Blindness Detection Using ML

As part of our Data Science Internship at Hamoye.com we were tasked to build a machine learning model that helps in detection of Diabetic Retinotherapy. So the first thing we did was to understand what Diabetic Retinopathy is.

So what is Diabetic Retinopathy?

Diabetes is a chronic, metabolic disease. An individual with diabetes has elevated blood glucose levels that predispose them to serious health problems, including damaged hearts, blood vessels, kidneys, eyes, and nerves.

DR (Diabetic retinopathy) is an issue caused by the degeneration or progressive nature of diabetes mellitus, which damages the blood vessels at the back of the eye.

A person with DR may have little or no symptoms at the beginning. As the condition progresses, they may experience: Sudden and total blindness, Poor night vision , Distorted colour perception, Spots or dark lines appearing in their vision and Fluctuating vision

According to the PubMed report of 2020, the number of middle-agers affected by DR worldwide was 103.12million, and the figure is expected to grow to 160.10million by the year 2045.Studies have shown that Africa is the most affected with the ailment while Central and South America is the least affected continent.

Okay, but why do we need to work on this?

In the absence of early detection and treatment, DR can lead to blindness. Unfortunately, DR is irreversible, and treatments only prolong vision, but early detection and treatment significantly reduce the risk of vision loss. Contrary to computer-aided diagnosis systems, DR retina fundus is best diagnosed with a comprehensive dilated eye exam.

<u>This is how it is traditionally done:</u>

1. Drops are used to dilate (widen) your pupils so that your doctor can see further within your eyes during the test.
2. Your close vision may get clouded when wearing the drops. It can take several hours to completely remove them. During your eye exam, your doctor will examine both the inside and outside of your eyes for any abnormalities.
3. A dye is injected into a vein in your arm after your eyes have been dilated. Pictures are obtained once the dye has circulated through your eye's blood vessels.

4. Blood vessels that are closed, or leaking can be identified using the photos. The test produces cross-sectional retinal pictures that reveal the retina's thickness. It will be possible to determine the amount of fluid that has entered the retinal tissue in this manner.
5. Optical coherence tomography (OCT) scans can be used to track the success of the treatment later on.

Furthermore, the use of Computer Aided Diagnosis (CAD) techniques for Ophthalmic Photography (OP) screening can help detect DR. An Ophthalmic Digital Camera can be used to take the OP, and changes in structural anatomy of the retina can be recorded on the OP.

Because of the large number of diabetic patients, traditional OP screening by experienced ophthalmologists requires a great deal of manpower and finances, making regular OP screening difficult. Therefore, it is of paramount importance to develop automated DR screening techniques based on OP images in order to improve the situation described above.

So have there been any improvements in DR Detection Methods yet?

In recent years, machine learning has been one of the most common techniques that have yielded better results, especially in medical image analysis and classification. Convolutional neural networks are becoming increasingly popular as deep learning methods in medical image analysis, and they are highly effective.

Multiple automated DR detection algorithms, as well as methods for segmenting lesions in Ophthalmic photography, have been developed as a result of developments in computer algorithm development for medical image processing. Grey-level thresholding can be used to segment anatomical structures and diseases from ophthalmic imagery. Despite this, the thresholding results in Ophthalmic photography are inconsistent due to uneven illumination. Instead, a combination of edge detection and mixture models was proposed to detect hard exudates, with a 95% accuracy rate.In addition to region-growing algorithms, adaptive region-growing algorithms, and Bayesian-based approaches have also been developed to segment lesions on ophthalmic photographs with an accuracy rate of 90 percent for DR identification. The colour and form features collected from Ophthalmic photography were utilised to classify microaneurysms and bleeding.

The use of artificial intelligence (AI) has resulted in a significant increase in the early diagnosis of Diabetic Retinopathy. It can be used as a sophisticated sort of triage, where patients with urgent conditions would be seen preferentially by a specialist while others with much milder symptoms can be scheduled to a later date.

Below are some of the more intriguing AI options that have contributed to this technological breakthrough and enhancement in the domain of DR detection.

IDx-DR (IDx Technologies) is an AI system certified by the FDA for the identification of DR. The operator collects two images per eye using a fundus camera, and these images are uploaded to the IDx-DR Client. In less than a minute, IDx-DR analyzes photos for indicators of DR and offers results. Patients who test negative for anything more than mild DR are encouraged to retest in 12 months, while those who test positive for more than mild DR are referred to an eye doctor. In a clinical investigation of 900 individuals with diabetes, IDx-DR achieved 87 percent sensitivity and 90 percent specificity at detecting more than mild DR in fundus pictures.

The EyeArt AI Eye Screening System (Eyenuk) uses artificial intelligence (AI) algorithms to analyse retinal images taken with a fundus camera and detect DR. The system uses fundus images of the eye to perform automated screening in a single office visit, which includes retinal imaging, DR grading based on international standards, and report generation. The DR screening findings are accessible to read and export to a PDF report in less than 60 seconds after the patient's fundus photos have been collected and submitted to the EyeArt AI System.

