Automatic Detection and Grade Classification of Diabetic Retinopathy using Deep CNN

Aniruddha Mulay

Abstract— Diabetic Retinopathy is a complication of diabetes, caused by high blood sugar levels which results in damage to the back of the eye or the retina. Diabetic Retinopathy is the leading cause of blindness in the working age population of the world. An estimated 93 million people are affected by this disease. As per the NHS, an estimated 1280 new cases of diabetic retinopathy are recorded each year in England alone. Currently, the methods employed for the detection of diabetic retinopathy are manual and time consuming. The manual method involves a trained clinician evaluating retina fundus images. Although this method is effective, this process is time consuming and delays can be caused due to late submission of retina fundus images, miscommunication resulting in delayed treatment of the patients. Also, the lack of expertise and employing this method in highly populated areas is another major problem with this approach. To overcome this problem, this paper proposes a deep learning system which will detect diabetic retinopathy from retina fundus images and classify it as per the severity levels with high degree of accuracy and store the results in a CSV file.

Keywords—Diabetic Retinopathy, Keras, Convolutional Neural Networks, Computer Vision, ResNet-50, ResNet-101, DenseNet-121, Xception

I. INTRODUCTION

Diabetes mellitus, also commonly known as diabetes.[1], is a metabolic disease which causes high blood sugar. To be used as a source of energy, a hormone by the name insulin is responsible for moving sugar from blood into cells. When a person is affected with diabetes, his body is unable to make enough insulin to transport the sugar from blood to cells or the body is unable to effectively utilize the insulin which it makes. If these high levels of blood sugar are left untreated, they can damage nerves, eyes, kidneys and other organs. Individuals who suffer from diabetes are also at risk at developing diabetic retinopathy, diabetic retinopathy is the damage caused to blood vessels in the retina as a result of high blood sugar levels. Diabetic retinopathy is the leading cause of blindness among adults as well as the most common cause of vision loss among people suffering from diabetes. Sugar blocks the tiny blood vessels that go into the retina, which causes them to leak or bleed. Diabetic Retinopathy does not usually produce symptoms during the early stages. The symptoms usually become more noticeable when the condition becomes more advanced. Some of the symptoms are blurred vision, impaired color vision, poor night vision, sudden and total loss of vision, patches which block a person's vision, etc. The image below shows the retina of a person affected with diabetic retinopathy:

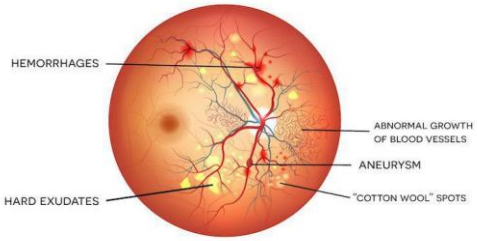


Figure: Diabetic Retinopathy affected retina

Diabetic retinopathy is usually classified into 5 stages on the scale of 0 to 4.[2]: *0-No DR, 1-Mild DR, 2-Moderate DR, 3-Severe DR, 4-Proliferative DR.*

Stage	Dilated Ophthalmoscopy Observable Findings	Severity
I	No abnormalities	No DR
II	Micro-aneurysms only	Mild non-proliferative DR
III	Any of the following: - micro-aneurysms - retinal dot and blot haemorrhages - hard exudates or cotton wool spots No signs of severe non-proliferative diabetic retinopathy	Moderate non-proliferative DR
IV	Any of the following: - more than 20 intra-retinal hemorrhages in each of 4 quadrants - definite venous beading in 2 or more quadrants - prominent intra-retinal microvascular abnormality (IRMA) in 1 or more quadrants No signs of proliferative retinopathy	Severe non-proliferative DR
V	One or both of the following: - Neovascularization - Vitreous/pre-retinal hemorrhage	Proliferative DR

Vision impairment caused by diabetic retinopathy can be slowed down or averted completely if diabetic retinopathy gets detected in its early stages. The current problem while detecting and classifying diabetic retinopathy is that it is a time consuming, manual process which involves a trained clinician going over retina fundus images. Although this method is effective, delays in submitting the retina fundus images, miscommunication, chances of human error might lead to delayed treatment. Manual detection of diabetic retinopathy is also a problem in densely populated areas where the number of people affected by DR is higher and also in rural/remote areas of low to middle income countries

where there simply aren't enough trained clinicians. This paper proposes a Convolutional Neural Network approach to overcome this problem, the CNN will be trained using a labeled dataset consisting of thousands of retina fundus images so as to automatically classify and label the test set images as per the severity levels from 0 to 4 and output the generated results to a CSV file.

II. Related Work

Ever since the introduction of Deep Learning architectures, different methods have been proposed to automate the process of detection and grade classification of diabetic retinopathy. A paper titled 'Automated Early Detection of Diabetic Retinopathy'.[3] was published in June 2010, this paper proposes an algorithm named 'Challenge2009' and compares its performance with 'Eyecheck' which was widely used during that time. The 'Eyecheck' algorithm which was first proposed in the year 2000 firstly detects all the pixels in the image and creates pixel clusters which are further clustered in candidate lesion. Then, a KNN classifier is applied on these clusters to assign it a probability indicating the likelihood is a red lesion. This algorithm was capable of detecting exudates and cotton wool spots. The 'Challenge2009' algorithm on the other hand was capable of detecting microaneurysms in the fundus image by assigning a probability to candidate lesions that are searched for using an adaptive wavelet domain. The paper titled 'Automatic Detection of Diabetic Retinopathy using Deep Learning'.[4] proposes an automated DR grading system trained and tested using two CNN architectures namely AlexNet and GoogleNet. These architectures were trained and evaluated on a Kaggle dataset containing 35,000 retinal images spread out over 5 different severity classes and the Messidor-1 dataset containing 1200 physician verified retina fundus images. The paper titled 'Automated detection of diabetic retinopathy using SVM'.[5] takes a completely different approach compared to other deep learning approaches for detecting and classifying retina fundus images on the DR severity scale. The approach proposed in this paper will first isolate different features of a DR affected retina (blood vessels, microaneurysms, hard exudates) in order to extract these features from the fundus image. These extracted features will then be fed to a Support Vector Machine (SVM) network to figure out the retinopathy grade of each retinal image. The study used the Messidor database for carrying out this experiment and was able to achieve a prediction accuracy of 93.8% and sensitivity of 94.6% when tested on 400 retina fundus images. This proposed method was only applicable to classify non proliferative DR images. In the paper 'Convolutional Neural Networks for Diabetic Retinopathy'.[6], a CNN approach is proposed to classify retina fundus images as per severity levels. This approach is trained on a dataset containing 80,000 train images on which it is able to achieve a classification accuracy of 95% and a 75% validation accuracy when tested on 5000 test images. This paper also discusses how a CNN approach yields significantly better results compared to a SVM approach when trained and evaluated on such a large dataset. The architecture used in this study is the one developed by Visual Geometry Group of the University of Oxford, the VGG-16 architecture. The paper furthers talks about the class imbalance in the dataset used for training the network and the approach taken to reduce the risk of overfitting. The paper 'DREAM: Diabetic Retinopathy analysis using Machine Learning'.[7] proposes a novel two step approach compared to previous approaches used for detecting and grade classifying diabetic retinopathy. In the first step, classifiers like GMM, KNN, SVM and Adaboost are evaluated for classifying retinopathy lesions from non-lesions. Out of the classifiers compared, GMM and KNN are found out to be ideal for bright lesion and red lesion classification. The step one filters out non lesions. In step two, bright lesions get classified as hard exudates and cotton wool spots while red lesions get classified as hemorrhages and micro-aneurysms. This novel approach helps in dealing with problems arising due to dataset imbalance. This grading system was tested on the MESSIDOR dataset consisting of 1200 images. The paper 'Detection and classification of exudates in retinal image using image processing techniques'.[8] talks about a system designed and developed to specifically detect exudates in a DR affected retina fundus image. In the preprocessing step, fundus images are resized, noise removal techniques are used and contrast enhancement is performed on them. Gray level Covariance Matrix (GLCM) is used for feature extraction. Classification is then performed using FCM and KNN algorithms. Another novel approach is proposed by the paper 'Exudate Detection for Diabetic Retinopathy using Pretrained CNN'.[9] wherein different CNN architectures (VGG-19, Inception-v3, ResNet-50) are used for extracting features from the fundus image. The features extracted using the three architectures are then fused together and then fed to fully connected layers with 'softmax' acting as the activation function. This combined classifier outperforms the three independent architectures when comparing the classification accuracy parameter. This is some of the background/related work that has been performed/proposed when it comes to applying Deep Learning to automate the process of detection and classification of diabetic retinopathy.

