Homework : 7 Machine Learning - 1 (Supervised Methods)

Importing all necessary libraries

Loading the saved dataset using numpy

Instantiating an object for the Classifier class in engine.py

Splitting the training data into train data and validation data (80/20 split)

```
----------------------------------------------------------------------------------------------------
Total number of images in Training data:  48000
Total number of images in Validation data:  12000
Total number of images in Test data:  10000
Total number of classes in the output lables: 10
----------------------------------------------------------------------------------------------------
```

Normalizing the pixel values in the training, validation and test data.

```
----------------------------------------------------------------------------------------------------
Shape of Flattened Training data:  (48000, 784)
Shape of Flattened Valdation data:  (12000, 784)
Shape of Flattened Test data:  (10000, 784)
We have 784 features in the input data.
----------------------------------------------------------------------------------------------------
```

Hyper-parameter Optimization for MLP on FMNIST 1]

1] (A) Start with M = 48 hidden nodes, η = 0.01, λ = 10 and B = 32 and vary the batch size and calculate the time taken to reach an accuracy of 80%

Model Summary for One Hidden Layer with 48 neurons.

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

 Hidden_Layer (Dense)        (None, 48)                37680

 Output_Layer (Dense)        (None, 10)                490

=================================================================
Total params: 38,170
Trainable params: 38,170
Non-trainable params: 0
_________________________________________________________________
```

Select the batch size that has the smallest run-tme. Report this batch size. Also report the sample mean and sample standard deviation of the 5 runs times.

Out[8]:
```
{32: 4.271796178817749,
 64: 5.985004568099976,
 128: 4.863578271865845,
 256: 11.695928525924682,
 512: 9.853559207916259}
```

Out[9]:
```
{32: 0.8438213739199153,
 64: 2.517410708836467,
 128: 0.6179979150825756,
 256: 2.229329935207475,
 512: 1.3635249894867774}
```

```
The batch size that has the smallest average run-time is: 32 with a run-t
ime of 4.271796178817749
Sample mean of the 5 run times is: 4.271796178817749
Sample standard deviation of the 5 run times is: 0.8438213739199153
```

Inference:

We can see from the results obtained the average time taken to get an accuracy greater than 80% was lesser for a batch size of 32 images. Hence, I will be using batch_size = 32 for 1] (b) ¶

Type *Markdown* and LaTeX: α^2

Homework : 7 Machine Learning - 1 (Supervised Methods)

Perform a grid search over the following hyper-parameters:

• $\eta \in$ {0.001,0.01, 0.1}

• $\lambda \in$ {1e−4,1e−3,1e−2}

• Number of hidden nodes $M \in$ {40,80, 160}

Importing all necessary libraries

```
TensorFlow version: 2.12.0
```

Loading the saved training and test data

Instantiating an object for the Classifier class in engine.py

```
Metal device set to: Apple M1

systemMemory: 8.00 GB
maxCacheSize: 2.67 GB
```

Splitting the training data into train data and validation data (80/20 split)

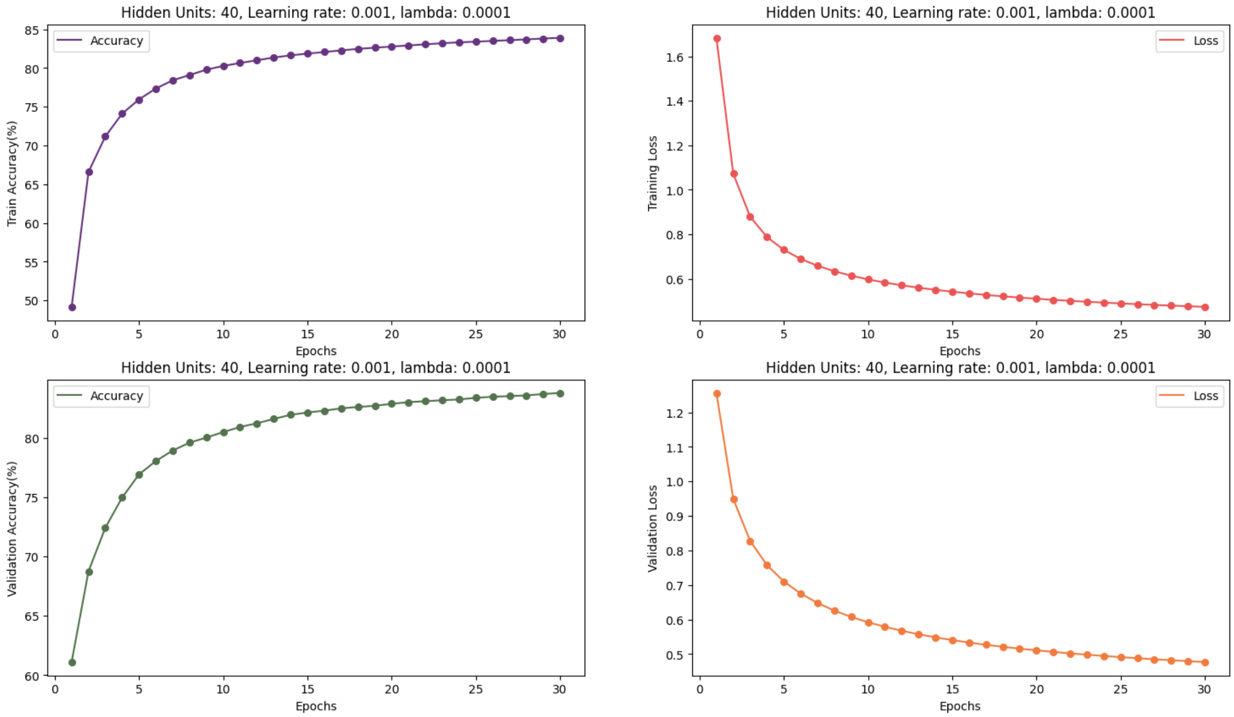
```
-------------------------------------------------------------------------
--------------------
Total number of images in Training data:  48000
Total number of images in Validation data:  12000
Total number of images in Test data:  10000
Total number of classes in the output lables: 10
-------------------------------------------------------------------------
--------------------
```

Normalizing the pixel values.

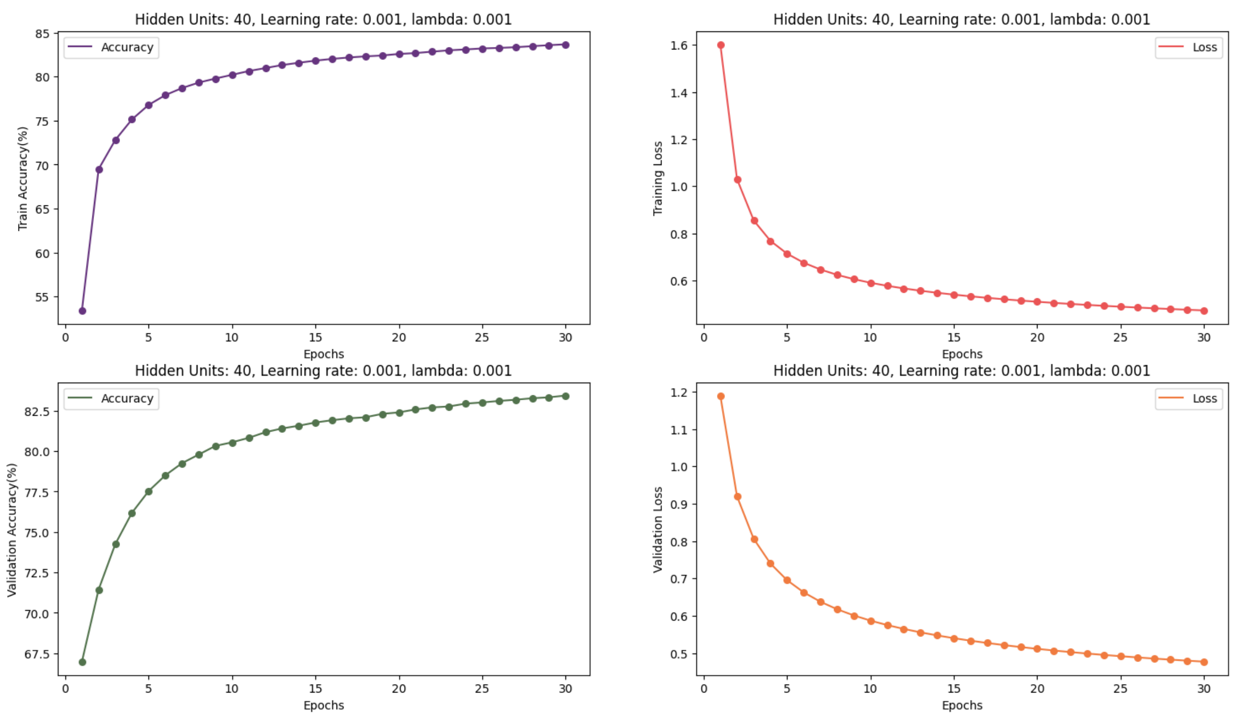
Intializing the three lists for learning_rates, regularization_parameters and hidden_nodes and also fixing the batch size for the training and validation data. ¶

Out[20]:
```
{'hidden_units': 160,
 'learning_rate': 0.1,
 'reg_param': 0.001,
 'final_validation_accuracy': <tf.Tensor: shape=(), dtype=float32, numpy=
88.308334>}
```

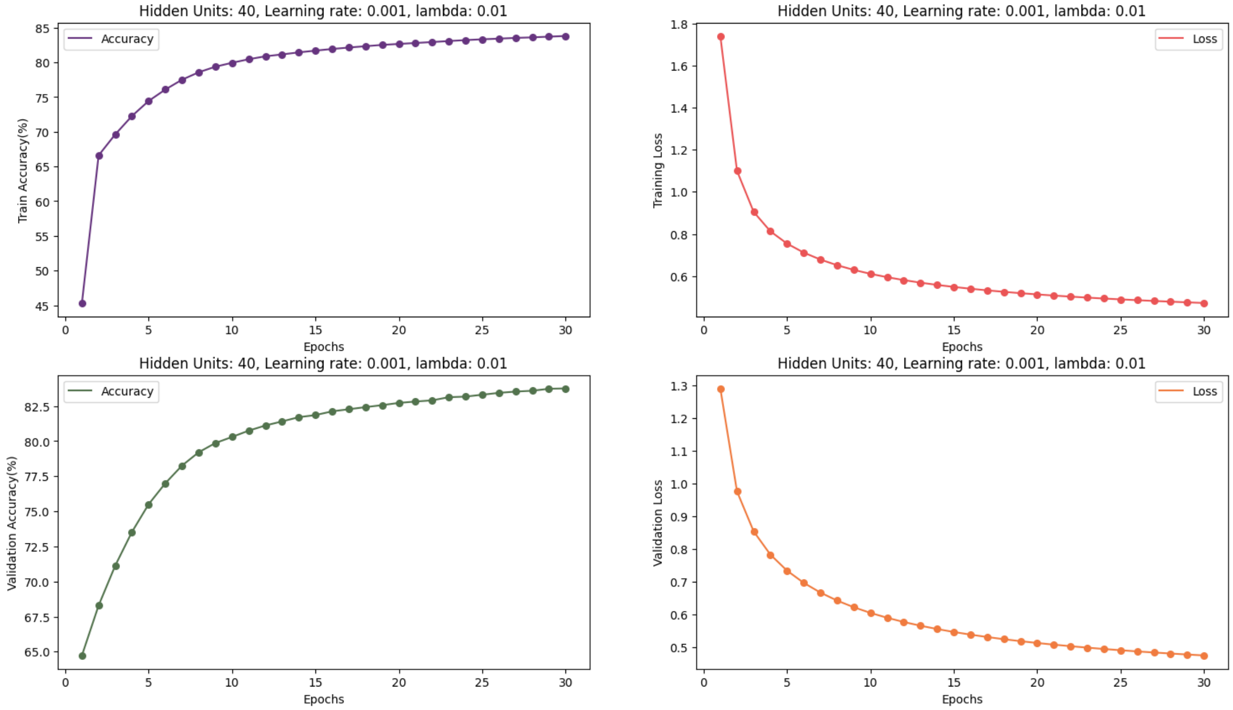
```
40 0.001 0.0001
```



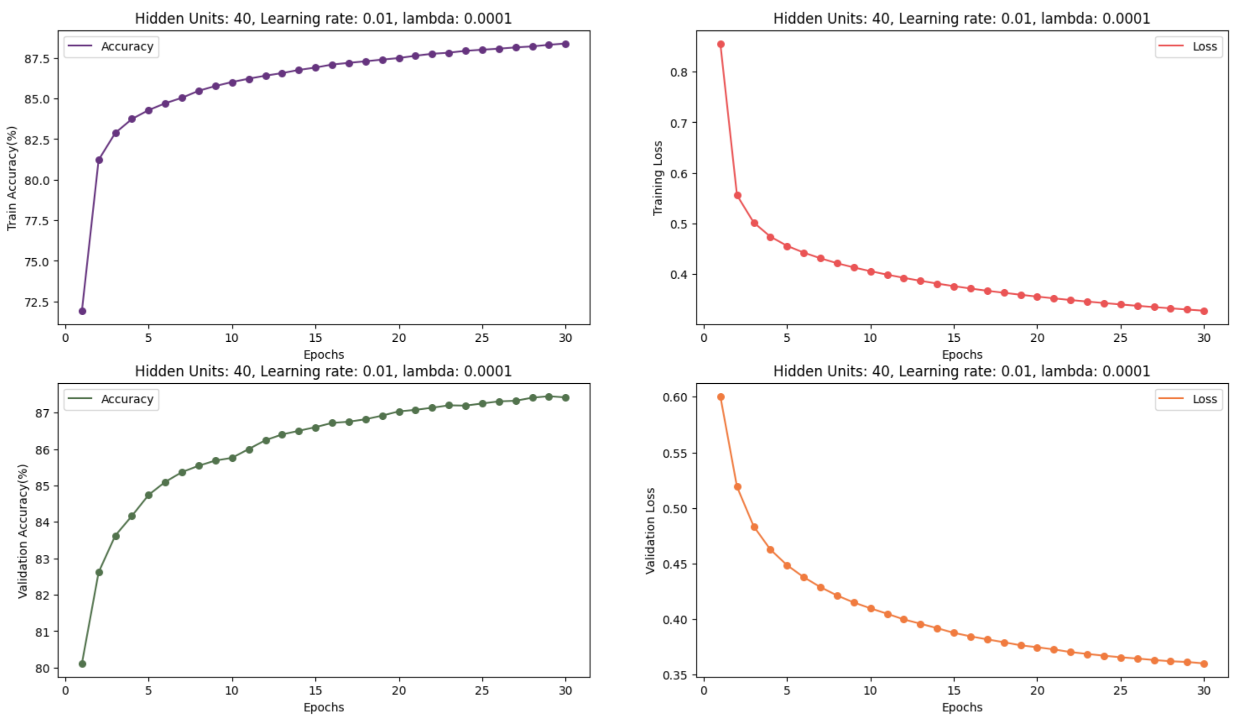

