VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
STUDIJŲ PROGRAMA: INFORMATIKA

Machine Learning-Based Retinal Image Processing for Detection Possibility of Blindness

Mašininiu mokymusi grįsta akies dugno vaizdų analizė galimam apakimui nustatyti

Master‘s thesis

By: Mehmet Furkan Sarac

VU email: furkan.sara-@mif.stud.vu.lt

Supervisor: prof. dr. Olga Kurasova

Reviewer: dr. Jolita Bernatavičienė

Vilnius

2022

Summary

The purpose of the thesis is to provide a different point of view to perform diabetic retinal image classification problem and the influences of various preprocessing methods on deep learning methods to solve the problem of the detection of diabetic retinopathy. To experiment the preprocessing methods such as HSV, HSL, and Lab color spaces, EfficientNet and ResNet deep learning architectures are used. Conclusions and detailed experimental results are provided at the end of the research.

Key Words: Image Classification, Image Preprocessing, Color Spaces, ResNet, EfficientNet, Diabetic Retinopathy, Deep Learning, Machine Learning.

Summary

Baigiamojo darbo tikslas – pateikti kitokį požiūrį į diabetinės tinklainės vaizdo klasifikavimo problemą ir įvairių išankstinio apdorojimo metodų įtaką giluminio mokymosi metodams, sprendžiant diabetinės retinopatijos nustatymo problemą. Norint eksperimentuoti su išankstinio apdorojimo metodais, tokiais kaip HSV, HSL ir Lab spalvų erdvės, naudojami EfficientNet ir ResNet gilaus mokymosi architektūros. Išvados ir išsamūs eksperimentų rezultatai pateikiami tyrimo pabaigoje.

Pagrindiniai žodžiai: vaizdo klasifikacija, išankstinis vaizdo apdorojimas, spalvų erdvės, ResNet, EfficientNet, diabetinė retinopatija, gilus mokymasis, mašininis mokymasis.

Contents

1. Introduction

Diabetic retinopathy is a complication caused by diabetes in the retinal part of our body. This complexity is one of the main reasons for blindness [1]. High blood sugar levels cause the reasons for retinopathy, destruction of the retina becomes easier with the help of sugar anomalies in the blood, and an eye with diabetic retinopathy can be seen in Figure 1. Diabetic retinopathy has different stages, and early diagnosis plays an essential role to prevent blindness.

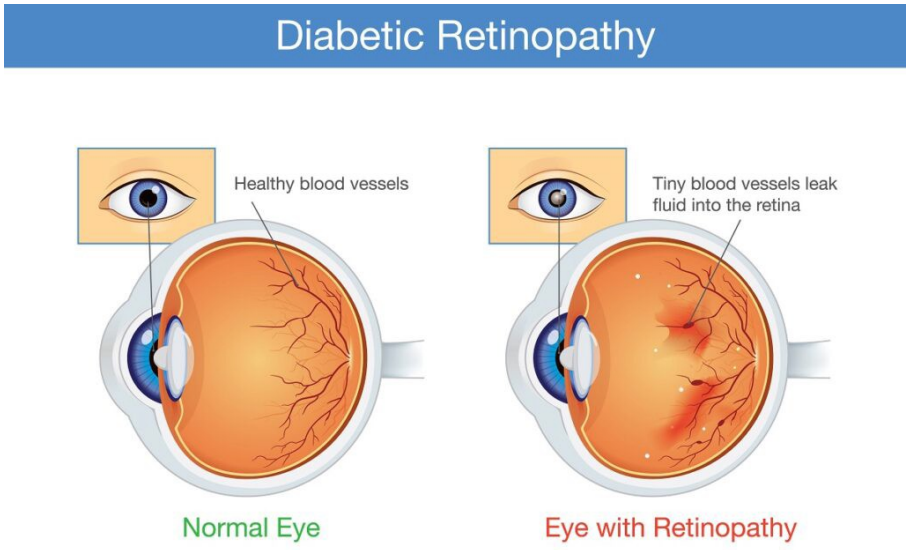


Figure 1. Comparison of healthy eye and eye with retinopathy [1]

With help of technology, it became easier to avoid these kinds of diseases year by year. The application of machine learning algorithms makes much easier the diagnosis process of illnesses including the possibility of blindness caused by diabetic retinopathy. In order to provide such an opportunity, image classification methods will be taken into consideration in this thesis. Classification of diabetic retinopathy is complex work and professionals should evaluate it. However, with the help of Convolutional Neural Networks (CNN) algorithms, we have a chance to predict the current situation of the disease. With the knowledge provided by previous research and experiments on diseases that causes blindness and machine learning, data preprocessing methods that might be useful for early diagnosis of diabetic retinopathy.

1.1 Problem Definition

Labeling images manually is one of the biggest problems while working on so much data and image classification applications with deep learning methods play essential roles to reduce workload and increase the efficiency and accuracy of the work. In the thesis, the aim is to investigate the influences of preprocessing methods mainly color spaces and other basic methods for the purpose of image classification with deep learning algorithms for eye fundus images. To examine the problem, experiments will be handled with the preprocessed training images. According to knowledge gained by previous research, different parts can affect the accuracy and efficiency of models and image preprocessing is one of the most important elements for purpose of image classification as well as blindness detection. Hereby, various image preprocessing methods related to color spacing such as HSV, HSL, LAB, and other hue-based methods will be applied to the images, and influences on the results obtained with deep learning methods will be evaluated at the end of the training with EfficientNet and ResNet architectures

There are some main challenges needed to be solved during the research:

1. **Investigation of Datasets**

   Some datasets like APTOS 2019 can have an imbalanced distribution of images through training and test files and to prevent bad effects of this issue, it is important to have a balanced dataset. Therefore, some image augmentation methods might apply to balance the image sets.

2. **Implementation of Preprocessing Methods**

   This task is the actual focus of the whole research and plays a crucial role in the further improvements of the experimental results. All deep learning architectures require different sizes of images and that's why resizing and other basic preprocessing methods will be used alongside color spacing methods.

3. Optimization of The Preprocessing Methods.

Most of the color spacing models focus on color in specific ranges that are manually determined and it is important to find the best range of colors for most images in the dataset.

4. Implementation of Deep Learning Architectures.

To have good test results, it is important to use some well-known and efficient proven models. Thus, it is decided to use EfficientNet and ResNet algorithms to have pieces of training. More explanations on this topic will be given in the further section.

5. Experimental Analysis To Investigate Influences of Image Preprocessing

To test the results, various experiments will be handled with various types of experiments with different image preprocessing methods mentioned in this thesis. Visualization and interoperation of the test results will be handled.

2. Literature Review

Diabetic retinopathy (DR) [1] is one of the common and dangerous reasons that can lead to blindness. The main complication is caused by high blood sugar in regards to diabetes and it damages the back of the eye where the retina is placed. Many things can cause this disease such as stress and life conditions. Early diagnosis of this treatment plays a crucial role to prevent blindness, however, it was not very easy to diagnose it in its early stages back then. Developments in technology and computer sciences have provided new opportunities for the diagnosis of DR as well as other medical research.

During the early stages of computer-based predictions of DR, support vector machines (SVM) [2] were popular to proceed. In 2013 *Adarsh et al.* [3] classified the severity levels of DR from mild to proliferate into 4 different classes by using multi-class SVM. Another method they used in that research was feature extraction but they handled this process manually based on morphological operators. However, they could achieve 95% average accuracy on the final version of the research. Another research held by *J.Calleja et al.* [4] has been completed by a two-stage method. In comparison to previous research mentioned earlier, *J. Calleja et al.* have used local binary patterns (LBP) for feature extraction and used SVM and random forest methods for classification. The dataset used during this research consists of 71 images and they had 97.46% accuracy by using the random forest for classification. Some other researchers are also focused on manual feature extraction with different methods for DR detection.

As time progresses, different points of view showed up in different researches. *Adityan Jothi et al*. [5] introduced a method to classify anomalies in the blood vessels that might cause blindness in fundus photographs provided at DIARETDB [6] and STARE [7] datasets. The Frangi filter technique is implemented in the algorithm for more accurate results in the research. Another methodis used by *T. Karim et al.* [8] and they could obtain improved sensitivity and accuracy results by using MATLAB Neural Network Pattern Recognition Tool (NPRTOOL). This research aimed to detect DR by detecting micro-aneurysm and the results are compared with other machine learning (ML) methods such as Naive Bayes and SVM.

To increase efficiency and accuracy obtained from the research, deep learning approaches became more popular over time. Many kinds of research are held with deep learning methods for automatic computer-based feature extraction, predictions, and classification. *I. Sadek et al.* [9] studied the topic of automatic DR detection by using methods. They aimed to classify DR into three different classes Normal, Exudates, and Drusen. To achieve this classification different experiments were held with ResNet [10], GoogLeNet [11], VGG-VD, and VGG [12] convolutional neural networks (CNN) on a custom dataset consisting of fundus images collected from six different datasets such as STARE, MESSIDOR [13], and other datasets. At the end of the research, they had 92% accuracy for the classification. Another research carried out by *Xu et al.* [14] has implemented data augmentation techniques onto CNN networks for DR classification and obtained results with 95% accuracy. Research carried out by *D. Doshi et al.* [15] completed the research with the kappa metric as an evaluation metric and this metric is different from than metrics used in the previous research mentioned before in this thesis. Their proposed method is completed by three CNN models and an ensemble learning model trained with those CNN architectures. They also used GPU computation and the aim of the research is to classify retinal images into 5 classes according to their severity levels. The best result they have achieved is provided by an ensemble model with a 0.3996 kappa score.

Some other researchers are focusing on image preprocessing methods, *Maya K. V. et al.* [16] used various image preprocessing methods such as Gaussian Noise, grey scaling, and green channel to detect bright lesions in the fundus images on the MESSIDOR dataset. To predict DR, they used a CNN architecture with 4 convolutional and 2 fully connected layers for the classification. The results of this study were achieved with 98% accuracy and 94.52% sensitivity. *G. Zago et al.* [17] also study on lesions. The purpose of the study is to design a lesion localization model with the help of VGG-16 CNN architecture that is customized with three dense layers. The model is pre-trained with the DIARETDB1 dataset and the tests were held with five different datasets, the best result for the DR screening was obtained with the MESSIDOR dataset with 91% accuracy. *Hagos et al*. [18] tried to use the transfer learning method by pre-training a model with InceptionNet-V3 and five different classification layers on the ImageNet dataset at the end of the research they achieved 90.9% accuracy. Another research held by *Sarki et al. [19]* in 2019 focused on pre-training of a model with ResNet-50, VGG, XceptionNets [20], and DenseNet [21] also with ImageNet dataset and could achieve 81.3% accuracy as a result of experiments.

In other research done in APTOS 2019 competition on the Kaggle website, Guanshuo Xu [22] has won that competition with 0.936129 Kappa score accuracy, and to get the first place in the competition, the ensemble learning method is used by blending InceptionNet and ResNet architectures. Resizing is the only preprocessing method used. It is noted that a larger input size provides better performance. For the loss function nn.SmoothL1Loss() is used and basic augmentation methods are used such as brightness, saturation, blur, mirror, rotation, and so on. The training part is divided into two parts, in the first stage, all 8 networks are trained and validated. To prevent overfitting, optimization of hyperparameters is handled in this stage. In the second stage, MESSIDOR and The Idrid datasets are added to training and lastly, by changing quadratic weighted kappa (qwk) thresholds from [0.5, 1.5, 2.5, 3.5] to [0.7, 1.5, 2.5, 3.5] best result is obtained.

For the second place, team [ods.ai] Eye of Private LB [23] got a 0.934310 kappa score by using pre-trained models with the 2015 competition dataset, and the most they could get 0.75 qwk score. Different network architectures are used but according to their research EfficientNet architecture is the best performed one and the final solution contains EfficientNet b3, b4, and b5 networks. Cropping black background and resizing are used as preprocessing methods and various augmentation methods are used such as blur, flip, random brightness, and others. Pre-training is done with old competition data by training 80 epochs with b3 models and 15 epochs with b5 models. Then, normal training is done on the 2019 competition dataset 50 epochs on b3 and 15 epochs on b5 architectures. The most important process during this research is pseudo-labeling previous competition test data and this process had a huge impact on the final result.

With a 0.933720 kappa score team Best Over Fitting [24] got fourth place in the competition with the method, “Crop From Gray” to the images of APTOS 2019 and APTOS 2015 datasets as preprocessing method for both training and prediction processes then followed by classified images into three different classes by their brightness of image boundaries of training images of both datasets and applied transformation to those images to make the same type images. They have also applied some augmentation methods such as Dihedral, RandomCrop, Rotation, Contrast, Brightness, Cutout, PerspectiveTransform, and Clahe. It is noted that he noticed, that large model training on high-resolution images tends overfitting due to the small size of the training set. So, he used a small model for high resolution and a large model for low resolution. The team pretrained the 2015 full dataset for 25 epochs without validation and then applied 5-fold cross-validation (CV)

to the 2019 dataset. EfficientNet B2, B3, B4, B5, B6, and B7 architectures are used in this research. Best results are obtained with an average of B7 and B5 architecture. For the prediction Dihedral Test-Time Augmentation (TTA) is used along with the CV. He did image preprocessing once for 40 tests (5 models X 8 Dihedral TTA) to save time.

Another competitor in the competition was Eugene Khvedchenya [25], he got the seventh place with 0.933155 kappa accuracy on the leaderboard, and he used, SeResNext50, SeResNext101, and InceptionV4 architectures are used. For the preprocessing different methods are used including Clahe, red channel dropping, and spherical image unwarping, but their performance was not as expected. According to the author's theory, all models are more complex than it seems (in several parameters) and have bigger capacity and generalization power so it leads vanishing impact on data. As a result, a simple preprocessing method is used by applying crop black areas out of the eye's boundaries, padding the image to a square size, and resizing it to 512x512 pixels. APTOS 2015 and a test set of IDRID datasets are pretrained. Before the training process, the sufficiency of the pooling method is considered and various pooling methods are tested. The global average pooling (GAP) method is performed better and applied for the rest of the research. In the training process, APTOS 2019, IDRID (train set), and MESSIDOR datasets are trained then, the Pseudo-labeling technique is applied to test datasets with MESSIDOR and APTOS 2014 datasets. Horizontal flip TTA is used for the final prediction.

The last competitor is reviewed in this report is team Eye of Stars [26], they got an eighth-place with 0.932106 accuracies by using datasets of APTOS 2019 and APTOS 2015 are combined and tried to improve the generalization part of this research so, this method has a different approach than the previous ones mentioned in this report. As other competitors did, also on this project the team have removed the black backgrounds of images. One other important task for them was to remove duplications and confusing labels from the datasets as the final preprocessing. Before the training, they applied some augmentation methods such as rotation, flip, zoom up to 1.35x, and lightning. It is noted that these augmentation techniques play a crucial role to generalize better and not overfitting. One cycle learning policy is taken into consideration for the training process and started with low-resolution images and continued with the high-resolution image. EfficientNet B4 and B5 architectures are used to train models and 5-fold cross-validation is applied.

According to observations I claimed during my research on previous works, most of the competitors that won prizes, focused on ensemble learning methods and did not focus a lot on preprocessing part of their research. Cropping images from black backgrounds and resizing are the most popular preprocessing methods that have been used during competition. Additionally, augmentation methods play a crucial part to have good experimental results due to imbalanced dataset issues. However, simple augmentation methods such as random rotating and mirroring images are applied commonly by competitors, it looks like they have good effects on the results. To train a model, EfficientNet, ResNet, and InceptionNet architectures are mostly used and combined to computer better and receive better results. To the words and results of research, ResNet has more stable and provides more satisfying results. However, they have mostly used basic image preprocessing methods as it is mentioned earlier. Hereby, it is hard to see the effects of color spacing methods on deep learning image classification for diabetic retinopathy diagnosis.

3. Preprocessing Methods

In this section, methods used for the purpose of preprocessing the retinal images are explained. Grey scaling, auto-crop, resizing and Gaussian blur methods are used for several small experiments to have an idea about the effects of preprocessing on the fundus images. However, the main aim of the thesis focuses on diabetic retinopathy detection by deep learning image classification, accordingly, to evaluate this topic, the effects of color spaces on image classification will be taken into consideration and some of the color space algorithms that will be used for experiments are explained in this section.

3.1 Grey Scaling

Grey scaling helps to reduce color differences between samples, thus makes easier to spot anomalies. With the help of the OpenCV library, this method is implemented in the research. Diagnose 3-severe sample is shown below, Figure 2 represents the original version of the image from the dataset, and Figure 3 denotes the converted version of the same sample.

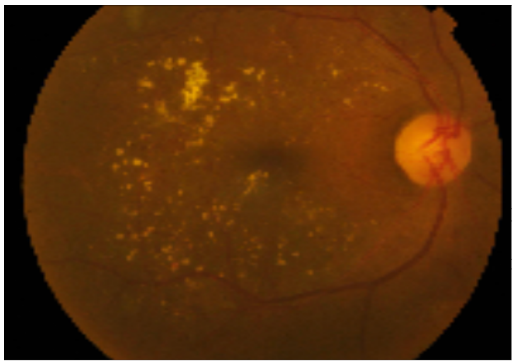


Figure 2. Original Severe Diagnose

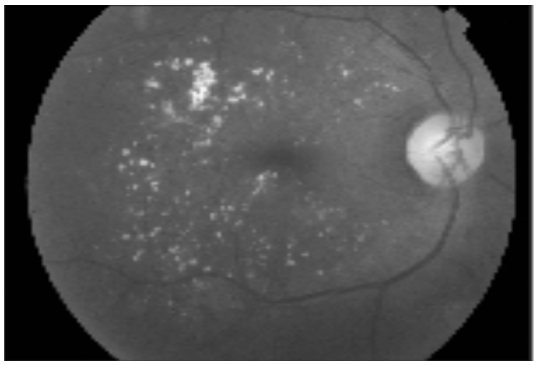


Figure 3. Grey Scaled Version of Image

As it can be seen from the examples in Figure 3, It is possible to observe blood vessels and leaks more clearly on the grey scaled fundus image. This method was the initial technique for preparing the dataset for model training.

3.2 Gaussian Filter

Gaussian filter is a 2D convolutional operator to blur and reduce noises from the images [27]. This method can be calculated with equation (1) which is given below;

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}} \quad (1)$$

Where shows the standard deviation of the distribution, x and y represent distances to the origin of the pixels and σ is the standard deviation of the Gaussian distribution. Depending on, the blurring level changes, greater values increase the blurring degree. In this part of preprocessing Gaussian filter is applied to the images that were converted into greyscale.

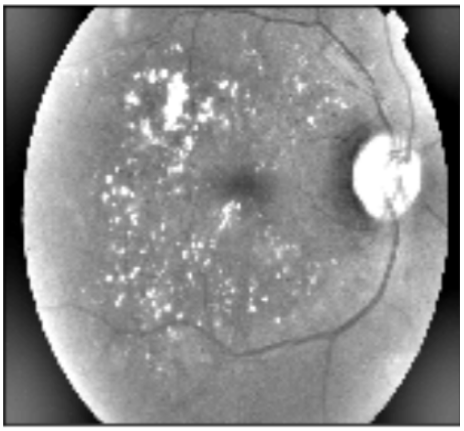


Figure 4. Gaussian Filter Applied Image

In comparison to Figure 3, observation of details becomes easier on the Gaussian filter applied version of the grey scaled image in Figure 4. Some unseen or hard-to-see anomalies are easier to detect but there are still some useless parts that might affect the result of the model training process.

3.3 Auto Crop and Re-Coloring

To focus on the retinal part of the image, all the unnecessary parts like grey or dark zones out of the fundus might be cropped for better training results. This process is followed by re-coloring part of the grey-scaled images. According to research on the Kaggle website I have seen the previous winner of the 2015 APTOS competition on diabetic retinopathy, Ben Graham [35] has used a method to improve lighting quality. During the application process of color cropping,

sigmaX parameter plays a crucial role in color images. It is noted that Ben Graham used sigmaX = 10 during his experiments and that parameter is not changed in the experiments performed in this thesis.. This method applies color to every part of the image with the grey color according to the depth of the grey color's scaler dimensions. Examples of recolored and cropped images for different sigmaX values from the train set are attached to Figure 5 and Figure 6 below.

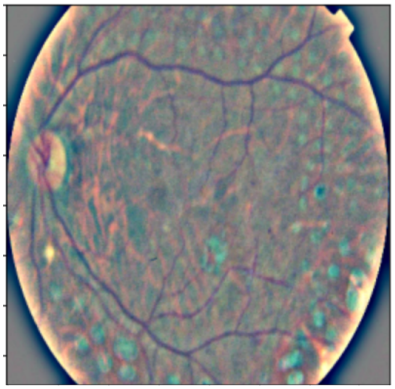


Figure 5. Recolored and Cropped Image with sigmaX = 10

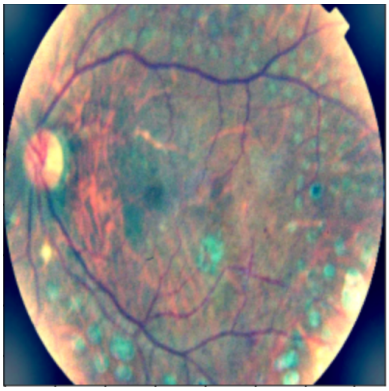


Figure 6. Recolored and Cropped Image with sigmaX = 30

Effects of the sigmaX parameter can be seen easily on the indicated images. Damages and blood vessels on the retina become more visual than in the previous versions of the samples from the dataset. With the methods applied for preprocessing, it is expected to have meaningful results at the end of the initial experiments.

3.4 HSV Color Space

HSV color space is a color model like the RGB model to describe the way colors combine to create the spectrum we see. The difference between RGB and HSV is, that HSV is more close to how humans see the world and H stands for hue, S saturation and V is value. The main reason to use this method is actually to spot the anomalies easier on the fundus images provided on the dataset. With the help of the HSV method, separation and segmentation of different parts of the images become easier but thresholding colors is should be taken into consideration to have proper results.

Figure 7 shows a fundus image from APTOS 2019 dataset in order to have an idea about the threshold values we need to examine the image in more detail with RGB and HSV histograms.

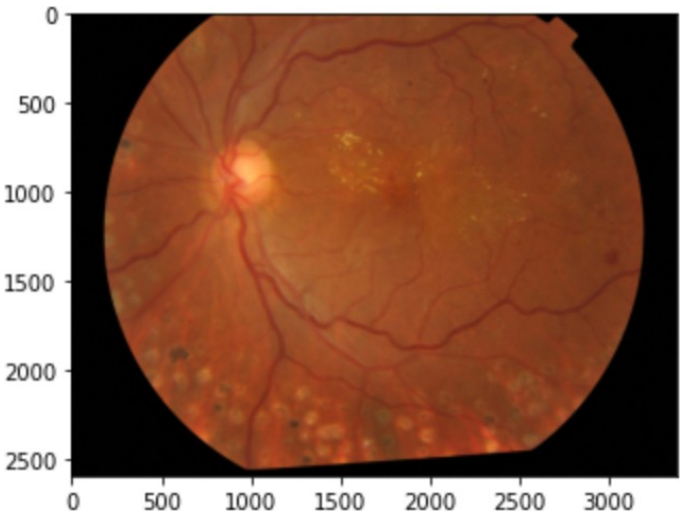


Figure 7. Fundus Image From APTOS 2019

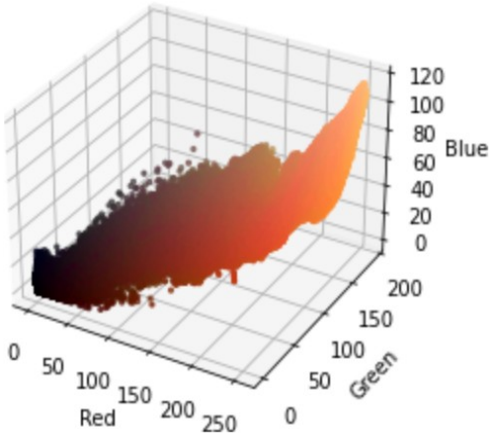


Figure 8. RGB Histogram

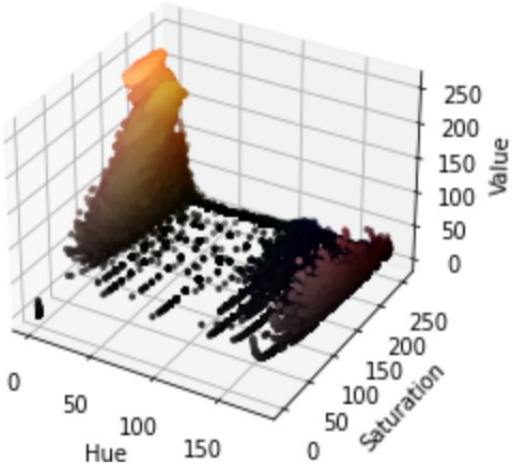


Figure 9. HSV Histogram

Figure 8 and Figure 9 show color histograms of the fundus image provided in Figure 7. To decide on threshold values, the HSV histogram gives an idea. Here in Figure 9, we can easily see that our boundaries should be between grey and orange color but it can change for other images in the dataset, however, most of the images have this color scale so thresholds will be in this range.

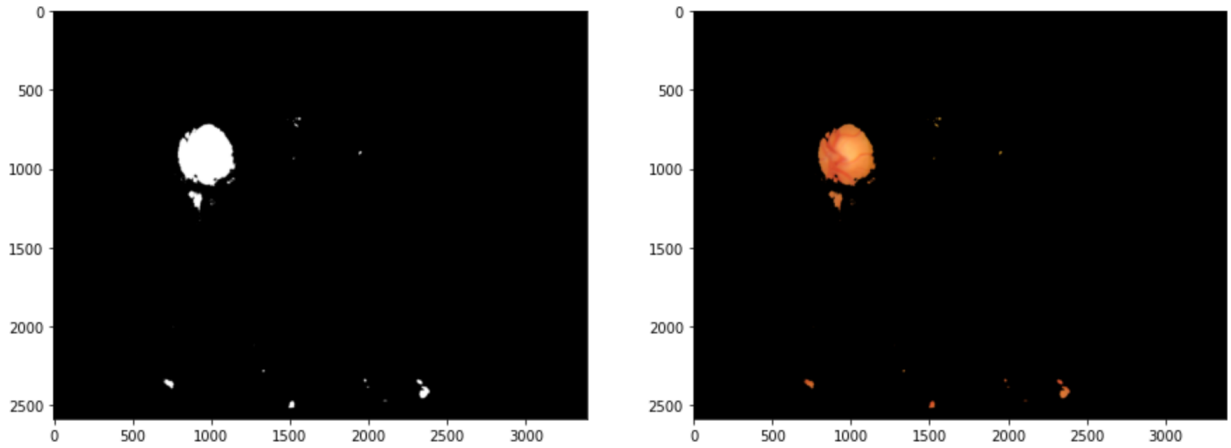


Figure 10. Masking

Figure 10 shows two different masks of the fundus images after thresholds are applied. With masking, we will be able to capture white and orange colors on the intended areas of the retina.

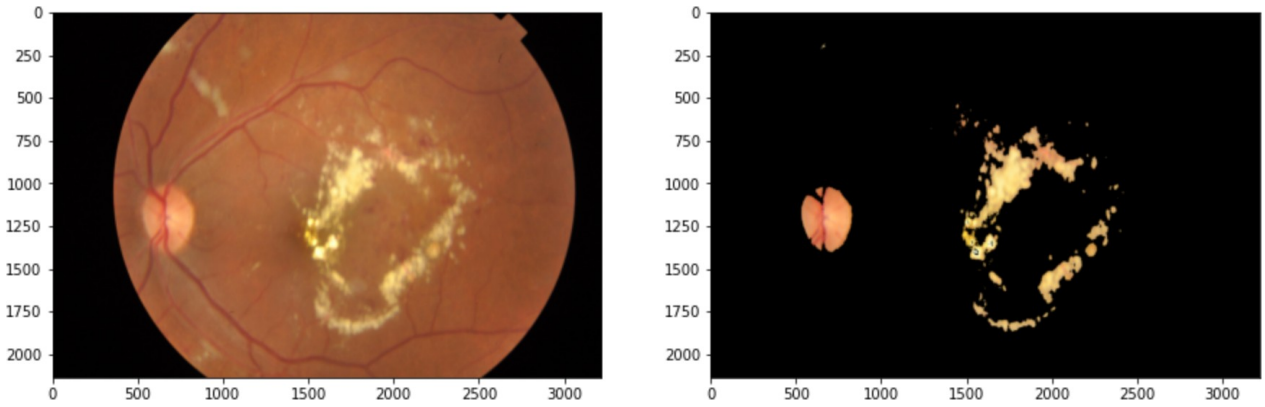


Figure 11. An Example of Thresholded Fundus Image

We can see a good result provided after HSV color spacing in Figure 11. However, some of the photos are not that successful but promising.

