ANALYZING THE IMPACT OF OPTIMIZATION ALGORITHMS IN TRAINING NEURAL NETWORKS –*FOCUSED ON VGG16 GLAUCOMA DETECTION*

Amril Prasad

Abstract: Glaucoma is one of the leading causes of blindness or vision impairment according to World Health Organization. It requires lot of expertise and practice to detect glaucoma from the retinal color fundus images. To make it scalable and efficient, various machine Learning models have been proposed over the years. Our study is aimed at analyzing the impact of various optimization algorithms and learning rates on training a pre-trained convolutional neural network model for detecting glaucoma from retinal color fundus images. Six different optimizers were studied with seven different learning rates ranging from 0.0005 to 0.1. The Experimental results show that there is a significant impact in Loss and Accuracy of the model with different optimizers and at different learning rates.

1.Introduction:

1.1 Background:

Glaucoma is the group of eye condition that causes progressive damage to the optic nerve due to the increased pressure in the eye. Certain types of glaucoma are referred as "Sneak thief of Sight" because there are usually no symptoms of the disease except for the loss of vision and pain at a very advanced stage of the disease. So, early detection during regular screening process can help in preventing vision loss. The common screening procedures for detecting glaucoma include,

- Tonometry
- Visual Field Perimetry Tests
- Fundus and Optic nerve screening
- Gonioscopy
- Pachymetry
- Optical Coherence Tomography

Our study focusses on Color Fundus Images obtained from "Fundus and Optic Nerve Screening". Fundus is the interior surface of the eye which includes retina, optic disc, blood vessels, macula and fovea and posterior pole. A fundus camera helps in capturing the fundus images which gives the advantage of documenting the structure of the fundus over the period of time. Changes associated with glaucoma include optic nerve head cupping, neuro-retinal rim thinning, retinal nerve fiber layer defect and peripapillary atrophy [2]. Although the digital fundus imaging camera is widely available in primary health care center and extensively being used for large scale screening, the interpretation of color fundus images with specific to glaucomatous signs is still very challenging and requires years of practice and strong expertise. This hinders the process of quick detection of glaucoma with the huge number of populations screened.

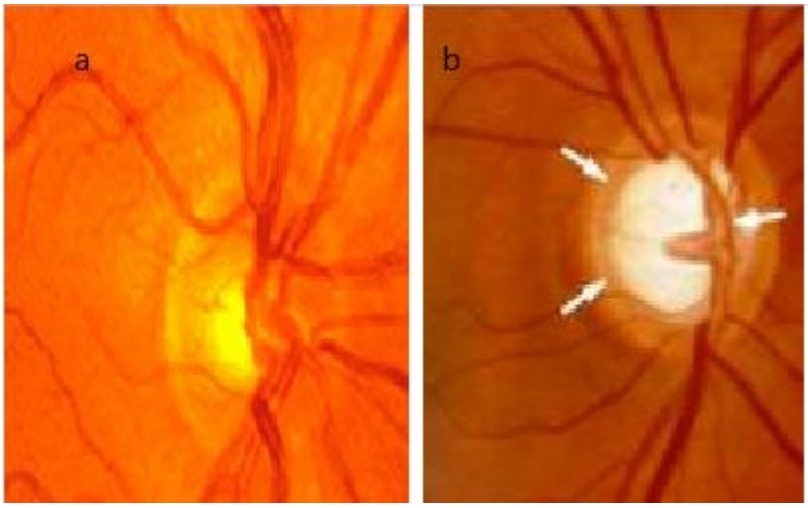

Fig 1.1 Example images featuring difference between a) Normal Optic Nerve b) Glaucoma affected

Artificial Intelligence has changed the way medical images are processed with its powerful machine learning algorithms. In [3] author has detailed about the way machine learning fits into medical imaging and diagnosis. Especially Deep Learning Networks have made a big leap in medical imaging in order to identify, classify and quantify patterns in medical imaging [4].The most successful type of deep learning algorithms for medical analysis is a sub class of neural networks called convolutional neural networks (CNN)[1].Our base Paper[1] compares the performance of various CNN models in classifying color fundus images as Glaucoma and Non-Glaucoma and found that VGG-19 model performs the best when combined with transfer learning. Our study is based on VGG-16 model which is less complex and equally efficient to the VGG1-19 model [5].

1.2 Dataset Summary:

Our dataset is obtained from GitHub, contains color fundus image with accompanying class labels as follows,

	Class 0 (Non-Glaucoma)	Class 1 (Glaucoma)
Train	255	200
Test	32	32

Table 1.1 Dataset distribution

The dataset is a binary class dataset with class balance ensured. The images are of varying sizes and will be handled as part of pre-processing data.

1.3 Hypothesis:

- Can the choice of Optimizer and Learning Rate can have significant impact on the performance of VGG16 CNN model?

The evolving research on various optimization algorithms and the development of new algorithms signifies the important of optimizers in deep learning architecture. Optimization involves fine tuning hyper parameters that gives the best fit with the smallest prediction error with observed data. Optimizers uses algorithms to change the parameters of the neural network such as weights and learning rates in order to reduce the losses by finding the global optimum. Our study aims at understanding the significance of different optimizers and its impact on VGG16 model performance in detecting glaucoma at different learning rates

2.Experimental Designs and Methods:

2.2 Data Pre-Processing:

Our dataset has images of varying sizes. To make it compatible with the pre-trained VGG16 model they have been rescaled to 224 x 224 pixel size. No other augmentation has been performed in order to capture the variations only with respect to optimizer and learning rates.

2.3 Method:

Using Python Libraries Keras, TensorFlow, a sequential model was created with a pre-trained VGG16 model and all the layers are frozen in order to enable transfer learning. It also helps in preventing the weights being modified resulting in reduced training time. In addition, a final Dense layer with SoftMax activation is added to form a fully connected layer whose output will be a probability in the range of 0 and 1, which in our case will indicate the class of our fundus images.

We chose six different optimizers Stochastic Gradient Descent (SGD), Adadelta, Adam, RMSprop, Adagrad and Nadam with seven different learning rates (0.0005, 0.0001, 0.005, 0.001, 0.05, 0.01, 0.1)

The batch size is set to 32 and Epoch is set to 25 for training the same model in Fig 2.1 with the above-mentioned optimizers and learning rates. The loss function used is “Binary Cross Entropy” since the model is a binary classifier. ‘Accuracy’ is the performance metric for evaluating the model.

A Random Normal initializer is used to initialise weights. Bias is initialised to zero. The seed is set to zero for all the models to get same sequence of numbers across trials. The Test Data is used as the validation data set to validate the models. Loss and accuracy against epochs are documented and plotted using ‘matplotlib’ library.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
fc1 (Dense)                  (None, 4096)              102764544
_________________________________________________________________
fc2 (Dense)                  (None, 4096)              16781312
_________________________________________________________________
dense (Dense)                (None, 2)                 8194
=================================================================
Total params: 134,268,738
Trainable params: 8,194
Non-trainable params: 134,260,544
_________________________________________________________________
```

Fig 2.1 VGG-16 model under study with customized final layer (Base Model)

The training time of the models were considerably less since the layers of the pre-trained model are frozen. Both the epochs and batch size are also chosen reasonably with respect to the sample size.

A total of 42 models were trained and validated as specified above.

2.4 Optimizers:

Stochastic Gradient Descent: Stochastic gradient descent is a variation to the gradient descent algorithm in which the updates are made to the coefficient after each training instance, instead of updating at the end of batch of instances. The learning will be much faster and the randomness of training data will be ensured.

Adagrad: It is one of the gradient descent algorithms which adapts learning rate and is highly suitable for sparse data since it considers low learning rates for frequently occurring parameters and high learning rates for infrequent parameters. It takes a subset of training data and update the weight by computing the gradient and squared error.

RMS Prop: RMS Prop computes the learning rate with an exponential average of squared gradients. Similar to Adagrad it takes a subset of training data and compute the gradient and squared error along with decay rate and then update the weights.

Adam: It is one of the most efficient algorithms which computes learning rate for each parameter. It takes into account both the exponentially decaying average of gradients and squared gradients which is the first moment and second moment. Both gradient and squared gradient are computed and biased towards zero and weights are updated by bias corrected gradients and squared gradients. Combination of momentum and RMS prop.

Adadelta: It is a stronger extension of Adagrad. It adapts learning rate based on moving window of gradient updates instead of considering all the past gradients. It enables continuous learning even after many updates.

Nadam: It incorporates Nesterov accelerated gradient with Adam. It is beneficial for noisy gradients and for gradients with high curvatures. With Nadam, the learning process is accelerated by summing up the exponential decay of moving averages for the previous and current gradient.

