OCULAR DISEASE RECOGNITION USING CNN

A PROJECT REPORT

Submitted by

ADITYA MUKHERJEE [RA1911003010210]

Under the guidance of

Dr P.Madhavan

(Associate Professor, Department of Department of Computing Technologies)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
Deemed to be University u/s 3 of UGC Act, 1956

S.R.M. Nagar, Kattankulathur, Kancheepuram District

October 2021

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that this project report titled "**OCULAR DISEASE RECOGNITION USING CNN** " is the bonafide work of "**ADITYA MUKHERJEE [RA1911003010210], , , ,** ", who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE	**SIGNATURE**
Dr P.Madhavan **GUIDE** Associate Professor Dept. of Department of Computing Technologies	**HEAD OF THE DEPARTMENT** Dept. of Computer Science and Engineering
Signature of the Internal Examiner	Signature of the External Examiner

ABSTRACT

.

Approximately 12 million people every year suffer from eye diseases like macular degeneration, glaucoma, and cataracts Some of these are not even in detected and if left alone it can be fatal sometimes .The equipment required for the detection can also be costly also weather the doctor remained equipped to decipher is a question too .Early ocular disease detection is an economic and effective way to prevent blindness caused by diabetes, glaucoma, cataract, age-related macular degeneration (AMD), and many other diseases. According to World Health Organization (WHO) at present, at least 2.2 billion people around the world have vision impairments, of whom at least 1 billion have a vision impairment that could have been prevented. Rapid and automatic detection of diseases is critical and urgent in reducing the ophthalmologist's workload and prevents vision damage of patients. Computer vision and deep learning can automatically detect ocular diseases after providing high-quality medical eye fundus images. Thus my answer to this problem statement was to create a Image classifier using pytorch, that is trained on over 2000 images of eye problems and can then help you identify if you have any problem in your eye or not from just a click. This projects includes the pipeline and also the code on how to clean the data and use it to make an accurate assumptions.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my guide, Dr P.Madhavan his valuable guidance, consistent encouragement, personal caring, timely help and providing me with an excellent atmosphere for doing research. All through the work, in spite of his busy schedule, he has extended cheerful and cordial support to me for completing this research work.

Author

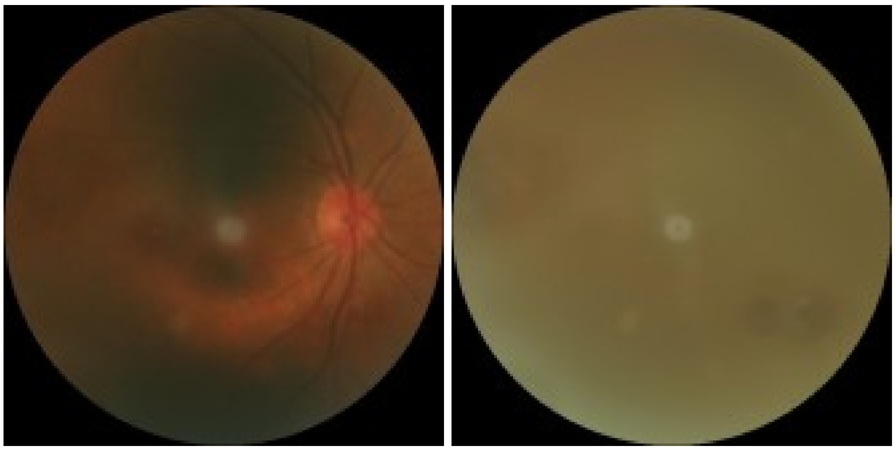
TABLE OF CONTENTS

CHAPTER 1

INTODUCTION

1.1 Title Selection

For the title selection, I had chosen a title in relation to AI ML as I aspire to make a career out in that field. I have been working with various AI ML tools in past and have been improving my skills continuously and felt that this was a good time to showcase and also hone my skills. I looked through various datasets and various use cases of AI/ML and after through research I finalized on ODIR dataset from Kaggle[2].



1.2 Motivation

The motivation to select this topic was to use AI/ML is such a filed where it can affect lives of thousands of people directly. Medical field is such a field where there are problems like availability of Doctors in remote areas like villages and also the human error aspect. The Doctors that are sometimes available in these remote areas are not properly equipped or not qualified enough. This introduces a level of error in the final decision which can prove to be fatal

Here is where the use of AI/ML comes in , we can create various different models that can trained and do the work of doctors in remote areas. Devices like mobiles with internet connections are available in most of the places and this can used to our advantage. With connecting people to over the internet to these various software can be a revolutionary idea and can save people lives who don't have access to doctors or high level equipment.

1.3 Objective

The Objective of this project was to create a pipeline where we could analyze the eye disease scan available and be able to recognize various diseases. This will help in the availability if a robust scalable model that can be used to create software that can reach remote places where the doctors are not available or enough amenities are not present to have a correct and efficient check up

Another objective of this project was to create an elaborative yet simple and self explanatory code, so that anyone who uses this code can even learn from it and also make modification easily.

CHAPTER 2

LITERATURE SURVEY

2.1 AI/ML Use-case

To start the project, the first step was to research the use cases of AI/ML. As I had selected on doing a project on AI/ML I had to go thought various articles and this article by Forbes[1], gave me various different ideas on how to and where AI/ML were used and I selected Health care as my topic .

2.2 Image Segmentation techniques

In this paper[3], it investigates evaluation of networks of increasing depth using an architecture with very small (3x3) convolution filters, which shows that a significant improvement on the prior-art configurations can be achieved by pushing the depth to 16-19 weight layers. But as this paper is a bit outdated a more robust CNN could have been developed with a better architecture.

After going through this paper,it came to light there was a place where it could have been improved in providing more depth in layers and also bigger convolution layers.

2.3 Dataset analysis

In this paper[3],the data set used was ODIR. After going through kaggle there were various data sets that were available . But All the data-sets while having good images did not have the volume that the ODIR data-set had and as for training and accuracy of the CNN model accuracy is an important factor and thus it was decided that the ODIR data-set would be used.

2.4 Model Selection

In this paper[4],its is clearly visible they have used predefined models that are available in the net to train and have also combined various data-sets and passed on to doctors for classification,But as we do not have the same resources available we have to use a more student friendly approach. For this we have created a more simpler and customize able model with a more student friendly pipeline ,where the data-set can be pre-processed in a more friendly manner

CHAPTER 3

DATASET SELECTION

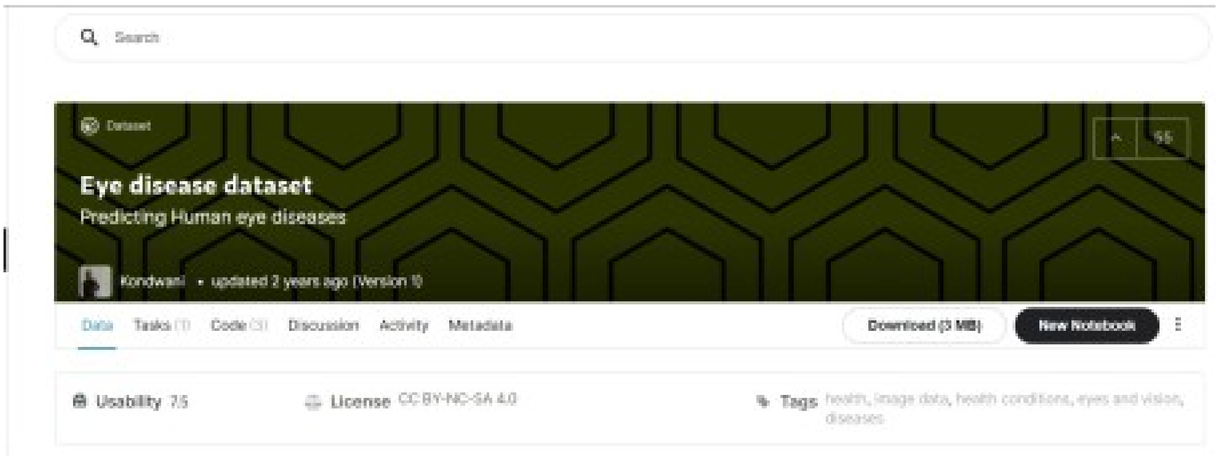
There are different modules in this model, in this chapter we will describe the various datasets available and discuss why which was used

3.1 Eye disease dataset

This is one of the first data set that will be available online this contains around 5000 images . They contain diseases like Bulging Eyes,Cataracts,Crossed Eyes,Glaucoma and Uveitis

while this data set had good amount of images it contained only single classification of eyes ,the is one eye had only one disease per eyes. And while that is still use full , my use case was of multi classification.