3.5 HSL Color Space

HSL is a cylindrical color model and has similarities with HSV color space. Hue and Saturation values are the same with HSV, however, the L letter stands for the value of light and it is the difference between HSV and HSL color spaces. The role of the hue is to specify the angle of the color on the RGB space as well as HSV and purity of the color check by saturation and this is also the same in HSV color space. As mentioned earlier, lightness is the only difference between HSV and HSL and the role of that value is to control the luminosity of the color if the color has a 0% value it means that the color is black, when that value gets higher color gets close to white.

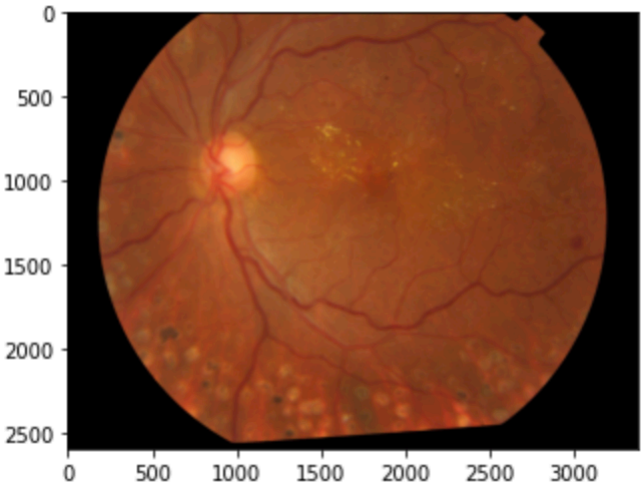


Figure 12. RGB Fundus Image

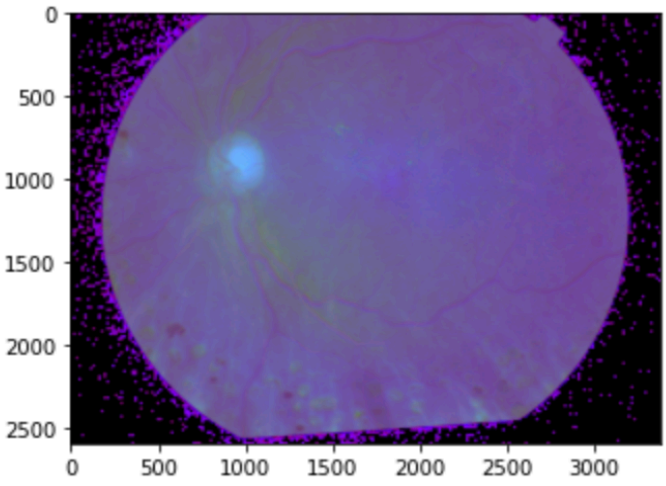


Figure 13. HSL Filtered Image

Figure 12 and Figure 13 show the difference between RGB fundus images and HSL filtered images. It is easy to observe the color difference between example images. HSL filter takes hue in the red channel, saturation in green, and luminosity in the blue channel and converts it as can be seen in Figure 13.

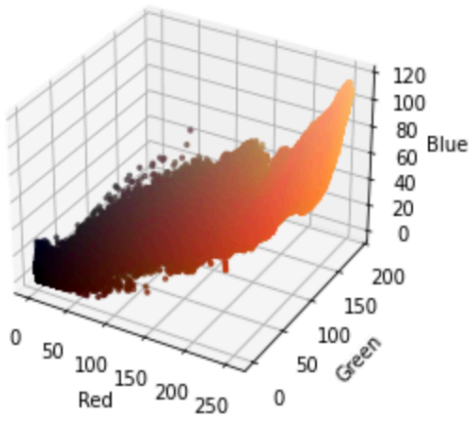


Figure 14. RGB Histogram

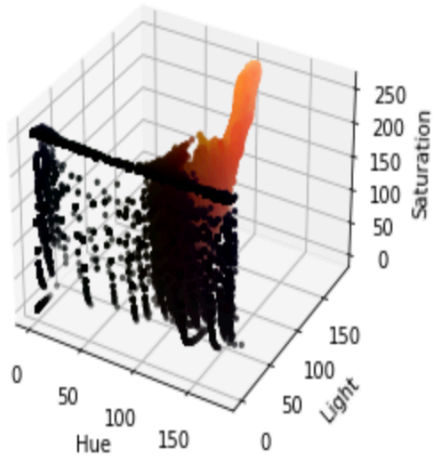


Figure 15. HSL Histogram

Figure 14 and Figure 15 show the histogram of the distribution histogram of the values of both RGB and HSL filters. These pieces of information can be used while deciding the threshold values if desired to have images in Figure 16.

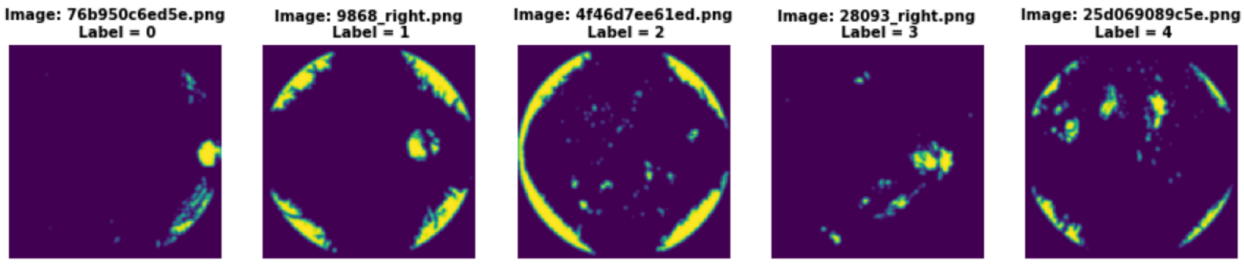


Figure 16. HSL Thresholded Fundus Images

Figure 16 shows the HSL filter applied and thresholded fundus images from every label from the preprocessed custom dataset (see section 6.3). With this method, it is easier to focus on the problematic parts of the diabetic retinal images.

3.6 L*a*b Color Space

The L*a*b* color space is invented by Richard Hunter in the 1900s. This method is proper to quantify the colors. 3 axis of the LAB color space represents light (L), red to green (a), and green to yellow (b). If values of the axis a* are negative, it means the color is green else red and the same thing can be said for the axis b* but the color blue is negative and yellow is positive. Axis L changes between 0 and 100 and shows the value of the lightness. Moreover, the L value shows the contrast between gray and black colors where the a and b axes show chromatic colors of the respective elements of the colors by showing the closest color and it is good to understand which side is the color closer on the color scale. Figure 17 and Figure 18 show the difference between RGB and Lab filter on fundus images from APTOS 2019 dataset.

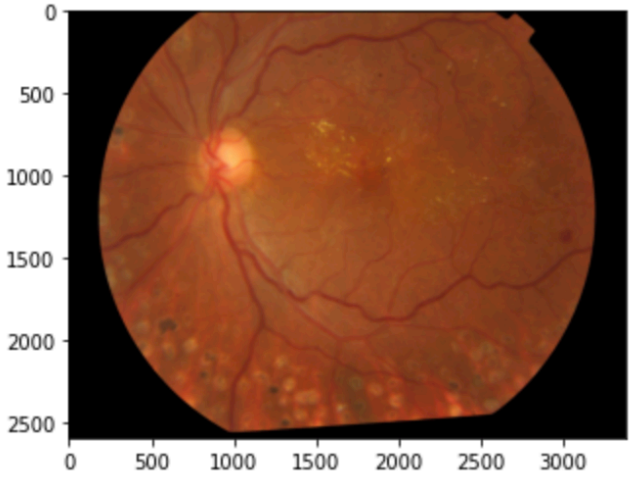


Figure 17. RGB Fundus Image

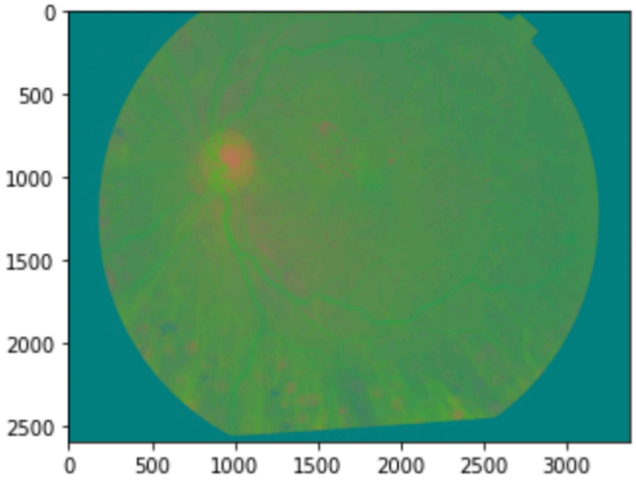


Figure 18. L*a*b* Filtered Image

Figure 19 and Figure 20 show the histogram of the colors on the fundus images and those histograms help us a little more to understand color values on the provided retinal images.

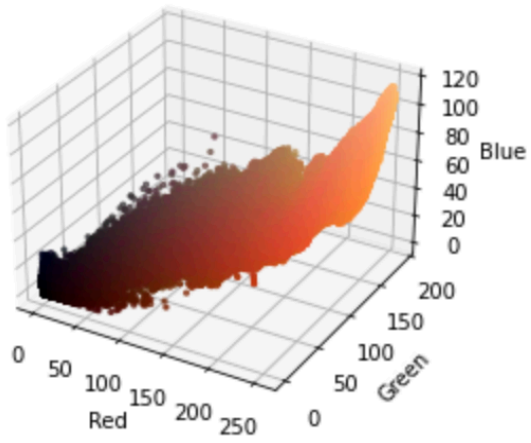


Figure 19. RGB Histogram

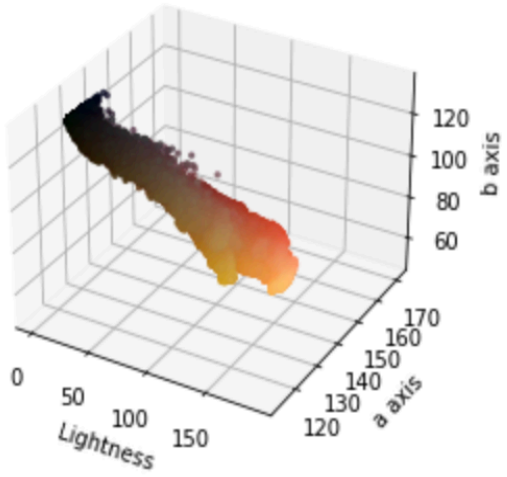


Figure 20. LAB Histogram

4. Deep Learning Algorithms

In this section, network architectures that will be used during experiments are explained. According to the literature review and APTOS 2019 competition solutions mentioned in the earlier sections of this report, it is seen that some deep learning architectures are used more than others and their accuracy levels are higher and more effective in the classification of images. Most of the solutions in the APTOS competition are considered ensemble learning methods and combined couple algorithms such as ResNet, EfficientNet, InceptionNet, and so on to increase accuracy and efficiency. D. Doshi et al. [15] were another ensemble learning method appliers for their research and they took help from three different architectures and applied ensemble learning. I. Sadek et al. [9] made research with ResNet, GoogleNet, and VGG algorithms on the custom dataset and had good accuracy results at the end of the research. Modern image classification methods and architectures for diabetic retinopathy detection will be taken into consideration to complete the tasks. Figure 12 represents the performances of the deep learning architectures below. ResNet, and EfficientNet has the best results for most of the research, so these architectures are implemented into the research to make reasonable recommendations for diabetic retinopathy detection. Additionally, different methods like pseudo labeling test time augmentations are the other methods that might have a crucial role to increase and make observations over the data.

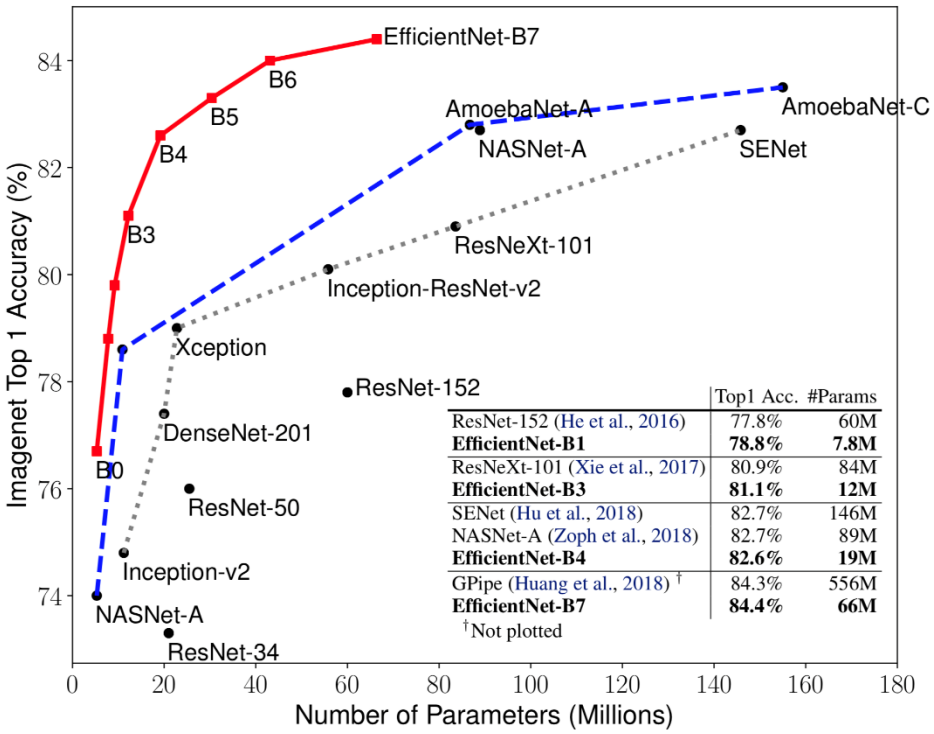


Figure 21. Network Accuracy Comparisons [28]

4.1 ResNet Architecture

Residual Network (ResNet) was introduced first in a paper named Deep Residual Learning for Image Recognition [29] in 2015 by Kaiming He, Jian Sun, Shaoqing Ren, and Xiangyu Zhang and the popularity and success of the model have increased since then. In the paper, the authors are also explained residual blocks, and ResNet is created by these blocks.

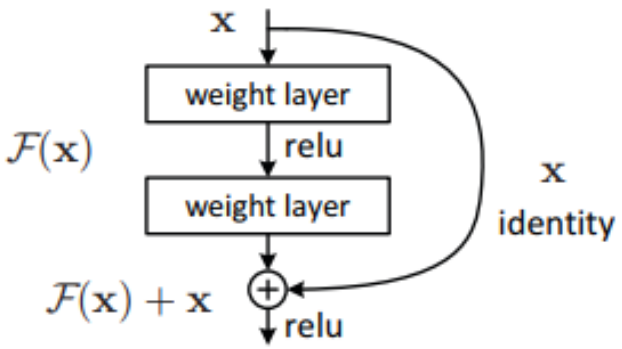


Figure 22. Residual Block **[29]**

Figure 22 shows the architecture of residual blocks. One of the most important parts of these blocks is a type of connection called “skip connection”. This method has an essential role to solve the vanishing gradient problem by letting gradients use a shortcut to flow. The stop process applies to negative weights, preventing them to pass to the convolutional layer. Those negative weights pass to the ReLu activation function and in this way, we are able to reduce calculation steps and parameters. Figure 23 shows the relations between VGG-19, 34 layer plain network, and 34 layers residual network. Basically, the plain network is inspired by the VGG-19 network, and then skip connections are added to the plain network to create a residual network.

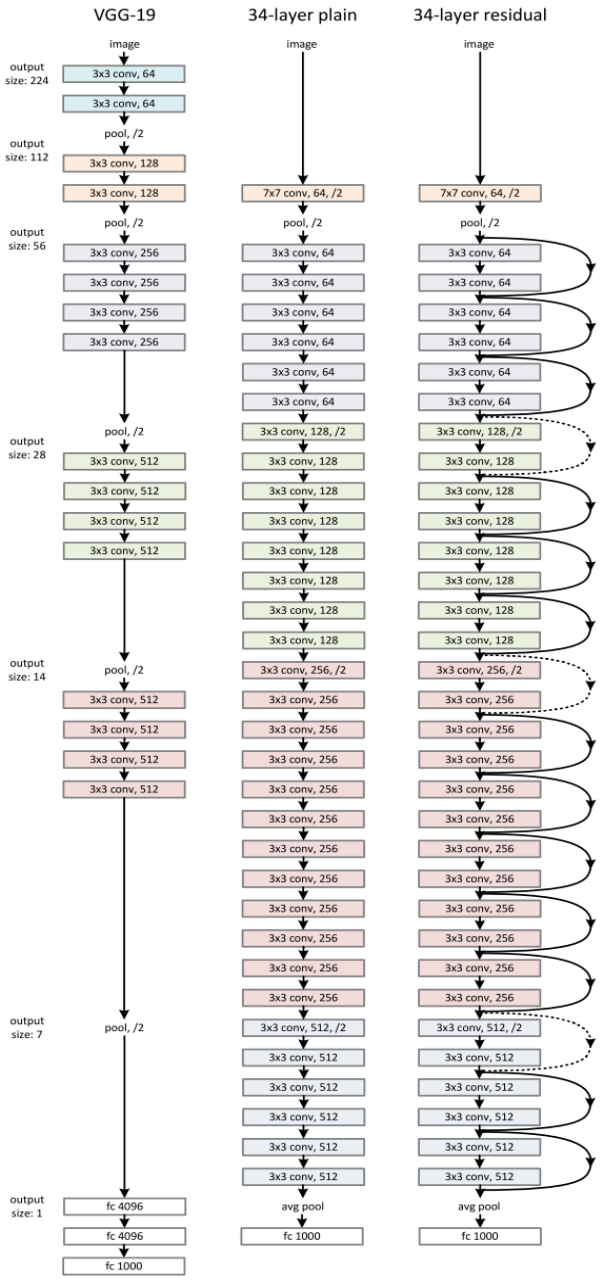


Figure 23. ResNet Architecture [29]

4.2 EfficientNet Architecture

According to the research made in the previous research work report, EfficientNet is a Convolutional Neural Network (CNN) model that achieves high accuracy level after its release in comparison to other CNN methods. It is first described by Mingxing Tan and Quoc V. Le in [28] and relies on the AutoML framework. This framework has provided an opportunity to create EfficientNet B0 then it's improved by the compound scaling method to develop the rest of the EfficientNet family from B0 to B7. EfficientNet uses a compound coefficient to scale all dimensions of depth, width, and resolution. However, scaling can cause some effects on accuracy. Scaling up dimensions, width, depth, or resolution may increase accuracy, also, it is important to have better accuracy and efficiency, and it is essential to balance all mentioned aspects. In the paper proposed for EfficientNet by Mingxing Tan and Quoc V. Le, the compound scaling method is used to scale the dimensions of the network. The researchers have executed a grid search method to find out the relations between various scaling sizes within the fixed resource constraints of networks with the help of this strategy they were able to find convenient scaling coefficients for every dimension to be scaled-up. Depending on these coefficients, the baseline network is scaled to the desired size. At the end of their experiments, they claimed that the compound scaling method has improved the model accuracy and efficiency. The method of compound scaling is illustrated in Figure 24.

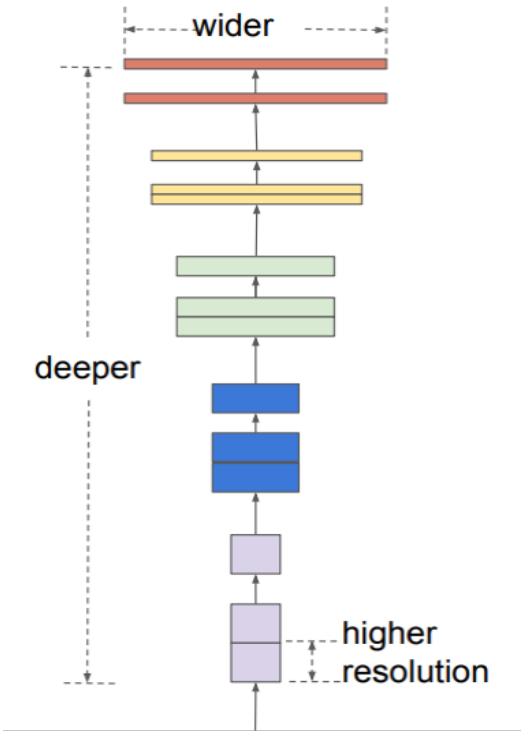


Figure 24. Compound Scaling Method [28]

Stage i	Operator F_i	Resolution $H_i \times W_i$	Channels C_i	Layers L_i
1	Conv3x3	299 x 299	40	1
2	MBConv1, k3x3	150 x 150	24	2
3	MBConv6, k3x3	75 x 75	32	3
4	MBConv6, k5x5	38 x 38	48	3
5	MBConv6, k3x3	19 x 19	96	5
6	MBConv6, k5x5	19 x 19	136	5
7	MBConv6, k5x5	10 x 10	232	6
8	MBConv6, k3x3	10 x 10	384	2
9	Conv1x1	10 x 10	1536	1
10	AveragePooling & dropout	2 x 2	1536	1
11	Dense	1 x 1	8	1
12	Sigmoid	1 x 1	8	1

Figure 25. EfficientNet Structure [30]

The model in Figure 25 was designed by Jing Wang et al. [30] for the purposes of classification of the fundus images.

5. Performance Metrics

It is important to calculate the performances of the models used to train the datasets. To perform a good performance test, accuracy, sensitivity, and specificity should be calculated. The confusion matrix gives chance to find values of true positive (TP), true negative (TN), false positive (FP), and false-negative (FN). To explain these values we can give example from the retinal image classification process from the Messidor dataset [31]. This dataset consists of four different types of attributes such as healthy retina, DR. Stages 1,2, and 3. If an image is predicted correctly from a specific class then it represents true positive and it will be true negative for all other correct predictions for alternative attributes. When a healthy retina image is predicted as a DR image of any stage then it is termed a false positive. And lastly, if a DR input image of any stage is predicted as a healthy retina, so it represents a false negative. We can calculate all metrics from the following equations (2,3,4).

$$Accuracy = \frac{|TP| + |TN|}{|TP| + |TN| + |FP| + |FN|} \quad (2)$$

$$Specificity = \frac{|TN|}{|TN| + |FP|} \quad (3)$$

$$Sensitivity = \frac{|TP|}{|TP| + |FP|} \quad (4)$$

With the help of the knowledge mentioned earlier in this section, it is possible to calculate desired metrics such as precision, recall, and F1 score. These three metrics can be counted as a part of a union and should be evaluated together. Precision gives the answer of between all positive predictions how many of them are really positive can be calculated on the equation (5),

$$Precision = \frac{TP}{TP + FP} \quad (5)$$

Recall, answers how much data is predicted positive between all real positive cases and can be calculated with the help of formula (6) as follows,

$$Recall = \frac{TP}{TP + FN} \quad (6)$$

The final part of this union is the F1 score which takes into account both precision and recall to calculate its value as on the equation (7) below,

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (7)$$

F1 score or also called F-measure basically combines precision and recall as can be seen in formula (7) and changes between 0 and 1 depending on precision and recall. The higher the F1 score goes, means better model we have to evaluate. Another metric is called F-beta score (8), this metric allows us to decide how to weight the balance between precision and recall using the beta parameter. When beta=1, the F-beta score is equivalent to the F-1 Score. When beta=0.5, this score is the F-0.5 score.

$$F\text{-}beta\ score = \frac{((1+beta^2)*Precision*Recall)}{beta^2*Precision+Recall} \quad (8)$$

5.1 Quadratic Weighted Kappa

This is another metric used during the experiment part of the research and this method will be explained in this section.

Quadratic weighted kappa (QWK) is a kind of classification accuracy and works well with unbalanced data by guessing randomly according to the frequency of each class, so it reduces the problems caused by imbalanced data. It basically tells how well the classifier performs and QWK calculates relations between two parameters. In order to measure kappa value, attribute parameters should be assigned by a researcher. Additionally, predicted results are also needed to calculate the kappa score. This metric might vary between 0 and 1. Interpretation of kappa scores is given in Figure 26 below;

Value of Kappa	Level of Agreement	% of Data that are Reliable
0 - .20	None	0 - 4 %
.21 - .39	Minimal	4 - 15 %
.40 - .59	Weak	15 - 35 %
.60 - .79	Moderate	35 - 63 %
.80 - .90	Strong	64 - 81 %
Above .90	Almost Perfect	82 - 100 %

Figure 26. Interpretation of Cohen's Kappa [32]

Calculation of quadratic weighted kappa is handled as follows:

$$\kappa = 1 - \frac{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} o_{ij}}{\sum_{i=1}^{k} \sum_{j=1}^{k} w_{ij} e_{ij}}, \quad (9)$$

$$w_{ij} = \frac{(i-j)^2}{(k-1)^2}, \quad (10)$$

where k represents the number of categories o_{ij} *that* stands with observed e_{ij} and is expected matrices (9). To calculate QWK, we need to calculate the confusion matrix to find prediction results for the classification problem. It is mentioned in previous researches that the calculation of the weight matrix w_{ij} (10) plays a crucial role to have proper calculation. A couple of problems like raters with the same percentage of an agreement, but different proportions of ratings might cause drastic changes in the kappa score.

6. Datasets

Here in this section, the datasets used for the research will be explained. Explained datasets are chosen concerning factors of quality and quantity of images, and detailed labels of diagnosis. Additionally, those datasets are newer in comparison to most of the other datasets and that was another reason to choose explained datasets in upcoming parts of the thesis.

6.1. APTOS 2019 Dataset

This dataset is published by the Asia Pacific Tele-Ophthalmology Society (APTOS) for the competition held on the Kaggle platform to detect diabetic retinopathy. The image set contains 3662

train and 1928 publicly available test images. Imbalanced training images in this dataset are the biggest problem for the training and can be seen in Figure 27.

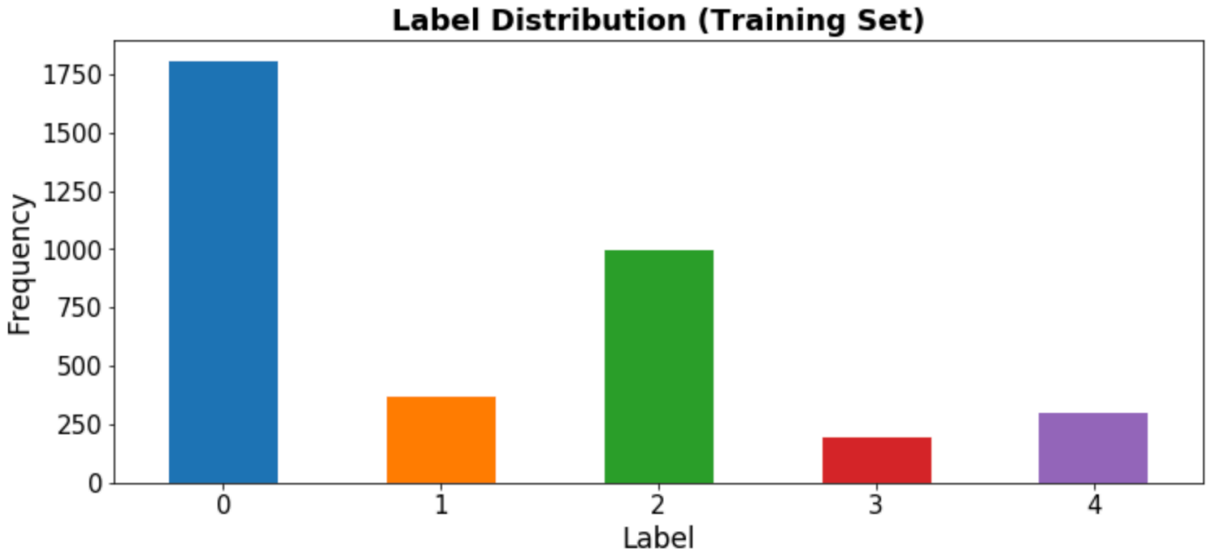


Figure 27. APTOS 2019 Dataset Training Images Distribution

Figure 27 shows the labels from 0 to 4, label 0 represent the healthy samples, and label 4 proliferative DR samples and it is easy to observe that label 0 has much more samples than the others and it causes an imbalance dataset problem as it is mentioned earlier. In order to prevent this issue, basic augmentation methods are applied before the training such as rotation, and horizontal and vertical flips. Images from every label can be seen in Figure 28.

