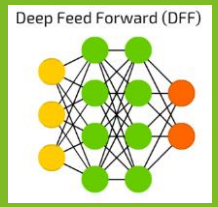
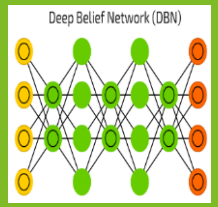
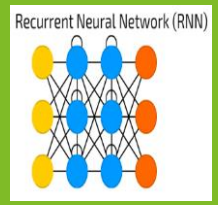
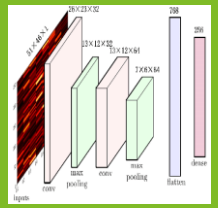
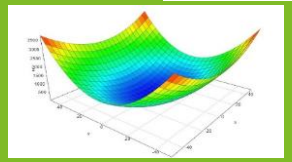
Deep Learning CSI_7_DEL

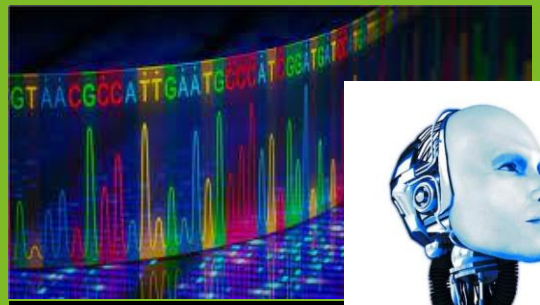


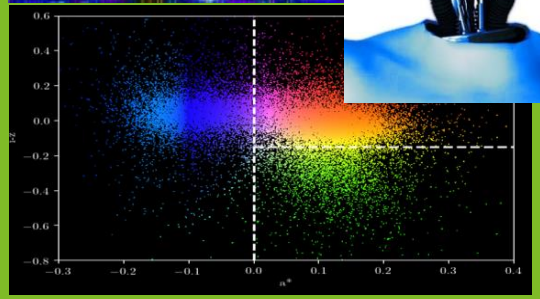


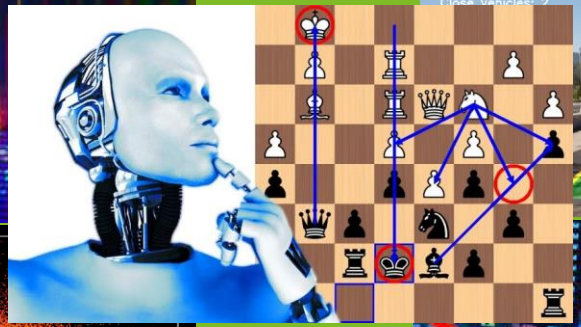


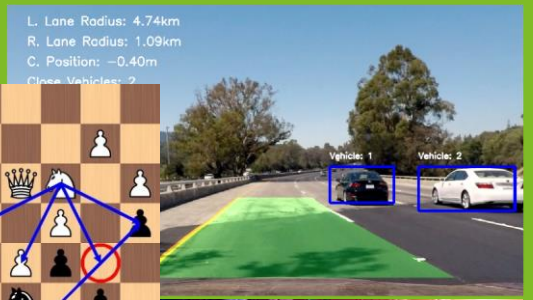












Tutorial 10:Deep Belief Networks

London South Bank University
EST 1892

Import the libraries

```
import numpy as np
import matplotlib.pyplot as plt
#Logistic layer from sklearn
from sklearn.neural_network import MLPClassifier
#MNIST dataset
from sklearn.datasets import load_digits
#Pareto's principle library
from sklearn.model_selection import train_test_split
#RBM model
from sklearn.neural_network import BernoulliRBM
#This is used to clone same model multiple times
from sklearn.base import clone
#Pipeline for linking RBM layers and the logistic layer
from sklearn.pipeline import Pipeline
#For classification accuracy report
from sklearn.metrics import classification_report
```

Load MNIST dataset

```
# load MNIST dataset
digits= load_digits()
#Separate data from labels
X, Y = digits.data, digits.target
#Normalise the data
X = (X / 16).astype(np.float32)
```