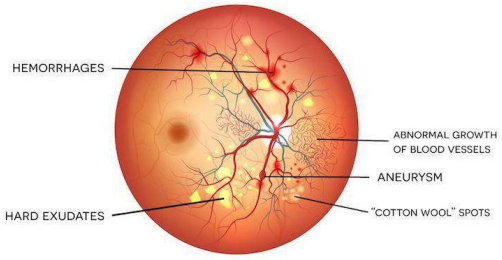
Benefits of Applying These Technological Tools in DR detection.

AI screening technologies have aided the screening process by putting it in the hands of a broader spectrum of healthcare practitioners, allowing for earlier detection of DR and thus earlier intervention, reducing the need for costly and intrusive procedures and follow-up visits. It is too late, and so DR-related blindness will be reduced.

AI has also aided in the simplification of screening while ensuring that speed and accuracy are not sacrificed. With the ability to be used by technicians or trained laypeople, it has the potential to aid patients who, for whatever reason, are unable to get to an eye care professional's office for baseline screening. It also offers the ability to provide on-the-spot fundus imaging screening for patients who are unable to see an eye doctor due to financial constraints. This covers high-risk groups including those without health insurance and indigenous people living on government-run reserves.

Introduction:

In order to speed up disease detection, a machine learning model was created based on analysing thousands of images that were collected in rural areas and processed in order to automatically identify diabetic retinopathy.



Objective:

This project aims to detect blindness before it occurs. It is a machine learning model developed to accelerate the detection of diseases.

Dataset:

The data used for this project is sourced from Kaggle at:

https://www.kaggle.com/c/aptos2019-blindness-detection/data

Libraries :

In [6]:

```
import os
import cv2
import random
import warnings
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, cohen_kappa_score,classification_report

import tensorflow as tf
from tensorflow.keras.models import Model,load_model
from tensorflow.keras import optimizers, applications
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D, Input


# Set seeds to make the experiment more reproducible.
def seed_everything(seed=0):
    random.seed(seed)
    os.environ['PYTHONHASHSEED'] = str(seed)
    np.random.seed(seed)
    tf.random.set_seed(0)
seed_everything()

%matplotlib inline
sns.set(style="whitegrid")
warnings.filterwarnings("ignore")
```

Methodology:

A. Data Exploration Analysis (EDA):

The first thing we did with the data was to apply EDA on it. The exploratory data analysis method involved analysing the dataset to summarise its most important characteristics, using statistical graphics and other data visualisation methods.

Load Data

In [7]:
```
train_df = pd.read_csv('../input/aptos2019-blindness-detection/train.csv')
test_df = pd.read_csv('../input/aptos2019-blindness-detection/test.csv')
```

In [8]:
```
x = train_df['id_code']
y = train_df['diagnosis']
```

In [9]:
```
print('Number of train samples: ', train_df.shape[0])
print('Number of test samples: ', test_df.shape[0])
display(train_df.head())
```

```
Number of train samples:  3662
Number of test samples:  1928
```

	id_code	diagnosis
0	000c1434d8d7	2
1	001639a390f0	4
2	0024cdab0c1e	1
3	002c21358ce6	0
4	005b95c28852	0

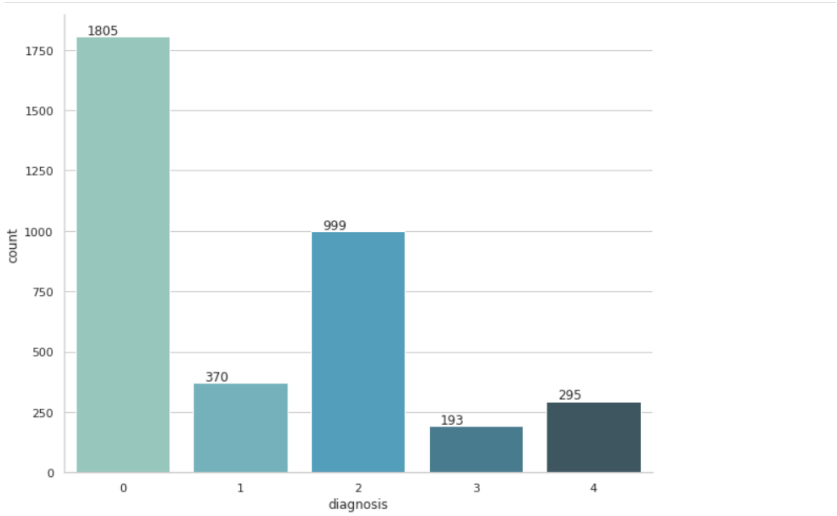
A1. Labelled Dataset Analysis:

We have aptos-2019-Blindness-Detection Dataset from kaggle consisting of 3662 labelled dataset in train.csv having features id_code and diagnosis with no missing data. We also have 1928 unlabeled dataset test.csv dataset

A clinical specialist labels the photographs. On a scale of 0 to 4, the integer labels represent the severity of DR, with 0 indicating no disease and 5 indicating the proliferative stage of DR.

