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

Ordre N° : IA StartUp 04/M2/2023

Thesis

Submitted in fulfilment of the requirements for the Masters degree in

Computer science

Option : **Artificial Intelligence**

Diabetic retinopathy detection based on retinography images

By :
BOUHITEM FAYEZ MOKHTAR

Members of the jury :

Belouaar Houcine	MCA	President
Aloui Ahmed	MCA	Supervisor
Bendahmane Asma	MAA	Examiner

Academic year 2022-2023

Acknowledgements

First of all, I thank God for giving me the courage and health to carry out this work.

I would like to express my deep gratitude to Dr. Aloui Ahmed, my thesis supervisor. I would like to thank him sincerely for his attentive supervision, regular follow-up, invaluable advice and unfailing support throughout my project. His presence and expertise were invaluable to my success.

Finally, I would like to express my gratitude and sincere thanks to all those who helped and encouraged us in any way in the realization of this project. I would also like to express my gratitude to all my friends for their unconditional support.

Abstract

The number of people with diabetes is expanding at an alarming rate throughout the world. According to the World Health Organization, there are currently 347 million cases of diabetes globally, and there will be 552 million cases by the year 2030. 1 in 3 of diabetes patients suffers from diabetic retinopathy, where DR is considered the fastest disease and the number one that leads to blindness because DR damages the retina's blood vessels and DR is time sensitive side that require early detection to stop it from progressing to irreversible vision loss. Deep learning is one of the new and powerful solutions to detect and classify various diseases that are diagnosed by medical imaging like X-ray, MRI, and Fusndus imaging.

The presented work it was with medical follow-up by an ophthalmologist surgeon in the eye hospital of Biskra. In our research, it is a deep learning methodology. We used the famous convolutional neural network AlexNet, a densely connected convolutional network DenseNet-121, and Yolov8. The two first models are used to classify the fundus images into five classes based on their severity: no DR, mild, moderate, severe, and porlifrate DR and the other model is used to detect the lesion of DR. We applied the three different models mentioned on different dataset with different structures. All used images are prepossessed with a Gaussian filter to improve the quality of the images and to emphasize the DR lesion like the blood vessels, microaneurysms and hemorrhages present in the retina. , also we used augmentation in the prepossessing phase and the circle crop method to give all images the circle retina form, and then we fed our data to our training models. Our results were very good and impressed the doctors at the hospital.

Keywords: Healthcare, Diabetes, Diabetic retinopathy, Artificial intelligence, Machine learning, Deep learning, CNN, DenseNet.

Résumé

Le nombre de personnes atteintes de diabète augmente à un rythme alarmant dans le monde entier. Selon l'Organisation mondiale de la santé, il y a actuellement 347 millions de cas de diabète dans le monde et il y en aura 552 millions d'ici 2030. 1 patient diabétique sur 3 souffre de rétinopathie diabétique, la RD étant considérée comme la maladie la plus rapide et la première à entraîner la cécité, car la RD endommage les vaisseaux sanguins de la rétine et la RD est un aspect sensible au temps qui nécessite une détection précoce pour l'empêcher d'évoluer vers une perte irréversible de la vision. L'apprentissage profond est l'une des solutions nouvelles et puissantes pour détecter et classer diverses maladies qui sont diagnostiquées par l'imagerie médicale comme les rayons X, l'IRM et l'imagerie Fusndus.

Le travail présenté a fait l'objet d'un suivi médical par un chirurgien ophtalmologiste à l'hôpital ophtalmologique de Biskra. Dans notre recherche, il s'agit d'une méthodologie d'apprentissage profond. Nous avons utilisé le célèbre réseau de neurones convolutifs AlexNet, un réseau convolutif densément connecté DenseNet-121, et Yolov8. Les deux premiers modèles sont utilisés pour classer les images du fond d'œil en cinq classes en fonction de leur gravité : pas de DR, légère, modérée, sévère et porlifrate DR, et l'autre modèle est utilisé pour détecter les lésions de DR. Nous avons appliqué les trois modèles mentionnés sur différents ensembles de données avec différentes structures. Toutes les images utilisées sont prétraitées avec un filtre gaussien pour améliorer la qualité des images et mettre en évidence les lésions de la RD comme les vaisseaux sanguins, les microanévrismes et les hémorragies présents dans la rétine. Nous avons également utilisé l'augmentation dans la phase de prépositionnement et la méthode du cercle de culture pour donner à toutes les images la forme d'un cercle rétinien, puis nous avons transmis nos données à nos modèles d'entraînement. Nos résultats ont été très bons et ont impressionné les médecins de l'hôpital.

Contents

List of Figures

List of Tables

General introduction

Overview

Deep learning is not new it existed from 90s but in that time it was rarely used because of the weakens of computers and hardware that time. but now with development of computers and with the availability of enhanced computational resources and capabilities has opened up opportunities for developing DL models that accurately detect and classify various pathologies and this is why DL become used in various applications self driving ,chat bots and healthcare.

In recent years with scary growing of diabetic patients causes increasing in the number of diabetic retinopathy patient because the DR is a complication associated with diabetes. DR leads to retinal abnormalities where in severe causes cause partial or even the total blindness .So to avoid blindness and the retina eye damage we need to detect DR in early stages and this is time consuming and labor-intensive. this present challenge in the poor or populated countries that suffer from lack of ophthalmologist like India and other countries in Asia.

Problematic

The world is witnessing a scary increasing in the number of patient with diabetes. According to the World Health Organization, there are currently 347 million cases of diabetes globally, and there will be 552 million cases by the year 2030. 1 in 3 of diabetes patients suffers from diabetic retinopathy, where DR is considered the fastest disease and the number one that leads to blindness because DR is a time sensitive side that require early detection to stop if from progressing to irreversible vision loss. It also needs a specialist and experienced doctor to diagnose it and more than that it's expensive.

Objective of the work

This work aims to present web application use deep learning model for the detection and classification of the various diabetic retinopathy levels using retinal pictures. After this general introduction, the dissertation is structured as follows:

- **Chapter 1: AI and healthcare** the use of artificial intelligence in healthcare then provide an overview of machine learning and deep learning finally we will see together popular CNN models
- **Chapter 2: Diabetic retinopathy and related works** This chapter begins by providing an overview of diabetes including its effects on the body then describes diabetic retinopathy and discusses its various stages and progression. Finally, provides some recent research and related works in the field
- **Chapter 3: System design** This chapter provides an overview of the system design, the datasets utilized, their structures, the preprocessing phase, and the architectures of our proposed deep learning models
- **Chapter 4: implementation 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.

Chapter 1

AI and healthcare

1.1 Introduction

When we talk about healthcare comes to our mind's diagnosis, treatment and injuries and when we mention AI comes to our mind's Intelligent robots, automation like self-driving cars like tesla but may we forget the important role that it plays in healthcare. in recent years AI showed an excellent result by improving the accuracy and efficiency of diagnoses by reducing medical errors and providing more personalized treatment plans because AI can analyze vast volumes of medical data and helps healthcare professionals to make wise decisions about patient care by using a variety of methodologies and technologies including machine learning and deep learning.

In this chapter will examine the use of artificial intelligence in healthcare then provide an overview of machine learning and deep learning finally we will see together popular CNN models .

1.2 Healthcare

After conducting an extensive search to ascertain the definition of healthcare, I found many definitions and this is some definition according to different sites.

1.2.1 Definitions

According to medical dictionary health care is "*services provided to people or communities by agents of the health services or professions for the purpose of promoting, maintaining, monitor-*

ing, or restoring health” [5]

According to oxford dictionary healthcare is “*the service of providing medical care*” [8]

According to oxford dictionary healthcare is “*the activity or business of providing medical services*” [6]

According to Merriam webster dictionary healthcare “*is efforts made to maintain or restore physical, mental, or emotional well-being especially by trained and licensed professionals usually hyphenated when used attributively*” [7]

1.3 AI in Healthcare

Artificial intelligence (AI) is transforming the healthcare industry by altering how care is provided, enhancing patient outcomes and cutting costs by enhance the accuracy of medical diagnoses, enable personalized treatments, and provide more efficient and effective healthcare services.

we can describe AI as technology that enables robots to see patterns, make decisions, and learn from feedback and from data like analyzing vast volumes of patient data and spot patterns and trends and develop predictive models that can be applied to enhance patient care.

The main benefits of ai in healthcare:

- Improved diagnosis accuracy
- Enhanced patient outcomes
- Reduced healthcare costs:
- Personalized medicine
- Improved patient experience
- More efficient healthcare services

After going over just a few of the numerous advantages of AI in healthcare, the AI technology continues to advance, it will likely play a bigger part in the healthcare like patient outcomes and assisting healthcare providers in providing better treatment at lower costs. And these is some of application of Ai in healthcare:

1. **AI in medical diagnosis**

   AI algorithms can analyze radiology and radioscopy like X-rays, MRIs, and CT scans to detect for changes or anomalies that can be signs of sickness or injury. This can help doctors make more accurate diagnoses. [4]

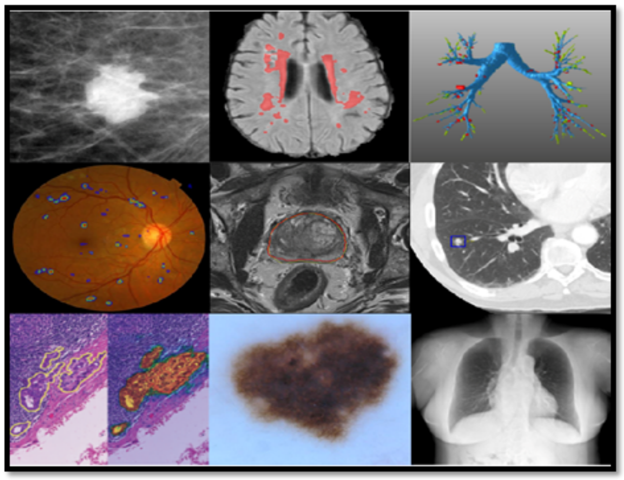
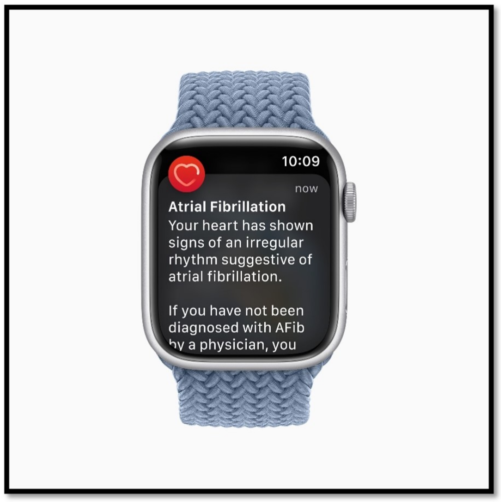


Figure 1.1: Medical imaging analysis

   AI achieved excellent accuracy in diagnosing various diseases which is based on its diagnosis in radiology and radioscopy like cancer detection, neurology, chronic disease management.

2. **AI in Early Detection**

   AI can also be used for early detection in different domains such as Alzheimer's and Parkinson's disease, heart attacks. By analyzing patient data such as medical history and brain scans, AI algorithms can detect patterns and identify early signs of these diseases in figure 1.2, leading to earlier interventions and improved outcomes for patients.


10:09

now

Atrial Fibrillation
Your heart has shown signs of an irregular rhythm suggestive of atrial fibrillation.

If you have not been diagnosed with AFib by a physician, you

Figure 1.2: Apple watch message [26]

3. **AI in medical assistance**

   AI has shown tremendous potential in providing medical assistance to healthcare providers, reducing workload, improving accuracy, and providing better patient outcomes. Medical assistance through AI includes a wide range of applications, such as virtual health assistants, chatbots, and smart medical devices. Virtual nurses, often referred to as virtual health assistants, are AI-powered technologies that offer individualized care to patients, including keeping track of prescription schedules and responding to inquiries from patients. These assistants can monitor patients remotely, alert healthcare providers to potential issues, and help prevent hospital readmissions.

Figure 1.3: virtual nurse [33]

1.4 Machine learning

1.4.1 Introduction

As we all know that Machine learning is a subset of artificial intelligence that involves training computer algorithms to learn from data, without being explicitly programmed. This allows computers to make predictions or take actions based on patterns in data, which can be used in a variety of applications across industries, including healthcare.

In fact, Machine learning algorithms can be trained on large datasets of patient data, medical images, or electronic health records to identify patterns and predict outcomes. This can be used to improve the accuracy of diagnosis, personalize treatment plans, and predict patient outcomes. For example, machine learning algorithms have been used to identify patients at high risk for heart disease or to predict hospital readmissions [40].

The relationship between artificial intelligence (AI), machine learning (ML), and deep learning (DL) is depicted in figure 1.4

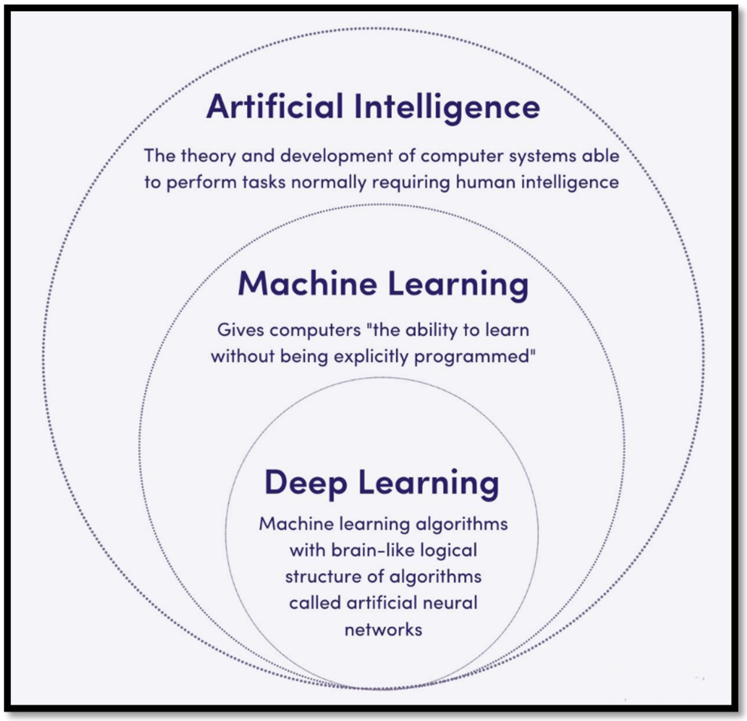

Artificial Intelligence
The theory and development of computer systems able to perform tasks normally requiring human intelligence
Machine Learning
Gives computers "the ability to learn without being explicitly programmed"
Deep Learning
Machine learning algorithms with brain-like logical structure of algorithms called artificial neural networks

Figure 1.4: AI vs ML vs DL [42]

1.4.2 Types of Machine learning algorithms

There are three main types of machine learning:

1. **Supervised Learning**

   A labeled dataset with known inputs and expected outputs is used to train the algorithm in this type of learning. The algorithm gains the ability to extrapolate and predict outcomes from, unseen data. Regression and classification are both examples of supervised learning [40].

   For example classification of dogs and cats in figure 1.5 the model learn from labeled data , after training the model now can predict outputs as cat or dog from new data or test data (unseen data)

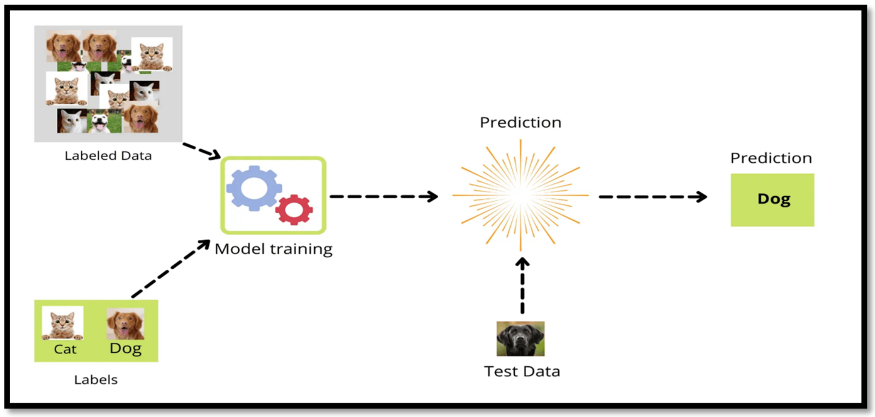


Figure 1.5: supervised classification cats and dogs

2. **Unsupervised Learning**

   In this kind of learning, the algorithm is trained on a dataset that has no labels and inputs without corresponding outputs. The algorithm picks up on the structure and patterns in the data over time. Clustering and dimensionality reduction are two examples of unsupervised learning [40].

   In figure 1.6 we have the same example but in unsupervised learning, after training the model was able to recognize differences between the two classes and create classes according to the similar images

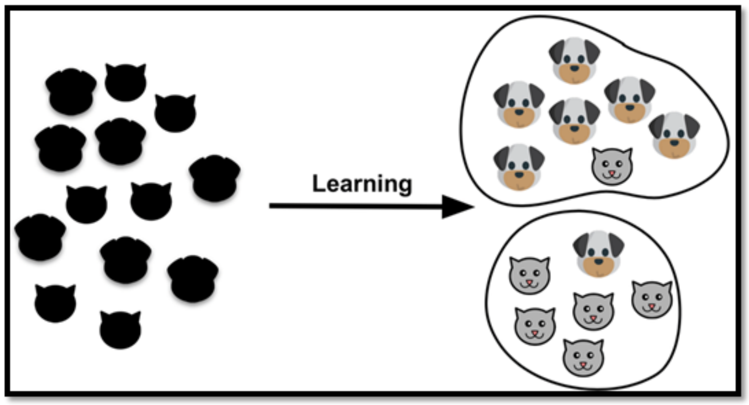


Figure 1.6: unsupervised learning cats and dogs [1]

3. **Reinforcement Learning**

   In this kind of learning, the algorithm picks up new skills through interaction with the environment and feedback in the form of rewards or penalties. Over time, the algorithm develops the ability to make decisions that maximize the reward and minimizing penalty. Playing video games and using robots are two examples of reinforcement learning [40].

   In figure 1.7 the dog is playing role of the agent and the environment is the persons that he trains the dog so he will give him a reward if the dog act correctly and punishment if the dog didn't behave well.

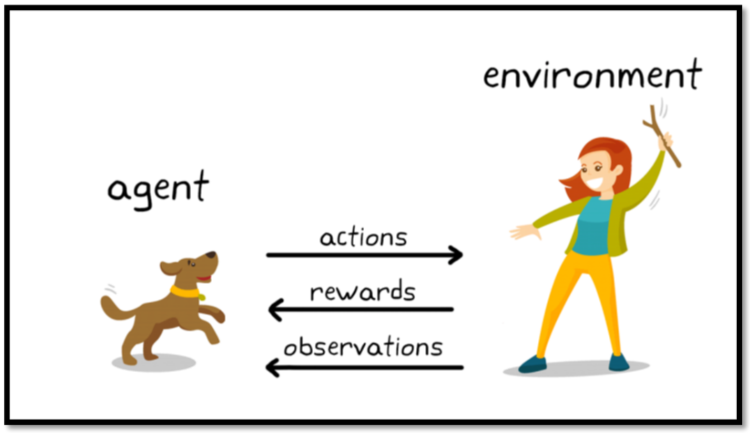


Figure 1.7: Reinforcement Learning example

1.5 Deep learning

Deep learning is a type of machine learning that deals with training neurons network. This is the most powerful technique in classification that justifies its existence in all applications and disciplines that using machine learning.

DL is neural network architecture with many hidden layers the D is abbreviation of deep means depth of layers, more than 2 hidden layers [32], DL process and analyze complex data, such as images, sounds, and text. The layers in these neural networks allow them to learn increasingly abstract features of the data, enabling them to make more accurate predictions or classifications

1.5.1 Difference between DL and ML

Key differences between deep learning and machine learning algorithms (see figure 1.8 is that deep learning algorithms focus on automatic and machine learning of features. Learning that features must be extracted manually.

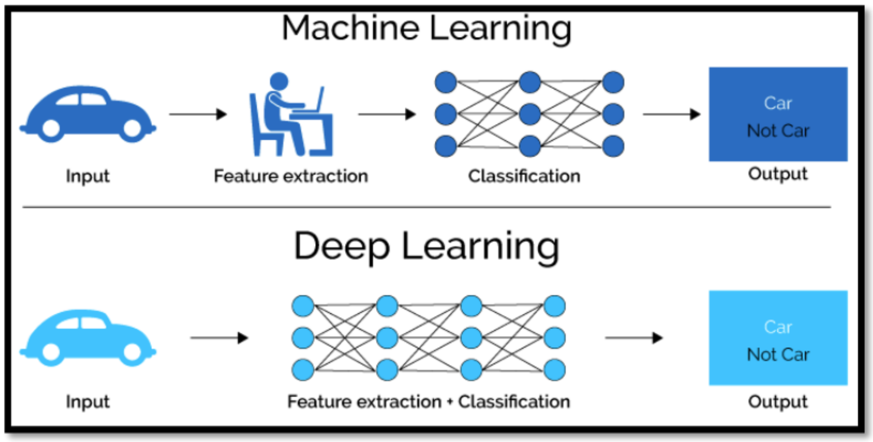


Figure 1.8: DL vs ML [47]

1.5.2 Artificial neural networks

In fact the artificial neural network inspired from neurons of human brain so we need to know some basics of humans neurons to understand the artificial neural network of machines.

1.5.2.1 Biological neuron

Is a cell that forms basic structural and functional unit of nervous system of humans and it's the cell that responsible for receiving, processing and transmitting information through signals .its composed of cell body (soma),dendrites, axon and axon terminals [23]. look at figure 1.9.

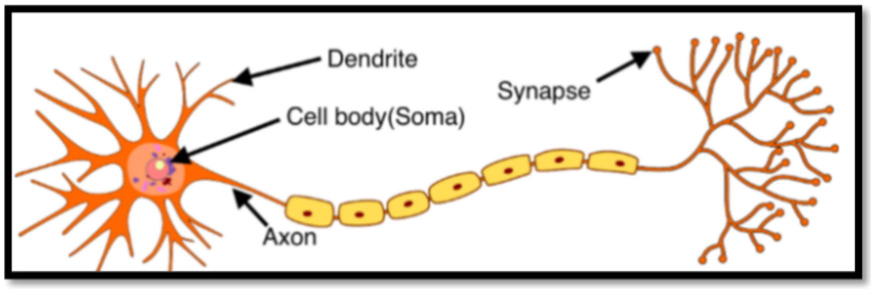


Figure 1.9: Biological neuron [37]

1.5.2.2 Artificial neuron

Like we said in the beginning of this part the artificial neuron is modeled from biological neuron, also known as a computational neuron, is a mathematical function that models the behavior of biological neurons, it takes a number of variables (x) as input then applies a weight to each input ***, and sums them up to produce an output. This output is then passed through an activation function, which determines the final output of the neuron [40].

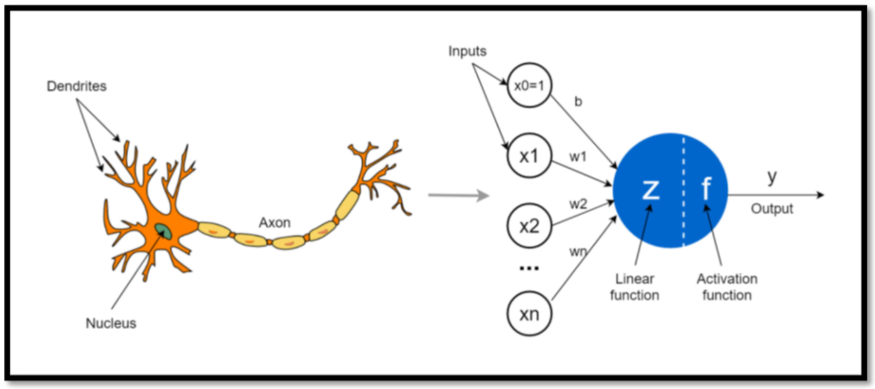


Figure 1.10: Biological neuron to an artificial neuron [37]

1.5.2.3 Artificial neural network

Artificial neural network is a computational model that consists of interconnectedus nodes, or artificial neurons, organized into layers, an input layer , a hidden layer, and an output layer, The output of one layer becomes the input to the next layer, allowing the network to process and transform information in a hierarchical manner see Figure 1.11.