Check the dataset shape and the first row

```
[4]  X.shape
```

```
(1797, 64)
```

```
[7]  X[0]
```

```
array([0.    , 0.    , 0.3125, 0.8125, 0.5625, 0.0625, 0.    , 0.    ,
       0.    , 0.    , 0.8125, 0.9375, 0.625 , 0.9375, 0.3125, 0.    ,
       0.    , 0.1875, 0.9375, 0.125 , 0.    , 0.6875, 0.5   , 0.    ,
       0.    , 0.25  , 0.75  , 0.    , 0.    , 0.5   , 0.5   , 0.    ,
       0.    , 0.3125, 0.5   , 0.    , 0.    , 0.5625, 0.5   , 0.    ,
       0.    , 0.25  , 0.6875, 0.    , 0.0625, 0.75  , 0.4375, 0.    ,
       0.    , 0.125 , 0.875 , 0.3125, 0.625 , 0.75  , 0.    , 0.    ,
       0.    , 0.    , 0.375 , 0.8125, 0.625 , 0.    , 0.    , 0.    ],
      dtype=float32)
```

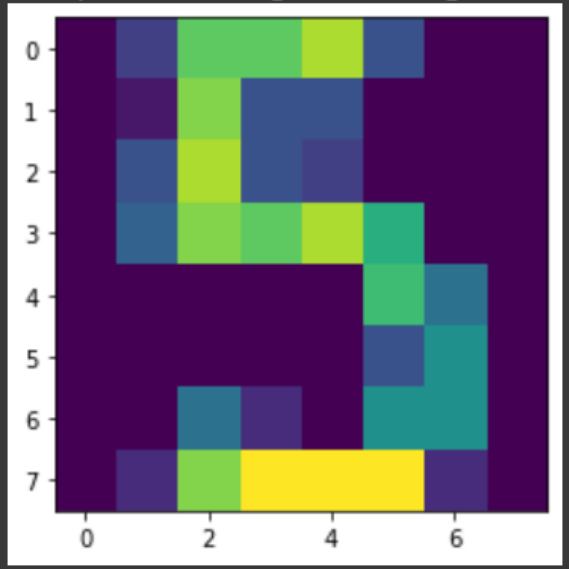
Apply 80/20 Pareto's principle

```
# split into train, validation, and test data sets
X_train, X_test, Y_train, Y_test = train_test_split(X,       Y,       test_size=0.2, random_state=0)
```

Visualise the second training input row

```
plt.imshow(X_train[1].reshape(8,8))
```

```
<matplotlib.image.AxesImage at 0x7f1cd0bbaaf0>
```



Set DBN parameters

```
# set hyperparameters
#RBM learning rate
learning_rate = 0.4
#RBM hidden neurons
hidden_neurons   =  256
#Gibbs sampling steps
total_epochs  =   50
#Batch size
batch_size    =  32
```

Create DBN model

```
# construct models
# RBM
rbm = BernoulliRBM(n_components=hidden_neurons, learning_rate=learning_rate, batch_size=batch_size, n_iter=total_epochs, verbose=1, random_state=0)
# Output Layer
neural_net=MLPClassifier(activation='logistic',hidden_layer_sizes=(256,10),learning_rate_init=0.008,max_iter=150)
# merge layers using pipeline: rbm1 > rbm2 > logistic
dbn=Pipeline(steps=[('rbm1', clone(rbm)), ('rbm2', clone(rbm)),('logistic layer', neural_net)])  # RBM stack / DBN
```

