CS671: Deep Learning and Applications
ASSIGNMENT 3 REPORT

Backpropagation Optimization Algorithms

Ankit Mehra(S22020)
Urvashi Goswami(S22024)
Ashish Rana(S22019)

S.No	Content	Page No
1	Introduction	1
2	Optimizers	2
3	Description of Architectures	7
4	Architectures with Hidden Layer 3	8 - 20
5	Architectures with Hidden Layer 4	21 - 31
6	Best Architecture	32
7	Results	33

March 22, 2023

Introduction

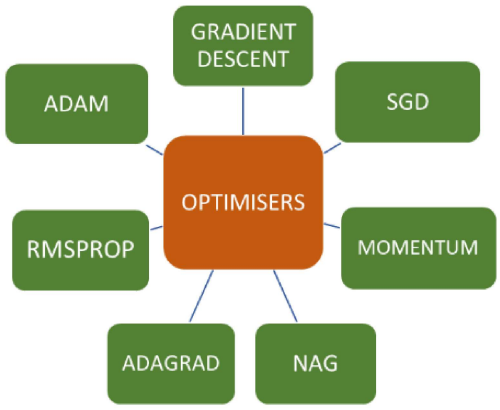


Fig 1. Types of Optimisers

In deep learning, backpropagation is the most widely used algorithm for training neural networks.Backpropagation optimization algorithms are responsible for reducing the losses and to provide the most accurate results possible. The weight is initialized using some initialization strategies and is updated with each epoch according to the update equation.

$$w_{new} = w_{old} + lr * (\nabla w L)$$

The above equation is the update equation using which weights are updated to reach the most accurate result. The best result can be achieved using some optimization strategies or algorithms called optimizers.Various optimizers are researched within the last few couples of years each having its advantages and disadvantages.

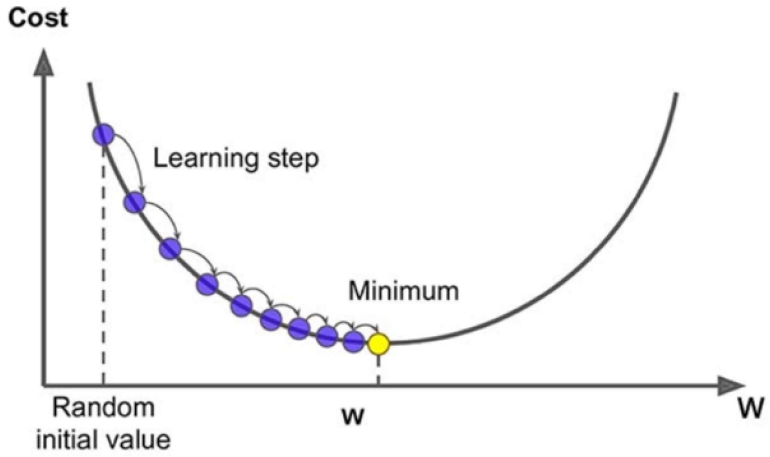


Fig 2. Image Source : codingninjas.com

Gradient Descent (GD):

Gradient descent is the most basic and first-order optimization algorithm which is dependent on the first-order derivative of a loss function. It calculates which way the weights should be altered so that the

function can reach a minimum. Through backpropagation, the loss is transferred from one layer to another and the model's parameters also known as weights are modified depending on the losses so that the loss can be minimized.

$$w_{new} = w_{old} + lr * (\nabla wL)$$

From the image above (Fig 1) it is observed that weights are updated to converge to the minima.Gradient descent may get stuck at local minima or saddle point and con never converge to minima. To find the best solution the algorithm must reach global minima.

Advantages

1. Very simple to implement.

Disadvantages

1. This algorithm takes an entire dataset of n-points at a time to compute the derivative to update the weights which require a lot of memory.
2. Minima is reached after a long time or is never reached.
3. This algorithm can be stuck at local minima or saddle point.

Stochastic Gradient Descent (SGD):

SGD algorithm is an extension of the GD algorithm and it overcomes some of the disadvantages of the GD algorithm. The GD algorithm has a disadvantage that it requires a lot of memory to load the entire dataset of n-points at a time to the computer derivative. In the case of the SGD algorithm derivative is computed taking one point at a time.

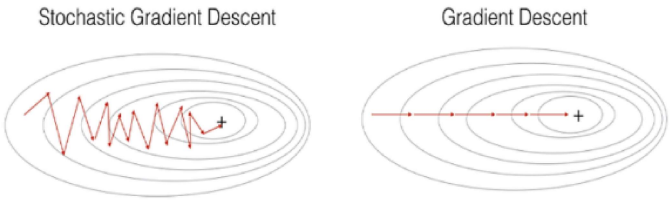


Fig 3. Image Source : www.kdnuggets.com

From the above diagram (Fig 2), it is observed that the updates take more number of iterations compared to gradient descent to reach minima. In the right part of Fig 2, the GD algorithm takes fewer steps to reach minima but the SGD algorithm is noisier and takes more iterations.

Advantages

1. Memory requirement is less compared to the GD algorithm as the derivative is computed taking only 1 point at once.

Disadvantages

1. The time required to complete 1 epoch is large compared to the GD algorithm.
2. Takes a long time to converge.
3. May stuck at local minima.

SGD with momentum:

A major disadvantage of the MB-SGD algorithm is that updates of weight are very noisy. SGD with momentum overcomes this disadvantage by denoising the gradients. Updates of weight are dependent on noisy derivatives and if we somehow denoise the derivatives then converging time will decrease.

The idea is to denoise derivative using an exponential weighting average that is to give more weightage to recent updates compared to the previous update.

SGD update equation:

- $\boldsymbol{w_{new} = w_{old} - lr * (\nabla wL)}$
- $\boldsymbol{v_{new} = \alpha * v_{old} + lr * (\nabla wL)}$
- $\boldsymbol{w_{new} = w_{old} - (\alpha * v_{old} + lr * (\nabla wL))}$

  where 0<$\alpha < 1$

Momentum at time 't' is computed using all previous updates giving more weightage to recent updates compared to the previous update. This leads to speed up the convergence.

When the new weight is calculated by subtracting only the gradient term with a previous weight the update moves in direction 1 and if new weight is calculated by subtracting momentum term with a previous weight then update moves in direction 2.

If we combine both the equation and calculate the new weight by subtracting previous weight with the sum of momentum and gradient term then the update will move towards direction 3 which results in denoising the update.

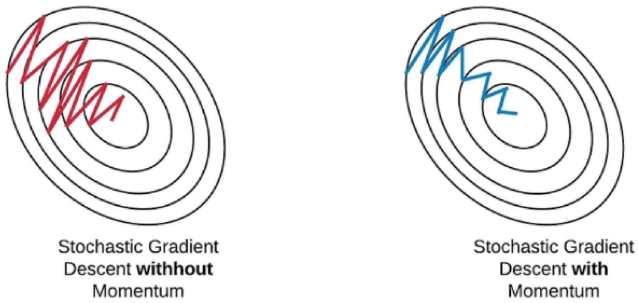


Fig 4. Image Source : www.kdnuggets.com

The diagram above (Fig 4) concludes SGD+momentum denoises the gradients and converges faster as compared to SGD.

Advantages

1. Has all advantages of the SGD algorithm.
2. Converges faster than the GD algorithm.

Disadvantages

1. We need to compute one more variable for each update.

Nesterov Accelerated Gradient (NAG):

The idea of the NAG algorithm is very similar to SGD with momentum with a slight variant. In the case of SGD with momentum algorithm, the momentum and gradient are computed on previous updated weight.

- $\boldsymbol{W_{dash} = W_{t-1} - \alpha * V_{t-1}}$
- $\boldsymbol{W_t = W_{dash} - (lr * \nabla W_{dash})L}$
- $\boldsymbol{W_t = W_{t-1} - ((momentum) + (lr * deviate))}$
- $\boldsymbol{W_t = W_{t-1} - ((\alpha * V_{t-1}) + (lr * \nabla W_{dash}L))}$

According to the NAG algorithm, firstly compute momentum V_t for point W_{t-1} and move in that direction to reach W_{dash}, then compute the gradient at new updated weight W_{dash} and again move towards the gradient (Image 6). The net movement results in the direction towards the minima.

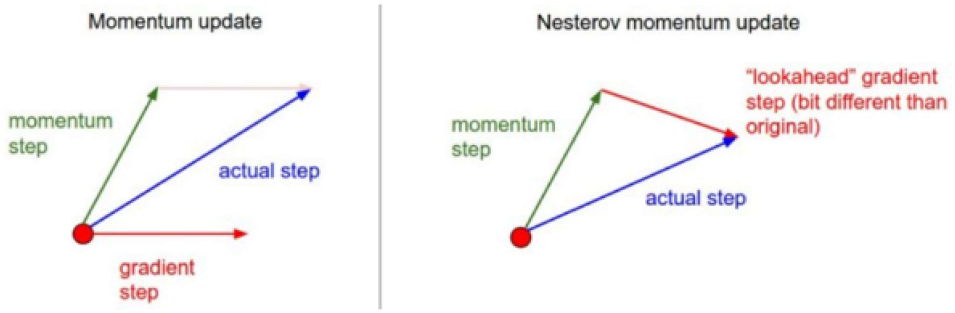


Fig 5. Image Source : Stanford CS231n class

As shown in Fig ,in Nesterov momentum, instead of evaluating gradients at the current position (red circle), we know that our momentum is about to carry us to the tip of the green arrow. With Nesterov momentum we therefore instead evaluate the gradient at this "looked-ahead" position.

Both NAG and SGD with momentum algorithms work equally well and share the same advantages and disadvantages.

Advantages

1. Does not miss the local minima.
2. Slows if minima's are occurring.

Disadvantages

1. Still, the hyperparameter needs to be selected manually.

Adaptive Gradient (AdaGrad):

For all the previously discussed algorithms the learning rate remains constant. So the key idea of AdaGrad is to have an adaptive learning rate for each of the weights. The learning rate for weight will be decreasing with the number of iterations.

- $\boldsymbol{W_t = W_{t-1} - (lr_t * (\nabla w_{t-1} L)}$
- $\boldsymbol{g_t = \nabla w_{t-1} L}$
- $\boldsymbol{W_t = W_{t-1} - (lr_t * g_t)}$
- $\boldsymbol{lr_t = \frac{lr}{\sqrt{\alpha_{t-1} + \varepsilon}}}$
- $\boldsymbol{\alpha_{t-1} = \sum_{i=1}^{t-1} g_i^2}$

So with the increase in the number of the iteration (t) learning rate alpha increasing, which results in an adaptively decreasing learning rate.

