RPEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIE
Ministery of Higher Education and Scientific Research
University Mohamed Khider – BISKRA
Faculty of Exact Sciences, Natural and Life Sciences
Computer Science Departement

Order N° : IA23/M2/2021

Dissertation

Presented to obtain the academic master's degree in

Computer science

Option : **Artificial Intelligence (AI)**

Deep learning approach for early diabetic retinopathy diagnosis

By :
FELLAH KAOUTHAR MANAR

Members of the jury :

FULL NAME	Grade	President
TIGANE Samir	MCB	Supervisor
FULL NAME	Grade	Examiner

Année universitaire 2021-2022

dedicace

To my parents Youssef and Naima

To my sisters Rihab and Meriem

To all my family members

Acknowledgements

First of all, I would like to thank God Almighty for giving me the opportunity and guidance to achieving my goal and to be successful in this part of my life's journey so far.
I would like to thank my parents for their love and support, especially within my last three months of graduate studies. Without them, this day would not have been possible.
I would also like to thank my sisters Rihab and Meriem for not allowing me to give up when the chips were down, for always being there.
Also, i would like to thank my supervisor Doctor **TIGANE Samir** and Professor **KAHLOUL Laid** for their consistent support and guidance during the running of this project.
Then I would like to thank all AI teachers, I learned a lot from them.
Finally, I would like to thank all my colleagues and my friends. They always kept me motivated and helped me by giving me mental support and strength to finish my thesis.
Thank you for supporting and guiding me to where I am today. May the Almighty God continue to bless you all!

FELLAH Kaouthar Manar

Abstract

Diabetes Mellitus (DM) is a metabolic illness that occurs when the body's blood sugar levels become excessively high. It is a serious public health problem, affecting 463 million people worldwide and this number is projected to rise to 700 million by 2045. Diabetic Retinopathy (DR) is the most common specific complication of DM. DR is a leading cause of blindness among working-age adults. Early identification and treatment of DR can lower the risk of vision loss greatly. Since a manual diagnosis is prone to misdiagnosis and requires more effort, the automated methods for DR detection are cost and time effective. Deep learning has recently been one of the most popular strategies for improving experience in a range of fields, particularly medical image analysis and classifications. In this research, we demonstrate the use of convolutional neural networks (CNNs) on color fundus images. These images are preprocessed with various filters before being fed into the training model. Finally, experimental results show that the proposed approach outperforms similar works in the literature.

Key-words:Healthcare, Diabetic Retinopathy, CNN, Artificial Intelligence, Image Processing, Convolutional Neural Networks.

Résumé

Récemment, l'intelligence artificielle (IA) a envahi tous les domaines de la recherche scientifique, pour ce qu'elle apporte en termes de solutions. Le domaine de la santé n'est pas une exception. Le diabète est l'une des maladies les plus fréequente dans le monde et en Algérie. La Rétinopathie diabétique est la principale complication ophtalmologique chez les patients diabétiques. Les études épidémiologiques citent la rétinopathie diabétique parmi les 5 premières causes de cécité et la première cause de cécité avant l'âge de 50 ans. Une détection précoce et un traitement adapté permettent de réduire considérablement le risque de perte de vue. Les autorités médicales recommandent un examen annuel pour les patients diabétiques. Dans notre projet de master, nous avons exploré le potentiel de l'apprentissage profond pour l'analyse d'images rétine. Nous avons étudié les concepts de DL avec un algorithme de réseau neuronal convolutionnel (CNN) pour construire un modèle efficace qui peut détecter et classer automatiquement les niveaux de la rétinopathie diabetique. Dans ce projet, nous avons appliqué une architecture CNN avec plusieurs paramètres sur 2 base de données différentes de rétinopathie diabétique avec des différentes structures.

Mots-clés: Soins de santé, Diabète, Rétinopathie diabétique, Intelligence artificielle, réseau neuronal convolutionnel, Apprentissage profond.

Contents

List of source code

List of Tables

List of Figures

General introduction

Computer Science is the field of knowledge that deals with the study of computers and computational systems. Its principal areas include artificial intelligence and machine learning, computer systems and networks, security, databases, human-machine interaction, computer vision, numerical analysis, programming languages, software engineering, bio informatics and theory of computing.

Computer vision is an interdisciplinary sub-field of Computer Science that deals with methods for understanding relevant information present in images. From an engineering perspective, its main purpose is developing methods and algorithms for automatically acquiring, processing, analyzing and understanding images. Typical problems addressed by computer vision include image classification, object detection, segmentation, semantic segmentation and text explanation generation.

Recently, Artificial Intelligence (AI), and in particular deep learning algorithms have been significantly progressing in many applications in a way that exceeds human potential. Moreover, the increase in computational resources and capabilities has created an opportunity to develop Deep Learning (DL) models for accurate detection and classification of many pathologies.

In recent years, the number of diabetic patients suffering from diabetic retinopathy (DR) is dramatically increased. DR is a complication of diabetes, causing abnormalities in the retina, and in the worst case blindness. Regular DR screening is important so that timely treatment can be implemented to prevent vision loss. Early detection, which is critical for an adequate prognosis, is labor-intensive and time-consuming. This presents a challenge in areas where skilled clinical facilities are traditionally rare.

Problematic

Diabetic Retinopathy is considered to be the world's first cause of blindness. The World Health Organization estimates that 347 million people have diabetes worldwide and the number will increase to 552 million by the year 2030 [1]. DR is a time sensitive complication of diabetes that needs to be diagnosed early in order to prevent its ultimate evolution towards irrevocable vision loss. It is frequent and needs a competent specialist to detect it. This isn't always possible in resource-limited regions.

Objective of the work

We aim in our research to build an efficient model that is capable of classifying DR images into different levels, and try new preprocessing methods that will help the model to learn better. We use color fundus images of Diabetic Retinopathy from the KAGGLE website [2].

Our dissertation is structured as follows:

- **Chapter 1: State of the art** This chapter discusses healthcare, how artificial intelligence has impacted the healthcare field and will continue to do so, the state of the art of machine learning and deep learning techniques, as well as the various levels of diabetes and diabetic retinopathy.
- **Chapter 2: System design and implementation** This chapter describes the datasets and their structure, the system's overall architecture, the tools for implementation, and the code in detail.
- **Chapter 3: Experimentation and Results** This chapter presents the results of our work, discusses how specific parameters influence the obtained results, and includes a comparative section to highlight the differences between this work and previous ones.
- **Conclusion and perspectives.** In this final chapter, we summarize and review our ideas and results, as well as provide some perspectives.

Chapter 1

State of the art

1.1 Introduction

Ever since the industrial revolution, there happened a vast development in the field of technology. Many hard manual works had been replaced by technology, which helps humankind a lot. Artificial Intelligence (AI) is one of the technological innovations that happened, to replace the manual work that is done by human in various fields. Artificial Intelligence is a branch of science and technology that creates intelligent machines and computer programs to perform various tasks which requires human intelligence.

The rise of artificial intelligence has brought a positive shift in the sector by providing accurate data-driven decisions. The data from large systems is used for the early detection of chronic illnesses. These illnesses include cancer, diabetes, and cardiovascular diseases, etc. With the advent of ML/AI in the healthcare system, we expect to see much automation in clinical decision-making.

In this chapter we will investigate Artificial Intelligence in healthcare and its application in the domain, take an overview on Machine Learning and Deep Learning, we will describe diabetes and the different levels of diabetic retinopathy, and finally we will mention some previous related works.

1.2 Artificial intelligence in healthcare

Artificial intelligence (AI) is gradually transforming medical practice. AI applications are expanding into areas that were previously thought to be only the domain of human experts.

The primary aim of health-related AI applications is to analyze relationships between clinical techniques and patient outcomes [3]. AI programs are applied to practices such as diagnostics, treatment protocol development, drug development, personalized medicine, and patient monitoring and care.

What differentiates AI technology from traditional technologies in healthcare is the ability to gather data, process it, and produce a well-defined output to the end-user. AI does this through machine learning algorithms and deep learning. These processes can recognize patterns in behavior and create their own logic.

Clinical decision support and imaging analysis are currently the most common roles for AI in medical settings. Clinical decision support tools assist providers in making decisions about treatments, medications, mental health, and other patient needs by providing them with quick access to relevant information or research. AI tools are being used in medical imaging to analyze CT scans, x-rays, MRIs, and other images for lesions or other findings that a human radiologist might miss.

1.2.1 Clinical applications

1. **Cardiovascular**

   In cardiovascular medicine today ML/AI has found wide range of applications in cardiovascular drug therapy, pharmacogenomics, heart failure management, cardiovascular imaging, and diagnostics. AI can provide tools to apply precision medicine and big data in cardiovascular medicine therefore, augmenting the effectiveness of the cardiologist.

   For example, a Mayo Clinic study applied AI techniques to a new screening tool for left ventricular dysfunction [1] in people without noticeable symptoms [4] [5].

Figure 1.1 – ECG signals generated from the single lead on an Apple Watch.

2. **Gastroenterology**

   AI has the potential to play a role in many aspects of gastroenterology [6]. Endoscopic [2] exams such as esophagogastroduodenoscopies (EGD) [3] [7] and colonoscopies rely on the detection of abnormal tissue as quickly as possible. By incorporating AI into these endoscopic procedures, clinicians can more quickly identify diseases, assess their severity, and visualize blind spots. Early trials of AI detection systems for early gastric cancer have shown sensitivity comparable to expert endoscopists [8] .

1. LV dysfunction occurs when the left ventricle of the heart is either defective or damaged
2. An endoscopy is a procedure used in medicine to look inside the body
3. EGD is an endoscopic procedure that allows the doctor to examine your esophagus, stomach and duodenum

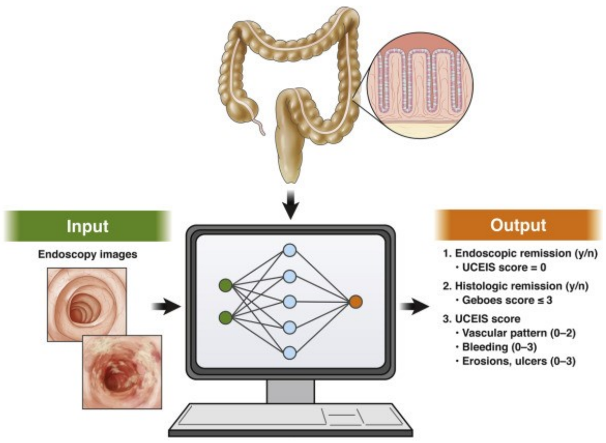


Figure 1.2 – AI application in gastroenterology.

3. **Radiology**

   AI is being studied within the field of radiology to detect and diagnose diseases through Computerized Tomography (CT) [4][9] and Magnetic Resonance (MR) Imaging. It may be particularly useful in settings where demand for human expertise exceeds supply, or where data is too complex to be efficiently interpreted by human readers.

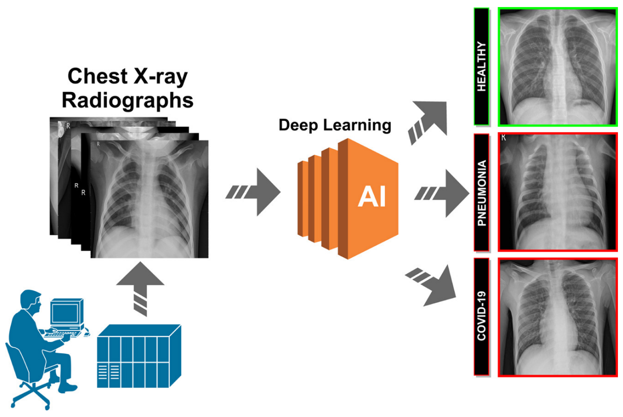


Figure 1.3 – AI application in radiology.

4. **Ophthalmology**

   Ophthalmology, especially retina, is an area AI plays a prejudicial role given the use of a multitude of digital images.
   Two hundred forty-three articles of AI application in diagnosing ophthalmological diseases have been published (search by PubMed, Sep 20, 2018)[10] [11]. Among them, the most intensively studied are DR, glaucoma, AMD, and cataract.

4. refers to a computerized x-ray imaging procedure in which a narrow beam of x-rays is aimed at a patient and quickly rotated around the body, producing signals that are processed by the machine's computer to generate cross-sectional images, or "slices"

1.3 Machine Learning

Machine learning is a subset of the Artificial Intelligence (AI) field in which we can build a model or algorithm for specific purposes based on given data using special techniques such as programs and statistical computation [12]. The main types of ML include:

- Supervised learning.
- Unsupervised learning.
- Reinforcement learning.
- Semi-supervised learning.

1.3.1 Supervised learning

Algorithms require all the instances of the training set to be labeled. From this previous knowledge the algorithm is able to learn and generalize, being able to predict new never seen before samples. From a probability perspective these types of algorithms learn a conditional distribution, i.e. P(c—X), being c the class to predict and X the sample [13].

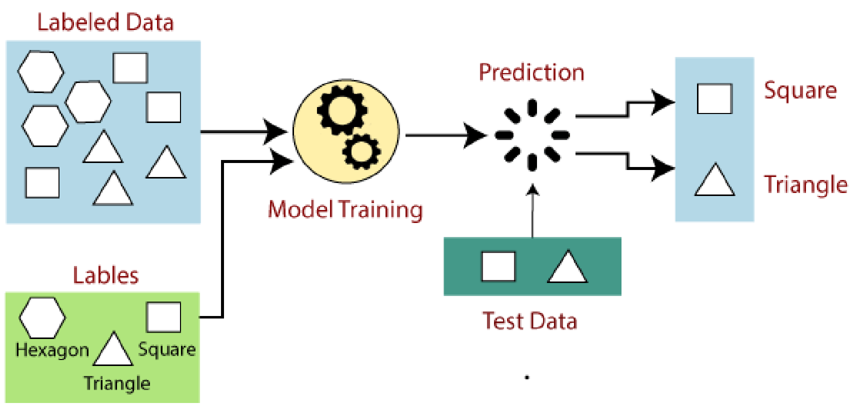


Figure 1.4 – Example of supervised learning.

1.3.2 Unsupervised learning

Unsupervised learning, also known as unsupervised machine learning, uses machine learning algorithms to analyze and cluster unlabeled datasets. These algorithms discover hidden patterns or data groupings without the need for human intervention. Its ability to discover similarities and differences in information make it the ideal solution for exploratory data analysis, cross-selling strategies, customer segmentation, and image recognition [14].

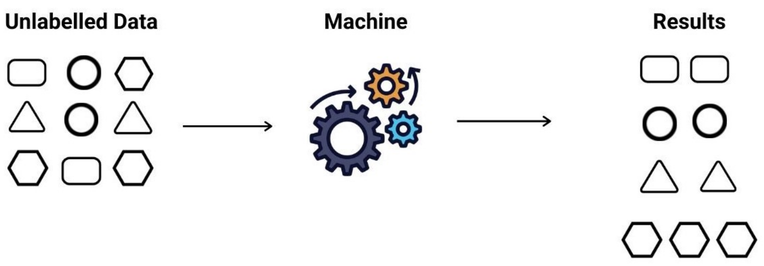


Figure 1.5 – Example of unsupervised learning.