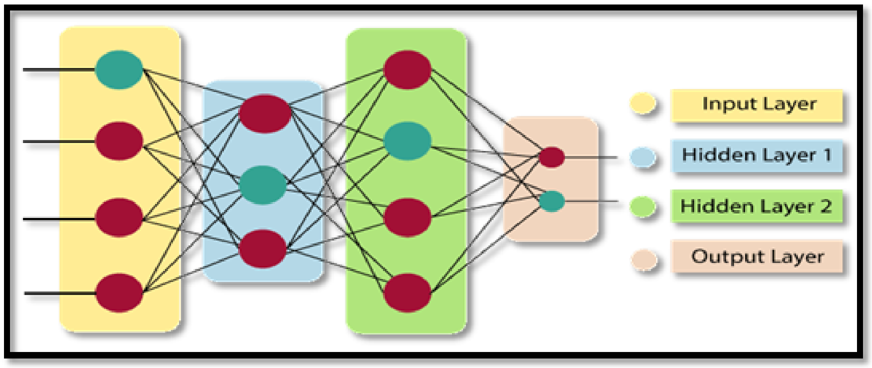


Figure 1.11: Artificial Neural Network [30]

1.6 Convolutional Neural Networks

1.6.1 Definition

A Convolutional neural networks is a sub of neural network, They possess all the traits of neural networks. The key feature of CNNs is their ability to automatically learn and extract features from images without any manual intervention, the first CNN architecture focused on text recognition [27].

A basic CNN architecture contains one convolutional layer and pooling layer, and also sometimes contains fully connected layers for supervised prediction. So, let us discuss the layers of CNN in detail as shown in Figure 1.12

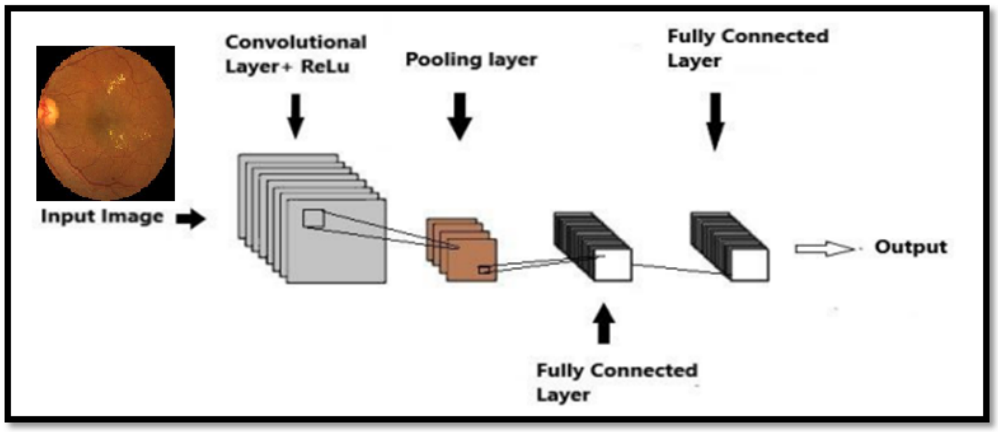


Figure 1.12: Convolutional neural network composition

1.6.2 Convolutional neural network composition and main operations

1.6.2.1 Input layer

The image's data should be in the CNN's input layer. The image data is a three-dimensional matrix that needs to be converted into a single column before being input. Each layer's output will be fed into the next layer.

1.6.2.2 Convolutional layer

Convolutional layer is the one where the action starts, is the most important layer in a CNN, it is responsible for extracting features from the input image, A number of filters or kernels make up the convolutional layer. which are small matrices that slide over the input image and perform a dot product operation at each position. Each filter develops the ability to recognize a particular feature or pattern, such as edges, corners, or textures, in the input image [43].

Here is an example of filter or 3x3 see figure 1.13

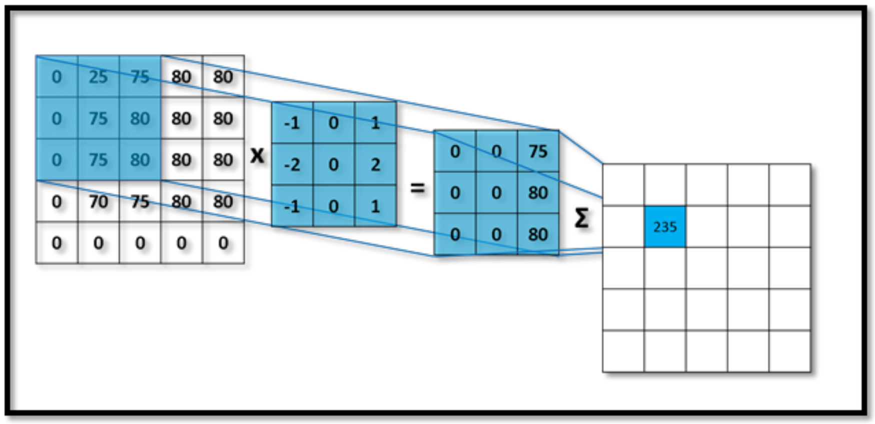


Figure 1.13: layer filter 3x3 [9]

We get 235 from applicating filter in image matrix like that:

0 * -1 + 25 * 0 + 75 * 1 + 0 * -2 + 75 * 0 + 80 * 2 + 0 * -1 +75 * 0 + 1 * 80 =235

Here is some Additional parameters, such as padding, stride, and dilation, can be adjusted for the convolutional layer.

- **Padding** To keep the input image's size constant during convolution, padding adds zeros to the edges of the image. See Figure 1.14

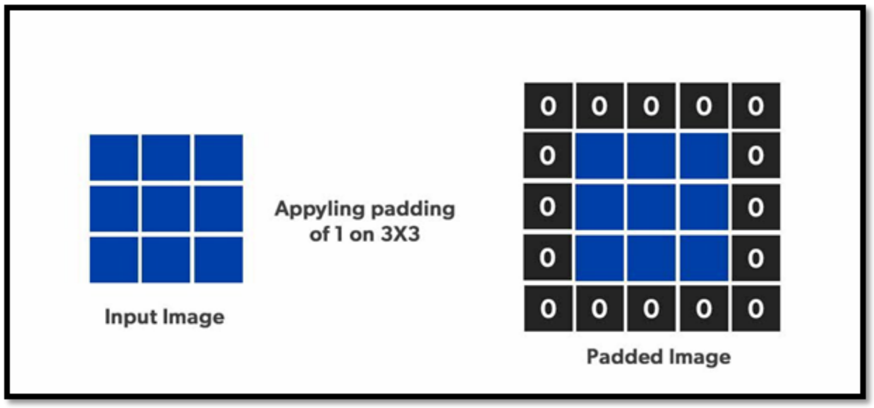


Figure 1.14: Padding [19]

- **Stride** Stride determines the spacing between the positions at which the filter is applied to the input image, by default equal 1 [2].

  Here is formula to calculate the output dimension after using stride *******

  Here is an example of using stride=2 see Figure 1.15

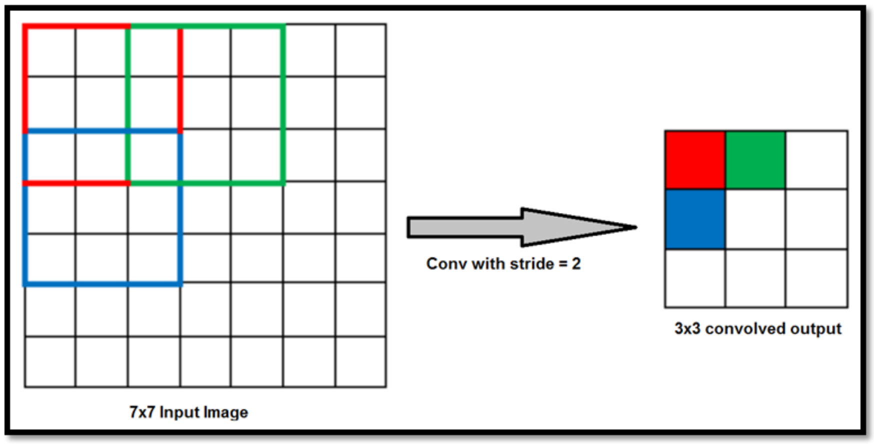


Figure 1.15: Stride [21]

1.6.2.3 Pooling layer

The pooling layer is a type of layer in a CNN that performs down-sampling of the feature maps produced by the convolutional layer , the goal of using down sampling is to reduce the spatial size of the feature maps while keeping the most crucial details about the features. The most common type of pooling used in CNNs is max pooling [43].

The method of max pooling involves dividing the feature map into non-overlapping rectangular regions called pooling regions and then determining the maximum value inside each pooling region, this process reduces the size of the feature map by a factor of the pooling region size.

For example, if the pooling region size is 2x2 then the feature map's size will be reduced in half in both directions. See Figure 1.16

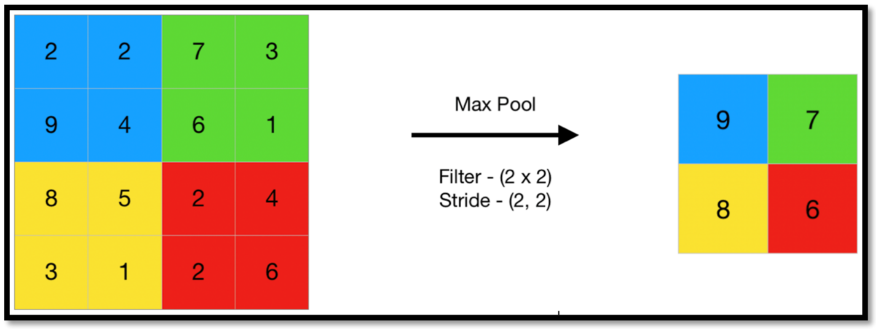


Figure 1.16: Max pooling [18]

1.6.2.4 The activation layer

The activation layer applies a non-linear activation function to the output of the previous layer, this nonlinear function's objective is to add nonlinearity to the network. which enables the network to learn more complex and non-linear relationships between the input and output [29].

Here is some of activation function see Figure 1.17

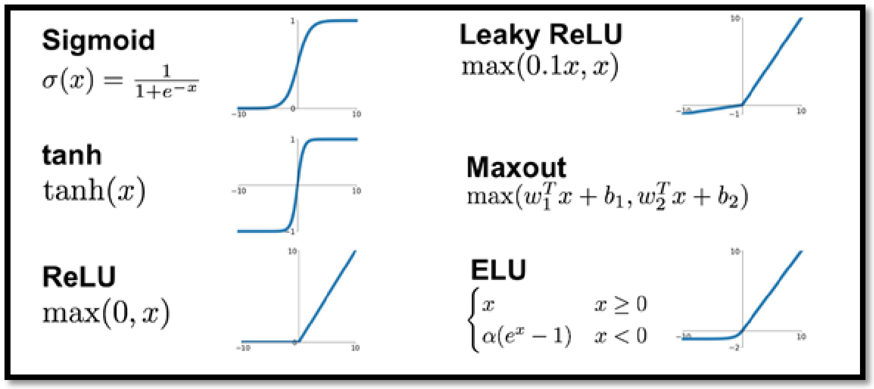


Figure 1.17: Activation function [16]

1.6.2.5 The fully connected layer

Every neuron in the previous layer is connected to every neuron in the current layer (see Figure 1.18) This layer is similar to the fully connected layers in traditional neural networks. The purpose of the fully connected layer in a CNN is to combine the features learned by the convolutional and pooling layers into a global representation that can be used for classification or other tasks [51].

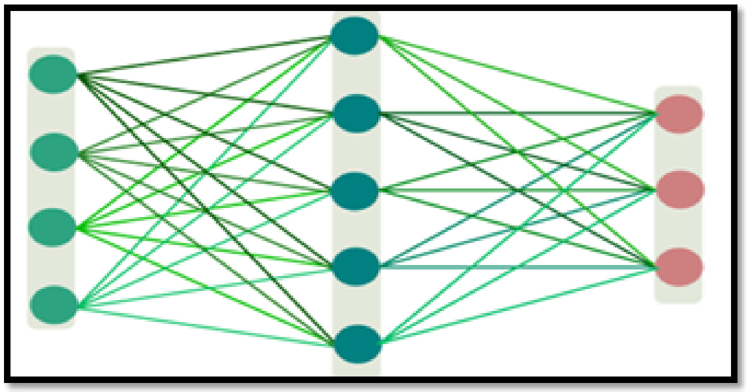


Figure 1.18: fully connected layer

1.6.2.6 Dropout

the dropout layer typically comes after the convolutional or fully connected layer. This technique used to prevent overfitting by randomly dropping out during training some of the neurons in the layer.

Example when dropout=0.2 that means each of neuron has a probability 0.2 of being "dropped out" at each iteration see Figure 1.19.

Drop out has many benefits

- Regularization: to prevent overfitting.
- improved generalization: By learning more robust features, the network is better able to generalize to new, unseen data
- Speed up training: the network converges faster during training.

- Ensemble learning: By randomly dropping out neurons at each iteration, the Dropout layer effectively creates an ensemble of smaller sub-networks, which can improve the overall performance of the network [53].

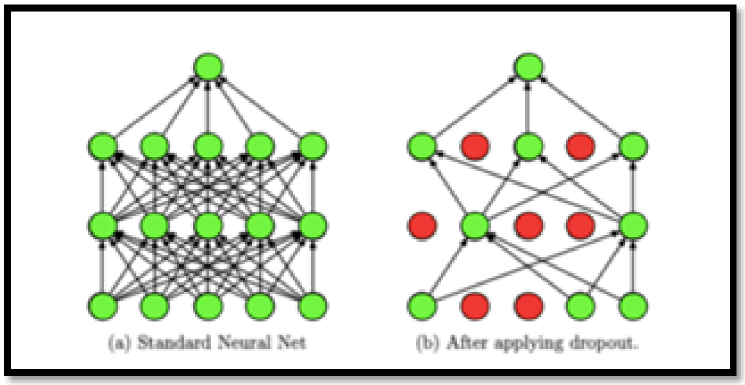


Figure 1.19: Applying dropout [14]

1.6.2.7 Output layer

The output layer of a Convolutional Neural Network (CNN) is the final layer of the network that produces the final predictions. The type and structure of the output layer depend on the specific task that the CNN is designed to solve.

Example if you have problem of classification the output layer is usually a fully connected layer with a SoftMax activation function [43].

1.7 Popular CNN Architectures

1.7.1 ResNet-50

ResNet-50 is a deep Convolutional Neural Network (CNN) architecture that was proposed by Microsoft researchers in 2015. ResNet50 is a deep neural network architecture consisting of 50 layers, that's why he has the number 50 in his name.

Resnet has ability to train very deep neural networks with many layers while avoiding the problem of vanishing gradients.

ResNet50 has shown state-of-the-art performance on various benchmark datasets, such as ImageNet, CIFAR-10, and COCO. It is widely used as a backbone architecture for many computer vision tasks, such as object detection, semantic segmentation, and image captioning [24].The Figure 1.20 explain Resnet50 architecture.

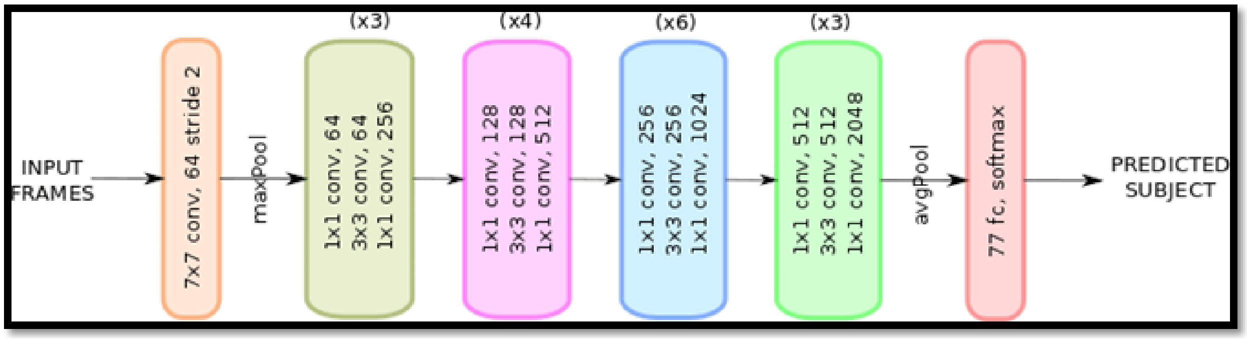


Figure 1.20: Resnet50 [20]

1.7.2 EfficientNet

The EfficientNet family consists of seven different models: EfficientNet-B0 to EfficientNet-B7.EfficientNet is a family of convolutional neural network (CNN) architectures introduced in 2019 by Google. Its architecture uses a number of methods ,in contrast to other architectures, this one is able to achieve good accuracy (look at Figure 1.21) with fewer parameters and computational resources by utilizing a novel method for scaling neural networks that balances network depth, width, and resolution [54].

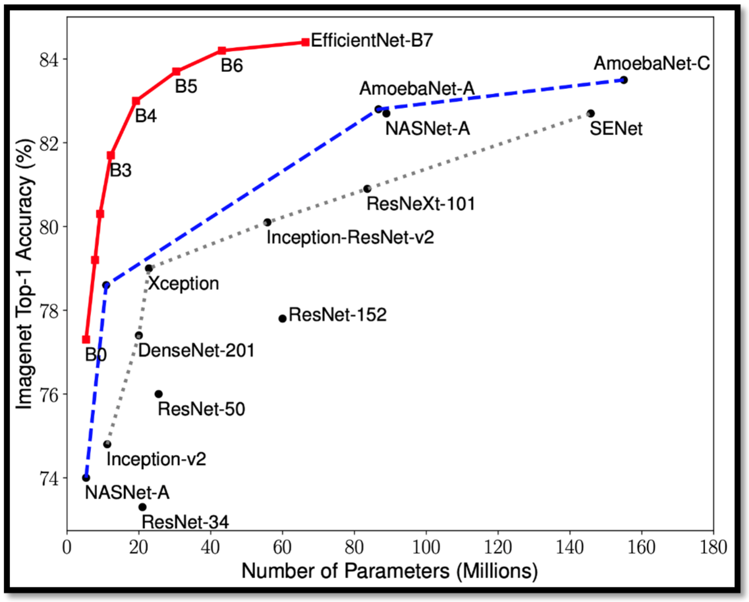


Figure 1.21: Efficientnet [15]

1.7.3 DenseNet

DenseNet is a deep convolutional neural network (CNN) architecture released in 2017 by Facebook AI. Each of the dense blocks that make up the DenseNet architecture has many convolutional layers, batch normalization, and a dense connection method. A transition layer is placed after each dense block and is composed of a convolutional layer, batch normalization, and a pooling process and each layer receives direct feedback from all preceding layers [39] look at Figure 1.22.

When compared DenseNet to other CNN architectures, it shows good performance on benchmark datasets like ImageNet, CIFAR-10, and CIFAR-100 while using fewer parameters and processing resources.

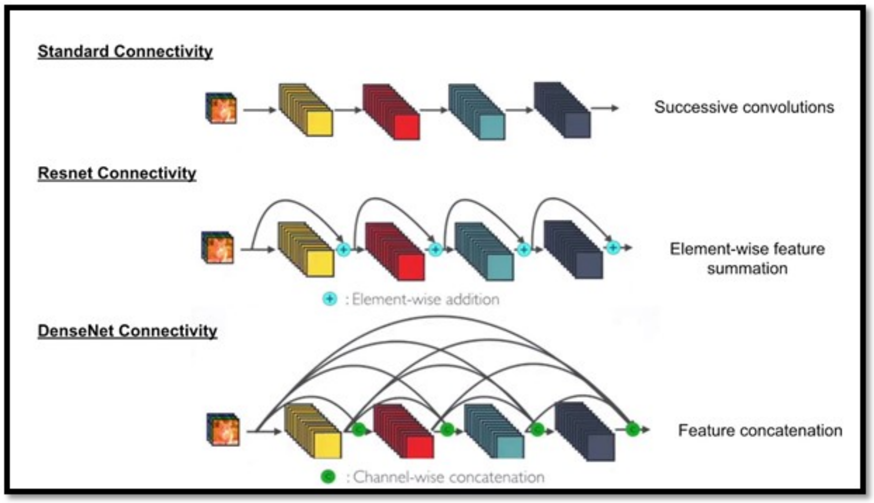


Figure 1.22: DenseNet [11]

1.8 Transfer learning

Instead of training a new model from scratch we use Transfer learning technique to gain time and to produce accurate model by taking an existing model has been trained on a large dataset then by training it on the new dataset [52].See Figure 1.23 explain deference between usual approach and Transfer learning approach.

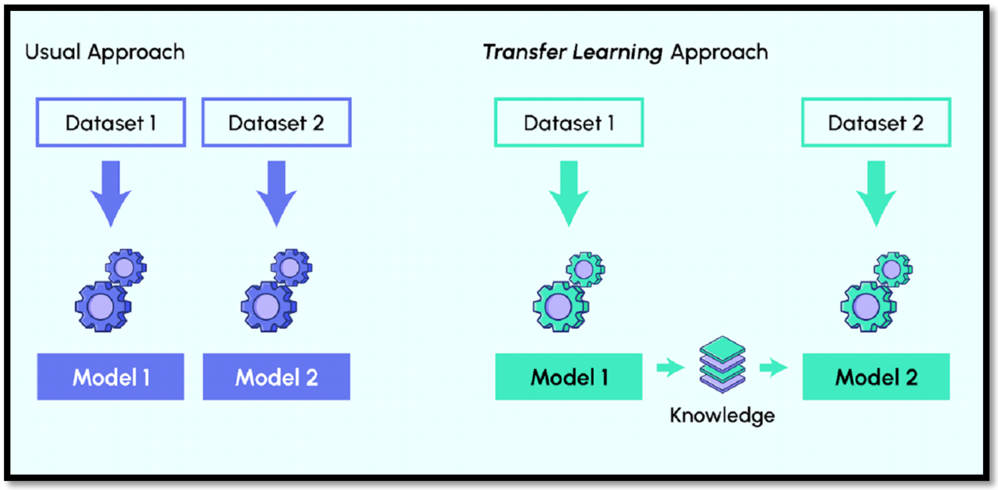


Figure 1.23: usual Approach VS Transfer learning Approach [44]

1.9 Conclusion

In this chapter we have covered some main aspects which are the basic of our current work: healthcare, Artificial intelligence. We have described the different use cases of AI in healthcare, we detailed the basics of machine learning and deep learning and popular CNN models. The next chapter will talk about diabets and we will introduce our main topic diabetic retinopathy and its stages and some recently related works.

Chapter 2

Diabetic retinopathy and related works

2.1 Introduction

Diabetes is a global epidemic has significant complications, one of which is diabetic retinopathy. This progressive eye disease damages the retina due to high blood glucose levels. Diabetic retinopathy causes visual impairment, reduces quality of life, and imposes a substantial socioeconomic burden. Researchers have made notable efforts to understand and manage this condition.

This chapter begins by providing an overview of diabetes including its effects on the body then describes diabetic retinopathy and discusses its various stages and progression. Finally, provides some recent research and related works in the field.

2.2 Diabetes

2.2.1 what is diabetes

Diabetes, also known as diabetes mellitus, is a chronic metabolic disorder that affects how our bodies processes blood sugar (glucose).as we all know that Glucose is the main source of energy for our bodies ,pancreas produce hormone known as Insulin which helps in adjusting the level of sugar in the blood. The high level of glucose in blood can damage various organs like the eyes heart, kidneys, nerves and blood vessels.[3].See Figure **??** .

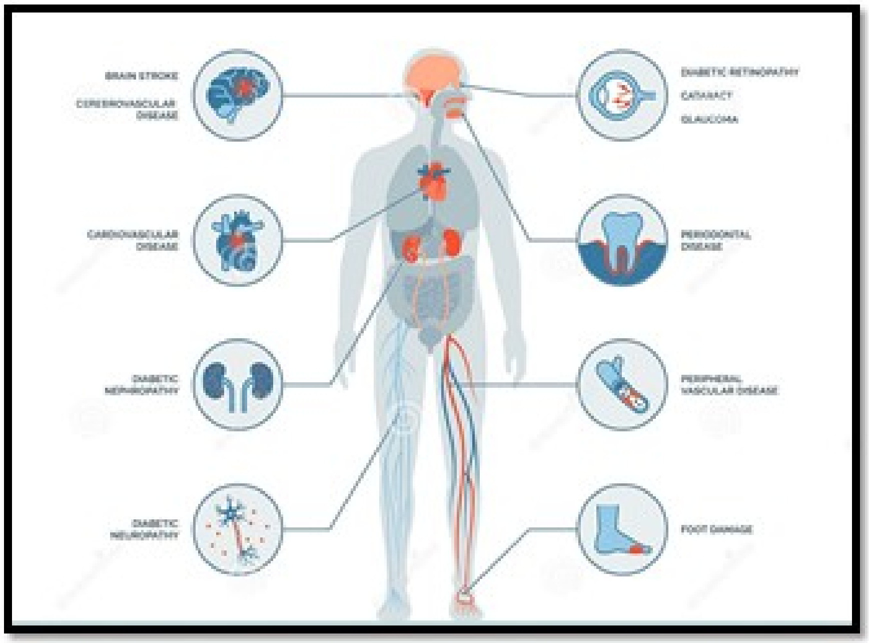


Figure 2.1: Effects of diabetes on human body [12]

2.2.2 Types of diabetes

1. **Type 1 Diabetes**

   Autoimmune condition, usually diagnosed in childhood or adolescence, where the body doesn't produce insulin [34].

