ICP -9

VVS Murthy Kolla
700729142

Summary of question:

1. Add one more hidden layer to autoencoder
2. Do the prediction on the test data and then visualize one of the reconstructed version of that test data. Also, visualize the same test data before reconstruction using Matplotlib
3. Repeat the question 2 on the denoisening autoencoder
4. plot loss and accuracy using the history object

Github link: https://github.com/murthykolla/ICP-9.git
Video link
https://drive.google.com/file/d/1gaEOfq_giU4BsgGoIsTxjXrcuZhW5zPS/view?usp=share_link

ICP - 9

VVS MURTHY KOLLA

700729142

Autoencoders

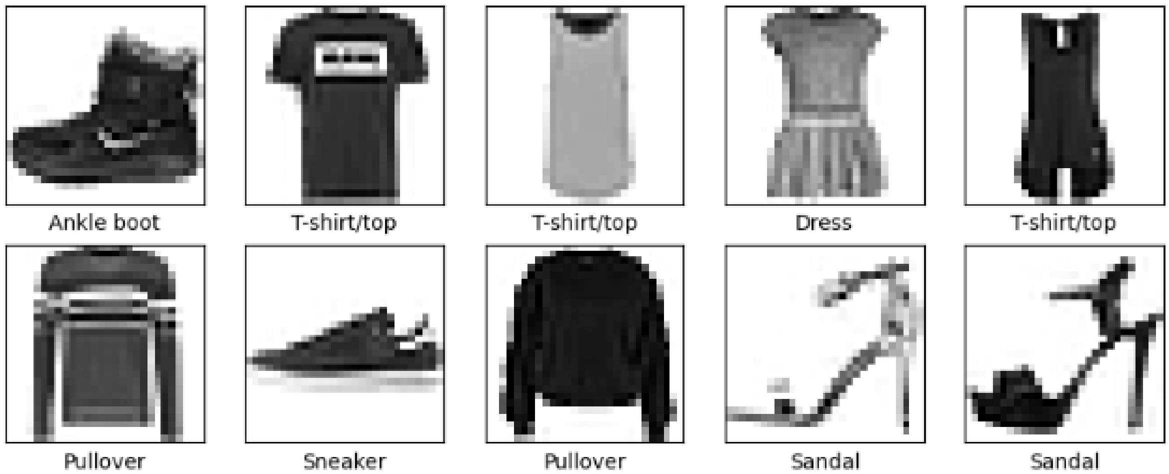
```
from keras.layers import Input, Dense
from keras.models import Model
from keras.datasets import fashion_mnist
import numpy as np

# Load Fashion MNIST dataset
(x_train, _), (x_test, _) = fashion_mnist.load_data()

# Normalize pixel values between 0 and 1
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.

# Reshape images to vectors
x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

# Define autoencoder model
input_img = Input(shape=(784,))
encoded = Dense(256, activation='relu')(input_img)
encoded = Dense(128, activation='relu')(encoded)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(128, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(encoded)
decoded = Dense(256, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
autoencoder = Model(input_img, decoded)

# Compile model
autoencoder.compile(optimizer='adam', loss='binary_crossentropy',metrics=['accuracy'])

# Train model
history = autoencoder.fit(x_train, x_train,
                epochs=50,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

# Evaluate model
```

```
235/235 [==============================] - 1s 5ms/step - loss: 0.2683 - accuracy: 0.0414 - val_loss: 0.2707 - val_accuracy: 0.0403
Epoch 39/50
235/235 [==============================] - 1s 6ms/step - loss: 0.2682 - accuracy: 0.0417 - val_loss: 0.2707 - val_accuracy: 0.0399
Epoch 40/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2680 - accuracy: 0.0426 - val_loss: 0.2706 - val_accuracy: 0.0421
Epoch 41/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2678 - accuracy: 0.0427 - val_loss: 0.2707 - val_accuracy: 0.0413
Epoch 42/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2676 - accuracy: 0.0432 - val_loss: 0.2701 - val_accuracy: 0.0405
Epoch 43/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2674 - accuracy: 0.0424 - val_loss: 0.2699 - val_accuracy: 0.0406
Epoch 44/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2673 - accuracy: 0.0439 - val_loss: 0.2704 - val_accuracy: 0.0458
Epoch 45/50
235/235 [==============================] - 1s 6ms/step - loss: 0.2673 - accuracy: 0.0431 - val_loss: 0.2701 - val_accuracy: 0.0410
Epoch 46/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2671 - accuracy: 0.0436 - val_loss: 0.2698 - val_accuracy: 0.0422
Epoch 47/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2670 - accuracy: 0.0440 - val_loss: 0.2700 - val_accuracy: 0.0426
Epoch 48/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2668 - accuracy: 0.0446 - val_loss: 0.2695 - val_accuracy: 0.0386
Epoch 49/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2666 - accuracy: 0.0441 - val_loss: 0.2692 - val_accuracy: 0.0452
Epoch 50/50
235/235 [==============================] - 1s 5ms/step - loss: 0.2666 - accuracy: 0.0456 - val_loss: 0.2696 - val_accuracy: 0.0388
Test loss: [0.2696399390697479, 0.03880000114440918]
```