1.3.3 Reinforcement learning

Algorithms give models the capacity of learning from environment, i.e. accumulating experience from its interaction with the surroundings. Such models are goal oriented, having an internal representation of the environment that is updated periodically with the objective of maximizing gain [15].

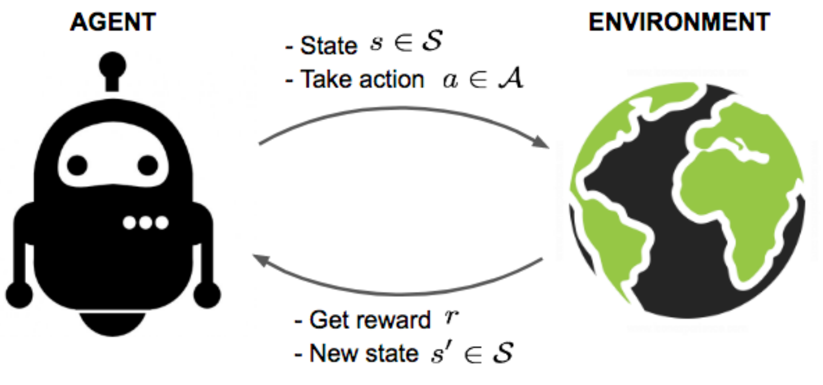

AGENT
ENVIRONMENT
- State $s \in \mathcal{S}$
- Take action $a \in \mathcal{A}$
- Get reward r
- New state $s' \in \mathcal{S}$

Figure 1.6 – Example of Reinforcement learning.

1.3.4 Semi-supervised learning

There are techniques which combine supervised and unsupervised learning to build a model which is able to predict a large number of unlabeled data. Supervised learning uses a small number of label data initially to train the model with a known target to build the model. Unsupervised learning is used by unlabeled data for the same model which is trained prior to predicting this kind of data. This operation is named semi supervised learning. This technique is used in web mining, text mining and video mining in which there are huge numbers of unlabeled data and a small number of labeled data [16].

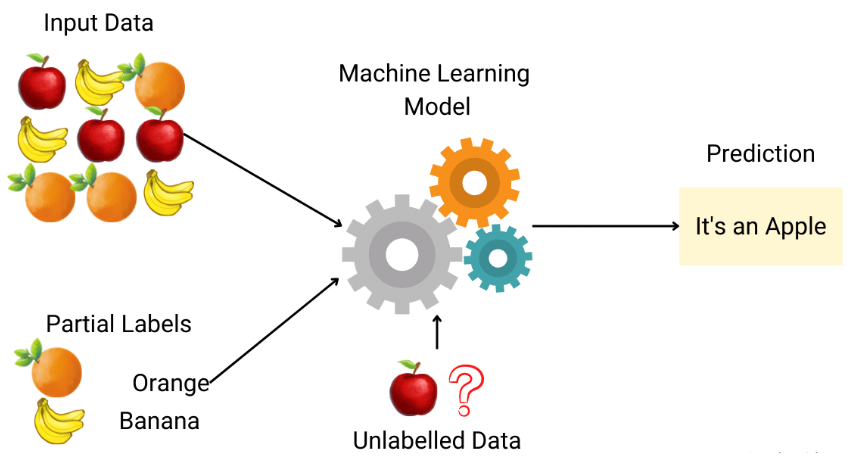


Figure 1.7 – Example of semi-supervised learning.

1.4 Deep Learning

1.4.1 Definition

Deep learning is a type of Machine learning, which deals with the training of neural networks. It is the most powerful technique in classification, which justifies its presence in almost all applications and fields that use machine learning.
The main definition of DL is that it is a Neural Network with many hidden layers, so "deep" here refers to the depth of layers consisting more than 2 hidden layers. It provides automatic feature extraction by determining the properties of the input data which can be used as a pointer to label the input data accurately. Each layer extracts features from the output of previous layer [17].

1.4.2 Types of Deep Learning architectures

Neural networks are the architectures lying under the term of deep learning. They are directed graphical models with a defined architecture formed by their building block, the neuron. We can differentiate between three typical base architectures: fully connected, convolutional and recurrent neural networks [18] [19].

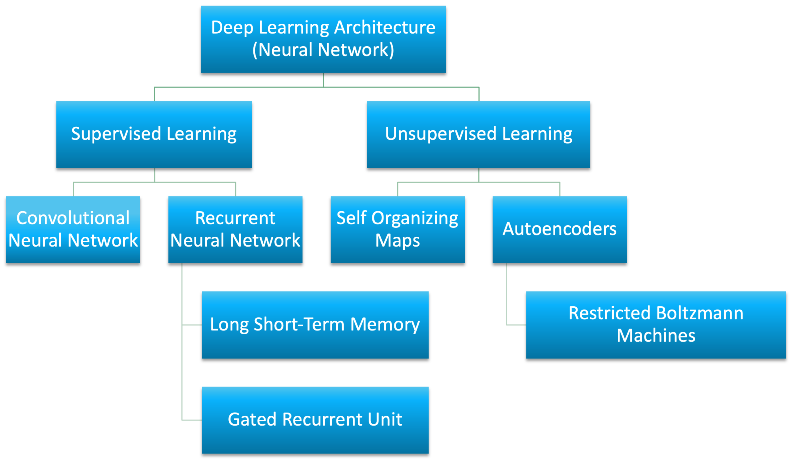


Figure 1.8 – Types of DL architectures

1.5 Artificial neural networks

1.5.1 Artificial neuron

Artificial neurons are modeled after the hierarchical arrangement of neurons in biological sensory systems. It takes as input a number of variables (X), each input of artificial neuron is associated to a weight w representing the value of the connection.

An activation function f transforms the weighted sum of the input variables and their weights $\sum_{i=1}^{n}(wi \times xi)$ to a value. The value will be transmitted to the output layer to be compared with a threshold value, and then provide an output response [20] [21].

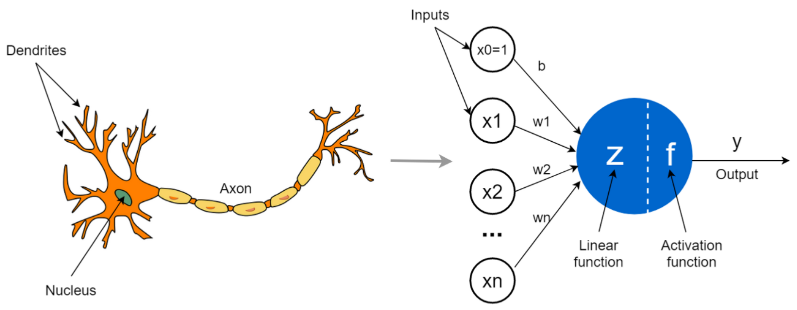


Figure 1.9 – Biological neuron to an artificial neuron.

1.5.2 Artificial neural network

The ANN (multi-layer) consists of an input layer to receive the external data to perform pattern recognition, an output layer which gives the problem solution, and a hidden layer (an intermediate layer) which separates the other layers [22].

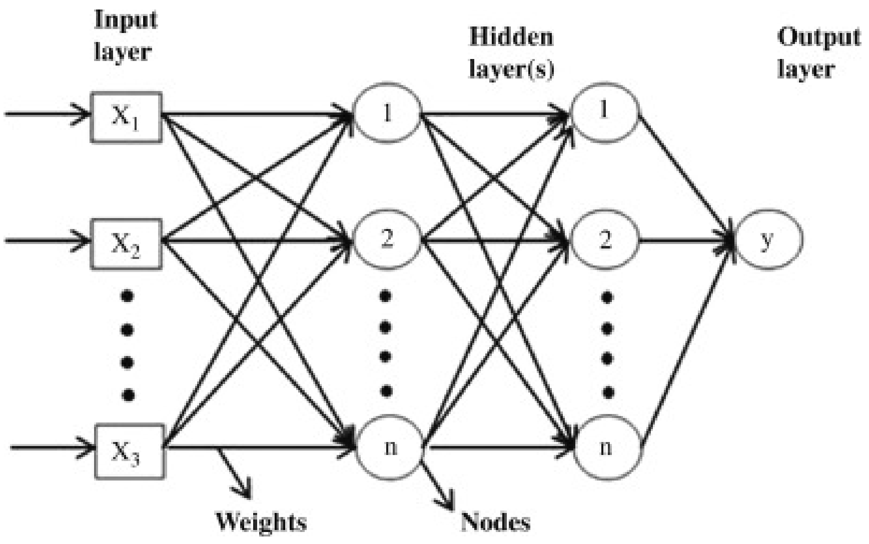


Figure 1.10 – An artificial neural network.

1.6 Convolutional neural networks

1.6.1 Definition

A convolutional neural network (CNN) is a type of artificial neural network used primarily for image recognition and processing, due to its ability to recognize patterns in images. The first CNN was created by Yann LeCun; at the time, the architecture focused on handwritten character recognition, such as postal code interpretation [23].

1.6.2 Main operations of convolution

1. **Edge Detection** is known as the operation of specifying a method for finding vertical or horizontal edges in images using a filter [24].

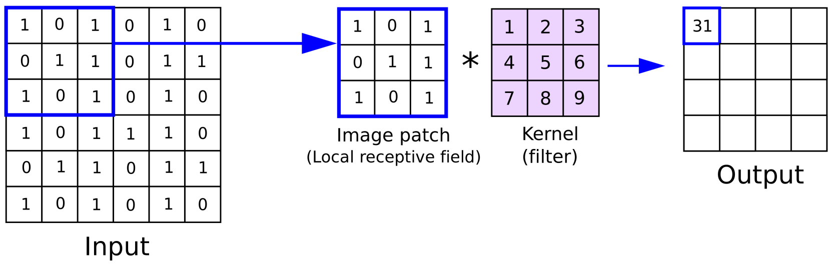


Figure 1.11 – Edge detection using a convolution operator.

According to Figure 1.11, a 3*3 kernel is constructed (also known as a filter or vertical detector) and a 6*6 image is taken. The convolution operation (*) occurs by applying the element product followed by summation. The first element on the top right corner will be:

$$1*1+0*2+1*3+0*4+1*5+1*6+1*7+0*8+1*9=31 \tag{1.1}$$

We get the second element on the 4 * 4 output matrix by moving the blue box one step to the right and performing the same convolution operation. This is done for each element.

2. **Padding**

   It refers to the number of pixels added to an image when it is being processed by the kernel of a CNN [25]. It is used to avoid image downsizing when applying the convolution operator every time (edge detection), and also to limit throwing away a large amount of information near the image's edges [24].
   We used the following equation to specify the dimension of the output matrix:

$$(n+2p-f+1)\times(n+2p-f+1) \tag{1.2}$$

   where n is the input image dimension, p is the padding and f is the filter.
   In order to verify whether padding is to be used, we use:

   - **Valid convolution:** there is no padding and the output dimension will be:

$$(n-f+1)\times(n-f+1) \tag{1.3}$$

   - **Same convolution:** the output size is equal to the input size:

$$p=\frac{f-1}{2} \tag{1.4}$$

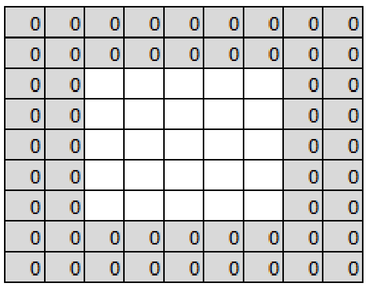


Figure 1.12 – Zero padding of size 2.

3. **Stride**

   It is a parameter of the neural network's filter representing the value by which the kernel slides over the input data. By default, it is 1 [26]. The output dimension is:

$$(\frac{n+2p-f}{s}+1)\times(\frac{n+2p-f}{s}+1) \tag{1.5}$$

   where n is the input image dimension, p is the padding, f is the filter, and s is the stride.

1.6.3 Types of layers in a CNN

1. **Convolutional layer**

   A convolutional layer is the main building block of a CNN. It contains a set of filters (or kernels), and parameters of which are to be learned throughout the training. The size of the filters is usually smaller than the actual image. Each filter convolves with the image and creates an activation map [27].

   Every component of the activation map can be thought to be the output of a neuron. Therefore each neuron is connected to a small local region in the input image, and the size of the area equals the size of the filter. Every one of the neurons calculates convolutions with small portions in LeCun et al. (2010), as shown in Eq. (2.6).

$$y_i = b_i + \sum_{x_i \epsilon x} W_{ij} * x_i \tag{1.6}$$

   Where $y_i \epsilon Y$, $i = 1, 2, \ldots D$. D is the depth of the convolutional layer. Each filter W_{ij} is a 3D matrix of size [F × F × CX]. Its size is determined by a selected receptive field (F), and its feature-map input's depth (CX).

   The main hyper-parameters of Convolution layer are size and number of filters, padding, stride, and a activation function (linear and non-linear).

2. **Non-linearity layer (activation layer)**

   This layer receives the Conv layer's feature map stack and performs the non linearity operation by adding bias and applying the activation function used for CNN.

   The activation function of artificial neuron has continuous values allowing an infinity of possible values included in an interval of [-1,1] or [0,1].

   The artificial neuron calculates the sum of the inputs and their weight $\sum_{i=1}^{n}(wi \times xi) + bias$. There are several forms of activation functions, each one is used in a specific context [28].

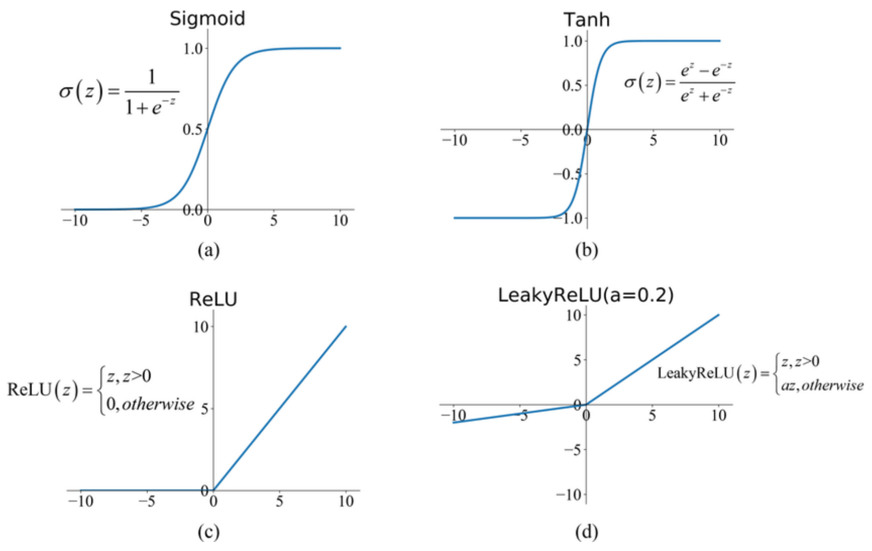


Figure 1.13 – Activation functions.

3. **Pooling Layer**

This type of layers is often placed between two convolution layers. It receives as input several feature maps, and applies to each of them the pooling operation. The pooling operation consists of reducing the size of the images, while preserving their important features [29].

There are two types of pooling layers (avg-pooling and max-pooling). Max-pooling is widely used and by applying the max pool, we slide the window of 2 $\times$ 2 regions and take the max value on each part and place them into the new matrix. Average pooling takes the average of the window of 2 $\times$ 2 regions.

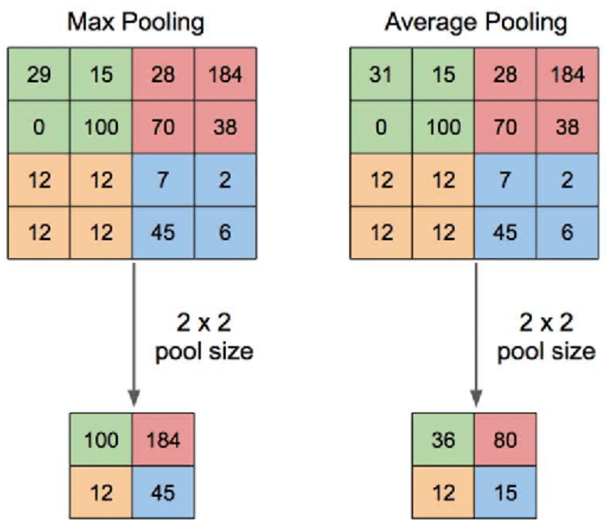


Figure 1.14 – Difference between max and average pooling.

4. **Fully Connected neural networks** are a set of dependent non-linear functions. Each individual function consists of a neuron (or a perceptron). In fully connected layers, the neuron applies a linear transformation to the input vector through a weights matrix. A non-linear transformation is then applied to the product through a non-linear activation function f [28].
   We can also visualize this layer as follows:

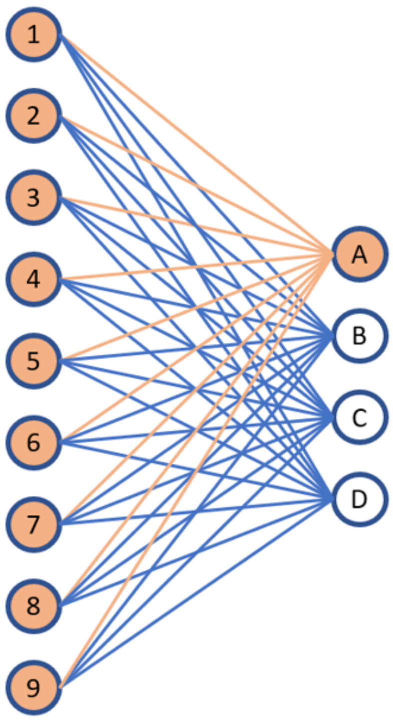


Figure 1.15 – Fully Connected layer.

   Figure 1.15 shows why we call these kinds of layers "Fully Connected" or sometimes "densely connected". All possible connections layer to layer are present, meaning every input of the input vector influences every output of the output vector. However, not all weights affect all outputs. Look at the lines between each node above. The orange lines represent the first neuron (or perceptron) of the layer. The weights of this neuron only affect output A, and do not have an effect on outputs of B, C or D.

5. **Batch normalization layer**

   Batch normalization (BN) is a layer that allows every layer of the network to do learning more independently. It is used to normalize the output of the previous layers. The activations scale the input layer in normalization. BN is used to reduce the variety of distributions in the input layer during the training process using two parameters:

   - Shifting, by subtracting the mean.

- Scaling, by dividing the batch standard. deviation

6. **Dropout layer**

   This layer is a mask that nullifies the contribution of some neurons towards the next layer and leaves unmodified all others. We can apply a Dropout layer to the input vector, in which case it nullifies some of its features, but we can also apply it to a hidden layer, in which case it nullifies some hidden neurons [30].

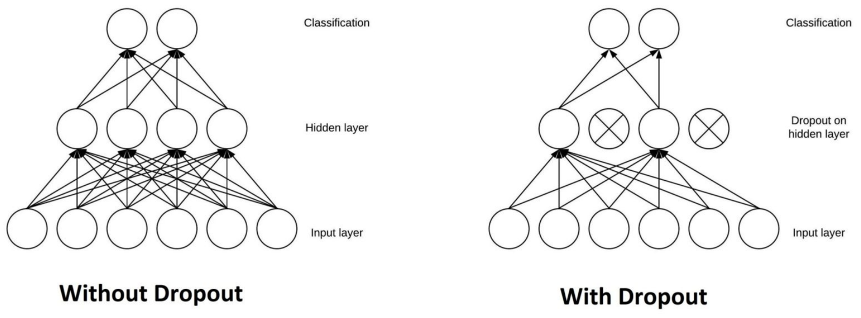


Figure 1.16 – Dropout explanation.

1.6.4 Popular CNN architectures

CNNs have been demonstrated to be very effective in solving complex classification, detection and segmentation problems.

There are an infinite number of architectures, we will discuss some of the most popular ones, such as VGG16, AlexNet, and the LetNet-5.

1. **VGG16**

   is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper "Very Deep Convolutional Networks for Large-Scale Image Recognition". The model achieves 92.7% top-5 test accuracy in ImageNet, which is a dataset of over 14 million images belonging to 1000 classes [28].

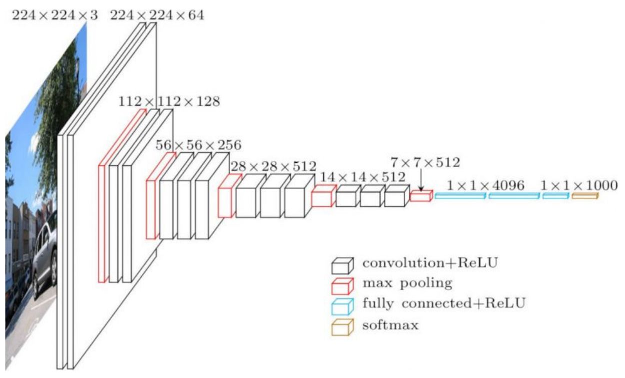
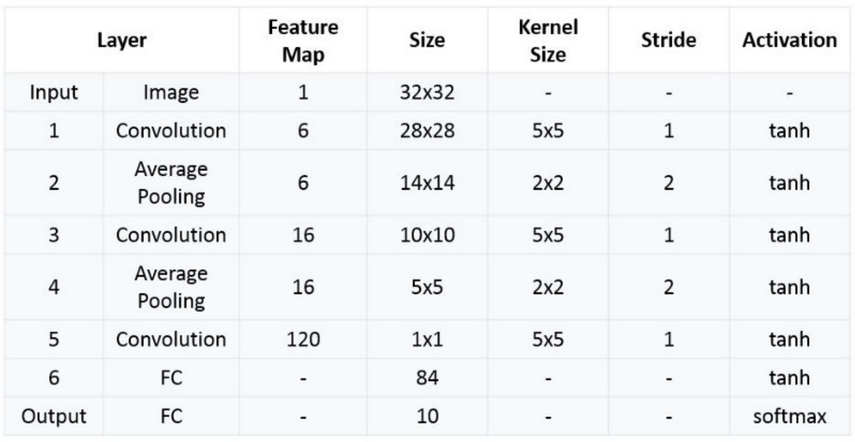


Figure 1.17 – VGG16 architecture.

2. **LetNet-5**

   Yann LeCun et al. introduced the first CNN architecture in 1995. The main goal was to use Gradient-Based learning algorithms to classify low-dimensional pattern recognition with minimal computational expenditure. The goal was to create an automatic handwritten binary classification system using the MNIST dataset, which included 50,000 training images, 10,000 validation and test images, and a grayscale resolution of 28x28 [31].



Layer		Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	32x32	-	-	-
1	Convolution	6	28x28	5x5	1	tanh
2	Average Pooling	6	14x14	2x2	2	tanh
3	Convolution	16	10x10	5x5	1	tanh
4	Average Pooling	16	5x5	2x2	2	tanh
5	Convolution	120	1x1	5x5	1	tanh
6	FC	-	84	-	-	tanh
Output	FC	-	10	-	-	softmax

Figure 1.18 – LetNet-5 Architecture Summary.

3. **AlexNet**

   Alexnet won the Imagenet large-scale visual recognition challenge in 2012. The model was proposed in 2012 in the research paper named Imagenet Classification with Deep Convolution Neural Network by Alex Krizhevsky and his colleagues.

   The Alexnet has eight layers with learnable parameters. The model consists of five layers with a combination of max pooling followed by 3 fully connected layers and they use Relu activation in each of these layers except the output layer [32].

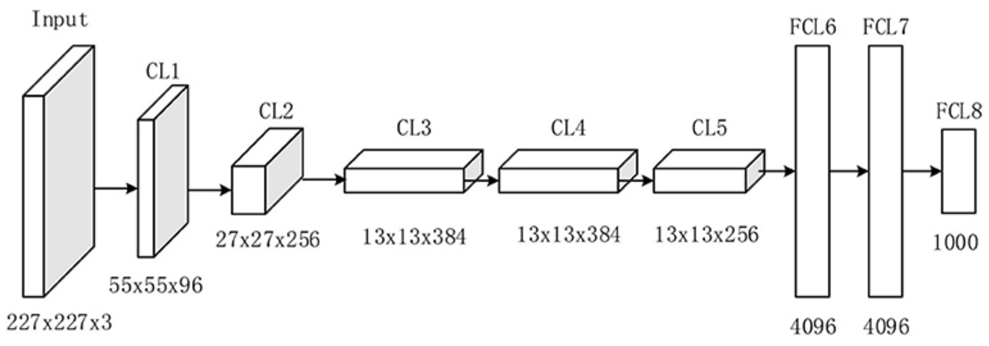


Figure 1.19 – AlexNet architecture.

1.7 Transfer learning

The reuse of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning.
Transfer learning offers a number of advantages, the most important of which are reduced training time, improved neural network performance (in most circumstances), and the absence of a large amount of data [33].

1.8 Diabetic retinopathy

1.8.1 The retina of the eye

The retina is the layer at the very back of your eyeball. It captures the light that enters your eye and helps translate it into the seen images [34]. Light passes through the lens at the front of the eye and hits the retina. Photoreceptors[5] change light energy into an electrical signal. This signal travels through the optic nerve and into the brain to become the picture of the seen world.

5. cells inside the retina that react to light

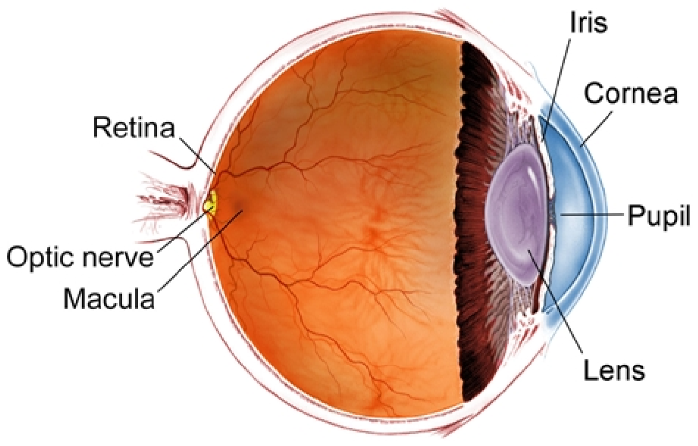


Figure 1.20 – The structure of the humain eye.

1.8.2 Diabetes

Diabetes mellitus is a disorder in which the body does not produce enough or respond normally to insulin, causing blood sugar (glucose) levels to be abnormally high [34].
Such high blood glucose levels can damage blood vessels, nerves, and even organs , increasing the probability in diabetic patients of developing other derived diseases, like in the retina (diabetic retinopathy), kidneys (diabetic nefropathy), and nervous system (diabetic neuropathy)
There are three main types of diabetes: type 1, type 2, and gestational diabetes (diabetes while pregnant) [35].

- **Type 1 diabetes** This type is an autoimmune disease, meaning your body attacks itself. In this case, the insulin-producing cells in your pancreas are destroyed. Up to 10% of people who have diabetes have Type 1. It's usually diagnosed in children and young adults (but can develop at any age).
- **Type 2 diabetes** With this type, the body either doesn't make enough insulin or the body's cells don't respond normally to the insulin. This is the most common type of diabetes. Up to 95% of people with diabetes have Type 2. It usually occurs in middle-aged and older people.
- **Gestational diabetes** This type develops in some women during their pregnancy. Gestational diabetes usually goes away after pregnancy. However, if you have gestational diabetes you're at higher risk of developing Type 2 diabetes later on in life.

1.8.3 Diabetic retinopathy

Diabetic Retinopathy is an associated disease derived from diabetes. It is caused by the damage of the small blood vessels of the retina. Due to diabetes disease related secondary effects, retinal blood vessels can break down, leak or become blocked, affecting the transport of nutrients and oxygen to parts of the retina, causing impaired vision over time. Due to the blockages, abnormal blood vessels can grow on the retina surface, causing an increment of the probability of bleeding and liquid leakages. Such structural changes can result initially in vision blurring and in last stages, even in retinal detachment and/or glaucoma[6] [36, 37].

Diabetic retinopathy symptoms:

— Floaters.

— Spots in vision.

— gradually worsening vision.

— sudden vision loss.

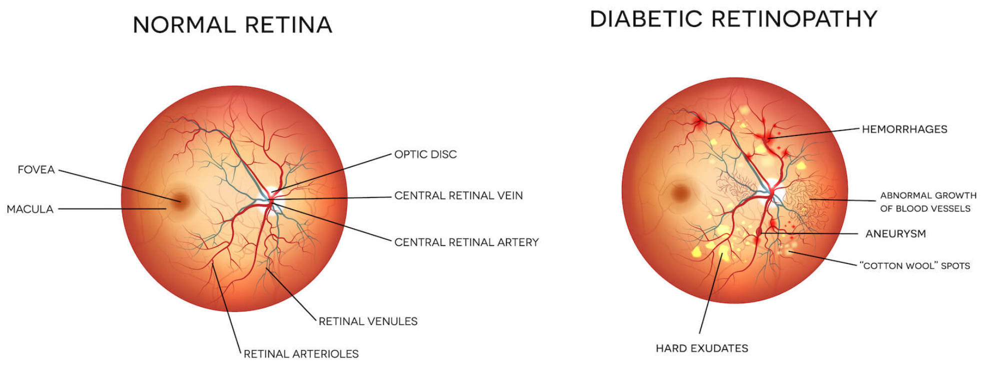


Figure 1.21 – The difference between a normal retina and diabetic retinopathy retina

DR is detected by the appearance of different types of lesions on a retina image. These lesions are microaneurysms (MA), haemorrhages (HM), soft and hard exudates (EX) [38][39].

- **Microaneurysms (MA)** is the earliest sign of DR that appears as small red round dots on the retina due to the weakness of the vessel's walls. The size is less than 125 µm and there are sharp margins.
- **Haemorrhages (HM)**appear as larger spots on the retina, where its size is greater than 125 m with an irregular margin. There are two types of HM, which are flame (superficial HM) and blot (deeper HM).

