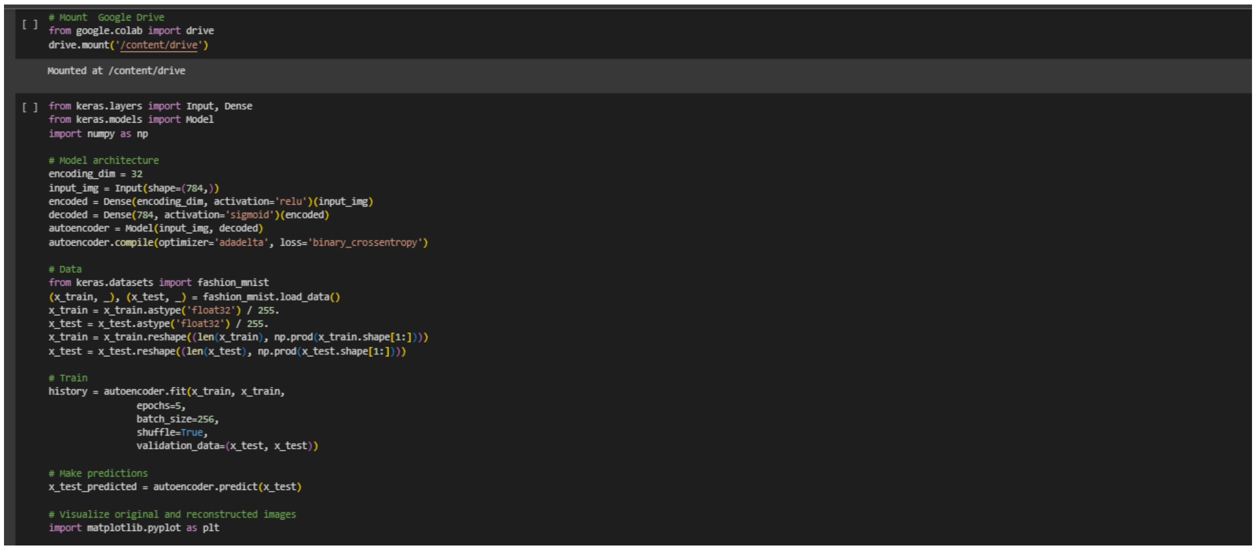
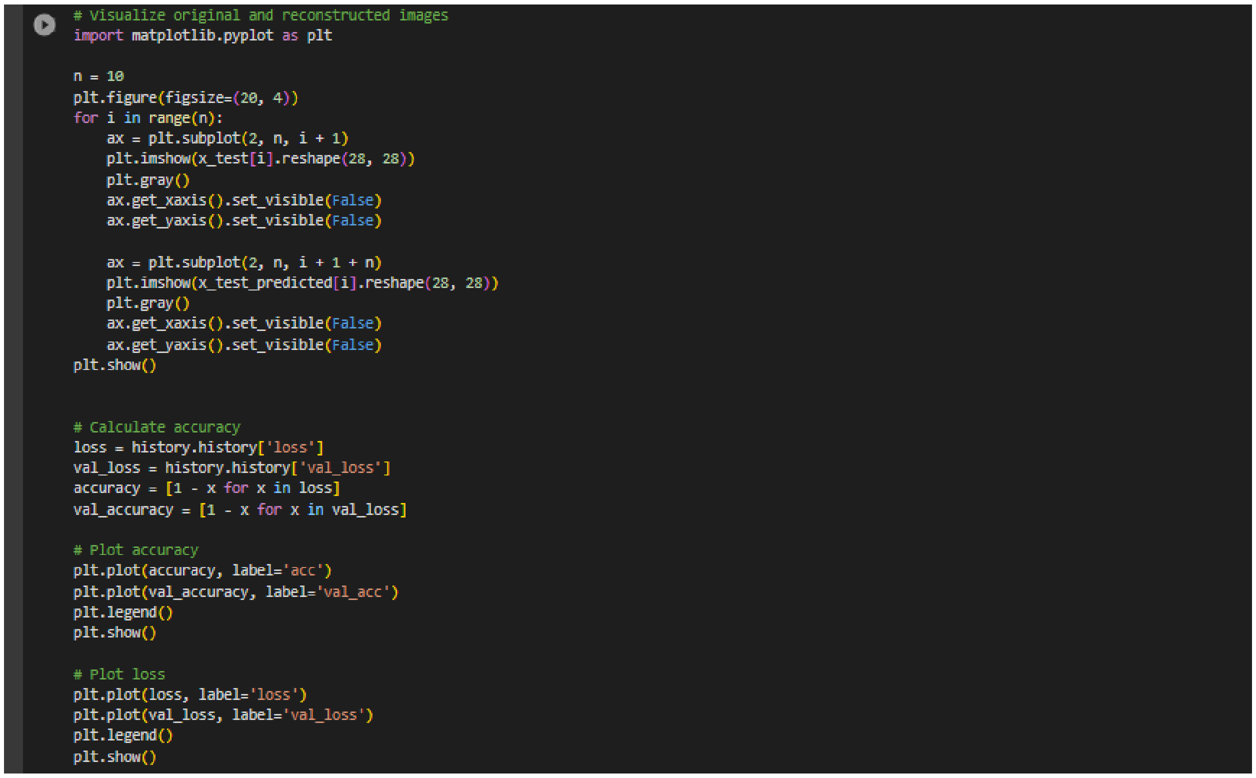
ICP 8 REPORT



