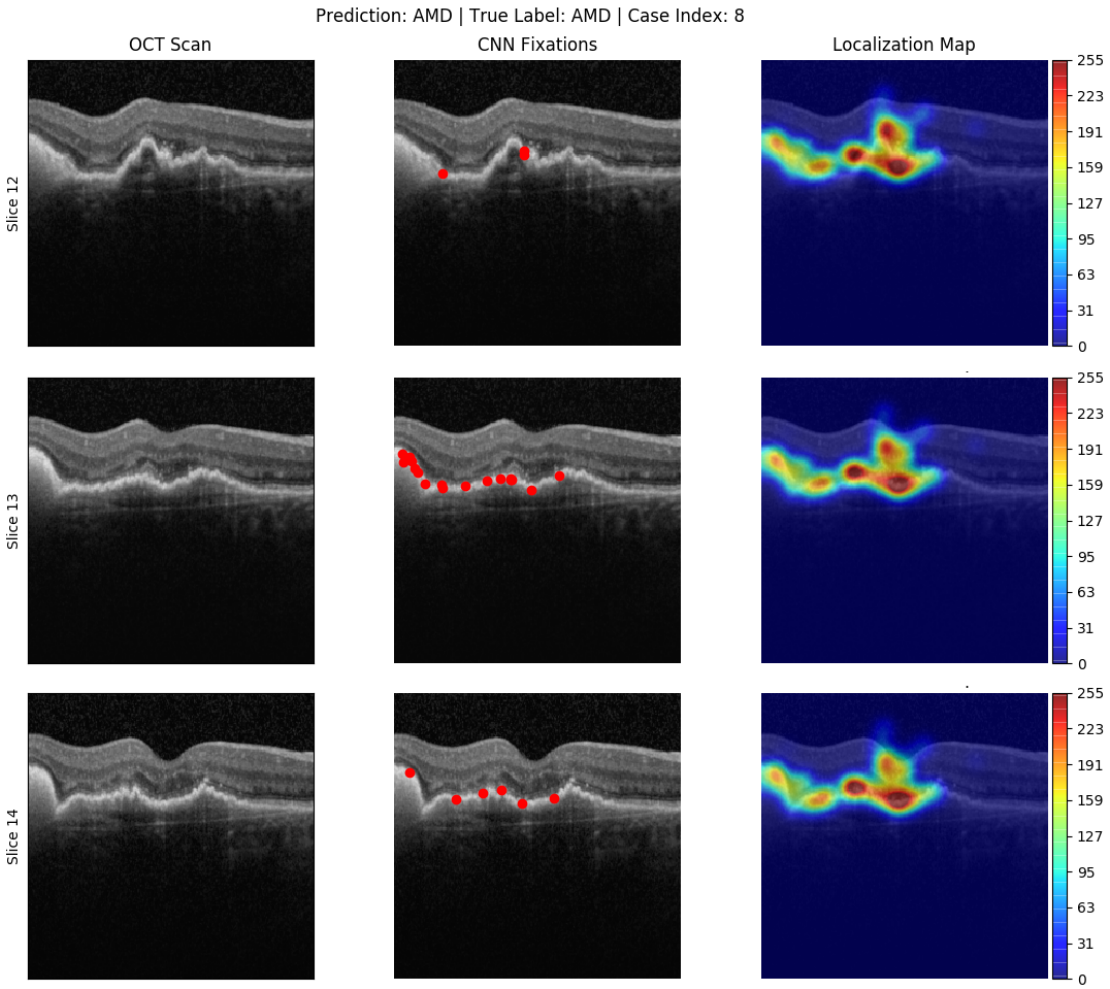


Figure 5.4: Visualization of focus points made from OptiNet using (2+1)D convolution on the St. Olav's dataset. Each row displayed a slice of an OCT scan. The OCT scan displays a case of AMD. In the system, each slice is displayed sequentially after each other by scrolling the mouse wheel back and forth.

5.4 Summary

The experimental plan consisted of three experiments: One to train 10 models using both types of convolutions on both dataset, second one is to do the same, but with shuffled partitions on the Duke dataset, and the final one was to evaluate CNN fixations made from running OptiNet on the St. Olav's dataset.

The two datasets used were pre-processed to make them easier to handle. The St. Olav's dataset had a slightly more advanced partitioning to ensure no OCT scans from one patient ended up in both partitions. Both datasets had an approximate train/validation split of 70/30 in number of OCT scans.