III. Dataset

The dataset used for this experiment is the dataset used for the 4th Asia Pacific Tele-ophthalmology Society (APTOS) 2019 blindness detection competition.[10]. This dataset consists of two folders namely 'train_images' and 'test_images'. The training images folders contains 3662 images spread out over 5 different classes marked on the diabetic retinopathy severity scale from 0 to 4. These 3662 images are to be used for training and validating the network's performance. These 3662 images are mapped to a CSV file which contains the 'id_code' of the image and its corresponding severity on the DR scale. The 'test_images' folder contains 1923 images for which the severity level needs to be predicted. The images for testing the model have been mapped to a CSV file by the name of 'test.csv'. This CSV file contains the unique 'id_code' for each of the images in the testing folder. Out of the 3662 training images in the dataset, 1805 images are of level 0(No DR), 370 images are of level 1(Mild DR), 999 images are of level 2(Moderate DR), 193 images are of level 3(Severe DR), 295 images are of level 4(Proliferative DR). As it can be seen the dataset is quite imbalanced and steps will need to be taken so that images of any particular level will not dominate the training process. Like any real-world medical dataset, these images have been obtained from different clinics from different regions over the world over a certain time period. Some images in the dataset are

underexposed, out-of-focus, overexposed, artefacts, etc. These images are the result of the retina fundus image being captured in low lighting conditions. Such images might pose a problem when performing multi-label classification. Image samples from the dataset are as follows:

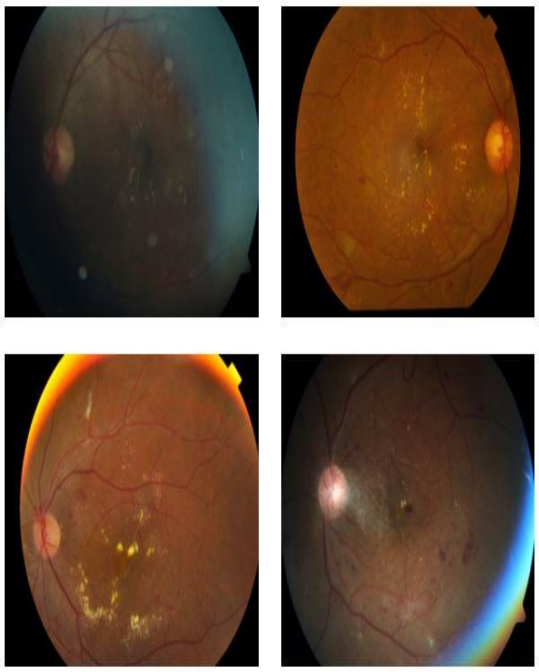


Figure: APTOS 2019 Dataset samples

As it can be seen, the image in the first frame is both underexposed, out of focus and also suffers from lens flare. While fixing the lens flare in the images within the dataset would be a complex task in the current scope of the project the exposure element of these images can be fixed rather easily by introducing the concept of Gaussian blurring. In the Gaussian blur operation, the input images are convolved with a Gaussian filter which is basically a low pass filter designed to filter out high frequency components. As a result, the defining features of a diabetic retinopathy image which are essential for telling them apart from a normal retina image get enhanced thus making them easier to detect and classify. Some of the processed image samples from the dataset are as follows:

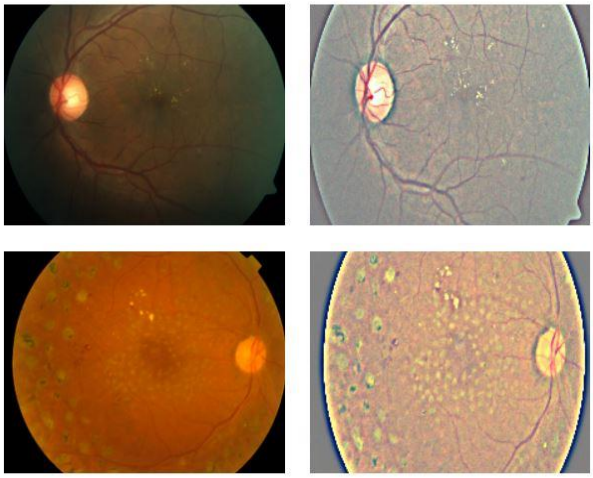


Figure: Input retina fundus images and processed images

Visualizing the image, we can clearly see the difference the Gaussian filter makes in improving the visibility of the features in the retina fundus images. Also, the gaussian filtered images have been resized to 700x700 dimensions since all the images in the train and test folders are of different sizes and CNN needs images with a uniform dimension for training the model. 700X700 dimension has been chosen taking into consideration the computation cost and that the image retains its details in spite of being resized to a lower dimension.

