Stacked autoencoder

This is a tactic to producd a low dimensional visulalization of a very high dimension set, in this case the MNIST fashion data set

We will reduce the dimension using an autoencoder and then further reduce it using t-SNE

"Hands on Machine Learning with Scikit-Learn, Keras and Tensorflow, 2nd Edition", A. Geron, O'Reilly

Checked on 1/18/2023

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import tensorflow as tf

from sklearn.metrics import accuracy_score, precision_score, recall_score
from sklearn.model_selection import train_test_split
from tensorflow.keras import layers, losses
from tensorflow.keras.datasets import fashion_mnist
from tensorflow.keras.models import Model
```

In [3]:
```
fashion_mnist = fashion_mnist
(X_train_full, y_train_full), (X_test, y_test) = fashion_mnist.load_data()
```

In [4]:
```
X_train_full.shape
```

Out[4]:
```
(60000, 28, 28)
```

Be sure to always scale or standardize data when working with autoencoders, they always seem to fail if you don't

In [5]:
```
X_train_full=X_train_full/255
X_test=X_test/255
```

Okay, here are the encoder and decoder pair

The encoder is going from 784 variables down to 30, that's quite a reduction

In [6]:
```
stacked_encoder = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=[28, 28]),
    tf.keras.layers.Dense(100, activation="selu"),
    tf.keras.layers.Dense(30, activation="selu"),
])
stacked_decoder = tf.keras.models.Sequential([
    tf.keras.layers.Dense(100, activation="selu", input_shape=[30]),
    tf.keras.layers.Dense(28 * 28, activation="sigmoid"),
    tf.keras.layers.Reshape([28, 28])
])
```

```
stacked_ae = tf.keras.models.Sequential([stacked_encoder, stacked_decoder])

stacked_ae.compile(loss="binary_crossentropy",
                   optimizer=tf.keras.optimizers.SGD(lr=1.5))

history = stacked_ae.fit(X_train_full, X_train_full, epochs=20,
                         validation_data=(X_test, X_test))
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use
the legacy optimizer, e.g.,tf.keras.optimizers.legacy.SGD.
```

```
Epoch 1/20
1875/1875 [==============================] - 13s 6ms/step - loss: 0.6625 - val_loss:
0.6004
Epoch 2/20
1875/1875 [==============================] - 10s 6ms/step - loss: 0.5367 - val_loss:
0.5041
Epoch 3/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.4954 - val_loss:
0.4896
Epoch 4/20
1875/1875 [==============================] - 12s 7ms/step - loss: 0.4816 - val_loss:
0.4737
Epoch 5/20
1875/1875 [==============================] - 12s 6ms/step - loss: 0.4612 - val_loss:
0.4488
Epoch 6/20
1875/1875 [==============================] - 12s 6ms/step - loss: 0.4339 - val_loss:
0.4212
Epoch 7/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.4087 - val_loss:
0.3998
Epoch 8/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3914 - val_loss:
0.3869
Epoch 9/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3816 - val_loss:
0.3798
Epoch 10/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3760 - val_loss:
0.3755
Epoch 11/20
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3724 - val_loss:
0.3724
Epoch 12/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3696 - val_loss:
0.3698
Epoch 13/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3671 - val_loss:
0.3675
Epoch 14/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3648 - val_loss:
0.3652
Epoch 15/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3625 - val_loss:
0.3629
Epoch 16/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3602 - val_loss:
0.3606
Epoch 17/20
1875/1875 [==============================] - 10s 5ms/step - loss: 0.3579 - val_loss:
0.3583
Epoch 18/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3555 - val_loss:
0.3559
Epoch 19/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3532 - val_loss:
0.3537
Epoch 20/20
1875/1875 [==============================] - 11s 6ms/step - loss: 0.3510 - val_loss:
0.3515
```

In [ ]:
```
Now we can plot some images and some reconstructions of the images using the encoder-c
```

In [ ]:
```
def plot_image(image):
    plt.imshow(image, cmap="binary")
    plt.axis("off")

def show_reconstructions(model, n_images=5):
    reconstructions = model.predict(X_test[:n_images])
    fig = plt.figure(figsize=(n_images * 1.5, 3))
    for image_index in range(n_images):
        plt.subplot(2, n_images, 1 + image_index)
        plot_image(X_test[image_index])
        plt.subplot(2, n_images, 1 + n_images + image_index)
        plot_image(reconstructions[image_index])

show_reconstructions(stacked_ae)
```

In [7]:
```
from sklearn.manifold import TSNE

X_test_compressed = stacked_encoder.predict(X_test)
tsne = TSNE()
X_valid_2D = tsne.fit_transform(X_test_compressed)
```

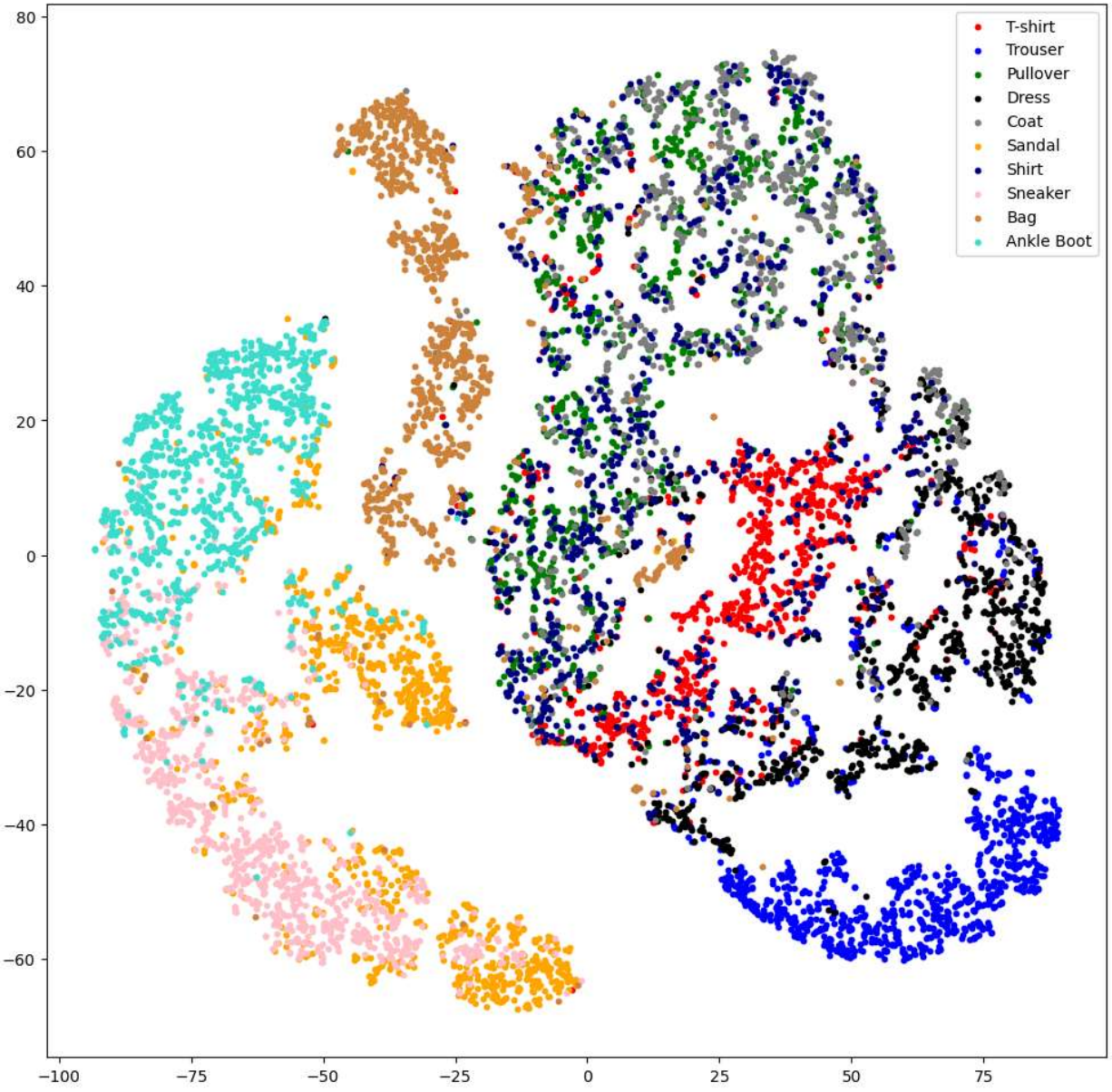
```
313/313 [==============================] - 1s 2ms/step
```

In [8]:
```
plt.figure(figsize=(12,12))
plt.scatter(X_valid_2D[y_test==0, 0], X_valid_2D[y_test==0, 1], c='r', s=10, cmap="tab
plt.scatter(X_valid_2D[y_test==1, 0], X_valid_2D[y_test==1, 1], c='b', s=10, cmap="tab
plt.scatter(X_valid_2D[y_test==2, 0], X_valid_2D[y_test==2, 1], c='g', s=10, cmap="tab
plt.scatter(X_valid_2D[y_test==3, 0], X_valid_2D[y_test==3, 1], c='k', s=10, cmap="tab
plt.scatter(X_valid_2D[y_test==4, 0], X_valid_2D[y_test==4, 1], c='grey', s=10, cmap='
plt.scatter(X_valid_2D[y_test==5, 0], X_valid_2D[y_test==5, 1], c='orange', s=10, cmap
plt.scatter(X_valid_2D[y_test==6, 0], X_valid_2D[y_test==6, 1], c='navy', s=10, cmap='
plt.scatter(X_valid_2D[y_test==7, 0], X_valid_2D[y_test==7, 1], c='pink', s=10, cmap='
plt.scatter(X_valid_2D[y_test==8, 0], X_valid_2D[y_test==8, 1], c='peru', s=10, cmap='
plt.scatter(X_valid_2D[y_test==9, 0], X_valid_2D[y_test==9, 1], c='turquoise', s=10, c