Clinicians graded each image for the severity of diabetic retinopathy on a scale of 0 to 4, with 0 representing No DR, Mild DR, Moderate DR, Severe DR, and Proliferative DR and 1, 2, 3, 4 representing No DR, Mild DR, Moderate DR, Severe DR, and Proliferative DR, respectively.

The dataset has 1805 images with No DR , 370 with mild DR, 999 with moderate DR, 193 with severe DR and 295 with Proliferative DR which is visualised in the graph below.



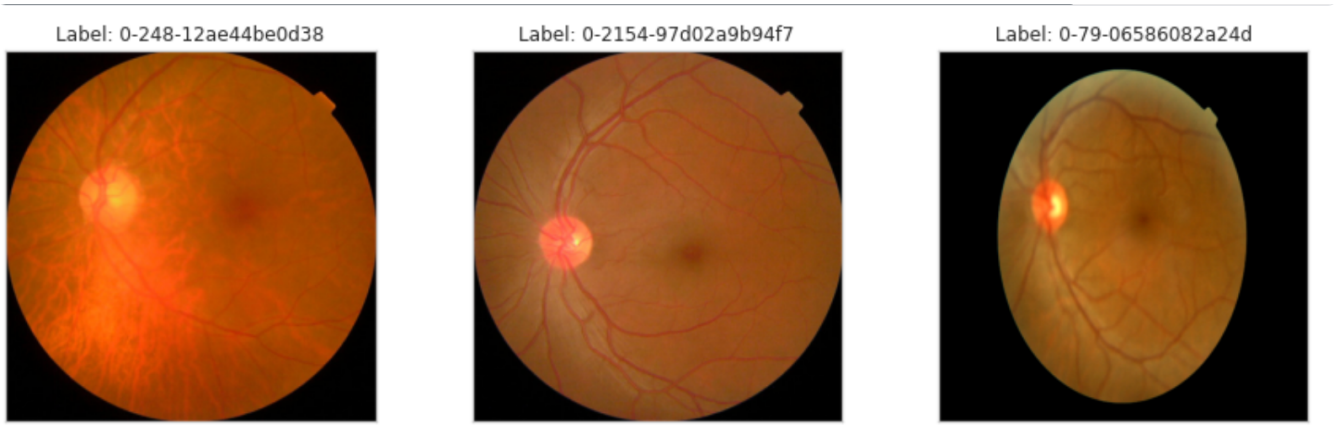

```
# plot categorical graph based on diagnosis class (0 = No DR, 1 - Mild , 2 - Moderate, 3 - Severe, 4 - Proliferative DR)

f, ax = plt.subplots(figsize=(10, 8))
ax = sns.countplot(x="diagnosis", data=train_df, palette="GnBu_d")
for p in ax.patches:
        ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()+10))
sns.despine()
plt.show()
```

A2. Image Analysis:

Image analysis or imagery analysis is the extraction of meaningful information from images; mainly from digital images by means of digital image processing techniques

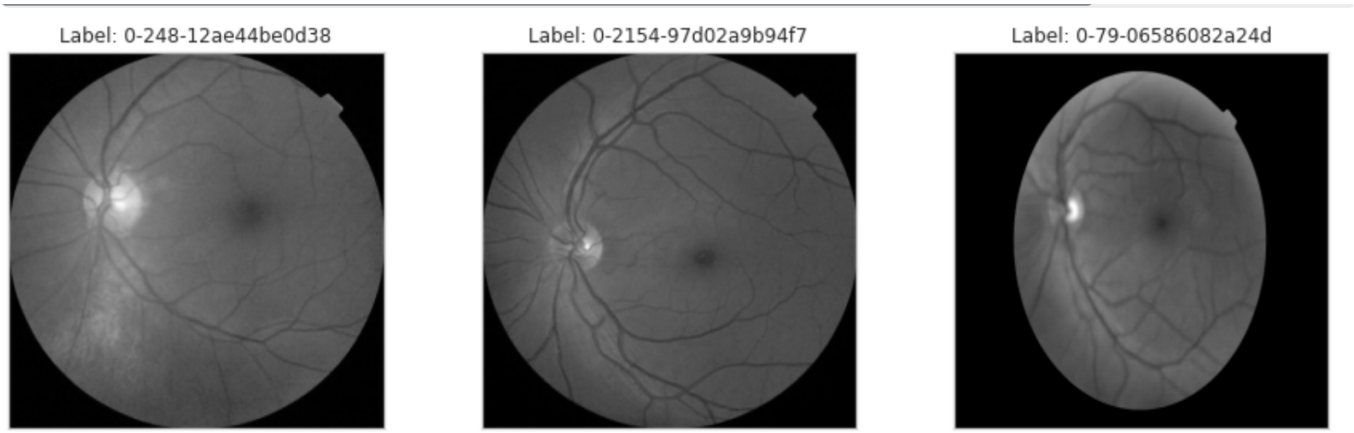
Original Images:

As we need to feed the same size, shape and dimension of images in the same format, we resize the images into (512, 512).