```
40 0.001 0.001
```



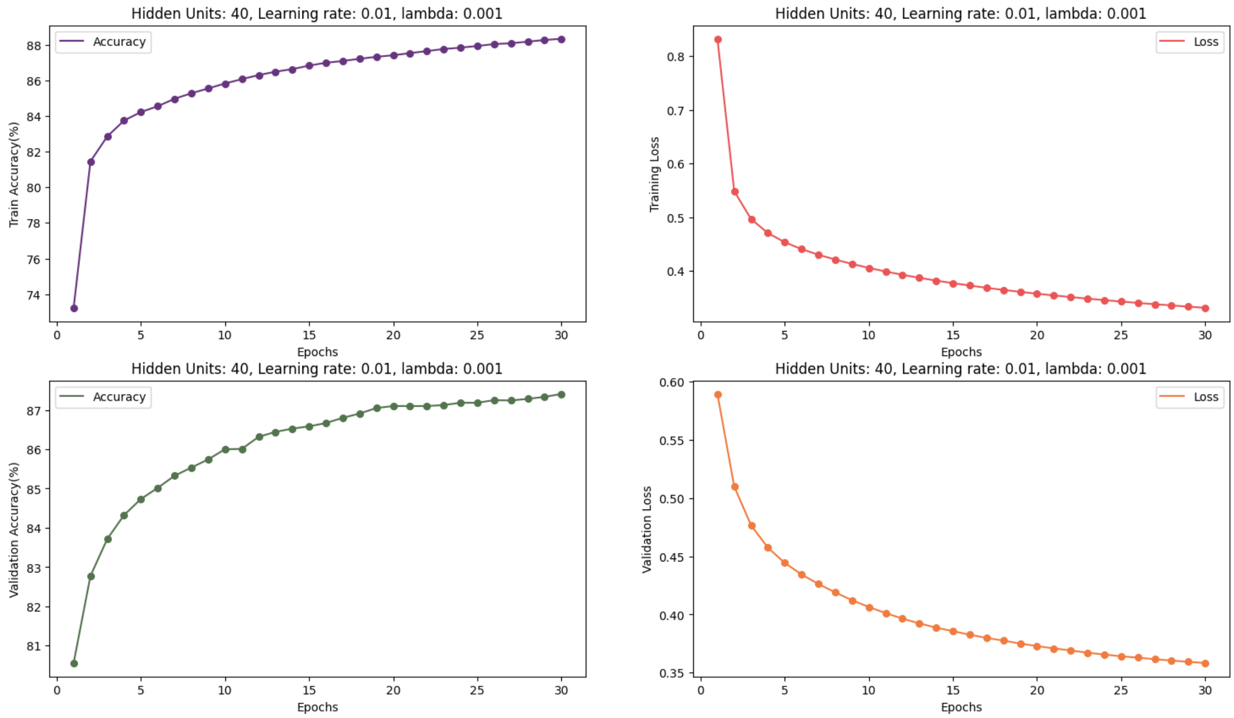

```
40 0.001 0.01
```



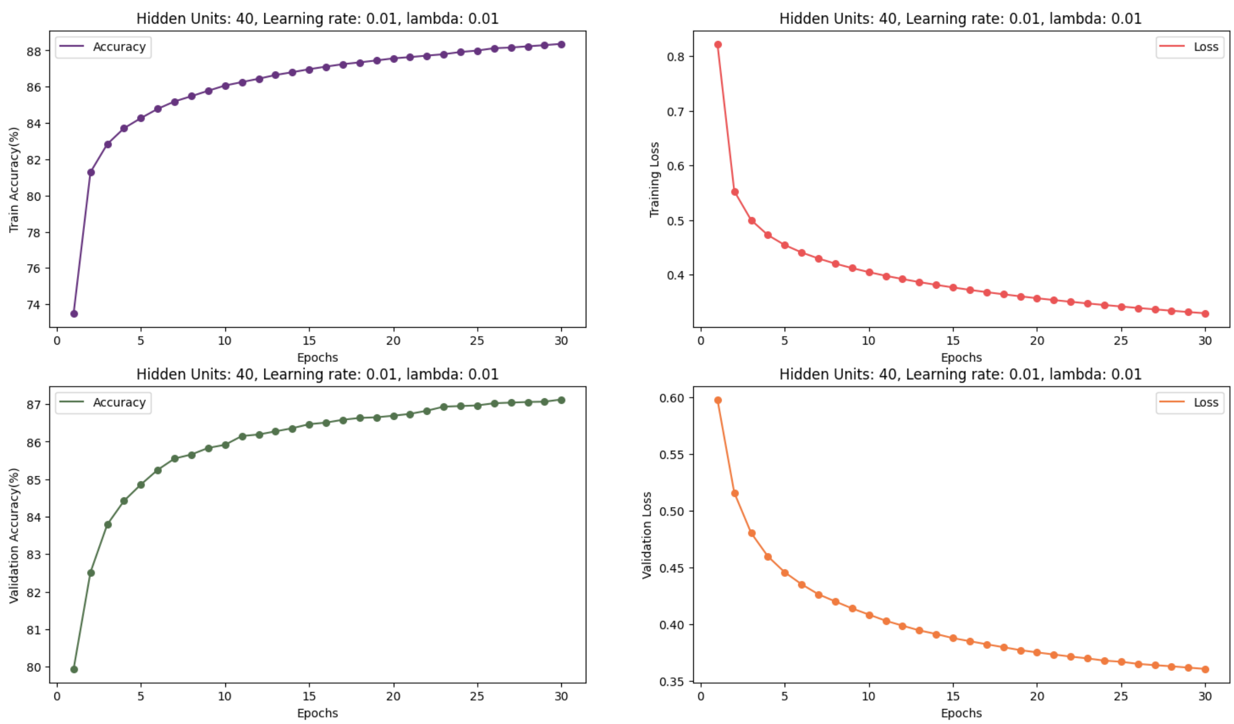

```
40 0.01 0.0001
```



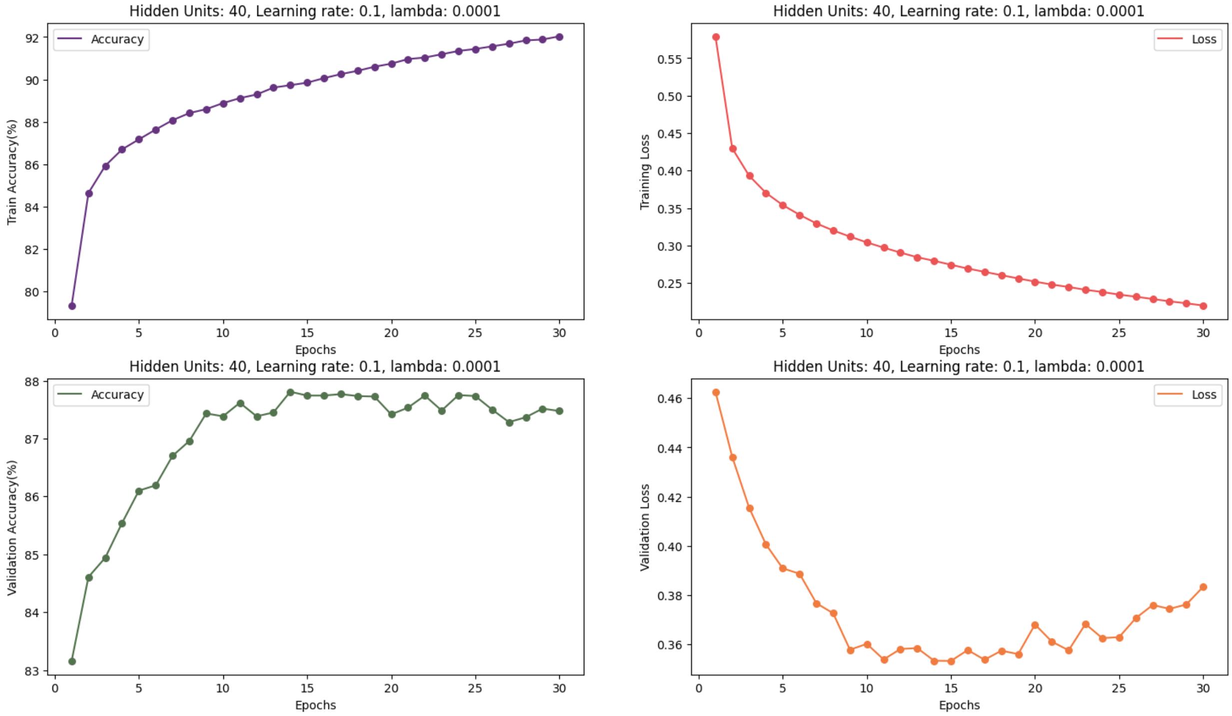

```
40 0.01 0.001
```



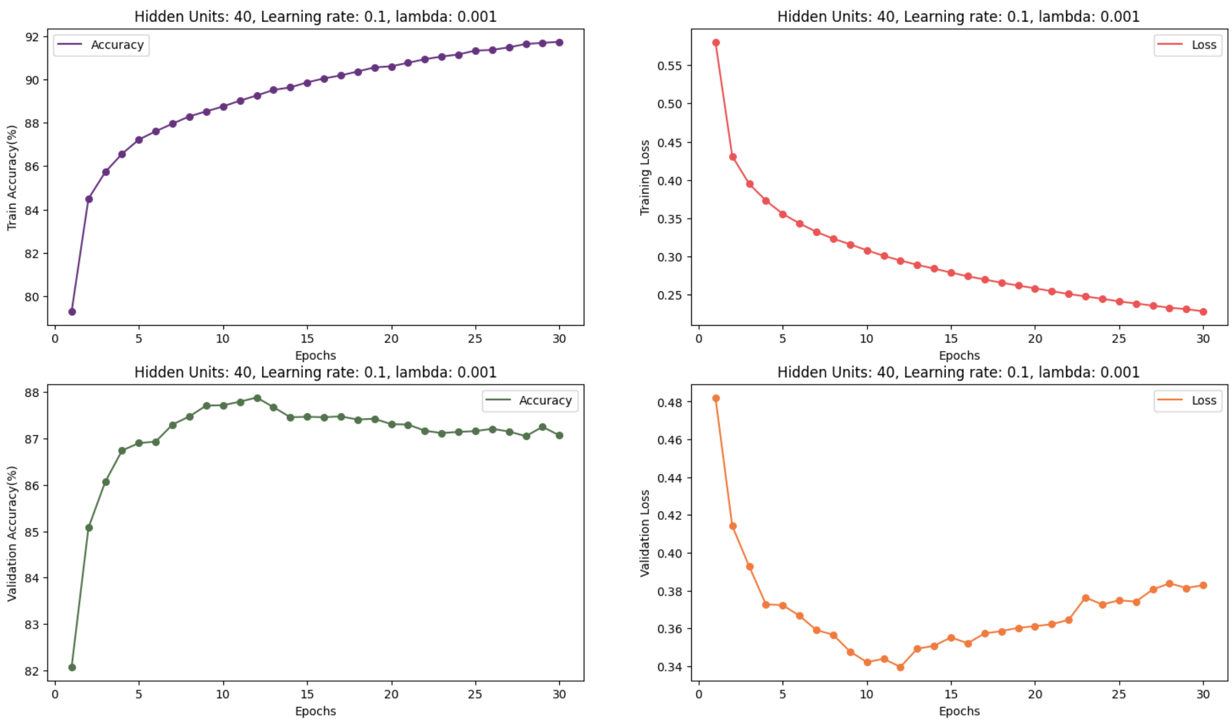
40 0.01 0.01



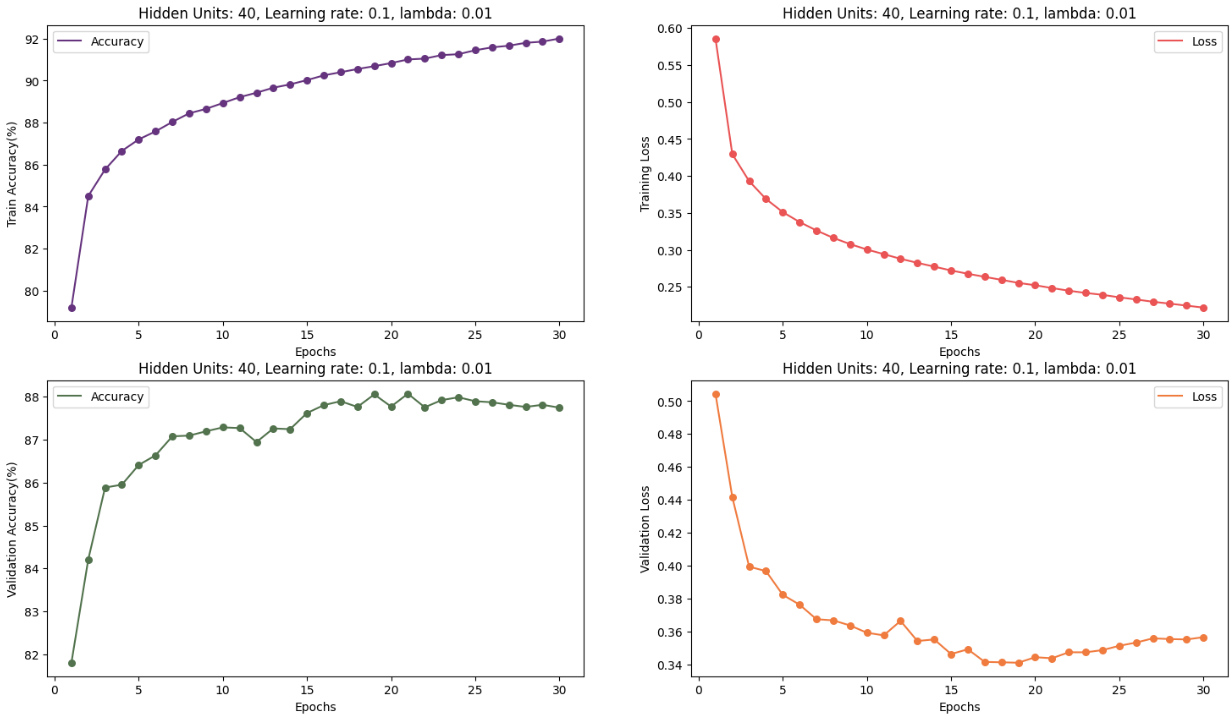
40 0.1 0.0001



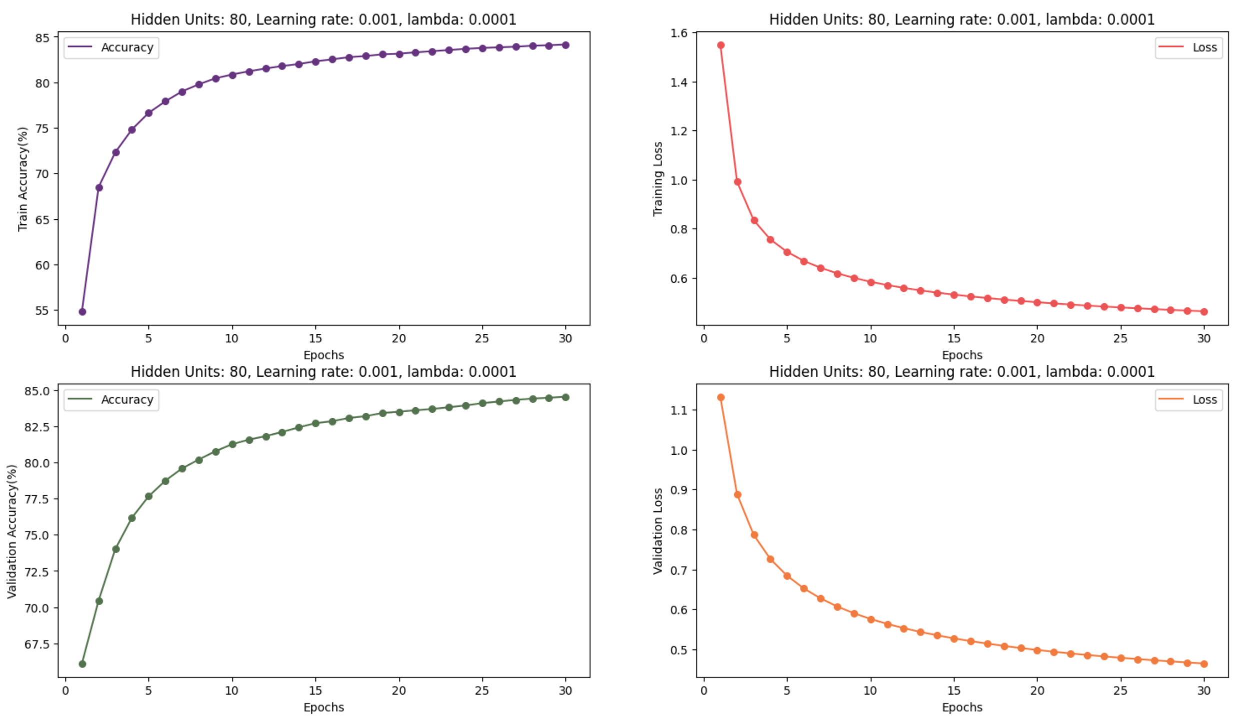
40 0.1 0.001



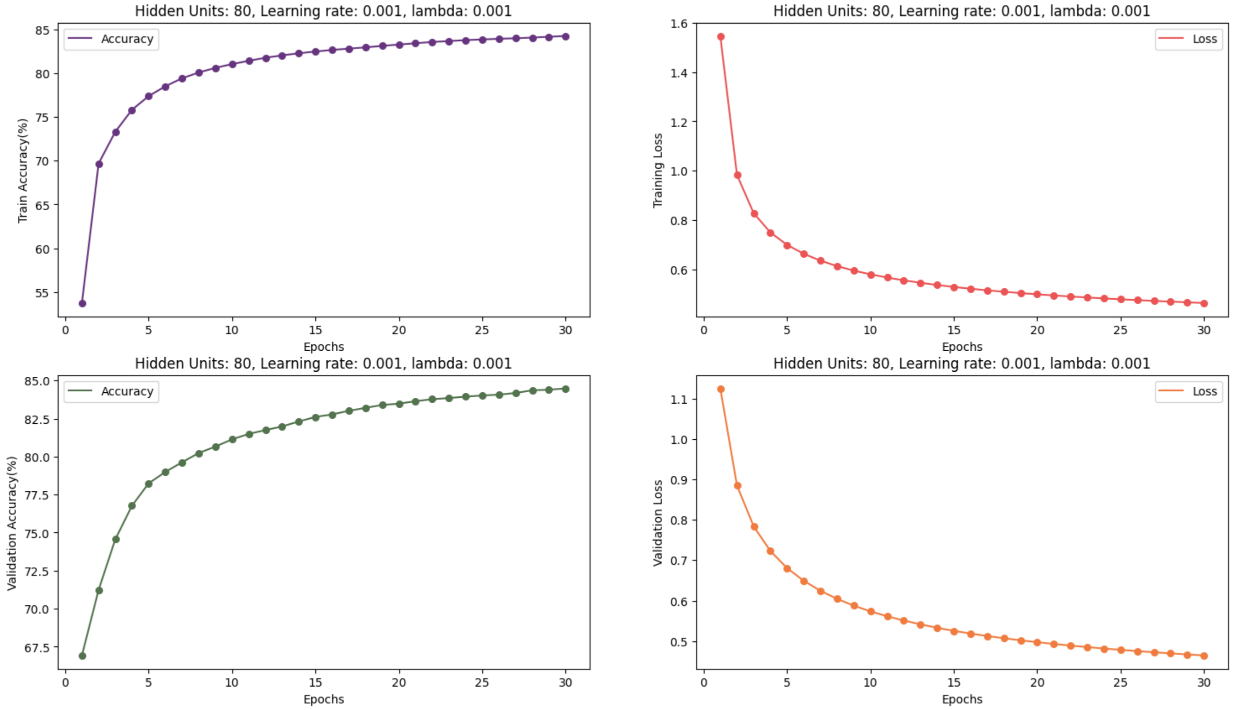
40 0.1 0.01



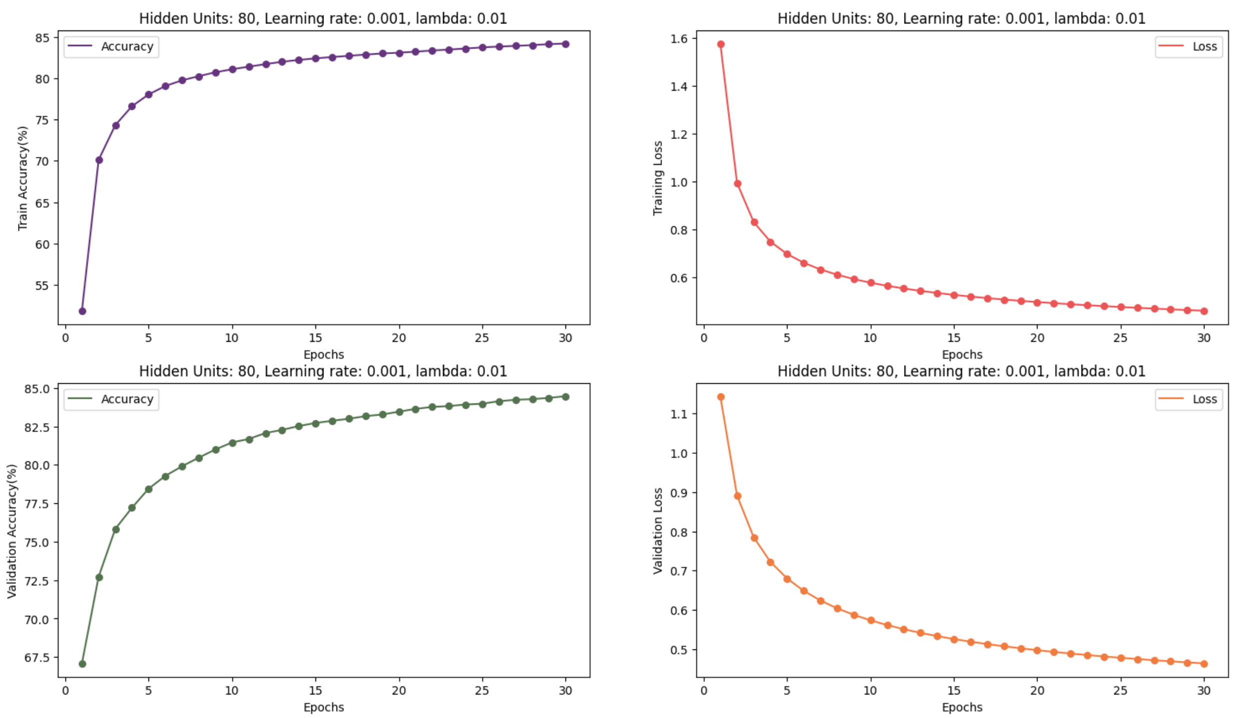