Advantages

1. No need to update the learning rate manually as it changes adaptively with iterations.

Disadvantages

1. As the number of iteration becomes very large learning rate decreases to a very small number which leads to slow convergence.

RMSprop

Root mean square prop or RMSprop is another adaptive learning rate that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, we take the exponential moving average (again!) of these gradients. Similar to momentum, we will slowly see that this update becomes the standard update for the learning rate component for most optimisers.

$w_{t+1} = w_t - \frac{\alpha}{\sqrt{v_t + \varepsilon}} . \frac{\partial L}{\partial w_t}$ where,

$v_t = \beta v_{t-1} + (1 - \beta)[\frac{\partial L}{\partial w_t}]^2$, 0 < $\beta < 1$

t- time step, w — weight/parameter, — learning rate, L/w — gradient of L wrt w

Advantages

1. Instead of inefficiently storing all previous squared gradients, - recursively defines as a decaying average of all past squared gradients, keeping the learning rate optimally high.

Disadvantages

1. Does not keep an exponentially decaying average of past gradients (similar to momentum).

Adam:

In the case of the AdaDelta algorithm, we were storing exponential decaying averages of the square of gradients to modify the learning rate. For Adam optimizer, the idea is to store both 1st order of moment (g_t) and 2nd order moment of the gradient (square of g_t).

EDA for 1st order moment:

$$\boldsymbol{m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t}$$

EDA for 2nd order moment:

$$\boldsymbol{v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t}$$

Bias Correction:

$$\boldsymbol{m_t^* = \frac{m_t^*}{1-\beta_1^t}}$$
$$\boldsymbol{v_t^* = \frac{v_t^*}{1-\beta_2^t}}$$

Update Function:

$$\boldsymbol{w_{t+1} = w_t - \frac{m_t^*}{\sqrt{v_t + \varepsilon}}} \, , where 0 < \beta_! 1, \beta_2, \alpha < 1$$

Advantages

1. The method is too fast and converges rapidly.
2. Rectifies vanishing learning rate, high variance.
3. Adam is considered the best algorithm amongst all the algorithms discussed above.

Disadvantages

1. Computationally costly.

Training Dataset Samples

Number of datapoints used to perform the task

Digits	Training Data	Validation Data	Testing Data
0	2277	759	759
1	2277	759	759
2	2277	759	759
4	2277	759	759
7	2277	759	759
Total	**11385**	**3795**	**3795**

Table 2: Dataset description

We have MNIST data set with 5 sets of digits [0,1,2,4,7] . The Deep Neural network is trained using these images of size (28×28) using different architectures. We have prepared 6 different architectures , three with 3 hidden layers and three with 4 hidden layers.

Details of Architectures.

Architecture	Input Layer	Hidden Layer1	Hidden Layer2	Hidden Layer 3	Hidden Layer 4	Output Layer
1	784	512	512	512	-	5
2	784	512	256	256	-	5
3	784	512	256	128	-	5
4	784	256	128	128	64	5
5	784	512	256	128	64	5
6	784	512	512	256	128	5

Table 3: Architecture description

The input layer has 784 neurons that is 28×28 and the output layer has 5 neurons representing each of the 5 digits (0,1,2,4,7) given to us.

Initial weights for various optimizers are taken the same so that comparison can be done in an unbiased way. **model.save_weights("path for saving weights ")** and **model.load_weights("path for loading weights ")**

Values of parameters taken over all architectures

Parameters	Values
Learning Rate	0.001
Activation (Hidden)	Sigmoidal
Activation (Output)	Softmax
Batch Size	1
Loss Function	Cross Entropy
Stopping Criterion	Early Stopping (Patience=1)

Table 4: Parameter value description

Architectures with hidden layer 3

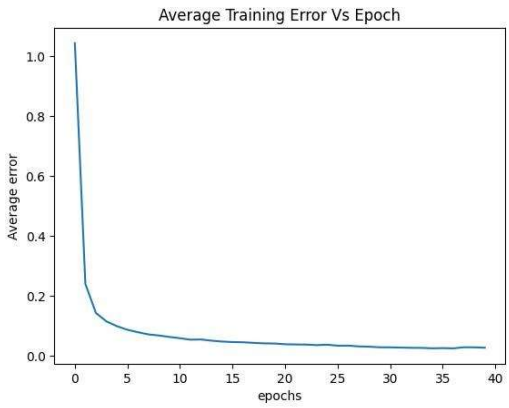
Architecture 1

Hidden layers = 3 , Neurons in H.L1=512 ,Neurons in H.L2=512 ,Neurons in H.L3=512.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 Input_Layer (Flatten)        (None, 784)               0

 Hidden_Layer_1 (Dense)       (None, 512)               401920

 Hidden_Layer_2 (Dense)       (None, 512)               262656

 Hidden_Layer_3 (Dense)       (None, 512)               262656

 Output_Layer (Dense)         (None, 5)                 2565

=================================================================
Total params: 929,797
Trainable params: 929,797
Non-trainable params: 0
```

Fig 6: Architecture 1 details

Optimizer 1 **(SGD Pattern mode)**



Results

Set	Loss	Accuracy
Training set	0.0350	99.07%
Validation set	0.0525	98.42%

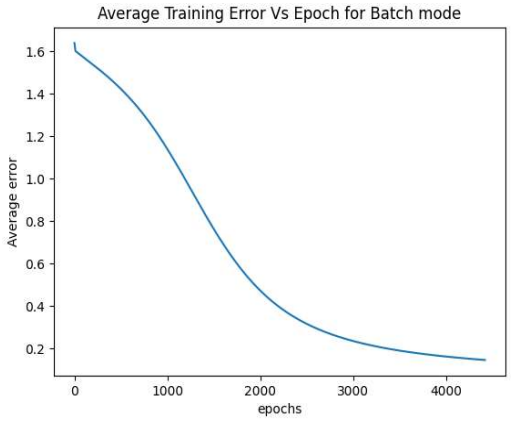
This architecture converged in **39 epochs**

Optimizer 2 **(SGD Batchmode)**

Results

Set	Loss	Accuracy
Training set	0.1179	97.00%
Validation set	0.1578	95.76%

This architecture converged in **4417 epochs** .
Momentum =0.6

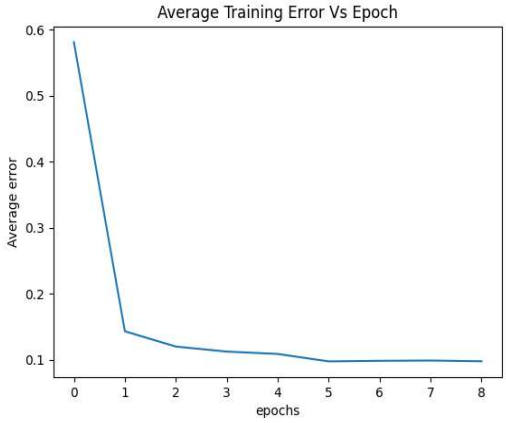


Optimizer 3 **(SGD with momentum)**

Results

Set	Loss	Accuracy
Training set	0.1207	96.37%
Validation set	0.1004	95.97%

This architecture converged in **9 epochs** .
Momentum =0.6

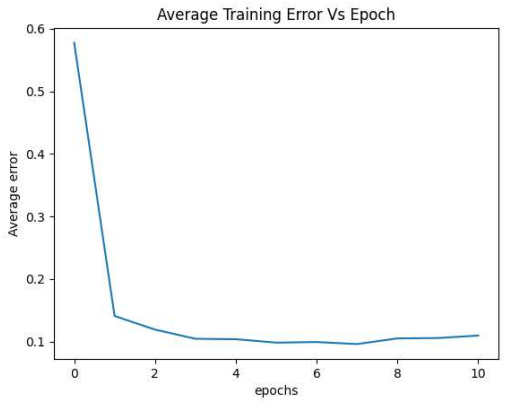


Optimizer 4 **(NAG)**

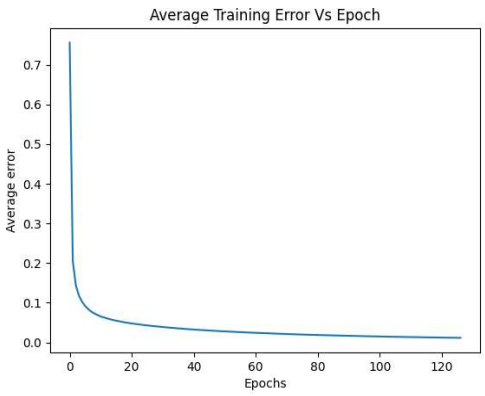
Results

Set	Loss	Accuracy
Training set	0.1042	97.04%
Validation set	0.1115	96.73%

This architecture converged in **11 epochs** .
Momentum =0.6



Optimizer 5 **(AdaGrad)**

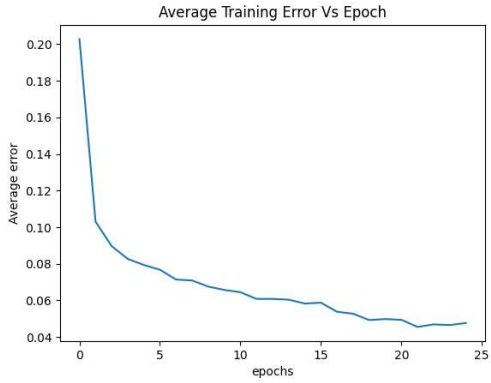


Results

Set	Loss	Accuracy
Training set	0.0051	99.93%
Validation set	0.0628	98.81%

This architecture converged in **127 epochs** .

Optimizer 6 **(RMSProp)**



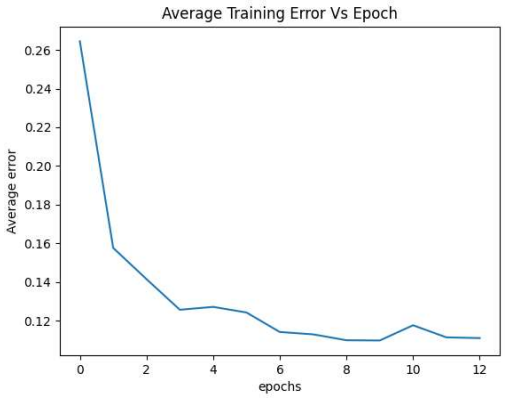
Results

Set	Loss	Accuracy
Training set	0.0784	98.74%
Validation set	0.0828	98.18%

This architecture converged in **25 epochs** .

$\beta = 0.99$ *and* $\varepsilon = 10^{-8}$

Optimizer 7 **(Adam)**



Results

Set	Loss	Accuracy
Training set	0.165	93.58%
Validation set	0.187	92.33%

This architecture converged in **13 epochs** .

$\beta_1 = 0.9,\ \beta_2 = 0.999$ *and* $\varepsilon = 10^{-8}$

Superimposition of the plot of each optimizer for architecture 1 (HL3)

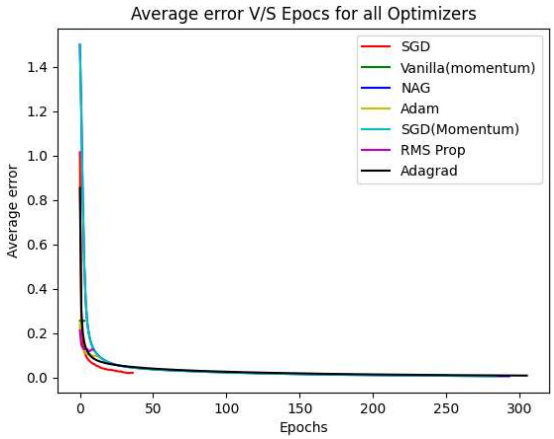


Fig 7: Superimposed plot of all optimizers

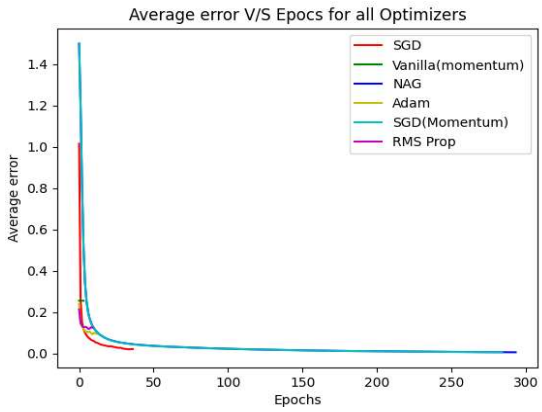


Fig 8: Superimposed plot of some optimizers

Comparison of number of epochs considered by each optimizer for architecture 1(HL3)

Optimizer	Number of Epochs
SGD(Pattern mode)	39
SGD(Batch Mode)	4417
SGD(Momentum)	9
NAG	11
AdaGrad	127
RMSProp	25
Adam	13

Table 4: Comparison of various optimizers

In architecture 1 it can be observed that AdaGrad took the longest time to converge, approximately 3 hours. It can be because of accumulation of the squared gradients in the denominator causing the learning rate to shrink and become infinitesimally small.

Architecture 2

Hidden layers = 3 , Neurons in H.L1=512 ,Neurons in H.L2=256 ,Neurons in H.L3=256.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (Flatten)        (None, 784)               0
_________________________________________________________________
Hidden_Layer_1 (Dense)       (None, 512)               401920
_________________________________________________________________
Hidden_Layer_2 (Dense)       (None, 256)               131328
_________________________________________________________________
Hidden_Layer_3 (Dense)       (None, 256)               65792
_________________________________________________________________
Output_Layer (Dense)         (None, 5)                 1285
=================================================================
Total params: 600,325
Trainable params: 600,325
Non-trainable params: 0
```

