1.0 Data Set

1.1 Description of Data Set

The dataset used consists of 50 major companies daily stock prices downloaded from Yahoo Finance from a period of ***2/8/2013*** to ***2/7/2018***. The closing stock prices of these companies for days in which the stock market was open were used for the analysis. **Table 1** shows the list of first 5 stocks of the companies used and 50th stocks chosen for this study.

Table 1: Stock prices used for analysis.

Time	A	AAL	AAP	AAPL	ABBV	ATVI
2013-02-08	45.08	14.75	78.90	67.85	36.25	13.41
2013-02-11	44.60	14.46	78.39	68.56	35.85	13.57
2013-02-12	44.62	14.27	78.60	66.84	35.42	13.51
2013-02-13	44.75	14.66	78.97	66.72	35.27	13.73
2013-02-14	44.58	13.99	78.84	66.66	36.57	14.00

The main objective is to predict the future stock price for Activision Blizzard Inc. (ATVI) at day *t+1* given the past evolution of 50 stocks observed up to time "***t***". **Table 2** shows the size of the original dataset downloaded. There are 1,259 daily stock prices for the 50 companies that we considered.

Table 2: Size of data set.

ID	Data	Size
1	Number of cases	1,259
2	Number of features	50

1.2 Preprocessing of Data Set

A check was done across the whole data set to see if there are null values present in the dataset. From **Figure 1**, it was found that there were zero missing values. Consequently, there was no need to replace isolated missing values Sj(t) by the mean of two actual values close to time t.

```
In [37]: #Check that no missing values
         bba.isnull().sum(axis=1)

Out[37]: 0       0
         1       0
         2       0
         3       0
         4       0
                ..
         1254    0
         1255    0
         1256    0
         1257    0
         1258    0
         Length: 1259, dtype: int64
```

Figure 1: Missing values.

The average moving mean was then calculated and normalized to give the final series. In order to create the training and test data sets for the MLP predictor, the recent past of the series was estimated and this gave an input, xt, of size 1,254 cases and 250 features for $5 \leq t \leq N-1$. To train the MLP, the output was considered the same as the inputs and subsequently split in ratio 90% and 10% for both the train and tests respectively as shown in **Table 2**.

Table 2: Training and testing data

ID	Data	Size
1	Training Set (PredTrain)	1128 cases with 250 features (90%)
2	Test Set (PredTest)	126 cases with 250 features (10%)

2.0 Application of AutoEncoder to compress the input vectors Xt

2.1 Compute a plausible dimension h for H

It is essential that the number of hidden layers needs to be selected very carefully. This is because selecting a value of h that is large could result in the model performing well on the training set but poorly on the test set. This situation is known as overfitting. Selecting a value of h that is small could also lead to a weak architecture and consequently underfit without capacity to generalize. As a result, it is essential to have a value that is stable for both the training and test set. The PCA analysis was performed on the input data to estimate the value of h that will be used to design the architecture of the MLP.

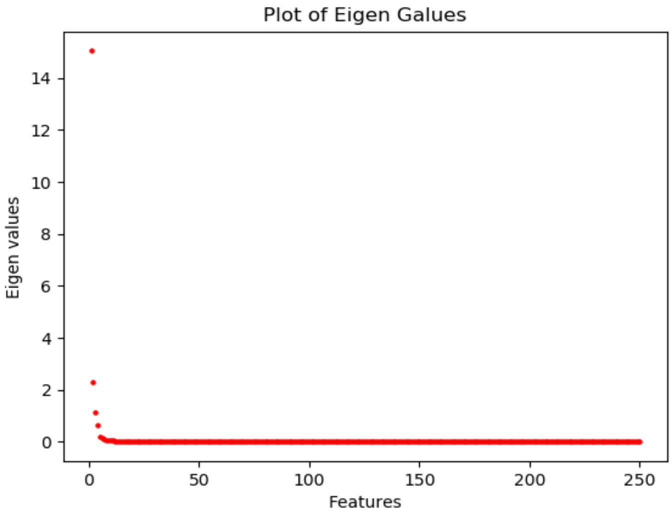


Figure 2: Plot of eigen values.

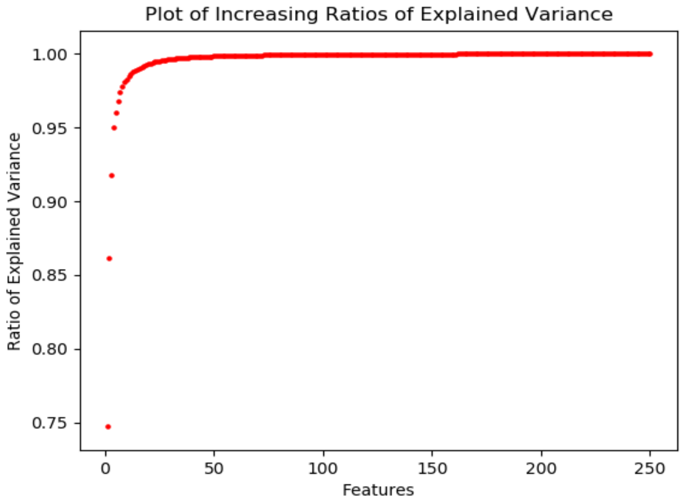


Figure 3: Plot of increasing Rat j.

Figure 2 shows the decreasing trend of the positive eigen values while **Figure 3** represents the cumulative sum of the eigen vectors which gives the proportion of the explained variance. The eigen vectors generally explained certain percentage of the total dispersion of the data. Hence, the explained dispersion Rati tends to 1 when R gets to *p*. Generally, there is a need to find a good truncation value of R such that the ratio is close to 90%. This corresponds to 3 as we estimated from the graph and calculated using the code in the appendix. Thus, the value h=3 was set as the number of neurons in the yet to be developed MLP architecture.

2.2 Description of the MLP Architecture

An MLP architecture defined by three layers was selected. The first is the input layer with number of neurons equals 250 (p=250). This represent the number of features of in our data. The second layer is the hidden layer with dimension equals to the value set as h estimated previously with a RELU activation function and lastly the output layer (dimension=250 which represent the number of prediction). **Figure 4** shows the structure of the MLP as defined by the autoencoder.

The loss function was defined as the mean-squared-error between the output of the MLP and the true value.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.initializers import Constant

# constructing the autoencoder

# determine h through PCA on your own data
# try to find suitable initializers for your own data
h = h90
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=250, bias_initializer=Constant(value=10)))
model.add(Dense(250, activation='relu', bias_initializer=Constant(value=5)))
model.summary()
```

Figure 4: First MLP Architecture.

More so, stochastic gradient descent gradient algorithm was implemented using keras by setting the learning rate at 0.05 and minimizing the mean square error. For the batch learning, a batch size of 100 was chosen and the epoch was set at 300 (**Table 4**). Early stopping technique which is a form of regularization was used to help prevent overfitting and was defined as shown in **Figure 4.**

Table 4: Parameters for first mlp

ID	Data	Value
1	Batch Size	100
2	Epoch	300
3	Number of hidden neurons	3
4	Learning rate	0.05

```
from tensorflow.keras import optimizers, losses

model.compile(optimizer=optimizers.SGD(learning_rate=0.05, decay=1e-7), loss='mean_squared_error')
from tensorflow.keras import callbacks

# the following callback to record losses after each batch
class MyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(x_train,x_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(x_test,x_test,verbose = 0))

MyMonitor = MyHistory()

# Keras built-in early-stopping callback
# https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patience=1000, restore_best_weights=True)
```

```
Monitor = model.fit(x_train, x_train, epochs=300, batch_size=100, callbacks = [MyMonitor, es], validation_data = (x_test, x_te

# After training, access MSE(AutoTrain) and MSE(AutoTest) through MyMonitor.MSEtrain and MyMonitor.MSEtest.
```

Figure 4: SGD, Batching learning and early stopping.

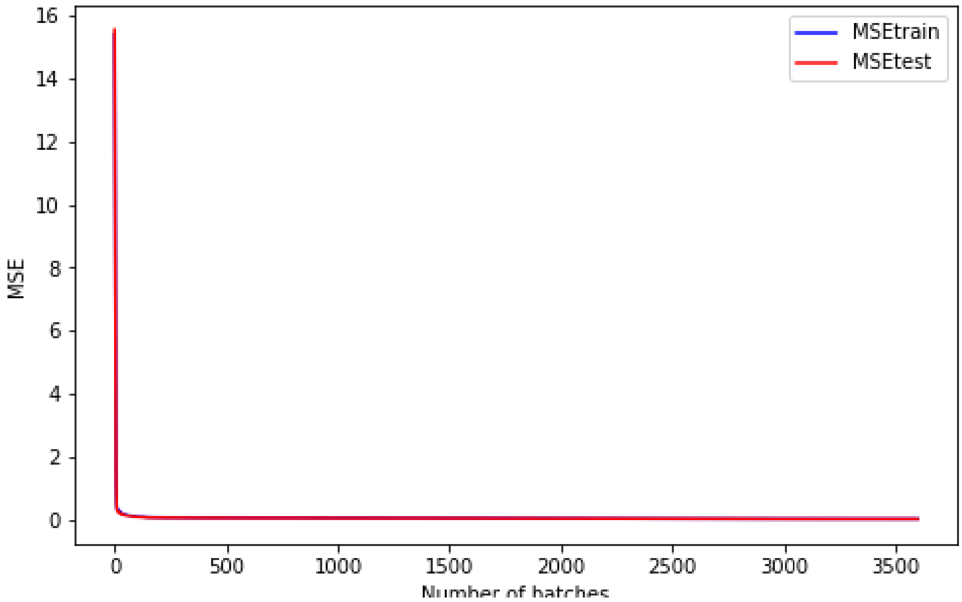


Figure 5: MSE(AutoTrain) and MSE(AutoTest) versus the number of batches.

Figure 5 shows the result for the AutoTrain and AutoTest that gave a very close match. We can observe from the graph that the MSE drops sharply from about 15 to close to zero during the initial training and after that stabilizes and doesn't change much for both.

2.3. Compute the compressed Inputs

To consider the deep learning method, we extracted the states of the hidden neurons from both the ***AutoTrain*** and ***AutoTest.*** These states were considered as inputs to the next autoencoder that will be constructed where output will be the target variable (stocks at ATLP at time *t+1*).

3.0 MLP predictor (deep learning method)

3.1 Description of the MLP Architecture

For the deep learning MLP architecture, we defined three layers like the previous architecture. However, the first input layer contains 3 neurons. This represent the number of neurons in the hidden layers defined in the first architecture. The second layer is the new hidden layer with dimension yet unknown and a RELU activation function, and lastly the output layer (dimension=1 which represent our prediction). The loss function was defined as the mean-squared-error between the output of the MLP and our targets at (t+1).

A stochastic gradient descent gradient was implemented using keras (as done in the first architecture) by setting the learning rate at 0.0001 and minimizing the mean square error. For the batch learning, a batch size of 320 was chosen and the epoch was set at 50 (**Table 5**). Early stopping technique which is a form of regularization was used to help prevent overfitting.

As opposed to using the previous PCA method, the number of neurons in the new hidden layer was selected by chosen the largest value of h such that the total number of weights and hidden layer is less than total the total number of cases. **Equations 1 and 2** show how the k value was estimated.

Table 5: Parameters for second mlp

ID	Data	Value
1	Batch Size	320
2	Epoch	50
3	Number of hidden neurons	varies (10, 30,50, 100,150, 200, 225)
5	Learning rate	0.0001

$$h \times k + k + k + 1 < 1128 \qquad (1)$$

Since h=3, therefore

$$k < 225 \qquad (2)$$

Consequently, we tried different values of *k* such that *k* is less than 225 and then calculate the Mean Relative Errors of Prediction (MREP) on NewTrain for each case of *k* considered. It is important to state that this analysis was done for the same MLP Architecture and Hyperparameters.

Table 6 shows the result that was obtained. It can be seen that the lowest MREP is given by k=100 as the MREP is 0.287. Also, even though the best value of k is 225, it did not perform well when compared to k=100. This value of k (k=100) was subsequently used to build the new architecture by setting a reasonable value for the epoch and batch size.

Table 6: Mean Relative Errors of Prediction MREP for different values of k

k=10	**k=30**	**k=50**	**k=100**	**k=150**	**k=200**	**k=225**
0.444	0.385	0.289	0.287	0.321	1.0	1.0

3.2 Evaluation of Results

Table 7 shows the parameters that was used for building the final model. The batch size was reduced to 20 while the epoch was increased to 500. The number of hidden neurons used was 100. The results are described below.

Table 7: Parameters for final architecture

ID	**Data**	**Value**
1	Batch Size	20
2	Epoch	500
3	Number of hidden neurons	100
5	Learning rate	0.0001

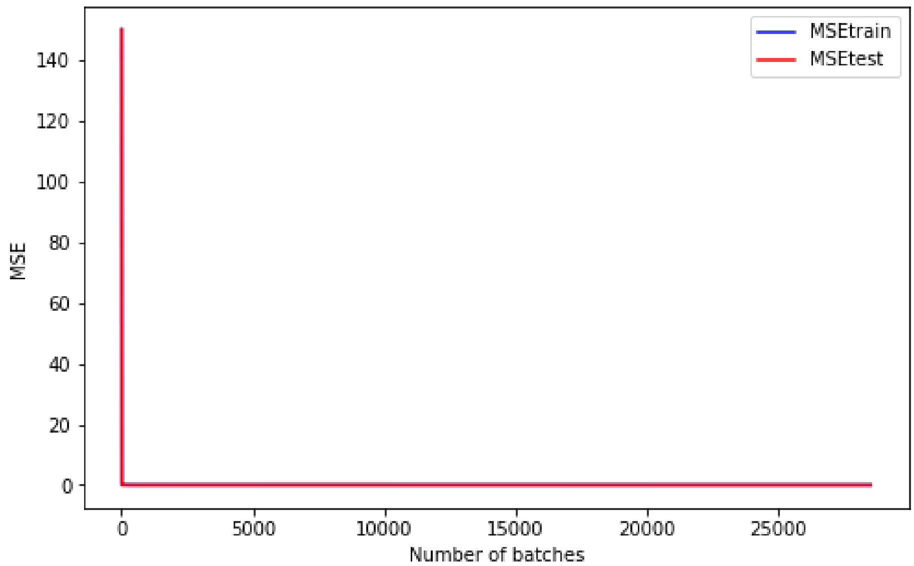


Figure 6: Plot of MSENewTrain and MSENewTest.

Figure 6 shows the plot that compares the MSE for both the train and test set. This result shows a very close match for the two graphs. It is also close to what was observed for the first MLP but this one started a lot higher before dropping. We can observe from the graph that the MSE drops sharply from about 150 to close to zero during the initial training and after that stabilizes and doesn't change much for both.

To analyze the output of our prediction, we plotted the MLP prediction against the target stock price. Because the number of cases is large (i.e. 1,128 cases for the trainset), the first 100 cases was plotted for both the train and test sets to show how closely the prediction match the target.

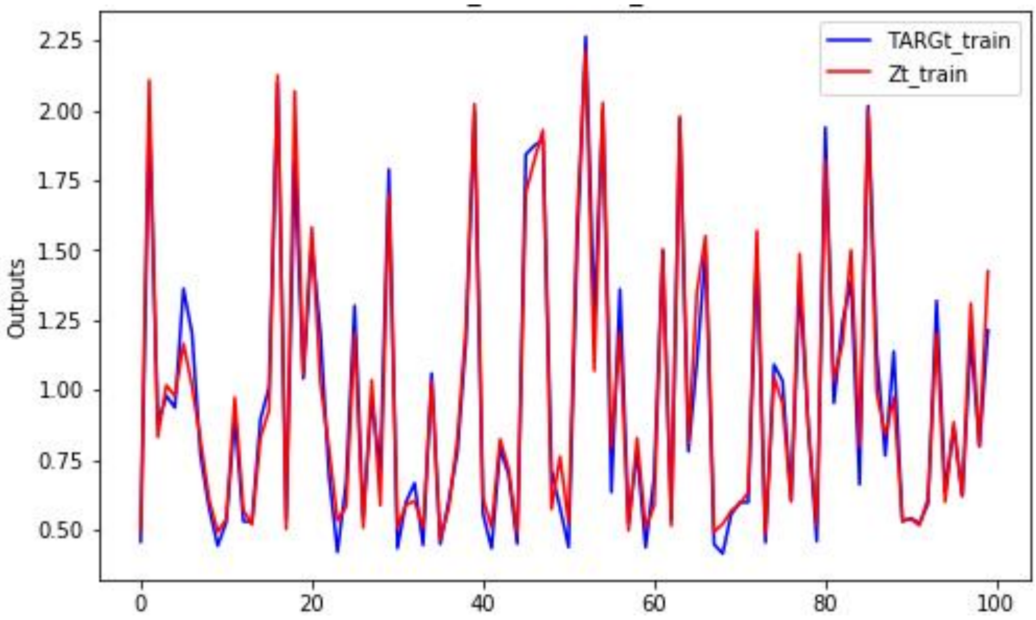


Figure 7: Plot of prediction for MLP and Target (Train Set).

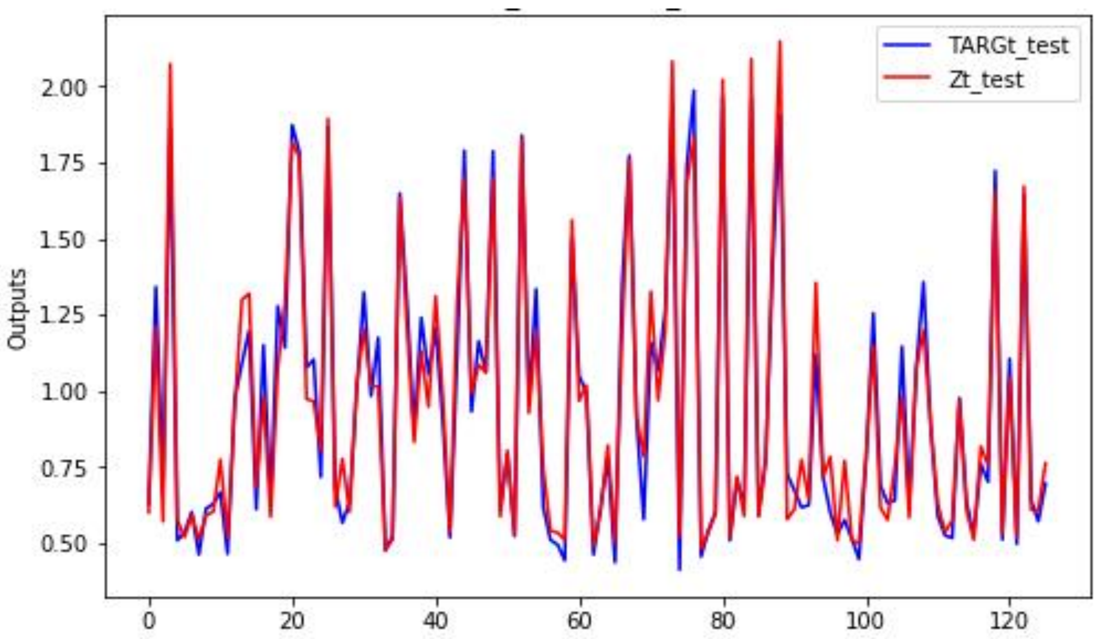


Figure 8: Plot of prediction for MLP and Target (Test Set).

Figures 7 and 8 show that the deep learning method does a very good job in predicting the future stock price. As we can see, there is a very close overlap between the predicted and true output of the train and test set.

More so, the MREP for both the train and test set were calculated for the purpose of comparison. **Table 7** shows the result of the analysis. As we can see, the value of MREP for both the train and test set are very low and almost the same. This shows that the model does not only perform well (in terms of low MREP) but also does a very good job in generalization. The MREP for the training set is lower than that of the test set which is expected as the model was fitted using the training data. The time taken to run the model is about ***11.133 mins.***

Table 7: Results of MREP for both train and test set

ID	Data	Values
1	MREP (Train)	0.079
2	MREP (Test)	0.081

In [3]:
```
import tensorflow as tf
tf.enable_eager_execution()
import numpy as np
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler
from numpy import linalg as LA
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from mpl_toolkits import mplot3d
from sklearn.utils import shuffle
import random
```

Data Set

In [4]:
```
#import dataset
data1 = pd.read_csv("C:/Users/jamiu/Desktop/Azencott - Deep Learning/all_stock
s_5yr.csv")
dc=data1[['date','close','Name']]
bb=dc.pivot(index='date', columns='Name', values='close').iloc[:,0:52].drop([
'APTV','ALLE'], axis=1)
```

In [5]:
```
bb.head()
```

Out[5]:

Name	A	AAL	AAP	AAPL	ABBV	ABC	ABT	ACN	ADBE	ADI	...	ANTM	AON
date													
2013-02-08	45.08	14.75	78.90	67.8542	36.25	46.89	34.41	73.31	39.12	45.70	...	66.28	56.53
2013-02-11	44.60	14.46	78.39	68.5614	35.85	46.76	34.26	73.07	38.64	46.08	...	66.01	56.66
2013-02-12	44.62	14.27	78.60	66.8428	35.42	46.96	34.30	73.37	38.89	46.27	...	66.01	56.70
2013-02-13	44.75	14.66	78.97	66.7156	35.27	46.64	34.46	73.56	38.81	46.26	...	63.00	57.28
2013-02-14	44.58	13.99	78.84	66.6556	36.57	46.77	34.70	73.13	38.61	46.54	...	63.29	57.30

5 rows × 50 columns

Processing of time series

In [6]:
```
bba=bb.iloc[:,:52].reset_index(drop=True)
bba.head()
```

Out[6]:

Name	A	AAL	AAP	AAPL	ABBV	ABC	ABT	ACN	ADBE	ADI	...	ANTM	AON
0	45.08	14.75	78.90	67.8542	36.25	46.89	34.41	73.31	39.12	45.70	...	66.28	56.53
1	44.60	14.46	78.39	68.5614	35.85	46.76	34.26	73.07	38.64	46.08	...	66.01	56.66
2	44.62	14.27	78.60	66.8428	35.42	46.96	34.30	73.37	38.89	46.27	...	66.01	56.70
3	44.75	14.66	78.97	66.7156	35.27	46.64	34.46	73.56	38.81	46.26	...	63.00	57.28
4	44.58	13.99	78.84	66.6556	36.57	46.77	34.70	73.13	38.61	46.54	...	63.29	57.30

5 rows × 50 columns

In [7]:
```
#Check that no missing values
bba.isnull().sum(axis=1)
```

Out[7]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1254    0
1255    0
1256    0
1257    0
1258    0
Length: 1259, dtype: int64
```