Models were trained for about 3 hours each, and various metrics were collected for live monitoring. In the evaluation process, the metrics accuracy, precision, recall and AUC were of the highest interest. The first experiment found that OptiNet using (2+1)D convolution performed slightly better on the Duke dataset, a seen in table 5.2. The opposite was found for the St. Olav's dataset shown in table 5.3.

The second experiment once again confirm that OptiNet using (2+1)D convolution performed better on the Duke dataset. Notably, it obtained an AUC of $\sim 99.7\%$. This value would be the most natural to use when comparing OptiNet to related works. This is because this experiment resembles the experimental setup of related works the most.

The third experiment found interessting visualization of what OptiNet focused on in an OCT scan. Figure 5.4 shows a case of AMD, figure 5.2 shows a control case, while figure 5.3 shows an AMD case where OptiNet predicts correctly, but the focus points are questionable.

Chapter 6

Evaluation and Conclusion

In this chapter, the results will be dissected to better understand what they mean and give answers to the research questions. OptiNet's performance will be discussed and its usability will be evaluated. Is (2+1)D convolution superior to 3D convolution? How well did OptiNet actually perform? These questions and more will be reflected upon. From the pondering of OptiNet's performance comes a final paragraph on the project's contributions to the field and what can be done in the future.

6.1 Evaluation

In light of how related works has performed on the Duke dataset, it seems OptiNet has performed quite well. RetiNet [Apostolopoulos et al., 2016] reported getting an AUC of 99.7% using the Duke dataset, while Farsiu et al. [2014] obtained an AUC of 99.17%. Venhuizen et al. [2015] obtained a slightly lower AUC of 98.4%. It would be the most fair to compare OptiNet's performance on shuffled partitions on the Duke dataset in table 5.4. OptiNet obtains an AUC of $\sim$ 99.7%, identical to RetiNet. It seems OptiNet performs at a state-of-the-art level, though it should be said that testing procedures are not identical. Furthermore, OptiNet proves it self to perform even better on the St. Olav's dataset with a mean AUC of $\sim$ 99.9% using (2+1)D convolution and an AUC of $\sim$ 1.00% using 3D convolution.

The models are quite stable across random factors with standard deviation of AUC ranging from 0.05% to 0.3%. It can be concluded that the project has accomplished both of its goals. The models perform within the range of related works, far beyond random. Furthermore, it accomplishes this without any pre-training step and proves it self on two separate datasets, showing that this architecture is able to complete its task on a small dataset and limited GPU capacity.

OptiNet does show high level performance, however, the AUC values are not vastly superior to previous works. In regards to research question 1, the experiments show no evidence that the tested 3D convolutions are superior to 2D convolution. However, the problem setting is already handled at a very high level, which leaves little room for improvement. There are also other factors that can affect the results, such as other architectural choices and hyper-parameters. Nonetheless, through the exploration of 3D convolutions, a highly capable system was developed that can indeed handle the problem setting.

The results on (2+1)D convolution versus 3D convolution are divided. Table 5.2 and table 5.4 show that (2+1)D convolution has a clear advantage on the Duke dataset. On shuffled partitions there is an AUC difference of 0.012, which in the scope of the literature is quite significant. However, it should also be noted that 3D convolution performed better on the St. Olav's dataset as indicated in table 5.3. The AUC difference is much smaller at 0.00074, but still outside the range of one standard deviation, which should not be dismissed. No clear conclusion can be made on which convolutional type is superior. It is likely that other architectural choices had greater significance for the performance.

Table 5.2 under "OptiNet with (2+1)D convolution" shows a precision of ~ 0.98, which indicates when the network is diagnosing a patient to have AMD, OptiNet will get it right $\sim 98\%$ of the time. This is of interest because it urges a caution when using the network. If OptiNet is used blindly it would have a 2% chance of giving the wrong diagnosis. This can be costly for the hospital because resources can be spent that does not help the patient. Furthermore, recall indicates that of all the patients with AMD, OptiNet will detect $\sim 98\%$ of them. With collaboration and critical use of OptiNet, this gap can be further reduced. The visualization closes the gap between the health professional and the model by giving an idea of how OptiNet made its decision. A doctor do not need to blindly accept or disregard the assessment, but rather be convinced by a correct assessment or disregards an erroneous one.