Also the images here did not seem to be from a hospital , and hence a more approved dataset would be better


Dataset

Eye disease dataset

Predicting Human eye diseases

Kondwani • updated 2 years ago (Version 1)

Data Tasks (1) Code (3) Discussion Activity Metadata Download (3 MB) New Notebook

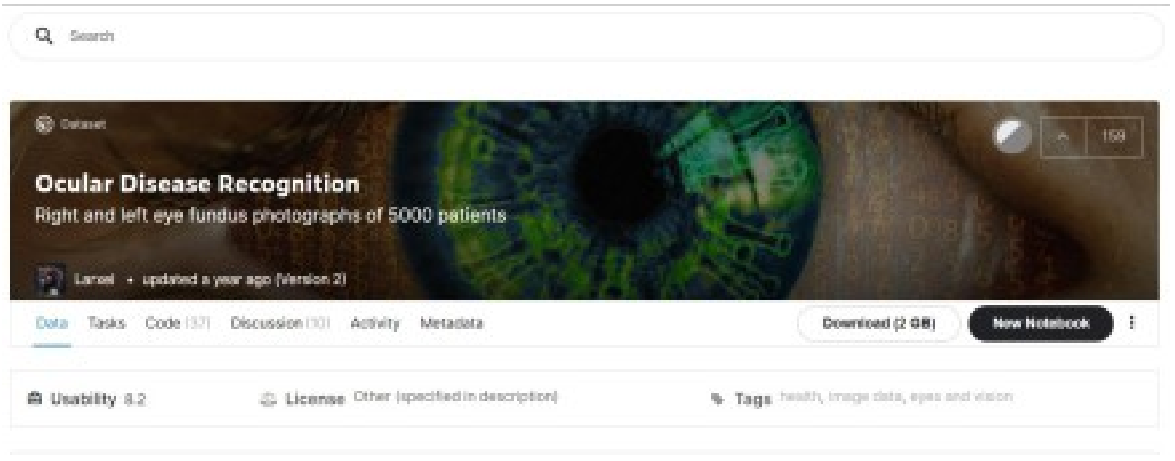
Usability 7.5 License CC BY-NC-SA 4.0 Tags health, image data, health conditions, eyes and vision, diseases

3.2 Ocular Disease Recognition

It is a Right and left eye fundus photographs of 5000 patients. Ocular Disease Intelligent Recognition (ODIR) is a structured ophthalmic database of 5,000 patients with age, color fundus photographs from left and right eyes and doctors' diagnostic keywords from doctors.

This dataset is meant to represent "real-life" set of patient information collected by Shanggong Medical Technology Co., Ltd. from different hospitals/medical centers in China. In these institutions, fundus images are captured by various cameras in the market, such as Canon, Zeiss and Kowa, resulting into varied image resolutions. Annotations were labeled by trained human readers with quality control management. They classify patient into eight labels including:

- Normal (N),
- Diabetes (D),
- Glaucoma (G),
- Cataract (C),
- Age related Macular Degeneration (A),
- Hypertension (H),
- Pathological Myopia (M),
- Other diseases/abnormalities (O)



Ocular Disease Recognition

Right and left eye fundus photographs of 5000 patients

Larxel • updated a year ago (Version 2)

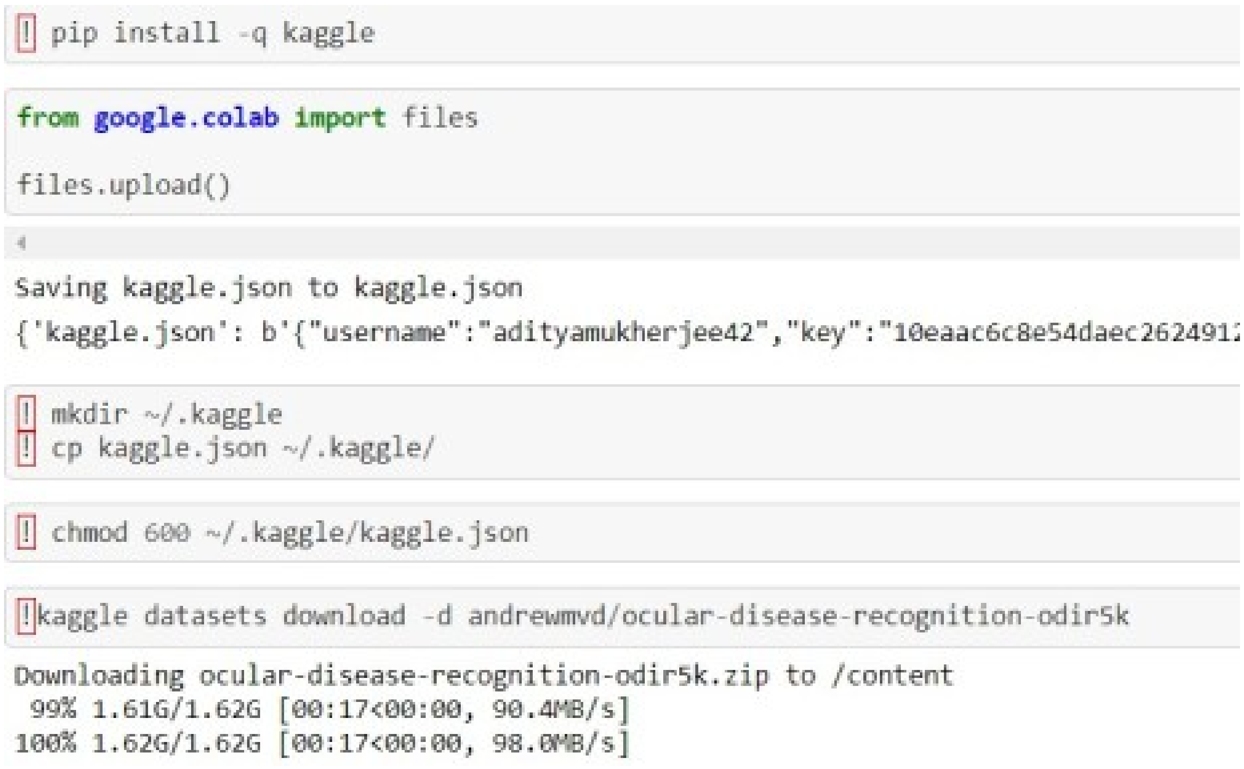
Usability 8.2 License Other (specified in description) Tags health, image data, eyes and vision

CHAPTER 4

DATSET PREPROCESSING

As I don’t have a proper Graphic card to work with i have used google colab as my IDE.

The initial step is to download the data set from kaggle ,for that first i have to create a json file from my kaggle account. Then create a kaggle folder where the json file will be stored . Then using chmod we change the user status of the json file



```
! pip install -q kaggle
```

```
from google.colab import files

files.upload()
```

```
Saving kaggle.json to kaggle.json
{'kaggle.json': b'{"username":"adityamukherjee42","key":"10eaac6c8e54daec262491
```

```
! mkdir ~/.kaggle
! cp kaggle.json ~/.kaggle/
```

```
! chmod 600 ~/.kaggle/kaggle.json
```

```
!kaggle datasets download -d andrewmvd/ocular-disease-recognition-odir5k
```

```
Downloading ocular-disease-recognition-odir5k.zip to /content
 99% 1.61G/1.62G [00:17<00:00, 90.4MB/s]
100% 1.62G/1.62G [00:17<00:00, 98.0MB/s]
```

Now after this we create two folders that are the testing and training folder. The images will be segregated so that we can different images to test and train the data.Then we will import all the required modules.Then a list was created with all the diseases and appropriate labels were added to them.

