PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
Ministry of Higher Education and Scientific Research
University of Mohamed Khider – BISKRA
Faculty of Exact Sciences, Science of Nature and Life
Computer Science Department

Order N°: GLSD02/M2/2021

Thesis

Presented to obtain the academic master's degree in

Computer science

Option: **GLSD**

Diabetic retinopathy detection based on retinography images

By:

SEBTI MOHAMED RIAD

Members of the jury :

Full Name	Grade	President
KAHLOUL Laid	Professor	Supervisor
Full Name	Grade	Examiner

Session 2020-2021

Acknowledgements

First and Foremost I would like to show gratitude and praise to Allah almighty for giving me the strength, knowledge to complete this work.

Secondly, I would like to thank my parents for believing in me and supporting me in all the difficult situations.

Thirdly, I would like to thank my teacher and my supervisor Professor **KAHLOUL Laid**, for guiding me in the right direction throughout this project.

Then I would like to thank all GLSD teachers, I learned a lot from them.

Finally, I would like to thank all the other members of the department, my colleagues and my friends. They always kept me motivated and helped me by giving me mental support and strength to finish my thesis.

To my family, my friends, and everyone I love and I care about, thank you.

SEBTI Mohamed Riad

Abstract

Recently, Artificial Intelligence (AI) has invaded all areas of scientific research, for what it brings in terms of solutions. The field of health is not an exception. Diabetes is one of the most common diseases in the world and in Algeria. One of its complications is diabetic retinopathy, which can blur or distort the vision of the patient, it is one of the main causes of blindness. Early detection of diabetic retinopathy can greatly help in the treatment. The recent development in the field of AI and especially Deep learning provides ambitious solutions that can be used to predict, forecast and diagnose several diseases in their early phases.

In our master project we have explored the potential of deep learning (DL) for retina image analysis. We have studied DL concepts with a convolutional neural network (CNN) algorithm to build a multi-classification model that can detect and classify retinopathy levels automatically. We have used google colab which is an accessible platform by everyone to train our models. In this project we have applied a CNN architecture with several parameters on 2 different datasets of diabetic retinopathy with different structures.

Keywords: Healthcare, Health care Diabetes, Diabetic retinopathy, Artificial intelligence, Machine learning, Deep learning.

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. L'une de ses complications est la rétinopathie diabétique, qui peut peut brouiller ou déformer la vision du patient, elle est l'une des principales causes de cécité. La détection précoce de la rétinopathie diabétique peut grandement aider au traitement. Le développement récent dans le domaine de l'IA et surtout de l'apprentissage profond fournit des solutions ambitieuses qui peuvent être utilisées pour prédire, prévoir et diagnostiquer plusieurs maladies dans leurs phases précoces.

Dans notre projet de master, nous avons exploré le potentiel de l'apprentissage profond (DL) 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 de multi-classification qui peut détecter et classer automatiquement les niveaux de la rétinopathie diabetique. Nous avons utilisé google colab qui est une plateforme accessible par tout le monde pour entraîner nos modèles. 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, Santé, Diabète, Rétinopathie diabétique, Intelligence artificielle , Apprentissage automatique, Apprentissage profond.

Contents

List of Figures

List of source code

List of Tables

Chapter 1

General introduction

Nowadays, with the development of computers and their ability to perform high computational tasks, many algorithms and learning techniques have been implemented to automate tasks or to help humans in decisions making. The application of these techniques is used in several areas such as online shopping, web search, Cybersecurity, Healthcare, etc. Indeed, healthcare is one of the fields that has been well developed by the automation of tasks by machines [29]. Several systems have been created to diagnose diseases to help clinicians in decisions making [38]. The majority of these systems use machine learning and deep learning to extract clinical information from images.

Diabetes is one of the most widespread diseases, affecting about 425 million people worldwide [1], one of its complications is the diabetic retinopathy [35].

Problematic

Diabetic retinopathy (which damages to the eyes: eye and retina) is a serious complication of diabetes that affects 50% of type 2[1] diabetes patients [10]. Diabetic retinopathy is one of the leading causes of blindness if it was diagnosed at a late stage, the patient will loose his vision partially and maybe entirely. Diagnosing diabetic retinopathy at early stages can increase curing

[1]Type 1 diabetes is usually diagnosed in children, adolescents or young adults. Type 1 diabetes is not related to lifestyle or obesity but to an abnormality in the secretion of an endocrine hormone (insulin) and the immune system. Type 2 diabetes is mainly a lifestyle disease. This type is caused by a disruption of the blood sugar balance

the patients and restore vision ability.

Objective of the work

The objective of this work is to propose deep learning models in order to detect and classify the different diabetic retinopathy levels using retinal images.

After this general introduction, the dissertation is structured as follows:

- **Chapter 2: Background of the Work.** This chapter discusses healthcare, how Artificial intelligence impacted healthcare field, state of the art of machine learning and deep learning techniques beside diabetes and diabetic retinopathy different levels.

- **Chapter 3: System design and implementation of a deep learning architecture for diabetic retinopathy.** This chapter describes the datasets, the system design phases, the implementations tools, and details the code.

- **Chapter 4: Experimentation and Results.** This chapter presents the result of our work, discusses how specific parameters influence on the obtained results and provides a comparative section to show the novelties of this work compared to previous ones.

- **Conclusion and perspectives:** In this last chapter, we summarize and review our ideas and results and giving some perspectives.

Chapter 2

Background

2.1 Introduction

AI is a rapidly expanding field of research with a great future, its applications concern all human activities. The use of AI in healthcare has showed a big success. In this field, AI has the ability to analyze medical data, to better serve patient, and to help healthcare systems, etc.

In this chapter we will define healthcare and its different stages, we will 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.

2.2 Healthcare

Healthcare or Health care has many definitions according to different sites.

2.2.1 Definitions

In a medical dictionary healthcare is "*The prevention, treatment, and management of illness and the preservation of mental and physical well-being through the services offered by the medical and allied health professions.*" [15].

According to Cambridge dictionary it is "*The set of services provided by a country or an organization for the treatment of the physically and the mentally ill* "[16].

The Definition of MBN (Market business news) is "*Healthcare refers to the efforts that medical professionals make to restore our physical and mental well-being. The term also includes the* *provision of services to maintain emotional well-being*" [18].

2.2.2 Healthcare stages

Depending on health problem, healthcare is staged into 3 different stages.

1. **Primary healthcare**

   Primary healthcare is essential healthcare made universally accessible to individuals and families in the community. Primary healthcare addresses the majority of a person's health needs throughout their lifetime,including physical, mental and social well-being focused in people-centred rather than disease-centred, primary healthcare includes health promotion,disease prevention, treatment, rehabilitation and palliative care [34].

2. **Secondary healthcare**

   When your primary care provider refers you to a specialist, you are then in secondary care. Secondary care simply means, you will be taken care of by someone who has more specific expertise in what is ailing you. Specialists focus either on a specific system of the body or a specific disease or condition. For example, cardiologists focus on the heart and its pumping system. Endocrinologists focus on hormone systems and some specialize in diseases like diabetes or thyroid disease. Secondary healthcare is when people need special healthcare [28].

3. **Tertiary healthcare**

   Specialized care that offers a service to those referred from secondary care for diagnosis or treatment, and which is not available in primary or secondary care. Tertiary care has become a common feature in certain specialties for rare conditions, or where the diagnostic or treatment facilities are scarce or require scarce combinations of resources, or which remain essentially the subject of research. These facilities are commonly found in medical schools and teaching hospitals [19].

2.3 Artificial intelligence and healthcare

AI is the development of computer systems that simulate human intelligence such as decision making, object detection, solving complex problems. The main benefits of AI are:

1. **Prediction with high accuracy.**
2. **Helps us in decision making processes.**
3. **Solve complex problems.**
4. **Preforms high level computing that take days for human to solve.**

AI has been impacting various domains including marketing, finance, gaming, and even the musical arts, however the largest impact of AI has been in the field of healthcare for two major reasons.

1. **Availability of medical data.**

   Tons of medical data is available in the form of medical history and medical images, basically with the availability of data implementation of AI becomes much easier, AI is based on technologies such as Deep learning and Machine learning which require tons and tons of data, so with the availability of data it became easier to use AI in the healthcare industry.

2. **Development of complex algorithms.**

   Machine learning was not capable of handling high dimensional data, however medical data or healthcare data is very high dimension data, which contains thousands of attributes, it is hard to analyse this data using ML, as soon as deep learning was introduced so the development of ML and DL played a major role in healthcare.

These are some examples of how AI could impact different domains in healthcare field.

- **AI in medical data.**

  Nuance is an example of how AI was used in medical data. Nuance is a pioneer and a leading provider of conversational AI [23], Nuance could use AI to predict the intent of a particular user, helped in storing, reformatting, and collecting data in order to provide faster and more consistent access to all the data so that any further analysis or any diagnosis can be preformed.

By using chatbots Nuance accelerated services,minimized the volume of inbound calls, could also cancel or validate medical appointments based on the customers history, Nuance increased the revenue, and could also send notifications via SMS or emails to doctors and patients.

In the figure 2.1 are the services provided by Nuance using AI which we have explained in the last paragraph.

Figure 2.1: Provided services by Nuance

- **AI in medical diagnosis**

  MRI imaging is an example of how AI was used in medical diagnosis. With the help of deep learning models, AI is actually revolutionising the image diagnosis field in medicine, one of the major application of AI in medical diagnosis is MRI scan, AI has taken over the complex analysis of MRI scans and it has made it much simpler process using a deep learning models.

  In the figure 2.2 an example of MRI image .

Figure 2.2: MRI imaging

- **AI in early detection**

  Smartwatches or trackers are examples of how AI was used in medical detection. AI has played a very important role in the early predictions of medical condition such as heart attacks, there are many health trackers that are being developed to monitor the health of a person and display warnings, when the device detect something unusual.

  In the figure 2.3 a smartwatch which can measure heart rate.

Figure 2.3: Smart watch

- **AI in medical assistance**

  Our example here is virtual nurses. Virtual nurses are developed using AI, basically a nurse but not physically present, this nurse can help in self care, clinical advice, scheduling an appointment, etc.

With such revolutions in the filed of healthcare it is clear that AI is benefiting us in many ways, and we have a total believe that AI will help and improve our situation in each and every domain if we use it in the correct way.

2.4 Machine learning

2.4.1 Introduction

Machine learning is a branch of AI, according to Arthur Samuel definition it is: “the field of study that gives computers the ability to learn without being explicitly programmed“, but this definition is informal.

Tom Mitchell provides a more modern definition: “A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

The figure 2.4 shows the relation between Artificial intelligence (AI), Machine learning (ML), and Deep learning (DL).

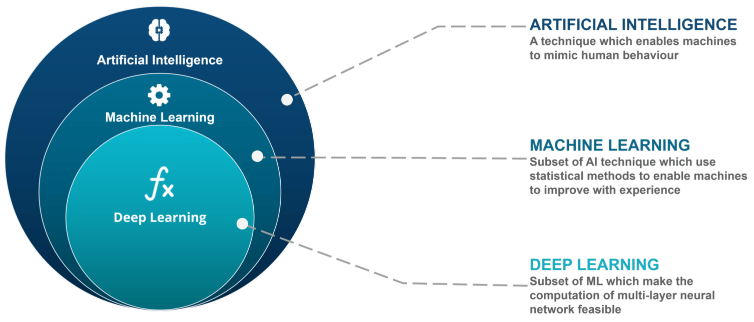

Artificial Intelligence
Machine Learning
Deep Learning
ARTIFICIAL INTELLIGENCE
A technique which enables machines to mimic human behaviour
MACHINE LEARNING
Subset of AI technique which use statistical methods to enable machines to improve with experience
DEEP LEARNING
Subset of ML which make the computation of multi-layer neural network feasible

Figure 2.4: AI vs ML vs DL

From our point of view if we want to define machine learning: it is a set of techniques which give the opportunity for a machine to learn, solve problems, do tasks. This set of techniques also allow machines to do analysis and conception in order to solve hard or even implement algorithms that we can’t implement.

2.4.2 Types of Machine learning algorithms

Depending on the nature of the data and the desired out-come of the algorithm, Machine learning algorithms can be divided into 3 types.

1. **Supervised Learning**

   In supervised learning we feed the algorithm with labeled data, after the training, our algorithm (Model) will be able to classify new data using a function that maps inputs to the desired output [20], in the figure 2.5 we did give apples to the model as a labeled data, after the training the model will be able to classify new apples images as apples.

   Examples of this type of algorithms : SVM, Random decision forest, etc.

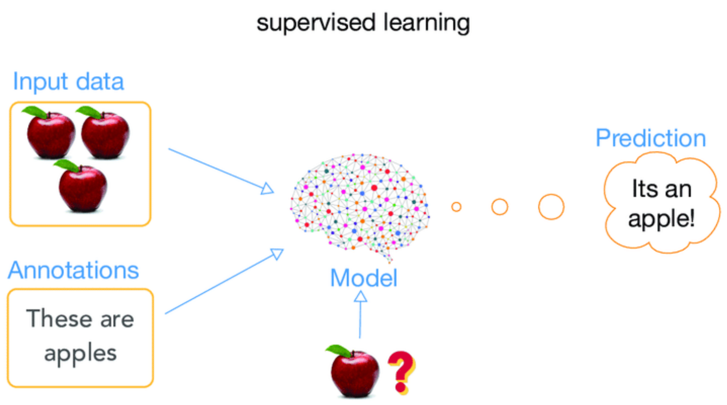


Figure 2.5: Supervised learning example

2. **Unsupervised Learning**

   In the unsupervised learning we feed the algorithm with unlabeled data, after the training, our algorithm (Model) will find labels according to the similarities shared between samples, our algorithm will be able to classify new data to the new labels created [20], in the figure 2.6 we did give unlabeled data to the model, after the training the model could make difference between the 3 classes, also the model created classes according to the similarities between samples.

   Examples of this type of algorithms : KNN, K-means, etc.

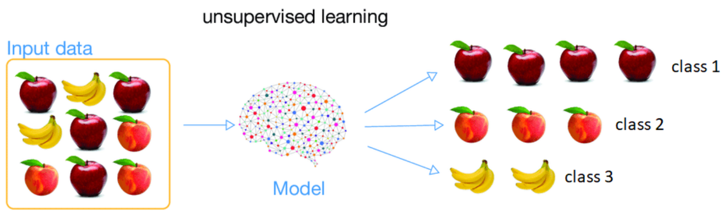


Figure 2.6: Unsupervised learning example

3. **Reinforcement Learning**

   It is the ability of an agent to interact with the environment and find out what is the best outcome. It follows the concept of hit and trial method. The agent is rewarded or penalized with a point for a correct or a wrong answer, and on the basis of the positive reward points gained the model trains itself. And again once trained it gets ready to predict the new data presented to it [20].