6.1.1 Evaluation of Visualization

CNN Fixations were calculated on the predictions made by OptiNet using (2+1)D convolution evaluating the St. Olav's dataset. Dr. Arnt-Ole Tvenning, responsible for the creation of the dataset, evaluated all results from the validation partition.

He found that OptiNet focused predominantly on the RPE as well as the retina. Figure 5.4 shows an example where the RPE is of interest. A surprising amount of focus is put on the retina right above the RPE rather than drusen, which doctors would typically focus on. OptiNet also focused on another less common area, the inner part of the retina below the RPE. Dr. Tvennig explains that in later years, studies have found that a secondary thickening in this layer happens when the upper part of the retina is thinning, which is an indicator of AMD. Furthermore, the model detects the thickening between the RPE and the top of the retina. This a sign sub-retinal deposition of drusen, something that has been focused on in later years. It is often difficult to detect this symptom in early stages of AMD, which makes the focus on this symptom exciting. OptiNet also detects fluid build up as a sign for some types of AMD, as well as the thinning of the RPE.

There was an unusual lack of focus on cysts in the retina and RPE. This might be because it is not a symptom in all cases of AMD, however, it could also be a lack of understanding from OptiNet. There was three cases of AMD that OptiNet got right, but seemingly on the wrong basis. An example of this is shown in figure 5.3. Though, some reasons can be found in these cases, they were hard to justify. This is an important insight into the true performance of the system and how it might perform in practice. Analysis on individual slices in the OCT scan was somewhat challenging due to the nature of the 3D localization map. The Gaussian blur is created in 3D and then sliced. Sometimes a warm area would start showing a few slices before the actual area of interest showed it self.

When it comes to healthy cases, the model mostly looked at the RPE and the shape of the upper part of the retina, for instance the dip of the fovea. Figure 5.2 shows an example where the shape of the RPE was the most important. It should be noted, however, the model can not communicate the absence of symptoms using CNN fixations. These are the existing signs of healthy cases, but there could also be other signs OptiNet can not show with this visualization.

Overall, the model give the impression that it has the right focus most of the time. Dr. Tvenning's assessment supports the achievement of goal 1, the network is indeed capable for learning symptoms of AMD at a level that could be usable in practice.

6.2 Discussion

The accuracy found in table 5.4 under "OptiNet with (2+1)D Convolution" indicates that OptiNet predicts correctly 97% of the time when the output is thresholded on 0.5. Comparing this to the AUC value, one can argue that if OptiNet would ever be used in practice, it might be more productive to return the output probability of AMD. This would provide the additional information of uncertainty. For instance, the case displayed in figure 5.3 has an output probability of $\sim 80\%$ while most AMD cases have output probabilities closer to 99%. This gives a good lesson in being sceptical of the results. Even though OptiNet performs well, signs of weakness was revealed through visualization. On the positive side, the tools used to make this discovery shows the usefulness of visualization together with deep learning methods. A system displaying the output probability as well as visualization can ease the work of health professionals and reduce uncertainty, making it a more viable tool to use in practice.

Though no clear conclusion was made on research question 2, there are certainly ways forward to get a more clear answer. Tran et al. claims that (2+1)D convolution performs better on deeper networks. With more GPU capacity, one can try a deeper version of OptiNet in an attempt to improve performance. Furthermore, one can rise the complexity of the problem setting by introducing more eye-related illnesses. One hypothesis is that the problem setting was too simple to see the advantages of (2+1)D convolution. Comparing table 5.2 and table 5.3, one can argue that the Duke dataset was more challenging as both architectural variations performed better on the St. Olav's dataset. The fact that (2+1)D convolution performs better on the Duke dataset could be a sign that (2+1)D convolution is better at higher complexity. On the other hand, figure 5.1 shows how similar most of the ROC curves are for these models. Further experimentation is needed to come to a conclusion.

Figure 5.3 and a few other cases had visualizations that could not be confidently justified by Dr. Tvenning. There could be a few possibilities in this scenario: OptiNet is using an unknown symptom for AMD, the visualization technique is failing, or OptiNet is making a correct assessment based on the wrong areas of the scan. Dr. Tvenning and I think the last scenario seem the most likely. Our reasoning is that visualizations did perform well overall, and the focus points in these odd cases were placed mostly above the retina. We also noticed that these images have more noise than usual. It seems the most likely that OptiNet got confused by the unusual noise in the image. Though, this lessens the confidence in OptiNet, it also displays the robustness of a complete system with visualization.