In [8]:
```
bb.shape
```

Out[8]:
```
(1259, 50)
```

In [9]:
```
data = bba
```

In [10]:
```
#data_mm.head()
data_mm=data.mean()
print('Normalized Data')
#normalized Yij
df=data/data_mm
df
#bb11= bba/(bba.rolling(window=2,min_periods=1).mean().reset_index(drop=True))
#df=bb11.iloc[:,:]
#df
```

```
Normalized Data
```

Out[10]:

Name	A	AAL	AAP	AAPL	ABBV	ABC	ABT	ACN	ADBE
0	0.916222	0.384182	0.595771	0.622135	0.595586	0.571182	0.801345	0.724987	0.432465
1	0.906467	0.376629	0.591920	0.628619	0.589014	0.569598	0.797851	0.722613	0.427159
2	0.906873	0.371680	0.593506	0.612862	0.581949	0.572034	0.798783	0.725580	0.429923
3	0.909515	0.381838	0.596299	0.611695	0.579485	0.568136	0.802509	0.727459	0.429038
4	0.906060	0.364387	0.595318	0.611145	0.600843	0.569720	0.808098	0.723207	0.426827
...	...	...	...	...	...	...	...	...	...
1254	1.480224	1.403372	0.885652	1.538325	1.911461	1.209482	1.448056	1.586842	2.204113
1255	1.448111	1.357009	0.860281	1.471577	1.892238	1.169649	1.436645	1.551636	2.162768
1256	1.386528	1.296061	0.829549	1.434810	1.799244	1.119462	1.367712	1.501497	2.103403
1257	1.391203	1.333047	0.847218	1.494773	1.827011	1.115077	1.370739	1.529781	2.149834
1258	1.383276	1.338777	0.830077	1.462775	1.866771	1.147723	1.366315	1.534330	2.126287

1259 rows × 50 columns

Create Training and Test sets for MLP

In [11]:
```
def series_to_supervised(data, n_in=1, n_out=1, dropnan=True):

    n_vars = 1 if type(data) is list else data.shape[1]
    df = pd.DataFrame(data)
    cols, names = list(), list()
    # input sequence (t-n, ... t-1)
    for i in range(n_in, 0, -1):
        cols.append(df.shift(i))
        names += [('var%d(t-%d)' % (j+1, i)) for j in range(n_vars)]

# forecast sequence (t, t+1, ... t+n)
    for i in range(0, n_out):
        cols.append(df.shift(-i))
        if i == 0:
            names += [('var%d(t)' % (j+1)) for j in range(n_vars)]
        else:
            names += [('var%d(t+%d)' % (j+1, i)) for j in range(n_vars)]
    # put it all together
    agg = pd.concat(cols, axis=1)
    agg.columns = names

#  Drop rows with NaN values
    if dropnan:
        agg.dropna(inplace=True)
    return agg
```

In [12]:
```
# frame as supervised learning
reframed = series_to_supervised(df, 4, 2)  ### specify how many days to look b
ack ###
# drop columns we don't want to predict
Xt=reframed.iloc[:,:250]
Yt=reframed.iloc[:,-1]
print('Dimension on Xt and TARGt')
Xt.shape, Yt.shape
```

```
Dimension on Xt and TARGt
```

Out[12]:
```
((1254, 250), (1254,))
```

In [13]:
```
x_train, x_test, y_train, y_test = train_test_split(Xt, Yt, test_size=0.1, ran
dom_state=1)
print('Dimension of train and test')
x_train.shape, x_test.shape
```

```
Dimension of train and test
```

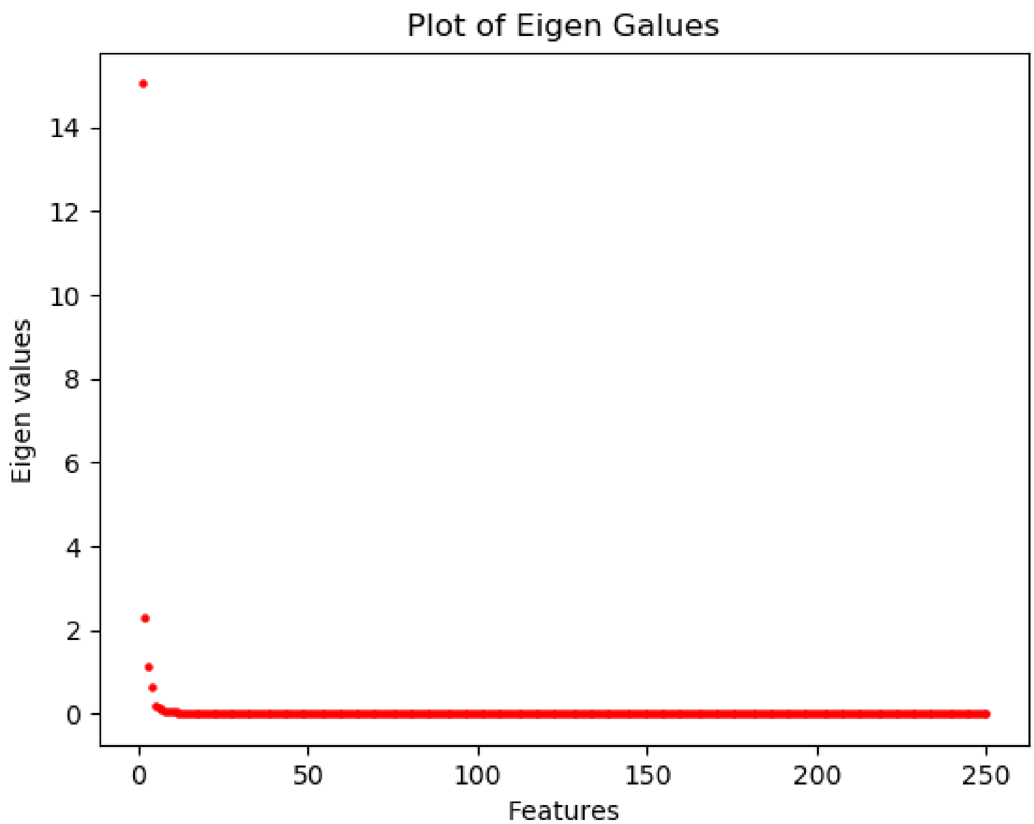
Out[13]:
```
((1128, 250), (126, 250))
```

Auto Encoder to compress input file

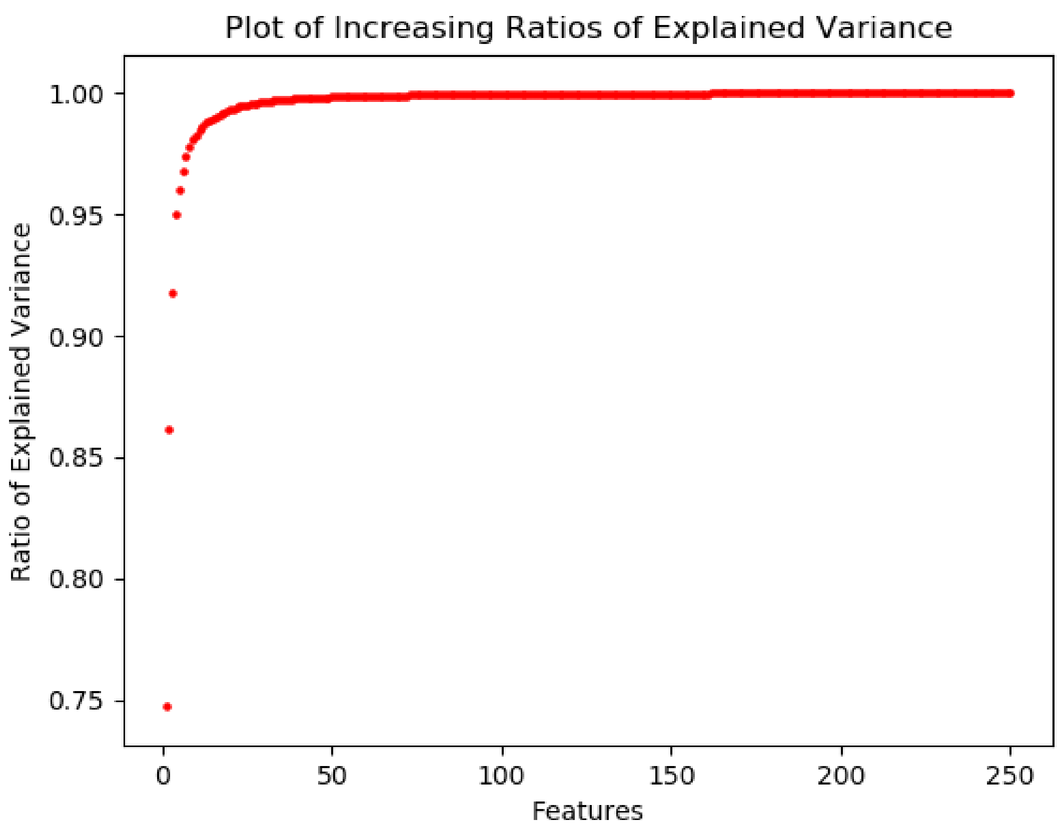
Compute a plausible dimension h for H

In [14]:
```
pca1=PCA(0.9999999999)
pca1.fit(x_train)
h100=pca1.n_components_
print('h100:',h100)
vals1=pca1.explained_variance_ #get eigen values
Lj=vals1.tolist()
j=list(range(1,len(Lj)+1))
%matplotlib notebook
plt.scatter(j,Lj,c='red',s=5)
plt.title('Plot of Eigen Galues')
plt.xlabel('Features')
plt.ylabel('Eigen values')
print('\n')
```

```
h100: 250
```



In [15]:
```
%matplotlib notebook
x=pd.DataFrame(Lj).cumsum()
RATj=x/sum(Lj)
plt.scatter(j,RATj,c='red',s=5)
plt.title('Plot of Increasing Ratios of Explained Variance')
plt.ylabel('Ratio of Explained Variance')
plt.xlabel('Features')
```

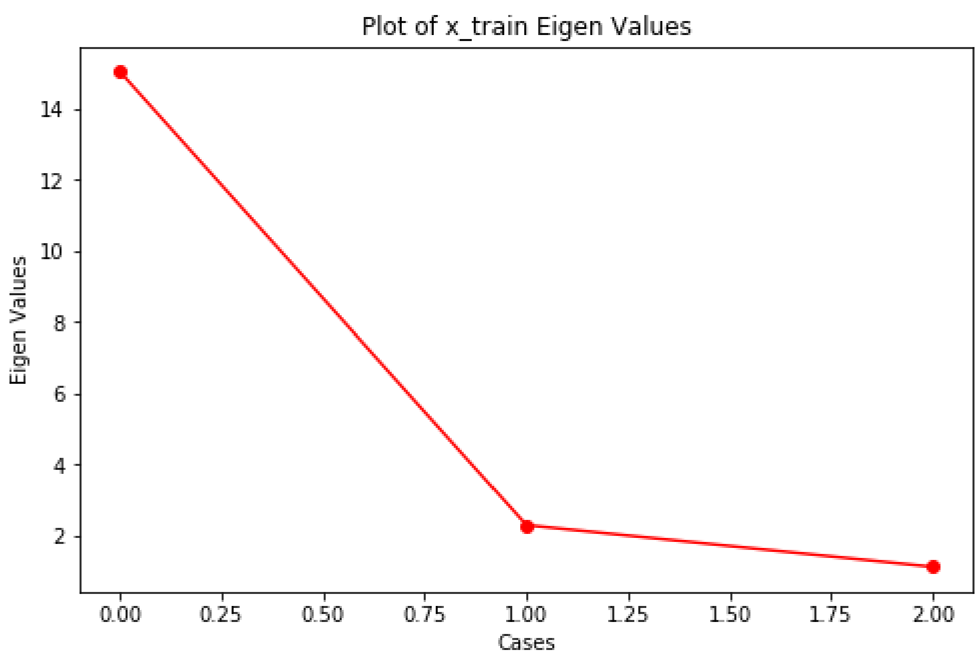


Out[15]: `Text(0.5, 0, 'Features')`

In [16]:
```
pca=PCA(0.90)
pca.fit(x_train)
h90=pca.n_components_
print('Number of components, h90:',h90)
eigval_train=pca.explained_variance_ #get eigen values
round_eig=[round(num, 2) for num in eigval_train]
print('eigen values from x_train:','\n',round_eig)

#plot eigen values
%matplotlib inline
plt.figure(figsize=(8, 5))
plt.figure(1)
plt.plot(eigval_train,  marker='o', label='Eigen Values', color='r')
plt.ylabel('Eigen Values')
plt.xlabel('Cases')
plt.title('Plot of x_train Eigen Values')
```

```
Number of components, h90: 3
eigen values from x_train:
 [15.04, 2.31, 1.13]
```

Out[16]:
```
Text(0.5, 1.0, 'Plot of x_train Eigen Values')
```



AutoEncoder Training

In [17]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.initializers import Constant

# constructing the autoencoder

# determine h through PCA on your own data
# try to find suitable initializers for your own data
h = h90
model = Sequential()
model.add(Dense(h, activation='relu', input_dim=250, bias_initializer=Constant
(value=10)))
model.add(Dense(250, activation='relu', bias_initializer=Constant(value=5)))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 3)                 753
_________________________________________________________________
dense_1 (Dense)              (None, 250)               1000
=================================================================
Total params: 1,753
Trainable params: 1,753
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```
from tensorflow.keras import optimizers, losses

model.compile(optimizer=optimizers.SGD(learning_rate=0.05, decay=1e-7), loss=
'mean_squared_error')
from tensorflow.keras import callbacks