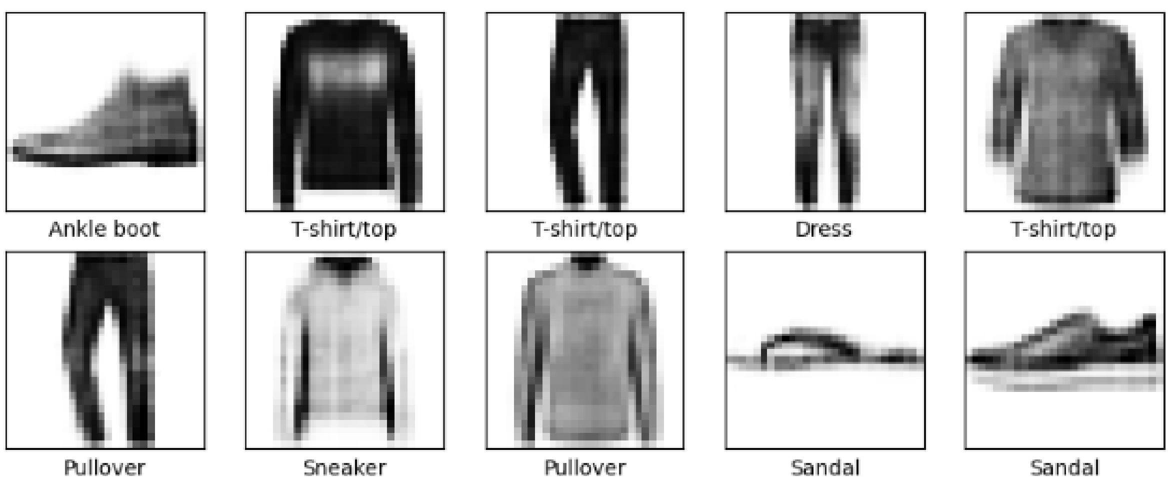
```
]  # Generate reconstructed images
   decoded_imgs = autoencoder.predict(x_test)
```

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

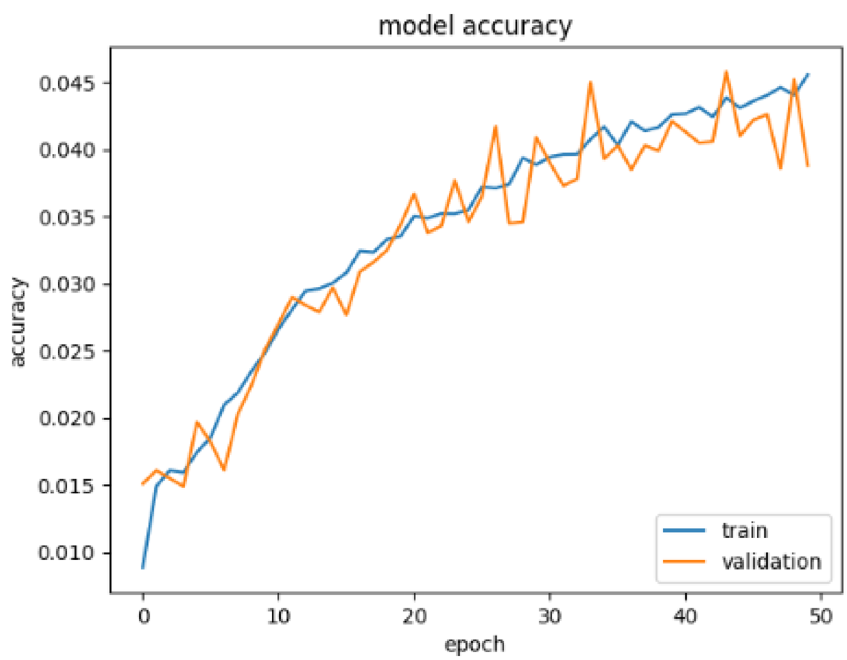
```
]  import tensorflow as tf
   from tensorflow import keras
   import numpy as np
   import matplotlib.pyplot as plt

   # Load the Fashion-MNIST dataset
   (x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()

   # Define the class names
   class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
                  'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

   # Plot some examples of images from the dataset
   plt.figure(figsize=(10,10))
   for i in range(10):
       plt.subplot(5,5,i+1)
       plt.xticks([])
       plt.yticks([])
       plt.grid(False)
       plt.imshow(x_train[i], cmap=plt.cm.binary)
       plt.xlabel(class_names[y_train[i]])
   plt.show()
```




```python
# Plot some examples of reconstructed images from the dataset
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

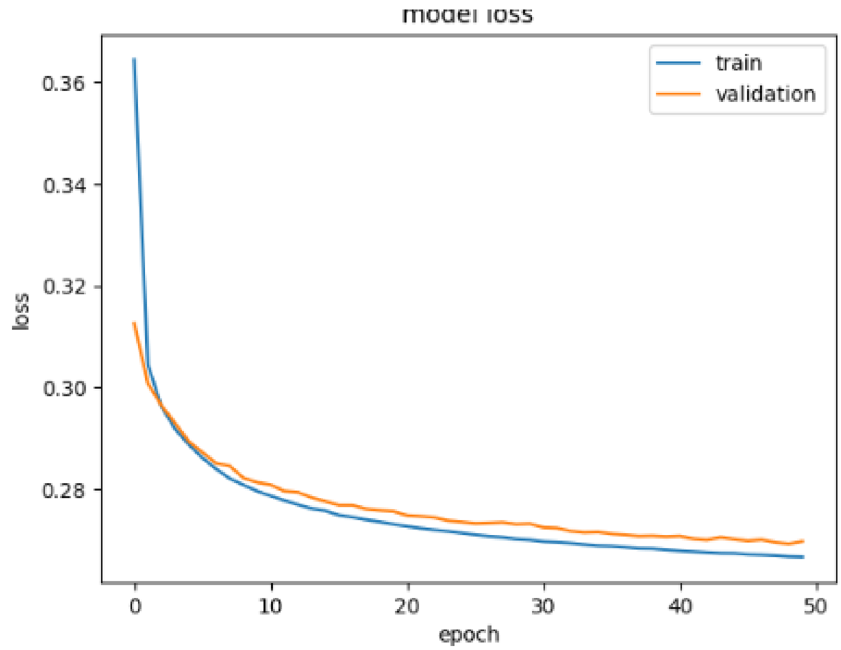


```python
import numpy as np
import matplotlib.pyplot as plt

# plot history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='lower right')
plt.show()
# plot history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
```




```
<matplotlib.legend.Legend at 0x7f30803d7760>
```

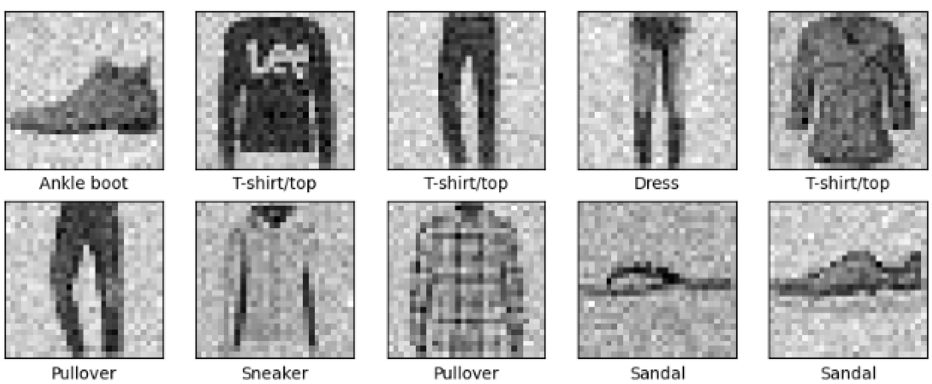
...



Denoisening Autoencoder

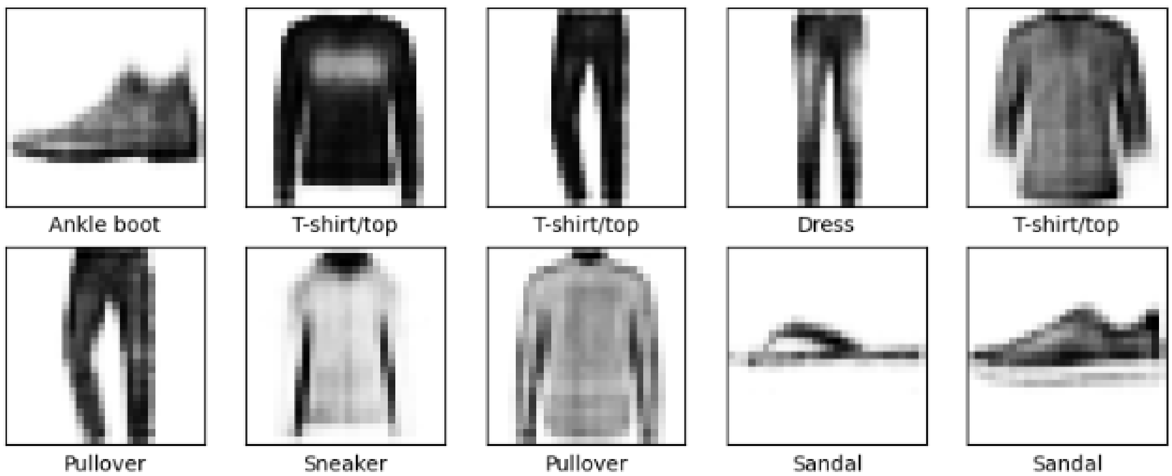
```
[ ]  from keras.layers import Input, Dense
     from keras.models import Model
     from keras.datasets import fashion_mnist
     import numpy as np