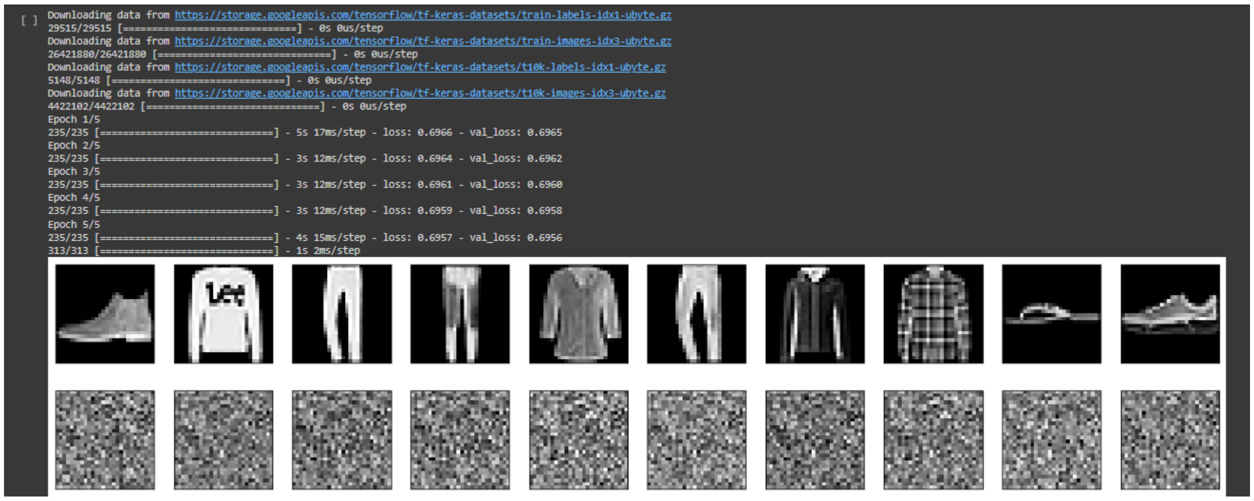
```
# Mount  Google Drive
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

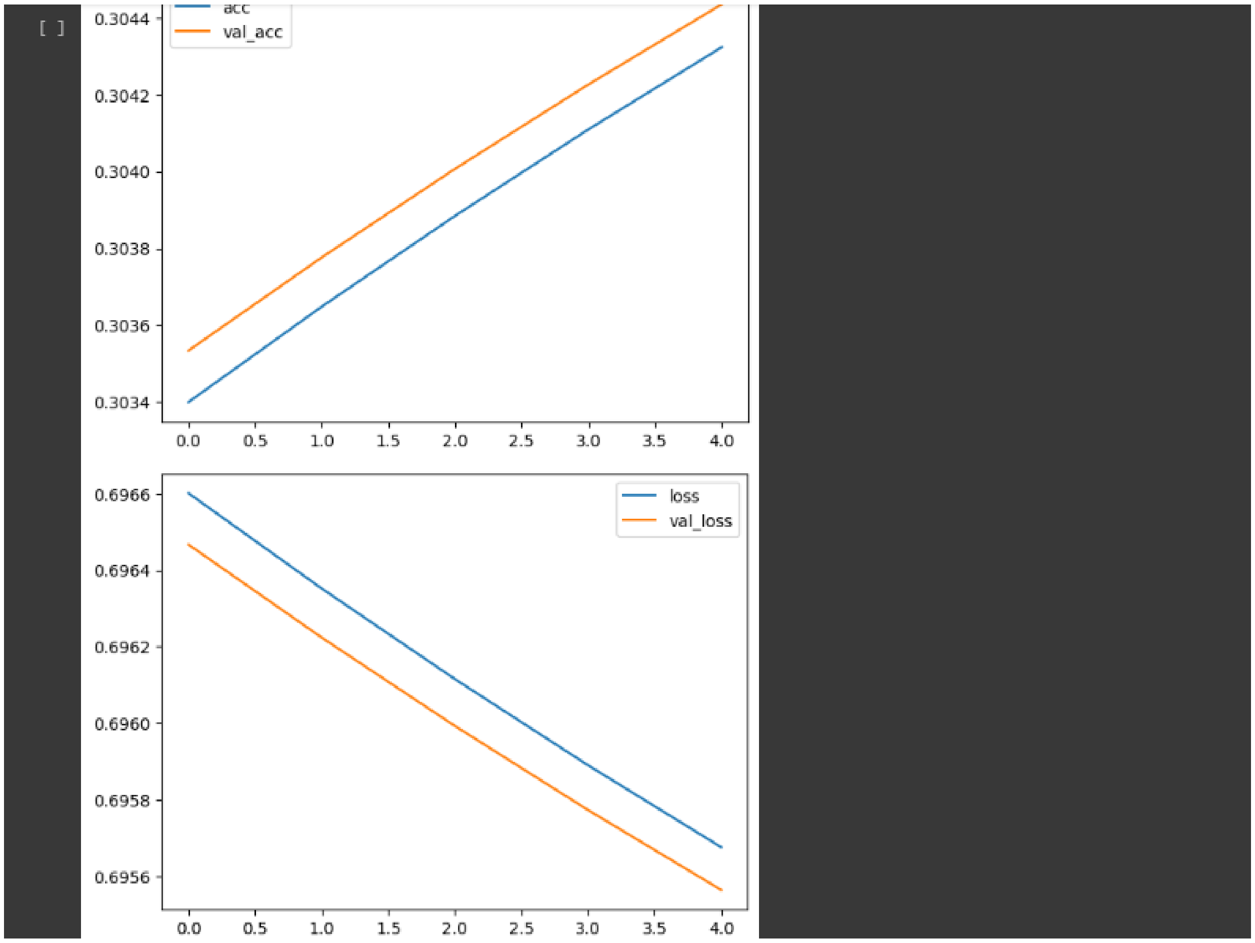
```
from keras.layers import Input, Dense
from keras.models import Model
import numpy as np