In the figure 2.7, the dog is playing the role of the agent, when the dog does a desired action or a close one the trainer who is playing the role of the environment will provide a reward for the dog otherwise no rewards or a negative reward.

Examples of this type of algorithms : Markov decision process, Approximate dynamic programming, etc.

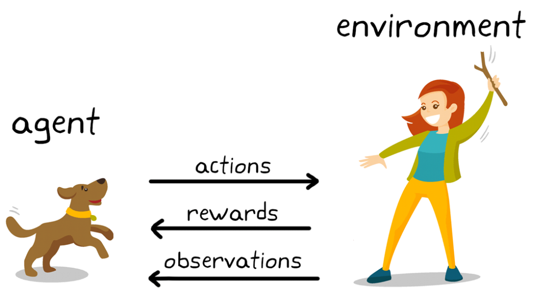


Figure 2.7: Reinforcement Learning example

2.5 Deep learning

Deep learning is an approach of Machine learning, which deals with the training of neural networks, DL is the most widely used in machine learning. It is also the most powerful technique in classification, which justifies its presence in almost all applications and fields that use machine learning.

The main difference between Deep learning and machine learning algorithms (see figure 2.8) is that deep learning algorithms focus on feature learning automatically but in machine learning we have to extract feature manually.

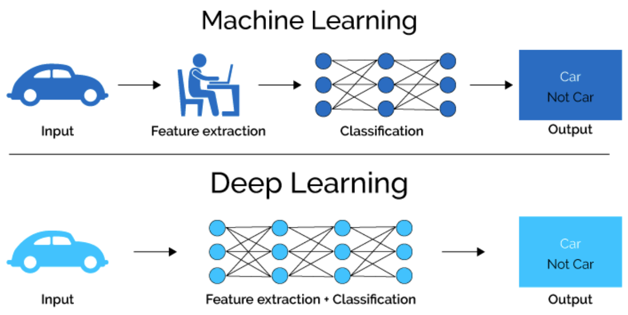


Figure 2.8: The main difference between ML and DL

2.5.1 Artificial neural networks

To understand artificial neural networks first, we need to know some basic notions of biological neural networks and the adaptations made in the artificial neural network model.

- **Biological neuron**

  A biological neuron (see figure 2.9) which is a cell composed of a nucleus. The cell body is divided to form the dendrites, it is by the dendrites that the information is transported from the outside to the soma (Cell body), the processed information continues its way through an axon to be transmitted to the other neurons.

  The transmission between two neurons is not direct, but rather there is a small space 10^{-9} between the axon of the afferent neuron and the dendrites of the other neuron. This space, defined as inter cellular, is called a synapse [2][3].

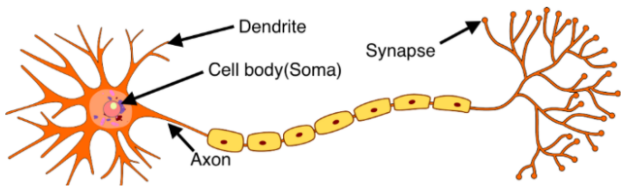


Figure 2.9: Biological neuron

- **Artificial neuron**

  An artificial neuron (see figure 2.10) which is a simplification of the biological neuron, each artificial neuron is an elementary processor. It takes as input a number of variables

X = $\{x1, x2, x3, ...xn\}$ called input layer, 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 which will then be transmitted to the output layer to be compared with a threshold value, and then provide an output response [39][3].

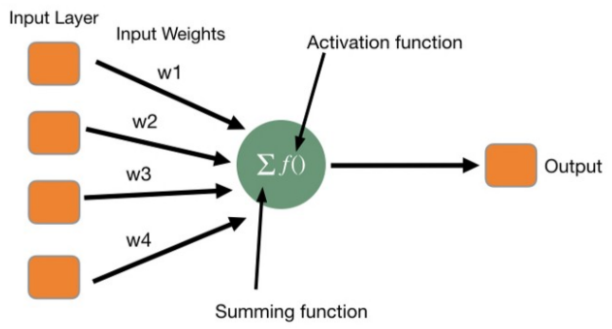


Figure 2.10: Artificial neuron

- **Artificial neural networks**

The artificial neural network (multi-layer) consisting of several layers of neurons: an input layer, a hidden layer, and an output layer. This neural network is an improvement of the previous one to address more complex problems.

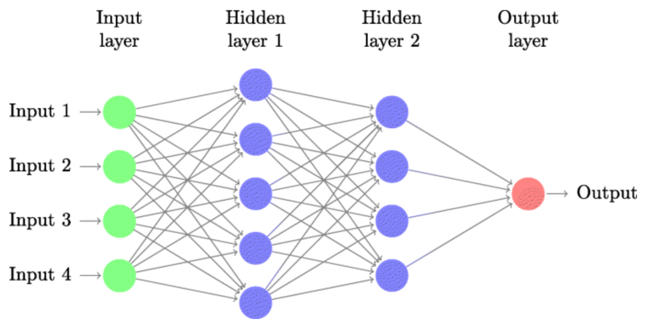


Figure 2.11: An artificial neural network

The artificial neural network has 3 types of layers (see figure 2.11) .

- **Types of layers**

  - **Input layer** : first layer in the neural network it is the layer which receive data, composed of input neurons.
  - **Hidden layer**: this layer or these layers are located between the input and the output layers, in these layers the problems will be solved, the choice of hidden layers depends on the complexity of the problem.
  - **Output layer**: this is the last layer of an artificial neural network, this last layer produces the output of the program, if we have a classification problem then the output layer neuron size will be equal to number of classes.

- **Activation function**

  We mentioned activation function in the last subsections, in this subsection we will detail more about activation function.

  The artificial neuron calculates the sum of the inputs and their weight $\sum_{i=1}^{n}(wi \times xi)+bias$, this sum can take any value between -∞ and ∞, so that the neuron can know the threshold value for its activation, an activation function is used.

  For the bias, we add a bias value to the weighted sum to get final value for prediction by our neuron [3].

  The activation function is used to introduce a non-linearity in the functioning of the artificial neuron unlike biological neurons which have a binary activation.

  The activation function of artificial neuron have continuous values which allows to have an infinity of possible values included in an interval of [-1,1] or [0,1].

  There are several forms of activation functions, each one is used in a specific context, we will mention the most used of them.

  - **ReLU Function**

    Defined by the following equation:

$$A(x) = max(0, x)$$

The activation function ReLU (Rectified Linear Unit) is a linear function that is considered the most popular and the most used activation function. Its operation is based on the fact that it replaces any negative input value with 0, and without modifying positive value [4].

The curve of ReLU function in the figure 2.12.

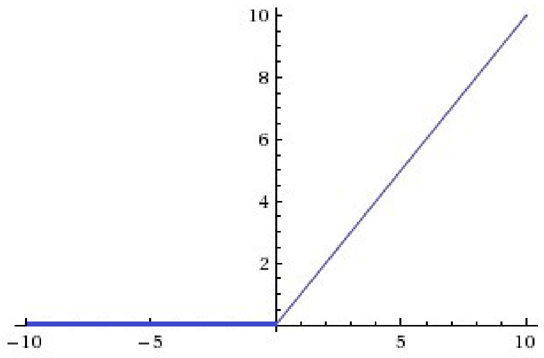


Figure 2.12: ReLU Function curve

- **Sigmoid function**

  Defined by the following equation:

$$f(x) = \frac{1}{1+e^{-x}}$$

Two values are possible for this activation function 0 or 1, its curve takes the form of an S ,this function is used in the output layer when we have a binary classification problem [4].

The curve of Sigmoid function in the figure 2.13.

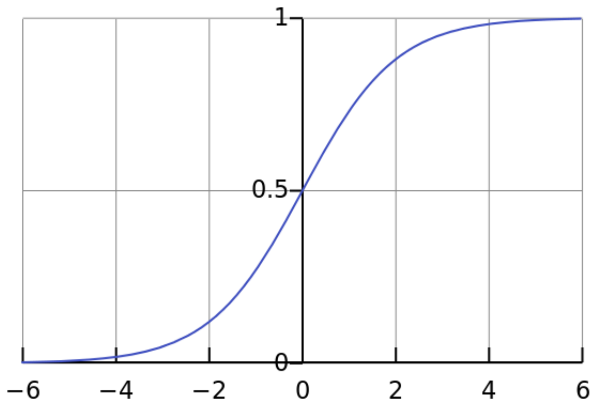


Figure 2.13: Sigmoid function curve

– **Tanh function curve**

Defined by the following equation:

$$f(x) = \frac{2}{1+e^{-2x}} - 1$$

This function transforms any real input into a value between [-1,1] [4].

The curve of Tanh function in the figure 2.14.

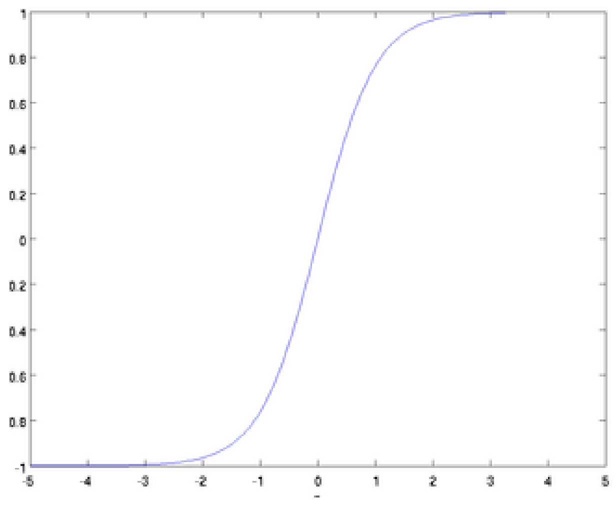


Figure 2.14: Tanh Function curve

Tanh is a variant of the sigmoid function, the relation between Tanh function and sigmoid function:

$$tanh(x) = 2sigmoid(2x) - 1$$

– **Softmax Function**

A normalized exponential function, it is used to represent a categorical law on a vector $z = (z1, z2, ..., zk)$ of K real numbers by transforming them to a vector $\sigma(z)$ with K probabilities of possible outcomes or the sum of K probabilities equal to 1.

$$softmax(z)i = \sigma(z)i = \frac{e^{zi}}{\sum_{k=1}^{K} e^{zk}}, z = (z1, z2, .., zK) \in R$$

This function is used in the output layer for the case of k-classes classification with $k >= 2$ in order to compute the probability to which class an input zi belongs (the class with the highest probability) [5].

2.5.2 Convolutional neural networks

- **CNN's description**

Convolutional neural networks are a sub-category of neural networks. They have all the characteristics of neural networks. And they are specifically designed to process images. The CNN's architecture is more specific, it is composed of two main blocks (see figure 2.15), and new layers types according to tasks[6].

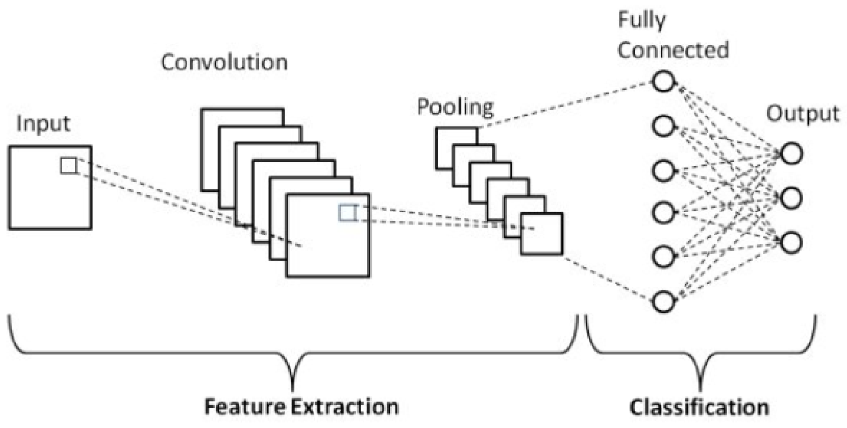


Figure 2.15: Convolutional neural network composition

- **CNN blocks**

  1. **The first block**

     The first block is the particularity of this type of neural network, since it works as a feature extractor by applying convolution filtering operations.The first layer filters the image with several convolution kernels, and returns feature maps [6].

  2. **The second block**

     The second block is not characteristic of a CNN, it is in fact found at the end of all neural networks used for classification. The values of the input vector are transformed (with several linear combinations and activation functions) to return a new output vector. The vector contains as many elements as there are classes, the element i represents the probability that the image belongs to class, each element is therefore between 0 and 1, and the sum of all is 1. These probabilities are calculated by the last layer of this block, which uses a sigmoid function (binary classification) or a softmax function (multi-class classification) as an activation function [6].

- **CNN layers types according to tasks**

  - **Convolutional layer**

    Its purpose is to identify the presence of a set of features in the images received as input. To do this, convolution filtering is performed, the principle is to drag a window representing the feature onto the image, and to calculate the convolution product between the feature and each portion of the scanned image.

    An image for a computer is a matrix of pixels with different values. Here is an example of how the CNN can apply a filter. before applying CNN filter we have our image as a matrix of pixels (see figure 2.16), with a filter of size = (3*3), this filter will be applied on this image [7] [8].

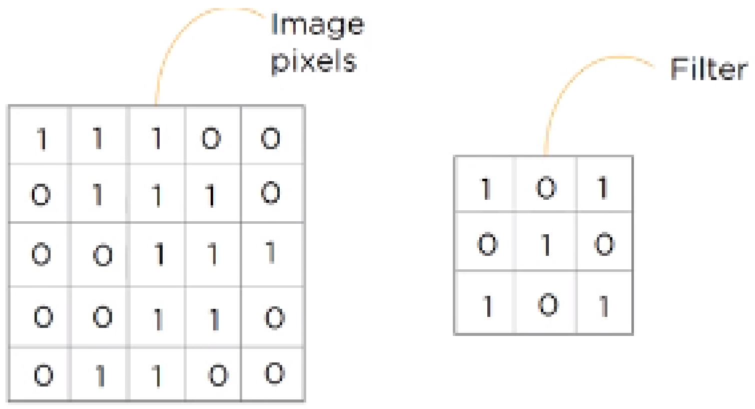


Figure 2.16: Convulitional layer filter

After taking a part of the image equal to the size of the filter, we will place the result of filter in the new matrix (feature map). By multiplying the image matrix with the filter, we sum the results, this is the first result, and we do the same to the other parts of image like what we did in figure 2.17.