Start training you DBN model

```
# train
dbn.fit(X_train, Y_train)
```

```
[BernoulliRBM] Iteration 3, pseudo-likelihood = -21.96, time = 0.25s
[BernoulliRBM] Iteration 4, pseudo-likelihood = -22.45, time = 0.22s
[BernoulliRBM] Iteration 5, pseudo-likelihood = -21.95, time = 0.20s
[BernoulliRBM] Iteration 6, pseudo-likelihood = -22.48, time = 0.20s
[BernoulliRBM] Iteration 7, pseudo-likelihood = -22.05, time = 0.17s
[BernoulliRBM] Iteration 8, pseudo-likelihood = -21.82, time = 0.21s
[BernoulliRBM] Iteration 9, pseudo-likelihood = -22.08, time = 0.18s
[BernoulliRBM] Iteration 10, pseudo-likelihood = -22.41, time = 0.16s
[BernoulliRBM] Iteration 11, pseudo-likelihood = -21.09, time = 0.17s
[BernoulliRBM] Iteration 12, pseudo-likelihood = -20.83, time = 0.16s
[BernoulliRBM] Iteration 13, pseudo-likelihood = -21.08, time = 0.20s
[BernoulliRBM] Iteration 14, pseudo-likelihood = -21.15, time = 0.20s
[BernoulliRBM] Iteration 15, pseudo-likelihood = -20.62, time = 0.18s
[BernoulliRBM] Iteration 16, pseudo-likelihood = -20.29, time = 0.20s
[BernoulliRBM] Iteration 17, pseudo-likelihood = -20.09, time = 0.20s
[BernoulliRBM] Iteration 18, pseudo-likelihood = -20.75, time = 0.21s
[BernoulliRBM] Iteration 19, pseudo-likelihood = -20.39, time = 0.23s
[BernoulliRBM] Iteration 20, pseudo-likelihood = -20.32, time = 0.18s
[BernoulliRBM] Iteration 21, pseudo-likelihood = -19.89, time = 0.19s
```

Evaluate your model

```
# evaluate using validation set
print("Model performance:\n%s\n" % (classification_report(Y_test, dbn.predict(X_test))))
```

```
Model performance:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        27
           1       0.89      0.91      0.90        35
           2       0.94      0.89      0.91        36
           3       0.84      0.90      0.87        29
           4       0.97      0.97      0.97        30
           5       0.95      0.97      0.96        40
           6       1.00      1.00      1.00        44
           7       0.95      0.97      0.96        39
           8       0.89      0.79      0.84        39
           9       0.86      0.88      0.87        41

    accuracy                           0.93       360
   macro avg       0.93      0.93      0.93       360
weighted avg       0.93      0.93      0.93       360
```

Try plotting RBM 1 components

```
plt.figure(figsize=(15, 6))
for i, comp in enumerate(dbn['rbm1'].components_):
    plt.subplot(10, 26, i + 1)
    cur_cmp=(comp*16).astype(np.float32)
    plt.imshow(cur_cmp.reshape((8, 8)), cmap=plt.cm.gray_r,interpolation="nearest")
    plt.xticks(())
    plt.yticks(())
plt.suptitle("256 components extracted by RBM 1", fontsize=16)
plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)

plt.show()
```