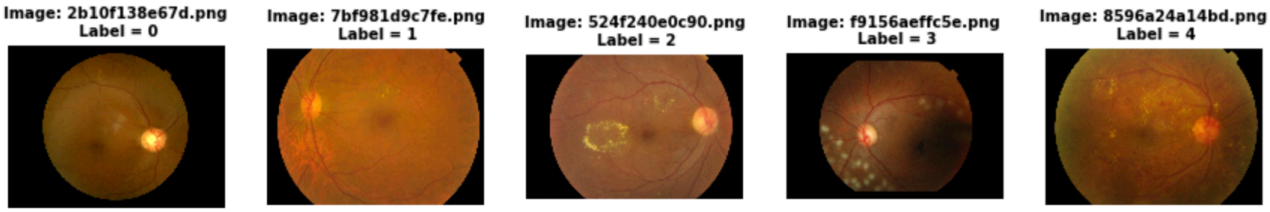


Figure 28. APTOS 2019 Dataset Example Images From Every Label

Since every specific deep learning architecture requires a specific image size, it is important to resize the images before the training as a part of preprocessing part of the experiments. Furthermore, it is possible to observe that the images have some differences in their sizes and it is essential to make them equal. From label 2 to label 4 it is easy to detect the anomalies that cause retinopathy on the images.

6.2 APTOS 2015 Dataset

This dataset was also published by APTOS and used for the diabetic retinopathy detection competition 2015 as well as the 2019 dataset. Images are collected by the fundus photography method and published publicly. As mentioned in the APTOS 2019 dataset explanation part, this dataset also has a big imbalanced data problem and it can be seen in Figure 29.

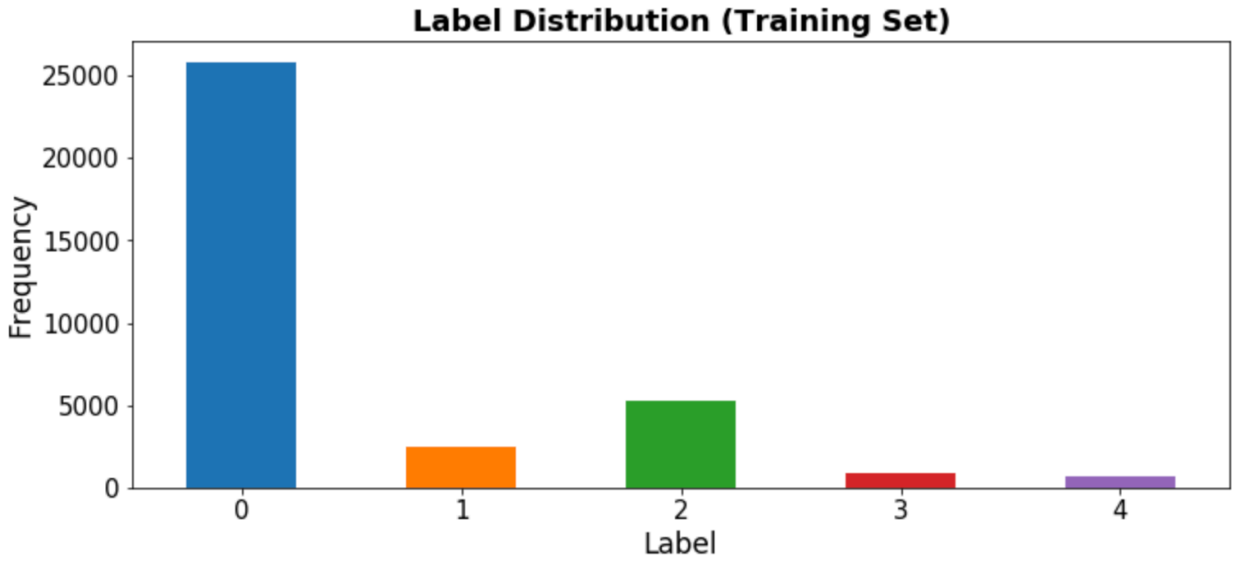


Figure 29. APTOS 2015 Dataset Training Images

In comparison to the 2019 dataset, this dataset has significantly much more images. Figure 29 shows the labels of the dataset and again label 0 has over 25000 images in training images and has very few labels 4 and 3 to have good results at the end of the training. Figure 30 shows example images from the dataset.

Figure 30. APTOS 2015 dataset example images

Apart from the images from the 2019 dataset, the 2015 dataset has images with different color scales and this is important to define thresholds for the color spaces, more information will be shared in the following sections.

6.3 Custom Balanced and Preprocessed Dataset

To prevent or reduce to chances of overfitting and other reasons for bad training results, this dataset is created by a Kaggle user called Alinusik [33] and used in this research to have a bit more reliable results. The image set consists of both APTOS 2019 and 2015 images, as it can be observed from the Figure 27 and Figure 29, labels except from 0 do not have many images and with the combination of these two datasets this problem is solved and it is possible to see it in Figure 31.

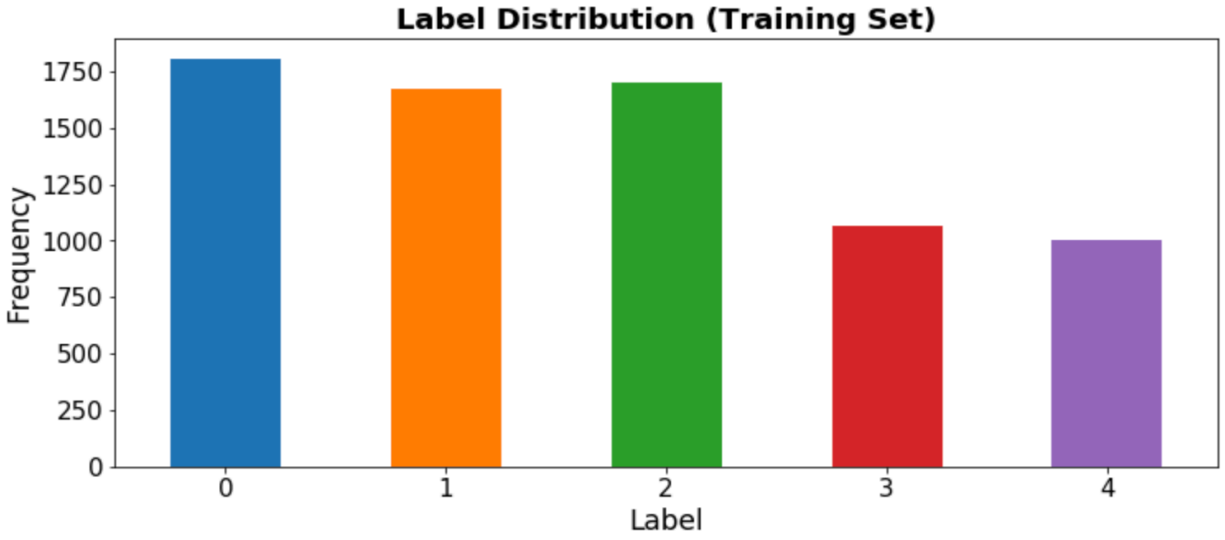


Figure 31. Custom Dataset Training Images Distribution

There are 7243 training images in total in this dataset and 2500 images for the testing. In comparison to previous image sets, this dataset is much more balanced as it is shown in Figure 23. But, it is not the only difference between the datasets. Images in this dataset are preprocessed with various methods such as conversion into gray, cropping unnecessary parts outside the fundus, resizing, and lastly gaussian blur to reduce noises on the images. Examples can be seen in Figure 32.

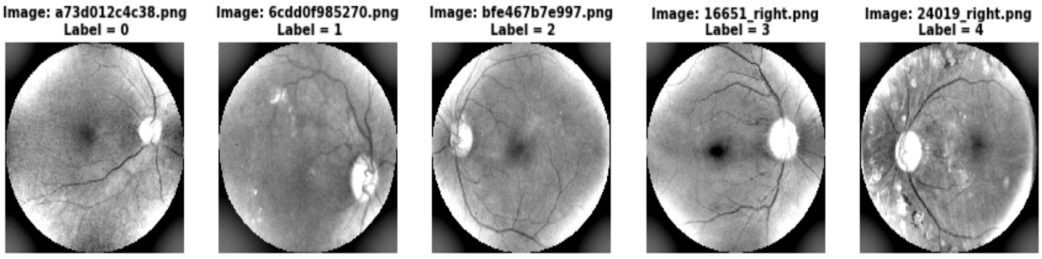


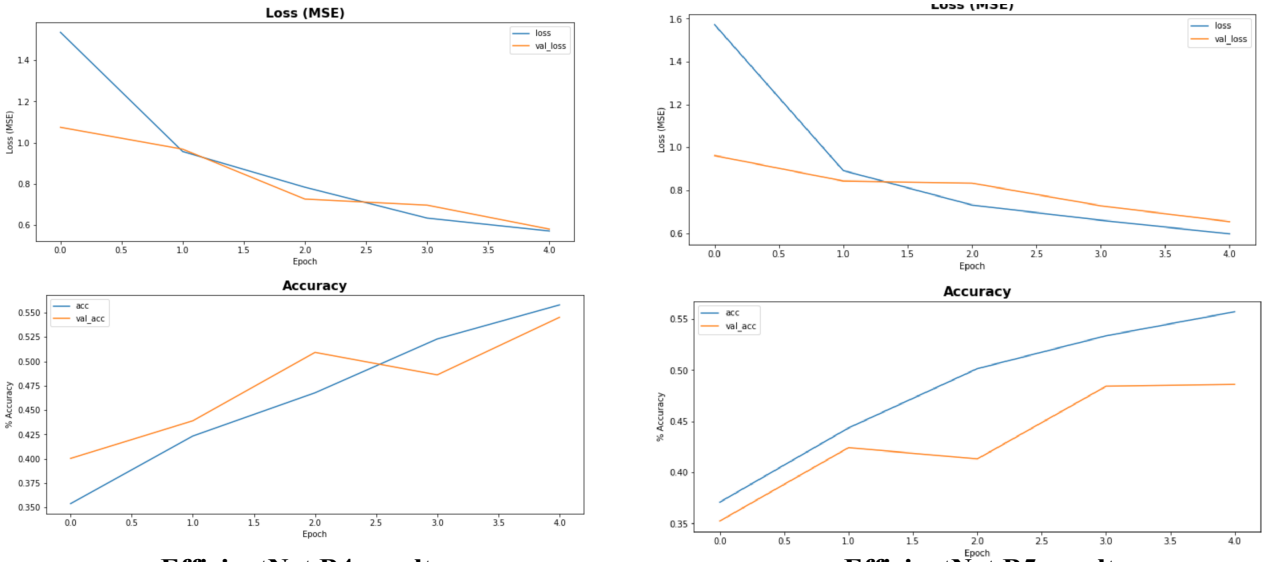
Figure 32. Custom dataset example images

7. Experimental Investigation

In this section, improvements in preprocessing methods, training, and prediction issues will be evaluated and explained. To execute experiments TensorFlow Framework is used and Cloud GPU is implemented provided by Kaggle Platform. Trainings are done with EfficientNet-B1, EfficientNet-B2, EfficientNet-B3, EfficientNet-B4, EfficientNet-B5 and ResNet50 architectures. Each experiment took approximately two hours to complete. The main focus of the experiments is to compare and combine the HSV, HSL, and L*a*b* color spacing methods with some other preprocessing methods. On the network part of the images, batch size was mostly equal to 4. For the normalization process, the group normalization method is applied, the Rectified Adam (RAdam) is a variant of the Adam stochastic optimizer that introduces a term to rectify the variance of the adaptive learning rate [34]. optimizer is used, and also to avoid overfitting as much as possible, an early stopping function is applied as well as updating the learning rate depending on validation loss. As mentioned in the problem definition part, the main focus of the experiments is to investigate the effects of color spaces on deep learning techniques in image classification problems. Results will be visualized and evaluated from different perspectives. Information and results of the experiments are explained in the following parts of the thesis.

7.1 Custom Dataset Experiments

7.1.1 Custom Dataset Experiment With EfficientNet



EfficientNet B4 results **EfficientNet B5 results**

Figure 33. Accuracy and Loss Results of Custom Dataset Experiment

Figure 33 represents experiment accuracy and loss results held with the plain custom dataset. In order to collect these results, experiments are done with 5 epochs and batch size 4 to check high-sized images easier and the mean square error loss function is used to calculate loss values.

Both results actually had optimal results and they are not overfitted and still have a tendency to increase their accuracies if they would have more epochs to train. Figure 34 shows Cohen kappa scores of the experiments and EfficientNet B4 is performed better for training and validation processes however it does not represent the correct predictions in Figure 35, B4 architecture was able to predict label 4 images in comparison to EfficientNet B5 architecture.

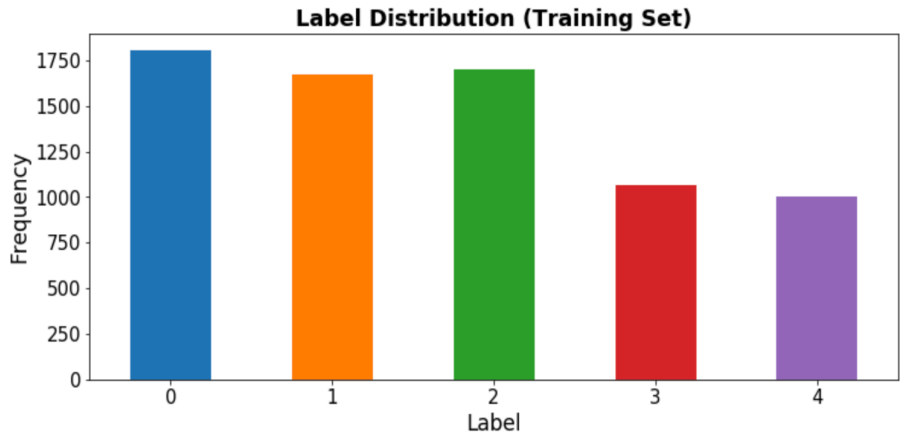
```
The Training Cohen Kappa Score is: 0.87795
The Validation Cohen Kappa Score is: 0.72498
```

EfficientNet B4 results

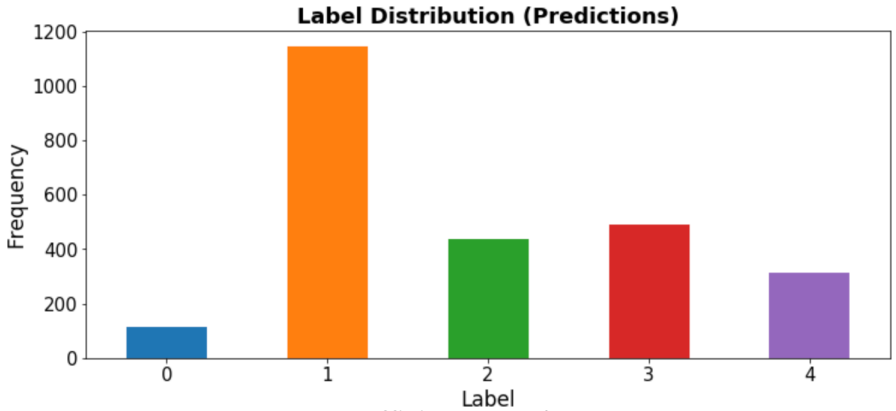
```
The Training Cohen Kappa Score is: 0.81879
The Validation Cohen Kappa Score is: 0.66711
```

EfficientNet B5 results

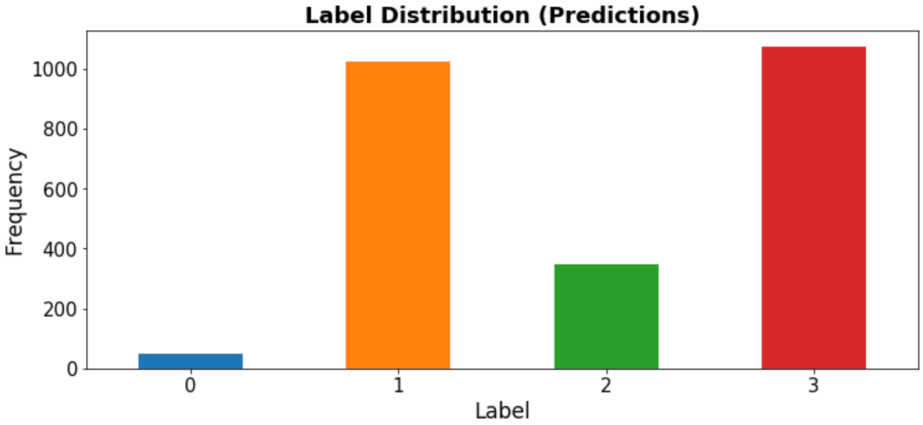
Figure 34. Kappa Score Results of Custom Dataset



Custom Dataset Training Images Distribution



EfficientNet B4 results



EfficientNet B5 results

Figure 35. Prediction Label Distribution of Custom Dataset Experiment

7.1.2 Custom Dataset Experiment With ResNet50

Figure 36. Custom dataset example train data images from every label

This experiment was held with 25 epochs and is not used any extra preprocessing methods than the mentioned ones in the dataset explanation section. Figure 36 shows training images from every label and Figure 37 presents the training report. Although the model has high F1, precision, and real values, the training accuracy is not good enough for the classification in comparison to EfficientNet experiments.

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.086804	0.382841	2.119095	1.027675	0.943727	0.977860	0.977860	0.977860	1.494519	0.351689	1.525976	0.825590	0.932857	0.847665	0.847665	0.847665	0.000050
1	0.963811	0.428835	1.975824	1.011091	0.979667	0.992782	0.992782	0.992782	0.985059	0.433447	1.578021	0.867707	0.954331	0.885820	0.885820	0.885820	0.000050
2	0.769392	0.423432	2.325319	1.003690	0.992620	0.997364	0.997364	0.997364	0.780177	0.494556	1.598816	0.893114	0.960561	0.906585	0.906585	0.906585	0.000050
3	0.984212	0.416821	1.846957	1.030191	0.913124	0.961623	0.961623	0.961623	0.683145	0.525110	1.606906	0.910179	0.963324	0.921099	0.921099	0.921099	0.000050
4	0.800263	0.471402	1.864326	1.005535	0.971402	0.985591	0.985591	0.985591	0.631966	0.548513	1.597206	0.921014	0.961861	0.928109	0.928109	0.928109	0.000050
5	0.854660	0.427911	1.921885	1.024646	0.925139	0.965285	0.965285	0.965285	0.611056	0.552576	1.596167	0.927136	0.966954	0.933942	0.933942	0.933942	0.000050
6	0.999098	0.429889	1.727091	1.020295	0.948339	0.977860	0.977860	0.977860	0.581451	0.563953	1.594158	0.931849	0.968362	0.938542	0.938542	0.938542	0.000050
7	0.641155	0.523105	2.054027	1.000000	0.992606	0.995775	0.995775	0.995775	0.496137	0.595482	1.599677	0.943496	0.976109	0.951030	0.951030	0.951030	0.000025
8	0.882263	0.479705	1.872124	1.012915	0.949262	0.975400	0.975400	0.975400	0.478754	0.610434	1.601752	0.951839	0.978547	0.957401	0.957401	0.957401	0.000025
9	0.750525	0.487061	2.227886	1.001232	0.978743	0.988029	0.988029	0.988029	0.456557	0.631237	1.591736	0.949943	0.975838	0.955006	0.955006	0.955006	0.000025
10	0.664521	0.513838	2.087594	1.009840	0.969557	0.986646	0.986646	0.986646	0.435528	0.632862	1.604045	0.960128	0.981635	0.964471	0.964471	0.964471	0.000025
11	0.816924	0.483364	1.958120	1.010474	0.952865	0.976763	0.976763	0.976763	0.437618	0.632374	1.593964	0.954927	0.972750	0.956356	0.956356	0.956356	0.000025
12	0.622756	0.523148	2.219029	1.002469	0.990741	0.995767	0.995767	0.995767	0.395894	0.654315	1.598292	0.966845	0.978655	0.967260	0.967260	0.967260	0.000012
13	0.703314	0.511091	2.157766	1.001232	0.980591	0.989173	0.989173	0.989173	0.379562	0.651227	1.597193	0.966629	0.979631	0.967773	0.967773	0.967773	0.000012
14	0.741625	0.525926	2.109137	1.006173	0.983333	0.993034	0.993034	0.993034	0.373722	0.668129	1.601814	0.968091	0.978764	0.968082	0.968082	0.968082	0.000012
15	0.774808	0.500000	2.034470	1.012939	0.961183	0.982572	0.982572	0.982572	0.361717	0.668779	1.592855	0.965220	0.973021	0.964148	0.964148	0.964148	0.000012
16	0.673579	0.540665	2.088738	1.008010	0.986137	0.995247	0.995247	0.995247	0.356511	0.674793	1.595680	0.975676	0.982881	0.974860	0.974860	0.974860	0.000012
17	0.723704	0.523105	2.146012	1.007394	0.987985	0.996303	0.996303	0.996303	0.350794	0.679018	1.596021	0.968850	0.977680	0.968437	0.968437	0.968437	0.000006
18	0.653686	0.529630	2.263081	1.003704	0.991667	0.996825	0.996825	0.996825	0.328803	0.697058	1.597804	0.968091	0.977626	0.968074	0.968074	0.968074	0.000006
19	0.735336	0.521257	2.090446	1.008010	0.976895	0.989790	0.989790	0.989790	0.326102	0.691695	1.600230	0.971234	0.976651	0.969198	0.969198	0.969198	0.000006
20	0.674696	0.515741	2.203600	1.009259	0.975926	0.989153	0.989153	0.989153	0.312075	0.703722	1.593619	0.975080	0.978980	0.973269	0.973269	0.973269	0.000006
21	0.673877	0.541590	2.177774	1.002465	0.983364	0.991550	0.991550	0.991550	0.310454	0.699496	1.602618	0.970096	0.977626	0.969537	0.969537	0.969537	0.000003
22	0.709299	0.512015	2.218692	1.005545	0.977819	0.989262	0.989262	0.989262	0.299097	0.708597	1.595607	0.977193	0.978710	0.974459	0.974459	0.974459	0.000003
23	0.644574	0.560998	2.142076	1.003697	0.989834	0.995775	0.995775	0.995775	0.302210	0.711848	1.599428	0.974972	0.976922	0.972099	0.972099	0.972099	0.000003
24	0.718063	0.516636	2.130886	1.004929	0.976895	0.988909	0.988909	0.988909	0.300783	0.709410	1.598923	0.978547	0.983964	0.977631	0.977631	0.977631	0.000003

Figure 37. Experiment summary report

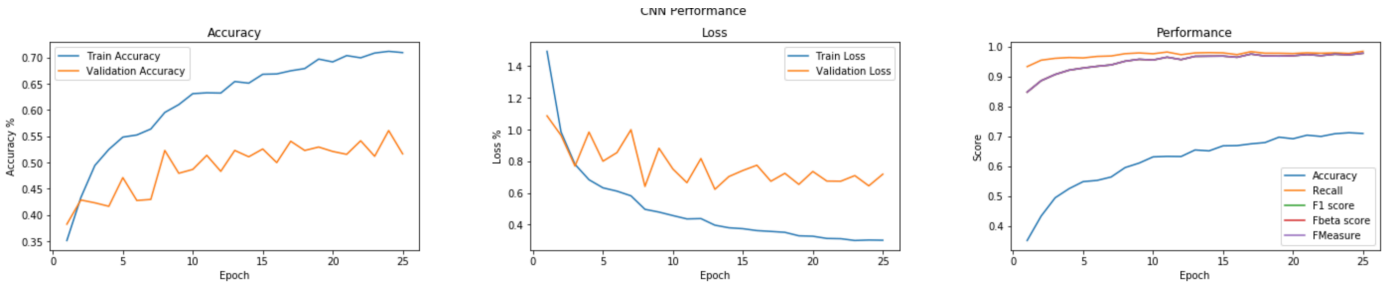


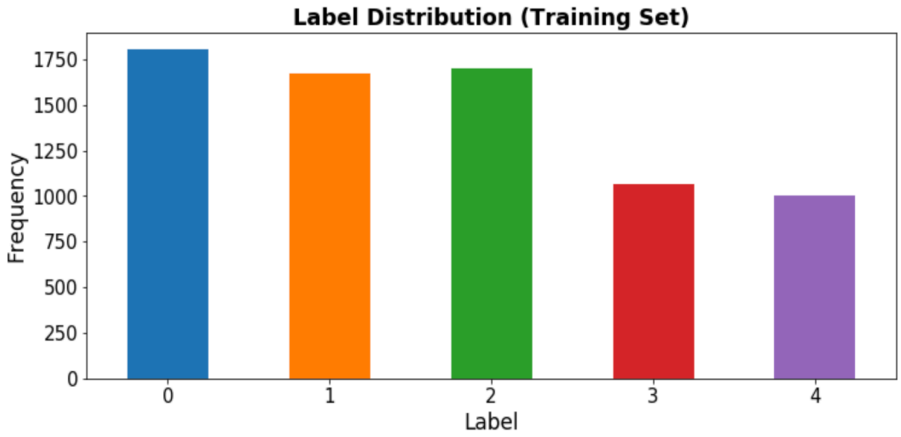
Figure 38. Experiment plots

Another proof to perform poor training can be seen in Figure 38. According to the validation and training loss plot, the model is overfitted obviously. The biggest difference between the EfficientNet model and the ResNet model is the image sizes. For this architecture, a lower input size might be led to the overfitting issue.

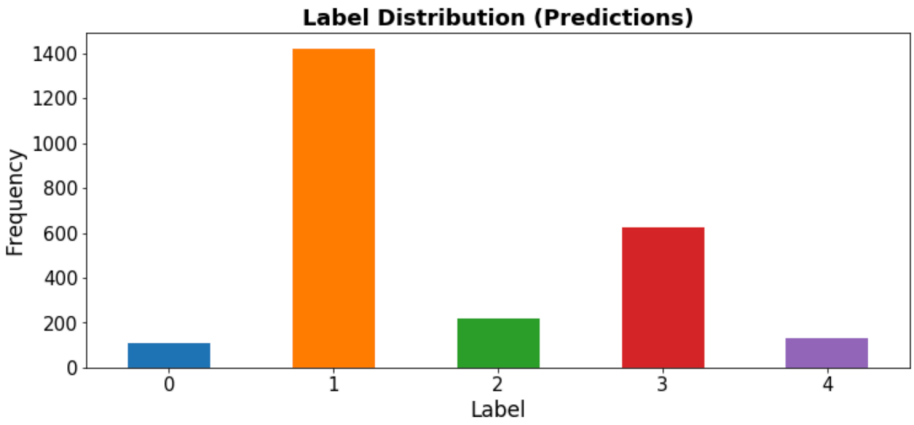
```
                                                Optimized Thresholds:
                                                [0.49959717 1.67501221 2.11083984 3.5494751 ]

The Training Cohen Kappa Score is: 0.90623    The Validation Quadratic Weighted Kappa (QWK)
The Validation Cohen Kappa Score is: 0.71967 with optimized rounding thresholds is: 0.74037

                                                This is an improvement of 0.02069
                                                over the unoptimized rounding
```

Figure 39. Kappa score summary reports

Figure 39 shows the kappa agreement results, in comparison to loss values this metric shows some positive side of the experiment but it is still possible to have a good kappa score for an overfitted model as can be seen in this experiment. Figure 40 shows how training set label distribution and predicted labels are evaluated with the test set of the dataset. The performance of the model for labels 1 and 3 look meaningful but it is hard to say it for label 0 and 2. But still, in comparison to prediction results obtained by EfficientNet models, the ResNet experiment has a more reasonable prediction session.



Custom Dataset Training Images Distribution



Predicted Label Distribution

Figure 40. Custom Dataset Training Images Distribution

7.2 APTOS 2019 Dataset Experiment

7.2.1 ResNet50 Experiment

Figure 41. Example training images from every sample from the dataset

The training images used for this experiment can be seen in Figure 41. The only pre-processing methods used for this experiment are resizing and cropping from grey. The main reason to have this test is to have an initial idea and make a comparison with color space applied experiments. The experiment is planned to execute with 25 epochs but because of the validation loss values, it stopped training at the 22nd epoch, details of the experiment are shown in Figure 42 below,

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	0.650543	0.478102	1.106598	0.900243	0.913625	0.902051	0.902051	0.902051	1.397047	0.351221	1.100459	12854.166881	0.881534	0.738622	0.738622	0.738622	0.000050
1	0.529064	0.532110	1.350052	0.879511	0.959633	0.905950	0.905950	0.905950	0.856967	0.439691	1.164179	12866.630213	0.919803	0.803596	0.803596	0.803596	0.000050
2	0.463163	0.687956	1.179016	0.939781	0.964720	0.945429	0.945429	0.945429	0.652408	0.542940	1.135316	0.806905	0.917122	0.839415	0.839415	0.839415	0.000050
3	0.451457	0.680734	1.120119	0.973700	0.948012	0.953622	0.953622	0.953622	0.507630	0.615632	1.117130	12866.697652	0.916908	0.853951	0.853951	0.853951	0.000050
4	0.500060	0.728102	0.993080	72993.626521	0.908759	0.916997	0.916997	0.916997	0.444015	0.658411	1.117160	0.864908	0.919588	0.881055	0.881055	0.881055	0.000050
5	0.493544	0.644161	1.007630	0.918491	0.874088	0.887800	0.887800	0.887800	0.393626	0.687681	1.118461	12866.751903	0.921625	0.890551	0.890551	0.890551	0.000050
6	0.409335	0.748175	1.162557	0.925182	0.935523	0.927424	0.927424	0.927424	0.364466	0.702798	1.122096	0.879811	0.919374	0.890346	0.890346	0.890346	0.000050
7	0.584003	0.700917	1.352339	0.933945	0.941896	0.932896	0.932896	0.932896	0.359004	0.722740	1.120634	0.880455	0.925807	0.892741	0.892741	0.892741	0.000050
8	0.762562	0.693578	1.423193	73395.432416	0.953517	0.952468	0.952468	0.952468	0.334255	0.737215	1.125424	0.900504	0.925592	0.906940	0.906940	0.906940	0.000050
9	0.317852	0.759124	1.256045	0.933090	0.956204	0.941223	0.941223	0.941223	0.319692	0.755548	1.110305	0.908867	0.925163	0.910653	0.910653	0.910653	0.000050
10	0.409661	0.755963	1.134906	0.962691	0.965138	0.961206	0.961206	0.961206	0.302993	0.749115	1.120892	12866.766806	0.910797	0.897560	0.897560	0.897560	0.000050
11	0.404327	0.728440	1.147889	0.942508	0.955963	0.946038	0.946038	0.946038	0.293305	0.770666	1.112284	0.893964	0.912405	0.896773	0.896773	0.896773	0.000050
12	0.399389	0.765683	1.144417	0.939114	0.938499	0.935108	0.935108	0.935108	0.276053	0.765519	1.113684	0.914120	0.925700	0.914463	0.914463	0.914463	0.000050
13	0.360157	0.747232	1.283007	73801.666052	0.924354	0.932666	0.932666	0.932666	0.291566	0.765841	1.125787	0.907473	0.910797	0.903295	0.903295	0.903295	0.000050
14	0.386616	0.771218	1.133147	0.943419	0.938499	0.936531	0.936531	0.936531	0.254252	0.799936	1.121182	0.923019	0.937279	0.925078	0.925078	0.925078	0.000025
15	0.406585	0.785978	1.289907	0.944649	0.952030	0.947338	0.947338	0.947338	0.237465	0.799614	1.109487	0.911869	0.916908	0.910392	0.910392	0.910392	0.000025
16	0.406284	0.758303	1.226656	0.937884	0.950184	0.940819	0.940819	0.940819	0.219968	0.813445	1.115594	12866.796290	0.924520	0.922106	0.922106	0.922106	0.000025
17	0.825524	0.744954	1.398186	0.952905	0.959021	0.952398	0.952398	0.952398	0.225251	0.803474	1.109955	0.908438	0.918087	0.909311	0.909311	0.909311	0.000025
18	0.476521	0.724771	1.410752	0.954740	0.965138	0.957833	0.957833	0.957833	0.203886	0.826954	1.120905	0.934598	0.943605	0.935683	0.935683	0.935683	0.000012
19	0.445827	0.777982	1.305351	0.963303	0.962079	0.961346	0.961346	0.961346	0.187964	0.823416	1.109948	0.905757	0.909724	0.904921	0.904921	0.904921	0.000012
20	0.436254	0.777982	1.237597	73395.414067	0.929664	0.930642	0.930642	0.930642	0.184413	0.832744	1.111847	0.932347	0.934598	0.930034	0.930034	0.930034	0.000012
21	0.505773	0.743542	1.312510	0.934194	0.936654	0.932578	0.932578	0.932578	0.184013	0.830814	1.117741	0.920768	0.921196	0.917729	0.917729	0.917729	0.000012

Figure 42. Experiment summary report

According to Figure 42, it is possible to observe that some good results are obtained with this experiment. At the end of the model training, reasonable training and validation accuracies can be seen in the report. Additionally, F1, precision, and recall values give a more detailed idea about the performance of the classifier and the results are good enough to classify diabetic retinopathy. Experiment plots are presented in Figure 43.

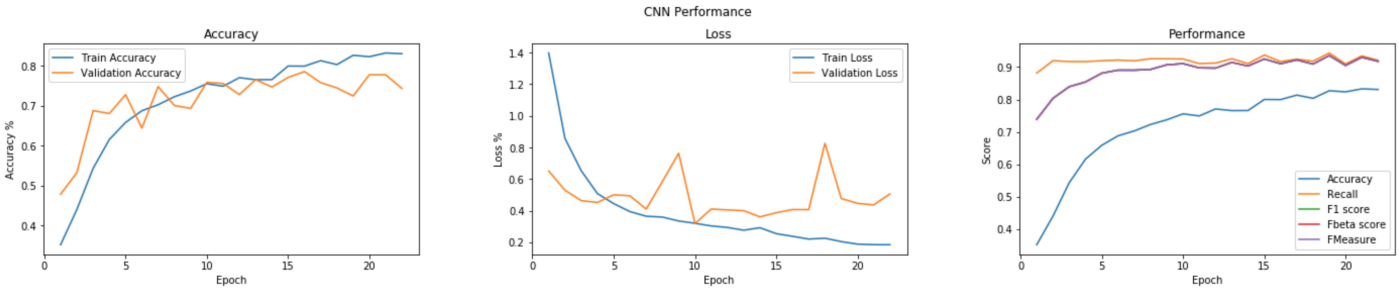


Figure 43. Experiment plots

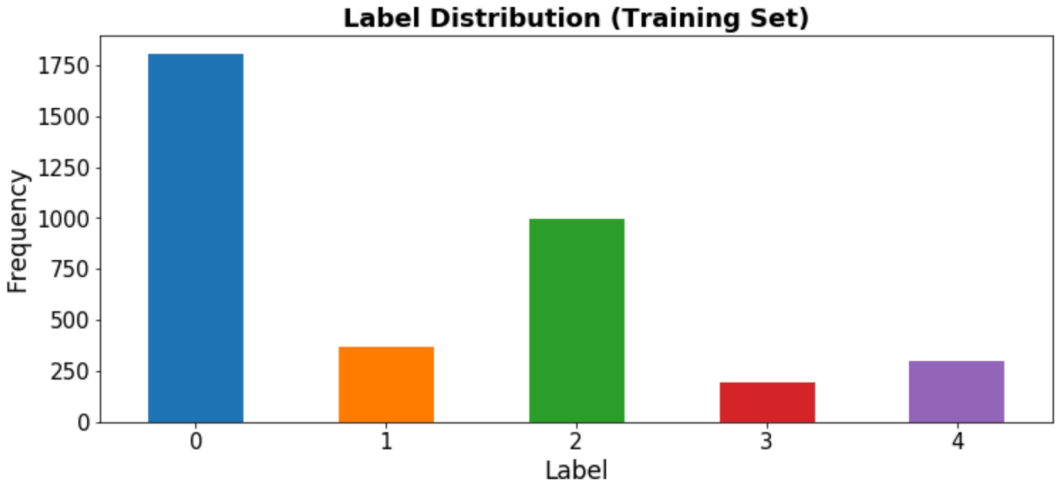
Despite having reasonable results in Figure 37, things change when the results are visualized. One of the most important plots to evaluate here is the loss graph of the training, validation loss has a slightly higher value than training loss and also has an unstable line, which means the model tends to overfit. Another metric to evaluate is the kappa score, Figure 44 shows the kappa performance of the training.

```
The Training Cohen Kappa Score is: 0.94627
The Validation Cohen Kappa Score is: 0.90952
```

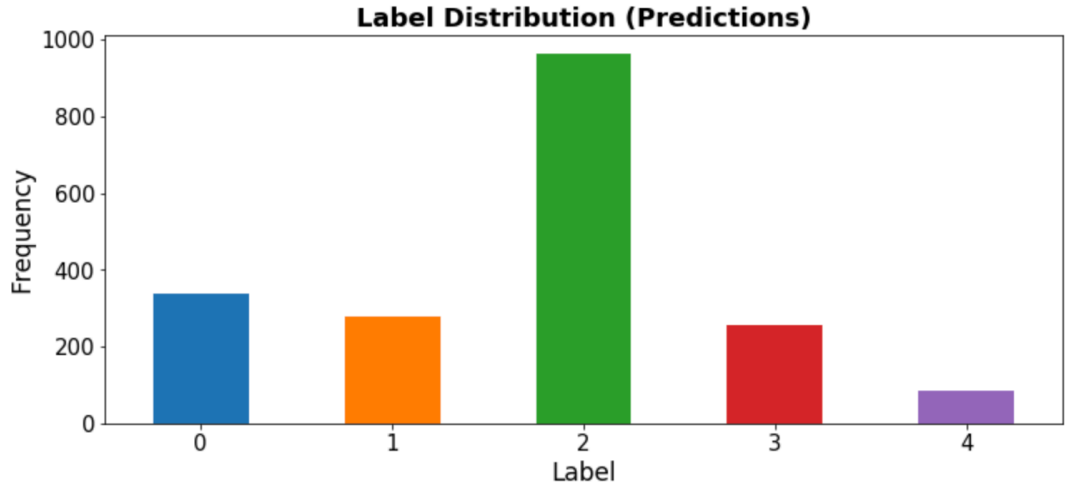
```
Optimized Thresholds:
[0.46418259 1.38766788 2.66730066 3.73869309]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.91429

This is an improvement of 0.00477
over the unoptimized rounding
```

Figure 44. Kappa score summary reports

Except for the overfitting issue, the kappa score also tells that model has a good classification perforce since it has over 0.90 kappa value. Another method to increase the validation kappa score is optimizing the validation data and making an evaluation again. On the right side of Figure 39 optimized kappa score and an increase in the validation kappa can be seen.



APTOS 2019 Dataset Training Images Distribution



Predicted Label Distribution

Figure 45. APTOS 2019 Dataset Training Images Distribution

Figure 45 represents the difference between the predicted label distribution obtained with test data and training data labels. Except for label 0 images prediction process looks reasonable in comparison to the training set. It is possible to say that the model has poor prediction performance for the label 0 and label 2 images but overall it is okay to classify the retinopathy.

7.2.2 EfficientNet B1 Experiment

Figure 46. Example training images from every sample from the dataset

In this experiment performance of the EfficientNet B1 model is tested with APTOS 2019 imbalanced dataset. After having slightly different results on the custom dataset experiments, it is planned to test out closer architectures by input sizes chosen and the required input size for EfficientNet B1 and ResNet50 architectures are the closest ones. Figure 46 shows training images of the dataset and experiment details with various metrics are represented in Figure 47. As well as the ResNet experiment this test section also has proper F1 score and accuracy results for both validation and training images. However, the investigation of loss values represented in Figure 48 tells some other stories as we met in previous experiments.

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	0.938398	0.616788	0.701931	218978.979319	0.888078	0.908133	0.908133	0.908133	2.042044	0.458548	0.344836	1.362468e+06	0.624893	0.677690	0.677690	0.677690	0.000050
1	0.569801	0.708257	1.031356	0.924771	0.933945	0.922202	0.922202	0.922202	0.845049	0.519460	1.043239	7.793503e-01	0.914120	0.811086	0.811086	0.811086	0.000050
2	0.572615	0.717153	0.866744	72993.675182	0.961071	0.973305	0.973305	0.973305	0.738700	0.571888	1.086609	8.132304e-01	0.921089	0.837620	0.837620	0.837620	0.000050
3	0.505664	0.710092	1.040598	0.939450	0.931498	0.930520	0.930520	0.930520	0.604272	0.601480	1.087033	8.366034e-01	0.920446	0.855829	0.855829	0.855829	0.000050
4	0.420682	0.732110	1.069033	73395.420795	0.916820	0.927829	0.927829	0.927829	0.502191	0.645224	1.089652	8.601908e-01	0.920660	0.871592	0.871592	0.871592	0.000050
5	0.431864	0.739051	1.003042	0.919708	0.911800	0.909663	0.909663	0.909663	0.438359	0.669347	1.106792	8.696258e-01	0.922054	0.881125	0.881125	0.881125	0.000050
6	0.468032	0.750459	0.958165	0.924159	0.929052	0.922901	0.922901	0.922901	0.404790	0.704728	1.097604	8.907473e-01	0.936850	0.901723	0.901723	0.901723	0.000050
7	0.384635	0.755474	1.093815	0.953771	0.950730	0.945499	0.945499	0.945499	0.379087	0.728852	1.097558	8.739144e-01	0.917122	0.885301	0.885301	0.885301	0.000050
8	0.404035	0.755963	1.023417	0.956575	0.926605	0.932215	0.932215	0.932215	0.365222	0.713091	1.098596	8.933204e-01	0.925807	0.899447	0.899447	0.899447	0.000050
9	0.365305	0.782847	1.147897	0.964720	0.945864	0.951130	0.951130	0.951130	0.338483	0.731425	1.106195	8.993245e-01	0.926128	0.904936	0.904936	0.904936	0.000050
10	0.352089	0.788991	1.059072	73395.418960	0.933945	0.934836	0.934836	0.934836	0.332762	0.742361	1.103729	9.057575e-01	0.920553	0.906719	0.906719	0.906719	0.000050
11	0.366298	0.768807	1.072428	0.931498	0.905199	0.910633	0.910633	0.910633	0.320383	0.752975	1.108553	8.987884e-01	0.921625	0.903071	0.903071	0.903071	0.000050
12	0.335313	0.772477	1.116542	0.899694	0.905199	0.898628	0.898628	0.898628	0.306753	0.754583	1.098021	8.958936e-01	0.915085	0.899459	0.899459	0.899459	0.000050
13	0.345473	0.761468	1.135323	0.945566	0.930887	0.931062	0.931062	0.931062	0.293692	0.769058	1.107073	9.029698e-01	0.917873	0.904330	0.904330	0.904330	0.000050
14	0.341295	0.777982	1.229521	0.934557	0.944342	0.935046	0.935046	0.935046	0.271067	0.778385	1.100079	9.119760e-01	0.920982	0.910095	0.910095	0.910095	0.000050
15	0.337367	0.782288	1.098995	0.956950	0.950799	0.948093	0.948093	0.948093	0.267297	0.773239	1.107614	9.089739e-01	0.917337	0.907997	0.907997	0.907997	0.000050
16	0.353358	0.757798	1.075323	0.973089	0.932110	0.943871	0.943871	0.943871	0.262812	0.782245	1.100321	1.286678e+04	0.911225	0.903641	0.903641	0.903641	0.000050
17	0.316278	0.795203	1.101011	0.955720	0.938499	0.938921	0.938921	0.938921	0.239807	0.777742	1.116789	9.272006e-01	0.927093	0.922455	0.922455	0.922455	0.000025
18	0.302075	0.805505	1.153086	0.957798	0.954740	0.950633	0.950633	0.950633	0.227439	0.799614	1.111915	9.131553e-01	0.919159	0.912267	0.912267	0.912267	0.000025
19	0.258004	0.812844	1.110960	0.949847	0.932722	0.934294	0.934294	0.934294	0.237245	0.796398	1.100839	1.286679e+04	0.919052	0.916574	0.916574	0.916574	0.000025
20	0.298500	0.797048	1.096073	0.916359	0.926199	0.917203	0.917203	0.917203	0.228357	0.795433	1.105819	9.236625e-01	0.922912	0.919408	0.919408	0.919408	0.000025
21	0.316363	0.793358	1.118377	0.934194	0.915744	0.921156	0.921156	0.921156	0.214542	0.805404	1.104733	9.159429e-01	0.919052	0.914074	0.914074	0.914074	0.000025
22	0.334913	0.795203	1.101365	0.912669	0.900984	0.901511	0.901511	0.901511	0.202676	0.811193	1.117598	9.211965e-01	0.921840	0.917046	0.917046	0.917046	0.000025
23	0.349937	0.759633	0.978411	0.949235	0.919878	0.927619	0.927619	0.927619	0.209515	0.811515	1.112766	9.222687e-01	0.917122	0.916142	0.916142	0.916142	0.000025
24	0.297825	0.785321	1.137689	73395.441590	0.938838	0.944325	0.944325	0.944325	0.208376	0.816018	1.108555	1.286680e+04	0.924306	0.922379	0.922379	0.922379	0.000012

Figure 47. Experiment summary report

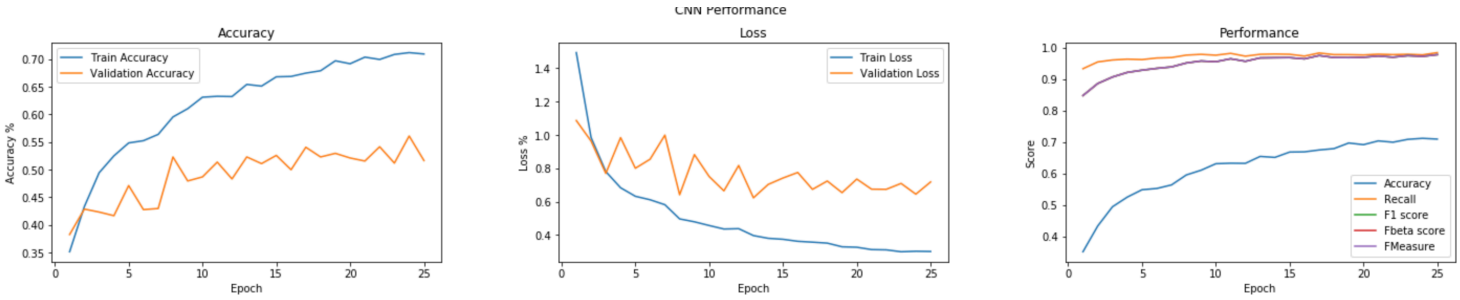
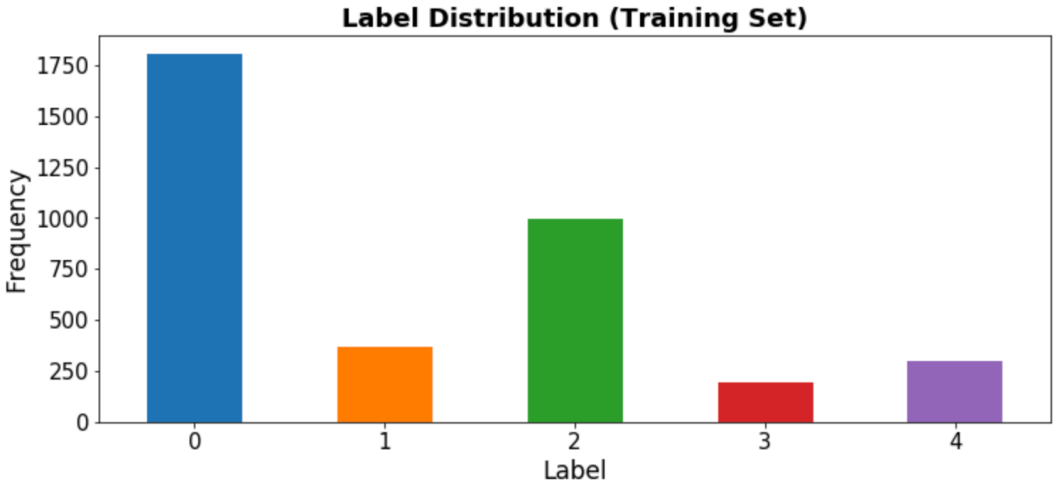


Figure 48. Experiment plots of B1 architecture

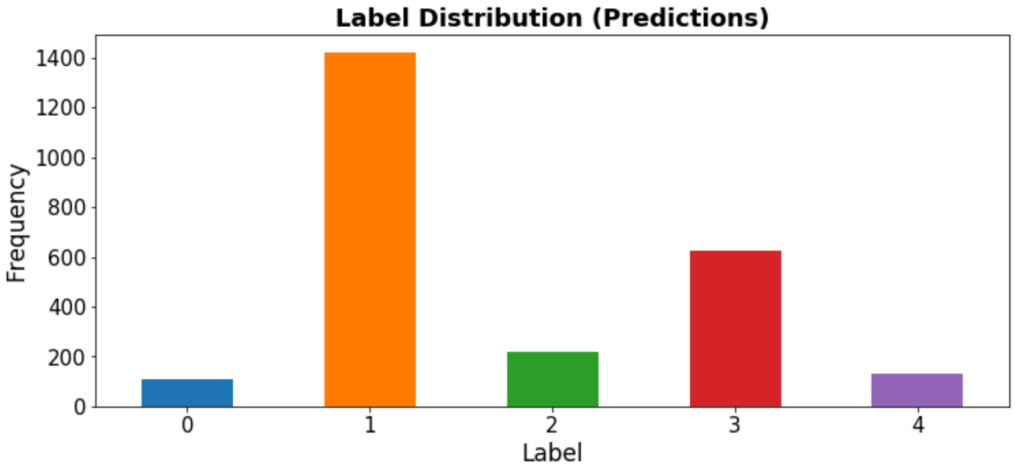
Experiment plots shown in Figure 48 give some hints that the model is also overfitted for that model too in the first plot for the accuracies it is possible to observe that the model reached nearly maximum accuracy levels for both training and validation sets. The second plot shows us how the model is actually poorly trained and overfitted. If the validation loss would not be that much unstable and would be closer to the training loss value a little more, it would be possible to say the experiment is handled quite well with the values of accuracy, F1, and the Kappa score results explained in Figure 49 but this test had similar results with ResNet50 experiment.

```
                                        Optimized Thresholds:
                                        [0.49959717 1.67501221 2.11083984 3.5494751 ]

The Training Cohen Kappa Score is: 0.90623  The Validation Quadratic Weighted Kappa (QWK)
The Validation Cohen Kappa Score is: 0.7196 with optimized rounding thresholds is: 0.74037

                                        This is an improvement of 0.02069
                                        over the unoptimized rounding
```

Figure 49. Kappa score summary reports of B1 architecture



APTOS 2019 Dataset Training Images Distribution



Predicted Label Distribution

Figure 50. APTOS 2019 Dataset Training Images Distribution for B1 Architecture

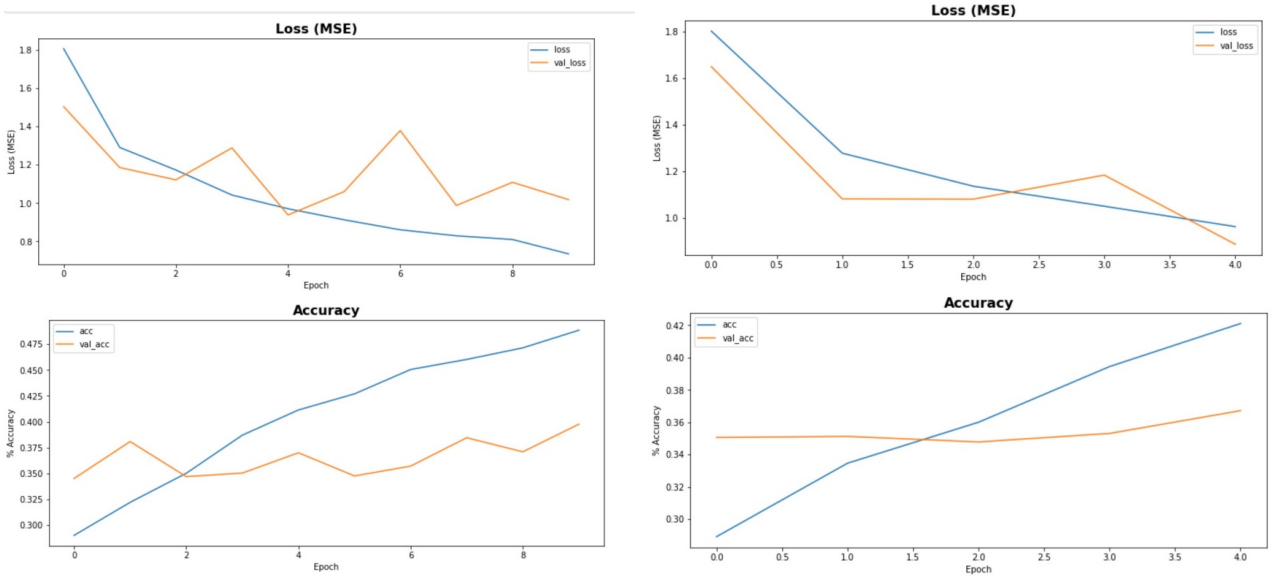
Figure 50 shows the predicted image labels and comparison to the previous experiment held with ResNet architecture, this model has much more predicted results for label 1. Again for this experiment prediction of label 0 images is hard for the models so far for both datasets have been used.

7.3 HSV Color Spacing Experiment

7.3.1 EfficientNet-B5 and EfficientNet-B4 Experiments

Figure 51. Thresholded Fundus Images B5-B4 Experiments

In this experiment effects of HSV color spacing on accuracy are examined and images in Figure 51 are used and examples from every label in the dataset can be seen above. Except for a number of epochs and image sizes, all other parameters were the same as in the previous experiment. Here in Figure 52. we can see that the 10 epoch experiment with EfficientNet B4 looks overfitted according to the loss plot of that experiment and the B5 model had more stable training in comparison to B4.