# Model architecture
encoding_dim = 32
input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')

# Data
from keras.datasets import fashion_mnist
(x_train, _), (x_test, _) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
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:])))

# Train
history = autoencoder.fit(x_train, x_train,
                epochs=5,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))

# Make predictions
x_test_predicted = autoencoder.predict(x_test)

# Visualize original and reconstructed images
import matplotlib.pyplot as plt
```

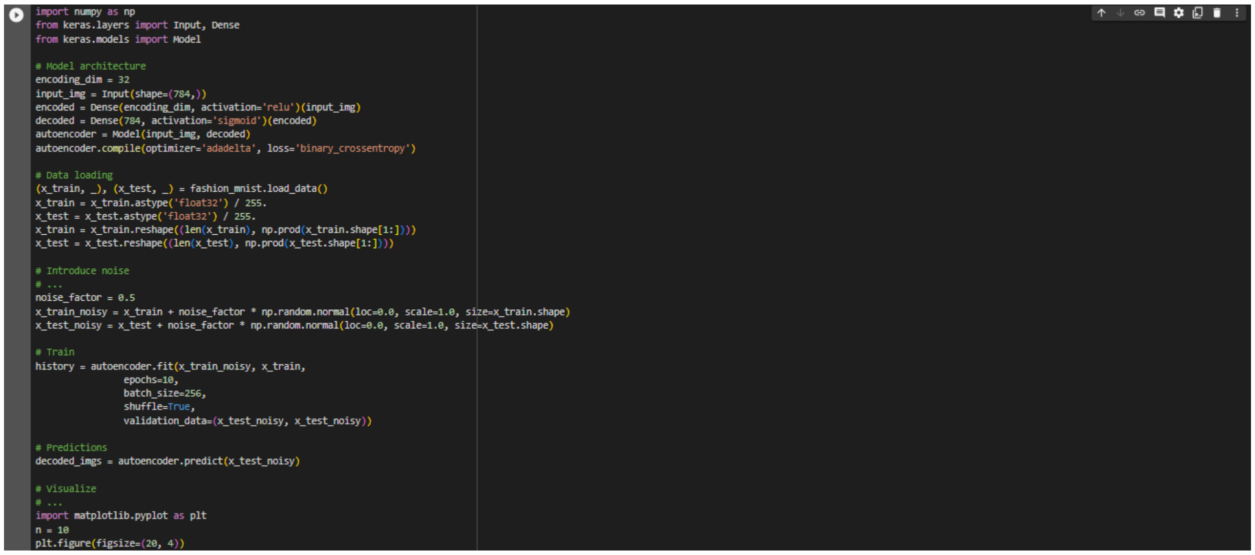


```
# Visualize original and reconstructed images
import matplotlib.pyplot as plt

n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(x_test_predicted[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()


# Calculate accuracy
loss = history.history['loss']
val_loss = history.history['val_loss']
accuracy = [1 - x for x in loss]
val_accuracy = [1 - x for x in val_loss]