IV. CNN Architectures

Convolutional Neural Networks.[11], or CNNs as they have been referred to form the backbone of many of today's image processing, image recognition and computer vision systems. In 1959, David Hubel and Torsten Wiesel described the concept of simple and complex cells in the human cortex. They stated that the simple cell responds to edges and bars of particular orientation while complex cells also can respond to these edges and bars but have the added advantage that the cells will still respond even if these edges and bars are move around in the space. They proposed that these kinds of cells are used in pattern recognition. Inspired by the work of David Hubel and Torsten Wiesel, Dr. Kunihiko Fukushima in the 1980s proposed a model in the paper titled 'Neocognitron: Neural Network model for a mechanism of pattern recognition unaffected by shift in position'.[12]. In this paper, Dr. Kunihiko talks about how the neocognitron model includes components like the 'S-cells' and 'C-cells' to be used for mathematical operations. These 'S-cells' sit in the first layer of the model and are connected to the 'C-cells' which sit in the second layer. The idea which was proposed was to turn the simple model into a complex model for visual pattern recognition. The first actual groundwork on Convolutional Neural Networks began when Yann LeCun published the paper 'Gradient based learning applied to Document Recognition'.[13] which demonstrated how a neural network can be applied to recognize handwritten digits. To be more specific, Yann LeCun trained a CNN model using the MNIST handwritten digits dataset to recognize handwritten digits. The CNN model was trained on the MNIST dataset by feeding it an input image and computing the output prediction, using the predictions, the model's settings were updated which was based upon the correct or incorrect prediction of the digits. The process will be repeated until reaching the optimal model settings and the loss being minimized. Over the years, researchers continued to make progress in developing more and more complex CNN architectures. The next breakthrough came in 2012 with the introduction of AlexNet which won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) with a top-5 error rate of 15.3%. The ILSVRC used the ImageNet dataset for evaluating different CNN architectures. The dataset contains 14 million images spread out over 1000 classes and is amongst the most popular datasets used by researchers today. This experiment will make use of 4 such CNN architectures, they are ResNet-50, ResNet-101, DenseNet-121 and Xception.

1] ResNet Architecture

Since the introduction of AlexNet in 2012, the goal of researchers is to make neural networks go deeper and deeper. AlexNet had only 5 convolutional layers, while the architectures GoogleNet and VGG networks released subsequently had 22 and 19 layers respectively. However, increasing the network's depth by simply stacking layers on top of each other does not always work. Deeper networks are notoriously hard to train because of the vanishing gradient problem. When the gradient is backpropagated multiple times to earlier layers, continuous multiplications make the gradient infinitively small. Due to the vanishing gradient problem, the performance of the network starts getting saturated and, in some cases, degrades rapidly as the network goes deeper. Due to the vanishing gradient problem, a 20-layer CNN demonstrated lower training and test error compared to a 56-layer CNN. The core idea behind ResNet.[14][15] is the introduction of a "identity shortcut connection" that skips one or more layers, as seen in the figure:

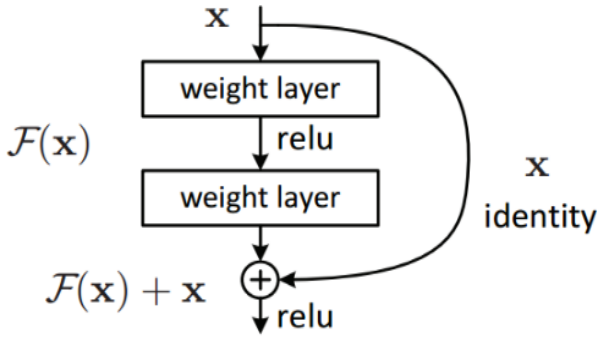


Figure: ResNet skip connection

The ResNet block or the skip connection block allows skipping of a few layers during the training process. The skip connection adds the output from the previous layers to the output of the stacked layers. This allows us to train far more deeper networks. This helps in training the deeper networks more efficiently compared to training deeper networks without the skip connection and also mitigates the problem of vanishing gradient by allowing the gradient to flow through an alternate path. The ResNet architecture won the ILSVRC 2015 classification competition with a 3.57% error rate. There are many variants of the ResNet architecture with varying number of layers. They are: ResNet-18, ResNet-34, ResNet-50, ResNet-101, ResNet-110, ResNet-152, ResNet-164, ResNet-1202.

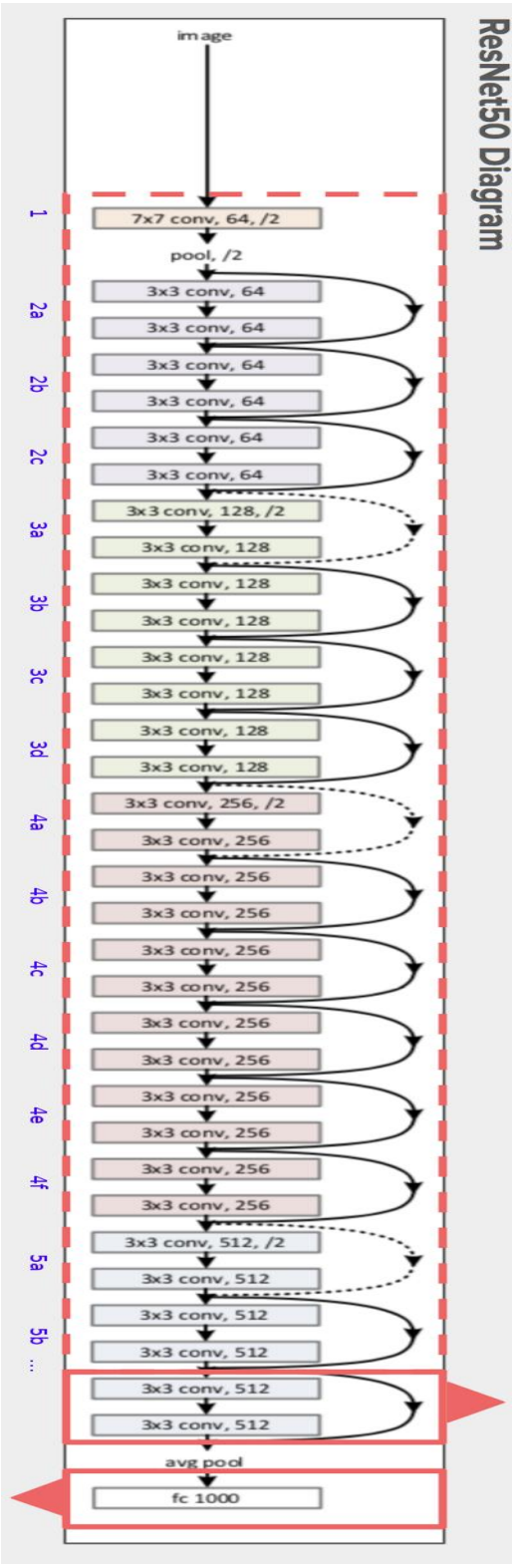


Figure: ResNet-50 architecture

2] DenseNet architecture

DenseNet.[16][17] architecture, similar to other neural network architectures aims to increase the depth of Deep CNN. The main problem faced by deeper CNN is that the information path from the input layer to the output layer becomes so big (in opposite direction, the backpropagation makes the gradient extremely small) that it gets vanished even before reaching the end of the other side. To counter

this problem, DenseNets simplify the connectivity pattern, they solve this problem by ensuring maximum information flow in forward direction (max gradient flow in reverse direction). They do this by simply connecting every layer in the network directly with one another.