A final consideration for OptiNet is how well it would actually perform when used in practice. In reality, subjects are not just healthy or have AMD. Patients can have a wide range of conditions and some may be similar to AMD, for example, Diabetic Macular Edema. It is likely OptiNet would have a higher number of false positives when used in a real environment. For this reason, the researchers at St. Olav's University Hospital has been advised to also look for similar conditions to add to their dataset, so that a future approach can be more robust against other conditions.

6.3 Contributions

The thesis have the following contributions:

- Like many of the related works, OptiNet shows that deep learning is a viable approach in detection of AMD.
- Testing of (2+1)D convolution on OCT scans with promising results.
- A 3D CNN architecture without any pre-training, trained on one GPU that can perform at a state-of-the-art level.
- The use of CNN fixations to display symptoms from a medical classification CNN.
- Implementation and use of CNN fixations for 3D convolution and (2+1)D convolution.
- The introduction and use of a new dataset from St. Olav's University Hospital.

6.4 Future Work

Going forward there are three aspects that will be of the highest interest: Improvement of OptiNet, improvement of visualization and improvement of the data. One way that might improve OptiNet's performance would be depth. With more computer resources a deeper network can be tested. As mentioned in the discussion, (2+1)D convolution might perform better with a deeper network. On top of this, one can add the skip connections used in Tran et al. [2017] to uphold performance once the network gets deeper. With more computer resources, bigger mini-batches could potentially improve training as well. Introducing some

pre-processing such as noise reduction might aid in the cases where OptiNet makes the right prediction, but based on strange focus points. A final suggestion from Prof. Masashi Sugiyama is to go forward with a semi-supervised method where only a small part of the dataset needs labeling. This method allows for more training without the need of doctors manually labeling new cases in the dataset.

As mentioned in section 6.1.1, CNN fixations can point at existing signs, but not at the absence of symptoms. It is not clear to me how this can be done, however, a technique capable of doing so would bring valuable insight for the reasoning of a model. One part of CNN fixations that can potentially be improved is the localization map. One could attempt to further adjust the sigma parameter in the 3D Gaussian blur. This parameter affects how far the "blobs" of interest stretch over the map. One could also attempt to perform 2D Gaussian blur for each slice, or another technique to display the density of points.

A bigger dataset with more variation can better prepare OptiNet for real-life use. One of the best ways to improve a deep learning method is to give it more data. The dataset developed at St. Olav's University Hospital could eventually become a new benchmark in the field. The dataset can grow significantly larger once permissions are granted. Benchmarking allows for better comparison between approaches which is advantageous for the field as a whole. Dr. Tvenning has also discussed testing against human professionals. Comparison to human performance is among the best indicators on how well an automated system could do in practice. A dataset with human performance statistics and specific guidelines could be a great benchmark for the field going forward.

Of the three aspects highlighted I find improvement of OptiNet and the improvement of data the most interesting. I suggest to expand the problem setting by introducing more illnesses to the dataset and further build on the OptiNet architecture to handle the new problem setting. By doing this, one moves closer to a highly capable system that can be used in practice.

6.5 Summary

OptiNet performed well with an AUC of $\sim 99.7\%$ on the shuffled Duke dataset. RetiNet [Apostolopoulos et al., 2016] reported getting an AUC of 99.7% using the Duke dataset, while Farsiu et al. [2014] obtained an AUC of 99.17%. Venhuizen et al. [2015] obtained a slightly lower AUC of 98.4%. This indicates that OptiNet performs at a state-of-the-art level, but it is important to note that the experimental setup is not identical.

It is concluded that goal 1 and 2 is achieved based on the following reasoning: The models achieve results at the level of related works, far beyond random. Furthermore, OptiNet proved it self on two separate datasets with no pre-training and limited GPU resources.

In regards to research question 1, the results indicate that there is no advantage to using the tested variations of 3D convolution over 2D convolution. This is because OptiNet did no outperform models such as RetiNet. It should be mentioned, however, that many factors could affect this result, such as other architectural choices and hyper-parameters.