```
80 0.001 0.0001
```



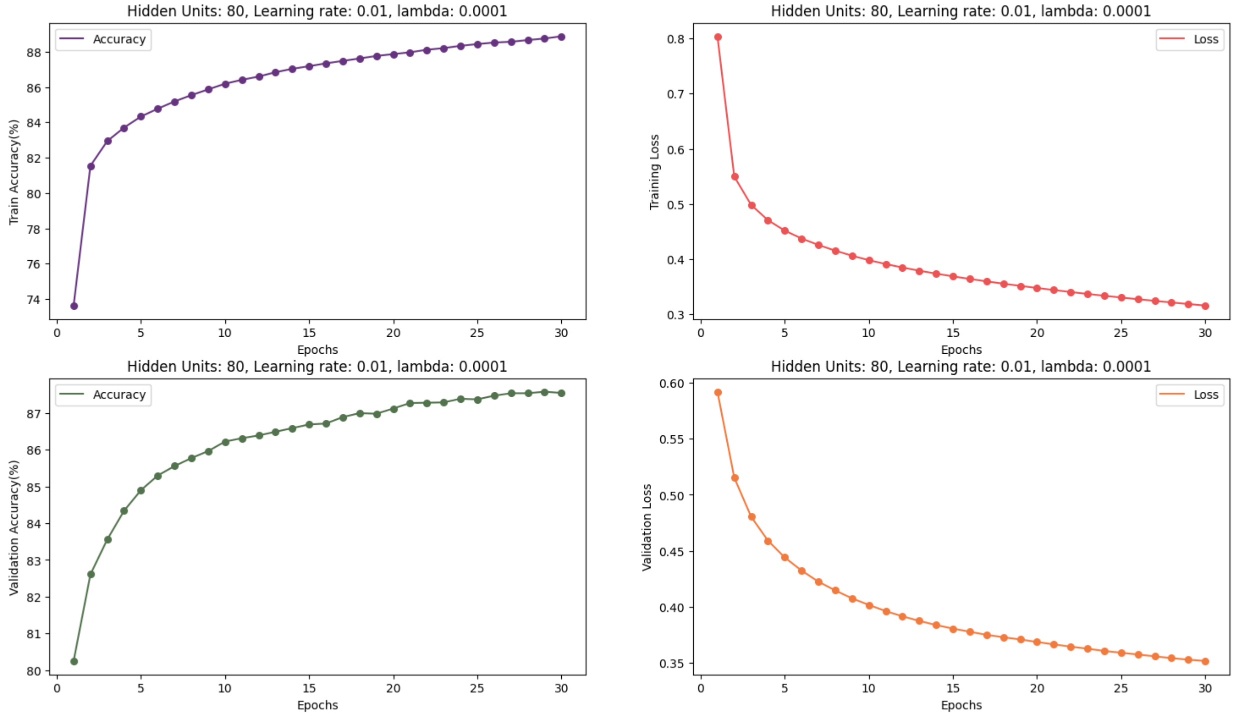

```
80 0.001 0.001
```



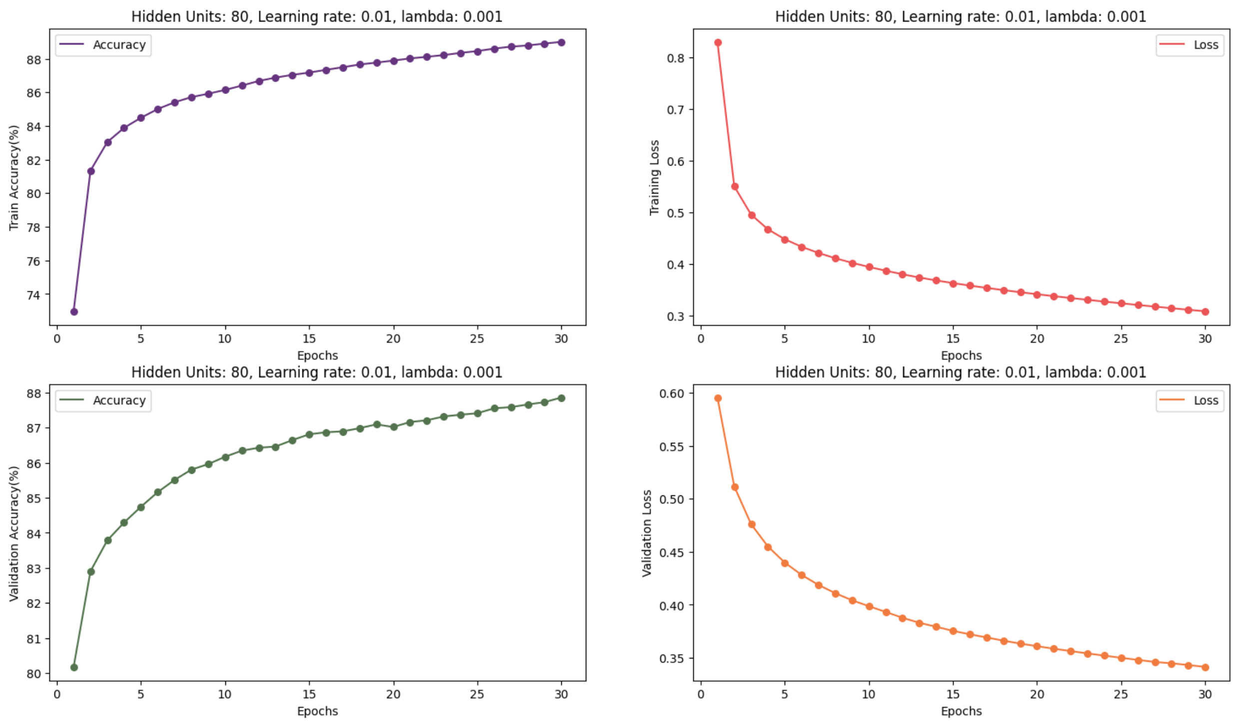

```
80 0.001 0.01
```



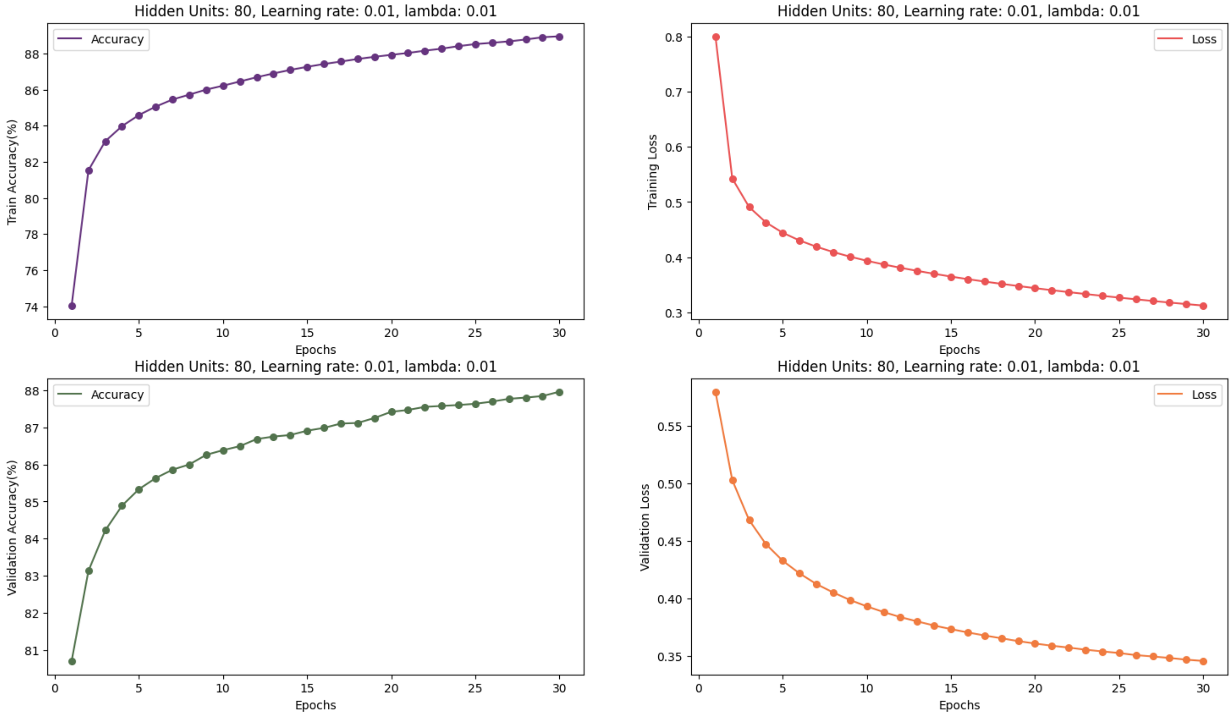

```
80 0.01 0.0001
```



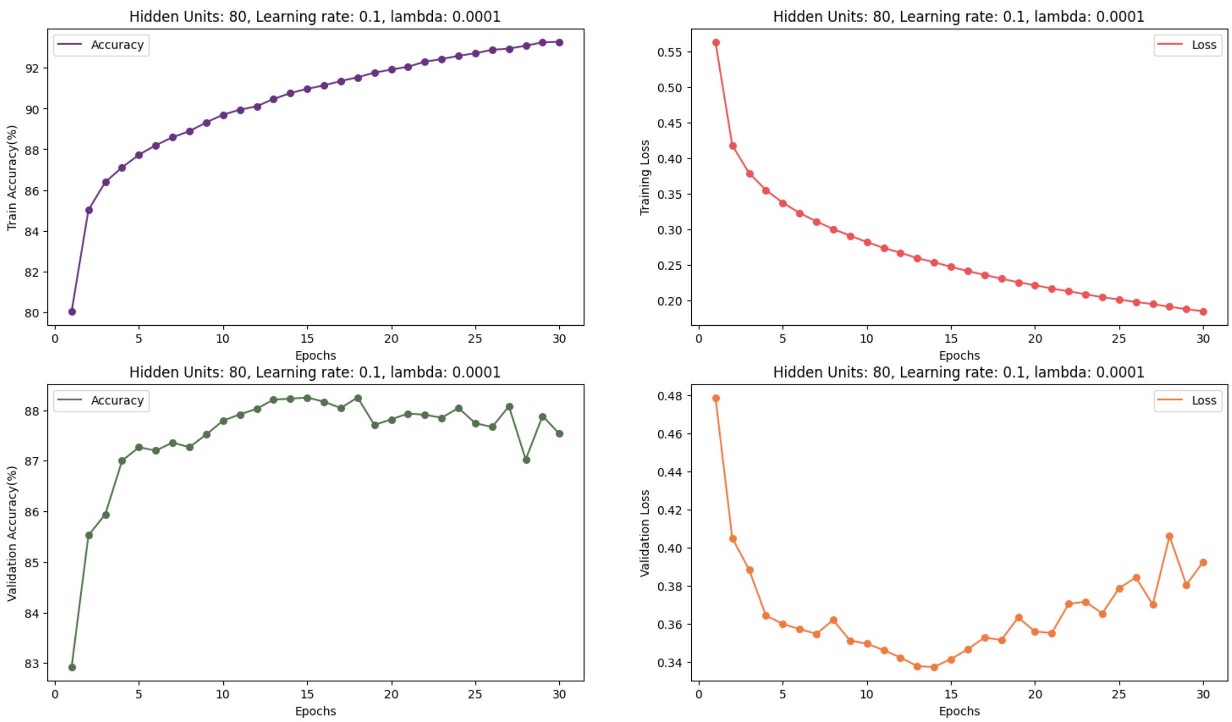

```
80 0.01 0.001
```



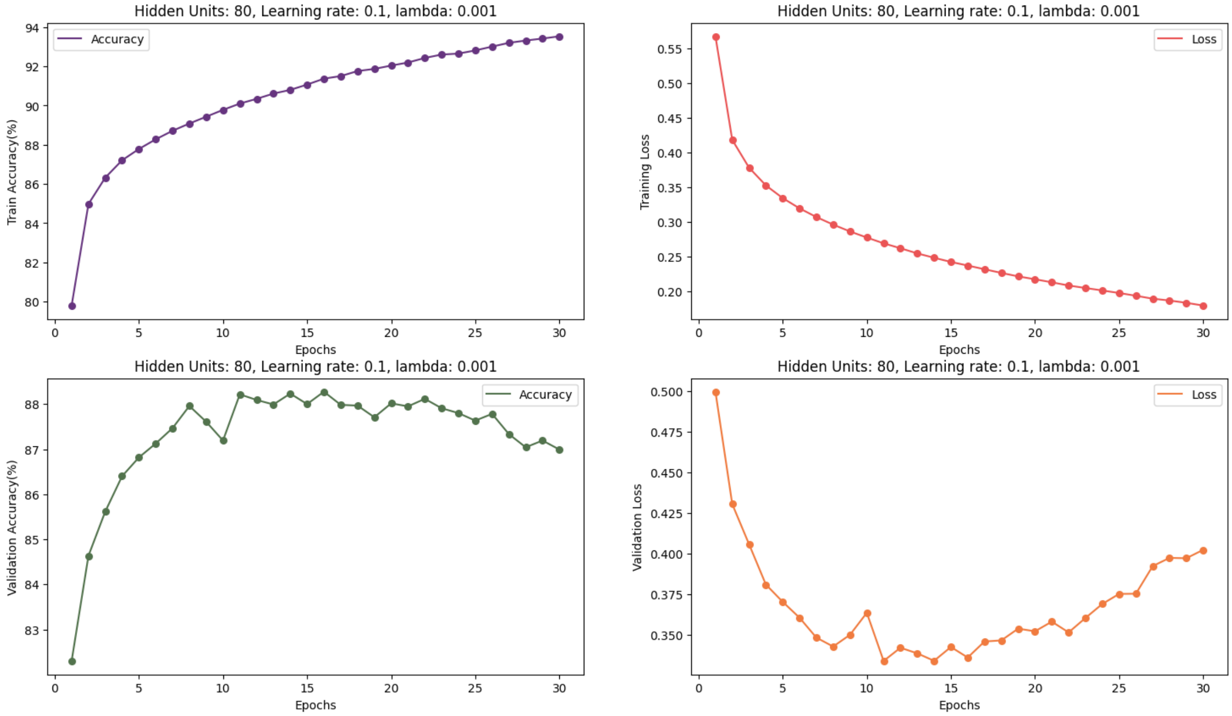

```
80 0.01 0.01
```