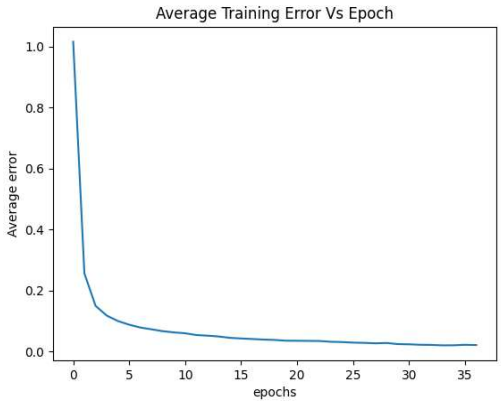
Fig 9: Architecture 2 details.

Optimizer 1 **(SGD Pattern mode)**

Results

Set	Loss	Accuracy
Training set	0.0243	99.45%
Validation set	0.0545	98.39%

This architecture converged in **37 epochs** .

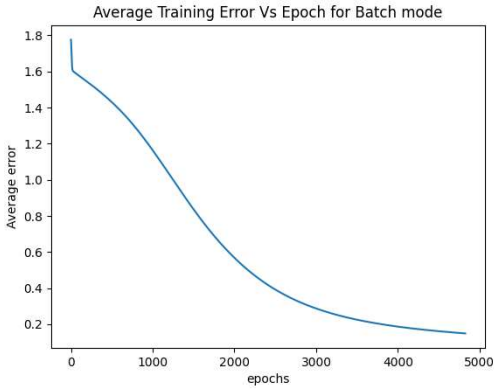


Optimizer 2 **(SGD Batch mode)**

Results

Set	Loss	Accuracy
Training set	0.1495	96.67%
Validation set	0.1608	96.07%

This architecture converged in **4825 epochs** .

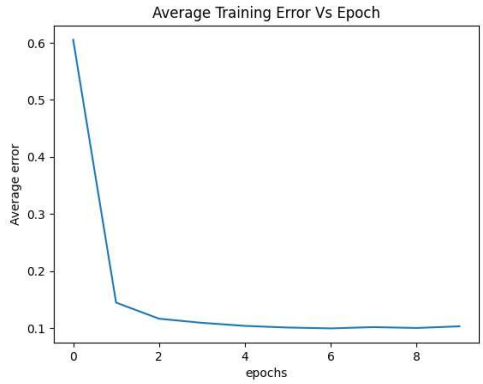


Optimizer 3 **(SGD with momentum)**

Results

Set	Loss	Accuracy
Training set	0.1033	97.23%
Validation set	0.1182	96.73%

This architecture converged in **10 epochs** .
Momentum=0.6

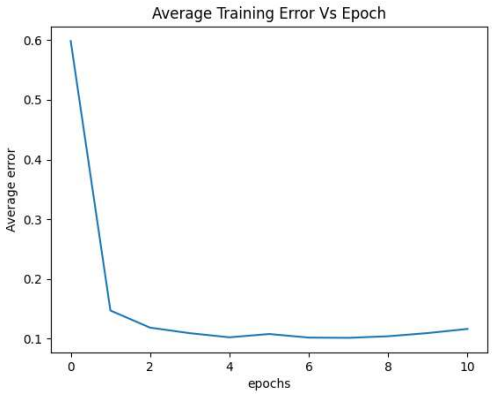


Optimizer 4 **(NAG)**

Results

Set	Loss	Accuracy
Training set	0.1165	96.60%
Validation set	0.231	96.36%

This architecture converged in **11 epochs** .

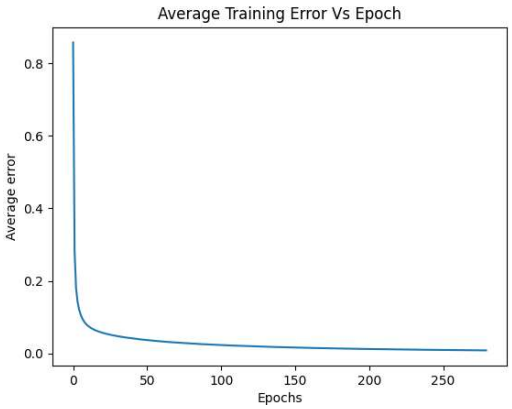


Optimizer 5 **(AdaGrad)**

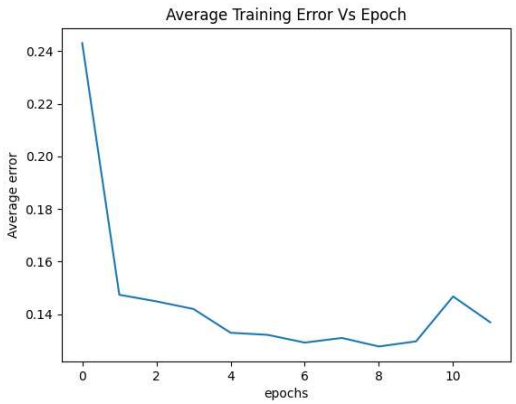
Results

Set	Loss	Accuracy
Training set	0.0395	98.98%
Validation set	0.0636	97.97%

This architecture converged in **280 epochs** .



Optimizer 6 **(RMSProp)**



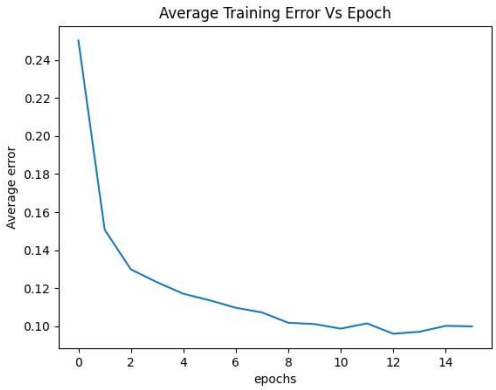
Results

Set	Loss	Accuracy
Training set	0.1369	97.79%
Validation set	0.1370	97.44%

This architecture converged in 1**2 epochs** .

$\beta = 0.99 \; and \; \varepsilon = 10^{-8}$

Optimizer 7 **(Adam)**



Results

Set	Loss	Accuracy
Training set	0.099	97.61%
Validation set	0.0930	97.52%

This architecture converged in **16 epochs** .