Images were resized while training the model as it could prevent time-consuming images resizing at once. Unfortunately, it was not a good decision, execution of one epoch could take even 15 minutes, so I created another function to resize images before creating the pytorch dataloader object. As a result, data are resized only once and saved to a different directory, thus

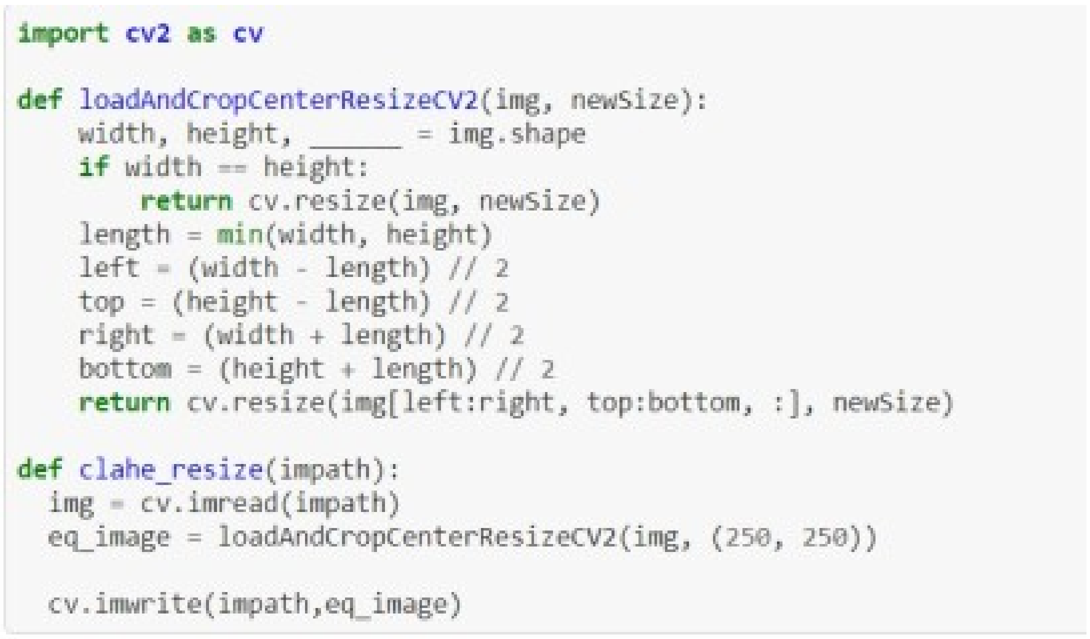
I could experiment with different training approaches using much faster training execution. Initially, all images were resized to 32x32 pixels size, but quickly I realized that compressing to such a low size, even though it speeds up the training process significantly, loses a lot of important image information, thus accuracy was very low. After several experiments I found that size of 250x250 pixels was the best in terms of compromising training speed and accuracy metrics, thus I kept this size on all images for all further experiments

Secondly, images are labeled. There is a problem with images annotations in the data.csv file because the labels relate to both eyes (left and right) at once whereas each eye can have a different disease. For example, if the left eye has a cataract and right eye has normal fundus, the label would be a cataract, not indicating a diagnosis of the right eye. Fortunately, the diagnostic keywords relate to a single eye. Dataset was created in a way to provide to the model as input both left and right eye images and return overall (for both eyes) cumulated diagnosis, neglecting the fact that one eye can be healthy. In my opinion, it does not make sense from a perspective of a final user of such a model, and it is better to get predictions separately for each eye, to know for example which eye should be treated. So, I enriched the dataset by creating a mapping between the diagnostic keywords to disease labels. This way, each eye is assigned to a proper label. Label information is added by renaming image names, and more specifically, by adding to the image file name one or more letters corresponding to the specific diseases. I applied this solution because this way I do not need to store any additional data frame with all labels. Renaming files is a very fast operation . Moreover, some images that had annotations not related to the specific disease itself, but to the low quality of the image, like “lens dust” or “optic disk photographically invisible” are removed from the dataset as they do not play a decisive role in determining patient’s disease.

Thirdly, the validation set is created by randomly selecting 30 percent of all available images. I chose this because this dataset is relatively small (only 7000 images in total), but I wanted to make my validation representative enough, not to have a bias when evaluating model, related to the fact, that many image variants or classes could not have their representation in the validation set. The ODIR dataset provides testing images, but unfortunately, no labeling information is provided to them in the data.csv file, thus I could not use available testing images to evaluate the model.

Now comes Image Augmentation , where we will use opencv module to augment the images . Now the question is what is Image Augmentation,Image augmentation artificially creates training images through different ways of processing or combination of multiple processing, such as random rotation, shifts, shear and flips, etc.

Now here we use two functions that was created by me so that the images could be Augmented(upto certain extent) and stored by me below is the image showing the two function

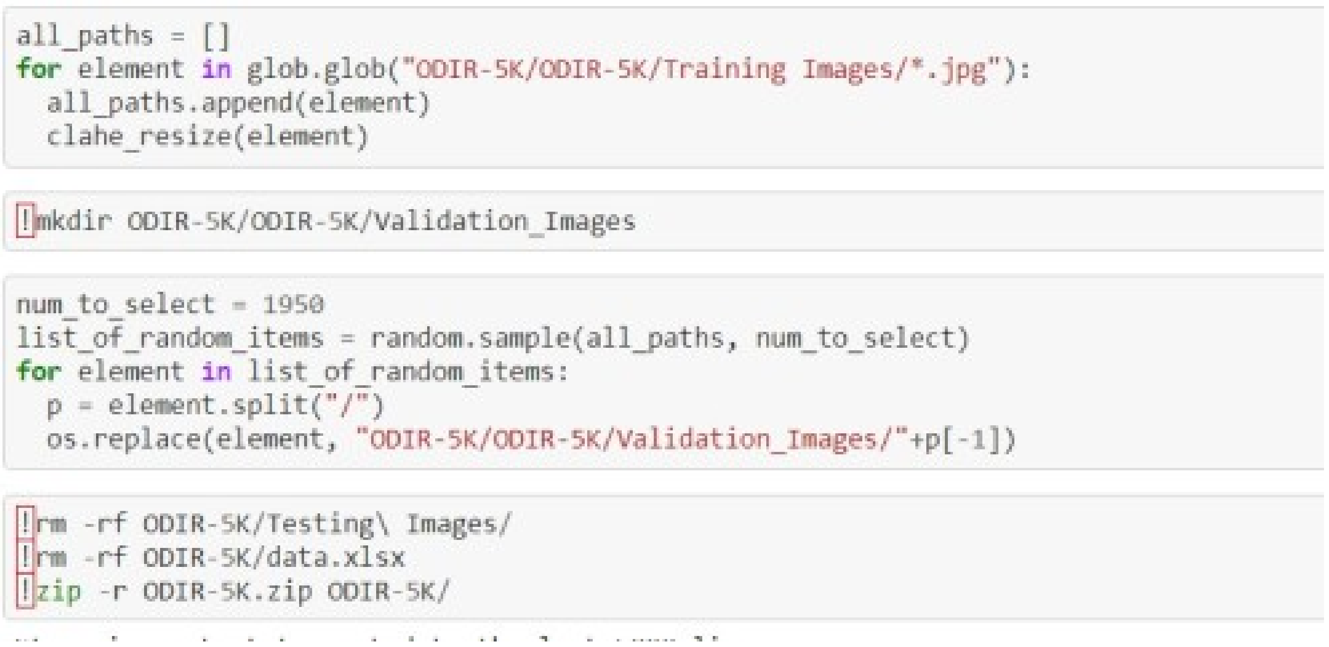


```
import cv2 as cv

def loadAndCropCenterResizeCV2(img, newSize):
    width, height, ______ = img.shape
    if width == height:
        return cv.resize(img, newSize)
    length = min(width, height)
    left = (width - length) // 2
    top = (height - length) // 2
    right = (width + length) // 2
    bottom = (height + length) // 2
    return cv.resize(img[left:right, top:bottom, :], newSize)

def clahe_resize(impath):
  img = cv.imread(impath)
  eq_image = loadAndCropCenterResizeCV2(img, (250, 250))

  cv.imwrite(impath,eq_image)
```