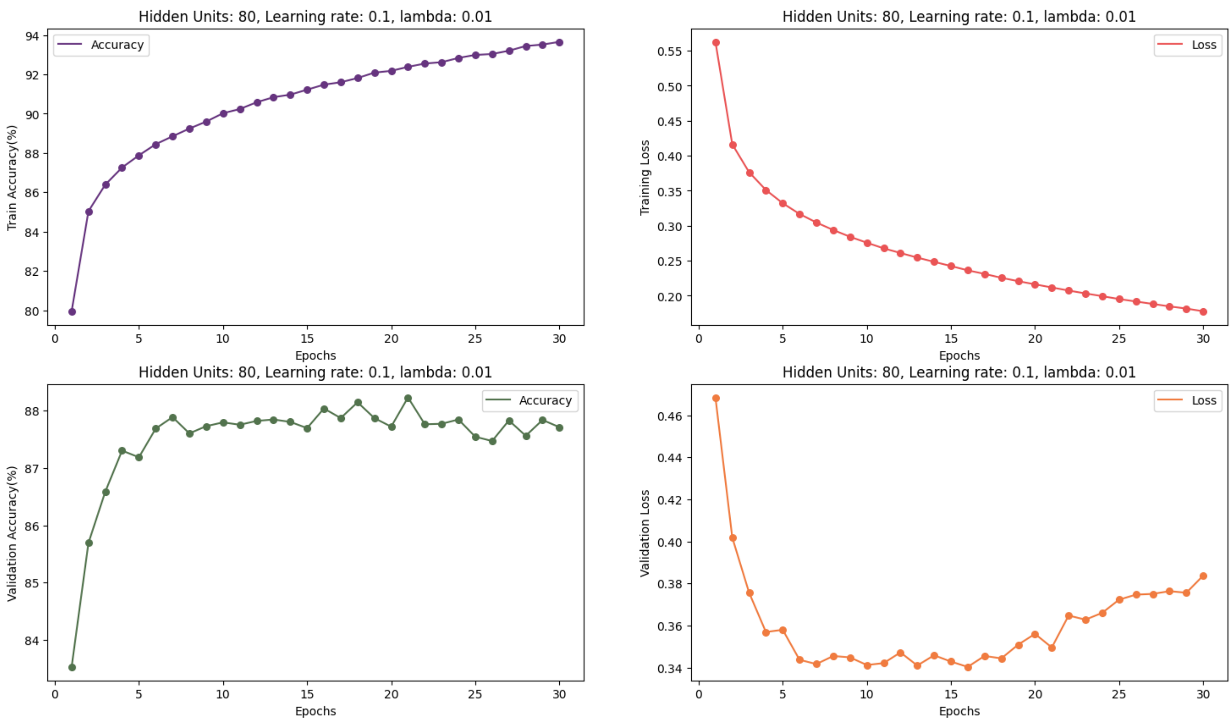
80 0.1 0.0001



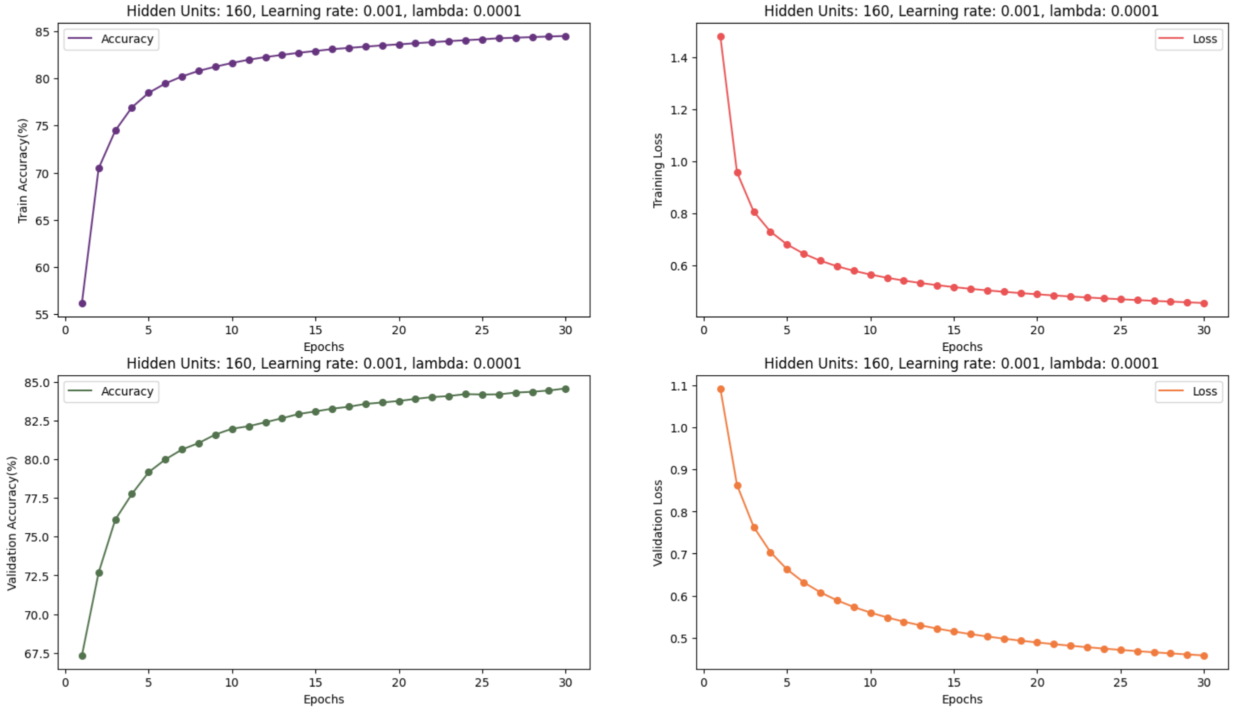
80 0.1 0.001



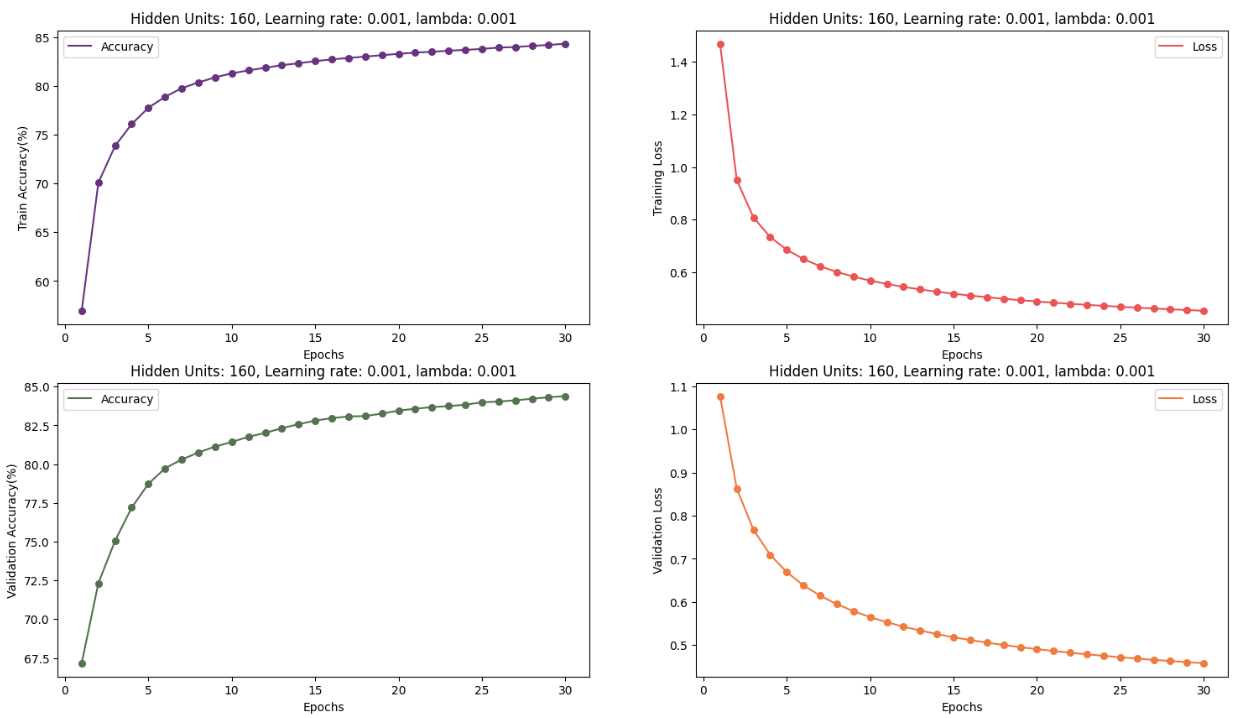
80 0.1 0.01



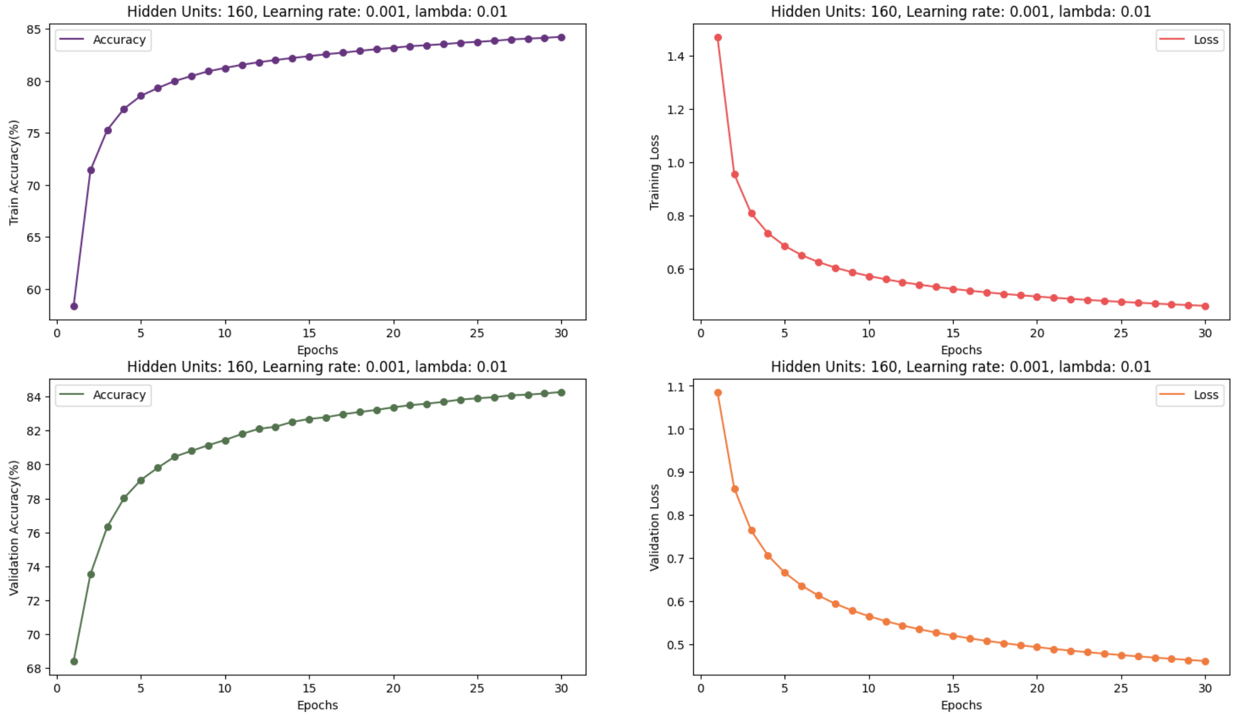
160 0.001 0.0001



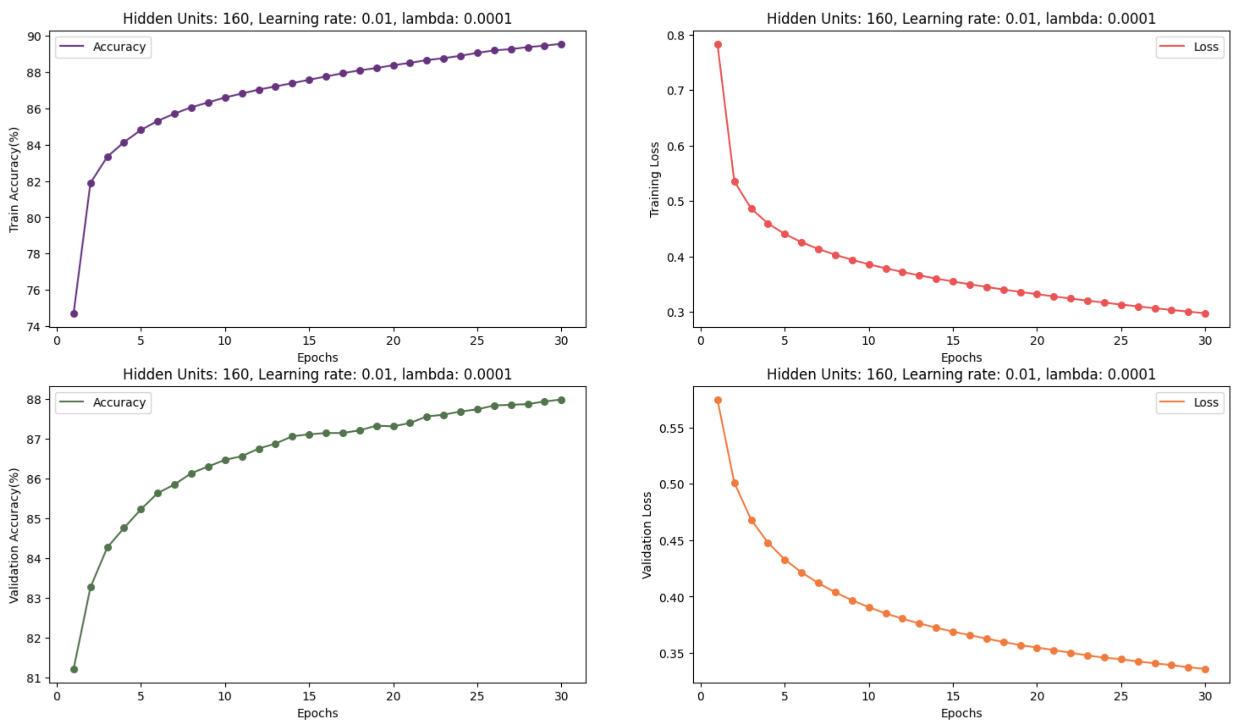
160 0.001 0.001



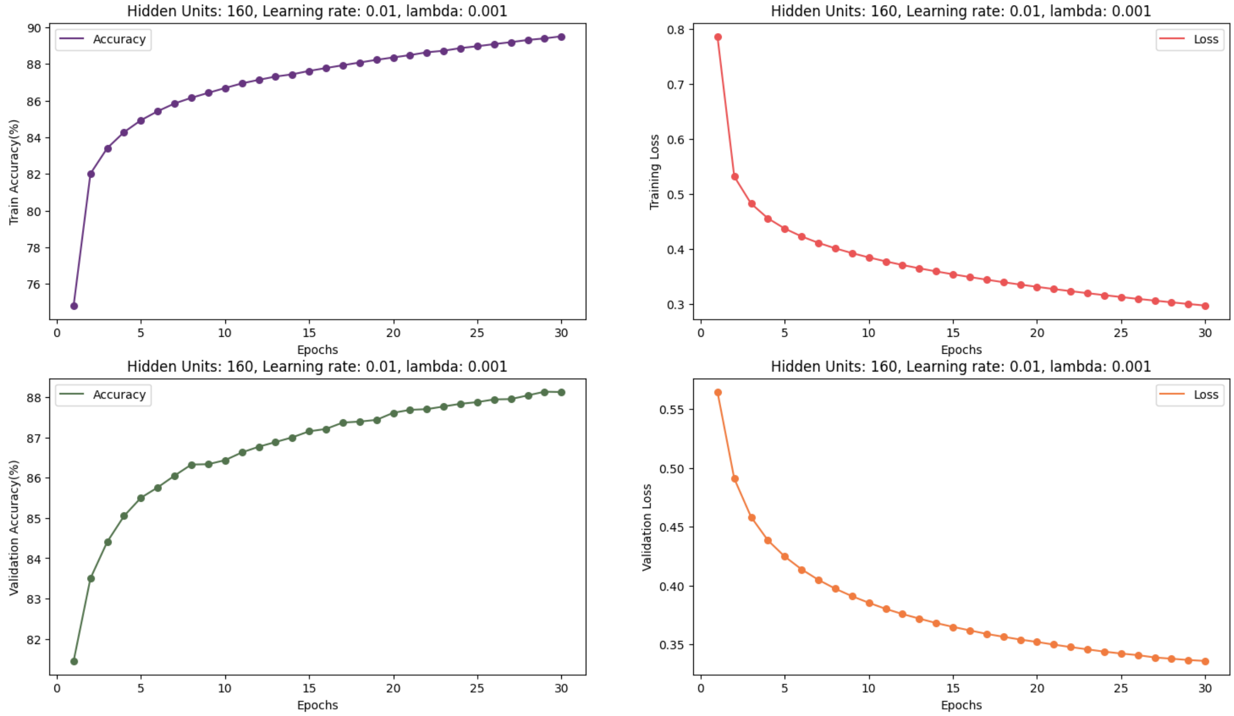
160 0.001 0.01



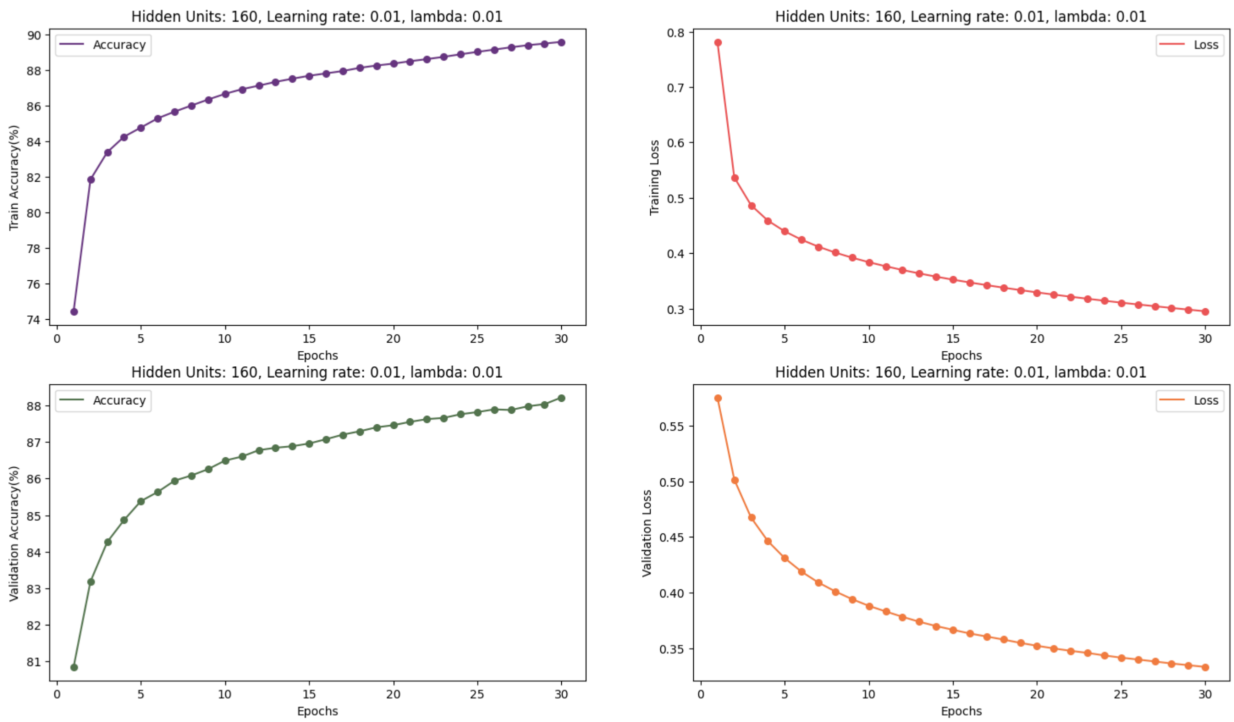

```
160 0.01 0.0001
```



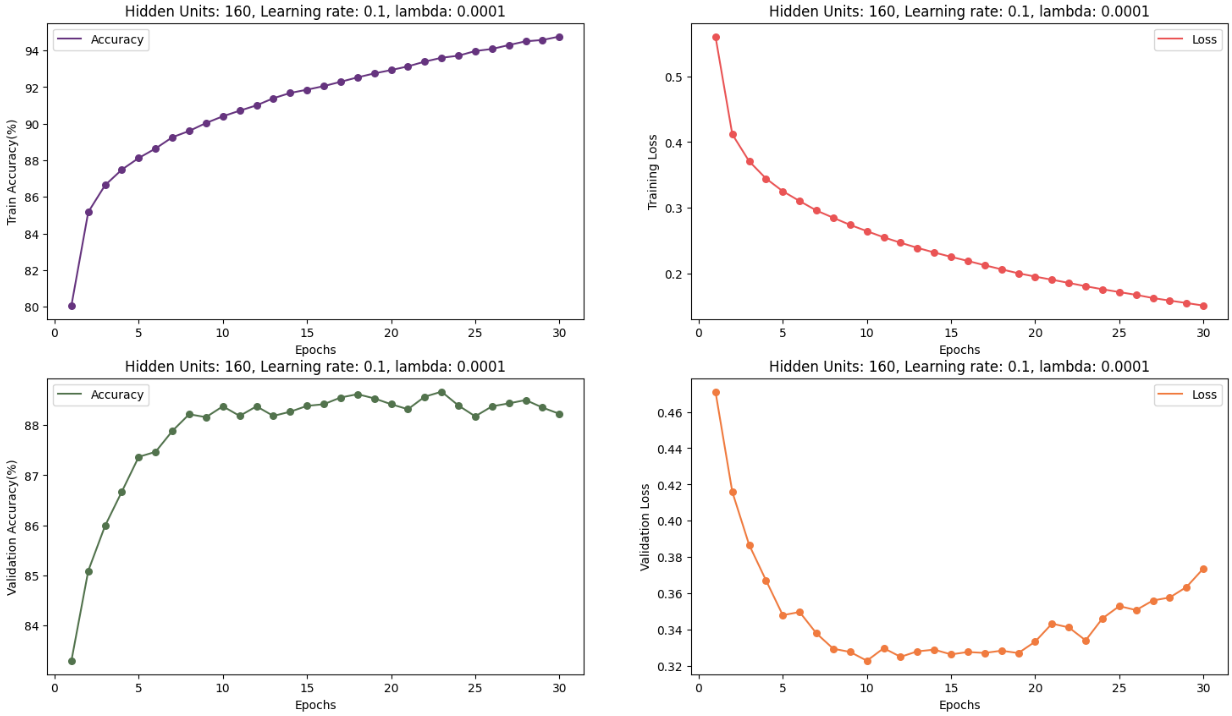

```
160 0.01 0.001
```



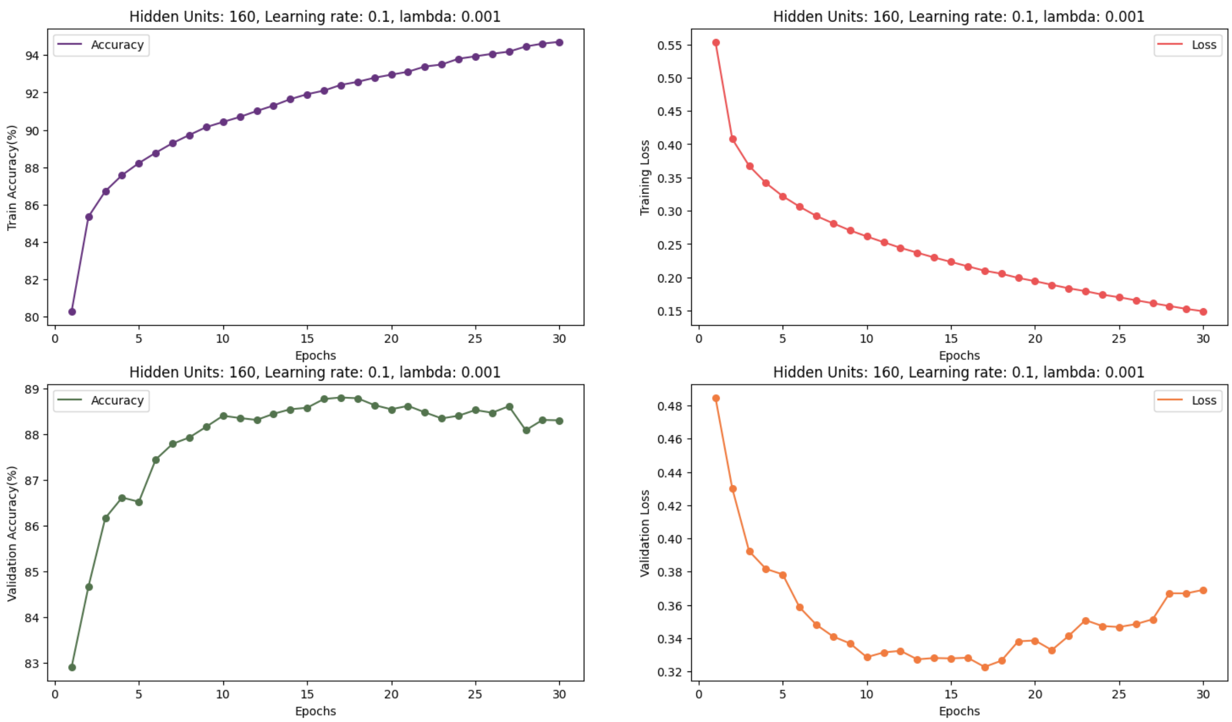

```
160 0.01 0.01
```



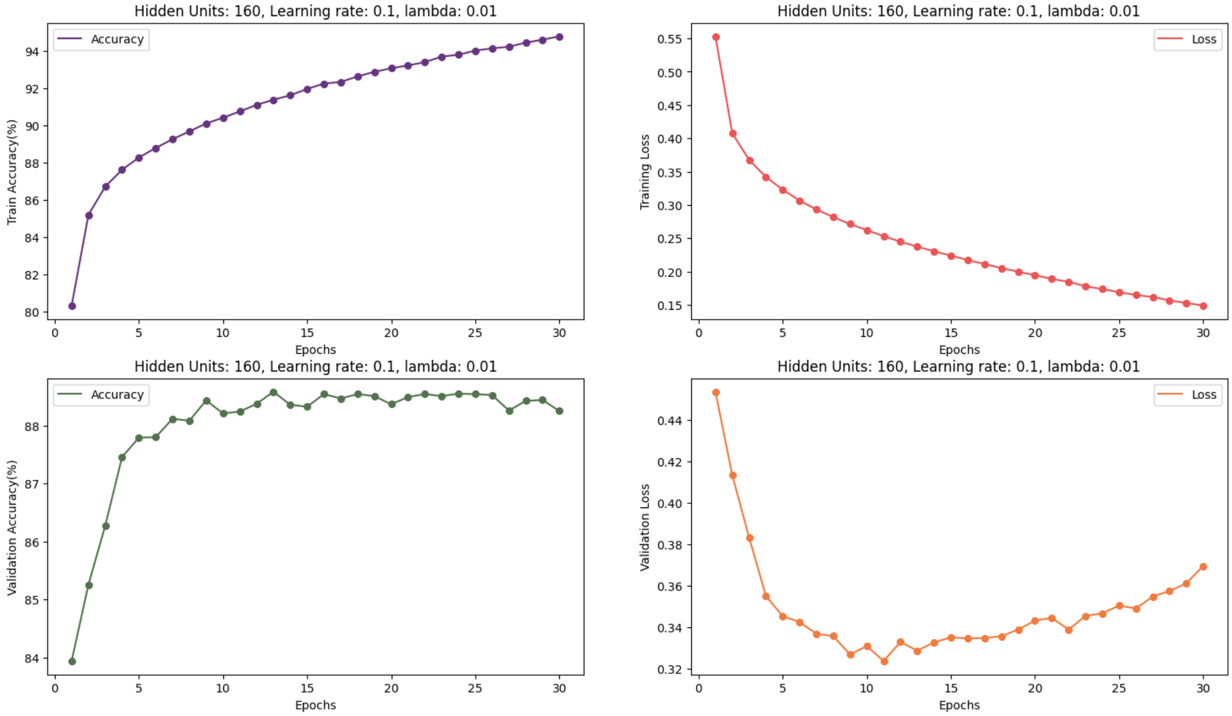

```
160 0.1 0.0001
```




```
160 0.1 0.001
```




```
160 0.1 0.01
```


Hidden Units: 160, Learning rate: 0.1, lambda: 0.01
Accuracy
Train Accuracy(%)
Epochs
Hidden Units: 160, Learning rate: 0.1, lambda: 0.01
Loss
Training Loss
Epochs
Hidden Units: 160, Learning rate: 0.1, lambda: 0.01
Accuracy
Validation Accuracy(%)
Epochs
Hidden Units: 160, Learning rate: 0.1, lambda: 0.01
Loss
Validation Loss
Epochs

Homework : 7 Machine Learning - 1 (Supervised Methods)

c) For the best hyper-parameters found in part (b), run 5 training runs out to 100 epochs. Report the best accuracy (over epochs) on val for each run - this is 5 numbers. Compute, mean, max, and std deviation for these 5 values.

Importing all necessary libraries

Loading the saved training and test data

Instantiating an object for the Classifier class in engine.py

Splitting the training data into train data and validation data (80/20 split)

```
----------------------------------------------------------------------------
--------------------
Total number of images in Training data:  48000
Total number of images in Validation data:  12000
Total number of images in Test data:  10000
Total number of classes in the output lables: 10
----------------------------------------------------------------------------
--------------------
```

Normalizing the pixel values.

Getting the best hyper-parameters from the "finalCombo.pkl" file saved in the "Results" Directory and also fixing the batch size for the training and validation data.

```
Hidden Units: 160, Learning Rate: 0.1, Regularization Parameter: 0.001
```

Training the model with best hyper-parameters for 5 iterations with 100 epochs per iteration and tracking the best validation accuracy over the 100 epochs for every iteration.

```
Model: "my_model_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_Layer (Flatten)       (None, 784)               0

 Hidden_Layer (Dense)        (None, 160)               125600

 Output_Layer (Dense)        (None, 10)                1610

=================================================================
Total params: 127,210
Trainable params: 127,210
Non-trainable params: 0
_________________________________________________________________
```

Out[12]:
```
[<tf.Tensor: shape=(), dtype=float32, numpy=88.9>,
 <tf.Tensor: shape=(), dtype=float32, numpy=88.74167>,
 <tf.Tensor: shape=(), dtype=float32, numpy=88.566666>,
 <tf.Tensor: shape=(), dtype=float32, numpy=88.941666>,
 <tf.Tensor: shape=(), dtype=float32, numpy=88.625>]
```

Out[13]:
```
{13: <tf.Tensor: shape=(), dtype=float32, numpy=88.9>,
 35: <tf.Tensor: shape=(), dtype=float32, numpy=88.74167>,
 16: <tf.Tensor: shape=(), dtype=float32, numpy=88.566666>,
 19: <tf.Tensor: shape=(), dtype=float32, numpy=88.941666>,
 20: <tf.Tensor: shape=(), dtype=float32, numpy=88.625>}
```

Computing Mean, Max and Standard Deviation for the 5 Highest Validation Accuracies computed.

```
Mean for the 5 Best Validation Accuracy Values: 88.75498962402344
Standard Deviation for the 5 Best Validation Accuracy Values: 0.147252887
4874115
Maximum Value from the 5 Best Accuracy Values: 88.94166564941406
```

Homework : 7 Machine Learning - 1 (Supervised Methods)

(d) Take best model from part c (highest val accuracy) and evaluate on test. Report the test accuracy. Report the number of trainable parameters and all hyper-parameters used to obtain this final best model.

Import all necessary libraries.

Loading the test images and labels from the saved dataset. Creating an instance for the Classifier class

Normalizing the pixel values on the test data.

Getting the best hyper-parameters from the "finalCombo.pkl" file saved in the "Results" Directory and Creating test_ds tio generate batches of 32

```
Hidden Units: 160, Learning Rate: 0.1, Regularization Parameter: 0.001
```

Testing the data with best hyper-parameters. We need to build the same model and load the trained weights and report the test accuracy.

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

 Hidden_Layer (Dense)        (None, 160)               125600

 Output_Layer (Dense)        (None, 10)                1610

=================================================================
Total params: 127,210
Trainable params: 127,210
Non-trainable params: 0
_________________________________________________________________
```

Out[7]:
```
<tensorflow.python.checkpoint.checkpoint.CheckpointLoadStatus at 0x132783
e20>

2023-04-07 19:36:04.948297: I tensorflow/core/common_runtime/executor.cc:
1197] [/device:CPU:0] (DEBUG INFO) Executor start aborting (this does not
indicate an error and you can ignore this message): INVALID_ARGUMENT: You
must feed a value for placeholder tensor 'Placeholder/_1' with dtype uint
8 and shape [10000]
         [[{{node Placeholder/_1}}]]

Test Accuracy: 88.0%
```

Report the number of trainable parameters and all hypter-parameters used to obtain the final besr model.

```
NUMBER OF TRAINABLE PARAMETERS
Model: "my_model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input_Layer (Flatten)       (None, 784)               0

 Hidden_Layer (Dense)        (None, 160)               125600

 Output_Layer (Dense)        (None, 10)                1610

=================================================================
Total params: 127,210
Trainable params: 127,210
Non-trainable params: 0
_________________________________________________________________

HYPER - PARAMETERS FOR THE FINAL BEST MODEL
Hidden Units: 160, Learning Rate: 0.1, Regularization Parameter: 0.001
```

Homework : 7 Machine Learning - 1 (Supervised Methods)

Importing all necessary libraries

Type *Markdown* and LaTeX: α^2

Loading the dataset using numpy

```
Shape of X_train: (4000, 2)
Shape of X_test: (2000, 2)
```

2] (a)

Given that,

$$0 \le x_1 \le 2 \; ; \quad 0 \le x_2 \le 2$$

Hence,

$$\Delta x_1 = 2 - 0 = 2$$

$$\Delta x_2 = 2 - 0 = 2$$

Average spacing, $\alpha = \left(\frac{\Delta x_1 \Delta x_2}{M} \right)^{1/2}$

$$\Rightarrow \alpha = \left(\frac{2 \times 2}{M} \right)^{1/2}$$

$$\alpha = \left(\frac{4}{M} \right)^{1/2}$$

$$\boxed{\alpha = \frac{2}{\sqrt{M}}}$$

Let $\sigma = 5\alpha$

$$\Rightarrow \exp\left\{-\gamma \|\underline{x} - \underline{\mu}_m\|^2\right\} = \exp\left\{-\frac{\|\underline{x} - \underline{\mu}_m\|^2}{2\sigma^2}\right\}$$

$$\Rightarrow \cancel{-}\gamma \|\underline{x} - \underline{\mu}_m\|^2 = \cancel{-}\frac{\|\underline{x} - \underline{\mu}_m\|^2}{2\sigma^2}$$

$$\Rightarrow \gamma = \frac{1}{2\sigma^2}$$

$$\gamma = \frac{1}{50\alpha^2}$$

$$\because \alpha = \frac{2}{\sqrt{M}}$$

$$\boxed{\gamma = \frac{M}{200}}$$

2 (b) For comparison to the below systems, compute the RMSE of a trivial system that always outputs the sample mean value y on the training-set data.

2] (b) RMSE of the Trivial System

```
The Root Mean Squared Error is: 3.2035150890062902
```

RMSE is 3.2035150890062902

2] (c) Choose the basis function centers as the data points: μm = xm , m = 1,2,!, N , in which N is the number of training data points during each fold in cross validation. For this part, the only hyperparameter to choose during model selection is γ.

2] (c) i] Use MSE linear regression for the second layer, without regularization.

2] (c) iii] Report on the cross validation RMSE for each value (c) or pair of values ((d) or (e)) tried, in 2 tables: one table for RMSE (mean over the 4 folds) and one table for RMSE (standard deviation over the 4 folds).