# the following callback to record losses after each batch
class MyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(x_train,x_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(x_test,x_test,verbose = 0))

MyMonitor = MyHistory()

# Keras built-in early-stopping callback
# https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping
es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=1000, restore_best_weights=True)
```

In [19]:
```
Monitor = model.fit(x_train, x_train, epochs=300, batch_size=100, callbacks =
[MyMonitor, es], validation_data = (x_test, x_test), verbose = 2)

# After training, access MSE(AutoTrain) and MSE(AutoTest) through MyMonitor.MS
Etrain and MyMonitor.MSEtest.
```

```
Train on 1128 samples, validate on 126 samples
Epoch 1/300
1128/1128 - 1s - loss: 7.6841 - val_loss: 0.3243
Epoch 2/300
1128/1128 - 0s - loss: 0.2943 - val_loss: 0.2236
Epoch 3/300
1128/1128 - 0s - loss: 0.2316 - val_loss: 0.1735
Epoch 4/300
1128/1128 - 0s - loss: 0.1819 - val_loss: 0.1475
Epoch 5/300
1128/1128 - 0s - loss: 0.1522 - val_loss: 0.1237
Epoch 6/300
1128/1128 - 0s - loss: 0.1348 - val_loss: 0.1149
Epoch 7/300
1128/1128 - 0s - loss: 0.1230 - val_loss: 0.1123
Epoch 8/300
1128/1128 - 0s - loss: 0.1109 - val_loss: 0.0882
Epoch 9/300
1128/1128 - 0s - loss: 0.0956 - val_loss: 0.0809
Epoch 10/300
1128/1128 - 0s - loss: 0.0915 - val_loss: 0.0789
Epoch 11/300
1128/1128 - 0s - loss: 0.0884 - val_loss: 0.0726
Epoch 12/300
1128/1128 - 0s - loss: 0.0794 - val_loss: 0.0647
Epoch 13/300
1128/1128 - 0s - loss: 0.0730 - val_loss: 0.0606
Epoch 14/300
1128/1128 - 0s - loss: 0.0685 - val_loss: 0.0588
Epoch 15/300
1128/1128 - 0s - loss: 0.0674 - val_loss: 0.0582
Epoch 16/300
1128/1128 - 0s - loss: 0.0662 - val_loss: 0.0575
Epoch 17/300
1128/1128 - 0s - loss: 0.0653 - val_loss: 0.0569
Epoch 18/300
1128/1128 - 0s - loss: 0.0647 - val_loss: 0.0555
Epoch 19/300
1128/1128 - 0s - loss: 0.0637 - val_loss: 0.0553
Epoch 20/300
1128/1128 - 0s - loss: 0.0630 - val_loss: 0.0552
Epoch 21/300
1128/1128 - 0s - loss: 0.0624 - val_loss: 0.0549
Epoch 22/300
1128/1128 - 0s - loss: 0.0618 - val_loss: 0.0536
Epoch 23/300
1128/1128 - 0s - loss: 0.0610 - val_loss: 0.0535
Epoch 24/300
1128/1128 - 0s - loss: 0.0604 - val_loss: 0.0524
Epoch 25/300
1128/1128 - 0s - loss: 0.0597 - val_loss: 0.0534
Epoch 26/300
1128/1128 - 0s - loss: 0.0593 - val_loss: 0.0520
Epoch 27/300
1128/1128 - 0s - loss: 0.0587 - val_loss: 0.0513
Epoch 28/300
1128/1128 - 0s - loss: 0.0583 - val_loss: 0.0515
```

```
Epoch 29/300
1128/1128 - 0s - loss: 0.0579 - val_loss: 0.0518
Epoch 30/300
1128/1128 - 0s - loss: 0.0575 - val_loss: 0.0508
Epoch 31/300
1128/1128 - 0s - loss: 0.0570 - val_loss: 0.0500
Epoch 32/300
1128/1128 - 0s - loss: 0.0567 - val_loss: 0.0500
Epoch 33/300
1128/1128 - 0s - loss: 0.0563 - val_loss: 0.0501
Epoch 34/300
1128/1128 - 0s - loss: 0.0561 - val_loss: 0.0505
Epoch 35/300
1128/1128 - 0s - loss: 0.0559 - val_loss: 0.0493
Epoch 36/300
1128/1128 - 0s - loss: 0.0554 - val_loss: 0.0491
Epoch 37/300
1128/1128 - 0s - loss: 0.0552 - val_loss: 0.0489
Epoch 38/300
1128/1128 - 0s - loss: 0.0549 - val_loss: 0.0501
Epoch 39/300
1128/1128 - 0s - loss: 0.0547 - val_loss: 0.0481
Epoch 40/300
1128/1128 - 0s - loss: 0.0544 - val_loss: 0.0481
Epoch 41/300
1128/1128 - 0s - loss: 0.0542 - val_loss: 0.0481
Epoch 42/300
1128/1128 - 0s - loss: 0.0539 - val_loss: 0.0478
Epoch 43/300
1128/1128 - 0s - loss: 0.0536 - val_loss: 0.0479
Epoch 44/300
1128/1128 - 0s - loss: 0.0536 - val_loss: 0.0476
Epoch 45/300
1128/1128 - 0s - loss: 0.0533 - val_loss: 0.0473
Epoch 46/300
1128/1128 - 0s - loss: 0.0531 - val_loss: 0.0477
Epoch 47/300
1128/1128 - 0s - loss: 0.0529 - val_loss: 0.0479
Epoch 48/300
1128/1128 - 0s - loss: 0.0527 - val_loss: 0.0470
Epoch 49/300
1128/1128 - 0s - loss: 0.0526 - val_loss: 0.0476
Epoch 50/300
1128/1128 - 0s - loss: 0.0525 - val_loss: 0.0470
Epoch 51/300
1128/1128 - 0s - loss: 0.0523 - val_loss: 0.0471
Epoch 52/300
1128/1128 - 0s - loss: 0.0522 - val_loss: 0.0467
Epoch 53/300
1128/1128 - 0s - loss: 0.0520 - val_loss: 0.0460
Epoch 54/300
1128/1128 - 0s - loss: 0.0519 - val_loss: 0.0462
Epoch 55/300
1128/1128 - 0s - loss: 0.0518 - val_loss: 0.0460
Epoch 56/300
1128/1128 - 0s - loss: 0.0517 - val_loss: 0.0457
Epoch 57/300
```

```
1128/1128 - 0s - loss: 0.0515 - val_loss: 0.0460
Epoch 58/300
1128/1128 - 0s - loss: 0.0514 - val_loss: 0.0456
Epoch 59/300
1128/1128 - 0s - loss: 0.0512 - val_loss: 0.0455
Epoch 60/300
1128/1128 - 0s - loss: 0.0512 - val_loss: 0.0454
Epoch 61/300
1128/1128 - 0s - loss: 0.0511 - val_loss: 0.0454
Epoch 62/300
1128/1128 - 0s - loss: 0.0510 - val_loss: 0.0457
Epoch 63/300
1128/1128 - 0s - loss: 0.0509 - val_loss: 0.0455
Epoch 64/300
1128/1128 - 0s - loss: 0.0507 - val_loss: 0.0456
Epoch 65/300
1128/1128 - 0s - loss: 0.0507 - val_loss: 0.0453
Epoch 66/300
1128/1128 - 0s - loss: 0.0506 - val_loss: 0.0453
Epoch 67/300
1128/1128 - 0s - loss: 0.0504 - val_loss: 0.0452
Epoch 68/300
1128/1128 - 0s - loss: 0.0504 - val_loss: 0.0449
Epoch 69/300
1128/1128 - 0s - loss: 0.0503 - val_loss: 0.0448
Epoch 70/300
1128/1128 - 0s - loss: 0.0502 - val_loss: 0.0452
Epoch 71/300
1128/1128 - 0s - loss: 0.0502 - val_loss: 0.0450
Epoch 72/300
1128/1128 - 0s - loss: 0.0501 - val_loss: 0.0450
Epoch 73/300
1128/1128 - 0s - loss: 0.0500 - val_loss: 0.0447
Epoch 74/300
1128/1128 - 0s - loss: 0.0499 - val_loss: 0.0446
Epoch 75/300
1128/1128 - 0s - loss: 0.0498 - val_loss: 0.0446
Epoch 76/300
1128/1128 - 0s - loss: 0.0498 - val_loss: 0.0452
Epoch 77/300
1128/1128 - 0s - loss: 0.0498 - val_loss: 0.0443
Epoch 78/300
1128/1128 - 0s - loss: 0.0496 - val_loss: 0.0452
Epoch 79/300
1128/1128 - 0s - loss: 0.0497 - val_loss: 0.0441
Epoch 80/300
1128/1128 - 0s - loss: 0.0495 - val_loss: 0.0439
Epoch 81/300
1128/1128 - 0s - loss: 0.0494 - val_loss: 0.0443
Epoch 82/300
1128/1128 - 0s - loss: 0.0493 - val_loss: 0.0442
Epoch 83/300
1128/1128 - 0s - loss: 0.0493 - val_loss: 0.0440
Epoch 84/300
1128/1128 - 0s - loss: 0.0492 - val_loss: 0.0439
Epoch 85/300
1128/1128 - 0s - loss: 0.0492 - val_loss: 0.0437
```

```
Epoch 86/300
1128/1128 - 0s - loss: 0.0490 - val_loss: 0.0437
Epoch 87/300
1128/1128 - 0s - loss: 0.0489 - val_loss: 0.0436
Epoch 88/300
1128/1128 - 0s - loss: 0.0488 - val_loss: 0.0438
Epoch 89/300
1128/1128 - 0s - loss: 0.0487 - val_loss: 0.0439
Epoch 90/300
1128/1128 - 0s - loss: 0.0487 - val_loss: 0.0435
Epoch 91/300
1128/1128 - 0s - loss: 0.0486 - val_loss: 0.0439
Epoch 92/300
1128/1128 - 0s - loss: 0.0486 - val_loss: 0.0433
Epoch 93/300
1128/1128 - 0s - loss: 0.0484 - val_loss: 0.0430
Epoch 94/300
1128/1128 - 0s - loss: 0.0484 - val_loss: 0.0430
Epoch 95/300
1128/1128 - 0s - loss: 0.0482 - val_loss: 0.0431
Epoch 96/300
1128/1128 - 0s - loss: 0.0482 - val_loss: 0.0429
Epoch 97/300
1128/1128 - 0s - loss: 0.0481 - val_loss: 0.0428
Epoch 98/300
1128/1128 - 0s - loss: 0.0480 - val_loss: 0.0429
Epoch 99/300
1128/1128 - 0s - loss: 0.0479 - val_loss: 0.0431
Epoch 100/300
1128/1128 - 0s - loss: 0.0479 - val_loss: 0.0426
Epoch 101/300
1128/1128 - 0s - loss: 0.0477 - val_loss: 0.0424
Epoch 102/300
1128/1128 - 0s - loss: 0.0476 - val_loss: 0.0424
Epoch 103/300
1128/1128 - 0s - loss: 0.0475 - val_loss: 0.0422
Epoch 104/300
1128/1128 - 0s - loss: 0.0474 - val_loss: 0.0428
Epoch 105/300
1128/1128 - 0s - loss: 0.0473 - val_loss: 0.0420
Epoch 106/300
1128/1128 - 0s - loss: 0.0472 - val_loss: 0.0421
Epoch 107/300
1128/1128 - 0s - loss: 0.0471 - val_loss: 0.0418
Epoch 108/300
1128/1128 - 0s - loss: 0.0470 - val_loss: 0.0422
Epoch 109/300
1128/1128 - 0s - loss: 0.0469 - val_loss: 0.0417
Epoch 110/300
1128/1128 - 0s - loss: 0.0468 - val_loss: 0.0420
Epoch 111/300
1128/1128 - 0s - loss: 0.0467 - val_loss: 0.0423
Epoch 112/300
1128/1128 - 0s - loss: 0.0466 - val_loss: 0.0415
Epoch 113/300
1128/1128 - 0s - loss: 0.0464 - val_loss: 0.0414
Epoch 114/300
```

```
1128/1128 - 0s - loss: 0.0463 - val_loss: 0.0420
Epoch 115/300
1128/1128 - 0s - loss: 0.0463 - val_loss: 0.0414
Epoch 116/300
1128/1128 - 0s - loss: 0.0460 - val_loss: 0.0414
Epoch 117/300
1128/1128 - 0s - loss: 0.0459 - val_loss: 0.0411
Epoch 118/300
1128/1128 - 0s - loss: 0.0457 - val_loss: 0.0411
Epoch 119/300
1128/1128 - 0s - loss: 0.0456 - val_loss: 0.0407
Epoch 120/300
1128/1128 - 0s - loss: 0.0455 - val_loss: 0.0408
Epoch 121/300
1128/1128 - 0s - loss: 0.0453 - val_loss: 0.0406
Epoch 122/300
1128/1128 - 0s - loss: 0.0452 - val_loss: 0.0408
Epoch 123/300
1128/1128 - 0s - loss: 0.0451 - val_loss: 0.0400
Epoch 124/300
1128/1128 - 0s - loss: 0.0449 - val_loss: 0.0400
Epoch 125/300
1128/1128 - 0s - loss: 0.0447 - val_loss: 0.0400
Epoch 126/300
1128/1128 - 0s - loss: 0.0445 - val_loss: 0.0398
Epoch 127/300
1128/1128 - 0s - loss: 0.0444 - val_loss: 0.0394
Epoch 128/300
1128/1128 - 0s - loss: 0.0442 - val_loss: 0.0397
Epoch 129/300
1128/1128 - 0s - loss: 0.0441 - val_loss: 0.0399
Epoch 130/300
1128/1128 - 0s - loss: 0.0439 - val_loss: 0.0393
Epoch 131/300
1128/1128 - 0s - loss: 0.0437 - val_loss: 0.0392
Epoch 132/300
1128/1128 - 0s - loss: 0.0435 - val_loss: 0.0390
Epoch 133/300
1128/1128 - 0s - loss: 0.0433 - val_loss: 0.0387
Epoch 134/300
1128/1128 - 0s - loss: 0.0431 - val_loss: 0.0388
Epoch 135/300
1128/1128 - 0s - loss: 0.0429 - val_loss: 0.0386
Epoch 136/300
1128/1128 - 0s - loss: 0.0427 - val_loss: 0.0387
Epoch 137/300
1128/1128 - 0s - loss: 0.0426 - val_loss: 0.0383
Epoch 138/300
1128/1128 - 0s - loss: 0.0423 - val_loss: 0.0385
Epoch 139/300
1128/1128 - 0s - loss: 0.0421 - val_loss: 0.0377
Epoch 140/300
1128/1128 - 0s - loss: 0.0418 - val_loss: 0.0377
Epoch 141/300
1128/1128 - 0s - loss: 0.0416 - val_loss: 0.0374
Epoch 142/300
1128/1128 - 0s - loss: 0.0414 - val_loss: 0.0371
```

```
Epoch 143/300
1128/1128 - 0s - loss: 0.0412 - val_loss: 0.0370
Epoch 144/300
1128/1128 - 0s - loss: 0.0409 - val_loss: 0.0368
Epoch 145/300
1128/1128 - 0s - loss: 0.0408 - val_loss: 0.0371
Epoch 146/300
1128/1128 - 0s - loss: 0.0405 - val_loss: 0.0363
Epoch 147/300
1128/1128 - 0s - loss: 0.0402 - val_loss: 0.0361
Epoch 148/300
1128/1128 - 0s - loss: 0.0400 - val_loss: 0.0358
Epoch 149/300
1128/1128 - 0s - loss: 0.0397 - val_loss: 0.0357
Epoch 150/300
1128/1128 - 0s - loss: 0.0395 - val_loss: 0.0354
Epoch 151/300
1128/1128 - 0s - loss: 0.0392 - val_loss: 0.0351
Epoch 152/300
1128/1128 - 0s - loss: 0.0389 - val_loss: 0.0351
Epoch 153/300
1128/1128 - 0s - loss: 0.0387 - val_loss: 0.0348
Epoch 154/300
1128/1128 - 0s - loss: 0.0384 - val_loss: 0.0346
Epoch 155/300
1128/1128 - 0s - loss: 0.0381 - val_loss: 0.0345
Epoch 156/300
1128/1128 - 0s - loss: 0.0378 - val_loss: 0.0341
Epoch 157/300
1128/1128 - 0s - loss: 0.0376 - val_loss: 0.0338
Epoch 158/300
1128/1128 - 0s - loss: 0.0373 - val_loss: 0.0336
Epoch 159/300
1128/1128 - 0s - loss: 0.0370 - val_loss: 0.0333
Epoch 160/300
1128/1128 - 0s - loss: 0.0367 - val_loss: 0.0332
Epoch 161/300
1128/1128 - 0s - loss: 0.0364 - val_loss: 0.0327
Epoch 162/300
1128/1128 - 0s - loss: 0.0362 - val_loss: 0.0327
Epoch 163/300
1128/1128 - 0s - loss: 0.0358 - val_loss: 0.0325
Epoch 164/300
1128/1128 - 0s - loss: 0.0355 - val_loss: 0.0321
Epoch 165/300
1128/1128 - 0s - loss: 0.0352 - val_loss: 0.0318
Epoch 166/300
1128/1128 - 0s - loss: 0.0350 - val_loss: 0.0315
Epoch 167/300
1128/1128 - 0s - loss: 0.0346 - val_loss: 0.0316
Epoch 168/300
1128/1128 - 0s - loss: 0.0343 - val_loss: 0.0314
Epoch 169/300
1128/1128 - 0s - loss: 0.0340 - val_loss: 0.0310
Epoch 170/300
1128/1128 - 0s - loss: 0.0337 - val_loss: 0.0306
Epoch 171/300
```