3. Results:

The results of training our model with six different optimizers and 7 different learning rates each are summarized in the Table 3.1. The loss graphs of each model is documented in section 3.1. The VGG-16 model with the “Adam” optimizer and Learning Rate 0.005 gives the lowest loss value 0.21 for training and 0.12 for validation in turn resulting in the training accuracy of 0.95 and validation accuracy of 0.97.

Optimizer	Learning Rate	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
SGD	0.0005	0.7	0.85	0.72	0.8
SGD	0.0001	0.58	1.43	0.57	1.57
SGD	0.005	0.81	0.64	0.79	0.71
SGD	0.001	0.75	0.68	0.77	0.62
SGD	0.05	0.43	8.66	0.5	7.67
SGD	0.01	0.79	1.33	0.77	1.61
SGD	0.1	0.57	6.62	0.5	7.67
Adadelta	0.0005	0.48	2	0.47	2.28
Adadelta	0.0001	0.48	2.01	0.47	2.29
Adadelta	0.005	0.51	1.8	0.5	2.05
Adadelta	0.001	0.49	1.97	0.47	2.25
Adadelta	0.05	0.66	1	0.68	0.99
Adadelta	0.01	0.54	1.61	0.53	1.81
Adadelta	0.1	0.73	0.75	0.74	0.69
Adam	0.0005	0.87	0.36	0.88	0.31
Adam	0.0001	0.72	0.78	0.74	0.72
Adam	0.005	0.95	0.21	0.97	0.12
Adam	0.001	0.91	0.28	0.91	0.26
Adam	0.05	0.72	4.22	0.75	3.81
Adam	0.01	0.79	2.88	0.78	2.9
Adam	0.1	0.44	8.58	0.5	7.67
RMSprop	0.0005	0.85	0.39	0.86	0.33
RMSprop	0.0001	0.73	0.75	0.75	0.68
RMSprop	0.005	0.44	8.58	0.5	7.67
RMSprop	0.001	0.87	0.36	0.88	0.31
RMSprop	0.05	0.56	6.73	0.5	7.67
RMSprop	0.01	0.83	0.83	2.02	2.02
RMSprop	0.1	0.56	6.74	0.5	7.67
Adagrad	0.0005	0.67	0.96	0.69	0.91
Adagrad	0.0001	0.53	1.65	0.52	1.83
Adagrad	0.005	0.85	0.43	0.85	0.38
Adagrad	0.001	0.74	0.7	0.76	0.61
Adagrad	0.05	0.85	1.48	0.84	1.59
Adagrad	0.01	0.88	0.39	0.89	0.34
Adagrad	0.1	0.44	8.55	0.5	7.67
Nadam	0.0005	0.86	0.38	0.87	0.34
Nadam	0.0001	0.71	0.83	0.72	0.76
Nadam	0.005	0.92	0.37	0.94	0.24
Nadam	0.001	0.91	0.27	0.91	0.24
Nadam	0.05	0.45	8.42	0.5	7.63
Nadam	0.01	0.84	1.83	0.85	1.69
Nadam	0.1	0.44	8.59	0.5	7.67

Table 3.1 Loss and Accuracy of models across each optimizer and learning rate

3.1 Graphs:

3.1.1 Loss Function Graphs of models with SGD optimizer at different learning rates

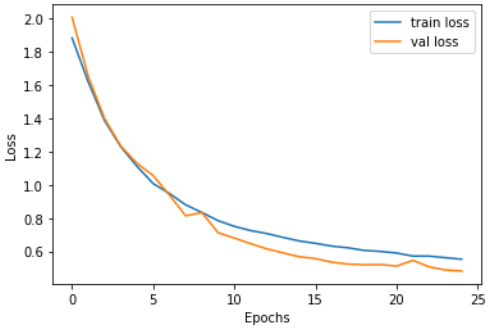

Fig 3.1.1.1 LR 0.0005

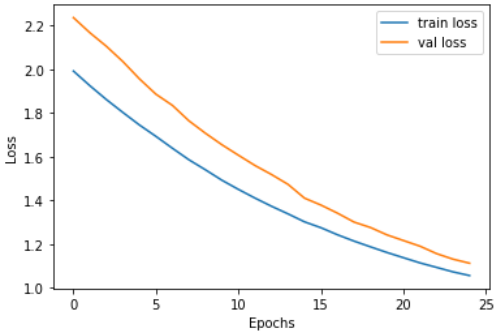

Fig 3.1.1.2 LR 0.0001

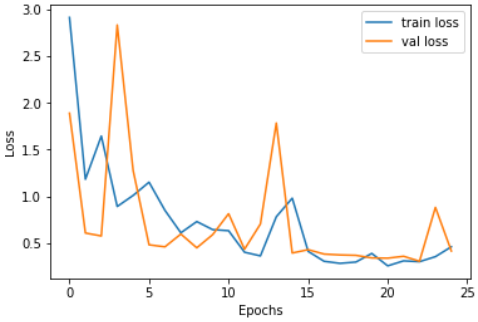

Fig 3.1.1.3 LR 0.005

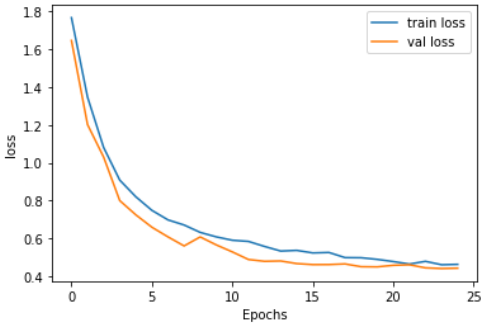

Fig 3.1.1.4 LR 0.001

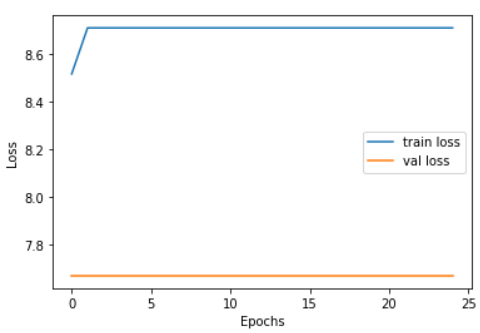

Fig 3.1.1.5 LR 0.05

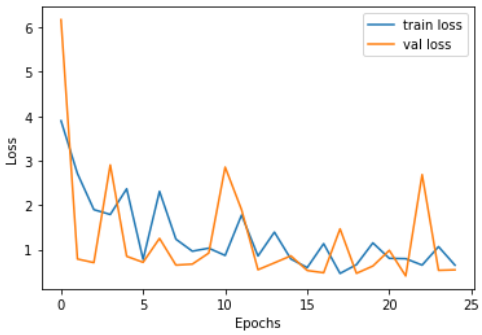

Fig 3.1.1.6 LR 0.01

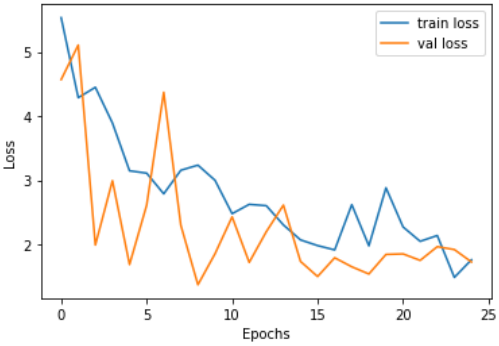

Fig 3.1.1.7 LR 0.1

3.1.2 Loss Function Graphs of models with Adadelta optimizer at different learning rates

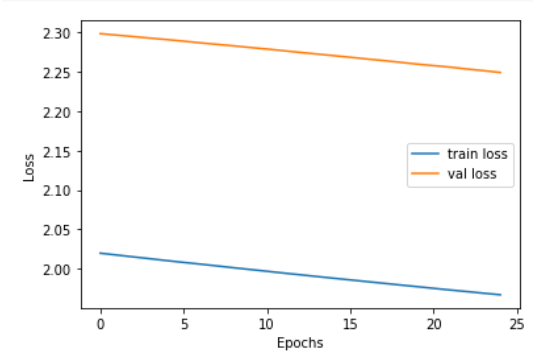

Fig 3.1.2.1 LR 0.0005

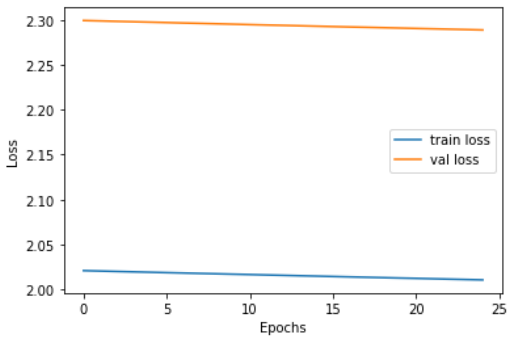

Fig 3.1.2.2 LR 0.0001

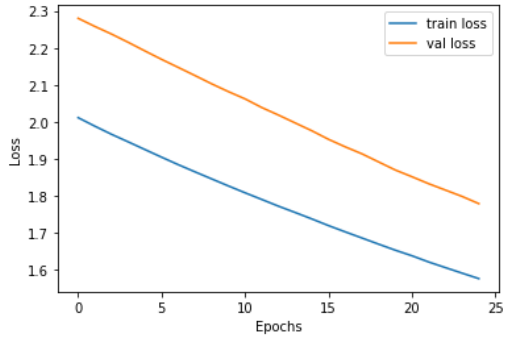

Fig 3.1.2.3 LR 0.005

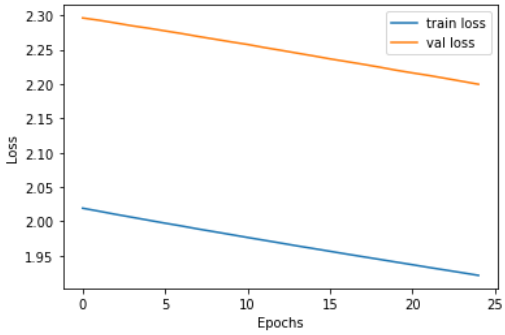

Fig 3.1.2.4 LR 0.001

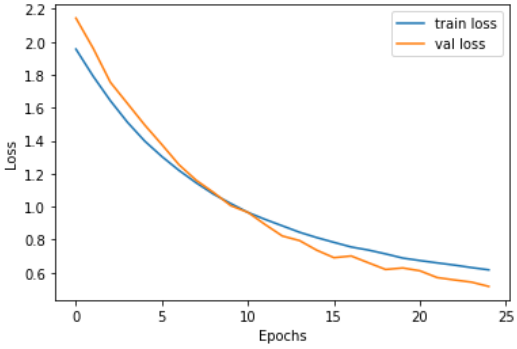

Fig 3.1.2.5 LR 0.05

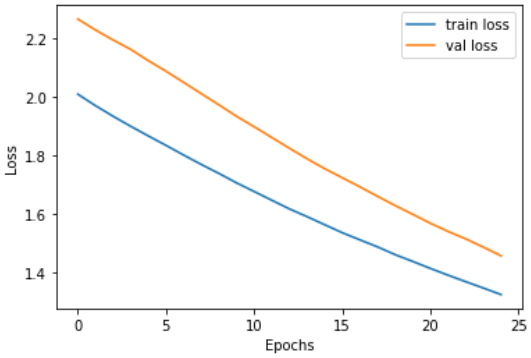

Fig 3.1.2.6 LR 0.01

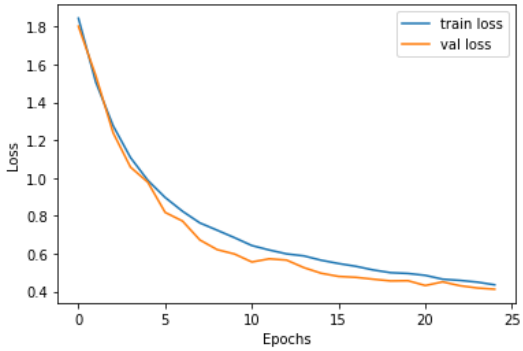

Fig 3.1.2.7 LR 0.1

3.1.3 Loss Function Graphs of models with Adam optimizer at different learning rates

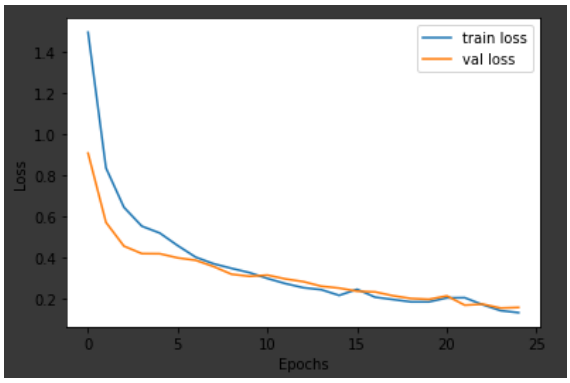

Fig 3.1.3.1 LR 0.0005

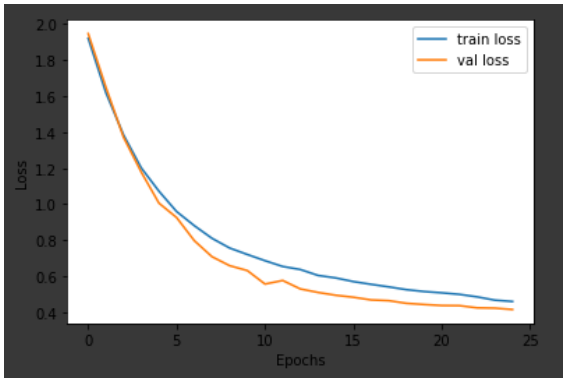

Fig 3.1.3.2 LR 0.0001

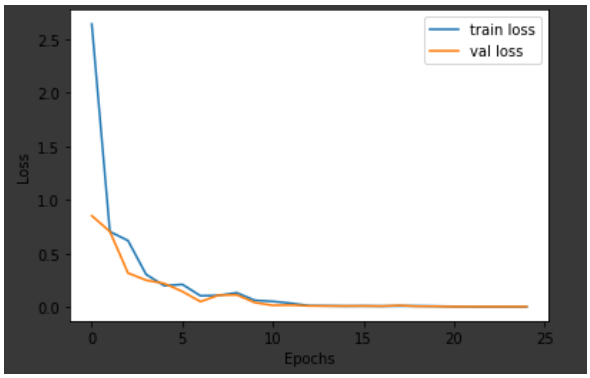

Fig 3.1.3.3 LR 0.005

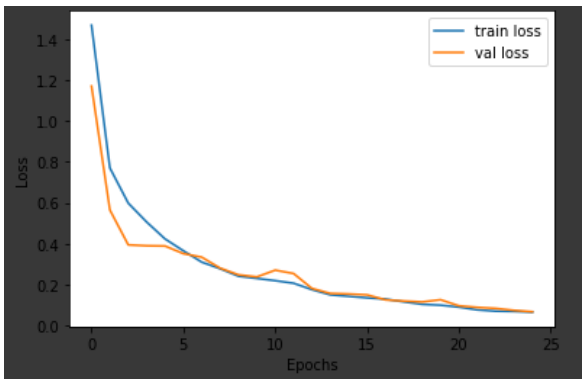

Fig 3.1.3.4 LR 0.001

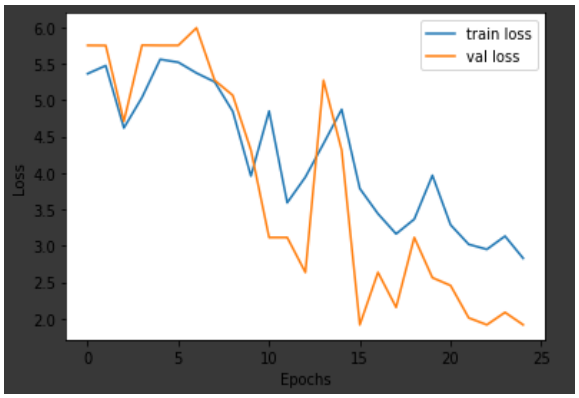

Fig 3.1.3.5 LR 0.05

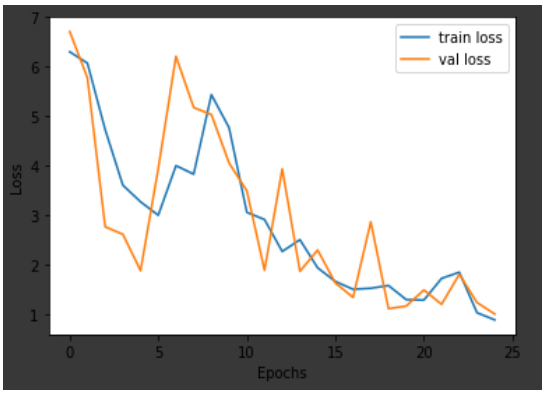

Fig 3.1.3.6 LR 0.01

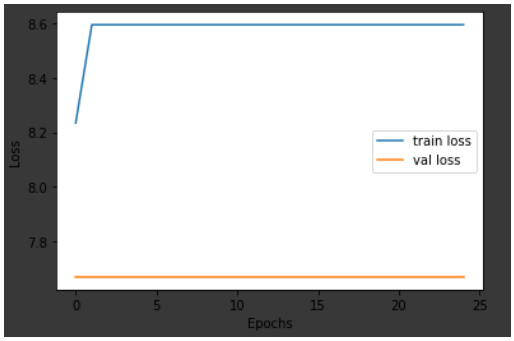

Fig 3.1.3.7 LR 0.1

3.1.4 Loss Function Graphs of models with RMSprop optimizer at different learning rates

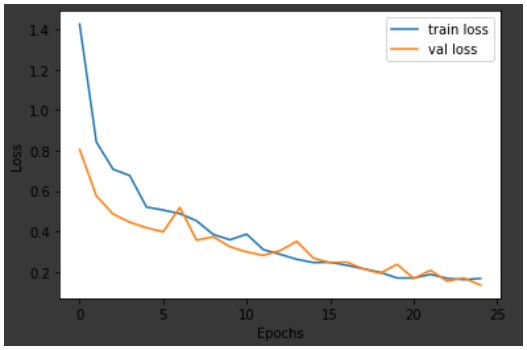

Fig 3.1.4.1 LR 0.0005

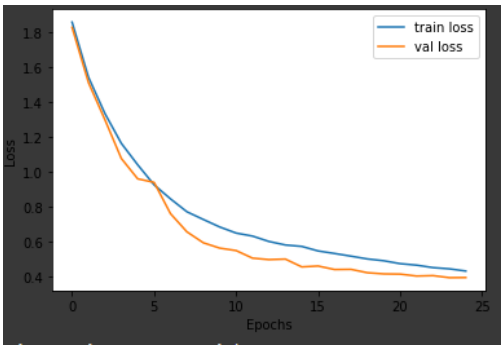

Fig 3.1.4.2 LR 0.0001

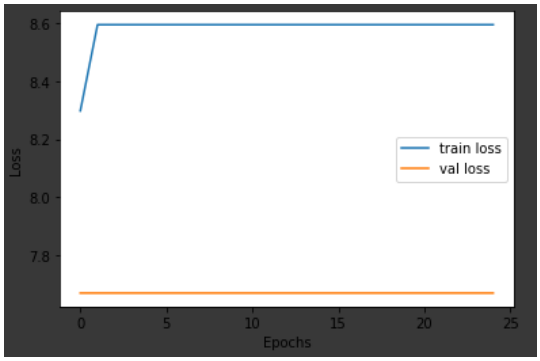

Fig 3.1.4.3 LR 0.005

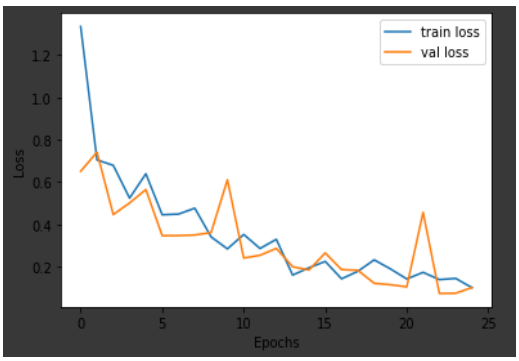

Fig 3.1.4.4 LR 0.001

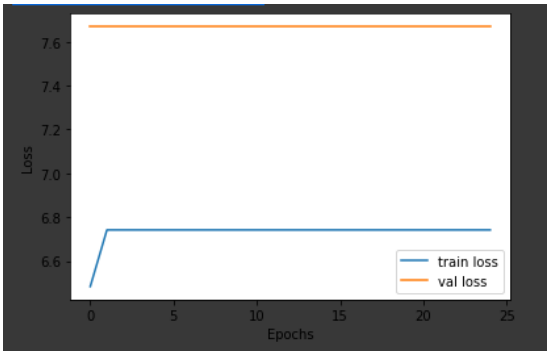

Fig 3.1.4.5 LR 0.05

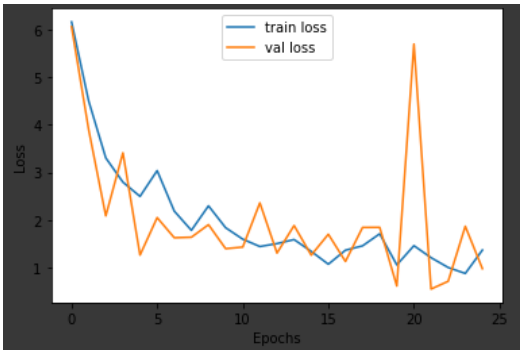

Fig 3.1.4.6 LR 0.01

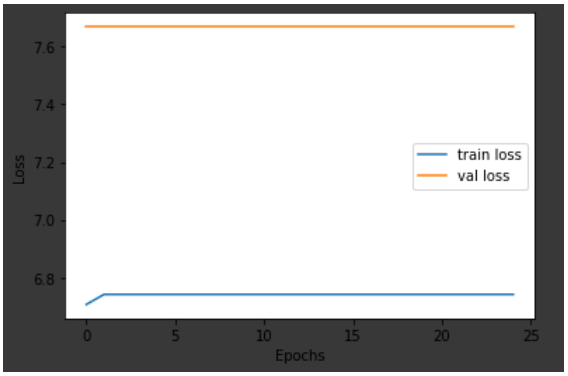

Fig 3.1.4.7 LR 0.1

3.1.5 Loss Function Graphs of models with Adagrad optimizer at different learning rates

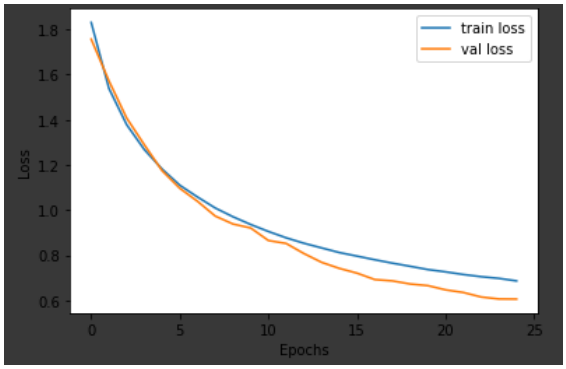

Fig 3.1.5.1 LR 0.0005

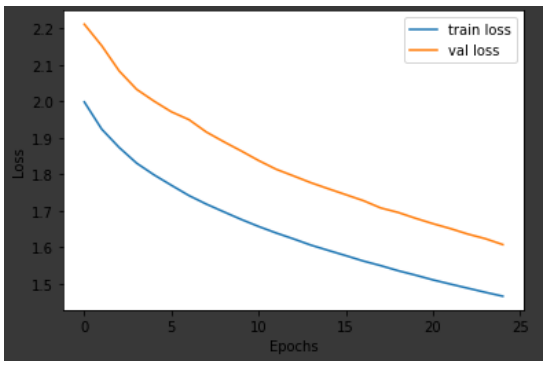

Fig 3.1.5.2 LR 0.0001

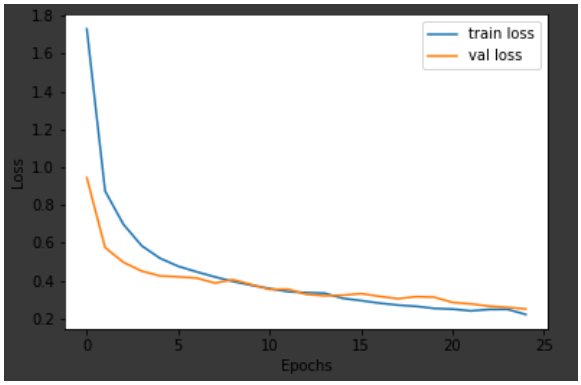

Fig 3.1.5.3 LR 0.005

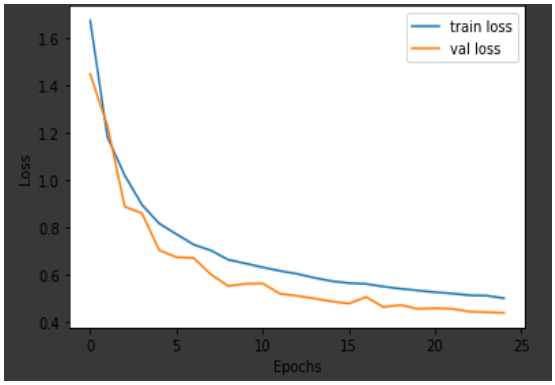

Fig 3.1.5.4 LR 0.001

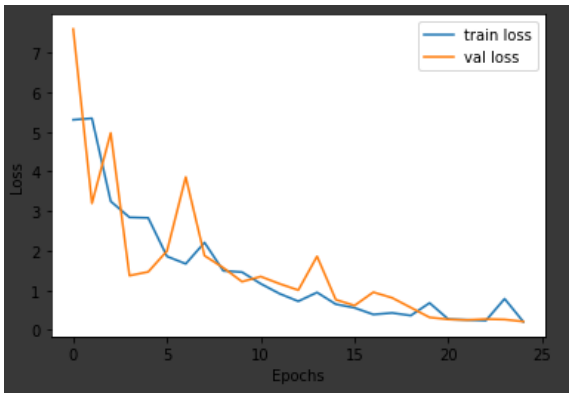

Fig 3.1.5.5 LR 0.05

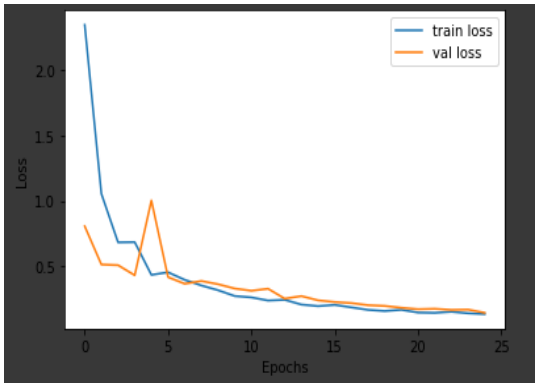

Fig 3.1.5.6 LR 0.01

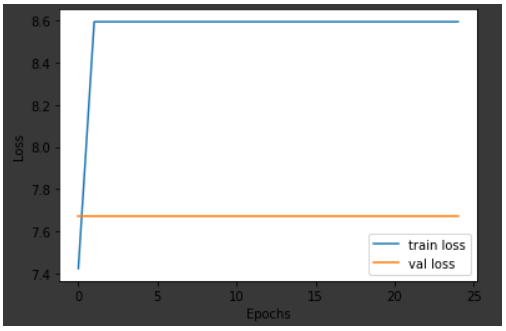

Fig 3.1.5.7 LR 0.1

3.1.6 Loss Function Graphs of models with Nadam optimizer at different learning rates

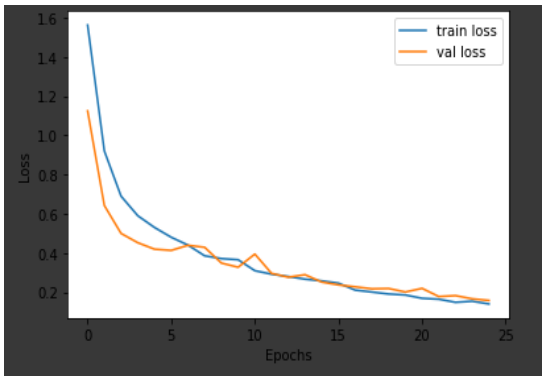

Fig 3.1.6.1 LR 0.0005

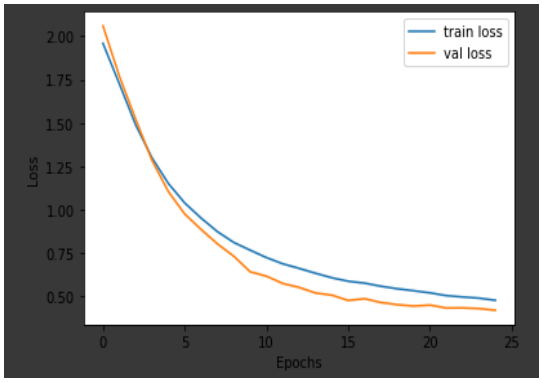

Fig 3.1.6.2 LR 0.0001

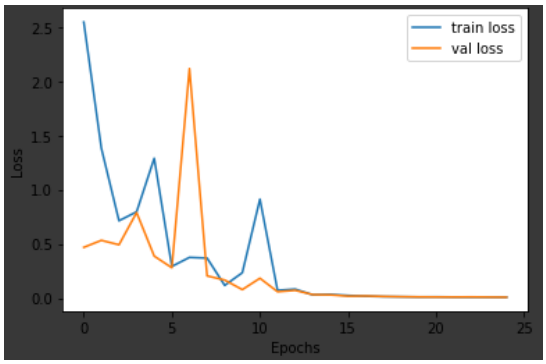

Fig 3.1.6.3 LR 0.005

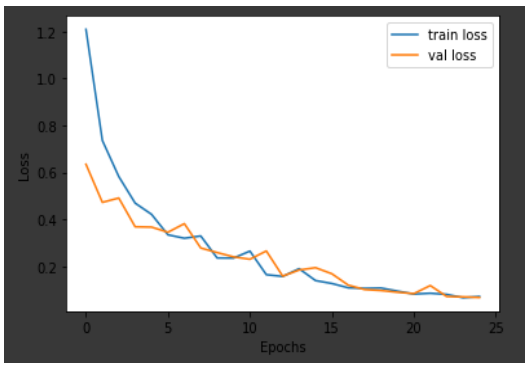

Fig 3.1.6.4 LR 0.001

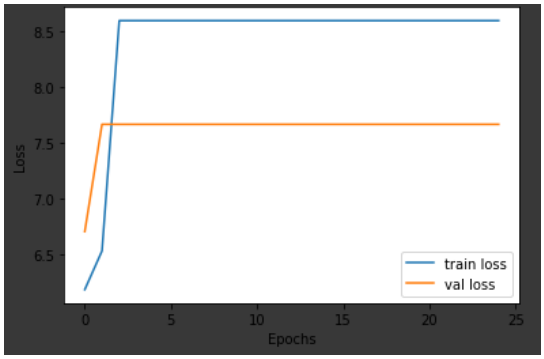

Fig 3.1.6.5 LR 0.05

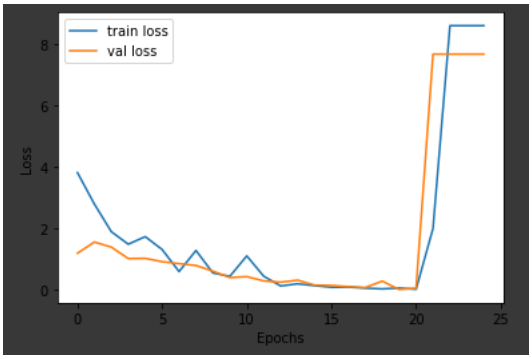

Fig 3.1.6.6 LR 0.01

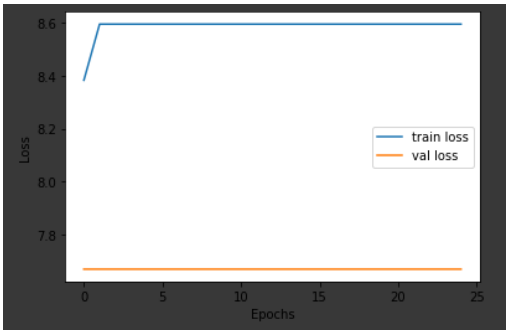

Fig 3.1.6.7 LR 0.1