Tabular Representation of Mean RMSE values

Gamma	Mean RMSE Train
0.15	1.0868751469897535
1.5	0.028192064976774445
15.0	3.127161418987823e-08
150.0	1.928418153651089e-12
1500.0	1.7348857708271595e-14
15000.0	1.452201992823574e-14

Gamma	Mean RMSE Val
0.15	1.1314046066337953
1.5	0.03319331251953468
15.0	8.449796167385362e-07
150.0	0.0053178191604607156
1500.0	1.7583423636641806
15000.0	2.8682317947183837

Tabular Representation of Std RMSE values

Gamma	STD RMSE Train
0.15	0.04714806346267406
1.5	0.00040685325793167416
15.0	9.978654400414821e-09
150.0	3.8608305022415494e-13
1500.0	1.6824874522002085e-15
15000.0	6.012693147960978e-16

Gamma	STD RMSE Val
0.15	0.05047317452558111
1.5	0.0033024685834434713
15.0	7.55530225216978e-07
150.0	0.0033012769371201407
1500.0	0.35743811678160403
15000.0	0.04684454229686801

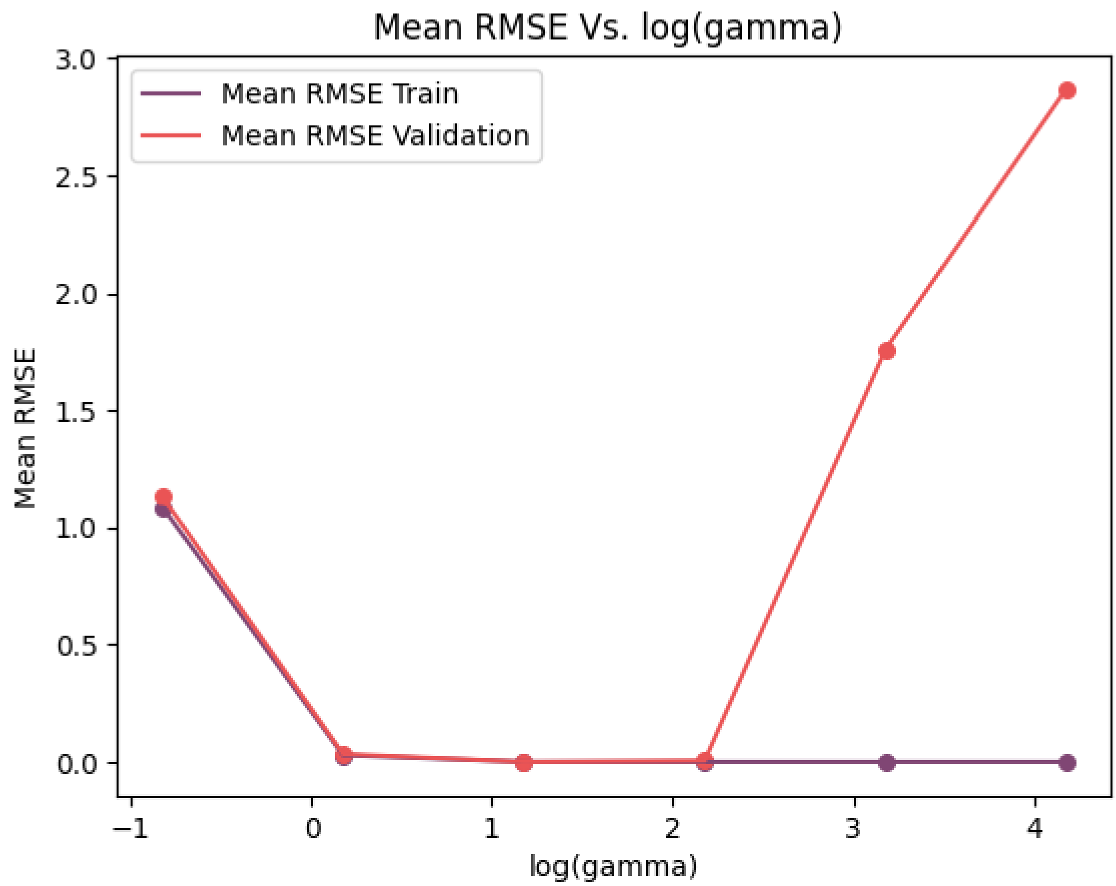
2] (c) iii] Report the best mean value (or pair of values) found.

The best mean value of the validation RMSE: 8.449796167385362e-07

2] (c) ii] Use model selection for finding a good value for γ.

Good value of the hyper-parameter gamma is 15.0

2] (c) iv] Plot training and validation RMSE vs. γ.



2] (d) Randomly choose the basis function centers, without replacement, from the training-set data. Use number of basis function centers M varying from 30 to 300 (e.g., values 30, 60, 100, 300, 600). In this part you have 2 hyperparameters to find during model selection (γ and M).

2] (d) i] Use MSE linear regression for the second layer, without regularization.

Creating a dictionary of key = m and values = list of mean of RMSE values (or) values = list of std of RMSE values.

2] (d) iii] Report on the cross validation RMSE for each value (c) or pair of values ((d) or (e)) tried, in 2 tables: one table for RMSE (mean over the 4 folds) and one table for RMSE (standard deviation over the 4 folds).

Printing the Tabular Representation of Mean and STD Value.

m = 30

Gamma	Mean RMSE Val
0.0015	4.715437036261309
0.015	1.948948718591237
0.15	1.6603193000926173
1.5	1.6091639751878826
15.0	1.7884678384010153
150.0	2.688934948496533

m = 60

Gamma	Mean RMSE Val
0.003	2.2015182082183395
0.03	1.6477424871512572
0.3	1.1573506985561315
3.0	0.813169495044279
30.0	1.531850531111318
300.0	2.776652318521858

m = 100

Gamma	Mean RMSE Val
0.005	3.018133266823477
0.05	1.4618910165333763
0.5	0.6628962513997799
5.0	0.1827883907734456
50.0	1.1698240829871596
500.0	2.8120144582013307

m = 300

Gamma	Mean RMSE Val
0.015	2.126017199711846
0.15	1.2093556462694497
1.5	0.04779848516385086
15.0	0.005249652344125224
150.0	1.0337928701625982
1500.0	2.80507041097321

m = 600

Gamma	Mean RMSE Val
0.03	1.8230663388457058
0.3	0.7066849282946088
3.0	0.0035784055930481932
30.0	0.0030103679520871772
300.0	0.9524367936051109

```
| 3000.0         | 2.8432433932186125       |
+----------------+--------------------------+
```

```
m = 30
+-----------------+--------------------------+
| Gamma           | STD RMSE Val             |
+-----------------+--------------------------+
| 0.0015          | 1.421217018441914        |
| 0.015           | 0.02828313584108904      |
| 0.15            | 0.116208806245442        |
| 1.5             | 0.030355506189749203     |
| 15.0            | 0.2098646639340821       |
| 150.0           | 0.11479887859794236      |
+-----------------+--------------------------+

m = 60
+-----------------+--------------------------+
| Gamma           | STD RMSE Val             |
+-----------------+--------------------------+
| 0.003           | 0.13758152227366904      |
| 0.03            | 0.13911330256092586      |
| 0.3             | 0.009331984859444182     |
| 3.0             | 0.02413007034985364      |
| 30.0            | 0.12134843468194005      |
| 300.0           | 0.058977372802306575     |
+-----------------+--------------------------+

m = 100
+-----------------+--------------------------+
| Gamma           | STD RMSE Val             |
+-----------------+--------------------------+
| 0.005           | 0.3063853395437327       |
| 0.05            | 0.04537710463705822      |
| 0.5             | 0.0382322429103942 8     |
| 5.0             | 0.020708538866547825     |
| 50.0            | 0.21200785550904155      |
| 500.0           | 0.09372881531994823      |
+-----------------+--------------------------+

m = 300
+-----------------+--------------------------+
| Gamma           | STD RMSE Val             |
+-----------------+--------------------------+
| 0.015           | 0.27037069631699884      |
| 0.15            | 0.01361300179130973      |
| 1.5             | 0.007979593749709754     |
| 15.0            | 0.0001762870033065202    |
| 150.0           | 0.06896733668415417      |
| 1500.0          | 0.04524199408463957      |
+-----------------+--------------------------+

m = 600
+-----------------+--------------------------+
| Gamma           | STD RMSE Val             |
+-----------------+--------------------------+
| 0.03            | 0.23526869758960053      |
| 0.3             | 0.021400130692310153     |
| 3.0             | 0.0004637632826080014    |
| 30.0            | 0.00041869166643692177   |
| 300.0           | 0.0238538900072484 2     |
```

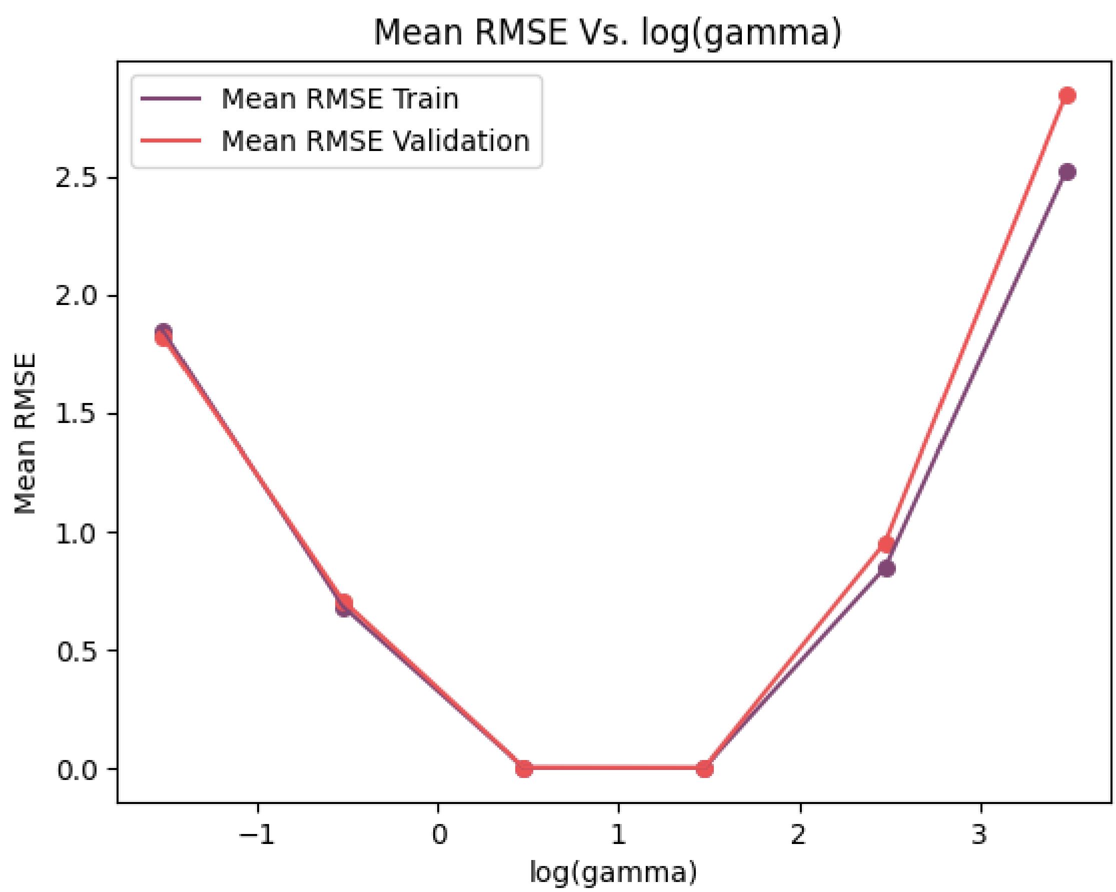
```
| 3000.0         | 0.0125238175771921 27    |
+----------------+--------------------------+
```

2] (d) iii] We can evidently see that for m = 600, gamma = 30, we get the minimum mean RMSE of 0.0030103679520871772

2] (d) ii] The optimized hyper-parameters are gamma = 30 amd M = 600

2] (d) iv] Plot training and validation RMSE vs. γ. (For parts (d) and (e), use your best value of M = M $*$ or K = $K*$ for the plot.)

Plotting Mean RMSE Validation Vs. log(Gamma)



2] (d) v] If computational complexity were an issue, what is the smallest value of M or K (and its associated γ) that would give RMSE at least a factor of 10 lower than the trivial system of (b)?

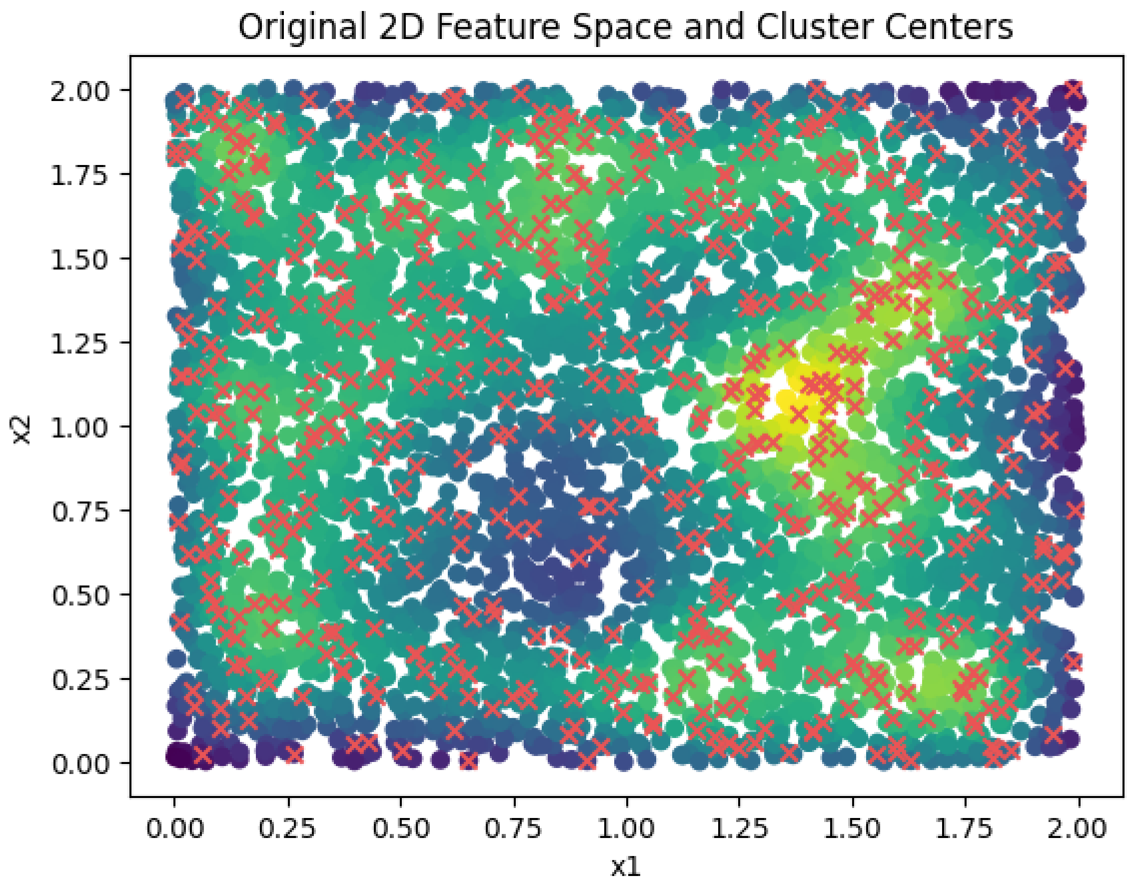
The smallest value of M is 100 and associated gamma = 5.0 with a RMSE = 0.1827883907734456 that would give RMSE at least a factor of 10 lower than the trivial system. This is the RCC Model.

What factor reduction in number of hidden units (dimensionality of the expanded feature space) from the original M=3000 in part (c) does this RCC model represent?

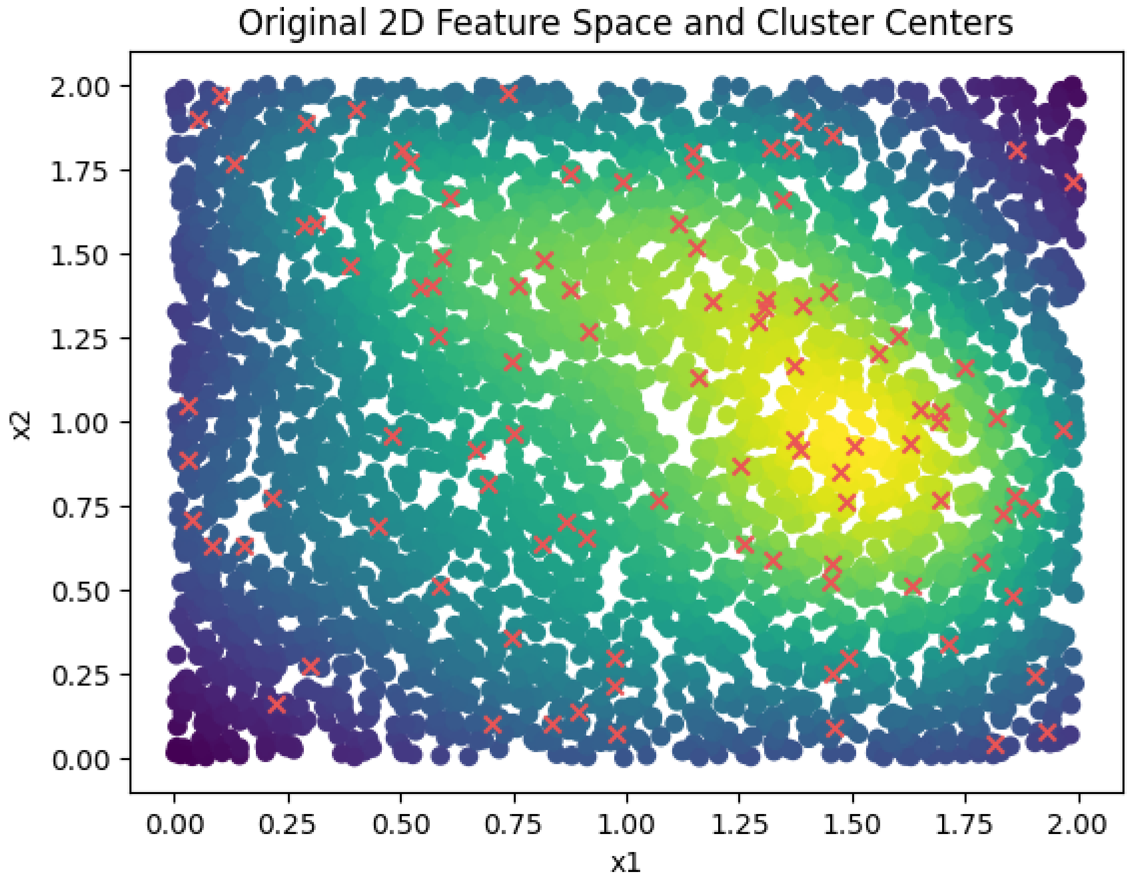
The RCC model represents reduction of hidden units by a factor of 30.

2] (d) vi] For parts (d) and (e), plot in original 2D feature space, the training data points x& and the cluster centers μ' for your best values of hyperparameters. Then repeat the plots for your RCC model, and again for your lowest-complexity-model (M=30 or K=30). (Total of 3 plots for (d))

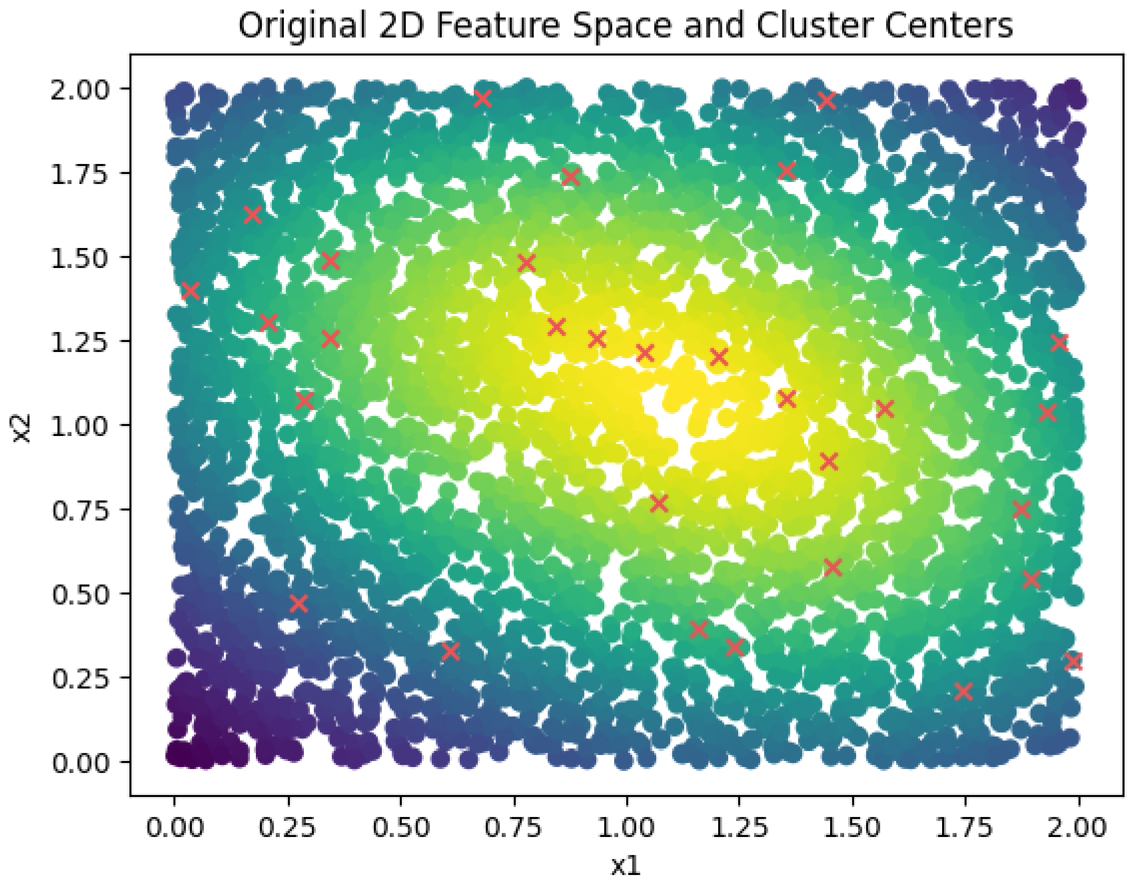
Plot in original 2D feature space, the training data points x& and the cluster centers μ' for your best values of hyperparameters.


Original 2D Feature Space and Cluster Centers
x2
x1
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

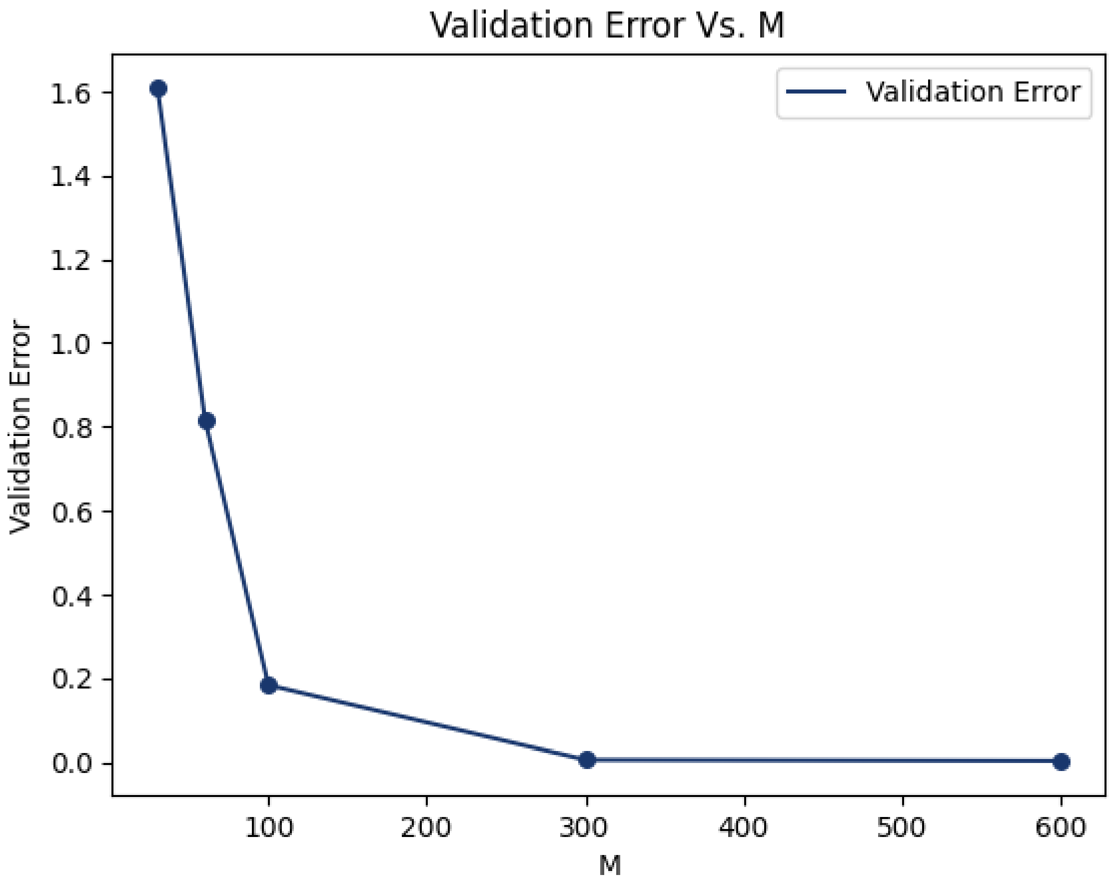
Plot in original 2D feature space, the training data points x& and the cluster centers μ' for the RCC Model. (M = 100, gamma = 5.0)

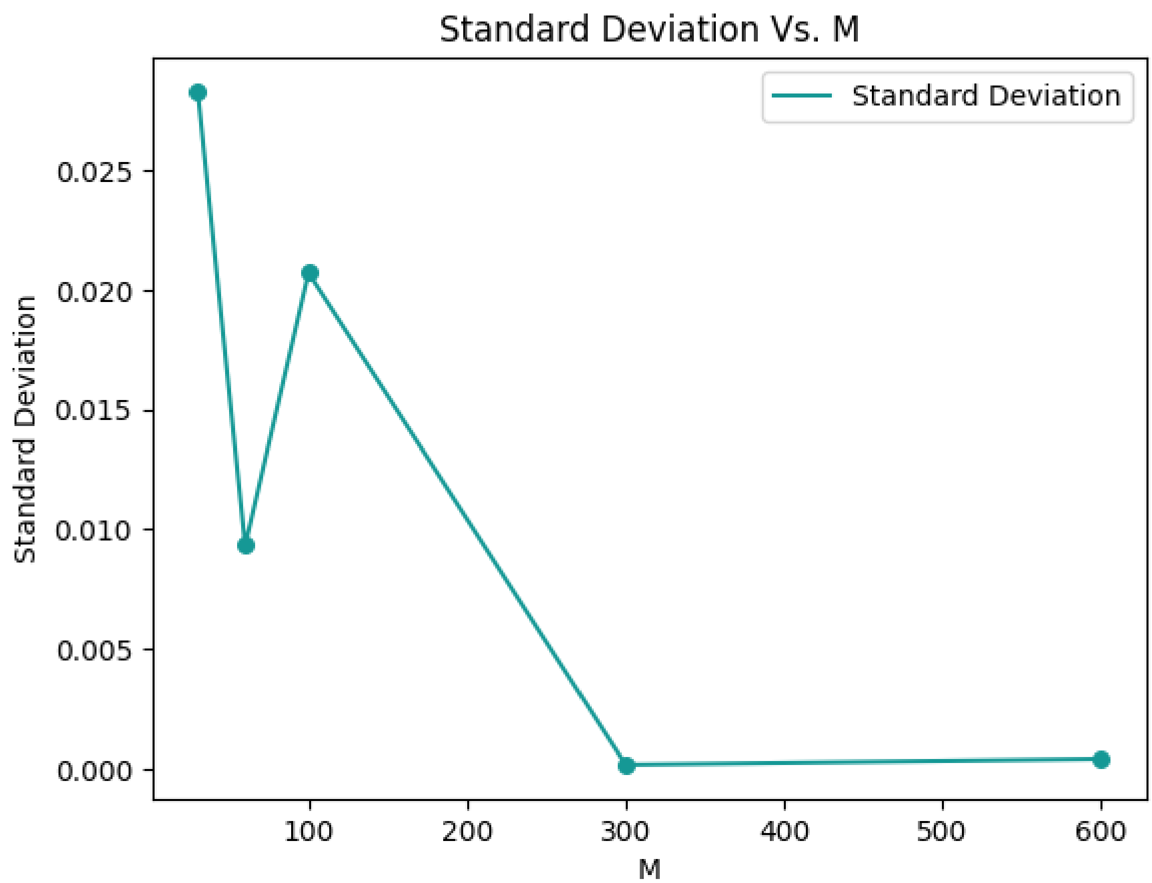


Plot in original 2D feature space, the training data points x& and the cluster centers μ' for the lowest-complexity-model (M=30, gamma = 1.5)



2] (d) vii] Plot the validation error and its standard deviation Vs. the second hyperparameter (M for (d), K for (e)), using the best γ for each value of M or K. (The value of best γ may depend on M or K.)





2] (e) Use K-means clustering to choose basis function centers for a given K; vary K using model selection (e.g., use values 30, 60, 100, 300, 600). For each value of K, choose your initial cluster centers randomly (i.e., in sklearn's K-means).

Creating a dictionary of key = k and values = list of mean of RMSE values (or) values = list of std of RMSE values.

2] (e) iii] Report on the cross validation RMSE for each value (c) or pair of values ((d) or (e)) tried, in 2 tables: one table for RMSE (mean over the 4 folds) and one table for RMSE (standard deviation over the 4 folds).

Printing the Tabular Representation of Mean and STD Value.

Type *Markdown* and LaTeX: α^2

k = 30

Gamma	Mean RMSE Val
0.0015	2.781403014113983
0.015	2.5347499037566887
0.15	1.5140042461841028
1.5	1.6279207364444022
15.0	1.7186802314872114
150.0	2.853963161835545

k = 60

Gamma	Mean RMSE Val
0.003	2.2913001902935104
0.03	1.7601490048119997
0.3	1.1788761966744619
3.0	0.8750528681062978
30.0	0.7191071334561288
300.0	2.735962285777041

k = 100

Gamma	Mean RMSE Val
0.005	3.2841865607298604
0.05	1.4740667601577164
0.5	0.587222792360727
5.0	0.18762609464511065
50.0	0.5974214332690146
500.0	2.7517893867086873

k = 300

Gamma	Mean RMSE Val
0.015	2.3927954181150284
0.15	1.208518651799153
1.5	0.04212384144673394
15.0	0.0068755165422162996
150.0	0.41759919520906
1500.0	2.8550082786616677

k = 600

Gamma	Mean RMSE Val
0.03	1.6696923293209172
0.3	0.7024665458704473
3.0	0.003410785183149106
30.0	0.002454457077742055
300.0	0.5942781739122412

```
| 3000.0          | 3.334740663717139        |
+-----------------+--------------------------+
```

k = 30

Gamma	STD RMSE Val
0.0015	0.2960635706374978
0.015	0.4209720204923199
0.15	0.01755715268095187
1.5	0.03184952394169455
15.0	0.04488304739707133
150.0	0.024915442438118593

k = 60

Gamma	STD RMSE Val
0.003	0.35602230092926107
0.03	0.13701607643903715
0.3	0.005871360806875238
3.0	0.027660201739291686
30.0	0.1751431098107133
300.0	0.05299801245676872

k = 100

Gamma	STD RMSE Val
0.005	0.47013394623247967
0.05	0.040182440651513864
0.5	0.03049707674163371
5.0	0.024497069240162074
50.0	0.06321150626805001
500.0	0.042718540261481794

k = 300

Gamma	STD RMSE Val
0.015	0.2296608303888253
0.15	0.01700393506364587
1.5	0.0014465767179988938
15.0	0.0017186920521525878
150.0	0.016913183517925777
1500.0	0.04266554224996233

k = 600

Gamma	STD RMSE Val
0.03	0.08495205582365
0.3	0.021624295611575688
3.0	0.0005997443880974495
30.0	0.0003481205543323593
300.0	0.0730131939156232

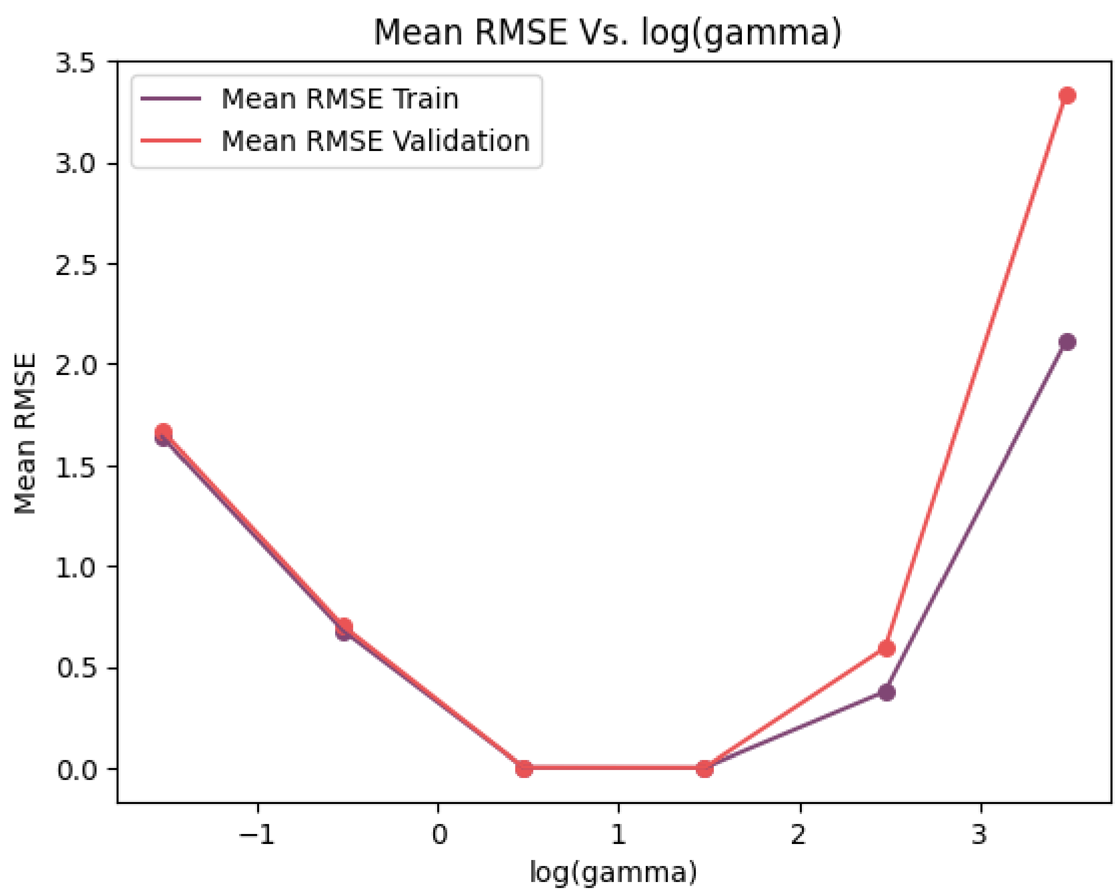
```
| 3000.0          | 0.4820442549739416       |
+-----------------+--------------------------+
```

2] (e) iii] We can evidently see that for k = 600, gamma = 30, we get the minimum mean RMSE of 0.002454457077742055

2] (e)) ii] The optimized hyper-parameters are gamma = 30 amd K = 600

2] (e) iv] Plot training and validation RMSE vs. γ. (For parts (d) and (e), use your best value of M = M $*$ or K = $K*$ for the plot.)

Plotting Mean RMSE Validation Vs. log(Gamma)



2] (e) v] If computational complexity were an issue, what is the smallest value of M or K (and its associated γ) that would give RMSE at least a factor of 10 lower than the trivial system of (b)?

