REPORT

Name: SHREY S KULKARNI

ID: 17XJ1A0348

College: MAHINDRA UNIVERSITY

Content: ANN ALGORITHM

The following parameters have been considered for comparison:

- Hidden Layers.
- Batch Size.
- Activation Function. (Only considered Tanh for Toy-Problem)
- λ (Decay parameter) η (Learning rate) for regularization.
- No of iterations for which the training is stopped.
- $MAPE_{Validation}$ & $MAPE_{Test}$ (Mean absolute percentage error).

NOTE:

The graphs have been pasted in the exact same order as the delineated parameters in the table.
The algorithms have been entered at the last

TOY PROBLEM

1. Hidden Layers & Batch Size:
   The hidden layers have been varied between 2, 3 and 5.
   The following parameters have been kept constant:
   - λ (Decay parameter) = 0
   - η (Learning rate): 0.01
   - Activation Function: Tanh (For all layers)

Table 1.1

No.	*Epochs*	*Batch Size*	*Hidden Layers (Varies)*
1.	1000	64	2
2.	1000	256	2
3.	1000	Full	2
4.	1000	64	3
5.	1000	256	3
6.	3000	Full	3
7	1000	64	5
8	1000	256	5
9	1000	Full	5

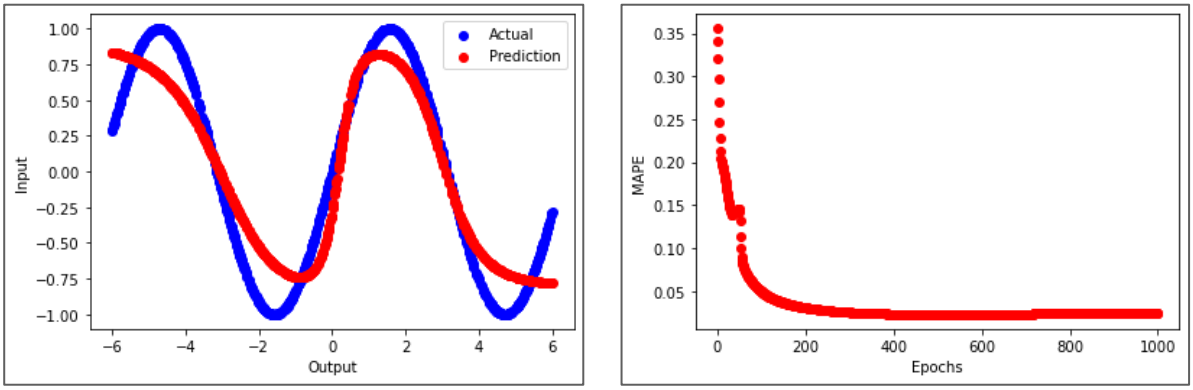
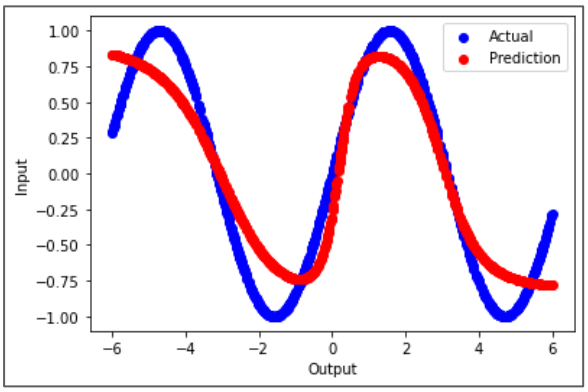


Figure 1.1 and 1.2 (Layers:2 ; Batch Size:64)



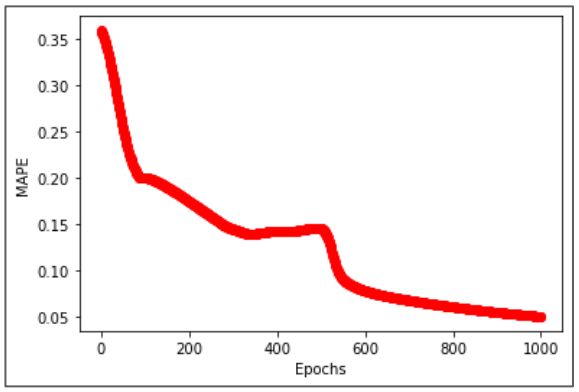
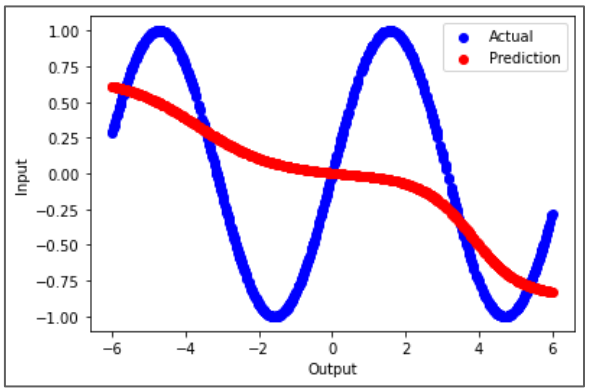


Figure 1.3 and 1.4 (Layers:2 ; Batch Size:256)



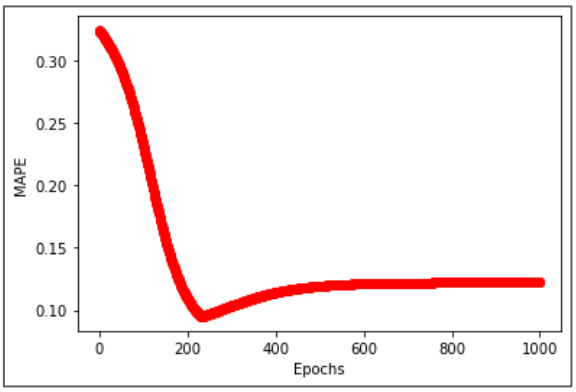
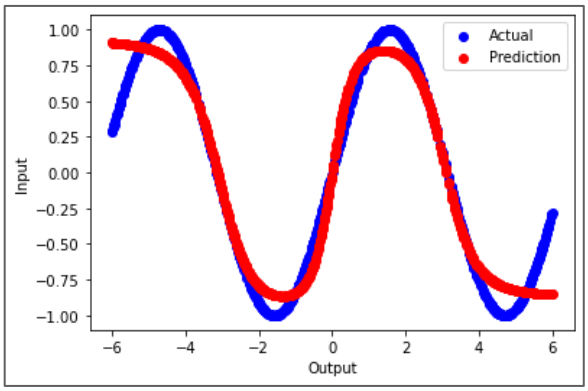


Figure 1.5 and 1.6 (Layers:2 ; Batch Size: Full)



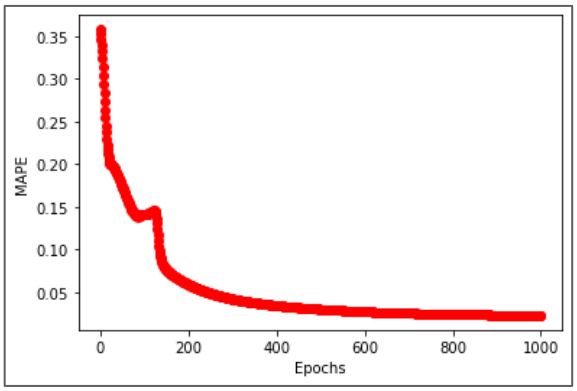
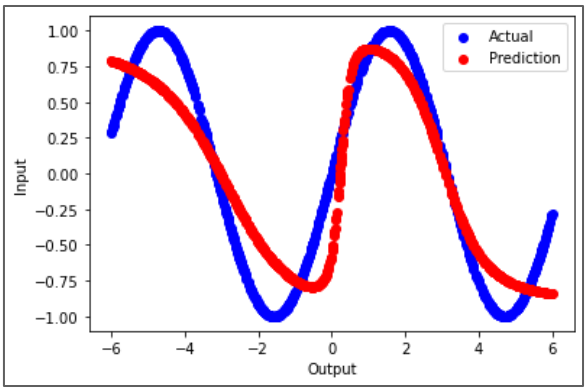


Figure 1.7 and 1.8 (Layers:3 ; Batch Size:64)



