Report on

Deep Learning and Applications (CS671)

Assignment 3

Submitted by:

GROUP - 9

Shashank Kapoor (S22022)

Shubham Patwar (T22108)

Syed Rizwan Ali Quadri (T22113)

Table of Content

Topic Page No.

1.4.6. Neural Architecture (512,256,128,64,32) Configuration

1.5. Adagrad Optimiser

1.5.1. Neural Architecture (128,64,32) Configuration

1.5.2. Neural Architecture (256,128,64) Configuration

1.5.3. Neural Architecture (512,256,128) Configuration

1.5.4. **Neural Architecture (256,128,64,32) Configuration(BEST)**

1.5.5. Neural Architecture (512,256,128,64) Configuration

1.5.6. Neural Architecture (512,256,128,64,32) Configuration

1.6. RMSProp Optimiser

1.6.1. Neural Architecture (128,64,32) Configuration

1.6.2. Neural Architecture (256,128,64) Configuration

1.6.3. **Neural Architecture (512,256,128) Configuration(BEST)**

1.6.4. Neural Architecture (256,128,64,32) Configuration

1.6.5. Neural Architecture (512,256,128,64) Configuration

1.6.6. Neural Architecture (512,256,128,64,32) Configuration

1.7. Adam Optimiser

1.7.1. **Neural Architecture (128,64,32) Configuration(BEST)**

1.7.2. Neural Architecture (256,128,64) Configuration

1.7.3. Neural Architecture (512,256,128) Configuration

1.7.4. Neural Architecture (256,128,64,32) Configuration

1.7.5. Neural Architecture (512,256,128,64) Configuration

1.7.6. Neural Architecture (512,256,128,64,32) Configuration

Assignment Problem Statement

In this assignment , the key objective is to get inference from each optimiser on various neural architecture used to get the better accuracy on validation set.

We are using MNIST dataset for classification of the number from the 28X28 size images into 10 classes (0 to 9), for training the fully connected neural architecture to be used with different optimisers and getting the best neural architecture for each optimiser.

Overall Result:

Optimiser	Neural Architecture	No. of Epochs	Training Accuracy	Validation Accuracy
Stochastic Gradient Descent	**SGD_(128,64,32)**	**44**	**0.9797**	**0.9623**
	SGD_(256,128,64)	31	0.9771	0.9615
	SGD_(512,256,128)	29	0.9915	0.9599
	SGD_(256,128,64,32)	25	0.9731	0.9610
	SGD_(512,256,128,64)	25	0.9848	0.9544
	SGD_(512,256,128,64,32)	27	0.9875	0.9573
Vanilla Gradient Descent	VGD_(128,64,32)	3055	0.411	0.4076
	VGD_(256,128,64)	4297	0.5625	0.5683
	VGD_(512,256,128)	**4074**	**0.6835**	**0.6938**
	VGD_(256,128,64,32)	2322	0.3641	0.3683
	VGD_(512,256,128,64)	2627	0.3578	0.3667
	VGD_(512,256,128,64,32)	2131	0.3516	0.3681
Stochastic Gradient Descent with Momentum	SGDMom_(128,64,32)	7	0.9838	0.9649
	SGDMom_(256,128,64)	**11**	**0.9971**	**0.9733**
	SGDMom_(512,256,128)	3	0.9732	0.9570
	SGDMom_(256,128,64,32)	5	0.9814	0.9683
	SGDMom_(512,256,128,64)	11	0.9996	0.9681
	SGDMom_(512,256,128,64,32)	3	0.9675	0.9491
Nesterov	NAG_(128,64,32)	10	0.9931	0.9699

Gradient Descent	NAG_(256,128,64)	6	0.9852	0.9625
	NAG_(512,256,128)	7	0.9982	0.9691
	NAG_(256,128,64,32)	**8**	**0.9916**	**0.9712**
	NAG_(512,256,128,64)	5	0.9870	0.9583
	NAG_(512,256,128,64,32)	3	0.9689	0.9509
Adagrad	Adagrad_(128,64,32)	373	0.9719	0.9641
	Adagrad_(256,128,64)	277	0.9832	0.9662
	Adagrad_(512,256,128)	195	0.9953	0.9678
	Adagrad_(256,128,64,32)	**288**	**0.9823**	**0.9681**
	Adagrad_(512,256,128,64)	155	0.9923	0.9631
	Adagrad_(512,256,128,64,32)	203	0.9934	0.9699
RMS	RMS_(128,64,32)	4	0.9880	0.9778
	RMS_(256,128,64)	2	0.9752	0.9620
	RMS_(512,256,128)	**4**	**0.9833**	**0.9810**
	RMS_(256,128,64,32)	4	0.9865	0.9786
	RMS_(512,256,128,64)	4	0.9823	0.9694
	RMS_(512,256,128,64,32)	3	0.9768	0.9762
Adam	**Adam_(128,64,32)**	**3**	**0.9856**	**0.9778**
	Adam_(256,128,64)	5	0.9903	0.9828
	Adam_(512,256,128)	4	0.9782	0.9778
	Adam_(256,128,64,32)	2	0.9704	0.9696
	Adam_(512,256,128,64)	6	0.9868	0.9736
	Adam_(512,256,128,64,32)	2	0.9610	0.9422

* Mark in bold is Best architecture according to the validation accuracy

Superimpose plot for the each optimiser:

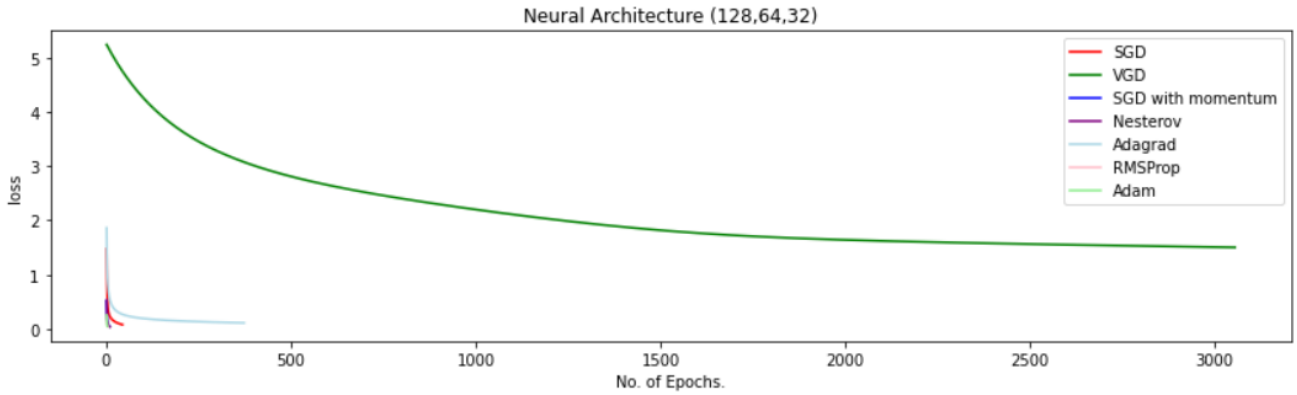


Fig.1: Superimpose plot of loss vs epochs for each optimiser for architecture 128,64,32

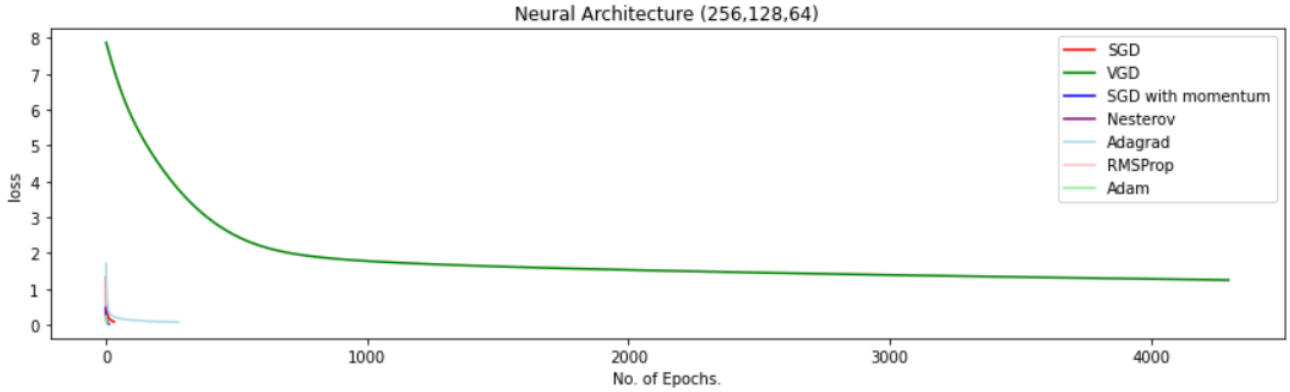


Fig.2: Superimpose plot of loss vs epochs for each optimiser for architecture 256,128,64

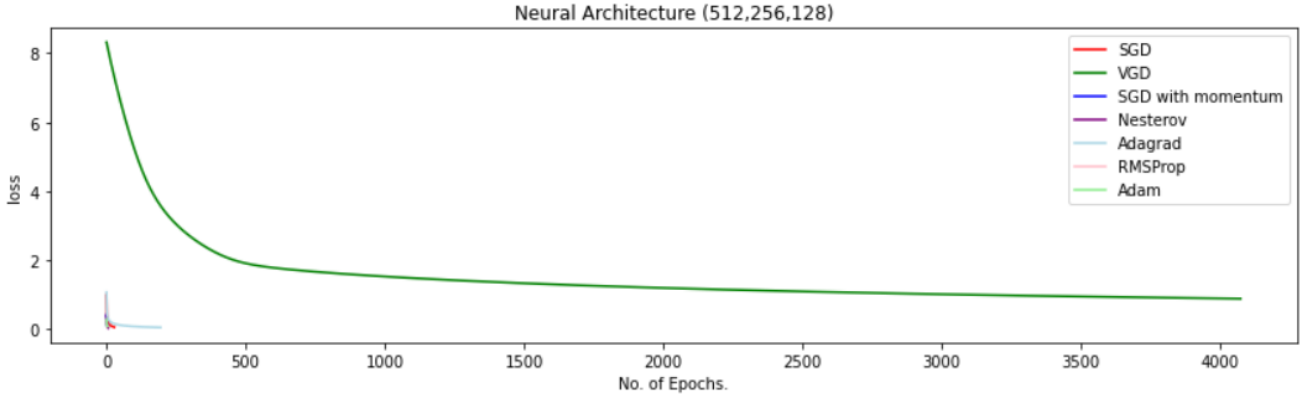


Fig.3: Superimpose plot of loss vs epochs for each optimiser for architecture 512,256,128

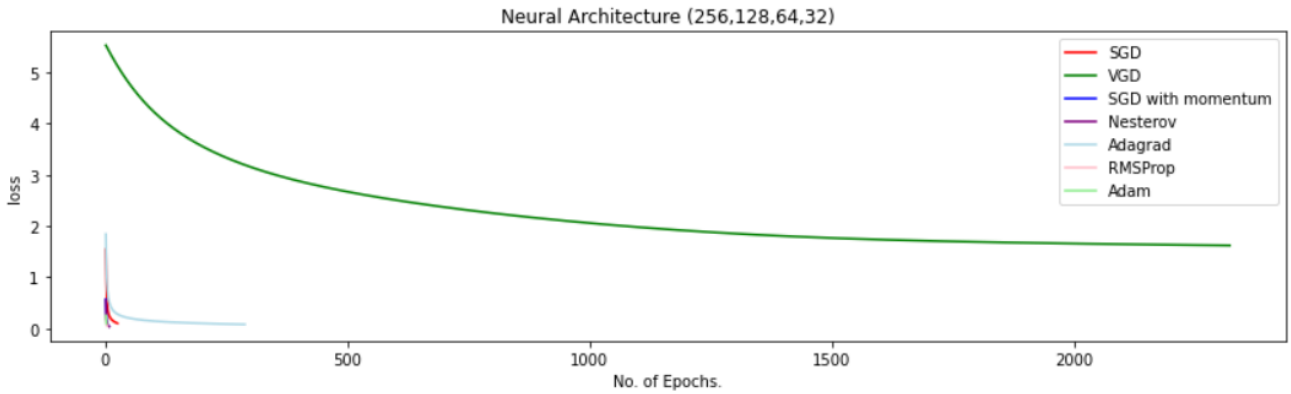


Fig.4: Superimpose plot of loss vs epochs for each optimiser for architecture 256,128,64,32