# Plot accuracy
plt.plot(accuracy, label='acc')
plt.plot(val_accuracy, label='val_acc')
plt.legend()
plt.show()

# Plot loss
plt.plot(loss, label='loss')
plt.plot(val_loss, label='val_loss')
plt.legend()
plt.show()
```

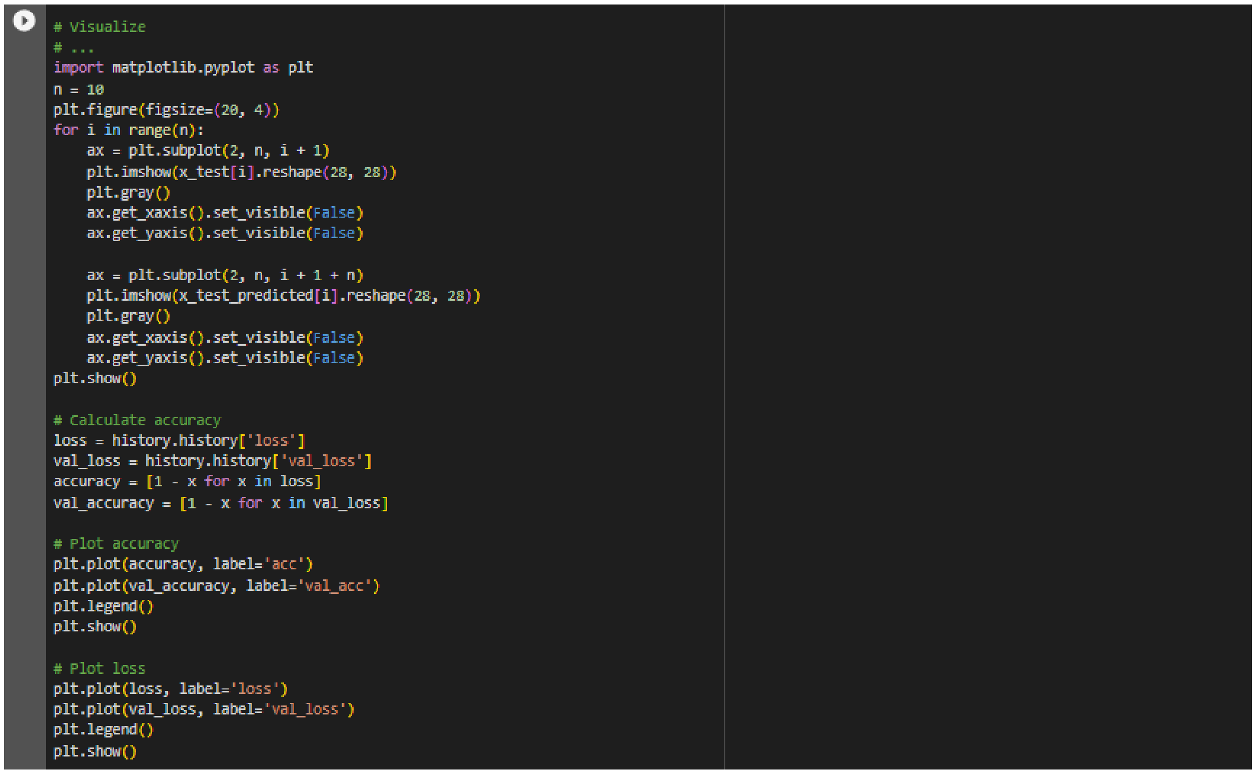