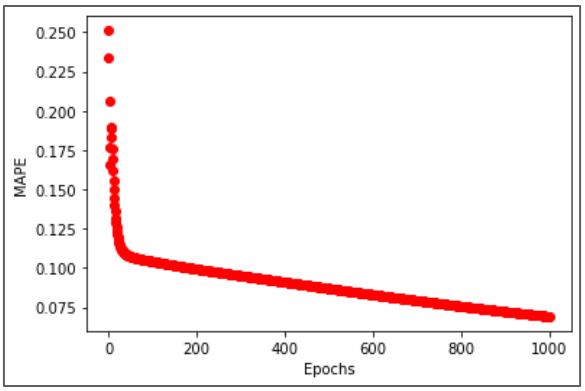
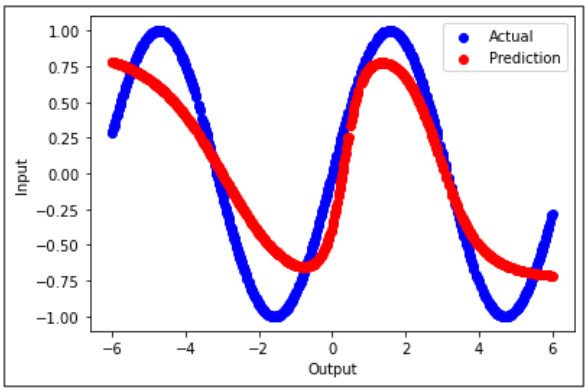


Figure 1.9 and 1.10 (Layers:3 ; Batch Size:256)



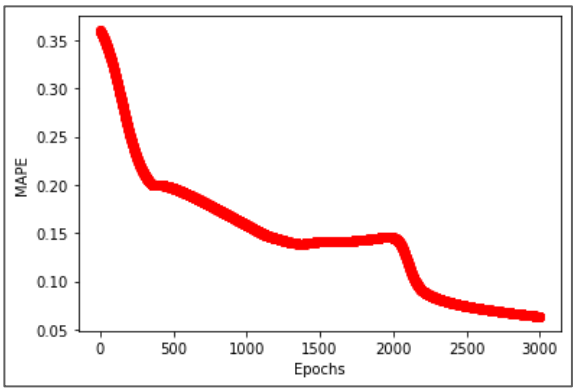
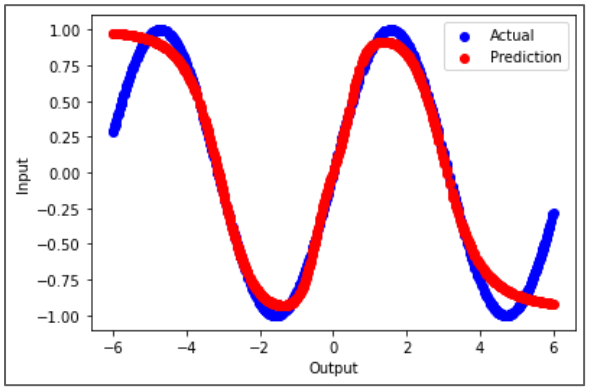


Figure 1.11 and 1.12 (Layers:3 ; Batch Size: Full)



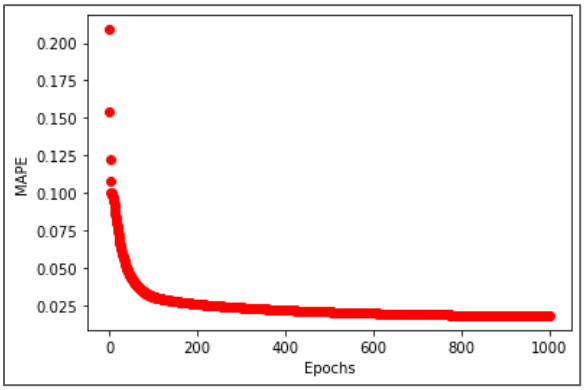
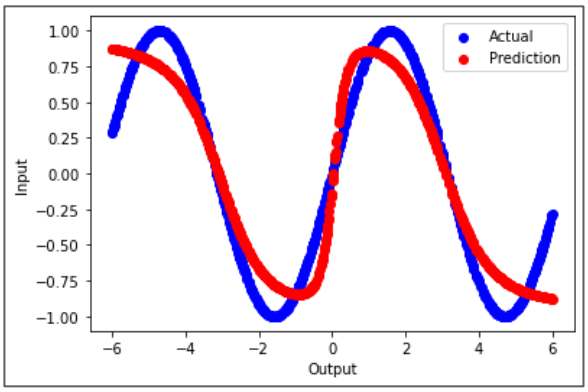


Figure 1.13 and 1.14 (Layers:5; Batch Size:64)



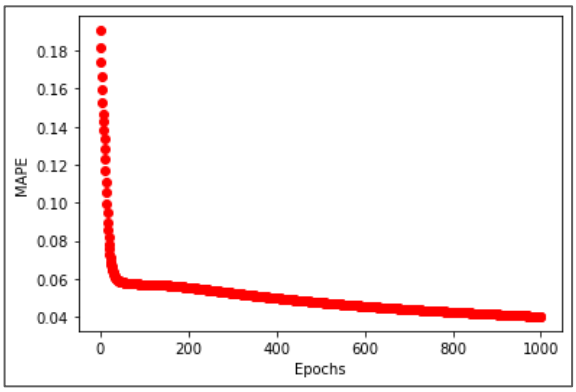
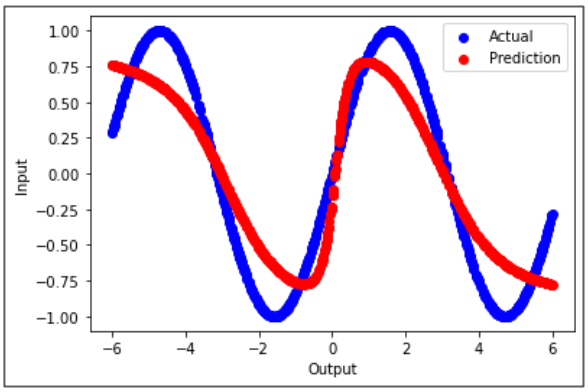


Figure 1.15 and 1.16 (Layers:5 ; Batch Size:256)



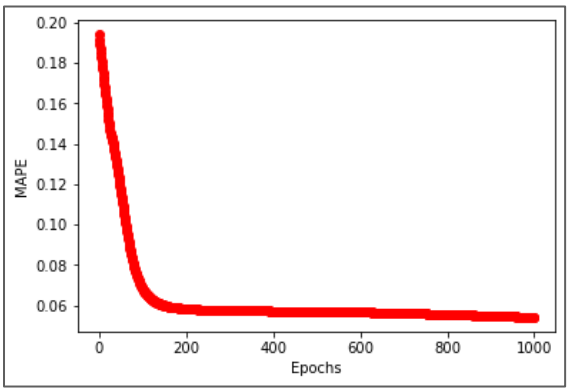


Figure 1.17 and 1.18 (Layers:5 ; Batch Size: Full)

2. Learning Rate Parameter:

The following parameters have been kept constant:

- Batch Size: 128
- λ (Decay parameter) = 0
- Hidden Layers: 5
- Activation Functions: Tanh and Logistic

Table 2.1

No.	Epochs	Learning Rate
1.	1000	0.4
2.	1000	0.1
3.	1000	0.04
4.	1000	0.01
5.	1000	0.004
6.	1000	0.001

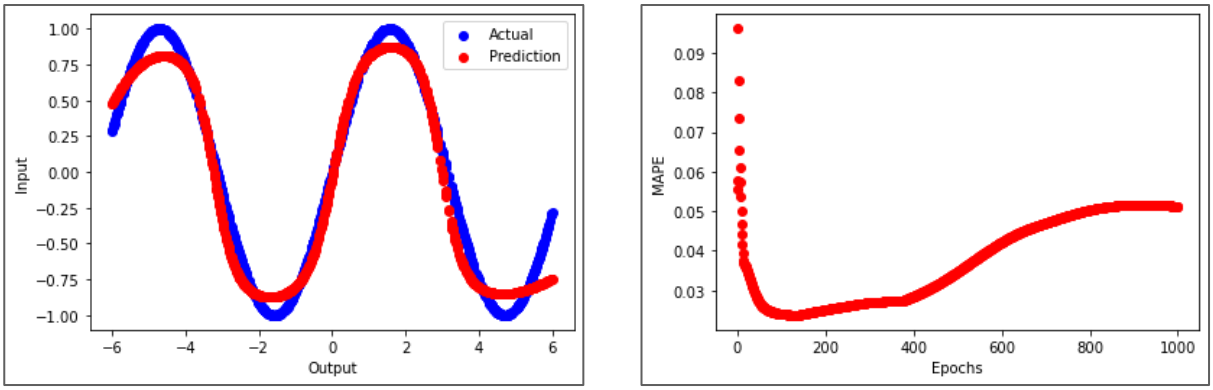


Figure 2.1 and 2.2 (Learning Rate: 0.4)