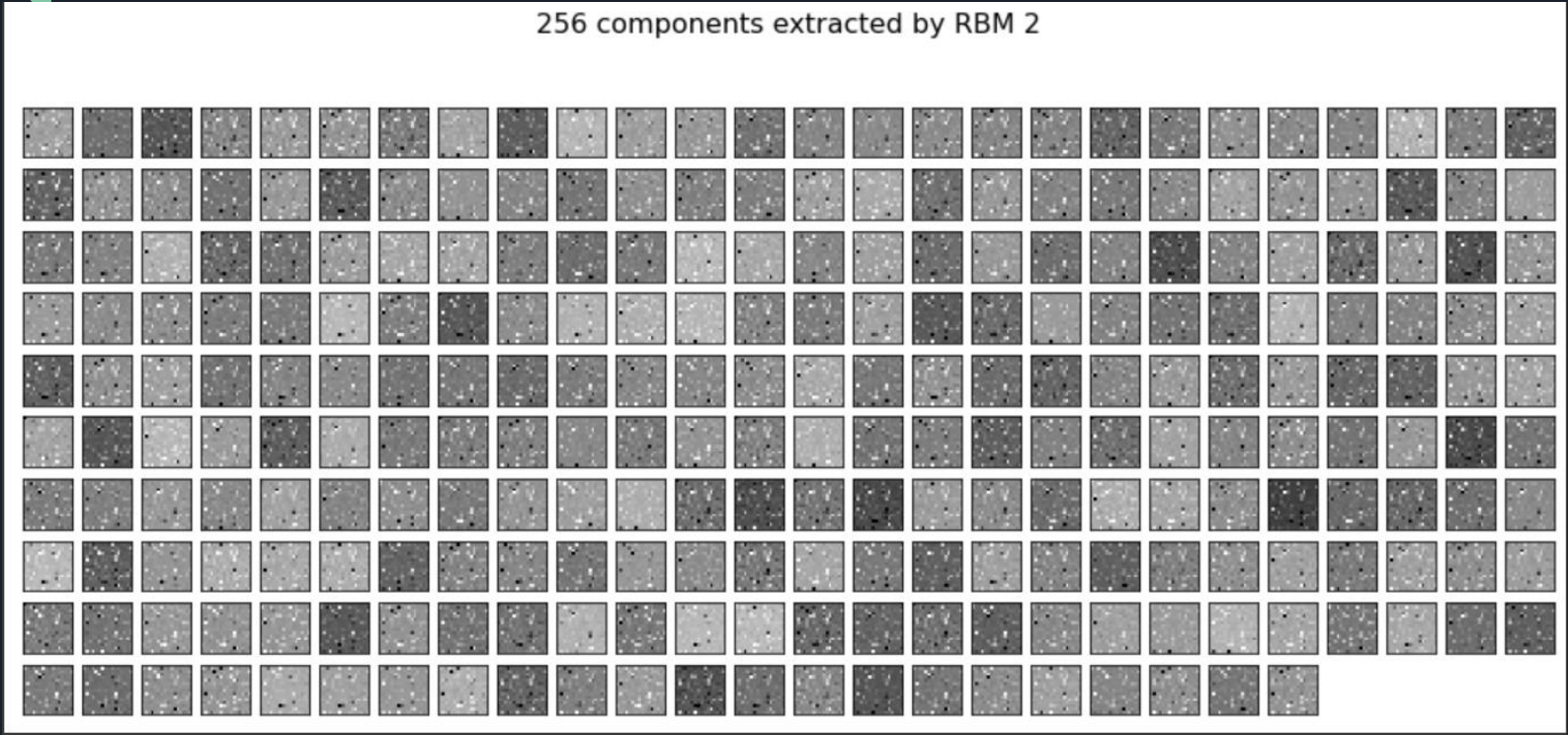
You should get an input like this

Plot RBM 2 components

```
plt.figure(figsize=(15, 6))
for i, comp in enumerate(dbn['rbm2'].components_):
    plt.subplot(10, 26, i + 1)
    plt.imshow(comp.reshape((16, 16)), cmap=plt.cm.gray_r, interpolation="nearest")
    plt.xticks(())
    plt.yticks(())
plt.suptitle("256 components extracted by RBM 2", fontsize=16)
plt.subplots_adjust(0.08, 0.02, 0.92, 0.85, 0.08, 0.23)

plt.show()
```