$\beta_1 = 0.9, \; \beta_2 = 0.999 \; and \; \varepsilon = 10^{-8}$

Superimposition of the plot of each optimizer for architecture 2 (HL3)

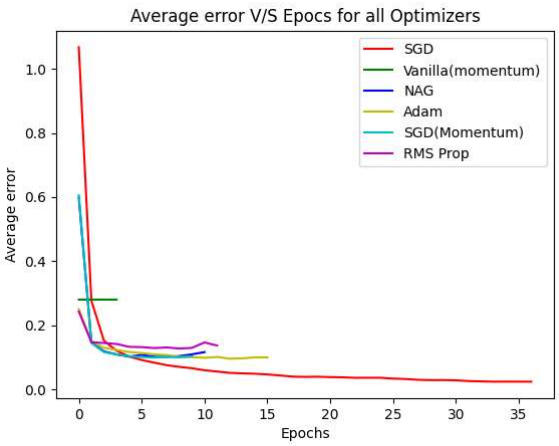


Fig 10: Superimposed plot of some optimizers

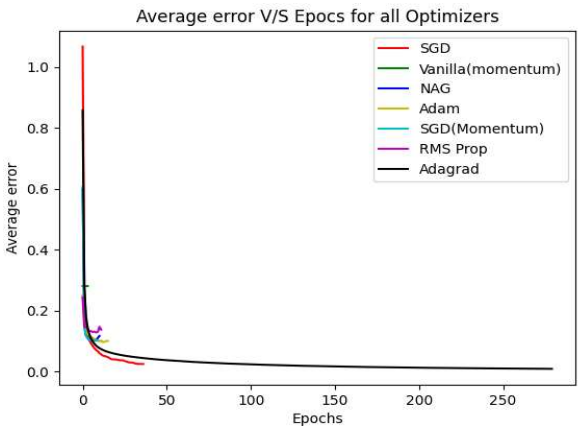


Fig 11: Superimposed plot of all optimizers

Comparison of number of epochs considered by each optimizer for architecture 2(HL3)

Optimizer	Number of Epochs
SGD(Pattern mode)	37
SGD(Batch Mode)	4825
SGD(Momentum)	10
NAG	11
AdaGrad	280
RMSProp	12
Adam	16

Table 5: Comparison of various optimizers

Architecture 3

Hidden layers = 3 , Neurons in H.L1=512 ,Neurons in H.L2=256 ,Neurons in H.L3=128.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (Flatten)        (None, 784)               0
_________________________________________________________________
Hidden_Layer_1 (Dense)       (None, 512)               401920
_________________________________________________________________
Hidden_Layer_2 (Dense)       (None, 256)               131328
_________________________________________________________________
Hidden_Layer_3 (Dense)       (None, 128)               32896
_________________________________________________________________
Output_Layer (Dense)         (None, 5)                 645
=================================================================
Total params: 566,789
Trainable params: 566,789
Non-trainable params: 0
```

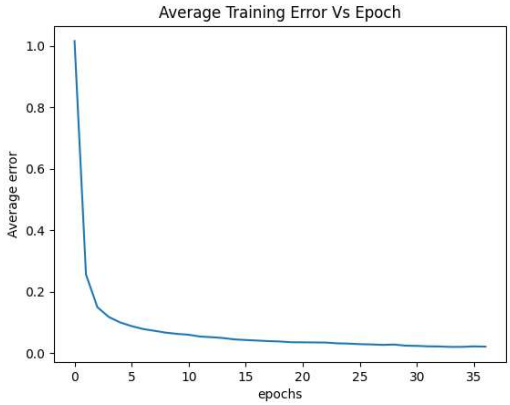
Fig 12: Architecture 3 details.

Optimizer 1 **(SGD Pattern mode)**

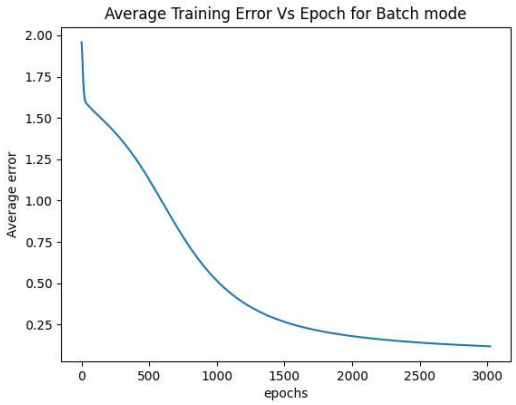
Results

Set	Loss	Accuracy
Training set	0.0207	99.53%
Validation set	0.0481	98.52%

This architecture converged in **37 epochs** .



Optimizer 2 **(SGD Batch mode)**

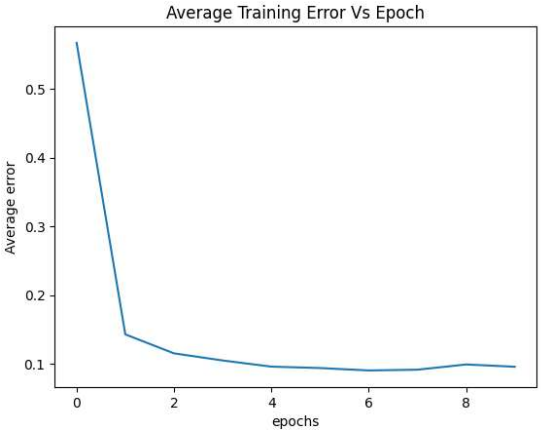


Results

Set	Loss	Accuracy
Training set	0.1189	97.23%
Validation set	0.1347	96.34%

This architecture converged in **3022 epochs** .
Momentum=0.9

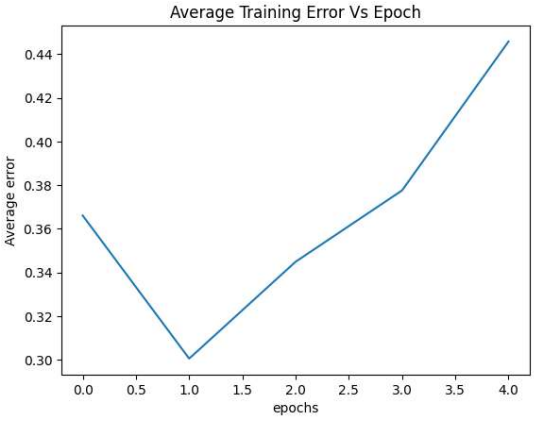
Optimizer 3 **(SGD with momentum)**



Results

Set	Loss	Accuracy
Training set	0.0957	97.34%
Validation set	0.0956	97.18%

This architecture converged in **10 epochs** .
Momentum=0.6



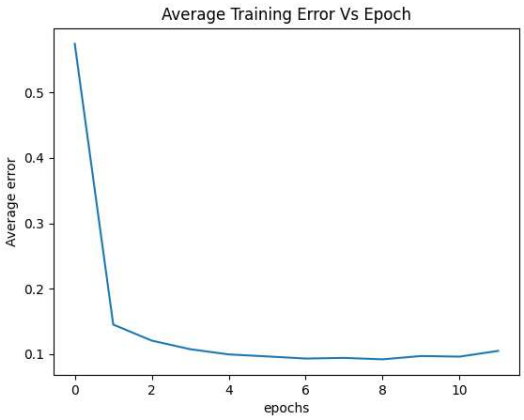
Set	Loss	Accuracy
Training set	0.3882	86.49%
Validation set	0.4533	88.12%

This architecture converged in **5 epochs** .
Momentum=0.9

Optimizer 4 **(NAG)**

Results

Set	Loss	Accuracy
Training set	0.0921	97.30%
Validation set	0.1059	96.89%



This architecture converged in **12 epochs** .
Momentum=0.6

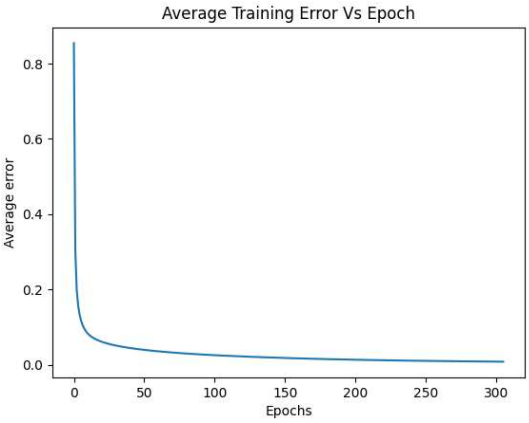
Set	Loss	Accuracy
Training set	0.2051	92.90%
Validation set	0.2187	92.19%

This architecture converged in **4 epochs** .
Momentum=0.9

Optimizer 5 **(AdaGrad)**

Results

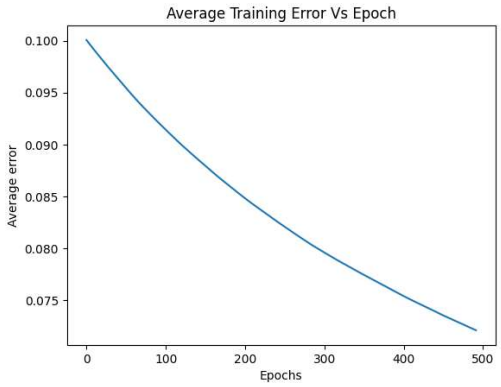
Set	Loss	Accuracy
Training set	0.0083	99.90%
Validation set	0.0626	98.08%



This architecture converged in **306 epochs** .

Results

Set	Loss	Accuracy
Training set	0.0510	97.93%
Validation set	0.0712	97.52%

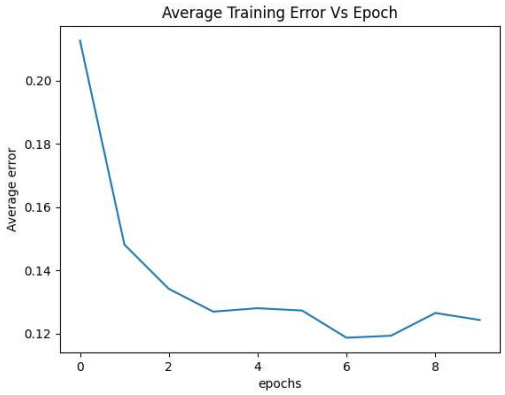


This architecture converged in **492 epochs** .
Batch size = 512

Optimizer 6 **(RMSProp)**

Results

Set	Loss	Accuracy
Training set	0.1243	97.19%
Validation set	0.1066	97.47%

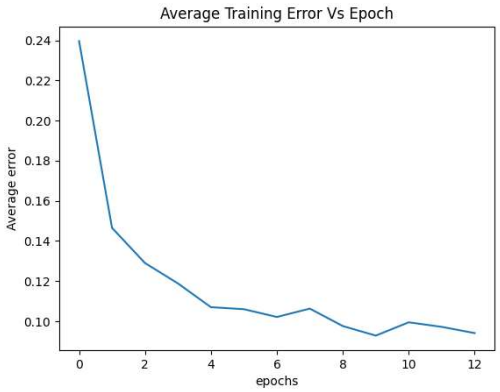


This architecture converged in **10 epochs** .
$\beta = 0.99$ *and* $\varepsilon = 10^{-8}$

Optimizer 7 **(Adam)**

Results

Set	Loss	Accuracy
Training set	0.0929-	97.58%
Validation set	0.0941	97.26%



This architecture converged in **13 epochs** .
$\beta_1 = 0.9,\ \beta_2 = 0.999$ *and* $\varepsilon = 10^{-8}$

OBSERVATION 1 :-We observed that AdaGrad is taking almost 3 hrs to converge in pattern mode so we tried running the model with batch mode with batch size 512 and it converged in less than 10 minutes. It is because in pattern mode over the iterations large update history for parameters are accumulated.

OBSERVATION 2 :-It can be observed that in NAG and SGD(Momentum) if the momentum factor is kept 0.9 the average error initially decreases and then increases , it is because of high momentum which causes the optimizer to overshoot the minimum , resulting in an increase in average error . So we tweaked the value of momentum to 0.6 for comparison and the average error kept on decreasing and converged after sufficient epochs as expected.

Superimposition of the plot of each optimizer for architecture 3 (HL3)

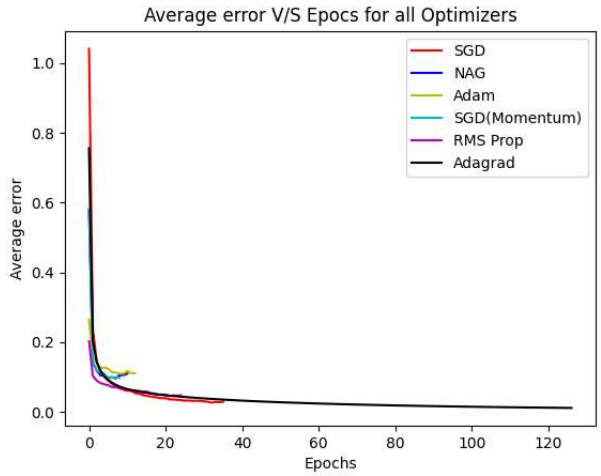


Fig 13: Superimposed plot of all optimizers

Comparison of number of epochs considered by each optimizer for architecture 1(HL3)

Optimizer	Number of Epochs
SGD(Pattern mode)	37
SGD(Batch Mode)	3022
SGD(Momentum)	10
NAG	12
AdaGrad	306
RMSProp	10
Adam	13

Table 6: Comparison of various optimizers

Architectures with 4 hidden layers

Architecture 1

Hidden layers = 3 , Neurons in H.L1=256 ,Neurons in H.L2=128 ,Neurons in H.L3=128 Neurons in HL4=64.

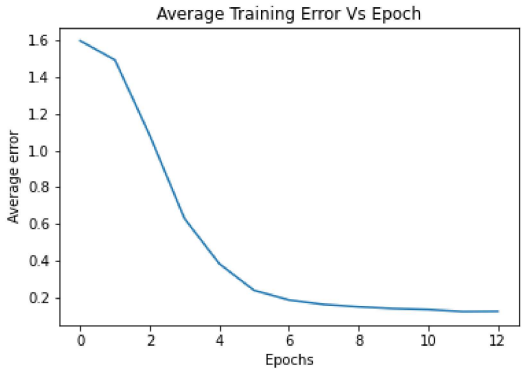
```
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 Input_Layer (Flatten)        (None, 784)               0

 Hidden_Layer_1 (Dense)       (None, 256)               200960

 Hidden_Layer_2 (Dense)       (None, 128)               32896

 Hidden_Layer_3 (Dense)       (None, 128)               16512

 Hidden_Layer_4 (Dense)       (None, 64)                8256

 Output_Layer (Dense)         (None, 5)                 325

=================================================================
Total params: 258,949
Trainable params: 258,949
Non-trainable params: 0
```

Fig 14: Architecture 1 details.

Optimizer 1 **(SGD Pattern mode)**

Results

Set	Loss	Accuracy
Training set	0.1232	97.12%
Validation set	0.1307	96.92%

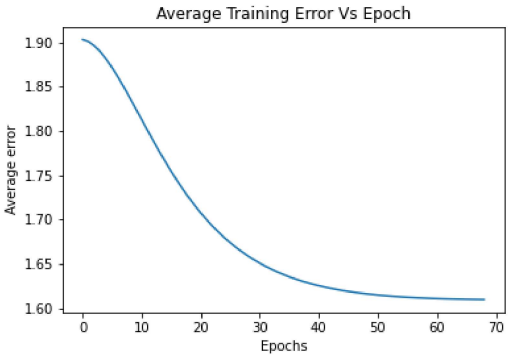


This architecture converged in **13 epochs** .

Optimizer 2 **(SGD Batch mode)**

Results

Set	Loss	Accuracy
Training set	1.602	20.0%
Validation set	1.609	19.87%

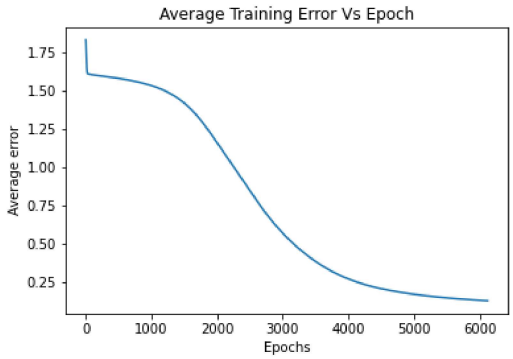


This architecture converged in **69 epochs** .

Optimizer 3 **(SGD Batch mode with momentum)**

Results

Set	Loss	Accuracy
Training set	0.1561	97.12%
Validation set	0.1347	96.34%

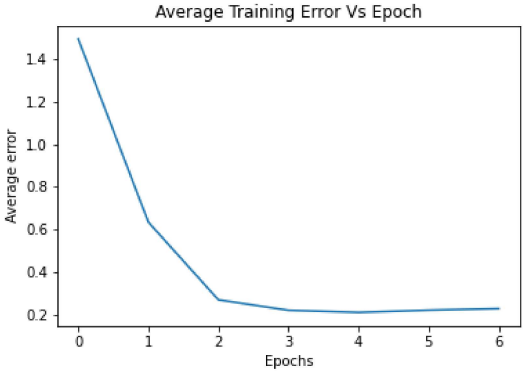


This architecture converged in **3022 epochs** .
Momentum=0.9

Optimizer 4 **(SGD with momentum)**

Results

Set	Loss	Accuracy
Training set	0.2210	94.49%
Validation set	0.2538	92.49%

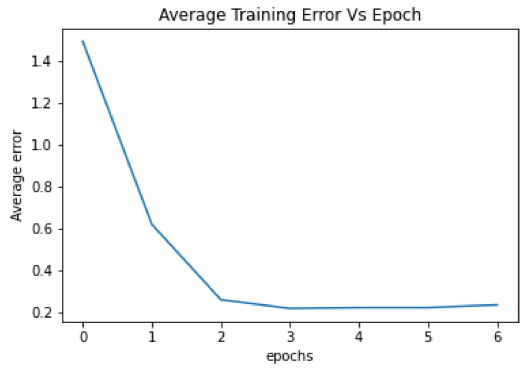


This architecture converged in **7 epochs** .
Momentum=0.6

Optimizer 5 **(NAG)**

Results

Set	Loss	Accuracy
Training set	0.2113	94.45%
Validation set	0.2117	93.89%

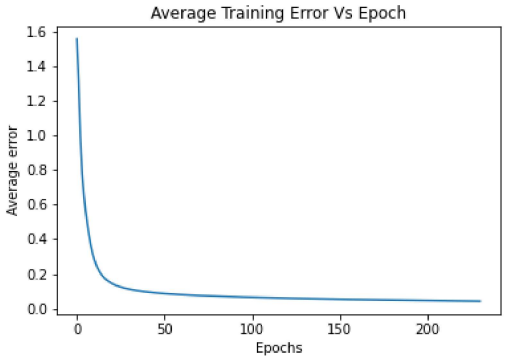


This architecture converged in **7 epochs** .
Momentum=0.6

Optimizer 6 **(AdaGrad)**

Results

Set	Loss	Accuracy
Training set	0.048	98.87%
Validation set	0.0757	98.80%

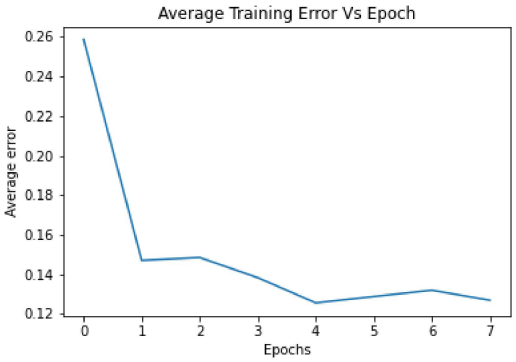


This architecture converged in **231 epochs** .

Optimizer 7 **(RMSProp)**

Results

Set	Loss	Accuracy
Training set	0.1256	97.58%
Validation set	0.1142	97.36%



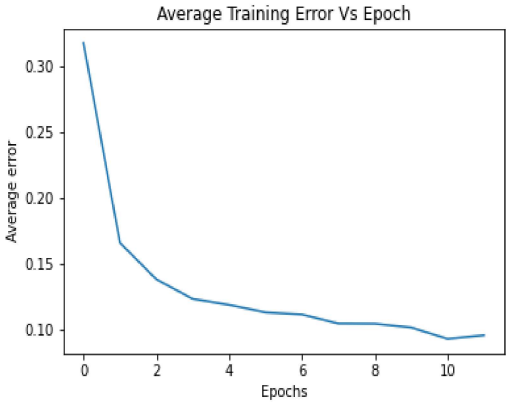
This architecture converged in 8 **epochs** .

$\beta = 0.99$ *and* $\varepsilon = 10^{-8}$

Optimizer 8 **(Adam)**

Results

Set	Loss	Accuracy
Training set	0.1043	97.36%
Validation set	0.1157	96.73%



This architecture converged in **12 epochs** .

$\beta_1 = 0.9,\ \beta_2 = 0.999$ *and* $\varepsilon = 10^{-8}$

Superimposition of the plot of each optimizer for architecture 1 (HL4)

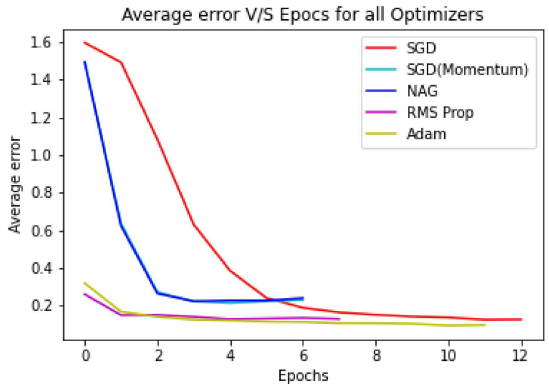


Fig 15: Superimposed plot of some optimizers

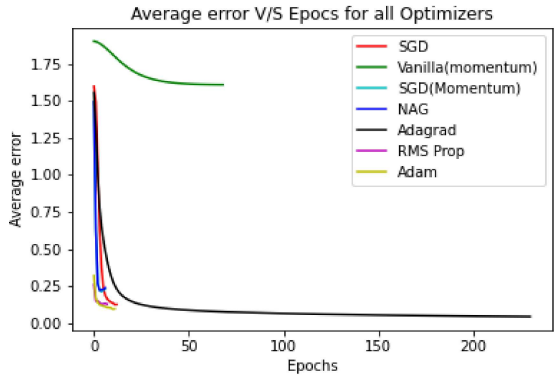


Fig 16: Superimposed plot of all optimizers

It can be observed in the result of SGD (Batch mode) that the model converges in 69 epochs and the accuracy value is stuck on 20% for whole of training , this is because the optimizer has gone into a local minima , this problem can be overcome by adding a little momentum . We can observe in SGD (Batch with momentum) that after adding a momentum of 0.9 the model converges in 3022 epochs but the final accuracy is 97.12%.

Comparison of number of epochs considered by each optimizer for architecture 1(HL4)

Optimizer	Number of Epochs
SGD(Pattern mode)	13
SGD(Batch Mode)	69
SGD(Batch with momentum)	3022
SGD(Momentum)	7
NAG	7
AdaGrad	231
RMSProp	8
Adam	12

Table 7: Comparison of various optimizers

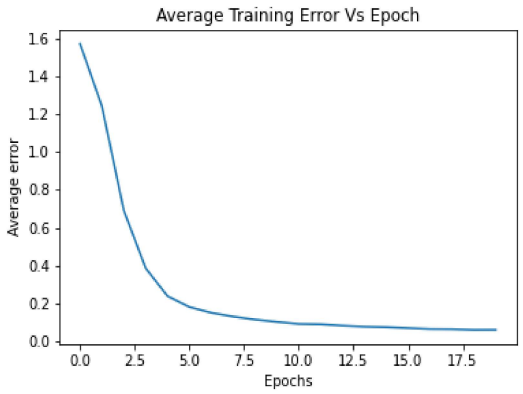
Architecture 2

Hidden layers = 4 , Neurons in H.L1=512 ,Neurons in H.L2=256 ,Neurons in H.L3=128 Neurons in HL4=64.

```
 Layer (type)                 Output Shape              Param #
=================================================================
 Input_Layer (Flatten)        (None, 784)               0

 Hidden_Layer_1 (Dense)       (None, 512)               401920

 Hidden_Layer_2 (Dense)       (None, 256)               131328

 Hidden_Layer_3 (Dense)       (None, 128)               32896

 Hidden_Layer_4 (Dense)       (None, 64)                8256

 Output_Layer (Dense)         (None, 5)                 325

=================================================================
Total params: 574,725
Trainable params: 574,725
Non-trainable params: 0
```

Fig 17: Architecture 2 details.

Optimizer 1 **(SGD Pattern mode)**

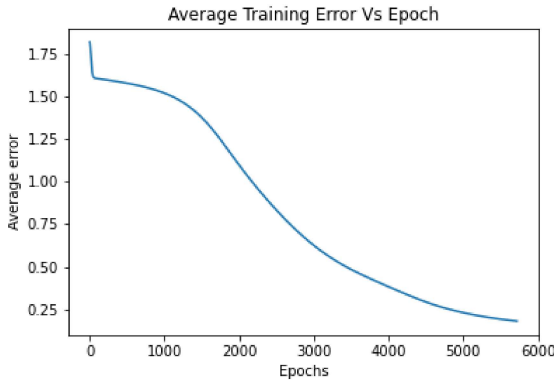


Results

Set	Loss	Accuracy
Training set	0.0617	98.44%
Validation set	0.0772	97.79%

This architecture converged in **20 epochs** .

Optimizer 2 **(SGD Batch mode)**



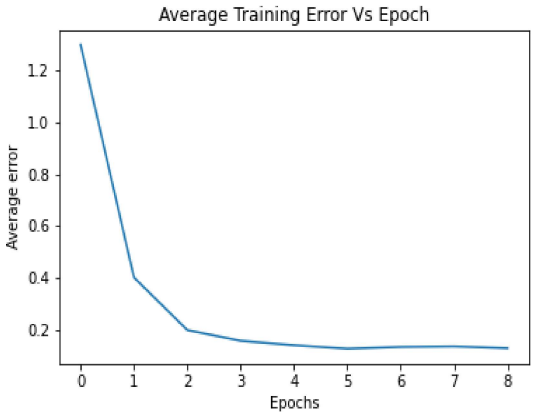
Results

Set	Loss	Accuracy
Training set	0.1817	96.56%
Validation set	0.2056	95.10%

This architecture converged in **5719 epochs** .
Momentum=0.9

Optimizer 3 **(SGD with momentum)**

Results

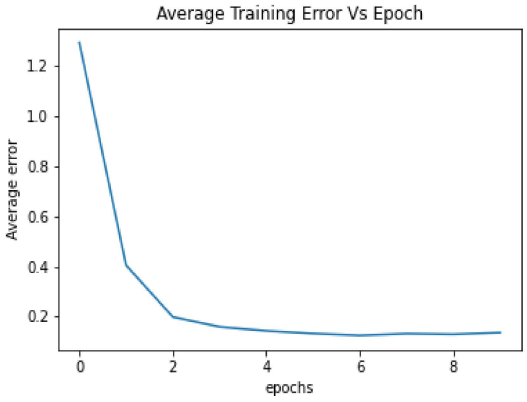


Set	Loss	Accuracy
Training set	0.1315	97.64%
Validation set	0.1023	97.29%

This architecture converged in **9 epochs** .
Momentum=0.6

Optimizer 4 **(NAG)**

Results

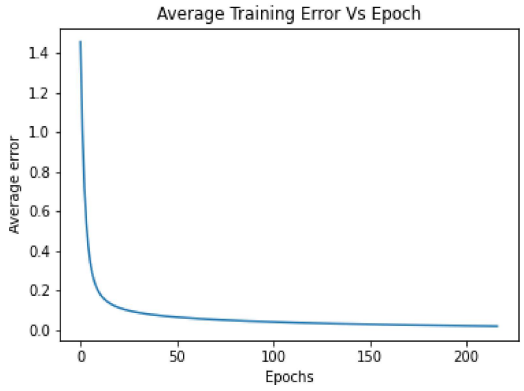


Set	Loss	Accuracy
Training set	0.1251	96.73%
Validation set	0.1437	96.36%

This architecture converged in **10 epochs** .
Momentum=0.6

Optimizer 5 **(AdaGrad)**

Results



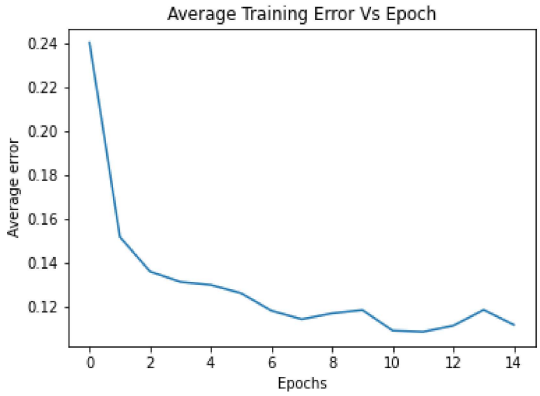
Set	Loss	Accuracy
Training set	0.0185	99.70%
Validation set	0.0703	97.94%

This architecture converged in **217 epochs** .

Optimizer 6 **(RMSProp)**

Results

Set	Loss	Accuracy
Training set	0.1084	97.84%
Validation set	0.1175	97.58%



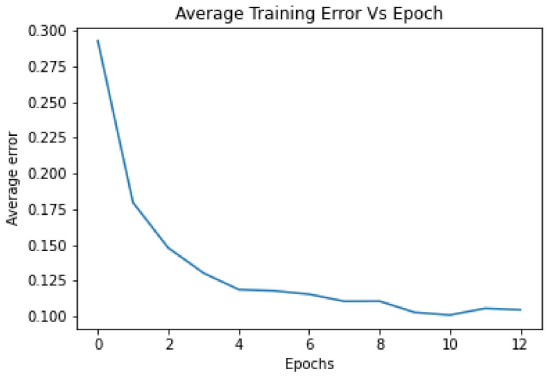
This architecture converged in **15 epochs** .

$\beta = 0.99$ *and* $\varepsilon = 10^{-8}$

Optimizer 7 **(Adam)**

Results

Set	Loss	Accuracy
Training set	0.1009	97.82%
Validation set	0.0854	97.60%



This architecture converged in **13 epochs** .

$\beta_1 = 0.9,\ \beta_2 = 0.999$ *and* $\varepsilon = 10^{-8}$

Superimposition of the plot of each optimizer for architecture 3 (HL4)

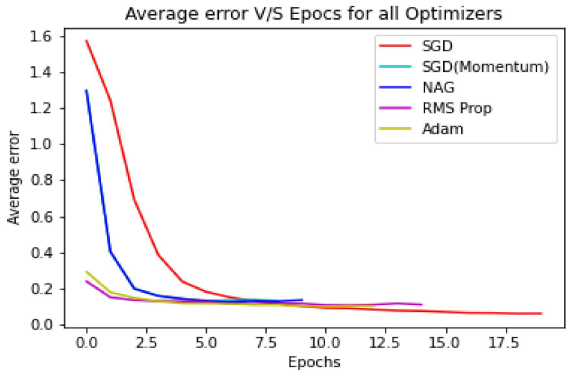


Fig 18: Superimposed plot of some optimizers

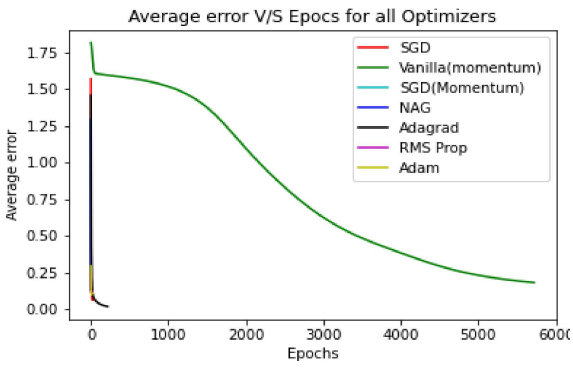


Fig 19: Superimposed plot of all optimizers

Comparison of number of epochs considered by each optimizer for architecture 1(HL4)

Optimizer	Number of Epochs
SGD(Pattern mode)	20
SGD(Batch Mode)	5719
SGD(Momentum)	9
NAG	10
AdaGrad	217
RMSProp	15
Adam	13

Table 8: Comparison of various optimizers

Architecture 3

Hidden layers = 3 , Neurons in H.L1=512 ,Neurons in H.L2=512 ,Neurons in H.L3=256 Neurons in HL4=128.

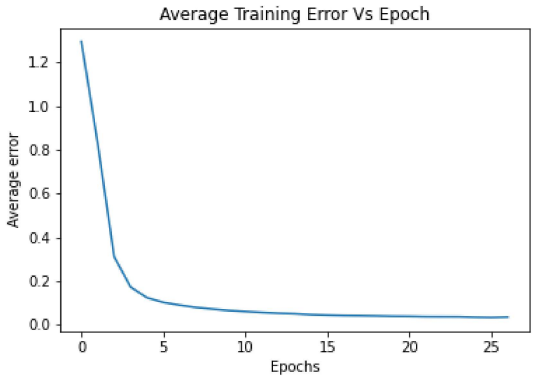
```
Layer (type)                 Output Shape              Param #
=================================================================
Input_Layer (Flatten)        (None, 784)               0

Hidden_Layer_1 (Dense)       (None, 512)               401920

Hidden_Layer_2 (Dense)       (None, 512)               262656

Hidden_Layer_3 (Dense)       (None, 256)               131328

Hidden_Layer_4 (Dense)       (None, 128)               32896

Output_Layer (Dense)         (None, 5)                 645

=================================================================
Total params: 829,445
Trainable params: 829,445
Non-trainable params: 0
```

Fig 20: Architecture 3 details.

Optimizer 1 **(SGD Pattern mode)**

Results

Set	Loss	Accuracy
Training set	0.0323	98.99%
Validation set	0.0529	98.34%

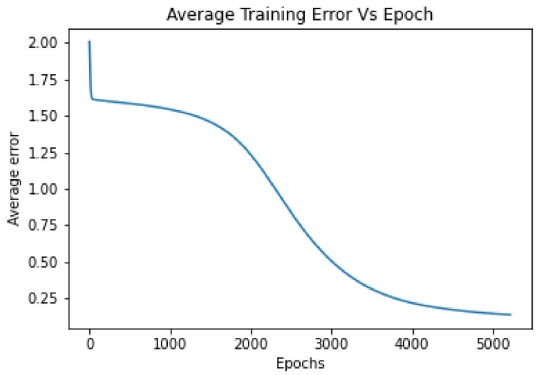


This architecture converged in **27 epochs** .

Optimizer 2 **(SGD Batch mode)**

Results

Set	Loss	Accuracy
Training set	0.1453	96.56%
Validation set	0.1505	96.02%

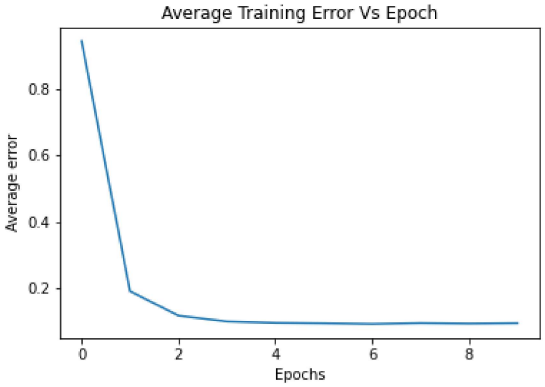


This architecture converged in **5719 epochs** .

Optimizer 3 **(SGD with momentum)**

Results

Set	Loss	Accuracy
Training set	0.0913	97.35%
Validation set	0.0930	97.18%

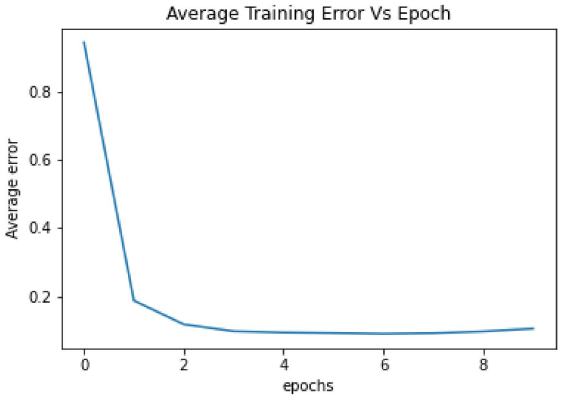


This architecture converged in **10 epochs** .
Momentum=0.6

Optimizer 4 **(NAG)**

Results

Set	Loss	Accuracy
Training set	0.0897	97.44%
Validation set	0.1046	96.65%



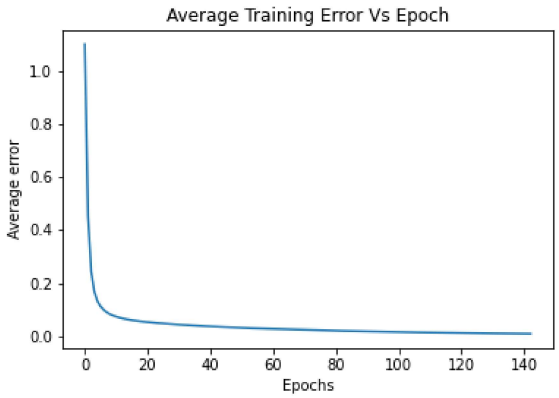
This architecture converged in **10 epochs** .
Momentum=0.6

Optimizer 5 **(AdaGrad)**

Results

Set	Loss	Accuracy
Training set	0.0092	99.83%
Validation set	0.0558	98.21%

This architecture converged in **143 epochs** .



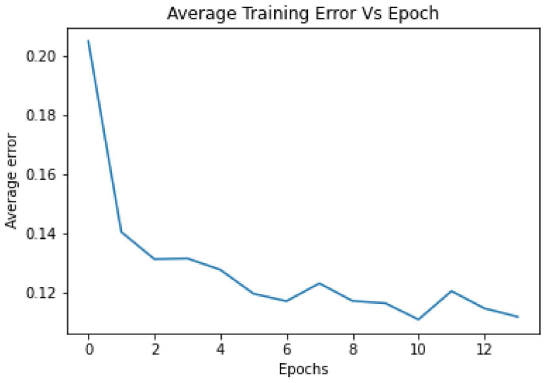
Optimizer 6 **(RMSProp)**

Results

Set	Loss	Accuracy
Training set	0.1121	97.82%
Validation set	0.1293	97.65%

This architecture converged in **14 epochs** .

$\beta = 0.99$ *and* $\varepsilon = 10^{-8}$



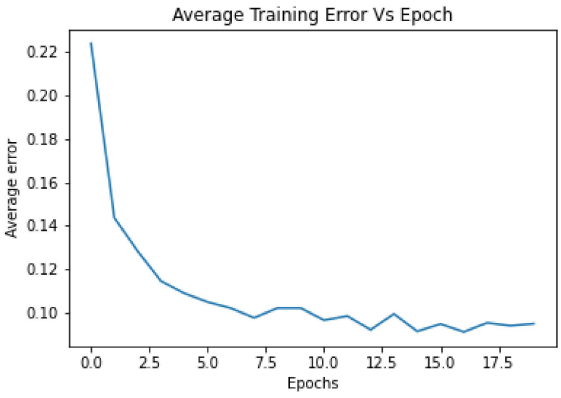
Optimizer 7 **(Adam)**

Results

Set	Loss	Accuracy
Training set	0.0947	97.48%
Validation set	0.0821	97.31%

This architecture converged in **20 epochs** .

$\beta_1 = 0.9,\ \beta_2 = 0.999$ *and* $\varepsilon = 10^{-8}$



Superimposition of the plot of each optimizer for architecture 3 (HL4)

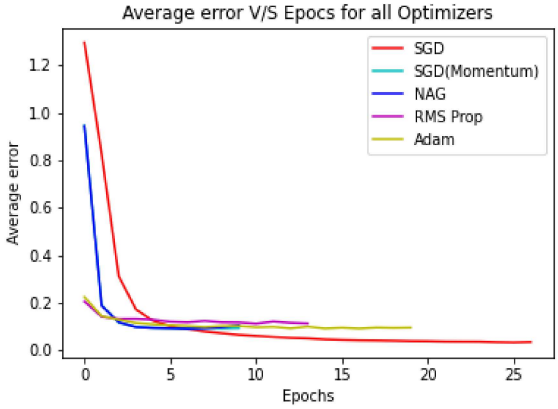


Fig 21: Superimposed plot of some optimizers

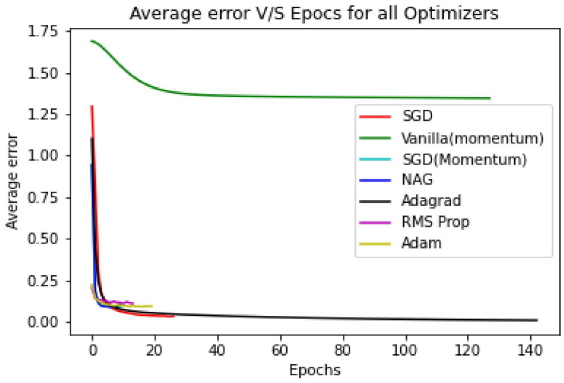


Fig 22: Superimposed plot of all optimizers

Comparison of number of epochs considered by each optimizer for architecture 1(HL4)

Optimizer	Number of Epochs
SGD(Pattern mode)	27
SGD(Batch Mode)	5719
SGD(Momentum)	10
NAG	10
AdaGrad	143
RMSProp	14
Adam	20

Table 9: Comparison of various optimizers

Best Architecture

The best architecture out of all based on the validation accuracy.
Architecture 1 , with 3 hidden layers with 512 neurons in each layer and **AdaGrad** optimizer.

Set	Loss	Accuracy	Epochs
Validation set	0.0628	98.81%	127

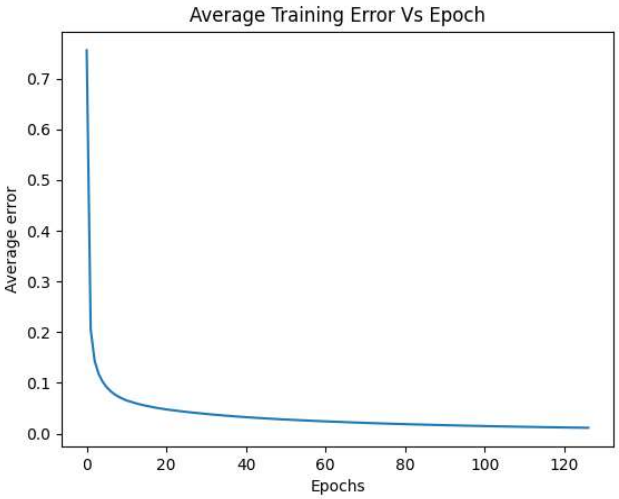


Fig 22:Error vs Epoch graph for AdaGrad

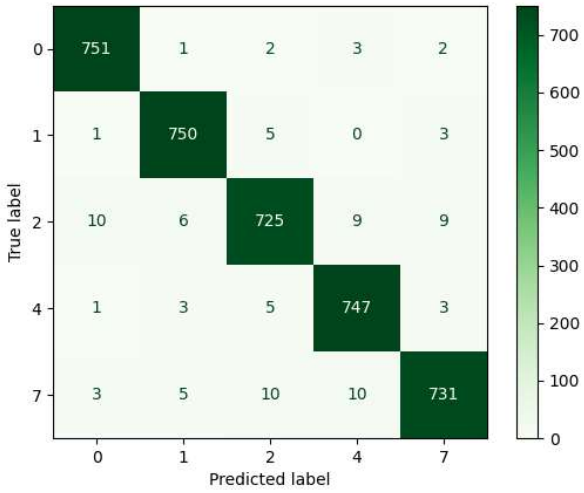


Fig 23:Confusion matrix for AdaGrad

We observed that on the basis of validation accuracy AdaGrad performed the best , but on the basis of the least time taken to converge with over 90% accuracy Adam performed best.

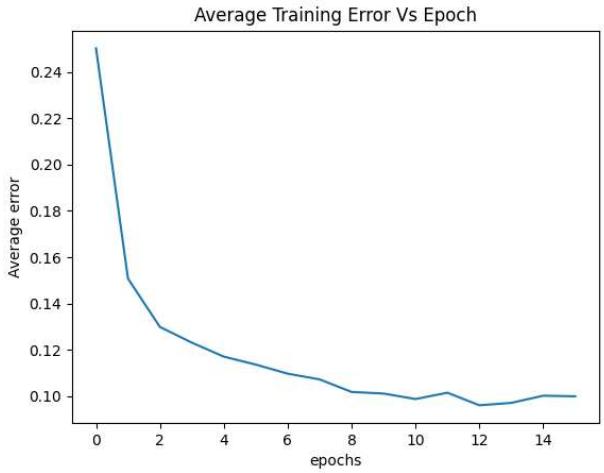


Fig 24:Error vs Epoch graph for Adam

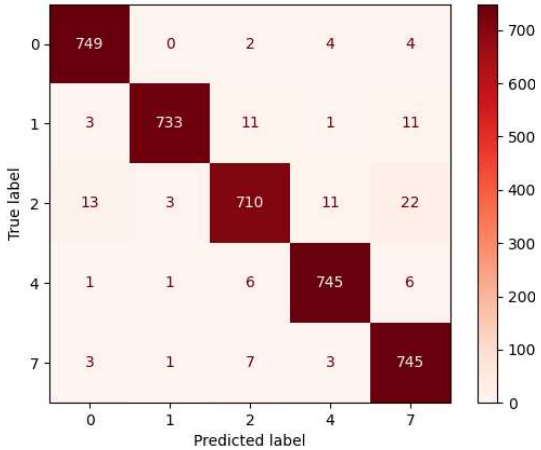
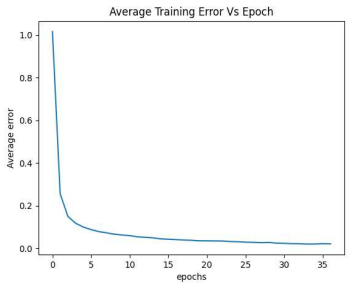
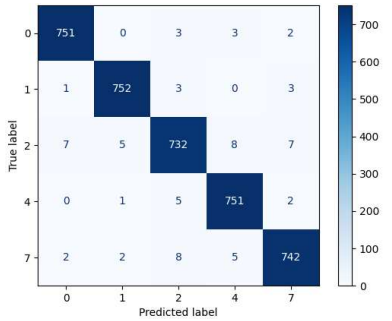
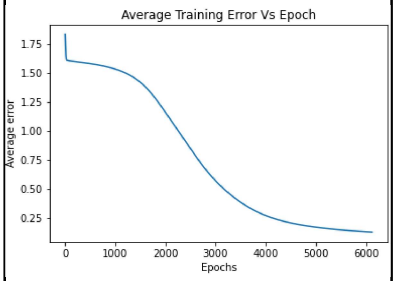
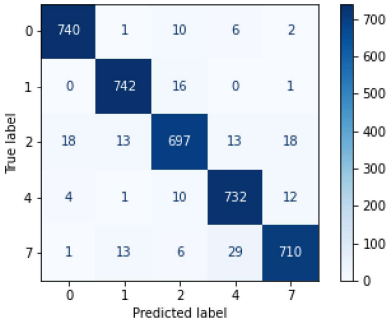
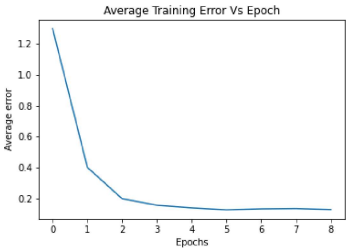
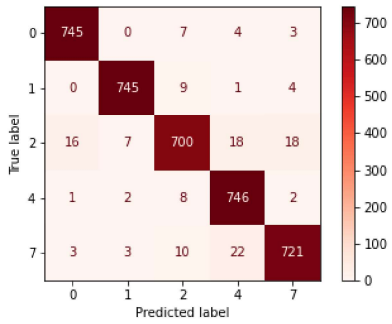
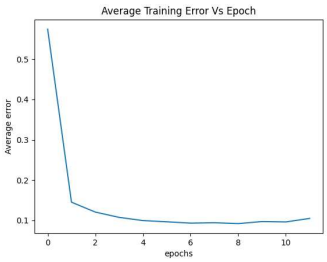
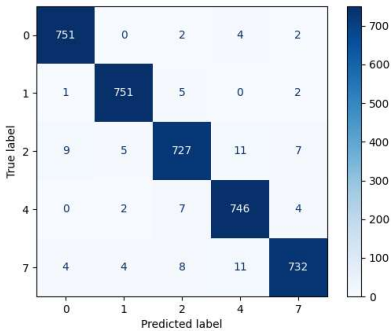


Fig 25:Confusion matrix for Adam

Best performance of optimizers in different architectures on the basis of validation accuracy.

Optimisers	Error V/S Epochs	Architecture	Confusion Matrix
SGD(Pattern mode) **Accuracy 98.52%**		3 Hidden Layers HL1-512 HL2-256 HL3-128	
SGD(Batch Mode) **Accuracy 96.34%**		4 Hidden Layers HL1-256 HL2-128 HL3-128 HL4-64	
SGD (Momentum) **Accuracy 97.29%**		4 Hidden Layers HL1-512 HL2-256 HL3-128 HL4-64	
NAG **Accuracy 96.89%**		3 Hidden Layers HL1-512 HL2-256 HL3-128	

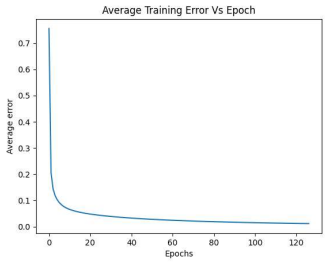
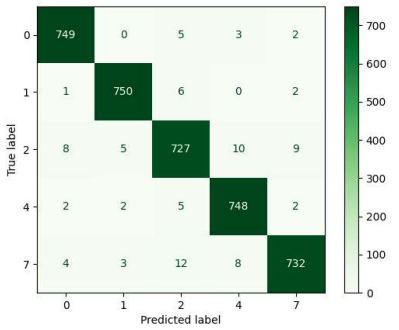
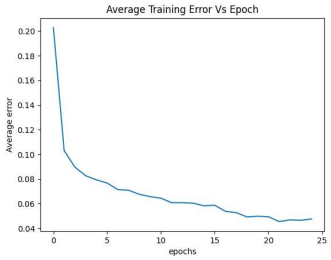
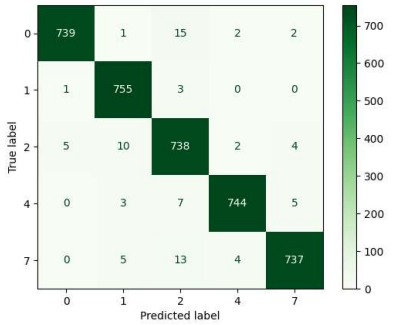
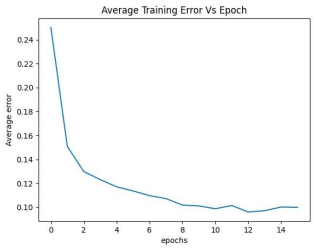
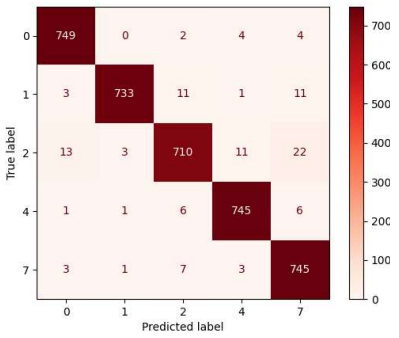
AdaGrad **Accuracy 98.81%**		3 Hidden Layers HL1-512 HL2-512 HL3-512	
RMSProp **Accuracy 98.18%**		3 Hidden Layers HL1-512 HL2-512 HL3-512	
Adam **Accuracy 97.52%**		3 Hidden Layers HL1-512 HL2-256 HL3-256	

Table 10: Best performance of optimizers across various architectures.