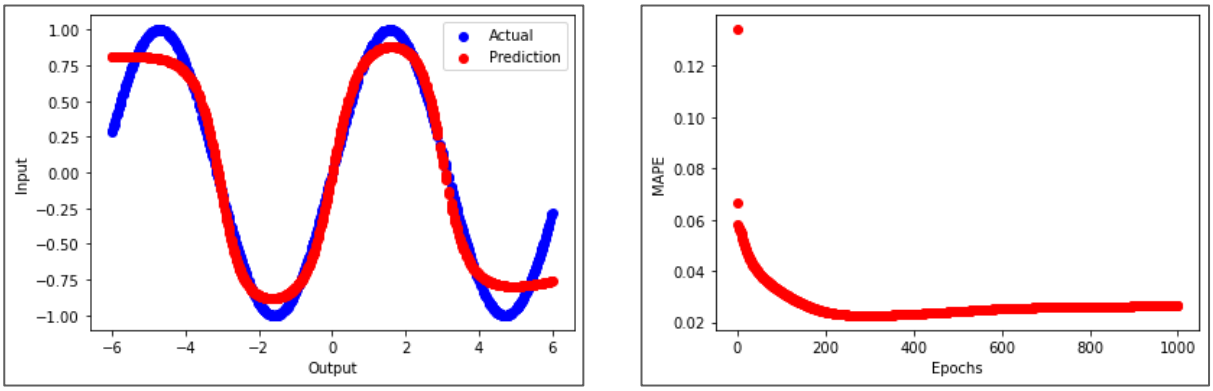


Figure 2.3 and 2.4 (Learning Rate: 0.1)

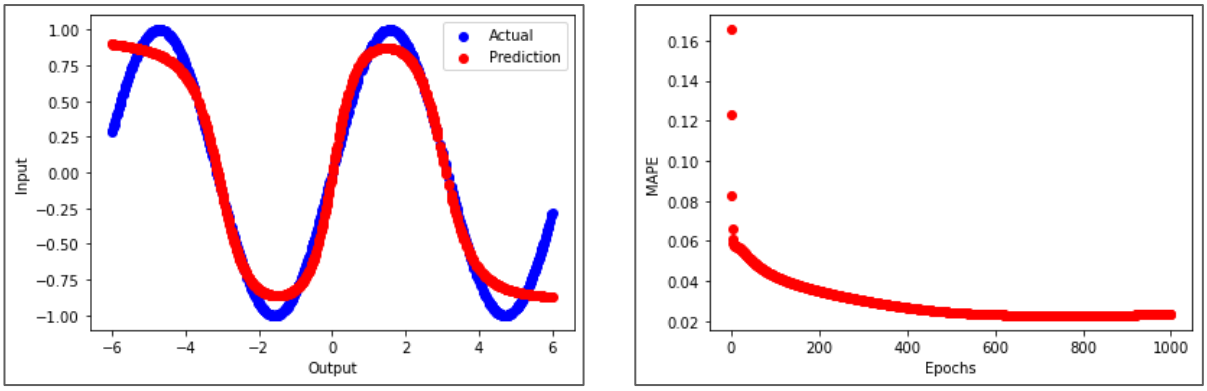
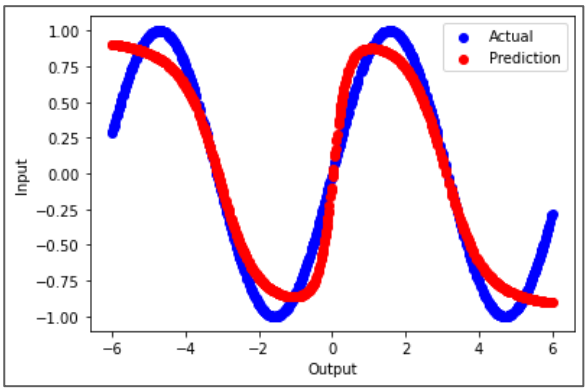


Figure 2.5 and 2.6 (Learning Rate: 0.04)



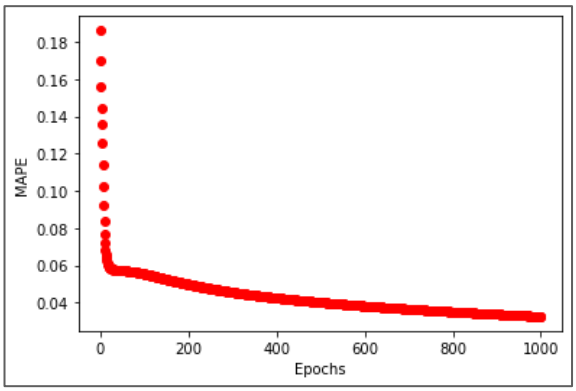
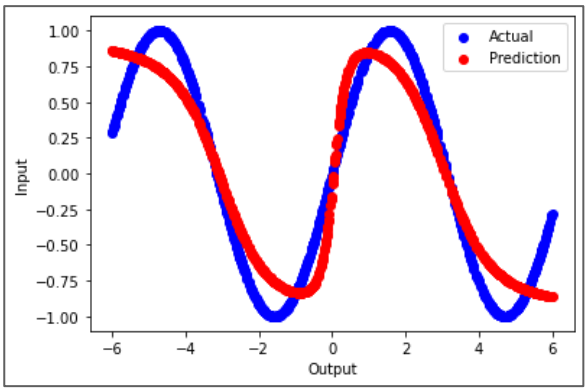


Figure 2.7 and 2.8 (Learning Rate: 0.01)



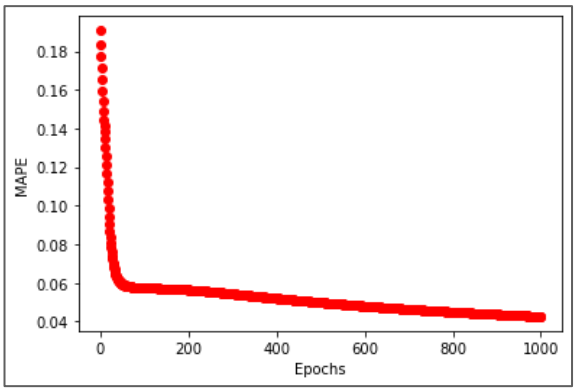
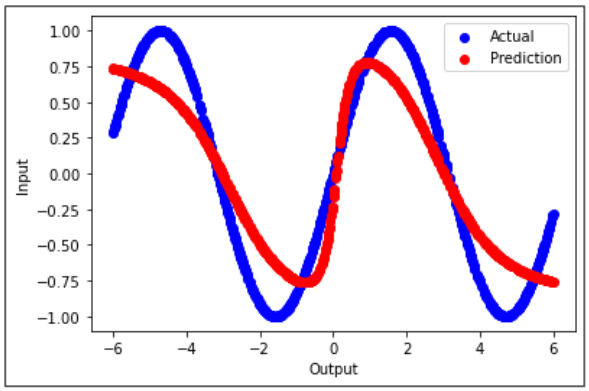


Figure 2.9 and 2.10 (Learning Rate: 0.004)



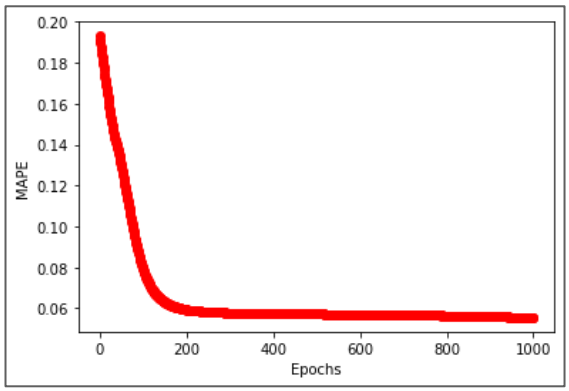
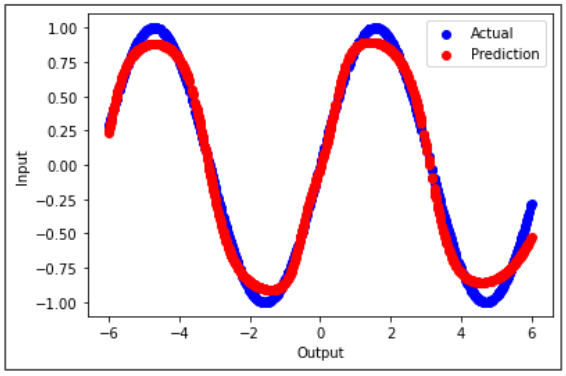
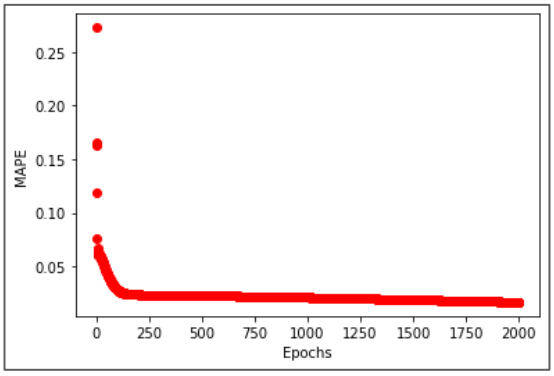


