**Diabetic Retinopathy Detection using ResNet50**

**Image Preprocessing**

Train images along with their labels are loaded using Keras Image Datagenerator\_flow\_from\_dataframe method. For the processing steps, images were normalized which will transform pixel intensity values of images from 0-255 to 0-1. All images are resized into 512 x 512 dimension to keep the size of each image constant. For data augmentation horizontal flipping was used. The dataset contains 3662 images in training set and 1928 images in the testing set. Labels are only provided with training set therefore we have used images in training set for further experiments. The dataset was divided into 80:20 ratios, where 80% of the train images are contained in training set and remaining 20% of train images are allocated for validation set. Dataset are divided into the batch size of 32.

**Model Training (ResNet50)**

We applied transfer learning of pre-trained CNN architecture in our experiment. We used ResNet50 pre-trained CNN architecture that is originally trained on ImageNet dataset as a backbone architecture. Classification layers of original model was replaced by user defined classification layers that contains Global Average Pooling 2D followed by a dropout of 0.5% and then a fully connected layer of 2048 units again followed by a dropout layer of 0.5% was added. Finally, an output layer of 5 units was added on top of base model as it is a multi-classification problem and we have five classes to predict. At first classification layers were trained keeping layers in convolutional base of ResNet50 frozen. The optimizer used in this case is Adam with a warm up learning rate of 0.001 and loss function was categorical\_cross\_entropy. ReLU activation function was used in hidden layers whereas for output layer softmax was used. The model was trained for 2 epochs at first. Once model is trained all the layers of ResNet50 are trained again with initial learning rate of 0.0001 and then learning rate is decayed with a decay rate of 0.5 till 1e-6. The model was trained for 20 epochs and in order to prevent the model from overfitting, we applied early stopping which will monitor validation loss after each epoch in which model will stop training if validation loss keeps on increasing with patience rate of 2.

**Evaluation Measure**

In order to evaluate the performance of trained model, accuracy, precision, recall (sensitivity), f1-score, cohen kappa was used as an evaluation measure. We have also drawn confusion matrix to get more insight on the performance of trained model. The model achieved 93.36% training accuracy, 84.20% validation accuracy and 85% overall test accuracy.