```
1128/1128 - 0s - loss: 0.0334 - val_loss: 0.0310
Epoch 172/300
1128/1128 - 0s - loss: 0.0331 - val_loss: 0.0302
Epoch 173/300
1128/1128 - 0s - loss: 0.0328 - val_loss: 0.0298
Epoch 174/300
1128/1128 - 0s - loss: 0.0325 - val_loss: 0.0296
Epoch 175/300
1128/1128 - 0s - loss: 0.0322 - val_loss: 0.0292
Epoch 176/300
1128/1128 - 0s - loss: 0.0320 - val_loss: 0.0292
Epoch 177/300
1128/1128 - 0s - loss: 0.0316 - val_loss: 0.0290
Epoch 178/300
1128/1128 - 0s - loss: 0.0313 - val_loss: 0.0289
Epoch 179/300
1128/1128 - 0s - loss: 0.0310 - val_loss: 0.0284
Epoch 180/300
1128/1128 - 0s - loss: 0.0308 - val_loss: 0.0280
Epoch 181/300
1128/1128 - 0s - loss: 0.0305 - val_loss: 0.0277
Epoch 182/300
1128/1128 - 0s - loss: 0.0302 - val_loss: 0.0278
Epoch 183/300
1128/1128 - 0s - loss: 0.0299 - val_loss: 0.0275
Epoch 184/300
1128/1128 - 0s - loss: 0.0296 - val_loss: 0.0271
Epoch 185/300
1128/1128 - 0s - loss: 0.0293 - val_loss: 0.0268
Epoch 186/300
1128/1128 - 0s - loss: 0.0290 - val_loss: 0.0267
Epoch 187/300
1128/1128 - 0s - loss: 0.0287 - val_loss: 0.0264
Epoch 188/300
1128/1128 - 0s - loss: 0.0285 - val_loss: 0.0262
Epoch 189/300
1128/1128 - 0s - loss: 0.0282 - val_loss: 0.0261
Epoch 190/300
1128/1128 - 0s - loss: 0.0279 - val_loss: 0.0257
Epoch 191/300
1128/1128 - 0s - loss: 0.0276 - val_loss: 0.0256
Epoch 192/300
1128/1128 - 0s - loss: 0.0274 - val_loss: 0.0253
Epoch 193/300
1128/1128 - 0s - loss: 0.0271 - val_loss: 0.0250
Epoch 194/300
1128/1128 - 0s - loss: 0.0268 - val_loss: 0.0249
Epoch 195/300
1128/1128 - 0s - loss: 0.0266 - val_loss: 0.0247
Epoch 196/300
1128/1128 - 0s - loss: 0.0264 - val_loss: 0.0245
Epoch 197/300
1128/1128 - 0s - loss: 0.0261 - val_loss: 0.0243
Epoch 198/300
1128/1128 - 0s - loss: 0.0259 - val_loss: 0.0245
Epoch 199/300
1128/1128 - 0s - loss: 0.0257 - val_loss: 0.0237
```

```
Epoch 200/300
1128/1128 - 0s - loss: 0.0254 - val_loss: 0.0237
Epoch 201/300
1128/1128 - 0s - loss: 0.0252 - val_loss: 0.0235
Epoch 202/300
1128/1128 - 1s - loss: 0.0250 - val_loss: 0.0233
Epoch 203/300
1128/1128 - 1s - loss: 0.0247 - val_loss: 0.0232
Epoch 204/300
1128/1128 - 0s - loss: 0.0245 - val_loss: 0.0230
Epoch 205/300
1128/1128 - 0s - loss: 0.0243 - val_loss: 0.0228
Epoch 206/300
1128/1128 - 0s - loss: 0.0241 - val_loss: 0.0224
Epoch 207/300
1128/1128 - 0s - loss: 0.0239 - val_loss: 0.0224
Epoch 208/300
1128/1128 - 0s - loss: 0.0237 - val_loss: 0.0222
Epoch 209/300
1128/1128 - 0s - loss: 0.0235 - val_loss: 0.0220
Epoch 210/300
1128/1128 - 0s - loss: 0.0233 - val_loss: 0.0219
Epoch 211/300
1128/1128 - 0s - loss: 0.0231 - val_loss: 0.0217
Epoch 212/300
1128/1128 - 0s - loss: 0.0229 - val_loss: 0.0216
Epoch 213/300
1128/1128 - 0s - loss: 0.0227 - val_loss: 0.0213
Epoch 214/300
1128/1128 - 0s - loss: 0.0226 - val_loss: 0.0213
Epoch 215/300
1128/1128 - 0s - loss: 0.0224 - val_loss: 0.0212
Epoch 216/300
1128/1128 - 0s - loss: 0.0222 - val_loss: 0.0212
Epoch 217/300
1128/1128 - 0s - loss: 0.0221 - val_loss: 0.0208
Epoch 218/300
1128/1128 - 0s - loss: 0.0219 - val_loss: 0.0206
Epoch 219/300
1128/1128 - 0s - loss: 0.0217 - val_loss: 0.0205
Epoch 220/300
1128/1128 - 0s - loss: 0.0216 - val_loss: 0.0205
Epoch 221/300
1128/1128 - 0s - loss: 0.0214 - val_loss: 0.0204
Epoch 222/300
1128/1128 - 0s - loss: 0.0213 - val_loss: 0.0203
Epoch 223/300
1128/1128 - 0s - loss: 0.0211 - val_loss: 0.0202
Epoch 224/300
1128/1128 - 0s - loss: 0.0211 - val_loss: 0.0201
Epoch 225/300
1128/1128 - 0s - loss: 0.0209 - val_loss: 0.0198
Epoch 226/300
1128/1128 - 0s - loss: 0.0207 - val_loss: 0.0197
Epoch 227/300
1128/1128 - 0s - loss: 0.0206 - val_loss: 0.0198
Epoch 228/300
```

```
1128/1128 - 0s - loss: 0.0205 - val_loss: 0.0195
Epoch 229/300
1128/1128 - 0s - loss: 0.0204 - val_loss: 0.0194
Epoch 230/300
1128/1128 - 0s - loss: 0.0202 - val_loss: 0.0193
Epoch 231/300
1128/1128 - 0s - loss: 0.0201 - val_loss: 0.0193
Epoch 232/300
1128/1128 - 0s - loss: 0.0200 - val_loss: 0.0197
Epoch 233/300
1128/1128 - 0s - loss: 0.0199 - val_loss: 0.0190
Epoch 234/300
1128/1128 - 0s - loss: 0.0198 - val_loss: 0.0190
Epoch 235/300
1128/1128 - 0s - loss: 0.0197 - val_loss: 0.0189
Epoch 236/300
1128/1128 - 0s - loss: 0.0196 - val_loss: 0.0188
Epoch 237/300
1128/1128 - 0s - loss: 0.0195 - val_loss: 0.0187
Epoch 238/300
1128/1128 - 0s - loss: 0.0194 - val_loss: 0.0187
Epoch 239/300
1128/1128 - 0s - loss: 0.0193 - val_loss: 0.0185
Epoch 240/300
1128/1128 - 0s - loss: 0.0192 - val_loss: 0.0185
Epoch 241/300
1128/1128 - 0s - loss: 0.0191 - val_loss: 0.0183
Epoch 242/300
1128/1128 - 0s - loss: 0.0190 - val_loss: 0.0184
Epoch 243/300
1128/1128 - 0s - loss: 0.0190 - val_loss: 0.0182
Epoch 244/300
1128/1128 - 0s - loss: 0.0189 - val_loss: 0.0184
Epoch 245/300
1128/1128 - 0s - loss: 0.0188 - val_loss: 0.0181
Epoch 246/300
1128/1128 - 0s - loss: 0.0187 - val_loss: 0.0180
Epoch 247/300
1128/1128 - 0s - loss: 0.0187 - val_loss: 0.0180
Epoch 248/300
1128/1128 - 0s - loss: 0.0186 - val_loss: 0.0180
Epoch 249/300
1128/1128 - 0s - loss: 0.0186 - val_loss: 0.0179
Epoch 250/300
1128/1128 - 0s - loss: 0.0185 - val_loss: 0.0180
Epoch 251/300
1128/1128 - 0s - loss: 0.0184 - val_loss: 0.0180
Epoch 252/300
1128/1128 - 0s - loss: 0.0183 - val_loss: 0.0177
Epoch 253/300
1128/1128 - 0s - loss: 0.0183 - val_loss: 0.0176
Epoch 254/300
1128/1128 - 0s - loss: 0.0182 - val_loss: 0.0176
Epoch 255/300
1128/1128 - 0s - loss: 0.0181 - val_loss: 0.0178
Epoch 256/300
1128/1128 - 0s - loss: 0.0181 - val_loss: 0.0175
```

```
Epoch 257/300
1128/1128 - 0s - loss: 0.0180 - val_loss: 0.0175
Epoch 258/300
1128/1128 - 0s - loss: 0.0179 - val_loss: 0.0174
Epoch 259/300
1128/1128 - 0s - loss: 0.0179 - val_loss: 0.0175
Epoch 260/300
1128/1128 - 0s - loss: 0.0179 - val_loss: 0.0174
Epoch 261/300
1128/1128 - 0s - loss: 0.0179 - val_loss: 0.0173
Epoch 262/300
1128/1128 - 0s - loss: 0.0178 - val_loss: 0.0172
Epoch 263/300
1128/1128 - 0s - loss: 0.0177 - val_loss: 0.0175
Epoch 264/300
1128/1128 - 0s - loss: 0.0177 - val_loss: 0.0171
Epoch 265/300
1128/1128 - 0s - loss: 0.0176 - val_loss: 0.0171
Epoch 266/300
1128/1128 - 0s - loss: 0.0176 - val_loss: 0.0171
Epoch 267/300
1128/1128 - 0s - loss: 0.0175 - val_loss: 0.0171
Epoch 268/300
1128/1128 - 0s - loss: 0.0175 - val_loss: 0.0172
Epoch 269/300
1128/1128 - 0s - loss: 0.0175 - val_loss: 0.0171
Epoch 270/300
1128/1128 - 0s - loss: 0.0174 - val_loss: 0.0170
Epoch 271/300
1128/1128 - 0s - loss: 0.0174 - val_loss: 0.0171
Epoch 272/300
1128/1128 - 0s - loss: 0.0173 - val_loss: 0.0169
Epoch 273/300
1128/1128 - 0s - loss: 0.0173 - val_loss: 0.0171
Epoch 274/300
1128/1128 - 0s - loss: 0.0173 - val_loss: 0.0169
Epoch 275/300
1128/1128 - 0s - loss: 0.0172 - val_loss: 0.0168
Epoch 276/300
1128/1128 - 0s - loss: 0.0172 - val_loss: 0.0167
Epoch 277/300
1128/1128 - 0s - loss: 0.0172 - val_loss: 0.0170
Epoch 278/300
1128/1128 - 0s - loss: 0.0171 - val_loss: 0.0167
Epoch 279/300
1128/1128 - 0s - loss: 0.0172 - val_loss: 0.0167
Epoch 280/300
1128/1128 - 0s - loss: 0.0171 - val_loss: 0.0166
Epoch 281/300
1128/1128 - 0s - loss: 0.0171 - val_loss: 0.0166
Epoch 282/300
1128/1128 - 0s - loss: 0.0170 - val_loss: 0.0167
Epoch 283/300
1128/1128 - 0s - loss: 0.0170 - val_loss: 0.0167
Epoch 284/300
1128/1128 - 0s - loss: 0.0170 - val_loss: 0.0165
Epoch 285/300
```

```
1128/1128 - 0s - loss: 0.0170 - val_loss: 0.0165
Epoch 286/300
1128/1128 - 0s - loss: 0.0169 - val_loss: 0.0173
Epoch 287/300
1128/1128 - 0s - loss: 0.0170 - val_loss: 0.0166
Epoch 288/300
1128/1128 - 0s - loss: 0.0169 - val_loss: 0.0165
Epoch 289/300
1128/1128 - 0s - loss: 0.0169 - val_loss: 0.0165
Epoch 290/300
1128/1128 - 0s - loss: 0.0168 - val_loss: 0.0164
Epoch 291/300
1128/1128 - 0s - loss: 0.0168 - val_loss: 0.0164
Epoch 292/300
1128/1128 - 0s - loss: 0.0168 - val_loss: 0.0166
Epoch 293/300
1128/1128 - 0s - loss: 0.0168 - val_loss: 0.0165
Epoch 294/300
1128/1128 - 0s - loss: 0.0168 - val_loss: 0.0163
Epoch 295/300
1128/1128 - 0s - loss: 0.0167 - val_loss: 0.0163
Epoch 296/300
1128/1128 - 0s - loss: 0.0167 - val_loss: 0.0163
Epoch 297/300
1128/1128 - 0s - loss: 0.0167 - val_loss: 0.0163
Epoch 298/300
1128/1128 - 0s - loss: 0.0167 - val_loss: 0.0163
Epoch 299/300
1128/1128 - 0s - loss: 0.0167 - val_loss: 0.0163
Epoch 300/300
1128/1128 - 0s - loss: 0.0166 - val_loss: 0.0163
```

In [20]:

```
eval=model.evaluate(x_test, x_test)
print('loss')
round(eval,3)
```

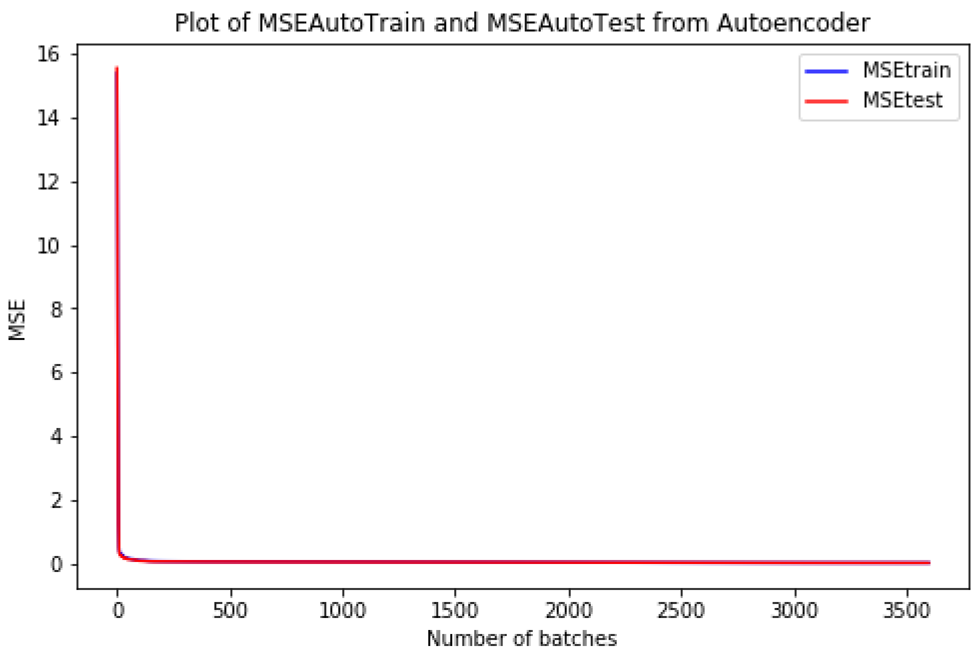
```
126/126 [==============================] - 0s 48us/sample - loss: 0.0163
loss
```

Out[20]:

```
0.016
```

In [21]:
```
#plot MSE
labels=['MSEtrain','MSEtest']
%matplotlib inline
plt.figure(figsize=(8, 5))
plt.figure(1)
plt.plot(MyMonitor.MSEtrain, label='MSEAutoTrain', color='b')
plt.plot(MyMonitor.MSEtest, label='MSEAutoTest', color='r')
plt.legend(labels)
plt.ylabel('MSE')
plt.xlabel('Number of batches')
plt.title('Plot of MSEAutoTrain and MSEAutoTest from Autoencoder')
```

Out[21]:
```
Text(0.5, 1.0, 'Plot of MSEAutoTrain and MSEAutoTest from Autoencoder')
```



Compute Compressed Inputs

In [22]:
```
# extract the hidden layer
Htrain = model.layers[0](np.asarray(x_train)).numpy()
Htest = model.layers[0](np.asarray(x_test)).numpy()
print(Htrain.shape)
Htrain
```

```
(1128, 3)
```

Out[22]:
```
array([[10.879543 , 12.2064295, 11.582977 ],
       [12.381304 , 12.3067255,  5.245779 ],
       [10.679415 , 10.988287 , 10.916448 ],
       ...,
       [12.039141 , 12.153407 ,  6.757164 ],
       [10.535016 , 11.84597  , 11.22563  ],
       [11.655713 , 11.67267  ,  7.2585554]], dtype=float32)
```

In [23]:
```
Htrain.shape,Htest.shape
```
Out[23]:
```
((1128, 3), (126, 3))
```
In [24]:
```
y_train=pd.DataFrame(y_train).values.reshape(1128,1)
y_test=pd.DataFrame(y_test).values.reshape(126,1)
```

MLP predictor (deep learning method)

In [74]:
```
# K=10
mlp10 = Sequential()
mlp10.add(Dense(10, activation='relu', input_dim=h, bias_initializer=Constant(
value=5)))
mlp10.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp10.summary()
mlp10.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
# configure suitable lr and decay

mlp10.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor10 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor10 = mlp10.fit(Htrain, y_train, epochs=50, batch_size=320, callbacks
= [mlpMyMonitor10, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_44 (Dense)             (None, 10)                40        
_________________________________________________________________
dense_45 (Dense)             (None, 1)                 11        
=================================================================
Total params: 51
Trainable params: 51
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
1128/1128 - 0s - loss: 9.2913 - val_loss: 0.7088
Epoch 2/50
1128/1128 - 0s - loss: 0.5042 - val_loss: 0.1995
Epoch 3/50
1128/1128 - 0s - loss: 0.2206 - val_loss: 0.1748
Epoch 4/50
1128/1128 - 0s - loss: 0.2031 - val_loss: 0.1745
Epoch 5/50
1128/1128 - 0s - loss: 0.2014 - val_loss: 0.1742
Epoch 6/50
1128/1128 - 0s - loss: 0.2010 - val_loss: 0.1738
Epoch 7/50
1128/1128 - 0s - loss: 0.2001 - val_loss: 0.1739
Epoch 8/50
1128/1128 - 0s - loss: 0.1998 - val_loss: 0.1741
Epoch 9/50
1128/1128 - 0s - loss: 0.1990 - val_loss: 0.1726
Epoch 10/50
1128/1128 - 0s - loss: 0.1987 - val_loss: 0.1734
Epoch 11/50
1128/1128 - 0s - loss: 0.1979 - val_loss: 0.1724
Epoch 12/50
1128/1128 - 0s - loss: 0.1971 - val_loss: 0.1717
Epoch 13/50
1128/1128 - 0s - loss: 0.1965 - val_loss: 0.1710
Epoch 14/50
1128/1128 - 0s - loss: 0.1960 - val_loss: 0.1709
Epoch 15/50
1128/1128 - 0s - loss: 0.1954 - val_loss: 0.1713
Epoch 16/50
1128/1128 - 0s - loss: 0.1949 - val_loss: 0.1702
Epoch 17/50
1128/1128 - 0s - loss: 0.1945 - val_loss: 0.1696
Epoch 18/50
1128/1128 - 0s - loss: 0.1937 - val_loss: 0.1689
Epoch 19/50
1128/1128 - 0s - loss: 0.1932 - val_loss: 0.1698
Epoch 20/50
1128/1128 - 0s - loss: 0.1928 - val_loss: 0.1690
Epoch 21/50
1128/1128 - 0s - loss: 0.1923 - val_loss: 0.1687
Epoch 22/50
1128/1128 - 0s - loss: 0.1916 - val_loss: 0.1675
```