The function loadAndCropCenterResize takes in two parameter that is the image and the new size we want to convert the image.In this function we take the images and the crop it from the center using the minimum of the width and length and using that as a reference and the resizes the image

the calhe resize is a function where the image is read and the iterated

And then to end it , 1950 images were selected in random and put into to a validation folder which will be used later while we train the model,This folder was the zipped and then saved in the google drive folder for later use.



```
all_paths = []
for element in glob.glob("ODIR-5K/ODIR-5K/Training Images/*.jpg"):
  all_paths.append(element)
  clahe_resize(element)
```

```
!mkdir ODIR-5K/ODIR-5K/Validation_Images
```

```
num_to_select = 1950
list_of_random_items = random.sample(all_paths, num_to_select)
for element in list_of_random_items:
  p = element.split("/")
  os.replace(element, "ODIR-5K/ODIR-5K/Validation_Images/"+p[-1])
```

```
!rm -rf ODIR-5K/Testing\ Images/
!rm -rf ODIR-5K/data.xlsx
!zip -r ODIR-5K.zip ODIR-5K/
```

Now the step of image augmentation and preprocessing has been completed now we will then move on to the next module of this project that is the model creation

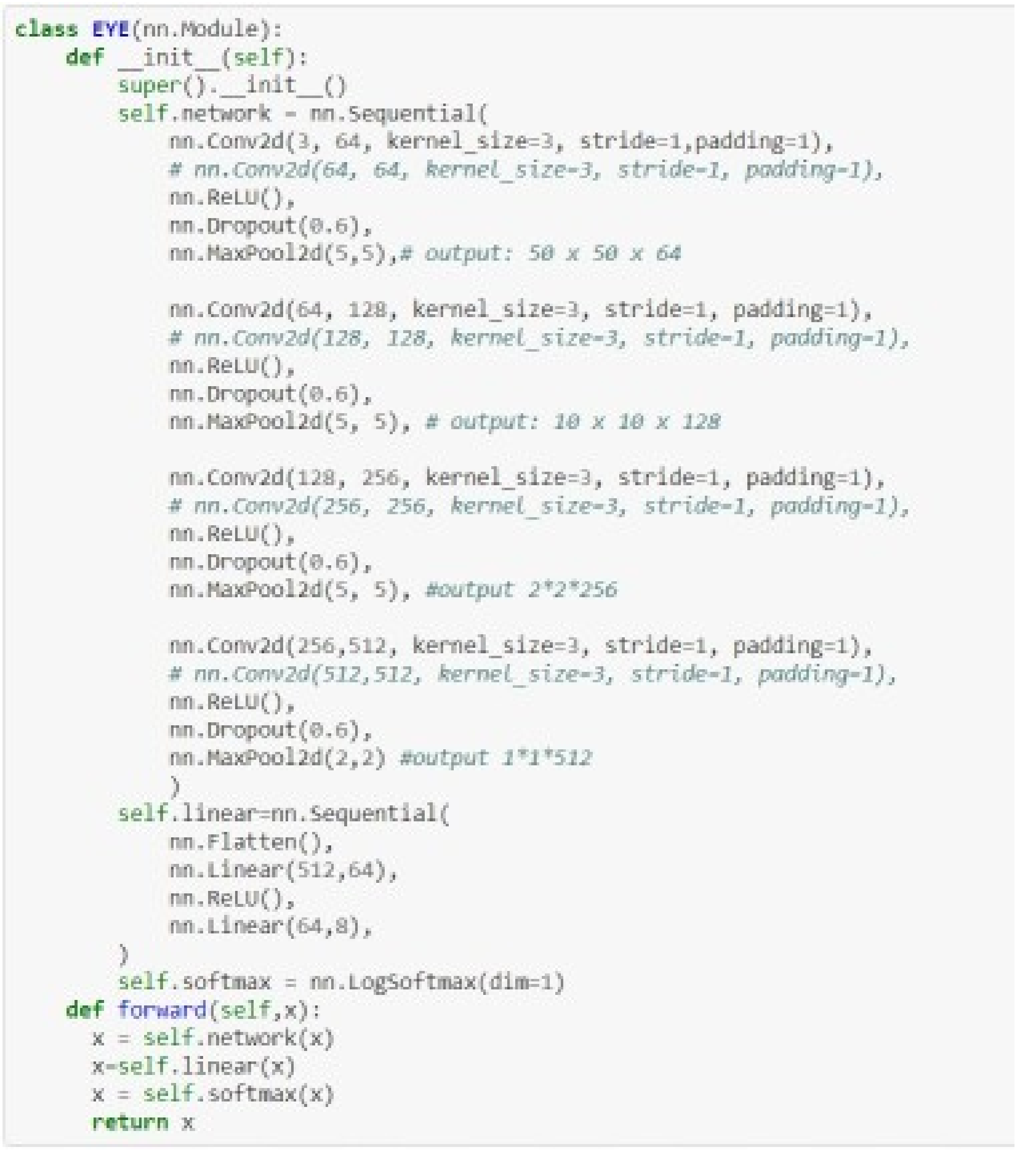
CHAPTER 5

MODEL CREATION

Today, with the increasing volatility, necessity and applications of artificial intelligence, fields like machine learning, and its subsets, deep learning and neural networks have gained immense momentum. The training needs softwares and tools like classifiers, which feed huge amount of data, analyze them and extract useful features. The intent of the classification process is to categorize all pixels in a digital image into one of several classes. Normally, multi-spectral data are used to perform the classification and, indeed, the spectral pattern present within the data for each pixel is used as the numerical basis for categorization. The objective of image classification is to identify and portray, as a unique gray level (or color), the features occurring in an image in terms of the object these features actually represent on the ground. Image classification is perhaps the most important part of digital image analysis. Classification between objects is a complex task and therefore image classification has been an important task within the field of computer vision. Image classification refers to the labelling of images into one of a number of predefined classes. There are potentially n number of classes in which a given image can be classified. Manually checking and classifying images could be a tedious task especially when they are massive in number and therefore it will be very useful if we could automate this entire process using computer vision. The advancements in the field of autonomous driving also serve as a great example of the use of image classification in the real-world. The applications include automated image organization, stock photography and video websites, visual search for improved product discoverability, large visual databases, image and face recognition on social networks, and many more; which is why, we need classifiers to achieve maximum possible accuracy.

5.1 Model Creation

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery. Input layer takes 250x250 RGB images. The first 2D convolution layer shifts over the input image using a window of the size of 3x3 pixels to extract features and save them on a multi-dimensional array, in my example number of filters for the first layer equals 64, so to (250, 250, 64) size cube. After each convolution layer, a rectified linear activation function (ReLU) is applied. Activation has the authority to decide if neuron needs to be activated or not measuring the weighted sum. ReLU returns the value provided as input directly, or the value 0.0 if the input is 0.0 or less. Because rectified linear units are nearly linear, they preserve many of the properties that make linear models easy to optimize with gradient-based methods. They also preserve many of the properties that make the linear model generalize well. To progressively reduce the spatial size of the input representation and minimize the number of parameters and computation in the network max-pooling layer is added. In short, for each region represented by the filter of a specific size, in my example it is (5, 5), it will take the max value of that region and create a new output matrix where each element is the max of the region in the original input. To avoid overfitting problems, two dropouts of 60 percent layers were added. Several batch normalization layers were added to the model. Finally, the "cube" is flattened. No fully connected layers are implemented to keep the simplicity of the network and keep training fast. The last layer is 8 dense because 8 is the number of labels (diseases) present in the dataset. Since we are facing multi-label classification (data sample can belong to multiple instances) softmax activation function is applied to the last layer. The softmax function converts each score to the final node between 0 to 1, we must go with the binary cross-entropy loss. The selected optimizer is Adam with a low learning rate of 0.0001 because of the overfitting problems that I was facing during the training.



```python
class EYE(nn.Module):
    def __init__(self):
        super().__init__()
        self.network = nn.Sequential(
            nn.Conv2d(3, 64, kernel_size=3, stride=1,padding=1),
            # nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Dropout(0.6),
            nn.MaxPool2d(5,5),# output: 50 x 50 x 64

            nn.Conv2d(64, 128, kernel_size=3, stride=1, padding=1),
            # nn.Conv2d(128, 128, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Dropout(0.6),
            nn.MaxPool2d(5, 5), # output: 10 x 10 x 128

            nn.Conv2d(128, 256, kernel_size=3, stride=1, padding=1),
            # nn.Conv2d(256, 256, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Dropout(0.6),
            nn.MaxPool2d(5, 5), #output 2*2*256

            nn.Conv2d(256,512, kernel_size=3, stride=1, padding=1),
            # nn.Conv2d(512,512, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Dropout(0.6),
            nn.MaxPool2d(2,2) #output 1*1*512
            )
        self.linear=nn.Sequential(
            nn.Flatten(),
            nn.Linear(512,64),
            nn.ReLU(),
            nn.Linear(64,8),
        )
        self.softmax = nn.LogSoftmax(dim=1)
    def forward(self,x):
      x = self.network(x)
      x=self.linear(x)
      x = self.softmax(x)
      return x
```