Research question 2 was harder to answer. As table 5.2 and table 5.3 show opposing results, it can not be clearly concluded which type of convolution is superior. It is thought that perhaps a deeper variation of OptiNet would give (2+1)D convolution the advantage, but that remains to be tested.

Dr. Tvenning found that OptiNet used many relevant symptoms when detection AMD, through evaluation of localization maps. However, he also found a few cases where OptiNet predicted correctly, but the focus points of the model could not be justified. It was assumed the model got confused by additional noise. Overall, the model gave the impression that it had the right focus most of the time.

The thesis made several contributions, some of the most notable was: Indication that deep learning is a viable approach for detection of AMD, the making of a 3D CNN architecture that perform at a state-of-the-art level on one GPU without pre-training, the use of CNN fixations to display symptoms of a medical classification CNN, and the introduction and use of a new dataset from St. Olav's University Hospital.

Going forward, there are three aspects that will be of the highest importance: Improvement of OptiNet, improvement of visualization and improvement of data. Of these three aspects I find improvement of OptiNet and improvement of data the most interesting. I suggest to expand the dataset to have more eye-related diseases and further build on the OptiNet architecture to handle the new problem setting.

Bibliography

Adhi, M. and Duker, J. S. (2013). Optical coherence tomography–current and future applications. *Current opinion in ophthalmology*, 24(3):213.

Akuffo, K. O., Nolan, J., Stack, J., Moran, R., Feeney, J., Kenny, R. A., Peto, T., Dooley, C., O'halloran, A. M., Cronin, H., et al. (2015). Prevalence of age-related macular degeneration in the republic of ireland. *British Journal of Ophthalmology*, 99(8):1037–1044.

Apostolopoulos, S., Ciller, C., Zanet, S. I. D., Wolf, S., and Sznitman, R. (2016). Retinet: Automatic AMD identification in OCT volumetric data. *CoRR*, abs/1610.03628.

Boureau, Y., Ponce, J., and LeCun, Y. (2010). A theoretical analysis of feature pooling in visual recognition. In *ICML 2010 - Proceedings, 27th International Conference on Machine Learning*, pages 111–118.

Carbonaro, G., Leanza, E., McCann, P., and Medda, F. (2018). Demographic decline, population aging, and modern financial approaches to urban policy. *International Regional Science Review*, 41(2):210–232.

Chiu, S. J., Izatt, J. A., O'Connell, R. V., Winter, K. P., Toth, C. A., and Farsiu, S. (2012). Validated automatic segmentation of amd pathology including drusen and geographic atrophy in sd-oct images. *Investigative ophthalmology & visual science*, 53(1):53–61.

Crabb, J. W., Miyagi, M., Gu, X., Shadrach, K., West, K. A., Sakaguchi, H., Kamei, M., Hasan, A., Yan, L., Rayborn, M. E., et al. (2002). Drusen proteome analysis: an approach to the etiology of age-related macular degeneration. *Proceedings of the National Academy of Sciences*, 99(23):14682–14687.

Cukras, C., Agrón, E., Klein, M. L., Ferris III, F. L., Chew, E. Y., Gensler, G., Wong, W. T., Group, A.-R. E. D. S. R., et al. (2010). Natural history of

drusenoid pigment epithelial detachment in age-related macular degeneration: Age-related eye disease study report no. 28. *Ophthalmology*, 117(3):489–499.

De Carlo, T. E., Romano, A., Waheed, N. K., and Duker, J. S. (2015). A review of optical coherence tomography angiography (octa). *International Journal of Retina and Vitreous*, 1(1):5.

Farsiu, S., Chiu, S. J., O'Connell, R. V., Folgar, F. A., Yuan, E., Izatt, J. A., Toth, C. A., Group, A.-R. E. D. S. . A. S. D. O. C. T. S., et al. (2014). Quantitative classification of eyes with and without intermediate age-related macular degeneration using optical coherence tomography. *Ophthalmology*, 121(1):162–172.

Feichtenhofer, C., Pinz, A., and Zisserman, A. (2016). Convolutional two-stream network fusion for video action recognition. *CoRR*, abs/1604.06573.

Fischer, H. (2013). Eye slide 3. [Online; accessed August 11, 2019], [Under creative commons licence: https://creativecommons.org/licenses/by/3.0].

Flaxman, S. R., Bourne, R. R., Resnikoff, S., Ackland, P., Braithwaite, T., Cicinelli, M. V., Das, A., Jonas, J. B., Keeffe, J., Kempen, J. H., et al. (2017). Global causes of blindness and distance vision impairment 1990–2020: a systematic review and meta-analysis. *The Lancet Global Health*, 5(12):e1221–e1234.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *CoRR*, abs/1512.03385.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Huang, D., Swanson, E. A., Lin, C. P., Schuman, J. S., Stinson, W. G., Chang, W., Hee, M. R., Flotte, T., Gregory, K., Puliafito, C. A., et al. (1991). Optical coherence tomography. *Science*, 254(5035):1178–1181.

Huang, G.-B., Zhu, Q.-Y., and Siew, C. (2006). Extreme learning machine: Theory and applications. *Neurocomputing*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

Jonas, J. B., Bourne, R. R., White, R. A., Flaxman, S. R., Keeffe, J., Leasher, J., Naidoo, K., Pesudovs, K., Price, H., Wong, T. Y., et al. (2014). Visual impairment and blindness due to macular diseases globally: a systematic review and meta-analysis. *American journal of ophthalmology*, 158(4):808–815.

Jonasson, F., Arnarsson, A., Eiríksdottir, G., Harris, T. B., Launer, L. J., Meuer, S. M., Klein, B. E., Klein, R., Gudnason, V., and Cotch, M. F. (2011). Prevalence of age-related macular degeneration in old persons: Age, gene/environment susceptibility reykjavik study. *Ophthalmology*, 118(5):825–830.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., and Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In *CVPR*.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kofod-Petersen, A. (2012). How to do a structured literature review in computer science. *Ver. 0.1. October*, 1.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Lecun, Y. (1988). A theoretical framework for back-propagation. In Touretzky, D., Hinton, G., and Sejnowski, T., editors, *Proceedings of the 1988 Connectionist Models Summer School, CMU, Pittsburg, PA*, pages 21–28. Morgan Kaufmann.

LeCun, Y., Bottou, L., Orr, G. B., and Müller, K.-R. (1998). Efficient backprop. In *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, pages 9–50, London, UK, UK. Springer-Verlag.

LeCun, Y., Haffner, P., Bottou, L., and Bengio, Y. (1999). Object recognition with gradient-based learning. In *Shape, Contour and Grouping in Computer Vision*, pages 319–, London, UK, UK. Springer-Verlag.

Lee, C. S., Baughman, D. M., and Lee, A. Y. (2017). Deep learning is effective for classifying normal versus age-related macular degeneration oct images. *Ophthalmology Retina*, 1(4):322–327.

Mopuri, K. R., Garg, U., and Venkatesh Babu, R. (2019). Cnn fixations: An unraveling approach to visualize the discriminative image regions. *IEEE Transactions on Image Processing*, 28(5):2116–2125.

Parmet, S., Lynm, C., and Glass, R. M. (2006). Age-related macular degeneration. *JAMA*, 295(20):2438–2438.

Pauleikhoff, D., Löffert, D., Spital, G., Radermacher, M., Dohrmann, J., Lommatzsch, A., and Bird, A. (2002). Pigment epithelial detachment in the elderly. *Graefe's Archive for Clinical and Experimental Ophthalmology*, 240(7):533–538.

Rosenfeld, P. J. (2016). Optical coherence tomography and the development of antiangiogenic therapies in neovascular age-related macular degeneration. *Investigative ophthalmology & visual science*, 57(9):OCT14–OCT26.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *Proceedings of the 20th International Conference on Artificial Neural Networks: Part III*, ICANN'10, pages 92–101, Berlin, Heidelberg. Springer-Verlag.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

Srinivasan, P. P., Kim, L. A., Mettu, P. S., Cousins, S. W., Comer, G. M., Izatt, J. A., and Farsiu, S. (2014). Fully automated detection of diabetic macular edema and dry age-related macular degeneration from optical coherence tomography images. *Biomedical optics express*, 5(10):3568–3577.

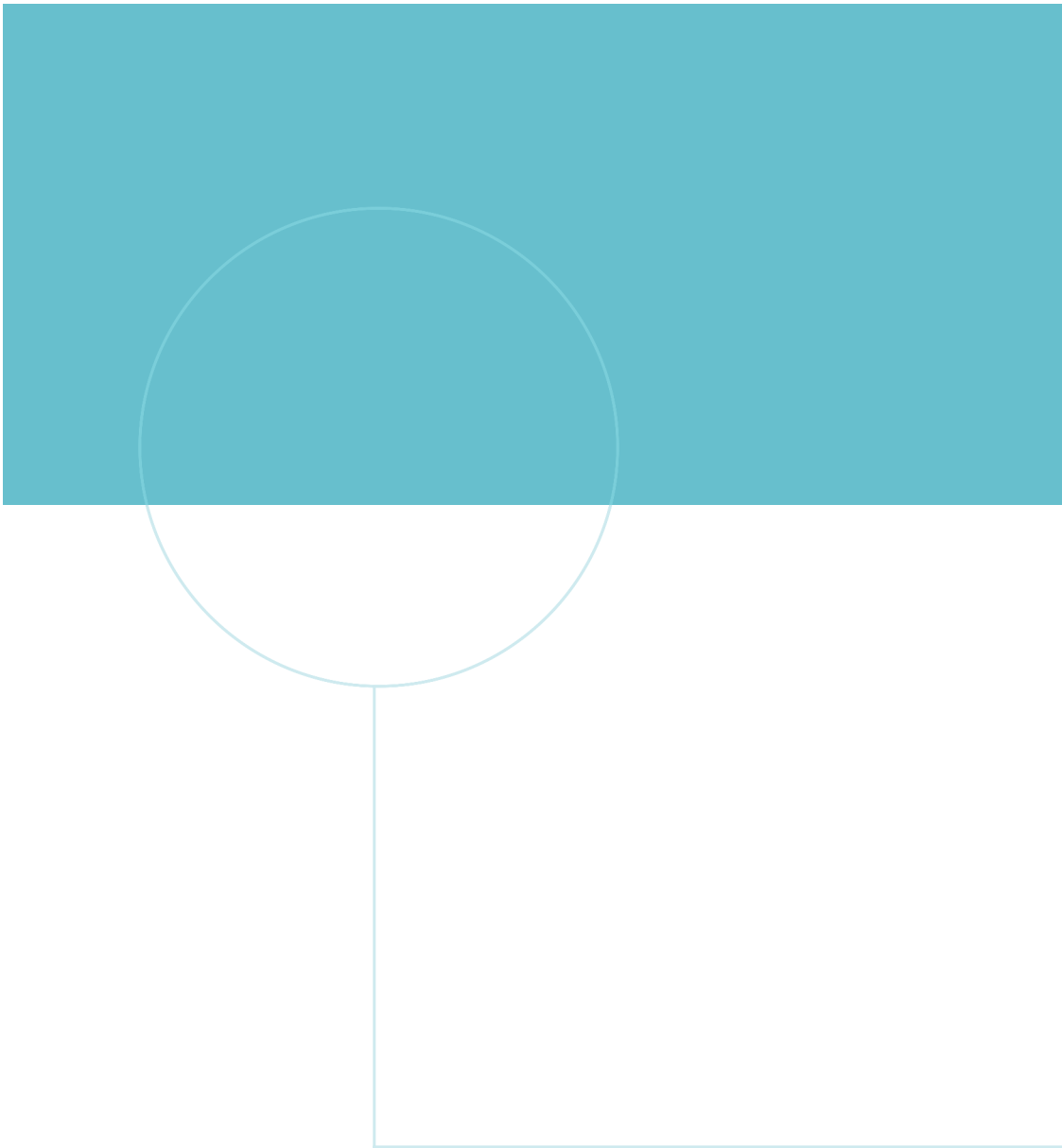
Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*.

Tran, D., Wang, H., Torresani, L., Ray, J., LeCun, Y., and Paluri, M. (2017). A closer look at spatiotemporal convolutions for action recognition. *CoRR*, abs/1711.11248.

Ulyanov, D., Vedaldi, A., and Lempitsky, V. (2016). Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*.

Venhuizen, F. G., van Ginneken, B., Bloemen, B., van Grinsven, M. J., Philipsen, R., Hoyng, C., Theelen, T., and Sánchez, C. I. (2015). Automated age-related macular degeneration classification in oct using unsupervised feature learning. In *Medical Imaging 2015: Computer-Aided Diagnosis*, volume 9414, page 94141I. International Society for Optics and Photonics.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer.



NTNU
Norwegian University of
Science and Technology