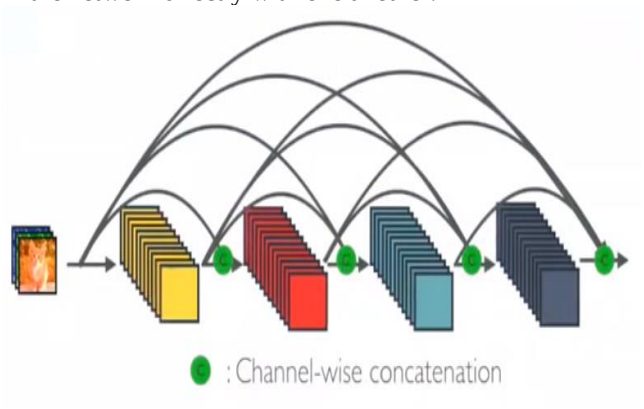


Figure: Dense block

Visualizing the Dense block of the DenseNet architecture, we can see that each that each layer obtains additional inputs from the preceding layers and passes its own feature maps to subsequent layers. This allows the network to be far more compact and thinner compared to other neural network architectures. DenseNet is both memory efficient and computationally cost effective. Some of the advantages of the DenseNet architecture are as follows:

- Strong Gradient flow
- Parameter efficiency
- Computational cost efficiency
- Diversified set of features
- DenseNet classifier uses features across all complexity levels.
- Prevent overfitting and make the training process simpler.

Probably the biggest advantage of DenseNet is feature reuse, it exploits the potential of the network by concatenating feature maps learned by different layers in the network. This is one of the major differentiating points between ResNet and DenseNet.

The different DenseNet variants are: DenseNet-121, DenseNet-169, DenseNet-201, DenseNet-264. The number of convolutional layers, pooling layers, transition layers and classification layers are similar across all variants with changes in number of dense blocks.

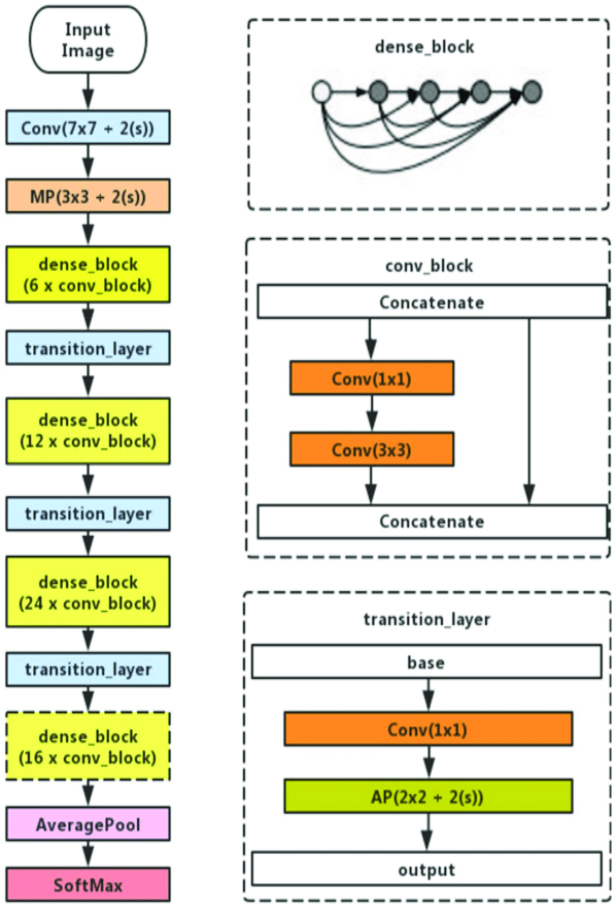


Figure: DenseNet-121 architecture

3] Xception architecture

Xception stands for 'Extreme Inception'. This architecture changes our perception of Convolutional Neural Networks in general. The Xception.[18][19] architecture takes the principles of the Inception architecture to the extreme. The author of the paper 'Deep Learning with Depthwise Separable Convolutions', François Chollet, who happens to be the founder of the Keras framework states that:

"We present an interpretation of Inception modules in convolutional neural networks as being an intermediate step in-between regular convolution and the depthwise separable convolution operation"

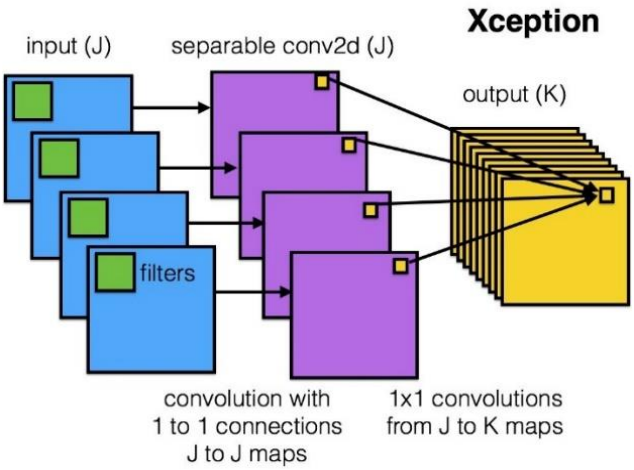
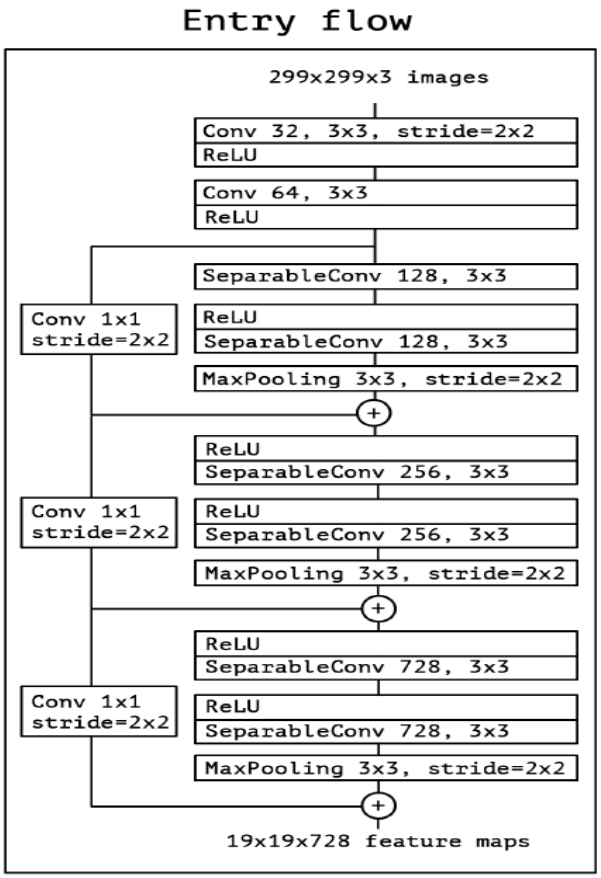
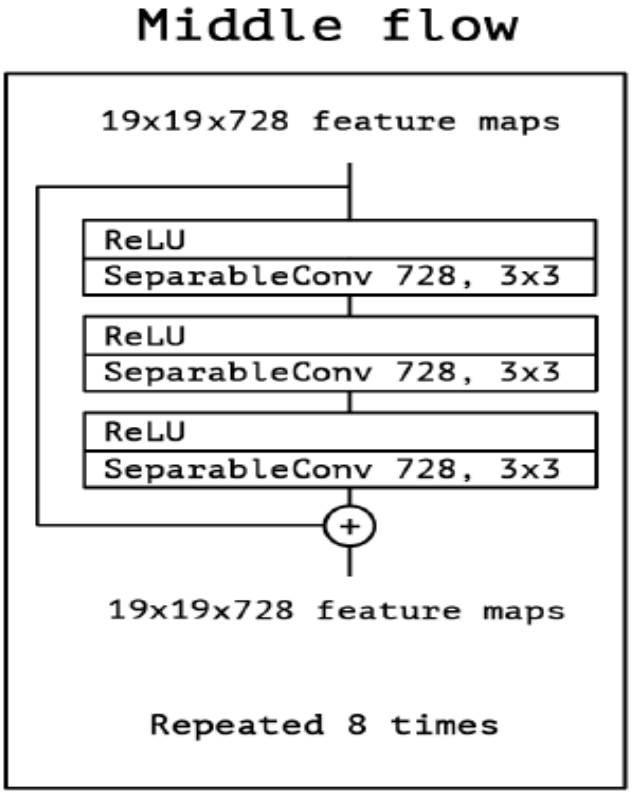