CHAPTER 6

TRAINING

The number of pictures in this data set is very limited.Thus while training it becomes of utmost importance to train it in a way that we can make the most use of the data set. Now lets first look what training is in ML and then which techniques were used here

6.1 What is Training

Training a model simply means learning (determining) good values for all the weights and the bias from labeled examples. In supervised learning, a machine learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss; this process is called empirical risk minimization.

Loss is the penalty for a bad prediction. That is, loss is a number indicating how bad the model's prediction was on a single example. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater. The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.

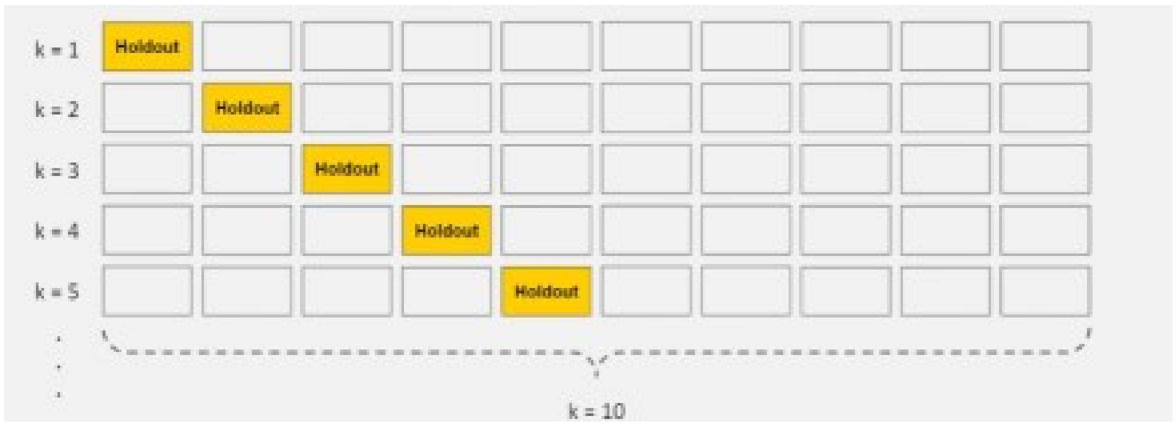
The techniques used here is cross validation , in the next section we will see what is cross validation

6.2 Cross-Validation

Cross-validation is a resampling technique for evaluating machine learning models on a small sample of data.

The process includes only one parameter, k, which specifies the number of groups into which a given data sample should be divided. As a result, the process is frequently referred to as k-fold cross-validation. When a specific value for k is chosen, it can be substituted for k in the model's reference, for example, k=10 for 10-fold cross-validation.

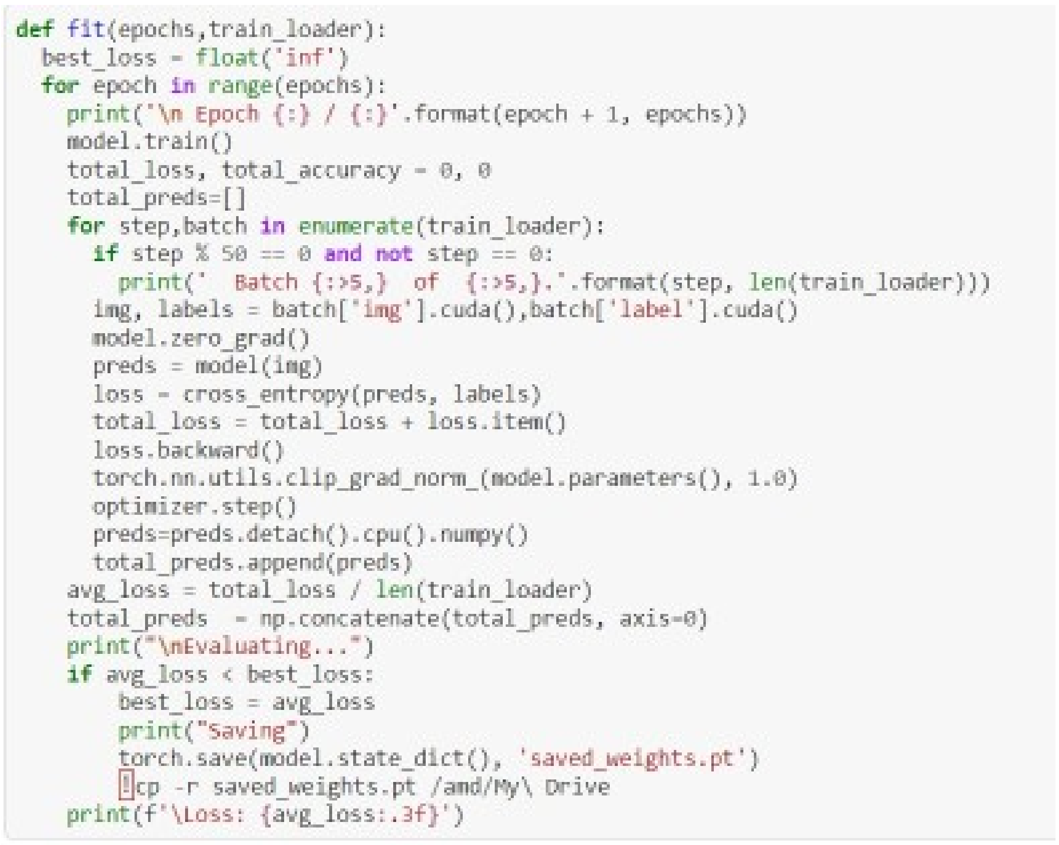
Cross-validation is a technique used in applied machine learning to estimate a machine learning model's skill on unknown data. That is, to use a small sample to forecast how the model will perform in general when used to make decisions.