```
[ ] Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
    29515/29515 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
    26421880/26421880 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
    5148/5148 [==============================] - 0s 0us/step
    Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
    4422102/4422102 [==============================] - 0s 0us/step
    Epoch 1/5
    235/235 [==============================] - 5s 17ms/step - loss: 0.6966 - val_loss: 0.6965
    Epoch 2/5
    235/235 [==============================] - 3s 12ms/step - loss: 0.6964 - val_loss: 0.6962
    Epoch 3/5
    235/235 [==============================] - 3s 12ms/step - loss: 0.6961 - val_loss: 0.6960
    Epoch 4/5
    235/235 [==============================] - 3s 12ms/step - loss: 0.6959 - val_loss: 0.6958
    Epoch 5/5
    235/235 [==============================] - 4s 15ms/step - loss: 0.6957 - val_loss: 0.6956
    313/313 [==============================] - 1s 2ms/step
```

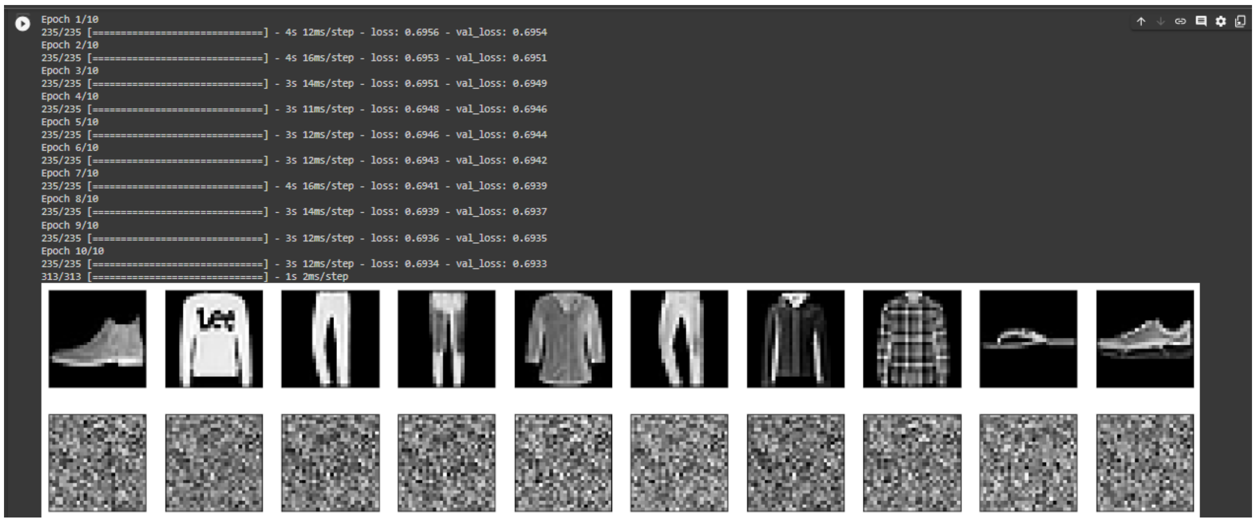




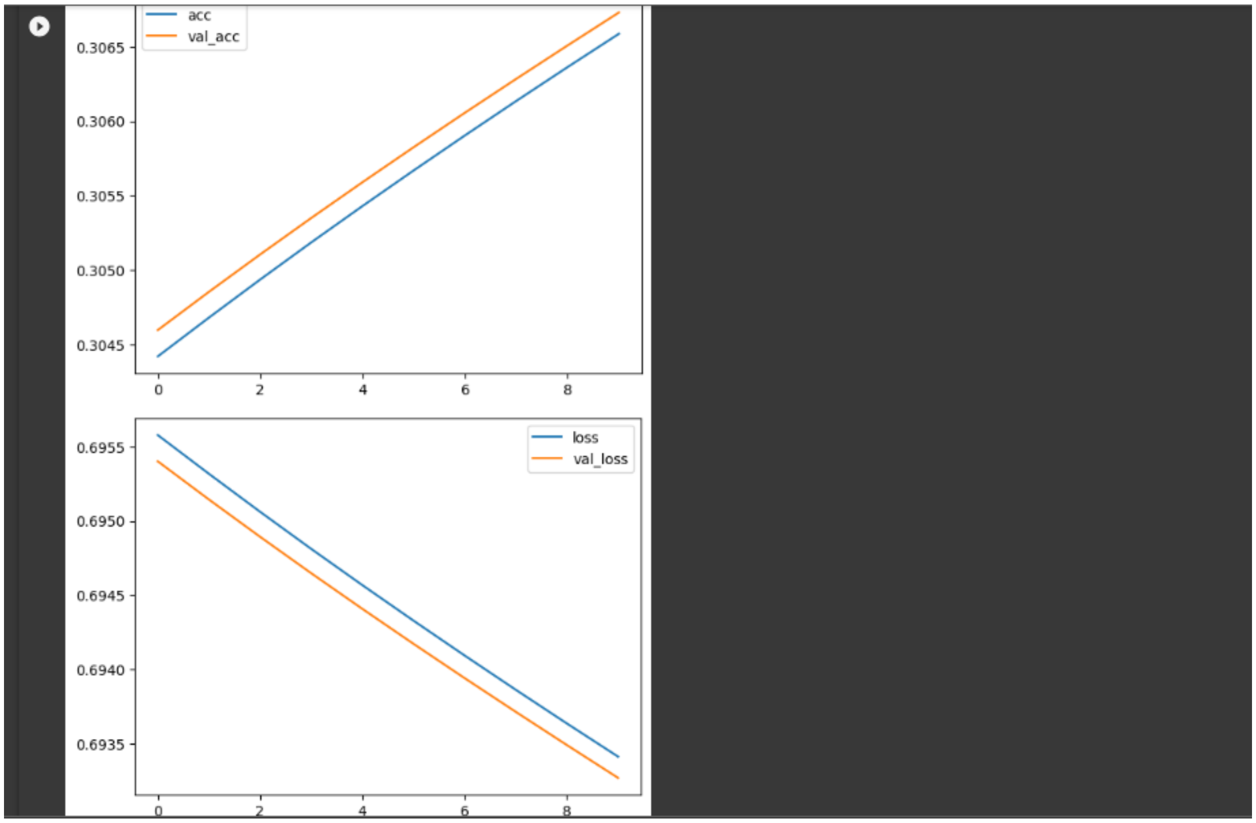
```python
import numpy as np
from keras.layers import Input, Dense
from keras.models import Model

