Task 1: The Challenge from Moriarty (50 points)

It is 01/01/2049. At around 11pm, an anonymous parcel arrived at 221B Baker Street addressed to Sherlock. The parcel contained a USB, which had a bunch of grayscale images depicting types of clothing such as shirts, pants, etc. Along with the images, there was an audio message. Dr. Watson played it and in the voice of Professor Moriarty, said "Hello Sherlock, it has been a while since we played a game. I have sent you a bunch of images, if you somehow visualize this in two dimensions, you might see some clusters, and the x-y coordinates of each of the 10000 points represent 10000 houses in London, whose wardrobe is going to burn in the next 100 days, taking down 100 houses a day. However smart you are or how much ever big your mind palace is, I am pretty sure that you cannot visualize these 28x28 images in two dimensions directly. My game starts tomorrow morning, which gives you about 12 hours to stop me. Good Luck!" Sherlock does not know how to solve this puzzle. Dr. Watson thinks getting a lower-dimensional projection of images might be possible using neural networks and unsupervised learning. Therefore, he calls you and asks you to help, and since you are a fan of Sherlock, you decide to solve the whole problem. You must think of training a network without using any labels that gives you a lower-dimensional representation of the images after which you could easily use tSNE from sklearn to bring the dimension down to two so that Sherlock can then take over. Your final output to Sherlock must be a 2D projection of the 10000 images. The clock is ticking! Sherlock is waiting! Oh, by the way, Sherlock prefers PyTorch. Images: https://github.com/zalandoresearch/fashion-mnist

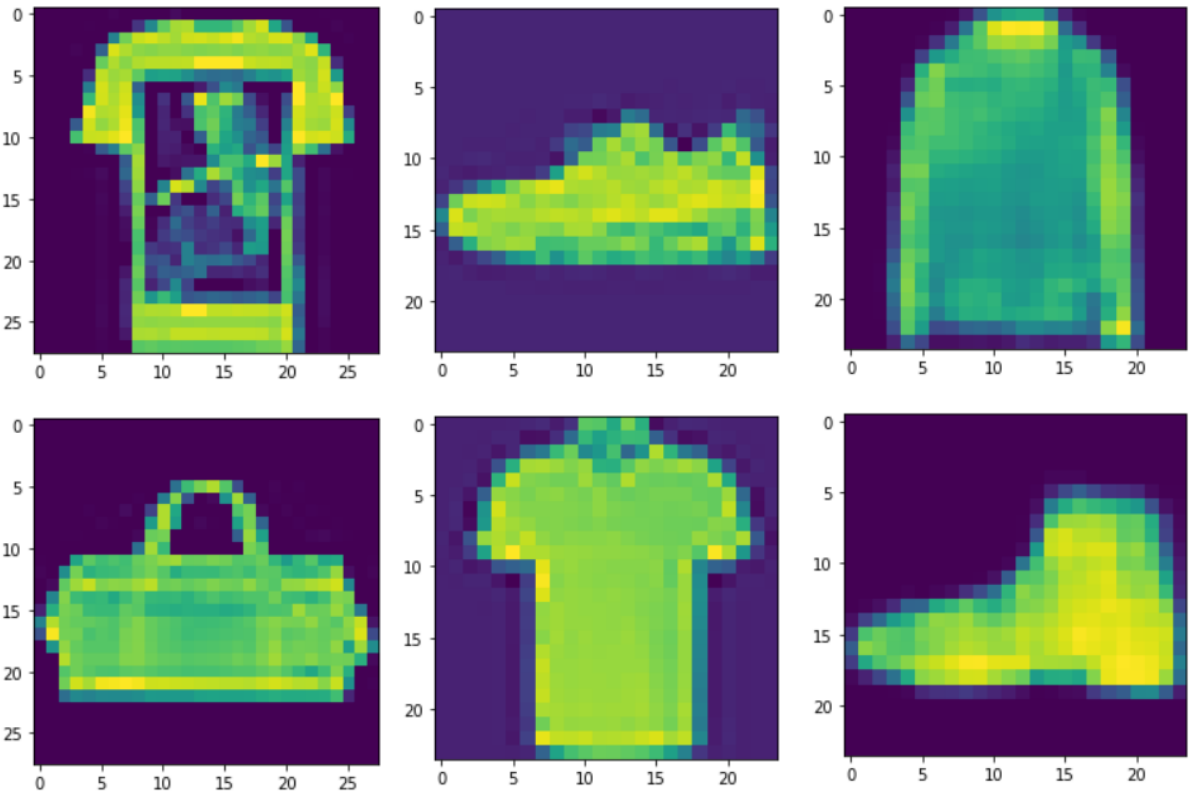
Solution:

For Task 1, the GitHub Repository mentioned above was used to load and read data from the data. The data was stored in form of .gz file and it was loaded by the load_mnist () function and read by read_mnist () function. From there on the data was visualized in form of Pandas data frame and it was found that the data was numbers between 0 and 255. Data processing was done followed by encompassing the data by dividing it by 255. Once the data was prepared, two processes were used in order to reduce the data down to certain dimensions, so that TSNE from sklearn can handle it well, by reducing the data further into only two dimensions. Two kinds of data reduction processes were used and tested. They are as follows:

Method 1: Image based dimension reduction

As we know, that the image in 784 dimension can be reshaped down to 28x28 and that forms the dimension of image. Plotting it gives back images from the MNIST dataset that is how the data can be visualized. In order to reduce the dimension from 784, the 28x28 dimensioned images are reduced to 24x24 by the PIL library. In this process, data is lost and noise is introduced. Features are lost in this process. A convoluted neural network is prepared with Input layer, 2D convolutional layer, Max pooling layer and 2D up sampling layer, with input and output dimension of 24x24x1. Then the reduced images are trained with itself, the training data due to absence of labels, and tested on the test data. This testing tells us how much the original features are retained after bringing down the

dimension to 24x24=576. The losses were reducing in nature, but the accuracies were too low. These were the images before and after training:

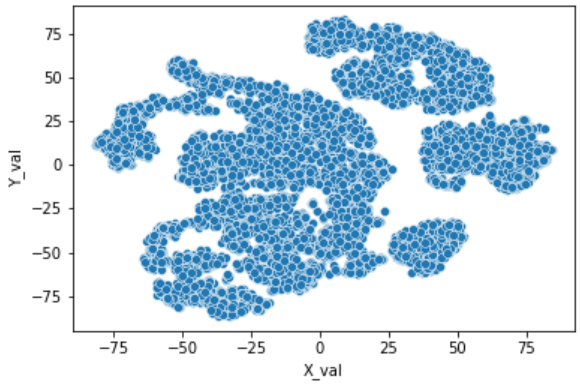


Images in 28x28 dimension **Resized image to lower dimension** **Prediction on test image**

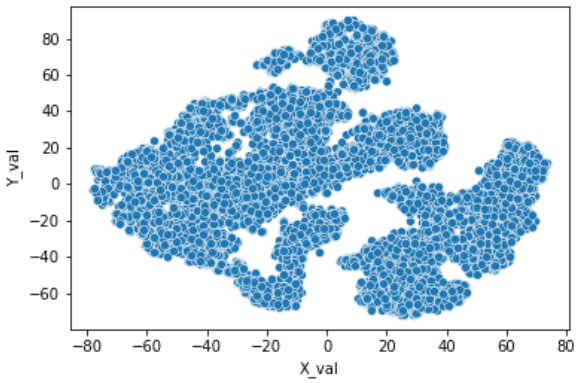
The Model's summary is as follows:

```
Model: "functional_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 24, 24, 1)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 24, 24, 16)        160
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 12, 12, 16)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 12, 12, 8)         1160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 8)           0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 6, 6, 8)           584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 3, 3, 8)           0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 3, 3, 8)           584
_________________________________________________________________
up_sampling2d (UpSampling2D) (None, 6, 6, 8)           0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 6, 6, 8)           584
_________________________________________________________________
up_sampling2d_1 (UpSampling2 (None, 12, 12, 8)         0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 16)        1168
_________________________________________________________________
up_sampling2d_2 (UpSampling2 (None, 24, 24, 16)        0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 24, 24, 1)         145
=================================================================
Total params: 4,385
Trainable params: 4,385
Non-trainable params: 0
_________________________________________________________________
```

From here on, the encoded images were again reshaped to the dimension of 576 and then it was fed to TSNE network. From there on, the data was converted to 10000x2 dimension and it was plotted. In addition, the original test dataset of shape 10000x784 was also reduced through TSNE and plotted.