6.3 Why Cross validation

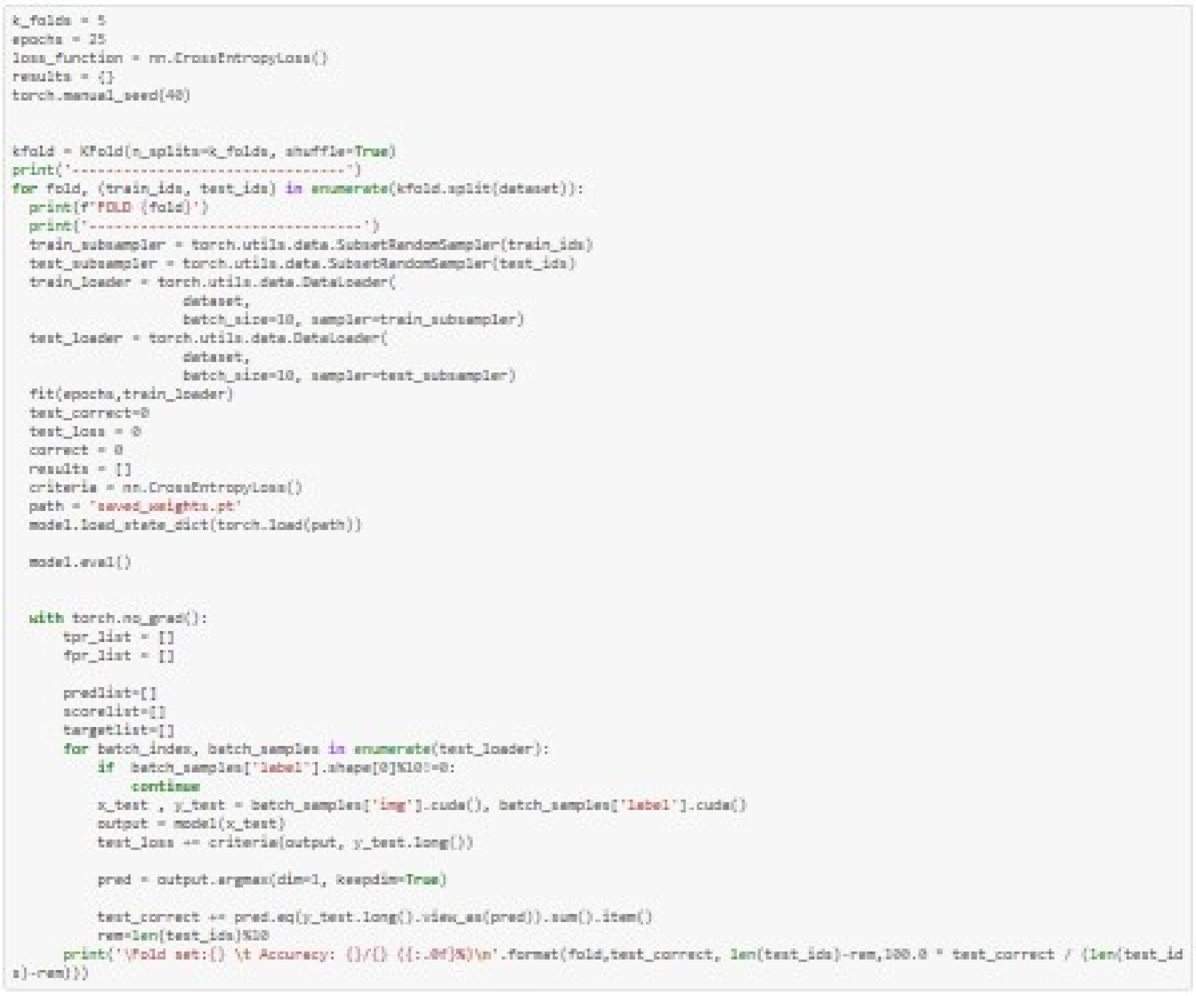
As mentioned above , the data here is very limited and thus cross validation gives a good solution to reuse data in such a manner that we can have more accurate results.As mentioned above we divide the dataset into two parts one part in used for training and the other part is used for testing . Here the training part is divided into 5 k folds . Here 5 k folds mean the the training dataset will be divided into 5 parts and one of the 5 parts will be kept for testing in the k fold.

This will then increase the amount of time the model will be trained on the dataset and thus will improve the accuracy if the dataset



```
def fit(epochs,train_loader):
  best_loss = float('inf')
  for epoch in range(epochs):
    print('\n Epoch {:} / {:}'.format(epoch + 1, epochs))
    model.train()
    total_loss, total_accuracy = 0, 0
    total_preds=[]
    for step,batch in enumerate(train_loader):
      if step % 50 == 0 and not step == 0:
        print('  Batch {:>5,}  of  {:>5,}.'.format(step, len(train_loader)))
      img, labels = batch['img'].cuda(),batch['label'].cuda()
      model.zero_grad()
      preds = model(img)
      loss = cross_entropy(preds, labels)
      total_loss = total_loss + loss.item()
      loss.backward()
      torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
      optimizer.step()
      preds=preds.detach().cpu().numpy()
      total_preds.append(preds)
    avg_loss = total_loss / len(train_loader)
    total_preds  = np.concatenate(total_preds, axis=0)
    print("\nEvaluating...")
    if avg_loss < best_loss:
        best_loss = avg_loss
        print("Saving")
        torch.save(model.state_dict(), 'saved_weights.pt')
        !cp -r saved_weights.pt /amd/My\ Drive
    print(f'\Loss: {avg_loss:.3f}')
```

The above image is the code , where the model is fitted , i.e how the model will be trained . here first the image and the label are now put on to the GPU. Then the image is passed through the model ,and then the output is passed through the cross entropy function where the loss is counted and then we use back propagation where we try to reduce the total loss that has inc cur in the model while predicting.The total loss is being stored to count the avg loss of the model and this is compared in every epoch and if the average loss of that epoch is less than the previous best loss the weights are stored .



```
k_folds = 5
epochs = 25
loss_function = nn.CrossEntropyLoss()
results = {}
torch.manual_seed(40)


kfold = KFold(n_splits=k_folds, shuffle=True)
print('--------------------------------')
for fold, (train_ids, test_ids) in enumerate(kfold.split(dataset)):
  print(f'FOLD {fold}')
  print('--------------------------------')
  train_subsampler = torch.utils.data.SubsetRandomSampler(train_ids)
  test_subsampler = torch.utils.data.SubsetRandomSampler(test_ids)
  train_loader = torch.utils.data.DataLoader(
                    dataset,
                    batch_size=10, sampler=train_subsampler)
  test_loader = torch.utils.data.DataLoader(
                    dataset,
                    batch_size=10, sampler=test_subsampler)
  fit(epochs,train_loader)
  test_correct=0
  test_loss = 0
  correct = 0
  results = []
  criteria = nn.CrossEntropyLoss()
  path = 'saved_weights.pt'
  model.load_state_dict(torch.load(path))

  model.eval()


  with torch.no_grad():
      tpr_list = []
      fpr_list = []

      predlist=[]
      scorelist=[]
      targetlist=[]
      for batch_index, batch_samples in enumerate(test_loader):
          if  batch_samples['label'].shape[0]%10!=0:
              continue
          x_test , y_test = batch_samples['img'].cuda(), batch_samples['label'].cuda()
          output = model(x_test)
          test_loss += criteria(output, y_test.long())

          pred = output.argmax(dim=1, keepdim=True)

          test_correct += pred.eq(y_test.long().view_as(pred)).sum().item()
          rem=len(test_ids)%10
      print('\Fold set:{} \t Accuracy: {}/{} ({:.0f}%)\n'.format(fold,test_correct, len(test_ids)-rem,100.0 * test_correct / (len(test_id
s)-rem)))
```

The above code if the K-fold code where the first number of epochs,number of folds and the loss function is specified . the using the Kfold function provided by the sklearn module a kfold object is created . This object will help us split the dataset into the training and testing part within the dataset. After creating the split the train id and test id is passed and then then the model is trained for 25 epochs using the fitting function and the after 25 epochs the model is tested on an part of the dataset and thus accuracy of the dataset is calculated after every epoch

CHAPTER 7

TESTING THE MODEL

Any project should include an evaluation of your machine learning method. When employing a metric like accuracy score, your model may produce satisfactory results, but when compared to other metrics like logarithmic loss or any other metric, it may produce terrible results. Most of the time, we utilise classification accuracy to assess our model's performance.

Below will be the metrics used to judge this model

7.1 Accuracy

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

7.2 Precision

It is the number of correct positive results divided by the number of positive results predicted by the classifier.

