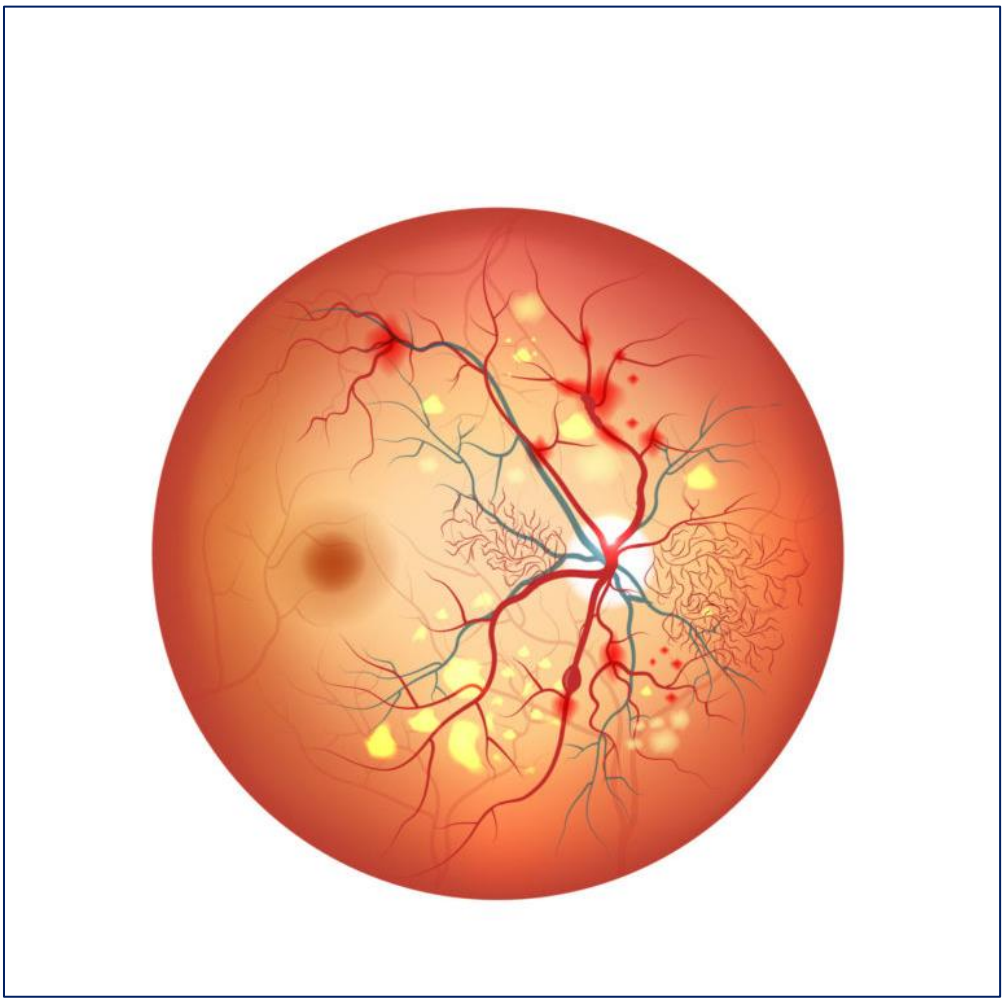
DETECTION OF DIABETIC RETINOPATHY USING CONVOLUTIONAL NEURAL NETWORK



Internship Report

Digital Innovation Lab,

IIM Bangalore

Authored by:

Shambhavi Krishna – 2nd year B. Tech, MIT, Manipal

Apurv Singh – 2nd year B. Tech, MIT, Manipal

IIMB
तेजस्वि नावधीतमस्तु

DIGITAL INNOVATION LAB

Contents

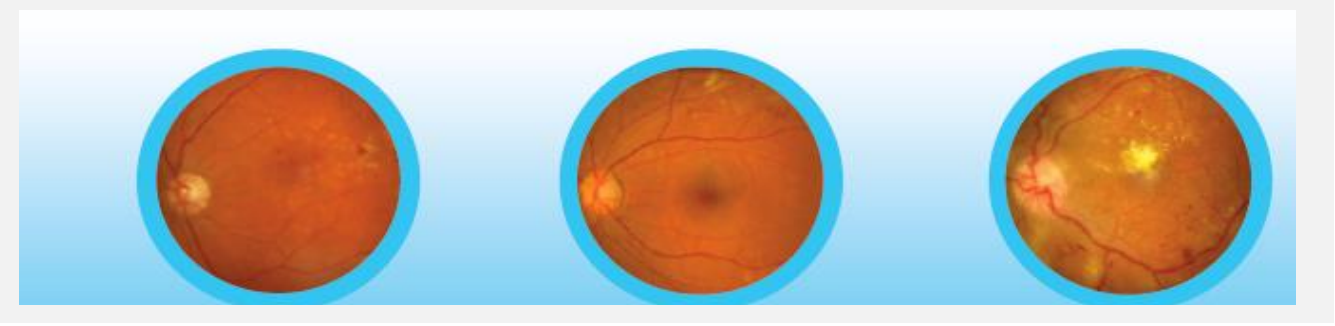
Diabetic Retinopathy

The disease….