     # Load Fashion MNIST dataset
     (x_train, _), (x_test, _) = fashion_mnist.load_data()

     # Normalize pixel values between 0 and 1
     x_train = x_train.astype('float32') / 255.
     x_test = x_test.astype('float32') / 255.

     # Reshape images to vectors
     x_train = x_train.reshape((len(x_train), np.prod(x_train.shape[1:])))
     x_test = x_test.reshape((len(x_test), np.prod(x_test.shape[1:])))

     # Define autoencoder model
     input_img = Input(shape=(784,))
     encoded = Dense(256, activation='relu')(input_img)
     encoded = Dense(128, activation='relu')(encoded)
```

```python
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt

# Load the Fashion-MNIST dataset
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()

# Define the class names
class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress',
               'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

# Plot some examples of images from the dataset
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(x_test_noisy[i].reshape(28, 28), cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```

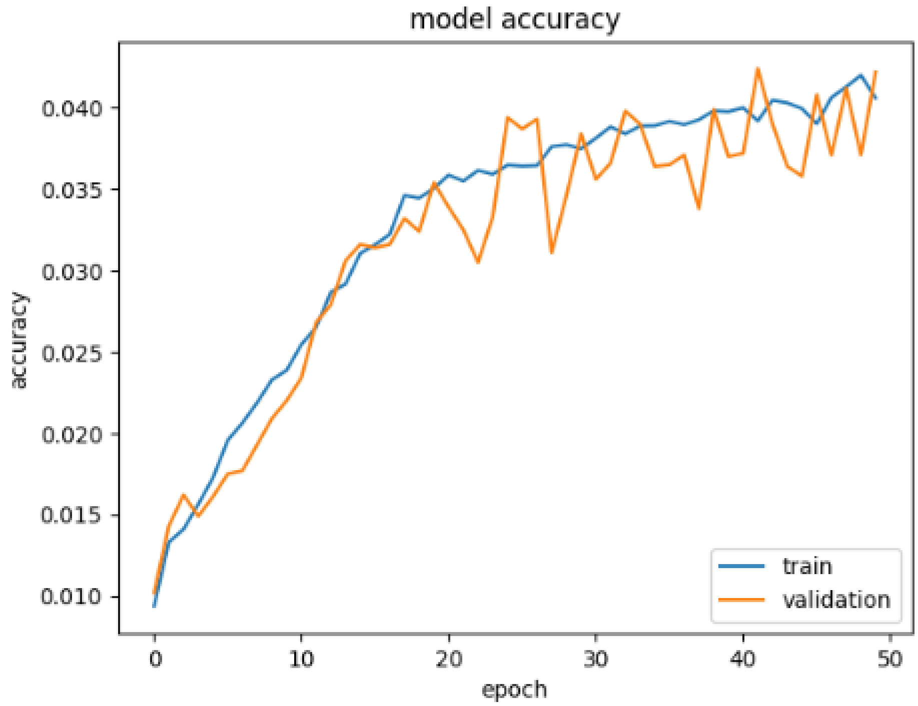



```python
# Plot some examples of reconstructed images from the dataset
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
```

```python
# Plot some examples of reconstructed images from the dataset
plt.figure(figsize=(10,10))
for i in range(10):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(decoded_imgs[i].reshape(28, 28), cmap=plt.cm.binary)
    plt.xlabel(class_names[y_train[i]])
plt.show()
```




```python
import numpy as np
import matplotlib.pyplot as plt

# plot history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='lower right')
plt.show()
# plot history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'validation'], loc='upper right')
```



`<matplotlib.legend.Legend at 0x7f308119d9d0>`