7.3 F1-Score

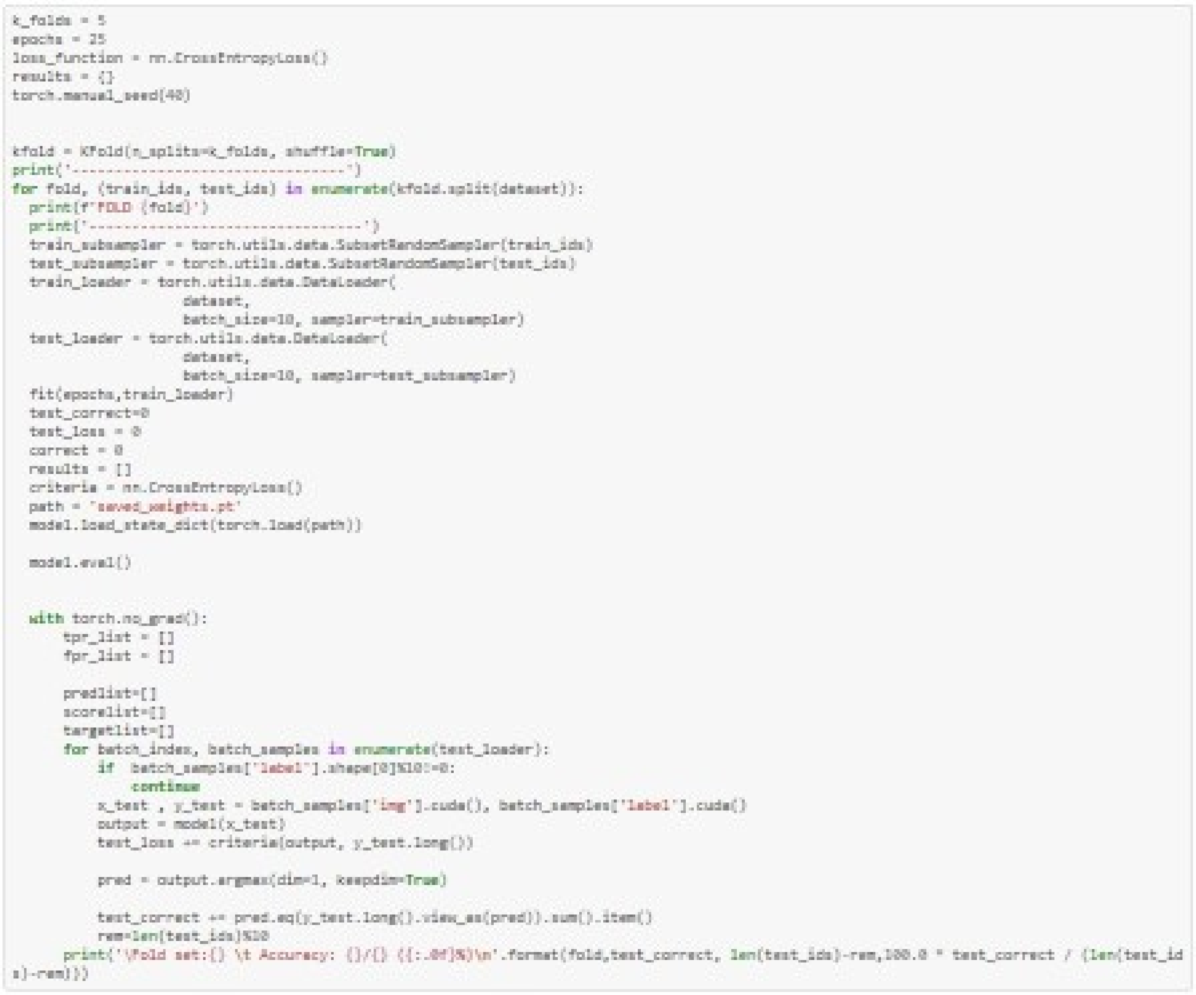
F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

7.4 Recall

It is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive).

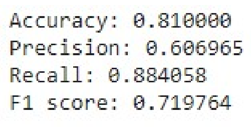
7.5 Code

Now below is the code where we can see the testing code that was used . Two arrays were created one was a prediction list and the other was a target list and then using sklearn modules provided by us and thus the metrics counted



```
k_folds = 5
epochs = 25
loss_function = nn.CrossEntropyLoss()
results = {}
torch.manual_seed(40)


kfold = KFold(n_splits=k_folds, shuffle=True)
print('--------------------------------')
for fold, (train_ids, test_ids) in enumerate(kfold.split(dataset)):
  print(f'FOLD {fold}')
  print('--------------------------------')
  train_subsampler = torch.utils.data.SubsetRandomSampler(train_ids)
  test_subsampler = torch.utils.data.SubsetRandomSampler(test_ids)
  train_loader = torch.utils.data.DataLoader(
                    dataset,
                    batch_size=10, sampler=train_subsampler)
  test_loader = torch.utils.data.DataLoader(
                    dataset,
                    batch_size=10, sampler=test_subsampler)
  fit(epochs,train_loader)
  test_correct=0
  test_loss = 0
  correct = 0
  results = []
  criteria = nn.CrossEntropyLoss()
  path = 'saved_weights.pt'
  model.load_state_dict(torch.load(path))

  model.eval()


  with torch.no_grad():
      tpr_list = []
      fpr_list = []

      predlist=[]
      scorelist=[]
      targetlist=[]
      for batch_index, batch_samples in enumerate(test_loader):
          if  batch_samples['label'].shape[0]%10!=0:
              continue
          x_test , y_test = batch_samples['img'].cuda(), batch_samples['label'].cuda()
          output = model(x_test)
          test_loss += criteria(output, y_test.long())

          pred = output.argmax(dim=1, keepdim=True)

          test_correct += pred.eq(y_test.long().view_as(pred)).sum().item()
          rem=len(test_ids)%10
      print('\Fold set:{} \t Accuracy: {}/{} ({:.0f}%)\n'.format(fold,test_correct, len(test_ids)-rem,100.0 * test_correct / (len(test_id
s)-rem)))
```

7.6 Final model metrics



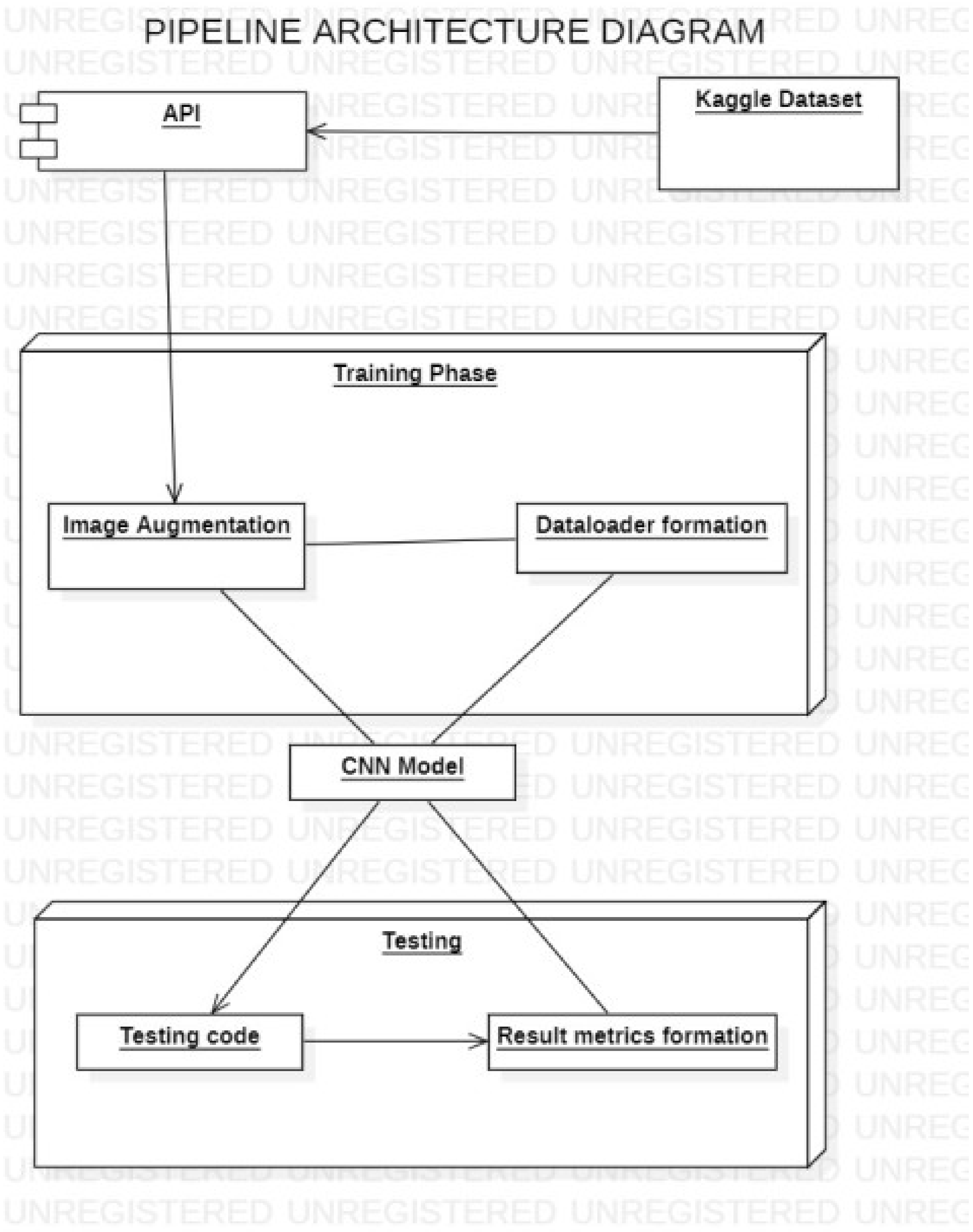
```
Accuracy: 0.810000
Precision: 0.606965
Recall: 0.884058
F1 score: 0.719764
```

