DIABETIC RETINOPATHY DETECTION

Introduction:

Diabetic Retinopathy is an illness which occurs due to increased insulin level in the body, as a consequence of Diabetes, and is characterized by loss of sight. Diabetes is incurable, so its effect can only be minimized by early detection. Different stages of Diabetic Retinopathy can be detected by analyzing retinal photographs. The purpose of this project is to build a Convolutional Neural Network for the classification of levels of severity of DR based on the fundus images based on the dataset provided by Simplilearn as part of the Advanced Deep Learning curriculum. Due to the advancement of Artificial Intelligence, there have been many attempts to perform this task, present in the literature on public datasets. In this project, I tried to replicate some of these models and then build a CNN model from scratch, combining some of the attributes as well as figuring out solutions to the problem faced, especially to that of data imbalance.

Preprocessing:

The dataset consists of 2 parts. One, the retinal images, and second, a csv file mapping image names to the level of illness. Following tasks were performed as part of preprocessing:

1. Image Cropping: The images were rectangular in shape, where the retinal part was placed on a black background. Images are cropped such that retinal image is centralized and black portion of image is even out from every side. This introduces a similarity in the region under investigation hence improving the efficiency and accuracy. [1]

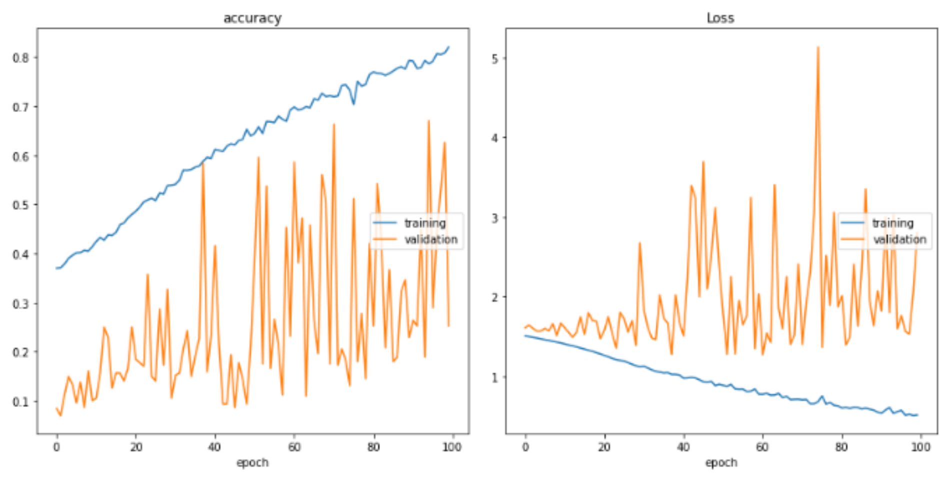
2. Histogram Equalization: Histogram Equalization increases contrast in images by detecting the distribution of pixel densities in an image and plotting these pixel densities on a histogram. The distribution of this histogram is then analyzed and if there are ranges of pixel brightnesses that aren't currently being utilized, the histogram is then "stretched" to cover those ranges, and then is "back projected" onto the image to increase the overall contrast of the image. [2] Since retina image consists of various objects like illuminated retina, nerve fibers, blood vessels and other deformities, hence contrast between different objects is essential. Histogram equalization provides the necessary contrast between illuminated retina and other objects hence making distinction between different objects easier to detect. [1]

3. Image Resizing: The images were of high resolution and therefore of significant memory size. The images were therefore resized to 512x512 which retained important features but also reduces the dataset size to something the machine can handle.

4. Balancing the Dataset: The original dataset is divided into 5 classes of DR, with following class distributions

class	frequency
0	1016
2	230
1	113
4	36
3	32

Here class 0 represents Healthy retina, where 4 represents Proliferative DR. As can be observed, the dataset is highly imbalanced with 71.2 percent of examples belonging to class 0 or healthy images, while only 2.2 percent of examples belonging to class 3 and 2.5 percent belonging to class 4. This imbalance posed a serious challenge to the ability of the classification algorithms to correctly identify the images with serious Retinopathy as the model consistently attained a local minima at 71.2% accuracy despite different architectures, optimizers and tuning these didn't affect the model accuracy. There were 3 options available. One, penalizing the loss function to learn more from classes with poor representation. Two, undersampling, and three, oversampling. The first modification drastically reduced the model's ability to learn and dipped model accuracy below par. Undersampling, was not successful either as the minority classes were too less in absolute numbers for the deep learning model to learn. Few different techniques were tried for oversampling. Firstly, SMOTE (synthetic minority oversampling technique) was tried on the images but it produced noisy graph and accuracy didn't increase from 68%. Moreover, the images generated were not good quality to add meaning to the model.