```
Epoch 23/50
1128/1128 - 0s - loss: 0.1911 - val_loss: 0.1669
Epoch 24/50
1128/1128 - 0s - loss: 0.1906 - val_loss: 0.1666
Epoch 25/50
1128/1128 - 0s - loss: 0.1901 - val_loss: 0.1664
Epoch 26/50
1128/1128 - 0s - loss: 0.1899 - val_loss: 0.1657
Epoch 27/50
1128/1128 - 0s - loss: 0.1891 - val_loss: 0.1657
Epoch 28/50
1128/1128 - 0s - loss: 0.1887 - val_loss: 0.1662
Epoch 29/50
1128/1128 - 0s - loss: 0.1882 - val_loss: 0.1647
Epoch 30/50
1128/1128 - 0s - loss: 0.1879 - val_loss: 0.1654
Epoch 31/50
1128/1128 - 0s - loss: 0.1874 - val_loss: 0.1648
Epoch 32/50
1128/1128 - 0s - loss: 0.1869 - val_loss: 0.1643
Epoch 33/50
1128/1128 - 0s - loss: 0.1862 - val_loss: 0.1635
Epoch 34/50
1128/1128 - 0s - loss: 0.1859 - val_loss: 0.1631
Epoch 35/50
1128/1128 - 0s - loss: 0.1854 - val_loss: 0.1633
Epoch 36/50
1128/1128 - 0s - loss: 0.1850 - val_loss: 0.1635
Epoch 37/50
1128/1128 - 0s - loss: 0.1846 - val_loss: 0.1625
Epoch 38/50
1128/1128 - 0s - loss: 0.1841 - val_loss: 0.1627
Epoch 39/50
1128/1128 - 0s - loss: 0.1837 - val_loss: 0.1621
Epoch 40/50
1128/1128 - 0s - loss: 0.1832 - val_loss: 0.1616
Epoch 41/50
1128/1128 - 0s - loss: 0.1829 - val_loss: 0.1615
Epoch 42/50
1128/1128 - 0s - loss: 0.1827 - val_loss: 0.1614
Epoch 43/50
1128/1128 - 0s - loss: 0.1821 - val_loss: 0.1606
Epoch 44/50
1128/1128 - 0s - loss: 0.1817 - val_loss: 0.1603
Epoch 45/50
1128/1128 - 0s - loss: 0.1813 - val_loss: 0.1599
Epoch 46/50
1128/1128 - 0s - loss: 0.1811 - val_loss: 0.1602
Epoch 47/50
1128/1128 - 0s - loss: 0.1805 - val_loss: 0.1597
Epoch 48/50
1128/1128 - 0s - loss: 0.1802 - val_loss: 0.1594
Epoch 49/50
1128/1128 - 0s - loss: 0.1799 - val_loss: 0.1589
Epoch 50/50
1128/1128 - 0s - loss: 0.1797 - val_loss: 0.1585
```

In [75]:
```
z_train10=mlp10.predict(Htrain)
MREPtrain10=np.average(abs(z_train10-y_train)/y_train)
print('k=10, MREP_train:',round(MREPtrain10,3))
```

```
k=10, MREP_train: 0.444
```

In [76]:
```
# K=30
mlp30 = Sequential()
mlp30.add(Dense(30, activation='relu', input_dim=h, bias_initializer=Constant(
value=5)))
mlp30.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp30.summary()
mlp30.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
# configure suitable lr and decay

mlp30.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor30 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor30 = mlp30.fit(Htrain, y_train, epochs=50, batch_size=320, callbacks
= [mlpMyMonitor30, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_23"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_46 (Dense)             (None, 30)                120       
_________________________________________________________________
dense_47 (Dense)             (None, 1)                 31        
=================================================================
Total params: 151
Trainable params: 151
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
1128/1128 - 0s - loss: 367.6738 - val_loss: 3.0259
Epoch 2/50
1128/1128 - 0s - loss: 1.1735 - val_loss: 0.1447
Epoch 3/50
1128/1128 - 0s - loss: 0.1499 - val_loss: 0.1332
Epoch 4/50
1128/1128 - 0s - loss: 0.1411 - val_loss: 0.1326
Epoch 5/50
1128/1128 - 0s - loss: 0.1407 - val_loss: 0.1323
Epoch 6/50
1128/1128 - 0s - loss: 0.1400 - val_loss: 0.1319
Epoch 7/50
1128/1128 - 0s - loss: 0.1395 - val_loss: 0.1316
Epoch 8/50
1128/1128 - 0s - loss: 0.1390 - val_loss: 0.1310
Epoch 9/50
1128/1128 - 0s - loss: 0.1389 - val_loss: 0.1306
Epoch 10/50
1128/1128 - 0s - loss: 0.1383 - val_loss: 0.1305
Epoch 11/50
1128/1128 - 0s - loss: 0.1377 - val_loss: 0.1298
Epoch 12/50
1128/1128 - 0s - loss: 0.1377 - val_loss: 0.1293
Epoch 13/50
1128/1128 - 0s - loss: 0.1371 - val_loss: 0.1293
Epoch 14/50
1128/1128 - 0s - loss: 0.1365 - val_loss: 0.1287
Epoch 15/50
1128/1128 - 0s - loss: 0.1359 - val_loss: 0.1282
Epoch 16/50
1128/1128 - 0s - loss: 0.1356 - val_loss: 0.1278
Epoch 17/50
1128/1128 - 0s - loss: 0.1352 - val_loss: 0.1272
Epoch 18/50
1128/1128 - 0s - loss: 0.1348 - val_loss: 0.1269
Epoch 19/50
1128/1128 - 0s - loss: 0.1343 - val_loss: 0.1267
Epoch 20/50
1128/1128 - 0s - loss: 0.1338 - val_loss: 0.1260
Epoch 21/50
1128/1128 - 0s - loss: 0.1335 - val_loss: 0.1259
Epoch 22/50
1128/1128 - 0s - loss: 0.1330 - val_loss: 0.1255
```

```
Epoch 23/50
1128/1128 - 0s - loss: 0.1327 - val_loss: 0.1245
Epoch 24/50
1128/1128 - 0s - loss: 0.1322 - val_loss: 0.1246
Epoch 25/50
1128/1128 - 0s - loss: 0.1319 - val_loss: 0.1239
Epoch 26/50
1128/1128 - 0s - loss: 0.1314 - val_loss: 0.1240
Epoch 27/50
1128/1128 - 0s - loss: 0.1309 - val_loss: 0.1236
Epoch 28/50
1128/1128 - 0s - loss: 0.1305 - val_loss: 0.1223
Epoch 29/50
1128/1128 - 0s - loss: 0.1302 - val_loss: 0.1218
Epoch 30/50
1128/1128 - 0s - loss: 0.1298 - val_loss: 0.1216
Epoch 31/50
1128/1128 - 0s - loss: 0.1292 - val_loss: 0.1215
Epoch 32/50
1128/1128 - 0s - loss: 0.1288 - val_loss: 0.1212
Epoch 33/50
1128/1128 - 0s - loss: 0.1285 - val_loss: 0.1208
Epoch 34/50
1128/1128 - 0s - loss: 0.1281 - val_loss: 0.1199
Epoch 35/50
1128/1128 - 0s - loss: 0.1280 - val_loss: 0.1192
Epoch 36/50
1128/1128 - 0s - loss: 0.1275 - val_loss: 0.1190
Epoch 37/50
1128/1128 - 0s - loss: 0.1271 - val_loss: 0.1193
Epoch 38/50
1128/1128 - 0s - loss: 0.1268 - val_loss: 0.1198
Epoch 39/50
1128/1128 - 0s - loss: 0.1266 - val_loss: 0.1196
Epoch 40/50
1128/1128 - 0s - loss: 0.1262 - val_loss: 0.1186
Epoch 41/50
1128/1128 - 0s - loss: 0.1261 - val_loss: 0.1184
Epoch 42/50
1128/1128 - 0s - loss: 0.1258 - val_loss: 0.1186
Epoch 43/50
1128/1128 - 0s - loss: 0.1253 - val_loss: 0.1173
Epoch 44/50
1128/1128 - 0s - loss: 0.1251 - val_loss: 0.1178
Epoch 45/50
1128/1128 - 0s - loss: 0.1250 - val_loss: 0.1163
Epoch 46/50
1128/1128 - 0s - loss: 0.1248 - val_loss: 0.1166
Epoch 47/50
1128/1128 - 0s - loss: 0.1242 - val_loss: 0.1160
Epoch 48/50
1128/1128 - 0s - loss: 0.1240 - val_loss: 0.1157
Epoch 49/50
1128/1128 - 0s - loss: 0.1238 - val_loss: 0.1167
Epoch 50/50
1128/1128 - 0s - loss: 0.1237 - val_loss: 0.1157
```

In [78]:
```
z_train30=mlp30.predict(Htrain)
MREPtrain30=np.average(abs(z_train30-y_train)/y_train)
print('k=30, MREP_train:',round(MREPtrain30,3))
```

```
k=30, MREP_train: 0.385
```

In [79]:
```
# K=50
mlp50 = Sequential()
mlp50.add(Dense(50, activation='relu', input_dim=h, bias_initializer=Constant(
value=5)))
mlp50.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp50.summary()
mlp50.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
# configure suitable lr and decay

mlp50.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor50 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor50 = mlp50.fit(Htrain, y_train, epochs=50, batch_size=320, callbacks
= [mlpMyMonitor50, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_24"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_48 (Dense)             (None, 50)                200       
_________________________________________________________________
dense_49 (Dense)             (None, 1)                 51        
=================================================================
Total params: 251
Trainable params: 251
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.101471). Check your callbacks.
1128/1128 - 0s - loss: 553.6958 - val_loss: 3.7321
Epoch 2/50
1128/1128 - 0s - loss: 1.3334 - val_loss: 0.0706
Epoch 3/50
1128/1128 - 0s - loss: 0.0716 - val_loss: 0.0643
Epoch 4/50
1128/1128 - 0s - loss: 0.0693 - val_loss: 0.0643
Epoch 5/50
1128/1128 - 0s - loss: 0.0692 - val_loss: 0.0642
Epoch 6/50
1128/1128 - 0s - loss: 0.0693 - val_loss: 0.0639
Epoch 7/50
1128/1128 - 0s - loss: 0.0690 - val_loss: 0.0637
Epoch 8/50
1128/1128 - 0s - loss: 0.0687 - val_loss: 0.0637
Epoch 9/50
1128/1128 - 0s - loss: 0.0686 - val_loss: 0.0636
Epoch 10/50
1128/1128 - 0s - loss: 0.0685 - val_loss: 0.0636
Epoch 11/50
1128/1128 - 0s - loss: 0.0684 - val_loss: 0.0634
Epoch 12/50
1128/1128 - 0s - loss: 0.0682 - val_loss: 0.0632
Epoch 13/50
1128/1128 - 0s - loss: 0.0681 - val_loss: 0.0634
Epoch 14/50
1128/1128 - 0s - loss: 0.0680 - val_loss: 0.0630
Epoch 15/50
1128/1128 - 0s - loss: 0.0680 - val_loss: 0.0629
Epoch 16/50
1128/1128 - 0s - loss: 0.0678 - val_loss: 0.0628
Epoch 17/50
1128/1128 - 0s - loss: 0.0677 - val_loss: 0.0631
Epoch 18/50
1128/1128 - 0s - loss: 0.0676 - val_loss: 0.0626
Epoch 19/50
1128/1128 - 0s - loss: 0.0675 - val_loss: 0.0630
Epoch 20/50
1128/1128 - 0s - loss: 0.0675 - val_loss: 0.0625
Epoch 21/50
1128/1128 - 0s - loss: 0.0673 - val_loss: 0.0624
```

```
Epoch 22/50
1128/1128 - 0s - loss: 0.0671 - val_loss: 0.0623
Epoch 23/50
1128/1128 - 0s - loss: 0.0671 - val_loss: 0.0623
Epoch 24/50
1128/1128 - 0s - loss: 0.0669 - val_loss: 0.0622
Epoch 25/50
1128/1128 - 0s - loss: 0.0669 - val_loss: 0.0623
Epoch 26/50
1128/1128 - 0s - loss: 0.0668 - val_loss: 0.0621
Epoch 27/50
1128/1128 - 0s - loss: 0.0667 - val_loss: 0.0620
Epoch 28/50
1128/1128 - 0s - loss: 0.0666 - val_loss: 0.0620
Epoch 29/50
1128/1128 - 0s - loss: 0.0666 - val_loss: 0.0619
Epoch 30/50
1128/1128 - 0s - loss: 0.0665 - val_loss: 0.0619
Epoch 31/50
1128/1128 - 0s - loss: 0.0663 - val_loss: 0.0617
Epoch 32/50
1128/1128 - 0s - loss: 0.0663 - val_loss: 0.0617
Epoch 33/50
1128/1128 - 0s - loss: 0.0662 - val_loss: 0.0618
Epoch 34/50
1128/1128 - 0s - loss: 0.0661 - val_loss: 0.0615
Epoch 35/50
1128/1128 - 0s - loss: 0.0661 - val_loss: 0.0616
Epoch 36/50
1128/1128 - 0s - loss: 0.0661 - val_loss: 0.0615
Epoch 37/50
1128/1128 - 0s - loss: 0.0660 - val_loss: 0.0614
Epoch 38/50
1128/1128 - 0s - loss: 0.0658 - val_loss: 0.0614
Epoch 39/50
1128/1128 - 0s - loss: 0.0657 - val_loss: 0.0613
Epoch 40/50
1128/1128 - 0s - loss: 0.0657 - val_loss: 0.0612
Epoch 41/50
1128/1128 - 0s - loss: 0.0656 - val_loss: 0.0612
Epoch 42/50
1128/1128 - 0s - loss: 0.0655 - val_loss: 0.0613
Epoch 43/50
1128/1128 - 0s - loss: 0.0656 - val_loss: 0.0611
Epoch 44/50
1128/1128 - 0s - loss: 0.0658 - val_loss: 0.0610
Epoch 45/50
1128/1128 - 0s - loss: 0.0654 - val_loss: 0.0614
Epoch 46/50
1128/1128 - 0s - loss: 0.0654 - val_loss: 0.0610
Epoch 47/50
1128/1128 - 0s - loss: 0.0652 - val_loss: 0.0609
Epoch 48/50
1128/1128 - 0s - loss: 0.0651 - val_loss: 0.0608
Epoch 49/50
1128/1128 - 0s - loss: 0.0652 - val_loss: 0.0609
```

```
Epoch 50/50
1128/1128 - 0s - loss: 0.0650 - val_loss: 0.0608
```

In [80]:
```
z_train50=mlp50.predict(Htrain)
MREPtrain50=np.average(abs(z_train50-y_train)/y_train)
print('k=50, MREP_train:',round(MREPtrain50,3))
```

```
k=50, MREP_train: 0.289
```

In [81]:
```
# K=100
mlp100 = Sequential()
mlp100.add(Dense(100, activation='relu', input_dim=h, bias_initializer=Constan
t(value=5)))
mlp100.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp100.summary()
mlp100.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
# configure suitable lr and decay

mlp100.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor100 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor100 = mlp100.fit(Htrain, y_train, epochs=50, batch_size=320, callbac
ks = [mlpMyMonitor100, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_25"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_50 (Dense)             (None, 100)               400       
_________________________________________________________________
dense_51 (Dense)             (None, 1)                 101       
=================================================================
Total params: 501
Trainable params: 501
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.216481). Check your callbacks.
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.127512). Check your callbacks.
1128/1128 - 1s - loss: 608.7245 - val_loss: 0.1285
Epoch 2/50
1128/1128 - 0s - loss: 0.1446 - val_loss: 0.1231
Epoch 3/50
1128/1128 - 0s - loss: 0.1367 - val_loss: 0.1221
Epoch 4/50
1128/1128 - 0s - loss: 0.1354 - val_loss: 0.1202
Epoch 5/50
1128/1128 - 0s - loss: 0.1333 - val_loss: 0.1181
Epoch 6/50
1128/1128 - 0s - loss: 0.1314 - val_loss: 0.1167
Epoch 7/50
1128/1128 - 0s - loss: 0.1297 - val_loss: 0.1165
Epoch 8/50
1128/1128 - 0s - loss: 0.1282 - val_loss: 0.1132
Epoch 9/50
1128/1128 - 0s - loss: 0.1264 - val_loss: 0.1155
Epoch 10/50
1128/1128 - 0s - loss: 0.1249 - val_loss: 0.1107
Epoch 11/50
1128/1128 - 0s - loss: 0.1232 - val_loss: 0.1088
Epoch 12/50
1128/1128 - 0s - loss: 0.1218 - val_loss: 0.1070
Epoch 13/50
1128/1128 - 0s - loss: 0.1202 - val_loss: 0.1065
Epoch 14/50
1128/1128 - 0s - loss: 0.1186 - val_loss: 0.1040
Epoch 15/50
1128/1128 - 0s - loss: 0.1172 - val_loss: 0.1051
Epoch 16/50
1128/1128 - 0s - loss: 0.1158 - val_loss: 0.1032
Epoch 17/50
1128/1128 - 0s - loss: 0.1144 - val_loss: 0.1012
Epoch 18/50
1128/1128 - 0s - loss: 0.1128 - val_loss: 0.0989
Epoch 19/50
1128/1128 - 0s - loss: 0.1113 - val_loss: 0.0998
Epoch 20/50
1128/1128 - 0s - loss: 0.1101 - val_loss: 0.0968
```