CHAPTER 8

ARCHITECTURE DIAGRAM

The below image is the Architecture diagram. It shows how the image is gathered and then how the data preprocessing and other modules in the pipeline work together . First the Kaggle dataset is accessed using the API instructions that have been provided by Kaggle where we have to create an account and then we have to create a json file which have the kaggle credentials and using them we have to log in . After downloading the data we enter the training phase , the training phase . In the training phase we have two main module , that is the image augmentation and dataloader formation. In image augmentation we have to pass image through various function which will help us elevate the dataset which will improve the result of the model . The data loader is an data function which helps during training and testing as it can help us iterate through our dataset

The next module in our diagram is the testing , here we use the weights that will be saved from the training and put it into the test code. the test code will test the model on the dataset that was splited . and then after testing the result metrics are formed . The result metrics will help us identify weather the model does well or not


PIPELINE ARCHITECTURE DIAGRAM
API
Kaggle Dataset
Training Phase
Image Augmentation
Dataloader formation
CNN Model
Testing
Testing code
Result metrics formation

CHAPTER 9

CONCLUSION

After the completion of training band testing , the model is ready for application . The model has been trained in a robust manner using cross validation and training matrix have given an average results

In the field of medical sciences it is important to have high accuracy as wrong diagnosis can have bad side effects. Also the aim of this project was to create a model that could help eliminate human error by not properly qualified doctors and help the people in remote areas like villages where equipment might not be available to have correct diagnosis , thus accuracy of 81 percent while suffice for now it will have to improve in the future .

Also another aim of this project , was to provide a customise pipeline for anyone who is starting with CNN to work with this model as it can help them increase their knowledge and give them hands on practice . Also having a customisable pipeline gives flexibility for the model to be used in various places

Also to add on , i have learnt many things from this project that i performed.The data preprocessing was the most time consuming step and in my previous projects i have worked with data sets that had preprocessing done on it , and thus much easier to work with , here with the images requiring more amount of image preprocessing as explained above and thus becomes a bit more challenging working with this tools.

Also creating a CNN From scratch makes is a very informative work to do ,as it makes one learn on what different function of different layers perform and combining them with various shapes for the images is always a task that is always required to do and thus ha been done in such a way that it can customised whenever it is warranted

9.1 Future Improvement

- The model that has been used here is quite simple,and thus works well on a small scale . But to scale up this project a more robust and a better model will be required.
- The Accuracy of this model is 0.81 which while for first training can suffice,but for medical usage the accuracy must be increased
- The model currently dose not have any user interface where a person can add an image and the model can give a result , and this while this is a start to make it more reachable to other people and can make use of it

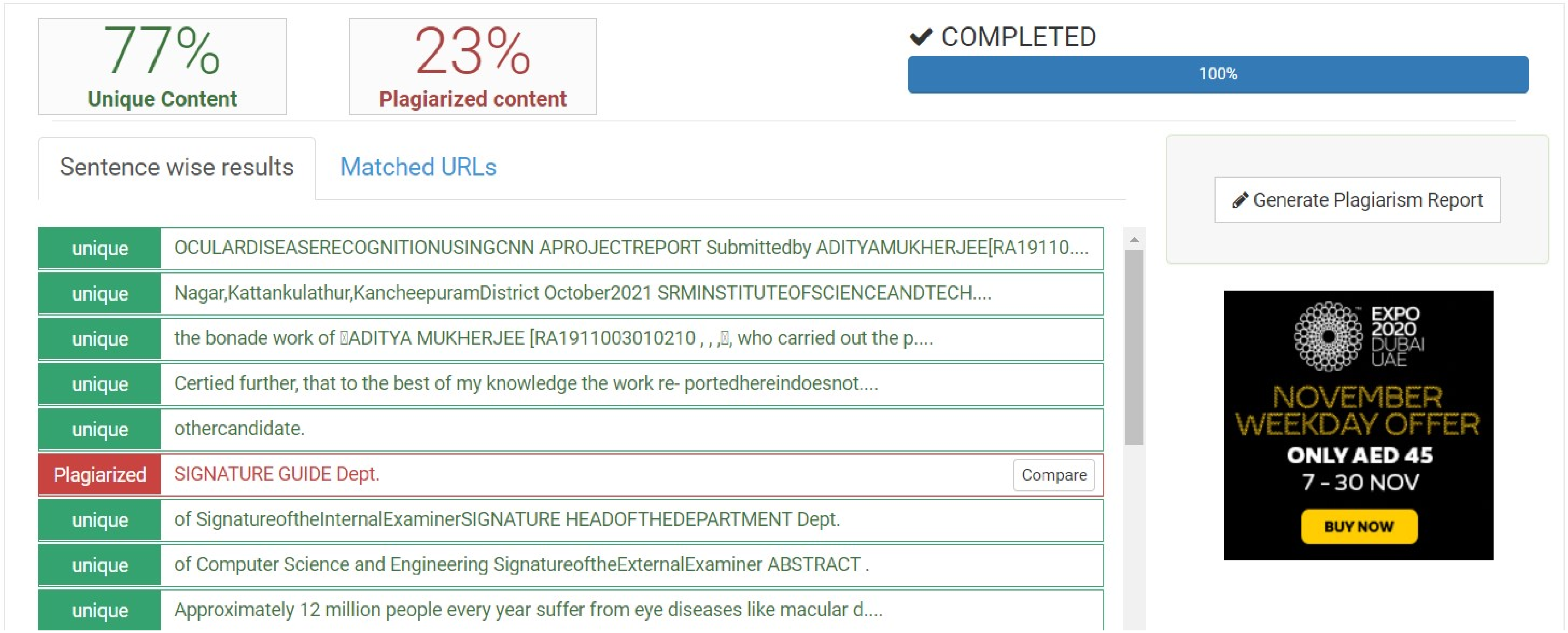
REFERENCES

[1] “https://www.forbes.com/sites/bernardmarr/2016/09/30/what-are-the-top-10-use-cases-for-machine-learning-and-ai/?sh=3715ae6f94c9.

[2] “Ocular disease recognition:right and left eye fundus photographs of 5000 patients.

[3] Karen Simonyan, A. Z. (2014). “Very deep convolutional networks for large-scale image recognition.

[4] Ning Li1, Tao Li1, C. H. K. W. K. a. (2020). “A benchmark of ocular disease intelligent recognition: One shot fo rmulti-diseasedetection.



77%
Unique Content

23%
Plagiarized content

✓ COMPLETED

100%

Sentence wise results | Matched URLs

unique	OCULARDISEASERECOGNITIONUSINGCNN APROJECTREPORT Submittedby ADITYAMUKHERJEE[RA19110....
unique	Nagar,Kattankulathur,KancheepuramDistrict October2021 SRMINSTITUTEOFSCIENCEANDTECH....
unique	the bonade work of ADITYA MUKHERJEE [RA1911003010210 , , ,, who carried out the p....
unique	Certied further, that to the best of my knowledge the work re- portedhereindoesnot....
unique	othercandidate.
Plagiarized	SIGNATURE GUIDE Dept. Compare
unique	of SignatureoftheInternalExaminerSIGNATURE HEADOFTHEDEPARTMENT Dept.
unique	of Computer Science and Engineering SignatureoftheExternalExaminer ABSTRACT .
unique	Approximately 12 million people every year suffer from eye diseases like macular d....