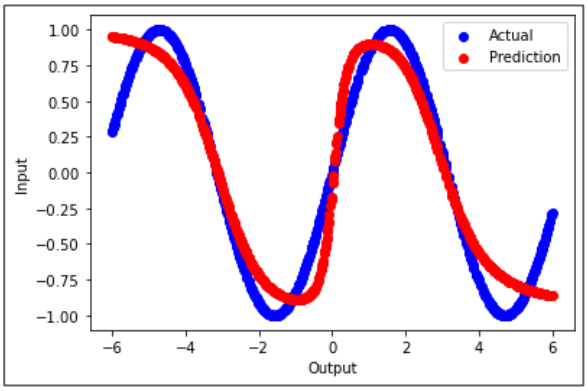
Figure 2.11 and 2.22 (Learning Rate: 0.001)

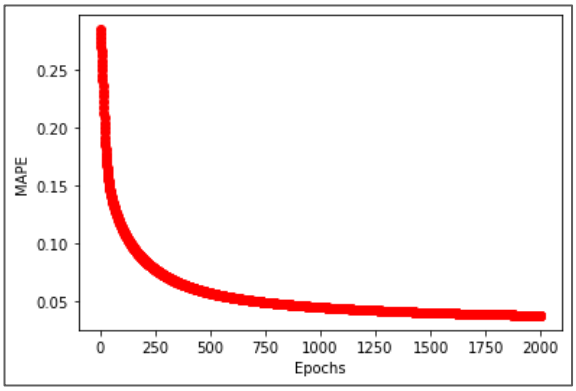
3. Keeping in mind that we need the best **fit**, the best combination of parameters for the toy-problem data set (i.e. the sin(x)) are as follows:
   - Hidden Layers: [1,6,1]
   - Epochs: 2000
   - Batch Size: 64
   - Learning Rate: 0.08
   - Activation Functions: Relu and Tanh (Last Layer)





4. Keeping in mind that we need the **optimal error curve**, the best combination of parameters for the toy-problem data set (i.e. the sin(x)) are as follows:
   - Hidden Layers: [1,5,1]
   - Epochs: 2000
   - Batch Size: 128
   - Learning Rate: 0.009
   - Activation Functions: Tanh and Logistic (Last Layer)





COMBINED CYCLE POWERPLANT:

1. Hidden Layers & Batch Size:

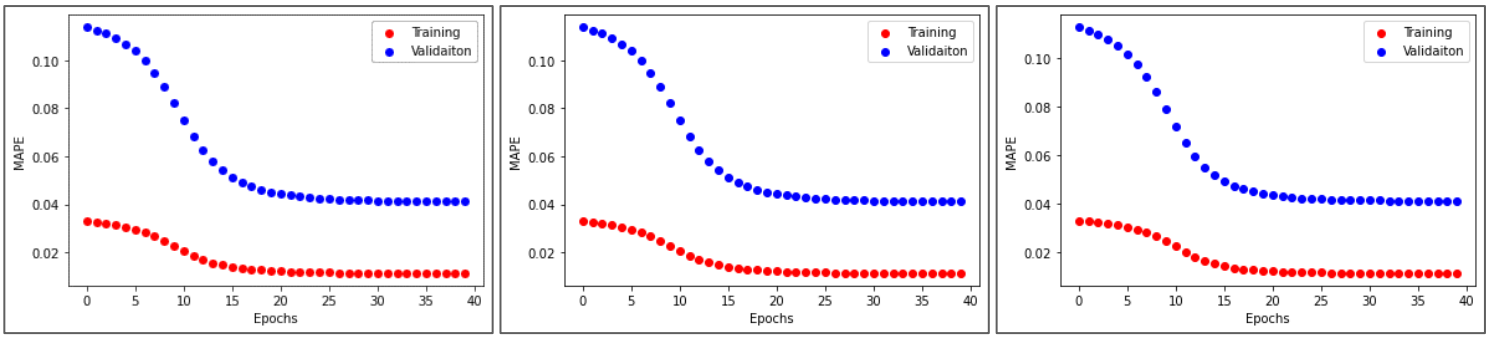
   The hidden layers have been varied between 2, 3 and 5.
   Epochs: 40
   η(Learning rate): 0.01
   λ(Regularization): 0.01
   Activation Function: Relu, Logistic (Last layer)

Table 1.1

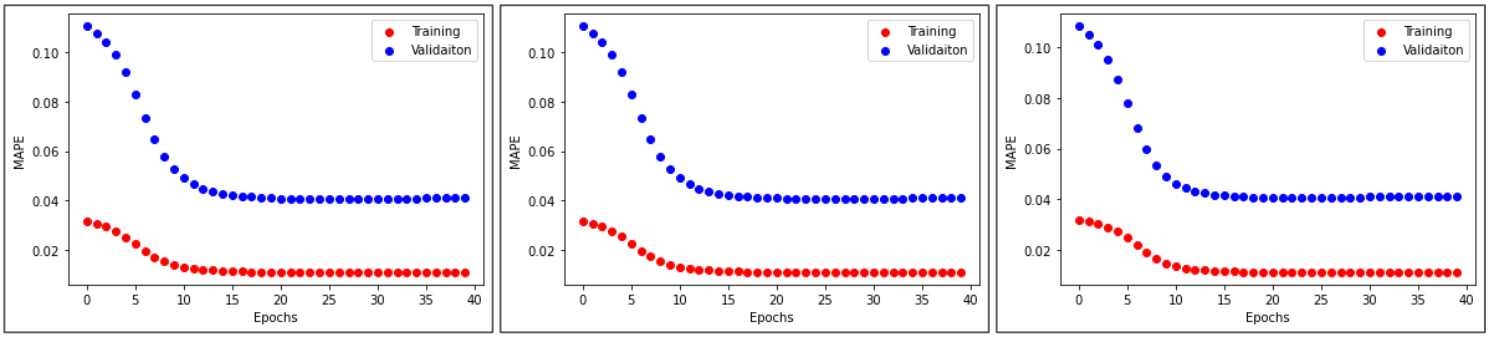
No.	Batch Size	Hidden Layers	$MAPE_{Test}$
1.	64	2	0.0481
2.	256	2	0.054
3.	Full	2	0.092
4.	64	3	0.043
5.	256	3	0.051
6.	Full	3	0.0502
7	64	5	0.0392
8	256	5	0.381
9	Full	5	0.045

NOTE:

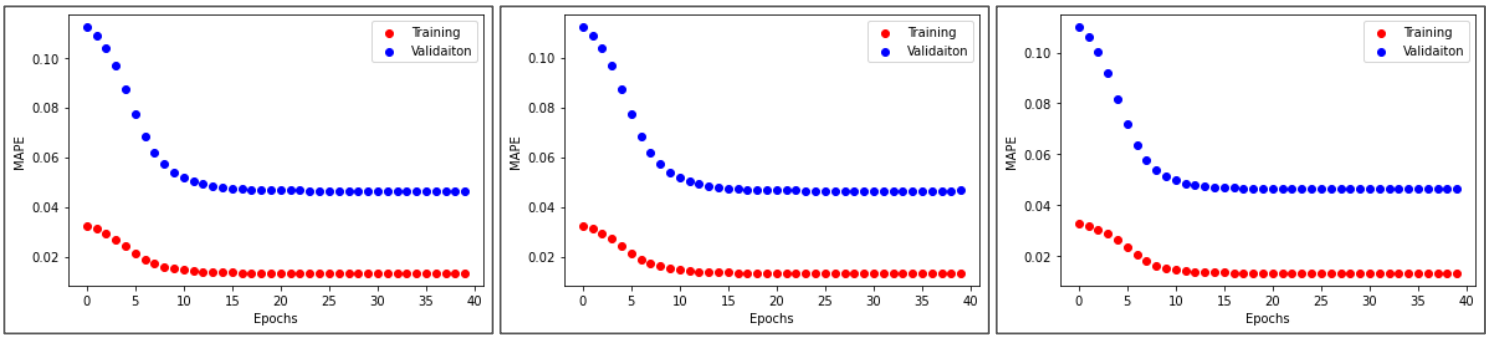
- ***The graphs have been pasted in the exact same order as the delineated parameters in the table.***
- ***Three Graphs having the same no of hidden layers and varying batch sizes have been compared side by side.***
- ***The three graphs for a given no of hidden layers look very similar, I assure you they are NOT the same.***