Figure: Xception architecture

Xception has the same number of parameters as the Inception architecture but is able to outperform the Inception architecture for image classification datasets containing thousands of classes. This indicates Xception is computationally efficient.





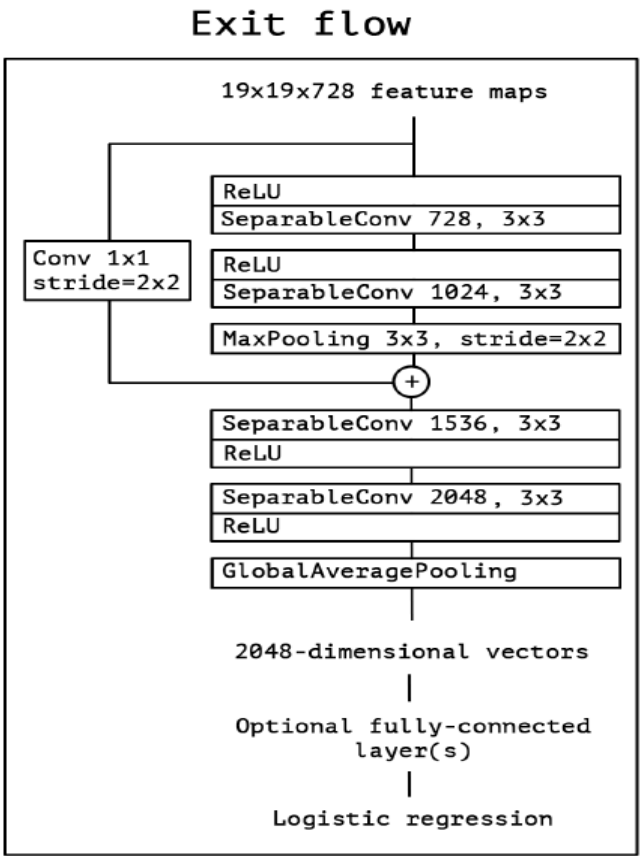


Figure: Xception architecture

V. TRAINING AND EVALUATION

In this section, we will be discussing how the CNN model was trained on the APTOS 2019 data and then utilized for computing the predictions of the test data.

1] Data Pre-processing and Evaluation

Data preprocessing is an extremely important step which needs to be taken before commencing the training process of any deep neural network. Data preprocessing helps us in improving the quality of data, removing noise which may be a hinderance in the training process, remove missing values, redundancies, etc. All these steps help us in making our model more accurate and hence improves the prediction accuracy on the test data as well. As discussed in Section 3, like any real-world dataset, a few of the images within the dataset are underexposed, out of focus and have artefacting issues. To counter this problem, the Gaussian Blur method which is basically a low pass filter which is convolved with the input images to remove high frequency components. This method helps in improving the exposure thereby making it easier for the neural network classifier to detect features peculiar to a DR affected retina. The APTOS 2019 competition dataset used for this experimentation has a total of 3662 images with their severity grade labels in a CSV file for training and 1923 images for testing. Like all other DR detection image datasets, this dataset is also imbalanced with more label 0(No DR) and label 2(Moderate DR) images compared to other label images. The images are of varying dimensions and hence have been resized to 700x700 as a CNN requires all images to be of

same dimensions. The train data of 3662 images has been split up into training and validation with 2930 images and 732 images respectively. The ImageDataGenerator function has been employed to rescale the image pixels between 0 to 1, rotate images, flip images, zoom in and out of images, shear the images, etc. All these augmentation techniques help the CNN model become more robust and accurate when exposed to images captured at different angles, zoom ranges, etc.

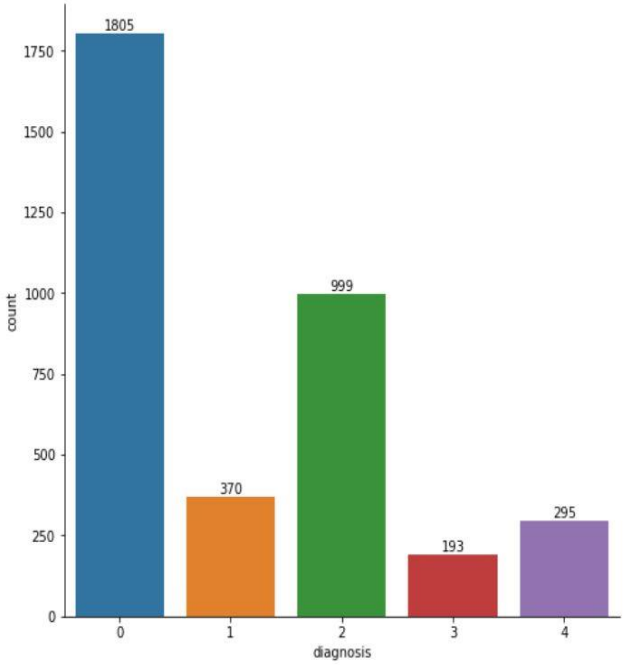


Figure: Train data distribution

2] Training

Transfer learning methodology has been applied for the training process. Transfer learning consists of taking learned features from one problem, which is the ImageNet dataset in our case and using the knowledge gained for a different dataset and problem. We begin by downloading the pre-trained ResNet-50, ResNet-101, DenseNet-121 and Xception networks without their fully connected layers. This is done because each network's fully connected layers can only accept a fixed image dimension as an input and the input size differs from network to network. Instead, we write a custom fully connected layer as follows:

```
x = GlobalAveragePooling2D()(base_model.output)
x = Dropout(0.5)(x)
x = Dense(2048, activation='relu')(x)
x = Dropout(0.5)(x)
final_output = Dense(n_out, activation='softmax', name='final_output')(x)
model = Model(input_tensor, final_output)
return model
```

Figure: Custom Fully connected layer

Dropout is used to drop a few neurons with random probability to prevent the model from overfitting. We freeze the entire network layers except for the custom FC layer to preserve the information and train it with a learning rate of 0.001 using Adam optimizer for a total of 3 epochs. The results are as follows:

Architecture	Epochs	Acc	Val Acc	Loss	Val Loss
ResNet-50	3	49.73%	45.89%	1.3187	1.3005
ResNet-101	3	51.32%	57.53%	1.2698	1.1950
DenseNet-121	3	69.44%	76.03%	0.8518	0.7107
Xception	3	72.58%	68.63%	0.7278	0.7597

Initial Training Results

The Xception architecture achieves the best results when compared to rest of the architectures. However, given that this is a Medical AI project, an accuracy of 72% is not satisfactory when performing classification on medical images. Hence, we now unfreeze the all the layers of the network and reduce the learning rate to 0.0001. 'Reduce Learning Rate on Plateau' metric has been applied to the network to monitor the validation accuracy parameter. This metric will reduce the learning rate stays the same even after 3 epochs. 'Early Stopping' metric has applied to the network to monitor the validation loss and this metric will stop the training process if the validation loss parameter remains constant even after 5 epochs. These metrics have been applied to prevent the model from overfitting. Using the Adam optimizer, we now train the networks for a total of 30 epochs. The results are as follows:

Architecture	Epochs	Acc	Val Acc	Loss	Val Loss
ResNet-50	21	92.00%	85.07%	0.0854	0.1585
ResNet-101	20	92.71%	83.70%	0.0750	0.1719
DenseNet-121	19	89.35%	84.79%	0.1138	0.1503
Xception	12	94.94%	82.74%	0.0572	0.1941

Finetuning Training Results

As it can be seen from the above table, all the 4 architectures have stopped the training process well before the designated 30 epochs due to the Early Stopping metric. The Xception architecture achieved the highest training accuracy and the lowest loss value but fell short when comparing the validation accuracy and validation loss values, suggesting it might be overfitting. The ResNet-50 and ResNet-101 networks achieved pretty similar results. The DenseNet-121 architecture registered the lowest training accuracy and highest loss value of all 4 architectures. All the trained model weights have been saved to a file in the .h5 format. The total number of trained parameters and the resulting size of the model weights file can be illustrated from the table given below:

Architecture	Trainable Parameters	Model Weights file size
ResNet-50	27,741,189	318.7 MB
ResNet-101	46,759,429	537.4 MB
DenseNet-121	9,063,301	106.1 MB
Xception	25,013,549	287.2 MB

3] Testing Process

In this section, we will talk about how the trained model is used for computing the predictions. For the training process, we begin by pre-processing the image data with the Gaussian blur filter operator like we did for the training process and store the processed images in a directory. Next, we load in the CSV file which contains the actual severity grade labels of the test data. Similar to the training process, we map the image data in the folder to the CSV file and resize all the images to 700x700 as the image dataset is of varying image sizes. Predictions are performed on the test data by issuing the model.predict_generator() function. The code outputs the list of predicted labels which are then converted into a dataframe and stored in the CSV file side by side to the 'Image ID' and the 'Actual' labels. The date and time parameter has been added while saving the CSV file so that the file does not get overwritten.

To visualize the predictions graphically and compare them with the actual labels we plot them on a bar graph using matplotlib and seaborn libraries.

VI. Building the EXE Application

Although running the Python script through the Command Line Interface (CLI) and the IDE is a good choice for testing the accuracy of the classifier and debugging any error in particular with the code in particular, it is not a good choice when the system has to be deployed in any environment and ran multiple times. Running the script multiple times through the CLI is not ideal as we have to navigate to the folder where it is stored and then run the python script. This

is particularly tedious and defeats the purpose of this proposed system which is to deliver the DR classification results speedily and with a high degree of accuracy. Also, the location where this system is to be deployed might not have the Tensorflow, Keras, deployed might not have the tensorflow, Keras, matplotlib, seaborn, pandas and numpy libraries installed. So, to overcome this problem, we will be converting the entire Python script and its associated dependencies into a single .exe application. To do

this, we will be making use of the PyInstaller.[20] library. PyInstaller bundles the python application, which is the python script in our case and all its dependencies into a single package which the user can run on multiple machines without needing to install the dependencies and even Python itself. The process of installation and conversion of the python script into quite simple. We begin by installing the Pyinstaller library into the directory where our python script is stored using the command prompt. Next, we issue the 'pyinstaller –onefile -w filename.py' command to create the executable application. The –onefile flag will tell pyinstaller to create a single exe application and the -w flag prevents the command prompt from popping up everytime the executable application is run.

VII. Conclusion

ACKNOWLEDGEMENT

I would like to thank my project supervisor Dr. Qianni Zhang for her invaluable guidance for making this project a success. I would also like to thank Dimitre Oliveira for his insightful python notebook: 'APTOS Blindness Detection-EDA and Keras ResNet50'which served as a useful reference for the implementation of my project(https://www.kaggle.com/dimitreoliveira/aptos-blindness-detection-eda-and-keras-resnet50).

VIII. References

1. https://www.healthline.com/health/diabetes
2. Wang, Xiaoliang, et al. "Diabetic retinopathy stage classification using convolutional neural networks." 2018 IEEE International Conference on Information Reuse and Integration (IRI). IEEE, 2018.
3. Abràmoff, Michael D., et al. "Automated early detection of diabetic retinopathy." *Ophthalmology* 117.6 (2010): 1147-1154.
4. Lam, Carson, et al. "Automated detection of diabetic retinopathy using deep learning." *AMIA summits on translational science proceedings* 2018 (2018): 147.
5. Carrera, Enrique V., Andrés González, and Ricardo Carrera. "Automated detection of diabetic retinopathy using SVM." *2017 IEEE XXIV international conference on electronics, electrical engineering and computing (INTERCON)*. IEEE, 2017.
6. Pratt, Harry, et al. "Convolutional neural networks for diabetic retinopathy." *Procedia computer science* 90 (2016): 200-205.
7. Roychowdhury, Sohini, Dara D. Koozekanani, and Keshab K. Parhi. "DREAM: diabetic retinopathy analysis using machine learning." *IEEE journal of biomedical and health informatics* 18.5 (2013): 1717-1728.
8. Janney, Bethanney, et al. "Detection and classification of exudates in retinal image using image processing techniques." *Journal of Chemical and Pharmaceutical Sciences* 8 (2015): 541-546.
9. Mateen, Muhammad, et al. "Exudate detection for diabetic retinopathy using pretrained convolutional neural networks." *Complexity* 2020 (2020).
10. https://www.kaggle.com/c/aptos2019-blindness-detection
11. https://towardsdatascience.com/a-short-history-of-convolutional-neural-networks-7032e241c483
12. Fukushima, Kunihiko. "Neural network model for a mechanism of pattern recognition unaffected by shift in position-Neocognitron." *IEICE Technical Report, A* 62.10 (1979): 658-665.
13. LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.
14. https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4
15. He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.
16. https://medium.com/analytics-vidhya/exploring-densenets-and-a-comparison-with-other-deep-architectures-85f02597400a
17. Huang, Gao, et al. "Densely connected convolutional networks." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
18. https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568
19. Chollet, François. "Xception: Deep learning with depthwise separable convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.
20. https://www.pyinstaller.org/

IX. Appendix

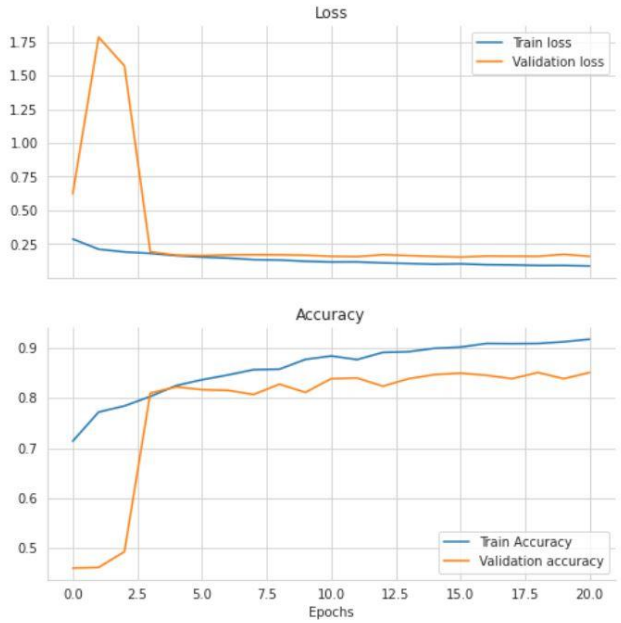


Fig.1: ResNet-50 Train Loss vs Val Loss & Train Acc vs Val Acc Graphs

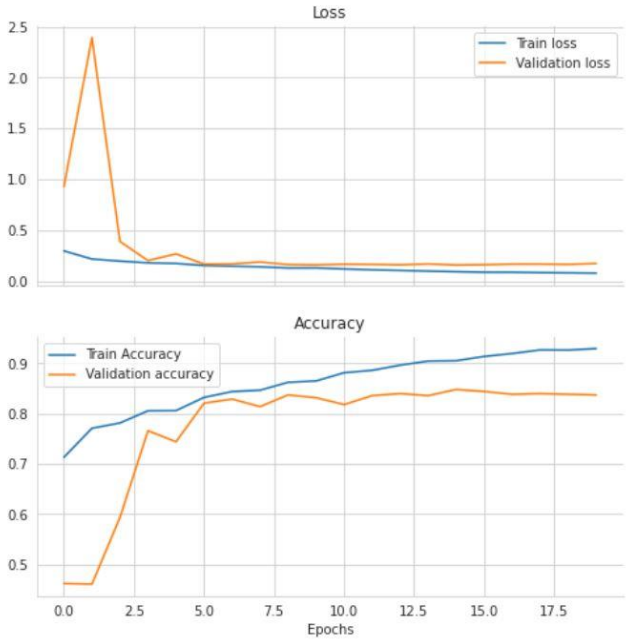


Fig.2: ResNet-101 Train Loss vs Val Loss & Train Acc vs Val Acc Graphs

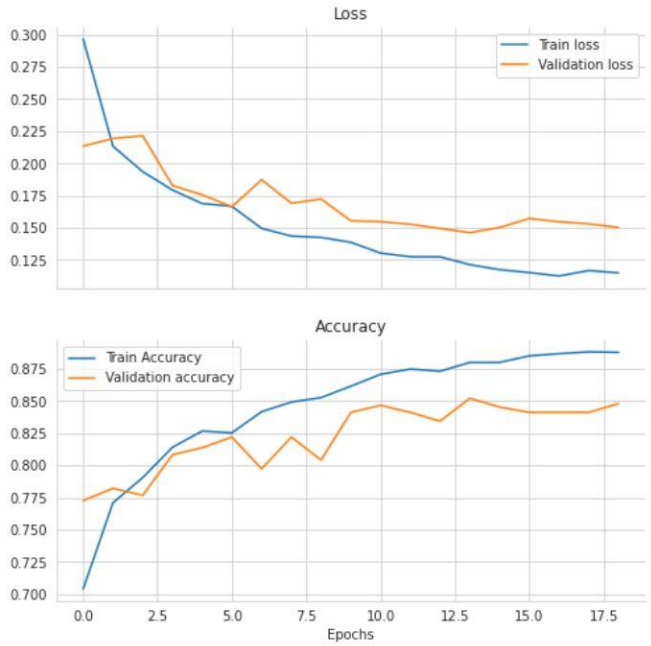


Fig.3: DenseNet-121 Train Loss vs Val Loss & Train Acc vs Val Acc Graphs

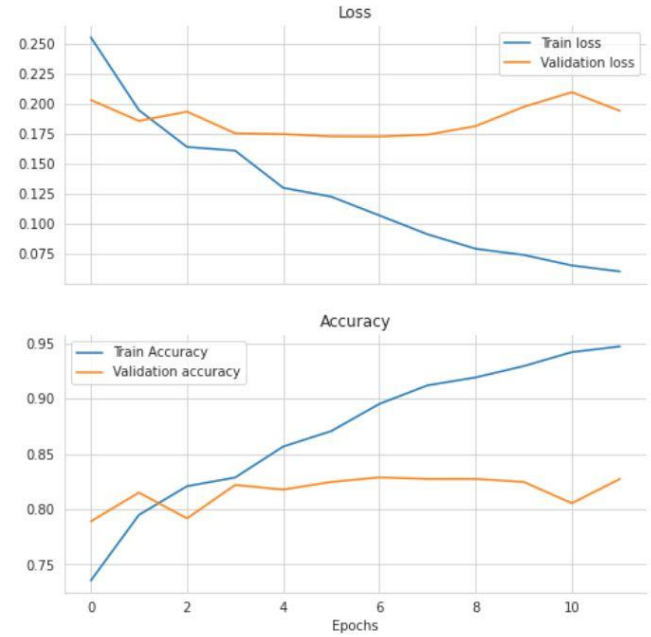


Fig.4: Xception Train Loss vs Val Loss & Train Acc vs Val Acc Graphs

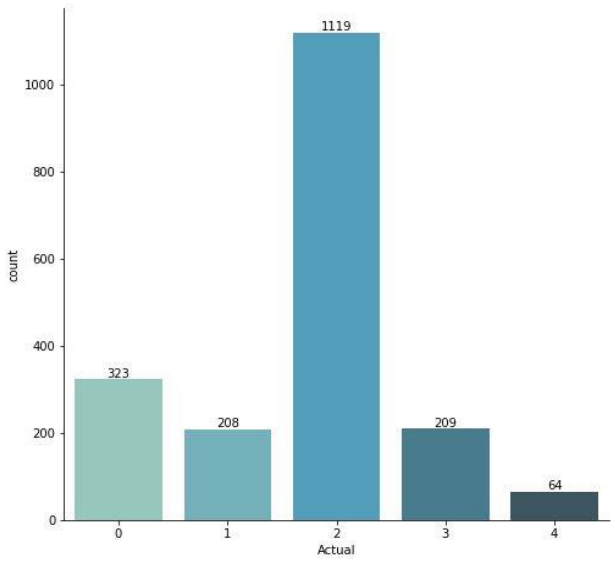


Fig.6: Test data actual labels distribution

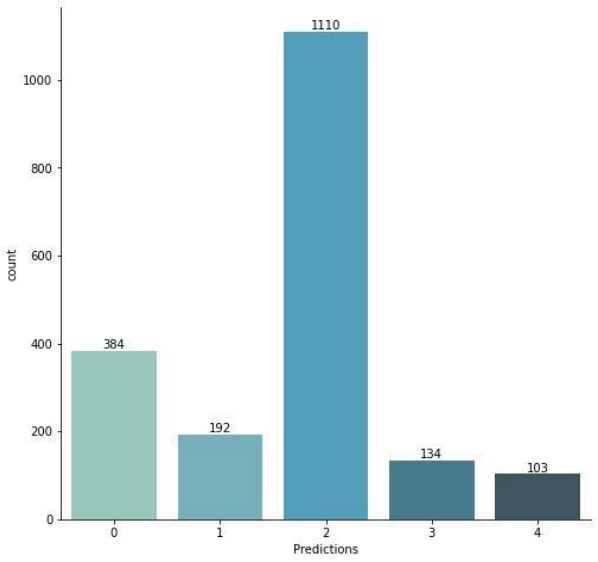


Fig.7: ResNet50 model predicted labels

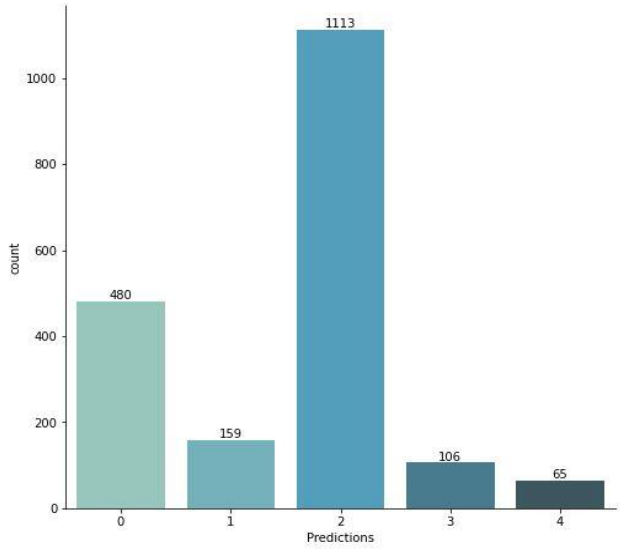


Fig.8: ResNet101 model predicted labels

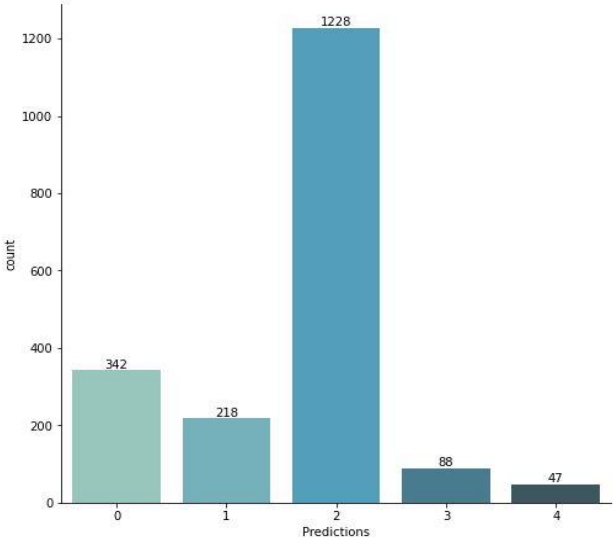


Fig.9: DenseNet-121 model predicted labels

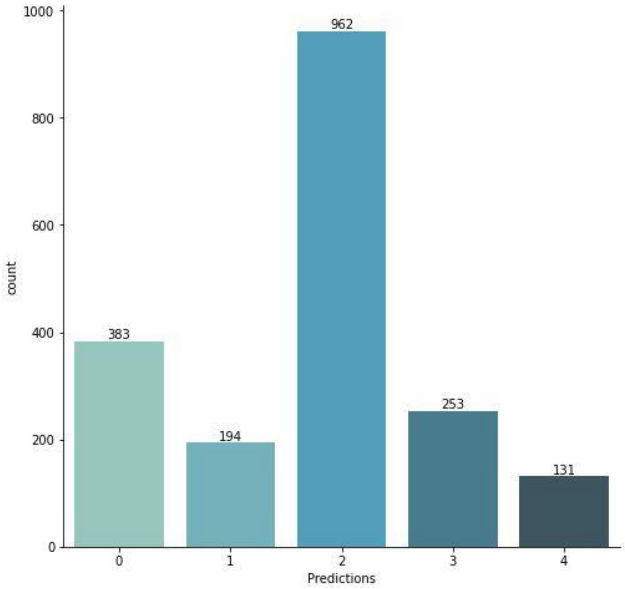


Fig.10: Xception model predicted labels