2. **Type 2 Diabetes**

   Most common form, often linked to lifestyle factors like obesity and physical inactivity. The body becomes resistant to insulin or doesn't produce enough [34].

3. **Gestational Diabetes**

   Occurs during pregnancy and usually resolves after childbirth, but increases the risk of developing type 2 diabetes later [34].

   See Figure 2.2 for more details

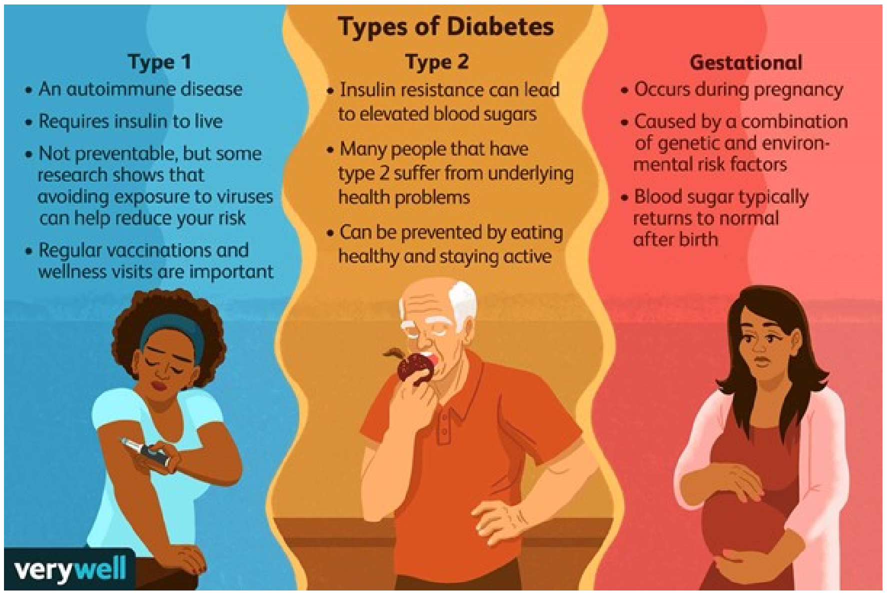

Types of Diabetes
Type 1
• An autoimmune disease
• Requires insulin to live
• Not preventable, but some research shows that avoiding exposure to viruses can help reduce your risk
• Regular vaccinations and wellness visits are important
Type 2
• Insulin resistance can lead to elevated blood sugars
• Many people that have type 2 suffer from underlying health problems
• Can be prevented by eating healthy and staying active
Gestational
• Occurs during pregnancy
• Caused by a combination of genetic and environmental risk factors
• Blood sugar typically returns to normal after birth
verywell

Figure 2.2: Types of diabetes [13]

2.3 Diabetic retinopathy

Diabetic retinopathy is a complication of diabetes that affects the eyes.DR can damage the small blood vessels of the retina leading to swelling, hemorrhage, and the growth of new abnormal blood vessels (see Figure 2.3)as result cause vision loss and blindness [45].

Here some of diabetic retinopathy symptoms:

- Blurred or distorted vision
- Dark or empty spots in your vision
- Eye pain or pressure
- Difficulty seeing at night

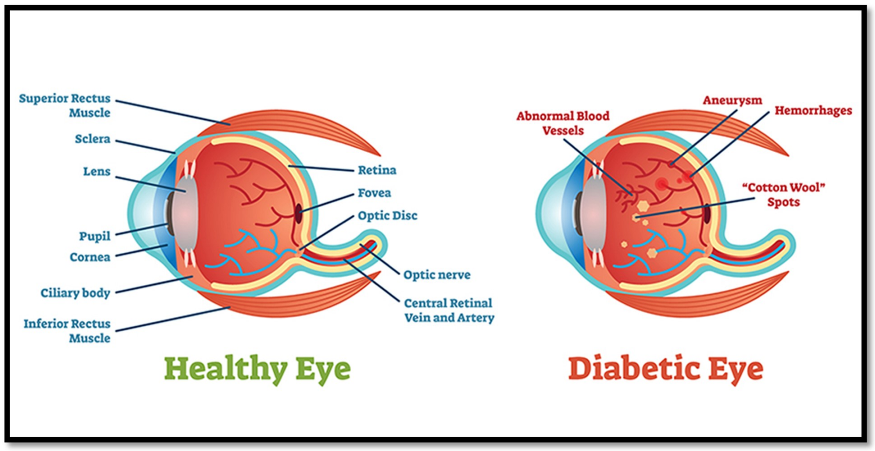


Figure 2.3: The difference between a normal retina and DR retina [10]

2.4 Retinal imaging

Retinal imaging is a diagnostic test used by Ophthalmologist to detect various eye diseases like diabetic retinopathy, macular degeneration, and glaucoma. They use imaging equipment to capture detailed images of the retina.

There is various types of retinal imaging techniques to diagnose retina like Optical coherence tomography, Indocyanine green angiography, Fundus photography (see Figure 2.4) [49]. for diabetic retinopathy Ophthalmologist use Fundus photography to diagnose patients

Figure 2.4: Fundus photography machine [17]

2.5 Diabetic retinopathy stages

Diabetic retinopathy can be classified into 4 stages based on blood vessels damage in the retina.

2.5.0.1 Mild

Is the first stage of DR , may not cause any noticeable symptoms and visual acuity may be normal but the doctor can notice some of small areas of the blood vessels called Microaneurysms [61].

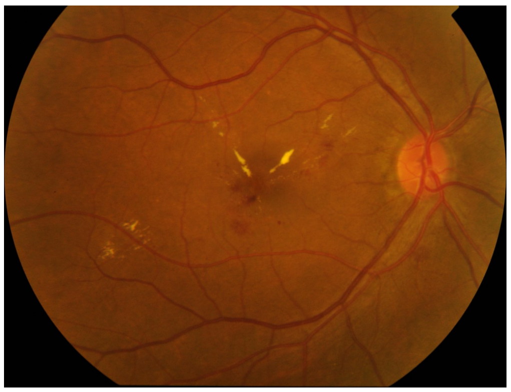


Figure 2.5: Mild DR [28]

2.5.0.2 Moderate

the damage to the blood vessels in the retina is more than the mild stage, because the blood vessels may become blocked and we will notice not just Microaneurysms we will see More widespread areas of hemorrhage, Cotton wool spots, IRMA [61].

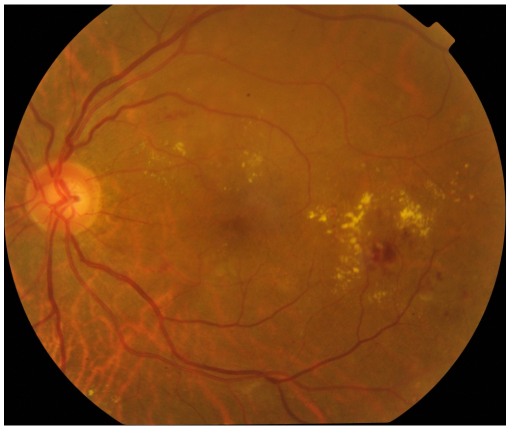


Figure 2.6: Moderate DR [28]

2.5.0.3 Severe

In this stage the patient will have more significant vision changes, such as blurred or distorted vision, floaters, and difficulty seeing at night because the blood vessels are more damaged than moderate stage and because of that we will notice more and big area of hemorrhage, multiple areas IRMA ex [61].

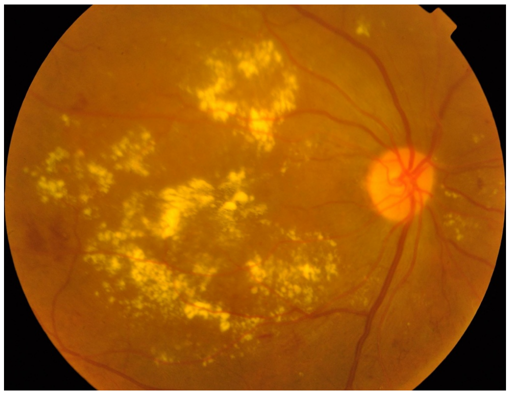


Figure 2.7: Severe DR [28]

2.5.0.4 Proliferate DR

This is advanced level of DR ,the eye will create new blood vessels and they are very fragile which may cause fluid leakage and the patient will have more problems in vision more than severe and even blindness [61].

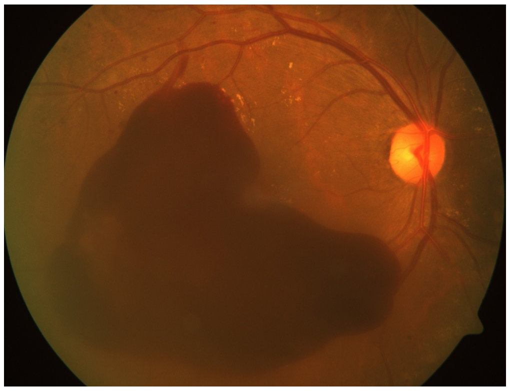


Figure 2.8: Proliferate DR [28]

2.6 Related works

The following piece we will mention some of previous works on classification retinal images using various ML and DL algorithms.

- **A Deep Learning Approach for the Diabetic Retinopathy Detection**

  M.R. Sebti and all [50], proposed two models, the first model was binary classification (no DR, DR) and the second model was multi class classification (No DR, Moderate, Severe). for the preprocessing they have used gaussian filter. The deep learning approach that he used for multi class classification its CNN architecture with total of 8 layers. From layer 1 to layer 5 are convolutional layers followed by a max pooling layer and from layer 6 to layer 8 are fully connected layers. His first model trained on aptos19 dataset, number images was 3662.the training accuracy and training validation were found to be 96.93% and 95.08%, respectively. And for the second proposed model the results were 93.06% for training accuracy, 81.12% for training validation.

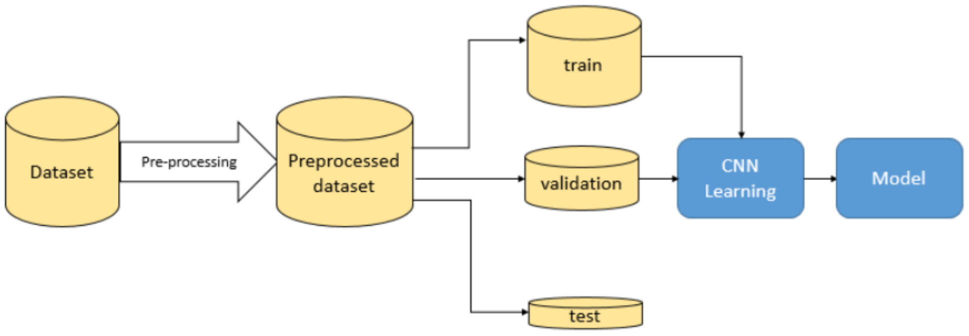


Figure 2.9: Diagram of Sebti System [50]

- **Deep learning approach for early diabetic retinopathy diagnosis**

  Falah and all [35], proposed two models, the first model was binary classification (no DR, DR) and the second model was multi class classification (No DR, Mild, Moderate, Severe). for the preprocessing they have used gaussian filter for all classes and Histogram equalization just in 2 classes mild and moderate. The deep learning approach that she used for multi class classification its CNN architecture with total of 11 layers. From layer 1 to layer 8 are convolutional layers followed by a max pooling layer and from layer 9 to layer 11 are fully connected layers. Her first model trained on aptos19 dataset, number images was 3662.the training accuracy and training validation were found to be 98.75% and 95.35%, respectively. And for the second proposed model the results were 96.73% for training accuracy, 93.45% for training validation.

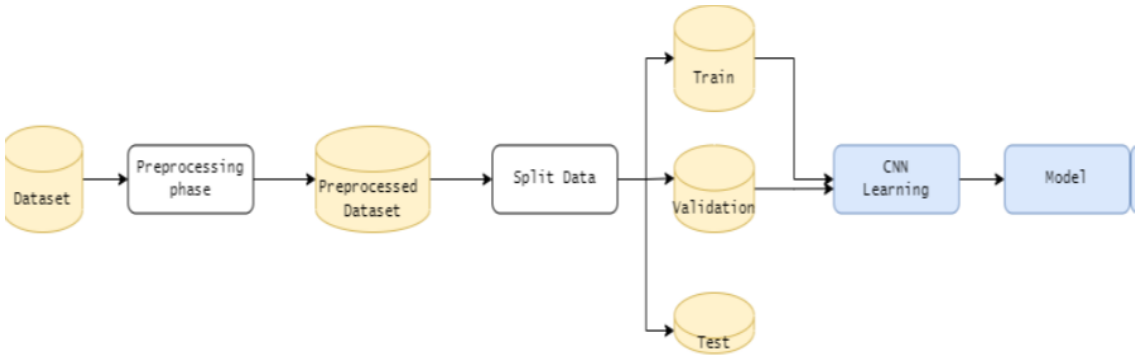


Figure 2.10: General architecture of Falah system [35]

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

  Esfahan and all [58], proposed just one model, he used popular CNN architecture Resnet34 for binary classification to classify DR as normal and DR (see Figure 2.11).

  Resnet34 is a pretrained model on ImageNet database. For the preprocessing they have used gaussian filter, weighted addition and image normalization. The size of the image, which had the number 35000, was (512,512) pixels. They reported 85% accuracy and 86% sensitivity.

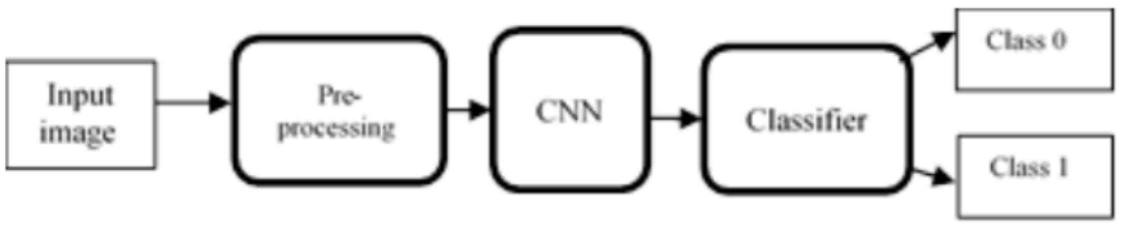


Figure 2.11: : Block diagram of the proposed method [58]

The table below 2.1 provides a concise comparison of relevant studies and research efforts, allowing us to identify research gaps and build upon previous findings. This analysis will inform the unique contributions of our research within the broader context of the field.

Model	**Accuracy**	**Validation accuracy**	**Test Accuracy**	**Number of classes**
Sebti riad binary[50]	96.93%	95.08%	95.65	2
Fallah kawther binary[35]	98.75%	95.35%	96.02%	2
Esfahan [58]	85%	86%	-	2
Sebti riad [50]	93.06%	81.12%	80,48%	3
Fallah kawther [35]	96.73%	93.45%	93,43%	4

Table 2.1: Comparison of relative work

‘

2.7 The critique of previous works

1. **Work of Fallah and all [35]**

   After consulting Dr.Hadrouk an surgeon specialist in ophthalmology in Biskra eye hospital. He said that the proposed achitecteur of Fallah Kawther and all [35] its wrong.Where we can not classify diabetic retinopathy like that. where we cannot put class severe with class proliferate DR (see Figure 2.12).

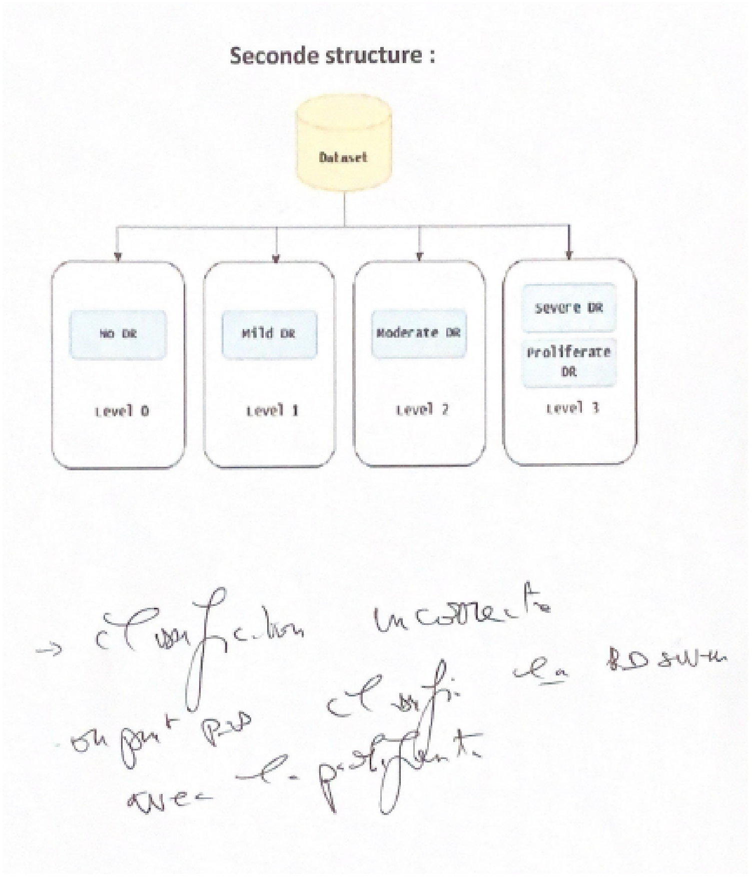


Figure 2.12: Fallah proposed structure to classify DR

2. **Work of Sebti and all [50]**

   the doctor also confirmed that the proposed achitecteur of Sebti and all [50] its wrong.Where we can not put class severe with class proliferate DR (see Figure 2.13).

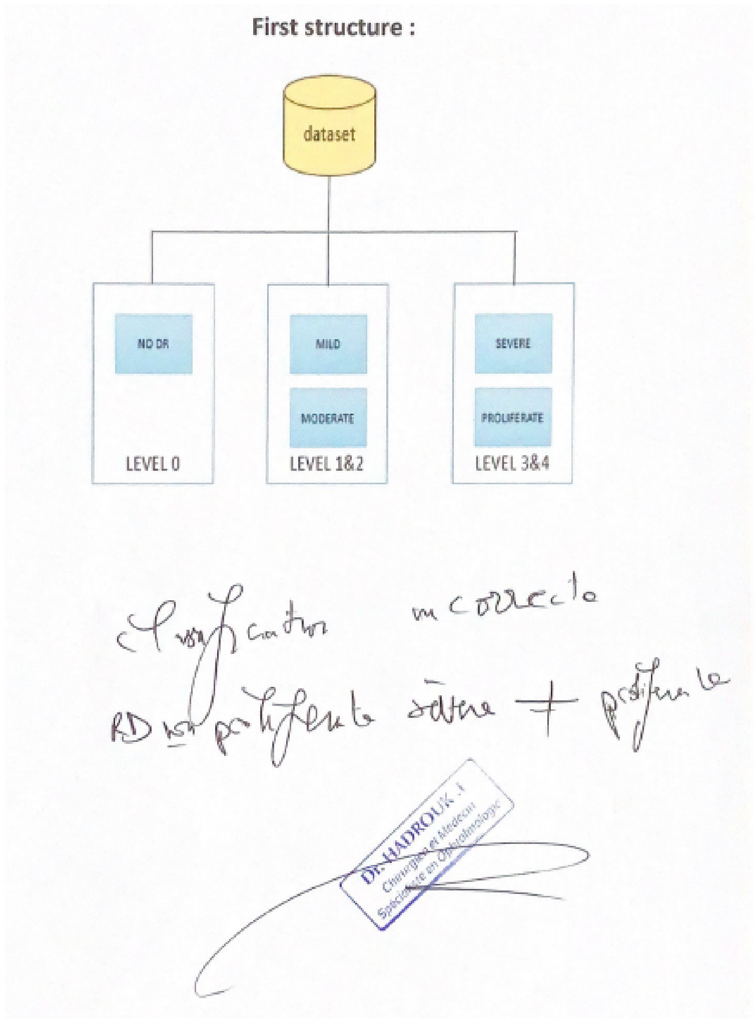


Figure 2.13: Sebti proposed structure to classify DR

2.7.1 Important note

all our proposed structure certified from doctor Hadrouk an surgeon specialist in ophthalmology in Biskra eye hospital. we will see them in the next chapter

2.8 Conclusion

In this chapter we covered diabetes ,diabetic retinopathy and some of previous related works, the preprocessing of data and type of deep learning technique that they used and the results achieved. Overall, we found that the related works that we covered are highly relevant to our own project then we explain the mistakes they made in their own proposed structure to classify DR. The previous ideas and findings presented have helped to shape our understanding of the topic and have provided valuable insights into the challenges we are likely to face. In the next following chapter we will go into our own strategy and how it fits into the field's larger context.

Chapter 3

System design of a deep learning architecture for Diabetic Retinopathy detection

3.1 Introduction

Recently, AI showed huge development in many applications in the real world until it became comparable to humans in intelligence, like Tesla's self-driving car.on the other side Diabetic retinopathy detection has received more attention recently. This is because there are more and more people suffering from this disease. A rapid and correct diagnosis becomes crucial as the risk of diabetic retinopathy grows. Artificial intelligence has emerged as a potential solution to address this demand. By leveraging the power of AI, we can effectively diagnose patients with diabetic retinopathy, providing timely and reliable results.
Our project focuses on developing three deep learning models that can accurately detect and classify different levels of diabetic retinopathy. We employ sophisticated image processing techniques within our model to accomplish this task. This chapter provides an overview of the system design, the datasets utilized, their structures, the preprocessing phase, and the architectures of our proposed deep learning models.

3.2 System design

Before we start in explain our work we were working and consulting with **DR.Hadrouk an surgeon specialist in ophthalmology in Biskra eye hospital**

3.2.1 General design

The goal of our research is to create a AI doctor to diagnose patient with diabetes if they have diabetic retinopathy.First we started by creating our model by choosing dataset then do prepossessing on it and split it then we training our model and testing it we will see more details in the next section .after we create model we deploy it in cloud to use it with web application that we are created to connect the users of our application with our intelligent model.when users use our web application the user send an image of retina imaging then our model can classify and detect retinopathy if its exist.See figure 3.1 to get general idea of our work.

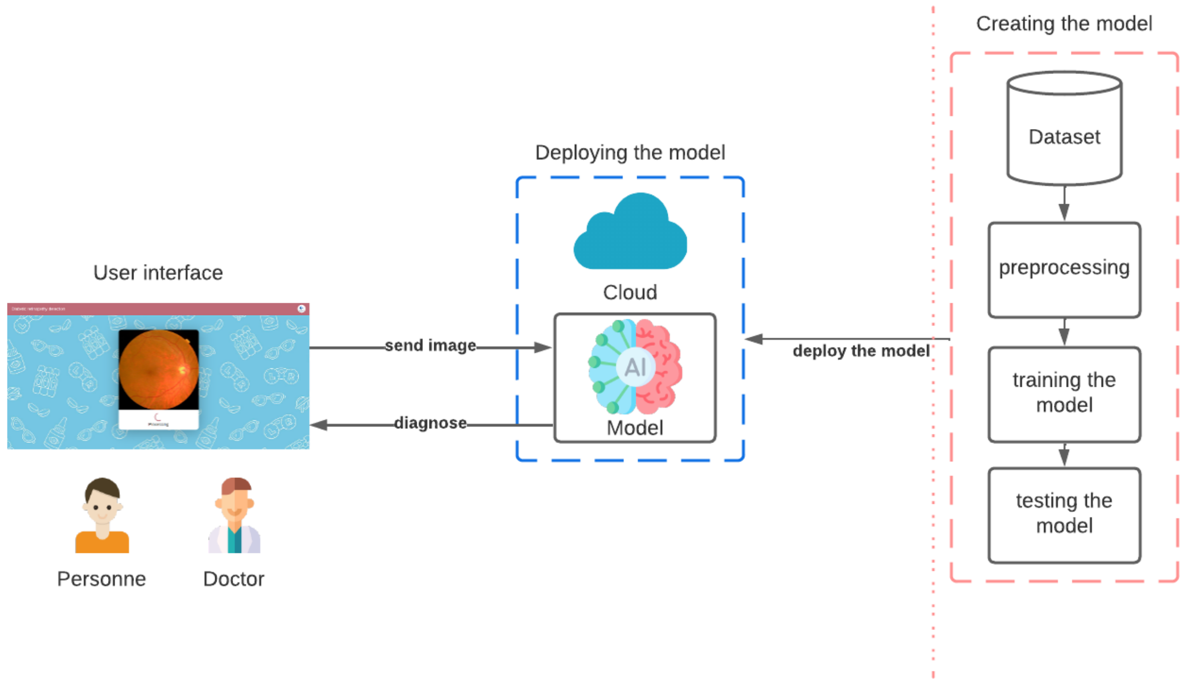


Figure 3.1: General design of our System

3.2.2 Detailed 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 3.2.

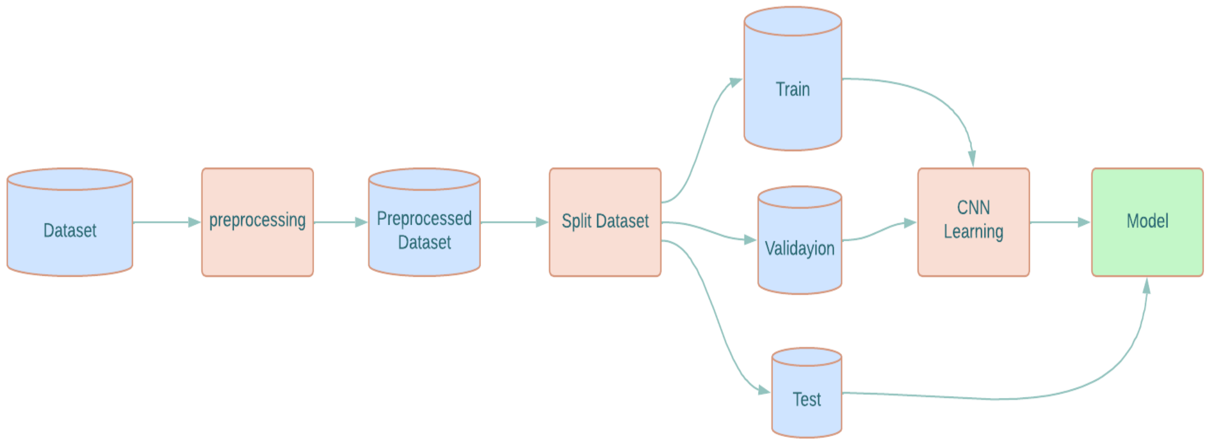


Figure 3.2: Detailed design of our System

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.

3.2.3 Dataset

After consult DR Hadrouk this is our dataset that we will work with

3.2.3.1 Datasets for classification

1. **Dataset description:**

   We utilized two Kaggle datasets that can be found at Resized 2015 and 2019 Blindness Detection Image [60].

(a) The first original (2019) Dataset is available at APTOS 2019 Blindness Detection [28]. This dataset was used in [50] [35] [58].

(b) The second original (2015) dataset is available at Diabetic Retinopathy (resized) [60] and was used in [50] [35].

(c) The third dataset created by overflow with team of ophthalmologist and me in biskra.

2. **Datasets structure:**

   (a) **Original structure of the available datasests:** The images are ranked into five different levels of diabetic retinopathy by specialists as it is shown in Figure 3.3

      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.

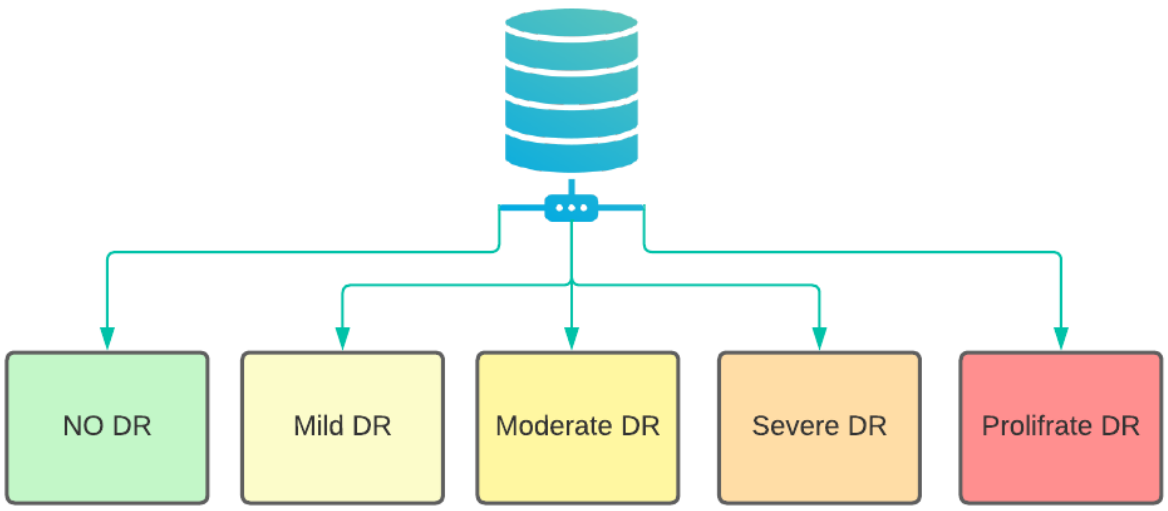


Figure 3.3: Original structure

(b) **The first proposed structure:** we change the original data set's organizational structure just to 2 classes NO DR and DR By combining all diabetic retinopathy levels (mild, moderate, sever, and proliferate) to reflect the DR class and keep class NO DR like its (see Figure 3.4).

i. **NO DR**: represent healthy retina people that not infected by DR

ii. **YES DR**: represent infected retina people with DR.

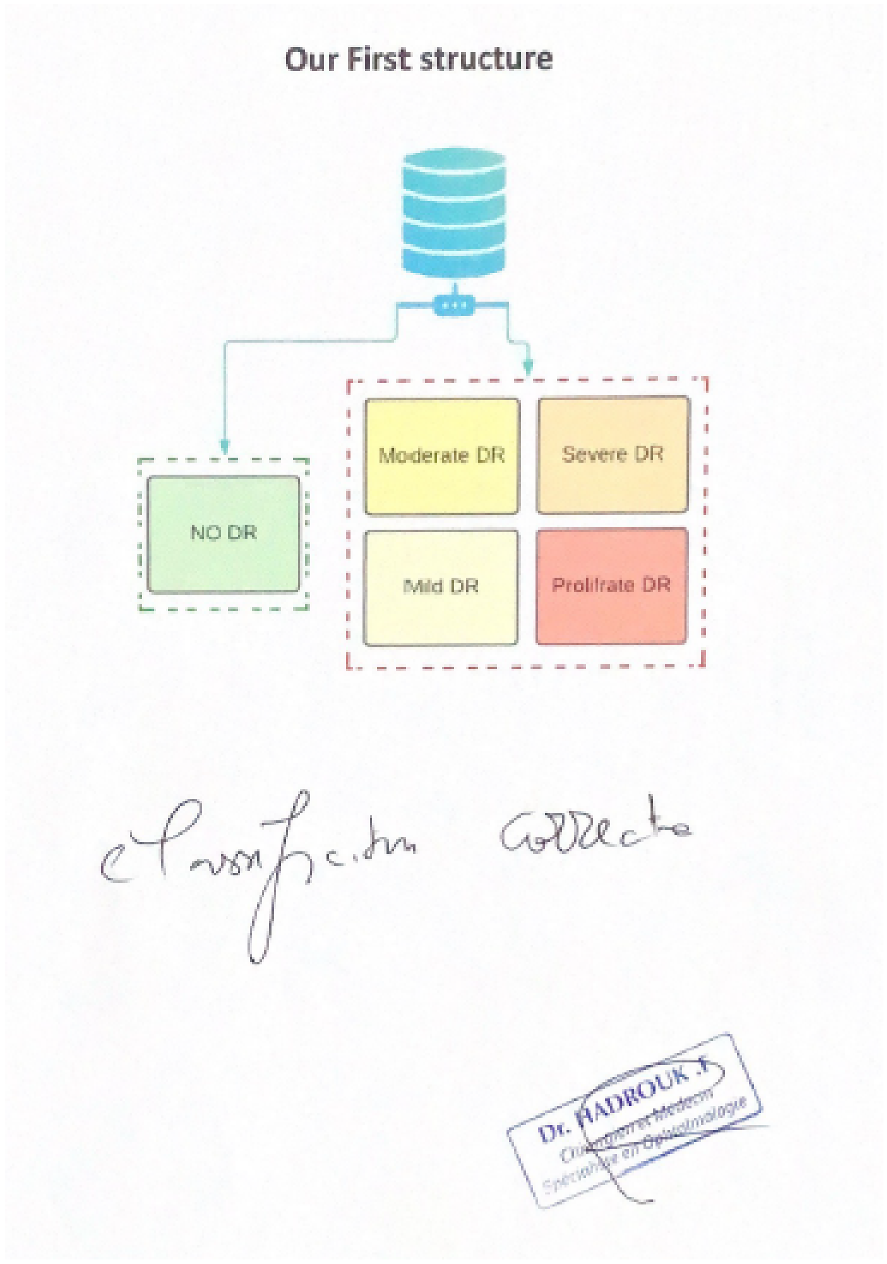


Figure 3.4: Our First structure

(c) **Our second structure:** We used the original structure of original dataset 5 classes .See Figure 3.5

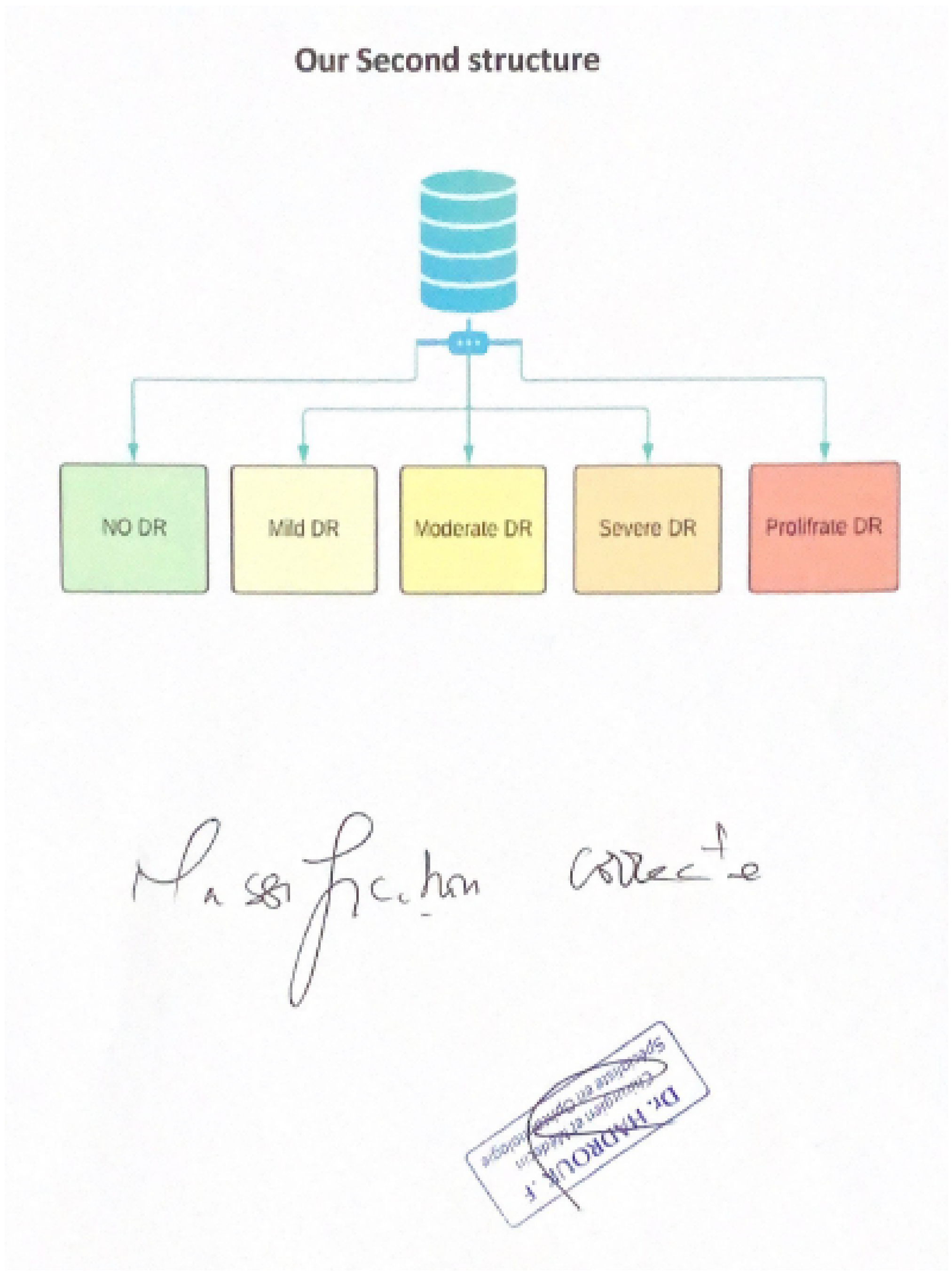


Figure 3.5: Our Second structure

3.2.3.2 creating our dataset for detection

1. **Data Collection**

   The dataset for diabetic retinopathy detection was collected from the APTOS 2019 dataset [28], which is a commonly used dataset for this task. The dataset consists of retinal images that were captured from patients with varying degrees of diabetic retinopathy.

2. **Data Labeling**

   The images in the dataset were labeled using an annotation tool roboflow see figure 3.6 . As the labeling was done by help of an ophthalmologist, it ensures accurate and reliable annotations for the presence of diabetic retinopathy. Each image was annotated with bounding boxes to indicate the regions associated with diabetic retinopathy.

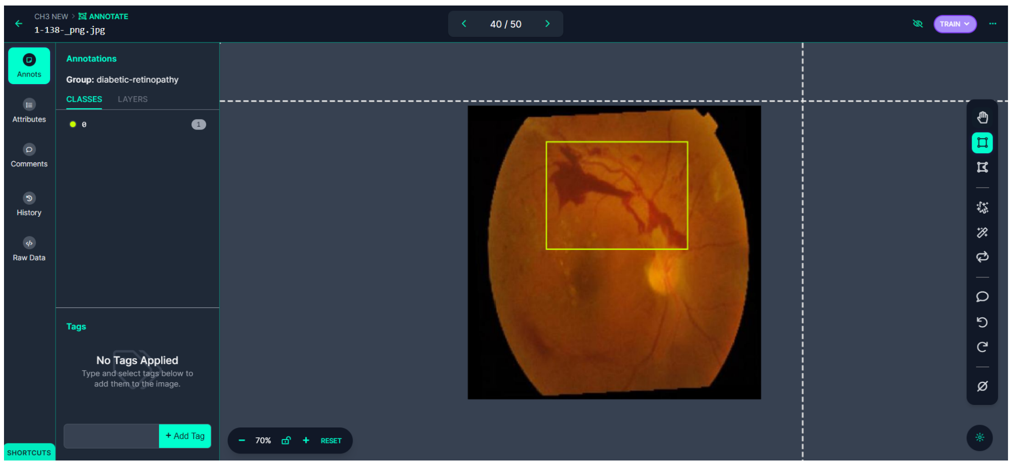


Figure 3.6: labeling images with roboflow

3. **Augmentation**

   To increase the diversity and variability of the dataset, data augmentation techniques were applied. These techniques included random rotations, flips (both vertical and horizontal), and adjustments to brightness. Augmentation helps in preventing overfitting and improves the model's ability to generalize to unseen data.

4. **Train/Test Split**

   The dataset was divided into three subsets: 80% for training, 10% for validation, and 10% for testing.

5. **Data Conversion**

   The annotations for the dataset were converted into YOLOv8 format, which uses a .txt file for each image. Each .txt file contains the bounding box annotations of region of DR in a specific image.

3.2.4 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**: A Gaussian filter is a commonly used image filtering technique that is effective in reducing noise and smoothing images [**?**].

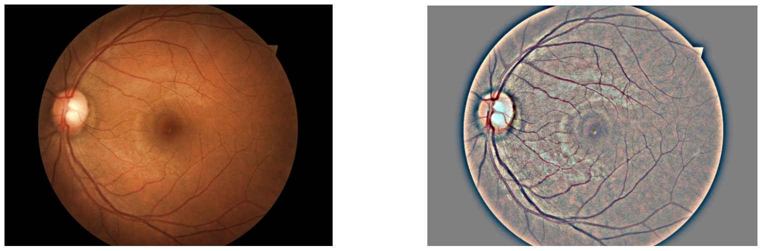


Figure 3.7: Gaussian filtering on diabetic retinopathy image

Why gaussian filter ?

The Gaussian filter is very helpful in the context of diabetic retinopathy and retinal imaging is used to improve the quality of the pictures and increase the precision of classification because it can help to emphasize the diabetic retinopathy lesions in retina image like the blood vessels, microaneurysms or Hemorrhages present in the retina.

2. **Resize**:

   The original dataset images were (1024,1024,3) We resize images to (224x224x3). The first number indicate the width of image and the second indicate Hight and the last number indicate image channels RGB (Red, Green, Blue).

   Why resizing images ?

   resizing images improve memory and computational efficiency, ensures consistent input size, standardizes the dataset and contributing to effective and efficient deep learning model training and inference.

3. **Circle crop**:

   I noticed that the images are not in same look. Some of them rounded and some of them zoomed ex...
   So, I create a method 'circle crop' to make them look the same and to give them the circle retina look

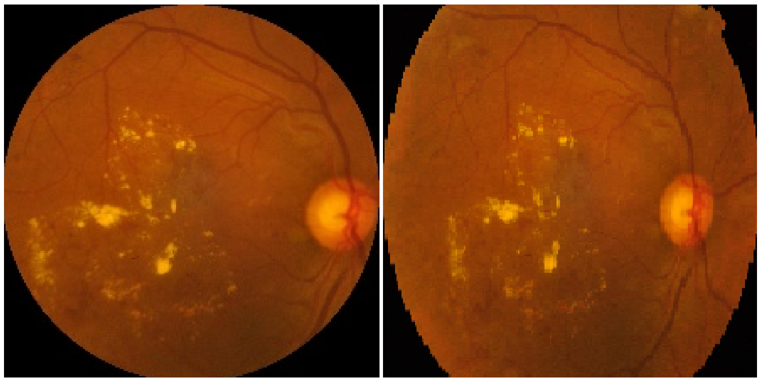


Figure 3.8: Cyrcle Crop on diabetic retinopathy image

3.2.5 Splitting dataset

We divide our dataset to three subsets (train, validation and test),70% of total images to train,20% for validation and 10% for test

3.2.6 Data Augmentation

Image augmentations are modifications applied to images to increase the number of images by applying different transformations for example rotating, scaling, flipping, brightness, cropping or adding some noise, to create new samples that are similar to the original images. Data augmentation also helpful in improves model generalization, reduces overfitting, addresses class imbalance, and helps in scenarios with limited data availability.
In this work we increase the number of images by different transformations like rotation, brightness by using ImageDataGenerator by TensorFlow.

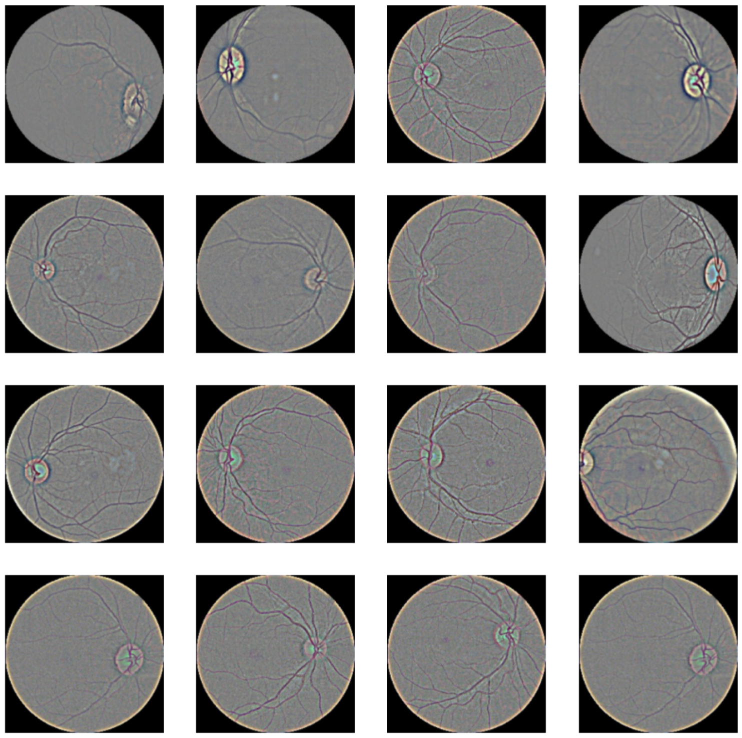


Figure 3.9: Data Augmentation using ImageDataGenerator

3.2.7 CNN Learning

Like we mention in chapter 1 CNNs have ability to automatically learn and extract features from images without any manual intervention. We mention also the composition of the Convolutional neural network (see Figure 1.12) like convolutional layer, pooling layer ,fully connected layer , dropout layer and some activation functions and in the same chapter we mention some of popular CNN architecture like DenseNet ,EfficientNet and ResNet50 .Now we will train this models for classification of diabetic retinopathy then we will move to detection to discuss yolov8.

3.2.8 AlexNet

Alexnet is the name of a convolutional neural network (CNN) that made a major contribution to the area of computer vision. It was developed by Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton and it won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 [25].
so after this excellent performance of AlexNet we choose it for classification of diabetic retinopathy

3.2.8.1 AlexNet architecture

in general AlexNet like we said is a convolutional neural network (CNN) so it have convolutional layers, pooling layers,fully connected layers.see figure 3.10 describes architecure of AlexNet.

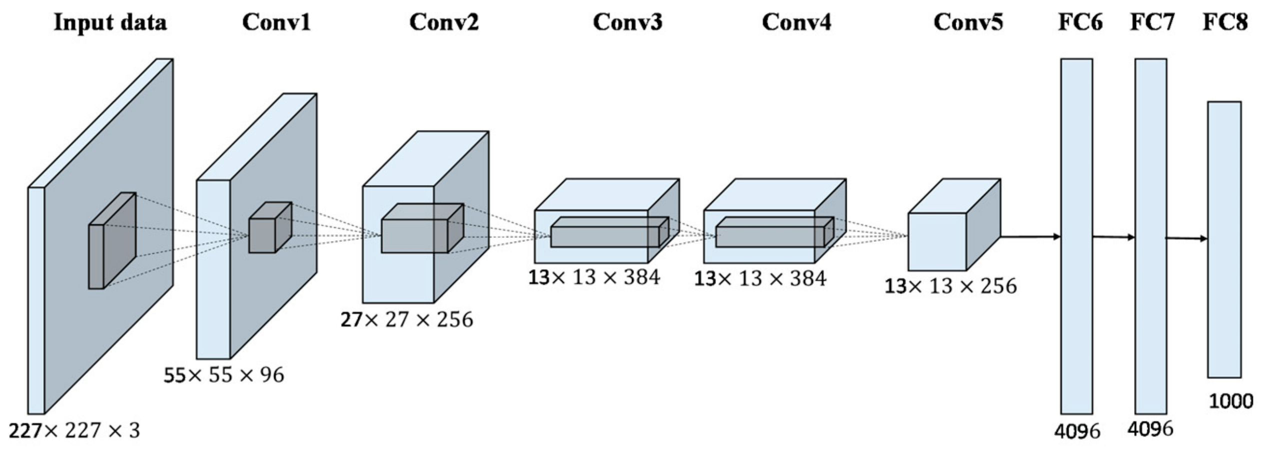


Figure 3.10: AlexNet architecture

1. **Input Layer**

   AlexNet takes an input image of size 227x227x3 (RGB color images). The three channels represent the red, green, and blue color channels.

2. **Convolutional Layers**

   The first convolutional layer has 96 filters with a stride of 4 pixels and a size of 11x11. Local response normalization (LRN) across channels is then used after rectified linear unit (ReLU) activation. By using max-pooling with a 3x3 window and a stride of 2 pixels, the output volume is downsampled.

3. **Second Convolutional Layer**

   The second convolutional layer has 256 filters of size 5x5. Again, ReLU activation and LRN are applied, followed by max-pooling.

4. **Third Convolutional Layer**

   This layer has 384 filters of size 3x3, with ReLU activation.

5. **Fourth and Fifth Convolutional Layers**

   The fourth and fifth convolutional layers, respectively, include 384 and 256 filters of size 3x3. ReLU activation is used by both layers.

6. **Fully Connected Layers**

   The last three layers are fully connected. The first fully connected layer has 4096 neurons, followed by a dropout layer for regularization. The second fully connected layer also has 4096 neurons, followed by another dropout layer. The final fully connected layer has 1000 neurons, corresponding to the 1000 classes in the ImageNet dataset.

7. **Output Layer**

   The output layer employs softmax activation to produce the probability distribution over the 1000 classes of the ImageNet dataset but in our case we just change number of classes to 5 beacause we have just 5 classes of diabetic retinopathy.

3.2.9 DenseNet-121

DenseNet-121 is a powerful deep learning model for image classification and is known for its dense connectivity and has achieved state-of-the-art performance on various computer vision tasks and this why we choose it for classification diabetic retinopathy

3.2.9.1 DenseNet-121 architecture:

In general, DenseNet-121 comprises of multiple dense blocks, transition layers, and a final classification layer. The feature that defines DenseNet-121 Each dense block consists of several convolutional layers, and each layer receives inputs from all preceding layers within the same block (see Figure 1.22).

After we talk in general now we will talk in detail about comprises of DenseNet-121

Here is architecture of DenseNet-121 Look at 3.11

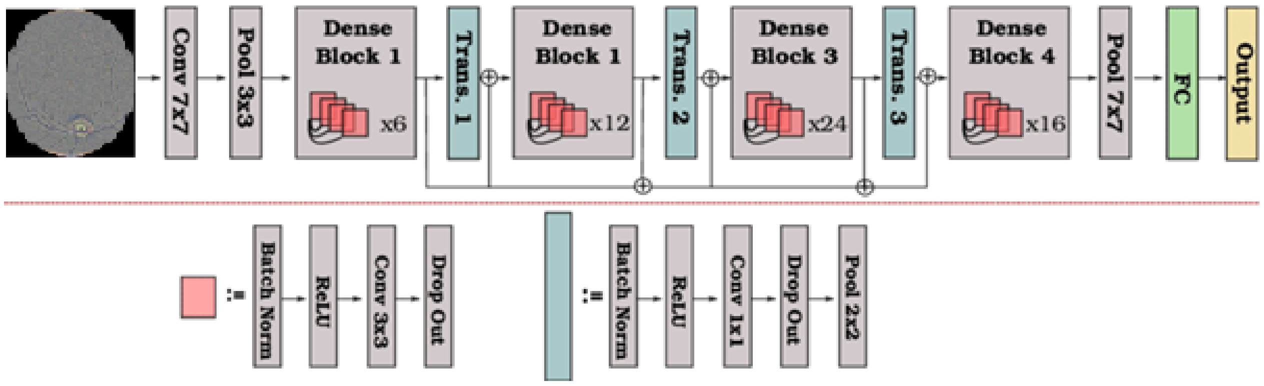


Figure 3.11: Densenet-121 architecture

1. **Dense Blocks**

   Like we say every dense block has many convolutional layers and within the same block, each layer's output is combined with the inputs from all layers that came before it and that lead to improve model performance. Look at figure 3.5 there is 4 dense blocks and each block have different layers like block 1 have x6 convolutions layers and each layer consist of batch normalization and relu function then 3*3 convolution layer then drop out (see Figure 3.11).

2. **Transition Layers**

   Transition layers are an essential component of DenseNet-121 positioned between his dense blocks (see Figure 3.11).They are responsible for reducing the spatial dimensions of the feature maps between dense blocks because they use downsampling operation and this operation can be achieved by techniques like average pooling or strided convulsion.

And also they play important role in controlling model complexity and facilitates the sharing of features, enhances the gradient flow and strengthens the representation capabilities of the network.

3. **Classification Layer**

   The classification layer is the final component of the DenseNet-121 architecture It applies operations on the feature maps produced by the dense blocks or transition layers before it in order to get the predicted class probabilities.it consists of Global average pooling and fully connect layer and finally SoftMax activation (see Figure 3.11).
   The classification layer typically begins with a global average pooling operation this operation reduces the spatial dimensions of the feature maps to a fixed size, regardless of the input image size. Then there is fully connected layer consists of a set of neurons and each connected to every element of the feature vector. The purpose of the fully connected layer is to learn complex relationships between the features and the target classes then finally the output of connected layer pass through an activation function like SoftMax which converts the raw scores into class probabilities.
   Now after we discuss the best model that we used it in our work classification of diabetic retinopathy now we will move to detection part, in this part we will discuss yolo architecture and why we choose yolo and how yolo learn

3.2.10 YOLOv8

You Only Look Once, YOLO gained popularity for object detection tasks due its speed and accuracy. Yolo's primary purpose is to detect and locate objects in images and videos. It focuses primarily on recognizing and categorizing objects based on their visual aspect. YOLO also used in medical imaging to detect and diagnose diseases like X-rays or MRIs because YOLO can identify abnormal regions, lesions or specific patterns associated with diseases and because of that we choose YOLO in our work for detection diabetic retinopathy.
here is deferent yolo version see Figure **??**

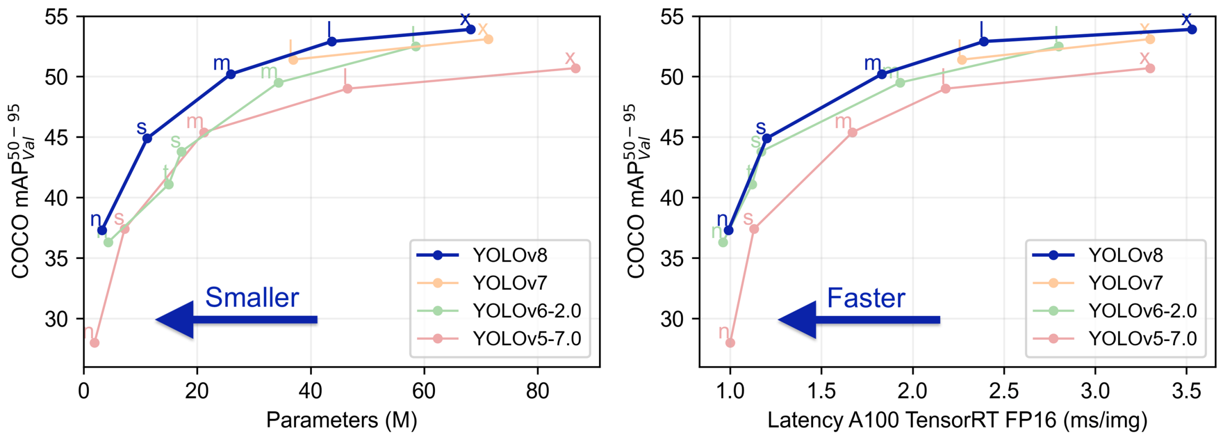


Figure 3.12: YOLO versions [59]

Like we see in figure **??** YOLOV8 is the most powerful and accurate that other YOLO versions. In our work we don't reel time detection we need accurate detection and because of that we choose YOLOv8 L model for our work.

3.2.10.1 YOLO learn

It works by dividing an input picture into a grid and determining the bounding boxes and class probabilities for each grid cell's objects. Here are the key aspects of how YOLO learns:

1. **Single Pass Detection** :

   YOLO processes the entire image in a single pass, enabling lesion detection of diabetic retinopathy. The network predicts bounding boxes and class probabilities directly using convolutional layers, without the need for a separate region.

2. **Loss Function** :

   YOLO uses a specific loss function that combines localization loss (to minimize bounding box coordinate errors), confidence loss (to penalize false positives and false negatives), and classification loss (to classify object classes). This loss guides the model during training to improve detection accuracy.

3. **Anchors** :

   YOLO uses predefined anchor boxes of different sizes and aspect ratios. These anchor boxes are used to anchor the predicted bounding boxes, allowing the model to detect objects at various scales and orientations.

4. **Non-Maximum Suppression** :

   YOLO applies non-maximum suppression during inference to eliminate duplicate detections and refine the final bounding boxes. This post-processing step removes overlapping boxes with lower confidence scores, keeping only the most confident detection.

3.2.11 Prediction

in this part the third split of dataset (test) will be used to test our models

in dense net model

- **Forward Pass**

  in dense net model The input image is forward-propagated through the DenseNet network where each layer receives feature maps from all preceding layers of DenseNet network

- **Feature Extraction**

  During the forward pass, DenseNet extracts hierarchical features from the image as it passes through different layers of the network

- **Global Average Pooling**

  After feature extraction DenseNet applies global average pooling. This process reduces the spatial dimensions and converts the feature maps into a fixed-length feature vector.

- **Classification**

  The global average pooled features are then fed into a fully connected layer or a softmax classifier, which then generates predicted probabilities for each class. The displayed class label for the input picture is taken from the class with the highest probability.

In YOLO model

- **Forward Pass**

  The input image is forward-propagated through the YOLO network then the network devides image into grid and applies a series of convolutional layers to extract features

- **Bounding Box and Class Prediction**

  For each grid cell, YOLO predicts multiple bounding boxes and their corresponding confidence scores, the confidence score represents the probability of containing an object and the predicted bounding boxes represent the location of detected object

- **Non-Maximum Suppression**

  YOLO uses non-maximum suppression to refine the predictions after collecting the bounding box predictions. This step removes overlapping boxes with lower confidence scores, keeping only the most confident detection

- **Post-Processing**

  The final item detection predictions are the bounding boxes, class labels, and confidence scores. The Bounding boxes on the input image show the recognized lesion of diabetic retinopathy and their positions.

3.2.12 Evaluation of Models

We need to evaluate our deep learning model before deploy it in real life because we need to make sure that our model has excellent performance to diagnose patient of diabetic retinopathy and to do that there are many different types of evaluation metrics available to evaluate the model like true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). All that predictions used to calculate accuracy, precision, F1 score and mean average precision according to 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 Conclusion

The purpose of this chapter was to present the general and specific design of our deep learning model of diabetic retinopathy classification system. Based on the different datasets and CNN architectures we employed, we could deduce in the following chapter how to implement diabetic retinopathy detection architectures and discuss implementation and experiments and the obtained results.

Chapter 4

implementation and Results

4.1 Introduction

the previous chapter covered The global and detailed architecture of our system and the dataset's structure.In this chapter we will mention tools and frameworks, implement our system design and present code for each system design phase.Following that, we will summarize the remaining parameters and give the findings for each proposed structure.Finally, we will compare our findings with previous related works.

4.2 implementation of a Deep Learning architecture

4.2.1 Frameworks , tools and libraries

- **Python**

  Guido van Rossum introduced the high-level programming language Python in 1991 . Python now is used extensively in various domains like web development, data analysis, machine learning, automation and scripting [36].

Figure 4.1: Python logo

- **React JS**

  React.js or React is a JavaScript library for building interactive user interfaces for web applications.powred by facebook [22].

Figure 4.2: React logo

- **FastApi**

  FastAPI is a Python web framework for building high performance APIs quickly and easily [48].

Figure 4.3: FastApi logo

- **Google Colab**

  Colab, or Google Colaboratory, is an online platform for coding in Python without the need for local installation. It provides a cloud-based environment with features like collaborative editing and access to powerful hardware resources [38].

Table 4.1: Google Colab resources

	GPU	**Runtime**	**RAM**	**disk capacity**
Google Colab	T4	12h	12GB	60GB

Figure 4.4: Google Colab logo

- **OpenCv**

  OpenCV or Open Source Computer Vision Library is a widely used open-source library for computer vision and image processing tasks. It offers a range of functions and algorithms for tasks like image manipulation, object detection and video analysis [46].

Figure 4.5: OpenCv logo

- **Pycharm**

  Pycharm is a Python development environment (IDE) .

Figure 4.6: Pycharm logo

- **Matplotlib**

  Matplotlib is a Python toolkit for making flexible plots and visualizations including line graphs and scatter plots. Data analysis and scientific study both make extensive use of it [56].

Figure 4.7: Matplotlib logo

- **Keras**

  An effective open source Python library for creating and analyzing deep learning models is called Keras [55].

Figure 4.8: Keras logo

- **TennsorFlow**

  TensorFlow is a powerful open source machine learning framework that was created by Google.it provides a variety of tools and resources for building and deploying machine learning models [57].

Figure 4.9: Tensorflow logo

- **NumPy**

  A well-known Python package for numerical computation is called NumPy. It offers mathematical functions for data management and analysis as well as efficient operations on huge multidimensional arrays [31].

Figure 4.10: NumPy logo

- **Kaggle**

  Kaggle is a platform for data science competitions and collaboration where users can access and use real world datasets [41].

Figure 4.11: Kaggle logo

4.2.2 Dataset preparation and preprocessing

1. **Download Dataset to Google Colab**

   First, we used the dataset API to download the Kaggle datasets to our Google Colab.

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

```

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

   go to Figure 3.8 to see image before and after using Gaussian filter

   (a) **cv2.GaussianBlur**(image, kernel size, sigmaX).

   applies a Gaussian blur filter to the input image "img". The parameter (0,0) specifies the size of the Gaussian kernel which is automatically calculated based on the sigma value of 10. This step blurs the image, reducing noise and smoothing out details.

   (b) **cv2.addWeighted**(image1,alpha,image2,beta,gamma).

   image1 = img - This parameter represents the input image.

   alpha = 4 – The second parameter is the weight or scaling factor applied to the input image (img). In this case, the weight of 4 means that the contribution of img will be significantly increased in the final result.

   image 2 = cv2.GaussianBlur(img, (0,0), 10) – read (a)

   beta = (-4) - The fourth parameter represents the weight or scaling factor applied to the Gaussian blurred image. In this case, a weight of -4 means that the contribution of the blurred image will be reduced in the final result.

   gamma = 128 - The fifth parameter is the optional scalar value gamma added to the weighted sum. It adjusts the brightness of the final image. In this case, a value of 128 is added to the weighted sum.

   . In the code below we apply gaussian filter , Circle crop and resizing image to 244*244

```
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
    #plt.imshow(img)
    gaussian = cv2.addWeighted (img , 4, cv2.GaussianBlur(img ,(0
   ,0) , 10) , -4, 128) # apply gaussin filtering
    processed_image  = circle_crop ( gaussian ) #applying Circle
   crop
     processed_image =cv2.resize(cv2.cvtColor(processed_image, cv2
   .COLOR_BGR2RGB),(224, 224)) #resizing
    cv2.imwrite (file , processed_image ) # save image

```

Listing 4.1: preprocessing

3. **Splitting and preparing Dataset**

   split_folder.ratio was used to divide our dataset into three subsets. split_folder.ratio parameters:

   splitfolders.ratio(path of input folder,path of output,seed,ratio)

```
pip install split-folders
import splitfolders
input_folder ="/content/dataset" # dataset path
output ="/content/retinopathy" # 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 4.2: Splitting Dataset

   Now after we split our dataset to 70% for training ,20% for validation and 10% for test we need to prepare it for our models

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
trdata = ImageDataGenerator()
traindata = trdata.flow_from_directory( directory ="/content/
    retinopathy/train"#training dataset directory
,target_size =(224 ,224) )
tsdata = ImageDataGenerator()
testdata = tsdata.flow_from_directory( directory ="/content/
    retinopathy/val"#validation dataset directory
```

```
                                                , target_size =(224 ,224) )
```

Listing 4.3: Dataset preparation

	description
ImageDataGenerator()	Generate tensor image data in batches using real-time data augmentation.
target_size	All photos will be downsized to (224,224) if the image size does not correspond to the desired size.For us, resizing images happens already in the preprocessing stage.

Table 4.2: Dataset preparation using ImageDataGenerator

4.2.3 Building our Models

4.2.3.1 Building our classification models DenseNet-121 and AlexNet

1. **Import libraries and modules**

   First we need to import important libraries likes keras and tensorflow numpy

```python
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,BatchNormalization,
    GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping,
    ReduceLROnPlateau
from sklearn.utils.class_weight import compute_class_weight



```

Listing 4.4: necessary imports for building a CNN Molde

2. **DenseNet-121 parameters initialization**

   Since we have various data structures we are going to delay initialization until the following chapter where we will provide initialization details for each structure.

3. **Creating Diabetic Retinopathy Classification DenseNet-121 Model**

   After testing many DenseNet-121 configurations the configuration that will be described produced the greatest results.

   (a) **Used Sequential Model**:

```
    model = Sequential()

```

Listing 4.5: Sequential model

We used Sequential() for building DenseNet-121 and AlexNet The sequential model is a technique for building deep learning models that involves making an instance of the Sequential class and adding model layers to it.

   (b) **Used DenseNet-121 Architecture**

```
# Load pre-trained model
base_model = DenseNet121(weights='imagenet', include_top=False,
    input_shape=(224,224,3))

# Add classification layers on top of base model
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.1),
    Dense(64, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(128, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(256, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(1024, activation='relu'),
    BatchNormalization(),
    Dropout(0.1),
    Dense(5, activation='softmax')
])

```

Listing 4.6: Diabetic retinopathy classification DenseNet Model Architecture

First we start our code by loading the **pre-trained DenseNet121 model** from the Keras library .The weights='imagenet' argument specifies that the pre-trained weights should be loaded from the **ImageNet** dataset .we used **"include_top=False"** argument to indicates that the last fully connected layer (top layer) should not be included, as we will be adding our own classification layers. Then we creat a sequential model .sequentail alow as to build a deeplearning model by staking layers on top of each other . The first layer added to the model is the **base_model**, which is the **pre-trained DenseNet121 model** loaded in the previous step. This serves as the feature extraction backbone for our model. Following the base_model, a **GlobalAveragePooling2D** layer is added .After the pooling layer we explained what is GlobalAveragePooling2D in densenet-121 architecture then, several **Dropout layers** are added with a **dropout rate of 0.1** Then **A series of Dense layers** are added with different output sizes **(64, 128, 256, 512, 512, 1024)** and **activation functions ('relu')**. These layers introduce non-linearity to the model and allow it to learn complex patterns from the extracted features and Between each **Dense layer**, a **BatchNormalization layer** is added. **Finally**, the last Dense layer is added with an output size of 5 and an **activation function of 'softmax'**. This layer outputs the predicted probabilities for the **5 classes in the classification task**.

(c) **Used AlexNet Architecture**

```
input_shape = (224, 224, 3)
num_classes = 5

alexnet_model = Sequential()

# Layer 1: Convolutional layer with 96 filters, 11x11 kernel size,
    and ReLU activation
alexnet_model.add(Conv2D(96, kernel_size=(11, 11), strides=(4, 4),
    padding='valid',
                         activation='relu', input_shape=input_shape))
```

```
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'))
# Layer 2: Convolutional layer with 256 filters, 5x5 kernel size, and ReLU activation
alexnet_model.add(Conv2D(256, kernel_size=(5, 5), strides=(1, 1), padding='same', activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'))
# Layer 3: Convolutional layer with 384 filters, 3x3 kernel size, and ReLU activation
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu'))
# Layer 4: Convolutional layer with 384 filters, 3x3 kernel size, and ReLU activation
alexnet_model.add(Conv2D(384, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu'))
# Layer 5: Convolutional layer with 256 filters, 3x3 kernel size, and ReLU activation
alexnet_model.add(Conv2D(256, kernel_size=(3, 3), strides=(1, 1), padding='same', activation='relu'))
alexnet_model.add(MaxPooling2D(pool_size=(3, 3), strides=(2, 2), padding='valid'))

# Flatten the feature maps
alexnet_model.add(Flatten())

# Layer 6: Fully connected layer with 4096 units and ReLU activation
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))

# Layer 7: Fully connected layer with 4096 units and ReLU activation
alexnet_model.add(Dense(4096, activation='relu'))
alexnet_model.add(Dropout(0.5))

# Layer 8: Output layer with num_classes units and softmax activation
alexnet_model.add(Dense(num_classes, activation='softmax'))


```

Listing 4.7: Diabetic retinopathy classification AlexNet Model Architecture

(d) **Used optimizer**

```
tf.keras.optimizers.legacy.Adam(lr=0.0001, decay=1e-9)
```

Listing 4.8: Used optimizer

- lr=0.0001: This sets the learning rate for the optimizer. The learning rate determines the step size taken during each update of the model's parameters. A smaller learning rate can lead to slower convergence but may result in better optimization, while a larger learning rate can lead to faster convergence but may risk overshooting the optimal solution.
- decay=1e-9: This parameter specifies the learning rate decay. It reduces the learning rate over time to fine-tune the optimization process. A small decay value ensures a gradual decrease in the learning rate throughout the training process, helping the model converge smoothly.

(e) **used loss function** because we have 5 classes (multiclassication) we used "categorical_crossentropy"

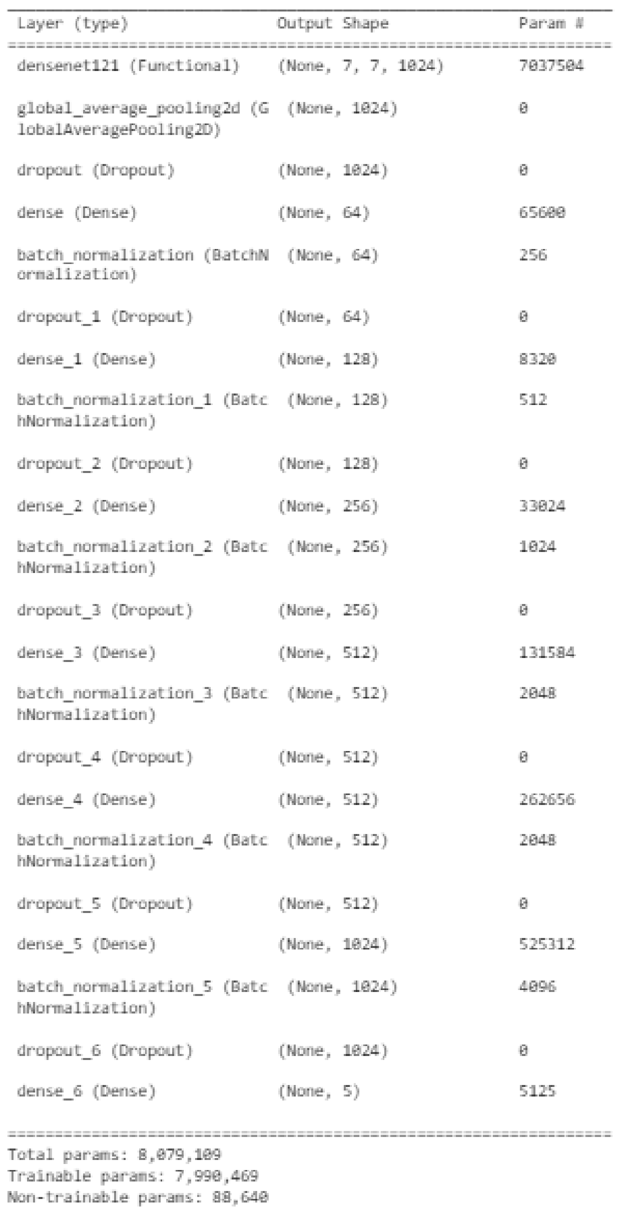
4. **Model summary**

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

```
model.summary()
```

Listing 4.9: Model summary

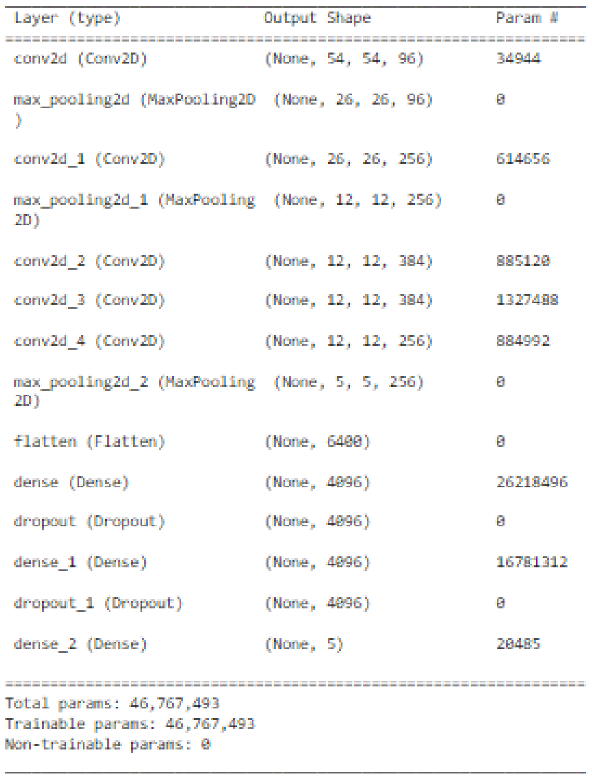
- DenseNet-121 summary



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 7, 7, 1024)        7037504

 global_average_pooling2d (G  (None, 1024)             0
 lobalAveragePooling2D)

 dropout (Dropout)           (None, 1024)              0

 dense (Dense)               (None, 64)                65600

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dropout_1 (Dropout)         (None, 64)                0

 dense_1 (Dense)             (None, 128)               8320

 batch_normalization_1 (Batc  (None, 128)              512
 hNormalization)

 dropout_2 (Dropout)         (None, 128)               0

 dense_2 (Dense)             (None, 256)               33024

 batch_normalization_2 (Batc  (None, 256)              1024
 hNormalization)

 dropout_3 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 512)               131584

 batch_normalization_3 (Batc  (None, 512)              2048
 hNormalization)

 dropout_4 (Dropout)         (None, 512)               0

 dense_4 (Dense)             (None, 512)               262656

 batch_normalization_4 (Batc  (None, 512)              2048
 hNormalization)

 dropout_5 (Dropout)         (None, 512)               0

 dense_5 (Dense)             (None, 1024)              525312

 batch_normalization_5 (Batc  (None, 1024)             4096
 hNormalization)

 dropout_6 (Dropout)         (None, 1024)              0

 dense_6 (Dense)             (None, 5)                 5125

=================================================================
Total params: 8,079,109
Trainable params: 7,990,469
Non-trainable params: 88,640
```

Figure 4.12: DenseNet summary

- AlexNet summary



```
Layer (type)                Output Shape              Param #
=================================================================
conv2d (Conv2D)             (None, 54, 54, 96)        34944

max_pooling2d (MaxPooling2D  (None, 26, 26, 96)       0
)

conv2d_1 (Conv2D)           (None, 26, 26, 256)       614656

max_pooling2d_1 (MaxPooling  (None, 12, 12, 256)      0
2D)

conv2d_2 (Conv2D)           (None, 12, 12, 384)       885120

conv2d_3 (Conv2D)           (None, 12, 12, 384)       1327488

conv2d_4 (Conv2D)           (None, 12, 12, 256)       884992

max_pooling2d_2 (MaxPooling  (None, 5, 5, 256)        0
2D)

flatten (Flatten)           (None, 6400)              0

dense (Dense)               (None, 4096)              26218496

dropout (Dropout)           (None, 4096)              0

dense_1 (Dense)             (None, 4096)              16781312

dropout_1 (Dropout)         (None, 4096)              0

dense_2 (Dense)             (None, 5)                 20485

=================================================================
Total params: 46,767,493
Trainable params: 46,767,493
Non-trainable params: 0
_________________________________________________________________
```

Figure 4.13: AlexNet summary

5. **Compiling classification Models** To build our model, we utilized the following code.

```
model.compile(tf.keras.optimizers.legacy.Adam(lr=0.0001, decay=1e-9),
    loss='categorical_crossentropy', metrics=['accuracy','Precision','
    FalseNegatives','FalsePositives','TrueNegatives','TruePositives'])
```

Listing 4.10: code of Compiling our models

Function	**Arguments**

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

Table 4.3: Compiling our Models arguments

4.2.3.2 Building our Detection model YOLOv8

1. **import libaries and modules** first we install library ultralytics and import YOLO from the same library

```
!pip install ultralytics -q
from ultralytics import YOLO

```

Listing 4.11: necessary imports for building a CNN Molde

2. **Creating Diabetic Retinopathy Detection YOLOv8 Mode** Now after we import necessary libraries we will upload a pretrained model YOLOv8 model

```
model = YOLO('yolov8l.pt')

```

Listing 4.12: necessary imports for building a CNN Molde

4.2.4 Training Our Models

4.2.4.1 AlexNet and DenseNet training

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)
earlystop = EarlyStopping(monitor='val_accuracy', min_delta=0.001,
   patience=10, verbose=1, mode='max', restore_best_weights=True)
lr_scheduler = ReduceLROnPlateau(monitor='val_accuracy', factor=0.5,
   patience=5, verbose=1, mode='max', min_lr=0.000001)

# Train model
history = model.fit(
    traindata,
    epochs=50,
    validation_data=testdata,
    class_weight=class_weights_dict,
    callbacks=[earlystop, lr_scheduler,checkpoint])
```

Listing 4.13: Fitting Model

- **ModelCheckpoint**

  Here, a ModelCheckpoint callback is created. This callback saves the model's weights at certain intervals during training. The arguments provided are as follows:

  – **modelName.h5** : The filename to save the model's weights.

  – **monitor='monitor'** : The value to monitor during training, typically a metric like accuracy or loss.

  – **verbose=1** : Verbosity mode. It displays information about the checkpoint being saved

  – **save_best_only=True** : Specifies to save only the best model based on the monitored value.

  – **save_weights_only=False** : Determines whether to save only the model's weights (True) or the entire model (False).

  – **mode='auto'** : The mode for comparison of the monitored value. 'auto' infers it automatically based on the monitored metric.

  – **period=1** : The frequency, in epochs, to save the model.

- **EarlyStopping**

  The EarlyStopping callback is created to stop the training early if a certain condition is met usualy used to avoid over fiting. The arguments are

  – **monitor='val_accuracy'** : The value to monitor, in this case, the validation accuracy.

  – **min_delta=0.001** : The minimum change in the monitored value to qualify as an improvement.

  – **patience=10** : The number of epochs with no improvement after which training will be stopped.

  – **verbose=1** : Verbosity mode. It displays information about early stopping being triggered.

  – **mode='max'** : The comparison mode. 'max' means the monitored value should be maximized.

  - **restore_best_weights=True** : Restores the weights of the best model found during training.

- **ReduceLROnPlateau**

  The ReduceLROnPlateau callback is used to reduce the learning rate when a certain metric plateaus. The arguments are:

  - **monitor=val_accuracy** : The value to monitor for a plateau, which is the validation accuracy.
  - **factor=0.5** : The factor by which the learning rate will be reduced (multiplied) when a plateau is detected.
  - **patience=5** : The number of epochs with no improvement after which the learning rate will be reduced.
  - **verbose=1** : Verbosity mode. It displays information about the learning rate reduction.
  - **mode='max'** : The mode for comparison of the monitored value. 'max' means it should be maximized.
  - **min_lr=0.000001** : The minimum learning rate to be reached.

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

  Because we have different structure binary and multi classification structures there is some parameters will be changed like the last dense output layer will be equal to number of classes of each structure.

Function	Arguments
fit_generator	-epochs = 50 -shuffle , when = true , dataset will be shuffled before the training CNN class_weight , when data is unbalanced , class weight takes the value 'balanced' callbacks=[earlystop, lr_scheduler,checkpoint] , fit_model uses the 2 previous functions

Table 4.4: Function description for the model fitting

4.2.4.2 YOLOv8 training

after we upload the pretrained model of YOLOv8 now we will do the training. we set the path of dataset of taring and validation in data.yaml file and we initialise number of epochs to 50 and size of images to 640*640.

```
!yolo task=detect mode=train model=yolov8l.pt data={dataset.location}/data
    .yaml epochs=150 imgsz=640

```

Listing 4.14: necessary imports for building a CNN Molde

4.3 Testing our Model

4.3.1 Testing DenseNet and AlexNet

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.

```python
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 tensorflow.keras.utils import img_to_array

import numpy as np
import argparse
import glob
import PIL
import os
import os.path
from PIL import Image
indiceDR0 =0 #NO DR counter
indiceDR1 =0# YES DR counter
indiceDR2 =0
indiceDR3 =0
indiceDR4 =0
total =0# total of images
listt=[]
listt2=[]
listt4=[]

f = r'/content/dataset15/2' # path of third subset for the no DR

for file in os. listdir(f):
  f_img = f+"/"+ file
  img = tf.keras.utils.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


  if output [0][0] > output [0][1] and output [0][0] > output [0][2] and
   output[0][0] > output [0][3] and output [0][0] > output [0][4]: # if
   the
    #probability that this image belongs to NO DR is bigger than the
    #probability of that this image belongs to Mild , moderate , and sever
    :
    print ("no", output [0])
    indiceDR0 += 1
    total += 1
  if output [0][1] > output [0][0] and output [0][1] > output [0][2] and
   output[0][1] > output [0][3] and output [0][1] > output [0][4] :
    print("mild", output [0])
    indiceDR1 += 1
    total += 1
  if output [0][2] > output [0][0] and output [0][2] > output [0][1] and
   output[0][2] > output [0][3] and output [0][2] > output [0][4] :
    print("moderate", output [0])
    indiceDR2 += 1
    total += 1
  if output [0][3] > output [0][0] and output [0][3] > output [0][1] and
   output[0][3] > output [0][2] and output [0][3] > output [0][4] :
    print ("sever", output [0])
    indiceDR3 += 1
    total += 1
  if output [0][4] > output [0][0] and output [0][4] > output [0][1] and
   output[0][4] > output [0][2] and output [0][4] > output [0][3] :
    print ("polifrate", output [0])
    indiceDR4 += 1
    total += 1

print(" =============== ")
print(" Diabetic retinopathy predict ")
print(" =============== ")
print("0: ",indiceDR0 ,", 1 ",indiceDR1 ,", 2: ",indiceDR2 ,", 3: ",
   indiceDR3 ,", 4: ",indiceDR4 )
```

Listing 4.15: Testning Model

4.3.2 Testing YOLOv8

Now the testing part we choose the detection task and val mode for test then we change the path of validation part of our dataset by path of test part.

```
!yolo task=detect mode=val model="/content/runs/detect/train2/weights/best
   .pt" data={dataset.location}/data.yaml

```

Listing 4.16: necessary imports for building a CNN Molde

4.4 Application interface

we devloped web appilication using react.js and FastApi to diagnose poeple if they have diabetic retinopathy or not.

The application is designed to be user-friendly and intuitive. When patients visit the app, they can easily upload their OCT images, which provide valuable insights into their eye health. Thanks to the power of React.js, the interface is sleek and responsive, ensuring a seamless user experience

Behind the scenes, FastAPI comes into action. It handles the communication between the frontend and backend, efficiently processing the uploaded OCT images. The image data is securely transmitted to our deep learning model, which has been trained to accurately diagnose diabetic retinopathy then the result return to patient.

our web application composed of 2 interfaces

- Submission interface of the medical image (fundus image)
- Results interface in which we will display the results related to the prediction performed on the submitted photo.

4.4.1 Submission Interface

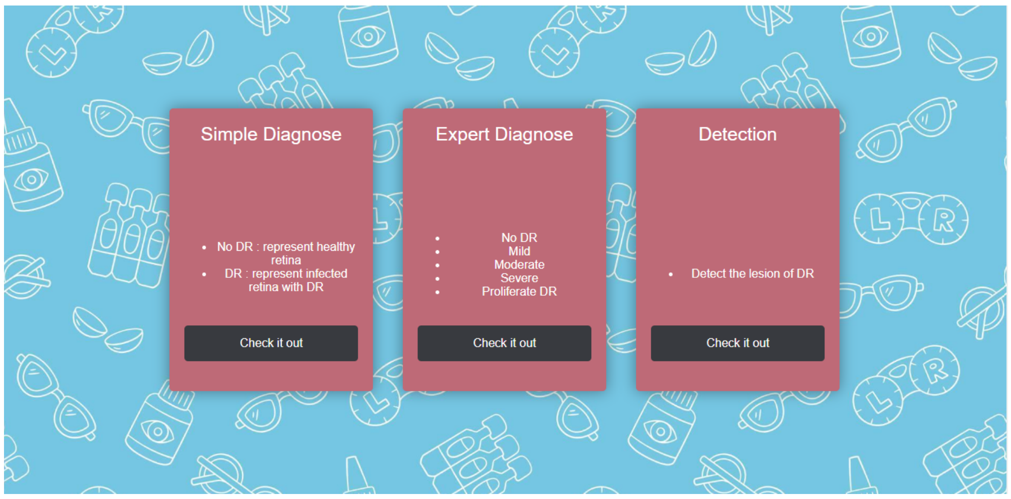


Figure 4.14: Home Interface

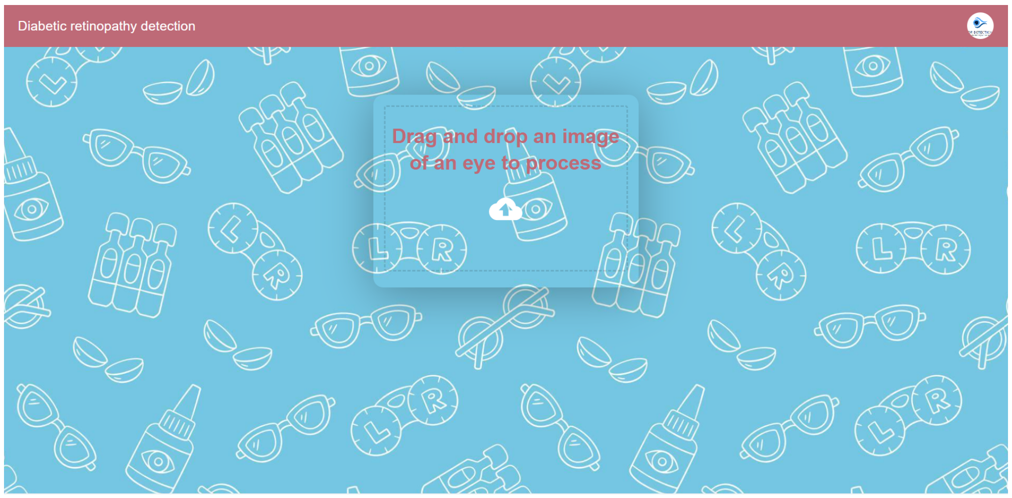

Diabetic retinopathy detection
Drag and drop an image of an eye to process

Figure 4.15: Submission Interface

4.4.2 Results interface

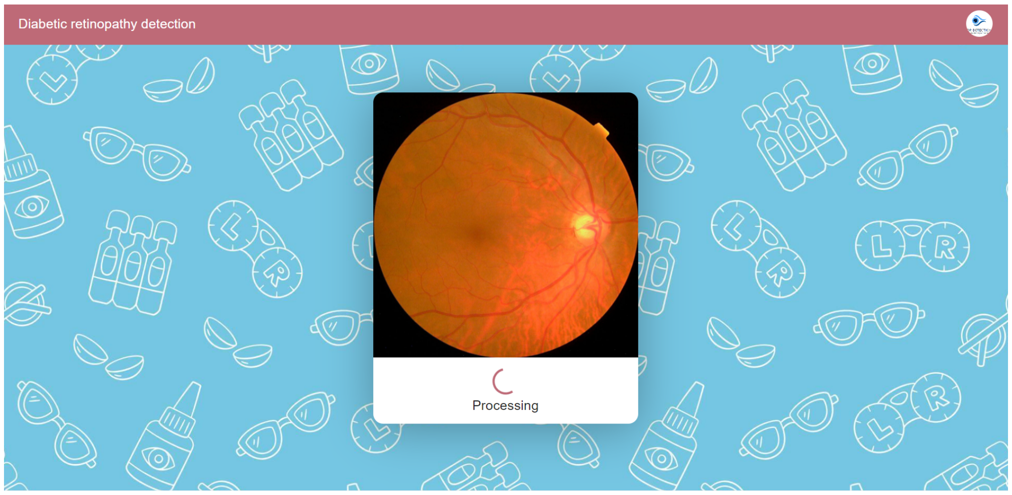


Figure 4.16: Results interface

Experimentation and Results

4.5 First proposed Structure (Binary classification)

4.5.1 The used Dataset

We employ the APTOS 2019 [48] data set 1 described in Chapter 3. The next table describes the original structure of the dataset, including the number of images per category.

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

Table 4.5: Original structure Dataset 1

we explained in chapter 3 how we groped our first structure to get just two classes NO DR and DR.We splited this dataset to 3 subset :training,validation and testing subset as showen in table 2.9

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

Table 4.6: first proposed structure Dataset 1

1. **Specification of parameters initialization for this structure:** The parameters utilized in this structure are shown in the table below.

	Informations
input shape	(224,224,3)
Number of Epochs	50 early stopped
Step per Epoch	total training images / batch size =81
Class weight	not used , balanced dataset
dropOutRate	10%
Patience	5
Last output Layer	Dense(2)
ModelCheckpoint Monitor	val_acc
EarlyStopping Monitor	val_acc

Table 4.7: Table of parameters for first proposed structure

4.5.2 AlexNet

1. **Results and discussion**:Training accuracy, validation accuracy, training loss, and validation loss have all been used to evaluate our model. We give the following graphs to show

how our model performed.

- A plot of accuracy and validation accuracy over epochs:

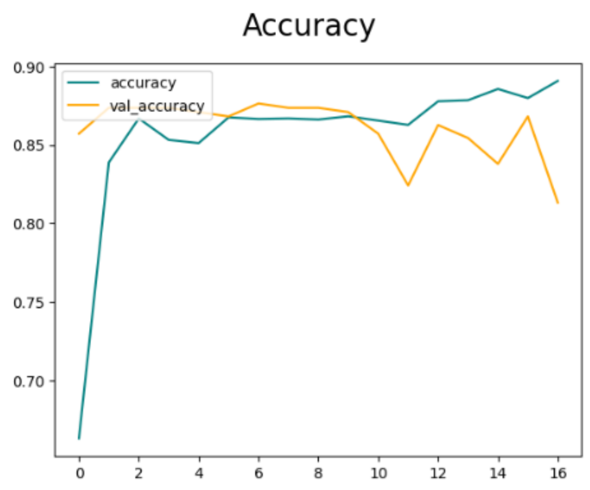


Figure 4.17: AlexNet Accuracy

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

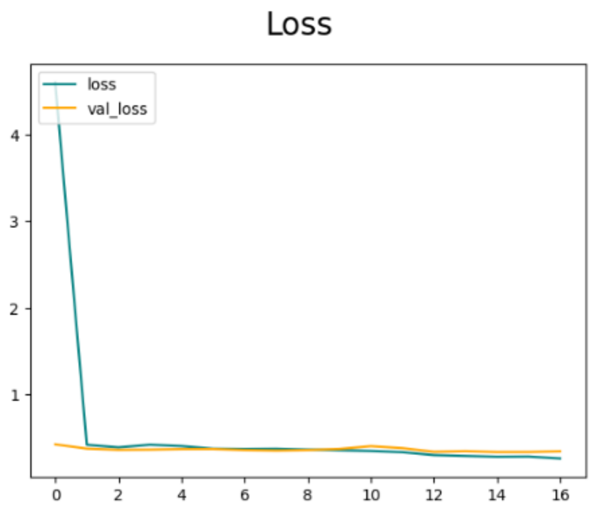


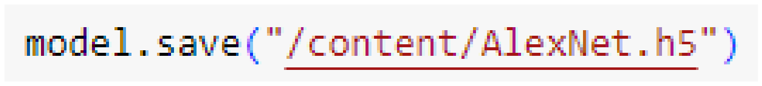
Figure 4.18: AlexNet Loss

- The plots of learning curves 4.17 and 4.18 show a good Fit because:
  - The accuracy and validation accuracy are converging to the same value but when we arrive to epoch 9 the accuracy still increasing and the validation accuracy going down to value 83%.

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

2. **Saving Model**

   the early stopping was in the epoch 17. Performance metrics of this model are in the table 4.8.



```
model.save("/content/AlexNet.h5")
```

Figure 4.19: Saving model

	Accuracy	Validation accuracy	Loss	Validation Loss
AlexNet Model	92.91%	89.07%	26.17%	81.32%

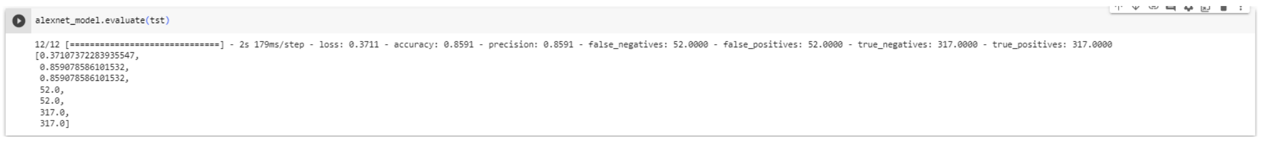
Table 4.8: Table of results Alexnet

3. **Testing Model**

   To see how well our CNN model performs on new images from the Dataset, we will employ the third subset in this phase.

   After testing our model on the third subset:

   - **Metrics results on test subset**



```
alexnet_model.evaluate(tst)

12/12 [==============================] - 2s 179ms/step - loss: 0.3711 - accuracy: 0.8591 - precision: 0.8591 - false_negatives: 52.0000 - false_positives: 52.0000 - true_negatives: 317.0000 - true_positives: 317.0000
[0.37107372283935547,
 0.859078586101532,
 0.859078586101532,
 52.0,
 52.0,
 317.0,
 317.0]
```

Figure 4.20: testing our model

4.5.3 DenseNet-121

1. **Results and discussion**:Training accuracy, validation accuracy, training loss, and validation loss have all been used to evaluate our model. We give the following graphs to show how our model performed.

- A plot of accuracy and validation accuracy over epochs:

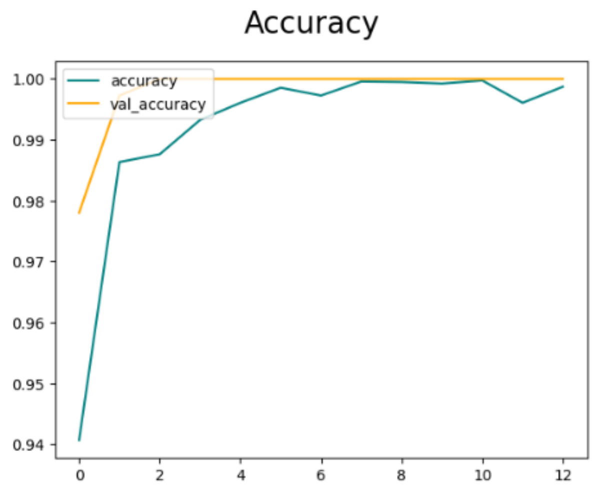


Figure 4.21: DenseNet-121 Accuracy

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

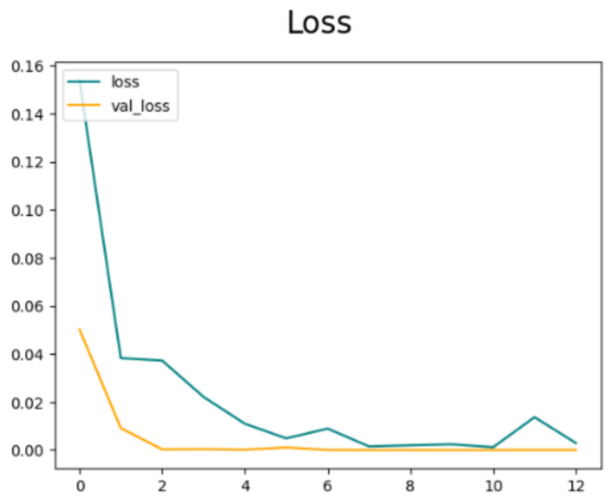


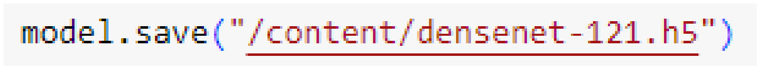
Figure 4.22: DenseNet-121 Loss

- The plots of learning curves 4.21 and 4.22 show a excellent Fit because:
  - The accuracy and validation accuracy are converging to the same value.
  - The loss and loss validation are decreasing to the same value.

in conclusion The training model shows promising performance with an initial accuracy of 94% and a validation accuracy of 97% at epoch 0. The accuracy curves of both the training and validation sets converge to a high accuracy of 100%, indicating that the model is consistently improving and generalizing well to unseen data.

2. **Saving Model**

   the the early stopping was in the epoch 31. Performance metrics of this model are in the table 4.23.



```
model.save("/content/densenet-121.h5")
```

Figure 4.23: Saving model

	Accuracy	Validation accuracy	Loss	Validation Loss
DenseNet-121 Model	99.90%	96.43%	0.33%	10.89%

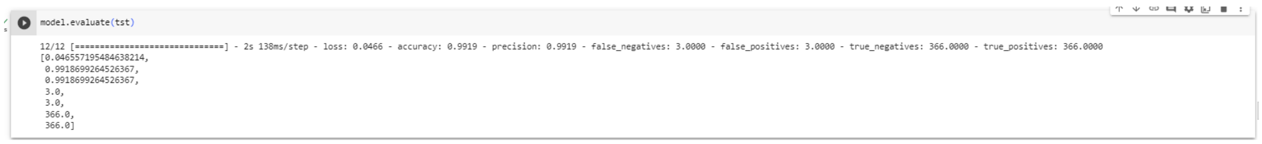
Table 4.9: Table of results DenseNet-121

3. **Testing Model**

   To see how well our CNN model performs on new images from the Dataset, we will employ the third subset in this phase.

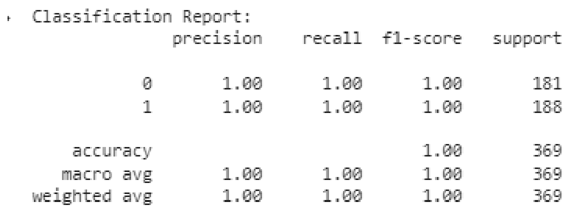
   After testing our model on the third subset:

   - **Metrics results on test subset**



```
model.evaluate(tst)

12/12 [==============================] - 2s 138ms/step - loss: 0.0466 - accuracy: 0.9919 - precision: 0.9919 - false_negatives: 3.0000 - false_positives: 3.0000 - true_negatives: 366.0000 - true_positives: 366.0000
[0.046557195484638214,
 0.9918699264526367,
 0.9918699264526367,
 3.0,
 3.0,
 366.0,
 366.0]
```

Figure 4.24: testing DenseNet-121



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       181
           1       1.00      1.00      1.00       188

    accuracy                           1.00       369
   macro avg       1.00      1.00      1.00       369
weighted avg       1.00      1.00      1.00       369
```

Figure 4.25: Classification Report

- **Confusion matrix**

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

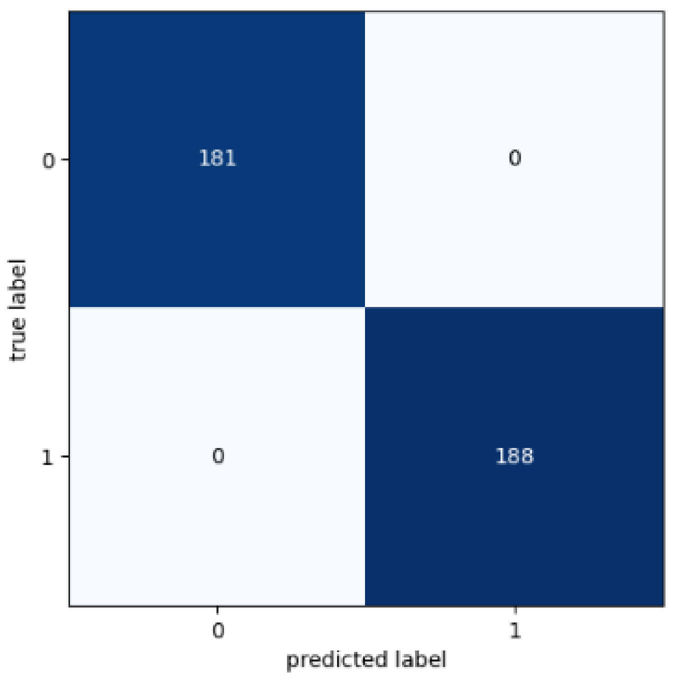


Figure 4.26: Confusion matrix

  - The top-left element, 181, represents the number of samples that belong to the "No DR" class and are correctly predicted as "No DR".
  - The bottom-right element, 188, represents the number of samples that belong to the "DR" class and are correctly predicted as "DR".
  - There are no misclassifications or errors in the prediction, as indicated by the zeros in the off-diagonal elements.

4.5.4 Comparisons of our experiments

After seeing the three trials, we determine that DenseNet-121 is the perfect model for the first structure.The table 4.10 show the comparisons of our experiments.

	Accuracy	**Validation accuracy**	**Test accuracy**	**Loss**	**Validation Loss**
AlexNet	92.91%	89.07%	85.91%	26.17%	81.32%
DenseNet-121	99.87%	100%	100%	0.0029%	0.002%

Table 4.10: Table of Comparisons of our experiments on the first Structure.

4.6 Second proposed Structure (Multi Classification)

4.6.1 The used Dataset

We combine two datasets, Aptos 2015 and Aptos 2019, for the Kaggle competition diabetic retinopathy detection. We use the entire Aptos 2019 dataset as is. From the Aptos 2015 dataset, we carefully select 17,003 samples, excluding low-quality, blurry, dark, and images with camera artifacts see figure 4.27. This curated combination enhances the dataset's quality and diversity for robust diabetic retinopathy detection algorithm development.

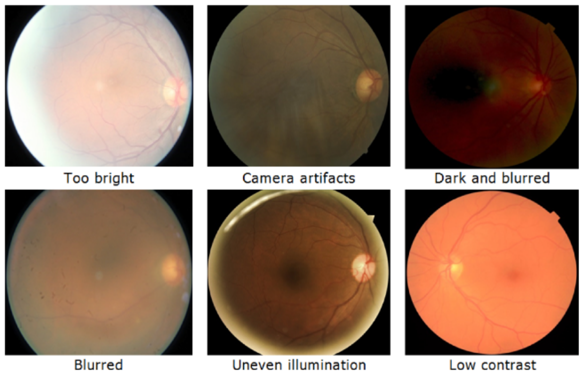


Figure 4.27: low-quality of some images

	Training	Validation	Test	Total
No DR	5770	1650	825	8245
Mild	4733	1367	683	6783
Moderate	3196	911	456	4563
Severe	2462	683	342	3487
Proliferate DR	3771	1082	541	5394
Total	19932	5695	2847	28472

Table 4.11: proposed structure 2

1. **Specification of parameters initialization for this structure:** The parameters utilized in this structure are shown in the table below.

	Informations
input shape	(224,224,3)
Number of Epochs	50 early stopped
Step per Epoch	total training images / batch size =81
Class weight	not used , balanced dataset
dropOutRate	10%
Patience	5
Last output Layer	Dense(2)
ModelCheckpoint Monitor	val_acc
EarlyStopping Monitor	val_acc

Table 4.12: Table of parameters for first proposed structure

4.6.2 AlexNet

1. **Results and discussion**:Training accuracy, validation accuracy, training loss, and validation loss have all been used to evaluate our model. We give the following graphs to show how our model performed.

   - A plot of accuracy and validation accuracy over epochs:

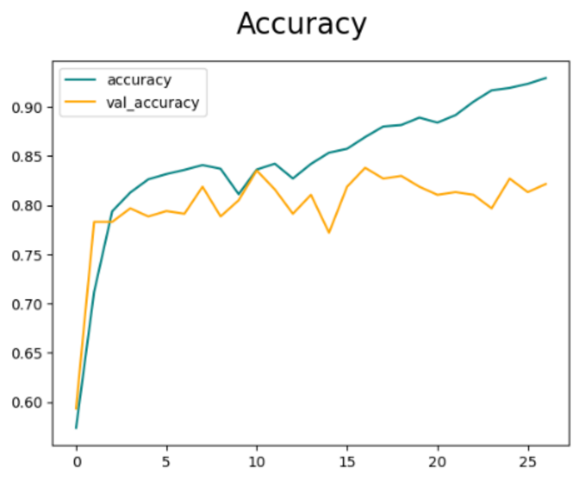


Figure 4.28: AlexNet Accuracy

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

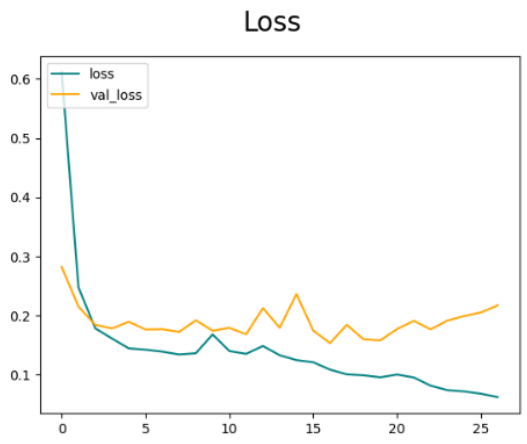


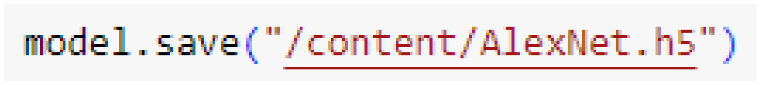
Figure 4.29: AlexNet Loss

   - The plots of learning curves 4.28 and 4.29 show a acceptable Fit because:

  - The accuracy and validation accuracy are converging to the same value but when we arrive to epoch 16 the accuracy still increasing and the validation accuracy going between value 82% and 85%.
  - The loss and loss validation are decreasing to the same value but in epoch 19 there is like overfiting the val loss curve going up.

2. **Saving Model**

   the the early stopping was in the epoch 27. Performance metrics of this model are in the table **??**.



```
model.save("/content/AlexNet.h5")
```

Figure 4.30: Saving model

	Accuracy	**Validation accuracy**	**Loss**	**Validation Loss**
AlexNet Model	92.91%	82.14%	6.15%	21.68%

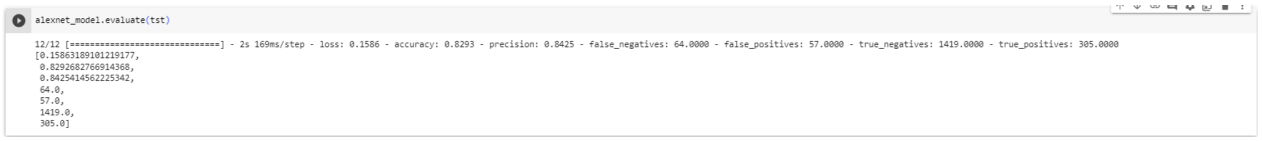
Table 4.13: Table of results Alexnet

3. **Testing Model**

   To see how well our CNN model performs on new images from the Dataset, we will employ the third subset in this phase.

   After testing our model on the third subset:

   - **Metrics results on test subset**



```
alexnet_model.evaluate(tst)

12/12 [==============================] - 2s 169ms/step - loss: 0.1586 - accuracy: 0.8293 - precision: 0.8425 - false_negatives: 64.0000 - false_positives: 57.0000 - true_negatives: 1419.0000 - true_positives: 305.0000
[0.15863189101219177,
 0.8292682766914368,
 0.8425414562225342,
 64.0,
 57.0,
 1419.0,
 305.0]
```

Figure 4.31: testing our model

4.6.3 DenseNet-121

1. **Results and discussion**:Training accuracy, validation accuracy, training loss, and validation loss have all been used to evaluate our model. We give the following graphs to show how our model performed.

   - A plot of accuracy and validation accuracy over epochs:

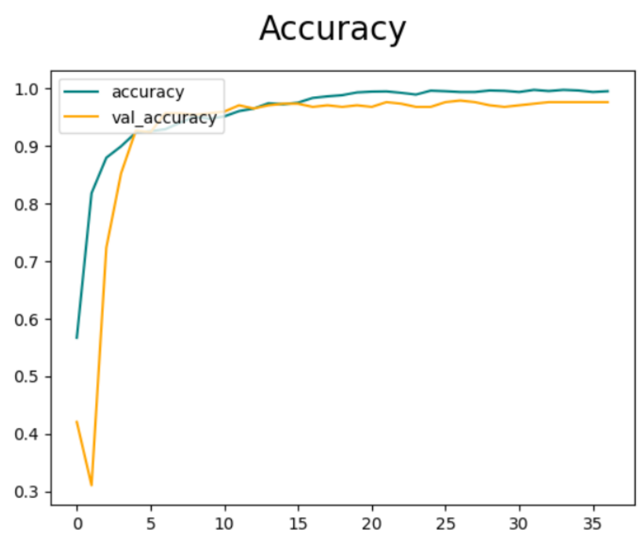


Figure 4.32: DenseNet-121 Accuracy

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

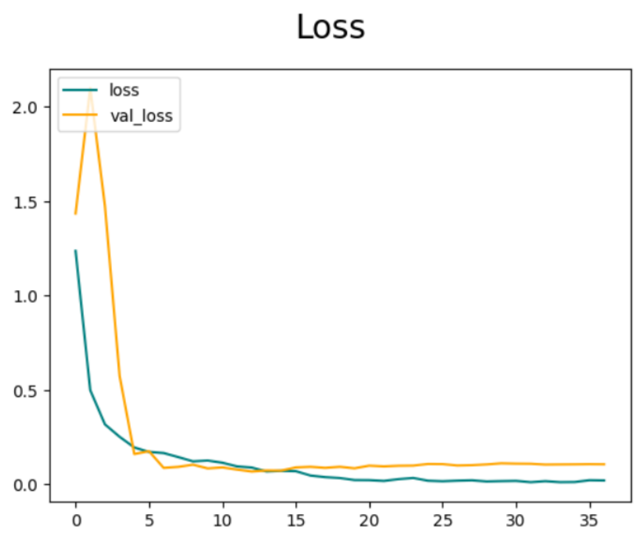
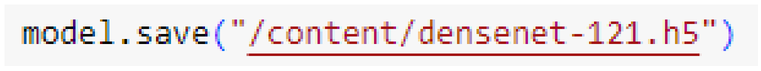


Figure 4.33: DenseNet-121 Loss

- The plots of learning curves 4.32 and 4.33 show a excellent Fit because:
  - The accuracy and validation accuracy are converging to the same value.
  - The loss and loss validation are decreasing to the same value.

2. **Saving Model**

   the the early stopping was in the epoch 27. Performance metrics of this model are in the table 4.34.



```
model.save("/content/densenet-121.h5")
```

Figure 4.34: Saving model

	Accuracy	Validation accuracy	Loss	Validation Loss
DenseNet-121 Model	99.47%	97.53%	2.14%	10.70%

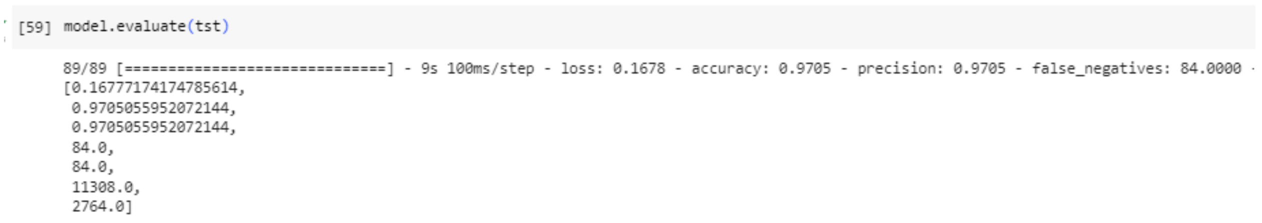
Table 4.14: Table of results DenseNet-121

3. **Testing Model**

   To see how well our CNN model performs on new images from the Dataset, we will employ the third subset in this phase.

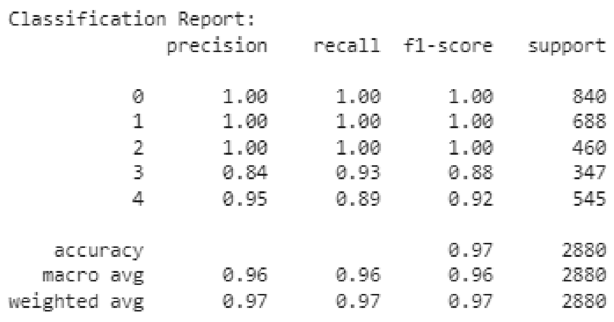
   After testing our model on the third subset:

   - **Metrics results on test subset**



```
[59] model.evaluate(tst)

     89/89 [==============================] - 9s 100ms/step - loss: 0.1678 - accuracy: 0.9705 - precision: 0.9705 - false_negatives: 84.0000 ·
     [0.16777174174785614,
      0.9705055952072144,
      0.9705055952072144,
      84.0,
      84.0,
      11308.0,
      2764.0]
```

Figure 4.35: testing DenseNet-121

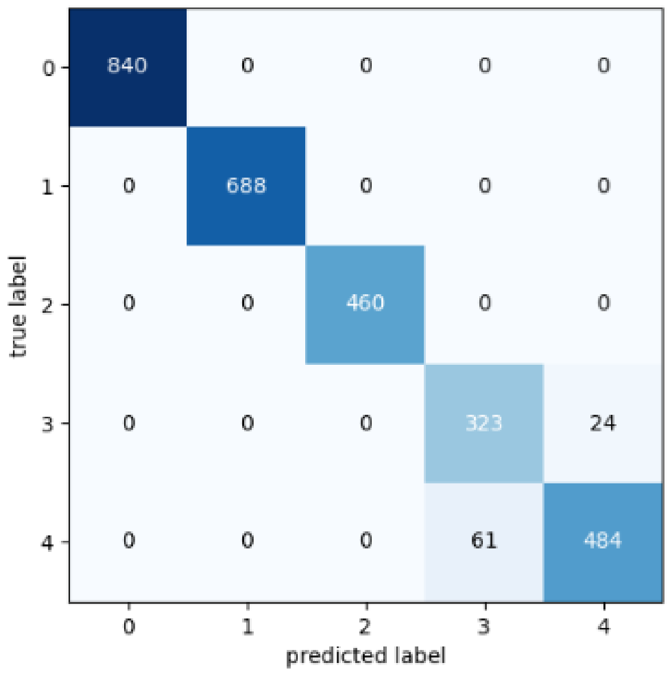


```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       840
           1       1.00      1.00      1.00       688
           2       1.00      1.00      1.00       460
           3       0.84      0.93      0.88       347
           4       0.95      0.89      0.92       545

    accuracy                           0.97      2880
   macro avg       0.96      0.96      0.96      2880
weighted avg       0.97      0.97      0.97      2880
```

Figure 4.36: Classification Report

- **Confusion matrix**

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



true label \ predicted label	0	1	2	3	4
0	840	0	0	0	0
1	0	688	0	0	0
2	0	0	460	0	0
3	0	0	0	323	24
4	0	0	0	61	484

Figure 4.37: Confusion matrix

  - The element at row 1, column 1 (top-left), 840, represents the number of samples that belong to the "No DR" class and are correctly predicted as "No DR".
  - The element at row 2, column 2, 688, represents the number of samples that belong to the "mild" class and are correctly predicted as "mild".

- The element at row 3, column 3, 460, represents the number of samples that belong to the "moderate" class and are correctly predicted as "moderate".
- The element at row 4, column 4, 323, represents the number of samples that belong to the "severe" class and are correctly predicted as "severe".
- The element at row 4, column 5, 24, represents the number of samples that belong to the "severe" class but are misclassified as "proliferate".
- The element at row 5, column 4, 61, represents the number of samples that belong to the "proliferate" class but are misclassified as "severe".
- The element at row 5, column 5, 484, represents the number of samples that belong to the "proliferate" class and are correctly predicted as "proliferate".

in conclusion:

Our model achieved high accuracy in predicting most of the classes, with the majority of the elements in the confusion matrix having high values. However, there are some misclassifications between the "severe" and "proliferate" classes, as indicated by the non-zero values in the off-diagonal elements for those classes

4.6.4 Comparisons of our experiments

After seeing the three trials, we determine that DenseNet-121 is the perfect model for the first structure.The table 4.15 show the comparisons of our experiments.

	Accuracy	**Validation accuracy**	**Test accuracy**	**Loss**	**Validation Loss**
AlexNet	92.91%	82.14%	82.93%	6.15%	21.68%
DenseNet-121	99.47%	97.53%	97.05%	2.14%	10.70%

Table 4.15: Table of Comparisons of our experiments on the first Structure.

4.7 third proposed structure (Detection DR)

4.7.1 The used Dataset

We utilized dataset 3 from Roboflow for this structure.the images annotated to just one classe DR to detect diabetic retinopathy.

Level	**Number of images**
DR	793

Table 4.16: our dataset

1. **Specification of parameters initialization for this structure:** The parameters utilized in this structure are shown in the table below.

Parameter	**Value**
Task	Detect
Mode	Train
Model	yolov8l.pt
Data	{dataset.location}/data.yaml
Epochs	100
Imgsz	640

Table 4.17: YOLO Parameter Initialization

2. **Results and discussion**:Training accuracy, validation accuracy, training loss, and validation loss have all been used to evaluate our model. We give the following graphs to show how our model performed.

    - **Precision Curve**

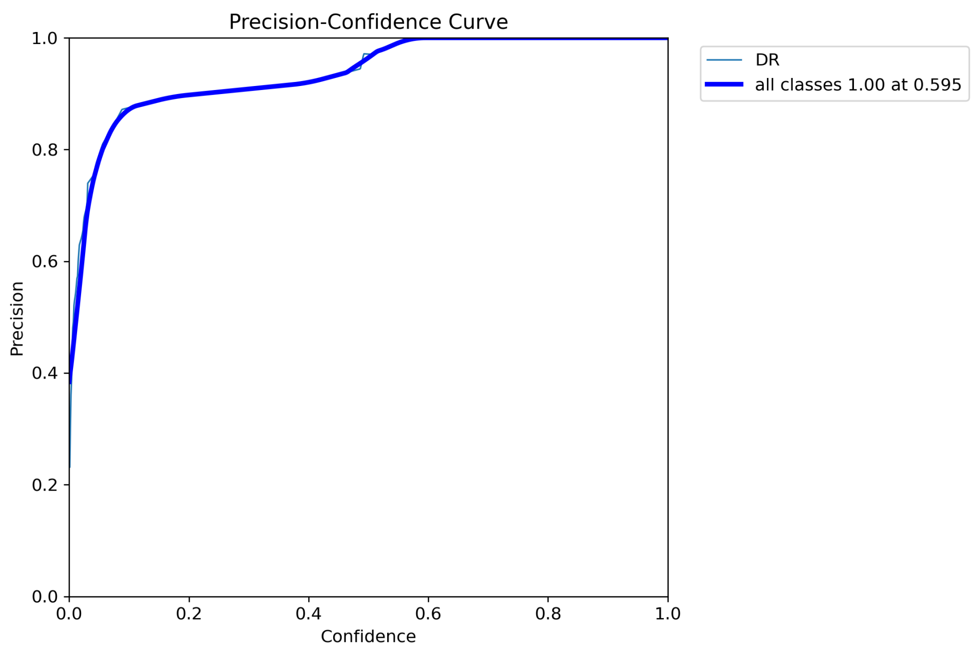


Figure 4.38: The model Accuracy

The precision curve showcases the exceptional precision achieved by the YOLO model . It demonstrates that the model accurately identifies DR with minimal false positives. The curve consistently maintains high precision values across different confidence thresholds , indicating that the YOLO model achieves excellent accuracy in detecting DR.

- **Recall Curve**

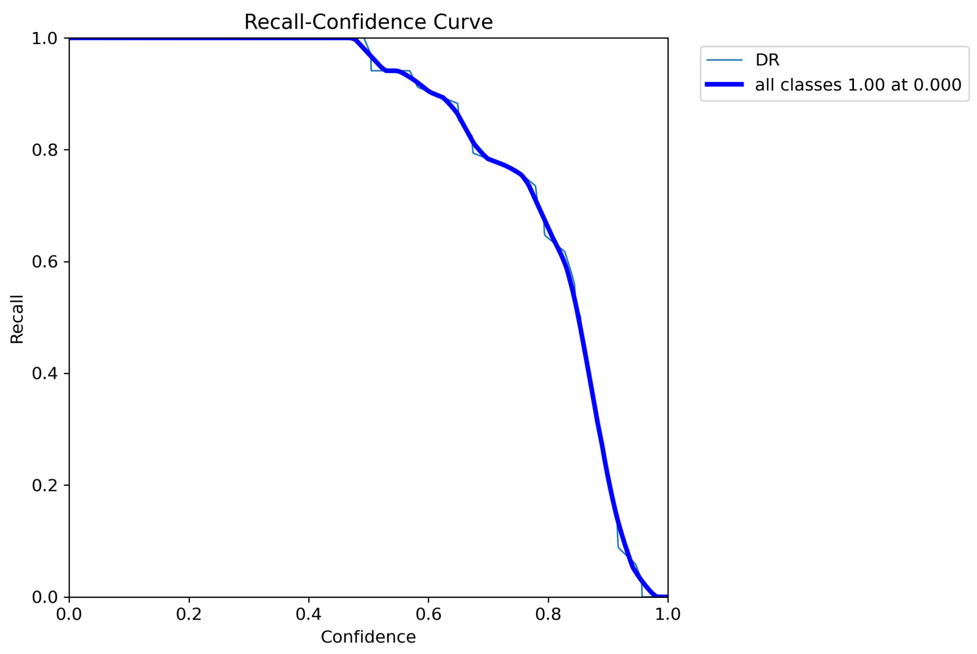


Figure 4.39: The model Accuracy

The recall curve reveals the YOLO model's outstanding ability to detect a high proportion of true positive objects while minimizing false negatives. It demonstrates that the model successfully captures a large number of objects in the dataset. The curve exhibits consistently high recall values across various confidence thresholds or IoU thresholds, underscoring the YOLO model's exceptional performance in capturing a vast majority of lession of DR.

- **Precision-Recall Curve**

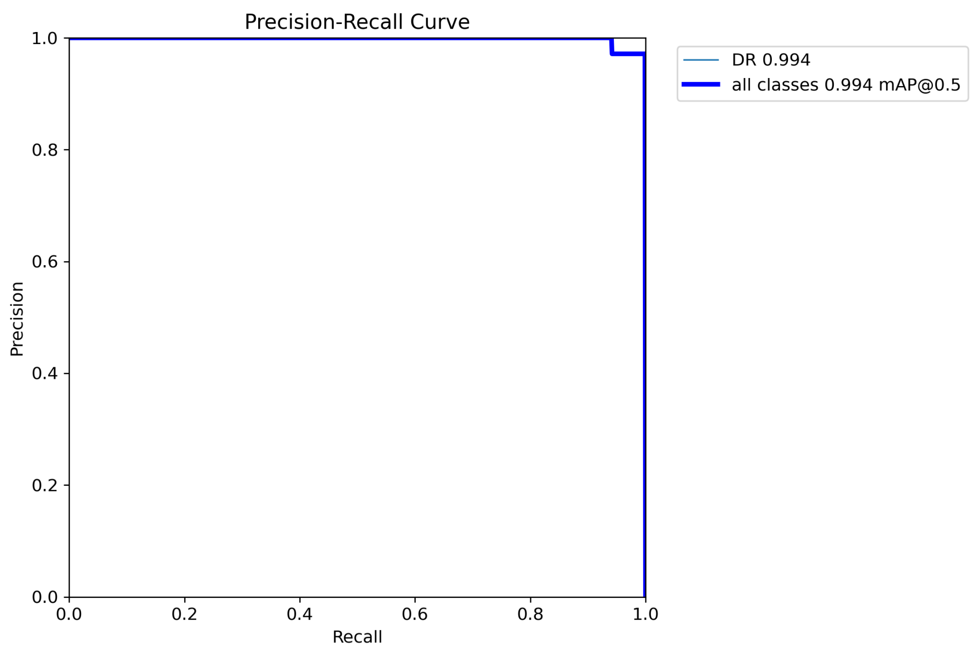


Figure 4.40: The model Accuracy

Looking at the precision-recall curve, we can observe that the precision remains consistently high as the recall increases. This signifies that the YOLO model achieves precise and reliable DR detection across a range of confidence thresholds.

- **F1 Curve**

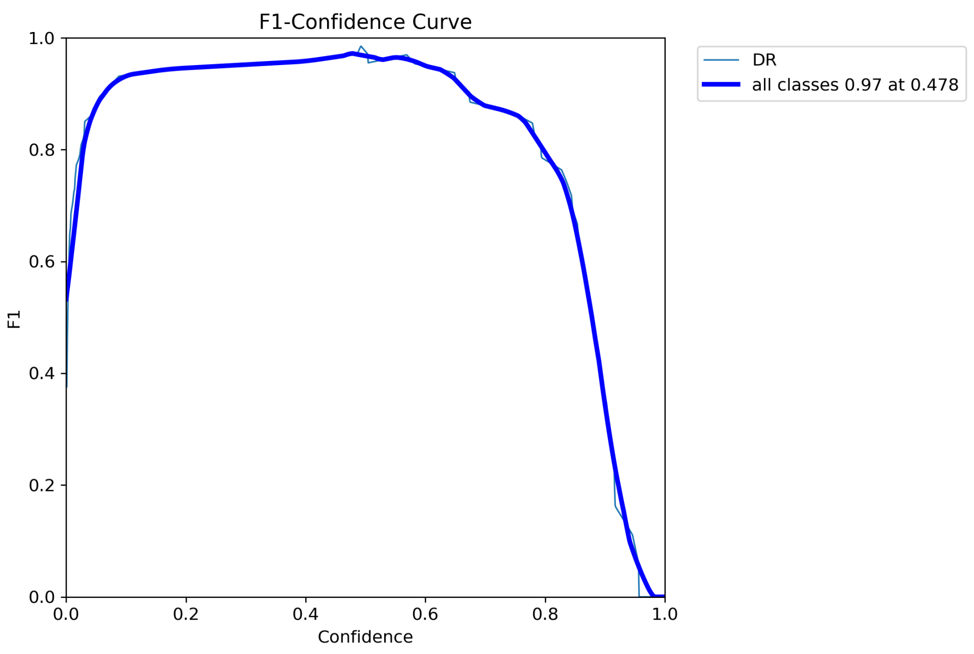


Figure 4.41: The model Accuracy

The F1 confidence curve for YOLO detection indicates that our model performs accurately across all classes. At a confidence threshold of 0.47, the model achieves an F1 score of 0.97. This high F1 score suggests a good balance between precision and recall, resulting in reliable and robust detections.

- **Confusion matrix**

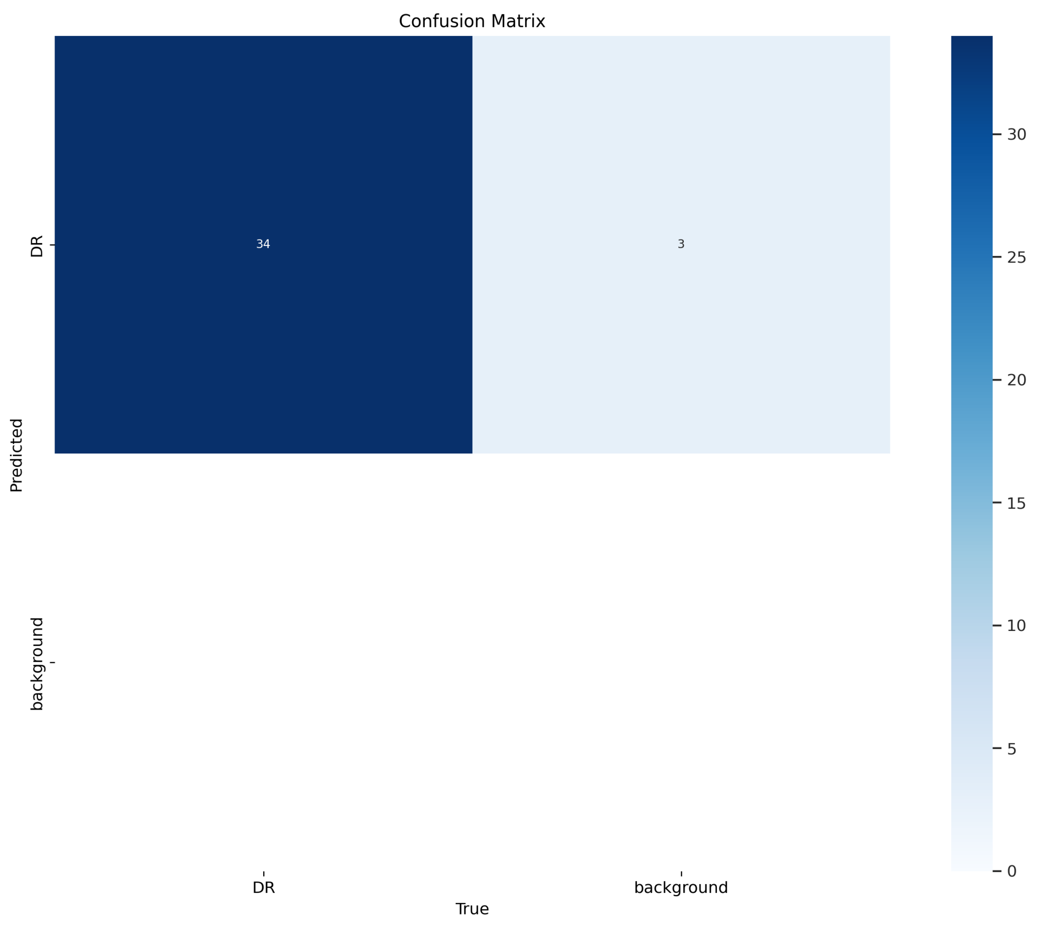


Figure 4.42: Confusion matrix

- The top-left element, 34, represents the number of instances that belong to the "DR" class and are correctly detected as "DR."
- The top-right element, 3, indicates that there are instances of the background that are incorrectly detected as "DR"

3. **Save the model**

   After training the model will be saved automatically to runs/detect/train/weights.we will found two models first named **best.pt** and the second one named **last.pt**

   The main difference is that **"best.pt"** captures the model's best performance, while **"last.pt"** represents the final state of the model after completing all training epochs.

Metric	Precision	Recall	F1 Score
DR	100%	100%	97%

Table 4.18: YOLO Evaluation Metrics

4.8 Comparison of our work with previous works

Model	Accuracy	Validation accuracy	Test Accuracy	Number of classes
Our DenseNet-121 (first structure)	99.87%	100%	100%	2
Our AlexNet (first structure)	92.91%	89.07%	85.91	2
Sebti riad binary[50]	96.93%	95.08%	95.65	2
Fallah kawther binary[35]	98.75%	95.35%	96.02%	2
Esfahan [58]	85%	86%	-	2
Our DenseNet-121 (second structure)	99.87%	97.53%	97.05	5
Our AlexNet (second structure)	92%	82.14%	82.93%	4
Sebti riad [50]	93.06%	81.12%	80,48%	3
Fallah kawther [35]	96.73%	93.45%	93,43%	4

Table 4.19: Table of comparison

Table 4.19 provides a comparison of different models, including our proposed models, with other works in the field of diabetic retinopathy detection. The table presents key metrics such as accuracy, validation accuracy, test accuracy, and the number of classes.

Our DenseNet-121 (first structure) achieved an accuracy of 99.87%, with both the validation accuracy and test accuracy reaching 100%. This indicates the excellent performance of the model in accurately classifying diabetic retinopathy images into two classes.

In comparison, our AlexNet (first structure) achieved a slightly lower accuracy of 92.91%, with a validation accuracy of 89.07% and a test accuracy of 85.91%. Despite the lower accuracy compared to DenseNet-121, AlexNet still demonstrates strong performance in classifying diabetic retinopathy images.

We also compared our models with two other works: Sebti riad binary [50] and Fallah kawther binary [35]. Sebti riad binary achieved an accuracy of 96.93% with a validation accuracy of 95.08% and a test accuracy of 95.65%. Fallah kawther binary achieved an accuracy of 98.75% with a validation accuracy of 95.35% and a test accuracy of 96.02%. Both of these works focus on the binary classification of diabetic retinopathy images.

Additionally, we included Esfahan [58], which achieved an accuracy of 85% and a validation accuracy of 86%. However, the test accuracy value is not provided.

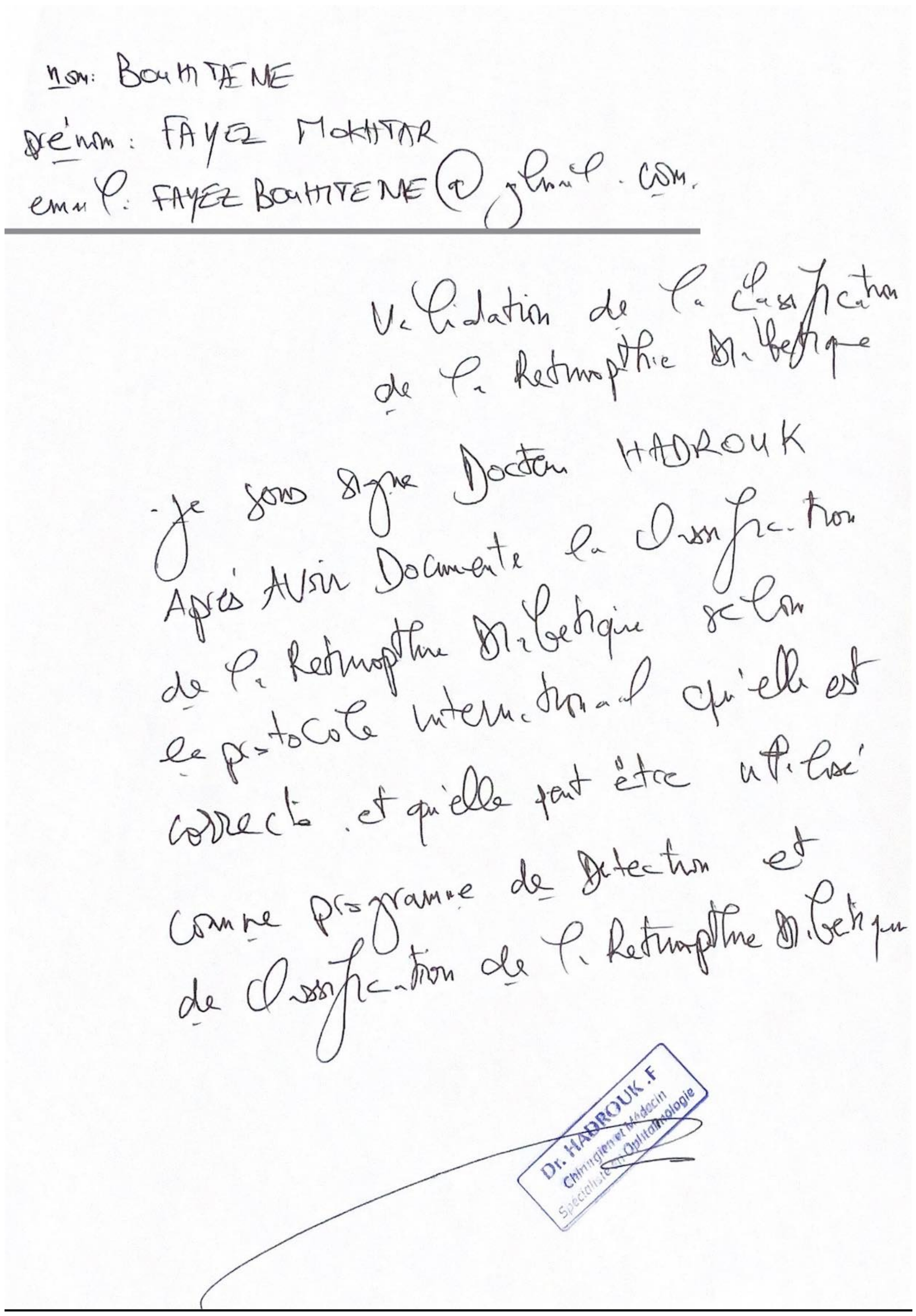
Moving to our DenseNet-121 (second structure), which can classify images into five classes, we obtained an accuracy of 99.87%, a validation accuracy of 97.53%, and a test accuracy of 97.05%. This highlights the effectiveness of the model in achieving high accuracy even with a larger number of classes.

Similarly, our AlexNet (second structure) achieved an accuracy of 92%, a validation accuracy of 82.14%, and a test accuracy of 82.93% when classifying images into four classes.

Furthermore, we compared our models with Sebti riad [50] and Fallah kawther [35], which focused on three and four classes, respectively. Sebti riad achieved an accuracy of 93.06% with a validation accuracy of 81.12%, while Fallah kawther achieved an accuracy of 96.73% with a validation accuracy of 93.45%.

Finally, our proposed models, particularly DenseNet-121, demonstrated superior performance in terms of accuracy and validation accuracy compared to other works. The ability to achieve high accuracy in classifying diabetic retinopathy images showcases the effectiveness of our models in contributing to the field of diabetic retinopathy detection.

Our work examined and approved by ophthalmologist in Biskra Hospital of Ophthalmologist. The following figure 4.43 illustrates the approval of our work.



nom: BOUHTAENE
prénom: FAYEZ MOKHTAR
email: FAYEZBOUHTTENE@gmail.com

Validation de la classification
de la Retinopathie Diabetique

Je sous signé Docteur HADROUK
Après avoir Documenté la classification
de la Retinopathie Diabetique selon
le protocole international qu'elle est
correcte et qu'elle peut être utilisé
comme programme de Detection et
de classification de la Retinopathie Diabetique

Dr. HADROUK.F
Chirurgien Médecin
Spécialiste en Ophtalmologie

Figure 4.43: Certificate of ophthalmologist

4.9 Conclusion

In this chapter, we described the many parameters for each structure, worked with various datasets, described all experiments, and presented the findings with plots and the confusion matrix for the test phase. The excellent results we have gotten motivate us to improve the model design.

Conclusion and Perspectives

Deep learning proved again its performance in detection and classification, especially in health-care, and the best proof is our work, where the traditional method for detection of DR is still consuming time, challenging, and it is more expensive.

In this research, we have suggested a variation of deep learning architecteurs to classify and detect DR, in fact, we have looked at a number of related works that have been put out to deal with image classification to diagnose DR. We correct some falsely proposed structures in the previous works and used different architectures to improve the existing results. We used AlexNet and DenseNet-121 to classify DR into 2 classes (No DR,DR) and 5 classes (No DR,Mild,Moderate,Severe, Proliferate), and the Yolov8 model to detect DR with bounding boxes. All used models were compared with different models like CNN and Resnet, where the best accuracy among all was obtained by the used model.

The result showed that DenseNet-121 and Yolov8 can be used in medical diagnosis, where they showed excellent results 99% in 2 classes and 93% in 5 classes.

In the near future,we will develop the data by increasing the number of images to get it bigger and bigger.We also, intend to work on other kinds of diseases especially diseases diagnosed by x rays or Mri.

References

[1] Cat and dog classification image. Online image. Retrieved from https://machine-learning-and-data-science-with-python.readthedocs.io/en/latest/_images/catdogcl.png.

[2] Calculate the output size of a convolutional layer. URL https://www.baeldung.com/cs/convolutional-layer-size. Accessed: Month Day, Year.

[3] What is diabetes? URL https://www.cdc.gov/diabetes/basics/diabetes.html. Accessed: Month Day, Year.

[4] Elsevier journal 2023 Medical Image Analysis. URL https://www.elsevier.com/journals/medical-image-analysis/1361-8415/guide-for-authors. Accessed: Month Day, Year.

[5] Healthcare. https://medical-dictionary.thefreedictionary.com/health+care, . Accessed: May 29, 2023.

[6] healthcare df3, . URL https://dictionary.cambridge.org/dictionary/english/healthcare. Accessed: Month Day, Year.

[7] healthcare df4, . URL https://www.merriam-webster.com/dictionary/health%20care. Accessed: Month Day, Year.

[8] Healthcare. https://www.oxfordlearnersdictionaries.com/definition/american_english/healthcare, . Accessed: May 29, 2023.

[9] Image title. Online image, . Retrieved from https://i.ibb.co/QYtzq56/image.png.

[10] Online image, . Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcTRYanUujVVUXIAi3luWDl-8GiP-FCV-ZJCje62hNSCGk_bjXRW.

[11] Online image, . Retrieved from https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcT1-Pal0hvVAZFfpJKDl_HtHJsQL_pNWAecaz0tUNJ1F_tta5tg.

[12] Online image, . Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRdSGG1z8y31eewjuUA-RmcCojxgRUDmTcQWU7SD1q_0fxNZgN_.

[13] Online image, . Retrieved from https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcQ9NPxmJK7uwjIWuFHvKuv4KSk0nAIDvzJymcPRUzA5KukTVaP9.

[14] Online image, . Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRq-z2xlsVPvQ2Qg4fcYG0ZQU0uBfs7Ub7YGAknvf9Ua6irH2RY.

[15] Online image, . Retrieved from https://encrypted-tbn3.gstatic.com/images?q=tbn:ANd9GcRD5y7I3sa939NnXsbTG6N_sC47oMjqz0kXaAvfzhVlb1fhBnJK.

[16] Online image, . Retrieved from https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcSzckr0hcWhrznwuTLx4I7mqFzfNy6vYQPaUJKL4q5roB0aC7Up.

[17] Online image, . Retrieved from https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcRGXSu1d9R5oJydSowkBX8BQ5Tt1u640j97VNwnT6CA8abpYF9-.

[18] Online image, . Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcRd8Xk_pAvtxJi7RPlT6zjXTWpagsudFJrqOP8sobAxGrZAzcIE.

[19] Online image, . Retrieved from https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS9S6FU12GJb6GU6ODw1oH9qvZRif2o7o9dlr2yN0nddn0H7tHA.

[20] Online image, . Retrieved from https://encrypted-tbn1.gstatic.com/images?q=tbn:ANd9GcSifRMb3PnGxuK9aX97BbK8qnZPhOA1JXQNA5YRxElu8JFPHsWw.

[21] Online image, . Retrieved from https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcTXsvRgA8iB2VMMTJSm8iIp2e8_MpWIfwUOPTJwjX-Unm1uKG6h.

[22] React documentation. https://react.dev/learn. Retrieved from the React website.

[23] The relation between artificial and biological neuron? URL https://smhatre59.medium.com/what-is-the-relation-between-artificialand-biological-neuron-18b05831036. Accessed: Month Day, Year.

[24] Resnet-50: The basics and a quick tutorial. URL https://datagen.tech/guides/computer-vision/resnet-50/. Accessed: Month Day, Year.

[25] Md Zahangir Alom, Tarek M. Taha, Christopher Yakopcic, Stefan Westberg, Paheding Sidike, Mst Shamima Nasrin, Brian C Van Esesn, Abdul A S. Awwal, and Vijayan K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches. *arXiv preprint arXiv:1803.01164*, 2018. Submitted on 3 Mar 2018 (v1), last revised 12 Sep 2018 (this version, v2).

[26] Apple Inc. Apple watch: Researchers explore new frontiers in heart health. Available at: https://www.apple.com/fr/newsroom/2023/02/with-apple-watch-researchers-explore-new-frontiers-in-heart-health/, 2023. Accessed: June 18, 2023.

[27] Appyhigh Technology Blog. Convolutional neural networks: A brief history of their evolution. URL https://medium.com/appyhigh-technology-blog/convolutional-neural-networks-a-brief-history-of-their-evolution-ee3405568597. Accessed: Month Day, Year.

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

[29] Mohamed H. Bakr and Mohamed H. Negm. Advances in imaging and electron physics. 2012.

[30] Weatherbit Blog. [title of the specific blog post].

[31] NumPy Community. Numpy documentation. https://numpy.org/devdocs/. Accessed: <accessed date>.

[32] deep AI. What is deep learning? URL https://deepai.org/machinelearningglossary-and-terms/deep-learning. Accessed: Month Day, Year.

[33] Depositphotos. Doctor interview. Available at: https://depositphotos.com/vector-images/doctor-interview.html, Year. Accessed: June 18, 2023.

[34] A. M. Egan and S. F. Dinneen. What is diabetes? *Medicine*, 47(1):1–4, 2019.

[35] Falah et al. Deep learning approach for early diabetic retinopathy diagnosis. *Unknown*, 2022.

[36] Python Software Foundation. Guido van rossum: The man behind python. https://www.python.org/doc/essays/blurb/. Retrieved from the Python.org website.

[37] Freepik. Cellule nerveuse.

[38] Google. Google colaboratory documentation. https://colab.research.google.com/notebooks/intro.ipynb. Accessed: <accessed date>.

[39] Gao Huang et al. Densely connected convolutional networks. 2018.

[40] F. et al. Jiang. Artificial intelligence in healthcare: past, present and future. *Stroke and Vascular Neurology*, svn-2017-000101, 2017. Accessed: Month Day, Year.

[41] Kaggle. Kaggle notebooks documentation. https://www.kaggle.com/docs/notebooks. Accessed: <accessed date>.

[42] Levity AI. Difference between machine learning and deep learning. Available at: https://levity.ai/blog/difference-machine-learning-deep-learning, Year. Accessed: June 18, 2023.

[43] W. H. Lopez Pinaya, S. Vieira, R. Garcia-Dias, and A. Mechelli. Convolutional neural networks. *Machine Learning*, pages 173–191, 2020. doi: 10.1016/b978-0-12-815739-8.00010-9.

[44] Author's Name. Usual machine learning vs transfer learning. Online image, 2022. Retrieved from https://data-science-blog.com/wp-content/uploads/2022/04/usual-machine-learning-vs-transfer-learning.png.

[45] National Eye Institute. Diabetic retinopathy. URL https://www.nei.nih.gov/learn-about-eye-health/eye-conditions-and-diseases/diabeticretinopathy. Accessed: Month Day, Year.

[46] OpenCV. Opencv documentation. https://docs.opencv.org/4.x/d1/dfb/intro.html. Accessed: <accessed date>.

[47] Ouidou. Indexation d'images par le deep learning dans le big data, Date of article. URL https://blog.ouidou.fr/indexation-dimages-par-le-deep-learning-dans-le-big-data-292d3d7a019e?gi=6c5e8e10ba02.

[48] Sebastián Ramírez. Fastapi documentation. https://fastapi.tiangolo.com/. Accessed: <accessed date>.

[49] David A. Salz and Andre J. Witkin. Imaging in diabetic retinopathy. *Middle East Afr J Ophthalmol*, 22(2):145–150, 2015. doi: 10.4103/0974-9233.151887.

[50] M.R. Sebti et al. A deep learning approach for the diabetic retinopathy detection. In *The Sixth Smart City Applications International Conference*, 2021.

[51] S.H. Shabbeer Basha et al. Impact of fully connected layers on performance of convolutional neural networks for image classification. *Neurocomputing*, 2020. ISSN 0925-2312.

[52] Hoo-Chang Shin et al. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35(5):1285–1298, 2016. doi: 10.1109/TMI.2016.2528162.

[53] Nitish Srivastava et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[54] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. 2020.

[55] Keras Team. Keras documentation. https://keras.io/, . Accessed: <accessed date>.

[56] Matplotlib Development Team. Matplotlib documentation. https://matplotlib.org/, . Accessed: <accessed date>.

[57] TensorFlow. Tensorflow documentation. https://www.tensorflow.org/learn. Accessed: <accessed date>.

[58] Mehdi Torabian Esfahani, Mahsa Ghaderi, and RJLEJPT Kafiyeh. Classification of diabetic and normal fundus images using new deep learning method. *Leonardo Electron. J. Pract. Technol*, 17(32):233–248, 2018.

[59] Ultralytics. ultralytics/ultralytics.

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

[61] W. L. Yun, U. Rajendra Acharya, Y. V. Venkatesh, C. Chee, L. C. Min, and E. Y. K. Ng. Identification of different stages of diabetic retinopathy using retinal optical images. *Information Sciences*, 178(1):106–121, 2008. doi: 10.1016/j.ins.2007.07.020.