6. Glaucoma is a common eye condition where the optic nerve, which connects the eye to the brain, becomes damaged.

- **Hard exudates** appear as bright-yellow spots on the retina caused by leakage of plasma.
- **Soft exudates** (also called cotton wool) appear as white spots on the retina caused by the swelling of the nerve fiber. The shape is oval or round.

Stages of diabetic retinopathy

There are three main stages of diabetic retinopathy which are:

- **Stage 1: background retinopathy**

  The first stage is also called Mild Nonproliferative Retinopathy. It means that tiny bulges (microaneurysms) have appeared in the blood vessels in the back of your eyes (retina), which may leak small amounts of blood [40].

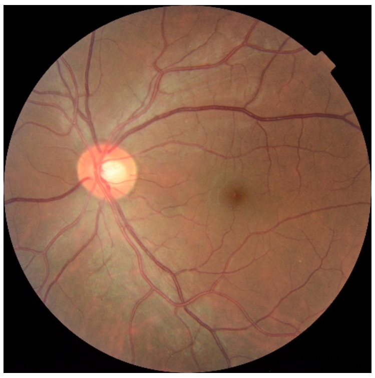


Figure 1.22 – Mild diabetic retinopathy.

- **Stage 2: pre-proliferative retinopathy**

  The second stage is also called Moderate Nonproliferative Retinopathy. At this stage, the blood vessels in the retinas swell. They may not carry blood as well as they used to. These things can cause physical changes to the retina. These changes can lead to diabetic macular edema (DME). This happens when blood and other fluids build up in a part of the retina called the macula [40].

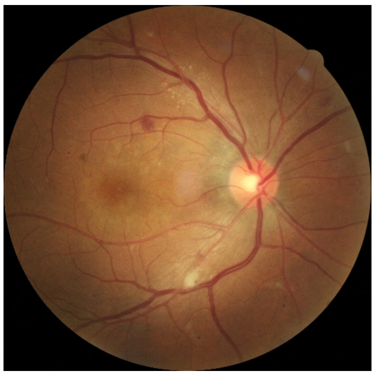


Figure 1.23 – Moderate diabetic retinopathy.

- **Stage 3: proliferative retinopathy** There are two levels in this stage severe retinopathy and Proliferative Diabetic Retinopathy (PDR) [40].

  - Severe retinopathy:

  The blood vessels become even more blocked. This means even less blood goes to the retinas. Because of this, scar tissue forms. The lack of blood triggers a signal to the retinas to create new blood vessels. If the blood vessels close off completely, it's called macular ischemia. This can lead to blurry vision with dark spots some people describe as "floaters ."

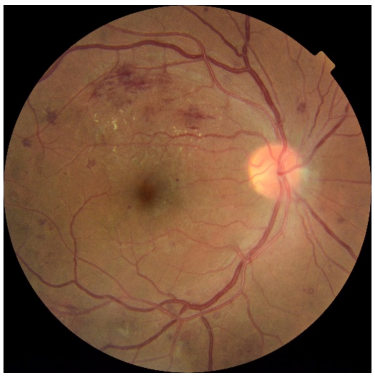


Figure 1.24 – Severe diabetic retinopathy.

- Proliferative Diabetic Retinopathy:

In this advanced level, new blood vessels grow in the retinas and into the gel-like fluid that fills the eyes. This growth is called neovascularization. These vessels are thin and weak. They often bleed which causes scar tissue.

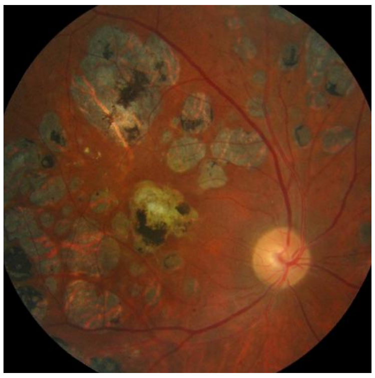


Figure 1.25 – Proliferative diabetic retinopathy.

1.8.4 Fundus photography

Color Fundus Retinal Photography uses a fundus camera to record color images of the condition of the interior surface of the eye, in order to document the presence of disorders and monitor their change over time. A fundus camera or retinal camera is a specialized low power microscope with an attached camera designed to photograph the interior surface of the eye, including the retina, retinal vasculature, optic disc, macula, and posterior pole [41].

Figure 1.26 – The machine used to get retina images.

1.8.5 Related works

In this subsection we will mention some of the previous works on classifying retina images using different ML and DL techniques.

- **A Deep Learning Approach for the Diabetic Retinopathy Detection**
  M.R. Sebti et al.[42] proposed two models binary classification and a multi class classification(No DR, moderate, severe). In the preprocessing stage, Gaussian filtering and image resizing to (224x224x3) were performed. The CNN architecture is composed firstly of convolutional layers each followed by a max pooling layer. Then, a first fully connected layer followed by a drop-out layer, and a second fully connected layer with Softmax as an activation function, are added. The proposed model was trained on the first structure. The training accuracy and training validation were found to be 96.93% and 95.08%, respectively. and than it was trained on the second structure with 32 epoch, the results were 93.06% for training accuracy, 81.12% for training validation.

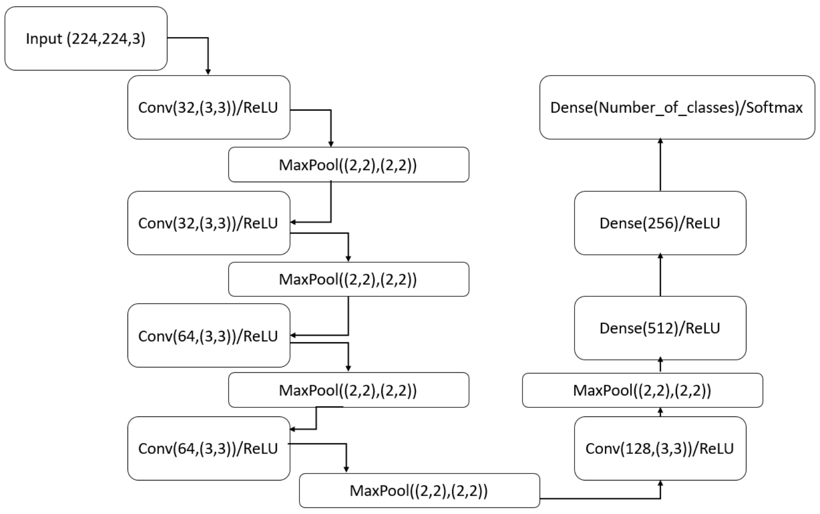


Figure 1.27 – The proposed architecture.

- **Feature Visualisation of Classification of Diabetic Retinopathy Using a Convolutional Neural Network**
  The authors of [43] proposed a method based on a CNN to classify images from the Kaggle dataset into five DR stages. In the preprocessing stage, color normalization and image resizing to 512 × 512 pixels were performed. Their custom CNN architecture contained 10 CONV layers, eight max-pooling layers, and three FC layers. The SoftMax function was used as a classifier for 80,000 test images. L2 regularization and dropout methods was used in CNN to reduce overfitting. Their results had a specificity of 95%, an accuracy of 75% and a sensitivity of 30%. Unfortunately, CNN does not detect the lesions in the images, and only one dataset was used to evaluate their CNN.

- **Classification of diabetic and normal fundus images using new deep learning method**
  Esfahan *et al.* [44] used a known CNN, which is ResNet34 in their study to classify DR images of the Kaggle dataset into normal or DR image (see figure 2.30). ResNet34 is one the available pre-trained CNN architecture on ImageNet database. They applied a set of image preprocessing techniques to improve the quality of images. The image preprocessing included the Gaussian filter, weighted addition and image normalization. The image number was 35000 images and its size was (512,512) pixels. They reported an accuracy of 85% and a sensitivity of 86%.

- **SVM and Neural Network based Diagnosis of Diabetic Retinopathy**
  The authors of [45] proposed a computer-vision-based approach for the detection of diabetic retinopathy stages using color fundus images. They tried to extract features from the raw image, using the image processing techniques, and fed them to the

SVM for binary classification and achieved a sensitivity of 98%, specificity 96%, and accuracy of 97.6% on a testing set of 250 images.

- **An interpretable ensemble deep learning model for diabetic retinopathy disease classification**
  The study of Jiang *et al.* reported in [46] integrates three pretrained CNN models, namely, Inception V3, Inception-Resnet-V2 and Resnet152 to classify their own dataset as referable DR or non-referable DR. The work reaches an accuracy of 88.21% and area under the curve (AUC) of 0.946.
- **Diabetic retinopathy stage classification using convolutional neural networks**
  X. Wang et al.[46] studied the performance of the three available pretrained architectures of CNN, VGG16 , AlexNet and InceptionNet V3 , to detect the five DR stages in the Kaggle dataset [47]. The images were resized to 224 $\times$ 224 pixels for VGG16, 227 $\times$ 227 pixels for AlexNet, and 299 $\times$ 299 pixels for InceptionNet V3 at the preprocessing stage. The dataset only contains 166 images. They reported an average accuracy of 50.03% in VGG16, 37.43% in AlexNet and 63.23% in InceptionNet V3.

1.9 Conclusion

In this chapter, we covered some of the most important aspects of our current work. We detailed the basics of machine learning and deep learning, then we discussed diabetes and its complications, diabetic retinopath and its levels. Finally we mentioned some previous related works. The following chapter will describe our system design and the implementation of a new deep learning architecture for diabetic retinopathy detection.

Chapter 2

System design and implementation

2.1 Introduction

Recently, Artificial Intelligence (AI) and deep learning algorithms have been significantly progressing in many applications in a way that exceeds human potential. Moreover, the increase in computational resources and capabilities has created an opportunity to develop Deep Learning (DL) models for accurate DR detection and classification. In this chapter we present the system design, the various datasets and their different structures, the preprocessing phase, and finally our proposed CNN-architecture. We will mention the used tools, frameworks, libraries, as well as how we implemented our system.

2.2 Global architecture of the system

The process used to detect and to classify DR images using DL begins by collecting the dataset and applying the needed preprocess to improve and enhance the images. Then, it is fed to the DL model to extract the features and to classify DR images.
Figure 2.1 depictes certain steps followed by our system.

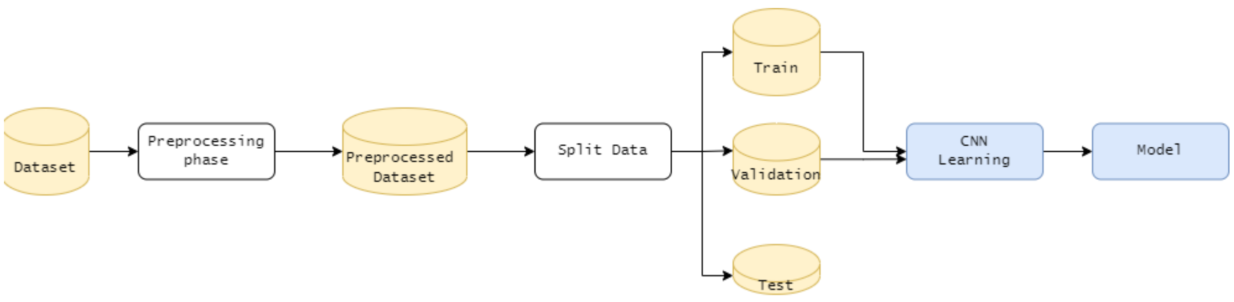


Figure 2.1 – General architecture.

2.2.1 Dataset

1. **Dataset description:** The two data sets that have been used in this study are available on [2].
   -The first original (2019) Dataset is available at APTOS 2019 Blindness Detection [48]. It contains 3662 of retina images.
   -The Second data also available on kaggle [47]. The training set contains 35126 of retina images with various resolutions, ranging from 433 × 289 pixels to 5184 × 3456 pixels.

2. **Original structure of the datasests:** The images are ranked into five different levels of diabetic retinopathy by specialists as it is shown in Figure 2.2.
   (a) **No DR** which represents **Level 0**: The patient retina has no diabetic retinopathy.
   (b) **Mild** which represents **Level 1**: The patient retina has a mild diabetic retinopathy.
   (c) **Moderate** which represents **Level 2**: The patient retina has a moderate diabetic retinopathy.
   (d) **Severe** which represents **Level 3**: The patient retina has a severe diabetic retinopathy.
   (e) **Proliferate DR** which represents **Level 4**: The patient retina has a proliferate diabetic retinopathy.

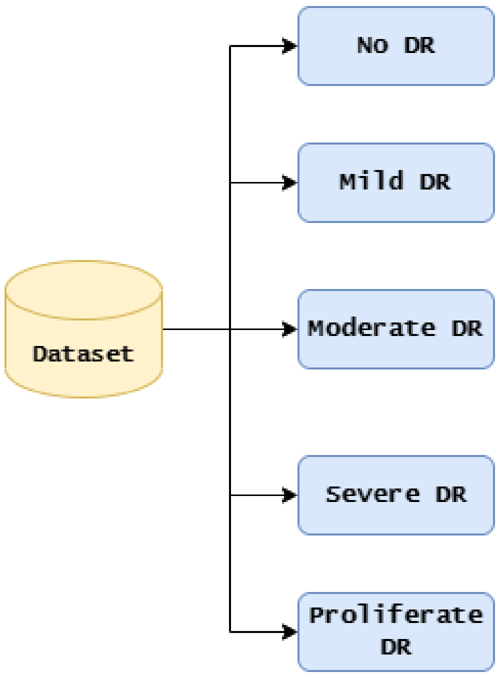


Figure 2.2 – Original Structure.

3. **The first proposed structure (Binary classification)**
   We modify the structure of the original data set by grouping all diabetic retinopathy levels (mild, moderate, sever, and proliferate) to represent class of DR. The other side contains the healthy ones which represents NO DR (see Figure 2.3).

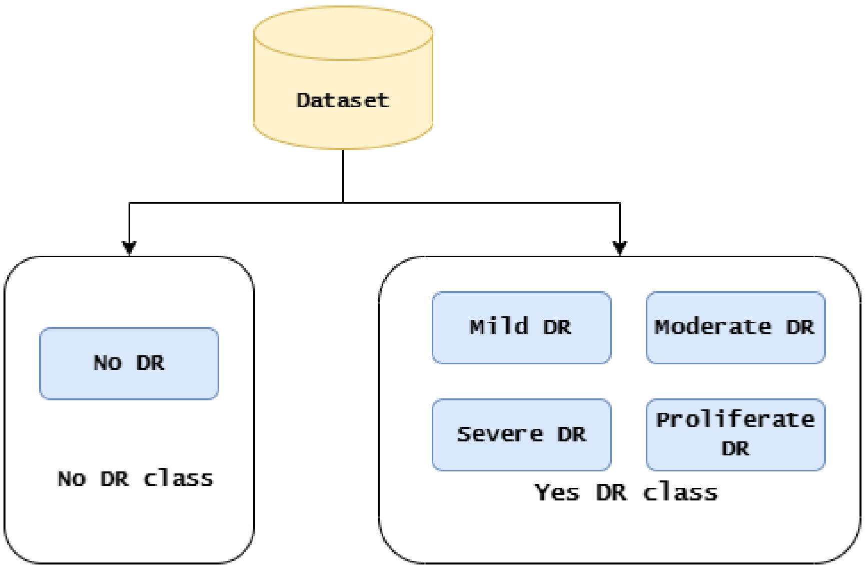


Figure 2.3 – The first proposed structure.