EfficientNet B4 results **EfficientNet B5 results**

Figure 52. Experiment Plots of B4-B5 Architectures

Kappa scores are shown in Figure 53 and despite having more training time EfficientNet B4 algorithm has very close results in comparison to the B5 model.

```
The Training Cohen Kappa Score is: 0.8076
The Validation Cohen Kappa Score is: 0.53548
```

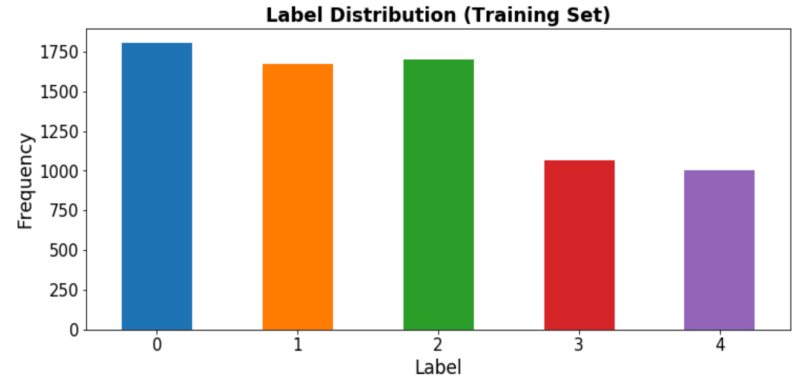
EfficientNet B4 results

```
The Training Cohen Kappa Score is: 0.75425
The Validation Cohen Kappa Score is: 0.46227
```

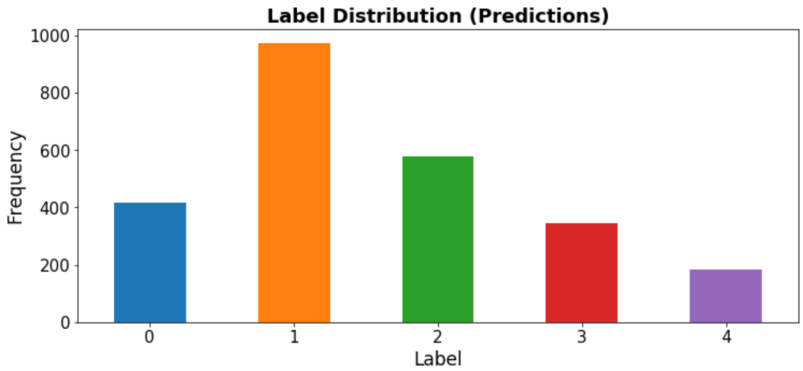
EfficientNet B5 results

Figure 53. Kappa Score Results of Custom Dataset B4-B5 Architectures

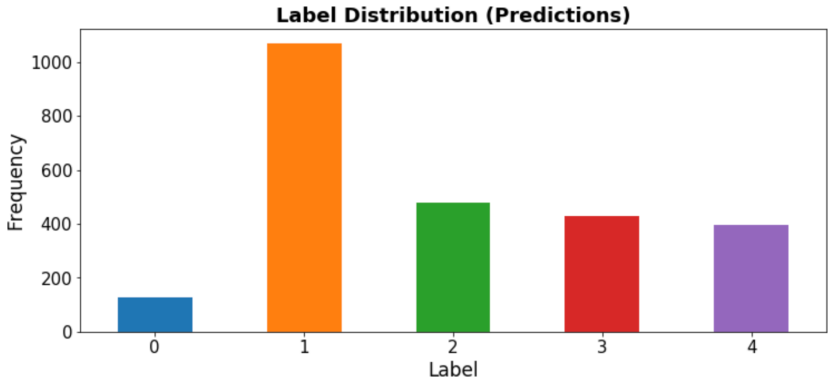
Another improvement in comparison to the previous custom dataset experiment can be seen in Figure 54, this experiment had more meaningful label distribution plots than the previous experiment.



Custom Dataset Training Images Distribution



EfficientNet B4 results



EfficientNet B5 results

Figure 54. Custom Dataset Training Images Distribution B4-B5 Architectures

7.3.2 EfficientNet-B5 Experiments with Different Thresholds

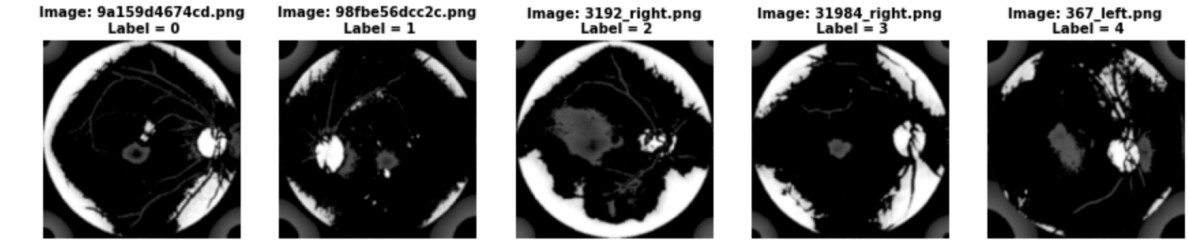


Figure 55. Training images after preprocess

This experiment was held with EfficientNet B5 architecture and different threshold values than the previous experiment. It is aimed to observe how different threshold values affect the results in training. This experiment has done with 10 epochs and the previous experiment was done with 5 epochs. It is possible to say that the results until the fifth epoch look similar and can be seen in Figures 56 and 57 below.

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score
0	1.439276	0.354244	1.604607	1.008610	0.977860	0.991038	0.991038	0.991038	1.786946	0.297596	1.385817	207927.992311	0.930908	0.816851	0.816851
1	1.167154	0.395564	1.760431	1.008626	0.982440	0.993486	0.993486	0.993486	1.254118	0.335771	1.578922	0.785741	0.974701	0.848090	0.848090
2	1.120463	0.377306	1.813791	1.011685	0.980627	0.993674	0.993674	0.993674	1.137451	0.359662	1.587681	0.805352	0.974917	0.859321	0.859321
3	1.137368	0.373383	1.803707	1.025878	0.951941	0.982748	0.982748	0.982748	1.040368	0.385015	1.590574	0.822201	0.976651	0.871434	0.871434
4	1.056205	0.420664	2.011356	1.019680	0.963100	0.986997	0.986997	0.986997	0.982690	0.398017	1.590488	0.825397	0.973184	0.872335	0.872335
5	0.918522	0.382625	2.099625	1.007394	0.981516	0.992606	0.992606	0.992606	0.916370	0.434422	1.592278	0.847771	0.976705	0.889726	0.889726
6	0.984707	0.402957	2.048098	1.008010	0.974122	0.988205	0.988205	0.988205	0.871956	0.445311	1.592597	0.856926	0.974972	0.894526	0.894526
7	0.823656	0.404059	2.079324	1.000000	0.996310	0.997891	0.997891	0.997891	0.823983	0.468714	1.596458	0.870090	0.975459	0.903480	0.903480
8	1.135950	0.349353	1.822122	1.017868	0.937153	0.970073	0.970073	0.970073	0.803076	0.466927	1.596259	0.872691	0.976380	0.905424	0.905424
9	1.055812	0.381919	2.016661	1.012915	0.955720	0.978914	0.978914	0.978914	0.770514	0.485129	1.597169	0.881630	0.972805	0.910560	0.910560

Figure 56. Experiment Summary Report of Updated B5 Experiment

Despite having a high F1 score, accuracy levels for both training and validation images are low. Figure 57 gives an idea of what would happen if more epochs were applied to the test. While training accuracy has a tendency to increase, validation accuracy has unstable and low values duration of the experiment. However, it is possible to comment positively by looking training and validation loss graph, the model is trained well enough and did not overfit but is close to overfitting.

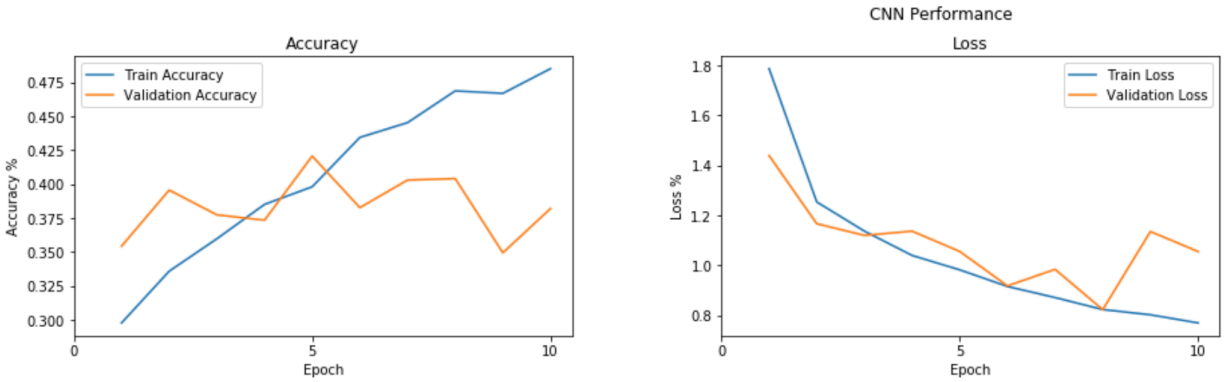


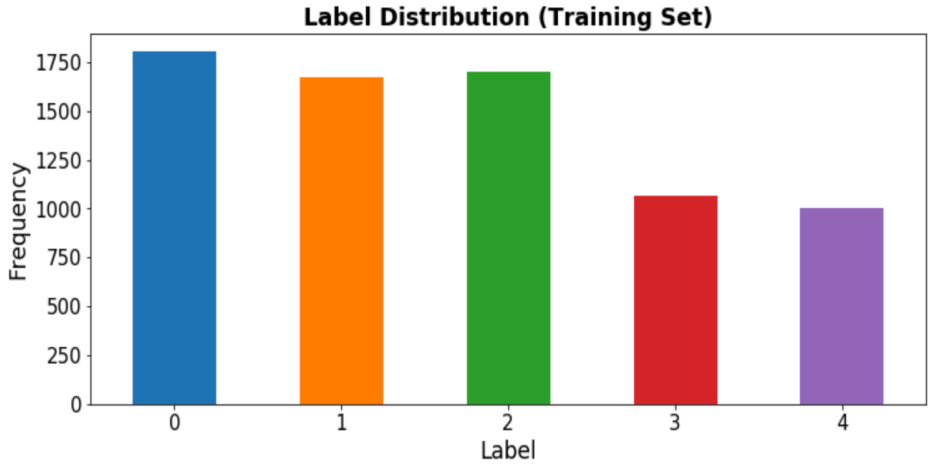
Figure 57. Experiment Plots of Updated B5 Experiment

Another metric to investigate is the kappa score presented in Figure 58,

```
The Training Cohen Kappa Score is: 0.72019
The Validation Cohen Kappa Score is: 0.51164
```

Figure 58. Experiment Kappa Results of Updated B5 Experiment

While having a higher kappa score on the training set, the validation kappa score has lower values parallel to accuracy results so it shows that the dataset has difficulty having an agreement on validation images. Figure 59 shows the comparison of predicted results distribution and training set labels. Despite having a more proper predicted label distribution compare to previous experiments, this model does not give so much trust in the purpose of image classification.



Custom Dataset Training Images Distribution

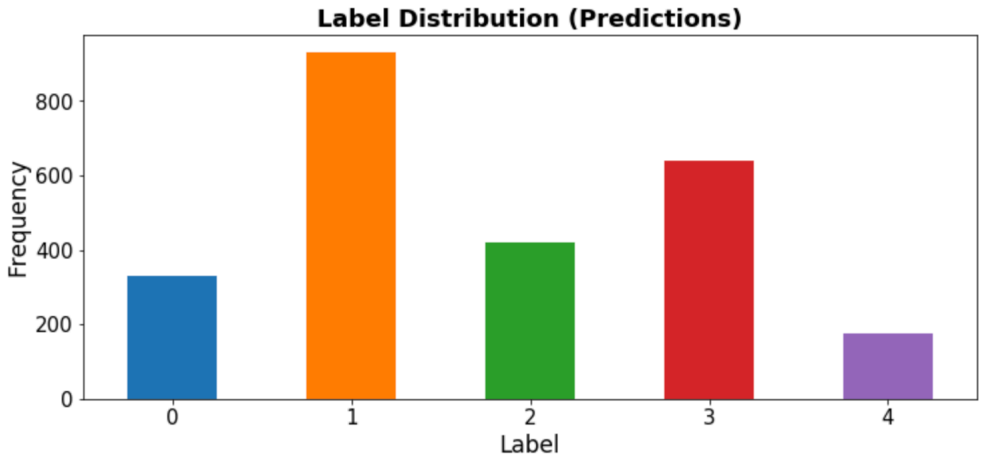


Figure 59. Experiment prediction label distribution of updated B5 experiment

7.3.3 EfficientNet-B3 Experiments

Figure 60. Training images after preprocess for B3 architecture

This experiment is planned to be done 25 in epochs but it has stopped at the 24th epoch and it took two hours to complete. Figure 60 shows the preprocessed training images by HSV color space. Figure 60 demonstrates the experiment result by consisting of metrics like F1, precision, and recall values. According to the results obtained, F1 values look proper enough to evaluate as well performed while validation accuracy stays stable at low levels as in the previous thresholded experiments. Figure 61 shows the performance of the experiment visually.

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.512115	0.333026	1.561068	1.029520	0.924354	0.966790	0.966790	0.966790	1.843168	0.295484	1.341487	279402.963396	0.896361	0.798072	0.798072	0.798072	0.000050
1	1.275491	0.341035	1.798534	1.017868	0.947320	0.976146	0.976146	0.976146	1.274118	0.347960	1.573055	0.793163	0.971125	0.848788	0.848788	0.848788	0.000050
2	1.144505	0.384686	1.869569	1.017220	0.952952	0.979265	0.979265	0.979265	1.154482	0.358687	1.582833	0.801994	0.968471	0.853519	0.853519	0.853519	0.000050
3	1.169009	0.387246	1.777677	1.011707	0.966728	0.985565	0.985565	0.985565	1.060783	0.389078	1.591539	0.821659	0.971234	0.866487	0.866487	0.866487	0.000050
4	1.198451	0.372694	1.751771	1.028290	0.919742	0.963451	0.963451	0.963451	0.965724	0.413457	1.588657	0.838561	0.973346	0.879831	0.879831	0.879831	0.000050
5	1.058315	0.386322	1.928685	1.014787	0.966728	0.986445	0.986445	0.986445	0.899015	0.442548	1.594995	0.854001	0.974376	0.890918	0.890918	0.890918	0.000050
6	1.105030	0.374539	1.839191	1.019065	0.964022	0.986470	0.986470	0.986470	0.843064	0.458801	1.592375	0.865594	0.972480	0.898352	0.898352	0.898352	0.000050
7	1.076254	0.392791	2.031651	1.024030	0.943623	0.976322	0.976322	0.976322	0.787494	0.485454	1.593374	0.878325	0.971559	0.908155	0.908155	0.908155	0.000050
8	1.074050	0.377306	1.875937	1.016605	0.964945	0.986646	0.986646	0.986646	0.748851	0.489680	1.598227	0.883201	0.970746	0.909870	0.909870	0.909870	0.000050
9	0.978090	0.403882	2.000874	1.025878	0.956562	0.983980	0.983980	0.983980	0.739495	0.496993	1.596018	0.883526	0.974917	0.912013	0.912013	0.912013	0.000050
10	0.948131	0.404806	2.191900	1.008626	0.982440	0.993663	0.993663	0.993663	0.730169	0.501219	1.594400	0.887697	0.972480	0.914766	0.914766	0.914766	0.000050
11	0.951182	0.398524	2.199206	1.012915	0.980627	0.994026	0.994026	0.994026	0.681237	0.521209	1.595747	0.896473	0.973130	0.920531	0.920531	0.920531	0.000050
12	0.967259	0.378704	2.243740	1.008025	0.978704	0.991005	0.991005	0.991005	0.676482	0.515521	1.592578	0.898315	0.964787	0.917891	0.917891	0.917891	0.000050
13	1.028181	0.413124	2.095362	1.026494	0.975046	0.996655	0.996655	0.996655	0.644301	0.541687	1.594866	0.911209	0.971396	0.927674	0.927674	0.927674	0.000050
14	1.091685	0.409259	2.161740	1.028395	0.941667	0.977249	0.977249	0.977249	0.630309	0.541037	1.593893	0.912671	0.972317	0.929910	0.929910	0.929910	0.000050
15	0.986435	0.431608	2.105755	1.020333	0.970425	0.991374	0.991374	0.991374	0.587953	0.557939	1.594854	0.915380	0.970583	0.931580	0.931580	0.931580	0.000025
16	0.928370	0.429630	2.109824	1.008642	0.982407	0.993474	0.993474	0.993474	0.554371	0.567040	1.599147	0.932391	0.970258	0.941489	0.941489	0.941489	0.000025
17	0.953766	0.419444	2.159386	1.018519	0.982407	0.996614	0.996614	0.996614	0.559614	0.572079	1.593059	6501.825397	0.974592	0.943982	0.943982	0.943982	0.000025
18	1.033059	0.402957	1.975113	1.028343	0.958410	0.987677	0.987677	0.987677	0.546625	0.571591	1.602430	0.928869	0.973996	0.941823	0.941823	0.941823	0.000025
19	0.945685	0.431481	2.096965	1.014198	0.977778	0.992416	0.992416	0.992416	0.539205	0.579717	1.596198	0.929194	0.974972	0.941232	0.941232	0.941232	0.000025
20	1.014391	0.420518	2.040483	1.014171	0.974122	0.991022	0.991022	0.991022	0.524603	0.590281	1.590646	0.936562	0.978655	0.947706	0.947706	0.947706	0.000025
21	0.982269	0.406654	2.106836	1.019100	0.977819	0.995247	0.995247	0.995247	0.498240	0.599545	1.598744	0.937104	0.976922	0.947785	0.947785	0.947785	0.000012
22	0.913566	0.450092	2.169752	1.022797	0.984288	0.998451	0.998451	0.998451	0.483172	0.598732	1.595227	0.940734	0.974376	0.948081	0.948081	0.948081	0.000012
23	0.980783	0.421442	2.066240	1.016020	0.966728	0.987149	0.987149	0.987149	0.484324	0.610922	1.599242	0.934937	0.972696	0.944902	0.944902	0.944902	0.000012

Figure 61. Experiment summary report for B3 architecture

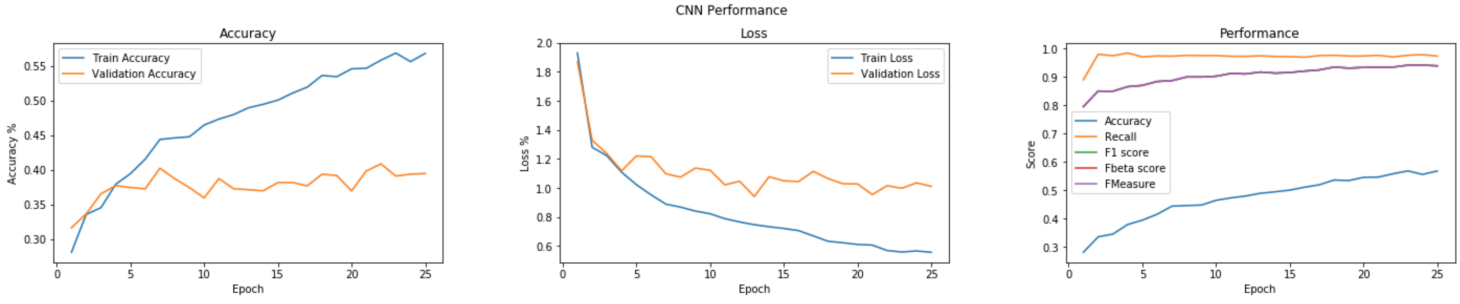


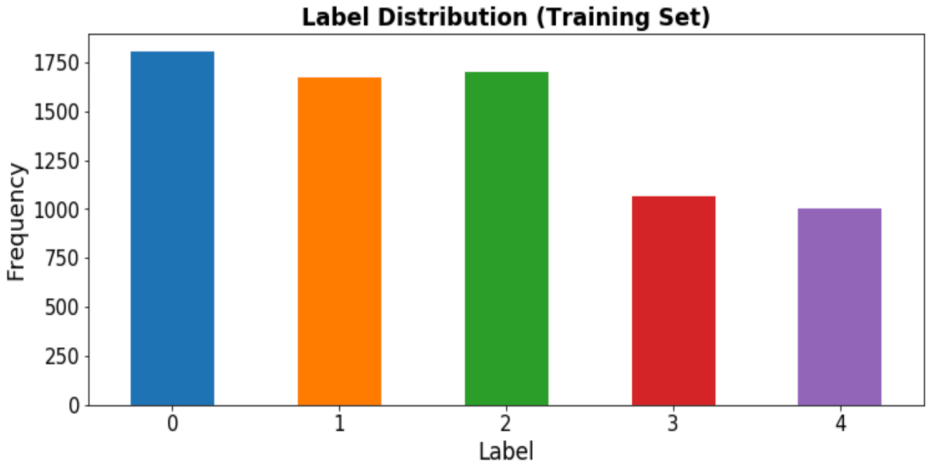
Figure 62. Experiment Plots for B3 architecture

According to the plots represented in Figure 62, validation accuracy reached the top level and will not increase anymore and early stop is quite proves that. Loss graphs look a little more stable than in the previous experiments but there is a difference more than it should be between validation and training loss values so, this model also tends to overfit. Figure 63 shows the kappa evaluation of the model and again validation kappa score is lower than it should be.

```
The Training Cohen Kappa Score is: 0.84651
The Validation Cohen Kappa Score is: 0.51585
```

```
Optimized Thresholds:
[0.53999677 1.68110177 2.17038298 2.98358497]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.56516

This is an improvement of 0.04932
over the unoptimized rounding
```

Figure 63. Experiment kappa results for B3 architecture



Custom Dataset Training Images Distribution

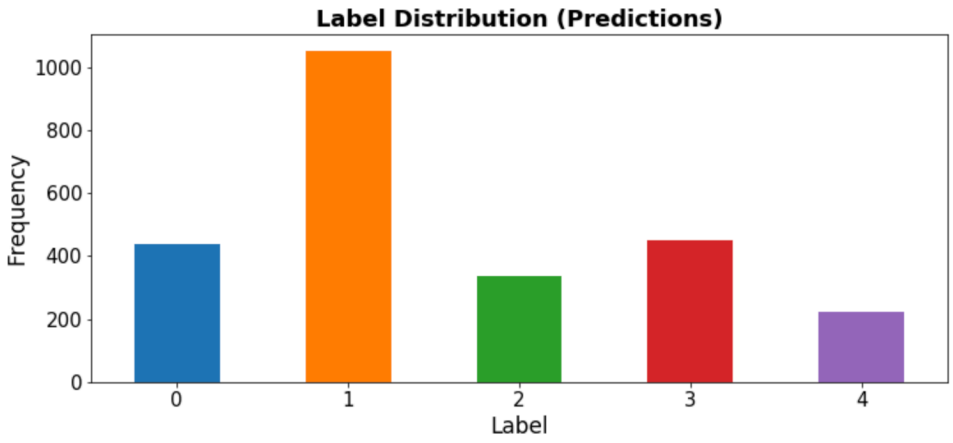


Figure 64. Experiment prediction label distribution for B3 architecture

Figure 64 represents the predictions at the end of the training and in compare to the EfficientNet B5 experiment prediction labels look so similar. Despite having different input sizes image quality does not affect the results significantly in EfficientNet architecture, this information might be commented on according to obtained results.

7.3.4 EfficientNet-B2 Experiments

Figure 65. Training images after preprocess for B2 architecture

To have a better idea of EfficientNet architecture it is aimed to have experimented with most of the models in EfficientNet. This experiment is executed with 25 epochs and took two hours to complete. Training images are used in the experiment represented in Figure 65. As well as the previous experiments, this test also has proper results for the F1, precision, and recall results represented in Figure 67. Also for this experiment, validation accuracy is not good enough to use for the classification of diabetic retinopathy images. Figure 66 shows the graphical results of the experiment the results are very similar to the previous experiments and this model is also close to overfit.

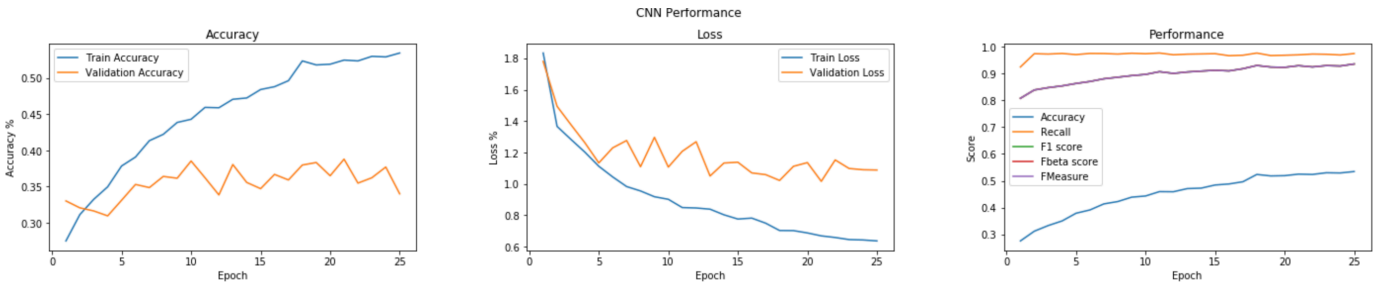


Figure 66. Experiment plots for B2 architecture

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.780081	0.330258	1.489130	1.047355	0.949262	0.989457	0.989457	0.989457	1.833593	0.275016	1.380085	77973.479370	0.925439	0.808289	0.808289	0.808289	0.000050
1	1.493767	0.320702	1.665737	1.033888	0.957486	0.988557	0.988557	0.988557	1.365636	0.311393	1.581413	0.774311	0.975026	0.839477	0.839477	0.839477	0.000050
2	1.376367	0.316421	1.740184	1.021525	0.953875	0.981726	0.981726	0.981726	1.282808	0.332521	1.587521	0.790129	0.973671	0.848210	0.848210	0.848210	0.000050
3	1.261394	0.309612	1.794718	1.026494	0.967652	0.991902	0.991902	0.991902	1.201657	0.349748	1.585973	0.796847	0.975838	0.854720	0.854720	0.854720	0.000050
4	1.133219	0.331181	1.915708	1.007380	0.964945	0.982956	0.982956	0.982956	1.111855	0.378515	1.585548	0.810824	0.971667	0.863746	0.863746	0.863746	0.000050
5	1.229194	0.353050	1.898527	1.025262	0.966728	0.991198	0.991198	0.991198	1.043837	0.390866	1.589961	0.820846	0.976001	0.871211	0.871211	0.871211	0.000050
6	1.275866	0.348708	1.849814	1.033210	0.928967	0.971358	0.971358	0.971358	0.982532	0.413457	1.593932	0.834931	0.975622	0.881267	0.881267	0.881267	0.000050
7	1.109000	0.364140	1.906106	1.015404	0.985213	0.997888	0.997888	0.997888	0.954472	0.422233	1.583987	0.848367	0.973671	0.887126	0.887126	0.887126	0.000050
8	1.296219	0.361624	1.758346	1.031365	0.929889	0.971534	0.971534	0.971534	0.917667	0.438648	1.598319	0.852538	0.976705	0.893147	0.893147	0.893147	0.000050
9	1.106759	0.385397	1.956880	1.013555	0.963956	0.984596	0.984596	0.984596	0.901218	0.443036	1.585686	0.860339	0.974863	0.897619	0.897619	0.897619	0.000050
10	1.206921	0.362292	2.049574	1.024030	0.956562	0.984244	0.984244	0.984244	0.848615	0.459451	1.600868	0.875129	0.977518	0.907879	0.907879	0.907879	0.000050
11	1.268125	0.338561	1.824190	1.030135	0.906827	0.955228	0.955228	0.955228	0.845881	0.458963	1.594038	0.868682	0.970963	0.901467	0.901467	0.901467	0.000050
12	1.049764	0.380556	2.069127	1.012346	0.974074	0.989947	0.989947	0.989947	0.838215	0.470665	1.591319	0.876429	0.973238	0.906777	0.906777	0.906777	0.000050
13	1.132534	0.355823	2.004393	1.013555	0.964880	0.985125	0.985125	0.985125	0.802300	0.472452	1.594485	0.880004	0.974159	0.910037	0.910037	0.910037	0.000050
14	1.137636	0.347222	1.925977	1.011728	0.973148	0.989418	0.989418	0.989418	0.775259	0.484154	1.600618	0.885368	0.975080	0.913100	0.913100	0.913100	0.000050
15	1.069583	0.366913	2.030971	1.016636	0.961183	0.984508	0.984508	0.984508	0.781226	0.488055	1.588494	0.884122	0.967550	0.910760	0.910760	0.910760	0.000050
16	1.058467	0.359259	2.082514	1.009259	0.968519	0.985273	0.985273	0.985273	0.748861	0.496343	1.592883	0.899615	0.969283	0.918963	0.918963	0.918963	0.000050
17	1.021436	0.379852	2.167255	1.017868	0.967652	0.988381	0.988381	0.988381	0.702167	0.523647	1.595340	0.912021	0.977409	0.931069	0.931069	0.931069	0.000025
18	1.111537	0.383333	2.098988	1.022222	0.961111	0.986772	0.986772	0.986772	0.701497	0.518121	1.595090	0.908500	0.968091	0.925189	0.925189	0.925189	0.000025
19	1.135174	0.365065	2.057283	1.014787	0.952865	0.978417	0.978417	0.978417	0.686777	0.519096	1.596955	6501.797876	0.969283	0.924105	0.924105	0.924105	0.000025
20	1.015787	0.387963	2.111185	1.010494	0.978704	0.991887	0.991887	0.991887	0.667973	0.524785	1.593799	0.914947	0.970800	0.930382	0.930382	0.930382	0.000025
21	1.152079	0.354898	2.073569	1.015404	0.958410	0.982044	0.982044	0.982044	0.657663	0.523647	1.600749	0.905195	0.973671	0.925611	0.925611	0.925611	0.000025
22	1.097702	0.362292	2.113306	1.010474	0.965804	0.984156	0.984156	0.984156	0.644364	0.529985	1.592044	0.915272	0.972859	0.930618	0.930618	0.930618	0.000025
23	1.089752	0.377079	2.008955	1.013555	0.950092	0.976411	0.976411	0.976411	0.641911	0.529173	1.600397	0.913863	0.970313	0.929026	0.929026	0.929026	0.000025
24	1.087918	0.340111	2.141983	1.012939	0.958410	0.980635	0.980635	0.980635	0.635835	0.534536	1.594202	0.920635	0.975513	0.936641	0.936641	0.936641	0.000025

Figure 67. Experiment summary report for B2 architecture

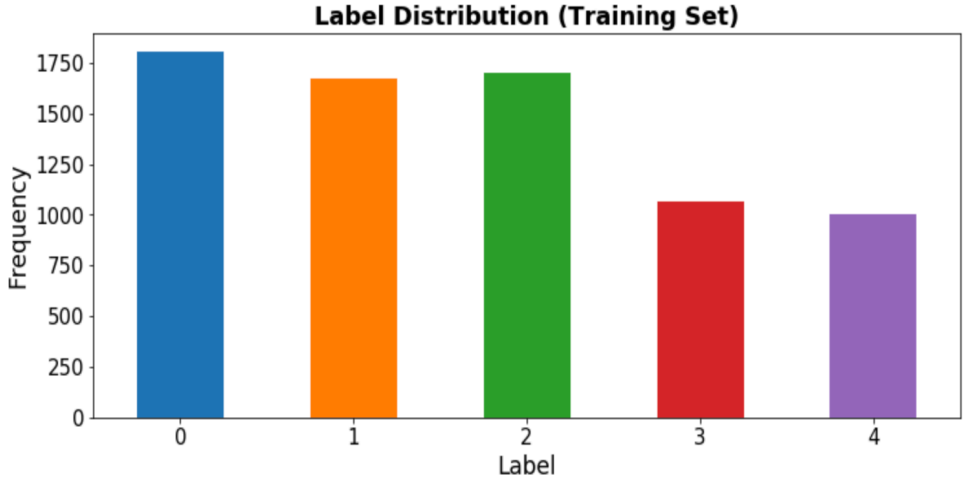
Investigation of the kappa score is shown in Figure 68 and also is not very different from the previous one's validation score has undesired results for this experiment too.

```
The Training Cohen Kappa Score is: 0.82422
The Validation Cohen Kappa Score is: 0.50007
```

```
Optimized Thresholds:
[0.5256871  1.61853298 2.54488175 3.30201876]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.53408

This is an improvement of 0.03401
over the unoptimized rounding
```

Figure 68. Experiment kappa results for B2 architecture

The distribution of the predicted image labels is given below in Figure 69.



