DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE and ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade, Shavige Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078.

A Project Report On

"Diabetic Retinopathy Detection Using Convolutional Neural Networks"

Submitted By

Kaustavjit Saha	**- 1DS18CS168**
Kumar Gaurav	**- 1DS18CS170**
Nikhil Raj	**- 1DS18CS172**
Susheel Kumar	**- 1DS18CS126**

[8th Semester, B.E.(CSE)]

Under The Guidance Of

Dr. Mohammed Tajuddin
Professor Dept. of CSE
DSCE, Bangalore

Department of Computer Science and Engineering
Dayananda Sagar College of Engineering
Bangalore-78

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

DAYANANDA SAGAR COLLEGE OF ENGINEERING

Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore - 560078
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the project entitled **Diabetic Retinopathy Detection Using Convolutional Neural Networks** is a bonafide work carried out by **Kaustavjit Saha [1DS18CS168]**, **Kumar Gaurav [1DS18CS170]**, **Nikhil Raj [1DS18CS172]**, **Susheel Kumar [1DS18CS126]** in partial fulfilment of 8th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2021-22.

Dr. Mohammed. Tajuddin (Internal Guide) Professor, Dept. of CSE, DSCE	**Dr. Ramesh Babu** Vice Principal & HOD CSE, DSCE	**Dr. C P S Prakash** Principal, DSCE
Signature:..................	Signature:..................	Signature:..................

Name of the Examiners:	Signature with date:
1....................................	....
2..................................	....

Abstract

Diabetes is a typical term that we go over in this day and age. It is something normal in pretty much every family. Diabetes is a chronic metabolic disease characterised by elevated blood glucose (or glucose) levels. Over time, diabetes causes major damage to the heart, veins, eyes, kidneys, and nerves. The most well-known type of diabetes is type 2, which often affects adults, and is brought on when the body stops responding to insulin or produces insufficient amounts of it.

Diabetes has a problem called diabetic retinopathy, which affects the eyes. Damage to the veins of the light-sensitive tissue at the back of the eye is what causes it (retina). Diabetic retinopathy may initially have no negative effects or very mild vision problems. Since the sickness is a dynamic cycle, clinical specialists recommend that diabetic patients should be distinguished at least two times every year to analyze indications of disease opportune. In the ongoing clinical analysis, the location mostly depends on the ophthalmologist looking at the variety fundus picture and afterward assesses the patient's condition. This location is burdensome and tedious, which brings about more mistakes.

Profound learning based Convolutional Neural Network (CNN) has as of late been demonstrated a promising methodology in biomedical examination. In this work, delegate Diabetic Retinopathy (DR) pictures have been collected into five classes as per the ability of ophthalmologist. For DR stage order, a collection of Deep Convolutional Neural Network techniques has been used. We are proposing a technique to execute a programmed determination of DR utilizing fundus picture order.

Acknowledgement

We have taken efforts in this project on **Diabetic Retinopathy Detection using Convolutional Neural Networks**. However, it would not have been possible without the kind support and help of many individuals and organizations. We would like to extend my sincere thanks to all of them.

We would like to take this opportunity to express our gratitude to **Dr. C P S Prakash**, Principal of DSCE, for permitting us to utilize all the necessary facilities of the institution.

We are also very grateful to our respected Vice Principal, HOD of Computer Science Engineering, DSCE, Bangalore, **Dr. Ramesh Babu D R**, for his support and encouragement.

We are immensely grateful to our respected and learned guide, **Dr. Mohammed Tajuddin,** Professor, CSE, DSCE for their valuable help and guidance. We are extremely thankful to them for all the encouragement and guidance they have given us during every stage of the project.

We would like to thank our project coordinators **Dr. Vindhya Malagi**, Professor, CSE, DSCE and Dr. Ramya R.S, Associate Professor, CSE, DSCE for their guidance and support.

We are also thankful to all the other faculty and staff members of our department for their kind co-operation and help.

Lastly, Our thanks and appreciations also go to my classmates and friends in developing the project and people who have willingly helped me out with their abilities.

Kaustavjit Saha[1DS18CS168]
Kumar Gaurav[1DS18CS170]
Nikhil Raj[1DS18CS172]
Susheel Kumar[1DS18CS126]

Contents

List of Figures

1
Introduction

1.1 Diabetic Retinopathy

The condition known as diabetic retinopathy, also known as diabetic eye disease (DED), is one in which diabetes mellitus causes damage to the retina. In developed countries, it is the leading cause of vision impairment. Diabetic retinopathy influences up to 80 percent of the individuals who have had both sort 1 and type 2 diabetes for a very long time or more. In something like 90% of new cases, movement to additional forceful types of sight undermining retinopathy and maculopathy could be diminished with appropriate treatment and checking of the eyes. The more drawn out an individual has diabetes, the higher their possibilities creating diabetic retinopathy.

There are typically no early warning signals of diabetic retinopathy. Even macular edoema, which can lead to quick focusing vision problems, may not have any advance notification or completed documentation for a considerable amount of time. However, a person with macular edoema is typically going to have distorted vision, making it challenging to read or drive. The vision may occasionally get better during the day or get worse. There are no negative effects in the primary stage, also known as non-proliferative diabetic retinopathy (NPDR). Despite having 20/20 eyesight, the patients could miss the warnings. The best way to diagnose NPDR is by performing a fundus assessment with an immediate or circuitous ophthalmoscope by a qualified ophthalmologist or optometrist. Fundus photography can be used to accurately record the fundus findings, and microaneurysms (tiny blood-filled swells in the corridor walls) should be apparent.Assuming there is diminished vision, fluorescein angiography can show limiting or hindered retinal veins obviously (absence of blood stream or retinal ischemia).

The result of damage to the retina's tiny veins and neurons is diabetic retinopathy. The earliest changes causing diabetic retinopathy include constriction of the retinal veins associated with decreased retinal blood flow; breakdown of the internal retinal neurons, followed later by modifications to the functionality of the external retina associated with unpretentious changes in visual ability. Afterward, the cellar film of the retinal veins thickens, vessels degenerate and lose cells, especially pericytes and vascular smooth muscle cells. This results in blood stream loss, mild ischemia, tiny aneurysms that appear as inflatable-like designs stretching away from the thin walls and target fiery cells, as well as severe bleeding and degeneration of the retina's neurons and glial cells. Typically, the illness develops 10 to 15 years after the diagnosis of diabetes mellitus.

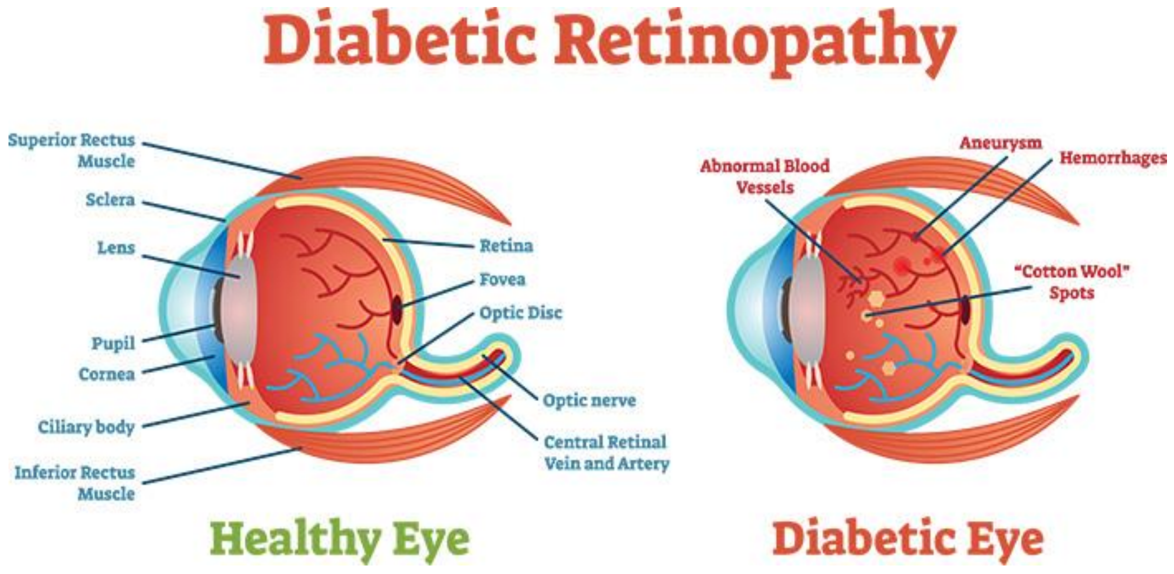


Figure 1.1.1: Difference between a healthy eye and Diabetic Eye

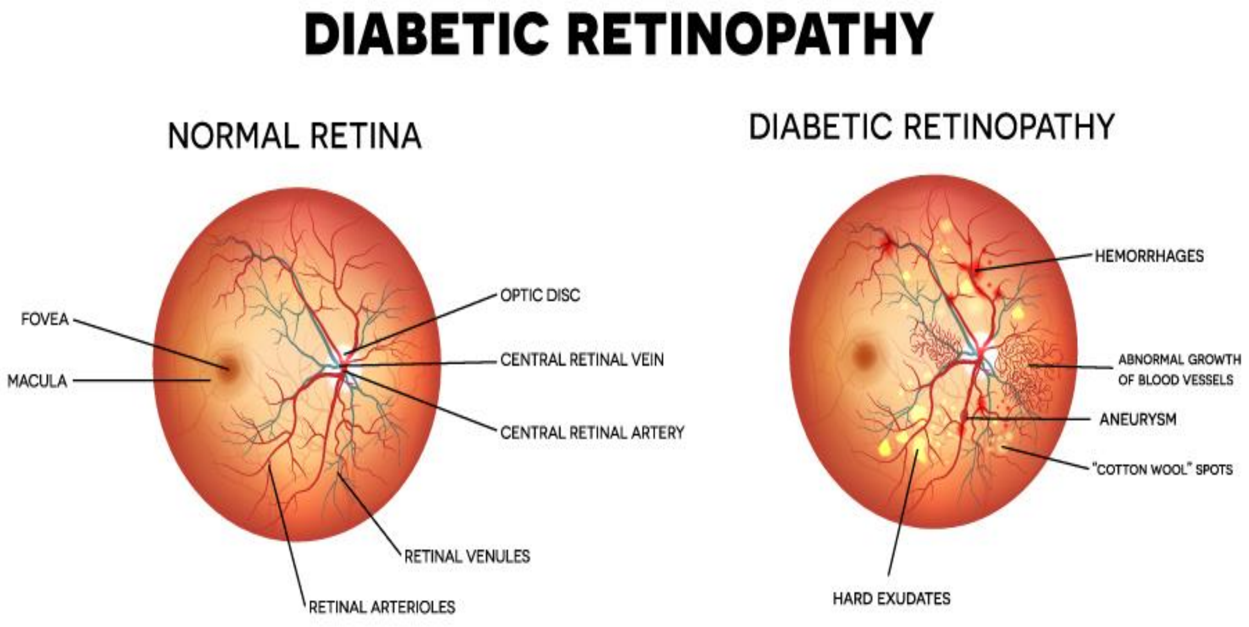


Figure 1.1.2: Retinal image of a healthy eye and a Diabetic Eye

1.2 Deep Learning

Individuals gain for a fact. The more extravagant our encounters, the more we can learn. In the man-made consciousness (AI) discipline known as profound learning, the equivalent can be said for machines controlled by AI equipment and programming. The encounters through which machines can learn are characterized by the information they secure, and the amount and nature of information decide the amount they can learn. AI, which is essentially a brain network with at least three layers, is a subset of profound learning. These neural networks make an effort to behave like the human brain. While a neural network with a single layer can still produce accurate predictions, additional hidden layers help to streamline and improve your accuracy.

Some of the information pre-handling that is typically associated with AI is removed by profound learning. These equations can process unstructured data like text and images and automate the highlight extraction process, reducing the need for human experts by a certain amount. Consider the situation if we required to sort a collection of pictures of different pets into categories like "feline," "canine," "hamster," etc. Deep learning algorithms can determine which features, like ears, are often essential for differentiating one creature from another.

Profound learning networks advance by finding many-sided structures in the information they experience. By building computational models that are made out of various handling layers, the organizations can make different degrees of deliberation to address the information. For instance, a profound learning model known as a convolutional brain organization can be prepared utilizing enormous numbers (as in great many) pictures, for example, those containing felines. This sort of brain network normally gains from the pixels contained in the pictures it gets. It can characterize gatherings of pixels that are illustrative of a feline's elements, with gatherings of highlights like paws, ears, and eyes showing the presence of a feline in a picture.

We are living in a period of uncommon open door, and profound learning innovation can assist us with accomplishing new leap forwards. Profound learning has been instrumental in the revelation of exoplanets and novel medications and the recognition of illnesses and subatomic particles. It is generally enlarging how we might interpret science, including genomics, proteomics, metabolomics, the immunome, the tip of the iceberg, to be precise.

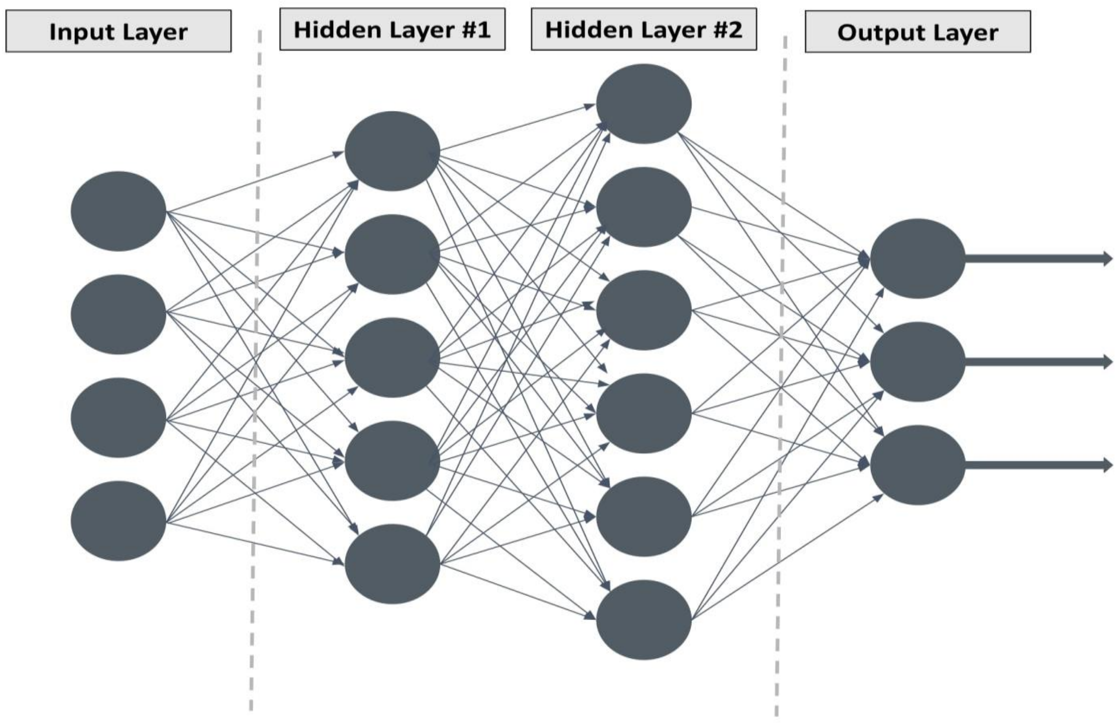


Figure 1.2.1: Neural Network Layers

1.2.1 CNN

A convolutional brain organisation (CNN) is a type of fictitious neural network used in image recognition and handling that is specifically designed to handle pixel input.The CNN has many layers some of them are the conventional layers and large numbers of them are the sub-examining layers which is at some point likewise have a layers which are associated start to finish. Multi-facet perceptrons are modified into CNNs. Multi-facet perceptrons typically refer to entirely related networks, in which each neuron in one layer is connected to every neuron in the layer above it. The conventional layer present any measure of information to figure out the superfluous ones and I is the little layer of the information or the characters. These information will shift in light of the info and different factors so that square which is measures will have an alternate examining of yi+1 or some other by variable of 1.

Natural cycles stimulated convolutional networks because the connectivity pattern between neurons resembles that of the organism visual brain. Only a small portion of the visual field, known as the open field, is where individual cortical neurons respond to upgrades. To the extent that they overlap, the receptive fields of different neurons cover the whole visual field. Comparative to other picture order calculations, CNNs typically employ very little pre-handling. According to this, whilst in traditional calculations these channels are hand-designed, the organisation learns how to develop the channels (or portions) through mechanical learning. A key advantage is the freedom from prior knowledge and human intervention in highlight extraction.

A convolutional brain network comprises of an information layer, stowed away layers and a result layer. In any feed-forward brain organization, any center layers are called stowed away on the grounds that their bits of feedbacks and results are covered by the enactment capability and last convolution. In a convolutional brain organization, the secret layers incorporate layers that perform convolutions. Normally this incorporates a layer that plays out a speck result of the convolution bit with the layer's feedback network. As the convolution piece slides along the information framework for the layer, the convolution activity creates a component map, which thus adds to the contribution of the following layer. Following this are several layers, such as standardisation, pooling, and entirely associated layers.