D.R. is a disease of the eye that people with diabetes are at a risk of developing which involves the damaging of retina due to abnormal growth of blood vessels in the eye.

If not treated at the right time, it could lead to permanent loss of vision. It is one of the leading causes of blindness in the world, especially in India where for every 10 million patients, there are only 15,000 certified Ophthalmologist, which makes the diagnosis difficult for people living in the remote parts of the country.

There are different grades of DR, based on the severity of the disease. The earlier the detection, the better it is for treatment.



During the course of this project, we aimed to create a solution for this epidemic by making an inference engine with a trained deep learning model, which can take a retinal scan and be able to, with high accuracy, make a diagnosis for each patient for different grades associated with the disease.

DATA

Probably the most important part of the project, the data required to train the model have been taken largely from the open-source data libraries, namely Kaggle, Messidor, IDRiD and Aravind-Kaggle competition.

We also used a fresh data from Vittala International Institute of Ophthalmology, which was collected by our project leader Dr. Shashank Garg as part of his collaboration with the institution for this project.

The main difference between all those open-source data and the new data was that all of the open-source data was pre-processed and clean, whereas with the institute's data, a lot of data cleaning and pre-processing had to be done which included:

- Removing redundant entries
- Removing undiagnosed entries
- Removing null entries
- Anonymizing the entries by renaming it
- Extracting the balanced set to train/test the model with all the possible scenario

THE MODEL

We used Inception V3 (released by people at Google) as our base model which has been already trained on more than a million images from an open-source dataset called ImageNet for weeks to do the basic job of image classification. This process of using an existing model over making a model from scratch is known as Transfer Learning.

Transfer learning is preferred over training a model from scratch because in doing so, we would require a large amount of computation resources which is not easily available. It would also require less images as compared to what could be required while training a model from scratch.

The Inception V3 model has 10 clusters of different layers called 'mixed#'. Since the last few layers would recognize pattern too specific to the data, we ignored the last few clusters of layers and flattened the output of 'mixed7' or 'mixed8' to train our classifier.
We tweaked different parameters and functions of the Inception V3 to suit our objective and data. We changed the optimizer from RMSprop to ADAM.
We also used SGD (Stochastic Gradient Descent) to get different results as these both are considered as the best optimizers for an Image Classification problem.

IMAGE AUGMENTATION

Other than tweaking different parameters and training different layers, we used different augmentation techniques like rotating the image, taking the mirror image, rescaling the size, etc. Augmentation ensures that the model is familiar with all variations of data. It also virtually increases the size of our dataset without having the need of generating new data.

TENSORFLOW

TensorFlow is a framework developed by Google on 9th November 2015. It is written in Python, C++ and Cuda. It supports platforms like Linux, Microsoft Windows, macOS, and Android. TensorFlow provides multiple API's in Python, C++, Java etc. The most widely used API is Python and we are using the same for our project.

The name TensorFlow is derived from the operations, such as adding or multiplying, that artificial neural networks perform on multidimensional data arrays. These arrays are called tensors in this framework.

The benefit of using TensorFlow is its higher lever abstraction and user-friendly support. It is the most widely used framework when it comes to the job of Image Classification as it supports Keras in its new versions which contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier.

ARTEMIS – HPC

We were extremely fortunate to have accessed one of the best computing resources present in the world, Artemis – HPC, which was possible due to the collaboration between DILab and University of Sydney. We had to use shell scripts in a Linux Environment to avail resources of the computers and train our models.

We used FileZilla as our primary FTP Client, PuTTY for using Linux in Windows and Cisco VPN to get into the network of University of Sydney and access the HPC.

ACKNOWLEDGEMENT

We would like to thank Dr. Shashank Garg from the bottom of our hearts for giving us an opportunity to work on this project and for his constant mentoring and guidance. We consider ourselves extremely fortunate to work for him.

The very approachable and helping nature of Dr. Garg was something which enabled us to question and think and act in ways we could never have done otherwise. The kind of access of he provided in terms of data as well as technical source was unparallel. We would really love to work with him again in the future. His patience and constant support were one of the biggest reasons that we were able to come to a meaningful result during the course of this internship.

Learning Experience

During the course of this internship we gained a lot of insights on how to handle, manipulate and process data. We also learned how Neural Networks work, specifically Convolutional Neural Networks and how it uses filters to extract feature and use it to adjust weights which eventually predicts an outcome for a previously unseen data. We also leaned how to use shell scripts in a Linux environment.

The experience at DILab was one of the best experience of our lives, both professionally and personally.

APPENDIX

Code for Labelling Images in different grades

```
import pandas as pd
import os
import shutil


# In[44]:


source_mess = r'E:\Aravind\Images\\'

dest_0 = r'E:\Dataset\IDRiD\grade_0\\'
dest_1 = r'E:\Dataset\IDRiD\grade_1\\'
dest_2 = r'E:\Dataset\IDRiD\grade_2\\'
dest_3 = r'E:\Dataset\IDRiD\grade_3\\'
dest_4 = r'E:\Dataset\IDRiD\grade_4\\'

csv_mess = r'E:\Aravind\train.csv'



df_mess = pd.read_csv(csv_mess, encoding = 'ISO-8859-1')




dir_mess = os.listdir(source_mess)
print(dir_mess)


sub_mess_0 = df_mess[df_mess.grade.isin(['0'])]
sub_mess_1 = df_mess[df_mess.grade.isin(['1'])]
sub_mess_2 = df_mess[df_mess.grade.isin(['2'])]
sub_mess_3 = df_mess[df_mess.grade.isin(['3'])]
sub_mess_4 = df_mess[df_mess.grade.isin(['4'])]

files_mess_0 = sub_mess_0.Image.tolist()
files_mess_1 = sub_mess_1.Image.tolist()
files_mess_2 = sub_mess_2.Image.tolist()
files_mess_3 = sub_mess_3.Image.tolist()
files_mess_4 = sub_mess_4.Image.tolist()
string = '.png'
files_mess_00 =[x + string for x in files_mess_0]
files_mess_11 =[x + string for x in files_mess_1]
files_mess_22 =[x + string for x in files_mess_2]
files_mess_33 =[x + string for x in files_mess_3]
files_mess_44 =[x + string for x in files_mess_4]


# my_list = ['foo', 'fob', 'faz', 'funk']
# string = 'bar'
# my_new_list = [x + string for x in my_list]
```

```
# In[40]:


for file in files_mess_00:
    print(file)
    if file in dir_mess:
        shutil.move(source_mess + file, dest_0)
        print(file + '--->'+dest_0)
print('moved messidor 0')

for file in files_mess_11:
    if file in dir_mess:
        shutil.move(source_mess + file, dest_1)
    else:
        continue
print('moved messidor 1')

for file in files_mess_22:
    if file in dir_mess:
        shutil.move(source_mess + file, dest_2)
print('moved messidor 2')

for file in files_mess_33:
    if file in dir_mess:
        shutil.move(source_mess + file, dest_3)
print('moved messidor 3')



for file in files_mess_44:
    if file in dir_mess:
        shutil.move(source_mess + file, dest_4)
print('moved messidor 4')



# In[47]:


source_mess = r'E:\Aravind\Images\\'
dir_mess = os.listdir(source_mess)
print(dir_mess)
```

Code for anonymizing file names

```
import os
import pandas as pd
import shutil


def read_files(fld):
    for root, dirs, files in os.walk(fld):
        return files

def files_and_extensions(paths):

    extensions = []
    files = []
    for fl in paths:
        extensions.append(os.path.splitext(fl)[1])
        files.append(os.path.splitext(fl)[0])
    return files, extensions

def main():
    input_folder = 'combined'
    output_folder = 'output_files' # directory created if does not exist - if exists
process is stopped
    label = 'test_image' #new file name description

    directory = os.getcwd()
    file_location = os.path.join(directory,input_folder)
    files = read_files(file_location)

    if not os.path.exists(output_folder):
        os.makedirs(output_folder)
    else:
        raise ValueError("the output folder",output_folder,"already exists. Create a
fresh folder or change output folder name")
    if not files:
        raise ValueError("The folder you specified",input_folder," is empty")

    filenames, extensions = files_and_extensions(files)

    count = 0
    new_files = []
    for name, ext, fullname in  zip(filenames, extensions,files):
        count +=1
        new_name = label + str(count) + ext
        new_files.append(new_name)
        new_path = os.path.join(directory,output_folder,new_name)
        old_path = os.path.join(directory,input_folder, fullname)
        shutil.copy(old_path,new_path)
        print("copying file from {} to {} ".format(old_path,new_path))

    reference = pd.DataFrame(zip(files,new_files),columns=('old_path','new_path'))
    reference.to_csv('output_reference.csv')
    print("done")
```