(Figures 1.1, 1.2, 1.3; Hidden Layers: 2 ; Batch Size: 64 – 256 - Full)



(Figures 1.4, 1.5, 1.6; Hidden Layers: 3 ; Batch Size: 64 – 256 - Full)



(Figures 1.7, 1.8, 1.9; Hidden Layers: 5 ; <u>Batch Size: 64 – 256 - Full</u>)

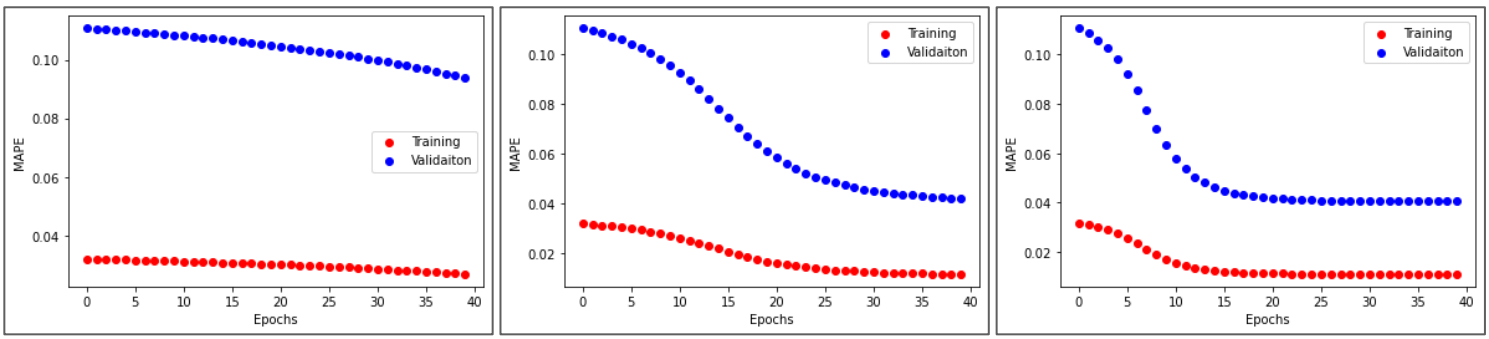
2. <u>Learning rate parameter</u>
   - The hidden layers have been varied between 2, 3 and 5.
   - Epochs: 40
   - Activation Function: Tanh
   - λ(Regularization): 0.01
   - Batch Size: 256
   - Hidden Layers: 3
   - (MAPE Test rounded up, becomes asymptotic)

Table 2.1

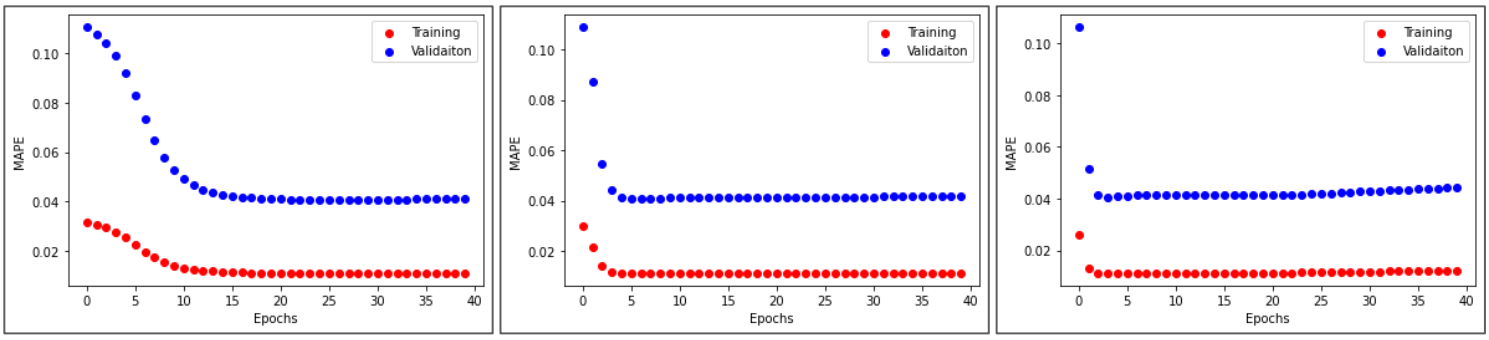
No.	Learning Parameter	$MAPE_{Test}$
1.	0.001	0.0962
2.	0.004	0.0593
3.	0.008	0.0424
4.	0.01	0.04126
5.	0.04	0.04143
6.	0.08	0.04263

<u>NOTE</u>:

- ***The graphs have been pasted in the exact same order as the delineated parameters in the table.***
- ***Three Graphs having the same no parameters and varying Learning have been compared side by side***



(Figures 2.1, 2.2, 2.3; Learning Rate: 0.001 – 0.004 – 0.008)

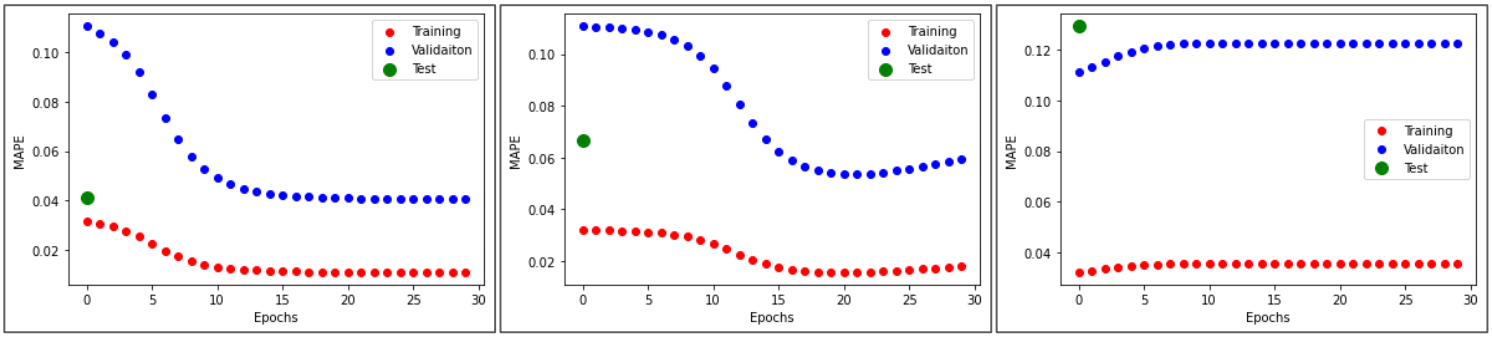


(Figures 2.1, 2.2, 2.3; Learning Rate: 0.01 – 0.04 – 0.08)

3. <u>Regularization Parameter:</u>
   - The parameter λ has been varied between 0, 0.1 and 0.8
   - Epochs: 40
   - Activation Function: Tanh
   - Batch Size: 256
   - Hidden Layers: 3
   - Learning Parameter: 0.01
   - (MAPE Test rounded up, becomes asymptotic)

Table 3.1

No.	Regularization parameter	$MAPE_{Test}$
1.	0	0.043
2.	0.1	0.071
3.	0.8	0.129



(Figures 3.1, 3.2, 3.3; Regularization parameter: 0 – 0.1 – 0.8)

4. Activation Function:
    - Epochs: 40
    - Batch Size: 256
    - Activation Function: Tanh
    - λ(Regularization): 0.01
    - Hidden Layers: 3
    - Learning Parameter: 0.01
    - "First two Layer", "Last Layer" = Activation functions used
    - (MAPE Test rounded up, becomes asymptotic)

Table 4.1

No.	Activation Function	$MAPE_{Test}$
1.	Tanh, Tanh	0.0438
2.	Logistic, Logistic	0.0452
3.	Relu, Tanh	0.0423
4.	Relu, Logistic	0.0409

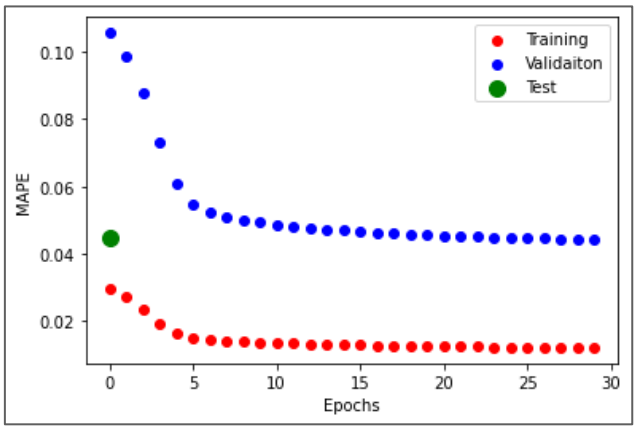


Figure 3.1 (Activation Function: Tanh, Tanh)