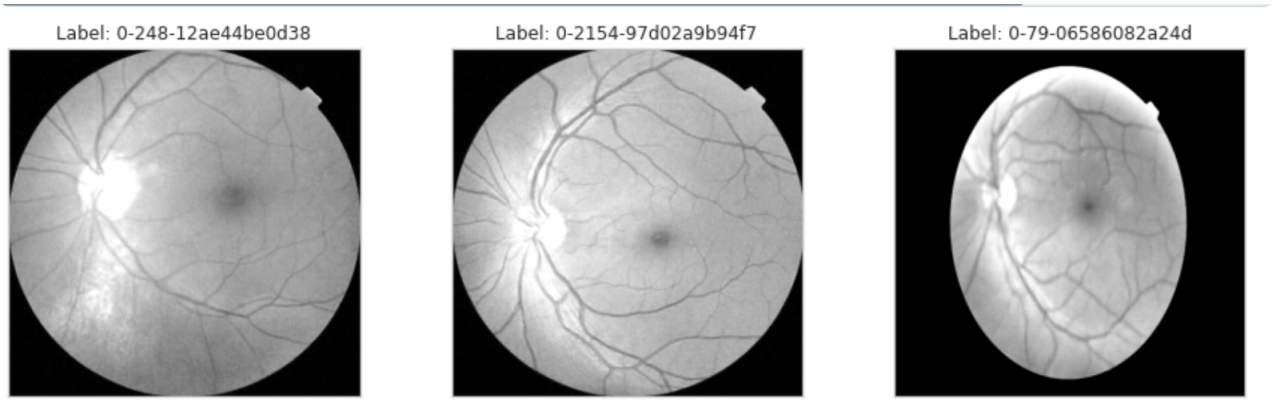
Green Channel Separation:

Green Channel Separation of the three colour channels in the image (Red, Green, and Blue) is done and the contrast between the blood vessels, exudates and haemorrhages is best seen in the green channel and these channels are neither under- illuminated nor over-saturated.



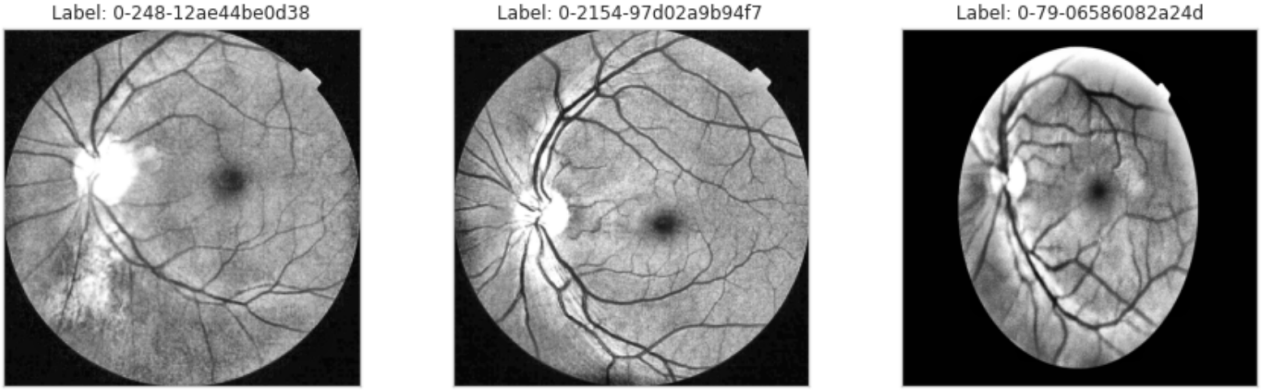
Contrast Stretching:

It is an image enhancement technique that improves contrast in an image by stretching the range of intensity values .



CLAHE:

Contrast Limited AHE (CLAHE) is a type of adaptive histogram equalisation in which contrast amplification is limited to reduce noise amplification. The slope of the transformation function determines the contrast amplification in the neighbourhood of a specific pixel value in CLAHE.



```
fig = plt.figure(figsize=(15,25))
for label in sorted(y.unique()):
    for i, (idx, row) in enumerate(train_df.loc[train_df['diagnosis'] == label].sample(3, random_state=rand).iterrows()):
      ax = fig.add_subplot(5, 3, label * 3 + i + 1, xticks=[], yticks=[])
      path=f"../input/aptos2019-blindness-detection/train_images/{row['id_code']}.png"
      image = cv2.imread(path)
      image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
      green_channel = image[:,:,1]
      p2, p98 = np.percentile(green_channel, (2, 98))
      img_rescale = exposure.rescale_intensity(green_channel, in_range=(p2, p98))
      img_adapteq = exposure.equalize_adapthist(img_rescale, clip_limit=0.03)

      # Morphological Structuring
      kernel = np.ones((5,5),np.uint8)
      erode_image = cv2.erode(img_adapteq, kernel, iterations=1)
      dilate_image = cv2.dilate(erode_image, kernel, iterations=1)

      image = cv2.resize(dilate_image, (size,size))

      plt.imshow(image, cmap = 'gray')
      ax.set_title('Label: %d-%d-%s' % (label, idx, row['id_code']) )
```

B. Modelling:

Modelling is done by using ResNet50 architecture.