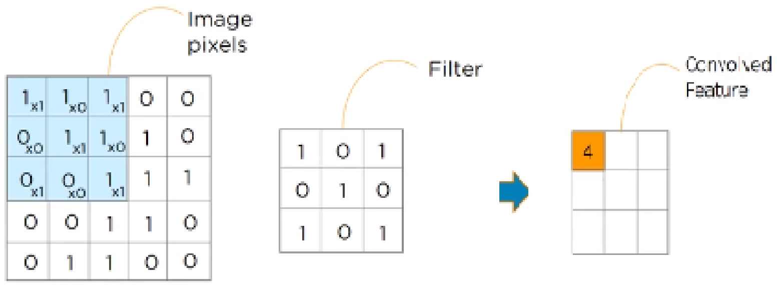


Figure 2.17: Convulitional layer filter first result

– **Pooling layer**

This type of layer 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.

In this example for the the blue part in the left-bottom of the map features in figure

2.18, the important feature is 4, and we do the same to the rest parts of the map features [7] [8].

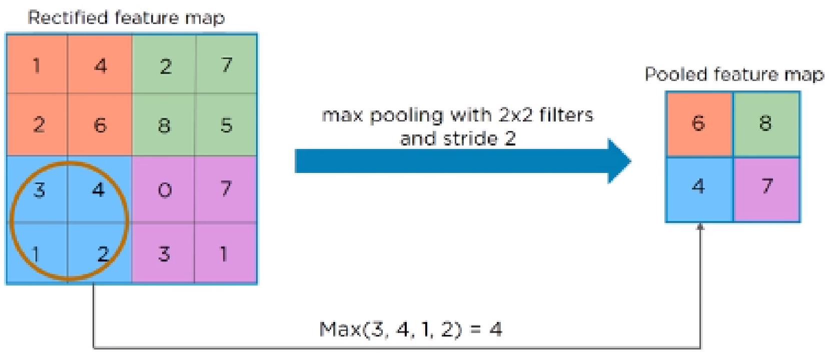


Figure 2.18: Maxpooling layer

– **Fully connected layers**

The last fully-connected layer is used to classify the input image of the network, this layer computes the score for each class from the extracted features from the first block[7] [8].

The examples given are simple when compared to the real CNN learning process, the objective was to explain how CNN layers do tasks.

This is an abstract example (see figure 2.19) of how the full process of learning is done by a CNN.

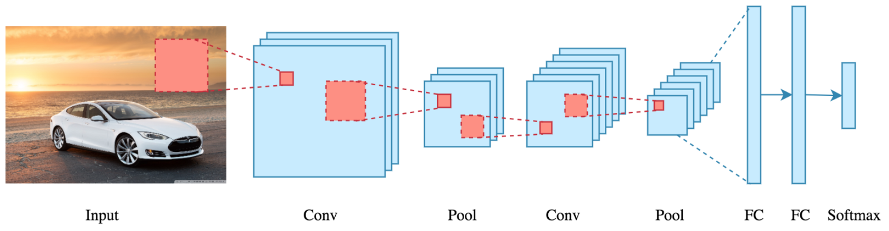


Figure 2.19: CNN learning process

2.6 Popular CNN Architectures

There is an infinite number of architecture, we will describe some of the most popular architectures like VGGNet, LeNet, and EfficientNet family.

- **VGGNet**

This architecture, which was one of the first to appear, was introduced by Simonyan and Zisserman in 2014. The VGGNet architecture in the figure 2.20 is composed of convolutional layers, ReLU as an activation function, Maxpooling layers are inserted between convoolutional layers in order to reduce the size of features parameters, the classificasion block is composed from dense layers which uses ReLU also, the final layer uses softmax for classification [37].

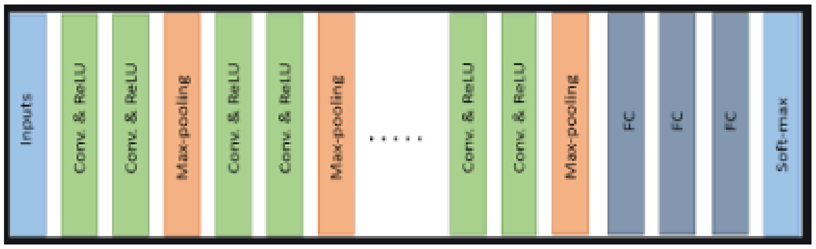


Figure 2.20: Basic architecture of VGGNet

- **EfficientNet**

This architecture or this family of architecture (B0,B1,..,B7), were realised by google in 2019. The EfficientNet-B0 architecture in figure 2.21 wasn't developed by engineers but by the neural network itself.They developed this model using a multi-objective neural architecture search that optimizes both accuracy and floating-point operations. Taking B0 as a baseline model, the authors developed a full family of EfficientNets from B1 to B7 which achieved state of the art accuracy on ImageNet while being very efficient to its competitors [9].

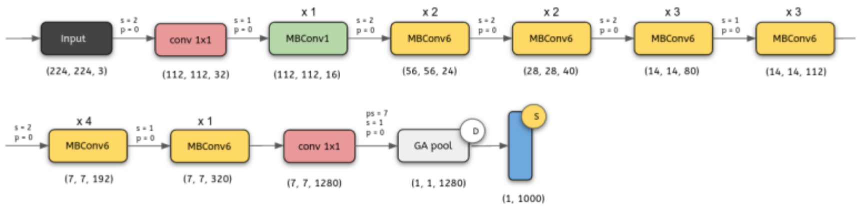


Figure 2.21: EfficientnetB0 architecture

- **LeNet**

  LeNet was introduced by Yan LeCun for digit recognition. The basic configuration of LeNet-5 is in the figure 2.22, 2 convolutions layers, 2 sub-sampling layers, 2 fully connected layers, and an output layer with the Gaussian connection. The total number of weights and Multiply and Accumulates (MACs) are 431k and 2.3M respectively.

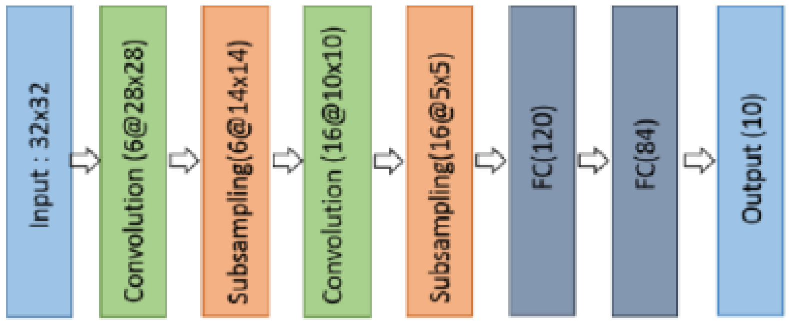


Figure 2.22: Basic architecture LeNet-5

2.7 Transfer learning

Transfer Learning is a technique of deep learning which allows Deep Learning to be done without the need to spend a lot of time on calculations. The principle is to use the knowledge acquired by a neural network when solving a problem to solve another more or less similar one. In this way, a transfer of knowledge is achieved [24].

Most of the popular CNN architectures can be used for transfer learning after some modifications.

2.8 Diabetic retinopathy

2.8.1 Diabetes

Diabetes is an abnormal rise in blood sugar. This increase in blood sugar can lead to damage to various organs, such as the eyes, heart, kidneys, nerves and blood vessels. Almost 90% of diabetics live with the disease for years without knowing it, as diabetes does not usually cause any symptoms at the beginning of its course. Diabetes is diagnosed when fasting blood glucose levels are greater than or equal to 1.26 g/l [17].

Diabetes can effect different parts of the body (see figure 2.23) like:

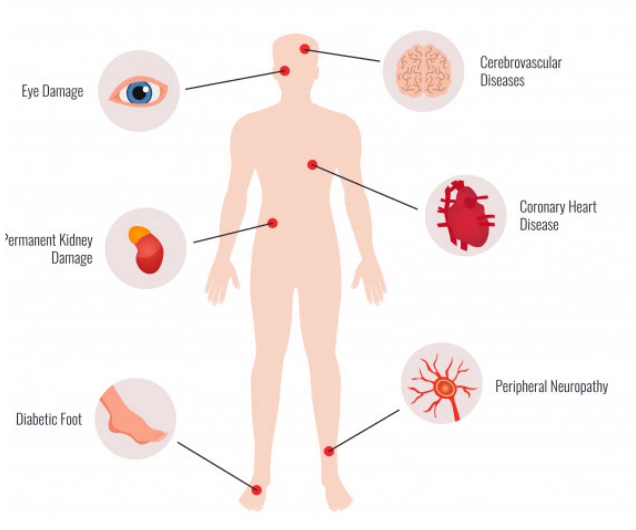


Figure 2.23: Effects of diabetes on human body

- **The effect of diabetes on the eyes**

  Diabetic retinopathy (eye and retina) is a serious complication of diabetes that affects 50% of type 2 diabetic patients[11]. The eyes are particularly susceptible to damage to small vessels.

- **The effect of diabetes on the heart**

  Diabetes can lead to cardiovascular complications as it also affects the large blood vessels.

- **The effect of diabetes on the kidneys**

  The kidney forms urine by filtering the blood. Because of diabetes, the kidney filter becomes clogged and can no longer eliminates certain waste products and allows molecules to pass into the urine that should not.

- **The effect of diabetes on the nerves**

  Diabetes can affect all the nerves in the body. It affects two types of nerves: the peripheral nerves that control muscles and sense the skin; and the nerves of the autonomic nervous system that control the functioning of the viscera.

2.8.2 Diabetic retinopathy

The disease often sets in without giving any warning signs.It is therefore possible to suffer from retinopathy even with good eyesight and in the absence of any symptoms. It is therefore important to have regular check-ups by a specialist and to have early detection. If the disease is allowed to spread, it will eventually affect the centre of the eye and the retina, creating serious and irreparable vision problems (see figure 2.24) [12].

Diabetic retinopathy symptoms:

- Eye pain or redness.
- blurred or patchy vision.
- gradually worsening vision.
- sudden vision loss.

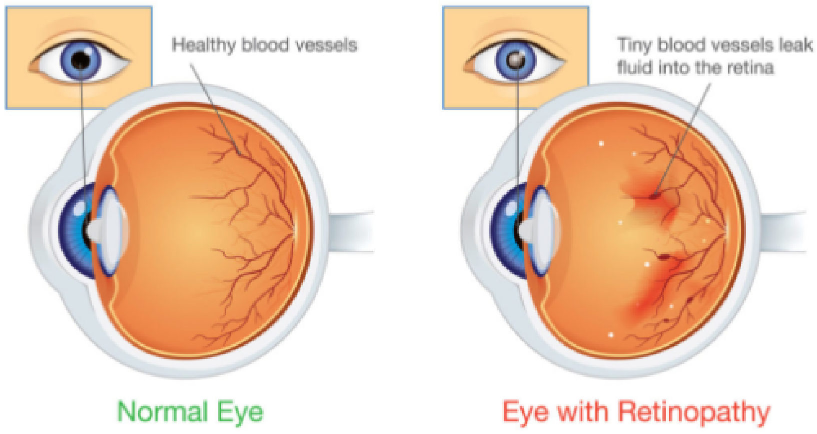


Figure 2.24: The difference between a normal retina and diabetic retinopathy retina

Levels of diabetic retinopathy

1. **Mild diabetic retinopathy**

The first stage of diabetic retinopathy (see figure 2.25), at this level the doctor can see tiny areas of swelling in the blood vessels of the retina [12].

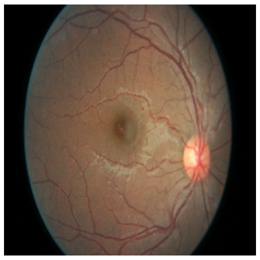


Figure 2.25: Mild diabetic retinopathy

2. **Moderate diabetic retinopathy**

   Increased swelling of tiny blood vessels begins to interfere with blood flow to the retina (see figure 2.26) [12].

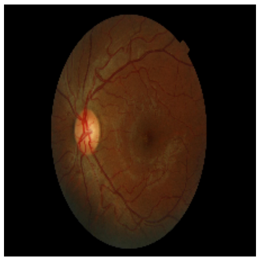


Figure 2.26: Moderate diabetic retinopathy

3. **Severe diabetic retinopathy**

   More of the blood vessels in the retina become blocked, causing a significant decrease in blood flow to that area. At this point, the body receives signals to start growing new blood vessels in the retina (see figure 2.27) [12].

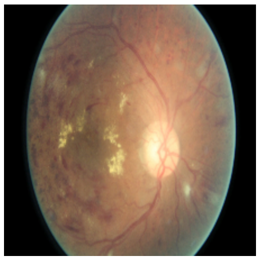


Figure 2.27: Severe diabetic retinopathy

4. **Proliferate diabetic retinopathy**

   This is an advanced stage of the disease, in which new blood vessels form in the retina. As these blood vessels are often fragile, the risk of fluid leakage is higher.This triggers various vision problems such as blurring, a reduced field of vision and even blindness (see figure 2.28)[12].

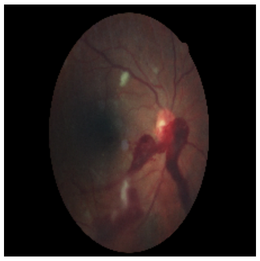


Figure 2.28: Proliferate diabetic retinopathy

2.8.3 Retinal imaging

Ophthalmologist uses a machine to get retina images (see figure 2.29) to monitor changes in the fundus This machine can be used to assess and monitor symptoms of retinal detachment, diabetic retino- pathy, vascular or inflammatory conditions and various eye diseases such as glaucoma [13].