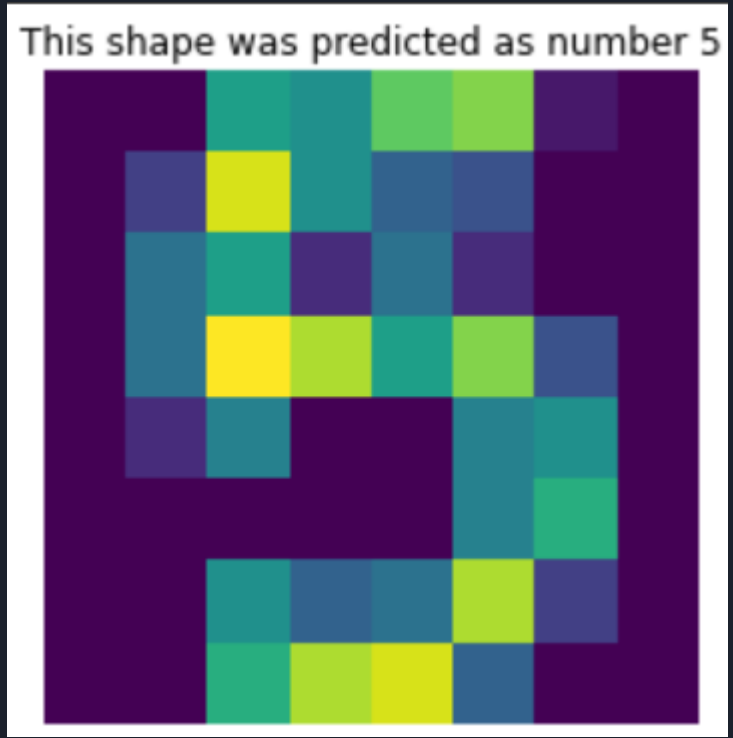
You should see something like this



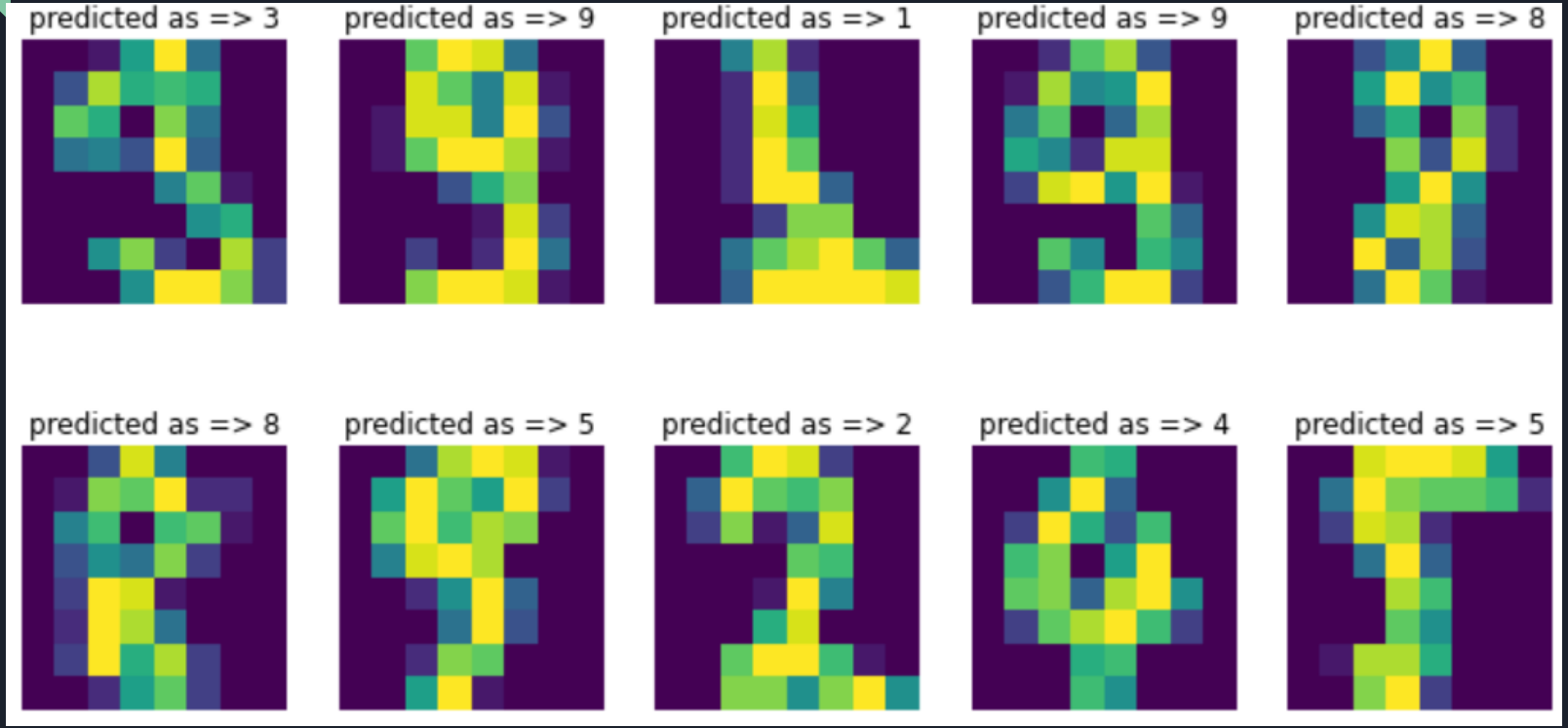
Predict single random digit with DBN

```
import random
i=random.randint(0, 359)
plt.imshow(X_test[i].reshape(8,8))
plt.axis("off")
plt.title(f"This shape was predicted as number {dbn.predict(X_test[i].reshape(1,-1))[0]}")
```

Output sample



Your turn: Try to modify the previous code and make 10 classifications. Your output should be like this



Tasks:

- Add an extra RBM to your model.
- Try decreasing the number of iterations for RBM. Do you see any improvements?
- Change the learning rate for Logistics layer to (0.001, 0.005, 0.01, 0.1) and retrain your model. What are the classification accuracy results?
- Work on your coursework for the rest of this tutorial.