The second technique I tried to oversample was by downloading the APTOS dataset from Kaggle, mixing the images belonging to different classes and then downsampling to make the dataset balanced, but the validation accuracy didn't improve from 53% and model lost its capacity to learn. This was probably because the labels in this dataset didn't match with the labels in the provided dataset. More research needs to be done on why the two datasets didn't add up. Finally, I decided to augment the dataset, but instead of online transformations as with Keras ImageDataGenerator, I decided to manually increase the size of the dataset by applying specific transformations which were more suitable for the model to learn – namely, 90 degrees rotation, 180 degrees rotation, vertical flip and shear with angle -15. To avoid too many duplicates, class 3 and class 4 samples were combined into a single class, and renamed as Severe DR. After applying all transformations, the dataset was downsampled with the minimum count for all classes, making it 816 for each class. Following table summarizes the transformations applied to different classes:

Original classes	New classes	Transformations	Original Count	Transformed Count	Final Count
0	No DR	NONE	1016	1016	816
1	Mild DR	Rotate 90, vertical flip, shear	113	896	816
2	Moderate DR	Vertical flip, shear	230	920	816
3	Severe DR	Rotate 90, 180, vertical flip, shear	32	816	816
4	Severe DR		36		

5. Train-Test Split and Class Segregation: As dataset was input through flow_from_directory of Keras ImageDataGenerator, all the images which were present in the local directory needed to be segregated into separate class folders as well as train and test folders. Labels were present in the dataframe created from the csv file, which was then split into train-test arrays using sklearn train_test_split and then using the file names from these dataframes, images were moved into respective directories. Using similar logic, using the shutil utility, images in both train and test folder were segregated according to the new class names. The dataframe had already been updated during augmentation for transformed images.

Model Building:

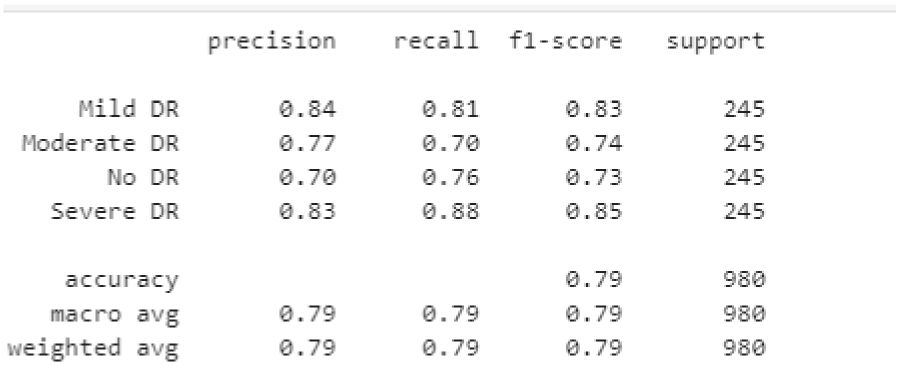
Architecture:
In the preprocessing step, all the images were resized to standard 512x512x3 pixels. The CNN model was built from scratch, and designed to be deep so that it learns complex features. There are 9 convolutional blocks. In each block, the kernel size is fixed to be 3x3. 2x2 Strided convolutions along with maxpooling layers are used to downsample the feature map. In each block except 5th and 7th block, batch normalization is applied. After the last convolutional block, the network is flattened to one dimensional. Then 2 dense layers are added each with 1024 neurons. To avoid the problem of overfitting, 0.5 dropout layers are added after each fully connected layer. To further avoid overfitting, 0.2 dropout is added after the last convolutional layer.

Compilation and Training:
Model was compiled with Stochastic gradient descent optimization algorithm with a learning rate of 0.001. The loss function used was categorical cross-entropy. Model was trained for 100 epochs initially and then 15 epochs to confirm convergence.

Results:

For the classification task, 3 metrics were chosen: 1. Accuracy, 2. Recall, 3. Precision and were evaluated on the validation data. The following confusion matrix and classification report summarize the model results:

Classification Report:

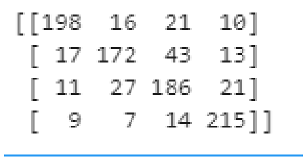


```
              precision    recall  f1-score   support

     Mild DR       0.84      0.81      0.83       245
 Moderate DR       0.77      0.70      0.74       245
       No DR       0.70      0.76      0.73       245
   Severe DR       0.83      0.88      0.85       245

    accuracy                           0.79       980
   macro avg       0.79      0.79      0.79       980
weighted avg       0.79      0.79      0.79       980
```

Confusion Matrix:



```
[[198  16  21  10]
 [ 17 172  43  13]
 [ 11  27 186  21]
 [  9   7  14 215]]
```

The model after balancing the data performed better than with imbalanced data, which produced an accuracy of 71.2% which is just the proportion of the majority class.

Autoencoder Task:

The task was to see if the latent representation in the bottleneck of autoencoder forms as many clusters as number of classes. To accomplish the task, first the images were converted to numpy arrays, then flattened into a 2 dimensional array, compressed by T-SNE. The lower dimensional representation was then plotted using matplotlib. The image array was also trained on an Autoencoder model and we were interested in the latent representation of these images. The model was trained 2 times. Once, on a 2 dimensional bottleneck with full dataset, and again on a 3 dimensional bottleneck with undersampled but balanced dataset. Then the latent representation was plotted using matplotlib. No clusters were observed.

Conclusion:

The imbalance of the dataset made it hard to get good classification accuracy and there was no other solution other than combining the two minority classes into one and then manually augmenting the images to increase the size of the dataset. Due to lack of time and resources, memory management on disk was less than ideal and I ended up creating multiple folders of the dataset to perform separate tasks. Histogram Equalization could've been made a part of a custom ImageDataGenerator. Hopefully, in future, we'd able to use GANs to generate a more balanced dataset, but due to lack of time and resources, that couldn't be experimented with. Also, transfer learning could’ve been applied to get better results, but due to lack of time and resources, that couldn’t be experimented with either.

References:

[1] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0233514

[2] https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085

[3] https://www.sciencedirect.com/science/article/pii/S1877050916311929?via%3Dihub

[4] https://www.sciencedirect.com/science/article/abs/pii/S0045790618334190?via%3Dihub

[5] https://ieeexplore.ieee.org/document/8701231