4. **The second structure (Multi class classification)**
   In this structure we use the second data set, we group the last two classes together due to lack of images. The structure becomes: (i) mild, (ii) moderate, (iii) sever with proliferate, and healthy ones which represents NO DR. There are four levels in this structure (see Figure 2.4).

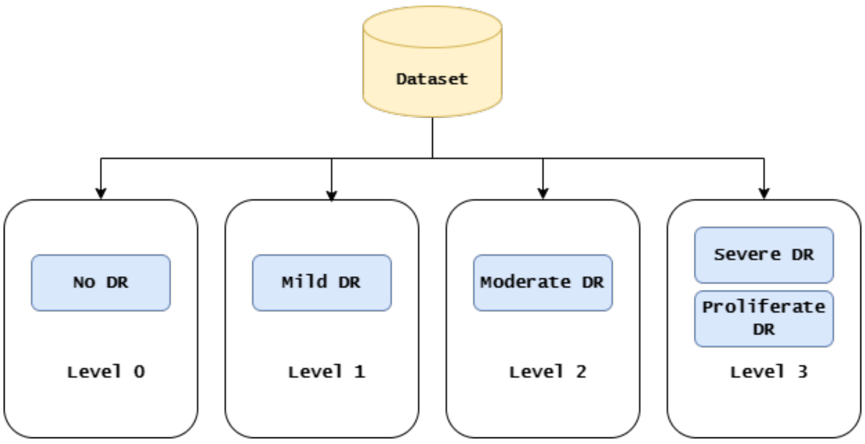


Figure 2.4 – The second proposed structure.

2.2.2 Preprocessing

1. **Resize Images Using Rescaling and Cropping**

Rescaling multiplies the height and width of the image by a scaling factor. If the scaling factor is not identical in the vertical and horizontal directions, then rescaling changes the spatial extents of the pixels and the aspect ratio. Cropping extracts a subregion of the image and preserves the spatial extent of each pixel [49] [50].
In this work, the dataset images were cropped in order to minimise the background by detecting the contour of each retina image and cropping according to its position then resized to (224x224).

2. **Gaussian Filtering**

   A Gaussian filter is a low pass filter used for reducing noise (high frequency components) and blurring regions of an image. The filter is implemented as an Odd sized Symmetric Kernel (DIP version of a Matrix) passed through each pixel of the region of interest to get the desired effect [51].
   The result of applying this filter is shown in Fig. 2.5.

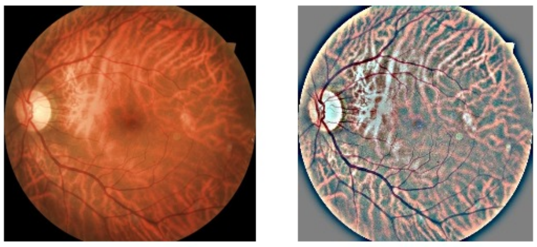


Figure 2.5 – Gaussian filtering.

3. **Histogram equalization**

   - **what's an Histogram?**
     A histogram of an image is the graphical interpretation of the image's pixel intensity values. It can be interpreted as the data structure that stores the frequencies of all the pixel intensity levels in the image.
   - **Histogram equalization** is an image processing technique that adjusts the contrast of an image by using its histogram. To enhance the image's contrast, it spreads out the most frequent pixel intensity values or stretches out the intensity range of the image. By accomplishing this, histogram equalization allows the image's areas with lower contrast to gain a higher contrast [52] [53].

   The result of applying this technique is shown in Fig 2.6.

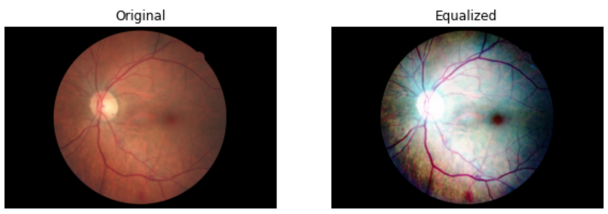


Figure 2.6 – histogram equalization.

4. **Data Augmentation**
   Image augmentations are manipulations applied to images to create different versions of similar content in order to expose the model to a wider array of training examples. For example, randomly altering rotation, brightness, or scale of an input image requires that a model consider what an image subject looks like in a variety of situations.

   In this work, we increase the number of images in real-time to improve network localization capability and reduce overfitting. During each epoch, a random rotation, shear, and zoom were performed using ImageDataGenerator by TensorFlow.

2.2.3 Splitting dataset

In Deep Learning, dataset splitting is the process of dividing a dataset into training, validation, and testing subsets. Our dataset is divided into three subsets: 70% images per class for training, 20% for validation, and 10% for testing.

2.2.4 CNN Learning

CNN is very similar to artificial neural network. The network input consists of neurons that are organized in layers with trainable weights and biases. Each neuron receives a number of inputs and then calculates the multiplication of weights in inputs and finally, using an error function which is a nonlinear conversion, generates the result. Next we will detail the parameters and hyper-parameters of each layer.

1. **CNN layers**
   The hyper-parameters of each layer are mentiend in Table 2.1.

	parameters	**Hyper-parameters**
Convolutional layer	Kernels	Kernel size, number of kernels stride, padding and activation function
Pooling layer	None	Pooling method ,filter size ,padding and strides
Fully connected layer	weights	number of weights
dropout layers	None	dropout rate

Table 2.1 – CNN layers parameters and hyper-parameters

2. **Loss function**
   The loss function in a neural network quantifies the difference between the expected outcome and the outcome produced by the machine learning model.
3. **Optimizers in CNN**
   Optimizers are algorithms or methods used to change the attributes of the neural network such as weights and learning rate in order to reduce the losses.
   Examples: RMSprop, SGD, adam, adamx, etc.

   Before we proceed to model training, some parameters must be explained due to their importance.

   - **Input shape**(width, height, channels): It is an image with the first two values refers to the width and the height of the image, and the third one refers to the image channels.
   - **Epoch**: The number of epochs is a hyperparameter that defines how many times the learning algorithm will work through the entire training dataset.
   - **Batch size**: The batch size is a hyperparameter that defines the number of samples to work through before updating the internal model parameters.
   - **Shuffle**: When this value is set to 'True', the dataset will be shuffled before training.
   - **Step per epoch**: The number of batch iterations before a training epoch is considered finished. It is generally calculated by dividing the number of samples in training subset on batch size.
   - **Learning rate**: The learning rate controls how quickly the model is adapted to the problem (usually very small).

2.2.5 Prediction

Following the training phase, the prediction phase is when a CNN model is ready to classify images. For us, the third subset from the dataset splitting will be used to test our CNN model.

2.2.6 Evaluation of a CNN model

Evaluation metrics are used to measure the quality of the statistical or deep learning model. Evaluating deep learning models or algorithms is essential for any project.
There are many different types of evaluation metrics available to test a model. These include true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). These values are used to calculate variety of performance metrics including Accuracy, Precision, and F1score according to the three equations:

$$ACCURACY = \frac{TP + TN}{TP + TN + FP + FN}$$

$$PRECISION = \frac{TP}{TP + FP}$$

$$F1SCORE = \frac{TP}{TP + \frac{1}{2}(FP + FN)}$$

2.3 Implementation of a Deep Learning architecture

We described our system design in detail in the previous section. This section's goal is to discuss tools and frameworks, implement our system design, and present code.

2.3.1 Frameworks , tools and libraries

- **Python**
  It is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
- **Google Colab**
  Google Colab was developed by Google to provide free access to GPU's and TPU's to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook [54].

Table 2.2 – Google Colab resources

	GPU	**Runtime**	**RAM**	**disk capacity**
Google Colab	K80	12h	12GB	60GB

- **OpenCv**
  Open Source Computer Vision Library is an open source computer vision and ma-

chine learning software library. Originally developed by Intel. It was later supported by Willow Garage then Itseez [55].

- **Anaconda spyder**

  Spyder, the Scientific Python Development Environment, is a free integrated development environment (IDE) that is included with Anaconda. It includes editing, interactive testing, debugging, and introspection features [56].

- **Matplotlib**

  Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy [57].

- **Keras**

  Keras is a powerful open source Python library for developing and evaluating deep learning models [58].

- **TensorFlow**

  TensorFlow is open-source library for machine learning. It provides a collection of workflows to develop and train models using Python or JavaScript, and to easily deploy in the cloud, on-prem, in the browser, or on-device [59].

- **NumPy**

  NumPy is a python fundamental scientific computing library. It provides a high performance multidimensional array object, and tools for working with these arrays [60].

- **Kaggle**

  Kaggle is an online community of data scientists and machine learning practitioners. Kaggle allows users to find and publish datasets [2].

2.3.2 Dataset preparation and preprocessing

1. **Download Dataset to Google Colab**

   - First, we used the first dataset [48] API to download the datasets from Kaggle to our Google Colab account.

   ```
   !kaggle competitions download -c aptos2019-blindness-detection
   ```

   - The second dataset [47] was downloaded to our local machine.

2. **Functions and parameters of the used filters**

   (a) **Gaussian filter cv2.addWeighted**(source1,alpha,source2,beta,gamma).

   source1 - first input.

   alpha – weight of the first array.

   source2 – second input array of the same size and channel. number as src1.

   beta - weight of the second array elements.

gamma - scalar added to each sum.

other function parameters were not modified [55].

(b) **cv2.GaussianBlur**(src, ksize, sigmaX)

src - input.

ksize - Gaussian Kernel Size.

sigmaX - Kernel standard deviation along X-axis (horizontal direction).

other function parameters were not modified.

```python
import cv2
import os
import glob
os.chdir("/content/dataset")         # dataset path example
for file in glob.glob("*.jpeg"):# look for this image format using glob library ,our dataset images are in this format
    img = cv2.imread(file)           # load an image
    gaussian = cv2.addWeighted(img, 4, cv2.GaussianBlur(img, (0,0), 10), -4, 128) #apply gaussin filtering
    cv2.imwrite(file,gaussian) # save image

```

Listing 2.1 – Gaussian filter

(c) **Histogram equalization** this technique was used only to the mild and moderate classes.

```python
import cv2
import os
import glob
os.chdir("/content/dataset")
for file in glob.glob("*.jpeg"):
    image_src = cv2.imread(file)

        r_image, g_image, b_image = cv2.split(image_src)#split the three chanels of the image

        r_image_eq = cv2.equalizeHist(r_image)#equilise the red channel
        g_image_eq = cv2.equalizeHist(g_image)#equilise the green channel
        b_image_eq = cv2.equalizeHist(b_image)#equilise the blue channel

        image_eq = cv2.merge((r_image_eq, g_image_eq, b_image_eq))
        cv2.imwrite(filename, image_eq)
```

Listing 2.2 – Histogram equalization

(d) **Cropping and resizing**

```
import PIL
import os
import cv2
from skimage import io
from matplotlib import pyplot as plt
import glob
from PIL import ImageFile


desired_size=224
os.chdir("C:/Users/kaouk/OneDrive/Desktop/retinopathy dataset")
for file in glob.glob("*.jpeg"):
    img = cv2.imread(file)
    img = cv2.copyMakeBorder(img,10,10,10,10,
    cv2.BORDER_CONSTANT,
    value=[0,0,0])
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    ret,gray = cv2.threshold(gray,10,255,cv2.THRESH_BINARY)

    contours,hierarchy = cv2.findContours(gray,
                                          cv2.RETR_EXTERNAL,
                                          cv2.
   CHAIN_APPROX_SIMPLE)
    contours = max(contours, key=cv2.contourArea)
    x,y,w,h = cv2.boundingRect(contours)

    if w>200 and h>200:
        new_img = img[y:y+h,x:x+w]
        height, width, _= new_img.shape

        if max([height, width]) > desired_size:
            ratio = float(desired_size/max([height, width]))
            new_img = cv2.resize(new_img,
                                 tuple([int(width*ratio), int(
   height*ratio)]),
                  interpolation = cv2.INTER_CUBIC)

        cv2.imwrite(file, new_img)
    else:
        print(f'No bounding for {file}')
        cv2.imwrite(file, img)
```

Listing 2.3 – Cropping and resizing

3. **Preparing Dataset**

(a) **Dividing data into categories**

We divide data to five classes according to the labels file.

```python
from __future__ import print_function
import pandas as pd
import shutil
import os
import sys

labels = pd.read_csv(r'/content/train.csv')#csv file path of
   labels

train_dir =r'/content/train_images' #dataset path
DR = r"/content/retinopathyD/DR_" #destination file path
if not os.path.exists(DR):
    os.mkdir(DR)

for filename, class_name in labels.values:
    # Create subdirectory with `class_name`
    if not os.path.exists(DR + str(class_name)):
        os.mkdir(DR + str(class_name))
    src_path = str(train_dir) + '/' + str(filename) + '.png'
    dst_path = DR + str(class_name) + '/' + str(filename) + '.
   png'
    try:
        shutil.copy(src_path, dst_path)
        print("sucessful")
    except IOError as e:
        print('Unable to copy file {} to {}'
              .format(src_path, dst_path))
    except:
        print('When try copy file {} to {}, unexpected error:
   {}'
              .format(src_path, dst_path, sys.exc_info()))

```

(b) **Splitting dataset**

We have used split_folder.ratio in order to split our dataset into 3 subsets as we mentioned before. split_folder.ratio parameters:

splitfolders.ratio(input_folder,output,seed,ratio)

```python
import splitfolders
input_folder="/content/dataset" #dataset path
output="/content/newDataset"    #splitted dataset new path
splitfolders.ratio(input_folder,output,
seed=1337,# set seed value for shuffling the items.
ratio=(.7,.2,.1)) # 70% for training 20% for validation  10% for
```

```
    testing
```

Listing 2.4 – Splitting dataset

Now we have completed the preprocessing phase and splitted dataset, we will prepare it for the CNN Model.

```
DATADIR = "/content/DataSplitedBinary/train"
TESTDIR = "/content/DataSplitedBinary/val"
LABELS = ["DR_0","DR_1"]
X_TRAIN = []
Y_TRAIN = []
X_VAL=[]
Y_VAL=[]
#Add train images to a numpy array
for label in LABELS:
        path = os.path.join(DATADIR, label)
        class_num = LABELS.index(label)
        for img in os.listdir(path):
            try:
                img_array = cv.imread(os.path.join(path, img))
                new_array = cv.resize(img_array, (224, 224))
                X_TRAIN.append(new_array)
                Y_TRAIN.append(class_num)
            except Exception as e:
                pass
#Add Validation images to a numpy array
for label in LABELS:
        path = os.path.join(TESTDIR, label)
        class_num = LABELS.index(label)
        for img in os.listdir(path):
            try:
                img_array = cv.imread(os.path.join(path, img))
                new_array = cv.resize(img_array, (224, 224))
                X_VAL.append(new_array)
                Y_VAL.append(class_num)
            except Exception as e:
                pass
```

Listing 2.5 – Dataset preparation

2.3.3 Building the CNN Model

1. **Import libraries and modules**

   For building a CNN with keras we need to import libraries first.

```
import keras,os
from keras.models import Sequential
```

```python
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten,Dropout
from keras.preprocessing.image import ImageDataGenerator
import numpy as np
from keras.models import Sequential
```

Listing 2.6 – necessary imports for building a CNN Model

2. **CNN parameters initialization**

   Since we have different data structures, we will delay the initialization to the next chapter, in the next chapter we will give details of initialization for each structure.