A simple example can be used to explain residual learning. When learning to ride a bike, we make mistakes and learn from them. Once we can ride a bike, our brain stops activating the neurons responsible for learning the ability, allowing us to concentrate on other aspects of riding the bike.

Every epoch, the ResNet50 design does not need to fire all neurons. This cuts training time in half and enhances accuracy. It does not attempt to learn a feature again once it has been learned, instead focusing on learning fresh features. A brilliant strategy that significantly improved model training results.

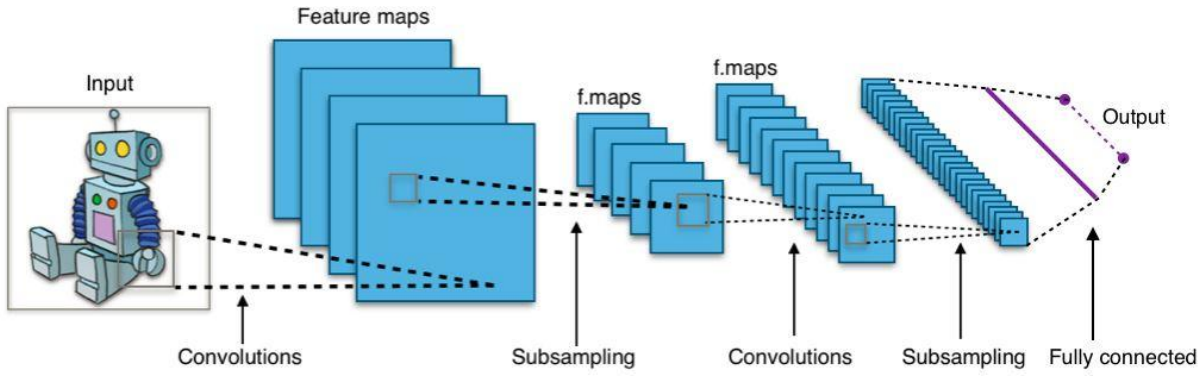


Image source: https://miro.medium.com/max/1400/1*3BRLw4lsANPEfGgimG3YVQ.png

```
train_datagen=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255,
                                  validation_split=0.2,
                                  horizontal_flip=True,
                                  featurewise_std_normalization = True,
                                  preprocessing_function = preprocessing
                                 )

train_generator=train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory= "../input/aptos2019-blindness-detection/train_images",
    x_col="id_code",
    y_col="diagnosis",
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    target_size=(HEIGHT, WIDTH),
    subset='training')

valid_generator=train_datagen.flow_from_dataframe(
    dataframe=train_df,
    directory= "../input/aptos2019-blindness-detection/train_images",
    x_col="id_code",
    y_col="diagnosis",
    batch_size=BATCH_SIZE,
    class_mode="categorical",
    target_size=(HEIGHT, WIDTH),
    subset='validation')

test_datagen = ImageDataGenerator(rescale=1./255)

test_generator = test_datagen.flow_from_dataframe(
        dataframe=test_df,
        directory = "../input/aptos2019-blindness-detection/test_images/",
        x_col="id_code",
        target_size=(HEIGHT, WIDTH),
        batch_size=1,
        shuffle=False,
        class_mode=None)
```

```
[ ]:
from tensorflow.keras.applications.resnet50 import ResNet50
```

```
def dr_detection_model(input_shape, n_out):
    ip_tensor = Input(shape=input_shape)
    baseModel = ResNet50(weights='imagenet',
                                    include_top=False,
                                    input_tensor=ip_tensor)


    x = GlobalAveragePooling2D()(baseModel.output)
    x = Dropout(0.5)(x)
    x = Dense(2048, activation='relu')(x)
    x = Dropout(0.5)(x)
    final_op = Dense(n_out, activation='softmax', name='final_output')(x)
    model = Model(ip_tensor, final_op)

    return model
```

+ Code + Markdown

```
[ ]:
model = dr_detection_model(input_shape=(HEIGHT, WIDTH, CANAL), n_out=N_CLASSES)

for layer in model.layers:
    layer.trainable = False

for i in range(-5, 0):
    model.layers[i].trainable = True

metric_list = ["accuracy"]
optimizer = tf.keras.optimizers.Adam(lr=WARMUP_LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy",  metrics=metric_list)
model.summary()
```

B1. Model training:

Transfer learning occurs when a model created for one activity is applied to a different task. Fine-tuning is a method of transfer learning in which the model output is changed to meet the new goal and only the output model is trained.