The models with SGD optimizer performed decently at lower learning rates with the loss values below 1.5. The model performed best at 0.005 learning rate yielding 81% accuracy for both training and validation set. Even the loss was almost equal in both cases. This was a good sign because the model was not overfitting or underfitting the data. But as the learning rate is increased, the loss values Increased up to 8 which is pretty worse.

The models with Adadelta optimizer performed reasonably well at all learning rates with the loss values less than 2. The best accuracy was at learning rate 0.1.

As we saw earlier Adam was the best performing optimizer but at 0.1 the loss value was so high. The model fitness is also inline between training and validation at all rates except 0.1

The RMS prop models were not consistent and the performance fluctuated inconsistently between the learning rates.
The models with Adagrad optimizer had a high loss value at a learning rate of 0.1 resulting in only 44% accuracy at training and 50% at validation. At most of the learning rates the training results and validation results were same indicating the good fit of the model.

The Nadam optimizer model performed next best to Adam optimizer with 92% and 94% in training and validation respectively. But the model behaved poorly with increasing learning rates with the loss value up to 8.59

All loss graphs show a decreasing trend at the end of epochs signifying that the number of epochs was sufficient to train the model. One exception to this Nadam at 0.01 which shows an increase in loss value post 20 epochs.(Fig 3.1.6.6)

The graph below (Fig 3.2) shows the accuracy of the best performing model Adam at 0.005. It can be seen that the model reached its peak accuracy at almost 10 epochs. The curve flattened before 15 epochs.

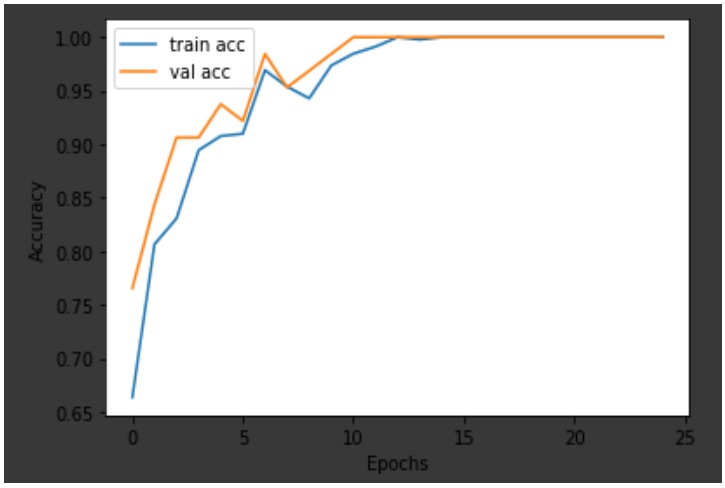

Fig3.2: Accuracy of model with Nadam optimizer and learning rate 0.005

3.2 Conclusion:

From the above results it is clear that the choice of optimizer and the learning rate has significant impact on the performance of model. The learning rates below 0.005 allow model to train better when compared to higher learning rates like 0.1. Adam optimizer performs best for our pre-trained VGG16 model in classifying glaucoma images. Further research can be drilled down to increase the model efficiency by performing augmentations and cross-validations.

4.References:

[1] Gómez-Valverde, Juan J et al. "Automatic glaucoma classification using color fundus images based on convolutional neural networks and transfer learning." Biomedical optics express vol. 10,2 892-913. 25 Jan. 2019, doi:10.1364/BOE.10.000892

[2] Hagiwara, Yuki & Koh, Joel En Wei & Tan, Jen Hong & Bhandary, Sulatha & Laude, Augustinus & Ciaccio, Edward & Tong, Louis & Acharya, U Rajendra. (2018). Computer-Aided Diagnosis of Glaucoma Using Fundus Images: A Review. Computer methods and programs in biomedicine. 165. 10.1016/j.cmpb.2018.07.012.

[3] Latif, Jahanzaib & Xiao, Chuangbai & Imran, Azhar & Tu, Shanshan. (2019). Medical Imaging using Machine Learning and Deep Learning Algorithms: A Review *. 10.1109/ICOMET.2019.8673502.

[4] Shen, Dinggang et al. "Deep Learning in Medical Image Analysis." *Annual review of biomedical engineering* vol. 19 (2017): 221-248. doi:10.1146/annurev-bioeng-071516-044442

[5] Shu, Mengying, "Deep learning for image classification on very small datasets using transfer learning" (2019). Creative Components. 345.

[6] Reddy, S. et al. "Optimization of Deep Learning using Various Optimizers, Loss Functions and Dropout." (2019).