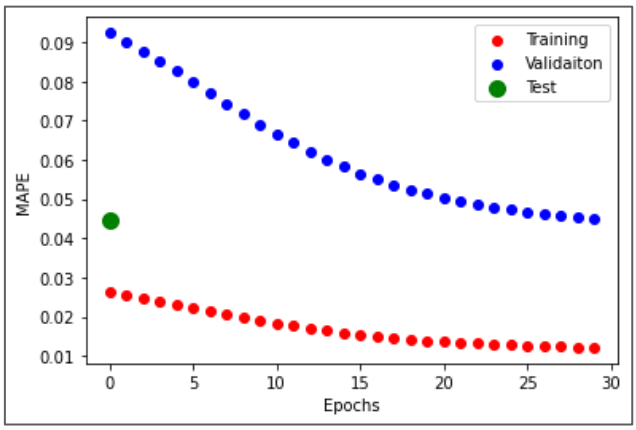


Figure 3.2 (Activation Function: Logistic, Logistic)

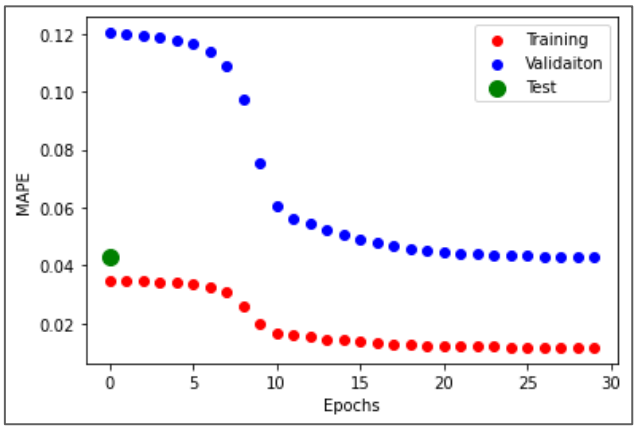


Figure 3.3 (Activation Function: Relu, Tanh)

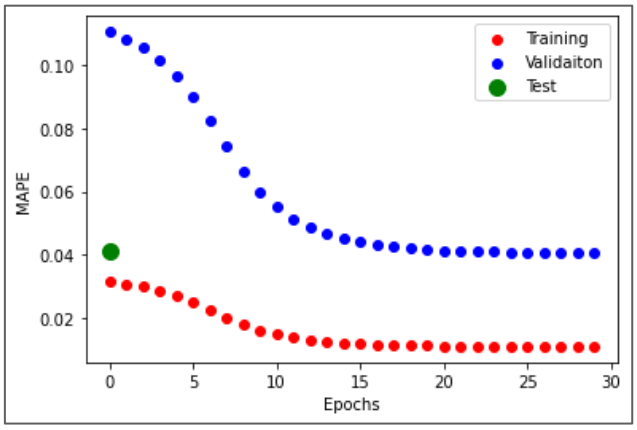


Figure 3.4 (Activation Function: Relu, Logistic)

5. <u>No of Iterations where error starts to increase:</u>

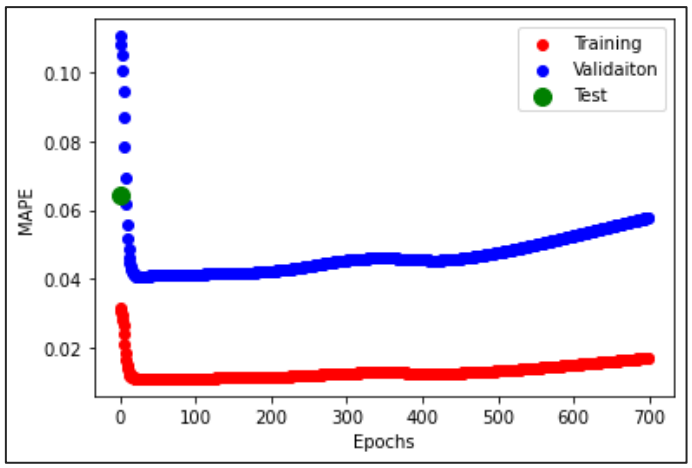


Figure 4.1 (No of iterations = 90)

6. RMS Error:

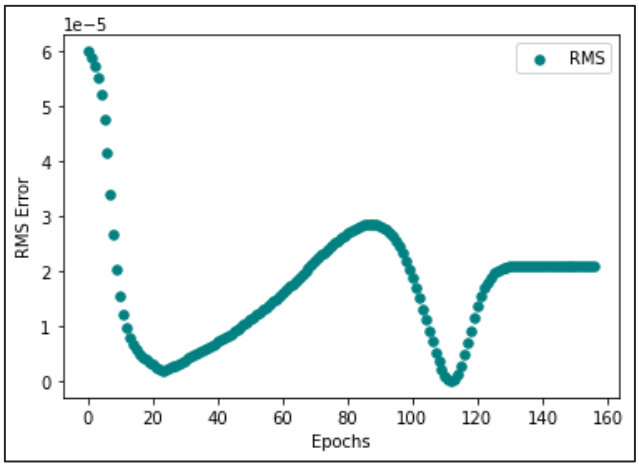


Figure 6.1

COMBINED CYCLE POWERPLANT: (SGD-MOMENTUM & ADAM)

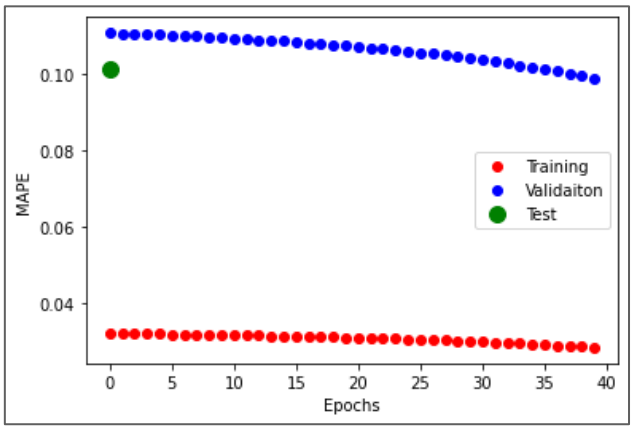
7. Learning rate parameter:
   - Epochs: 40
   - Activation Function: Tanh
   - λ(Regularization): 0.01
   - Batch Size: 256
   - Hidden Layers: 3
   - (MAPE Test rounded up, becomes asymptotic)

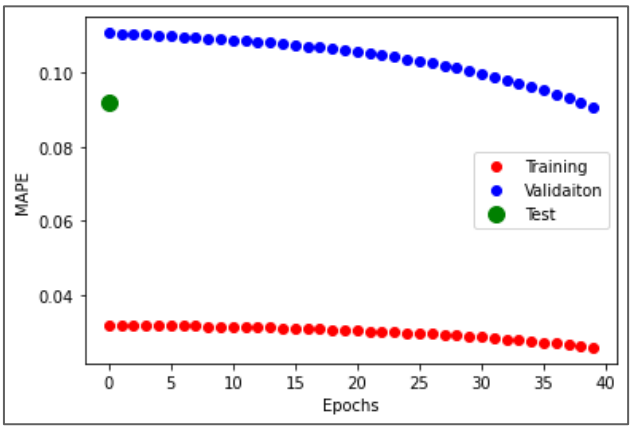
Table 1.1

No.	Learning Parameter	$MAPE_{Test}$
1.	0.008	0.1014
2.	0.01	0.0921
3.	0.08	0.0681
4.	0.1	0.0652

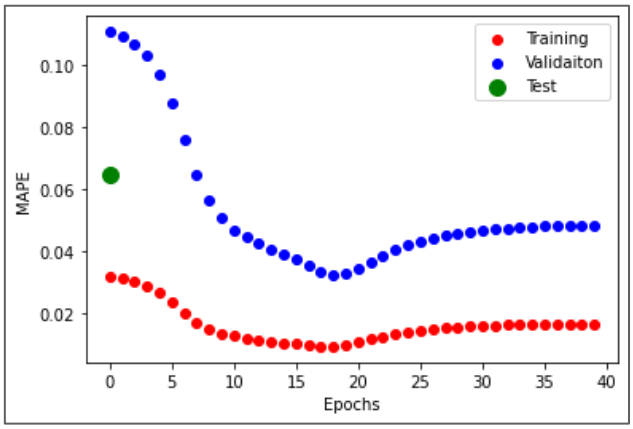
NOTE:

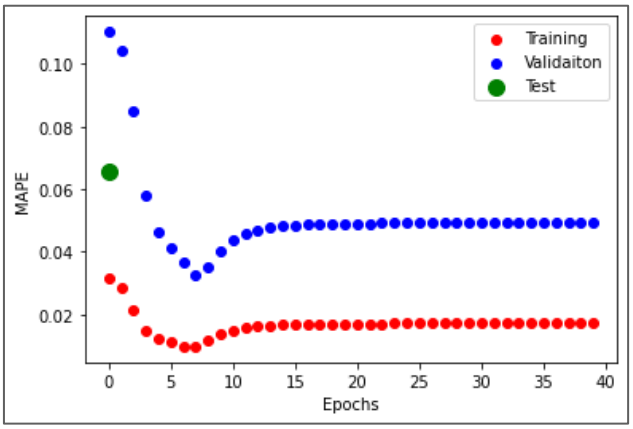
- ***The graphs have been pasted in the exact same order as the delineated parameters in the table.***
- ***Three Graphs having the same no parameters and varying Learning have been compared side by side***





(Figures 2.1, 2.2, 2.3; Learning Rate: 0.008 – 0.01)



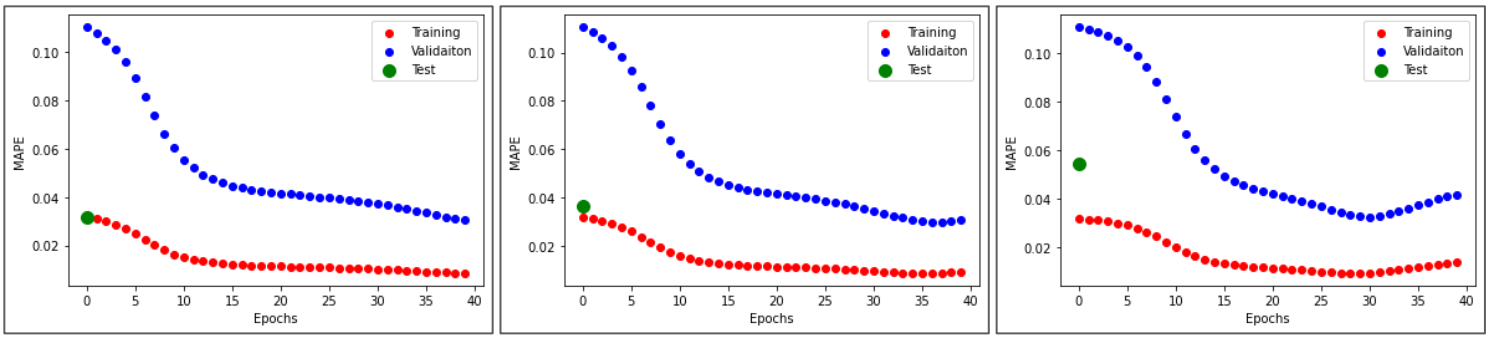


(Figures 2.1, 2.2, 2.3; Learning Rate: 0.08 – 0.1)

8. <u>Regularization Parameter:</u>
   - The parameter λ has been varied between 0, 0.1 and 0.8
   - Epochs: 40
   - Activation Function: Tanh
   - Batch Size: 256
   - Hidden Layers: 3
   - Learning Parameter: 0.01
   - (MAPE Test rounded up, becomes asymptotic)

Table 2.1

No.	Regularization parameter	$MAPE_{Test}$
1.	0	0.0315
2.	0.005	0.0382
3.	0.05	0.0545



(Figures 3.1, 3.2, 3.3; Regularization parameter: 0 – 0.005 – 0.05)

9. Activation Function:
    - Epochs: 40
    - Batch Size: 256
    - Activation Function: Tanh
    - λ(Regularization): 0.01
    - Hidden Layers: 3
    - Learning Parameter: 0.01
    - "First two Layer", "Last Layer" = Activation functions used
    - (MAPE Test rounded up, becomes asymptotic)

Table 4.1

No.	Activation Function	$MAPE_{Test}$ (ADAM)	$MAPE_{Test}$ (SGD+MOM)
1.	Tanh, Tanh	0.0261	0.0453
2.	Logistic, Logistic	0.0288	0.0419
3.	Relu, Tanh	0.0696	0.0438
4.	Relu, Logistic	0.0451	0.0421

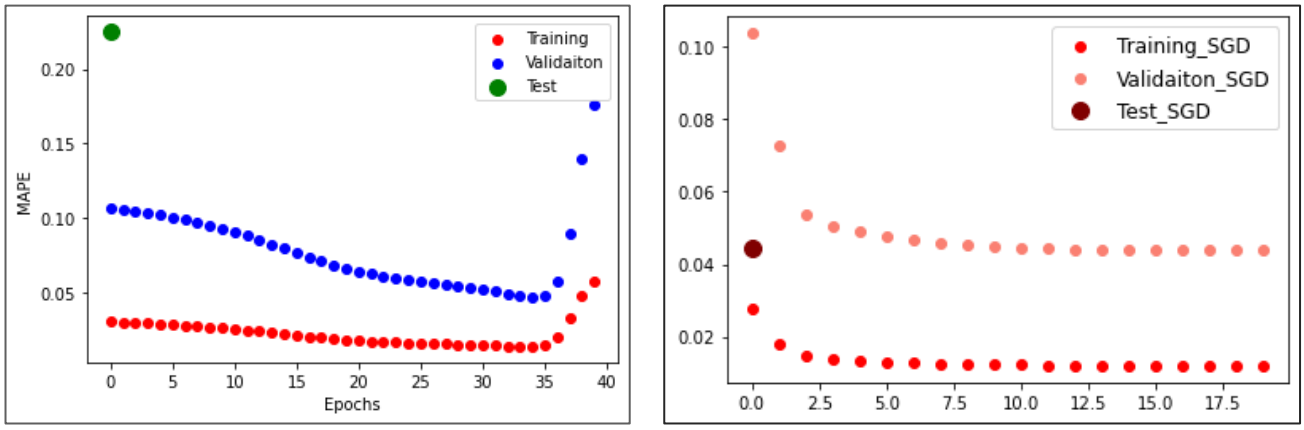


Figure 3.1 (Activation Function: Tanh, Tanh)

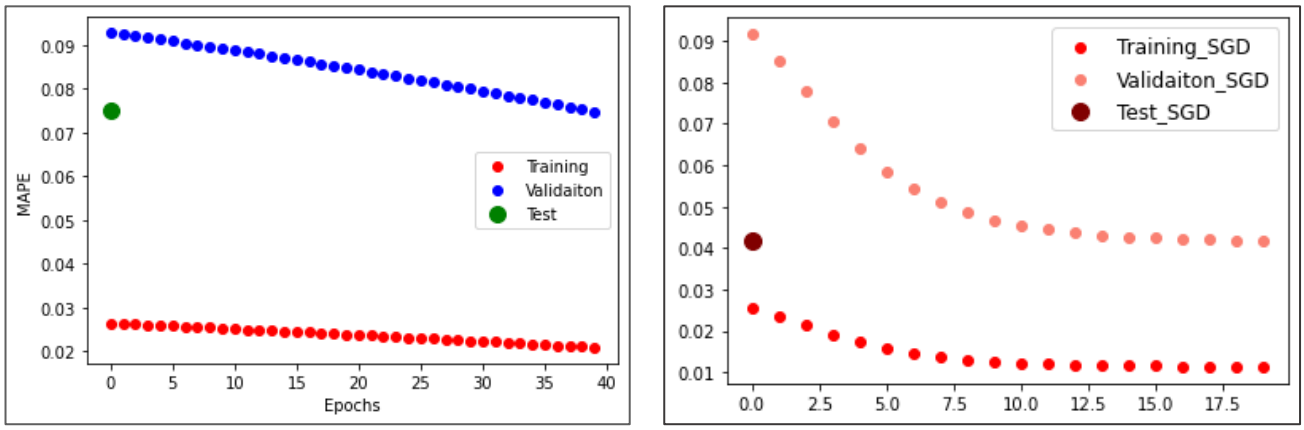


Figure 3.2 (Activation Function: Logistic, Logistic)