Advantage :

- Every new model no longer requires a big collection of tagged training data.
- Improving machine learning development and deployment efficiency for many models.
- A broader approach to machine problem solving that employs a variety of algorithms to address new problems.
- Models can be trained in simulations rather than in real-world settings.

The following parameters were used for modelling :

BATCH_SIZE = 8 , EPOCHS = 20 , WARMUP_EPOCHS = 2

LEARNING_RATE = 1e-4 , WARMUP_LEARNING_RATE = 1e-3

HEIGHT = 512 , WIDTH = 512 , CANAL = 1

N_CLASSES = train_df['diagnosis'].nunique()

ES_PATIENCE = 5 RLROP_PATIENCE = 3, DECAY_DROP = 0.5

Top layer Training:

In transfer learning we are pre-training our model with 2 epochs, 366 step size and 8 batch size. In top layer training we achieved train accuracy of 0.4582 and validation accuracy of 0.4615 after 2 epochs.

[ ]:

```
training_step_values = train_generator.n//train_generator.batch_size
validation_step_values = valid_generator.n//valid_generator.batch_size

# history_warmup = model.fit(trainX, batch_size=32, epochs=50)

history_warmup = model.fit_generator(generator=train_generator,
                                     steps_per_epoch=training_step_values,
                                     validation_data=valid_generator,
                                     validation_steps=validation_step_values,
                                     epochs=WARMUP_EPOCHS,
                                     verbose=1).history
```

Fine Tuning :

Fine-tuning is the process of fine-tuning or changing a model that has already been trained for one task to make it execute a second related task.

Deep learning models that have been fine-tuned can be used to train new models. It makes the process easier by saving time. Previous models include a lot of data that needs to be imported. As a result, it can help you save a lot of time. It also gives a large amount of data for the new model, making it much more dependable. Fine-tuning can help push the boundaries of deep learning by making the process of generating new algorithms considerably easier and faster.

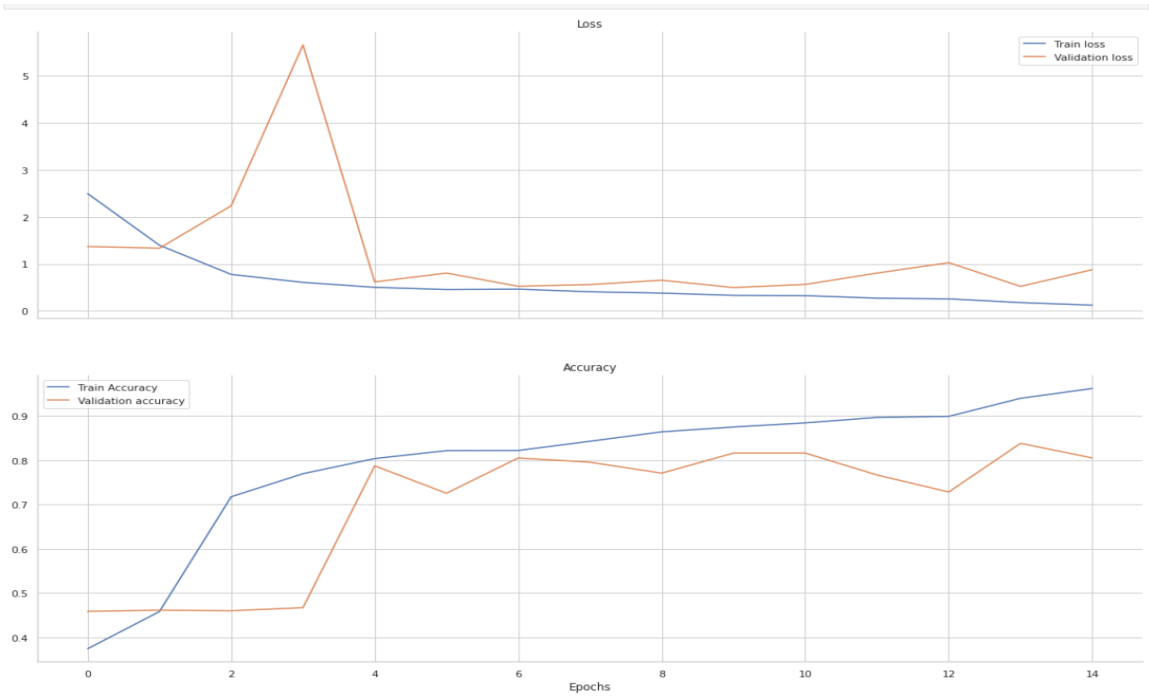
[ ]:
```
for layer in model.layers:
    layer.trainable = True

early_stop = EarlyStopping(monitor='val_loss', mode='min', patience=ES_PATIENCE, restore_best_weights=True, verbose=1)
reducing_LRplt = ReduceLROnPlateau(monitor='val_loss', mode='min', patience=RLROP_PATIENCE, factor=DECAY_DROP, min_lr=1e-6, verbose=1)

callback_list = [early_stop, reducing_LRplt]
optimizer = optimizers.Adam(lr=LEARNING_RATE)
model.compile(optimizer=optimizer, loss="categorical_crossentropy",  metrics=metric_list)
model.summary()
```

- Total params: 27,794,309
- Trainable params: 27,741,189
- Non-trainable params: 53,120

[ ]:
```
history_finetunning = model.fit_generator(generator=train_generator,
                                          steps_per_epoch=training_step_values,
                                          validation_data=valid_generator,
                                          validation_steps=validation_step_values,
                                          epochs=EPOCHS,
                                          callbacks=callback_list,
                                          verbose=1).history
```

B2. Loss Graph

One of the most widely used metric combinations is training loss + validation loss over time. The training loss is used to determine how well the model fits the training data, whereas the validation loss is used to determine how well it fits new data.



[ ]:

```
history = {'loss': history_warmup['loss'] + history_finetunning['loss'],
           'val_loss': history_warmup['val_loss'] + history_finetunning['val_loss'],
           'accuracy': history_warmup['accuracy'] + history_finetunning['accuracy'],
           'val_accuracy': history_warmup['val_accuracy'] + history_finetunning['val_accuracy']}

sns.set_style("whitegrid")
fig, (ax1, ax2) = plt.subplots(2, 1, sharex='col', figsize=(20, 14))

ax1.plot(history['loss'], label='Train loss')
ax1.plot(history['val_loss'], label='Validation loss')
ax1.legend(loc='best')
ax1.set_title('Loss')

ax2.plot(history['accuracy'], label='Train Accuracy')
ax2.plot(history['val_accuracy'], label='Validation accuracy')
ax2.legend(loc='best')
ax2.set_title('Accuracy')

plt.xlabel('Epochs')
sns.despine()
plt.show()
```

The categorical_crossentropy after 2 epochs, loss on the validation set approaches a minimum, and after 3 epochs, loss declines fast, nearing the saturated range of loss until 13 epochs.
Accuracy of Validation Dataset increased rapidly after 2 epochs and kept on increasing till it hit the stopping epochs..

B3. Model Evaluation:

Model evaluation is important to assess the efficacy of a model during initial research phases, and it also plays a role in model monitoring.

Confusion matrix :

	precision	recall	f1-score	support
0	0.99	0.97	0.98	1805
1	0.66	0.81	0.72	370
2	0.78	0.86	0.82	999
3	0.69	0.49	0.57	193
4	0.84	0.49	0.62	295
accuracy			0.86	3662
macro avg	0.79	0.73	0.74	3662
weighted avg	0.87	0.86	0.86	3662

```
cmplt_datagen = ImageDataGenerator(rescale=1./255)
cmplt_generator = cmplt_datagen.flow_from_dataframe(
        dataframe=train_df,
        directory = "../input/aptos2019-blindness-detection/train_images/",
        x_col="id_code",
        target_size=(HEIGHT, WIDTH),
        batch_size=1,
        shuffle=False,
        class_mode=None)

complete_step_size = cmplt_generator.n//cmplt_generator.batch_size
train_preds = model.predict_generator(cmplt_generator, steps=complete_step_size)
train_preds = [np.argmax(pred) for pred in train_preds]
```

[ ]:

```
labels = ['0 - No DR', '1 - Mild', '2 - Moderate', '3 - Severe', '4 - Proliferative DR']
cnf_mtx = confusion_matrix(train_df['diagnosis'].astype('int'), train_preds)
cnf_mtx_norm = cnf_mtx.astype('float') / cnf_mtx.sum(axis=1)[:, np.newaxis]
df_cnf_mtx = pd.DataFrame(cnf_mtx_norm, index=labels, columns=labels)
plt.figure(figsize=(16, 7))
sns.heatmap(df_cnf_mtx, annot=True, fmt='.2f', cmap="Blues")
plt.show()
```

+ Code + Markdown

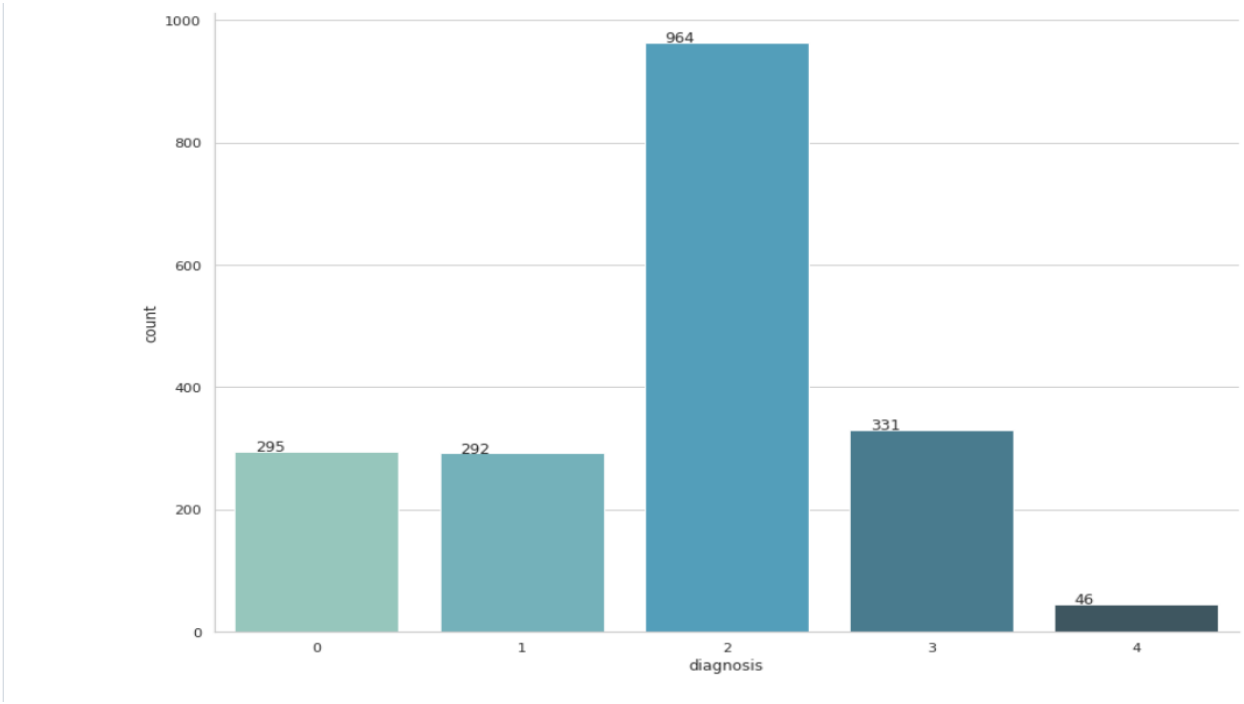
[ ]:

```
print(classification_report(train_df['diagnosis'].astype('int'), train_preds))
```

Quadratic Weighted Kappa

The Quadratic Weighted Kappa (QWK) statistic is used to determine how well two annotators agree. When the class labels have a natural ordering, QWK has been frequently employed as an evaluation metric for a variety of medical imaging difficulties.
Cohen Kappa score for training: 0.904

Deploying model on the test dataset



[ ]:

```
test_generator.reset()
test_step_values = test_generator.n//test_generator.batch_size
preds = model.predict_generator(test_generator, steps=test_step_values,verbose =1)
test_preds = [np.argmax(pred) for pred in preds]
```

[ ]:

```
img_names = test_generator.filenames
res = pd.DataFrame({'id_code':img_names, 'diagnosis':test_preds})
res['id_code'] = res['id_code'].map(lambda x: str(x)[:-4])
res.to_csv('submission.csv',index=False)
res.head(10)
```

[ ]:

```
f, ax = plt.subplots(figsize=(14, 10))
ax = sns.countplot(x="diagnosis", data=res, palette="GnBu_d")
for p in ax.patches:
        ax.annotate('{:}'.format(p.get_height()), (p.get_x()+0.1, p.get_height()))
sns.despine()
plt.show()
```

C. Result:

After applying our Model on Test Dataset consisting 1928 images, Model prediction shows:

Proliferative DR	46
Severe DR	331
Moderate DR	964
Mild DR	292
No DR	295

Conclusion:

We introduced an automated Diabetic Retinopathy Detection System based on the CNN model and the ResNet50 architecture in this research. To enrich the dataset to feed the deep network, a dataset of fundus images was reorganised, pre-processed, and augmented. The constructed model looked into different layers, activation functions, loss functions, and optimization algorithms in order to reduce computational costs while maintaining model accuracy. The model's accuracy in detecting Diabetic Retinopathy in terms of training (96.17 %) and validation (80.49 %) Because it is highly accurate, cost-effective, and time-efficient, ophthalmologists will be able to detect Diabetic Retinopathy earlier and more precisely, allowing them to prevent blindness.

References:

https://www.webmd.com/diabetes/diabetic-retinopathy

https://www.researchgate.net/publication/336829911_Early_Blindness_Detection_Based_on_Retinal_Images_Using_Ensemble_Learning

https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy/diagnosis-treatment/drc-20371617

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6901900/

https://pubmed.ncbi.nlm.nih.gov/33940045/

https://www.mayoclinic.org/diseases-conditions/diabetic-retinopathy/symptoms-causes/syc-20371611

https://collaborativeeye.com/articles/nov-dec-19/the-role-of-artificial-intelligence-in-detecting-diabetic-retinopathy/