Reduced Image based on CNN network



Reduced image just by TSNE

Method 2: Feature based dimension reduction:

For feature based dimension reduction, an auto-encoder is formed which decreases the dimension of 784 to 256. In this case, if we plot the encoded data, it does not reproduce any image. However, it protects the features more in this case. We get an idea about this, when we train and test the dataset and get to see that the accuracy is high. These are some of the examples:

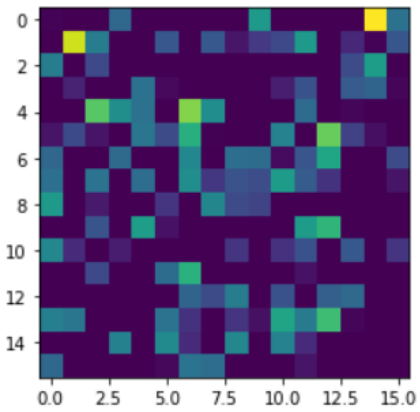


Image 1 after encoding

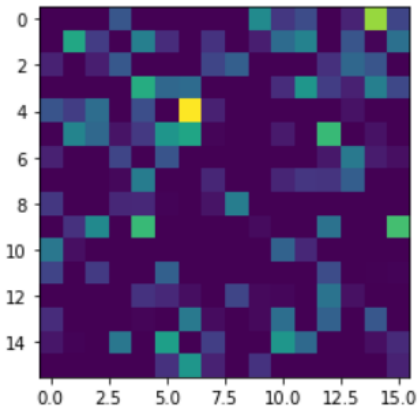
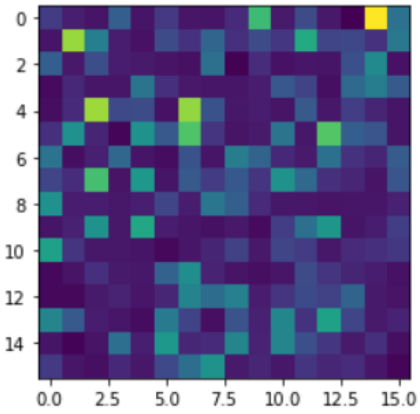
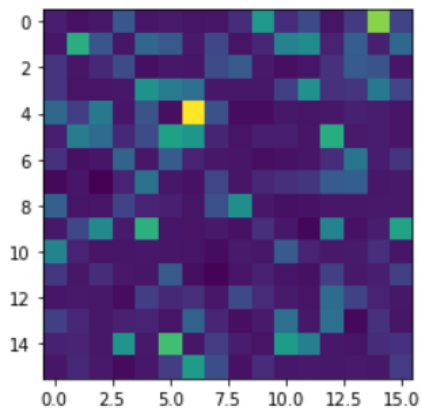


Image 2 after encoding



Predicted image 1 on encoded test data



Predicted image 2 on encoded test data

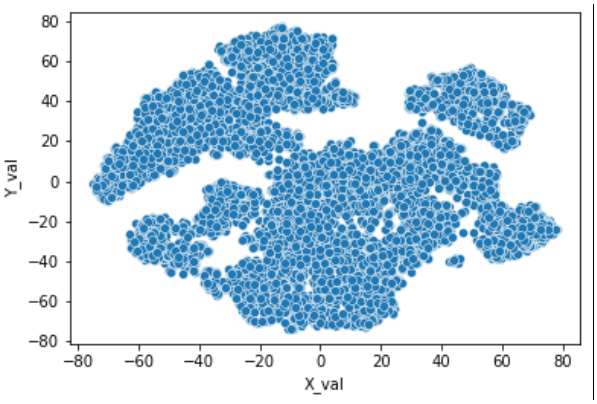
Only Flatten, Dense and Sequential Layers were used. We see a very good prediction for this unsupervised learning although. The model is as follows:

```
Model: "functional_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 256)]             0
_________________________________________________________________
sequential (Sequential)      (None, 16)                4112
_________________________________________________________________
sequential_1 (Sequential)    (None, 256)               4352
=================================================================
Total params: 8,464
Trainable params: 8,464
Non-trainable params: 0
_________________________________________________________________
None
```

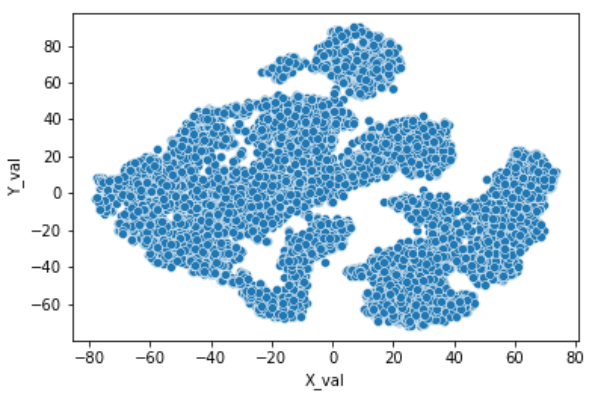
Then, it is checked how accurate the encoded dimensions are, with respect to the original dimension. The following model is used in order to train the original dimensioned dataset.

```
Model: "functional_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_14 (InputLayer)        [(None, 784)]             0
_________________________________________________________________
sequential_6 (Sequential)    (None, 256)               200960
_________________________________________________________________
sequential_7 (Sequential)    (None, 784)               201488
=================================================================
Total params: 402,448
Trainable params: 402,448
Non-trainable params: 0
_________________________________________________________________
None
```

After this, the encoded dataset is fed to TSNE in order to bring it down to two dimensions. Then the 2D array is plotted.



Reduced Image based on Auto encoder network

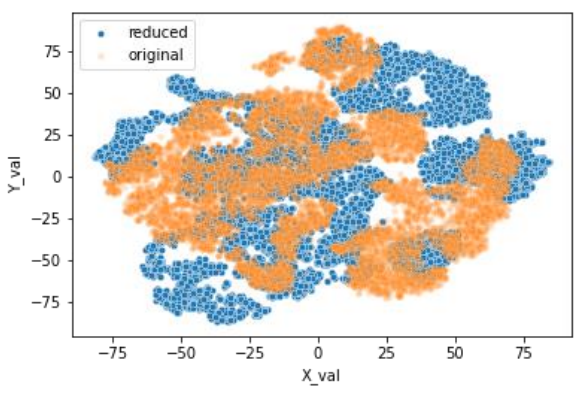


Reduced image just by TSNE

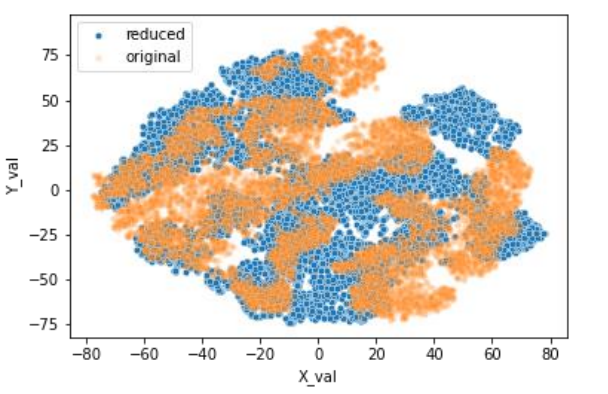
Two plots are plotted with overlapping maps:

1. CNN, TSNE based data reduction, and only TSNE based dimension reduction.
2. Auto encoder, TSNE based, and only TSNE based dimension reduction.

The plots are as follows:



CNN and TSNE based Dimension Reduction



Auto encoder and TSNE based Dimension Reduction

Task 2: The Devil Is in The Detail (50 points)

You successfully solved the first problem and rescued all the wardrobes. Five days later, Sherlock receives an email from Moriarty, which says, “Hello Sherlock, the first game was just a warm-up. This time, I assure you it will be more interesting. I have sent you a bunch of satellite images and I have encoded a 5-letter secret code in them, one alphabet in each of the randomly chosen 5 images. Since you are not that smart to solve it, I have also sent you five images, which are similar to the five images containing the code. I have full confidence that you will fail this time. If you are still wondering what the code might be, that code will give you away my next move in ending you and your little doctor friend. Good Luck!” Since you helped solve the first problem, Sherlock trusts you with this too. You need to search the five images containing code with the help of the given query images. Second time is the charm. Do not let Sherlock down. You can download the database & the five query images at here. Submit the work done (Notebook) to search for these images and the final five images that you think contain the code. Do you know what the code is now?

Note: For all the two tasks, feel free to use any APIs that you see fit. You should document your solutions (and the APIs you used) in detail but concisely as a PDF, which will be used for grading. Your code will be used as an evidence to support your solution document.

Solution:

In this notebook, the SIFT library is used to detect and match features of two sets of images. The dataset contains two sets of images, as per the folders 'database' and 'queries'. These sets of images both relate to satellite images. By visualizing the 'database' dataset, it can be said that a larger satellite picture has been used and smaller overlapping blocks of the image has been cropped out. Before cropping out, alphabets were printed on them. When going through the dataset, instead of finding one single image for one alphabet, ending up finding a continuous series of some images where the alphabet slowly appears and then disappears, travelling horizontally in the real larger satellite image. Also at an offset of 260 images from the blocks where the alphabet is found, the alphabet itself is seen to shift vertically. Therefore, every image represents an overlapping patch from a bigger image. In this Task, it is to be found out that which image from the 'database' actually matches the most with the images from the database. Around that image patch, the alphabet should be present. One