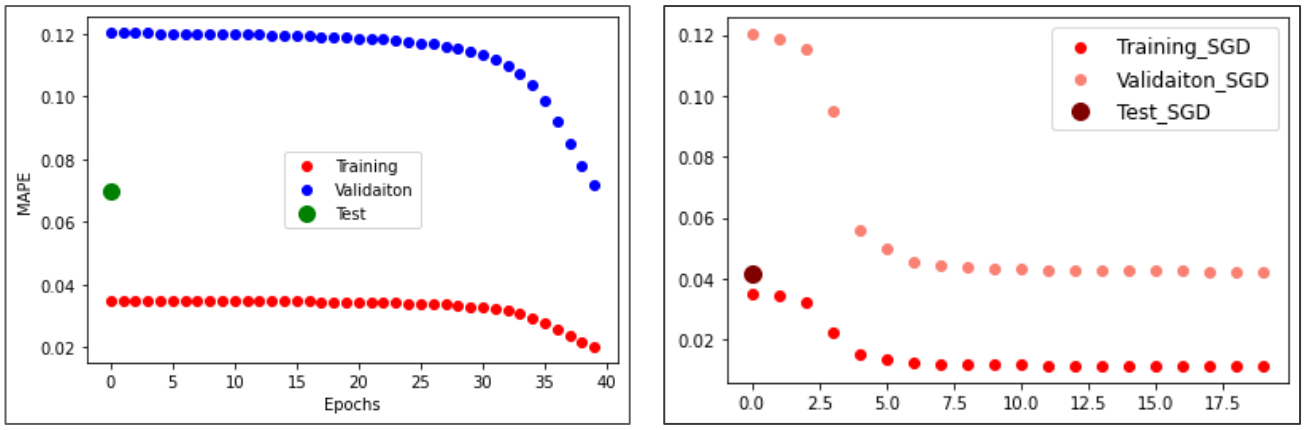


Figure 3.3 (Activation Function: Relu, Tanh)

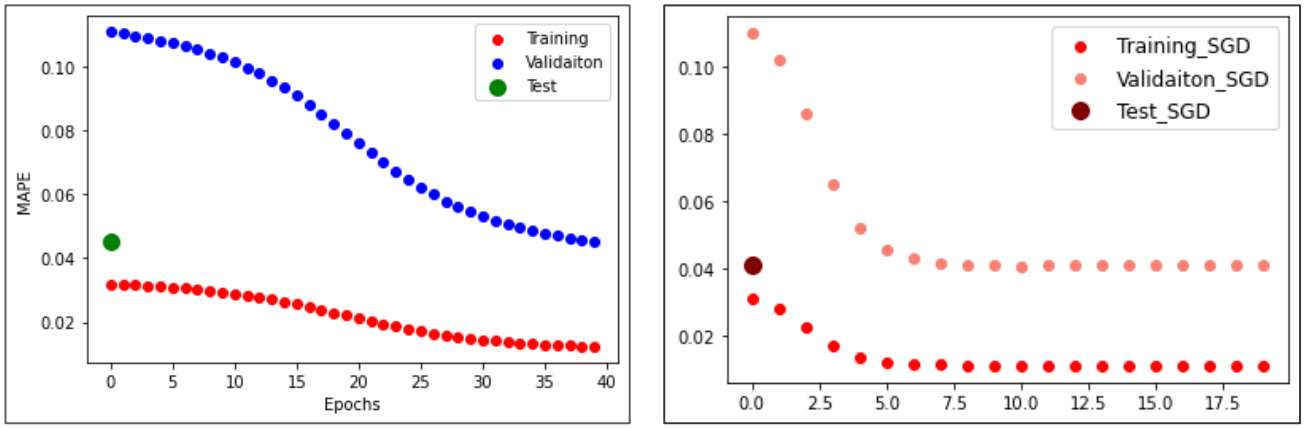


Figure 3.4 (Activation Function: Relu, Logistic)

10. No of Iterations where error starts to increase:

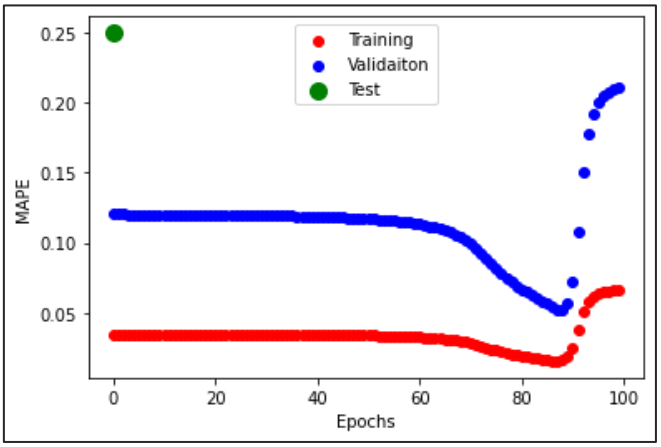
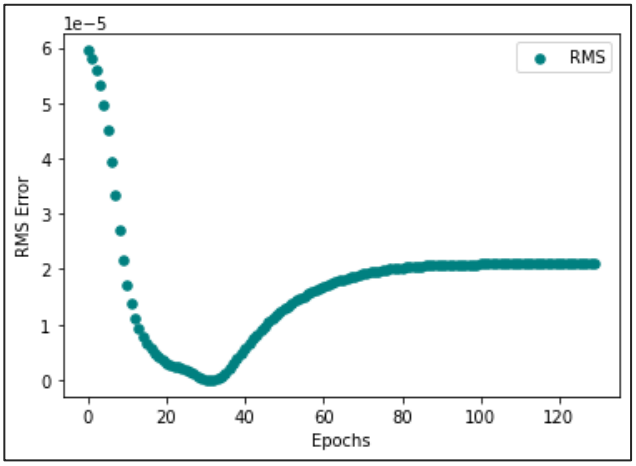


Figure 4.1 (No of iterations = 90)

11. RMS Error for ADAM and SGD+MOMENTUM respectively:



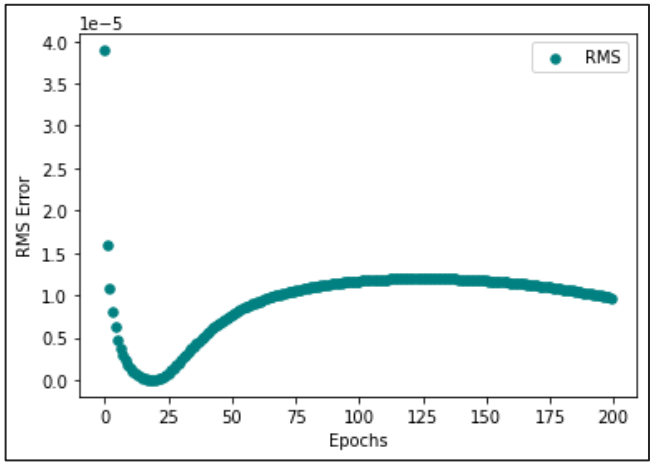


Figure 5.1, 5.2 (First figure corresponds to ADAM, Second corresponds to SGD+MOM)

ALGORITHM:

```
class ANN:

  def __init__(self. Activation, layers):
    Initialize weights and biases randomly using random. randn

  def Activation_Func(x, name, deriv=False):
     if (name == 'Logistic'):
     elif (name == 'Tanh'):

     elif (name == 'Relu'):



  def Feedforward (self, x):
    Multiply randomly initialized weights with the given input
    Append intermediate layers
     return(output, intermediate)



  def Backpropagation (self, y, output, intermediate):
     Initiate deltas
     Enter delta for last layer
     Initiate parameters
     for i in reversed(range(len(deltas)-1)):
     Enter weights in terms of deltas
     Enter derivatives in terms of weights and learning rate parameter
     return Dw, Db
```

```
#BACKPROP FOR ADAM
 def Backpropagation (self, y, output, intermediate):
      Initiate deltas
      Enter delta for last layer
      Initiate parameters
      for i in reversed(range(len(deltas)-1)):
      Enter weights in terms of deltas
      Enter derivatives in terms of weights and learning rate parameter
      Vw,Vb = self.SGD(Dw,Db)
      Sw,Sb = self.RMS_Prop(Dw, Db)
      W,B = self.ADAM(Vw, Sw, Vb, Sb)
      Update weights and biases
      return Dw, Db




  def MAPE (self, y_true, y_pred):


  def RMS (self, y_true, y_pred):



  def train (self, input, output, batch_size, epochs, lr):

     for e in range(epochs):
        initiate all lists

        while i<len(y):
           Divide dataset into batches specified
           Update iterative parameter

           output, intermediate = self.FeedForward(x_batch)
           Dw, Db = self.Backpropagation(y_batch, output, intermediate)

           Update weights
           Update biases

           Append the last row of intermediate into a list (recently updated output)

        E. append(self. MAPE(y, y_pred))
        rms. append(self.RMS(y, y_pred))


        Append the latest weights into feedforward for validation

     return (E, y_pred,Ev,y2_pred,w,z,rms)


  def Test(self,x,y,weight,bias):
    Repeat Feedforward using the weights from training
     return(Et)
```

Read and split dataset
Normalize input and ouput

Neural = ANN(activation=['Relu','Logistic'], layers=[4,3,1])
Use train and test functions to get ouputs and consequently the graphs