plt.legend()
```

```
<ipython-input-8-3f5ee9261eaf>:2: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==0, 0], X_valid_2D[y_test==0, 1], c='r', s=10, cmap
="tab10",label="T-shirt")
<ipython-input-8-3f5ee9261eaf>:3: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==1, 0], X_valid_2D[y_test==1, 1], c='b', s=10, cmap
="tab10",label="Trouser")
<ipython-input-8-3f5ee9261eaf>:4: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==2, 0], X_valid_2D[y_test==2, 1], c='g', s=10, cmap
="tab10",label="Pullover")
<ipython-input-8-3f5ee9261eaf>:5: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==3, 0], X_valid_2D[y_test==3, 1], c='k', s=10, cmap
="tab10",label="Dress")
<ipython-input-8-3f5ee9261eaf>:6: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==4, 0], X_valid_2D[y_test==4, 1], c='grey', s=10, cma
p="tab10",label="Coat")
<ipython-input-8-3f5ee9261eaf>:7: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==5, 0], X_valid_2D[y_test==5, 1], c='orange', s=10, c
map="tab10",label="Sandal")
<ipython-input-8-3f5ee9261eaf>:8: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==6, 0], X_valid_2D[y_test==6, 1], c='navy', s=10, cma
p="tab10",label="Shirt")
<ipython-input-8-3f5ee9261eaf>:9: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==7, 0], X_valid_2D[y_test==7, 1], c='pink', s=10, cma
p="tab10",label="Sneaker")
<ipython-input-8-3f5ee9261eaf>:10: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==8, 0], X_valid_2D[y_test==8, 1], c='peru', s=10, cma
p="tab10",label="Bag")
<ipython-input-8-3f5ee9261eaf>:11: UserWarning: No data for colormapping provided via
'c'. Parameters 'cmap' will be ignored
  plt.scatter(X_valid_2D[y_test==9, 0], X_valid_2D[y_test==9, 1], c='turquoise', s=1
0, cmap="tab10",label="Ankle Boot")
```

Out[8]: `<matplotlib.legend.Legend at 0x7ea6b9ee4b80>`


T-shirt
Trouser
Pullover
Dress
Coat
Sandal
Shirt
Sneaker
Bag
Ankle Boot
80
60
40
20
0
−20
−40
−60
−100
−75
−50
−25
0
25
50
75

In [ ]: