BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

AUTOMATIC DETECTION OF EYE RETINAL PATHOLOGIES

AUTOMATICKÁ DETEKCE PATOLOGIÍ SÍTNICE OKA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **VÍT TLUSTOŠ**
AUTOR PRÁCE

SUPERVISOR **prof. Ing. Dipl.-Ing. MARTIN DRAHANSKÝ, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Intelligent Systems (DITS) Academic year 2021/2022

Bachelor's Thesis Specification

25232

Student: **Tlustoš Vít**
Programme: Information Technology
Title: **Automatic Detection of Eye Retinal Pathologies**
Category: Image Processing

Assignment:

1. Study the literature on eye retinal pathologies, image processing and convolutional neural networks.
2. Propose a solution architecture that will detect exudates/drusens and hemorrhages.
3. Implement the proposed architecture from the previous point.
4. Test the solution. Summarise the results and discuss possible improvements.

Recommended literature:

- AKIL, Mohamed; ELLOUMI, Yaroub; KACHOURI, Rostom. Detection of retinal abnormalities in fundus image using CNN deep learning networks. In: *State of the Art in Neural Networks and their Applications*. Academic Press, 2021. p. 19-61.
- PACHADE, Samiksha, et al. Retinal Fundus Multi-Disease Image Dataset (RFMiD): A Dataset for Multi-Disease Detection Research. *Data*, 2021, 6.2: 14.
- SEMERÁD Lukáš a DRAHANSKÝ Martin. Retinal Vascular Characteristics. *Handbook of Vascular Biometrics*. Advances in Computer Vision and Pattern Recognition. Londýn: Springer International Publishing, 2019, s. 309-354. ISBN 978-3-030-27730-7.

Requirements for the first semester:

- Items 1 and 2.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Drahanský Martin, prof. Ing., Dipl.-Ing., Ph.D.**
Consultant: Mňuk Tomáš, MUDr., FNUSA
Head of Department: Hanáček Petr, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: May 11, 2022
Approval date: November 15, 2021

Abstract

This thesis aims to design and implement a system that automatically detects retinal eye pathologies. The retina is an essential part of the eye that, as the only organ in the body, contains light-sensitive cells that make a vision possible. For the treatment of eye disease to be successful, early detection and precise examination of its extent are crucial. The proposed system based on the supplied image automatically generates masks representing occurrences of individual pathologies. The result is then presented to the user. A convolutional neural network based on the U-Net architecture handles the evaluation. The network was trained on the Indian Diabetic Retinopathy Image Dataset (IDRiD), which contains 81 images of the retina and associated annotations. The system was evaluated using the AUC-PR score (area under the precision-recall curve). Segmentation of hard exudates, soft exudates, hemorrhages and microaneurysms achieved an AUC-PR score of 74%, 50%, 45% and 33%, respectively. This work proposes an innovative architecture that, if further developed, has the potential to be used by ophthalmologists for diagnosing and determining the extent of retinal disease.

Abstrakt

Cílem této práce je navrhnout a vyvinout systém, který automaticky odhalí vybrané patologie nacházející se na snímcích sítnice lidského oka. Sítnice jako jediný orgán v těle obsahuje světlocitlivé buňky potřebné k vidění. Proto, aby byla léčba onemocnění sítnice úspěšná je klíčové jeho včasné zachycení a přesné určení rozsahu. Navržený systém automaticky k dodanému snímku vygeneruje segmentační masky reprezentující výskyt jednotlivých patologií. Výsledek je poté prezentován uživateli. O samotné vyhodnocení se stará konvoluční neuronová síť jejímž základem je architektura U-Net. Síť byla natrénovaná na datasetu Indian Diabetic Retinopathy Image Dataset zkráceně IDRiD, který obsahuje 81 snímků sítnice a k nim příslušících anotací. Úspěšnost navrhovaného systému byla stanovena pomocí AUC-PR skóre (plocha pod precision-recall křivkou). Segmentace tvrdých exsudátů, měkkých exsudátů, hemoragií a mikroaneuryzmat dosáhla hodnot AUC-PR 74%, 50%, 45% a 33%, v daném pořadí. Tato práce přináší inovativní architekturu, která má v případě dalšího rozvoje potenciál být využita oftalmology pro diagnostiku a stanovení rozsahu onemocnění sítnice oka.

Keywords

bioinformatics, human eye, eye retina, image processing, segmentation, eye retinal pathologies, machine learning, neural networks

Klíčová slova

bioinformatika, lidské oko, sítnice oka, zpracování obrazu, segmentace, patologie sítnice oka, strojové učení, neuronové sítě

Reference

TLUSTOŠ, Vít. *Automatic Detection of Eye Retinal Pathologies.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor prof. Ing. Dipl.-Ing. Martin Drahanský, Ph.D.

Rozšířený abstrakt

Zrak hraje zásadní roli v kvalitě lidského života. Umožňuje nám nejen se orientovat v prostoru, ale i vnímat nebezpečí. Oko je párový orgán, bez něhož by vidění nebylo možné. Detekuje světlo ve viditelném spektru a informace o pozorované realitě následně posílá do mozku, který je interpretuje. I malé poškození zraku může výrazně ovlivnit kvalitu života, je tedy třeba zrak chránit tak, jak je to jen možné. Jednou z nejdůležitějších částí oka je sítnice. Sítnice jako jediný orgán v těle obsahuje světlocitlivé buňky potřebné k vidění. Sítnice je náchylná k řadě onemocnění, která mohou být charakterizována výskytem některé z následujících patologií. Častý je výskyt tvrdých a měkkých exsudátů, microaneurysmat a hemoragií. Tvrdé exsudáty jsou nažloutlá ložiska tvořené nahromaděným lipidovým a proteinovým materiálem. Měkké (vatovité) exsudáty jsou tvořeny nahromaděnými mrtvými nervovými buňkami. Vznikají většinou v důsledku nedostatečného přísunu krve (ischemie). Microaneurysmata jsou drobné výdutě na cévách, které mohou prasknout a vytvořit tak krevní výron. Hemoragie neboli krvácení mohou vznikat právě v důsledku prasklého microaneurysma. Proto, aby byla léčba onemocnění sítnice úspěšná je klíčové jeho včasné zachycení a přesné určení rozsahu.

Cílem této práce je navrhnout a vyvinout systém, který automaticky odhalí patologie nacházející se na snímcích sítnice lidského oka, který by mohl pomáhat lékařům rychle a efektivně dané patologie odhalovat.

Navržený a implementovaný systém automaticky k dodanému snímku sítnice vygeneruje segmentační masky reprezentující výskyt jednotlivých patologií. Výsledek je poté prezentován uživateli. Pro prezentaci výsledku je použita knihovna napari, která umožnuje uživateli se snímkem pohybovat, přibližovat si jej a vypínat jednotlivé masky. O samotné vyhodnocení se stará konvoluční neuronová síť. Architektura neuronové sítě se zakládá na modelu U-Net, který byl zvolen, protože v oblasti segmentace medicínských dat v mnoha případech dosahuje výborných výsledků. Navržená architektura byla oproti své standardní variantě rozšířena využitím residual bloků, které umožňují lepší optimalizaci hlubokých neuronvoých sítí a aplikací mechanismu pozornosti (attention), který výrazným způsobem urychluje učení tím, že relevantní části vstupů přiřadí vyšší váhu. Jako datový zdroj tohoto projektu, byl využit dataset Indian Diabetic Retinopathy Image Dataset (IDRiD), který disponje 81 barevnými snímky sítnice a k nim příslušících anotací. Aby data mohla být vyžita pro trénink a následnou evaluaci modelu, byla nejdříve předzpracována. Předzpracování spočívá ve zmenšení jejich rozměrů, zakódování masek jednotlivých patologií do podoby sparse encoded vektorů a následné vytvoření trénovacích a validačních datasetů aplikací metody cross validace. Daná data jsou dále rozdělena na malé překrýcajicí se části, které jsou použity pro samotný trénik/evaluaci. Výsledný prodkut byl natrénován na takto připravených datech pro segmentaci čtyř výše představených typů lézí – tvrdé exsudáty, měkké exsudáty, hemoragie a mikroaneuryzmata.

Jelikož bylo potřeba natrénovat relitivně velké množství různých modelů, je další postraním produktem této práce U-Net framework. U-Net framework je vysokoúrovňové API, které umožňuje uživateli vytvářet modely založené na architektuře U-Net jednoduše bez nutnosti programování. Uživatel specifikuje parametry modelu a ten mu je následně automaticky vygenerován.

Úspěšnost navrhovaného systému byla kvantifikována pomocí AUC-PR skóre (plocha pod precision-recall křivkou). Segmentace tvrdých exsudátů, měkkých exsudátů, hemoragií a mikroaneuryzmat dosáhla hodnot AUC-PR 74% , 50%, 45% a 33%, v daném pořadí. Podrobná analýza navrženého modelu včetně návrhu dalšího vylepšíní je důkladně diskutována v závěru práce. Tato práce přináší inovativní architekturu, která má v případě dalšího

rozvoje potenciál být využita oftalmology pro diagnostiku a stanovení rozsahu onemocnění sítnice oka.

Automatic Detection of Eye Retinal Pathologies

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of prof. Ing. Dipl.-Ing. Martin Drahanský Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Vít Tlustoš
May 16, 2022

Acknowledgements

I would like to express my gratitude for the guidance and support that I have received from my supervisor prof. Ing. Dipl.-Ing. Martin Drahanský Ph.D.

Contents

List of Figures

Chapter 1

Introduction

Sight plays a vital role in the quality of human life. It allows us not only to navigate in the surrounding environment, but also to perceive danger. The eye is a paired organ without which the vision would not be possible. It detects light in the visible spectrum and sends information about the observed reality to the brain. The brain, specifically the visual cortex, then creates an image of the observed reality. The most important part of the eye is the retina. As the only organ in the body, the retina contains light sensitive cells needed for vision. Thanks to its complexity, it is susceptible to a number of diseases. Even a minor vision impairment can affect the quality of life highly. Therefore we should do everything in our power to preserve a good vision. Early diagnosis and ongoing monitoring of the retinal disease are essential for its successful treatment. This is both a time-consuming and challenging process. Therefore, there is a need for an automated solution to help health care professionals with such tasks.

To solve any problem with the aid of a computer, we need an algorithm. The algorithm can be viewed as a sequence of steps that need to be performed to solve some problem. However, we do not know the exact sequence for complicated problems like autonomous driving or, in this case, the detection of retinal eye pathologies. Machine learning is an application of artificial intelligence featuring algorithms that can learn through experience, thus making it suitable for such tasks. One of the sub-areas of machine learning is deep learning. Deep learning algorithms are collectively referred to as neural networks due to their inspiration from the human brain. Nowadays, artificial neural networks, thanks to their high success rate, are widely used almost in any field of data processing.

To properly understand the problematic Chapter 2 presents needed theoretical knowledge. Anatomy of the eye is discussed in Chapter 2.2. A Particular focus is applied to the retina and its diseases. Needed technical knowledge is presented in Chapter 2.3 and Chapter 2.4. Chapter 2.3 introduces general concepts in artificial intelligence and machine learning. Neural networks and relevant state of the art techniques and architectures are discussed in Chapter 2.4. Chapter 3 introduces the proposed system and its implementation. The proposed model is thoroughly analysed in Chapter 4.

1.1 Notation

Following symbols and notation are used throughout this thesis.

Notation	Description
k	A scalar
$\mathbf{v}$	A vector
$\mathbf{M}$	A set
$\frac{dy}{dx}$	A derivative of y with respect x
$\nabla_{\mathbf{v}} y$	A gradient of y with respect to $\mathbf{v}$
x_i	An index i represents i-th element
$\mathbf{h}^l$	An output of n-th layer of the network
$\varphi(\mathbf{x})$	A mapping applied to the $\mathbf{x}$

Table 1.1: Notation and symbols

Chapter 2

State of the art

This chapter summarises the state of the art knowledge. The basics of optics needed for understanding a vision are introduced in Chapter 2.1. The anatomy of the eye is described in Chapter 2.2. Fundamentals of artificial intelligence and machine learning are presented in Chapter 2.3. Lastly, neural networks are discussed in depth in Chapter 2.4.

2.1 Optics

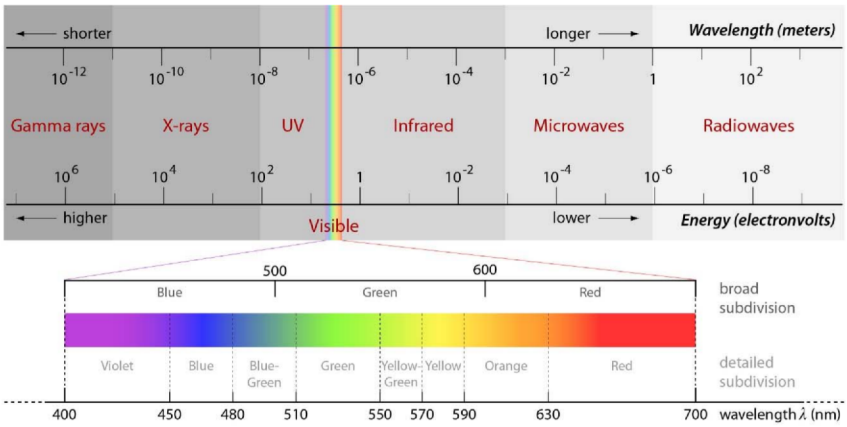

Figure 2.1: Electromagnetic spectrum, source [34]

Light is electromagnetic radiation that a wavelength can characterise. A sensory organ, the eye, perceives different wavelengths later interpreted by the brain as colours. The human eye can see only a small portion (400-700nm) of the whole electromagnetic spectrum, ranging from short gamma waves to long radio waves. [34]

2.2 Human eye

The human eye, described as both perfect and complex by Charles Darwin, is the first in a series of organs that makes vision possible. The eye converts the incident light into electrochemical impulses that are transmitted to the brain. When the light encounters

the eye, it first passes through a cornea layer. The cornea is a transparent layer where the light is refracted, allowing it to converge. After the cornea, the light reaches a pupil. The pupil is an opening in the iris that controls the amount of light let into the eye. It dilates or contracts based on the intensity, thereby controlling the amount of light. Once through the pupil, the light is processed by a lens. The lens is a bi-convex transparent disc that, with the aid of auxiliary muscles, changes its shape to focus the light rays onto the retina. The retina is the only place in the eye with light sensitive cells (rods and cones) that convert the light energy into electrochemical impulses. The electrochemical impulses are then propagated to the brain that interprets the observed reality. [36] [2]

2.2.1 Anatomy

This subchapter has been adopted from [2] and [36].

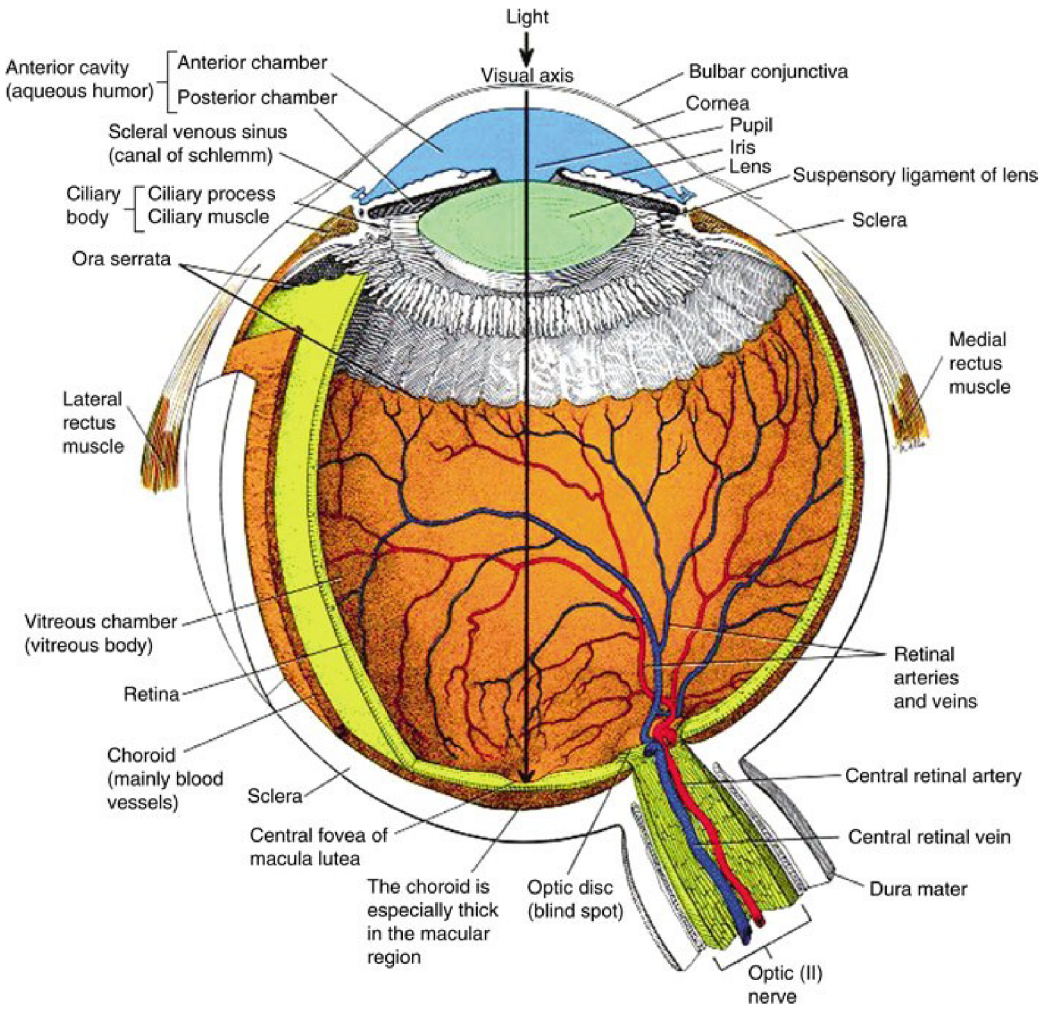

Figure 2.2: Eye anatomy, source: [20]

The eye can be divided into anterior and posterior segments. The anterior segment consists of the cornea, iris, pupil, conjunctiva, ciliary body, anterior chamber, aqueous humour, trabecular meshwork, and lens. The posterior segment comprises vitreous humour,

sclera, choroid, retina, macula and optic nerve. Structures involved in light processing are discussed below.

- **A sclera** is an opaque white outer layer surrounding most of the eyeball. Its primary responsibility is to protect and maintain the shape of the eye.
- **A cornea** is a transparent layer covering the front of the eyeball. It is the first layer encountered by the light. It protects the rest of the eye from microorganisms, injuries and artificial substances like dust. It is also responsible for focusing the light rays onto the retina (light-sensitive layer). It has a greater refractive index than the air. Therefore, the light is slowed down. Since there are no blood vessels, it receives its nourishment by tears and by the aqueous humour (fluid inside the anterior chamber).
- **An aqueous humour** occupies a space between the cornea and the lens. It is responsible for the nutrition of the cornea and the lens. Therefore it is constantly refiled.
- **An iris** is a thin circular structure responsible for controlling the size of the pupil. The colour of the iris is determined genetically.
- **A pupil** is a small opening in the iris that controls the amount of light that is let into the eye. In low-light environments, the pupil is dilated, allowing more light to enter the eye. In high-light environments, it is shrieked, thereby protecting the photo-receptive cells and providing a better image. It is controlled by the sphincter and dilator muscles of the iris.
- **A lens** is a biconvex elastic layer that can change its shape with the aid of ciliary muscles. Once the optimal amount of light reaches the lens, it is further focused, resulting in a more refined image. The process is called accommodation. During accommodation, the lens changes the degree of refraction, providing a better and more precise picture. When the eye focuses on a near object, the lens is forced into a bulging shape providing a greater optic power which brings the focal point closer.
- **A ciliary body** is a ring-shaped tissue surrounding the lens. It is composed of ciliary muscles and ciliary zonule. Ciliary muscles help the lens change its shape while the ciliary zonule holds it in position.
- **A vitreous humour** is a gel-like fluid mainly composed of water. Once the amount of light is reduced by the pupil and refracted by the cornea and the lens, it needs to pass through the vitreous humour. The vitreous humour is located in the cavity between the lens and the retina occupying about two-thirds of the eye volume. Its primary function is to maintain the shape of the eye.
- **A retina** is the only part of the eye containing photo-receptive cells. Since the topic of the work is the detection of retinal eye pathologies, the Chapter 2.2.2 is devoted to the retina.
- **An optic nerve** connects the eye to the brain. The information is transmitted in the form of electrochemical impulses.

2.2.2 Retina

This subchapter has been adopted from [2] and [36]. The retina is a thin circular disc with a diameter of 30–40mm [2] located at the back of the eyeball. It is the only organ in the human body containing light-sensitive cells needed for vision. It is responsible for transforming the incident light into signals that the brain can interpret. This is a rather complex process involving multiple neurons. Individual neuron types are discussed below.

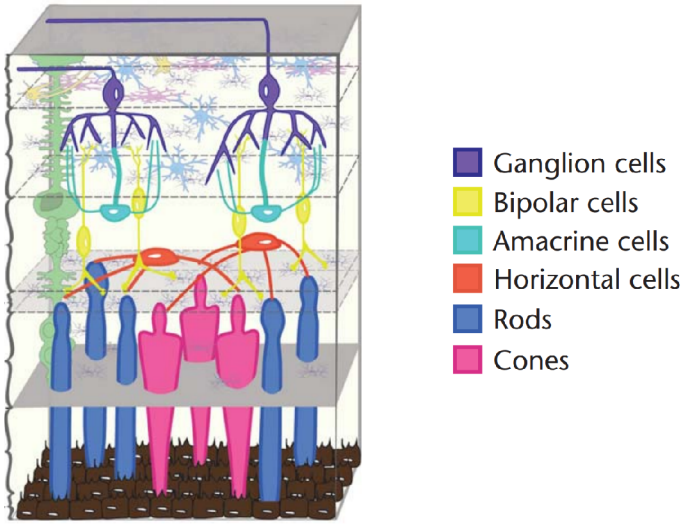

Figure 2.3: Retinal cells, source: [36]

- **Transmission neurons** are involved in the transmission of signals generated by photoreceptors (light sensitive cells) to the brain. The ganglion neurons are the innermost cells that connect to the optic nerve. Bipolar neurons work as a bridge between the ganglion and photoreceptor cells. They synapse with the photoreceptor on the one end and with the ganglion neuron on the other.

- **Photoreceptive neurons**, the rods and cones, are located in the outermost layer of the retina. They absorb photons of the incident light, and when the absorbed amount is significant enough, they initiate a signal transmitted to the brain. Rods are highly susceptible to light but only detect shades of grey. Many different rod cells connect via bipolar neurons to a single ganglion cell. Therefore, they provide the brain with information about the object's general shape or whether it is light or dark, but not with fine detail. Conversely, cones are susceptible to specific light frequencies providing the brain with information interpreted as colour. Unlike the rod cell, each cone cell has a personal ganglion cell. Therefore the brain can distinguish individual cells and provide a detailed image.

- **Lateral neurons**, horizontal and amacrine cells, link impulses originating from the photoreceptors. The horizontal cells interconnect multiple photoreceptors, thereby increasing the contrast. Amacrine cells form links between vertical pathway neurons. It is supposed that they contribute to better contrast as well.

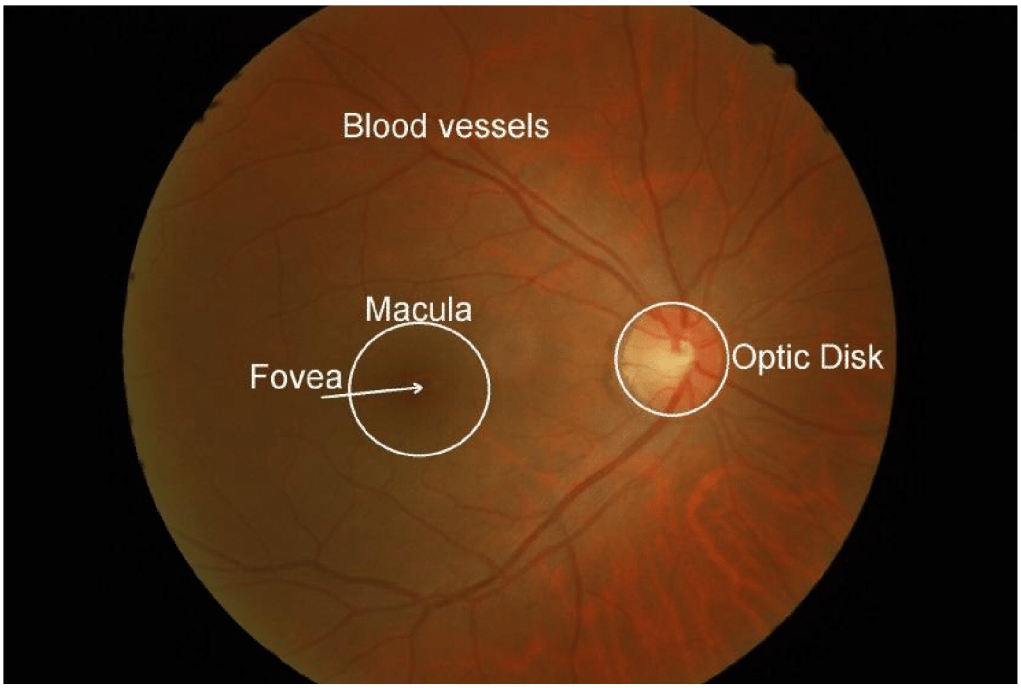

Figure 2.4: Retina regions, source: [6]

A method called fundoscopic examination may be used to assess the state of the retina. The ophthalmoscope, a device for acquiring the images, is placed in front of the patient eye. The inside of the eye is illuminated with sufficiently intense light. The rays forming the image of the retina re-emerge through the pupil, and the ophthalmoscope captures the image. [29] An image of a healthy retina is shown in Figure 2.4. There are three regions to recognise.

- **An optic disc** forms a passage for the transmission neurons. Since it does not contain photoreceptors, it is often called a blind spot.
- **A macula** is an area with a very high concentration of photoreceptors on which the light is being focused.
- **A fovea**, the dark part of the macula, is in the centre of the macula. Thanks to this, it provides the sharpest vision.

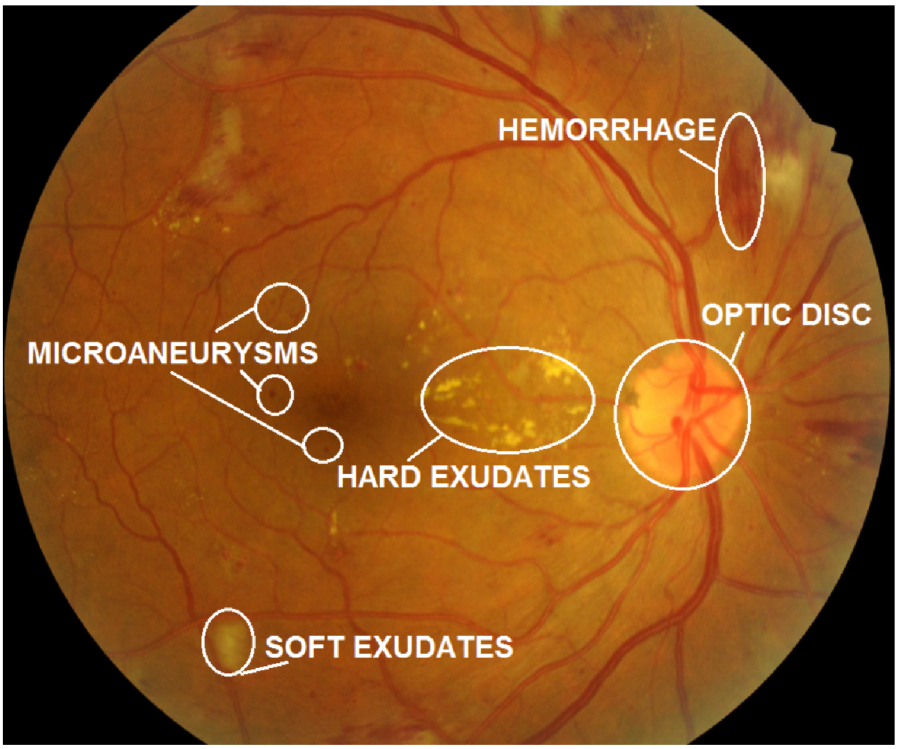

Figure 2.5: Retinal eye pathologies, source: [7]

The following text has been adopted from [33]. Due to its complexity, the retina may be subject to many diseases. Pathologies that are subject to this thesis are discussed below.

- **Hard exudates** are lesions, mainly formed as a result of leaking microaneurysms, composed of lipid and proteinaceous material.
- **Soft exudates**, also called cotton wool spots, are accumulations of dead nerve cells caused mainly by ischaemic damage (restricted blood supply).
- **Hemorrhages** arise as a result of bleeding capillaries.
- **Microaneurysms** are outpouchings of the capillary walls that may leak fluids into the retina, thus creating hemorrhages or hard exudates.

2.3 Artificial intelligence

This chapter discusses general concepts in artificial intelligence and presents needed theoretical knowledge. Artificial intelligence (AI) has become a vital part of computer science. To solve any problem with the aid of a computer, we need an algorithm. The algorithm can be viewed as a computational step sequence that transforms input to output. However, we do not know the exact transformation for complicated problems like autonomous driving or stock price prediction. In that case, we use machine learning algorithms.

2.3.1 General workflow

This subchapter has been adopted from [1]. The workflow of any data-science project can be divided into at least four stages.

- **Data collection**. Enough data has to be collected to train and evaluate the model. In some cases, existing datasets can be used. It is important to note that the usability of the final product is greatly affected by data quality.
- **Data preprocessing**. Once the data is collected, it needs to be preprocessed. Data preprocessing involves cleaning, verifying and transforming the data into a usable form.
- **Building datasets**. The preprocessed data needs to be divided into training, validation and testing subsets. The training subset is used for model optimisation, whereas the validation subset is used only for model evaluation. The testing subset is used for assessing multiple trained models on unseen data.
- **Model training and refinement**. Once the datasets are built, models need to be designed, trained and evaluated. Generally, multiple different models are trained, and the best or a combination of them (ensemble) is used for the final product.

2.3.2 Common tasks

When designing an algorithm, the first thing to be done is to determine the fundamental objective. From a top-level perspective, there are three groups of tasks - classification, segmentation and regression.

- **Classification**, as the name suggests, aims to classify (divide) data into categories. Its typical application might be written digits recognition.
- **Segmentation** is only a different type of classification. Instead of assigning a category (class) to the data entry as a whole, each part has its class. A typical segmentation application is an automatic map creation.
- **Regression**, based on hidden patterns in the data, predicts the value of some phenomenon. A typical example of regression is a weather forecast or stock price prediction.

2.3.3 Learning

The way the machine learning algorithm learns highly depends on the data and task.

- **The supervised learning** is a process when annotated data is fed to the model. The model periodically updates its parameters until a required performance is reached. Its typical application might be classifying written digits. The model would receive an image representing the number and a correct label, e.g. 5.
- **The reinforcement learning** is a trial and error based method. The model gains rewards or penalties based on how it performs. This technique is employed when a model needs to make a series of decisions in a potentially complex environment. A typical application might be training a self-driving car algorithm.
- **Unsupervised learning** also referred to as clustering, is a set of techniques that are used when dealing with unlabeled data. Its typical use is to categorise data into groups based on some similarity without the categories. One typical deployment of this technique is product recommendation.

Loss function

This subchapter has been adopted from [14], [32] and [13] .

$$L(\mathbf{w}, \mathbf{X}) \tag{2.1}$$

$$\mathbf{X} = \{(\mathbf{x_1}, \mathbf{y_1}), (\mathbf{x_2}, \mathbf{y_2}), \ldots, (\mathbf{x_N}, \mathbf{y_N}))\} \tag{2.2}$$

The loss function, also referred to as cost or objective function, is a mathematical function that quantifies how bad the model performs. Generally, it can be expressed as Equation 2.1. Given training data $\mathbf{X}$ we try to find model parameters $\mathbf{w}$ that minimise the loss function. A supervised learning data can view as a finite set of ordered pairs where $\mathbf{x}_i$ represents encoded data and $\mathbf{y}_i$ the ground truth. Popular loss functions are discussed below.

- **Binary-cross entropy loss**

$$L_i(\mathbf{x_i}, y_i) = -1 \times (y_i \times \log(p_i) + (1 - y_i) \times \log(1 - p_i)) \tag{2.3}$$

$$z = \varphi(\mathbf{x_i}) \tag{2.4}$$

$$p_i = \sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.5}$$

The binary-cross entropy, also called a sigmoid loss, is a widely used loss function for binary classification or binary segmentation problems. The data entry $\mathbf{x_i}$ can be labeled as either positive or negative class $y_i \in 0, 1$. The term $p_i \in < 0, 1 >$ represents predicted probability of the positive class. For numerical stability, every computation takes place in the logarithmic domain. The term z represents the learned underlying mapping of the network applied to the data-entry x_i, producing a value that is in the context of deep learning called „logit“.

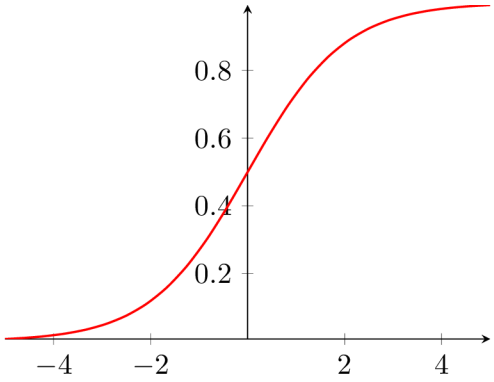

Figure 2.6: Sigmoid function

The „logit“ value ranges from $-\infty$ to $+\infty$. To convert the „logit“ to a valid probability range p_i a sigmoid function is used. The course of the sigmoid function is shown in

Figure 2.6. Intuitively, for small „logit“ values probability p_i reaches 0 while for large values p_i reaches 1.

- **Categorical cross entropy loss**

$$L_i(\mathbf{x_i}, \mathbf{y_i}) = -\log(p_i) \tag{2.6}$$

$$\mathbf{z} = \varphi(\mathbf{x_i}) \tag{2.7}$$

$$\log(p_i) = \log(softmax_i(\mathbf{z})) = \log(\frac{e^{z_i}}{\sum_{c=0}^{N} e^{z_c}}) = z_i - \log \sum_{c=0}^{N} e^{z_c} \tag{2.8}$$

Categorical cross-entropy, also called softmax loss, is a generalisation of the binary-cross entropy for multi-class classifications and segmentation problems. In the case of categorical cross-entropy, the network produces a vector of „logits“ $\mathbf{z}$. A softmax function is used to transform the vector to a valid probability range. It is worth noting that also the softmax function is usually calculated in the logarithmic domain providing several advantages - better numerical stability and reduced computational expensiveness.

- **Dice loss**

$$Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{2.9}$$

$$L_i(p_i, y_i) = 1 - \frac{2 \times p_i \times y_i}{p_i + y_i} \tag{2.10}$$

The dice loss is founded on a geometric basis. Given two sets **A** and **B**, the ratio of intersection and addition is measured. The function must be differentiable to be usable for training. Therefore its differentiable approximation is being used. The main advantage of using dice loss over cross-entropy is that it is usable even for highly imbalanced datasets. [22]

- **Taversky loss**

$$L_i(p_i, y_i) = 1 - \frac{p_{i,1} \times y_{i,1}}{p_{i,1} \times y_{i,1} + \alpha \times p_{i,1} \times y_{i,0} + \beta \times p_{i,0} \times y_{i,1}} \tag{2.11}$$

The Taversky loss is a generalization of the Dice loss. It introduces two hyper-parameters α and β allowing us to control the importance of false-positives and false-negatives. For hyper-parameters $\alpha = \beta = 0.5$ the function simplifies to a dice loss. Indexes i,c represent i-th data entry and predicted/ground truth class $c \in \{0, 1\}$. [22] [28]

Metrics

This chapter introduces metrics used for the model evaluation.

Confusion matrix

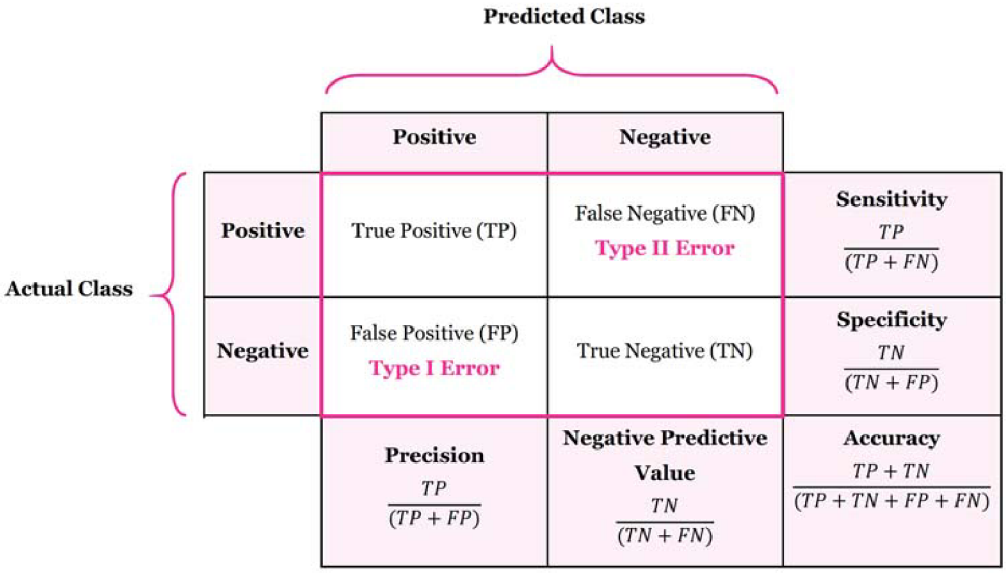

Figure 2.7: Confusion matrix, source: [10]

The confusion matrix, also called the error matrix, is used for assessing the model performance. The prediction made by the model can fall into one of the following categories. If the model correctly predicted a positive/negative class, it is labelled as a true-positive/true-negative. If the model misclassified the positive (predicted negative class) / negative (predicted positive class) class, it is labelled as a false-negative/false-positive.

Metrics that are based on the confusion matrix are listed below.

- **The sensitivity** measures the ability of the model to classify positive class samples correctly.
- **The specificity** measures the ability of the model to classify negative class samples correctly.
- **The positive predictive value** measures the ratio of true positive predictions and all positive predictions.
- **The negative predictive value** measures the ratio of true negative predictions and all negative predictions.

Jacquard coefficient

$$IoU(A,B) = \frac{|A \cap B|}{|A \cap B|} \tag{2.12}$$

The Jacquard coefficient, also called the Intersection over Union metrics, is formed on the same basis as the Dice coefficient (2.9). Given two sets A and B, the ratio of intersection and union is measured. A value close to one/zero indicates that the model performs well/bad on the data. [22]

Model training

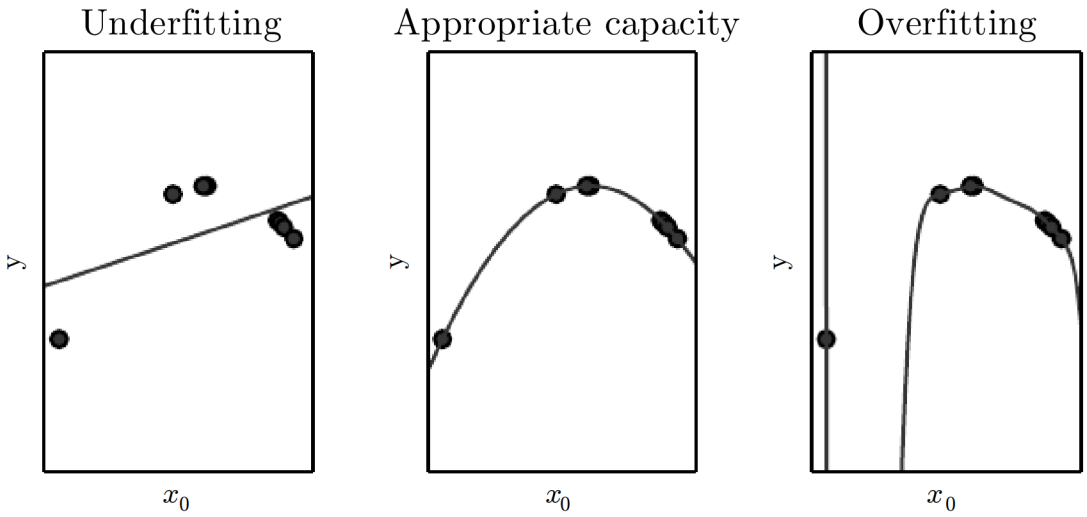

Figure 2.8: Underfitting/overfitting, source: [14]

When training a model, one thing to keep in mind is to monitor whether the model is underfitting or overfitting 2.8. This phenomenon is largely influenced by the model's capacity (number of parameters). For a model to perform well on real-world data, its capacity should be right. If the capacity is too big, the model overfits (it will respond well to the training data, but the performance on real-world data will suffer). On the other hand, the model will underfit if the capacity is too low (it will not be able to recognise enough features leading to poor performance on both training and real-world data). [14]

Data augmentation

Data augmentation is a set of techniques used for reducing overfitting. The overfitting is reduced by applying random transformations to the training data. Some of the important techniques are discussed below. This subchapter has been adopted from [24] and [19].

- **Scaling**

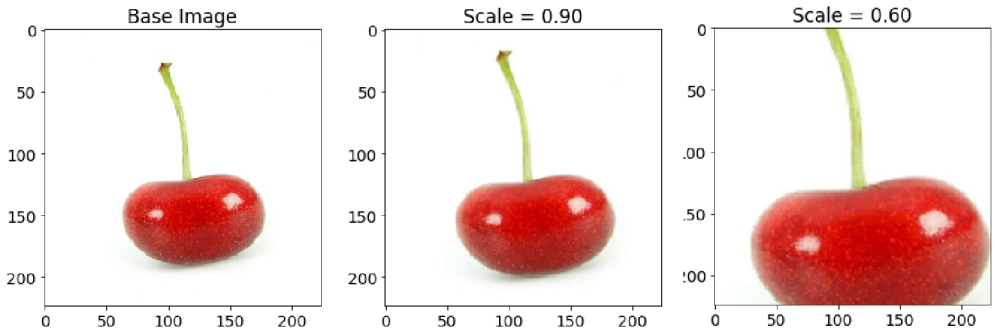

Figure 2.9: Scale, source: [24]

The same object may be captured in the image at various scales depending on the distance to the camera. Therefore, the model should provide a uniform prediction even for differently scaled objects. Scaling introduces multiple differently scaled representations of the same image to the dataset.

- **Translation**

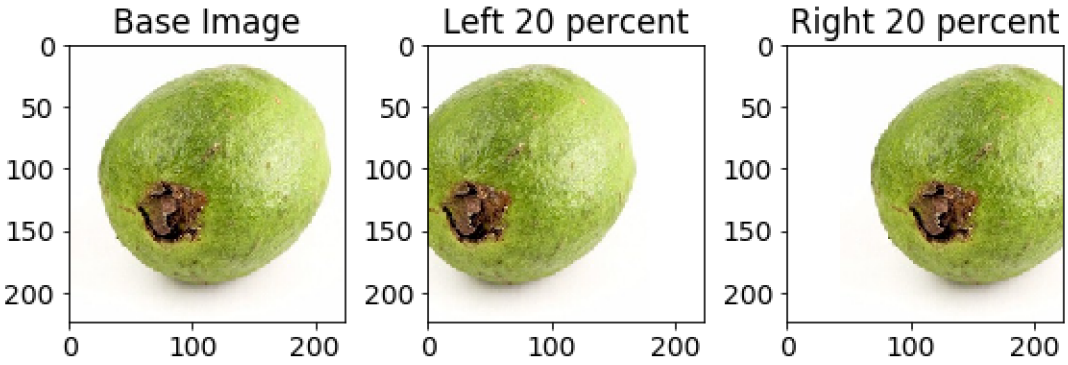

Figure 2.10: Translation, source: [24]

The object may appear anywhere in the image, depending on how the picture is taken. Therefore, the model needs to be invariant against the object's position. Positional diversity can be increased by shifting the object within the image.

- **Rotation**

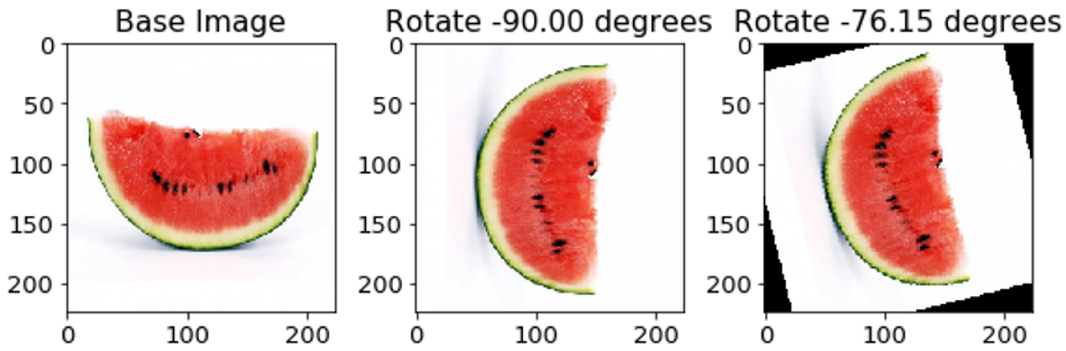

Figure 2.11: Rotation, source: [24]

A rotation of the image should not affect the model's prediction. A random rotation might be applied to the image to make the model more resistant to rotations. If the rotation angle is not a multiple of 90 degrees, then the rotated image does not fit perfectly with the input image.

- **Brightness**

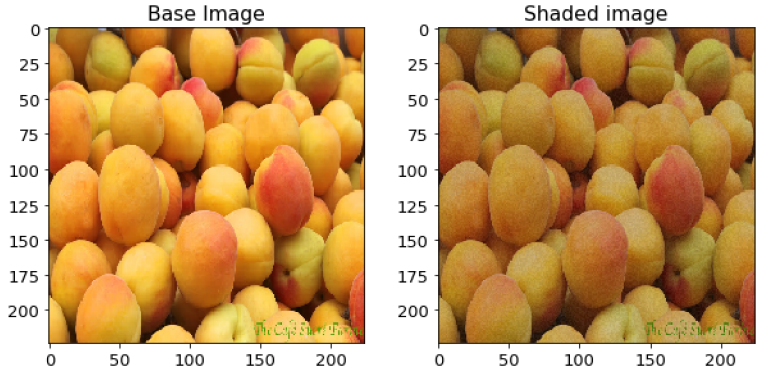

Figure 2.12: Brightness, source: [24]

The lighting conditions may significantly affect the model's performance. For this reason, brightness and contrast may be adjusted randomly to improve the lighting conditions dependency.

- **Noise**

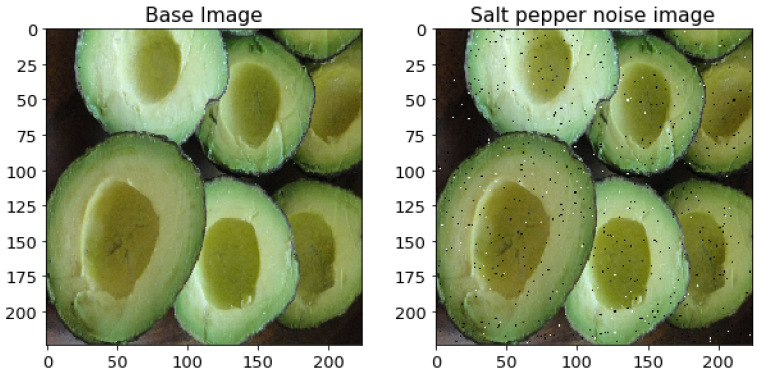

Figure 2.13: Noise, source: [24]

It has been observed that adding a small amount of noise to the input images improves network generalisation. [8] A salt pepper noise or Gaussian noise might be used.

- **Dropout**

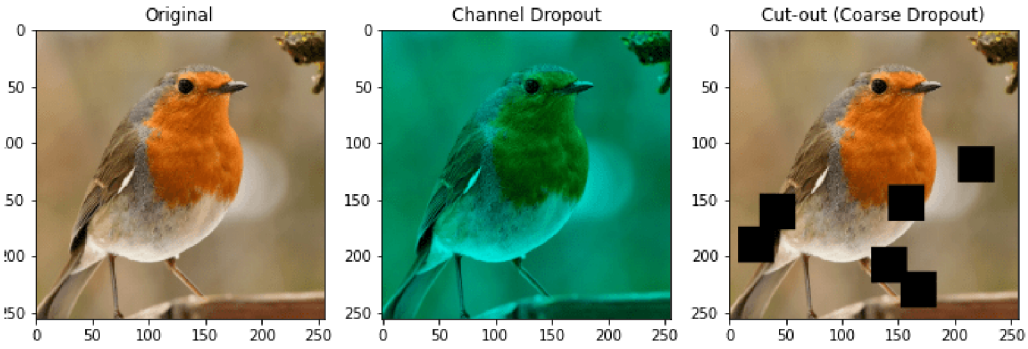

Figure 2.14: Dropout, source: [19]

Coarse and channel dropout are two techniques used for preventing the model from overfitting. The coarse dropout omits small parts of the image, whereas the channel dropout randomly drops some of the channels.

2.4 Deep learning

Deep learning is a set of techniques inspired by the human brain. Due to insufficient computational power, artificial neural networks have not become popular until recently. Nowadays, artificial neural networks are widely used almost in any field of data processing (visual data, natural language, ...).

2.4.1 Neural networks

A fundamental building block of any artificial neural network is a neuron. A real neuron inspires the function and structure of the artificial neuron but in no case replicates it. Like real neurons in the brain, artificial neurons are placed in many interconnected layers.

Neuron

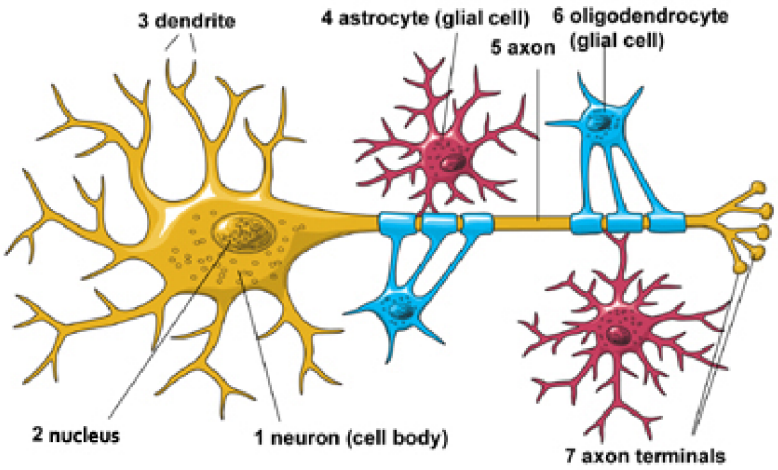

Figure 2.15: Real neuron, source: [12]

A real neuron has three basic parts: a cell body, an axon and dendrites. Both the axon and dendrites are protrusions of the neuron. Axon propagates the output signal, whereas dendrites are designated for receiving input signals. The place where an axon of one neuron connects to a dendrite of a different neuron is called a synapse. The cell body accumulates incoming signals, and when certain conditions are met, an outgoing signal is sent via the axon for other neurons to receive. [12]

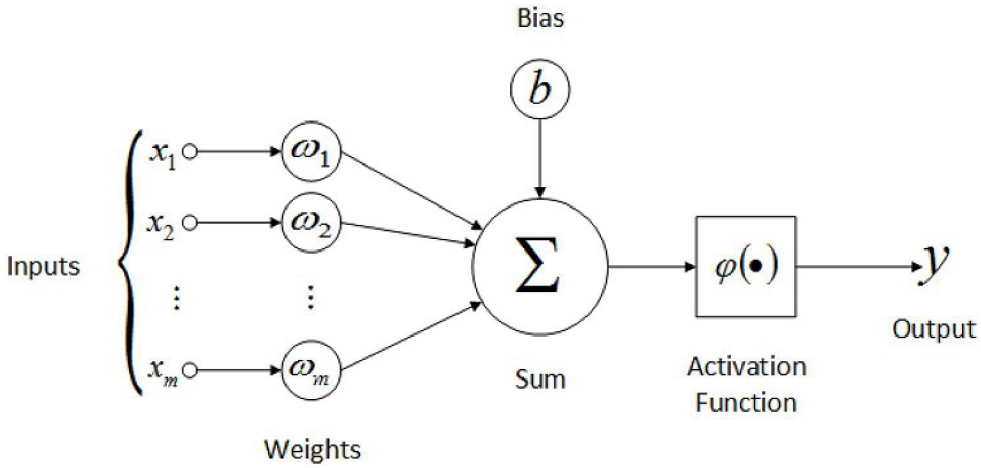

Figure 2.16: Artificial neuron, source: [11]

An artificial neuron is composed of three operations - an element-wise dot product (dendrites), a summation operator (cell body) and a mapping function (axon). First, the element-wise dot product of incoming signals $\mathbf{x}$ (outputs of predeceasing layer) with neuron weights $\mathbf{w}$ is calculated. Then all the multiplied elements are summed by the summation operator. Lastly, an output is generated by a mapping function φ that transforms summed signals into an output signal.

Activation function

Another crucial building block of any neural network is an activation function. Since both the dot product and the summation are linear operations, the neural network would not be able to separate linearly inseparable data. That would make the network unusable for complex problems like autonomous driving. The activation function introduces the non-linearity into the neural network allowing the network to recognise complex data patterns. This subchapter has been adopted from [18] and [4].

- **Rectified linear unit (ReLU)**

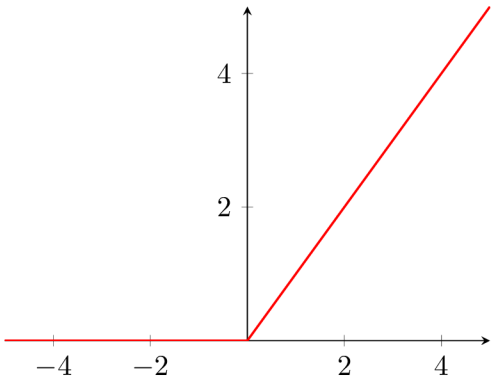

Figure 2.17: ReLU - function

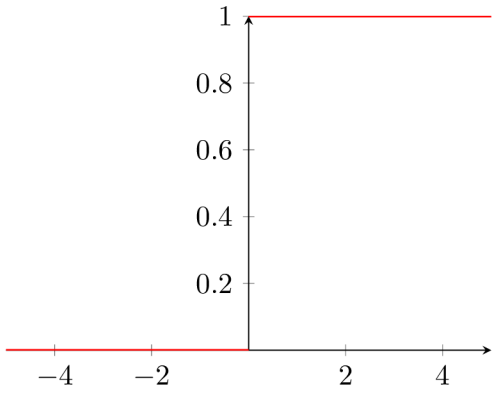

Figure 2.18: ReLU - first derivative

$$h(a) = max(0, a) \tag{2.13}$$

$$\nabla h(a) = \begin{cases} 1 & a \geq 0 \\ 0 & a < 0 \end{cases} \tag{2.14}$$

Rectified linear unit (ReLU) is probably the most commonly used activation function in hidden layers.

The ReLU is computationally inexpensive - both the forward (2.13) (maximum operation) and backward (constant) (2.14) passes are computationally non-demanding. Another advantage is that the function does not saturate for positive activation values thanks to its linear characteristics.

The most significant limitation faced by the function is a dying ReLU problem. The derivative of any negative activation value is zero making the neuron „dead“ (weights will never be updated).

- **Leaky Rectified linear unit (Leaky ReLU)**

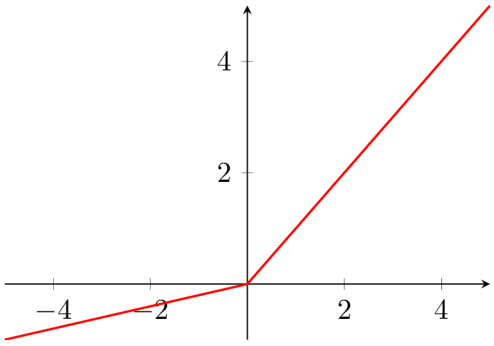

Figure 2.19: Leaky ReLU - function

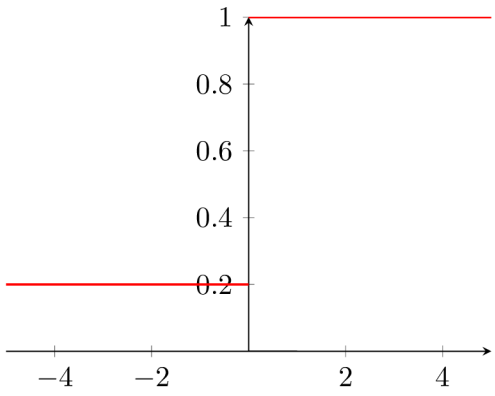

Figure 2.20: Leaky ReLU - first derivative

$$h(a) = max(\alpha \times a, a) \tag{2.15}$$

$$\nabla h(a) = \begin{cases} 1 & a \geq 0 \\ \alpha & a < 0 \end{cases} \tag{2.16}$$

The leaky rectified linear unit (Leaky ReLU) solves the dying ReLU problem. Other than that, the Leaky ReLU has the same advantages as the ReLU. The problem is fixed by mapping all negative activation values to non-zero values making the gradient non-zero. The constant α determines the slope of the activation function for negative values.

Pooling layer

Since we have limited computational resources available, there is a need to be efficient. Pooling is used for a spatial reduction of the input. It also helps the network become invariant against a precise position of certain features. A slight difference in the position of the feature might result in a completely different feature map. A common approach to address this issue is to use pooling. Pooling reduces the size of an input, thereby extending the neuron's receptive field (at the expense of fine detail). [14] [9]

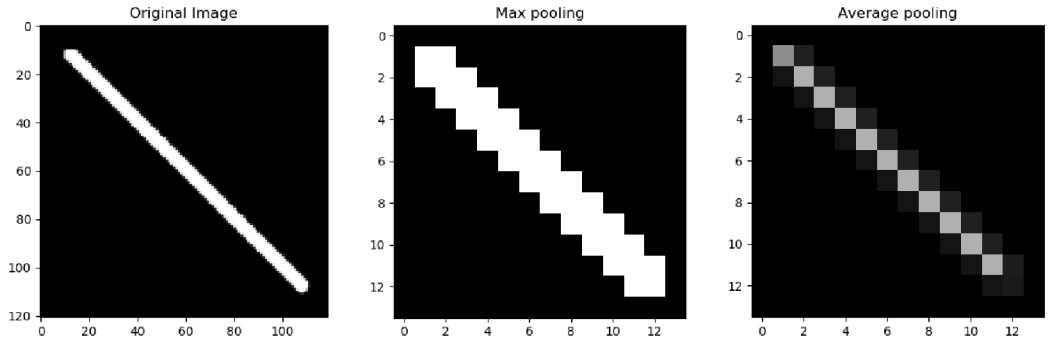

Figure 2.21: Pooling, source: [5]

$$\mathbf{h^l_{max}} = max_{i=0,\ldots,p;j=0,\ldots,q}(\mathbf{h^{l-1}_{(x+i,y+j)}}) \tag{2.17}$$

$$\mathbf{h^l_{avg}} = \frac{\sum_{i=0}^{p}\sum_{j=0}^{q}\mathbf{h}^{l-1}_{(x+i,y+j)}}{p+q+2} \tag{2.18}$$

Commonly used pooling functions are a maximum h^l_{max} and an average h^l_{avg} pooling functions. The pooling function shifts a fixed-size window over the input and computes a maximum or average value at each position. Figure 2.21 demonstrates the use of an average and maximum pooling on a straight line. While it cannot be explicitly said that one is better than the other, a few characteristics are obvious. The average pooling smooths out the feature map meaning sharp features may not be recognised. On the other hand, the maximum pooling recognises only the sharpest feature. [5]

2.4.2 Optimization

At the beginning of the training, a model is initialised with small randomly distributed weights. Then a process called optimisation starts. The weights are systematically adjusted to minimise a loss function (2.3.3).

Optimizers

This subchapter has been adopted from [14], [3] and [31].

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \times \nabla_{\mathbf{w}_i} \tag{2.19}$$

Optimisation of a neural network is an iterative process based on gradient calculation. The gradient $\nabla_{\mathbf{w}_i}$ represents the steepest ascend of the loss function with respect to model weights. The index i represents the current iteration. Once the gradient is computed, a small step (determined by the learning rate η) is performed in the negative gradient (steepest descent) direction, thereby minimising the loss function. In the context of deep learning, the process is called gradient descent. Performing the step is a non-trivial problem because the gradient space is usually non-convex and contains many local minimums and maximums. Therefore there is a need for smart optimisation algorithms, also called optimisers.

- **Learning rate**

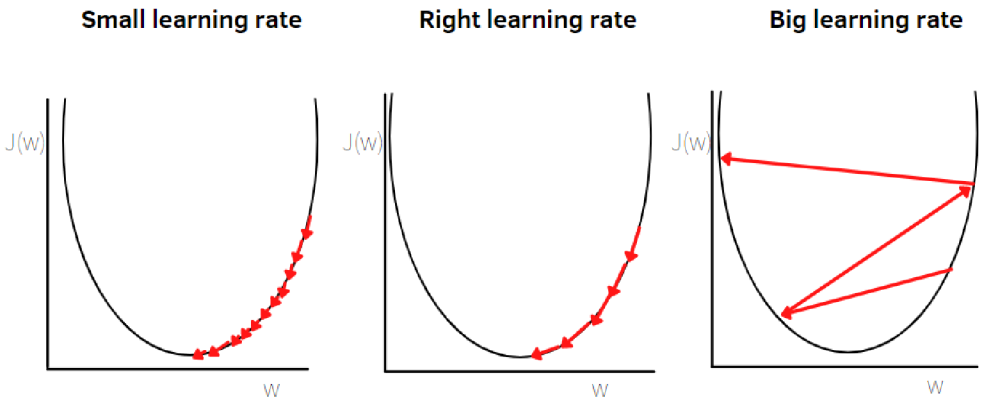

Figure 2.22: Learning rate, source: [3]

Choosing a correct learning rate is crucial for the model to converge. As shown in Figure 2.22 if the learning rate is too high, the loss function fluctuates. Conversely, the optimisation takes unreasonably long if the learning rate is too low.

- **Gradient descent**

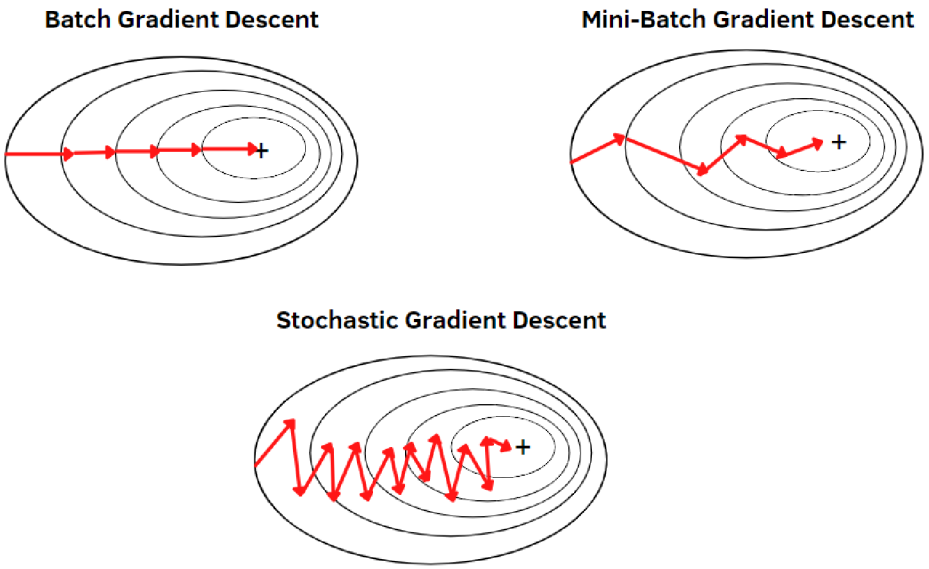

Figure 2.23: Batch vs. mini-batch vs. stochastic gradient, source: [3]

The next challenge comes when choosing when should the weights updates occur. There are essentially three methods - batch, mini-batch and stochastic gradient descent.

Batch gradient descent updates the weights once the gradient is computed for the whole dataset. This method produces the most accurate gradients of the three but is very computationally expensive. Therefore it might be impractical for large datasets.

Mini-batch gradient descent method introduces a hyper-parameter called *batch_size* that determines how many data-set entries are used for a gradient computation and weights update. As a result, weight updates are more frequent but not as accurate. However, it has been observed that the network converges similarly but in a fraction of the time.

Stochastic gradient descent method is a special case of the mini-batch gradient descent method when $batch\_size = 1$. It further reduces the computational expensiveness at the expense of accuracy. This approach might not apply to some problems because the loss function might fluctuate (due to the noisy gradients).

- **Momentum**

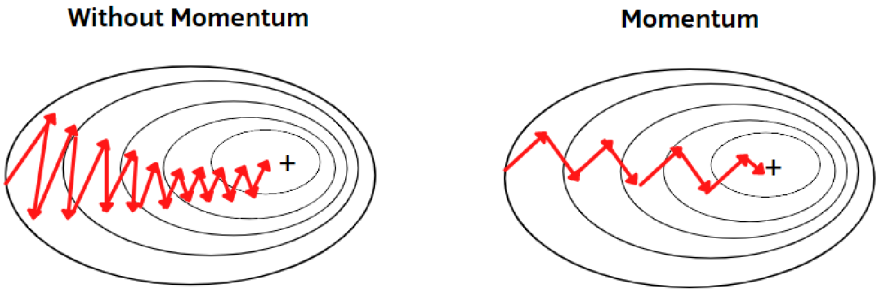

Figure 2.24: Momentum, source: [3]

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \times \nabla_{\mathbf{w}_i} + \beta \times (\mathbf{w}_i - \mathbf{w}_{i-1}) \tag{2.20}$$

A trick called gradient descent with momentum can be used to accelerate the training. The new weights are computed as a weighted sum of a gradient $\nabla_{\mathbf{w}_i}$ from the current position and the last step $\mathbf{w}_i - \mathbf{w}_{i-1}$. The hyper-parameter $\beta \in (0, 1)$ determines the importance of the last step.

- **Adaptive learning**

  The training can be further improved by adaptive training methods such as RMSProp or Adam. The learning rate is no longer global. Instead, it is computed for each weight individually.

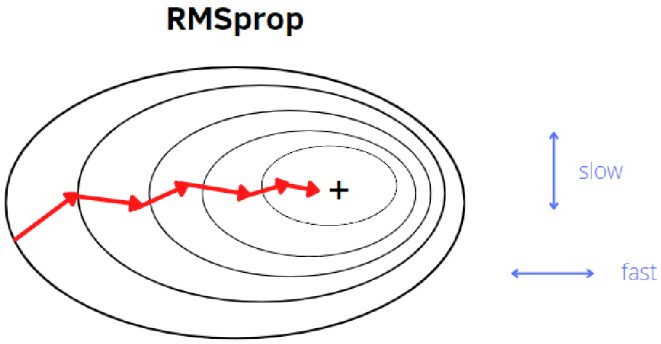

Figure 2.25: RMSProp, source: [3]

As shown in the Figure 2.25 by introducing different learning rates for individual weights, depicted as slow and fast, the training becomes smoother.

$$\mathbf{s}_{i+1} = \alpha \times \mathbf{s}_t + (1 - \alpha)\nabla^2_{\mathbf{w}_i} \tag{2.21}$$

$$\mathbf{w}_{i+1} = \mathbf{w}_i - \eta \times \frac{\nabla_{\mathbf{w}_i}}{\sqrt{\mathbf{s}_{i+1}} + \epsilon} \tag{2.22}$$

This can be achieved by introducing the exponential moving average s_{i+1} to the Equation 2.22. The exponential moving average is represented by the Equation 2.21. It can be viewed as a window that slides over past gradients and computes an „average“ of squared derivatives for each weight. It is desirable to slow down the learning in the direction where the derivatives are large. Therefore the gradient is divided by the square root of the exponential moving average. The ϵ in the denominator prevents a division by zero.

The holy grail of modern optimisers is the Adam algorithm. Adam uses the above described adaptive learning combined with the momentum (2.4.2).

Computation

Every deep learning framework represents a neural network as a computational graph. The nodes represent elementary mathematical operations, and edges connect the operations, creating more or less complex functions. Decomposing neural networks into computational graphs makes an automated computation and differentiation possible. However, a fundamental assumption for the automatic differentiation is the existence of the first derivative of each operation in the graph. The gradient computation is done by applying the rules of differentiation and the chain rule, which lays the foundations for the derivation of composite functions. [13] [30]

The calculation itself consists of two passes - a forward pass and a backward pass. The forward pass, calculated „from left to right“, evaluates the loss function for current model weights. The backwards pass, calculated „from right to left“, minimises the loss function by using the gradient descent technique (2.4.2).

2.4.3 Convolutional neural networks

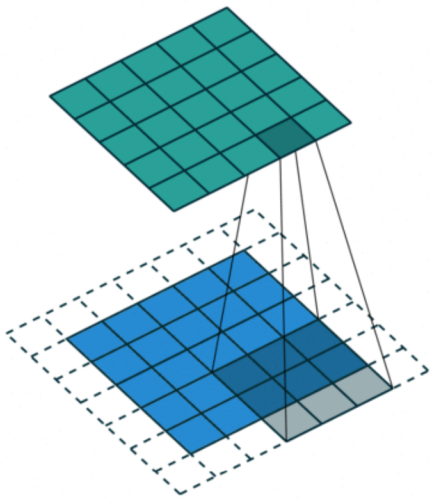

Figure 2.26: Convolution, source: [26]

Convolutional neural networks change the way neurons are connected. The output of every neuron is no longer directly connected to the input of every neuron in the following layer. Instead, a convolution operation is performed. Neurons called kernels or filters are small

matrices of weights. They might be viewed as windows that are being shifted over the input. An element-wise dot product is taken at each position creating activation maps. It has been observed that this type of computation dramatically improves network performance, especially in computer vision. During the training kernel, weights learn to represent more or less complex shapes. The number of filters is the first in a series of parameters that need to be experimented with. If the amount is not big enough, the network underfitts. Conversely, if the amount is too big, the network is prone to overfitting 2.8. A hyper-parameter kernel size determines the receptive field of the neuron. The larger the kernel size is, the bigger the receptive field (at the expense of computational complexity). [13] [14]

2.4.4 Popular architectures and techniques

This chapter introduces state of the art architectures and techniques.

Residual networks

Residual connections were firstly introduced in the paper [15]. Since then, the principle has been used by many engineers and scientists. This subchapter has been adopted from [15].

It has been observed that the model benefits from more layers (accuracy improves) until, at some point, the model starts to perform much worse - degrades. As described in the paper, such degradation is not caused by overfitting. As the depth of the model increases, the model becomes more difficult to train due to the vanishing/exploding gradient types of problems. Residual networks are based on the same hypothesis as regular neural networks. Let $g(\mathbf{h})$ be an underlying mapping to be learned by a part of the neural network. The input to the mapping is denoted as $\mathbf{h}$. It is assumed that multiple non-linear layers can asymptotically approximate the mapping, which also applies to the residual function $g(\mathbf{h}) - x$. While both functions should approximate the desired function, the training difficulty might differ in favour of the residual networks.

- **Vanilla network**

$$g_n(\mathbf{h}^{n-1}) = non\_linearity(linearity(\mathbf{h}^{n-1})) \tag{2.23}$$

$$\mathbf{h^n} = g_n(\mathbf{h}^{n-1}) \tag{2.24}$$

  The output of the n-th layer $\mathbf{h^n}$ of the vanilla network, a network that does not employ identity (skip) connections, solely depends on the mapping function g_n. The function g_n represents a transformation performed by the n-th layer of the network. The transformation comprises a linear (linearity) and a non-linear (non_linearity) component. The linear component involves weights and biases, whereas the non-linear component represents an activation function. Typical examples of the linear and the non-linear components are convolution and ReLU activation, respectively.

- **Residual network**

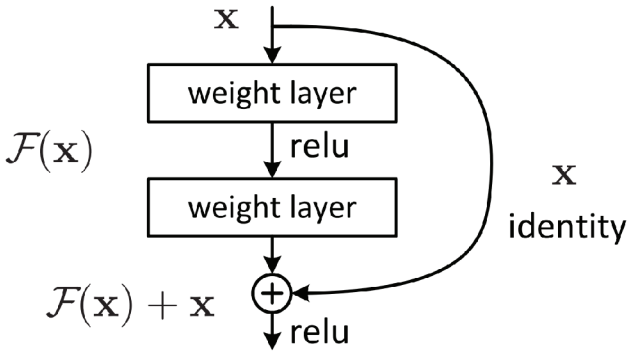

Figure 2.27: Residual block, source: [5]

$$\mathbf{h^n} = g_n(\mathbf{h}^{n-1}) + \mathbf{h}^{n-1} \tag{2.25}$$

A residual block introduces identity (skip) connections to the architecture. The output of the current layer $\mathbf{h^n}$ no longer depends only on the g_n, it also depends directly on the output of the predeceasing layer $\mathbf{h^{n-1}}$.

Dense networks

Densely connected networks improve the vanishing gradient types of problems. One key difference compared to the residual networks is the number of parameters. As described in the paper, residual networks suffer from a significant redundancy, meaning the parameters are not efficiently used. Densely connected networks reduce the number of parameters while achieving similar or better performance by introducing dense connections to the framework. This subchapter has been adopted from [17].

- **Dense block**

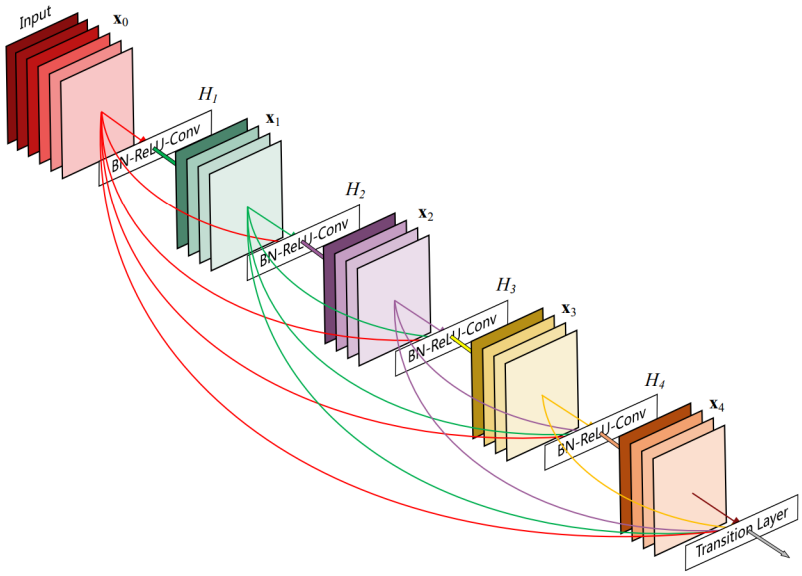

Figure 2.28: Dense block, source: [17]

A dense block is a series of convolutional layers where each layer is directly (densely) connected to all outputs of predeceasing layers. The information flow is even better compared to the residual networks. Another great benefit of the dense connections is collective knowledge. Later layers can use all the predeceasing outputs resulting in fewer redundant feature maps.

$$\mathbf{h}^n = g_n([\mathbf{h}^0, \mathbf{h}^1, \ldots, \mathbf{h}^{n-1}]) \tag{2.26}$$

The output of the n-th layer $\mathbf{h}^n$ can be calculated using the Equation 2.26. Term $[\mathbf{h}^0, \mathbf{h}^1, \ldots, \mathbf{h}^{n-1}]$ represents a concatenation of feature maps that are produced by predeceasing layers.

- **Transition block**

  The transition block is placed directly after the dense block. Its purpose is to reduce the spatial resolution and number of filters.

U-Net

The U-Net architecture was firstly introduced in the paper [27]. The U-Net is widely used for the segmentation of medical data. This subchapter has been adopted from [27].

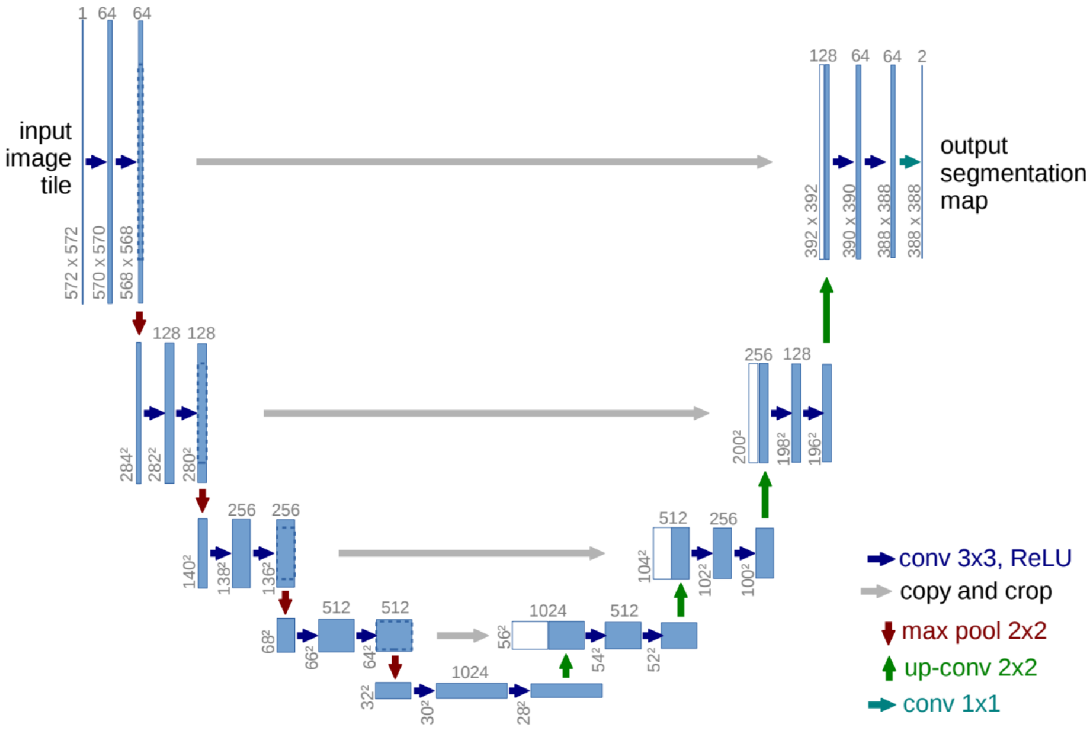

Figure 2.29: U-Net architecture, source [27]

The architecture is composed of two symmetric paths - counteractive and expansive paths. The counteractive path (encoder) is used to build a feature-rich representation. The other expansive path (decoder) makes precise localisation possible. The biggest drawback of using encoder-decoder architectures is losing spatial information at later layers. The U-Net skip connections are introduced to preserve more spatial information between adjacent levels of the encoder-decoder.

$$\mathbf{h}^{n}_{expansive} = [\mathbf{h}^{n}_{contractive}, \mathbf{h}^{n-1}_{expansive}] \tag{2.27}$$

A skip connection connects each level of the counteractive path to the adjacent level of the expansive path, thereby allowing more precise localisation. Input $\mathbf{h}^{n}_{expansive}$ to the first convolution of the n-th layer of the expansive path can be calculated using Equation 2.27. The $[\mathbf{h}^{n}_{contractive}, \mathbf{h}^{n-1}_{expansive}]$ represents concatenation of the contracting and expansive feature maps. Note last skip connection is applied at the depth $n = 1$ because $\mathbf{h}^{-1}_{expansive}$ does not exist.

Attention

Loosely defined attention is a mechanism that lets the network focus on things we care about. The attention speeds up the training process by highlighting relevant features. The attention mechanisms can be divided into spatial and channel attention. This subchapter has been adopted from [23] and [35].

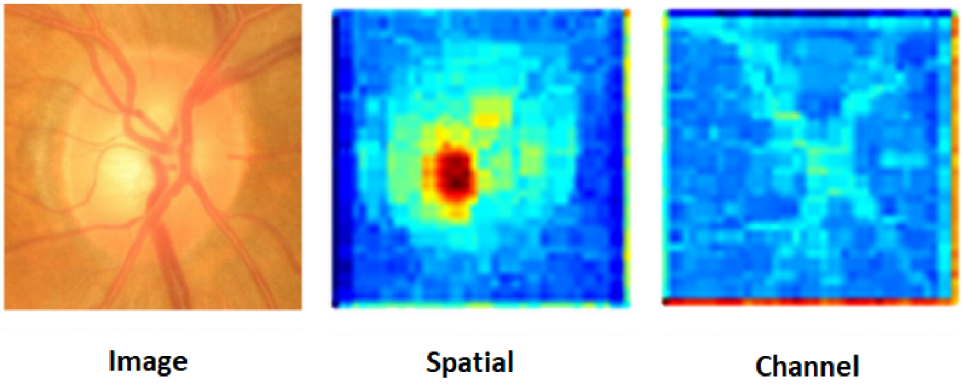

Figure 2.30: Attention, source: [21]

As the name suggests, spatial attention highlights relevant regions of the feature map. On the other hand, the channel attention highlights relevant feature maps as a whole.

- **Spatial attention module (SAM)**

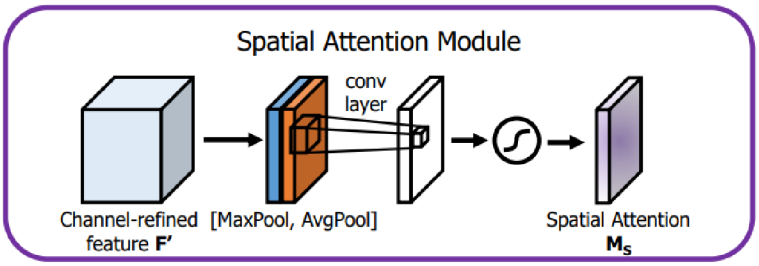

Figure 2.31: Spatial attention module, source: [35]

As shown in Figure 2.31, spatial attention consists of three sequential operations. First, a spatial average and maximum pooling are performed. Thereby, maximum and average feature representations are exposed to the following layers. The convolutional layer then consumes the output. Lastly, the sigmoid activation layer transforms the intermediate feature map values into the range of $< 0, 1 >$, thereby the spatial attention map is created.

- **Squeeze and excitation block (SE)**

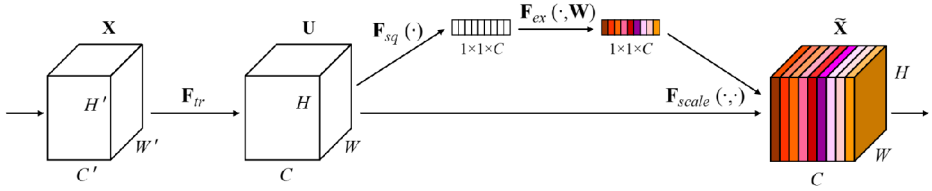

Figure 2.32: Squeeze and excitation block, source: [16]

Computation of the channel attention while using the squeeze and excitation block is visualised in Figure 2.32. First, an global average or maximum pooling is performed. It has been observed that the global average pooling produces slightly better results [16]. The intermediate feature map is then squeezed to the lower-dimensional vector and excited to the original-sized vector. Lastly, the sigmoid activation layer transforms the feature map values into the range of $< 0, 1 >$. Thus an attention map is created. [16]

- **Channel attention module (CAM)**

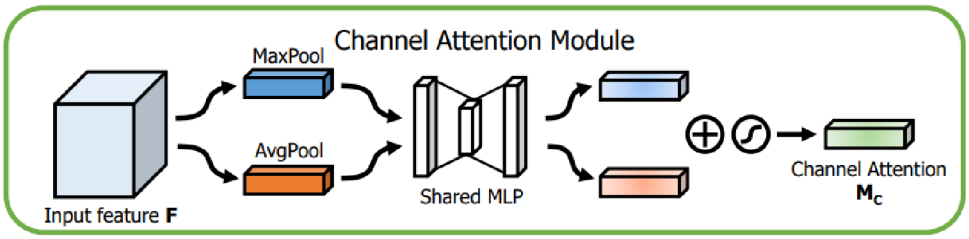

Figure 2.33: Channel attention module, source: [35]

The architecture of the channel attention module shown in Figure 2.33 is similar to the squeeze and excitation block except for the use of a global pooling layer. Both the global maximum and the global average poolings are used simultaneously. The abbreviation Shared MLP stands for Shard multi-level preceptor used for squeezing and the excitation of the intermediate outputs.

- **Combined attention module (CBAM)**

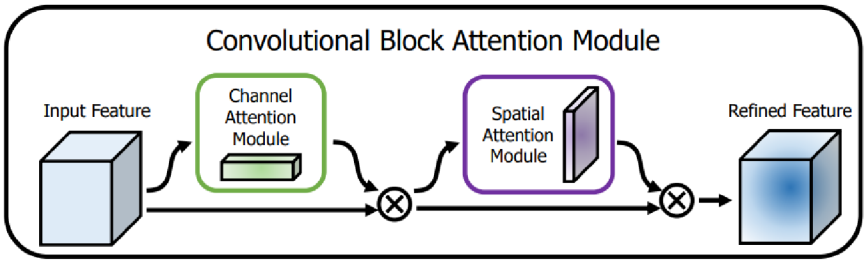

Figure 2.34: Convolutional block attention module, source: [35]

It has been observed that channel and spatial attention work well together. In the case of the CBAM module, channel attention is followed by spatial attention.

Chapter 3

Algorithm design and implementation

Correctly identifying and segmenting retinal eye pathologies is a challenging task. Some pathologies are only a few pixels large or/and similarly looking. Even an experienced ophthalmologist (eye doctor) might have difficulty identifying them correctly. This chapter introduces the proposed solution and its fundamental principles.

3.1 Architecture

A variety of models has been trained, resulting in several observations. Based on those observations, the proposed architecture has been designed.

Design choices

- **The U-Net architecture**, described in Chapter 2.4.4, achieves the state of the performance on medical problems. Therefore it has been selected as the basis for the proposed architecture.
- **Residual blocks**, described in Chapter 2.27, were used to provide the expanding path with even more detailed positional information. They also improve the exploding/vanishing gradient types of problems.
- **Attention**, described in Chapter 2.4.4, is a mechanism used for speeding up the training by applying bigger weights to regions/channels that are relevant. The best results were achieved using the CBAM block. The CBAM block applies a channel followed by spatial attention to the input.
- **The batch normalisation** has proved to be an absolute necessity. Without the batch normalisation, the model was unable to learn anything.
- **Dropout layers** were used to reduce overfitting (2.8). A dropout layer randomly disables a specified amount of kernels during training.

Proposed architecture

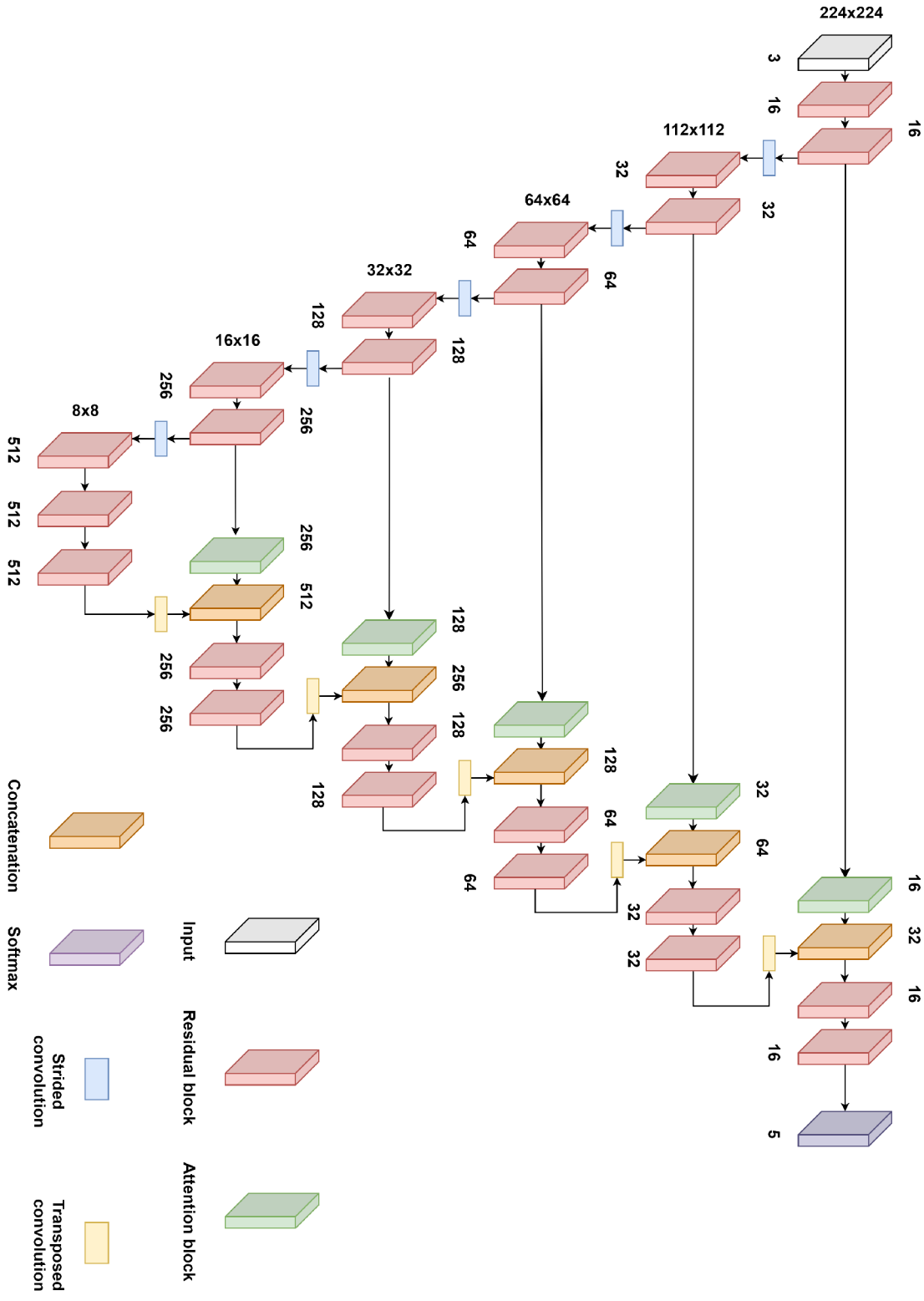

Figure 3.1: Proposed architecture

The architecture of the proposed system is shown in the Figure 3.1. For clarity, layers are consolidated into blocks. The implementation of individual blocks is explained in detail in Chapter 3.2.

The contracting path of the architecture comprises residual blocks (2.4.4) and strided convolution layers (2.26). The strided convolution is used for spatial dimensions reduction (downsampling). The following pattern is repeated throughout the path. First, a residual block is applied to the intermediate output. Once two consecutive residual blocks process the input, a strided convolution downsamples the intermediate output by half.

The expanding path comprises transposed convolution layers, attention and residual blocks. The transposed convolution is used for upsampling the input in a learnable way. The following sequence of operations is repeated through the expanding path. First, attention is applied to the adjacent level output of the contracting path. Then predeceasing level output of the expanding path is via the transposed convolution upsampled. Both intermediate outputs are concatenated, producing a feature map later consumed by two consecutive residual blocks.

Model hyper-parameters

- **Activation function**. A ReLU activation functions are used as the non-linear components of the network.
- **Kernels size**. Convolutional layers use kernels of size three, whereas transposed convolution layers use kernels of size two.
- **Number of kernels**. The first level of the U-Net has 16 kernels. The amount increases with a factor of two up to 512 kernels in the last level.
- **Dropout rate** of 0.2 (20% of neurons are dropped during training) was used.
- **Kernel weights initialisation**. Kernel weights were initialised using the HE uniform distribution. The HE uniform distribution is specially designed to be used with the ReLU activation function.

3.2 Implementation

This chapter introduces used technologies and describes individual components of the proposed system.

3.2.1 Technologies

The proposed system is developed using the Python programming language. Python was chosen because it has become a standard for data science projects and the enormous availability of data science libraries. Important used libraries are listed below.

- Tensorflow[1]

  Tensorflow is an open-source framework developed by the Google Brain Team. It is widely used for various machine and deep learning applications.

[1] https://www.tensorflow.org/

- keras[2]

  Keras is a high-level Tensorflow API that further simplifies the model design, training and evaluation.

- TensorBoard [3]

  The TensorBoard is a visualisation toolkit used for model evaluation.

- scikit-image[4]

  The scikit-image is an open-source library that provides a rich API for image processing.

- napari[5]

  The napari is an image viewer used for viewing and annotating multi-dimensional data.

- jupyter[6]

  The juypter is a web-based service used for interactive code execution.

- Albumentations [7]

  The albumantion is a library providing a variety of augmentation techniques.

3.2.2 System architecture

The project structure is presented below:

- `data/` Data used for training and evaluation
- `docs/` Documentation of the project
- `logs/` TensorBoard logs directory
- `models/` Model weights directory
- `notebooks/` Jupyter notebooks used for interactive code execution
- `src/` Source files of the proposed system
  - `architecture/` Implementation of the U-Net framework (3.2.3)
  - `data/` Data processing scripts
  - `metrics/` Metrics used for model evaluation
  - `losses/` Implemented loss functions
  - `handlers/` Training, evaluation and prediction scripts

[2]https://keras.io/
[3]https://www.tensorflow.org/tensorboard
[4]https://scikit-image.org/
[5]https://napari.org/index.html
[6]https://jupyter.org/
[7]https://albumentations.ai/

3.2.3 U-Net framework

A U-Net framework has been built to provide a simple and efficient interface for model design and creation. The basic building blocks of the framework are model skeletons and blocks. Blocks assemble individual layers like convolution into higher functional units like residual blocks, whereas model skeletons connect individual blocks, thus forming the architecture.

Layers

Before the architecture of individual blocks can be discussed, layers must be introduced.

- **Conv** layer performs a spatial convolution.
- **BN** stands for batch normalization. The layer transforms inputs to maintain mean close to 0 and standard deviation close to 1.
- **Activation** represents an activation layer. The layer transforms inputs by the activation function.
- **Add** performs element-wise addition.
- **Multiply** performs element-wise multiplication.
- **Concat** concatenates inputs by the last axis.
- **Dropout** randomly sets units to 0.
- **Pooling** reduces inputs spatial dimensions by applying either the maximum (2.17) or average pooling (2.18). Special case of the pooling is a global maximum/average pooling that reduces the spatial dimension to the size of 1.

Blocks

This chapter introduces individual blocks and describes their architecture. Layers represented by a dashed line are optional.

- **ConvBlock**

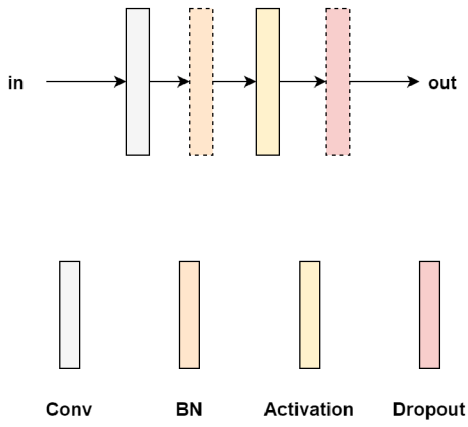

Figure 3.2: ConvBlock architecture

The ConvBlock forms a basic convolutional block comprising a convolutional, batch normalisation, activation and dropout layers.

- **ResConvBlock**

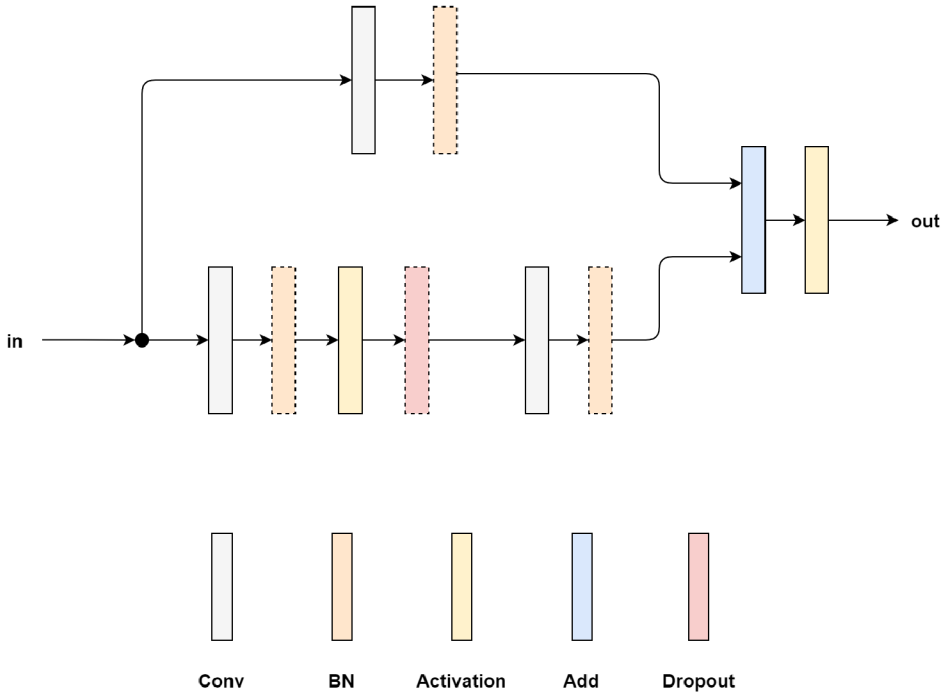

Figure 3.3: ResConvBlock architecture

The ResConvBlock forms a residual convolutional block (2.27).

- **DenseBlock**

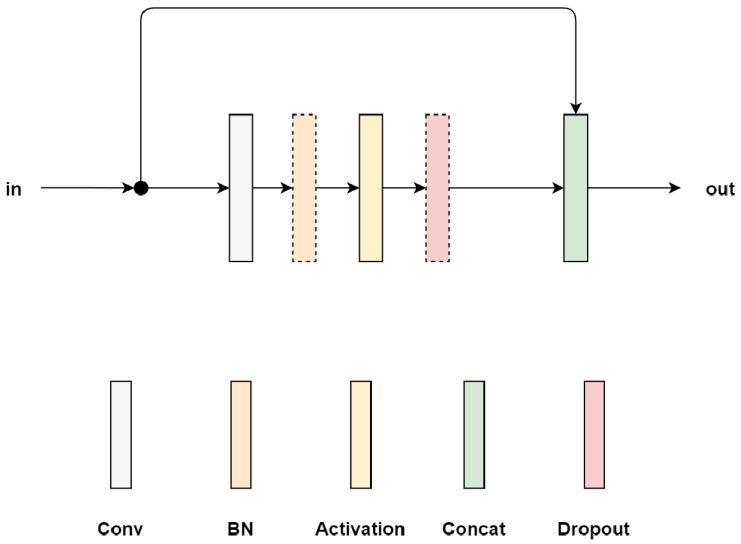

Figure 3.4: DenseBlock architecture

The DenseBlock is an implementation of the densely connected block (2.28). Only one iteration of convolutional layers within the block is shown for better clarity.

- **SEBlock**

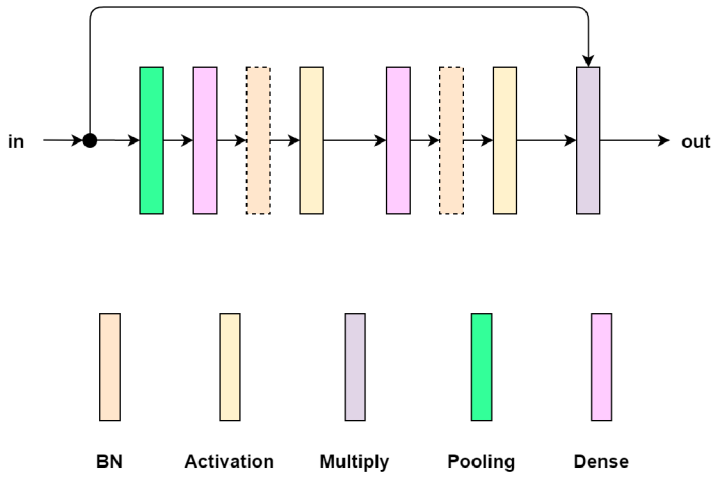

Figure 3.5: SEBlock architecture

The SEBlock forms a squeeze and excitation block (2.32). Based on the configuration, the pooling layer represents either a GlobalMaxPooling or a GlobalAveragePooling layer.

- **CBAMBlock**

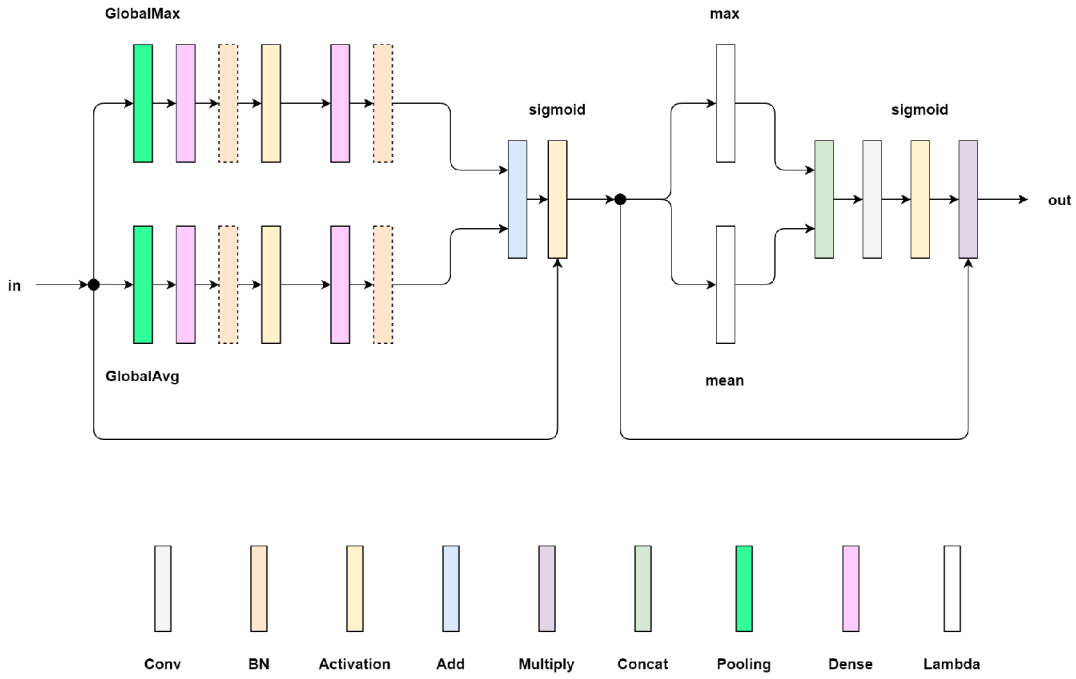

Figure 3.6: CBAMBlock architecture

The CBAMBlock implements a channel followed by a spatial attention mechanism applied to the input. The GlobalMax/GlobalMin pooling layer produces a single maximum/average value per channel. The Lambda max/mean layer computes a max/mean value over the last (channel) axis.

U-Net model skeletons

The implementation of the U-Net model skeleton can be found within the sub-directory `/skeletons`.

```
model = compose(input_shape=(224, 224, 3), num_chanels=5,
    enc_kernels=[(2, 16), (2, 32), (2, 64), (2, 128), (2, 256), (2, 512)],
    enc_conv_block=ResConvBlock('relu', dropout_rate=0.2),
    dec_kernels=[(2, 16), (2, 32), (2, 64), (2, 128), (2, 256), (2, 512)],
    dec_conv_block=ResConvBlock('relu', dropout_rate=0.2),
    downsample_block=lambda x, size: Conv2D(K.int_shape(x)[-1], (3, 3),
        strides=size, padding='same',kernel_initializer='he_uniform')(x),
    upsample_block=lambda x, size: Conv2DTranspose(K.int_shape(x)[-1] // 2,
        (2,2), strides=size, padding='same',
        kernel_initializer='he_uniform')(x),
        attention_block=CBAMBlock('relu'), attention_mode='dec')
```

Figure 3.7: Proposed model configuration

The model skeleton implements a compose() method used for the model creation. Its function can be shown in the way the proposed model is created. The model is composed by providing the configuration shown in Figure 3.7.

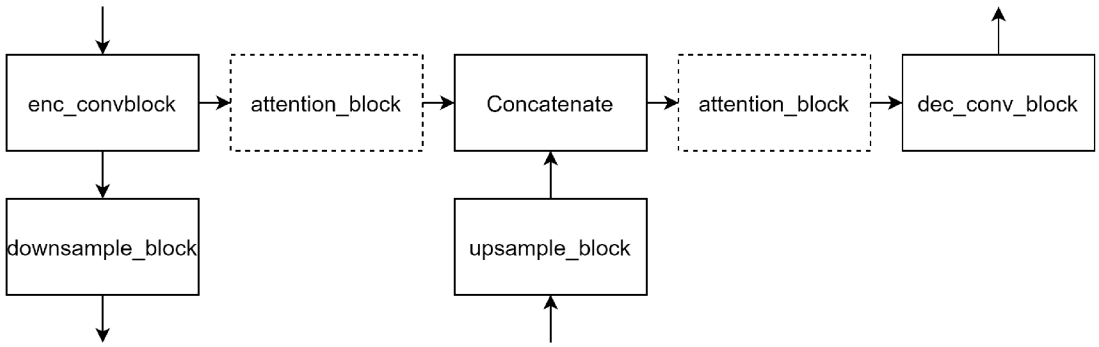

Figure 3.8: U-Net skeleton blocks connections

For a better understanding, Figure 3.8 shows the interconnection of individual blocks.

3.2.4 Preprocessing

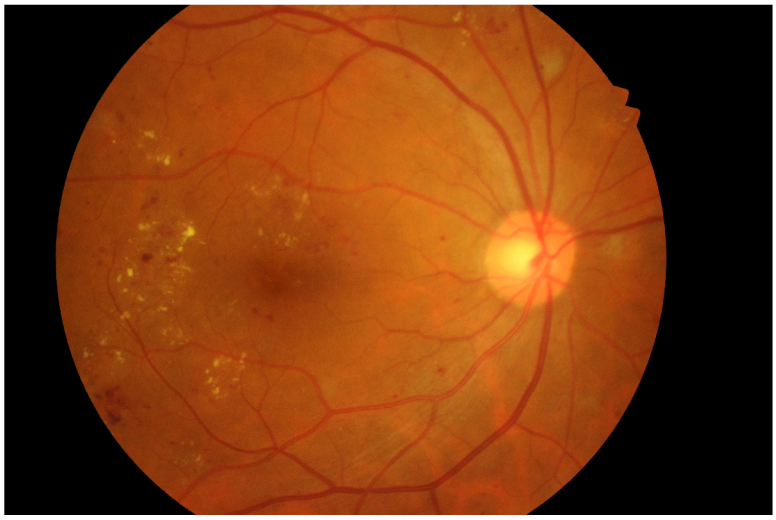

Figure 3.9: Original image

Figure 3.10: Preprocessed image

A green channel extraction and CLAHE preprocessing method has been implemented. The green channel is selected because it offers the best amount of contrast. Once the green channel is extracted, an adaptive contract limited equalisation (CLAHE) is applied to improve the contrast further.

3.2.5 Prediction

The prediction from raw data is handled by the Predictor class, whose implementation is located in the `/src/model/predictor.py` file.

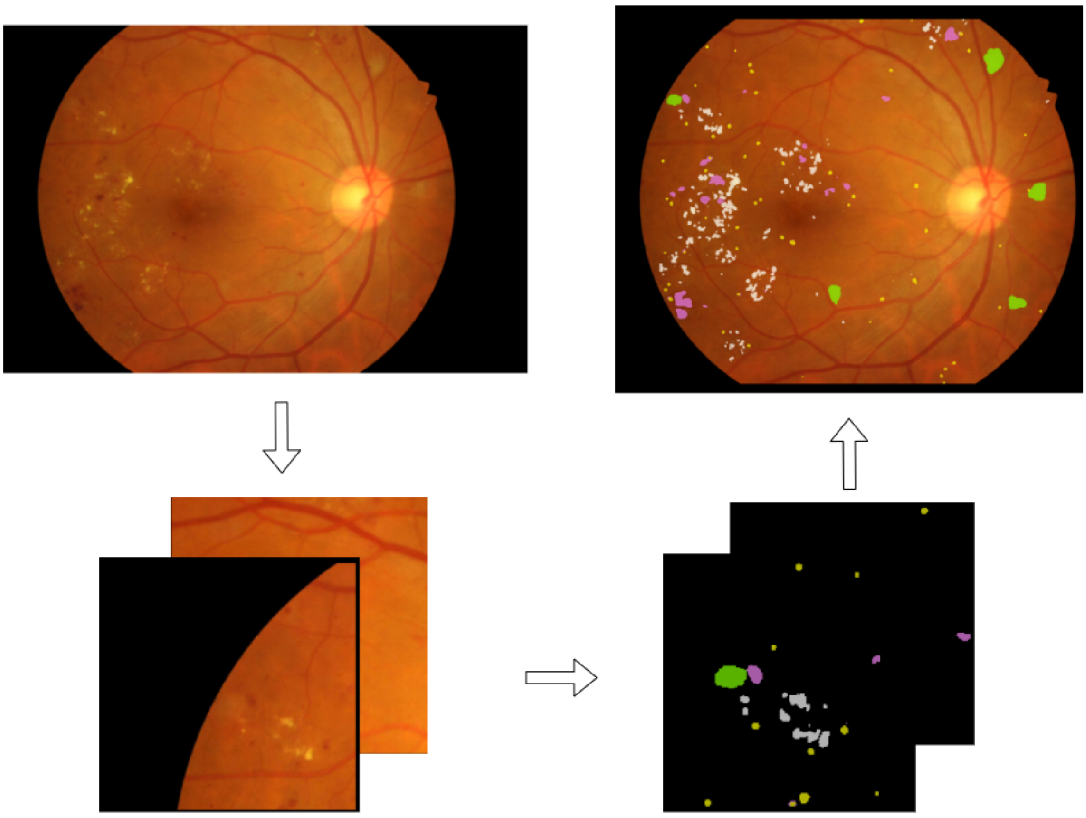

Figure 3.11: Prediction pipeline

The prediction process is discussed below.

1. Original (to be annotated) image is loaded from the disk.
2. Spatial dimensions of the image are reduced to 1072x712 pixels.
3. If the model was trained using preprocessed data, the image is preprocessed.
4. The image is divided into small overlapping patches. The overlap is used for extending the contextual region of the prediction.
5. The patches are fed to the model resulting in the same number of prediction masks.
6. The final prediction masks are created by appropriately arranging the prediction masks.

3.3 Running the code

This chapter describes how to use the implemented system.

3.3.1 Environment

First, the environment needs to be prepared. This can be done by issuing the following command from the project's top directory.

- `pip install -r requirements.txt`

3.3.2 Data

Before any training or evaluation can occur, the datasets need to be created. The Jupyter notebook `/project/src/notebooks/data.ipynb` encapsulates the data preparation process. To prepare the data, execute the following sequence of steps.

1. Inside the notebook, set the `TOP_DIR` variable to the project's top directory location.
2. Run cells `# conf_0`, `# conf_1` and `# organize_0` to create a directory structure accepted by the implemented methods.
3. Download the segmentation part, denoted as `A. Segmentation.zip`, of the dataset [8] or use the provided dataset `/idrid.zip`.
4. Unzip the dataset into the folder `TOP_DIR/data/unpacked`.
5. Process the data by running the cell `# extract_0`. This action can take up to 45 minutes. After processing the data, patches (small images used for training and evaluation) will appear in the `TOP_DIR/data/patches` directory.

3.3.3 Training

The Jupyter notebook `/project/src/notebooks/model.ipynb` encapsulates the model training, evaluation and prediction. To train the model, execute the following sequence of steps.

1. Inside the notebook, set the `TOP_DIR` variable to the project's top directory.
2. Run cells `# conf_0`, `# conf_1` and `# model_0` to instantiate the model.
3. Run the cell `# train_0` to create a directory that will contain the model weights.
4. Run cells `# train_1` and `# train_2` to instantiate the model training wrapper and train the model.

3.3.4 Evaluation

Execute the following sequence of steps to evaluate the model.

1. If using the provided model weights, unzip the `/weights.zip` to the `TOP_DIR/models` directory.
2. Run cells `# conf_0`, `# conf_1` and `# model_0` to instantiate the model.
3. Run cells `# evaluate_0` and `# evaluate_1` to instantiate the model evaluation wrapper and evaluate the model. Once computed the model evaluation report will appear at the bottom of the cell.

[8] https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid

3.3.5 Prediction

To make a prediction execute the following sequence of steps.

1. If using the provided model weights, unzip the `/weights.zip` to the `TOP_DIR/models` directory.
2. Run cells `# conf_0`, `# conf_1`, `# model_0` and `# predict_0` to instantiate the model and the model prediction wrapper.
3. Set the `IMG_PATH` to the to be predicted image location.
4. Run cells `# predict_1` to make the prediction. Once computed, a window containing the original image with generated predictions will appear.

Chapter 4

Experiments and evaluation

Over a hundred different models have been trained using various training methods and loss functions leading up to the proposed architecture. This chapter discusses how the proposed architecture was trained and presents achieved results.

4.1 Data

This subchapter introduces the IDRiD dataset used for training and discusses the dataset preparation. Every model has been trained on the Indian Diabetic Retinopathy Image Dataset (IDRiD). [25] The IDRiD is a multipurpose data-set created by retinal specialists at Eye Clinic located in Nanded, (M.S.), India. The dataset is available from IEEE Dataport Repository [1]. It is divided into three parts - segmentation, disease grading and localisation. Every part contains colour retina images.

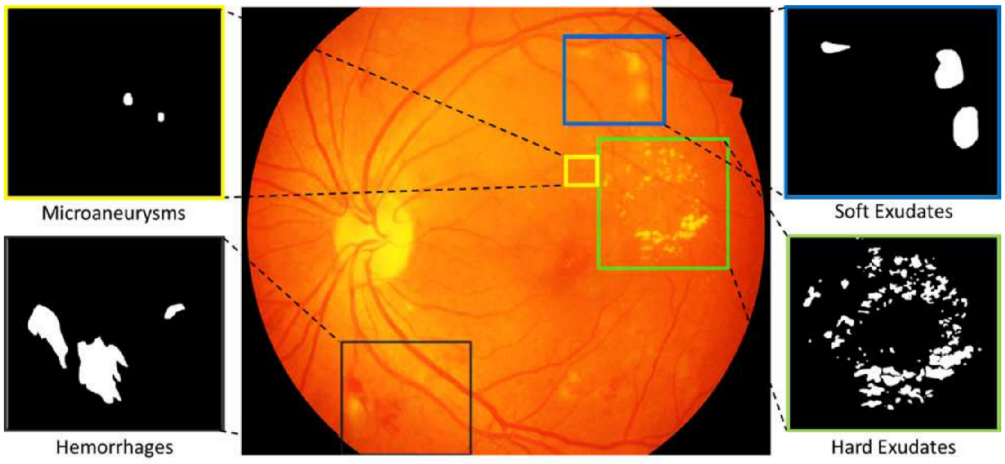

Figure 4.1: IDRiD annotations, source: [25]

Only the segmentation part has been used for model training and evaluation. It contains 81 colour images and belonging pixel-level annotations. The annotations are provided as

[1] https://ieee-dataport.org/open-access/indian-diabetic-retinopathy-image-dataset-idrid

binary masks for the following lesions - hard exudates, soft exudates (cotton wool spots), microaneurysms and hemorrhages.

Before the training or evaluation of the model could begin, the datasets had to be prepared. The process is described below.

1. Spatial dimensions of the data are reduced to 1072x712 pixels.
2. Data is divided into training and validation datasets using the 5-fold cross-validation technique.
3. To increase the amount of data, original images are divided into small overlapping patches. The spatial dimensions of the patches were set to 224x224 pixels. The overlap was set to 112x112 pixels.

4.2 Training

This chapter summarises the training process.

- **Batches**. The data was fed to the model in the form of mini-batches. The batch size of 15 patches has proved to produce the best results.
- **Loss function**. Taversky loss function 2.11 has proved to provide much better results out of all tried loss functions. Best results were achieved for parameters set to $\alpha = \beta = 0.5$ simplifying the loss function to be a Dice loss 2.10.
- **Augmentation**. To reduce the overfitting 2.8 several augmentation techniques 2.3.3 specifically scaling, rotation by a multiple of 90 degrees, random brightness and contrast were used.
- **Optimizer**. The model was trained using the Adam optimizer 2.21 with the first and second moment estimates set to 0.9 and 0.999.
- **Learning rate**. The initial learning rate was set to 10^{-4}.

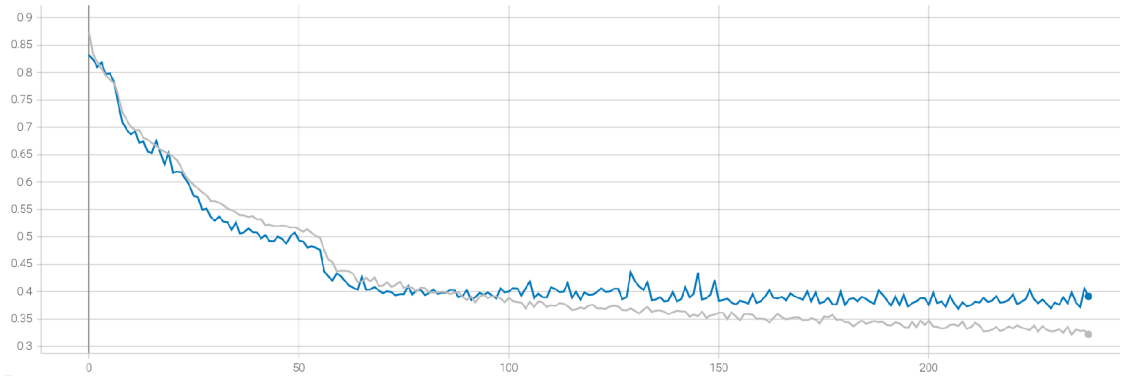

Figure 4.2: Loss function

Figure 4.2 shows the loss function of the best performing model (discussed in detail below) trained on fold three. The model has been trained for 250 epochs on the Nvidia RTX4000 graphic card. The best performance (smallest validation loss) was achieved at the 208 epoch. Therefore those weights are used for the final product.

4.3 Evaluation

This chapter presents achieved results. First, performance on individual folds is evaluated then the best performing model is discussed in detail.

Performance on folds

The data, as previously stated, has been divided into five-folds, each containing training and validation datasets.

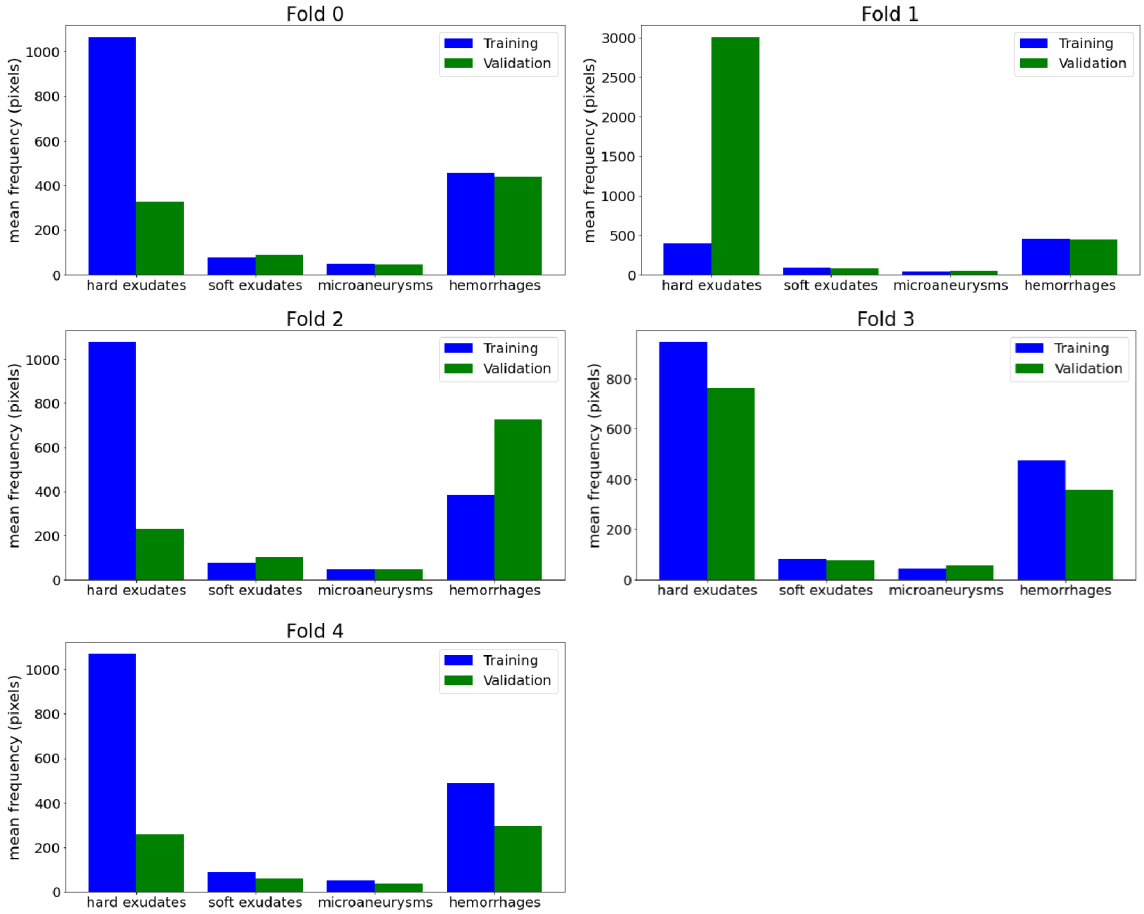

Figure 4.3: Mean lesion pixel frequencies per fold distributions

Figure 4.3 shows the mean frequencies of lesion pixels per dataset. A blue/green colour represents the training/validation dataset frequencies. The dataset is relatively small because all of the presented cases are more or less imbalanced. Fold 3 is the most balanced fold out of all the presented folds. Therefore, it should produce results close to the performance of the real-world data.

fold	hard exudates	soft exudates	hemorrhages	microaneurysms
0	0.73	0.46	**0.53**	0.33
1	0.25	0.42	0.32	**0.55**
2	**0.76**	0.43	0.44	0.33
3	0.74	**0.5**	0.45	0.33
4	0.74	0.39	0.32	0.44
mean	0.62	0.4525	0.435	0.385

Table 4.1: AUC-PR scores obtained on folds one to five

The Table 4.1 presents achieved results on folds one to five. The AUC-PR scores were obtained by computing the area under the precision-recall curve. The model trained on fold three has been selected as the final model. This decision was made based on the AUC-PR score and the knowledge of mean lesion pixel frequencies per fold distributions as shown in Figure 4.3.

Final model

The model trained on fold three is discussed in detail in this chapter.

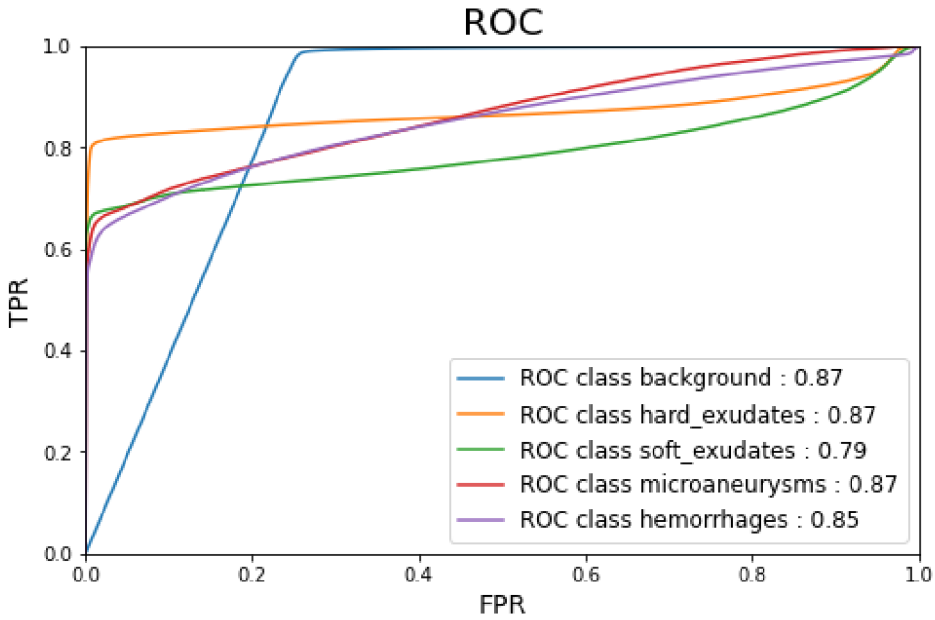

Figure 4.4: Receiver operating characteristic curve

$$TPR = \frac{TP}{TP + FN} \tag{4.1}$$

$$FPR = \frac{TP}{TP + FN} \tag{4.2}$$

Figure 4.4 shows the receiver operating characteristic curve. The x-axis represents the true positive rate (4.1) also referred to as sensitivity. The y-axis represents a false positive rate (4.2). However, given such a highly imbalanced dataset, it presents too optimistic

results. The false-positive rate is due to the high amount of true-negatives, always close to 1 except for the background region.

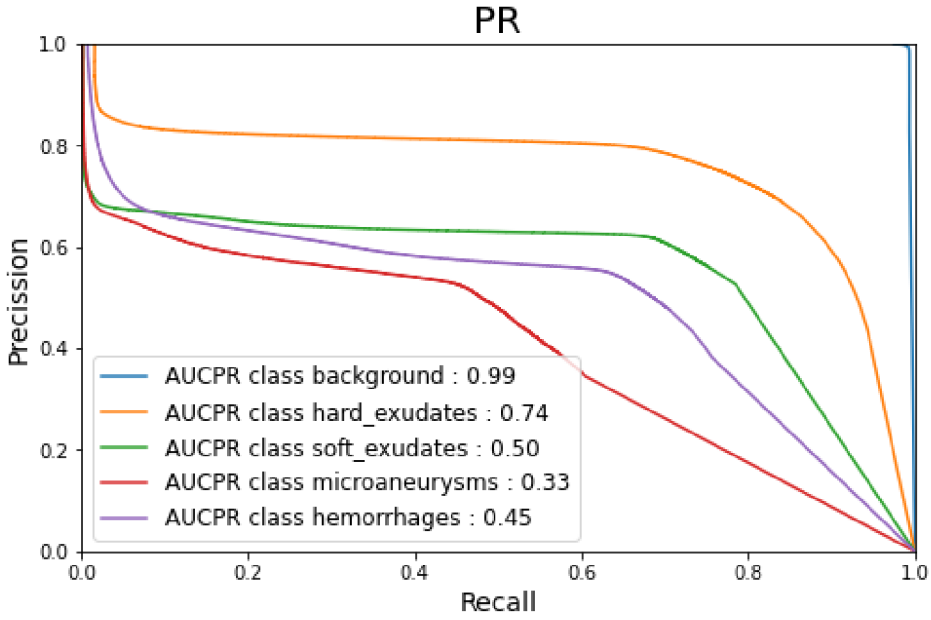

Figure 4.5: Precision-recall curve

$$precision = \frac{TP}{TP + FP} \tag{4.3}$$

$$recall = \frac{TP}{TP + FN} \tag{4.4}$$

Since the precision-recall curve does not consider true negatives, it can be used even for highly imbalanced datasets. Figure 4.5 demonstrates the segmentation abilities of the model. The x-axis represents the recall (4.4) also referred to as sensitivity. The recall determines the ratio of correctly predicted diseased regions to all diseased areas). The y-axis represents a precision (4.3) also called the positive predictive value. The precision is determined by calculating the ratio of correctly predicted diseased regions to all regions predicted as diseased. As shown in Figure 4.5 the model performs well considering the hard exudates lesions. A good precision is achieved at a reasonable recall in addressing the haemorrhages and soft exudates lesions. Correctly segmenting microaneurysm is a challenging task. Some are almost indistinguishable from the background. In this case, a steeply decreasing precision with increasing recall is evident.

		predicted				
ground truth		background	EX	SE	MA	HE
	background	**48910130**	139953	16052	12667	85753
	hard exudates (EX)	207755	**559372**	1536	3	18
	soft exudates (SE)	28913	2159	**45529**	68	341
	microaneurysms (MA)	27451	25	0	**25130**	4352
	hemorrhages (HE)	164270	364	100	9550	**185389**

Table 4.2: Confusion matrix

The confusion matrix for the model evaluated on the validation dataset is shown in Figure 4.2. The main problem of the model is to distinguish lesions from the background. If the pixel is labelled as a lesion, it is likely to be correctly classified. The most significant overlap of about 7% was measured between hemorrhages and microaneurysms.

lesion	**sensitivity**	**specificity**	**precision**
hard exudates	0.73	0.9971	0.8
soft exudates	0.59	0.9996	0.72
hemorrhages	0.52	0.9982	0.67
microaneurysms	0.44	0.9995	0.53

Table 4.3: Confusion matrix based metrics

The performance considering the sensitivity, specificity and positive predictive value metrics is presented in Table 4.3. Given the task's difficulty, it can be said that the model performs well while segmenting the hard exudates. A good performance was achieved while segmenting the soft exudates and hemorrhages. The microaneurysms are usually only a few pixels wide or blend in with the background making them challenging to segment correctly. Thus, the lower performance is somewhat expected.

Chapter 5

Conclusion

This thesis aimed to design and implement a system that automatically detects retinal eye pathologies, specifically exudates/drusens and hemorrhages.

The implemented system based on the supplied image of the retina automatically generates masks representing occurrences of individual pathologies. The result is then presented to the user. For this purpose, the napari library is used, enabling the user to manipulate the images (move, zoom in/out, turn on/off individual masks). A convolutional neural network handles the segmentation. The network architecture is based on the U-Net architecture enhanced by residual blocks combined with an attention mechanism. The model was trained on the Indian Diabetic Retinopathy Image Dataset (IDRiD) to segment four lesions - hard exudates, soft exudates, hemorrhages and microaneurysms. The dataset was divided into five parts using the k-fold cross-validation technique. Then five models were trained, and the best was selected as the final product. The models were thoroughly analysed using the receiver operating characteristic curves, precision-recall curves and confusion matrix based metrics (sensitivity, specificity, precision). Since the dataset is highly imbalanced, most insights were obtained using the PR curve. Based on testing, I can state the segmentation of hard exudates works reasonably well since it achieves an AUC-PR (area under the precision-recall curve) of 74%. Segmentation of soft exudates, hemorrhages, and microaneurysms achieves AUC-PR scores of 50%, 45% and 33%, respectively. For such a challenging task, I believe that this is a good result (see the previous chapter).

Another output of this project is the U-Net framework. Since various models had to be trained, there was a need for a tool that would simplify the model creation process. Therefore, the U-Net framework has been developed. The U-Net framework is a high-level API that lets the user create U-Net-based models without the need for programming simply by providing a model configuration.

Since there are still things to improve, I intend to continue working on this project as part of my follow-up master's degree thesis. I believe that the proposed architecture is powerful enough to achieve even better performance if the preprocessing of the data and method for composing the prediction are improved. Another thing to try is to create an ensemble model containing several networks, each trained on a different input size resolution. Without a doubt, the model would benefit from a larger dataset. Therefore I would like to create a larger dataset with the cooperation of ophthalmologists or synthesise the data using the generative adversarial network.

Bibliography

[1] *Machine Learning Workflow* [online]. [cit. 2022-04-01]. Available at: https://www.run.ai/guides/machine-learning-engineering/machine-learning-workflow.

[2] Addo, E., Bamiro, O. A. and Siwale, R. Anatomy of the eye and common diseases affecting the eye. In: *Ocular drug delivery: Advances, challenges and applications*. Springer, 2016, p. 11–25. ISBN 3319476890.

[3] Anello, E. *Understanding Optimization Algorithms* [online]. March 2021 [cit. 2022-04-01]. Available at: https://laptrinhx.com/understanding-optimization-algorithms-3818430905/.

[4] Baheti, P. *12 Types of Neural Network Activation Functions: How to Choose?* [online]. May 2022 [cit. 2022-04-01]. Available at: https://www.v7labs.com/blog/neural-networks-activation-functions.

[5] Basavarajaiah, M. *Maxpooling vs minpooling vs average pooling* [online]. February 2019 [cit. 2022-04-01]. Available at: https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9.

[6] Basit, A. and Egerton, S. Bio-medical imaging: Localization of main structures in retinal fundus images. *IOP Conference Series: Materials Science and Engineering*. november 2013, vol. 51. DOI: 10.1088/1757-899X/51/1/012009.

[7] Borsos, B., Nagy, L., Iclănzan, D. and Szilagyi, L. Automatic detection of hard and soft exudates from retinal fundus images. *Acta Universitatis Sapientiae, Informatica*. august 2019, vol. 11, p. 65–79. DOI: 10.2478/ausi-2019-0005.

[8] Brownlee, J. *Train Neural Networks With Noise to Reduce Overfitting* [online]. December 2018 [cit. 2022-04-01]. Available at: https://machinelearningmastery.com/train-neural-networks-with-noise-to-reduce-overfitting/.

[9] Brownlee, J. *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks* [online]. April 2019 [cit. 2022-04-01]. Available at: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[10] Chakravorty, D. *General Idea on Confusion Matrix* [online]. [cit. 2022-04-01]. Available at: https://www.debadityachakravorty.com/ai-ml/cmatrix/.

[11] Chu, L. *Human Neuron vs. Artificial Neuron: Similarities and Discrepancies* [online]. September 2021 [cit. 2022-04-01]. Available at: https://lanstonchu.wordpress.com/2021/09/06/human-neuron-vs-artificial-neuron-similarities-and-discrepancies/.

[12] Communications, O. of and Liaison, P. *Brain Basics: The Life and Death of a Neuron* [online]. National Institute of Neurological Disorders and Stroke, december 2019 [cit. 2022-03-25]. Available at: https://www.ninds.nih.gov/Disorders/Patient-Caregiver-Education/Life-and-Death-Neuron.

[13] Fei Fei Li, R. G. *CS231n: Convolutional Neural Networks for Visual Recognition* [online]. 2022. Available at: https://cs231n.github.io/.

[14] Goodfellow, I., Bengio, Y. and Courville, A. *Deep Learning*. MIT Press, 2016. ISBN 0262035618. http://www.deeplearningbook.org.

[15] He, K., Zhang, X., Ren, S. and Sun, J. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, p. 770–778. DOI: 10.1109/CVPR.2016.90.

[16] Hu, J., Shen, L. and Sun, G. Squeeze-and-Excitation Networks. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, p. 7132–7141. DOI: 10.1109/CVPR.2018.00745.

[17] Huang, G., Liu, Z., Van Der Maaten, L. and Weinberger, K. Q. Densely connected convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, p. 4700–4708. DOI: 10.48550/ARXIV.1608.06993.

[18] Jain, V. *Everything you need to know about "Activation Functions" in Deep learning models* [online]. December 2019 [cit. 2022-04-01]. Available at: https://towardsdatascience.com/everything-you-need-to-know-about-activation-functions-in-deep-learning-models-84ba9f82c253.

[19] Kvifte, T. *Improve Your Deep Learning Models with Image Augmentation* [online]. April 2021 [cit. 2022-04-01]. Available at: https://dev.to/itminds/improve-your-deep-learning-models-with-image-augmentation-4j7f.

[20] Lagali, N. Corneal Stromal Regeneration: Current Status and Future Therapeutic Potential. *Current Eye Research*. Taylor and Francis. 2020, vol. 45, no. 3, p. 278–290. DOI: 10.1080/02713683.2019.1663874. PMID: 31537127. Available at: https://doi.org/10.1080/02713683.2019.1663874.

[21] Li, S., Ge, C., Sui, X., Zheng, Y. and Jia, W. Channel and Spatial Attention Regression Network for Cup-to-Disc Ratio Estimation. *Electronics*. 2020, vol. 9, no. 6. DOI: 10.3390/electronics9060909. ISSN 2079-9292. Available at: https://www.mdpi.com/2079-9292/9/6/909.

[22] Li, X., Sun, X., Meng, Y., Liang, J., Wu, F. et al. Dice loss for data-imbalanced NLP tasks. *ArXiv preprint arXiv:1911.02855*. arXiv. 2019. DOI: 10.48550/ARXIV.1911.02855. Available at: https://arxiv.org/abs/1911.02855.

[23] Misra, D. *Attention Mechanisms in Computer Vision: CBAM* [online]. 2020 [cit. 2022-04-01]. Available at: https://blog.paperspace.com/attention-mechanisms-in-computer-vision-cbam/.

[24] Pai, P. *Data Augmentation Techniques in CNN using Tensorflow* [online]. October 2017 [cit. 2022-04-01]. Available at: https://medium.com/ymedialabs-innovation/data-augmentation-techniques-in-cnn-using-tensorflow-371ae43d5be9.

[25] Porwal, P., Pachade, S., Kamble, R., Kokare, M., Deshmukh, G. et al. Indian diabetic retinopathy image dataset (IDRiD): a database for diabetic retinopathy screening research. *Data*. Multidisciplinary Digital Publishing Institute. 2018, vol. 3, no. 3, p. 25. DOI: 10.21227/H25W98.

[26] Pröve, P.-L. *An Introduction to different Types of Convolutions in Deep Learning* [online]. July 2017 [cit. 2022-04-01]. Available at: https://towardsdatascience.com/types-of-convolutions-in-deep-learning-717013397f4d.

[27] Ronneberger, O., Fischer, P. and Brox, T. U-net: Convolutional networks for biomedical image segmentation. In: Springer. *International Conference on Medical image computing and computer-assisted intervention*. 2015, p. 234–241. DOI: 10.48550/ARXIV.1505.04597.

[28] Salehi, S. S. M., Erdogmus, D. and Gholipour, A. Tversky loss function for image segmentation using 3D fully convolutional deep networks. *CoRR*. 2017, abs/1706.05721. Available at: http://arxiv.org/abs/1706.05721.

[29] Schneiderman, H. *The Funduscopic Examination*. 3rdth ed. 1990. Available at: https://www.ncbi.nlm.nih.gov/books/NBK221.

[30] Simmons, G. F. and Simmons, G. F. *Calculus with analytic geometry*. McGraw-Hill New York, 1996. ISBN 0070576424.

[31] Sweta. *Batch, Mini Batch and Stochastic gradient descent* [online]. August 2020 [cit. 2022-04-01]. Available at: https://sweta-nit.medium.com/batch-mini-batch-and-stochastic-gradient-descent-e9bc4cacd461.

[32] Tengyu Ma, C. R. *CS229: Machine Learning* [online]. 2022. Available at: https://cs229.stanford.edu/.

[33] Tuteja, D. S. *Fundoscopic Appearances of Retinal Pathologies* [online]. November 2021 [cit. 2021-01-21]. Available at: https://geekymedics.com/fundoscopic-appearances-of-retinal-pathologies/.

[34] Verhoeven, G. J. J. The reflection of two fields – Electromagnetic radiation and itsrole in (aerial) imaging. [online]. october 2017, [cit. 2021-12-25]. Available at: https://www.researchgate.net/publication/320616988_The_reflection_of_two_fields_-_Electromagnetic_radiation_and_its_role_in_aerial_imaging.

[35] Woo, S., Park, J., Lee, J.-Y. and Kweon, I. S. Cbam: Convolutional block attention module. In: *Proceedings of the European conference on computer vision (ECCV)*. 2018, p. 3–19. DOI: 10.48550/ARXIV.1807.06521.

[36] Zhu, J., Zhang, E. and Rio Tsonis, K. Eye Anatomy. In:. November 2012. DOI: 10.1002/9780470015902.a0000108.pub2. ISBN 0470016175.

Appendix A

Contents of the DVD

The directory structure of the provided DVD:

- `/project` Source files of the system
- `/idrid.zip` Segmentation part of the IDRiD dataset
- `/weights.zip` Trained model weights
- `/text` Source files of the thesis
- `/thesis.pdf` Text of the thesis

Appendix B

Predictions

The following screenshots show the pathology prediction based on the unseen retinal image created by the final model.

Figure B.1: Retinal image with predicted occurrences of individual pathologies

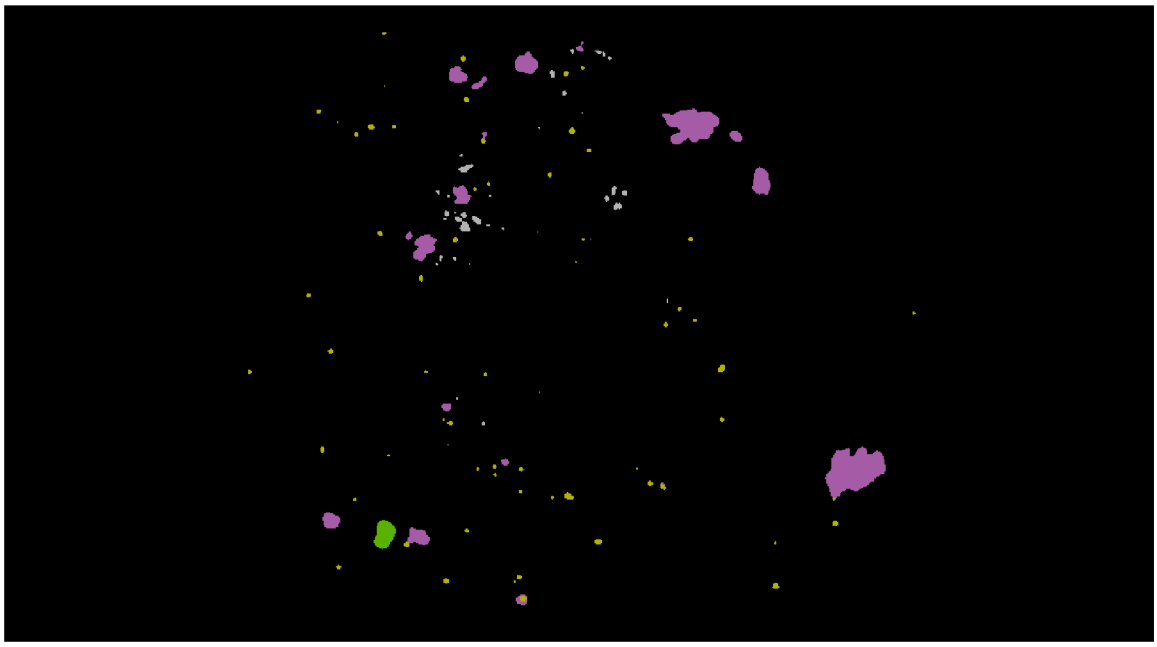

Figure B.2: Masks capturing the predicted occurrences of individual pathologies

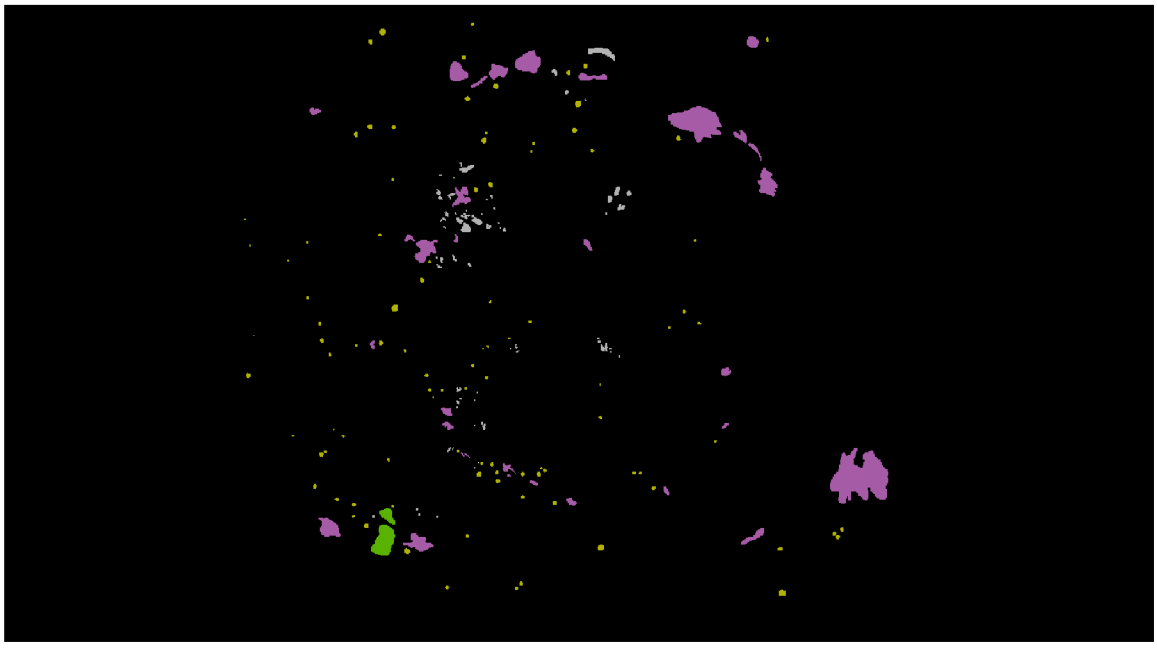

Figure B.3: Masks provided by an ophthalmologist capturing the real occurrences of individual pathologies, source: [25]