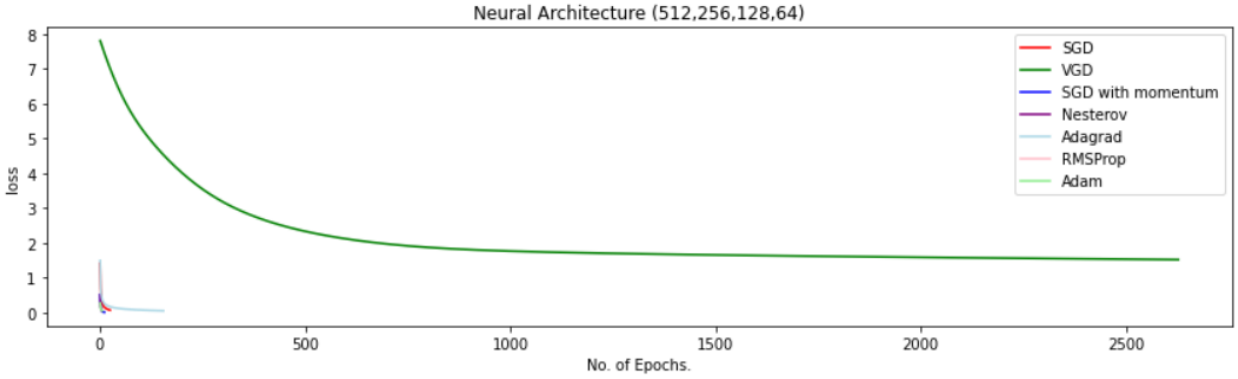


Fig.5: Superimpose plot of loss vs epochs for each optimiser for architecture 512,256,128,64

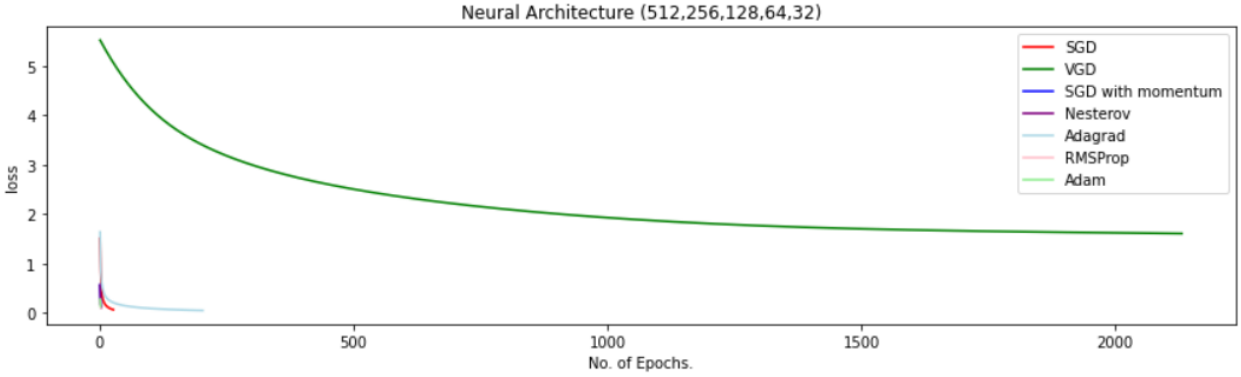


Fig.6: Superimpose plot of loss vs epochs for each optimiser for architecture 512,256,128,64,32

1. Introduction to optimiser

An optimizer is an algorithm or method used to find the optimal values for the parameters of a mathematical function, typically used in deep learning models. It is basically an optimization method that helps improve a deep learning model's performance. These optimization algorithms or optimizers widely affect the accuracy and speed training of the deep learning model.

The primary goal of an optimizer is to minimize the loss function, which represents the difference between the predicted output and the actual output of a model.

We train models by adjusting the parameters based on the input data and the desired output. Optimizers help us find the best set of parameters that can minimize the error between the predicted output and the actual output.

There are many different optimizers available, each with their own strengths and weaknesses. Some of the most commonly used optimizers include;

- Stochastic Gradient Descent
- Vanilla Gradient Descent
- Stochastic Gradient Descent with momentum
- Nastrov
- Adagrad
- RMSprop
- Adam.

These optimizers differ in terms of how they calculate the gradients and update the parameters.

The choice of optimizer can have a significant impact on the performance of a model. A good optimizer should be able to converge quickly to the optimal values while avoiding getting stuck in local minima. It should also be able to handle noise and fluctuations in the input data without overfitting or underfitting the model.

1.1. Stochastic Gradient Descent:

Stochastic Gradient Descent (SGD) is a popular optimization algorithm used in machine learning to minimize the cost or loss function. It is a variant of Gradient Descent (GD), but instead of computing the gradient over the entire dataset, SGD updates the model parameters based on a single sample. This makes it more computationally efficient and faster than GD.

The basic idea behind SGD is to iteratively update the model parameters based on the gradient of the loss function with respect to the current parameters.

The update rule is given by:

$$W = W - (\square * \nabla W)$$

Where,

W is the vector of model parameters (weights),

□ is the learning rate ($0 < \square < 1$)

$$\nabla w = \frac{\partial E_n}{\partial w}$$

$\nabla \mathbf{W}$ is the gradient of the loss function with respect to W, given as

In other words, for each sample in the training dataset, we compute the gradient of the loss function with respect to the model parameters (weights), and then update the parameters by taking a step in the opposite direction of the gradient with a certain learning rate □. As shown in the figure,

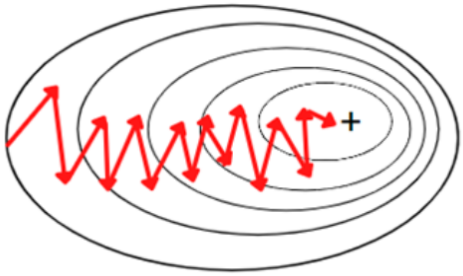


Fig. 7 : Convergence of Stochastic Gradient Descent

SGD is a widely used optimization algorithm, but it has some limitations. It can get stuck in local minima or saddle points, and it may take longer to converge to the optimal solution compared to other optimization algorithms such as Adam and Adagrad. However, it is still an important algorithm to know and understand in the context of machine learning.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.1.1. Neural Architecture (128,64,32) Configuration (BEST):

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes
  - Hidden Layer 2 : 64 nodes
  - Hidden Layer 3 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **44** epochs

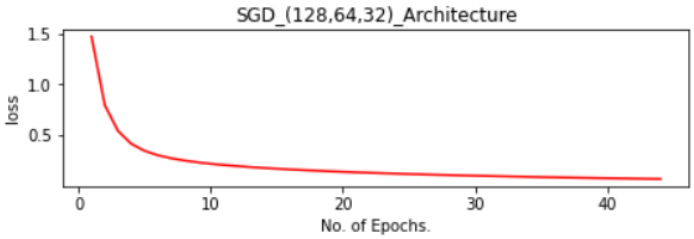
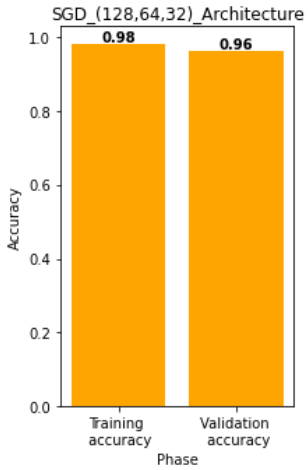


Fig.8: loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9797
- Validation Accuracy : 0.9623

Fig.9: Bar plot between training and validation accuracy

		Actual Label									
		0	1	2	3	4	5	6	7	8	9
Predicted Label	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	707	0	17	0	8	11	16	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	9	0	732	0	5	6	7	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	8	0	9	0	731	1	10	0
	7	0	0	13	0	11	0	2	727	6	0
	8	0	0	17	0	10	0	7	7	718	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy :0.9526

Fig. Confusion Matrix of the Best Possible SGD Architecture

1.1.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **31** epochs

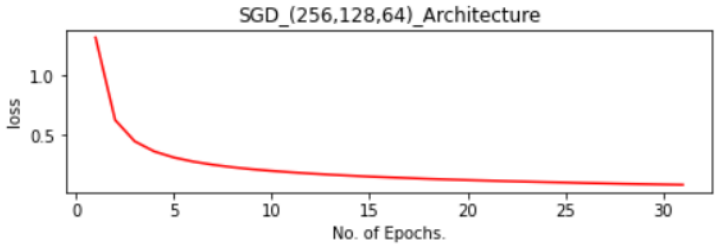
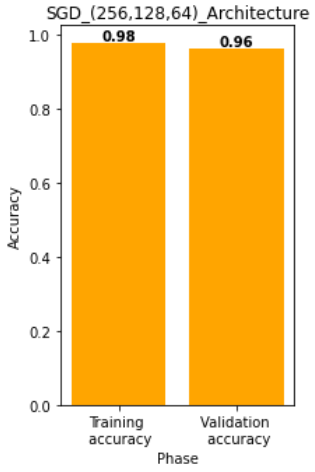


Fig.10: loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9771
- Validation Accuracy : 0.9615

Fig.11: Bar plot between training and validation accuracy

1.1.3. Neural Architecture (512,256,128) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **29** epochs

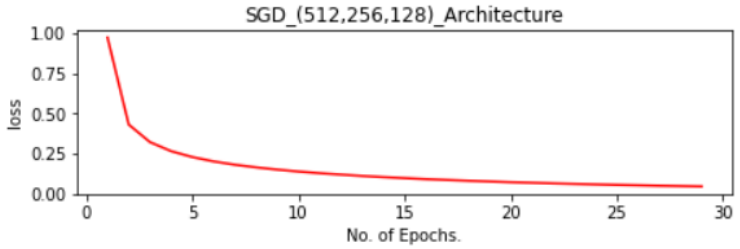
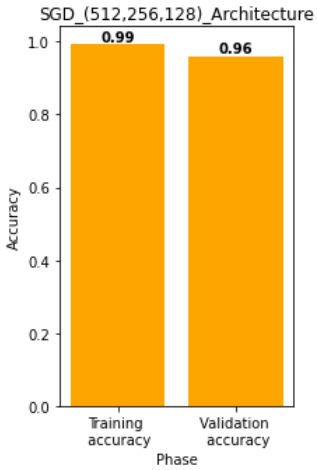


Fig.12: loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9915
- Validation Accuracy : 0.9599

Fig.13: Bar plot between training and validation accuracy

1.1.4. Neural Architecture (256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **25** epochs

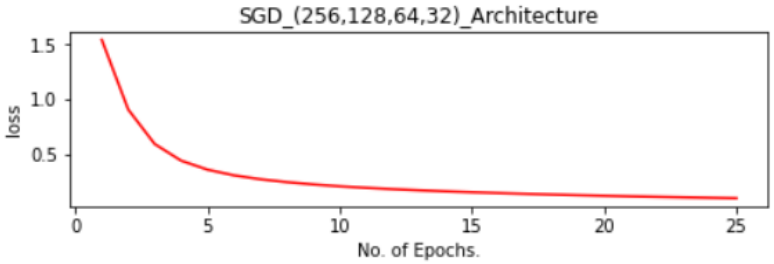
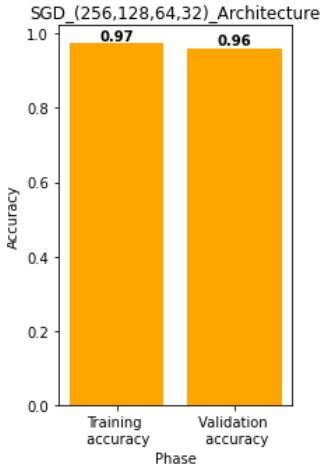


Fig.14: loss vs epochs for training data with architecture 256,128,64,32



- Training Accuracy : 0.9731
- Validation Accuracy : 0.9610

Fig.15: Bar plot between training and validation accuracy

1.1.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **25** epochs

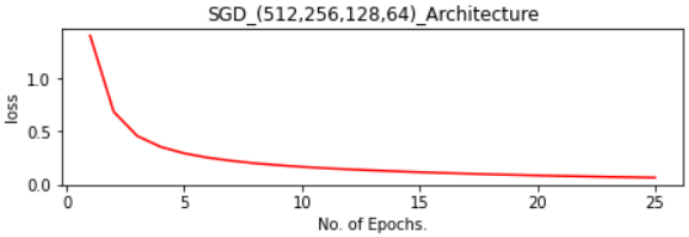
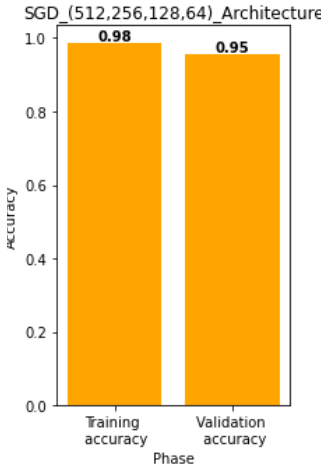


Fig.16:loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9848
- Validation Accuracy : 0.9544

Fig.17: Bar plot between training and validation accuracy

1.1.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4: 64 nodes
  - Hidden Layer 5: 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **27** epochs

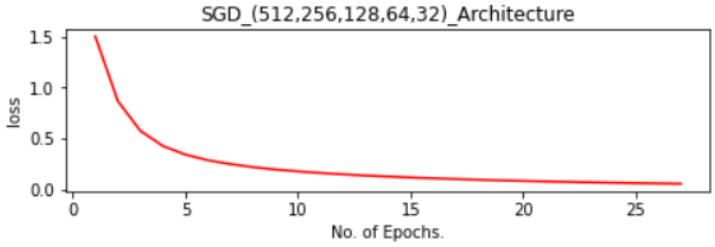
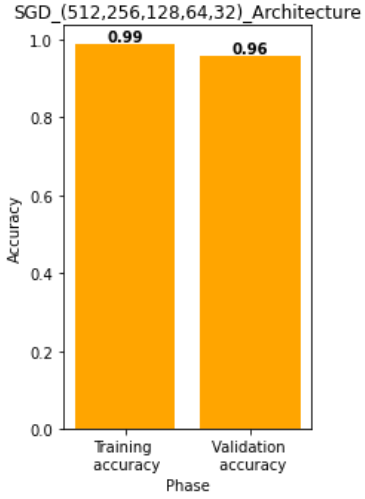


Fig.18: loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9875
- Validation Accuracy : 0.9573

Fig.19: Bar plot between training and validation accuracy

1.2. Vanilla Gradient Descent:

Vanilla Gradient Descent (also known as Batch Gradient Descent) is a popular optimization algorithm used in machine learning to find the optimal parameters of a model. The algorithm works by collecting Gradient values over an entire epoch and the model parameters (weights) are updated at once.

In other words, it calculates the gradient of the cost function with respect to the model parameters, and then updates the parameters by subtracting a fraction of this gradient, multiplied by the learning rate □, from the current parameter values. The learning rate determines the size of the step taken in the direction of the gradient. Weight change is decided using single resultant (global) gradient values.

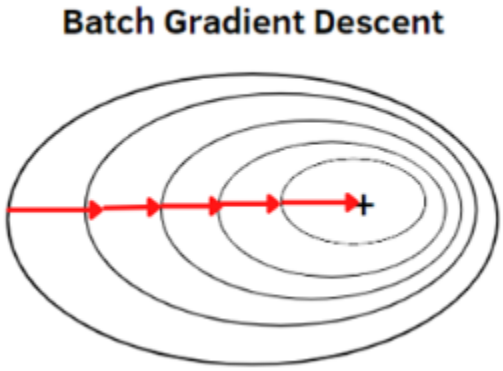


Fig. 20:Convergence of Batch/Vanilla Gradient Descent

The update rule for vanilla gradient descent can be written as:

$$W = W - (□ * \nabla W)$$

Where,

W is the vector of model parameters (weights),
□ is the learning rate (0 < □ < 1),
∇**W** is the gradient of the loss function with respect to W, given as $\nabla w = \frac{\partial E_n}{\partial w}$

Vanilla gradient descent has some limitations. One of the main challenges is that it can be slow to converge, especially when dealing with large datasets or complex models. Additionally, it is sensitive to the choice of learning rate, which can affect the convergence and stability of the algorithm. Finally, it is prone to getting stuck in local minima.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.2.1. Neural Architecture (128,64,32) Configuration:

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes

- Hidden Layer 2 : 64 nodes
- Hidden Layer 3 : 32 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3055** epochs

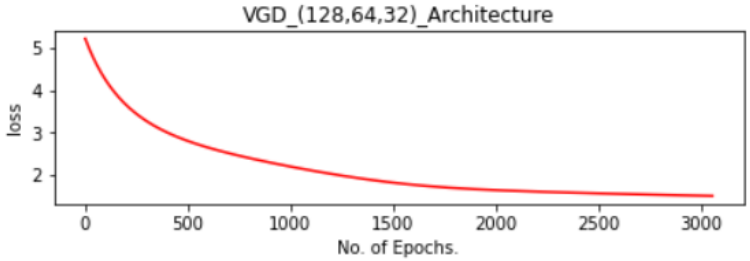
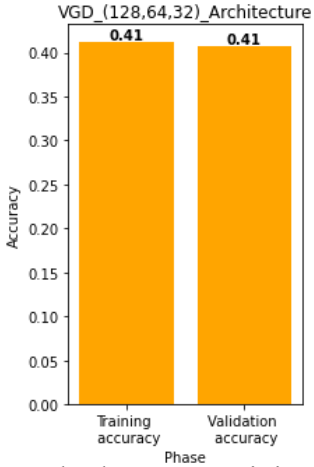


Fig.21: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.4115
- Validation Accuracy : 0.4076

Fig.22: Bar plot between training and validation accuracy

1.2.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes

    - Hidden Layer 3 : 64 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4297** epochs

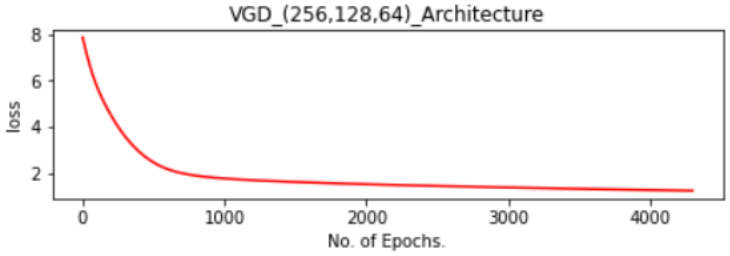
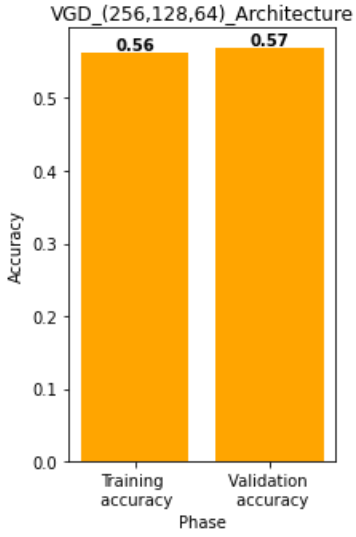


Fig.23: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.5625
- Validation Accuracy : 0.5683

Fig.24: Bar plot between training and validation accuracy

1.2.3. Neural Architecture (512,256,128) Configuration (BEST)

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4074** epochs

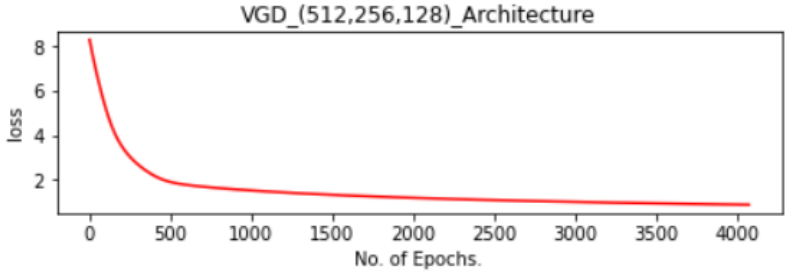
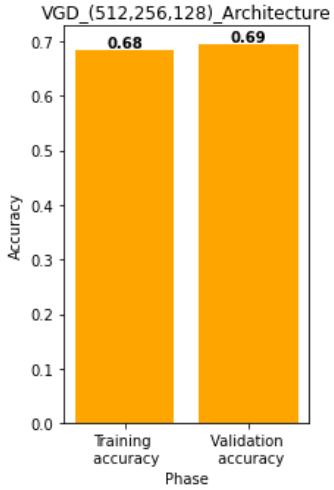


Fig.25: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.6835
- Validation Accuracy : 0.6938

Fig.26: Bar plot between training and validation accuracy

		Actual Label									
Predicted Label		0	1	2	3	4	5	6	7	8	9
	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	464	0	71	0	95	29	100	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	73	0	462	0	107	65	52	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	81	0	76	0	554	15	33	0
	7	0	0	41	0	72	0	18	570	58	0
	8	0	0	86	0	55	0	22	61	535	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy : 0.6811

Fig. Confusion Matrix of the Best Possible VGD Architecture

1.2.4. Neural Architecture (256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2322** epochs

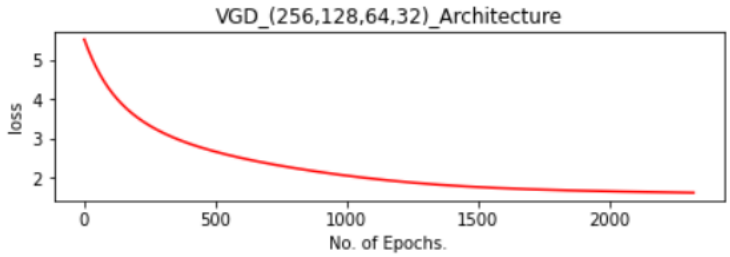
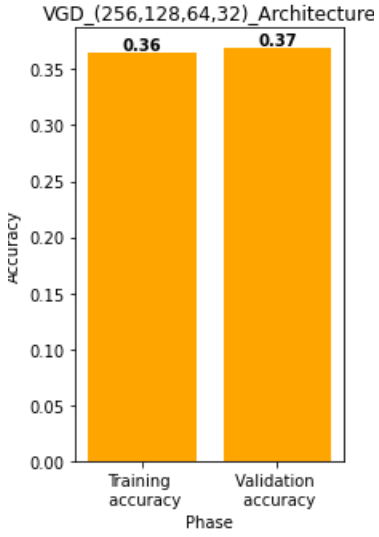


Fig.27: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.3641
- Validation Accuracy : 0.3683

Fig.28: Bar plot between training and validation accuracy

1.2.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points

- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2627** epochs

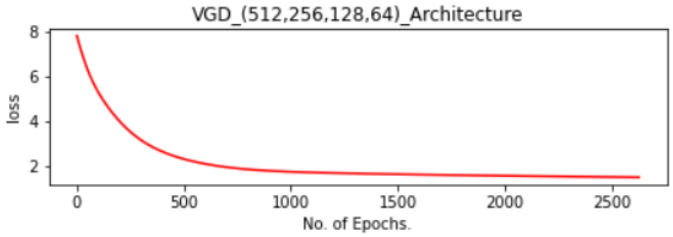
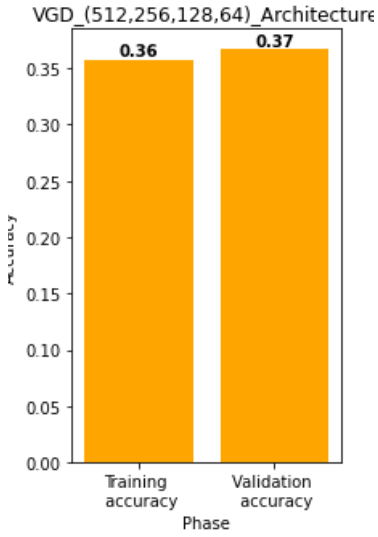


Fig.29: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.3578
- Validation Accuracy : 0.3667

Fig.30: Bar plot between training and validation accuracy

1.2.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
    - Input Layer : 784 features
    - Hidden Layer 1 : 512 nodes
    - Hidden Layer 2 : 256 nodes
    - Hidden Layer 3 : 128 nodes

    - Hidden Layer 4: 64 nodes
    - Hidden Layer 5: 32 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2131** epochs

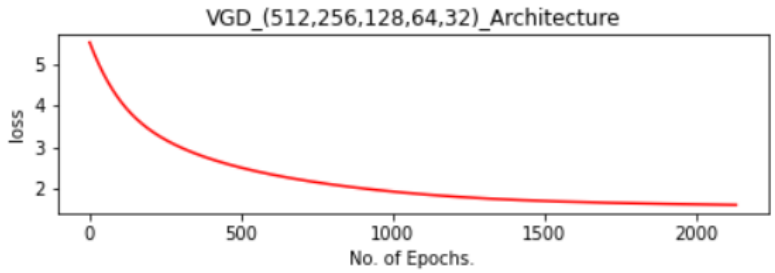
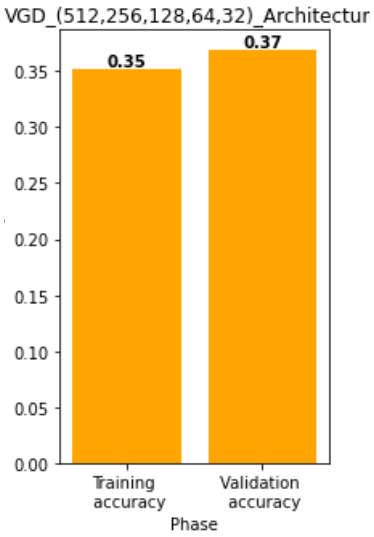


Fig.31: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.3516
- Validation Accuracy : 0.3681

Fig.32: Bar plot between training and validation accuracy

1.3. Stochastic Gradient Descent with momentum (Generalized delta rule):

Stochastic Gradient Descent with momentum is an optimization algorithm used in machine learning for finding the optimal parameters of a model. It is an algorithm that includes a momentum term to help the algorithm converge faster and avoid getting stuck in local minima. The intuition behind this algorithm is that if we repeatedly move in the same direction then we probably gain some confidence and start taking bigger steps in that direction.

The momentum term is a moving average of the gradients of the objective function with respect to the parameters. The idea behind momentum is to keep track of the direction of the previous gradients and use this information to update the parameters in a more consistent direction.

The update rule for Stochastic Gradient Descent with momentum can be expressed as follows:

$$W(m+1) = W(m) - (\square * \nabla W(m)) + (\alpha * \Delta W(m-1))$$

Where,

W(m+1) is the new vector of model parameters (weights),
W(m) is the present vector of model parameters (weights),
□ is the learning rate (0 < □ < 1),
∇ **W(m)** is the gradient of the loss function with respect to W, given as $\nabla w = \frac{\partial E_n}{\partial w}$
α is the momentum (0 < α < 1),
ΔW(m-1) is the history of model parameters (weights).

The momentum term is updated using a hyperparameter α, which typically has a value between 0.9 and 0.99. A larger value of α gives more weight to the previous gradients and results in a smoother update trajectory.

Stochastic Gradient Descent with momentum is particularly useful for optimizing models with high-dimensional parameter spaces or models with noisy gradients. The momentum term helps the algorithm move more efficiently through the parameter space and reduces the effect of noisy gradients.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.3.1. Neural Architecture (128,64,32) Configuration:

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes

- Hidden Layer 2 : 64 nodes
- Hidden Layer 3 : 32 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **7** epochs

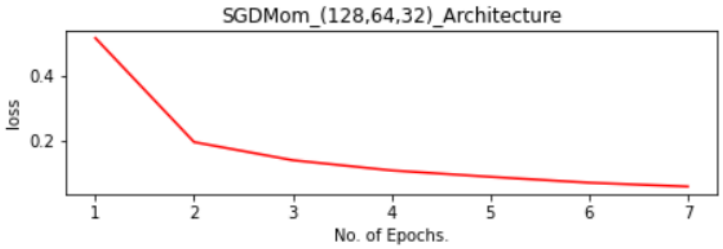
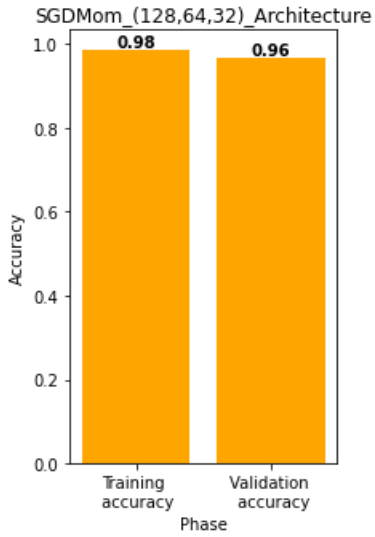


Fig.33: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9838
- Validation Accuracy : 0.9649

Fig.34: Bar plot between training and validation accuracy

1.3.2. Neural Architecture (256,128,64) Configuration (BEST)

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:

    - Input Layer : 784 features
    - Hidden Layer 1 : 256 nodes
    - Hidden Layer 2 : 128 nodes
    - Hidden Layer 3 : 64 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

-Total number of epochs needed to reach threshold **11** epochs

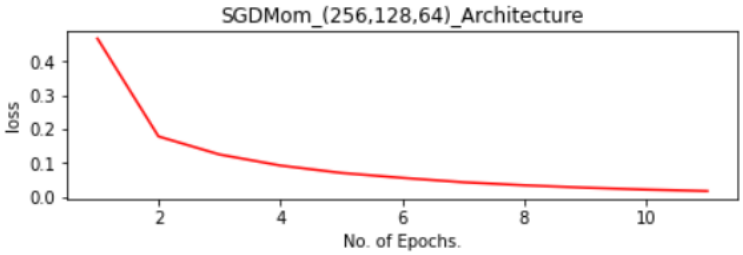
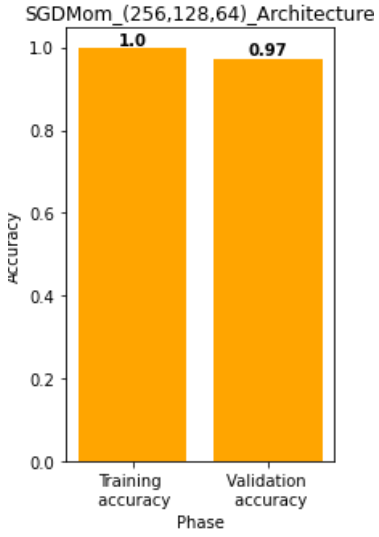


Fig.35: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9971
  Validation Accuracy : 0.9733

Fig.36: Bar plot between training and validation accuracy

		Actual Label									
		0	1	2	3	4	5	6	7	8	9
Predicted Label	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	717	0	13	0	3	8	18	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	6	0	742	0	3	6	2	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	6	0	6	0	740	0	7	0
	7	0	0	7	0	12	0	2	734	4	0
	8	0	0	10	0	12	0	6	6	725	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy: 0.9639

Fig. Confusion Matrix of the Best Possible SGD with Momentum Architecture

1.3.3. Neural Architecture (512,256,128) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

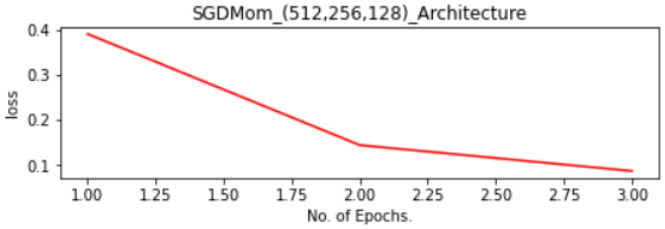
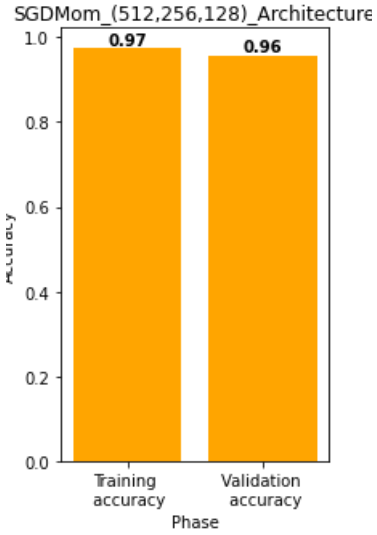


Fig.37: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9971
  Validation Accuracy : 0.9733

Fig.38: Bar plot between training and validation accuracy

1.3.4. Neural Architecture (256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points

- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **5** epochs

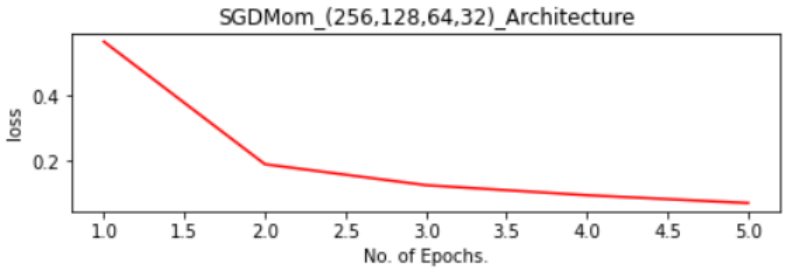
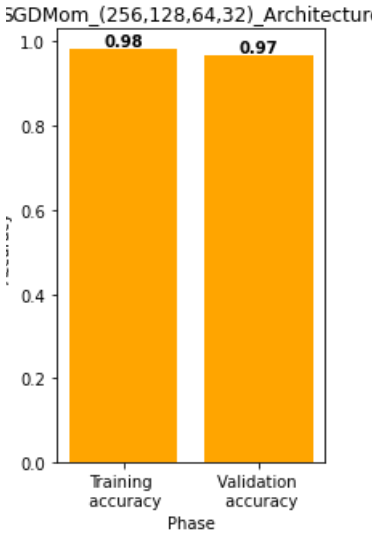


Fig.39: Loss vs epochs for training data with architecture 256,128,64,32



- Training Accuracy : 0.9814
-Validation Accuracy : 0.9683

Fig.40: Bar plot between training and validation accuracy

1.3.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
    - Input Layer : 784 features
    - Hidden Layer 1 : 512 nodes
    - Hidden Layer 2 : 256 nodes

- Hidden Layer 3 : 128 nodes
- Hidden Layer 4 : 64 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **11** epochs

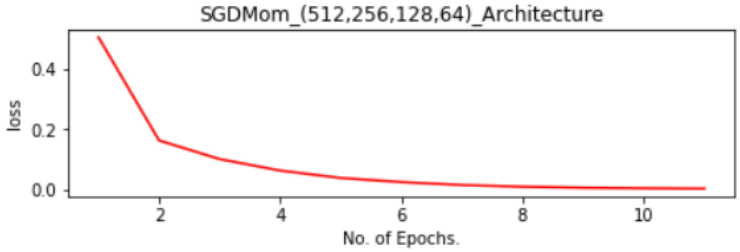
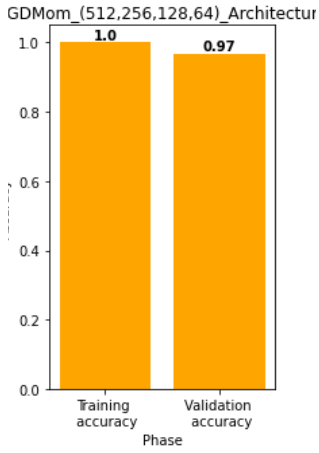


Fig.41: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9814
-Validation Accuracy : 0.9683

Fig.42: Bar plot between training and validation accuracy

1.3.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes

    - Hidden Layer 2 : 256 nodes
    - Hidden Layer 3 : 128 nodes
    - Hidden Layer 4: 64 nodes
    - Hidden Layer 5: 32 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

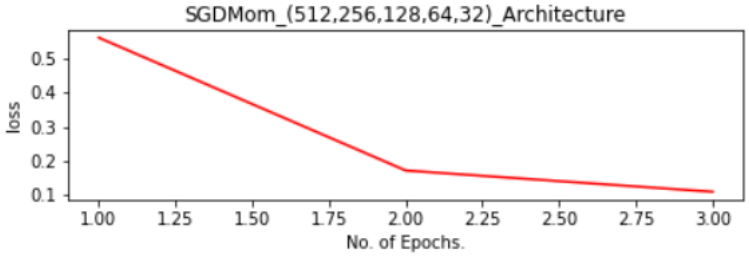
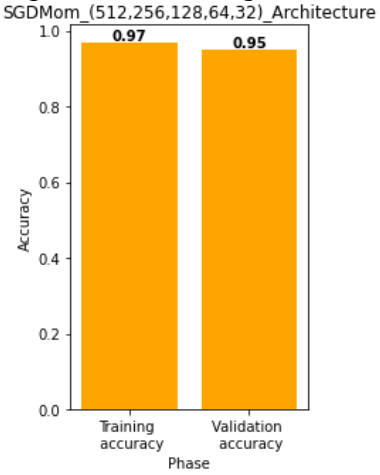


Fig.43: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.9971
- Validation Accuracy : 0.9733

Fig.44: Bar plot between training and validation accuracy

1.4. Stochastic Gradient Descent with momentum (Nesterov Accelerated Gradient or NAG):

Nesterov Momentum (also known as Nesterov Accelerated Gradient or NAG) is a variant of the popular Momentum optimizer that aims to improve the convergence speed and stability of the optimization process in deep learning models.

The standard Momentum optimizer uses a moving average of the past gradients to update the model's parameters. In contrast, Nesterov Momentum adds an extra term to the standard momentum update that takes into account the momentum's contribution in the direction of the current gradient. This allows the optimizer to more accurately anticipate the next set of model parameters and adjust the momentum accordingly.

The update rule for Stochastic Gradient Descent with momentum can be expressed as follows:

$$W_{Lookahead} = W(m) + \alpha * \Delta W(m-1)$$
$$W(m+1) = W(m) - (\square * \nabla W_{lookahead}) + (\alpha * \Delta W(m-1))$$

Where,

W(m+1) is the new vector of model parameters (weights),
W(m) is the present vector of model parameters (weights),
□ is the learning rate (0 < □ < 1),
$\mathbf{W_{Lookhead}}$ is a look ahead point is considered as a partial update,
$\nabla \mathbf{W_{lookahead}}$ is a gradient at look ahead point.
$\nabla \mathbf{W(m)}$ is the gradient of the loss function with respect to W, given as $\nabla w = \frac{\partial E_n}{\partial w}$
α is the momentum (0 < α < 1),
ΔW(m-1) is the history of model parameters (weights).

Nesterov Momentum has been shown to work well in many deep learning applications, particularly those with sparse or noisy gradients. It can help accelerate convergence, reduce oscillations, and improve the overall performance of the model.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.4.1. Neural Architecture (128,64,32) Configuration:

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes
  - Hidden Layer 2 : 64 nodes

- Hidden Layer 3 : 32 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **10** epochs

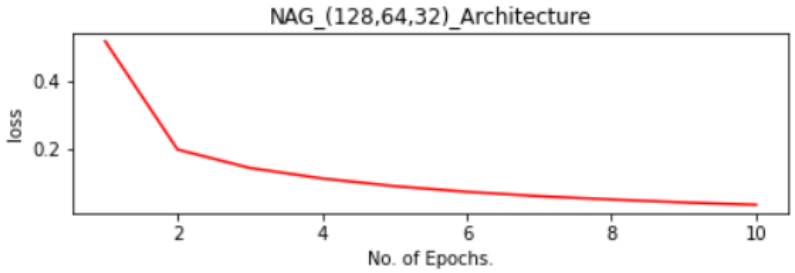
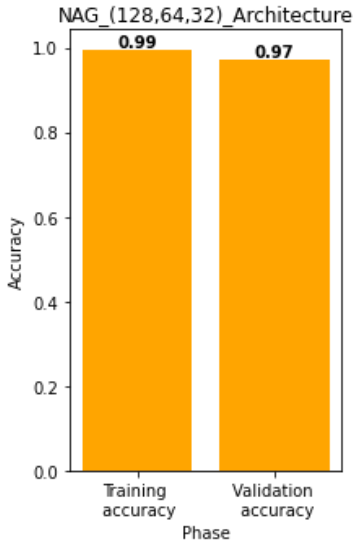


Fig.45: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9931
-Validation Accuracy : 0.9699

Fig.46: Bar plot between training and validation accuracy

1.4.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:

- Input Layer : 784 features
- Hidden Layer 1 : 256 nodes
- Hidden Layer 2 : 128 nodes
- Hidden Layer 3 : 64 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **6** epochs

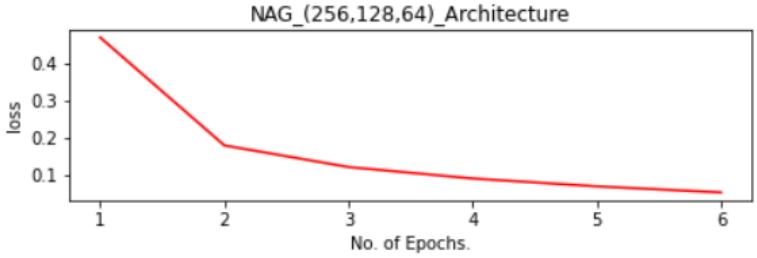
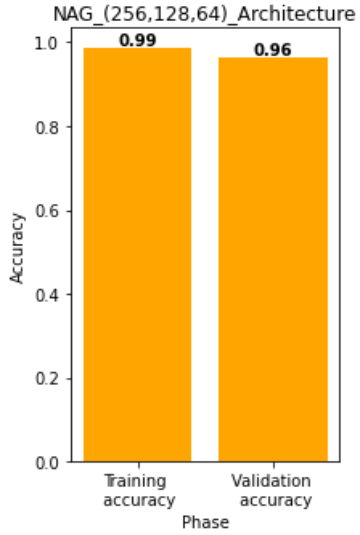


Fig.47: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9852
-Validation Accuracy : 0.9625

Fig.48: Bar plot between training and validation accuracy

1.4.3. Neural Architecture (512,256,128) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **7** epochs

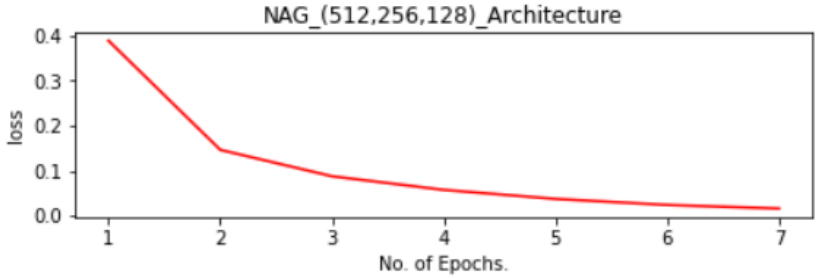
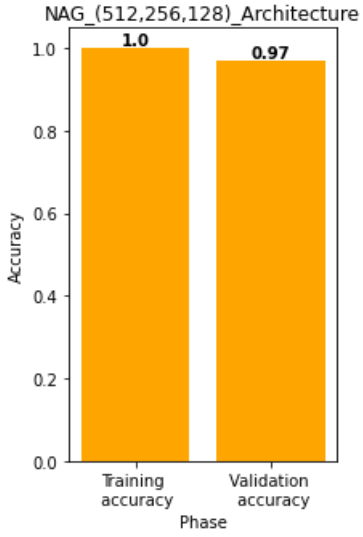


Fig.49: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9982
-Validation Accuracy : 0.9691

Fig.50: Bar plot between training and validation accuracy

1.4.4. Neural Architecture (256,128,64,32) Configuration (BEST)

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **8** epochs

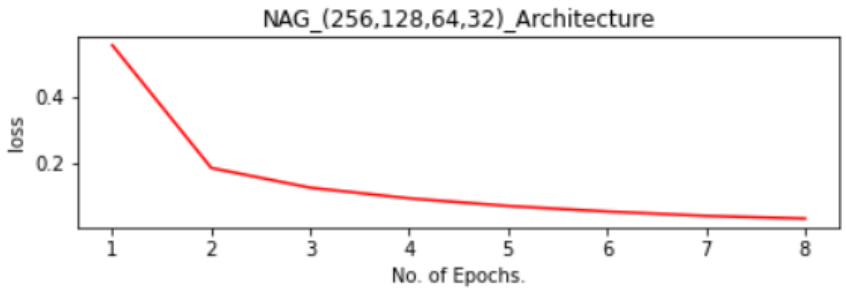
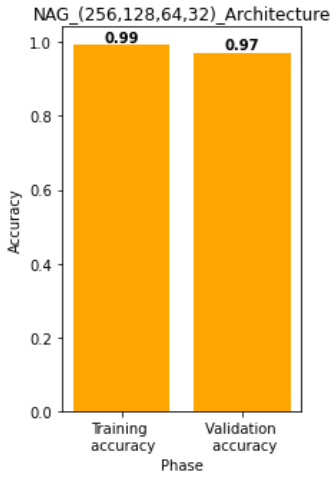


Fig.51: Loss vs epochs for training data with architecture 256,128,64,32



- Training Accuracy : 0.9916
-Validation Accuracy : 0.9712

Fig.52: Bar plot between training and validation accuracy

		Actual Label									
		0	1	2	3	4	5	6	7	8	9
Predicted Label	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	722	0	9	0	8	6	14	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	10	0	741	0	1	4	3	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	13	0	5	0	736	0	5	0
	7	0	0	18	0	11	0	2	724	4	0
	8	0	0	22	0	5	0	7	3	722	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy : 0.9604

Fig. Confusion Matrix of the Best Possible NAG Architecture

1.4.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **5** epochs

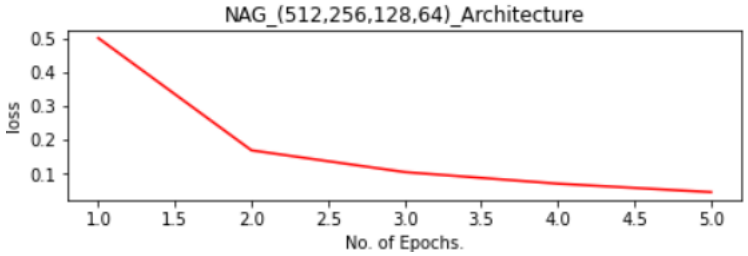
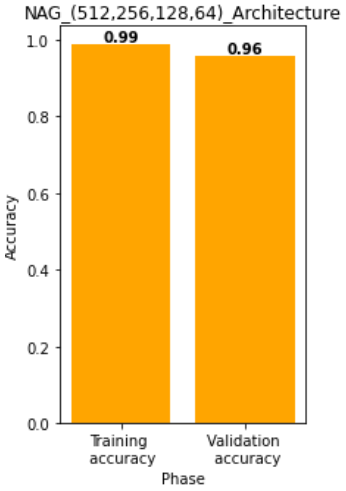


Fig.53: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.98700
-Validation Accuracy : 0.9583

Fig.54: Bar plot between training and validation accuracy

1.4.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4: 64 nodes
  - Hidden Layer 5: 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points

- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

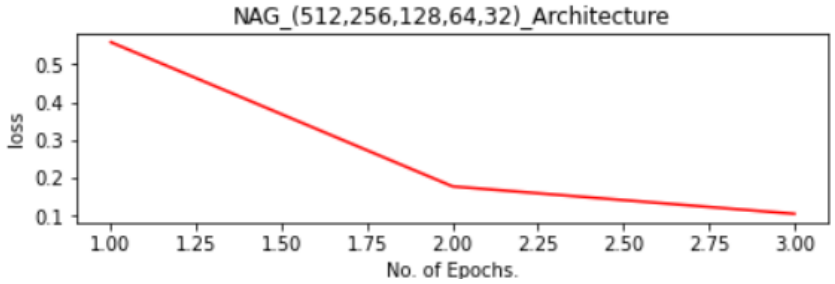
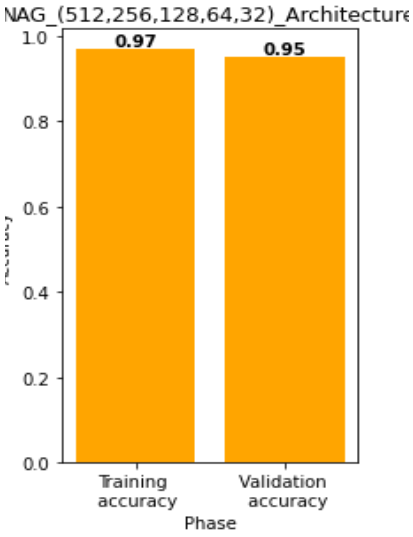


Fig.55: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.9689
-Validation Accuracy : 0.9509

Fig.56: Bar plot between training and validation accuracy

1.5. Adagrad:

Adagrad, short for Adaptive Gradient, is a gradient-based optimization algorithm for training machine learning models. It is an extension of stochastic gradient descent (SGD) that adapts the learning rate for each parameter based on its historical gradient information.

In standard SGD, the learning rate is a fixed parameter that is set in advance and used to update all model parameters equally. However, this approach can be suboptimal when the gradients of some parameters are much larger than others. This can cause the learning rate to be too small for some parameters, resulting in slow convergence, while too large for others, leading to oscillations and instability.

Adagrad addresses this issue by scaling the learning rate for each parameter based on its past gradient history. Specifically, Adagrad divides the learning rate by the square root of the sum of the squared gradients for each parameter up to the current iteration. This scaling results in a larger learning rate for parameters with smaller gradients and a smaller learning rate for those with larger gradients.

The update rule for Adagrad can be expressed as follows:

$$v(m) = v(m-1) + (\nabla w(m))^2$$

$$W(m+1) = W(m) - \left(\frac{\square}{\sqrt{v(m) + \varepsilon}} * \nabla W(m)\right)$$

Where,

v(m) is the update history variable (running estimate of square of the gradient of current weight)
W(m+1) is the new vector of model parameters (weights),
W(m) is the present vector of model parameters (weights),
□ is the learning rate (0 < □ < 1),
∇ **W(m)** is the gradient of the loss function with respect to W, given as $\nabla W = \frac{\partial E_n}{\partial w}$
ε is small positive value such as 10^{-8}

One potential drawback of Adagrad is that the sum of the squared gradients in the denominator can become very large over time, leading to a very small effective learning rate. To address this issue, later optimization algorithms like Adadelta and RMSprop were proposed as improvements on Adagrad.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.5.1. Neural Architecture (128,64,32) Configuration:

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes

- Hidden Layer 2 : 64 nodes
- Hidden Layer 3 : 32 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **373** epochs

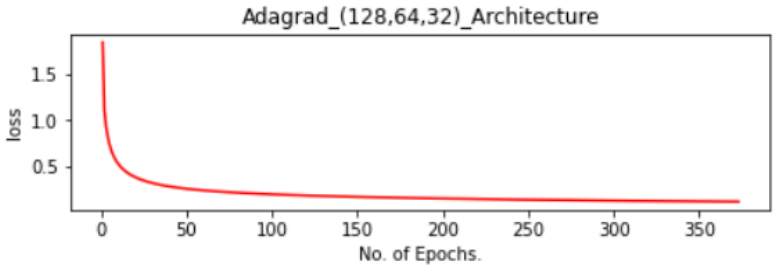
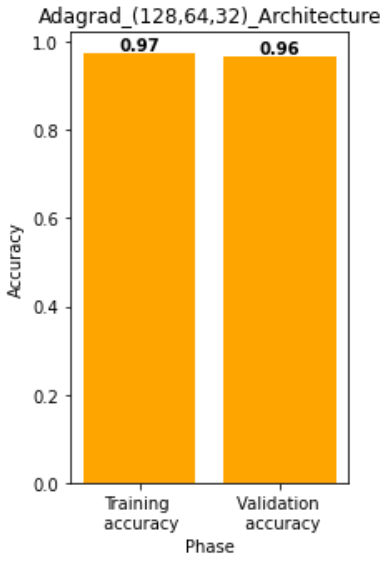


Fig.57: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9719
-Validation Accuracy : 0.9641

Fig.58: Bar plot between training and validation accuracy

1.5.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:

    - Input Layer : 784 features
    - Hidden Layer 1 : 256 nodes
    - Hidden Layer 2 : 128 nodes
    - Hidden Layer 3 : 64 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **277** epochs

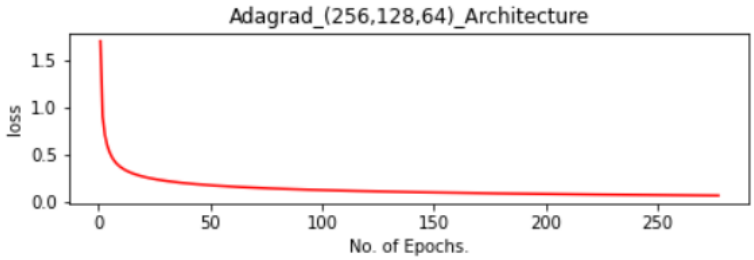
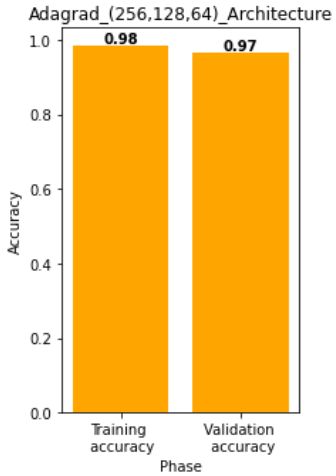


Fig.59: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9832
-Validation Accuracy : 0.9662

Fig.60: Bar plot between training and validation accuracy

1.5.3. Neural Architecture (512,256,128) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000
  -

Results:

- Total number of epochs needed to reach threshold **195** epochs

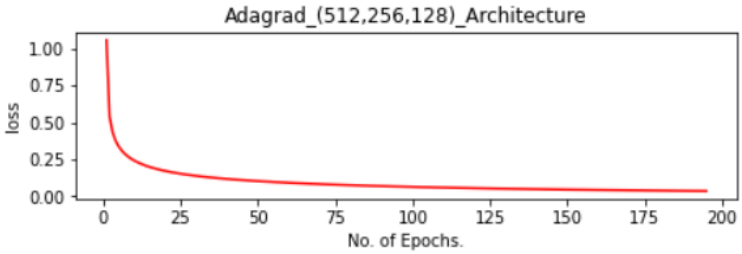
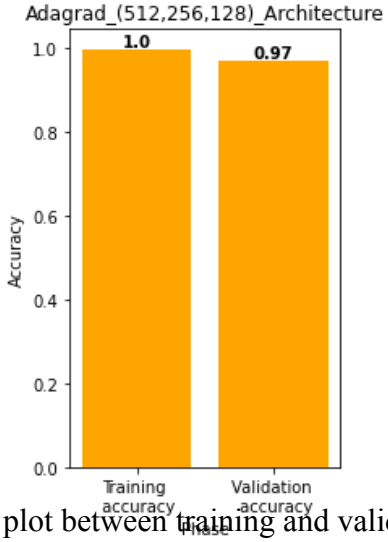


Fig.61: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9953
-Validation Accuracy : 0.9678

Fig.62: Bar plot between training and validation accuracy

1.5.4. Neural Architecture (256,128,64,32) Configuration (BEST)

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
    - Input Layer : 784 features
    - Hidden Layer 1 : 256 nodes
    - Hidden Layer 2 : 128 nodes
    - Hidden Layer 3 : 64 nodes
    - Hidden Layer 4 : 32 nodes
    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **288** epochs

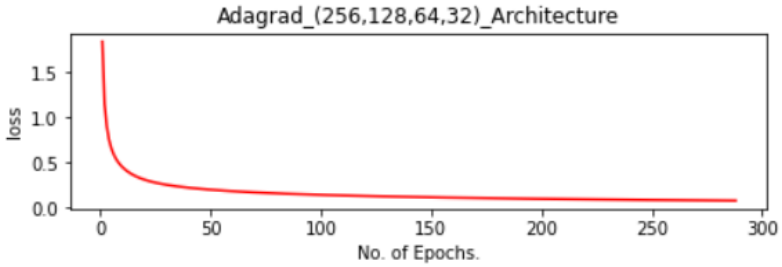
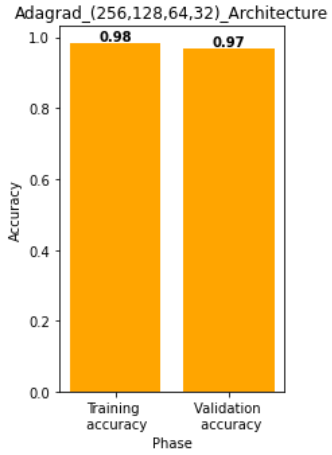


Fig.63: Loss vs epochs for training data with architecture 256,128,64,32



- Training Accuracy : 0.9823
-Validation Accuracy : 0.9681

Fig.64: Bar plot between training and validation accuracy

		Actual Label									
		0	1	2	3	4	5	6	7	8	9
Predicted Label	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	703	0	14	0	6	13	23	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	11	0	734	0	3	8	3	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	10	0	4	0	734	1	10	0
	7	0	0	13	0	5	0	3	735	3	0
	8	0	0	17	0	8	0	7	5	722	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy : 0.9560

Fig. Confusion Matrix of the Best Possible Adagrad Architecture

1.5.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **155** epochs

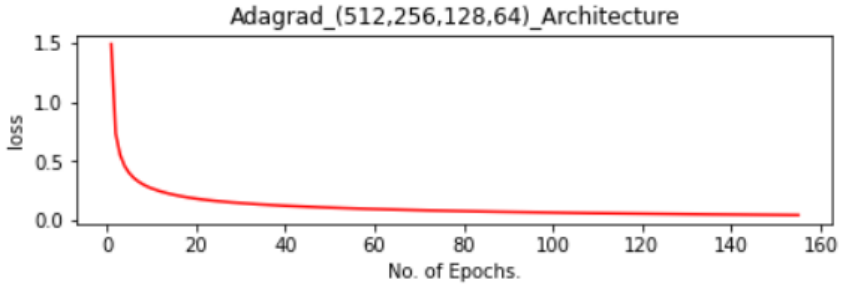
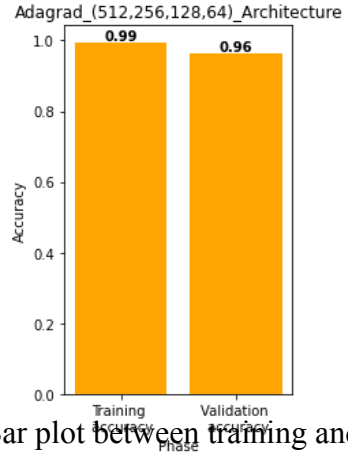


Fig.64: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9923
-Validation Accuracy : 0.9631

Fig.66: Bar plot between training and validation accuracy

1.5.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4: 64 nodes
  - Hidden Layer 5: 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001

- Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **203** epochs

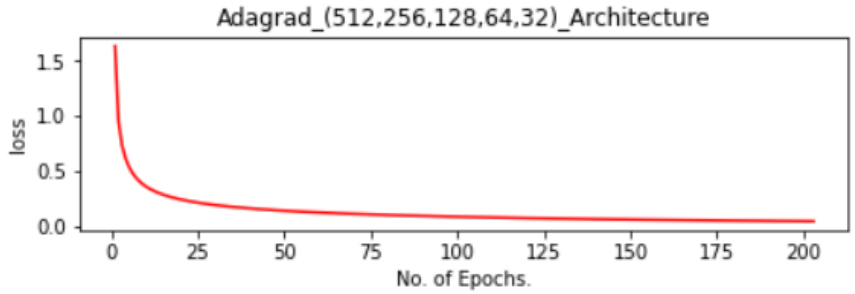
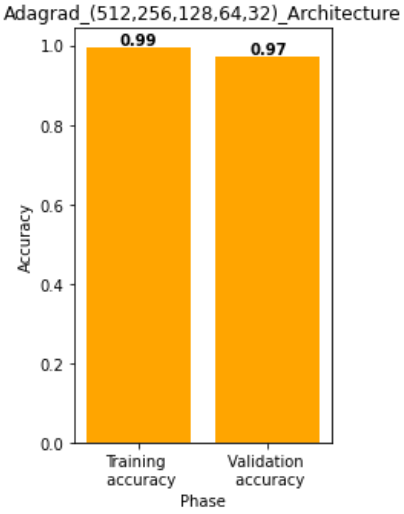


Fig.67: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.9934
-Validation Accuracy : 0.9699

Fig.68: Bar plot between training and validation accuracy

1.6. RMSprop:

RMSprop, short for Root Mean Square Propagation, is a gradient-based optimization algorithm for training deep learning models. It is a modification of the Adagrad algorithm that addresses its tendency to reduce the learning rate too much over time.

Like Adagrad, RMSprop adapts the learning rate for each parameter based on its past gradient history. However, instead of summing the squared gradients over all past iterations, RMSprop uses a moving average of the squared gradients. Specifically, RMSprop computes an exponentially weighted moving average of the squared gradients, with the decay rate controlled by a hyperparameter. This moving average smooths out the historical gradient information and prevents the learning rate from decreasing too quickly.

The RMSprop algorithm updates each parameter by dividing the current gradient by the square root of the moving average of the squared gradients for that parameter. The hyperparameters of RMSprop are the learning rate.
The update rule for RMSprop can be expressed as follows:

$$v(m) = \beta * (v(m-1)) + (1-\beta) * (\nabla w(m))^2$$

$$W(m+1) = W(m) - \left(\frac{\square}{\sqrt{v(m) + \varepsilon}} * \nabla W(m)\right)$$

Where,

v(m) is running estimate of square of the gradient of current weight:exponential moving average
β is a decay factor that weight the squared gradient and accumulated history ($0 < \beta < 1$)
W(m+1) is the new vector of model parameters (weights),
W(m) is the present vector of model parameters (weights),
□ is the learning rate ($0 < \square < 1$),
∇**W(m)** is the gradient of the loss function with respect to W, given as $\nabla W = \frac{\partial E_n}{\partial w}$
ε is small positive value such as 10^{-8}

RMSprop has been shown to be effective for a variety of machine learning tasks, particularly for deep neural networks. It is widely used in practice and is one of the most popular optimization algorithms for training deep learning models.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.6.1. Neural Architecture (128,64,32) Configuration:

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes
  - Hidden Layer 2 : 64 nodes
  - Hidden Layer 3 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4** epochs

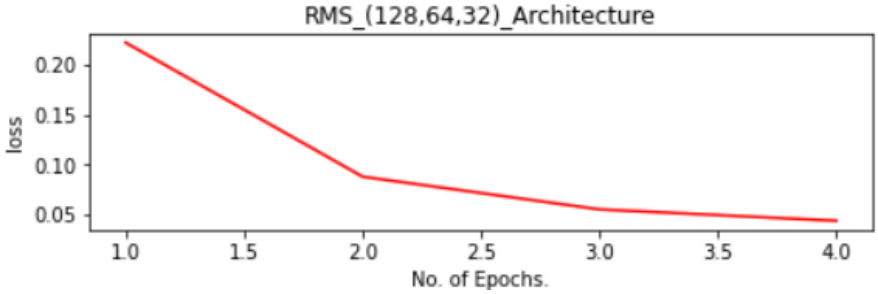
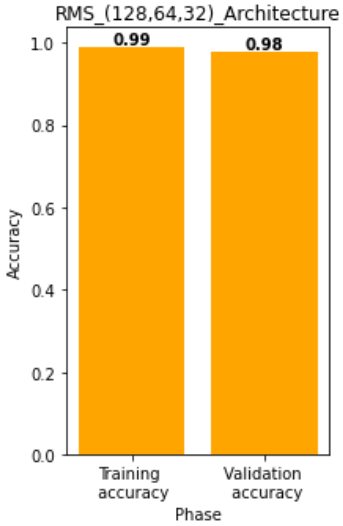


Fig.69: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9880
-Validation Accuracy : 0.9778

Fig.70: Bar plot between training and validation accuracy

1.6.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2** epochs

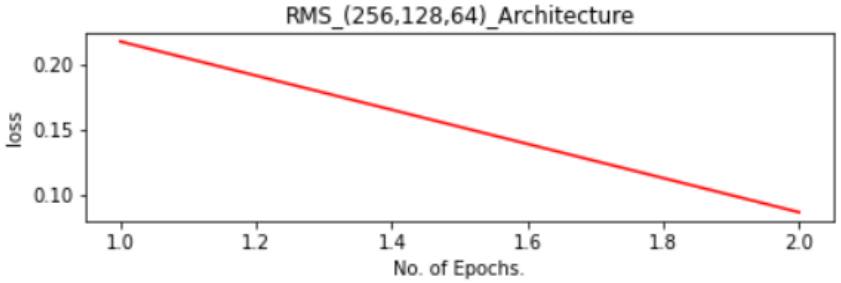
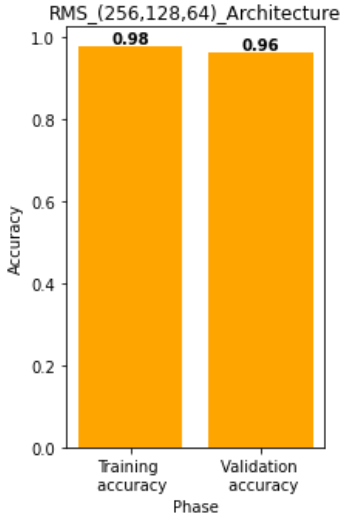


Fig.71: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9752
-Validation Accuracy : 0.9620

Fig.72: Bar plot between training and validation accuracy

1.6.3. Neural Architecture (512,256,128) Configuration (BEST)

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4** epochs

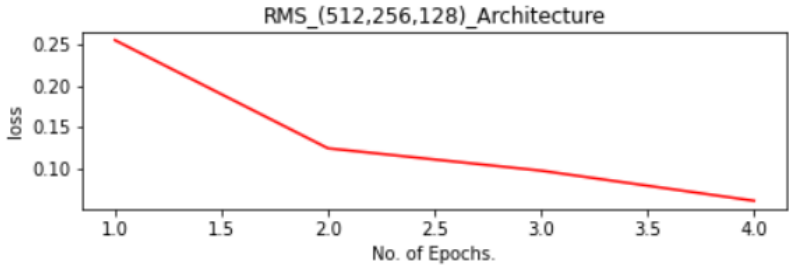
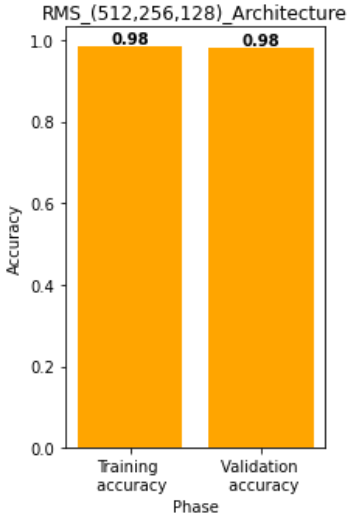


Fig.73: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9831
-Validation Accuracy : 0.9810

Fig.74: Bar plot between training and validation accuracy

		Actual Label									
		0	1	2	3	4	5	6	7	8	9
Predicted Label	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	740	0	1	0	0	8	10	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	5	0	734	0	3	9	8	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	5	0	0	0	738	1	15	0
	7	0	0	8	0	2	0	0	748	1	0
	8	0	0	15	0	1	0	1	5	737	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy: 0.9742

Fig. Confusion Matrix of the Best Possible RMS Prop Architecture

1.6.4. Neural Architecture (256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4** epochs

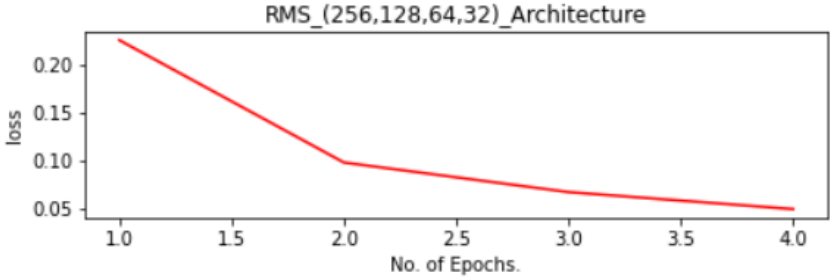
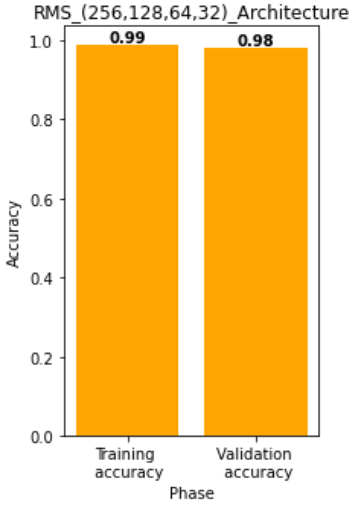


Fig.75: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9865
-Validation Accuracy : 0.9786

Fig.76: Bar plot between training and validation accuracy

1.6.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001

- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4** epochs

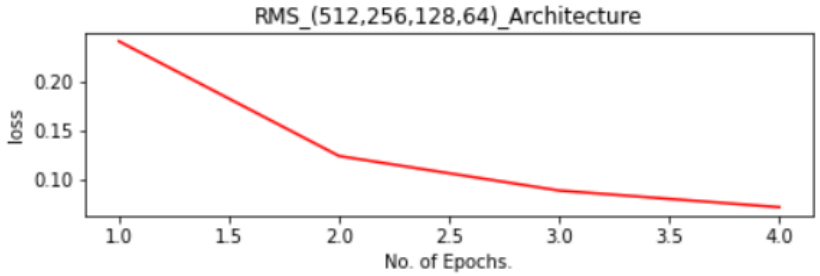
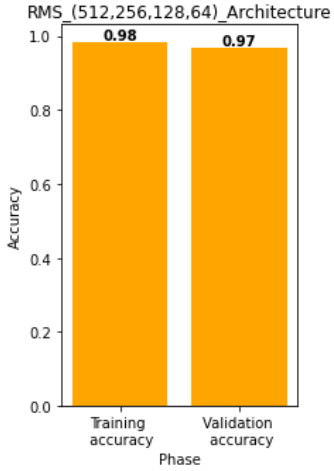


Fig. 77: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9823 -
-Validation Accuracy : 0.9694

Fig.78: Bar plot between training and validation accuracy

1.6.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Hidden Layer 4: 64 nodes
  - Hidden Layer 5: 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points

- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

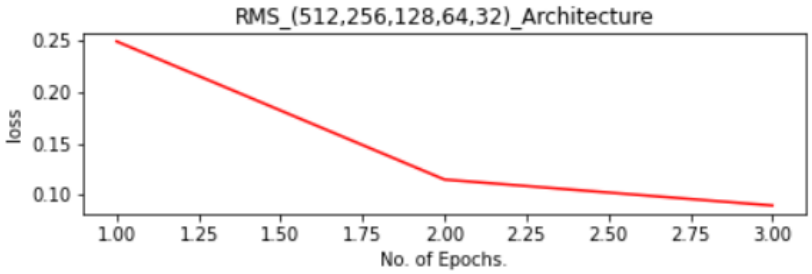
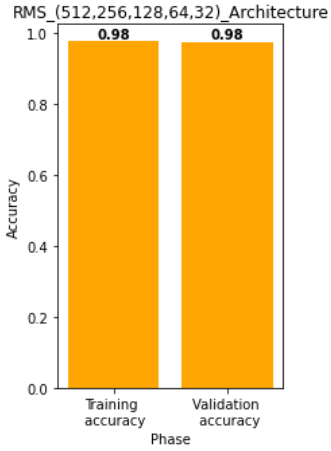


Fig.79: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.9768
-Validation Accuracy : 0.9762

Fig.80: Bar plot between training and validation accuracy

1.7. Adam:

Adam, short for Adaptive Moment Estimation, is a popular gradient-based optimization algorithm for training deep learning models. It is a combination of the ideas from two other optimization algorithms, namely RMSprop and momentum.

Like RMSprop, Adam uses a moving average of the gradients to adapt the learning rate for each parameter. However, in addition to the moving average of the squared gradients, Adam also computes a moving average of the gradients themselves. This allows the algorithm to take into account both the current gradient and the historical gradient information when computing the updates to the parameters.

Adam also includes a bias correction step to account for the fact that the moving averages are initialized to zero and are therefore biased towards zero in the early iterations. The bias correction is performed by dividing the moving averages by the corresponding bias correction terms.

The updates to the parameters in Adam are computed using a combination of the moving averages of the gradients and the squared gradients, along with the bias correction terms. The hyperparameters of Adam are the learning rate, the decay rates for the moving averages, and the epsilon value used to avoid division by zero.

The update rule for Adam can be expressed as follows:

$$u(m) = \beta_1 * u(m-1) + (1-\beta_1) * \nabla W(m)$$

$$v(m) = \beta_2 * v(m-1) + (1-\beta_2) * (\nabla W(m))^2$$

$$\hat{u}(m) = \frac{u(m)}{1-\beta_1^m}, \quad \hat{v}(m) = \frac{v(m)}{1-\beta_2^m}$$

$$W(m+1) = W(m) - \left(\frac{\square}{\sqrt{\hat{v}(m) + \varepsilon}} * \hat{u}(m)\right)$$

Where,

u(m) is a running estimate of the gradient of current weight: exponential moving average of gradient.

β_1 is a decay factor that weight the current gradient and accumulated history ($0 < \beta_1 < 1$)

v(m) is a running estimate of the squared gradient of current weight: exponential moving average of squared gradient

β_2 is a decay factor that weight the current squared gradient and accumulated history ($0 < \beta_2 < 1$)

$\hat{u}$(m) and $\hat{v}$(m) are the bias correction terms.

W(m+1) is the new vector of model parameters (weights),

W(m) is the present vector of model parameters (weights),

□ is the learning rate ($0 < \square < 1$),

∇**W(m)** is the gradient of the loss function with respect to W, given as $\nabla w = \frac{\partial E_n}{\partial w}$

ε is small positive value such as 10^{-8}

Adam has been shown to be effective for a wide range of deep learning tasks and is particularly popular for training deep neural networks. It is often considered the default optimization algorithm for deep learning models and is widely used in practice.

While training the Neural Architecture for the Classification of the numbers from the MNIST dataset. We have consider the different architectures as given below:

1.7.1. Neural Architecture (128,64,32) Configuration(BEST):

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 128 nodes
  - Hidden Layer 2 : 64 nodes
  - Hidden Layer 3 : 32 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

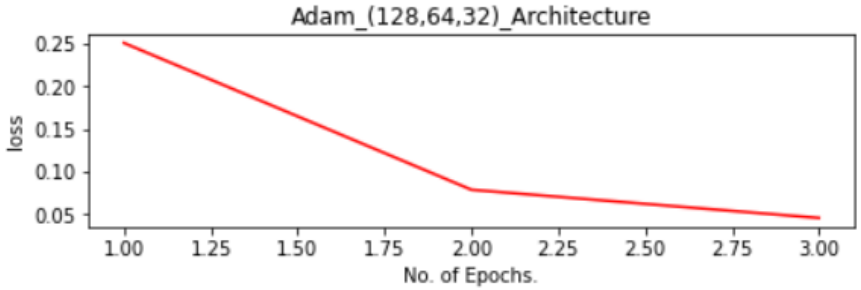
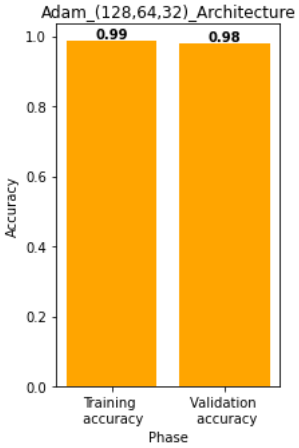


Fig.81: Loss vs epochs for training data with architecture 128,64,32



- Training Accuracy : 0.9856
-Validation Accuracy : 0.9778

Fig.82: Bar plot between training and validation accuracy

		Actual Label									
Predicted Label		0	1	2	3	4	5	6	7	8	9
	0	0	0	0	0	0	0	0	0	0	0
	1	0	0	0	0	0	0	0	0	0	0
	2	0	0	743	0	1	0	4	6	5	0
	3	0	0	0		0	0	0	0	0	0
	4	0	0	8	0	729	0	9	11	2	0
	5	0	0	0	0	0	0	0	0	0	0
	6	0	0	3	0	0	0	751	0	5	0
	7	0	0	10	0	1	0	2	743	3	0
	8	0	0	12	0	2	0	13	6	726	0
	9	0	0	0	0	0	0	0	0	0	0

- Testing Accuracy : 0.9728

Fig. Confusion Matrix of the Best Possible Adam Architecture

1.7.2. Neural Architecture (256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **3** epochs

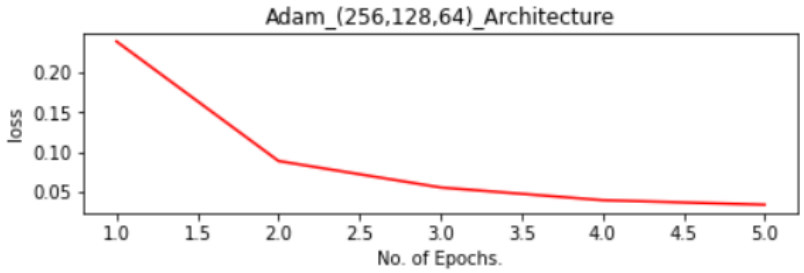
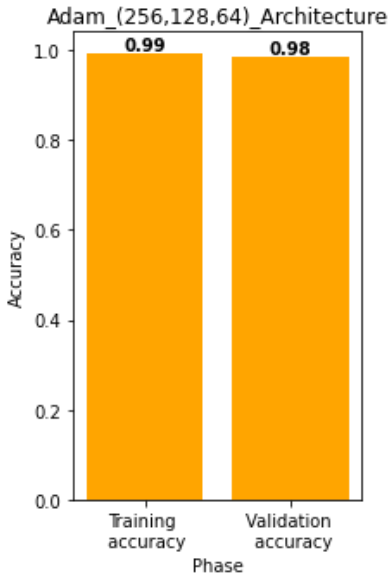


Fig.83: Loss vs epochs for training data with architecture 256,128,64



- Training Accuracy : 0.9903
-Validation Accuracy : 0.9828

Fig.84: Bar plot between training and validation accuracy

1.7.3. Neural Architecture (512,256,128) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 512 nodes
  - Hidden Layer 2 : 256 nodes
  - Hidden Layer 3 : 128 nodes
  - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001

- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **4** epochs

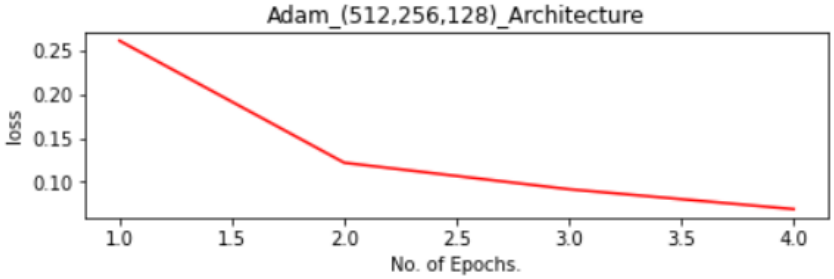
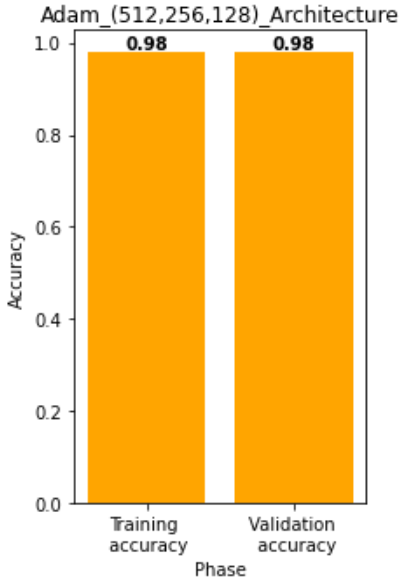


Fig.85: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9782
-Validation Accuracy : 0.9778

Fig.86: Bar plot between training and validation accuracy

1.7.4. Neural Architecture (256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
  - Input Layer : 784 features
  - Hidden Layer 1 : 256 nodes
  - Hidden Layer 2 : 128 nodes
  - Hidden Layer 3 : 64 nodes
  - Hidden Layer 4 : 32 nodes
  - Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2** epochs

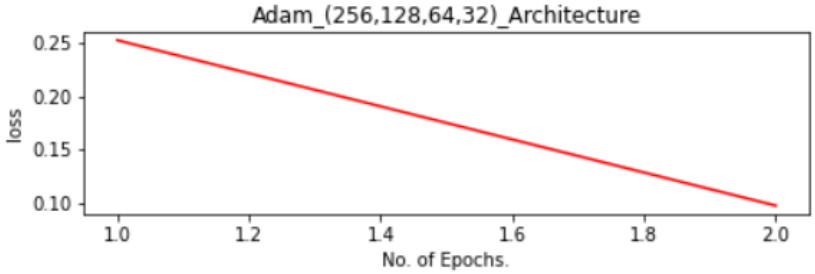
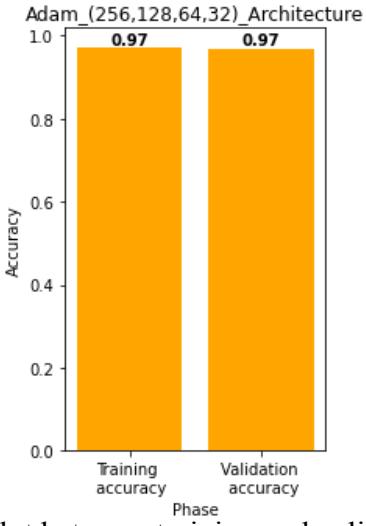


Fig.87: Loss vs epochs for training data with architecture 512,256,128



- Training Accuracy : 0.9704
-Validation Accuracy : 0.9696

Fig.88: Bar plot between training and validation accuracy

1.7.5. Neural Architecture (512,256,128,64) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:
    - Input Layer : 784 features
    - Hidden Layer 1 : 512 nodes
    - Hidden Layer 2 : 256 nodes
    - Hidden Layer 3 : 128 nodes
    - Hidden Layer 4 : 64 nodes

    - Output layer : 10 nodes
- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
    - Threshold: 0.0001
    - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **6** epochs

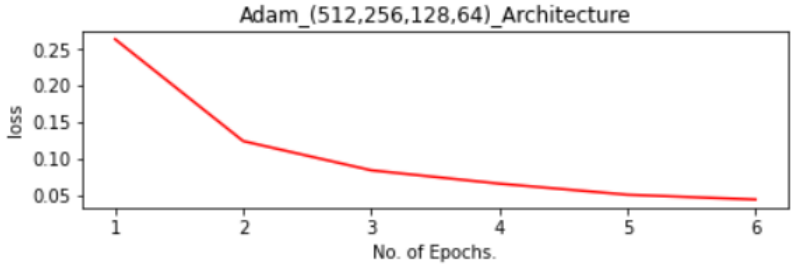
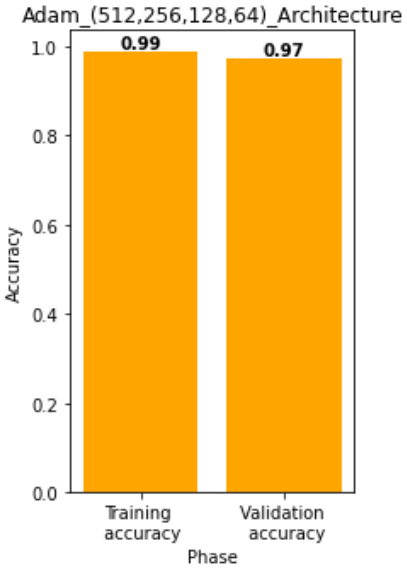


Fig.89: Loss vs epochs for training data with architecture 512,256,128,64



- Training Accuracy : 0.9868
-Validation Accuracy : 0.9736

Fig.90: Bar plot between training and validation accuracy

1.7.6. Neural Architecture (512,256,128,64,32) Configuration

In this Neural Architecture we have considered the following parameters

Hyperparameters:

- Neural Architecture:

- Input Layer : 784 features
- Hidden Layer 1 : 512 nodes
- Hidden Layer 2 : 256 nodes
- Hidden Layer 3 : 128 nodes
- Hidden Layer 4: 64 nodes
- Hidden Layer 5: 32 nodes
- Output layer : 10 nodes

- Training Dataset : 11385 data points
- Testing Dataset: 3795 data points
- Validation Dataset: 3795 data points
- Learning rate: 0.001
- Convergence Criteria (which ever reach earlier):
  - Threshold: 0.0001
  - Epochs: 500000

Results:

- Total number of epochs needed to reach threshold **2** epochs

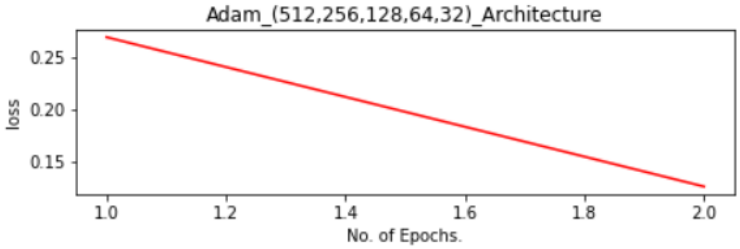
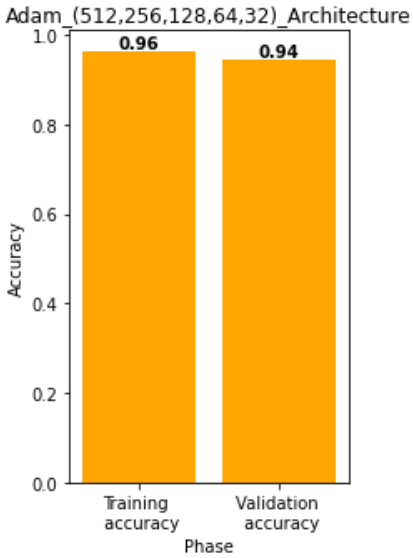


Fig.91: Loss vs epochs for training data with architecture 512,256,128,64,32



- Training Accuracy : 0.9610
-Validation Accuracy : 0.9422

Fig.92: Bar plot between training and validation accuracy