Custom Dataset Training Images Distribution

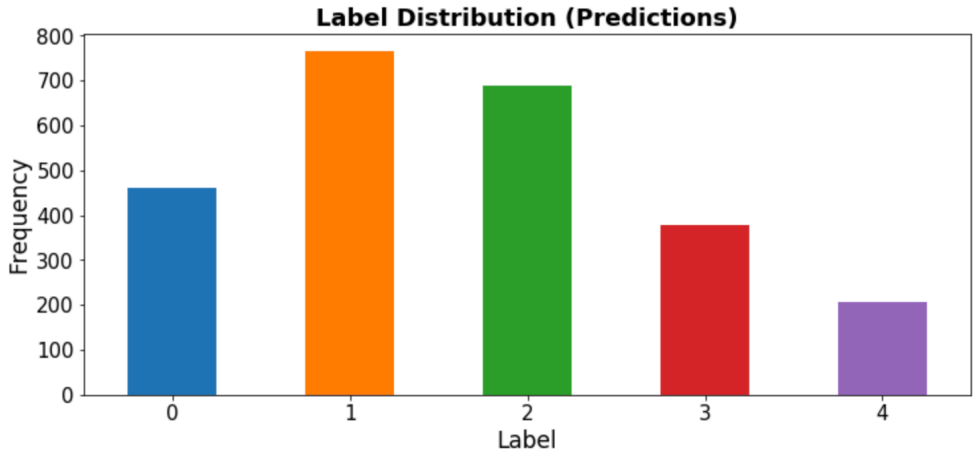


Figure 69. Experiment prediction label distribution for B2 architecture

According to the plots provided in Figure 69, it is possible to mention that the B2 architecture has behaved slightly better than previous models to predict labels for every class in comparison to the provided training set.

7.3.5 EfficientNet B1 Experiment

Among the EfficientNet architectures, this model has the lowest input size with 240x240 and the training images are shown in Figure 70.

Figure 70. Training images after preprocess for B1 architecture

The first experiment held with B1 architecture is done with 10 epochs and details of the experiment can be seen in Figure 71. Related to having experiment with fewer epochs than the previous experiments, the F1 score has lower results than the other experiments but validation accuracy does not show a big difference from the experiments held with more epochs. Overall this architecture also shows similarities to other models of EfficientNet.

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score
0	1.776305	0.314576	1.450306	1.025830	0.944649	0.977333	0.977333	0.977333	1.965923	0.282489	1.295456	168941.660927	0.899394	0.807565	0.807565
1	1.526137	0.313309	1.670862	1.034504	0.954713	0.986093	0.986093	0.986093	1.423493	0.305217	1.568347	0.771114	0.968633	0.834343	0.834343
2	1.319240	0.321956	1.784177	1.015375	0.951107	0.977157	0.977157	0.977157	1.293292	0.333658	1.584669	0.781299	0.973726	0.843578	0.843578
3	1.280747	0.333641	1.865296	1.016020	0.971349	0.990318	0.990318	0.990318	1.218096	0.346173	1.577757	0.785091	0.978330	0.848809	0.848809
4	1.307100	0.348708	1.784638	1.015375	0.953875	0.979441	0.979441	0.979441	1.177088	0.355761	1.583904	0.791267	0.974376	0.852168	0.852168
5	1.198238	0.338262	1.887883	1.010474	0.970425	0.987149	0.987149	0.987149	1.098559	0.372501	1.589964	0.806273	0.978276	0.862262	0.862262
6	1.290604	0.358856	1.884076	1.019680	0.943727	0.974697	0.974697	0.974697	1.032372	0.395254	1.591244	0.815104	0.980877	0.870426	0.870426
7	1.123797	0.357671	2.019092	1.019100	0.963031	0.985829	0.985829	0.985829	1.010516	0.407931	1.582304	0.821225	0.977301	0.872431	0.872431
8	1.282347	0.351476	1.915737	1.027675	0.946494	0.979265	0.979265	0.979265	0.959100	0.422721	1.594684	0.839861	0.976380	0.883971	0.883971
9	1.072834	0.354898	2.252102	1.009858	0.983364	0.994367	0.994367	0.994367	0.909550	0.438485	1.586553	0.851617	0.974267	0.892183	0.892183

Figure 71. Experiment summary report for B1 architecture

```
Optimized Thresholds:
[0.4200412  1.81790732 2.54448922 3.25718652]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.5184

This is an improvement of 0.03165
over the unoptimized rounding
```

```
The Training Cohen Kappa Score is: 0.74479
The Validation Cohen Kappa Score is: 0.48675
```

Figure 72. Experiment kappa results for B1 architecture

Figure 72 shows the kappa performance of the first experiment and once again validation kappa result is not enough to make a good comment on this experiment.

The second experiment done with B1 architecture has different inputs than the first one. Figure 73 shows the training images used for the second experiment.

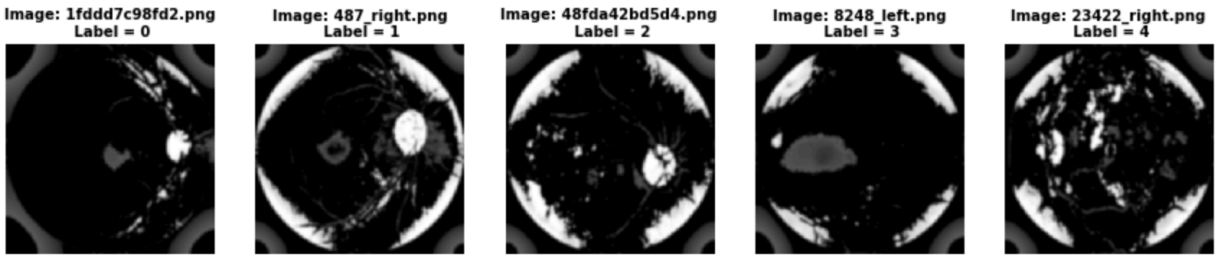


Figure 73. Updated training images after preprocess for B1 architecture

To obtain these training images, different threshold values were applied than the first one and it is aimed to observe the influences of the threshold values on results. The experiment is planned to be done with 25 epochs but the early stop function has cut the experiment at the 24th epoch. Details of the experiment are shown in Figure 75 below. This experiment has one of the highest F1 score and training accuracy among all experiments done by thresholding. However, it is hard to say the same thing for the validation accuracy. Figure 74 shows the Kappa score performance and also for this test kappa values are so low to consider as good.

```
Optimized Thresholds:
[0.54992805 1.65315561 2.11150351 3.20372412]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.56622

This is an improvement of 0.04411
over the unoptimized rounding
```

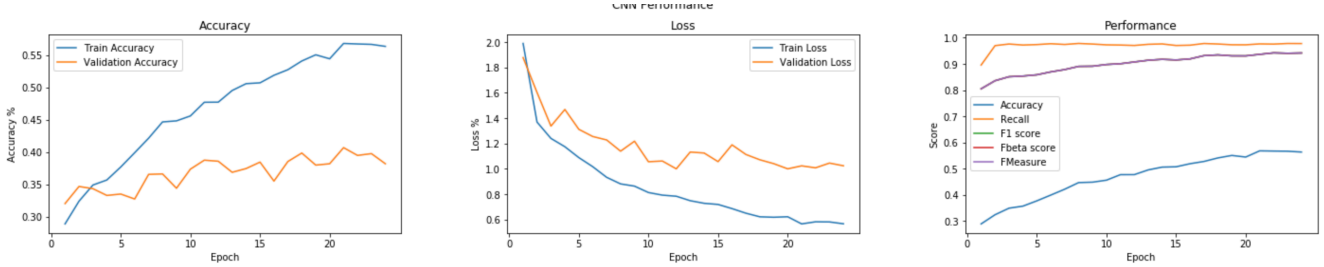
```
The Training Cohen Kappa Score is: 0.84381
The Validation Cohen Kappa Score is: 0.52211
```

Figure 74. Experiment Kappa results for updated B1 experiment

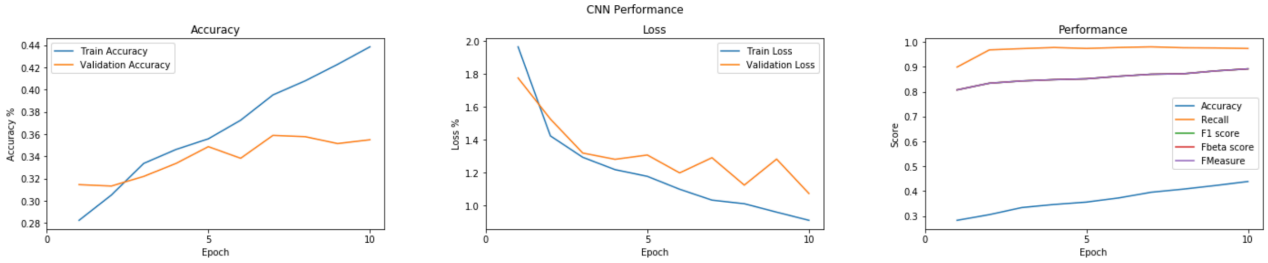
	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.878479	0.320111	1.379452	1.039975	0.934502	0.976805	0.976805	0.976805	1.990080	0.288661	1.279441	181937.110624	0.895657	0.804818	0.804818	0.804818	0.000050
1	1.603148	0.346580	1.511265	1.035736	0.930684	0.971059	0.971059	0.971059	1.369580	0.323745	1.579365	0.776478	0.969771	0.835818	0.835818	0.835818	0.000050
2	1.338023	0.343173	1.698810	1.018450	0.962177	0.985943	0.985943	0.985943	1.240733	0.348773	1.582646	0.793542	0.975947	0.851021	0.851021	0.851021	0.000050
3	1.469237	0.332717	1.625977	1.043130	0.928835	0.975178	0.975178	0.975178	1.175158	0.356574	1.581553	0.798635	0.972046	0.853657	0.853657	0.853657	0.000050
4	1.312719	0.334871	1.807586	1.043050	0.953875	0.989808	0.989808	0.989808	1.089590	0.376727	1.590882	0.804648	0.973780	0.858448	0.858448	0.858448	0.000050
5	1.256014	0.327172	1.866489	1.017252	0.955638	0.981516	0.981516	0.981516	1.017731	0.398830	1.588143	0.819979	0.977084	0.869519	0.869519	0.869519	0.000050
6	1.228580	0.365314	1.848463	1.030750	0.945572	0.980671	0.980671	0.980671	0.933799	0.421258	1.587305	0.834606	0.974213	0.878246	0.878246	0.878246	0.000050
7	1.140111	0.365989	1.915420	1.007394	0.987061	0.995775	0.995775	0.995775	0.881798	0.446449	1.589779	0.851292	0.978168	0.890468	0.890468	0.890468	0.000050
8	1.218790	0.343808	2.116087	1.027726	0.968577	0.992606	0.992606	0.992606	0.864296	0.448074	1.596683	0.850371	0.975838	0.891095	0.891095	0.891095	0.000050
9	1.056684	0.373383	2.008268	1.019717	0.963031	0.985741	0.985741	0.985741	0.813875	0.455713	1.588927	0.864023	0.972750	0.897659	0.897659	0.897659	0.000050
10	1.062300	0.387246	2.042875	1.014171	0.970425	0.988381	0.988381	0.988381	0.793281	0.476841	1.590763	0.866407	0.971992	0.900651	0.900651	0.900651	0.000050
11	1.000741	0.385609	2.033396	1.019065	0.981550	0.996837	0.996837	0.996837	0.784534	0.477003	1.598878	0.878975	0.970367	0.907461	0.907461	0.907461	0.000050
12	1.133338	0.368519	1.964676	1.017284	0.964815	0.986243	0.986243	0.986243	0.749300	0.495043	1.596848	0.888022	0.974809	0.914036	0.914036	0.914036	0.000050
13	1.125583	0.374307	2.023122	1.017252	0.978743	0.994895	0.994895	0.994895	0.728741	0.505607	1.591388	0.890839	0.976272	0.917544	0.917544	0.917544	0.000050
14	1.057352	0.384259	2.028060	1.012346	0.971296	0.987372	0.987372	0.987372	0.719815	0.506907	1.593865	0.891435	0.970258	0.915122	0.915122	0.915122	0.000050
15	1.190075	0.354898	2.062695	1.022797	0.975046	0.994895	0.994895	0.994895	0.687056	0.518609	1.597884	0.896907	0.971450	0.919157	0.919157	0.919157	0.000050
16	1.114090	0.385185	2.122072	1.008642	0.973148	0.988183	0.988183	0.988183	0.650777	0.527222	1.594969	0.911371	0.977951	0.931604	0.931604	0.931604	0.000025
17	1.072132	0.398336	2.058759	1.026494	0.965804	0.991022	0.991022	0.991022	0.622276	0.540712	1.596507	0.916680	0.976001	0.934542	0.934542	0.934542	0.000025
18	1.041389	0.379630	2.072088	1.023457	0.969444	0.992593	0.992593	0.992593	0.618662	0.550301	1.595491	0.913917	0.973075	0.930874	0.930874	0.930874	0.000025
19	1.000783	0.381701	2.158303	1.007394	0.977819	0.990494	0.990494	0.990494	0.622663	0.544125	1.592952	0.913972	0.972913	0.930713	0.930713	0.930713	0.000025
20	1.024316	0.406654	2.149795	1.014171	0.975046	0.991374	0.991374	0.991374	0.566244	0.567853	1.599255	0.919931	0.976380	0.936554	0.936554	0.936554	0.000012
21	1.008701	0.394640	2.067051	1.016020	0.962107	0.983699	0.983699	0.983699	0.582712	0.567040	1.594744	0.931361	0.975622	0.942277	0.942277	0.942277	0.000012
22	1.046052	0.397412	2.074226	1.014787	0.974122	0.991022	0.991022	0.991022	0.581708	0.566390	1.598344	0.926811	0.978005	0.940752	0.940752	0.940752	0.000012
23	1.024783	0.381701	2.146673	1.011091	0.975046	0.989966	0.989966	0.989966	0.567102	0.563302	1.598271	0.928003	0.977734	0.941795	0.941795	0.941795	0.000012

Figure 75. Experiment summary report for updated B1 experiment

Figure 76 presents the both experiment's graphical summary and can be seen below,



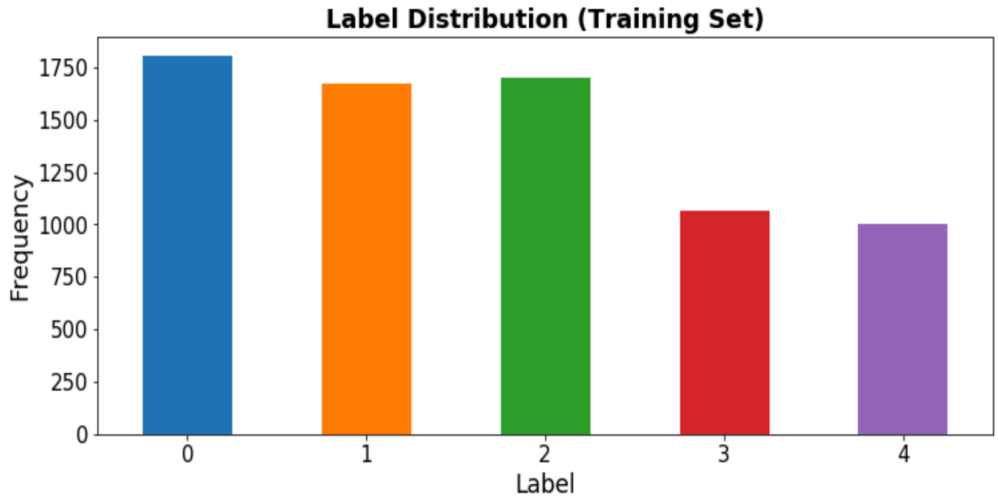
Second Experiment Plots



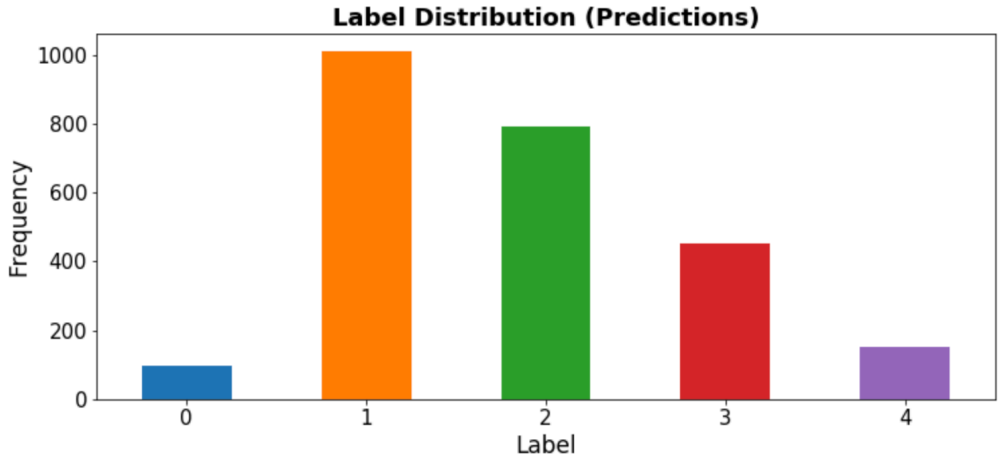
First Experiment Plots

Figure 76. Experiment plots of B1 experiments

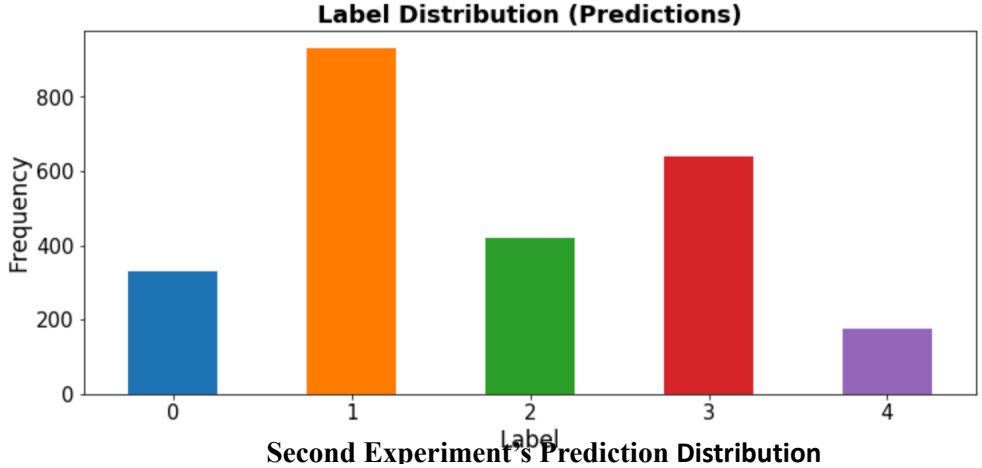
It is possible to observe that the results show similarities, the first experiment has more optimal loss values and is less likely to have overfitting issues until the 10th epoch, in the second experiment, the loss graph shows less stable validation loss and is more likely to overfit. Also having more epochs did not affect the validation accuracy for this experiment. Another comparison might be handled in Figure 77. The prediction results are given for both experiments.



Custom Dataset Training Images Distribution



First Experiment's Prediction Distribution



Second Experiment's Prediction Distribution

Figure 77. Experiment prediction label distribution for B1 experiments

7.3.6 ResNet50 Experiment

This experiment is done with ResNet50 architecture to compare models from EfficientNet experiments. The experiment took almost three hours to complete and it was one of the slowest test among the types of experiments held previously.

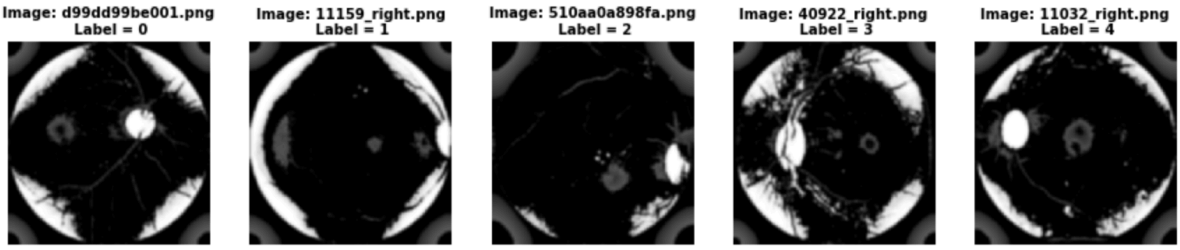


Figure 78. Training images of ResNet50 after preprocess

The same threshold values were used in the second experiment of the EfficientNet B1 experiment for the training images for this experiment, preprocessed images can be seen in Figure 78. As well as the second experiment of the B1 experiment, this experiment also has a high F1 score and training accuracy. However, validation accuracy is not good enough to perform trustful classification. Details about the experiments are presented in Figure 79 below,

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.397157	0.363469	1.761618	1.014760	0.985240	0.997364	0.997364	0.997364	1.807615	0.282814	1.458503	51982.572775	0.920403	0.807073	0.807073	0.807073	0.000050
1	1.509058	0.365989	1.715053	1.056685	0.945471	0.987360	0.987360	0.987360	1.279140	0.330895	1.576789	0.777778	0.975730	0.841914	0.841914	0.841914	0.000050
2	1.415073	0.357934	1.745653	1.016605	0.970480	0.989984	0.989984	0.989984	1.163928	0.357224	1.591929	0.798581	0.973671	0.855736	0.855736	0.855736	0.000050
3	1.549640	0.310536	1.757282	1.028343	0.967652	0.992958	0.992958	0.992958	1.073593	0.389566	1.595553	0.812558	0.976001	0.865837	0.865837	0.865837	0.000050
4	1.184181	0.369004	2.006800	1.010455	0.970480	0.987348	0.987348	0.987348	1.042726	0.407769	1.593007	0.826318	0.972588	0.872627	0.872627	0.872627	0.000050
5	1.246832	0.336414	2.243055	1.011707	0.984288	0.995775	0.995775	0.995775	0.957899	0.428246	1.596584	0.844141	0.969879	0.884108	0.884108	0.884108	0.000050
6	1.267136	0.340406	1.842519	1.041820	0.936347	0.978000	0.978000	0.978000	0.916631	0.438810	1.597417	0.844358	0.971125	0.884886	0.884886	0.884886	0.000050
7	1.223310	0.338262	1.875933	1.020949	0.983364	0.998592	0.998592	0.998592	0.902526	0.451162	1.592798	0.857847	0.970367	0.892587	0.892587	0.892587	0.000050
8	1.470675	0.335793	1.899833	1.047970	0.923432	0.970937	0.970937	0.970937	0.871849	0.451812	1.603082	0.857468	0.971775	0.894667	0.894667	0.894667	0.000050
9	1.233796	0.358595	1.941729	1.022797	0.939002	0.973154	0.973154	0.973154	0.802361	0.484154	1.598572	0.875020	0.972642	0.906913	0.906913	0.906913	0.000025
10	1.047544	0.353974	1.995482	1.011707	0.986137	0.996655	0.996655	0.996655	0.763422	0.490817	1.603798	0.883580	0.967279	0.909576	0.909576	0.909576	0.000025
11	1.139771	0.358595	1.947514	1.020333	0.940850	0.973770	0.973770	0.973770	0.751131	0.504794	1.594167	0.888997	0.969337	0.914015	0.914015	0.914015	0.000025
12	1.016017	0.379630	2.163751	1.013580	0.975926	0.991534	0.991534	0.991534	0.712011	0.516659	1.596118	0.898044	0.971559	0.919762	0.919762	0.919762	0.000025
13	1.198122	0.351201	1.800959	1.027726	0.934381	0.972362	0.972362	0.972362	0.728049	0.517959	1.602387	0.897394	0.973888	0.920414	0.920414	0.920414	0.000025
14	1.072275	0.356481	2.010358	1.025309	0.967593	0.991534	0.991534	0.991534	0.687235	0.527548	1.601151	0.900049	0.971342	0.922091	0.922091	0.922091	0.000025
15	1.054583	0.350277	2.030469	1.014787	0.969501	0.988029	0.988029	0.988029	0.684350	0.526735	1.599687	0.898261	0.970638	0.920093	0.920093	0.920093	0.000025
16	0.957896	0.407407	2.164298	1.004938	0.979630	0.990476	0.990476	0.990476	0.662735	0.529335	1.600000	0.906116	0.971071	0.925186	0.925186	0.925186	0.000025
17	1.107088	0.385397	1.960024	1.020949	0.955638	0.981868	0.981868	0.981868	0.656884	0.539899	1.595268	0.911371	0.978818	0.932341	0.932341	0.932341	0.000025
18	1.027458	0.378704	2.073225	1.012346	0.979630	0.993122	0.993122	0.993122	0.648408	0.541850	1.598137	6501.811745	0.972209	0.933869	0.933869	0.933869	0.000025
19	1.223868	0.371534	1.848954	1.043130	0.935305	0.977713	0.977713	0.977713	0.640818	0.537786	1.597024	0.908067	0.972425	0.926473	0.926473	0.926473	0.000025
20	1.151153	0.381481	2.158115	1.019136	0.964815	0.986949	0.986949	0.986949	0.619430	0.554851	1.596509	0.916138	0.967062	0.929445	0.929445	0.929445	0.000025
21	1.056704	0.400185	2.093581	1.009858	0.977819	0.990846	0.990846	0.990846	0.566518	0.575004	1.602041	6501.819113	0.971775	0.939066	0.939066	0.939066	0.000012
22	1.003260	0.402033	2.126588	1.017868	0.980591	0.996303	0.996303	0.996303	0.552792	0.580530	1.599950	0.928003	0.977355	0.942018	0.942018	0.942018	0.000012
23	1.023112	0.404806	2.159744	1.004929	0.974122	0.986797	0.986797	0.986797	0.557362	0.574029	1.601985	0.924265	0.974917	0.938960	0.938960	0.938960	0.000012
24	1.064769	0.378004	2.256507	1.009858	0.979667	0.991990	0.991990	0.991990	0.533901	0.582643	1.601524	0.931741	0.973346	0.943179	0.943179	0.943179	0.000012

Figure 79. Experiment summary report of ResNet50

```
The Training Cohen Kappa Score is: 0.83077
The Validation Cohen Kappa Score is: 0.55181
```

```
Optimized Thresholds:
[0.53486729 1.67251255 2.20000711 3.18059527]

The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.56114

This is an improvement of 0.00933
over the unoptimized rounding
```

Figure 80. Experiment kappa results of ResNet50

In Figure 80 Kappa scores are shown and this experiment has one of the highest training and validation kappa scores but unfortunately below 0.65 kappa values do not consider as good so this experiment is not successful for classification. Figure 81 shows more information about the details of the experiment.