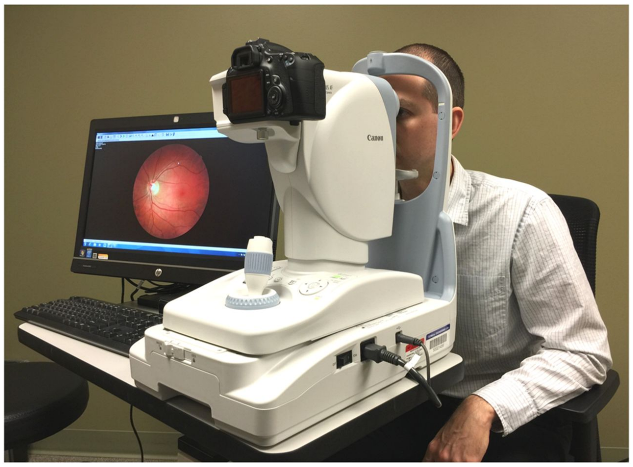


Figure 2.29: The machine used to get retina images

2.8.4 Related works

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

- **Healthcare using deep learning Diabetic retinopathy detection using deep learning**

  Berra erraid and Meadi nadjib, 2019 thesis from the university of mohamed khieder-Biskra, They used transfer learning with different architecture to classify DR images of the kaggle dataset into 2, 3, and 5 classes. for the preprocessing they have used gaussian filter and data augmentation.the image number used is 3662 .They reported an accuracy of 99.22% and a validation of 100% using EfficientNetB0 on binary classification, an accuracy of 98.82% and a validation of 100% using ResNet50 on 3 classes and an accuracy of 98.23% and a validation of 100% using RestNet50 on 5 classes [14].

- **Classification of diabetic and normal fundus images using new deep learning method**

  M. T. Esfahan et al. They 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 % [30].

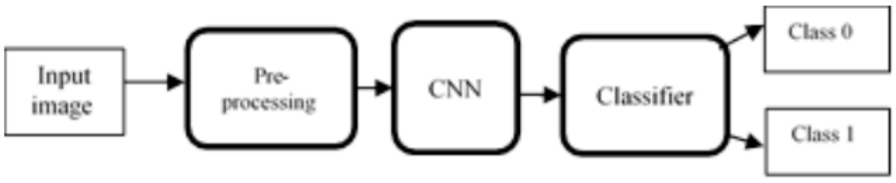


Figure 2.30: Block diagram of the proposed method

- **Decision tree CART algorithm for diabetic retinopathy classification**

  Aziza, Elaouaber Zineb, et al. Propose an automatic system for diabetic retinopathy detection from color fundus images. The proposed approach is based on the segmentation of blood vessels and extracts the geometric features, which are used in the early detection of diabetic retinopathy. The Hessian matrix, ISODATA algorithm and active contour are used for the segmentation of the blood vessels, they have used. Finally, they have applied the decision tree CART algorithm to classify images into normal (NO-DR) or DR (see figure 2.31). The proposed system was tested on the drive and Messidor datasets and achieved an average sensitivity, specificity and accuracy of 89%, 99% , and 96%, respectively for the segmentation of retinal vessels and 91%, 100% , and 93%, respectively for the classification of diabetic retinopathy[27].

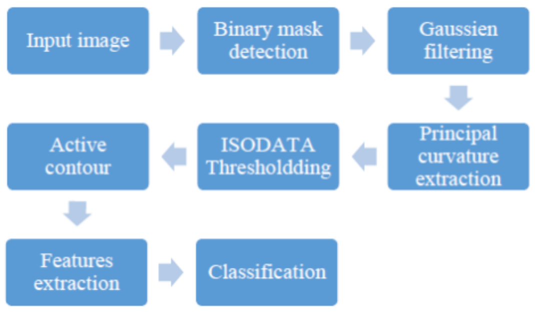


Figure 2.31: Flowchart of the proposed method

- **A convolutional neural network for the screening and staging of diabetic retinopathy**

  Mohamed Shaban, Zeliha Ogur et al, they proposed a CNN architecture (see figure 2.32) inspired form the VGGNet architecture, to classify DR images of the Kaggle dataset into normal, mild and moderate in the same level, sever and proliferate in the same level image, they augmented data , and resized images to (224,224) in order to improve the performance[36].

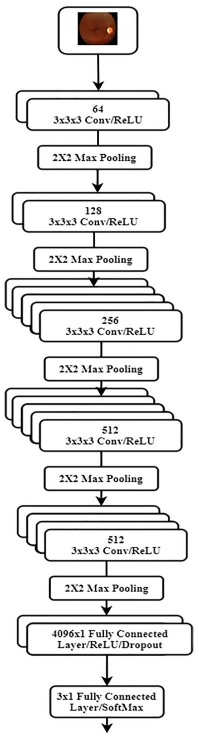


Figure 2.32: Proposed architecture

- **Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs**

  Varun Gulshan, Lily Peng et al, they trained CNN model using a retrospective development data set of 128 175 retinal images, which were graded 3 to 7 times for diabetic retinopathy, diabetic macular edema, and image gradability[31].

2.8.5 Conclusion

In this chapter we have covered some main aspects which are the basic of our current work: healthcare, Artificial intelligence, and diabetic retinopathy. We have described the different use cases of AI in healthcare, we detailed the basics of machine learning and deep learning then we have discussed diabetes and its complications, diabetic retinopathy, diabetic retinopathy levels, and finally we have mentioned some of the previous related works. The next chapter will introduce our system design and the implementation of a new deep learning architecture for the detection of the diabetic retinopathy.

Chapter 3

Design and implementation of a deep learning architecture for Diabetic Retinopathy detection

3.1 Introduction

In the past few years, diabetic retinopathy detection have gained huge attention, and due to the rapid development of Machine learning and Deep learning algorithms and the successful applications in this field, Deep learning convolutional neural networks have proven very effective in many areas such as image recognition applications, including those of diabetic retinopathy detection. In our project we are interested in creating a convolutional neural network in order to detect and classify diabetic retinopathy levels. Our models uses image processing methods to detect and classify diabetic retinopathy levels. In this chapter we present the system design, the different datasets and their different structures, the preprocessing phase, and finally our proposed CNN-architecture. We will mention the used tools, frameworks, libraries, and we will show how we have implemented our system.

3.2 System design

In order to build a deep learning Model for detecting and classifying diabetic retinopathy levels (No DR, mild , moderate, severe, proliferate), our System will be following certain steps depicted in figure 3.1.

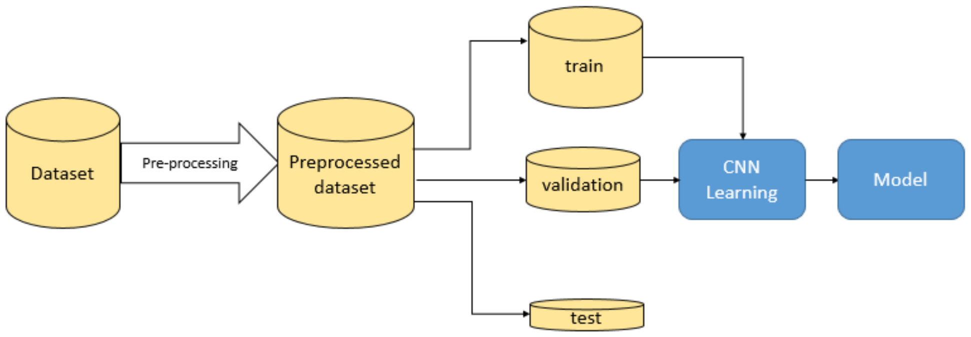


Figure 3.1: Diagram of our System

The system starts firstly by getting the dataset, do preproccessing on the dataset, split dataset, then we feed the CNN model with the splitted dataset, by the end we will have a model which can classify new diabetic retinopathy images.

3.2.1 Dataset

1. **Dataset description:**

   We have used 2 datasets from Kaggle which are available at Resized 2015 & 2019 Blindness Detection Image [40].

   (a) The first original (2019) Dataset is available at APTOS 2019 Blindness Detection [26]. This dataset was used in [36].

   (b) The second original (2015) dataset is available at Diabetic Retinopathy (resized) [32] and was used in [30].

   The third dataset from Kaggle is available at Diabetic retinopathy detection [33].

2. **Datasets structure:**

   (a) **Original structure of the available datasests:** In this dataset there is 5 classes (see figure 3.2) which are levels of diabetic retinopathy:

      i. **No DR** which represent **Level 0**: The patient retina has no diabetic retinopathy.

ii. **Mild** which represent **Level 1**: The patient retina has a mild diabetic retinopathy.

iii. **Moderate** which represent **Level 2**: The patient retina has a moderate diabetic retinopathy.

iv. **Severe** which represent **Level 3**: The patient retina has a Severe diabetic retinopathy.

v. **Proliferate DR** which represent **Level 4**: The patient retina has a proliferate diabetic retinopathy.

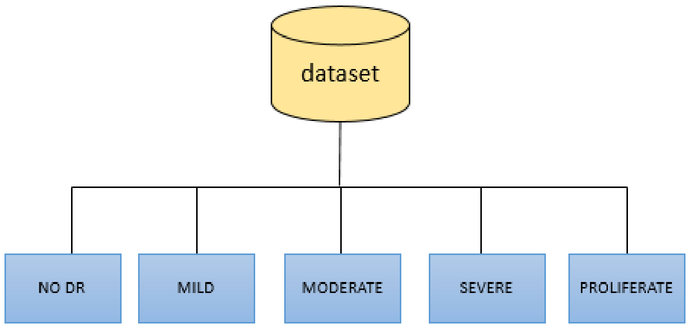


Figure 3.2: Original structure

(b) **Our first structure:** In this structure we will be working on Diabetic retinopathy detection, to clarify we will be working on 2 classes (see figure 3.3), which are:

i. **NO DR**: The patient retina is healthy and has no diabetic retinopathy.

ii. **YES DR**: The patient retina has diabetic retinopathy.

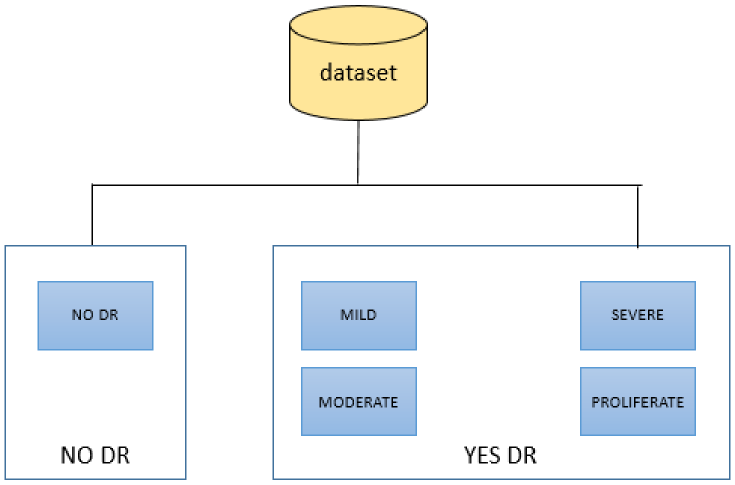


Figure 3.3: First proposed structure

(c) **Our second structure:** We proposed this structure (see figure 3.4)to replace the original (see figure 3.2) one due to the lack of data in the some classes (Mild -Severe - Proliferate) and we will be classifying 3 classes instead of 5:

i. **Level 0** : The patient retina has no diabetic retinopathy.

ii. **Level 1&2** : The patient retina has a level 1 or 2 diabetic retinopathy.

iii. **Level 3&4** : The patient retina has a level 3 or 4 diabetic retinopathy.

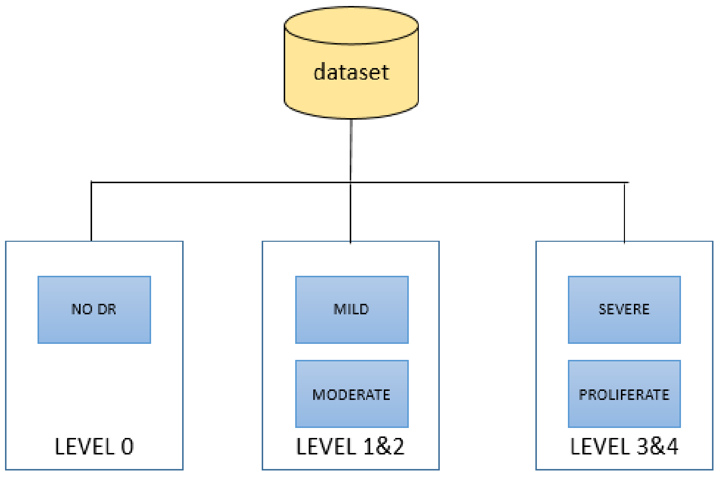


Figure 3.4: Second proposed structure

3.2.2 Preprocessing:

Before providing the data to CNN model for the training we need to pass by preprocessing, it is a crucial phase where we do modifications on data like resizing, applying filters, removing noises, etc. For that we applied gaussian filter then we resized all images.

1. **Gaussian filtering**:

   Gaussian filtering is used to blur images and remove noise and detail[25].

   In 2D dimension, the Gaussian function is:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

   (a) **Gaussian filtering for diabetic retinopathy**:

   we applied gaussian filtering on our dataset images, parameters and the way we applied this filter will be discussed in the implementation section.

   (b) **Gaussian filtering results**:

   By applying Gaussian filtering the quality of the retina has improved, blood vessels are clearer now (see figure 3.5).

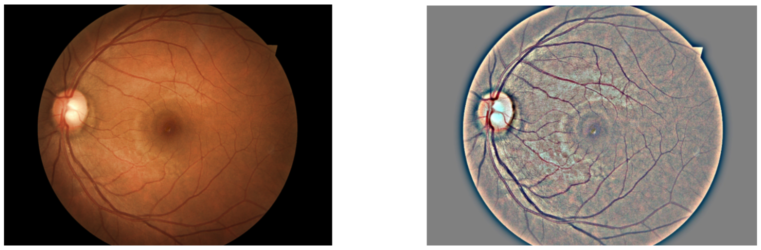


Figure 3.5: Gaussian filtering on diabetic retinopathy image

2. **Resize**:

   Original dataset images size were (1024x1024x3). We resized images to (224x224x3), where the first two values refers to the width and the height of the image, and the third one refers to the image channels, which means that images are in RGB (Red, Green, Blue).

   Why resizing images ?

If we do use images with a big size directly, we need to use a big input shape size in the CNN architecture, this will affect the number of parameters of the CNN model, and the training phase will take longer time.

3.2.3 Splitting dataset

In Deep learning, the splitting of dataset, is to divide the dataset into training subset, validation subset, and testing subset. Our dataset is splitted into 3 subsets with 70% images per class for training, 20% for validation, and 10% for testing.

