

ICP 8 REPORT

```
ICP8.ipynb
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

# 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
```

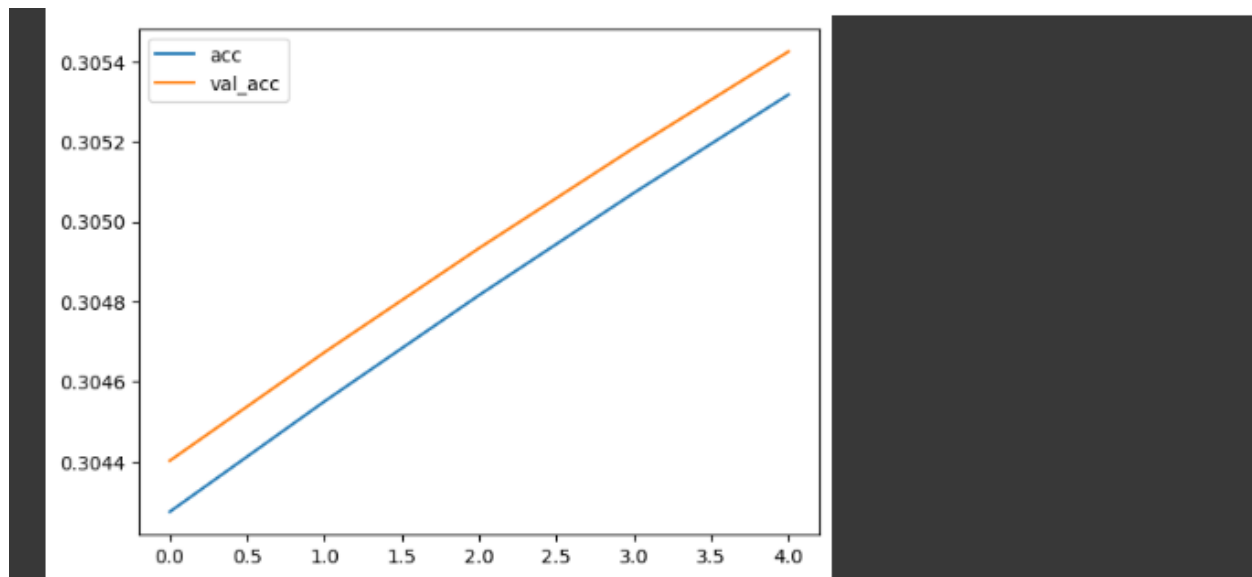
```
# Visualize original and reconstructed images
import matplotlib.pyplot as plt

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()
```



```
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='adam', 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
x_test_predicted = autoencoder.predict(x_test_noisy)
```

```

# 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]

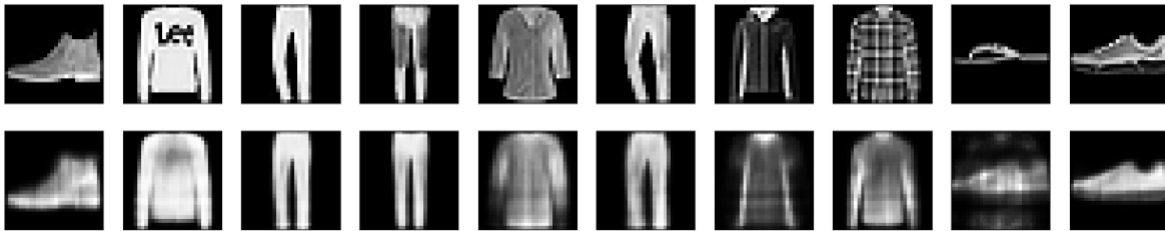
# Plot accuracy
plt.plot(accuracy, label='acc')
plt.plot(val_accuracy, label='val_acc')
plt.legend()
plt.show()

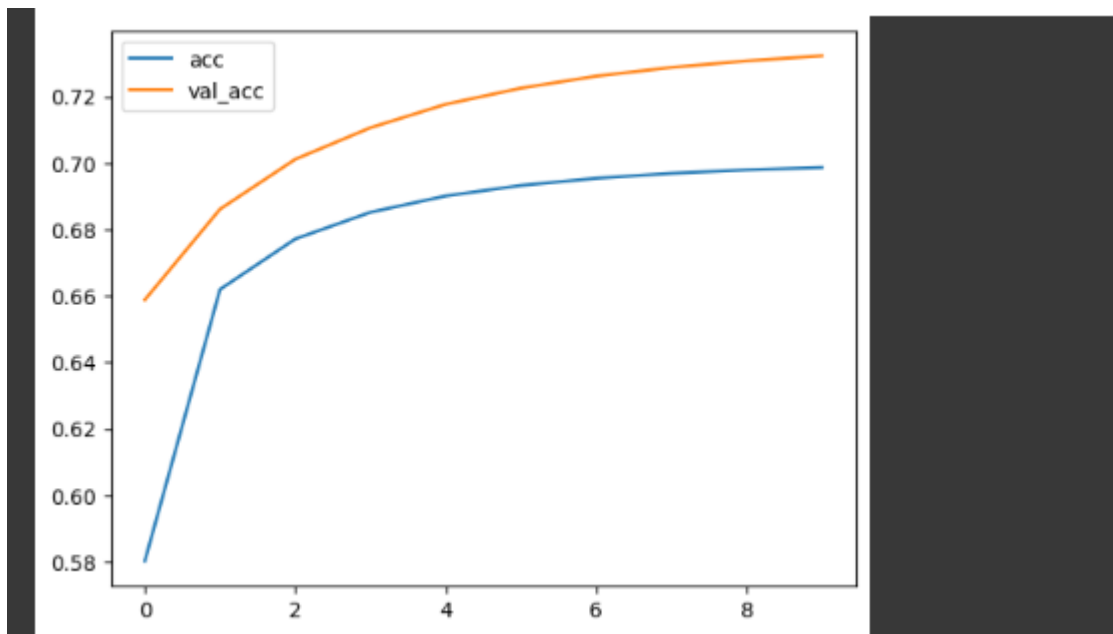
```

```

Epoch 1/10
235/235 [=====] - 4s 13ms/step - loss: 0.4197 - val_loss: 0.3411
Epoch 2/10
235/235 [=====] - 2s 10ms/step - loss: 0.3380 - val_loss: 0.3139
Epoch 3/10
235/235 [=====] - 2s 10ms/step - loss: 0.3228 - val_loss: 0.2988
Epoch 4/10
235/235 [=====] - 2s 10ms/step - loss: 0.3148 - val_loss: 0.2894
Epoch 5/10
235/235 [=====] - 3s 11ms/step - loss: 0.3099 - val_loss: 0.2823
Epoch 6/10
235/235 [=====] - 4s 15ms/step - loss: 0.3067 - val_loss: 0.2775
Epoch 7/10
235/235 [=====] - 3s 11ms/step - loss: 0.3046 - val_loss: 0.2738
Epoch 8/10
235/235 [=====] - 2s 10ms/step - loss: 0.3031 - val_loss: 0.2712
Epoch 9/10
235/235 [=====] - 2s 9ms/step - loss: 0.3020 - val_loss: 0.2693
Epoch 10/10
235/235 [=====] - 2s 10ms/step - loss: 0.3013 - val_loss: 0.2677
313/313 [=====] - 1s 2ms/step

```





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