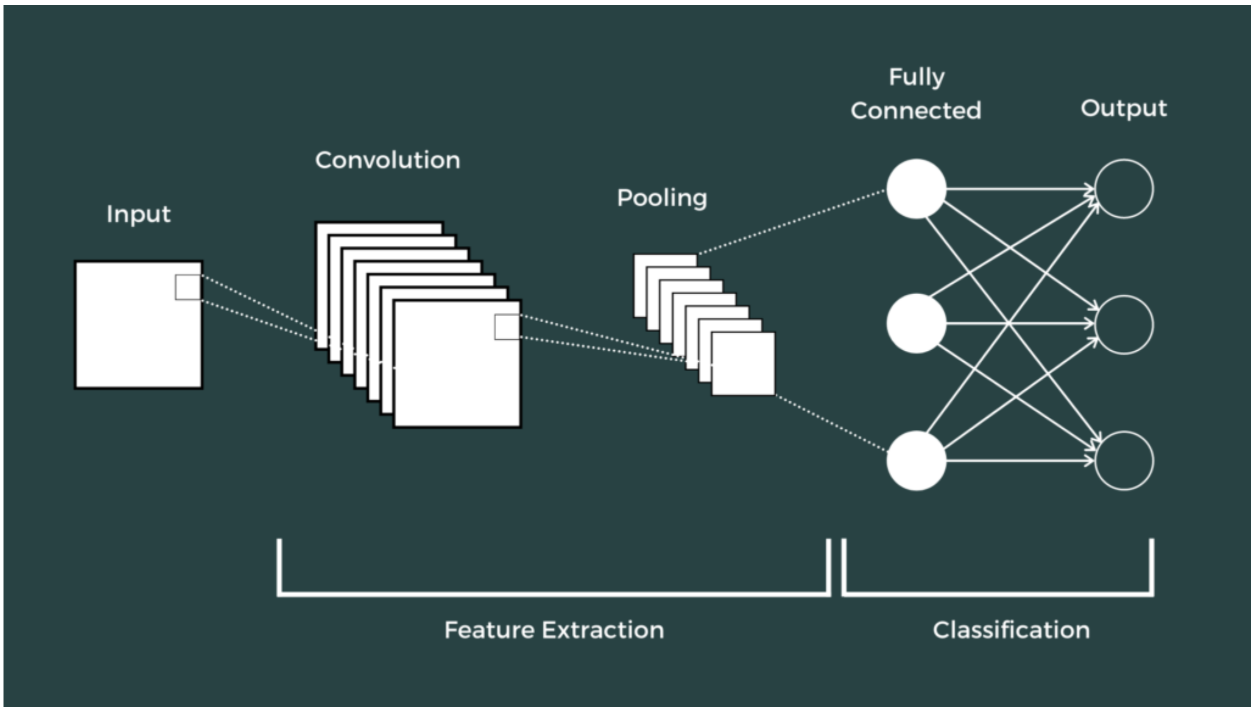


Figure 1.2.2: CNN Design

1.3 Real World-Application

The ongoing clinical finding mostly depends on the ophthalmologist's evaluation of the patient's condition after examining various fundus images. This location is difficult and tiresome, which results in greater error. In addition, many patients with DR cannot be properly assessed and treated due to the large number of diabetic patients and the lack of clinical resources in some areas. As a result, they miss out on the best treatment opportunities, which can eventually result in irreversible visual loss and even visual impairment.The various applications are:-

1. It is of immense use in the medical field as it will enable the doctors to detect the ailment at an early stage.

2. The use of automated method to treat Diabetic Retinopathy will increase the accuracy of the disease getting treated.

3. Diabetic Retinopathy programmed order of fundus pictures can actually help specialists in Diabetic Retinopathy determination, which can work on the analytic effectiveness.

4. Rather than the specialists' manual activity with time, it alleviates their tension on the finding for DR in a programmed and high-exactness way.

5. A regular checkup for the patients can be organized as now the process has become time efficient as well as more accurate.

1.4 Organisation of the Project Report

The project report is organized as given below:

In Chapter (2), we examine about the issue proclamation and our answer for the issue. A similar part likewise manages the other existing innovation. The Chapter that follows for example part (3) comprises of the subtleties on the writing overview of the papers to the issue explanation and the proposed arrangement. In Chapter (4), we present the System Outline and others framework as Data stream chart and the grouping graph. The following section, part (5) gives the necessities and confines about the execution of the proposed framework. Part 6 arrangements with the testing of the item and their ideal outcomes. In Chapter (7), we examine about the affecting boundaries and their impact on the framework. A similar section manages laying out the ideal boundaries for the framework. The part (8) closes the paper alongside notice of the Future Enhancements. Part (9) is insights concerning the references made during the improvement of the framework.

2

Problem Statement and Proposed Solution

2.1 Problem Statement

To develop a model that efficiently detects Diabetic Retinopathy in a set of fundus images and classifies the severity of damage to eye, thus, providing credible insights to the severity of diabetes.

2.2 Existing Systems

2.2.1 Screening of Diabetic Retinopathy

Because of its high pivotal goal and broad retinal output inclusion, optical cognizance tomography (OCT) imaging is the best technique for identifying DR. Spatial space optical rationality tomography (SD-OCT) uncovered DR-related changes in the retinal color epithelium layer, Bruch's layer, outer confining film, and photoreceptors. Furthermore, SD-OCT identified related epiretinal layers, changes in retinal degeneration, and the vitreoretinal interface. The difficulties of programmed arrangement of SD-OCT information for consequently recognizing patients with DME from typical subjects are additionally tended to. The recommended technique utilizes Local Binary Pattern (LBP) highlights to portray the surface of OCT pictures and contrasts a few LBP include extraction strategies and the strategy for involving a solitary mark for the full OCT volume. As per test results utilizing two datasets of freely 32 and 30 OCT volumes, highlights got from LBP surface have a huge level of discriminative potential, whether or not low or high state portrayals are utilized. The utilization of noncalibrated, numerous viewpoint fundus pictures to investigate the macula's broadening is a one of a kind methodology that is encouraged. This headway makes it simpler for far off ophthalmologists to distinguish and evaluate expanding zones. A sound system fundus camera is inclined to incorrectness and can't be utilized to get to this capacity Additionally, provide a programmed method for handling image retouching measurements that are useful for the POC robotized finish of early macular edoema, such as before the presence of exudation The suggested method is broken down into three stages: first, a preprocessing framework balances the image while simultaneously focusing on the faint microstructures and the capability of the macula; second, all available points of view are enrolled using non-morphological scant elements; and third, a thick pyramidal optical stream is registered for each image and then genuinely combined to create a nave level guide of the macula. Three setups of counterfeit photographs and two courses of action of true pictures are utilized to show the outcomes. These training tests show the capacity to decide a base expanding of 300 m and to interface the entertainment to the enlarged region.

2.2.2 Retinal Imagining Method

Versatile optics is a procedure for working on optical frameworks (AO). It is finished by decreasing the front wave impact twisting. Through AO retinal imaging, the dividing, mosaic, and photography cell thickness of sound and infected eyes might be thought about. The incendiary issues are followed for the back piece of fundus photography utilizing retinal imaging. The two techniques for retinal imaging that are most often used are fundus fluorescein angiography (FA) and OCT. For the right determination of the back post, these imaging modalities are helpful in the treatment of patients with fiery problems . When contrasted with fundus photos, FA is a nosy technique that utilizes fluorescein tone, while OCT is a costly imaging innovation. Furthermore, DR can be identified consequently utilizing fundus pictures. For telescreening, the DR reviewing strategy is utilized. Thus, in contrast with biomicroscopy, OCT, and FA modalities, the DR evaluating framework applied to fundus imaging modalities is a dependable and reasonable device. In contrast to SD-OCT based en face thickness guide or pseudoshading images obtained with Optomap, en face standard or advanced multicolor filtering laser ophthalmoscopy (SLO) images obtained with Scientifica 5 HRA2 consider a superior comprehension of epiretinal films for preoperative evaluation. Infrared or FA images are simply hardly reasonable to depict epiretinal layers.

2.2.3 Diabetic Retinopathy Detection By Computer-Aided Diagnostic System (CAD)

Utilizing highlights including the area of EXs, MAs, veins, surface, HMs, hub focuses, and different elements, PC helped analysis plans to recognize DR and typical pictures . For the DR screening gadget, clinical parts, explicitly Osareh et Recursive district development (RRGT) and versatile power thresholding (AIT) were utilized to remove EXs from dull injuries, and dim sores were sectioned out utilizing the channel administrator .To free MAs, Quellec et al. utilized the best channels. For arrangement, the fake brain organization (ANN) and Bayesian blueprint work were utilized. Their strategies yielded an explicitness of 46.3% and a responsiveness of 95.1%. In the CAD system, support vector machine (SVM) part classifiers are utilized to decide if DR is available or not. The order issues in DR have been settled by the arranged CAD engineering. PC helped conclusion has turned into a fundamental part of the clinical area. A calculation was contrived considering the review of a motorized fundus photograph to distinguish DME, DR, and significant harm. This technique is a more proficient option in contrast to the tedious fundus photograph manual assessment for evaluating for DR. It will be feasible to save a great deal of time in the event that this system is controlled by the specialists, which will expand the time dispensed to a mass screening program. Logical data assumes a critical part in clinical imaging. Logical data in fundus picture modalities has been utilized to recognize and separate splendid sores. This relevant data is utilized to analyze coronary calcifications and hard EXs in CT filters. Significant level context oriented components are used to make sense of the setting in the spatial connection. Logical CAD system is a defeated loom when diverged from neighborhood CAD structure. A few creators have recommended CAD techniques for the discovery of various stages (NPDR and PDR) of DR that are experienced and portrayed in this part. Table 3 sums up these CAD techniques.

2.3 Proposed Solution

2.3.1 Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm that can process information from a picture of information, assign significance (learnable loads and predispositions) to various views and objects in the picture, and then choose whether to separate one from the other. Comparing the pre-handling expected in a ConvNet to other characterisation calculations, it is much lower. While in crude tactics channels are hand-designed, ConvNets can become accustomed to these channels/qualities with adequate practise.

A ConvNet's design was inspired by the association of the Visual Cortex and is comparable to the network example of Neurons in the Human Brain. Individual neurons only respond to stimulation in a small region of the visual field called the Receptive Field. The entire visual region is covered by a variety of these fields that crossover.

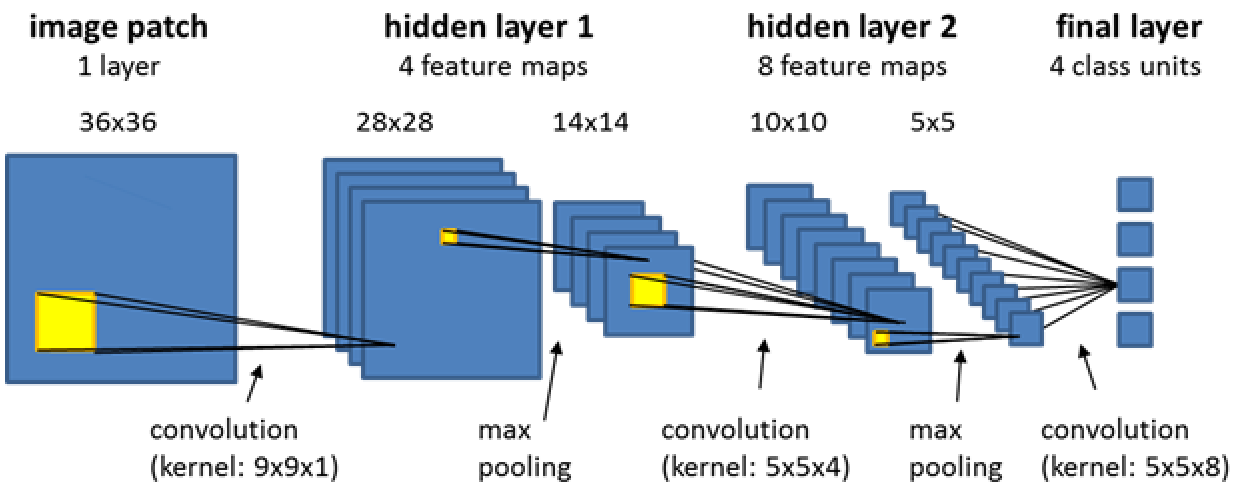


Figure 2.3.1: ConvNet Architecture

2.3.2 Residual Net (Residual Net)

Every successful solution that follows the primary CNN-based engineering (AlexNet) that won the ImageNet 2012 competition has more layers in a profound brain organisation to reduce the error rate. This works for fewer layers, but when the number of layers increases, a common problem in profound learning known as the Vanishing/Exploding slope arises. As a result, the angle is either zero or abnormally large. When a result, the preparation and testing error rate also rises as the number of layers is increased. This design put out the concept of Residual Blocks to address the evaporating/detonating angle problem. This company uses a technique called skip associations. By skipping a few levels in the middle, the skip association links the acts of one layer to others. This builds the last block. These remaining blocks are stacked to create ResNets. The organisational philosophy is that we let the organisation suit the remaining planning rather than layering on understanding the fundamental planning.

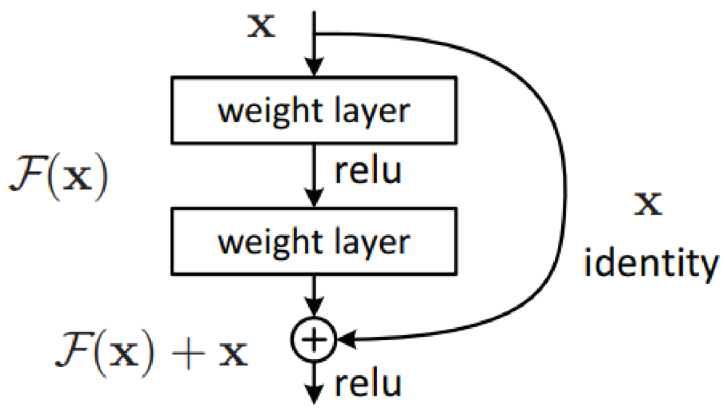

x
weight layer
relu
weight layer
$\mathcal{F}(\mathbf{x})$
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.3.2: Relu Activation Function

2.3.3 GoogleNet (InceptionV3)

A convolutional brain network of 48 layers deep is called Initiation v3. From the ImageNet data collection, you can build a pretrained version of the organisation trained on more than 1,000,000 images [1]. The organisation that has been trained can categorise images into 1000 different article classes, including console, mouse, pencil, and various animal types. The organisation has since discovered rich element depictions for many images. The company's photo input size is 299 by 299 pixels.Origin v3 is like and contains every one of the highlights of Inception V2 with following changes/increments:

- Utilization of RMSprop optimizer.
- Batch Normalization in the fully connected layer of Auxiliary classifier.
- Utilization of *7×7* factorized Convolution.
- Label Smoothing Regularization: This method involves analysing the effects of mark dropout when preparing in order to regularise the classifier. It prevents the classifier from predicting a class too quickly. The expansion of name smoothing improves the error rate by 0.2 percent.

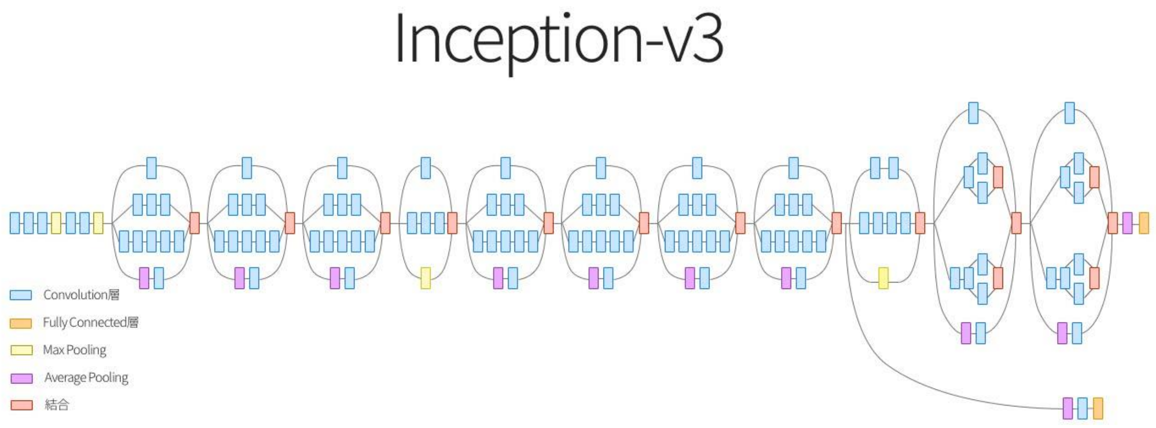
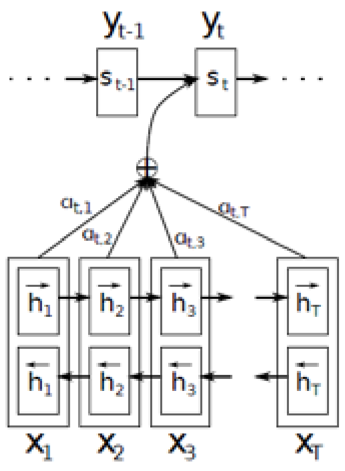


Figure 2.3.3: Inception-v3 Architecture

2.3.4 Improving GoogleNet Model with Attention Mechanism



y_{t-1}
y_t
s_{t-1}
s_t
$\alpha_{t,1}$
$\alpha_{t,2}$
$\alpha_{t,3}$
$\alpha_{t,T}$
$\overrightarrow{h}_1$ $\overrightarrow{h}_2$ $\overrightarrow{h}_3$ $\overrightarrow{h}_T$
$\overleftarrow{h}_1$ $\overleftarrow{h}_2$ $\overleftarrow{h}_3$ $\overleftarrow{h}_T$
X_1 X_2 X_3 X_T

Figure 2.3.4: Attention model

The Attention model is presented in Bahdanau's study with an outline similar to this. For each information sentence, the Bidirectional LSTM used here generates a series of comments (h1, h2,.....,hTx). Each of the vectors used in their research, h1, h2,.., and so on, is simply a link between the forward and reverse secret states in the encoder.

$$h_j = \left[\overrightarrow{h}_j^\top ; \overleftarrow{h}_j^\top\right]^\top$$

To put it simply, each of the vectors h1, h2, h3,., and hTx represents the Tx number of words in the information sentence. Only the last condition of the encoder LSTM (hTx in this case) was used as the setting vector in the simple encoder and decoder paradigm.

However, when creating the setting vector, Bahdanau et al focused on embeddings of the many words in the information (addressed by stored states). In essence, they did this by extracting a weighted portion of the secret states.Presently, the inquiry is how could the loads be determined? Indeed, the loads are likewise educated by a feed-forward brain organization and I've referenced their numerical condition underneath. The weighted total of the annotations is used to create the context vector ci for the output word yi:

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j.$$

A softmax function is used to calculate the weights ij and is represented by the following equation:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})},$$

$$e_{ij} = a(s_{i-1}, h_j)$$

A softmax function is used to calculate the weights ij and is represented by the following equation: The output score of a feedforward neural network, eij, seeks to measure the alignment between input at j and output at i. It is defined by the function a.

Fundamentally, the info aspect of the feedforward network is (Tx, 2d) at that time if the encoder generates Tx number of "explanations" (the secret state vectors) with each having aspect d. (accepting the past condition of the decoder likewise has d aspects and these two vectors are connected). To obtain scores eij (with an aspect (Tx, 1)), this information is repeated using a grid Wa of (2d, 1) aspects (clearly followed by extension of the predisposition term).

To obtain the normalised alignment scores for output j, a tan hyperbolic function is applied on top of these eij scores, followed by a softmax:

E = I [Tx*2d] * Wa [2d * 1] + B[Tx*1]

α = softmax(tanh(E))

C= IT * α

So, α is a (Tx, 1) dimensional vector and its elements are the weights corresponding to each word in the input sentence.

Let α is [0.2, 0.3, 0.3, 0.2] and the input sentence is “I am doing it”. Here, the context vector corresponding to it will be:

C=0.2*I”I” + 0.3*I”am” + 0.3*I”doing” + + 0.3*I”it” [Ix is the hidden state corresponding to the word x].

3

Literature Survey

Authors: Neelu K., S. Bhattacharya, "Early Detection of DR Using PCA-Firefly Based DL Model", MDPI Switzerland, 2020.
Description: The study published in MDPI 2020[1] suggested using Deep Neural Networks (DNN) to categorise and organise DR. Dataset was obtained from the University of California ML Repository and Messidor and included 64,000 images of the retinal fundus for the purpose of training the DNN to identify (if present in the patient) and classify the DR into two groups. Proliferative (PDR) and Non-Proliferative DR, for instance (NPDR). The AdamOptimizer was used to prepare the DNN, which made use of PCA and firefly calculations. The model's author claimed that it had a 96 percent accuracy that outperformed previous well-known mixture ML estimates.

Authors: Lifeng Qiao, Ying Zhu, Hui Zhou, "Diabetic Retinopathy Detection involving Prognosis of Microaneurysm and Early Diagnosis System for NonProliferative Diabetic Retinopathy Based on Deep Learning Algorithms", IEEE Access, 2017.
Description: This research [2] provides the Prognosis of Microaneurysm and earlydiagnostics framework for nonproliferative diabetic retinopathy (PMNPDR), which is ready to make DCNNs successfully for the semantic division of fundus pictures, which can further enhance NPDR identification productivity and exactness. It is advised to use a straightforward yet effective coordinated injury ID framework with LOG and MF channels connected by postprocessing methodology. These techniques, when combined sequentially and wisely, provide an incredibly strong framework for the identification of various sores, regardless of their surface, structure, scale, etc.

Authors: JIE LI, JIXIANG GUO, "Determination of DR Using DNN", IEEE, January 2019.
Description: Automates DR diagnosis using Deep CNN to provide the diabetic patients with the appropriate advice following an evaluation of the severity of their retinal images. ReLU, Inception v3, and Resnet calculations are weakened by this model. The E-Optha dataset, which includes 130 photos and 88000 more, is used to create the dataset. Delegated No Dr, NPDR, Mild PDR, and Severe PDR are the

outcomes.

Authors: Yung-Hui Li, Na-NingYeh, " Computer Assisted Diagnosis for DR Based on Fundus Images Using Deep CNN", Hindawi distributer for tech and science diaries, 2019.

Description: By using computations like fragmented max-pooling and support vector machine (SVM), the paper [4] offered a model for the conclusion of DR using DCNN to organise it into NPDR and PDR. The 34000 images in the Kaggle dataset were used to create the rough images, which were arranged with 91 percent accuracy using picture pre-handling techniques as rescaling variation disparity expulsion and fringe evacuation.

Authors: Joel J, J Kivinen, "DL Fundus Images Analysis of DR and Macular Edema Grading", Springer, 2019.

Description: One of the studies published in 2019 by Springer[5] focused on the application of DL techniques to analyse fundus images of reference DR and the use of reviewing techniques to organise macular edoema using deep CNN. It was prepared using the Origin V3 calculation, and the dataset, which included 41000 images of the retinal fundus, was obtained from Digi-Fundus Ltd. The result was referred to as referable and non-referable DR and had a precision of 91%.

Authors: Navoneel Chakrabarty, "A Deep Learning Method for the recognition of diabetic retinopathy", IEEE, 2018.

Description: The approach was developed in [6] to organically organise patients with DR and those without DR. Utilized was an informational index made up of 30 high-resolution fundus images, 15 of which were reliable and 15 of which experienced DR. Adam Optimizer was used in the DCNN's instructional experience.

Authors: SehrishQummar, FiazGul Khan, Sajid Shah, Ahmad Khan, ShahaboddinShamshirband, Zia Ur Rehman, Iftikhar Ahmed Khan, WaqasJadoon, "A Deep Learning Ensemble Approach for Diabetic Retinopathy Detection", IEEE, 2019.

Description: The experts worked on five complex CNN models (resnet50, Inception V3, Xception, Dense121, and Dense169) in order to get the optimum result by combining the models into one and arranging multiple DR stages. For planning, testing, and approval reasons, they used the Kaggle dataset containing 35,126 fundus photos with 64 percent, 20 percent, and 16 percent [7].

Authors: Nikhil M N, Angel Rose A, "Diabetic Retinopathy Stage Classification utilizing CNN", IRJET, 2019.

Description: CNN was used in [8] to organise the DR stages. They made use of images from the 500 retinal images in the Kaggle informative index. In pre-handling, HE sifting calculations are used, and stochastic slope drops with energy advancement calculations are used for preparation. The proposed model, which linked the VGG 16 Alex Net and the Inception net V3 resulted in an accuracy of 81.1 percent.

4

Architecture and Design

In order to put on their fantastic presentation on picture characterisation, CNN has achieved incredible feats. We have used GoogleNet (Inceptionv3) model, which is the most recent Deep CNNs, and do move learning along with hyper-boundary tweaking to see how well these models align with the DR image dataset. On the display of models, pertinent discussion for retinopathy discovery research is provided. The last fully associated layer of a previous prepared CNNs is erased and is then used as an element extractor in the move learning procedure. We train a classifier on the new dataset once all of the highlights for all of the clinical images have been successfully eliminated. Although the actual organisation did not propose the bounds of the hyperparameter-tuning approach, it is crucial to tune and upgrade these boundaries in order to improve the presentation as a result of creating the DR picture. The following fig illustrates the suggested DR detection process utilising CNNs:

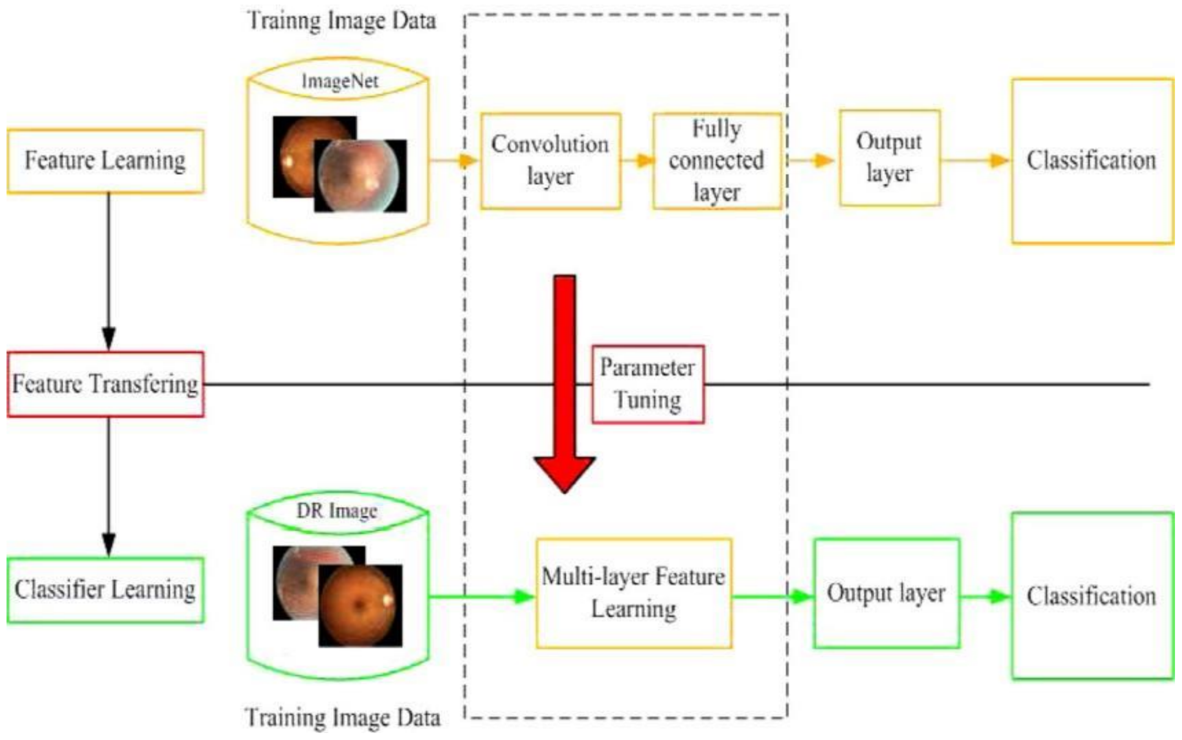


Figure 4.1: Base design of the project

The architecture mainly consists of two machine learning techniques namely Transfer Learning and Hyperparameter tuning.

4.1 Transfer Learning

Transfer learning (TL) is a research area in AI (ML) that focuses on storing knowledge obtained while addressing one problem and applying it to a different but connected problem. Given the enormous financial and time resources required to develop brain network models on these issues and the enormous advances in knowledge they provide on related issues, it is a well-known method in deep learning to use pre-prepared models as the initial stage on PC vision and normal language handling projects. In transfer learning, we first construct an initial base organisation on an initial dataset and task, and then we reuse or transfer the learned highlights to an additional objective organisation that will be trained on an additional objective dataset and task. When the elements are general—that is, applicable to both the base and the target tasks—rather than specifically designed for the base activity, this interaction will frequently succeed.

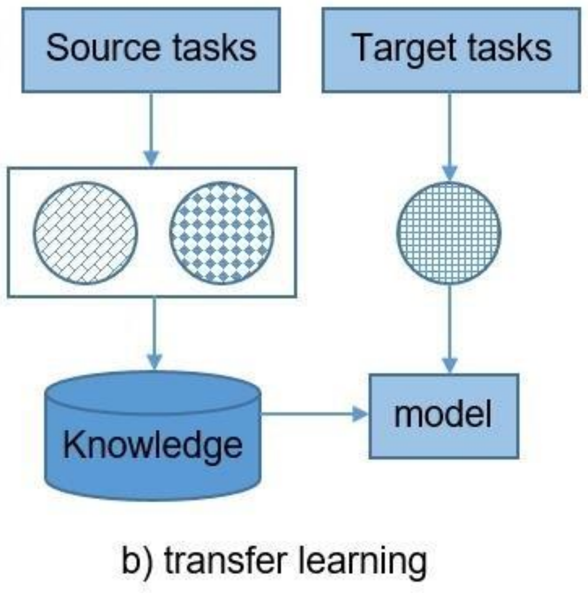


Figure 4.1.1: Transfer Learning

4.2 Hyperparameter Tuning

Selecting the best possible hyperparameters for a learning calculation is known as hyperparameter tuning. A model contention known as a hyperparameter has a value that is predetermined before the learning process begins. Optimizing the hyperparameters is the path to AI computations. A numerical model with numerous boundaries that should be learned from the data is known as a machine learning model. We can match the model bounds by creating a model using the data that is already available.

However, there is another class of boundaries known as Hyperparameters that are not easily obtained from routine preparation. Prior to beginning the actual preparation procedure, they are typically repaired.

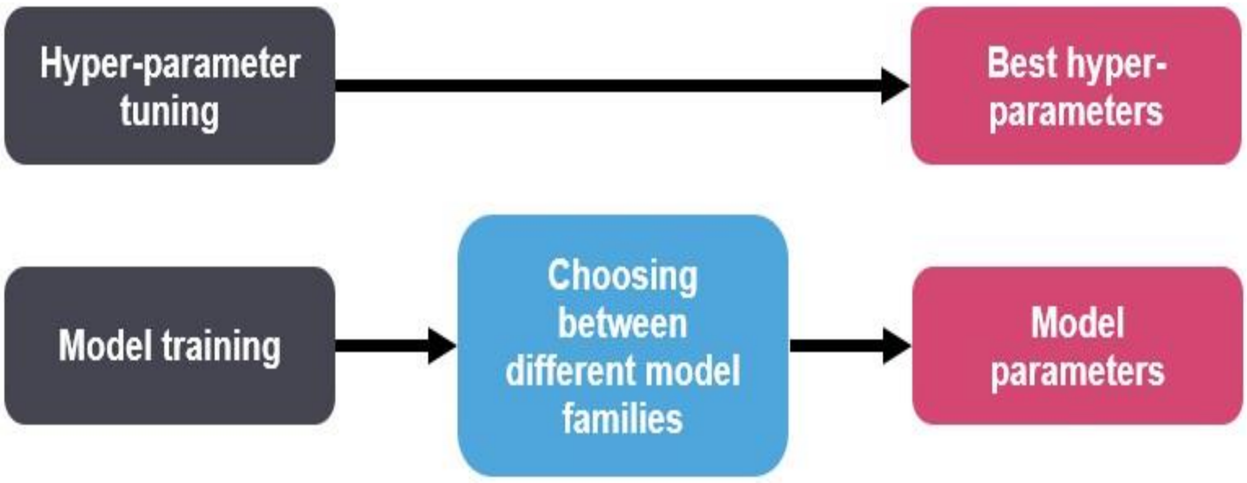


Figure 4.2.1: Hyperparameter Tuning

5

Implementation

5.1 Implementation Platform

5.1.1 Hardware

Following are the hardware specifications used on Kaggle platform for training the model:

1. **CPU**
    - Processor: Intel(R) Xeon(R) CPU @ 2.00GHz
    - Cores: 1

2. **RAM**
    - Total Memory: 16GB

3. **GPU**
    - Model: NVIDIA K80 GPUs
    - GPU Memory: 16GB

4. **HDD**
    - Total Memory: 72GB

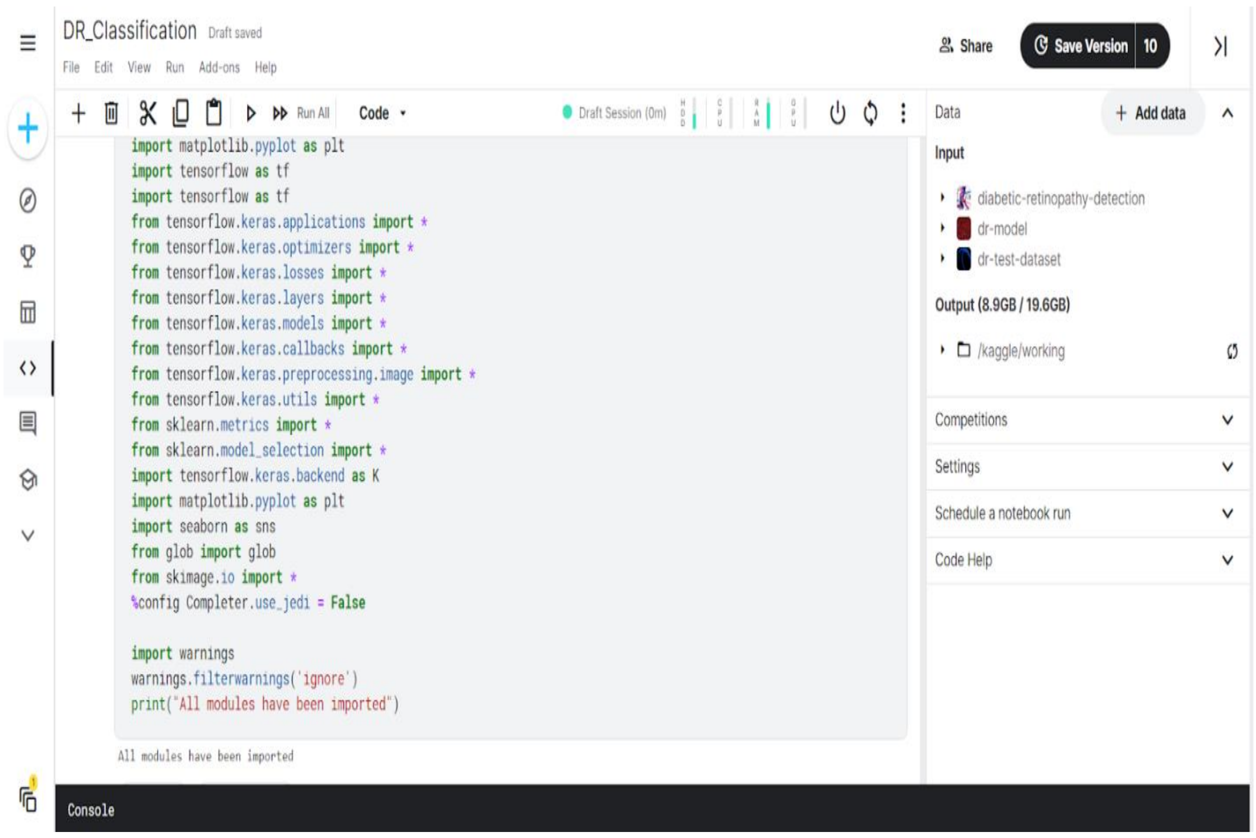
5.1.2 Software

1. **Platform:** Kaggle
2. **Operating System:** Linux 5.10.107
3. **Software Used:** Python
4. **Programming Languages:** Python 3.6

5.2 Implementation Details

5.2.1 Setting up Kaggle

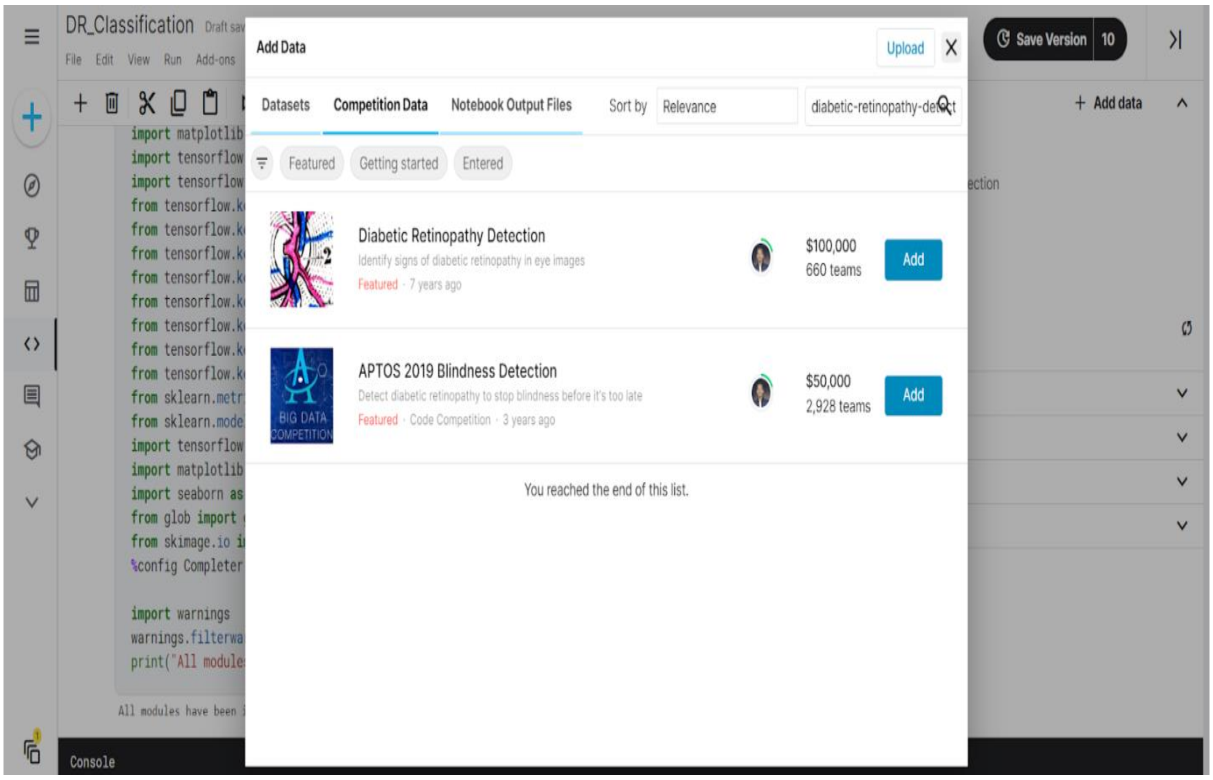
Kaggle is a data science and artificial intelligence platform. Kaggle allows clients to team up with different users, find and distribute datasets and use GPU coordinated journals to tackle information science-related issues. A user has to create an account on the platform to use it. Further, a jupyter notebook is created to write ML/DL code. The below diagram shows the UI of the Kaggle notebook. It contains an input directory and an output directory to store data. The input directory mainly contains dataset while the output directory is used to save models, weights, etc. A dataset already present on Kaggle is added for training the model. This is illustrated in Fig 5.2.2 In settings, we select language as Python and Accelerator as GPU. Finally, we start the session to work with the notebook.



```
import matplotlib.pyplot as plt
import tensorflow as tf
import tensorflow as tf
from tensorflow.keras.applications import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.losses import *
from tensorflow.keras.layers import *
from tensorflow.keras.models import *
from tensorflow.keras.callbacks import *
from tensorflow.keras.preprocessing.image import *
from tensorflow.keras.utils import *
from sklearn.metrics import *
from sklearn.model_selection import *
import tensorflow.keras.backend as K
import matplotlib.pyplot as plt
import seaborn as sns
from glob import glob
from skimage.io import *
%config Completer.use_jedi = False

import warnings
warnings.filterwarnings('ignore')
print("All modules have been imported")
```

```
All modules have been imported
```

Figure 5.2.1: Kaggle Notebook UI



```
import matplotlib
import tensorflow
import tensorflow
from tensorflow.k
from tensorflow.k
from tensorflow.k
from tensorflow.k
from tensorflow.k
from tensorflow.k
from tensorflow.k
from tensorflow.k
from sklearn.metr
from sklearn.mode
import tensorflow
import matplotlib
import seaborn as
from glob import
from skimage.io i
%config Completer

import warnings
warnings.filterwa
print("All module
```

```
All modules have been
```

Diabetic Retinopathy Detection
Identify signs of diabetic retinopathy in eye images
Featured · 7 years ago
$100,000
660 teams

APTOS 2019 Blindness Detection
Detect diabetic retinopathy to stop blindness before it's too late
Featured · Code Competition · 3 years ago
$50,000
2,928 teams

You reached the end of this list.

Figure 5.2.2: Adding dataset in kaggle

5.2.2 Importing Libraries

Neural network libraries are usually used to implement neural networks in computer programs. Below is the list of libraries used for various operations in the project like preprocessing, Augmentation, Normalization, Optimization, Parameter tuning etc.

- Tensorflow
- Keras
- Sklearn
- Matplotlib
- Seaborn

5.2.3 Preparing Dataset

The dataset used in our project for training CNN model is provided by California Healthcare Foundation and is present on the Kaggle platform. It contains more than 35000 high resolution fundus retinal images divided as test and train dataset in compressed form. A train labels csv file is also present to map all training images to their respective levels. Due to very large image file size and limited computing power, we are working on a small subset of data of 872 images. The dataset details and a sample image for each category of data is illustrated below.

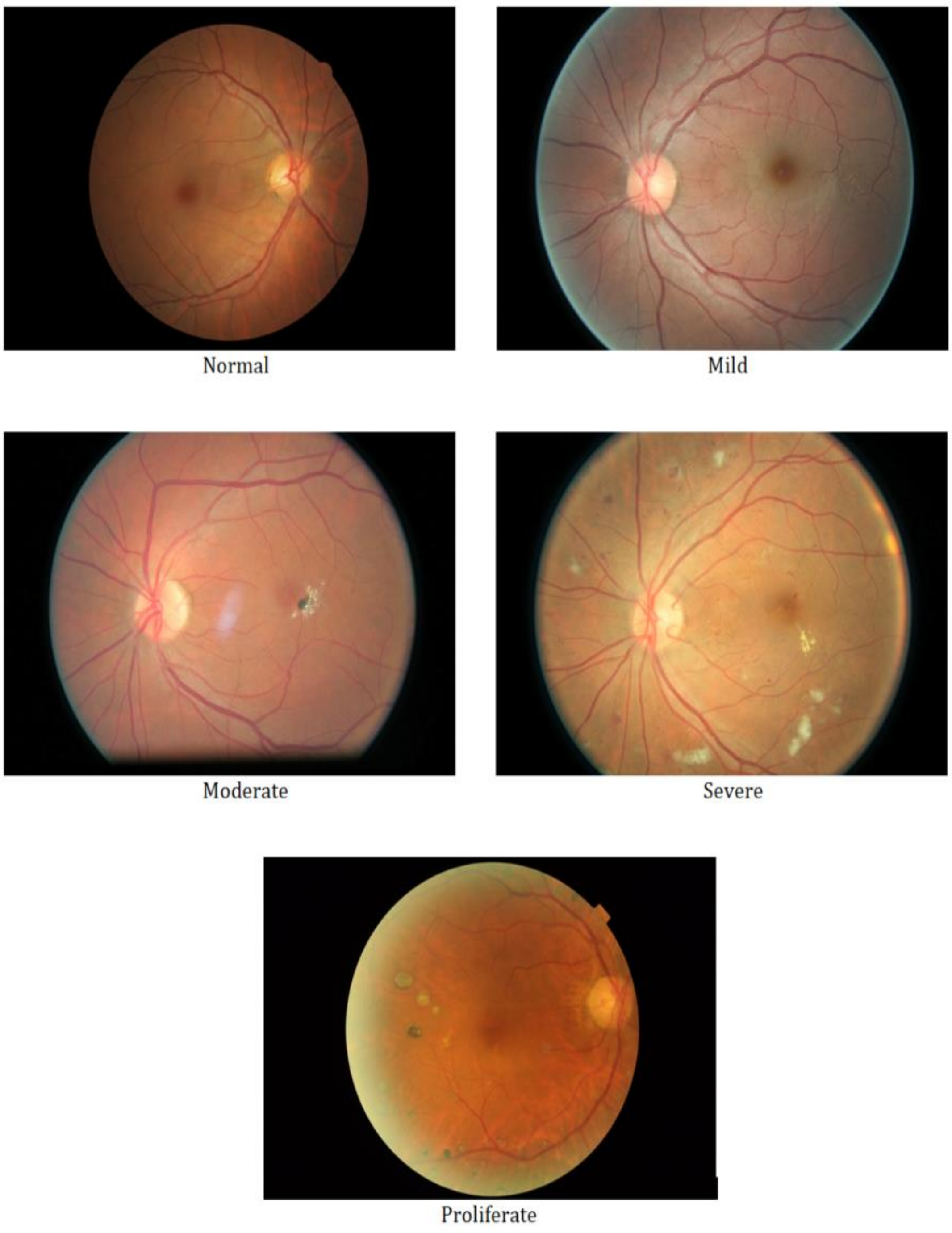
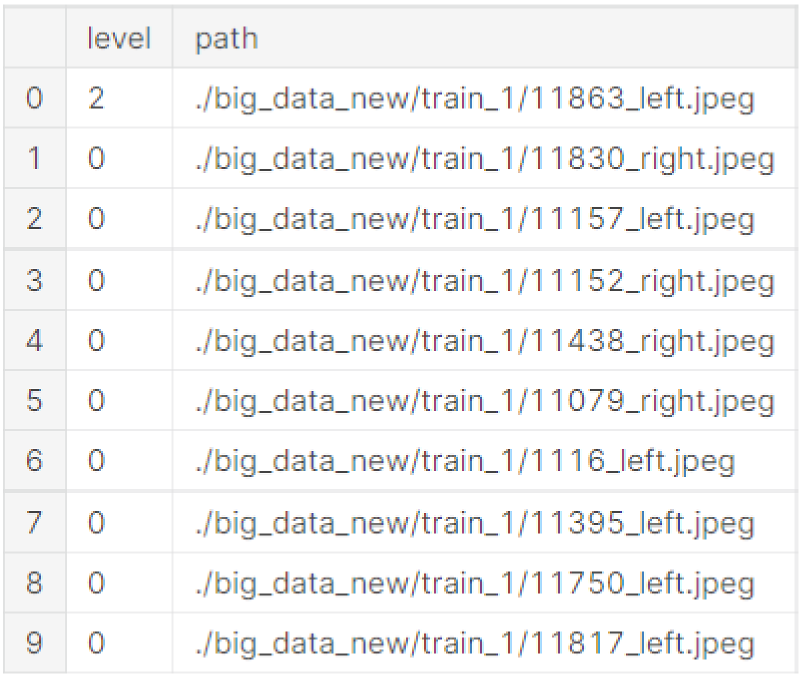


Figure 5.2.3: Sample Images from dataset for each level

In the next step, we filter the trainLabelscsv file and map the path of the image files to their levels for only those 872 image files. The first few outputs of the filtered dataset are represented below.



	level	path
0	2	./big_data_new/train_1/11863_left.jpeg
1	0	./big_data_new/train_1/11830_right.jpeg
2	0	./big_data_new/train_1/11157_left.jpeg
3	0	./big_data_new/train_1/11152_right.jpeg
4	0	./big_data_new/train_1/11438_right.jpeg
5	0	./big_data_new/train_1/11079_right.jpeg
6	0	./big_data_new/train_1/1116_left.jpeg
7	0	./big_data_new/train_1/11395_left.jpeg
8	0	./big_data_new/train_1/11750_left.jpeg
9	0	./big_data_new/train_1/11817_left.jpeg

Figure 5.2.4: Filtered dataset

5.2.4 Splitting dataset into train and test data

In order to assess the performance of our deep learning model, we need to divide a dataset into train and test sets. The statistics of the train set are known and are utilised to fit the model. The validation data set, which is the second set, is utilised to make predictions. For our dataset, we are performing an 75 % - 25 % split of the dataset, with stratification to keep a similar distribution in the validation set.

5.2.5 Balancing the dataset

The data imbalance can be defines as an unequal distribution of classes in a dataset. Our dataset is highly imbalanced with a lot of samples for class 0, and very little for the rest of the levels. The imbalance in the dataset can be observed by analyzing the below graph in figure 5.2.5. There are several methods to balance a dataset like undersampling, oversampling etc. Here, we are oversampling our data such that all classes have the same number of images as the maximum. The new balanced dataset statistics is shown in the figure 5.2.6.

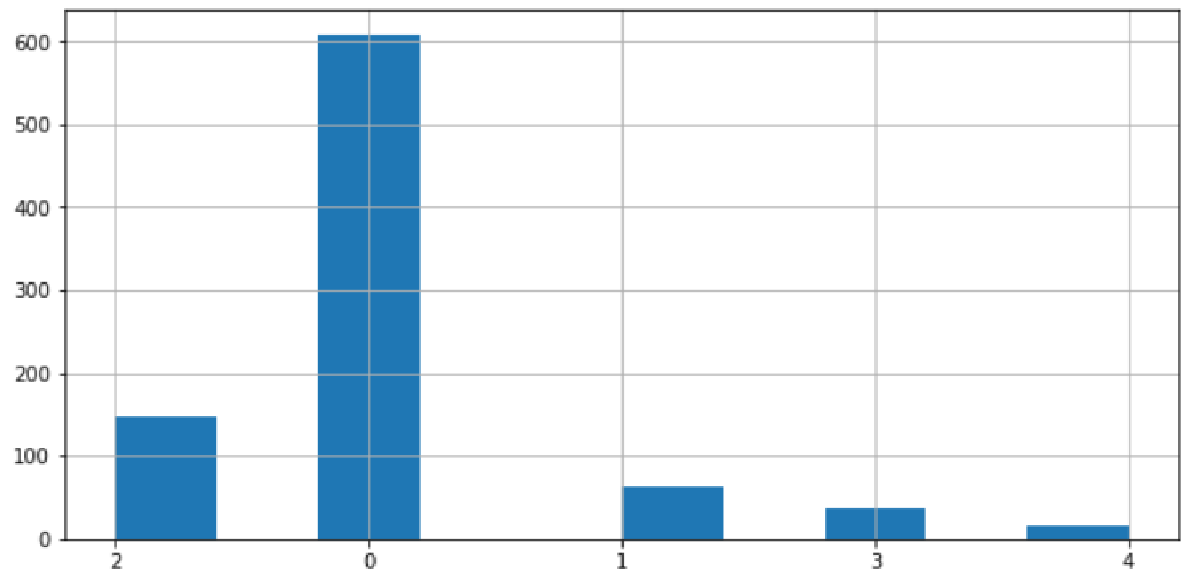


Figure 5.2.5: Imbalanced training data

```
New Data Size: 2255 Old Size: 654
```

```
<AxesSubplot:>
```

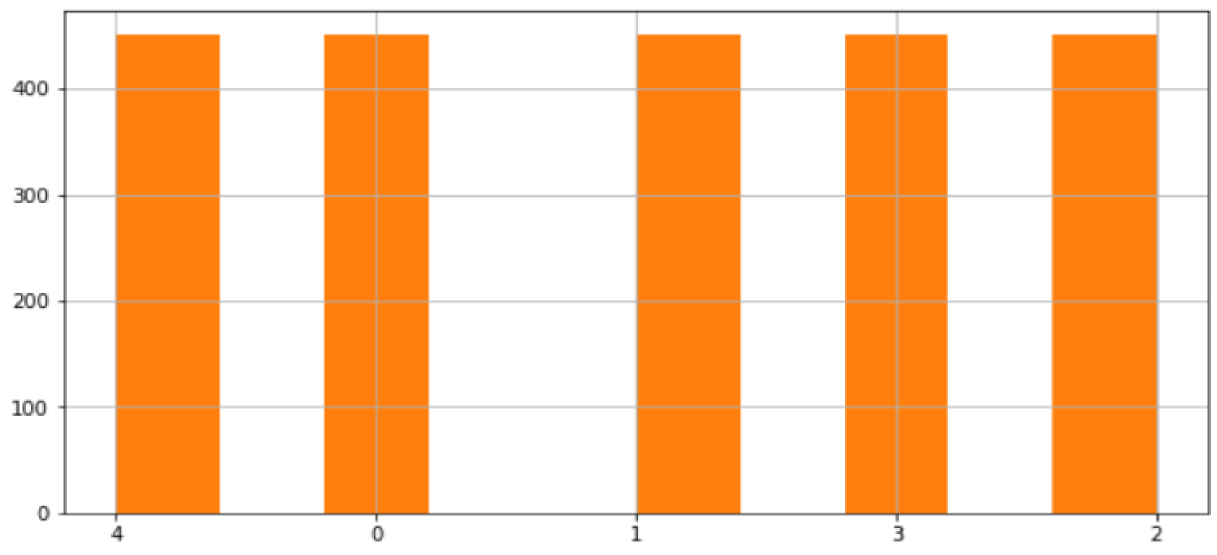
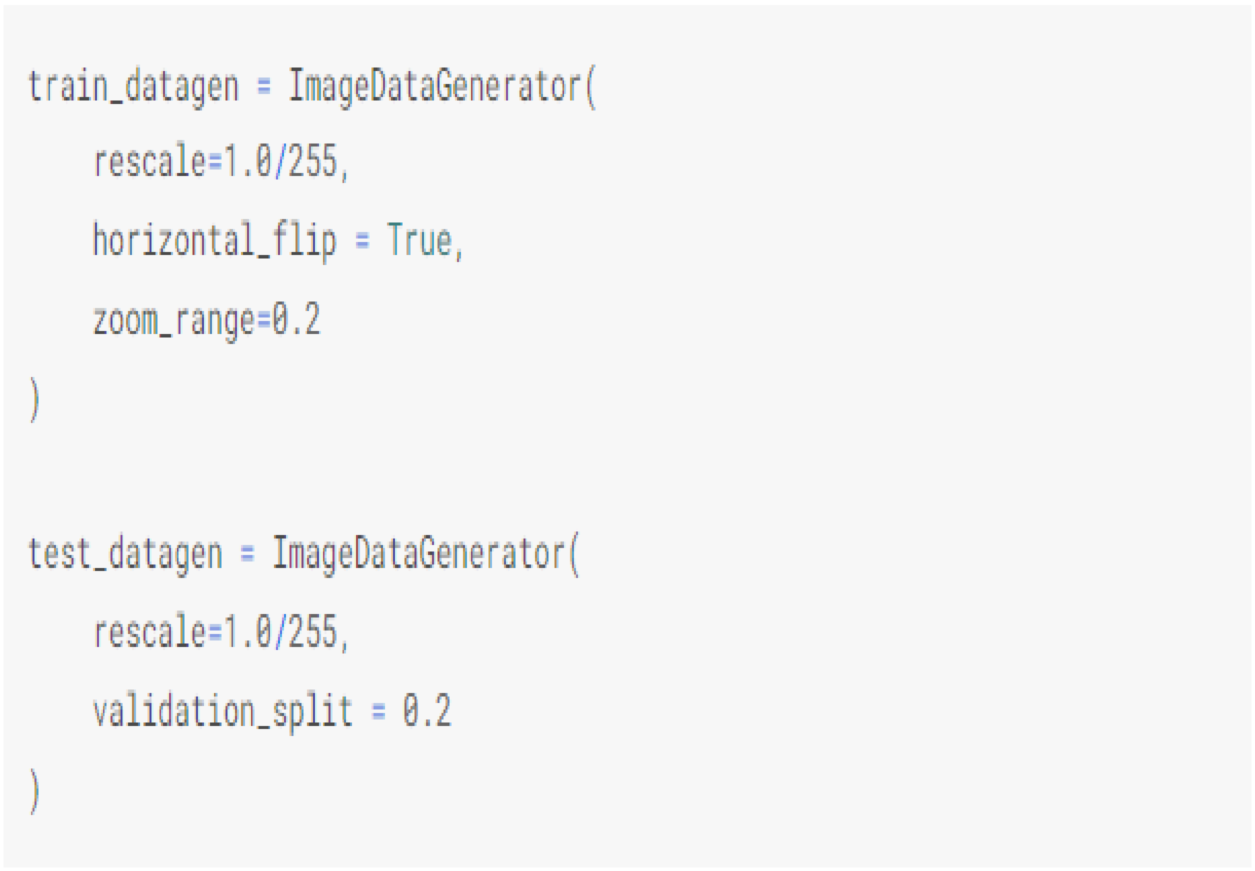


Figure 5.2.6: Balanced training data

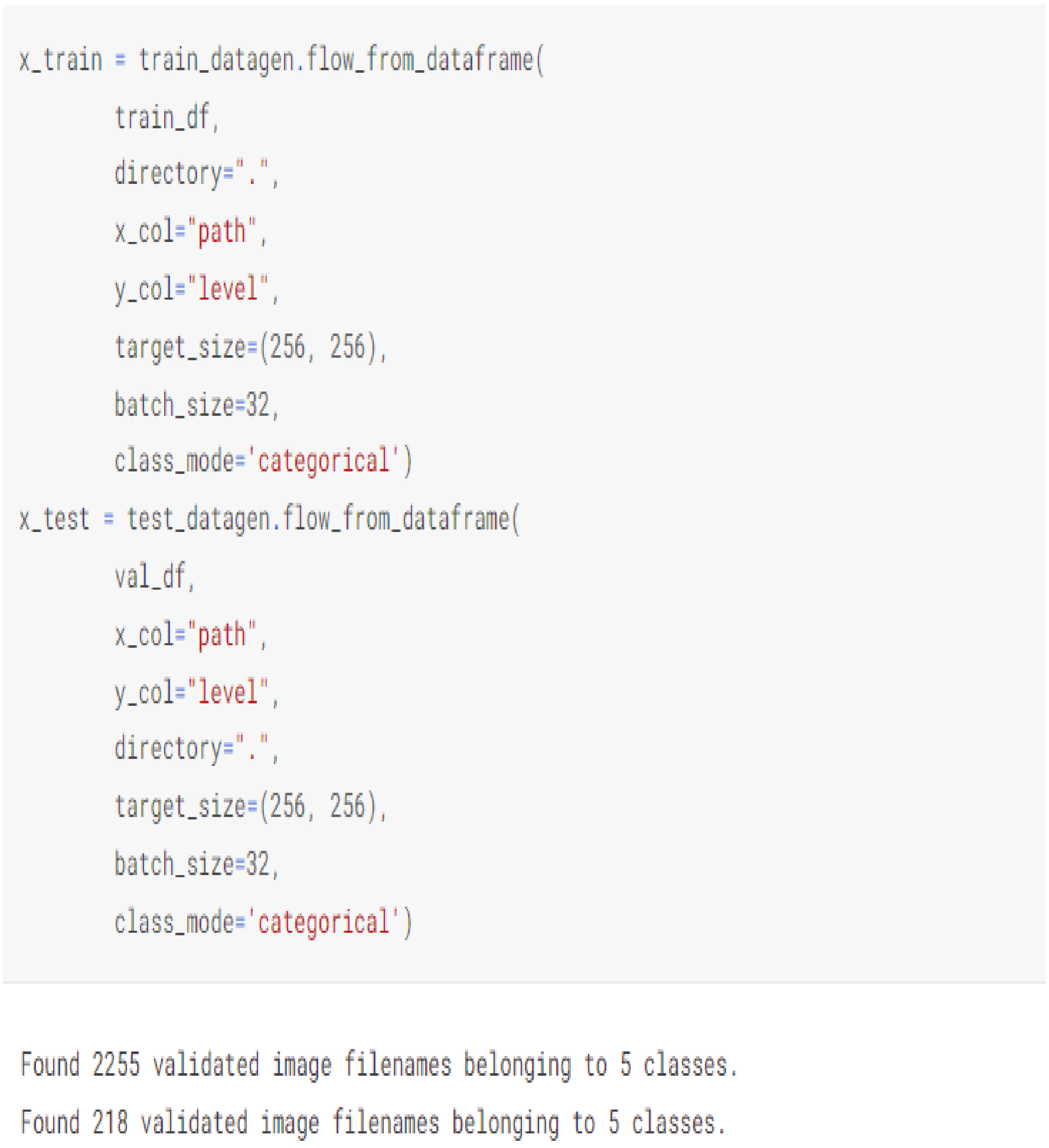
5.2.6 Data Pre-processing and augmentation

We use an ImageData Generator from Keraspre-processing library for data pre-processing. An ImageDataGenerator class generates batches of tensor image data and allows the users to perform image augmentation while training the model. Three separate functions are provided by the ImageDataGenerator class to load the picture dataset in memory and produce batches of augmented data. ".flow()," ".flow from directory()," and ".flow from dataframe.()" are the names of these three functions. The goal of each of these functions is the same—to load the picture dataset into memory and produce batches of modified data—but how they go about it differs. We are utilising the flow from dataframe() technique in our project. This function creates batches of augmented/normalized data from a Pandas DataFrame and a directory path. The implementation of ImageDataGenerator along with parameters used is illustrated in below code snippets.



```
train_datagen = ImageDataGenerator(
    rescale=1.0/255,
    horizontal_flip = True,
    zoom_range=0.2
)

test_datagen = ImageDataGenerator(
    rescale=1.0/255,
    validation_split = 0.2
)
```

Figure 5.2.7: Code Snippet for ImageDataGenerator



```
x_train = train_datagen.flow_from_dataframe(
        train_df,
        directory=".",
        x_col="path",
        y_col="level",
        target_size=(256, 256),
        batch_size=32,
        class_mode='categorical')
x_test = test_datagen.flow_from_dataframe(
        val_df,
        x_col="path",
        y_col="level",
        directory=".",
        target_size=(256, 256),
        batch_size=32,
        class_mode='categorical')
```

```
Found 2255 validated image filenames belonging to 5 classes.
Found 218 validated image filenames belonging to 5 classes.
```

Figure 5.2.8: Code Snippet for preprocessing dataset

5.2.7 Building CNN model

Transfer Learning

The transfer learning is an approach in which a previously learned model is used on a new problem. Using transfer learning, we can train deep neural networks with a small amount of data. For building our CNN, we are using "InceptionV3" as a base pre-trained model. Inception v3 is a CNN which was developed as a module for Googlenet to help with object detection and image analysis..

InceptionV3

The Inceptionv3 model has a total number of 42 layers and a lower error rate than its previous version - Inceptionv1. The design of Inception-v3 was performed to allow deeper neural networks while also keeping the number of parameters from getting too large. It has more than 25 million parameters. The Inceptionv3 model architecture is shown below.

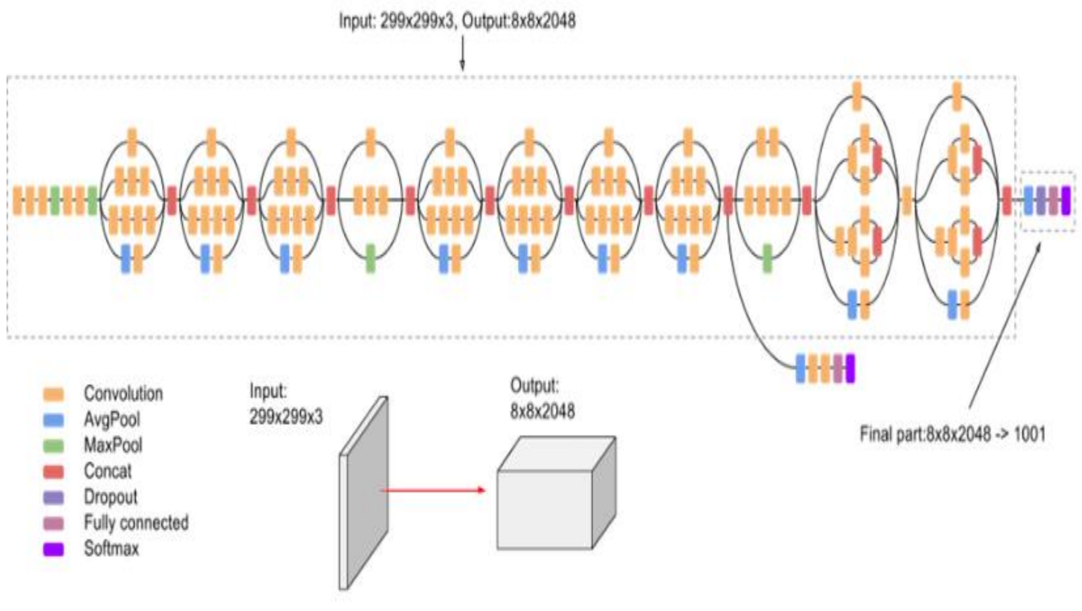
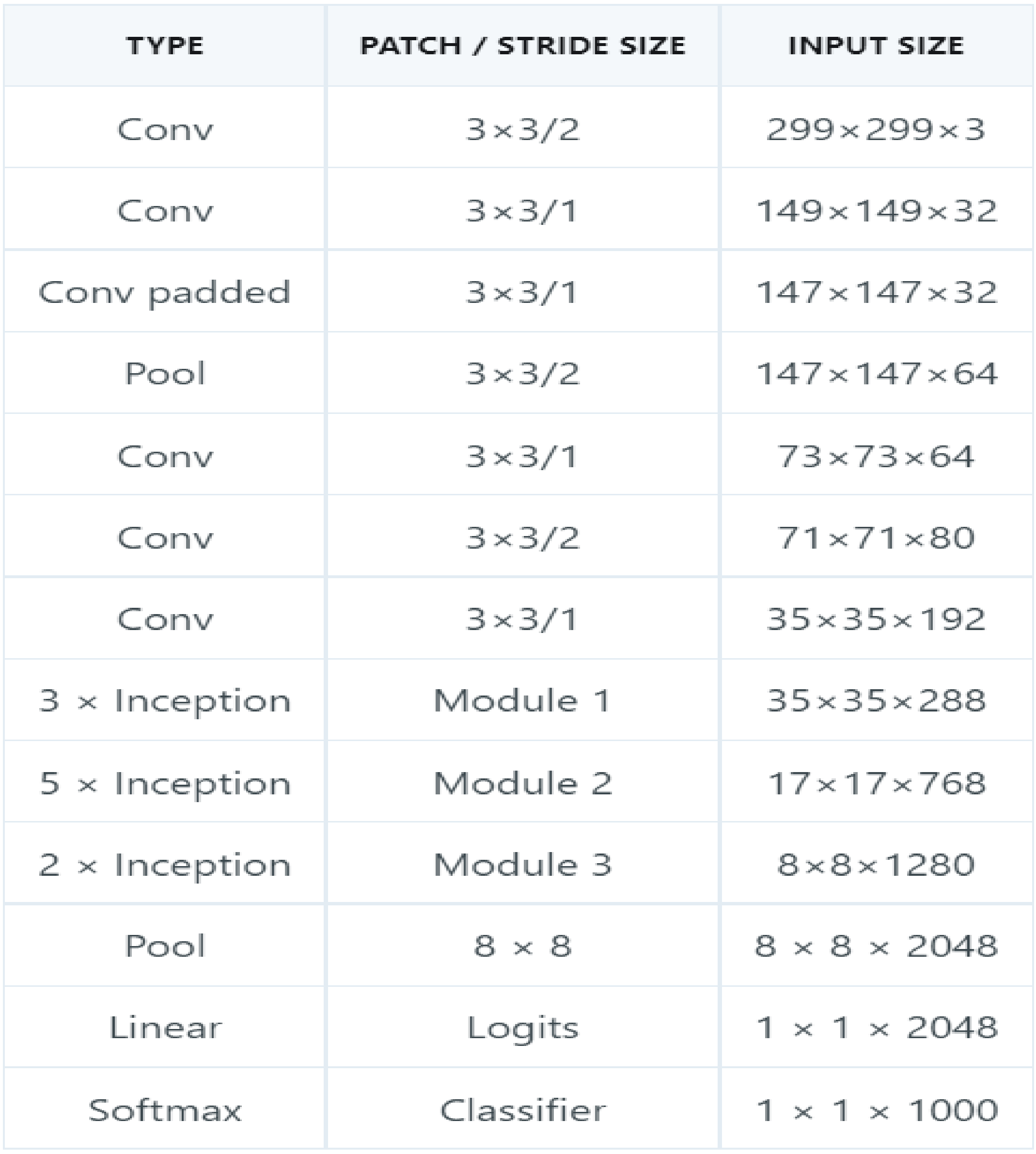


Figure 5.2.9: Inceptionv3 Architecture used



TYPE	PATCH / STRIDE SIZE	INPUT SIZE
Conv	3×3/2	299×299×3
Conv	3×3/1	149×149×32
Conv padded	3×3/1	147×147×32
Pool	3×3/2	147×147×64
Conv	3×3/1	73×73×64
Conv	3×3/2	71×71×80
Conv	3×3/1	35×35×192
3 × Inception	Module 1	35×35×288
5 × Inception	Module 2	17×17×768
2 × Inception	Module 3	8×8×1280
Pool	8 × 8	8 × 8 × 2048
Linear	Logits	1 × 1 × 2048
Softmax	Classifier	1 × 1 × 1000

Figure 5.2.10: Inceptionv3 layers details

Batch Normalization

When training very deep neural networks, a technique called batch normalisation is used to equalise the inputs to each layer for each mini-batch. We are using a batch normalization layer connected to Inceptionv3 output layer for bringing learning stability and significantly lowering the quantity of training epochs needed to build deep networks. This is used to accelerate the training process for deep learning neural networks.

Dropout

One of the general problems in deep learning applications is over-fitting. It happens when the model learns a training dataset well and shows high accuracy on the training dataset, but fails to provide better accuracy on the new test data so that the validation accuracy is much lower. Dropout is a technique used in neural networks to compensate for overfitting. Dropout layer employs a regularisation technique that simulates concurrent training of numerous neural networks with various designs. To account for data from the attention model that are absent, we are using a dropout layer.

GlobalAveragePooling2d

GlobalAveragePooling2D applies an average pooling on all spatial dimensions until each spatial dimension is equal to one, and leaves other dimensions unchanged. We used Global average pooling layer in our model as an replacement for the Flattening after the last pooling layer of our CNN. The 2D Global average pooling block is used instead of the fully connected blocks. The below figure can be referred to as a pictorial demonstration of the Global average pooling layer.

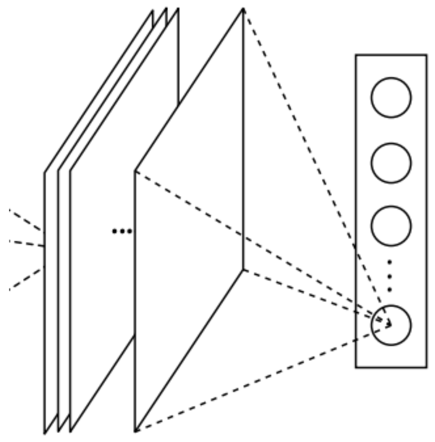


Figure 5.3.11: Global Average Pooling 2d

Attention

The attention mechanism is a cutting-edge method for capturing distant feature interactions and enhancing convolutional neural networks' capacity for representation. Before pooling, we construct an attention mechanism to switch on and off the pixels in the GAP, and we rescale (Lambda layer) the outcomes according to the amount of pixels.

Dense

A dense layer is one that has strong connections to the layer above it and uses matrix vector multiplication to alter the output's dimension. We are adding a dense layer after rescaling GAP in our CNN model.

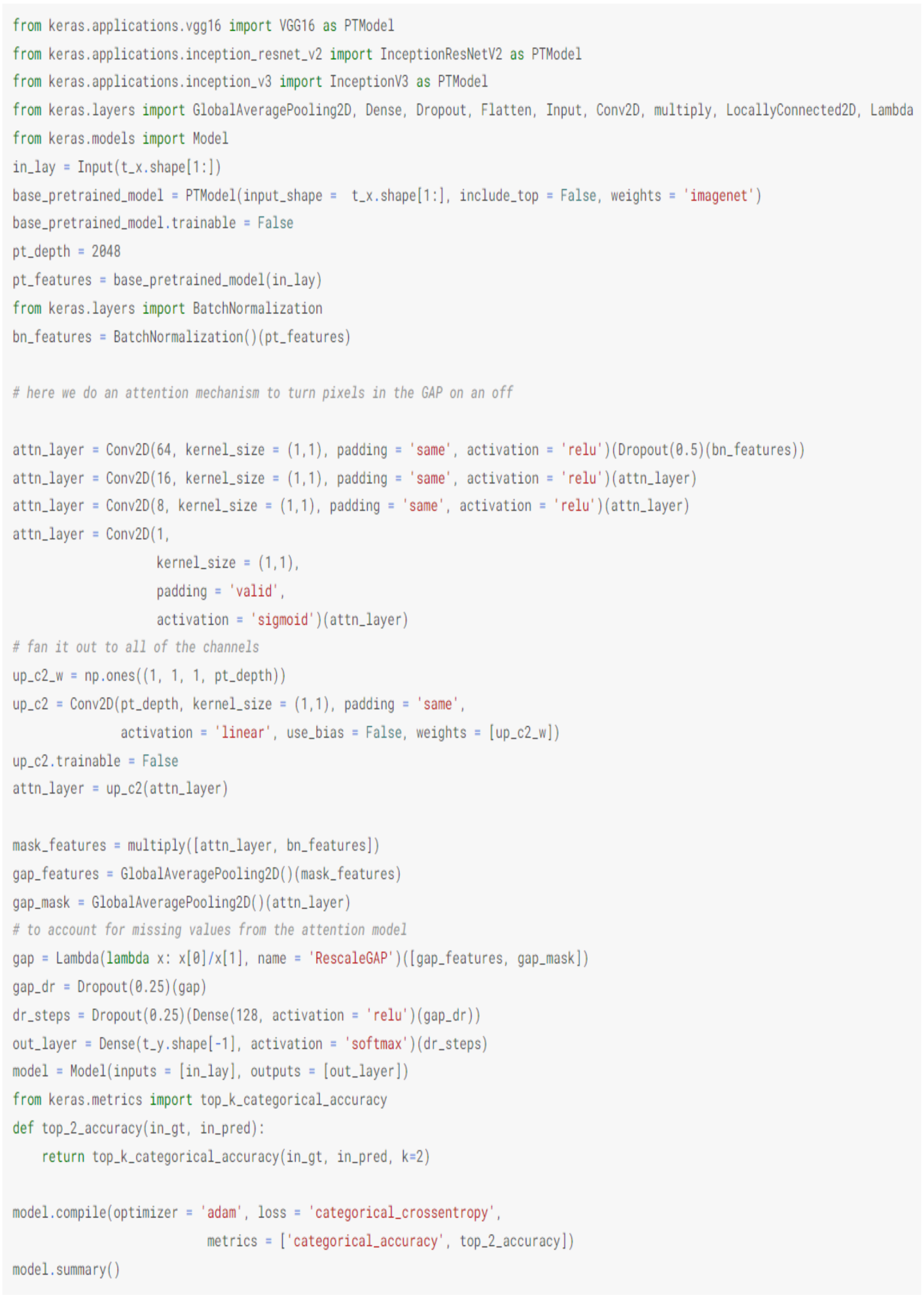
Hyperparameters

Hyperparameters are the set of variables that determine the network structure and determine how the network is trained. General hyperparameters used in our CNN model are: Number of Hidden Layers and units, kernel size, padding, Activation function, Batch size, Number of Epochs, etc.

Adam Optimizer

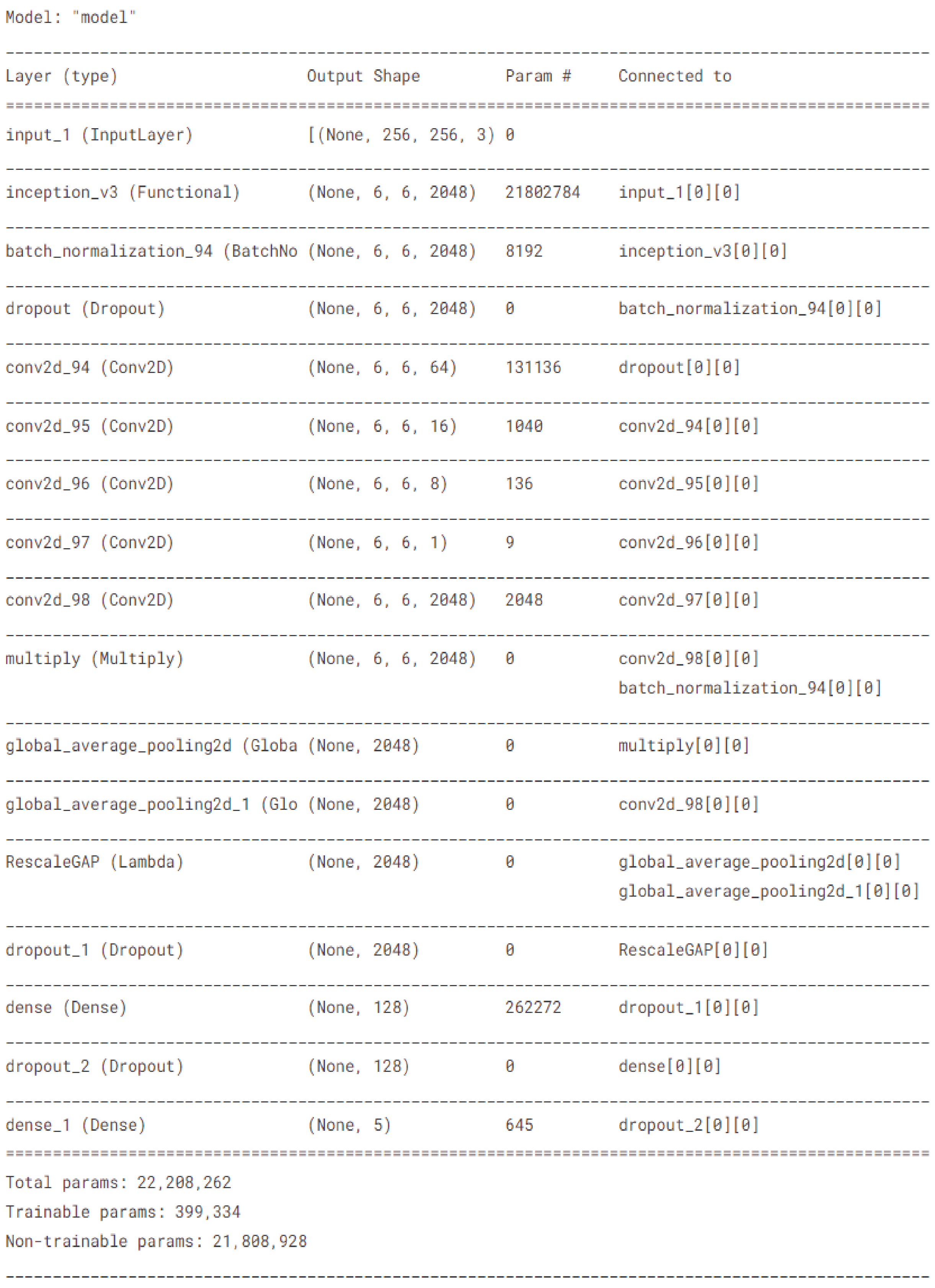
An optimizer is an algorithm that changes the weights, learning rate, and other characteristics of the neural network. Optimizers can help increase accuracy and decrease overall loss. Since Adam Optimizer is very effective when dealing with complex problems involving lots of data or parameters, we are utilising it to compile our model. The "exponentially weighted average" of the gradients is a factor that the Adam optimizer takes into account to speed up the gradient descent procedure. The technique converges quicker towards the minima when averages are used.

The code snippets for building CNN model and final model summary are illustrated below.



```python
from keras.applications.vgg16 import VGG16 as PTModel
from keras.applications.inception_resnet_v2 import InceptionResNetV2 as PTModel
from keras.applications.inception_v3 import InceptionV3 as PTModel
from keras.layers import GlobalAveragePooling2D, Dense, Dropout, Flatten, Input, Conv2D, multiply, LocallyConnected2D, Lambda
from keras.models import Model
in_lay = Input(t_x.shape[1:])
base_pretrained_model = PTModel(input_shape =  t_x.shape[1:], include_top = False, weights = 'imagenet')
base_pretrained_model.trainable = False
pt_depth = 2048
pt_features = base_pretrained_model(in_lay)
from keras.layers import BatchNormalization
bn_features = BatchNormalization()(pt_features)

# here we do an attention mechanism to turn pixels in the GAP on an off

attn_layer = Conv2D(64, kernel_size = (1,1), padding = 'same', activation = 'relu')(Dropout(0.5)(bn_features))
attn_layer = Conv2D(16, kernel_size = (1,1), padding = 'same', activation = 'relu')(attn_layer)
attn_layer = Conv2D(8, kernel_size = (1,1), padding = 'same', activation = 'relu')(attn_layer)
attn_layer = Conv2D(1,
                    kernel_size = (1,1),
                    padding = 'valid',
                    activation = 'sigmoid')(attn_layer)
# fan it out to all of the channels
up_c2_w = np.ones((1, 1, 1, pt_depth))
up_c2 = Conv2D(pt_depth, kernel_size = (1,1), padding = 'same',
               activation = 'linear', use_bias = False, weights = [up_c2_w])
up_c2.trainable = False
attn_layer = up_c2(attn_layer)

mask_features = multiply([attn_layer, bn_features])
gap_features = GlobalAveragePooling2D()(mask_features)
gap_mask = GlobalAveragePooling2D()(attn_layer)
# to account for missing values from the attention model
gap = Lambda(lambda x: x[0]/x[1], name = 'RescaleGAP')([gap_features, gap_mask])
gap_dr = Dropout(0.25)(gap)
dr_steps = Dropout(0.25)(Dense(128, activation = 'relu')(gap_dr))
out_layer = Dense(t_y.shape[-1], activation = 'softmax')(dr_steps)
model = Model(inputs = [in_lay], outputs = [out_layer])
from keras.metrics import top_k_categorical_accuracy
def top_2_accuracy(in_gt, in_pred):
    return top_k_categorical_accuracy(in_gt, in_pred, k=2)

model.compile(optimizer = 'adam', loss = 'categorical_crossentropy',
                           metrics = ['categorical_accuracy', top_2_accuracy])
model.summary()
```