# Model architecture
encoding_dim = 32
input_img = Input(shape=(784,))
encoded = Dense(encoding_dim, activation='relu')(input_img)
decoded = Dense(784, activation='sigmoid')(encoded)
autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')

# Data loading
(x_train, _), (x_test, _) = fashion_mnist.load_data()
x_train = x_train.astype('float32') / 255.
x_test = x_test.astype('float32') / 255.
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:])))

# Introduce noise
# ...
noise_factor = 0.5
x_train_noisy = x_train + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_train.shape)
x_test_noisy = x_test + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=x_test.shape)

# Train
history = autoencoder.fit(x_train_noisy, x_train,
                epochs=10,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test_noisy, x_test_noisy))

# Predictions
decoded_imgs = autoencoder.predict(x_test_noisy)

# Visualize
# ...
import matplotlib.pyplot as plt
n = 10
plt.figure(figsize=(20, 4))
```



```python
# Visualize
# ...
import matplotlib.pyplot as plt
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
    ax = plt.subplot(2, n, i + 1)
    plt.imshow(x_test[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)

    ax = plt.subplot(2, n, i + 1 + n)
    plt.imshow(x_test_predicted[i].reshape(28, 28))
    plt.gray()
    ax.get_xaxis().set_visible(False)
    ax.get_yaxis().set_visible(False)
plt.show()

# Calculate accuracy
loss = history.history['loss']
val_loss = history.history['val_loss']
accuracy = [1 - x for x in loss]
val_accuracy = [1 - x for x in val_loss]

# Plot accuracy
plt.plot(accuracy, label='acc')
plt.plot(val_accuracy, label='val_acc')
plt.legend()
plt.show()

# Plot loss
plt.plot(loss, label='loss')
plt.plot(val_loss, label='val_loss')
plt.legend()
plt.show()
```



```
Epoch 1/10
235/235 [==============================] - 4s 12ms/step - loss: 0.6956 - val_loss: 0.6954
Epoch 2/10
235/235 [==============================] - 4s 16ms/step - loss: 0.6953 - val_loss: 0.6951
Epoch 3/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6951 - val_loss: 0.6949
Epoch 4/10
235/235 [==============================] - 3s 11ms/step - loss: 0.6948 - val_loss: 0.6946
Epoch 5/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6946 - val_loss: 0.6944
Epoch 6/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6943 - val_loss: 0.6942
Epoch 7/10
235/235 [==============================] - 4s 16ms/step - loss: 0.6941 - val_loss: 0.6939
Epoch 8/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6939 - val_loss: 0.6937
Epoch 9/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6936 - val_loss: 0.6935
Epoch 10/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6934 - val_loss: 0.6933
313/313 [==============================] - 1s 2ms/step
```



Repository link: https://github.com/sxk7912/Bigdata