3.2.4 CNN Learning

In order to make a CNN learn from the training subset and the validation subset, first we will detail layers parameters, hyper-parameters, optimizers, and other model parameters.

How a CNN model can learn?

The CNN is a mathematical construction with layers of different types, we detailed layers and blocks in the previous chapter page 17-18-19, each layer does a task on the image or on the set of images in order to get features that can be anywhere in the image. From one layer to another the features become more complicated. The performance of a model is measured using kernels, weights and a function through forward propagation (see figure 3.6) on a training dataset, and learnable parameters are updated according to the loss value through back propagation [41].

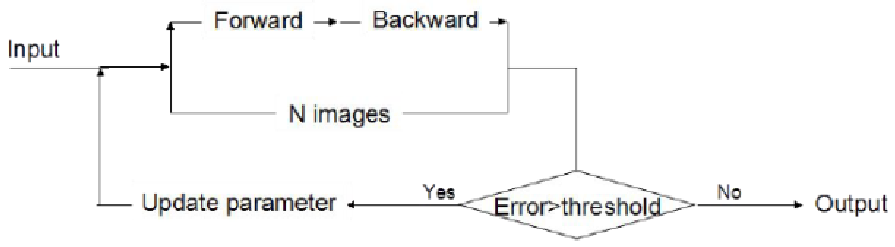


Figure 3.6: CNN Training algorithm

1. **Layers in CNN** :

A CNN is formed by a stack of distinct layers, convolution layers, pooling layers, fully connected layers, and dropout layers in particular cases, we described in chapter 2 page 18-19 CNN these layers, in the next table 3.1 we will detail the parameters and hyper-parameters of each layer.

	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 3.1: CNN layers parameters and hyper-parameters

Why we use dropout layer?

Dropout layer is used with an arbitrary rate(generally from 0.1 to 0.5)in order to avoid over fitting.

How dropout is done?

Dropout refers to dropping out units(from hidden and visible layers)in a neural network, to clarify those units from layers wont be used in the training.

2. **Loss function**

   Loss function quantifies how wrong would be if we use model to make prediction, we always try to minimize this function value.

3. **Optimizers in CNN**

   Optimizers are algorithms or methods used to change the attributes of a neural network such as weights and learning rate in order to reduce the losses. There are a number of available optimizers optimizers like:SGD, RMSprop, adam, adamx, etc.

4. **Other CNN parameters**:

   Before we passe to the model training, some parameters need to be explained due to the importance of these parameters.

- **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.
- **Padding**: One of the layers parameters, when this value is set to 'same', border elements of the shape wont be discarded.
- **Epoch**: When the entire dataset is passed through the CNN.
- **Batch size**: Total of training or validation example in a single batch,Our batch_size is set to 32 in our experimentations.
- **Shuffle**: When this value is set to 'True', the dataset will be shuffled before training.
- **Class weight**: How the CNN gives importance to classes, it is used when the dataset is imbalanced.
- **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.
- **Validation steps**: The number of batch iterations before a validation epoch is considered finished, it is generally calculated by dividing the number of samples in validation subset on batch size.
- **Lr or Learning rate**: The learning rate controls how quickly the model is adapted to the problem (usually very small).

3.2.5 Prediction

After the training phase, the prediction phase is the phase where a CNN model is ready to classify images, for us we have the third subset from the dataset splitting and we will use it to test our CNN model.

3.2.6 Evaluation of a CNN Model

In order to evaluate the performance of a CNN model, true positives(TP), true negatives(TN), false positives (FP), and false negatives(FN). These prediction ensembles are used to calculate variety of performance metrics including Accuracy, Precision , and F1score according to the following equations:

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

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

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

3.3 Implementation of a Deep Learning architecture

In the previous section we detailed our system design. The objective of this section is to mention tools and frameworks, implement our system design and present code.

3.3.1 Frameworks , tools and libraries

- **Python**

  Python is a programming language created by Guido van Rossum and first released in 1991, Python is a general-purpose, versatile, and powerful programming language.

  Figure 3.7: Python logo

- **Google Colab**

  Google Colab, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

while using Google colab code is executed in a virtual machine, Virtual machines are recycled when idle for a while.

Table 3.2: Google Colab resources

	GPU	**Runtime**	**RAM**	**disk capacity**
Google Colab	K80	12h	12GB	60GB
Google Colab pro	T4 & P100	24h	25GB	108GB

Figure 3.8: Google Colab logo

- **OpenCv**

  Developed by Intel, OpenCv provides a real-time optimized Computer Vision library, tools, and hardware.

Figure 3.9: OpenCv logo

- **Anaconda spyder**

  Spyder is a Python development environment (IDE) .

Figure 3.10: Python logo

- **Matplotlib**

  Matlplotlib is a plotting library for the python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+.

Figure 3.11: Matplotlib logo

- **Keras**

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

Figure 3.12: Keras logo

- **TennsorFlow**

  TensorFlow is open-source library for machine learning. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks and machine learning.

Figure 3.13: Tensorflow logo

- **NumPy**

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

Figure 3.14: NumPy logo

- **Kaggle**

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

kaggle

Figure 3.15: Kaggle logo

3.3.2 Dataset preparation and preprocessing

1. **Download Dataset to Google Colab**

   First of all we have downloaded the Datasets from Kaggle to our google Colab account using the dataset API.

   ```
   !kaggle datasets download -d benjaminwarner/resized-2015-2019-
      blindness-detection-images

   ```

2. **Functions and parameters of the used gaussin filter**

   The gaussin filtering shown in the last section (page 34 figure 3.5) was obtained by using 2 functions from OpenCv.

   (a) **cv2.addWeighted**(src1,alpha,src2,beta,gamma).

   src1 - first input.

   alpha – weight of the first array.

   src2 – 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 [21].

   (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 [22].

```python
import cv2
import os
import glob
os.chdir("/content/dataset")         # dataset path example
for file in glob.glob("*.jpg"):# 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
    gaussian = cv2.resize(gaussian, (224, 224))   # resize images
    cv2.imwrite(file,gaussian) # save image

```

Listing 3.1: preprocessing

3. **Splitting and preparing 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 3.2: Splitting Dataset

Now we have a splitted dataset, we will prepare it for the CNN Model.

```python
trdata = ImageDataGenerator()
traindata = trdata.flow_from_directory(directory="/content/out/train"#
   training dataset directory
,target_size=(224,224))
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory(directory="/content/out/val"#
   validation dataset directory
, target_size=(224,224))
```

Listing 3.3: Dataset preparation

	description
ImageDataGenerator()	Generate batches of tensor image data with real-time data augmentation.
target_size	if the images size is not equal to the target size , all images will be resized to (224,224) , for us images are already resized in the prepocessing phase

Table 3.3: Compiling Model arguments

3.3.3 Building our 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
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 3.4: necessary imports for building a CNN Molde

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**:

```
    model = Sequential()

```

Listing 3.5: Sequential model

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

(b) **Used CNN Architecture**

```
#Layer number 1
model.add(Conv2D(input_shape=(224,224,3),filters=32,kernel_size
    =(3,3),padding="same", activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 2
model.add(Conv2D(filters=32, kernel_size=(3,3), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 3
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 4
model.add(Conv2D(filters=64, kernel_size=(3,3), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 5
model.add(Conv2D(filters=128, kernel_size=(3,3), padding="same",
    activation="relu"))
model.add(MaxPool2D(pool_size=(2,2),strides=(2,2)))
#Layer number 6
model.add(Flatten())
model.add(Dense(units=512,activation="relu"))
#Layer number 7
model.add(Dense(units=256,activation="relu"))
model.add(Dropout(0.2))
#Layer number 8
model.add(Dense(number_of_classes, activation="softmax"))

```

Listing 3.6: Diabetic retinopathy Detection CNN Model Architecture

Layers	**Formed from**
Layer number 1	Conv2D(32,(3,3)),MaxPooling(2,2), padding("same"),activation("relu")
Layer number 2	Conv2D(32,(3,3)),MaxPooling(2,2), padding("same"),activation("relu")
Layer number 3	Conv2D(64,(3,3)),MaxPooling(2,2), padding("same"),activation("relu")
Layer number 4	Conv2D(64,(3,3)),MaxPooling(2,2), padding("same"),activation("relu")
Layer number 5	Conv2D(128,(3,3)),MaxPooling(2,2), padding("same"),activation("relu")
Layer number 6	Flatten(),Dense(512),activation("relu")
Layer number 7	Dense(256),activation("relu"), Dropout(dropOutRate)
Layer number 8	Dense(number_of_classes), activation("softmax")

Table 3.4: CNN Layers composition

We have used add function to add layers to our CNN model, for the 2 hyper-parameters number_of_classes and dropOutRate, these 2 hyper-parameters will be changed according to the dataset structure.

(c) **Used optimizer**

```
from keras.optimizers import Adam
opt = Adam(lr=1e-4)
```

Listing 3.7: 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= 0.0001.

(d) **Used loss function**

A loss function is the second required argument for compiling a Keras model, the loss function measures performance of a classification model.

(e) **Used metrics**

Metrics describe how good our model is doing in the training phase.

4. **Compiling CNN Model**

```
model.compile(optimizer=opt, loss=keras.losses.
    categorical_crossentropy, metrics=['Precision','accuracy','
    FalseNegatives','FalsePositives','TrueNegatives','TruePositives'])
```

Listing 3.8: Compiling Model

Function	**Arguments**
model.Compile()	optimizer = adam learning rate = 0.0001 Loss function = categorical crossentropy Metrics = ['Precision','accuracy','FalseNegatives' ,'FalsePositives','TrueNegatives','TruePositives']

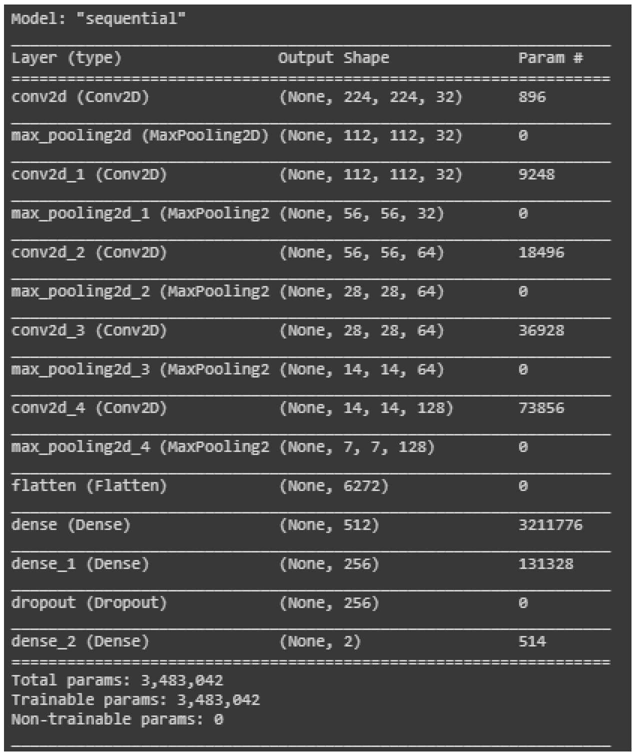
Table 3.5: Compiling Model arguments

5. **Model summary**

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

```
model.summary()
```

Listing 3.9: Model summary



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 224, 224, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 112, 112, 32)      0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 112, 112, 32)      9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 56, 56, 64)        18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 28, 28, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 14, 14, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 14, 14, 128)       73856
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 7, 7, 128)         0
_________________________________________________________________
flatten (Flatten)            (None, 6272)              0
_________________________________________________________________
dense (Dense)                (None, 512)               3211776
_________________________________________________________________
dense_1 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout (Dropout)            (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 514
=================================================================
Total params: 3,483,042
Trainable params: 3,483,042
Non-trainable params: 0
_________________________________________________________________
```

Figure 3.16: Model summary

6. **Training CNN Model**

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

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
checkpoint = ModelCheckpoint("modelName.h5", monitor='monitor',
    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(steps_per_epoch=len(traindata),generator=
    traindata, validation_data= testdata, validation_steps=len(testdata
    ),epochs=100,shuffle= True,class_weight=class_weights,callbacks=[
    checkpoint,early])
```

Listing 3.10: Fitting Model

-

- **ModelCheckpoint**

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

Function	**Arguments**
ModelCheckpoint	-Model name -monitor of performance -verbose mode 0,1 or 2 , we used 1 for an animated progress -save_best_only , if = true it only saves when the model is considered the "best" -save_weights_only , if = false , the full model is saved -mode ,one of max,min or auto ,in our case the save_best_only will overwrite the mode mode period is just an additional argument for backwards compatibility

Table 3.6: Function description for the Checkpoint

- **EarlyStopping**

  Early stop is a method, used to specify arbitrary the number of epochs according to a monitor once the model performance stops improving, the main objective of this method is to avoid over fitting.

Function	Arguments
EarlyStopping	-monitor of performance min_delta is the minimum change in the , monitored quantity if the change is less than min_delta ,will count as no improvement - patience is the number of epochs with no improvement after which training will be stopped verbose mode if = 0 , silent , no animation -mode is one of max,min or auto ,here it will follow the monitor if val accuracy then mode will be max,if val_loss then the mode will be min

Table 3.7: Function description for the earlyStopping

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

Function	Arguments
fit_generator	-step per epoch is calculated using len(traindata) -validation_steps is calculated using len(testdata)) -epochs = 100 -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 2 previous functions

Table 3.8: Function description for the model fitting

Since we have different data structures, some parameters will be changed according to the structure, in the next chapter we will give details and training results for each structure .

3.4 Testing our CNN Model

In order to test our model on the third subset (test) we have used this code. This code was used on the first proposed structure model, with some changes we used the same code for the other structures.

```
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 DR counter
total=0# total of images

f = r'/content/out/test/NO' #path of third subset for the no DR

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 predected , it is a matrix
     with j(second index of the matrix)size equal to the  same number of
   classes


    if output[0][0]>output[0][1] : # if the probability that this image
   belongs to NO DR is bigger than the probability of that this image
   belongs to the YES DR  .
        print("NO ", output[0])
        indiceDR0 += 1
        total += 1
    if output[0][1]>output[0][0]:
        print("YES ", output[0])
        indiceDR12 += 1
        total += 1
print("====================")
print("NO DR predict")
print("====================")
print("NO",indiceDR0)
print("YES",indiceDR12)
```

Listing 3.11: Testning Model

3.5 Conclusion

In this chapter we have presented our datasets structures, we have described also our system design, what we did exactly in the preprocessing phase, we have mentioned the used tools, libraries and frameworks used, we have presented the implementation of a big part of our system and we have detailed about our CNN model. In the next chapter, we will discuss different experimentations and obtained results.

Chapter 4

Experimentation and Results

4.1 Introduction

In the previous chapter, we have described our system design, we have showed the structures of the Datasets, and we have also presented code for each system design phase. In this chapter, we will specify the remaining parameters, hyper-parameters, and we will use the proposed structures. By the end of each subsection, we will discuss our experiments and the obtained results. Finally, we will compare our obtained results with some of the previous related works.

4.2 First Dataset

The next table 4.1 shows the original dataset structure[1] with the number of images per class (NO DR: patient has no diabetic retinopathy, Mild: patient has a mild diabetic retinopathy, Moderate: patient has a moderate diabetic retinopathy, Severe: patient has a sever diabetic retinopathy, Profile DR: patient has a proliferate diabetic retinopathy).

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

Table 4.1: Original structure Dataset 1

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

4.2.1 First proposed Structure (Binary classification)

This structure will be used to train a model for detecting diabetic retinopathy, which means we will have 2 levels: NO DR and YES DR, as we have mentioned before. The Dataset will be splitted into 3 subsets: training subset, validation subset and testing subset.

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

Table 4.2: first proposed structure Dataset 1

1. **Specification of parameters initialization for this structure:** In this first structure, we propose the following set of parameters in the table 4.3.

	Informations
input shape	(224,224,3)
Number of Epochs	31 early stopped
Step per Epoch	total training images / batch size =81
Class weight	not used , balanced dataset
dropOutRate	20%
Patience	15
Last output Layer	Dense(2)
ModelCheckpoint Monitor	val_acc
EarlyStopping Monitor	val_loss

Table 4.3: Table of parameters for first proposed structure

2. **Experimentation**: We have evaluated the model using 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:

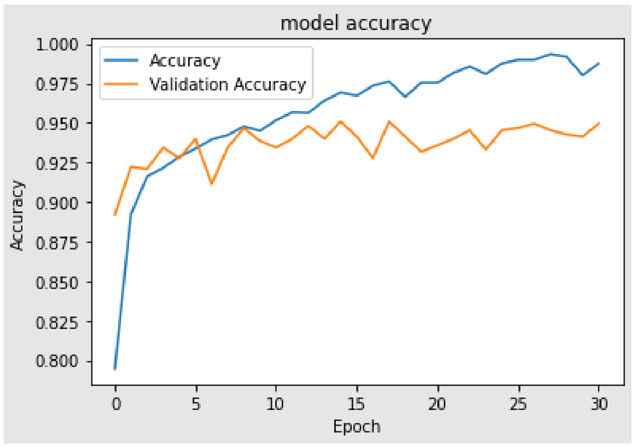


Figure 4.1: The model Accuracy

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

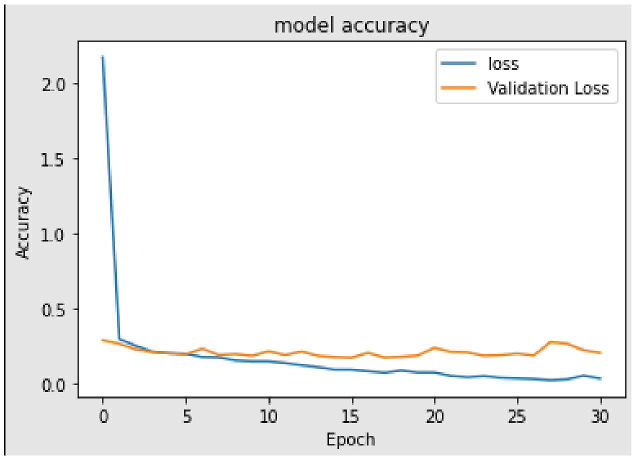
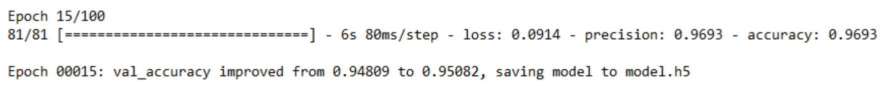


Figure 4.2: The model Loss

   - The plots of learning curves 4.1 and 4.2 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.

3. **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 4.3.



```
Epoch 15/100
81/81 [==============================] - 6s 80ms/step - loss: 0.0914 - precision: 0.9693 - accuracy: 0.9693

Epoch 00015: val_accuracy improved from 0.94809 to 0.95082, saving model to model.h5
```

Figure 4.3: Saving best model first structure Dataset 1

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	96.93%	95.08%	9,14%	17,07%

Table 4.4: Table of results first proposed structure first Dataset

4. **Testing Model**

   In this phase, we will use the third subset, in order to test how good our CNN model is doing on new images from the Dataset.

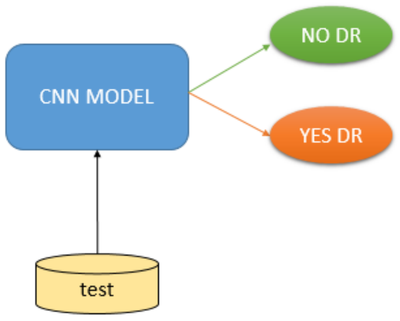
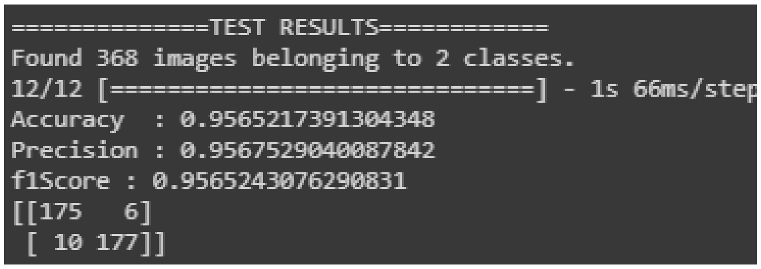


Figure 4.4: Model for detection DR first structure

5. **Results and discussion**

   After testing our model on the third subset:

   - **Metrics results on test subset**



```
=============TEST RESULTS===========
Found 368 images belonging to 2 classes.
12/12 [==============================] - 1s 66ms/step
Accuracy  : 0.9565217391304348
Precision : 0.9567529040087842
f1Score : 0.9565243076290831
[[175   6]
 [ 10 177]]
```

Figure 4.5: Model testing first structure Dataset 1

- **Confusion matrix**

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

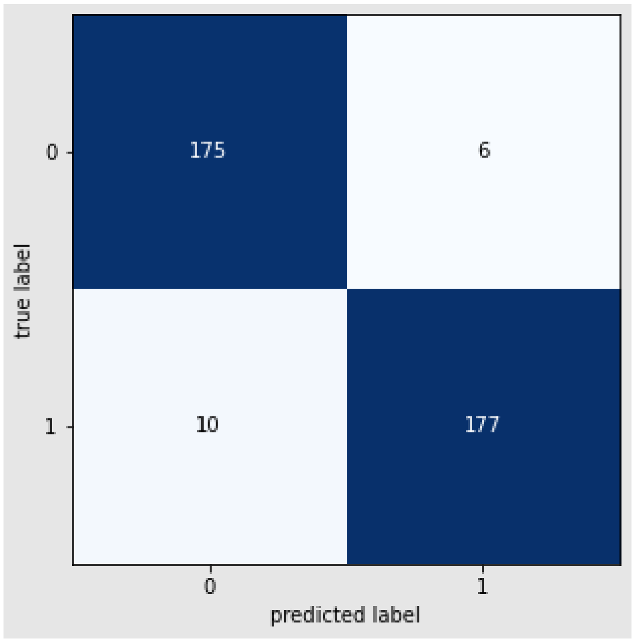


Figure 4.6: Confusion matrix first structure Dataset 1

  – The model could predict 175 images right out of 181 from the NO DR test folder.

  – The model could predict 177 images right out of 187 from the YES DR test folder.

  – Our model could detect and make difference between the 2 classes.

4.2.2 Second proposed structure

We have proposed this structure to replace the original one since we have a high unbalance between classes as well the 2 last classes are poor (give the number of images of each class). The idea is to keep level 0 as NO DR, to group the level 1 with level 2, the level 3 with level 4, after splitting dataset with this structure.

	Training	Validation	Test	Total
Lvl 0	1263	361	181	1805
Lvl 1&2	958	273	138	1369
Lvl 3&4	341	97	50	488
Total	2562	731	369	3662

Table 4.5: Second proposed structure Dataset 1

1. **Specification of parameters initialization for this structure:** The following table 4.6 provides the values of the used parameters.

	Informations
input shape	(224,224,3)
Number of Epochs	23 early stopped
Step per Epoch	total training images / batch size = 81
Class weight	'balanced'
dropOutRate	10%
Patience	30
Last output Layer	Dense(3)
ModelCheckpoint Monitor	Val_acc
EarlyStopping Monitor	val_loss

Table 4.6: Table of parameters for second proposed structure Dataset 1

2. **Experimentation:** We have evaluated the model using training accuracy, validation accuracy, training loss and validation loss. To visualise the performance of the model, we give the following plots.

   - A plot of accuracy and validation accuracy over epochs:

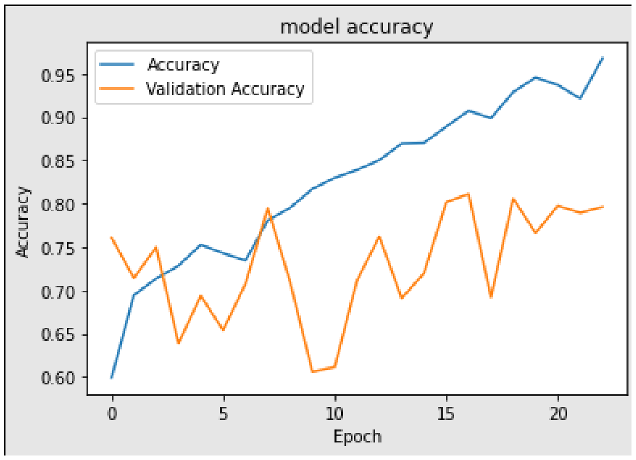


Figure 4.7: The model Accuracy

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

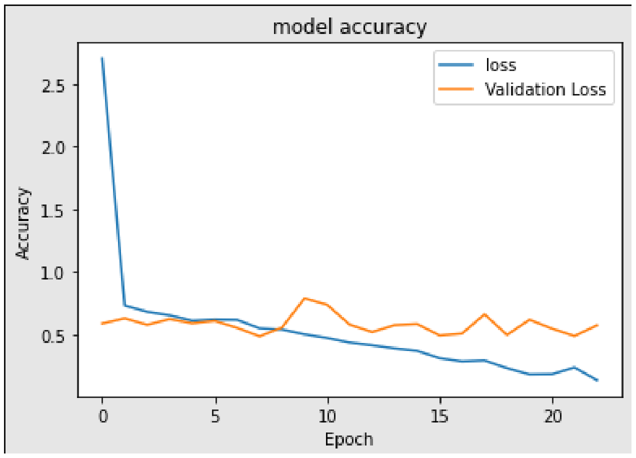


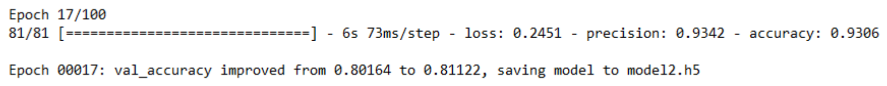
Figure 4.8: The model Loss

   - The plots of learning curves 4.7 and 4.8 show a good Fit.
     - The accuracy and validation accuracy are converging to the same values .
     - The loss and loss validation are decreasing to the same values .

- The lack of data in the last class 3&4 had an effect on the learning .

3. **Saving Model**

   The best model was saved in the epoch number 17, the early stopping was in the epoch 23. Performance metrics of this model are in the table 4.7.



```
Epoch 17/100
81/81 [==============================] - 6s 73ms/step - loss: 0.2451 - precision: 0.9342 - accuracy: 0.9306

Epoch 00017: val_accuracy improved from 0.80164 to 0.81122, saving model to model2.h5
```

Figure 4.9: Saving best model second proposed structure Dataset 1

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	93.06%	81.12%	24.51%	50.57%

Table 4.7: Table of results second proposed structure first Dataset

4. **Testing Model**

   In this phase, we will use the third subset, in order to test how good our CNN model is doing on new images from the dataset.

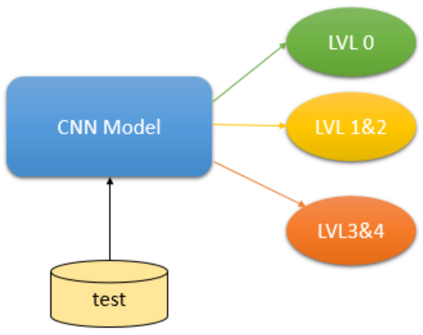
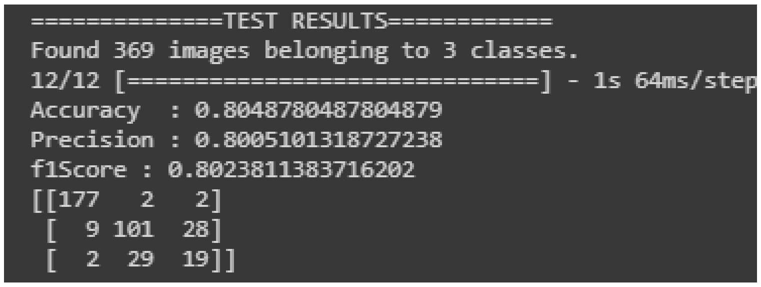


Figure 4.10: Model classifying DR second structure

5. **Results and discussion**

   After testing our model on the third subset:

   - **Metrics results on test subset**



```
==============TEST RESULTS============
Found 369 images belonging to 3 classes.
12/12 [==============================] - 1s 64ms/step
Accuracy  : 0.8048780487804879
Precision : 0.8005101318727238
f1Score : 0.8023811383716202
[[177   2   2]
 [  9 101  28]
 [  2  29  19]]
```

Figure 4.11: Model testing second structure Dataset 1

- **Confusion matrix**

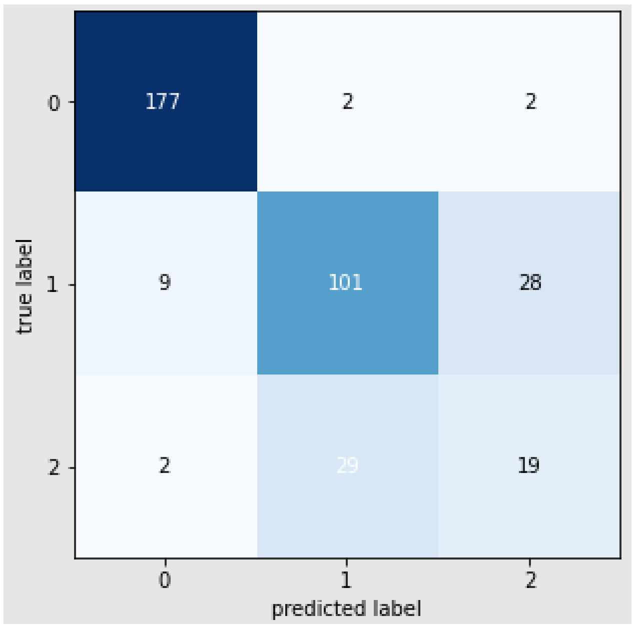


Figure 4.12: Confusion matrix second structure Dataset 1

  - The model could predict 177 images right out of 181 from the LVL0 test folder.
  - The model could predict 101 images right out of 138 from the LVL1&2 test folder.
  - The model predicted only 19 images right out of 50 from the LVL3&4 test folder.
  - Our model could detect and make difference between the 2 first classes but not the third class. This is caused by the lack of data in the third class. Our model could not learn much feature to make the difference, we have modified the Dataset in order to solve the problem. This modification provides a new structure which will be used in the section 4.2.3.

4.2.3 Second proposed structure with modifications

We have modified the first dataset by replacing its 2 last classes (severe DR, proliferate DR) by another 2 classes from the second dataset (see figure 4.13. This modification aims to provide more images in the last 2 classes.

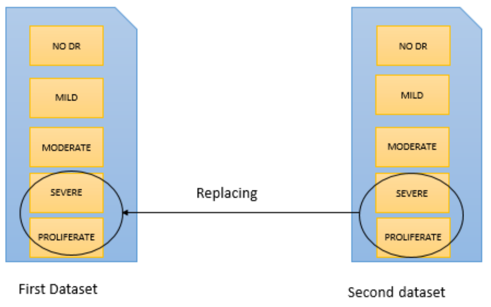


Figure 4.13: Modification of the first Dataset

Now we will work normally on 3 classes Lvl 0, Lvl 1&2 and a larger Lvl 3&4, after splitting the dataset, as depicted in the following table.

	Training	Validation	Test	Total
Lvl 0	1263	361	181	1805
Lvl 1&2	958	273	138	1369
Lvl 3&4	1106	316	159	1581
Total	3327	950	478	4755

Table 4.8: Second proposed structure Dataset 1 with modifications

1. **Specification of parameters initialization for this structure:** In this third structure (the second one with modification), we propose the following set of parameters in the table 4.9.

	Informations
input shape	(224,224,3)
Number of Epochs	37 early stopped
Step per Epoch	total training images / batch size = 104
Class weight	not used , balanced Dataset
dropOutRate	10%
Patience	30
Last output Layer	Dense(3)
ModelCheckpoint Monitor	Val_acc
EarlyStopping Monitor	val_loss

Table 4.9: Table of parameters for second proposed structure with modifications

2. **Experimentation**

   We have evaluated the model using training accuracy, validation accuracy, training loss and validation loss. To visualise the performance of the model, the following plots are provided.

   - A plot of accuracy and validation accuracy over epochs:

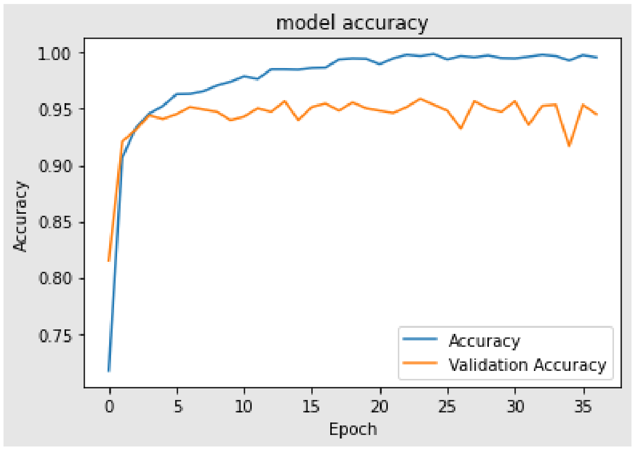


Figure 4.14: The model Accuracy

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

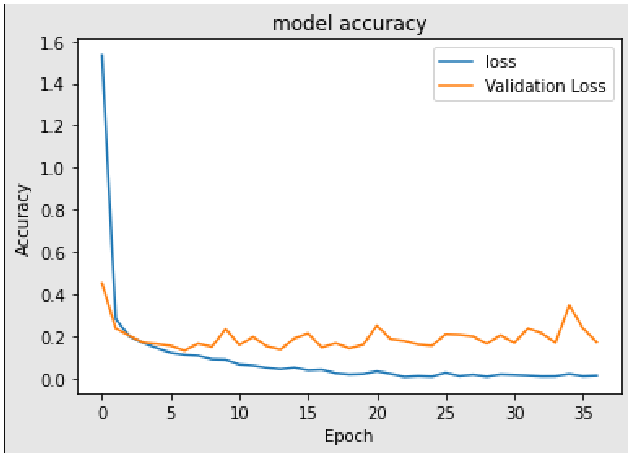
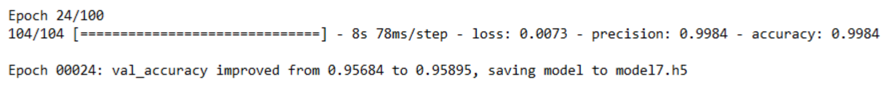


Figure 4.15: The model Loss

- The plots of learning curves 4.14 and 4.15 show a good Fit because:
  - The accuracy and validation accuracy are converging to the same values.
  - The loss and loss validation are decreasing to the same values.

3. **Saving Model**

   The best model was saved in the epoch number 17, and the early stopping was in the epoch 23. Performance metrics of this model are in the table 4.10.



```
Epoch 24/100
104/104 [==============================] - 8s 78ms/step - loss: 0.0073 - precision: 0.9984 - accuracy: 0.9984

Epoch 00024: val_accuracy improved from 0.95684 to 0.95895, saving model to model7.h5
```

Figure 4.16: Saving best model second proposed structure with modifications first Dataset

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	99.84%	95.89%	00.73%	16.12%

Table 4.10: Table of results second proposed structure with modifications first Dataset

4. **Testing Model**

   In this phase, we will use the third subset, in order to test how good our CNN model is doing on new images from the Dataset.

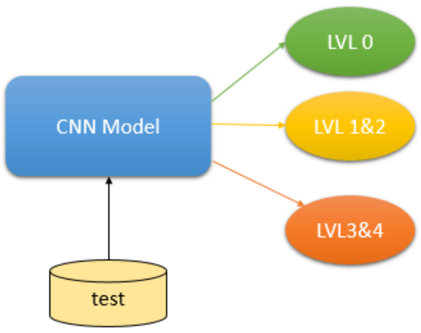
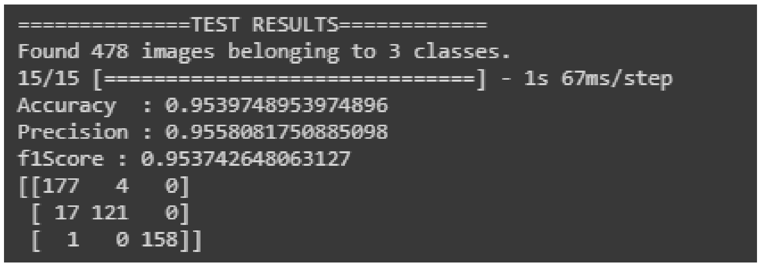


Figure 4.17: Model classifying DR second structure woth modifications

5. **Results and discussion**

   After testing our model on the third subset:

   - **Metrics results on test subset**



```
==============TEST RESULTS============
Found 478 images belonging to 3 classes.
15/15 [==============================] - 1s 67ms/step
Accuracy  : 0.9539748953974896
Precision : 0.9558081750885098
f1Score : 0.953742648063127
[[177   4   0]
 [ 17 121   0]
 [  1   0 158]]
```

Figure 4.18: Model testing second structure with modifications

- **Confusion matrix**

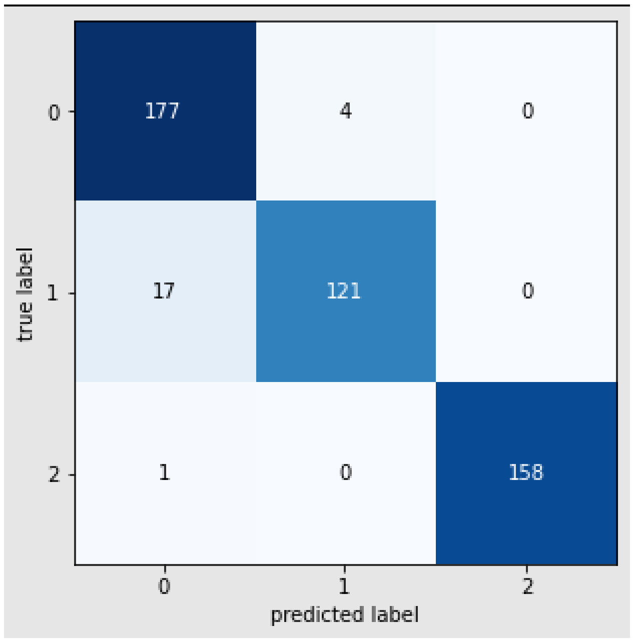


Figure 4.19: Confusion matrix second structure with modifications

- The model could predict 177 images right out of 181 for the LVL0 test folder.
- The model could predict 121 images right out of 138 for the LVL1&2 test folder.
- The model could predict 158 images right out of 159 for the LVL3&4 test folder.
- This update makes the model finding better features for the third class. Indeed, now the model makes difference between the 3 classes and could classify correctly.

4.3 Second dataset

For this dataset[1], we have taken our time working on the first proposed structure, since this structure is the less complicated and it has 2 classes only. The table below describes the number of samples for this Dataset before the splitting.

[1]https://www.kaggle.com/tanlikesmath/diabetic-retinopathy-resized

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

Table 4.11: Original structure Dataset 2

4.3.1 One proposed structure (Binary classification)

It is always better to balance the data manually if we can. In order to do that, in this structure we have taken 9500 images from the NO DR class and all the images from other classes. The table 4.12 describes the splitted dataset.

	Training	Validation	Test	Total
NO DR	6650	1900	951	9501
YES DR	6521	1863	932	9316
Total	13171	3763	1883	18817

Table 4.12: the proposed structure Dataset 2

1. **Specification of parameters initialization for this structure:**

   Notice: we have decreased the learning rate for this dataset from **0.0001** to **0.00001**.

   The following table 4.13 pro-vides the values of the used parameters.

	Informations
input shape	(224,224,3)
Number of Epochs	50 early stopped
Step per Epoch	total training images / batch size = 412
Class weight	not used , balanced Dataset
dropOutRate	10%
Patience	30
Last output Layer	Dense(2)
ModelCheckpoint Monitor	Val_acc
EarlyStopping Monitor	val_loss

Table 4.13: Table of parameters for second Dataset

2. **Experimentation**

   We have evaluated the model using training accuracy, validation accuracy, training loss and validation loss. To visualise the performance of the model, we give the following plots.

   - A plot of accuracy and validation accuracy over epochs:

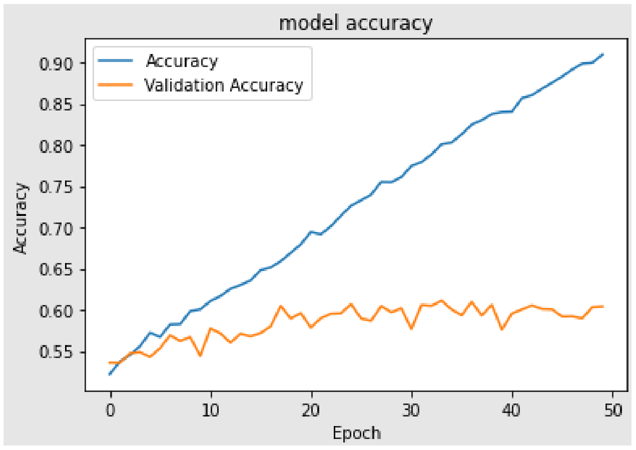


Figure 4.20: The model Accuracy

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

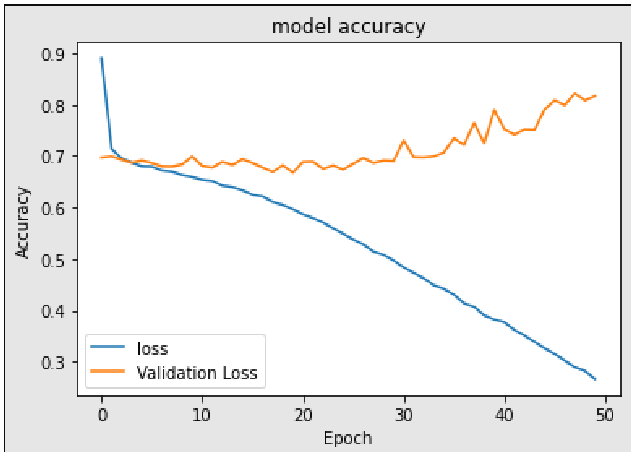
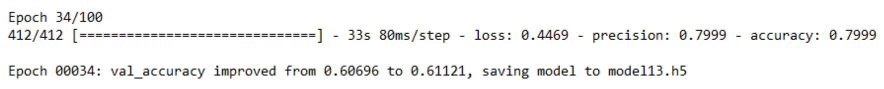


Figure 4.21: The model Loss

- The plots of learning curves 4.20 and 4.21 show an over Fitting because:
  - The accuracy and validation accuracy are not converging to the same values.
  - The loss is decreasing but not the validation loss.

3. **Saving Model** The best model was saved in the epoch number 34, the early stopping was in the epoch 50. Performance metrics of this model are in the table 4.14.



```
Epoch 34/100
412/412 [==============================] - 33s 80ms/step - loss: 0.4469 - precision: 0.7999 - accuracy: 0.7999

Epoch 00034: val_accuracy improved from 0.60696 to 0.61121, saving model to model13.h5
```

Figure 4.22: Saving best model second dataset

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
CNN Model	79.99%	61.12%	44.69%	69.90%

Table 4.14: Table of results second Dataset

4. **Testing Model**

   In this phase, we will use the third subset, in order to test how good our CNN model is doing on new images from the dataset.

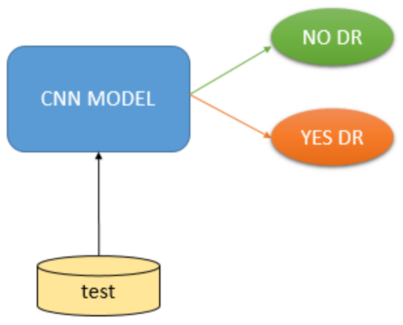
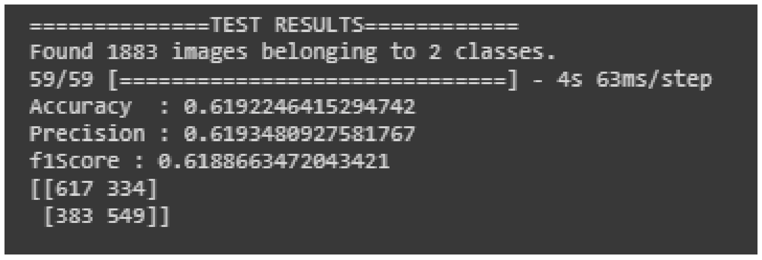


Figure 4.23: Model for detection DR first structure

5. **Results and discussion**

   After testing our model on the third subset:

   - metrics results on test subest:



```
==============TEST RESULTS============
Found 1883 images belonging to 2 classes.
59/59 [==============================] - 4s 63ms/step
Accuracy  : 0.6192246415294742
Precision : 0.6193480927581767
f1Score : 0.6188663472043421
[[617 334]
 [383 549]]
```

Figure 4.24: Model testing Second Dataset

- **confusion matrix**

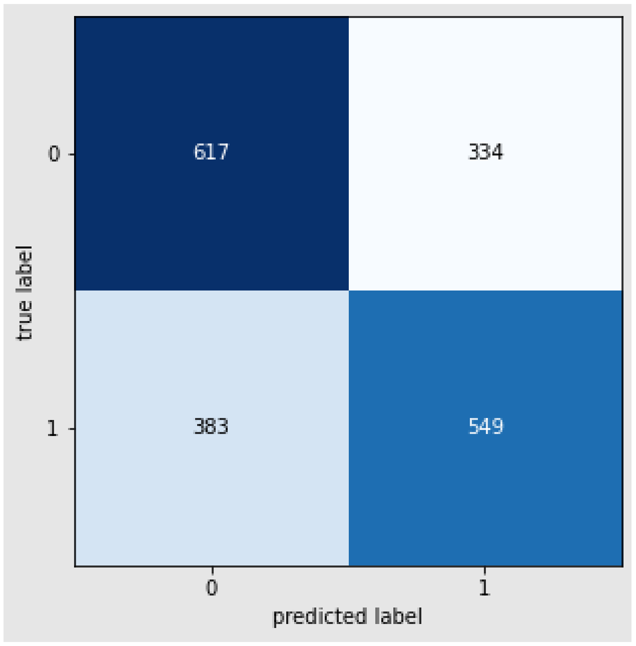


Figure 4.25: Confusion matrix Second Dataset

  - The model could predict 617 images right out of 951 for the NO DR test folder.
  - The model could predict 549 images right out of 932 for the YES DR test folder.
  - Our model is finding difficulty to make difference between the 2 classes, in the next subsection we will describe all what we tried like solutions.

6. **Other tried ideas to improve results**

   In order to improve our model performance for the second dataset, we have tried many solutions and updates on the second datset. However, those improvements were not considerable. All tried solutions are presented in table 4.15.

Manipulated parameters	**tested values**	**Description**
Optimizers	-Adam with different Lr. -Adamx with different Lr. -SGD with different Lr. -RMSprop with different Lr.	Optimizers are algorithms used to change the attributes of a neural network
Input shape	(224,224,3) (299,299,3) (512,512,3) (1024,1024,3)	It is an image with the first two values refers to the width and the height of the image, where the third one refers to the image channels
Learning rates	0.001 0.0001 0.00001	learning rate controls how quickly the model is adapted to the problem
Filter applied	grey scale guassian blur only	Filter tried in preprocessing phase
Transfer learning	-Efficentnet B0 -Efficentnet B4	Pretrained models tried
Data agumentation	-Augment both classes to 25 000 samples -Augment both classes to 50 000 samples -Augment both classes to 100 000 samples	Different data augmentation ways were tried in preprocessing phase to improve model performance
Architecture size	20 millions parameters 50 millions parameters 150 millions parameters	Many layers configurations were tried

Table 4.15: Tried solutions

Unfortunately, all these parameters were leading to an Over fit with a validations accuracy

< 65.

4.4 Third dataset

This is the biggest diabetic retinopathy detection dataset[1] available on Kaggle. It has the same original structure with 5 classes (No DR, mild, moderate, severe, proliferate), but with a huge number of samples **88 708**. The size of all zipped files is 82 GO. Unfortunately, we couldn't use this dataset since the disk capacity in Google Colab Pro is limited.

4.5 Comparison of our work with previous works

In this section, we will compare different results from different related works with ours. We have chosen accuracy, validation accuracy and dataset organization as Comparison criteria. We have also mentioned which dataset was used in each work.

4.5.1 Table of Comparison

We must explain some important points before presenting the table.

1. Since we have modified the dataset in the third experimentation, we will not compare its results with others works results. In deep learning, we compare results only when we use the same dataset.

2. For the last experimentation, our model did not learn and no solution was found even after trying many different parameters. The model was over fitting and so that this experimentation results will not be compared also. Indeed, when the model over fits its accuracy is no more realistic.

3. For the fifth related work [31] in page 29, we can not compare it with other works since the dataset used has an other structure and the accuracy was not mentioned in the paper [31], however the precision was equal to 96%.

[1]https://www.kaggle.com/c/diabetic-retinopathy-detection/data

‘

	Accuracy	**Validation accuracy**	**Dataset organization**	**Used Dataset**
Our architecture with first structure	96.93%	95.08%	2	First Dataset
Our architecture with second structure	93,06%	81,12%	3	First Dataset
EfficientNetB0 [14]	99.22%	100%	2	First Dataset
ResNet50 as mentioned in the work [14]	98.82%	100%	3	First Dataset
ResNet50 as mentioned in the work [14]	98.23%	100%	5	First Dataset
ResNet34 [30]	85%	-	2	Second Dataset
CNN Model Inspired from VGGNet [36]	92%	89%	3	First Dataset
Decision tree [27]	92%	89%	2	-

Table 4.16: Table of comparison

From the table 4.16, we can conclude that the more we have classes the less accuracy we get, also our architecture is performing good compared with the most popular and powerful architectures.

4.6 Conclusion

In this chapter we have detailed the different parameters for each structure, we have worked on different datasets, we have explained all experimentations, and we have illustrated results though plots and the confusion matrix for the test phase. The results obtained represent particularly good performances which encourage the improvement of our model architecture.

Chapter 5

Conclusion and Perspectives

The application of Deep Learning (DL) in healthcare attracts lot of researchers. This new approach have proved to be suitable in several applications. The diabetic retionpathy is one of the most serious diseases that affect a large amount of population over the world.

Throughout this dissertation, we have proposed a variant of deep learning architectures for the detection and classification of diabetic retinopathy levels. Indeed, we have investigated several works and architectures that were proposed to deal with images classification and diabetic retinopathy detection. Based on those proposed work, we have tried to propose a variant architecture to improve the existing results. Practically, two datasets [26] [32] were used to train our models and to validate their results.

The results have shown that Convolutional Neural Networks (CNN) models can be used in medical diagnosis, our model showed good results in term of classifying using the first dataset [26] but not the second one [32]. We did not tried enough preprocessing methods or other deep learning techniques due to hardware and time limitations.

Finally, this work opens the field to several future works such as the improvement of the model by using the third dataset, and many other deep learning techniques, and other preprocessing methods.

References

[1] Le diabete dans le monde. URL:https://www.federationdesdiabetiques.org/information/diabete/chiffres-monde.

[2] Overview of neuron structure and function. URL:https://www.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function.

[3] The relation between artificial and biological neuron? URL:https://smhatre59.medium.com/what-is-the-relation-between-artificial-and-biological-neuron-18b05831036.

[4] Activation function. URL: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0.

[5] softmax function. URL:https://deepai.org/machine-learning-glossary-and-terms/softmax-layer.

[6] reseau de neurones convolutif. URL:https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn.

[7] Convolutional neural network tutorial. https://www.simplilearn.com/tutorials/deep-learning-tutorial/convolutional-neural-network.

[8] les differentes couches d'un cnn. https://openclassrooms.com/en/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5083336-decouvrez-les-differentes-couches-dun-cnn.

[9] Image classification with efficientnet. URL:https://medium.com/analytics-vidhya/image-classification-with-efficientnet-better-performance-with-computational-efficiency-f480fdb00ac6.

[10] La retinopathie diabetique et les maladies des yeux. URL: https://www.federationdesdiabetiques.org/information/complications-diabete/retinopathie#:~:text=La%20rétinopathie%20diabétique%20(atteinte%20des,de%20cécité%20avant%2065%20ans..

[11] diabetes complications. URL:https://www.federationdesdiabetiques.org/information/complications-diabete/retinopathie.

[12] Diabetic retinopathy stages. URL:https://www.healthline.com/health/diabetes/diabetic-retinopathy-stages#symptoms.

[13] Non-mydriatic fundus photography: a practical review for the neurologist. URL:https://pn.bmj.com/content/16/5/343.

[14] Healthcare using deep learning diabetic retinopathy detection using deep learning. URL:https://www.theses-algerie.com/2522440798901925/memoire-de-master/universite-mohamed-khider---biskra/healthcare-using-deep-learning-diabetic-retinopathy-detection-using-deep-learning?size=n_10_n.

[15] healthcare. URL:https://medical-dictionary.thefreedictionary.com/health+care.

[16] healthcare. URL:https://dictionary.cambridge.org/dictionary/english/healthcare.

[17] Symptômes et diagnostic du diabète. URL:https://www.ameli.fr/assure/sante/themes/diagnostic/diagnostic-diabete#:~:text=Le%20diagnostic%20est%20posé%20lorsque,en%20l%27absence%20de%20symptômes..

[18] healthcare. URLhttps://marketbusinessnews.com/financial-glossary/health-care/.

[19] healthcare stages. https://pallipedia.org/tertiary-health-care/.

[20] What is machine learning? machine learning for beginners. URL:https://www.edureka.co/blog/what-is-machine-learning/.

[21] Opencv documentation ,operations on arrays „ . URL:https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#addweighted.

[22] Opencv documentation ,image filtering, . URL:https://docs.opencv.org/2.4/modules/ocl/doc/image_filtering.html?highlight=gaussianblur.

[23] Microsoft accelerates industry cloud strategy for healthcare with the acquisition of nuance. URL:https://news.microsoft.com/2021/04/12/microsoft-accelerates-industry-cloud-strategy-for-healthcare-with-the-acquisition-of-nuance/.

[24] Transfer learning. URL:https://openclassrooms.com/en/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5088816-apprenez-a-construire-un-cnn-et-gagnez-du-temps-avec-le-transfer-learning.

[25] Ashley Walker 2000 Robert Fisher, Simon Perkins and Erik Wolfart. Gaussian smoothing. https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm.

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

[27] Elaouaber Zineb Aziza, Lazouni Mohamed El Amine, Messadi Mohamed, and Bessaid Abdelhafid. Decision tree cart algorithm for diabetic retinopathy classification. In *2019 6th International Conference on Image and Signal Processing and their Applications (ISPA)*, pages 1–5. IEEE, 2019.

[28] Fiona M Burns, Anne M Johnson, James Nazroo, Jonathan Ainsworth, Jane Anderson, Ade Fakoya, Ibidun Fakoya, Andy Hughes, Eva Jungmann, S Tariq Sadiq, et al. Missed opportunities for earlier hiv diagnosis within primary and secondary healthcare settings in the uk. *Aids*, 22(1):115–122, 2008.

[29] Thomas Davenport and Ravi Kalakota. The potential for artificial intelligence in healthcare. *Future healthcare journal*, 6(2):94, 2019.

[30] Mehdi Torabian ESFAHANI, Mahsa GHADERI, and Raheleh KAFIYEH. Classification of diabetic and normal fundus images using new deep learning method. *Leonardo Electron J Pract Technol*, 17(32):233–248, 2018.

[31] Varun Gulshan, Lily Peng, Marc Coram, Martin C Stumpe, Derek Wu, Arunachalam Narayanaswamy, Subhashini Venugopalan, Kasumi Widner, Tom Madams, Jorge Cuadros, et al. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *Jama*, 316(22):2402–2410, 2016.

[32] ilovescience. Diabetic retinopathy (resized). URL https://www.kaggle.com/tanlikesmath/diabetic-retinopathy-resized.

[33] kaggle. Diabetic retinopathy detection. URL https://www.kaggle.com/c/diabetic-retinopathy-detection/overview.

[34] World Health Organization et al. *Primary health care: report of the International Conference on primary health care, Alma-Ata, USSR, 6-12 September 1978*. World Health Organization, 1978.

[35] Massimo Porta and F Bandello. Diabetic retinopathy. *Diabetologia*, 45(12):1617–1634, 2002.

[36] Mohamed Shaban, Zeliha Ogur, Ali Mahmoud, Andrew Switala, Ahmed Shalaby, Hadil Abu Khalifeh, Mohammed Ghazal, Luay Fraiwan, Guruprasad Giridharan, Harpal Sandhu, et al. A convolutional neural network for the screening and staging of diabetic retinopathy. *Plos one*, 15(6):e0233514, 2020.

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

[38] Peter Szolovits, Ramesh S Patil, and William B Schwartz. Artificial intelligence in medical diagnosis. *Annals of internal medicine*, 108(1):80–87, 1988.

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

[40] Benjamin Warner. Resized 2015 2019 blindness detection images. URL https://www.kaggle.com/benjaminwarner/resized-2015-2019-blindness-detection-images.

[41] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4): 611–629, 2018.