Figure 5.2.12: Code snippet for building CNN Model

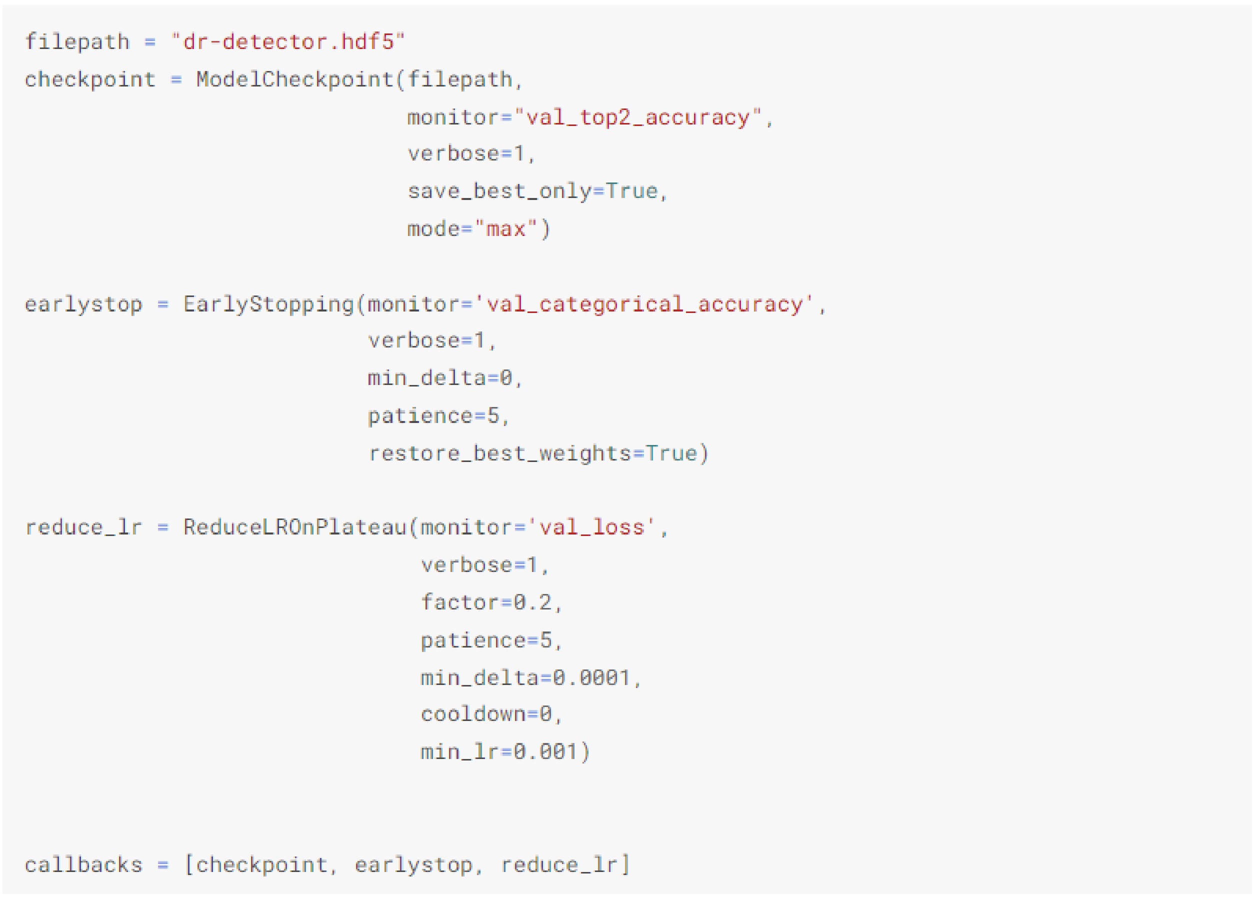


```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 256, 256, 3) 0
__________________________________________________________________________________________________
inception_v3 (Functional)       (None, 6, 6, 2048)   21802784    input_1[0][0]
__________________________________________________________________________________________________
batch_normalization_94 (BatchNo (None, 6, 6, 2048)   8192        inception_v3[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 6, 6, 2048)   0           batch_normalization_94[0][0]
__________________________________________________________________________________________________
conv2d_94 (Conv2D)              (None, 6, 6, 64)     131136      dropout[0][0]
__________________________________________________________________________________________________
conv2d_95 (Conv2D)              (None, 6, 6, 16)     1040        conv2d_94[0][0]
__________________________________________________________________________________________________
conv2d_96 (Conv2D)              (None, 6, 6, 8)      136         conv2d_95[0][0]
__________________________________________________________________________________________________
conv2d_97 (Conv2D)              (None, 6, 6, 1)      9           conv2d_96[0][0]
__________________________________________________________________________________________________
conv2d_98 (Conv2D)              (None, 6, 6, 2048)   2048        conv2d_97[0][0]
__________________________________________________________________________________________________
multiply (Multiply)             (None, 6, 6, 2048)   0           conv2d_98[0][0]
                                                                 batch_normalization_94[0][0]
__________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 2048)         0           multiply[0][0]
__________________________________________________________________________________________________
global_average_pooling2d_1 (Glo (None, 2048)         0           conv2d_98[0][0]
__________________________________________________________________________________________________
RescaleGAP (Lambda)             (None, 2048)         0           global_average_pooling2d[0][0]
                                                                 global_average_pooling2d_1[0][0]
__________________________________________________________________________________________________
dropout_1 (Dropout)             (None, 2048)         0           RescaleGAP[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 128)          262272      dropout_1[0][0]
__________________________________________________________________________________________________
dropout_2 (Dropout)             (None, 128)          0           dense[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 5)            645         dropout_2[0][0]
==================================================================================================
Total params: 22,208,262
Trainable params: 399,334
Non-trainable params: 21,808,928
__________________________________________________________________________________________________
```

Figure 5.2.13: CNN model summary

Callbacks for parameter tuning

A callback is defined a set of functions that can be applied at a given stage of training the model. During model training, the callbacks can be used to receive a view of the model's internal states and statistics. After each epoch, we automate activities via a callback, giving us control over the training process. This includes modifying the learning rates over time, ending training when you reach a specific accuracy/loss score, saving your model as a checkpoint following each successful epoch, etc. For training our model we are using three types of callbacks - checkpoint, which saves the model after each epoch in a file, early stop, which stops the training when a certain parameter doesn’t changes over epochs and reduce lr, which stops the training when the loss doesn’t changes over certain epoch. The code snippet for callbacks is shown in the figure 5.2.14



```
filepath = "dr-detector.hdf5"
checkpoint = ModelCheckpoint(filepath,
                             monitor="val_top2_accuracy",
                             verbose=1,
                             save_best_only=True,
                             mode="max")

earlystop = EarlyStopping(monitor='val_categorical_accuracy',
                          verbose=1,
                          min_delta=0,
                          patience=5,
                          restore_best_weights=True)

reduce_lr = ReduceLROnPlateau(monitor='val_loss',
                              verbose=1,
                              factor=0.2,
                              patience=5,
                              min_delta=0.0001,
                              cooldown=0,
                              min_lr=0.001)


callbacks = [checkpoint, earlystop, reduce_lr]
```

Figure 5.2.14: Code Snippets for callback

5.2.8 Training CNN model

We train our model by calling the model.fit() method from keras library. Model.fit() technique runs the model through a predetermined number of epochs (iterations on a dataset). The arguments passed as parameters are - training data, steps per epoch, number of epochs, the validation data, the validation steps and callbacks.

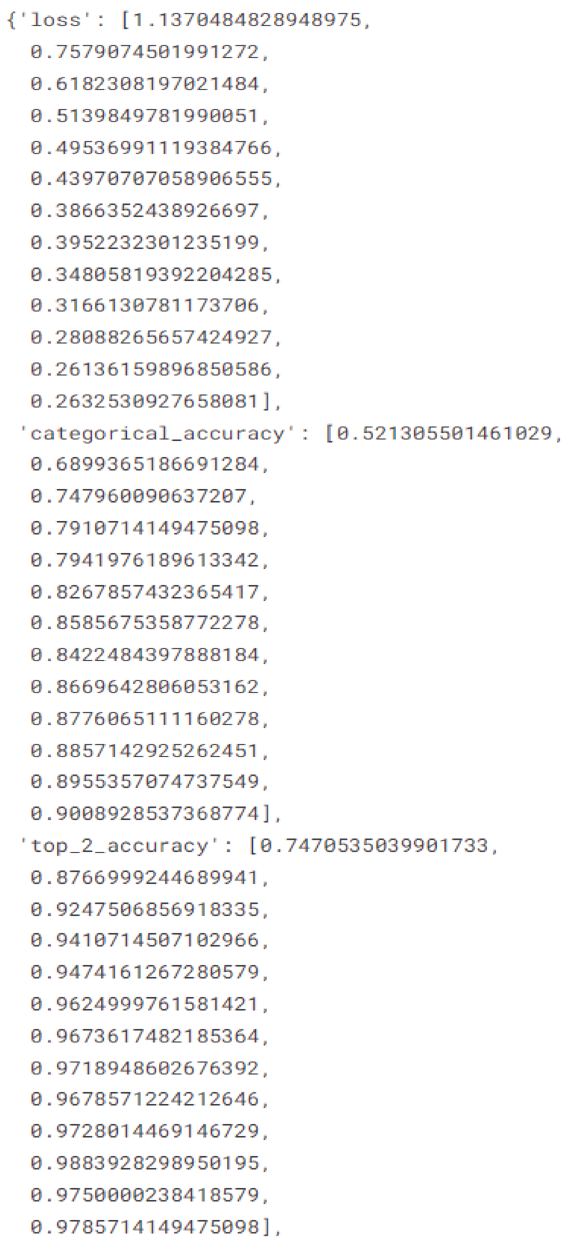
5.2.9 Saving CNN model

It generally takes a long time to train a deep learning model, especially if the hardware capacity of the system is inadequate. After training is complete, the model is saved to a file so that it can be used to generate predictions at a later time. Models can be saved in three different formats using Keras: YAML, JSON, and HDF5. Our model is being saved in the Kaggle output directory in HDF5 format.

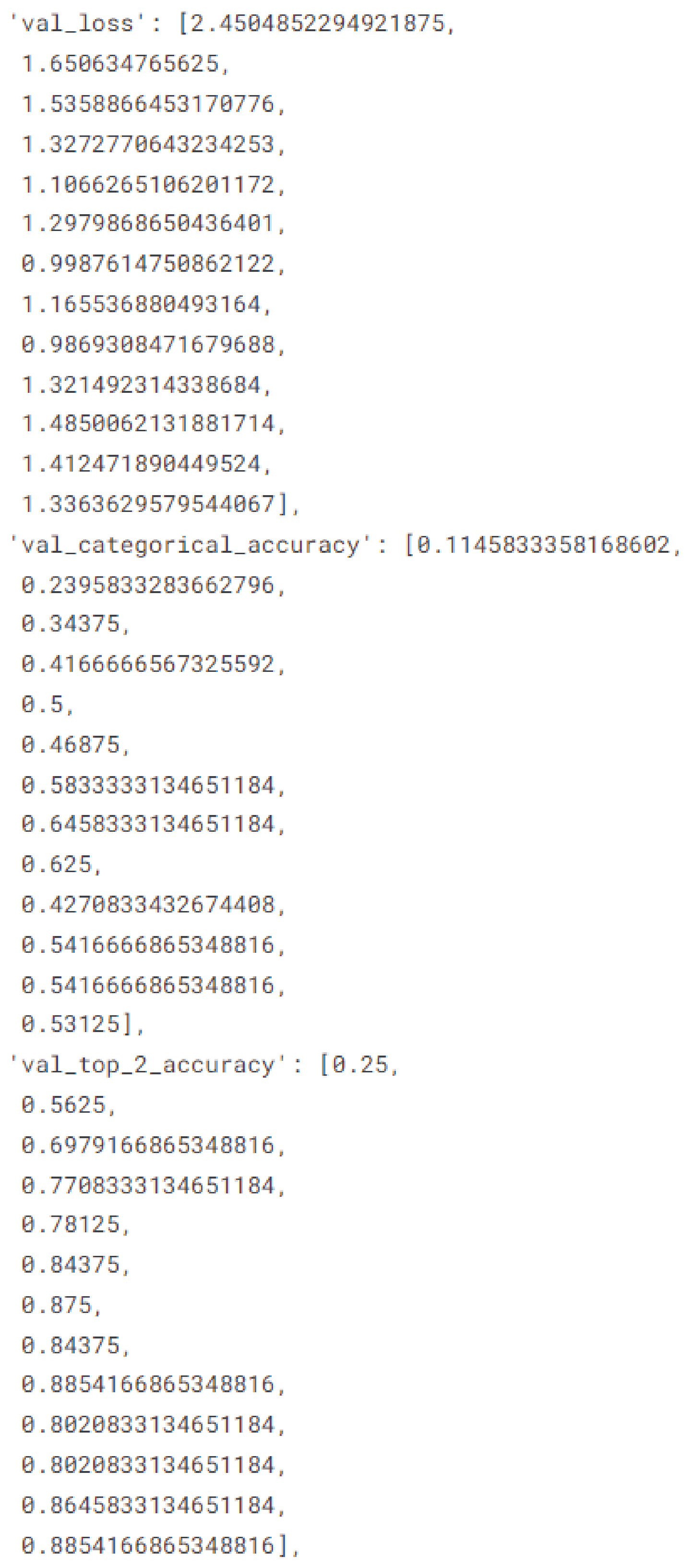
6

Testing

Testing of the CNN model is performed on validation data using batch size of 64. The results are calculated based on validation loss, validation categorical accuracy and validation top 2 accuracy which changes with every epoch. Finally we evaluate the score for both training and validation data. The results of the testing are shown in the following figures.



```
{'loss': [1.1370484828948975,
  0.7579074501991272,
  0.6182308197021484,
  0.5139849781990051,
  0.49536991119384766,
  0.43970707058906555,
  0.3866352438926697,
  0.3952232301235199,
  0.34805819392204285,
  0.3166130781173706,
  0.28088265657424927,
  0.26136159896850586,
  0.2632530927658081],
 'categorical_accuracy': [0.521305501461029,
  0.6899365186691284,
  0.747960090637207,
  0.7910714149475098,
  0.7941976189613342,
  0.8267857432365417,
  0.8585675358772278,
  0.842248397888184,
  0.8669642806053162,
  0.8776065111160278,
  0.8857142925262451,
  0.8955357074737549,
  0.9008928537368774],
 'top_2_accuracy': [0.7470535039901733,
  0.8766999244689941,
  0.9247506856918335,
  0.9410714507102966,
  0.9474161267280579,
  0.9624999761581421,
  0.9673617482185364,
  0.9718948602676392,
  0.9678571224212646,
  0.9728014469146729,
  0.9883928298950195,
  0.9750000238418579,
  0.9785714149475098],
```



```
'val_loss': [2.4504852294921875,
 1.650634765625,
 1.5358866453170776,
 1.3272770643234253,
 1.1066265106201172,
 1.2979868650436401,
 0.9987614750862122,
 1.165536880493164,
 0.9869308471679688,
 1.321492314338684,
 1.4850062131881714,
 1.412471890449524,
 1.3363629579544067],
'val_categorical_accuracy': [0.1145833358168602,
 0.2395833283662796,
 0.34375,
 0.4166666567325592,
 0.5,
 0.46875,
 0.583333134651184,
 0.6458333134651184,
 0.625,
 0.4270833432674408,
 0.5416666865348816,
 0.5416666865348816,
 0.53125],
'val_top_2_accuracy': [0.25,
 0.5625,
 0.6979166865348816,
 0.7708333134651184,
 0.78125,
 0.84375,
 0.875,
 0.84375,
 0.8854166865348816,
 0.8020833134651184,
 0.8020833134651184,
 0.8645833134651184,
 0.8854166865348816],
```

Figure 6.1: History of training for each epoch

The results of the testing are shown in the following tables.

Loss	0.26
Categorical Accuracy	90.08%
Top 2 accuracy	98.83%

Performance metrics for Training data

Loss	0.98
Categorical Accuracy	64.58%
Top 2 accuracy	88.54%

Performance metrics for Validation data

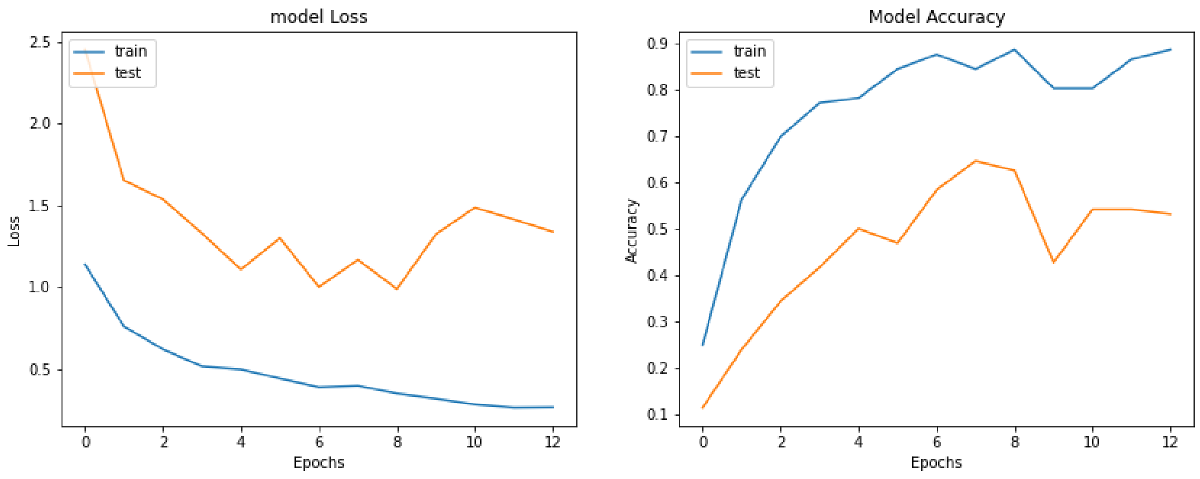


Figure 6.2: Graphical Representation for model loss and accuracy

The Attention map for some test images are represented in the figure below.