3. **Creating Diabetic Retinopathy Detection CNN Model**

   After trying many CNN configurations, the one that will be described had the best results.

   (a) **Used CNN Model**:

```python
    model = Sequential()

```

Listing 2.7 – Sequential model

The sequential model is a way of creating a new deep learning models where an instance of the Sequential class is created and model layers are added to it.

   (b) **Used CNN Architecture**

```python
#Layer number 1
model.add(Conv2D(input_shape=(224,224,3),filters=32,kernel_size
    =(5,5),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.2))
#Layer number 2
model.add(Conv2D(filters=32, kernel_size=(5,5), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.2))
#Layer number 3
model.add(Conv2D(filters=32, kernel_size=(5,5), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.2))
#Layer number 4
model.add(Conv2D(filters=32, kernel_size=(5,5), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 5
model.add(Conv2D(filters=64, kernel_size=(5,5), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
```

```python
#Layer number 6
model.add(Conv2D(filters=64, kernel_size=(5,5), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
model.add(Dropout(0.2))
#Layer number 7
model.add(Conv2D(filters=128, kernel_size=(5,5), padding="same",
     activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 8
model.add(Conv2D(filters=256, kernel_size=(5,5), padding="same",
     activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 9
model.add(Flatten())
model.add(Dense(units=512,activation="relu"))
#Layer number 10
model.add(Dense(units=256,activation="relu"))
model.add(Dropout(0.2))
#Layer number 11
model.add(Dense(number_of_classes, activation="softmax"))
```

Listing 2.8 – Diabetic retinopathy detection CNN model architecture

(c) **Used optimizer**

```python
from keras.optimizers import Adam
opt = Adam(lr=1e-5)
```

Listing 2.9 – Optimizer used

An optimizer is one of the two arguments required for compiling a Keras model, the optimizer controls learning rate, We have used adam as optimizer with a learning rate= 0.00001.

(d) **Used loss function**
The loss function measures performance of a classification model. We have used BinaryCrossentropy() for the binary classification and CategoricalCrossentropy() for multi class classification.

4. **Compiling CNN Model** We used the code below to compile our model.

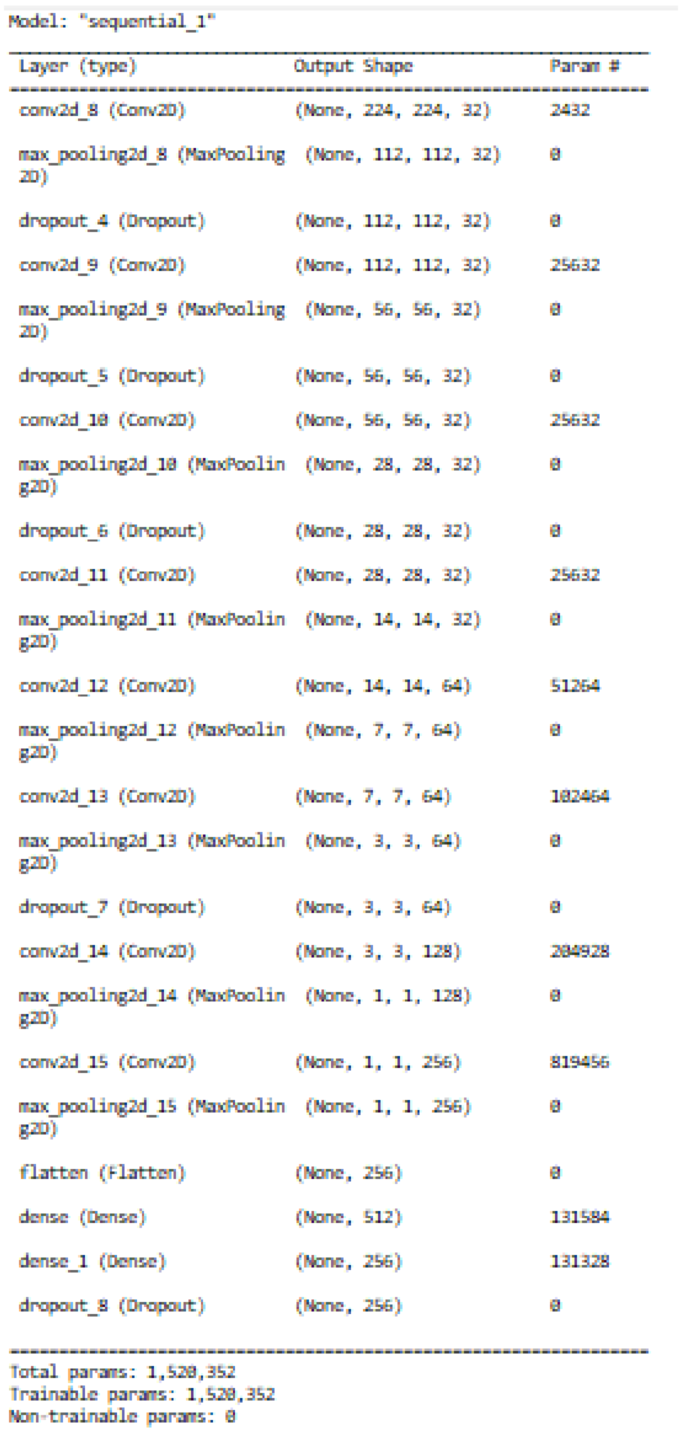
```python
from keras.optimizers import Adam,SGD,Adamax
opt = Adam(learning_rate=1e-3)
model.compile(optimizer=opt, loss=keras.losses.
    categorical_crossentropy, metrics=['accuracy','Precision','
    FalseNegatives','FalsePositives','TrueNegatives','TruePositives
    '])
```

5. **Model summary**

   Once our model is ready, we can call summary() method to display its contents see Figure 2.7.

```
model.summary()
```

Listing 2.10 – Model summary



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_8 (Conv2D)           (None, 224, 224, 32)      2432

 max_pooling2d_8 (MaxPooling  (None, 112, 112, 32)     0
 2D)

 dropout_4 (Dropout)         (None, 112, 112, 32)      0

 conv2d_9 (Conv2D)           (None, 112, 112, 32)      25632

 max_pooling2d_9 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 dropout_5 (Dropout)         (None, 56, 56, 32)        0

 conv2d_10 (Conv2D)          (None, 56, 56, 32)        25632

 max_pooling2d_10 (MaxPoolin  (None, 28, 28, 32)       0
 g2D)

 dropout_6 (Dropout)         (None, 28, 28, 32)        0

 conv2d_11 (Conv2D)          (None, 28, 28, 32)        25632

 max_pooling2d_11 (MaxPoolin  (None, 14, 14, 32)       0
 g2D)

 conv2d_12 (Conv2D)          (None, 14, 14, 64)        51264

 max_pooling2d_12 (MaxPoolin  (None, 7, 7, 64)         0
 g2D)

 conv2d_13 (Conv2D)          (None, 7, 7, 64)          102464

 max_pooling2d_13 (MaxPoolin  (None, 3, 3, 64)         0
 g2D)

 dropout_7 (Dropout)         (None, 3, 3, 64)          0

 conv2d_14 (Conv2D)          (None, 3, 3, 128)         204928

 max_pooling2d_14 (MaxPoolin  (None, 1, 1, 128)        0
 g2D)

 conv2d_15 (Conv2D)          (None, 1, 1, 256)         819456

 max_pooling2d_15 (MaxPoolin  (None, 1, 1, 256)        0
 g2D)

 flatten (Flatten)           (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dense_1 (Dense)             (None, 256)               131328

 dropout_8 (Dropout)         (None, 256)               0

=================================================================
Total params: 1,520,352
Trainable params: 1,520,352
Non-trainable params: 0
```

Figure 2.7 – Model summary

6. **Training CNN Model**

   We used fit generator, ModelCheckpoint, and EarlyStopping with different arguments depending on the structure to train the model, in this section, we will explain the role of each argument and other functions.

```python
from keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("modelNAME.h5", monitor='val_accuracy',
    verbose=1, save_best_only=True, save_weights_only=False, mode='
   auto',period=1)
early = EarlyStopping(monitor='monitor', min_delta=0, patience=
   Number_of_patience, verbose=0, mode='auto')
hist = model.fit_generator(X_TRAIN,Y_TRAIN,epochs=200,
   validation_data=(X_VAL,Y_VAL),epochs=100,shuffle= True,
   class_weight=class_weights,callbacks=[checkpoint,early])
```

Listing 2.11 – Fitting model

- **ModelCheckpoint**

  tf.keras.callbacks.ModelCheckpoint is used in conjunction with training using model.fit() to save a model [61].

  The arguments of ModelCheckpoint are:

  — **Filepath :** string or PathLike, path to save the model file.

  — **Monitor:** The metric name to monitor for example prefix the name with "val_accuracy" to monitor validation metrics.

  — **Verbose:** Verbosity mode, 0 or 1. Mode 0 is silent, and mode 1 displays messages when the callback takes an action.

  — **Save_best_only:** if save_best_only=True, it only saves when the model is considered the "best" and the latest best model according to the quantity monitored will not be overwritten.

  — **Mode:** one of 'auto', 'min', 'max'.

  — **Save_weights_only:** if True, then only the model's weights will be saved.

- **EarlyStopping**

  It is a feature that enables the training to be automatically stopped when a chosen metric has stopped improving. It is form of regularization used to avoid overfitting.

  The most important argument in using EarlyStopping is Patience, which is the number of epochs without improvement after which training will be early stopped [62].

  — **How EarlyStopping works ?**

  Early stopping monitors the evolution of the given metric after every epoch.

If the metric has not improved for Patience epochs in a row, the experiment is early stopped.

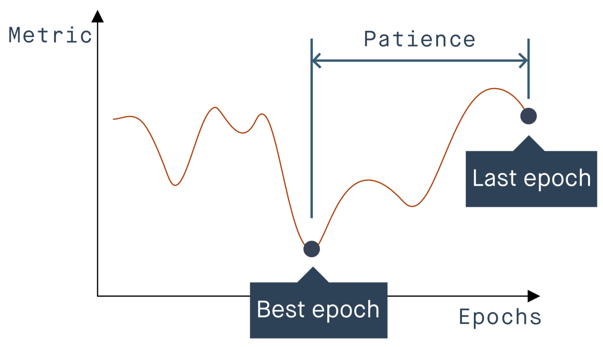


Figure 2.8 – Early stopping based on the metric loss

- **model.fit_generator** This function will train our model .

Function	**Arguments**
fit_generator	-epochs = 200 -shuffle , when = true , dataset will be shuffled before the training CNN class_weight , when data is unbalanced , class weight takes the value 'balanced' callbacks=[checkpoint,early] , fit_model uses the two previous functions

Table 2.3 – Function description for the model fitting

2.4 Testing the CNN Model

After training the model, we proceed to the testing phase, where we used the third subset (test).

```
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tensorflow import keras
from keras.models import load_model
from tensorflow import keras
from keras.preprocessing import image
import PIL
import os
import os.path
from PIL import Image
from os import listdir
from PIL import Image as PImage
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import img_to_array
from keras.preprocessing.image import load_img
import numpy as np
import argparse
import glob
import PIL
import os
import os.path
from PIL import Image
indiceDR0=0 #NO DR counter
indiceDR12=0#YES Mild DR counter
indiceDR2=0#YES moderate DR counter
indiceDR3=0#YES severe DR counter
total=0# total of images

f = r'/content/out/test/NO' #path of third subset

for file in os.listdir(f):
    f_img = f+"/"+file

    img = image.load_img(f_img, target_size=(224, 224)) # use the same
    target size of CNN model
    img = np.asarray(img) #converte image to an array
    img = np.expand_dims(img, axis=0)
    output = model.predict(img) #what the model pridected


    if output[0][0]>output[0][1] and output[0][0]>output[0][2] and output
    [0][0]>output[0][3] and output[0][0]>output[0][4] # if the
    probability that this image belongs to NO DR is bigger than the
    probability of that this image belongs to Mild, moderate, and sever:
        print("No DR", output[0])
        indiceDR0 += 1
        total += 1
    if output[0][1]>output[0][0] and output[0][1]>output[0][2] and output
    [0][1]>output[0][3] and output[0][1]>output[0][4] :
        print("MILD", output[0])
        indiceDR1 += 1
        total += 1
    if output[0][2]>output[0][0] and output[0][2]>output[0][1] and output
    [0][2]>output[0][3] and output[0][2]>output[0][4] :
        print("MODERATE", output[0])
        indiceDR2 += 1
        total += 1
    if output[0][3]>output[0][0] and output[0][3]>output[0][1] and output
   [0][3]>output[0][2] and output[0][3]>output[0][4] :
        print("SEVERE", output[0])
        indiceDR3 += 1
        total += 1

print("===============")
print("Diabetic retinopathy predict")
print("===============")
print("No DR",indiceDR0,"Mild",indiceDR1,"moderate",indiceDR2,"severe",
   indiceDR3)
```

Listing 2.12 – Testning model

2.5 Conclusion

In this chapter, we presented our dataset structures, described our system design, explained step by step the preprocessing phase, mentioned the tools, libraries, and frameworks we used, presented the implementation of a large part of our system, and discussed our CNN model in detail. In the following chapter, we will go over various experiments and their outcomes.

Chapter 3

Experimentation and Results

3.1 Introduction

In the previous chapter we have discussed the global architecture of our system, the structure of the dataset, and we have also presented code for each system design phase. In this chapter we will present the obtained results of every proposed structure and explain the remaining parameters. Finally, we will compare our results with some of the previous related works.

3.2 First proposed structure (Binary Classification)

3.2.1 The used Dataset

We use the APTOS 2019 [48] dataset [1] presented in chapter 3. The next table explains the orginal structure of the dataset with number of images per class.

Level	Number of images
NO DR	1805
Mild	370
Moderate	999
Severe	193
Proliferate DR	295

Table 3.1 – Original structure

We modify the original structure by grouping all diabetic retinopathy levels (mild, moderate, sever, and proliferate) to represent class of DR, and the healthy one represent the class of No DR. The Dataset is splitted into 3 subsets: training subset, validation subset and testing subset as it is shown in Table 3.2.1.

1. https://www.kaggle.com/c/aptos2019-blindness-detection/overview

	Training	Validation	Test	Total
NO DR	1263	361	181	1805
YES DR	1299	371	187	1857
Total	2562	732	368	3662

Table 3.2 – first proposed structure

3.2.2 Specification of the used parameters

The table below presents the used parameters in this structure.

	Informations
input shape	(224,224,3)
Number of Epochs	60 early stopped
Batch size	batch size =81
DropoutRate	20%
Patience	30
Last output Layer	Dense(2)
ModelCheckpoint Monitor	val_acc
EarlyStopping Monitor	val_acc

Table 3.3 – Table of parameters for first proposed structure

3.2.3 Results and discussion

The evaluation metrics used are training accuracy, validation accuracy, training loss and validation loss. To visualise the performance of the model, we present the following plots.

- A plot of accuracy and validation accuracy over epochs:

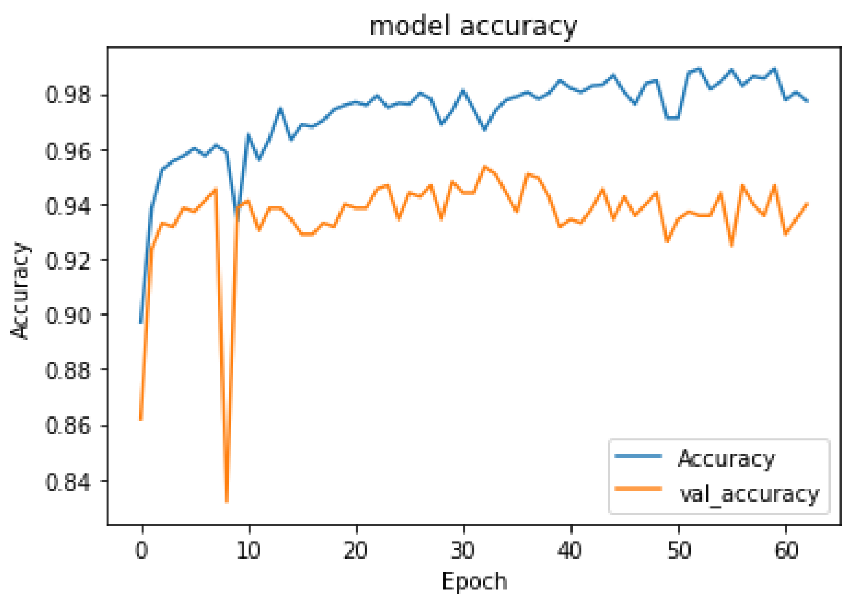


Figure 3.1 – The model Accuracy for binary classification

- A plot of loss and validation loss over epochs over epochs:

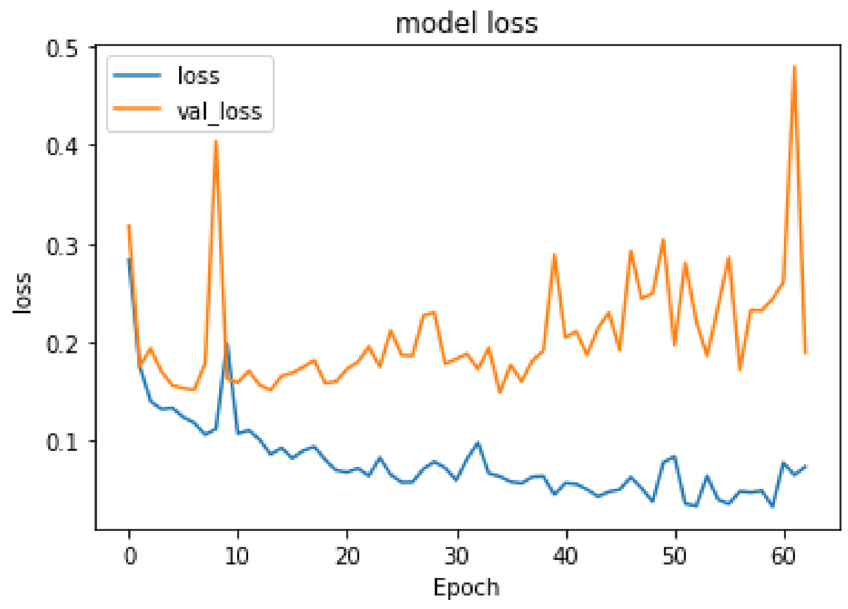


Figure 3.2 – The model Loss for binary classification

- The plots of learning curves 3.1 and 3.2 show a good fit because:
  — The accuracy and validation accuracy are converging to the same value.
  — The loss and loss validation are decreasing to the same value.

1. **Saving Model**

   The best model was saved in the epoch number 15, the the early stopping was in the epoch 31. Performance metrics of this model are in the Table 3.4.

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	98.75%	95.35%	3,30%	19,20%

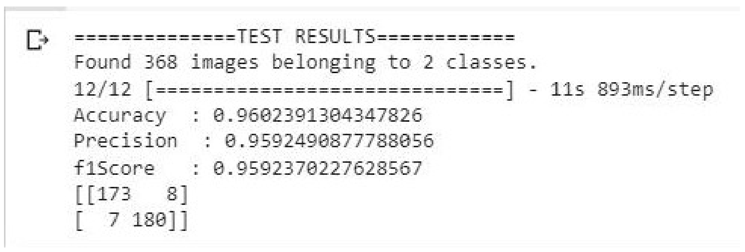
Table 3.4 – Table of results first proposed structure.

2. **Testing Model**

   In this phase, we will use the third subset, in order to test the performance of our CNN model on new images from the dataset.

   After testing our model on the third subset:

   (a) **Metrics results on test subset**



```
=============TEST RESULTS===========
Found 368 images belonging to 2 classes.
12/12 [==============================] - 11s 893ms/step
Accuracy  : 0.9602391304347826
Precision  : 0.9592490877788056
f1Score   : 0.9592370227628567
[[173   8]
 [  7 180]]
```

Figure 3.3 – Model testing first structure.

   (b) **Confusion matrix**

   The confusion matrix is a table that is often used to describe the performance of a classification model.

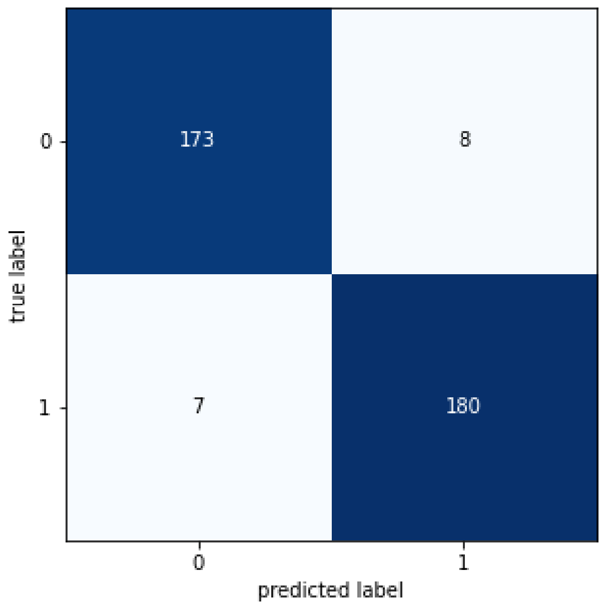


Figure 3.4 – Confusion matrix first structure.

— The model could predict 173 images right out of 181 from the NO DR test folder.

— The model could predict 180 images right out of 187 from the YES DR test folder.

3.3 Second Proposed Structure (Multiclass Classification)

3.3.1 The used Dataset

In this structure we have used the 2015 Kaggle competition dataset[2]. We only used 9726 images from the training set due to the lack of images in the two last classes (sever and proliferate).

Level	**Number of images**
NO DR	25810
Mild	2443
Moderate	5292
Severe	873
Proliferate DR	708

Table 3.5 – Original structure

We group the last two classes together due to lack of images. The structure becomes: (i) mild, (ii) moderate, (iii) sever with proliferate, and healthy ones which represent NO DR. There are four levels in this structure (see Table 3.6).

Table 3.6 – Second proposed structure.

Level of DR	Class	number of images used
NO DR	Class of NO DR	2811
Mild	Class of Mild DR	2443
Moderate	Class of Moderate DR	2891
Sever Proliferate	Class of Sever DR	873 + 708 = 1581

After splitting dataset with this structure and augmenting the train subset we obtain the Table 3.7.

- **Before data augmentation**

2. https://www.kaggle.com/c/diabetic-retinopathy-detection/data

	Training	Validation	Test	Total
Lvl 0	1968	561	282	2811
Lvl 1	1710	488	245	2443
Lvl 2	2024	577	290	2891
Lvl 3&4	1106	315	160	1581

Table 3.7 – Second proposed structure before data augmentation.

- **After Data augmentation**

	Training	Validation	Test
Lvl 0	19000	561	282
Lvl 1	17000	488	245
Lvl 2	20000	577	290
Lvl 3&4	11000	315	160

Table 3.8 – Second proposed structure after data augmentation.

3.3.2 Specification of the used parameters

Table 3.9 provides the values of the used parameters.

	Informations
input shape	(224,224,3)
Number of Epochs	26 early stopped
Batch size	batch size =115
DropoutRate	30%
Patience	10
Last output Layer	Dense(2)
ModelCheckpoint Monitor	val_acc
EarlyStopping Monitor	val_acc

Table 3.9 – Table of parameters for second proposed structure.

3.3.3 Results and discussion

The evaluation metrics used are training accuracy, validation accuracy, training loss and validation loss. To visualise the performance of the model, we present the following plots.

- A plot of accuracy and validation accuracy over epochs:

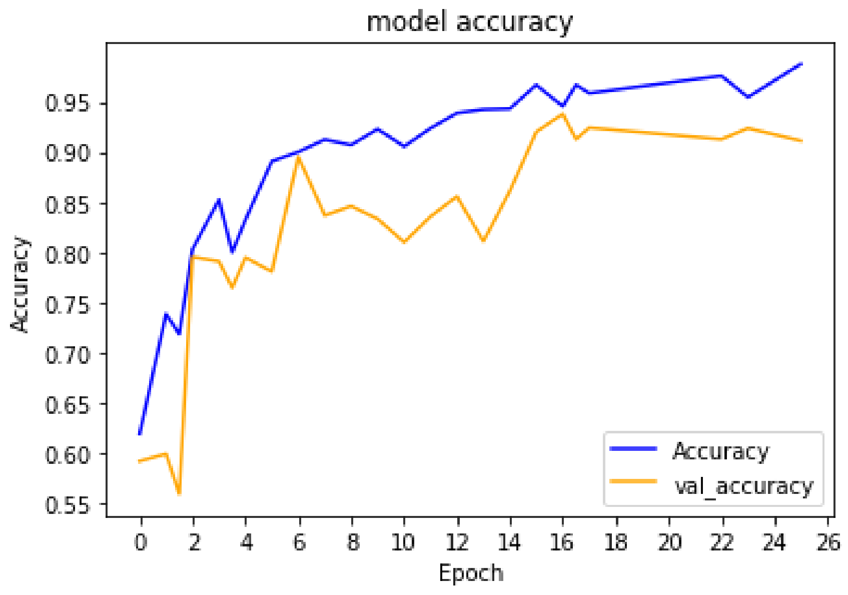


Figure 3.5 – The model Accuracy for multi-class classification.

- A plot of loss and validation loss over epochs over epochs:

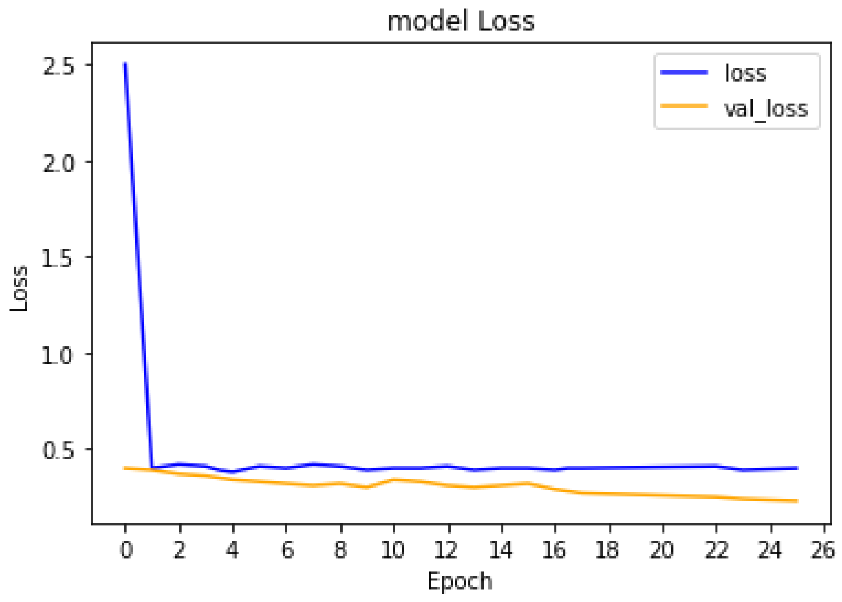


Figure 3.6 – The model Loss for multi-class classification.

- The plots of learning curves 3.5 and 3.6 show a good Fit because:

— The accuracy and validation accuracy are converging to the same value.

— The loss and loss validation are decreasing to the same value.

1. **Saving Model**

The early stopping was in the epoch 26. Performance metrics of this model are in the Table 3.10.

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	96.73%	93.45%	4.99%	3.24%

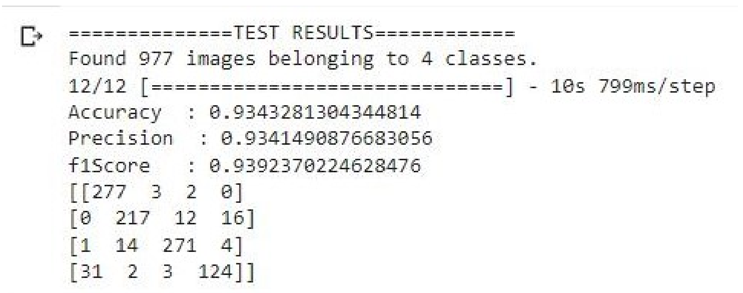
Table 3.10 – Table of results second proposed structure.

2. **Testing Model**

   In this phase, we will use the third subset, in order to test the performance of our CNN model on new images from the dataset.

   After testing our model on the third subset:

   (a) **Metrics results on test subset**