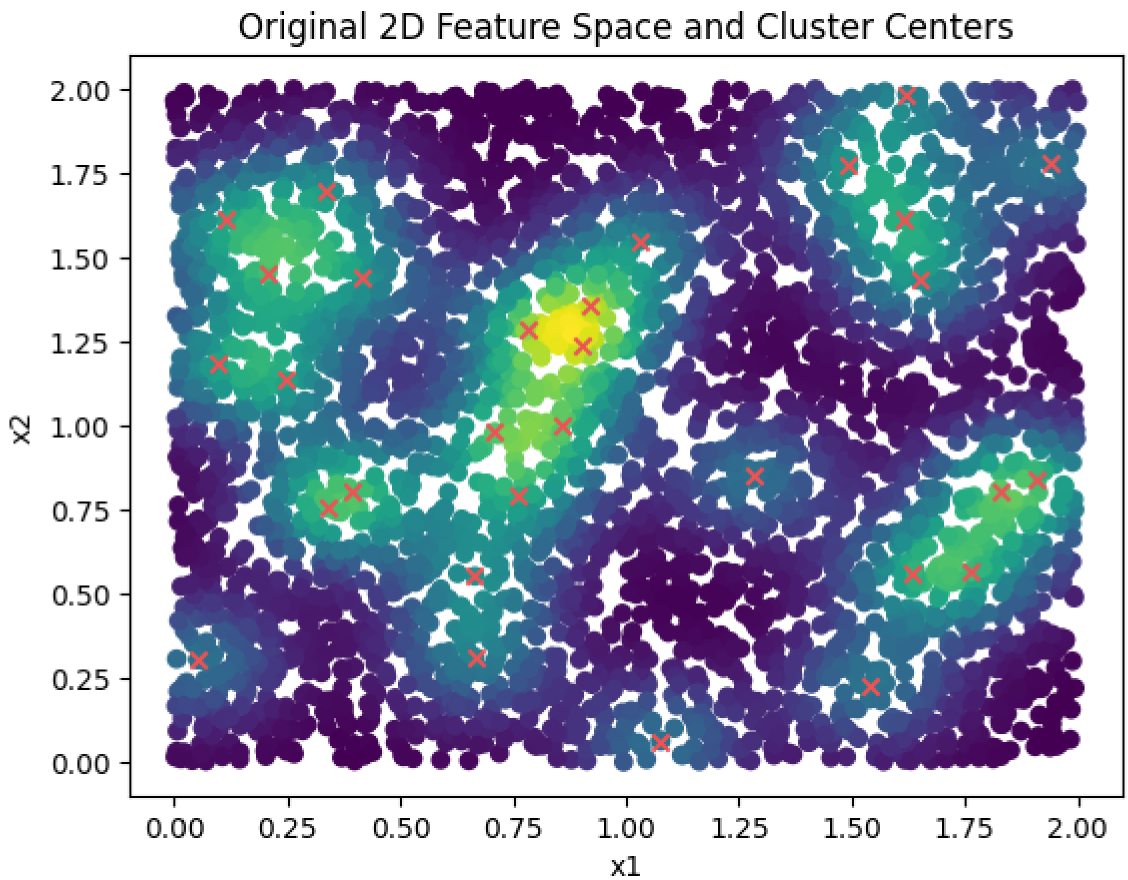
The smallest value of K is 100 and associated gamma = 5.0 with a RMSE = 0.18762609464511065 that would give RMSE at least a factor of 10 lower than the trivial system. This is the RCC Model.

What factor reduction in number of hidden units (dimensionality of the expanded feature space) from the original M=3000 in part (c) does this RCC model represent?

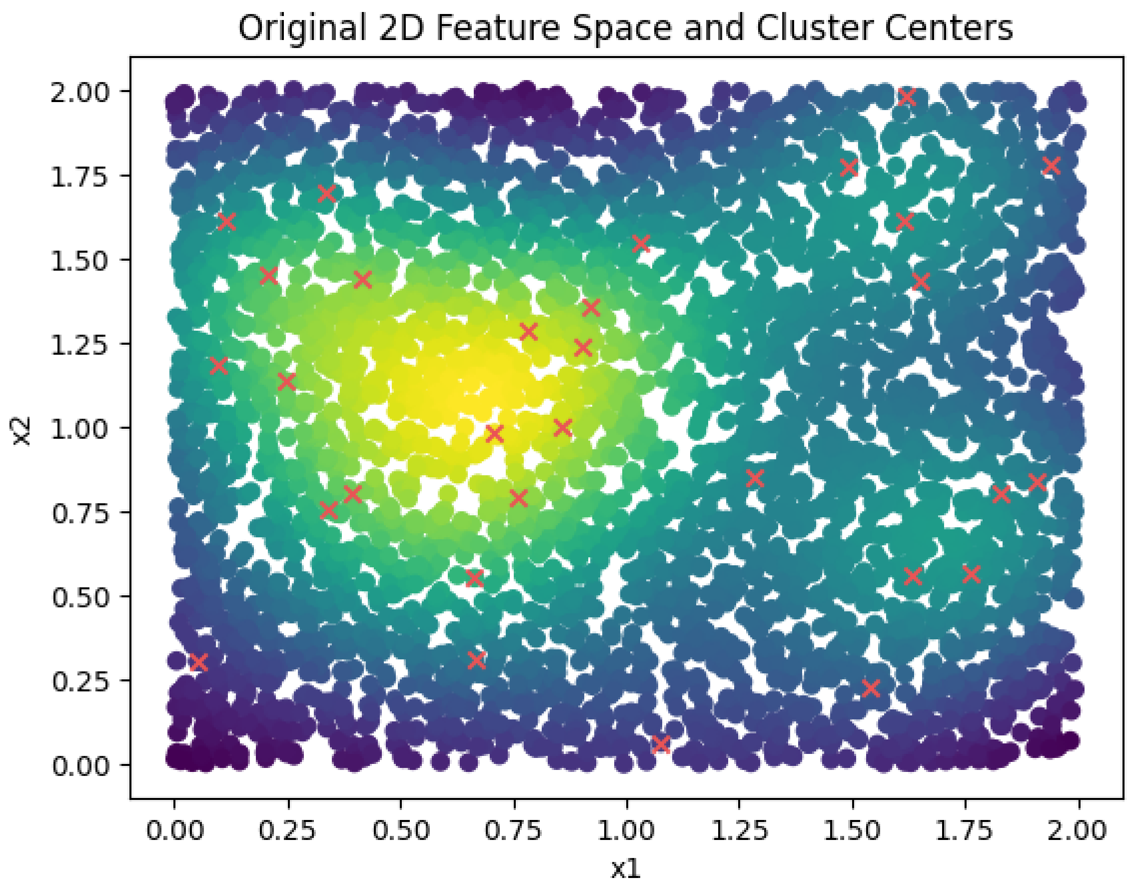
The RCC model represents reduction of hidden units by a factor of 30.

2] (e) vi] For parts (d) and (e), plot in original 2D feature space, the training data points x& and the cluster centers μ' for your best values of hyperparameters. Then repeat the plots for your RCC model, and again for your lowest-complexity-model (M=30 or K=30). (Total of 3 plots for (d))

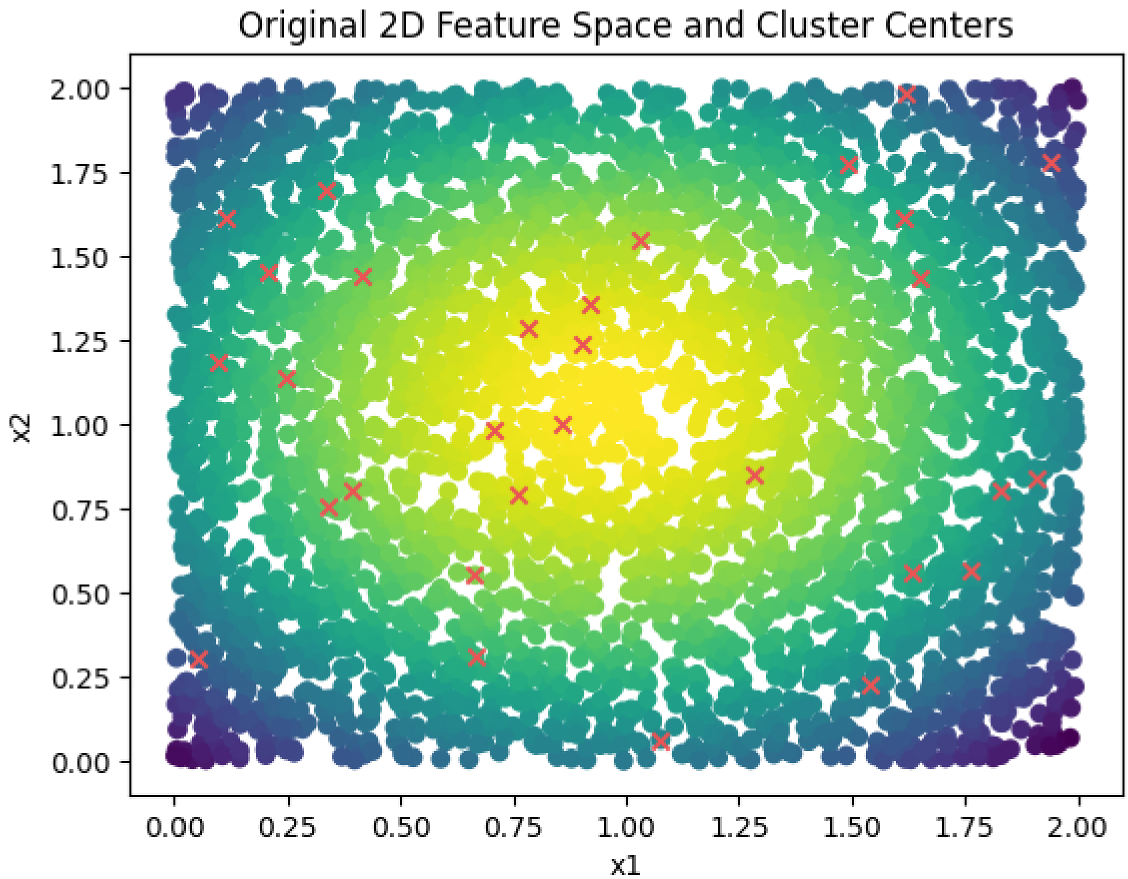
Plot in original 2D feature space, the training data points x& and the cluster centers μ' for your best values of hyperparameters.


Original 2D Feature Space and Cluster Centers
x2
x1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00 0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

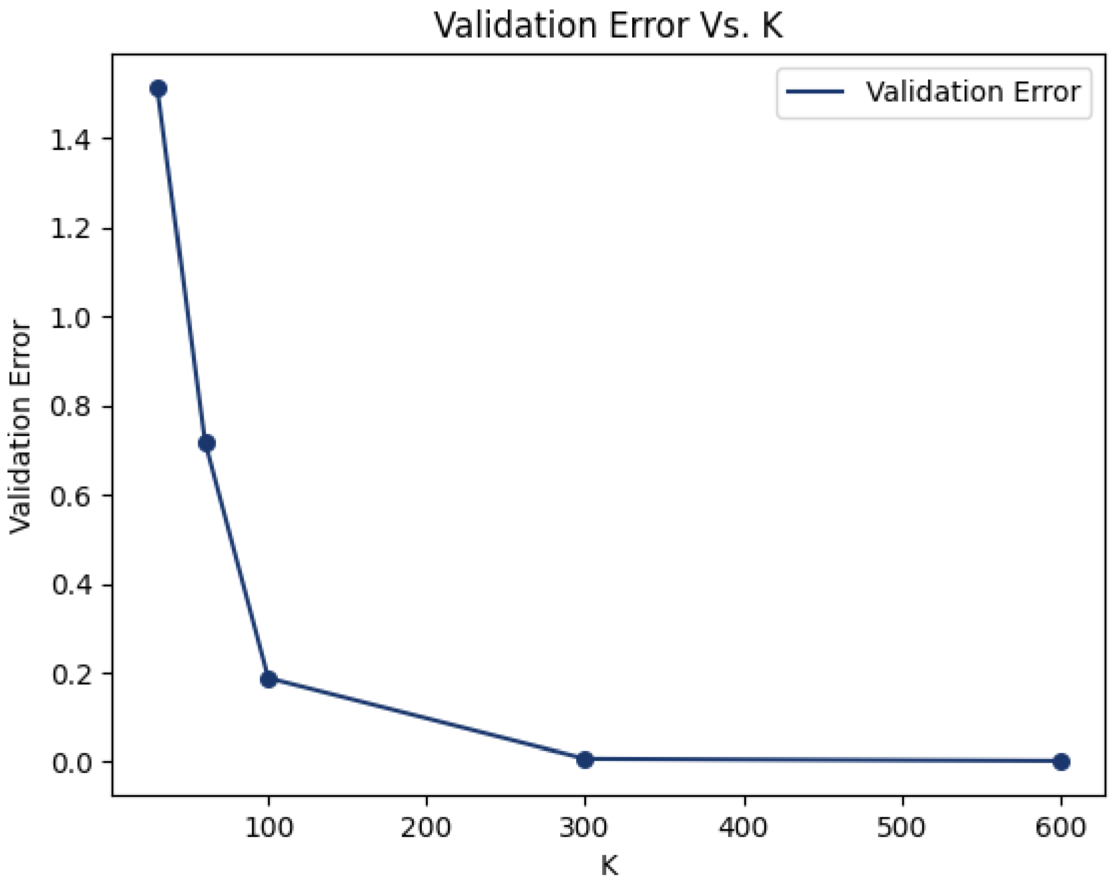
Plot in original 2D feature space, the training data points x& and the cluster centers μ' for the RCC Model. (K = 100, gamma = 5.0)

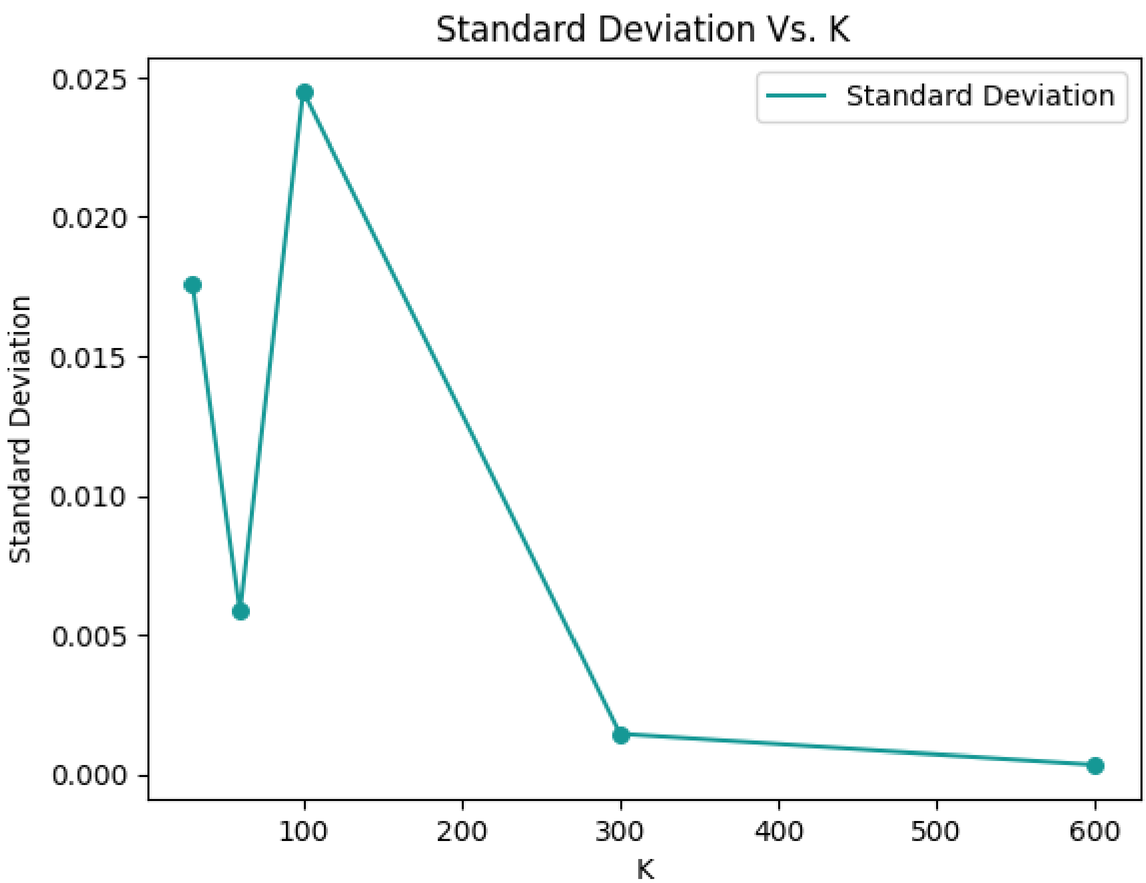


Plot in original 2D feature space, the training data points x& and the cluster centers μ' for the lowest-complexity-model (K=30, gamma = 0.15)



2] (e) vii] Plot the validation error and its standard deviation Vs. the second hyperparameter (M for (d), K for (e)), using the best γ for each value of M or K. (The value of best γ may depend on M or K.)





(f) Give the d.o.f. and number of constraints for the second layer (linear regressor) for each of (c), (d), and (e), for your best model of each; and again for your RCC model for each of (d), (e).

For Cross-Validation

BEST MODELS

c]

D.O.F ==> 3001

Constraints ==> 3000

d]

D.O.F ==> 601

Constraints ==> 3000

e]

D.O.F ==> 601

Constraints ==> 3000

RCC MODELS

d]

D.O.F ==> 101

Constraints ==> 3000

e]

D.O.F ==> 101

Constraints ==> 3000

(f) Give the d.o.f. and number of constraints for the second layer (linear regressor) for each of (c), (d), and (e), for your best model of each; and again for your RCC model for each of (d), (e).

For Full Train Dataset

BEST MODELS

c]

D.O.F ==> 4001

Constraints ==> 4000

d]

D.O.F ==> 601

Constraints ==> 4000

e]

D.O.F ==> 601

Constraints ==> 4000

RCC MODELS

d]

D.O.F ==> 101

Constraints ==> 4000

e]

D.O.F ==> 101

Constraints ==> 4000

(g) Run the best model from each of (c), (d), and (e); and run the RCC model of (d), (e), on your test set. Report the RMSE of each (5 models total).

Best Model (c)

```
RMSE of Best Model (c) for Test Data is: 1.4204961473780364e-07
```

Best Model (d)

```
RMSE of Best Model (d) for Test Data is: 0.0020840503693903565
```

Best Model (e)

```
RMSE of Best Model (e) for Test Data is: 0.001717223606446779
```

RCC Model (d)

```
RMSE of RCC Model (d) for Test Data is: 0.12617802459250077
```

RCC Model (e)

```
RMSE of RCC Model (e) for Test Data is: 0.1603978481225234
```

(h) Compare and comment on your results from (b)-(g). Specifically, observe and try to explain differences in performance for different values of M (or K) and γ during model selection.

(b) The trivial model outputs the sample mean of the y_train values. Hence, it gives the same y_pred for every single data-point. So, its RMSE will be very high and performance is very poor

For (c), using the entire data points as basis function centers resulted in the lowest RMSE value, however, it is computationally expensive.

For (d), randomly selecting m data points as the basis function centers resulted in a higher RMSE value than in (c), but it still outperformed the trivial system.

For (e), using K-means clustering to select k centers as the basis function centers resulted in similar results as (d).

Generally, increasing the number of basis function centers (M or K) leads to a decrease in RMSE values. Additionally, the mean RMSE decreases with an increase in gamma value, indicating better performance of the model. However, a gamma value that is too high can cause the model's performance to deteriorate.