```
Epoch 21/50
1128/1128 - 0s - loss: 0.1088 - val_loss: 0.0959
Epoch 22/50
1128/1128 - 0s - loss: 0.1078 - val_loss: 0.0955
Epoch 23/50
1128/1128 - 0s - loss: 0.1060 - val_loss: 0.0948
Epoch 24/50
1128/1128 - 0s - loss: 0.1046 - val_loss: 0.0924
Epoch 25/50
1128/1128 - 0s - loss: 0.1033 - val_loss: 0.0922
Epoch 26/50
1128/1128 - 0s - loss: 0.1020 - val_loss: 0.0900
Epoch 27/50
1128/1128 - 0s - loss: 0.1008 - val_loss: 0.0879
Epoch 28/50
1128/1128 - 0s - loss: 0.0999 - val_loss: 0.0885
Epoch 29/50
1128/1128 - 0s - loss: 0.0986 - val_loss: 0.0859
Epoch 30/50
1128/1128 - 0s - loss: 0.0973 - val_loss: 0.0850
Epoch 31/50
1128/1128 - 0s - loss: 0.0960 - val_loss: 0.0852
Epoch 32/50
1128/1128 - 0s - loss: 0.0951 - val_loss: 0.0849
Epoch 33/50
1128/1128 - 0s - loss: 0.0938 - val_loss: 0.0834
Epoch 34/50
1128/1128 - 0s - loss: 0.0926 - val_loss: 0.0818
Epoch 35/50
1128/1128 - 0s - loss: 0.0916 - val_loss: 0.0804
Epoch 36/50
1128/1128 - 0s - loss: 0.0904 - val_loss: 0.0805
Epoch 37/50
1128/1128 - 0s - loss: 0.0894 - val_loss: 0.0790
Epoch 38/50
1128/1128 - 0s - loss: 0.0883 - val_loss: 0.0781
Epoch 39/50
1128/1128 - 0s - loss: 0.0876 - val_loss: 0.0767
Epoch 40/50
1128/1128 - 0s - loss: 0.0865 - val_loss: 0.0771
Epoch 41/50
1128/1128 - 0s - loss: 0.0856 - val_loss: 0.0762
Epoch 42/50
1128/1128 - 0s - loss: 0.0845 - val_loss: 0.0744
Epoch 43/50
1128/1128 - 0s - loss: 0.0836 - val_loss: 0.0732
Epoch 44/50
1128/1128 - 0s - loss: 0.0828 - val_loss: 0.0729
Epoch 45/50
1128/1128 - 0s - loss: 0.0817 - val_loss: 0.0717
Epoch 46/50
1128/1128 - 0s - loss: 0.0812 - val_loss: 0.0712
Epoch 47/50
1128/1128 - 0s - loss: 0.0803 - val_loss: 0.0705
Epoch 48/50
1128/1128 - 0s - loss: 0.0797 - val_loss: 0.0703
Epoch 49/50
```

```
1128/1128 - 0s - loss: 0.0784 - val_loss: 0.0693
Epoch 50/50
1128/1128 - 0s - loss: 0.0778 - val_loss: 0.0685
```

In [82]:
```
z_train100=mlp100.predict(Htrain)
MREPtrain100=np.average(abs(z_train100-y_train)/y_train)
print('k=100, MREP_train:',round(MREPtrain100,3))
```

```
k=100, MREP_train: 0.287
```

In [83]:
```
# K=150
mlp150 = Sequential()
mlp150.add(Dense(150, activation='relu', input_dim=h, bias_initializer=Constan
t(value=5)))
mlp150.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp150.summary()
mlp150.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
# configure suitable lr and decay

mlp150.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor150 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor150 = mlp150.fit(Htrain, y_train, epochs=50, batch_size=320, callbac
ks = [mlpMyMonitor150, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_26"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_52 (Dense)             (None, 150)               600       
_________________________________________________________________
dense_53 (Dense)             (None, 1)                 151       
=================================================================
Total params: 751
Trainable params: 751
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.105934). Check your callbacks.
1128/1128 - 0s - loss: 620.1870 - val_loss: 1.0633
Epoch 2/50
1128/1128 - 0s - loss: 1.1788 - val_loss: 0.9490
Epoch 3/50
1128/1128 - 0s - loss: 0.8651 - val_loss: 0.4921
Epoch 4/50
1128/1128 - 0s - loss: 0.2466 - val_loss: 0.0927
Epoch 5/50
1128/1128 - 0s - loss: 0.1005 - val_loss: 0.0941
Epoch 6/50
1128/1128 - 0s - loss: 0.1008 - val_loss: 0.0937
Epoch 7/50
1128/1128 - 0s - loss: 0.0995 - val_loss: 0.0895
Epoch 8/50
1128/1128 - 0s - loss: 0.0988 - val_loss: 0.0910
Epoch 9/50
1128/1128 - 0s - loss: 0.0985 - val_loss: 0.0894
Epoch 10/50
1128/1128 - 0s - loss: 0.0980 - val_loss: 0.0884
Epoch 11/50
1128/1128 - 0s - loss: 0.0978 - val_loss: 0.0895
Epoch 12/50
1128/1128 - 0s - loss: 0.0969 - val_loss: 0.0877
Epoch 13/50
1128/1128 - 0s - loss: 0.0968 - val_loss: 0.0874
Epoch 14/50
1128/1128 - 0s - loss: 0.0964 - val_loss: 0.0870
Epoch 15/50
1128/1128 - 0s - loss: 0.0958 - val_loss: 0.0866
Epoch 16/50
1128/1128 - 0s - loss: 0.0953 - val_loss: 0.0864
Epoch 17/50
1128/1128 - 0s - loss: 0.0951 - val_loss: 0.0895
Epoch 18/50
1128/1128 - 0s - loss: 0.0955 - val_loss: 0.0861
Epoch 19/50
1128/1128 - 0s - loss: 0.0942 - val_loss: 0.0863
Epoch 20/50
1128/1128 - 0s - loss: 0.0943 - val_loss: 0.0856
Epoch 21/50
1128/1128 - 0s - loss: 0.0935 - val_loss: 0.0850
```

```
Epoch 22/50
1128/1128 - 0s - loss: 0.0932 - val_loss: 0.0846
Epoch 23/50
1128/1128 - 0s - loss: 0.0927 - val_loss: 0.0857
Epoch 24/50
1128/1128 - 0s - loss: 0.0928 - val_loss: 0.0840
Epoch 25/50
1128/1128 - 0s - loss: 0.0931 - val_loss: 0.0839
Epoch 26/50
1128/1128 - 0s - loss: 0.0917 - val_loss: 0.0834
Epoch 27/50
1128/1128 - 0s - loss: 0.0913 - val_loss: 0.0834
Epoch 28/50
1128/1128 - 0s - loss: 0.0912 - val_loss: 0.0832
Epoch 29/50
1128/1128 - 0s - loss: 0.0910 - val_loss: 0.0825
Epoch 30/50
1128/1128 - 0s - loss: 0.0903 - val_loss: 0.0831
Epoch 31/50
1128/1128 - 0s - loss: 0.0904 - val_loss: 0.0820
Epoch 32/50
1128/1128 - 0s - loss: 0.0902 - val_loss: 0.0823
Epoch 33/50
1128/1128 - 0s - loss: 0.0900 - val_loss: 0.0826
Epoch 34/50
1128/1128 - 0s - loss: 0.0895 - val_loss: 0.0811
Epoch 35/50
1128/1128 - 0s - loss: 0.0889 - val_loss: 0.0815
Epoch 36/50
1128/1128 - 0s - loss: 0.0886 - val_loss: 0.0819
Epoch 37/50
1128/1128 - 0s - loss: 0.0881 - val_loss: 0.0803
Epoch 38/50
1128/1128 - 0s - loss: 0.0882 - val_loss: 0.0802
Epoch 39/50
1128/1128 - 0s - loss: 0.0877 - val_loss: 0.0808
Epoch 40/50
1128/1128 - 0s - loss: 0.0875 - val_loss: 0.0805
Epoch 41/50
1128/1128 - 0s - loss: 0.0874 - val_loss: 0.0810
Epoch 42/50
1128/1128 - 0s - loss: 0.0870 - val_loss: 0.0794
Epoch 43/50
1128/1128 - 0s - loss: 0.0868 - val_loss: 0.0789
Epoch 44/50
1128/1128 - 0s - loss: 0.0864 - val_loss: 0.0787
Epoch 45/50
1128/1128 - 0s - loss: 0.0864 - val_loss: 0.0788
Epoch 46/50
1128/1128 - 0s - loss: 0.0862 - val_loss: 0.0785
Epoch 47/50
1128/1128 - 0s - loss: 0.0856 - val_loss: 0.0789
Epoch 48/50
1128/1128 - 0s - loss: 0.0853 - val_loss: 0.0779
Epoch 49/50
1128/1128 - 0s - loss: 0.0852 - val_loss: 0.0783
```

```
Epoch 50/50
1128/1128 - 0s - loss: 0.0850 - val_loss: 0.0792
```

In [84]:
```
z_train150=mlp150.predict(Htrain)
MREPtrain150=np.average(abs(z_train150-y_train)/y_train)
print('k=150, MREP_train:',round(MREPtrain150,3))
```

```
k=150, MREP_train: 0.321
```

In [85]:
```
# K=200
mlp200 = Sequential()
mlp200.add(Dense(200, activation='relu', input_dim=h, bias_initializer=Constan
t(value=5)))
mlp200.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp200.summary()
mlp200.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
# configure suitable lr and decay

mlp200.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor200 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor200 = mlp200.fit(Htrain, y_train, epochs=50, batch_size=320, callbac
ks = [mlpMyMonitor200, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_27"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_54 (Dense)             (None, 200)               800       
_________________________________________________________________
dense_55 (Dense)             (None, 1)                 201       
=================================================================
Total params: 1,001
Trainable params: 1,001
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.101748). Check your callbacks.
1128/1128 - 0s - loss: 573.0659 - val_loss: 1.0812
Epoch 2/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 3/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 4/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 5/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 6/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 7/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 8/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 9/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 10/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 11/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 12/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 13/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 14/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 15/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 16/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 17/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 18/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 19/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 20/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 21/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

```
Epoch 22/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 23/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 24/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 25/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 26/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 27/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 28/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 29/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 30/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 31/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 32/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 33/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 34/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 35/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 36/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 37/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 38/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 39/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 40/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 41/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 42/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 43/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 44/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 45/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 46/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 47/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 48/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 49/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

```
Epoch 50/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

In [86]:

```
z_train200=mlp200.predict(Htrain)
MREPtrain200=np.average(abs(z_train200-y_train)/y_train)
print('k=200, MREP_train:',round(MREPtrain200,3))
```

```
k=200, MREP_train: 1.0
```

In [87]:

```
# K=225
mlp225 = Sequential()
mlp225.add(Dense(225, activation='relu', input_dim=h, bias_initializer=Constan
t(value=5)))
mlp225.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlp225.summary()
mlp225.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
# configure suitable lr and decay

mlp225.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), los
s='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitor225 = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitor225 = mlp225.fit(Htrain, y_train, epochs=50, batch_size=320, callbac
ks = [mlpMyMonitor225, es], validation_data = (Htest, y_test), verbose = 2)
```

```
Model: "sequential_28"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_56 (Dense)             (None, 225)               900       
_________________________________________________________________
dense_57 (Dense)             (None, 1)                 226       
=================================================================
Total params: 1,126
Trainable params: 1,126
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/50
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch 
update (0.102935). Check your callbacks.
1128/1128 - 0s - loss: 408.6348 - val_loss: 1.0812
Epoch 2/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 3/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 4/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 5/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 6/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 7/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 8/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 9/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 10/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 11/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 12/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 13/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 14/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 15/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 16/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 17/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 18/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 19/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 20/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 21/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

```
Epoch 22/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 23/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 24/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 25/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 26/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 27/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 28/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 29/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 30/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 31/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 32/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 33/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 34/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 35/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 36/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 37/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 38/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 39/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 40/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 41/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 42/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 43/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 44/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 45/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 46/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 47/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 48/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
Epoch 49/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

```
Epoch 50/50
1128/1128 - 0s - loss: 1.2592 - val_loss: 1.0812
```

In [88]:
```
z_train225=mlp225.predict(Htrain)
MREPtrain225=np.average(abs(z_train225-y_train)/y_train)
print('k=225, MREP_train:',round(MREPtrain225,3))
```

```
k=225, MREP_train: 1.0
```

In [90]:
```
z_train=mlp100.predict(Htrain)
z_test=mlp100.predict(Htest)
```

Evaluation of Result

In [ ]:

In [98]:
```
import time
start = time.time()

# K=100
mlpff = Sequential()
mlpff.add(Dense(100, activation='relu', input_dim=h, bias_initializer=Constant
(value=5)))
mlpff.add(Dense(1, activation='relu', bias_initializer=Constant(value=40)))
mlpff.summary()
mlpff.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
# configure suitable lr and decay

mlpff.compile(optimizer=optimizers.SGD(learning_rate=0.0001, decay=1e-6), loss
='mean_squared_error')
class mlpMyHistory(callbacks.Callback):
    def on_train_begin(self, logs={}):
      self.MSEtrain = []
      self.MSEtest = []
    def on_batch_end(self, batch, logs={}):
      self.MSEtrain.append(self.model.evaluate(Htrain,y_train,verbose = 0))
      self.MSEtest.append(self.model.evaluate(Htest,y_test,verbose = 0))

mlpMyMonitorff = mlpMyHistory()

es = callbacks.EarlyStopping(monitor='val_loss', mode='min', verbose=1, patien
ce=200, restore_best_weights=True)
mlpMonitorff = mlpff.fit(Htrain, y_train, epochs=500, batch_size=20, callbacks
= [mlpMyMonitorff, es], validation_data = (Htest, y_test), verbose = 2)

end = time.time()
```

```
Model: "sequential_31"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense_62 (Dense)             (None, 100)               400       
_________________________________________________________________
dense_63 (Dense)             (None, 1)                 101       
=================================================================
Total params: 501
Trainable params: 501
Non-trainable params: 0
_________________________________________________________________
Train on 1128 samples, validate on 126 samples
Epoch 1/500
WARNING:tensorflow:Method (on_train_batch_end) is slow compared to the batch
update (0.125050). Check your callbacks.
1128/1128 - 1s - loss: 34.7964 - val_loss: 0.0623
Epoch 2/500
1128/1128 - 1s - loss: 0.0487 - val_loss: 0.0384
Epoch 3/500
1128/1128 - 1s - loss: 0.0385 - val_loss: 0.0319
Epoch 4/500
1128/1128 - 1s - loss: 0.0324 - val_loss: 0.0317
Epoch 5/500
1128/1128 - 1s - loss: 0.0286 - val_loss: 0.0247
Epoch 6/500
1128/1128 - 1s - loss: 0.0257 - val_loss: 0.0316
Epoch 7/500
1128/1128 - 1s - loss: 0.0237 - val_loss: 0.0228
Epoch 8/500
1128/1128 - 1s - loss: 0.0220 - val_loss: 0.0208
Epoch 9/500
1128/1128 - 1s - loss: 0.0207 - val_loss: 0.0220
Epoch 10/500
1128/1128 - 1s - loss: 0.0203 - val_loss: 0.0248
Epoch 11/500
1128/1128 - 1s - loss: 0.0198 - val_loss: 0.0201
Epoch 12/500
1128/1128 - 1s - loss: 0.0194 - val_loss: 0.0189
Epoch 13/500
1128/1128 - 1s - loss: 0.0190 - val_loss: 0.0185
Epoch 14/500
1128/1128 - 1s - loss: 0.0188 - val_loss: 0.0200
Epoch 15/500
1128/1128 - 1s - loss: 0.0186 - val_loss: 0.0195
Epoch 16/500
1128/1128 - 1s - loss: 0.0183 - val_loss: 0.0184
Epoch 17/500
1128/1128 - 1s - loss: 0.0181 - val_loss: 0.0182
Epoch 18/500
1128/1128 - 1s - loss: 0.0181 - val_loss: 0.0178
Epoch 19/500
1128/1128 - 1s - loss: 0.0180 - val_loss: 0.0182
Epoch 20/500
1128/1128 - 1s - loss: 0.0177 - val_loss: 0.0203
Epoch 21/500
1128/1128 - 1s - loss: 0.0178 - val_loss: 0.0188
```

```
Epoch 22/500
1128/1128 - 1s - loss: 0.0177 - val_loss: 0.0218
Epoch 23/500
1128/1128 - 1s - loss: 0.0178 - val_loss: 0.0255
Epoch 24/500
1128/1128 - 1s - loss: 0.0176 - val_loss: 0.0171
Epoch 25/500
1128/1128 - 1s - loss: 0.0171 - val_loss: 0.0169
Epoch 26/500
1128/1128 - 1s - loss: 0.0173 - val_loss: 0.0191
Epoch 27/500
1128/1128 - 1s - loss: 0.0170 - val_loss: 0.0167
Epoch 28/500
1128/1128 - 1s - loss: 0.0168 - val_loss: 0.0187
Epoch 29/500
1128/1128 - 1s - loss: 0.0167 - val_loss: 0.0165
Epoch 30/500
1128/1128 - 1s - loss: 0.0166 - val_loss: 0.0170
Epoch 31/500
1128/1128 - 1s - loss: 0.0166 - val_loss: 0.0169
Epoch 32/500
1128/1128 - 1s - loss: 0.0166 - val_loss: 0.0164
Epoch 33/500
1128/1128 - 1s - loss: 0.0164 - val_loss: 0.0174
Epoch 34/500
1128/1128 - 1s - loss: 0.0162 - val_loss: 0.0183
Epoch 35/500
1128/1128 - 1s - loss: 0.0162 - val_loss: 0.0163
Epoch 36/500
1128/1128 - 1s - loss: 0.0161 - val_loss: 0.0161
Epoch 37/500
1128/1128 - 1s - loss: 0.0163 - val_loss: 0.0158
Epoch 38/500
1128/1128 - 1s - loss: 0.0158 - val_loss: 0.0159
Epoch 39/500
1128/1128 - 1s - loss: 0.0160 - val_loss: 0.0164
Epoch 40/500
1128/1128 - 2s - loss: 0.0157 - val_loss: 0.0176
Epoch 41/500
1128/1128 - 2s - loss: 0.0159 - val_loss: 0.0156
Epoch 42/500
1128/1128 - 2s - loss: 0.0157 - val_loss: 0.0155
Epoch 43/500
1128/1128 - 2s - loss: 0.0155 - val_loss: 0.0193
Epoch 44/500
1128/1128 - 1s - loss: 0.0157 - val_loss: 0.0158
Epoch 45/500
1128/1128 - 1s - loss: 0.0154 - val_loss: 0.0174
Epoch 46/500
1128/1128 - 1s - loss: 0.0155 - val_loss: 0.0154
Epoch 47/500
1128/1128 - 1s - loss: 0.0154 - val_loss: 0.0171
Epoch 48/500
1128/1128 - 1s - loss: 0.0152 - val_loss: 0.0150
Epoch 49/500
1128/1128 - 2s - loss: 0.0152 - val_loss: 0.0164
Epoch 50/500
```

```
1128/1128 - 1s - loss: 0.0148 - val_loss: 0.0152
Epoch 51/500
1128/1128 - 1s - loss: 0.0148 - val_loss: 0.0160
Epoch 52/500
1128/1128 - 1s - loss: 0.0152 - val_loss: 0.0160
Epoch 53/500
1128/1128 - 1s - loss: 0.0150 - val_loss: 0.0148
Epoch 54/500
1128/1128 - 1s - loss: 0.0149 - val_loss: 0.0148
Epoch 55/500
1128/1128 - 1s - loss: 0.0147 - val_loss: 0.0152
Epoch 56/500
1128/1128 - 1s - loss: 0.0146 - val_loss: 0.0146
Epoch 57/500
1128/1128 - 1s - loss: 0.0148 - val_loss: 0.0145
Epoch 58/500
1128/1128 - 1s - loss: 0.0147 - val_loss: 0.0152
Epoch 59/500
1128/1128 - 1s - loss: 0.0147 - val_loss: 0.0157
Epoch 60/500
1128/1128 - 1s - loss: 0.0146 - val_loss: 0.0143
Epoch 61/500
1128/1128 - 1s - loss: 0.0145 - val_loss: 0.0142
Epoch 62/500
1128/1128 - 1s - loss: 0.0143 - val_loss: 0.0143
Epoch 63/500
1128/1128 - 1s - loss: 0.0145 - val_loss: 0.0143
Epoch 64/500
1128/1128 - 1s - loss: 0.0148 - val_loss: 0.0155
Epoch 65/500
1128/1128 - 1s - loss: 0.0142 - val_loss: 0.0148
Epoch 66/500
1128/1128 - 1s - loss: 0.0143 - val_loss: 0.0155
Epoch 67/500
1128/1128 - 1s - loss: 0.0142 - val_loss: 0.0143
Epoch 68/500
1128/1128 - 2s - loss: 0.0143 - val_loss: 0.0155
Epoch 69/500
1128/1128 - 1s - loss: 0.0138 - val_loss: 0.0141
Epoch 70/500
1128/1128 - 1s - loss: 0.0139 - val_loss: 0.0140
Epoch 71/500
1128/1128 - 1s - loss: 0.0141 - val_loss: 0.0139
Epoch 72/500
1128/1128 - 2s - loss: 0.0138 - val_loss: 0.0141
Epoch 73/500
1128/1128 - 1s - loss: 0.0137 - val_loss: 0.0137
Epoch 74/500
1128/1128 - 2s - loss: 0.0139 - val_loss: 0.0148
Epoch 75/500
1128/1128 - 1s - loss: 0.0137 - val_loss: 0.0140
Epoch 76/500
1128/1128 - 1s - loss: 0.0137 - val_loss: 0.0144
Epoch 77/500
1128/1128 - 2s - loss: 0.0139 - val_loss: 0.0135
Epoch 78/500
1128/1128 - 1s - loss: 0.0137 - val_loss: 0.0143
```

```
Epoch 79/500
1128/1128 - 1s - loss: 0.0136 - val_loss: 0.0135
Epoch 80/500
1128/1128 - 1s - loss: 0.0135 - val_loss: 0.0192
Epoch 81/500
1128/1128 - 2s - loss: 0.0137 - val_loss: 0.0141
Epoch 82/500
1128/1128 - 1s - loss: 0.0132 - val_loss: 0.0211
Epoch 83/500
1128/1128 - 1s - loss: 0.0134 - val_loss: 0.0140
Epoch 84/500
1128/1128 - 1s - loss: 0.0136 - val_loss: 0.0159
Epoch 85/500
1128/1128 - 1s - loss: 0.0135 - val_loss: 0.0137
Epoch 86/500
1128/1128 - 1s - loss: 0.0132 - val_loss: 0.0131
Epoch 87/500
1128/1128 - 1s - loss: 0.0133 - val_loss: 0.0138
Epoch 88/500
1128/1128 - 1s - loss: 0.0132 - val_loss: 0.0142
Epoch 89/500
1128/1128 - 1s - loss: 0.0128 - val_loss: 0.0164
Epoch 90/500
1128/1128 - 1s - loss: 0.0129 - val_loss: 0.0135
Epoch 91/500
1128/1128 - 1s - loss: 0.0132 - val_loss: 0.0133
Epoch 92/500
1128/1128 - 1s - loss: 0.0129 - val_loss: 0.0129
Epoch 93/500
1128/1128 - 2s - loss: 0.0129 - val_loss: 0.0146
Epoch 94/500
1128/1128 - 1s - loss: 0.0129 - val_loss: 0.0129
Epoch 95/500
1128/1128 - 1s - loss: 0.0130 - val_loss: 0.0134
Epoch 96/500
1128/1128 - 1s - loss: 0.0127 - val_loss: 0.0263
Epoch 97/500
1128/1128 - 1s - loss: 0.0130 - val_loss: 0.0131
Epoch 98/500
1128/1128 - 2s - loss: 0.0127 - val_loss: 0.0132
Epoch 99/500
1128/1128 - 1s - loss: 0.0127 - val_loss: 0.0147
Epoch 100/500
1128/1128 - 1s - loss: 0.0127 - val_loss: 0.0146
Epoch 101/500
1128/1128 - 1s - loss: 0.0125 - val_loss: 0.0126
Epoch 102/500
1128/1128 - 1s - loss: 0.0125 - val_loss: 0.0151
Epoch 103/500
1128/1128 - 1s - loss: 0.0125 - val_loss: 0.0125
Epoch 104/500
1128/1128 - 1s - loss: 0.0125 - val_loss: 0.0151
Epoch 105/500
1128/1128 - 1s - loss: 0.0126 - val_loss: 0.0126
Epoch 106/500
1128/1128 - 1s - loss: 0.0122 - val_loss: 0.0151
Epoch 107/500
```

```
1128/1128 - 1s - loss: 0.0126 - val_loss: 0.0124
Epoch 108/500
1128/1128 - 1s - loss: 0.0125 - val_loss: 0.0124
Epoch 109/500
1128/1128 - 1s - loss: 0.0124 - val_loss: 0.0127
Epoch 110/500
1128/1128 - 2s - loss: 0.0122 - val_loss: 0.0130
Epoch 111/500
1128/1128 - 1s - loss: 0.0123 - val_loss: 0.0130
Epoch 112/500
1128/1128 - 1s - loss: 0.0122 - val_loss: 0.0139
Epoch 113/500
1128/1128 - 1s - loss: 0.0119 - val_loss: 0.0141
Epoch 114/500
1128/1128 - 1s - loss: 0.0121 - val_loss: 0.0135
Epoch 115/500
1128/1128 - 1s - loss: 0.0120 - val_loss: 0.0122
Epoch 116/500
1128/1128 - 1s - loss: 0.0123 - val_loss: 0.0126
Epoch 117/500
1128/1128 - 1s - loss: 0.0121 - val_loss: 0.0125
Epoch 118/500
1128/1128 - 1s - loss: 0.0122 - val_loss: 0.0122
Epoch 119/500
1128/1128 - 1s - loss: 0.0121 - val_loss: 0.0124
Epoch 120/500
1128/1128 - 1s - loss: 0.0120 - val_loss: 0.0128
Epoch 121/500
1128/1128 - 1s - loss: 0.0119 - val_loss: 0.0120
Epoch 122/500
1128/1128 - 1s - loss: 0.0121 - val_loss: 0.0129
Epoch 123/500
1128/1128 - 1s - loss: 0.0119 - val_loss: 0.0160
Epoch 124/500
1128/1128 - 1s - loss: 0.0118 - val_loss: 0.0157
Epoch 125/500
1128/1128 - 1s - loss: 0.0120 - val_loss: 0.0119
Epoch 126/500
1128/1128 - 1s - loss: 0.0118 - val_loss: 0.0119
Epoch 127/500
1128/1128 - 1s - loss: 0.0120 - val_loss: 0.0120
Epoch 128/500
1128/1128 - 1s - loss: 0.0120 - val_loss: 0.0118
Epoch 129/500
1128/1128 - 1s - loss: 0.0117 - val_loss: 0.0128
Epoch 130/500
1128/1128 - 1s - loss: 0.0117 - val_loss: 0.0123
Epoch 131/500
1128/1128 - 1s - loss: 0.0115 - val_loss: 0.0135
Epoch 132/500
1128/1128 - 1s - loss: 0.0116 - val_loss: 0.0123
Epoch 133/500
1128/1128 - 1s - loss: 0.0116 - val_loss: 0.0118
Epoch 134/500
1128/1128 - 2s - loss: 0.0117 - val_loss: 0.0117
Epoch 135/500
1128/1128 - 1s - loss: 0.0115 - val_loss: 0.0121
```

```
Epoch 136/500
1128/1128 - 2s - loss: 0.0114 - val_loss: 0.0134
Epoch 137/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0129
Epoch 138/500
1128/1128 - 2s - loss: 0.0114 - val_loss: 0.0124
Epoch 139/500
1128/1128 - 1s - loss: 0.0114 - val_loss: 0.0121
Epoch 140/500
1128/1128 - 2s - loss: 0.0114 - val_loss: 0.0129
Epoch 141/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0117
Epoch 142/500
1128/1128 - 1s - loss: 0.0114 - val_loss: 0.0119
Epoch 143/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0115
Epoch 144/500
1128/1128 - 1s - loss: 0.0114 - val_loss: 0.0115
Epoch 145/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0114
Epoch 146/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0120
Epoch 147/500
1128/1128 - 2s - loss: 0.0114 - val_loss: 0.0114
Epoch 148/500
1128/1128 - 1s - loss: 0.0111 - val_loss: 0.0114
Epoch 149/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0114
Epoch 150/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0113
Epoch 151/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0121
Epoch 152/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0115
Epoch 153/500
1128/1128 - 1s - loss: 0.0112 - val_loss: 0.0115
Epoch 154/500
1128/1128 - 1s - loss: 0.0113 - val_loss: 0.0119
Epoch 155/500
1128/1128 - 1s - loss: 0.0112 - val_loss: 0.0112
Epoch 156/500
1128/1128 - 1s - loss: 0.0111 - val_loss: 0.0119
Epoch 157/500
1128/1128 - 1s - loss: 0.0108 - val_loss: 0.0115
Epoch 158/500
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0120
Epoch 159/500
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0112
Epoch 160/500
1128/1128 - 1s - loss: 0.0109 - val_loss: 0.0123
Epoch 161/500
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0119
Epoch 162/500
1128/1128 - 2s - loss: 0.0111 - val_loss: 0.0117
Epoch 163/500
1128/1128 - 1s - loss: 0.0109 - val_loss: 0.0122
Epoch 164/500
```

```
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0149
Epoch 165/500
1128/1128 - 1s - loss: 0.0112 - val_loss: 0.0134
Epoch 166/500
1128/1128 - 1s - loss: 0.0108 - val_loss: 0.0137
Epoch 167/500
1128/1128 - 1s - loss: 0.0108 - val_loss: 0.0112
Epoch 168/500
1128/1128 - 1s - loss: 0.0109 - val_loss: 0.0146
Epoch 169/500
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0110
Epoch 170/500
1128/1128 - 1s - loss: 0.0108 - val_loss: 0.0131
Epoch 171/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0119
Epoch 172/500
1128/1128 - 2s - loss: 0.0108 - val_loss: 0.0109
Epoch 173/500
1128/1128 - 1s - loss: 0.0106 - val_loss: 0.0127
Epoch 174/500
1128/1128 - 1s - loss: 0.0110 - val_loss: 0.0110
Epoch 175/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0112
Epoch 176/500
1128/1128 - 2s - loss: 0.0106 - val_loss: 0.0117
Epoch 177/500
1128/1128 - 1s - loss: 0.0106 - val_loss: 0.0149
Epoch 178/500
1128/1128 - 2s - loss: 0.0107 - val_loss: 0.0118
Epoch 179/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0108
Epoch 180/500
1128/1128 - 2s - loss: 0.0106 - val_loss: 0.0113
Epoch 181/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0118
Epoch 182/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0161
Epoch 183/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0128
Epoch 184/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0107
Epoch 185/500
1128/1128 - 1s - loss: 0.0106 - val_loss: 0.0108
Epoch 186/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0110
Epoch 187/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0106
Epoch 188/500
1128/1128 - 1s - loss: 0.0104 - val_loss: 0.0149
Epoch 189/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0112
Epoch 190/500
1128/1128 - 1s - loss: 0.0107 - val_loss: 0.0117
Epoch 191/500
1128/1128 - 1s - loss: 0.0103 - val_loss: 0.0136
Epoch 192/500
1128/1128 - 1s - loss: 0.0106 - val_loss: 0.0114
```

```
Epoch 193/500
1128/1128 - 2s - loss: 0.0105 - val_loss: 0.0109
Epoch 194/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0106
Epoch 195/500
1128/1128 - 2s - loss: 0.0103 - val_loss: 0.0105
Epoch 196/500
1128/1128 - 1s - loss: 0.0103 - val_loss: 0.0106
Epoch 197/500
1128/1128 - 1s - loss: 0.0104 - val_loss: 0.0141
Epoch 198/500
1128/1128 - 1s - loss: 0.0105 - val_loss: 0.0105
Epoch 199/500
1128/1128 - 2s - loss: 0.0103 - val_loss: 0.0113
Epoch 200/500
1128/1128 - 1s - loss: 0.0103 - val_loss: 0.0105
Epoch 201/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0104
Epoch 202/500
1128/1128 - 2s - loss: 0.0104 - val_loss: 0.0106
Epoch 203/500
1128/1128 - 1s - loss: 0.0104 - val_loss: 0.0104
Epoch 204/500
1128/1128 - 2s - loss: 0.0102 - val_loss: 0.0111
Epoch 205/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0114
Epoch 206/500
1128/1128 - 1s - loss: 0.0104 - val_loss: 0.0104
Epoch 207/500
1128/1128 - 2s - loss: 0.0102 - val_loss: 0.0112
Epoch 208/500
1128/1128 - 1s - loss: 0.0104 - val_loss: 0.0106
Epoch 209/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0107
Epoch 210/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0113
Epoch 211/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0104
Epoch 212/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0103
Epoch 213/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0103
Epoch 214/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0104
Epoch 215/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0103
Epoch 216/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0105
Epoch 217/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0104
Epoch 218/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0109
Epoch 219/500
1128/1128 - 1s - loss: 0.0102 - val_loss: 0.0111
Epoch 220/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0102
Epoch 221/500
```

```
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0109
Epoch 222/500
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0109
Epoch 223/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0102
Epoch 224/500
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0110
Epoch 225/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0102
Epoch 226/500
1128/1128 - 1s - loss: 0.0103 - val_loss: 0.0104
Epoch 227/500
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0113
Epoch 228/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0104
Epoch 229/500
1128/1128 - 1s - loss: 0.0101 - val_loss: 0.0104
Epoch 230/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0114
Epoch 231/500
1128/1128 - 2s - loss: 0.0102 - val_loss: 0.0101
Epoch 232/500
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0101
Epoch 233/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0114
Epoch 234/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0101
Epoch 235/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0101
Epoch 236/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0101
Epoch 237/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0100
Epoch 238/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0101
Epoch 239/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0102
Epoch 240/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0101
Epoch 241/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0101
Epoch 242/500
1128/1128 - 2s - loss: 0.0098 - val_loss: 0.0103
Epoch 243/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0102
Epoch 244/500
1128/1128 - 2s - loss: 0.0097 - val_loss: 0.0137
Epoch 245/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0107
Epoch 246/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0110
Epoch 247/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0132
Epoch 248/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0101
Epoch 249/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0132
```

```
Epoch 250/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0112
Epoch 251/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0099
Epoch 252/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0115
Epoch 253/500
1128/1128 - 1s - loss: 0.0100 - val_loss: 0.0108
Epoch 254/500
1128/1128 - 2s - loss: 0.0098 - val_loss: 0.0099
Epoch 255/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0112
Epoch 256/500
1128/1128 - 2s - loss: 0.0098 - val_loss: 0.0133
Epoch 257/500
1128/1128 - 1s - loss: 0.0099 - val_loss: 0.0100
Epoch 258/500
1128/1128 - 2s - loss: 0.0096 - val_loss: 0.0122
Epoch 259/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0108
Epoch 260/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0098
Epoch 261/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0104
Epoch 262/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0102
Epoch 263/500
1128/1128 - 1s - loss: 0.0098 - val_loss: 0.0101
Epoch 264/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0108
Epoch 265/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0110
Epoch 266/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0115
Epoch 267/500
1128/1128 - 2s - loss: 0.0096 - val_loss: 0.0109
Epoch 268/500
1128/1128 - 3s - loss: 0.0096 - val_loss: 0.0105
Epoch 269/500
1128/1128 - 2s - loss: 0.0096 - val_loss: 0.0112
Epoch 270/500
1128/1128 - 2s - loss: 0.0096 - val_loss: 0.0166
Epoch 271/500
1128/1128 - 2s - loss: 0.0098 - val_loss: 0.0107
Epoch 272/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0099
Epoch 273/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0098
Epoch 274/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0101
Epoch 275/500
1128/1128 - 1s - loss: 0.0097 - val_loss: 0.0101
Epoch 276/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0097
Epoch 277/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0104
Epoch 278/500
```

```
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0119
Epoch 279/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0117
Epoch 280/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0117
Epoch 281/500
1128/1128 - 2s - loss: 0.0095 - val_loss: 0.0100
Epoch 282/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0097
Epoch 283/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0096
Epoch 284/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0102
Epoch 285/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0096
Epoch 286/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0097
Epoch 287/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0109
Epoch 288/500
1128/1128 - 2s - loss: 0.0094 - val_loss: 0.0096
Epoch 289/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0097
Epoch 290/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0099
Epoch 291/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0098
Epoch 292/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0122
Epoch 293/500
1128/1128 - 1s - loss: 0.0096 - val_loss: 0.0097
Epoch 294/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0099
Epoch 295/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0101
Epoch 296/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0099
Epoch 297/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0101
Epoch 298/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0113
Epoch 299/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0096
Epoch 300/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0097
Epoch 301/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0096
Epoch 302/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0099
Epoch 303/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0095
Epoch 304/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0101
Epoch 305/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0108
Epoch 306/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0096
```

```
Epoch 307/500
1128/1128 - 2s - loss: 0.0094 - val_loss: 0.0107
Epoch 308/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0096
Epoch 309/500
1128/1128 - 1s - loss: 0.0095 - val_loss: 0.0108
Epoch 310/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0116
Epoch 311/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0097
Epoch 312/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0095
Epoch 313/500
1128/1128 - 2s - loss: 0.0093 - val_loss: 0.0145
Epoch 314/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0096
Epoch 315/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0101
Epoch 316/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0100
Epoch 317/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0096
Epoch 318/500
1128/1128 - 1s - loss: 0.0094 - val_loss: 0.0101
Epoch 319/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0097
Epoch 320/500
1128/1128 - 2s - loss: 0.0093 - val_loss: 0.0099
Epoch 321/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0096
Epoch 322/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0094
Epoch 323/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0109
Epoch 324/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0133
Epoch 325/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0097
Epoch 326/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0116
Epoch 327/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0095
Epoch 328/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0099
Epoch 329/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0145
Epoch 330/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0101
Epoch 331/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0094
Epoch 332/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0098
Epoch 333/500
1128/1128 - 2s - loss: 0.0093 - val_loss: 0.0100
Epoch 334/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0094
Epoch 335/500
```

```
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0096
Epoch 336/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0096
Epoch 337/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0096
Epoch 338/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0094
Epoch 339/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0107
Epoch 340/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0139
Epoch 341/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0109
Epoch 342/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0102
Epoch 343/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0099
Epoch 344/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0093
Epoch 345/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0112
Epoch 346/500
1128/1128 - 2s - loss: 0.0092 - val_loss: 0.0136
Epoch 347/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0094
Epoch 348/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0093
Epoch 349/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0164
Epoch 350/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0104
Epoch 351/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0093
Epoch 352/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0093
Epoch 353/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0094
Epoch 354/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0093
Epoch 355/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0092
Epoch 356/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0096
Epoch 357/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0093
Epoch 358/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0093
Epoch 359/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0107
Epoch 360/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0110
Epoch 361/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0095
Epoch 362/500
1128/1128 - 2s - loss: 0.0092 - val_loss: 0.0095
Epoch 363/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0093
```

```
Epoch 364/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0094
Epoch 365/500
1128/1128 - 1s - loss: 0.0092 - val_loss: 0.0094
Epoch 366/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0094
Epoch 367/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0092
Epoch 368/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0093
Epoch 369/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0093
Epoch 370/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0094
Epoch 371/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0097
Epoch 372/500
1128/1128 - 1s - loss: 0.0093 - val_loss: 0.0094
Epoch 373/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0092
Epoch 374/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0094
Epoch 375/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0092
Epoch 376/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0092
Epoch 377/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0092
Epoch 378/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0099
Epoch 379/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0110
Epoch 380/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0092
Epoch 381/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0103
Epoch 382/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0093
Epoch 383/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0093
Epoch 384/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0099
Epoch 385/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0109
Epoch 386/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0093
Epoch 387/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0099
Epoch 388/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0102
Epoch 389/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0118
Epoch 390/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0100
Epoch 391/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0094
Epoch 392/500
```

```
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0091
Epoch 393/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0097
Epoch 394/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0091
Epoch 395/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0092
Epoch 396/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0095
Epoch 397/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0094
Epoch 398/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0101
Epoch 399/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0092
Epoch 400/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0097
Epoch 401/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0097
Epoch 402/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0091
Epoch 403/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0145
Epoch 404/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0093
Epoch 405/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0094
Epoch 406/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0092
Epoch 407/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0112
Epoch 408/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0118
Epoch 409/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0107
Epoch 410/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0104
Epoch 411/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0118
Epoch 412/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0091
Epoch 413/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0095
Epoch 414/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0095
Epoch 415/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0107
Epoch 416/500
1128/1128 - 1s - loss: 0.0091 - val_loss: 0.0096
Epoch 417/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0098
Epoch 418/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0095
Epoch 419/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 420/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0092
```

```
Epoch 421/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0104
Epoch 422/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0091
Epoch 423/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0092
Epoch 424/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0092
Epoch 425/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0103
Epoch 426/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0099
Epoch 427/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0090
Epoch 428/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0090
Epoch 429/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0107
Epoch 430/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0092
Epoch 431/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 432/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0096
Epoch 433/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0092
Epoch 434/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 435/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0146
Epoch 436/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0091
Epoch 437/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0097
Epoch 438/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 439/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0094
Epoch 440/500
1128/1128 - 2s - loss: 0.0090 - val_loss: 0.0090
Epoch 441/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0089
Epoch 442/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0109
Epoch 443/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0091
Epoch 444/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0111
Epoch 445/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0106
Epoch 446/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0092
Epoch 447/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0090
Epoch 448/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0093
Epoch 449/500
```

```
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0091
Epoch 450/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0106
Epoch 451/500
1128/1128 - 2s - loss: 0.0088 - val_loss: 0.0091
Epoch 452/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 453/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0097
Epoch 454/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0098
Epoch 455/500
1128/1128 - 2s - loss: 0.0088 - val_loss: 0.0098
Epoch 456/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0089
Epoch 457/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0097
Epoch 458/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0094
Epoch 459/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0093
Epoch 460/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0090
Epoch 461/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0090
Epoch 462/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0090
Epoch 463/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0089
Epoch 464/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0090
Epoch 465/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0093
Epoch 466/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0090
Epoch 467/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0114
Epoch 468/500
1128/1128 - 2s - loss: 0.0088 - val_loss: 0.0094
Epoch 469/500
1128/1128 - 1s - loss: 0.0089 - val_loss: 0.0137
Epoch 470/500
1128/1128 - 1s - loss: 0.0090 - val_loss: 0.0092
Epoch 471/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0089
Epoch 472/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0100
Epoch 473/500
1128/1128 - 2s - loss: 0.0088 - val_loss: 0.0100
Epoch 474/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0111
Epoch 475/500
1128/1128 - 2s - loss: 0.0089 - val_loss: 0.0094
Epoch 476/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0102
Epoch 477/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0090
```

```
Epoch 478/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0091
Epoch 479/500
1128/1128 - 1s - loss: 0.0086 - val_loss: 0.0124
Epoch 480/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0091
Epoch 481/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0091
Epoch 482/500
1128/1128 - 1s - loss: 0.0086 - val_loss: 0.0089
Epoch 483/500
1128/1128 - 1s - loss: 0.0086 - val_loss: 0.0090
Epoch 484/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0088
Epoch 485/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0092
Epoch 486/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0105
Epoch 487/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0115
Epoch 488/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0088
Epoch 489/500
1128/1128 - 3s - loss: 0.0086 - val_loss: 0.0091
Epoch 490/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0090
Epoch 491/500
1128/1128 - 3s - loss: 0.0087 - val_loss: 0.0088
Epoch 492/500
1128/1128 - 2s - loss: 0.0087 - val_loss: 0.0091
Epoch 493/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0089
Epoch 494/500
1128/1128 - 2s - loss: 0.0088 - val_loss: 0.0091
Epoch 495/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0101
Epoch 496/500
1128/1128 - 1s - loss: 0.0088 - val_loss: 0.0094
Epoch 497/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0116
Epoch 498/500
1128/1128 - 1s - loss: 0.0087 - val_loss: 0.0106
Epoch 499/500
1128/1128 - 2s - loss: 0.0086 - val_loss: 0.0090
Epoch 500/500
1128/1128 - 1s - loss: 0.0086 - val_loss: 0.0088
```