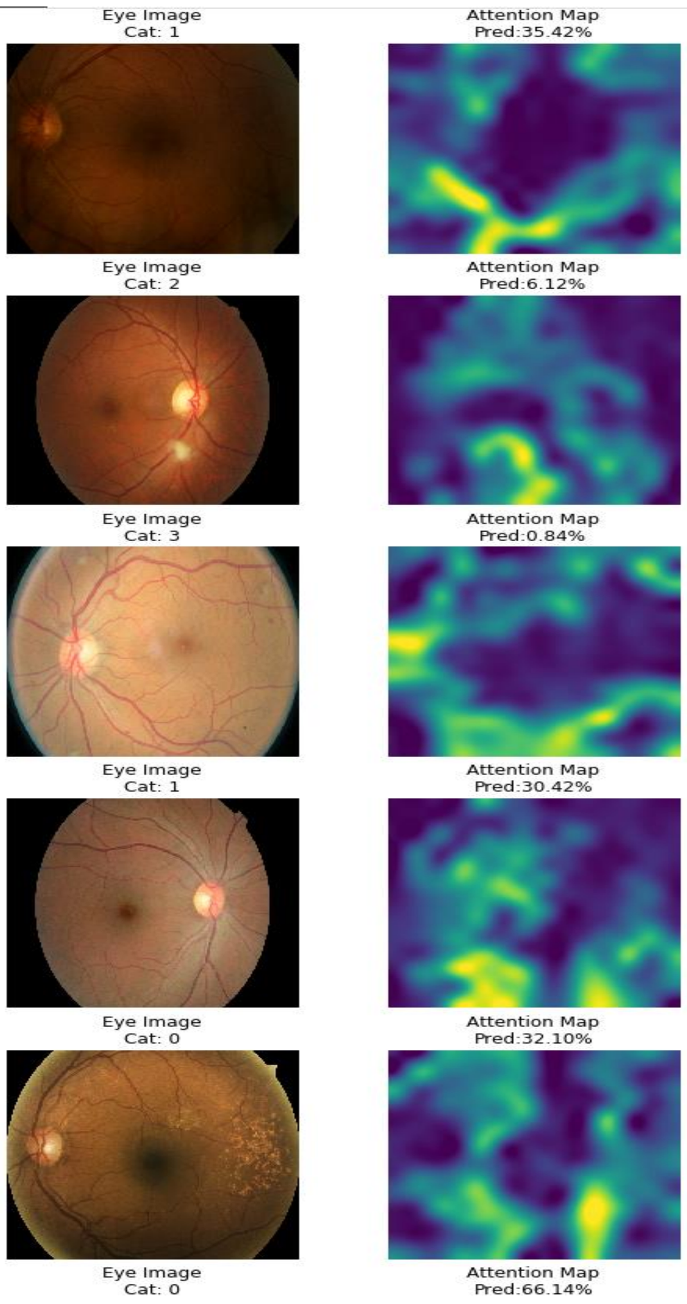
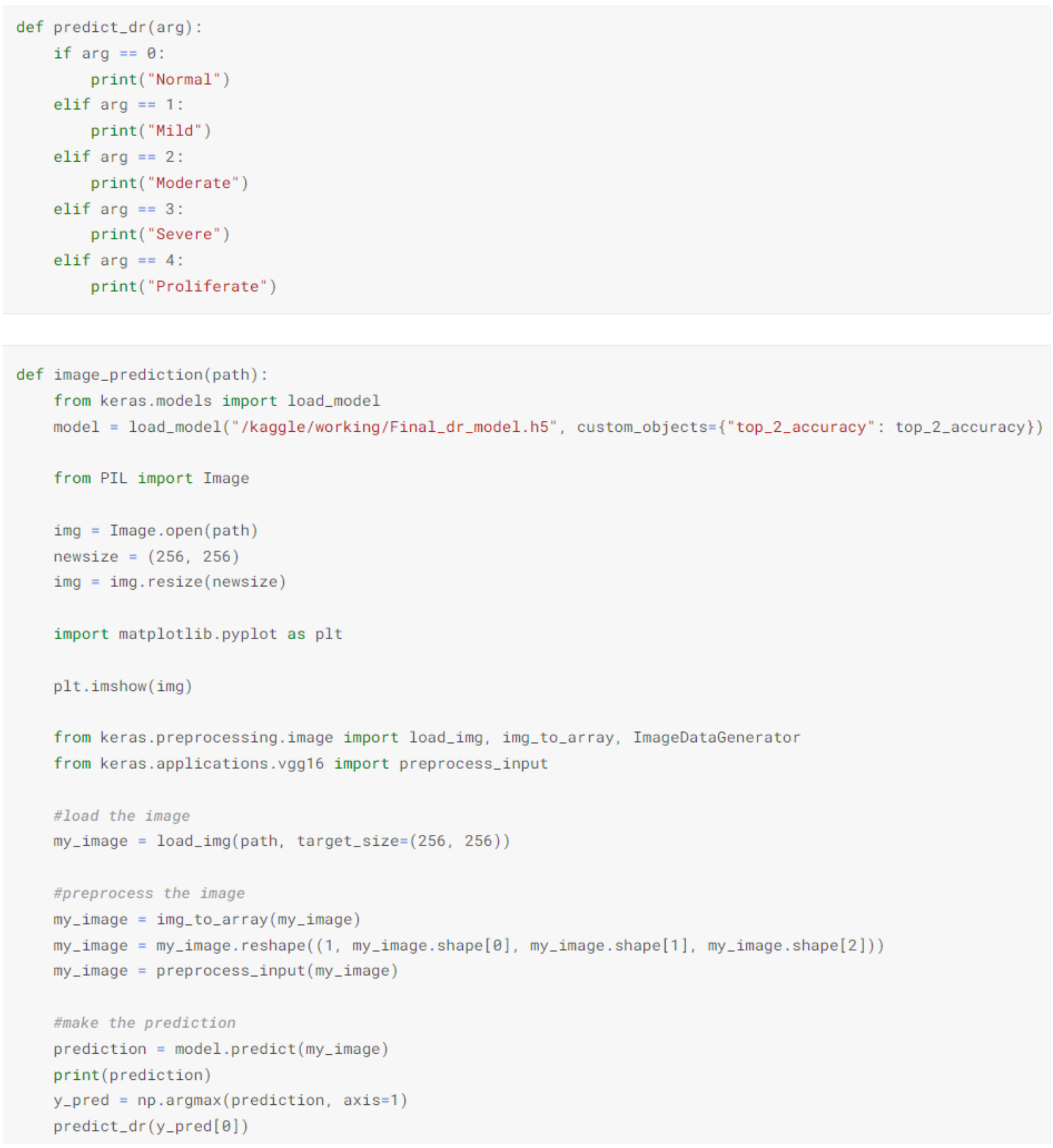


Figure 6.3: Attention map for some images

7
Experimentation and Results

We randomly took a few retinal images from the kaggle dataset and performed experimentation to analyze the results. We made a helper function for experimentation and test results for our CNN model. The helper function takes the image path as an argument, pre process the image as per the input requirements of the model and then predicts the classification output. The code snippet for the helper function is shown below.



```python
def predict_dr(arg):
    if arg == 0:
        print("Normal")
    elif arg == 1:
        print("Mild")
    elif arg == 2:
        print("Moderate")
    elif arg == 3:
        print("Severe")
    elif arg == 4:
        print("Proliferate")
```

```python
def image_prediction(path):
    from keras.models import load_model
    model = load_model("/kaggle/working/Final_dr_model.h5", custom_objects={"top_2_accuracy": top_2_accuracy})

    from PIL import Image

    img = Image.open(path)
    newsize = (256, 256)
    img = img.resize(newsize)

    import matplotlib.pyplot as plt

    plt.imshow(img)

    from keras.preprocessing.image import load_img, img_to_array, ImageDataGenerator
    from keras.applications.vgg16 import preprocess_input

    #load the image
    my_image = load_img(path, target_size=(256, 256))

    #preprocess the image
    my_image = img_to_array(my_image)
    my_image = my_image.reshape((1, my_image.shape[0], my_image.shape[1], my_image.shape[2]))
    my_image = preprocess_input(my_image)

    #make the prediction
    prediction = model.predict(my_image)
    print(prediction)
    y_pred = np.argmax(prediction, axis=1)
    predict_dr(y_pred[0])
```

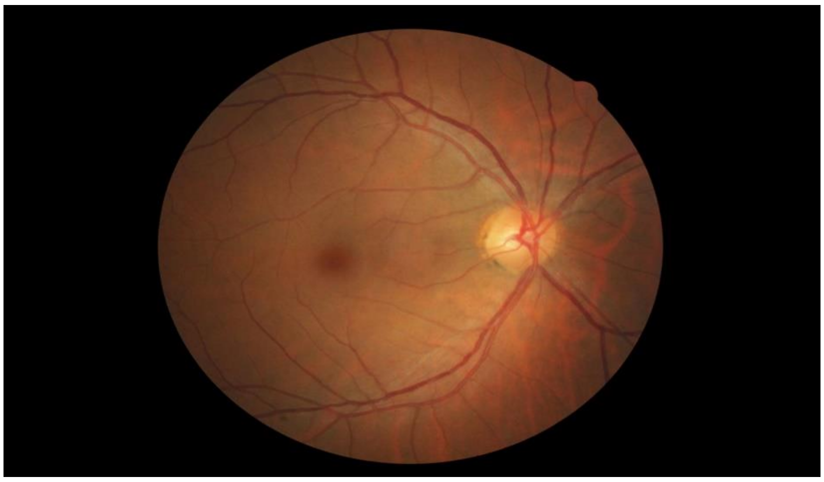
Figure 7.1: Code snippet for Helper function

Test case 1:

Image data: Test_1

Expected output: Normal

DR Image:



Result: Normal

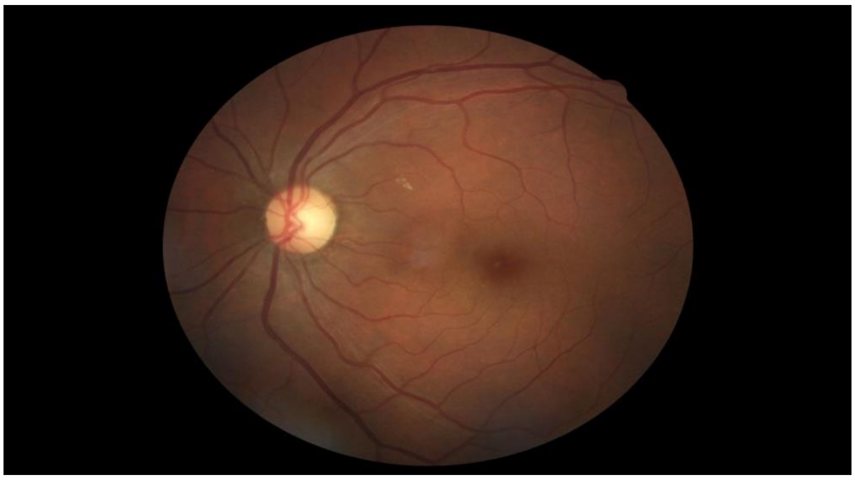
Test case passed

Test case 2:

Image data: Test_9

Expected output: Moderate

Image:



Result: Moderate

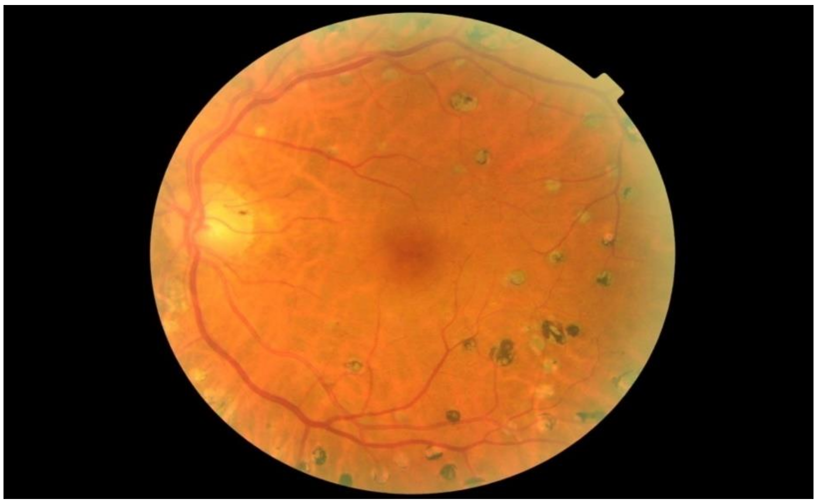
Test case passed

Test case 3:

Image data: Test_10

Expected output: Proliferate

Image:



Result: Proliferate

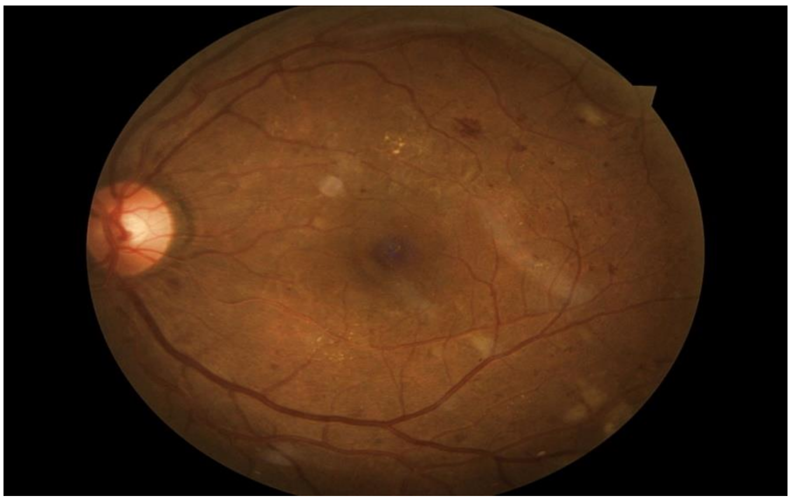
Test case passed

Test case 4:

Image data: Test_15

Expected output: Severe

Image:



Result: Moderate

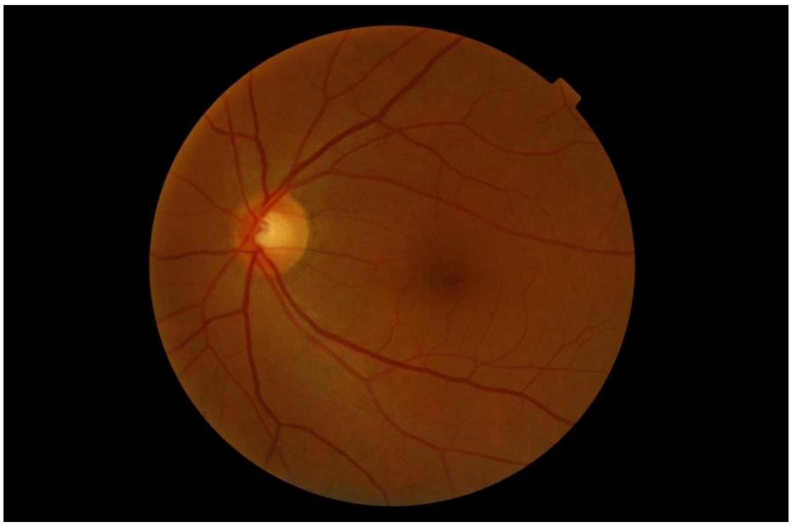
Test case Failed

Test case 5:

Image data: Test_3

Expected output: Normal

Image:



Result: Normal

Test case passed

Conclusion

In this work, we presented a model to detect diabetic retinopathy at its early stages. We have used the Kaggle competition dataset provided by the California Healthcare Foundation which helped us check the model's feature acquisition for different stages of DR directly from examination images. We have performed several preprocessing on datasets including oversampling to balance the data and data augmentation to improve the performance and outcomes of the model. Our CNN is based on a transfer learning approach in which we have used Inceptionv3 as our base pre-trained model and then added several layers like Batch Normalization, Attention, GlobalAveragePooling2d, and Dense layer to build a final CNN model. The model is optimized using the Adam optimizer. Our proposal achieved good performance with maximum top 2 accuracy of 88% and categorical accuracy of 64%. A fair comparison of this work with previous methods is still missing given the lack of resources and computing power to train models on a larger dataset and methodology to propose a new solution to classify DR subclasses.

In future works, we propose to train our model on a larger dataset and create a benchmark dataset to follow as a baseline comparison. Additionally, we can check different preprocessing techniques and architecture, such as a multilabel approach to improve the model performance.

References

[1] Neelu K., S. Bhattacharya, “Early Detection of DR Using PCA-Firefly Based DL Model”, MDPI Switzerland, 2020.
[2] Lifeng Qiao, Ying Zhu, Hui Zhou, “Diabetic Retinopathy Detection using Prognosis of Microaneurysm and Early Diagnosis System for NonProliferative Diabetic Retinopathy Based on Deep Learning Algorithms”, IEEE Access, 2017.
[3] JIE LI, JIXIANG GUO, “Diagnosis of DR Using DNN”, IEEE, January 2019.
[4] Yung-Hui Li, Na-NingYeh, “Computer Assisted Diagnosis for DR Based on Fundus Images Using Deep CNN”, Hindawi publisher for tech & science journals, 2019.
[5] Joel J, J Kivinen, “DL Fundus Images Analysis of DR and Macular Edema Grading”, Springer, 2019.
[6] Navoneel Chakrabarty, “A Deep Learning Method for the detection of diabetic retinopathy”, IEEE, 2018.
[7] SehrishQummar, FiazGul Khan, Sajid Shah, Ahmad Khan, Shahaboddin Shamshirband, Zia Ur Rehman, Iftikhar Ahmed Khan, WaqasJadoon, “A Deep Learning Ensemble Approach for Diabetic Retinopathy Detection”, IEEE, 2019.
[8] Nikhil M N, Angel Rose A, “Diabetic Retinopathy Stage Classification using CNN”, IRJET, 2019.
[9] Walter T, Klein J-C, Massin P, Erginay A. A contribution of image processing to the diagnosis of diabetic retinopathy-detection of exudates in color
fundus images of the human retina. IEEE Trans Med Imag 2002;21(10):1236–43.
[10] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: European conference on computer vision. Springer; 2014. p. 818–33.