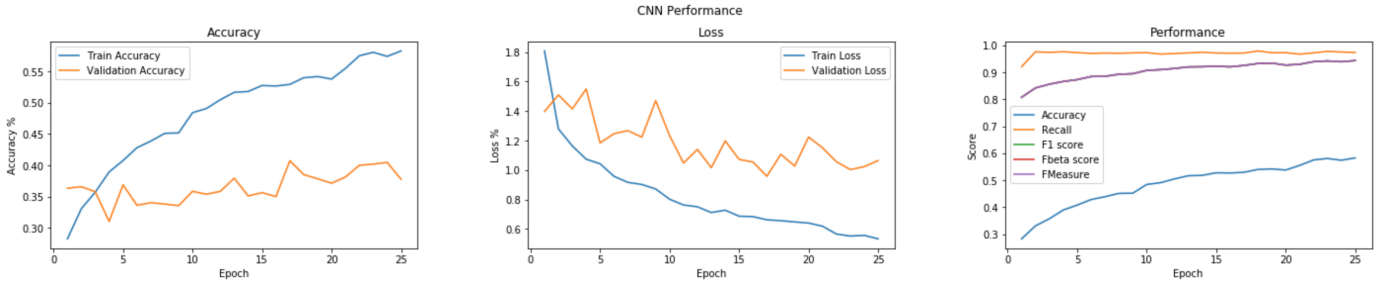


Figure 81. Experiment Plots of ResNet50

With regard to graphs, it is easy to say that the model is heavily overfitted and trained poorly. So this model is also not appropriate to use for classification purposes. Figure 82 shows the predicted labels at the end of the experiment with the test set and the prediction distribution is close to previous experiments.

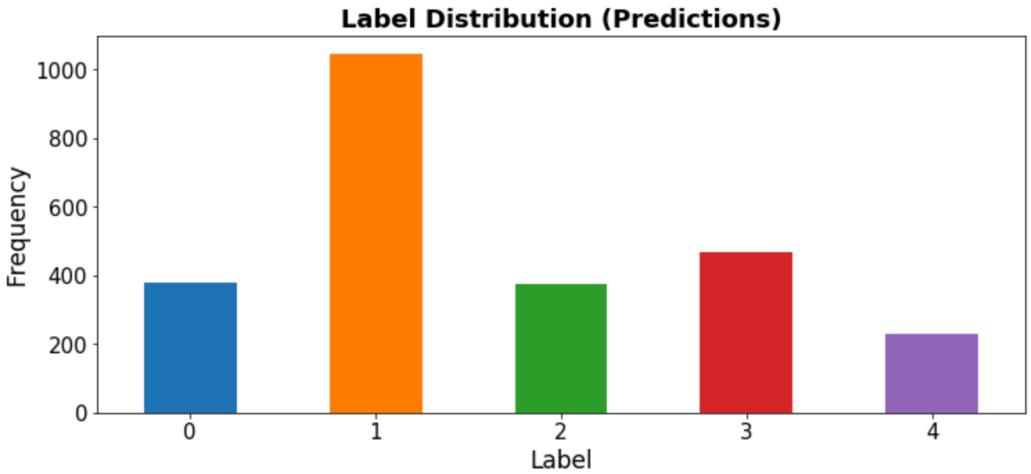
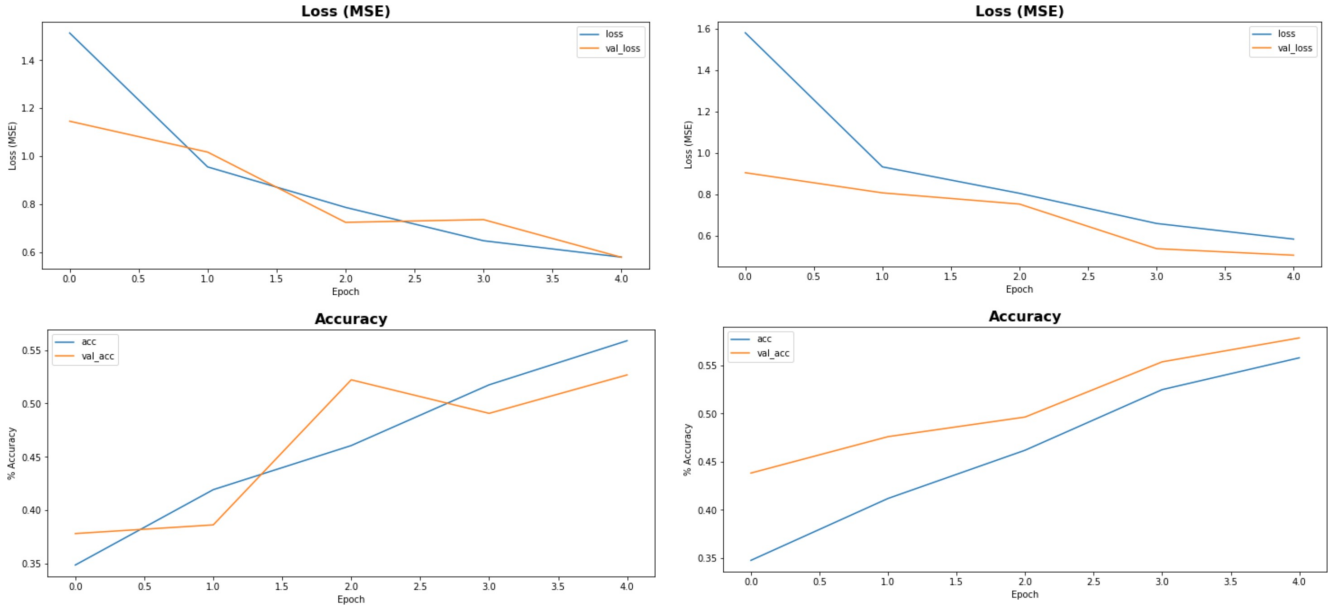


Figure 82. Experiment prediction label distribution of ResNet50

7.4 SigmaX Function Experiment

Figure 83. Training images of SigmaX experiment after preprocess

In order to execute this experiment, SigmaX = 10 function is used to get samples represented in Figure 83. As it is handled during previous experiments, except for image sizes, every other parameter was the same also in this experiment. As can be seen in Figure 84, both architectures had quite well training and were less likely to overfit in comparison to the B4 model in the HSV experiment. But again, the EfficientNet B5 model had a more balanced training and validation sessionthan the B4 model also in this experiment. According to kappa scores given in Figure 85 both models have led in some parts but with the combination of loss accuracy and kappa scores, EfficientNet B5 did a better job during this experiment and had better label distribution (Figure 86) than B4 architecture.



EfficientNet B4 results **EfficientNet B5 results**

Figure 84. Accuracy and Loss Results of SigmaX Experiment

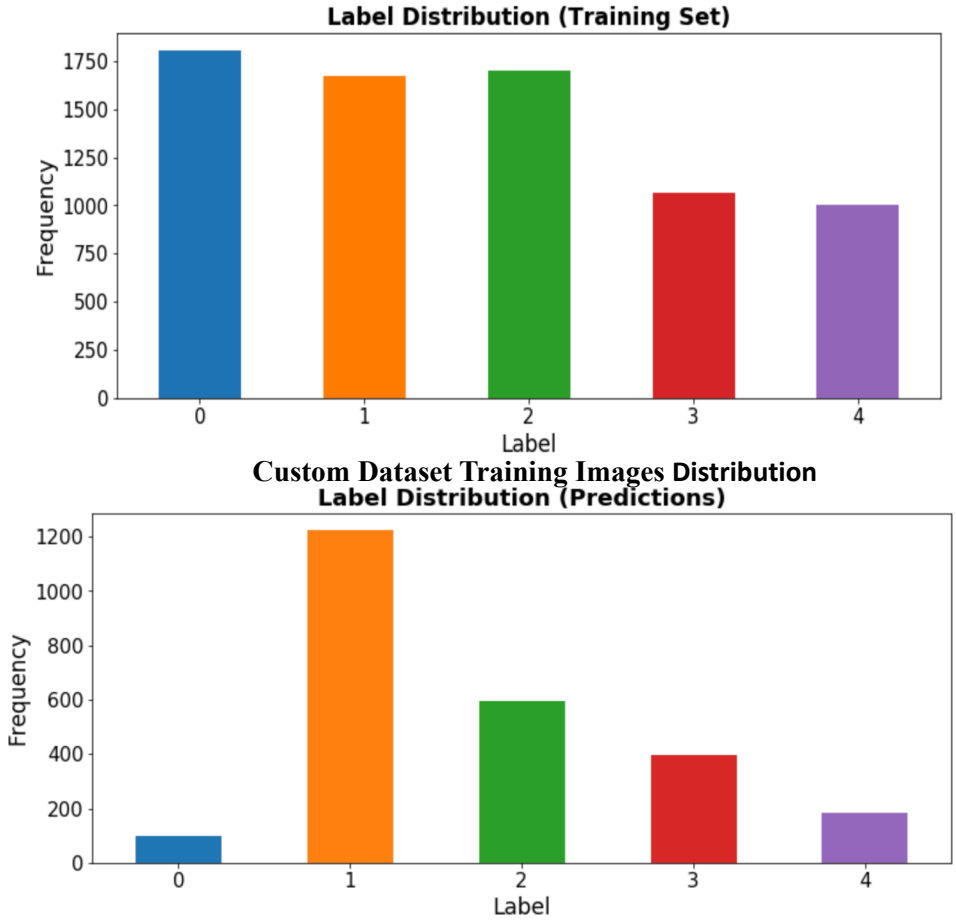
```
The Training Cohen Kappa Score is: 0.87454
The Validation Cohen Kappa Score is: 0.7155
```

EfficientNet B4 results

```
The Training Cohen Kappa Score is: 0.83839
The Validation Cohen Kappa Score is: 0.72319
```

EfficientNet B5 results

Figure 85. Kappa Score Results of SigmaX Experiment



EfficientNet B4 results

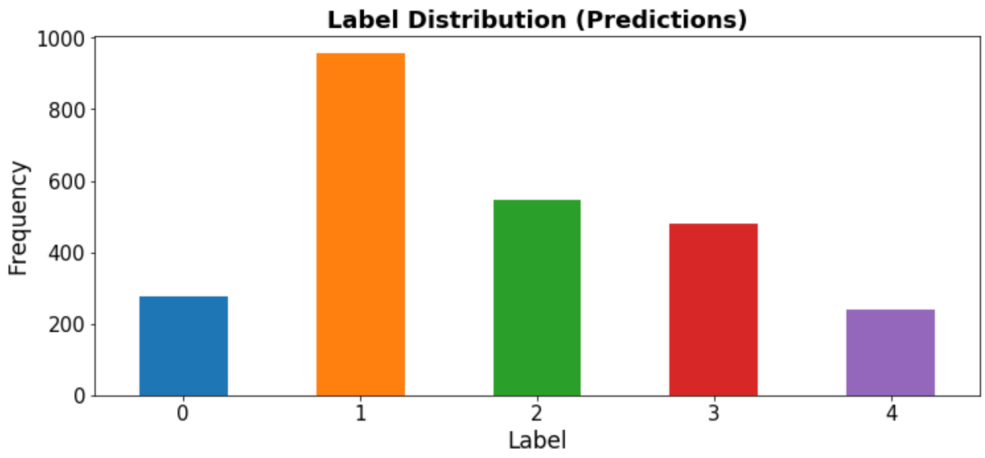


Figure 86. Experiment prediction label distribution of SigmaX experiment

7.5 EfficientNet B1 Experiments with Color Spaces Without Threshold

During this series of experiments, it is aimed to test the color spaces mentioned in the earlier sections without thresholding the images. As can be observed from the previous experiments done in the thesis, it is very hard to obtain good results with threshold applications in many ways. For the final experiments, HSV and HSL cylindric color spaces and Lab chromatic color space is used. Figure 87 represents the training images after preprocessing. Conversion of filters to the custom dataset images from RGB color filter to desired color space has significantly changed the colors of the training images. Figure 88 shows the details of both three experiments and among the preprocessed experiments the best results are obtained so far in these tests for the HSV and HSL filters. However, it is not possible to say the same thing for the Lab filter. The lab filter has very poor results and it can easily be seen in Figures 89 and 90. Loss graphs in Figure 88 show that while HSV and HSL filters have proper training and F1 scores, the Lab filter is totally overfitted and trained extremely badly. In figure 89, observation of the kappa scores tells similar stories as the other metrics. Despite having the same parameters for the experiments, the Lab filter once again has a very bad score as the Kappa score too and the other two filters have good agreement scores in comparison to the thresholded experiments. Figure 91 presents the final predictions of the experiments. HSV and HSL filters performed almost similarly and the Lab filter naturally made predictions very bad as well as the other scores. All figures are demonstrated below,

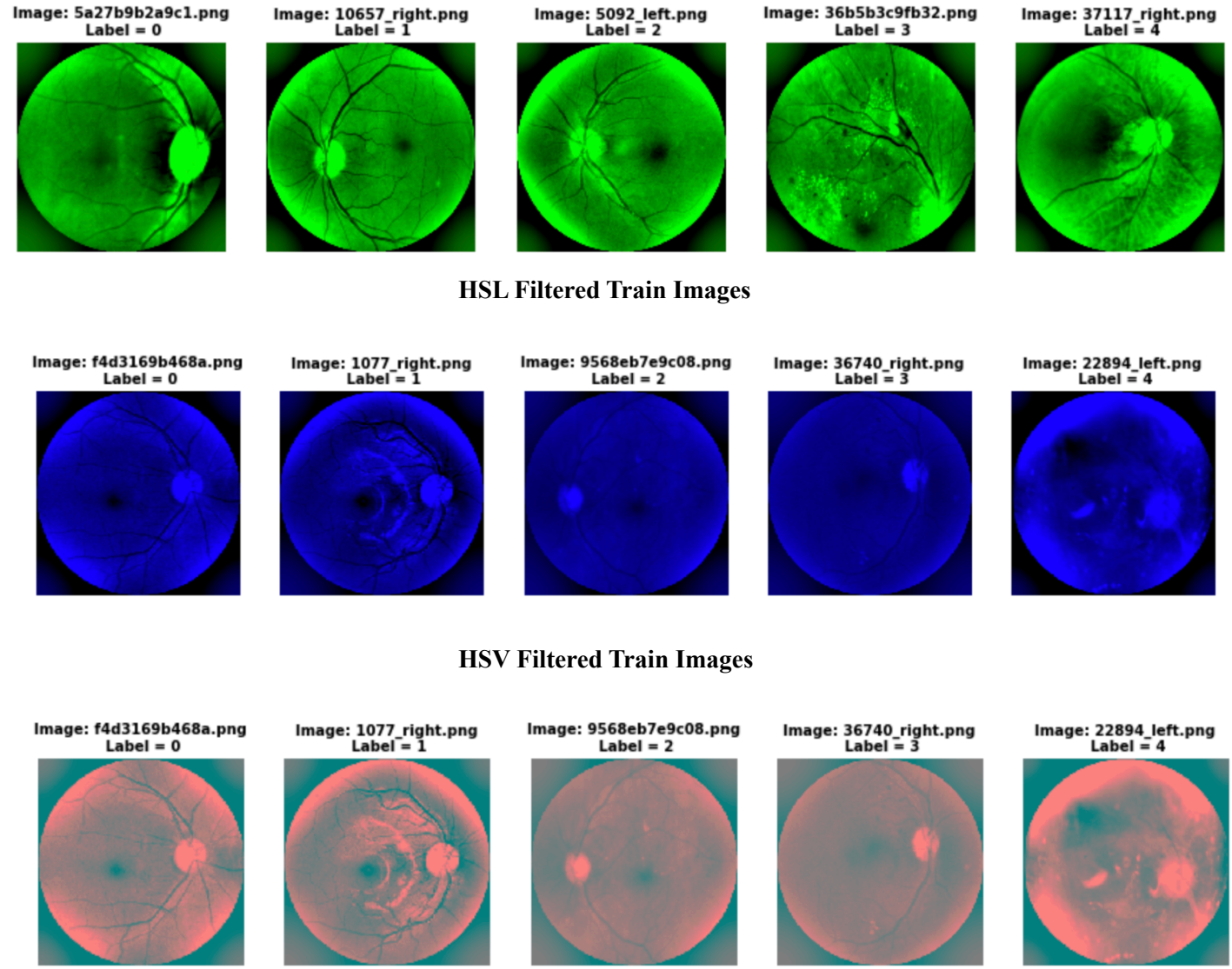


Figure 87. Training images of experiments after preprocess

	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.433515	0.335793	1.532239	1.033825	0.905904	0.955052	0.955052	0.955052	1.698679	0.326998	1.303851	188434.865605	0.893979	0.823318	0.823318	0.823318	0.000050
1	1.114808	0.395564	1.689610	1.039433	0.916821	0.964669	0.964669	0.964669	1.088057	0.387778	1.573441	0.823988	0.964462	0.864119	0.864119	0.864119	0.000050
2	0.936048	0.415129	1.744451	1.010455	0.948339	0.973467	0.973467	0.973467	0.970432	0.407769	1.583164	0.837261	0.967225	0.873028	0.873028	0.873028	0.000050
3	0.877506	0.433457	1.866118	1.011091	0.960259	0.980988	0.980988	0.980988	0.842493	0.450024	1.583942	0.851725	0.973834	0.886612	0.886612	0.886612	0.000050
4	0.759988	0.489852	2.047040	1.007380	0.977860	0.990511	0.990511	0.990511	0.769269	0.479928	1.581025	0.863102	0.967875	0.893381	0.893381	0.893381	0.000050
5	0.708766	0.502773	2.006362	1.008010	0.978743	0.991022	0.991022	0.991022	0.683412	0.514383	1.587330	0.879733	0.971559	0.906681	0.906681	0.906681	0.000050
6	0.736341	0.482472	2.004431	1.006150	0.985240	0.994201	0.994201	0.994201	0.625222	0.534536	1.592121	0.896473	0.977897	0.921180	0.921180	0.921180	0.000050
7	0.647065	0.525878	2.099694	1.007394	0.974122	0.988205	0.988205	0.988205	0.585279	0.554526	1.585061	0.909367	0.974917	0.927756	0.927756	0.927756	0.000050
8	0.731229	0.500924	1.997663	1.012939	0.960259	0.981868	0.981868	0.981868	0.580232	0.555176	1.597196	0.914892	0.977680	0.933536	0.933536	0.933536	0.000050
9	0.702404	0.502773	1.960538	1.006778	0.968577	0.984156	0.984156	0.984156	0.538998	0.580692	1.585126	0.928924	0.973346	0.939704	0.939704	0.939704	0.000050
10	0.644982	0.536044	2.059101	1.004929	0.977819	0.989438	0.989438	0.989438	0.514479	0.588818	1.600585	0.932499	0.976543	0.943777	0.943777	0.943777	0.000050
11	0.751994	0.503690	1.901475	1.011070	0.952030	0.976805	0.976805	0.976805	0.493906	0.600845	1.591840	0.931578	0.971884	0.941086	0.941086	0.941086	0.000050
12	0.682493	0.515741	1.973618	1.007407	0.981481	0.992593	0.992593	0.992593	0.473145	0.603120	1.595941	0.940625	0.976705	0.948344	0.948344	0.948344	0.000050
13	0.642432	0.507394	2.078695	1.003697	0.991682	0.996831	0.996831	0.996831	0.463314	0.609946	1.593887	0.945934	0.979089	0.953743	0.953743	0.953743	0.000050
14	0.681700	0.504630	2.011238	1.003704	0.992593	0.997266	0.997266	0.997266	0.451986	0.614985	1.597213	0.946693	0.976543	0.952443	0.952443	0.952443	0.000050
15	0.656081	0.526802	2.046923	1.010474	0.974122	0.989262	0.989262	0.989262	0.436730	0.627336	1.589236	6501.832439	0.969662	0.946552	0.946552	0.946552	0.000050
16	0.651870	0.537037	2.027985	1.006173	0.980556	0.991358	0.991358	0.991358	0.426804	0.633837	1.595568	0.942521	0.977409	0.952000	0.952000	0.952000	0.000050
17	0.589197	0.557301	2.065029	1.000000	0.987985	0.992958	0.992958	0.992958	0.413007	0.638550	1.596223	0.953952	0.978926	0.958726	0.958726	0.958726	0.000050
18	0.605094	0.534259	2.141246	1.004938	0.971296	0.984092	0.984092	0.984092	0.401329	0.649114	1.593379	0.958340	0.977139	0.962248	0.962248	0.962248	0.000050
19	0.636328	0.564815	2.146005	1.001235	0.994444	0.997354	0.997354	0.997354	0.392215	0.660978	1.596338	0.955523	0.975405	0.958919	0.958919	0.958919	0.000050
20	0.639508	0.523105	2.147022	1.003697	0.988909	0.995247	0.995247	0.995247	0.373971	0.659028	1.596274	0.956065	0.975351	0.959800	0.959800	0.959800	0.000050
21	0.659616	0.528651	2.091381	1.003697	0.987061	0.994191	0.994191	0.994191	0.372631	0.663741	1.597275	0.958882	0.976705	0.962128	0.962128	0.962128	0.000050
22	0.606229	0.555453	2.155900	1.002465	0.987061	0.993663	0.993663	0.993663	0.331961	0.680644	1.593699	0.965708	0.978114	0.966729	0.966729	0.966729	0.000025
23	0.667117	0.530499	1.987221	1.002465	0.985213	0.992430	0.992430	0.992430	0.331208	0.687307	1.599019	0.966466	0.979576	0.967969	0.967969	0.967969	0.000025
24	0.647545	0.529575	2.115462	1.004929	0.982440	0.992078	0.992078	0.992078	0.340299	0.682919	1.594834	0.973455	0.983152	0.974272	0.974272	0.974272	0.000025

HSL Experiment Summary

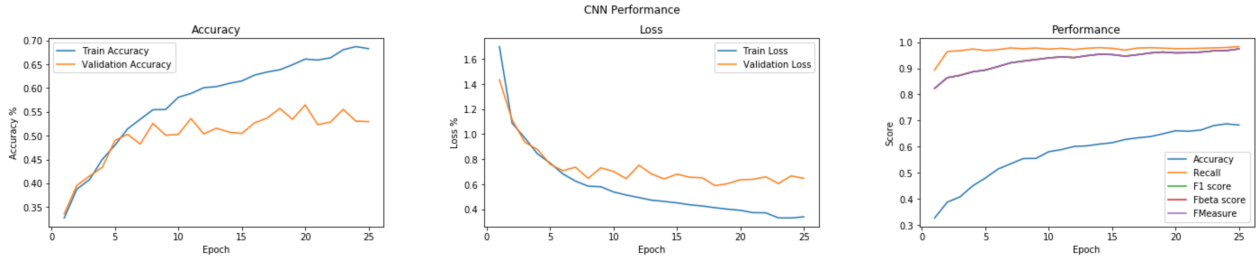
	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	1.567940	0.327491	1.445329	1.022755	0.893911	0.944368	0.944368	0.944368	1.717402	0.323749	1.305769	181937.149881	0.892571	0.825635	0.825635	0.825635	0.000050
1	1.057242	0.390943	1.695964	1.021565	0.927911	0.966816	0.966816	0.966816	1.105897	0.384365	1.573660	0.823176	0.967225	0.865373	0.865373	0.865373	0.000050
2	0.902296	0.438192	1.804045	1.007380	0.967712	0.984361	0.984361	0.984361	0.983657	0.405331	1.582667	0.835636	0.968796	0.873276	0.873276	0.873276	0.000050
3	0.761083	0.440850	2.089666	1.002465	0.989834	0.995247	0.995247	0.995247	0.874168	0.437185	1.583816	0.845875	0.970692	0.881015	0.881015	0.881015	0.000050
4	0.712473	0.497232	2.069586	1.001230	0.986162	0.992620	0.992620	0.992620	0.787686	0.471477	1.581706	0.862885	0.966629	0.892015	0.892015	0.892015	0.000050
5	0.717254	0.499076	2.019695	1.003697	0.987061	0.994015	0.994015	0.994015	0.687375	0.508532	1.591721	0.881413	0.970908	0.906483	0.906483	0.906483	0.000050
6	0.756617	0.473247	1.977564	1.006150	0.984317	0.993674	0.993674	0.993674	0.632693	0.529660	1.590651	0.899019	0.975622	0.921947	0.921947	0.921947	0.000050
7	0.645046	0.504621	2.074001	1.007394	0.971349	0.986797	0.986797	0.986797	0.591493	0.560540	1.585016	0.907579	0.970583	0.923967	0.923967	0.923967	0.000050
8	0.737621	0.500923	1.984986	1.009840	0.959410	0.980320	0.980320	0.980320	0.562779	0.565415	1.595560	0.919064	0.975730	0.935059	0.935059	0.935059	0.000050
9	0.682339	0.511091	2.111002	1.000000	0.981516	0.989085	0.989085	0.989085	0.542195	0.574842	1.586268	0.927515	0.975080	0.940230	0.940230	0.940230	0.000050
10	0.631621	0.521257	2.147713	1.003697	0.978743	0.989438	0.989438	0.989438	0.501671	0.599057	1.599669	0.939271	0.980877	0.950302	0.950302	0.950302	0.000050
11	0.691113	0.523063	2.013447	1.008610	0.971402	0.986646	0.986646	0.986646	0.491386	0.603283	1.591696	0.935695	0.974647	0.946508	0.946508	0.946508	0.000050
12	0.632634	0.535185	2.076406	1.002469	0.990741	0.995767	0.995767	0.995767	0.477847	0.609459	1.594934	0.943280	0.979522	0.952268	0.952268	0.952268	0.000050
13	0.673413	0.522181	2.116847	1.000000	0.993530	0.996303	0.996303	0.996303	0.459586	0.613684	1.592451	0.946043	0.976434	0.952876	0.952876	0.952876	0.000050
14	0.648694	0.512963	2.091145	1.000000	0.997222	0.998413	0.998413	0.998413	0.449358	0.612872	1.599892	0.952164	0.979522	0.957885	0.957885	0.957885	0.000050
15	0.652871	0.533272	2.149968	1.002465	0.985213	0.992430	0.992430	0.992430	0.416400	0.635462	1.592628	0.945067	0.973996	0.952666	0.952666	0.952666	0.000025
16	0.594951	0.549074	2.124627	1.003704	0.990741	0.996296	0.996296	0.996296	0.398074	0.650902	1.595360	0.959586	0.976488	0.962145	0.962145	0.962145	0.000025
17	0.647115	0.548059	2.052161	1.003697	0.981516	0.990758	0.990758	0.990758	0.392946	0.648952	1.594305	0.957690	0.980768	0.962765	0.962765	0.962765	0.000025
18	0.610015	0.533333	2.203348	1.002469	0.987963	0.994180	0.994180	0.994180	0.380716	0.652852	1.596303	0.959532	0.976109	0.962358	0.962358	0.962358	0.000025
19	0.654860	0.562037	2.130884	1.002469	0.989815	0.995238	0.995238	0.995238	0.375216	0.662441	1.595060	0.956607	0.975459	0.959677	0.959677	0.959677	0.000025
20	0.651154	0.525878	2.213701	1.002465	0.987985	0.994191	0.994191	0.994191	0.368583	0.666504	1.595207	0.959857	0.977518	0.963270	0.963270	0.963270	0.000025
21	0.655618	0.526802	2.153987	1.003697	0.983364	0.992078	0.992078	0.992078	0.365566	0.663091	1.602142	0.959044	0.980064	0.963815	0.963815	0.963815	0.000012
22	0.610703	0.553604	2.179106	1.001232	0.988909	0.994015	0.994015	0.994015	0.350387	0.672843	1.592572	0.963649	0.978655	0.965203	0.965203	0.965203	0.000012
23	0.639805	0.552680	2.035197	1.002465	0.984288	0.992078	0.992078	0.992078	0.352299	0.674630	1.598971	0.965274	0.979143	0.966508	0.966508	0.966508	0.000012
24	0.626795	0.544362	2.133409	1.003697	0.987985	0.994719	0.994719	0.994719	0.344536	0.674143	1.595016	0.971396	0.984669	0.973133	0.973133	0.973133	0.000012

HSV Experiment Summary

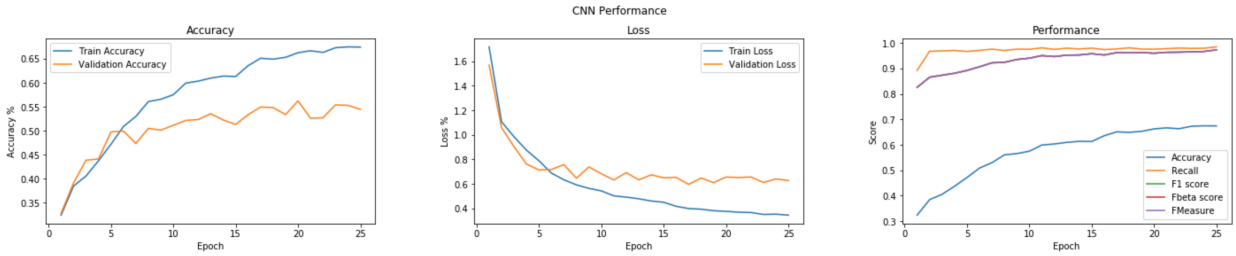
	val_loss	val_acc	val_mean_pred	val_precision	val_recall	val_f1_score	val_fbeta_score	val_fmeasure	loss	acc	mean_pred	precision	recall	f1_score	fbeta_score	fmeasure	lr
0	2.025562	0.311808	1.443437	1.004920	0.998155	1.001054	1.001054	1.001054	2.227216	0.223522	1.367139	110462.079435	0.935077	0.795702	0.795702	0.795702	0.000050
1	1.671456	0.246765	1.780077	1.000000	1.000000	1.000000	1.000000	1.000000	1.888322	0.226069	1.568655	0.709843	0.988623	0.805365	0.805365	0.805365	0.000050
2	2.011976	0.302583	1.527529	1.008610	0.998155	1.002636	1.002636	1.002636	1.835074	0.216967	1.586575	0.707351	0.991386	0.805467	0.805467	0.805467	0.000050
3	1.709967	0.262477	1.871535	1.000000	0.999076	0.999472	0.999472	0.999472	1.795322	0.221355	1.580556	0.708218	0.993012	0.805894	0.805894	0.805894	0.000050
4	1.673259	0.275830	1.785886	1.004920	0.996310	1.000000	1.000000	1.000000	1.779110	0.220543	1.585560	0.709356	0.985427	0.804647	0.804647	0.804647	0.000050
5	1.645097	0.235675	1.954929	1.003697	0.995379	0.998944	0.998944	0.998944	1.729355	0.229157	1.591756	0.710494	0.992145	0.806766	0.806766	0.806766	0.000050
6	1.753775	0.289668	1.738591	1.012915	0.981550	0.994553	0.994553	0.994553	1.722996	0.230457	1.588989	0.714340	0.992361	0.809590	0.809590	0.809590	0.000050
7	1.710910	0.306839	1.726693	1.005545	0.979667	0.990494	0.990494	0.990494	1.705343	0.236795	1.585413	0.716832	0.989653	0.808951	0.808951	0.808951	0.000050
8	1.586791	0.270295	1.895500	1.006150	0.995387	1.000000	1.000000	1.000000	1.695542	0.245896	1.597270	0.724037	0.991332	0.814390	0.814390	0.814390	0.000050
9	1.619427	0.258780	1.848967	1.004929	0.990758	0.996831	0.996831	0.996831	1.661640	0.247034	1.583925	0.724579	0.991061	0.815853	0.815853	0.815853	0.000050
10	1.598138	0.233826	2.088531	1.002465	0.994455	0.997888	0.997888	0.997888	1.644848	0.251097	1.592815	0.731621	0.989057	0.820024	0.820024	0.820024	0.000050
11	1.660644	0.265683	1.716602	1.010455	0.994465	1.001054	1.001054	1.001054	1.617970	0.259223	1.595446	0.737472	0.986565	0.823612	0.823612	0.823612	0.000050
12	1.636887	0.255556	1.746026	1.003704	0.990741	0.996296	0.996296	0.996296	1.608251	0.266537	1.593334	0.741535	0.991495	0.826228	0.826228	0.826228	0.000050
13	1.636313	0.250462	1.887067	1.004929	0.994455	0.998944	0.998944	0.998944	1.571389	0.274013	1.584001	0.745165	0.989869	0.828783	0.828783	0.828783	0.000025
14	1.607211	0.254630	1.782847	1.007407	0.987963	0.996296	0.996296	0.996296	1.558898	0.284577	1.600707	0.752045	0.989490	0.833464	0.833464	0.833464	0.000025
15	1.453370	0.257856	1.962630	1.004929	0.990758	0.996655	0.996655	0.996655	1.546693	0.287990	1.579894	0.752966	0.982610	0.832412	0.832412	0.832412	0.000025
16	1.549361	0.225000	1.857789	1.010494	0.990741	0.998942	0.998942	0.998942	1.553521	0.288965	1.599509	0.754754	0.984723	0.834312	0.834312	0.834312	0.000025
17	1.514689	0.256007	2.035100	1.004929	0.992606	0.997888	0.997888	0.997888	1.541215	0.288640	1.586794	0.761038	0.987702	0.839282	0.839282	0.839282	0.000025
18	1.503844	0.257407	1.951269	1.003704	0.989815	0.995767	0.995767	0.995767	1.515804	0.295953	1.598641	0.759900	0.985915	0.838066	0.838066	0.838066	0.000025
19	1.638342	0.270370	1.862616	1.010494	0.985185	0.995767	0.995767	0.995767	1.524645	0.292865	1.597468	0.759575	0.983964	0.837287	0.837287	0.837287	0.000025
20	1.562232	0.254159	1.870637	1.002465	0.986137	0.992958	0.992958	0.992958	1.506461	0.307655	1.586976	0.766618	0.982610	0.841242	0.841242	0.841242	0.000012
21	1.591636	0.255083	1.895212	1.009242	0.980591	0.992606	0.992606	0.992606	1.487850	0.309768	1.596149	0.768785	0.986294	0.844514	0.844514	0.844514	0.000012
22	1.623915	0.261553	1.908639	1.004929	0.973198	0.986797	0.986797	0.986797	1.475960	0.315293	1.590656	0.772198	0.988732	0.847617	0.847617	0.847617	0.000012
23	1.564515	0.241220	1.858030	1.003697	0.979667	0.989966	0.989966	0.989966	1.485171	0.307492	1.599886	0.769597	0.982989	0.842991	0.842991	0.842991	0.000012
24	1.611077	0.243993	1.904987	1.011091	0.984288	0.995247	0.995247	0.995247	1.468452	0.320007	1.593715	0.772631	0.988623	0.848111	0.848111	0.848111	0.000006

LAB Experiment Summary

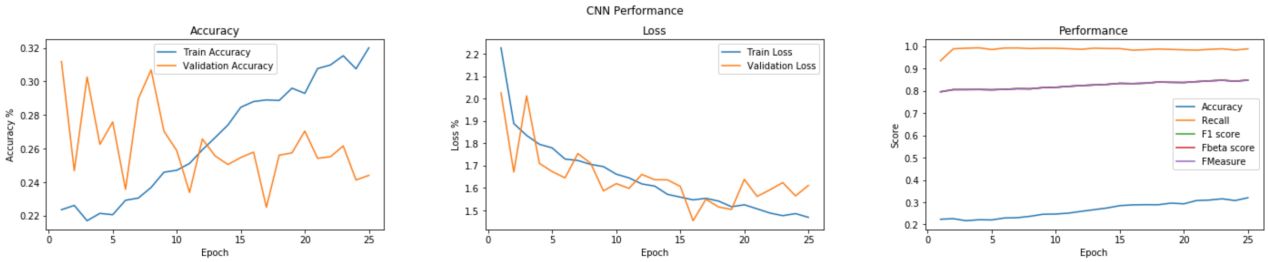
Figure 88. Summary report of experiments



HSL Experiment Plots



HSV Experiment Plots



Lab Experiment Plots

Figure 89. Graphical summaries of the experiments

```
Optimized Thresholds:
[0.53891451 1.79040709 2.24831516 2.98099956]

The Training Cohen Kappa Score is: 0.90341
The Validation Cohen Kappa Score is: 0.70918
The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.71796

This is an improvement of 0.00878
over the unoptimized rounding
```

HSL Kappa Results

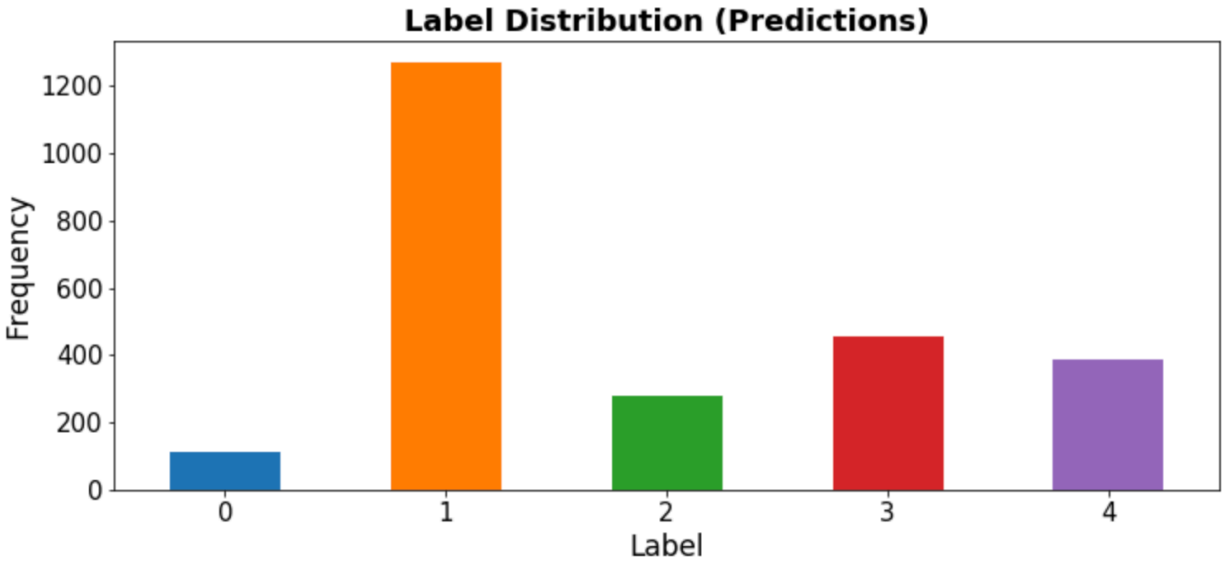
```
Optimized Thresholds:
[0.48764457 1.50449718 2.65621859 3.38538001]

The Training Cohen Kappa Score is: 0.9085
The Validation Cohen Kappa Score is: 0.71135
The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.7331

This is an improvement of 0.02176
over the unoptimized rounding
```

HSV Kappa Results

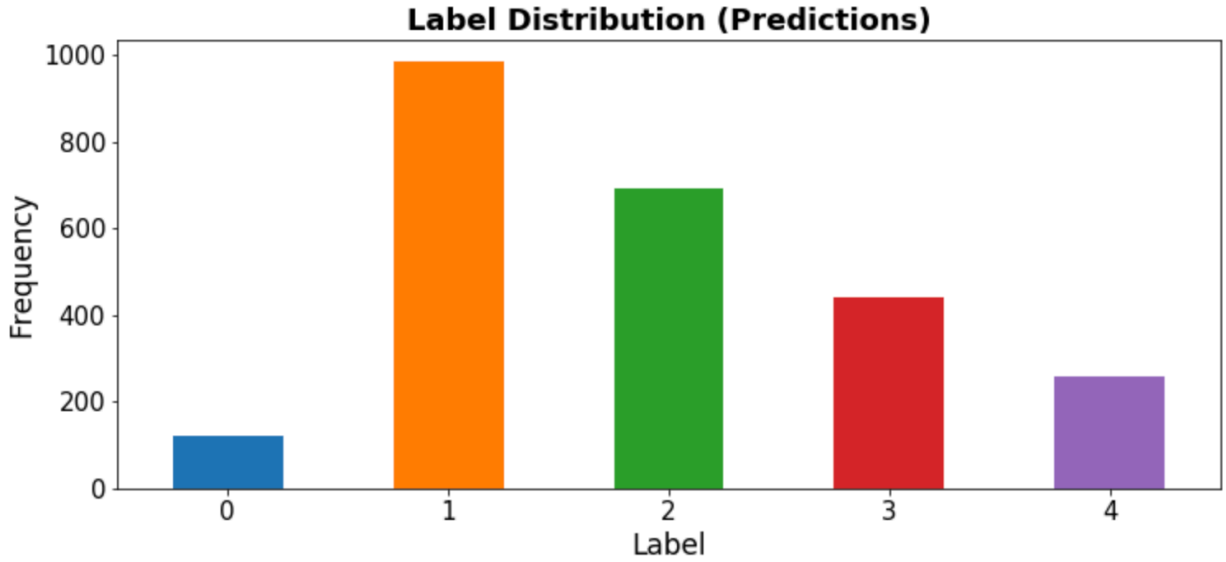
```
Optimized Thresholds:
[0.5058568  1.52260668 2.65610393 3.19573754]

The Training Cohen Kappa Score is: 0.24904
The Validation Cohen Kappa Score is: 0.11289
The Validation Quadratic Weighted Kappa (QWK)
with optimized rounding thresholds is: 0.12202

This is an improvement of 0.00913
over the unoptimized rounding
```

LAB Kappa Results

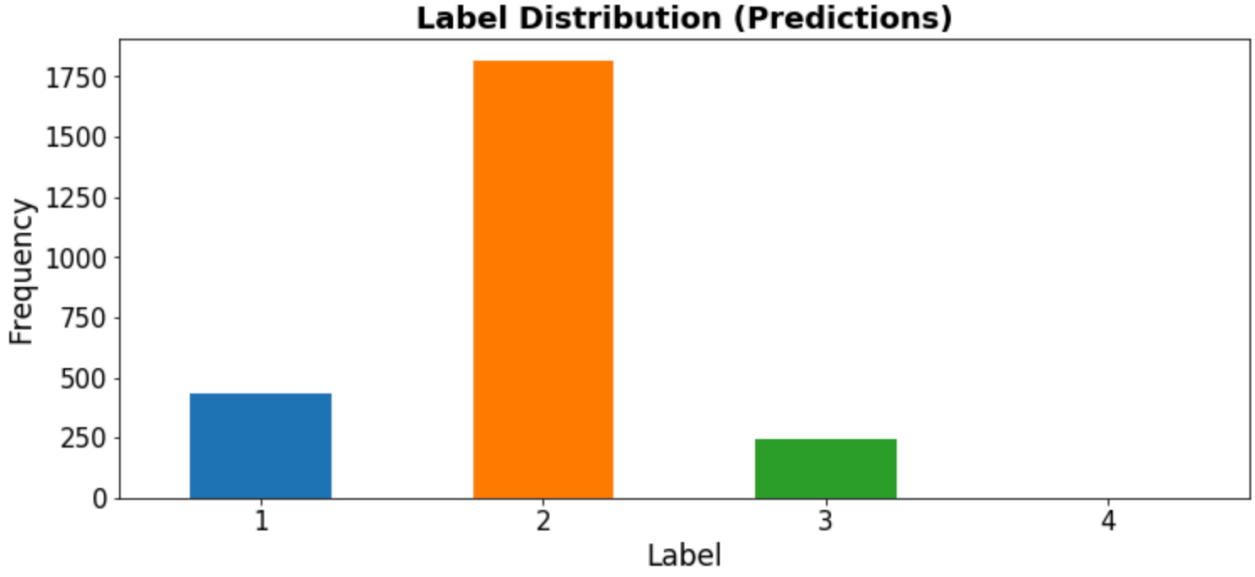
Figure 90. Kappa results of experiments



HSL Experiment Summary



HSV Experiment Summary



LAB Experiment Summary

Figure 91. Experiment Predictions Distributions

8. Conclusion and Result Evaluation

According to information shared in various resources about diabetic retinopathy detection, it is realized that many researchers did not use a lot of preprocessing methods and they have only used just basic methods such as crop from the gray area and resizing. The main reason to execute these experiments is to find out that is it possible to have good results also with more complex preprocessing methods. The main problem to resolve in this thesis was to investigate the influences of the color space methods on deep learning methods. Some researchers considered color spacing methods but they did not use modern architectures so testing these methods with deep learning algorithms was the main part of the thesis.

During the experiments, it is decided to use EfficientNet B1 to B5 and ResNet50 architectures due to their successful performance in the APTOS 2019 competition and other research. Various types of experiments are done with the deep learning architectures mentioned earlier to have an idea of the purpose of image classification with desired preprocessing methods. On the thresholding process of the images, it is seen that the color differences on images in APTOS 2019 dataset trained models badly and it is a very difficult challenge to assign private threshold values to every image in the dataset. Another problem was an imbalanced data problem in APTOS 2019 dataset. With regards to these two main problems, a custom dataset that consists of both APTOS 2019 and 2015 preprocessed to grey images and a balanced dataset is used for most of the experiments held in this thesis. For the initial experiments, a comparison of both datasets is done without applying any preprocessing method, and the performances of the models obtain similar results for this experiment.

After having experiments with plain datasets, training with preprocessed images by HSV color space with different threshold values, and deep learning architectures, it is seen that the method is not good enough for the purpose of retinal image classification to detect diabetic retinopathy.

Experiments at the end of the research with HSV, HSL, and Lab filters without thresholds show some hopeful results represented in the earlier sections.

At the end of the research held in this thesis, it is possible to make conclusions as follows below,

1. EfficientNet architectures mostly performed better than ResNet50 architecture to make predictions.
2. Input size does not have a big influence on the training performances for the images that are preprocessed in the desired way by thresholding the training images by color spaces.
3. Values of the thresholds have essential effects on the trainings, it is observed that balancing the color of the images is a very big challenge to resolve.
4. The results obtained by preprocessed images with HSV color spacing by thresholding are not trustable for the purpose of retinal image classification with deep learning methods.
5. Cylindric color space filters without thresholds like HSV and HSL performed significantly better than Lab chromatic color space. While HSV and HSL filters have validation kappa scores over 0.70, the Lab filter failed with a 0.11 kappa score. Additionally, F1 scores and other accuracy metrics show similar impacts to kappa metrics.
6. It is believed that more improvements can be done with HSV and HSL color spaces to improve the classification performances with deep learning methods in the research will be held in the same purpose of this thesis.

References

1 - Chelvin Sng Eye Center. 2022. *Diabetic Retinopathy: Causes & Treatments | Chelvin Sng Eye Center*. [online] Available at: <https://www.drchelvinsng.com/diabetic-retinopathy/ >

2- Ben-Hur, Asa, and Jason Weston. "A user's guide to support vector machines." *Data mining techniques for the life sciences*. Humana Press, 2010. 223-239.

3- Adarsh, P. and D. Jeyakumari, 2013. Multiclass SVMbased automated diagnosis of diabetic retinopathy. Proceedings of the International Conference on Communication and Signal Processing, Apr. 3-5, IEEE Xplore Press, Melmaruvathur, India, pp: 206-210. DOI: 10.1109/ iccsp.2013.6577044

4- J. De Calleja, L. Tecuapetla, and M. A. Medina, "LBP and Machine Learning for Diabetic Retinopathy Detection," pp. 110–117, 2014.

5- Jothi, A. and S. Jayaram, 2019. Blood Vessel Detection in Fundus Images Using Frangi Filter Technique. 1st Edn., Smart Innovations in Communication and Computational Sciences. Springer, Singapore, ISBN-13: 978-981-10-8970-1, pp: 49-57.

6- DIARETDB1 is Copyright © 2007 by Tomi Kauppi, Valentina Kalesnykiene, Joni-Kristian Kamarainen, Lasse Lensu, Iiris Sorri, Asta Raninen, Raija Voutilainen, Juhani Pietilä, Heikki Kälviäinen, and Hannu Uusitalo.
https://www.it.lut.fi/project/imageret/diaretdb1

7 - A. Hoover, V. Kouznetsova and M. Goldbaum, "Locating Blood Vessels in Retinal Images by Piece-wise Threhsold Probing of a Matched Filter Response", *IEEE Transactions on Medical Imaging* , vol. 19 no. 3, pp. 203-210, March 2000.

8- T. Karim, M. S. Riad and R. Kabir, "Symptom Analysis of Diabetic Retinopathy by Micro Aneurysm Detection Using NPRTOOL," 2019 International Conference on Robotics,Electrical and Signal Processing Techniques (ICREST), 2019, pp. 606-610, doi: 10.1109/ICREST.2019.8644439.

9- I. Sadek, M. Elawady, and A. E. R. Shabayek, "Automatic Classification of Bright Retinal Lesions via Deep Network Features," pp. 1–20, 2017.

10- He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. 52

11.- Richmond Alake, Deep Learning: GoogLeNet Explained. (2020) *https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765*

12.- Srikanth Tammina (2019); Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images; International Journal of Scientific and Research Publications (IJSRP) 9(10) (ISSN: 2250-3153), DOI: http://dx.doi.org/10.29322/IJSRP.9.10.2019.p9420

13- Decencière et al.. Feedback on a publicly distributed database: the Messidor database: the Messidor database. Image Analysis & Stereology, v. 33, n. 3, p. 231-234, aug. 2014. ISSN 1854-5165.

14- Xu, K., D. Feng and H. Mi, 2017. Deep convolutional neural network-based early automated detection of diabetic retinopathy using fundus image. Molecules, 22: 2054-2054. DOI: 10.3390/ molecules22122054

15- D. Doshi, A. Shenoy, D. Sidhpura, and P. Gharpure, "Diabetic retinopathy detection using deep convolutional neural networks," in International Conference on Computing, Analytics and Security Trends, CAST 2016, 2017, pp. 261–266, doi: 10.1109/CAST.2016.7914977

16- Maya, K.V. and K.S. Adarsh, 2019. Detection of retinal lesions based on deep learning for diabetic retinopathy. Proceedings of the 5th International Conference on Electrical Energy Systems, Feb. 21-22, IEEE Xplore Press, Chennai, India. DOI: 10.1109/ICEES.2019.8719242

17- G. T. Zago, R. V. Andreão, B. Dorizzi, and E. O. Teatini Salles, "Diabetic retinopathy detection using red lesion localization and convolutional neural networks," Comput. Biol. Med., vol. 116, p. 103537, 2020, doi: 10.1016/j.compbiomed.2019.103537.

18- Hagos, M. T. and Kant, S. (2019). Transfer learning based detection of diabetic retinopathy from small dataset. CoRR, abs/1905.07203

19 - Rubina Sarki, Sandra Michalska. Convolutional neural networks for mild diabetic retinopathy detection: an experimental study. (2019)

20- Chollet, François. " Xception: Deep learning with depthwise separable convolutions." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

21.- Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

22- Xu, G. (2019b). *APTOS 2019 Blindness Detection | Kaggle*. Blindness Detection. https://www.kaggle.com/c/aptos2019-blindness-detection/discussion/108065

23 - E.P.L.B. [ods.ai]. (2019). *APTOS 2019 Blindness Detection | Kaggle*. Blindness Detection.
https://www.kaggle.com/c/aptos2019-blindness-detection/discussion/107926

24- B.O.F. (2019). *APTOS 2019 Blindness Detection | Kaggle*. Blindness Detection. https://www.kaggle.com/c/aptos2019-blindness-detection/discussion/107987

25- Khvedchenya, E. (2019). *APTOS 2019 Blindness Detection | Kaggle*. Blindness Detection.
https://www.kaggle.com/competitions/aptos2019-blindness-detection/discussion/108058

26 - E.S. (2019). *APTOS 2019 Blindness Detection | Kaggle*. Blindness Detection. https://www.kaggle.com/competitions/aptos2019-blindness-detection/discussion/108030

27- Wikipedia contributors. (2022, April 11). *Gaussian filter*. Wikipedia. https://en.wikipedia.org/wiki/Gaussian_filter# %7E:text=In%20electronics% 20and %20signal%20processing,would%20have%20infinite%20impulse%20response).

28- Tan, Mingxing, and Quoc Le. "Efficientnet: Rethinking model scaling for convolutional neural networks." *International conference on machine learning*. PMLR, 2019.

29- He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

30- Wang, Jing, et al. "Multi-label classification of fundus images with efficientnet." *IEEE Access* 8 (2020): 212499-212508.

31- Decencière, Etienne, et al. "Feedback on a publicly distributed image database: the Messidor database." *Image Analysis & Stereology* 33.3 (2014): 231-234.

32- Henry, Felix, et al. "Sugarcane land classification with satellite imagery using logistic regression model." *IOP conference series: materials science and engineering*. Vol. 185. No. 1. IOP Publishing, 2017.

33- A. (2020, May 24). *APTOS 2015&2019 Preprocessed*. Kaggle. https://www.kaggle.com/datasets/alinusik/aptos-20152019-preprocessed

34- Liu, Liyuan, et al. "On the variance of the adaptive learning rate and beyond." *arXiv preprint arXiv:1908.03265* (2019).

35- E. E. Reber, R. L. Michell, and C. J. Carter.Kaggle diabetic retinopathy detection competition report.Technical report, Kaggle, 2015.