```
=============TEST RESULTS===========
Found 977 images belonging to 4 classes.
12/12 [=============================] - 10s 799ms/step
Accuracy  : 0.9343281304344814
Precision  : 0.9341490876683056
f1Score   : 0.9392370224628476
[[277  3  2  0]
[0  217  12  16]
[1  14  271  4]
[31  2  3  124]]
```

Figure 3.7 – Model testing second structure.

   (b) **Confusion matrix**

   This matrix shows the performance of our model.

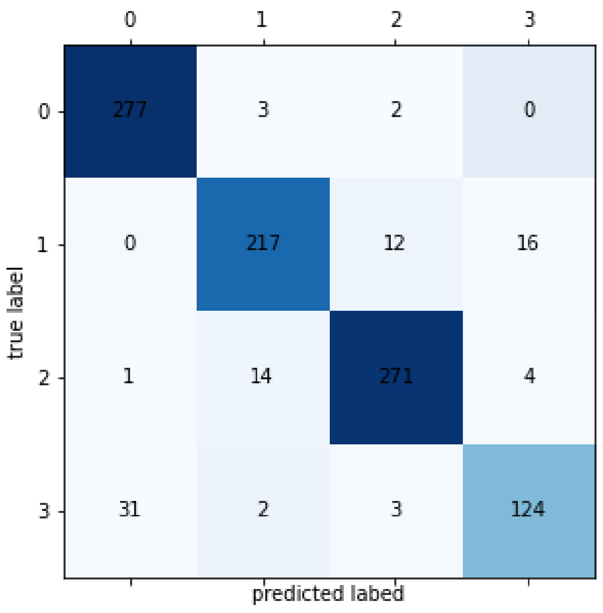


Figure 3.8 – Confusion matrix second structure.

The results of the confusion matrix are explained:

— Row 0 Column 0 = The NO DR correctly diagnosed as NO DR and its value is 277 image out of 282.

— Row 0 Column 1, Column 2, and Column = The NO DR incorrectly diagnosed as Mild DR or Moderate DR or Sever DR and the values are 3, 2, 0 ,respectively out of 282.

— Row 1 Column 0, Column 2, and Column 3 = The mild DR incorrectly diagnosed as NO DR, moderate DR, or sever DR and the values are 0, 12, 16 ,respectively out of 245.

— Row 1 Column 1 = The mild DR correctly diagnosed as mild DR and its value is 217 out of 245. Row 2 Column 0, 1, and 3 = The moderate DR incorrectly diagnosed as NO DR, Mild DR, or Severe DR, the values are 1, 14, 4 ,respectively out of 290.

— Row 2 Column 2 = The moderate DR correctly diagnosed as moderate DR its values is 271 out of 290.

— Row 3 Column 0, 1, and 2 = The Sever DR incorrectly diagnosed as NO DR, mild DR, or moderate DR, the values are 31, 2, 3 ,respectively out of 160.

— Row 3 Column 3 = The Severe DR correctly diagnosed as Sever DR its values is 124 out of 160.

3.4 Comparison with other related works

To conclude this chapter, Table 3.11 compares our proposition to some related works, where we have chosen accuracy and validation accuracy as comparison criteria.

	Accuracy	**Validation accuracy**	**Number of classes**	**Dataset**
Our work (Binary classification)	98.75%	95.35%	2	[48]
Our work (Multi class classification)	96.73%	93.45%	4	[47]
CNN model (Binary classification)[42]	96.93%	95.08%	2	[48]
CNN model (Multi class classification) [42]	93,06%	81,12%	3	[48]
ResNet34 [63]	85%	-	2	[47]
VGGNet [64]	92%	89%	3	[48]
Pretrained CNNs [65]	88.21%	-	2	-
Pretrained CNN InceptionNet V3 [46]	63.23%	-	5	[47]
A method based on CNN [43]	75%	-	5	[47]

Table 3.11 – Comparison of our proposition to other works.

3.5 Conclusion

In this chapter, we detailed the different parameters for each structure, worked on different datasets, explained all experiments, and illustrated results with plots and the confusion matrix for the test phase. The obtained outcomes are highly good, which encourages us to improve our model architecture.

Conclusion and Perspectives

Artificial Intelligence has grown to be very popular in today's world, it has undoubtedly brought new efficiencies and quality to healthcare outcomes. Deep learning enhanced algorithms have been demonstrated to achieve state-of-the-art results on complex computer vision tasks, including medical image diagnosis of Diabetic Retinopathy (DR) which is considered a devastating disease that causes visual loss and blindness.
Throughout this work we have proposed a CNN model for two different structure (Binary classification and multi class classification) for the detection and classification of diabetic retinopathy levels. The CNN model showed very excellent results compared to other works where accuracy for binary and multi class was 98,75% 96.73% ,respectfully.
Finally, from a technical engineering perspective this work paves the way for future work such as model improvement to avoid the overfitting, and using differents datasets, as well as many other deep learning techniques and preprocessing methods.

Bibliography

[1] World health organization. *Diabetic retinopathy.* URL: https://www.who.int/.

[2] kaggle. *Datasets.* URL: https://www.kaggle.com/datasets.

[3] Wikipedia. *Artificial intelligence in healthcare.* URL: https://en.wikipedia.org/wiki/Artificial_intelligence_in_healthcare#cite_note-1.

[4] Mayo clinic. *Cardiovascular Medicine.* URL: https://www.mayoclinic.org/departments-centers/ai-cardiology/overview.

[5] Terri Malloy. *Mayo researchers use AI to detect weak heart pump via patients' Apple Watch ECGs.* URL: https://newsnetwork.mayoclinic.org/discussion/mayo-researchers-use-ai-to-detect-weak-heart-pump-via-patients-apple-watch-ecgs/.

[6] Parambir S Dulai Ariela K Holmer. *Using Artificial Intelligence to Identify Patients With Ulcerative Colitis in Endoscopic and Histologic Remission.* URL: https://doi.org/10.1053/j.gastro.2020.04.01.

[7] JOHNS HOPKINGS medecine. *Gastroenterology and Hepatology.* URL: https://www.hopkinsmedicine.org/gastroenterology_hepatology/clinical_services/basic_endoscopy/esophagogastroduodenoscopy.html.

[8] Anirvan P et al. "Artificial Intelligence in Gastrointestinal Endoscopy in a Resource-constrained Setting: A Reality Check". In: *Euroasian journal of hepato gastroenterology* (2019).

[9] National Institute of Biomedical Imaging and Bioengineering (NIBIB). *Computed Tomography (CT).* URL: https://www.nibib.nih.gov/science-education/science-topics/computed-tomography-ct.

[10] *Pub med.* URL: https://pubmed.ncbi.nlm.nih.gov/?linkname=pubmed_pubmed&from_uid=30581604.

[11] Lu W et al. "Applications of Artificial Intelligence in Ophthalmology General Overview". In: *J Ophthalmol* (2018).

[12] Ed Burns. *Machine learning.* URL: https://www.techtarget.com/searchenterpriseai/definition/machine-learning-ML.

[13] Jordi DE LA TORRE GALLART. "Diabetic Retinopathy Classification and Interpretation using Deep Learning Techniques". In: (2018).

[14] IBM Cloud Education. *Unsupervised Learning*. URL: https://www.ibm.com/cloud/learn/unsupervised-learnin.

[15] edureka. *What is Machine Learning? Machine Learning For Beginners*. URL: https://www.edureka.co/blog/what-is-machine-learning.

[16] IBM Cloud Education. *Semi-supervised learning*. URL: https://www.ibm.com/cloud/learn/semi-supervised-learnin.

[17] deepAI. *What is Deep Learning?* URL: https://deepai.org/machine-learning-glossary-and-terms/deep-learning.

[18] Samaya Madhavan and M. Tim Jones. *IBM developer, Deep learning architectures*. URL: https://developer.ibm.com/articles/cc-machine-learning-deep-learning-architectures/.

[19] Saptarshi Sengupta et al. "A review of deep learning with special emphasis on architectures, applications and recent trends". In: *Knowledge-Based Systems* 194 (Feb. 2020), p. 105596. DOI: 10.1016/j.knosys.2020.105596.

[20] *The Relation Between Artificial And Biological Neuron?* URL: https://smhatre59.medium.com/what-is-the-relation-between-artificial-and-biological-neuron-18b05831036.

[21] Claude Touzet. *les réseaux de neurones artificiels, introduction au connexionnisme*. EC2, 1992.

[22] *Neural Network Architecture*. URL: https://www.sciencedirect.com/topics/engineering/neural-network-architecture.

[23] *CNN definition and tutorial*. URL: https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network.

[24] INAS AL-KAMACHY. *CLASSIFICATION OF DIABETIC RETINOPATHY USING PRE-TRAINED DEEP LEARNING MODELS*. ÇANKAYA UNIVERSITY, 2019.

[25] deepAI. *Padding (machine learning)*. URL: https://deepai.org/machine-learning-glossary-and-terms/padding.

[26] *Calculate output size of Convolution (stride)*. URL: https://iq.opengenus.org/output-size-of-convolution.

[27] Sakib Mostafa and Fang-Xiang Wu. "Chapter 3 - Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images". In: *Neural Engineering Techniques for Autism Spectrum Disorder*. Ed. by Ayman S. El-Baz and Jasjit S. Suri. Academic Press, 2021, pp. 23–38. ISBN: 978-0-12-822822-7. DOI: https://doi.org/10.1016/B978-0-12-822822-7.00003-X. URL: https://www.sciencedirect.com/science/article/pii/B978012822822700003X.

[28] PRAMOD GUPTA and NARESH K. SINHA. "CHAPTER 14 - Neural Networks for Identification of Nonlinear Systems: An Overview". In: *Soft Computing and Intelligent Systems*. Ed. by NARESH K. SINHA and MADAN M. GUPTA. Academic Press Series in Engineering. San Diego: Academic Press, 2000, pp. 337–356. ISBN: 978-0-12-646490-0. DOI: https://doi.org/10.1016/B978-012646490-0/50017-2. URL: https://www.sciencedirect.com/science/article/pii/B9780126464900500172.

[29] *A Gentle Introduction to Pooling Layers for Convolutional Neural Networks*. URL: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/.

[30] *How ReLU and Dropout Layers Work in CNNs*. URL: https://www.baeldung.com/cs/ml-relu-dropout-layers.

[31] *Understanding and Implementing LeNet-5 CNN Architecture (Deep Learning)*. URL: https://towardsdatascience.com/understanding-and-implementing-lenet-5-cnn-architecture-deep-learning-a2d531ebc342.

[32] *AlexNet*. URL: https://en.wikipedia.org/wiki/AlexNet.

[33] Hoo-Chang Shin et al. "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning". In: *IEEE Transactions on Medical Imaging* 35.5 (2016), pp. 1285–1298. DOI: 10.1109/TMI.2016.2528162.

[34] Erika F. Brutsaert. *Diabetes Mellitus (DM)*. URL: https://www.msdmanuals.com/home/hormonal-and-metabolic-disorders/diabetes-mellitus-dm-and-disorders-of-blood-sugar-metabolism/diabetes-mellitus.

[35] Centers for Disease Control and Prevention. *Diabetes Basics*. URL: https://www.cdc.gov/diabetes/basics/diabetes.

[36] Roy Taylor and Deborah Batey. *Handbook of retinal screening in diabetes: diagnosis and management*. John Wiley & Sons, 2012.

[37] Kierstan Boyd. "American Academy of Ophthalmology-What is Diabetic Retinopathy". In: *Accessed: Sep* 10 (2020), p. 2021.

[38] Early Treatment Diabetic Retinopathy Study Research Group et al. "Grading diabetic retinopathy from stereoscopic color fundus photographs—an extension of the modified Airlie House classification: ETDRS report number 10". In: *Ophthalmology* 98.5 (1991), pp. 786–806.

[39] Wejdan L. Alyoubi, Wafaa M. Shalash, and Maysoon F. Abulkhair. "Diabetic retinopathy detection through deep learning techniques: A review". In: *Informatics in Medicine Unlocked* 20 (2020), p. 100377. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2020.100377. URL: https://www.sciencedirect.com/science/article/pii/S2352914820302069.

[40] *Diabetic retinopathy stages.* URL: https://www.nhs.uk/conditions/diabetic-retinopathy/stages/.

[41] the Ophthalmic Photographers Society. *Fundus Photography Overview.* URL: https://www.opsweb.org/page/fundusphotography.

[42] M.R. Sebti et al. ""A Deep Learning Approach for the Diabetic Retinopathy Detection"". In: *The Sixth Smart City Applications Iinternational Conference* (2021).

[43] Harry Pratt et al. "Convolutional neural networks for diabetic retinopathy". In: *Procedia computer science* 90 (2016), pp. 200–205.

[44] Mehdi Torabian Esfahani, Mahsa Ghaderi, and RJLEJPT Kafiyeh. "Classification of diabetic and normal fundus images using new deep learning method". In: *Leonardo Electron. J. Pract. Technol* 17.32 (2018), pp. 233–248.

[45] Yung-Hui Li et al. "Computer-assisted diagnosis for diabetic retinopathy based on fundus images using deep convolutional neural network". In: *Mobile Information Systems* 2019 (2019).

[46] Xiaoliang Wang et al. "Diabetic retinopathy stage classification using convolutional neural networks". In: *2018 IEEE International Conference on Information Reuse and Integration (IRI)*. IEEE. 2018, pp. 465–471.

[47] ilovescience. *Diabetic Retinopathy Detection 2015 Competition.* URL: https://www.kaggle.com/c/diabetic-retinopathy-detection/data.

[48] Asia Pacific Tele-Ophthalmology Society (APTOS). *APTOS 2019 Blindness Detection.* URL: https://www.kaggle.com/c/aptos2019-blindness-detection/overview.

[49] Andrew G Howard. "Some improvements on deep convolutional neural network based image classification". In: *arXiv preprint arXiv:1312.5402* (2013).

[50] Stephan Zheng et al. "Improving the robustness of deep neural networks via stability training". In: *Proceedings of the ieee conference on computer vision and pattern recognition*. 2016, pp. 4480–4488.

[51] Suvajit Dutta et al. "Classification of diabetic retinopathy images by using deep learning models". In: *International Journal of Grid and Distributed Computing* 11.1 (2018), pp. 89–106.

[52] Mohammad Abdullah-Al-Wadud et al. "A dynamic histogram equalization for image contrast enhancement". In: *IEEE Transactions on Consumer Electronics* 53.2 (2007), pp. 593–600.

[53] Carson Lam et al. "Automated detection of diabetic retinopathy using deep learning". In: *AMIA summits on translational science proceedings* 2018 (2018), p. 147.

[54] *Goole Colaboratory*. URL: https://colab.research.google.com.

[55] *OpenCV Documentation ,Operations on Arrays* , URL:https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#addweighted.

[56] *Spyder IDE*. URL: https://www.spyder-ide.org/.

[57] *Matplotlib - visualization with python*. URL: https://matplotlib.org/.

[58] *Keras: the Python deep learning*. URL: https://keras.io/.

[59] *TensorFlow*. URL: https://www.tensorflow.org/resources/learn-ml?gclid=CjwKCAjwtcCVBhA0EiwAT1fY7_V3GuCJmsl_YsGhshq-3C7ySEknxCBQN5So2iflIaDxHQp4ckftsho
BwE.

[60] *NumPy*. URL: https://numpy.org/.

[61] *model checkpoint keras*. URL: https://keras.io/api/callbacks/model_checkpoint/.

[62] *Early stopping keras*. URL: https://keras.io/api/callbacks/early_stopping/.

[63] Mehdi Torabian ESFAHANI, Mahsa GHADERI, and Raheleh KAFIYEH. "Classification of diabetic and normal fundus images using new deep learning method". In: *Leonardo Electron J Pract Technol* 17.32 (2018), pp. 233–248.

[64] Mohamed Shaban et al. "A convolutional neural network for the screening and staging of diabetic retinopathy". In: *Plos one* 15.6 (2020), e0233514.

[65] Hongyang Jiang et al. "An Interpretable Ensemble Deep Learning Model for Diabetic Retinopathy Disease Classification". In: *2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*. 2019, pp. 2045–2048. DOI: 10.1109/EMBC.2019.8857160.