In [99]:

```
z_trainff=mlpff.predict(Htrain)
MREPtrainff=np.average(abs(z_trainff-y_train)/y_train)
print('k=ff, MREP_train:',round(MREPtrainff,3))
```

```
k=ff, MREP_train: 0.079
```

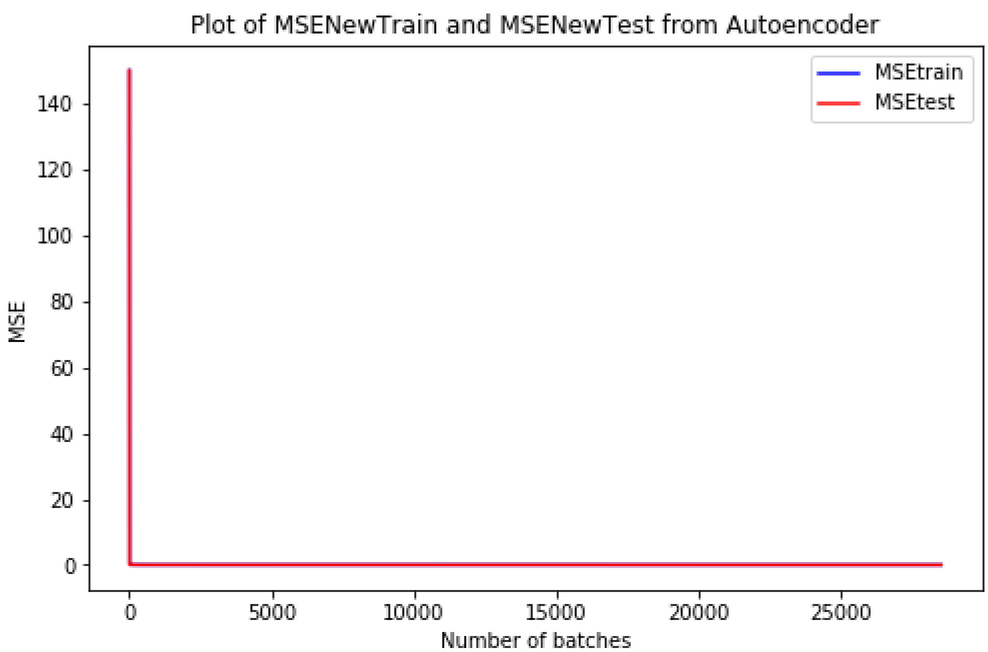
Time to run model

In [114]:
```
print('Time to run the model in mins is',round((end - start)/60, 3))
```

```
Time to run the model in mins is 11.133
```

In [101]:
```
#plot MSE
labels=['MSEtrain','MSEtest']
%matplotlib inline
plt.figure(figsize=(8, 5))
plt.figure(1)
plt.plot(mlpMyMonitorff.MSEtrain, label='MSENewTrain', color='b')
plt.plot(mlpMyMonitorff.MSEtest, label='MSENewTest', color='r')
plt.legend(labels)
plt.ylabel('MSE')
plt.xlabel('Number of batches')
plt.title('Plot of MSENewTrain and MSENewTest from Autoencoder')
```

Out[101]:
```
Text(0.5, 1.0, 'Plot of MSENewTrain and MSENewTest from Autoencoder')
```

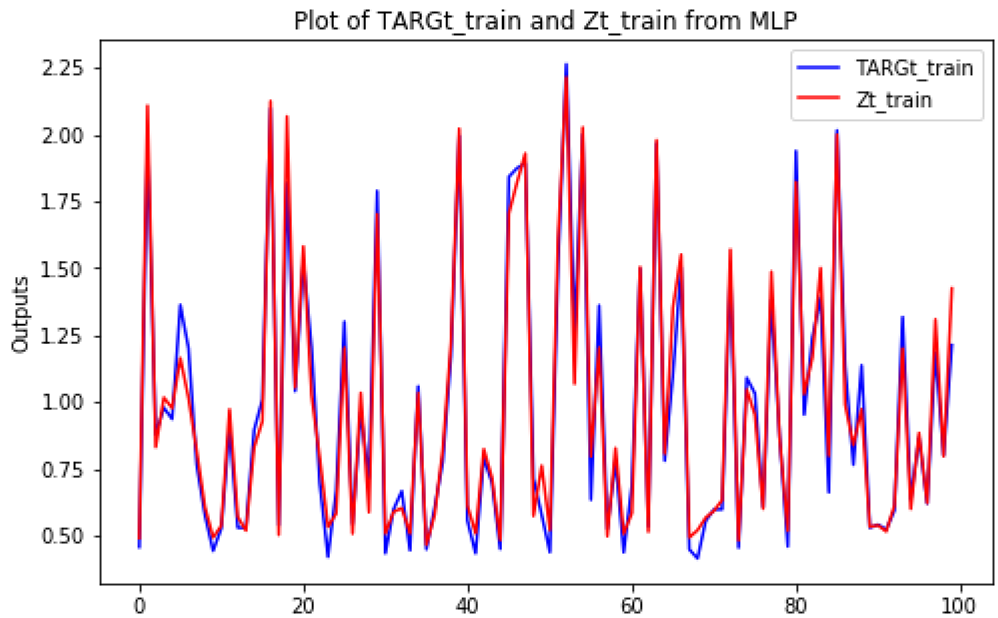


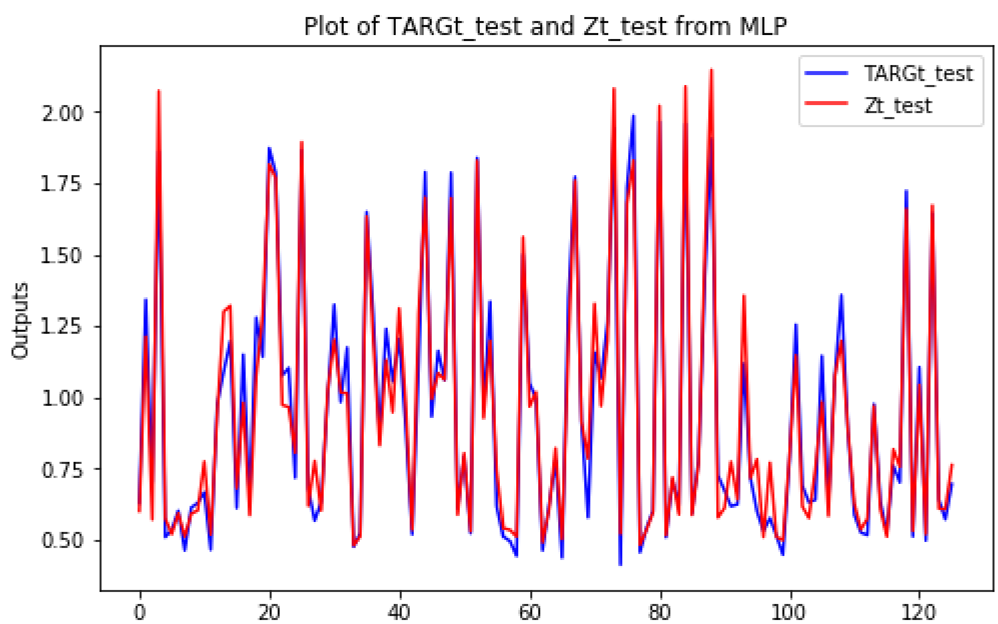
In [105]:
```
z_trainff=mlpff.predict(Htrain)
MREPtrainff=np.average(abs(z_trainff-y_train)/y_train)
z_testff=mlpff.predict(Htest)
MREPtestff=np.average(abs(z_testff-y_test)/y_test)
```

In [106]:
```
#plot TARGt vs Zt
labels1=['TARGt_train','Zt_train']
labels2=['TARGt_test','Zt_test']

%matplotlib inline
plt.figure(2)
plt.figure(figsize=(8, 5))
plt.plot(range(100), y_train[:100], color='b')
plt.plot(range(100),z_trainff[:100], color='r')
plt.legend(labels1)
plt.ylabel('Outputs')
plt.title('Plot of TARGt_train and Zt_train from MLP')


plt.figure(3)
plt.figure(figsize=(8, 5))
plt.plot(y_test, color='b')
plt.plot(z_testff, color='r')
plt.legend(labels2)
plt.ylabel('Outputs')
plt.title('Plot of TARGt_test and Zt_test from MLP')
```

Out[106]: Text(0.5, 1.0, 'Plot of TARGt_test and Zt_test from MLP')

<Figure size 432x288 with 0 Axes>





In [108]:
```
MREPtrainff=np.average(abs(z_trainff-y_train)/y_train)
MREPtestff=np.average(abs(z_testff-y_test)/y_test)
print('MREPtrain:',round(MREPtrainff,3))
print('MREPtest:',round(MREPtestff,3))
```

```
MREPtrain: 0.079
MREPtest: 0.081
```

In [ ]: