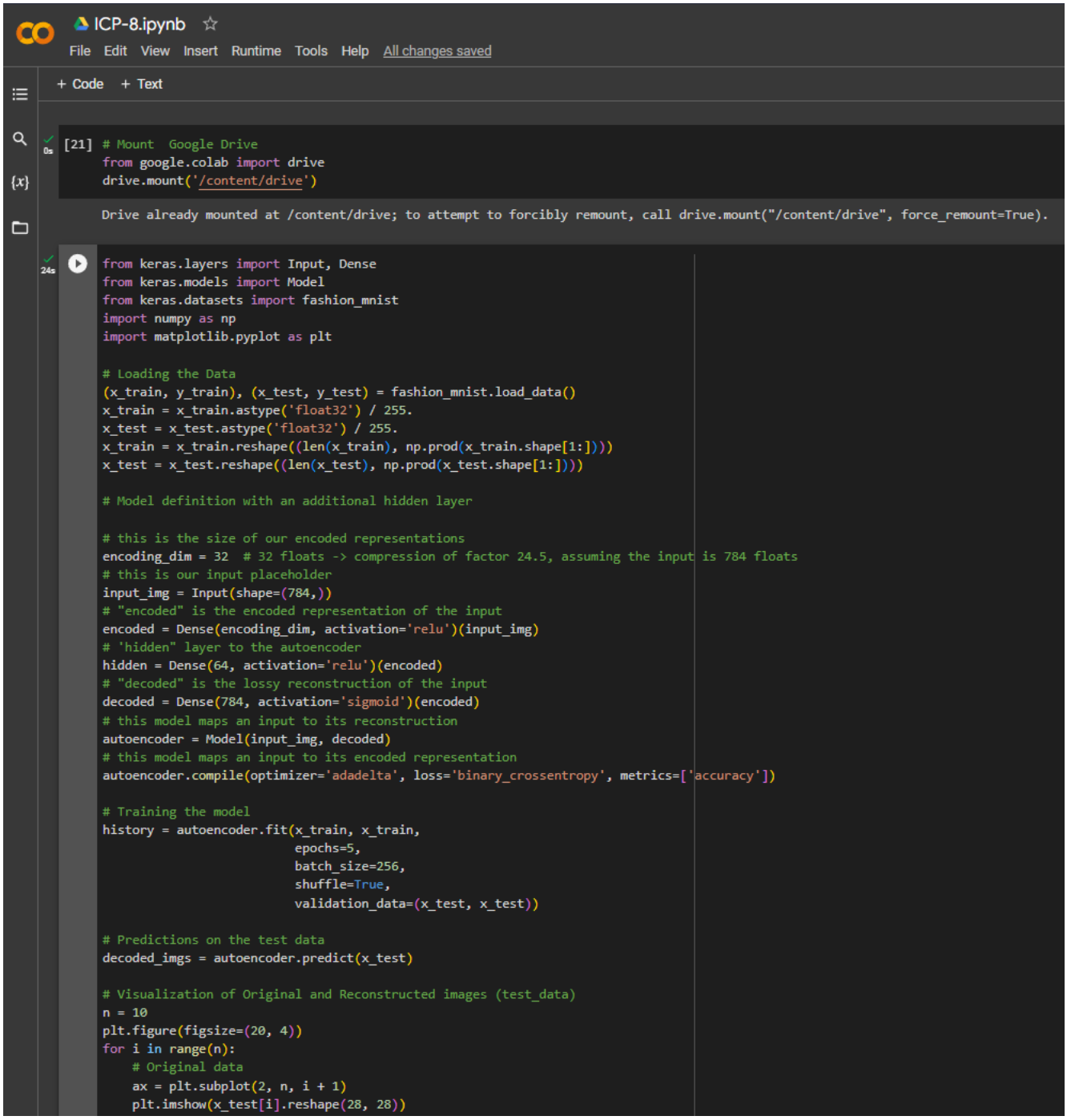

ICP-8.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[21] # Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
from keras.layers import Input, Dense
from keras.models import Model
from keras.datasets import fashion_mnist
import numpy as np
import matplotlib.pyplot as plt
# Loading the Data
(x_train, y_train), (x_test, y_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:])))
# Model definition with an additional hidden layer
# this is the size of our encoded representations
encoding_dim = 32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
# this is our input placeholder
input_img = Input(shape=(784,))
# "encoded" is the encoded representation of the input
encoded = Dense(encoding_dim, activation='relu')(input_img)
# 'hidden" layer to the autoencoder
hidden = Dense(64, activation='relu')(encoded)
# "decoded" is the lossy reconstruction of the input
decoded = Dense(784, activation='sigmoid')(encoded)
# this model maps an input to its reconstruction
autoencoder = Model(input_img, decoded)
# this model maps an input to its encoded representation
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])
# Training the model
history = autoencoder.fit(x_train, x_train,
epochs=5,
batch_size=256,
shuffle=True,
validation_data=(x_test, x_test))
# Predictions on the test data
decoded_imgs = autoencoder.predict(x_test)
# Visualization of Original and Reconstructed images (test_data)
n = 10
plt.figure(figsize=(20, 4))
for i in range(n):
# Original data
ax = plt.subplot(2, n, i + 1)
plt.imshow(x_test[i].reshape(28, 28))

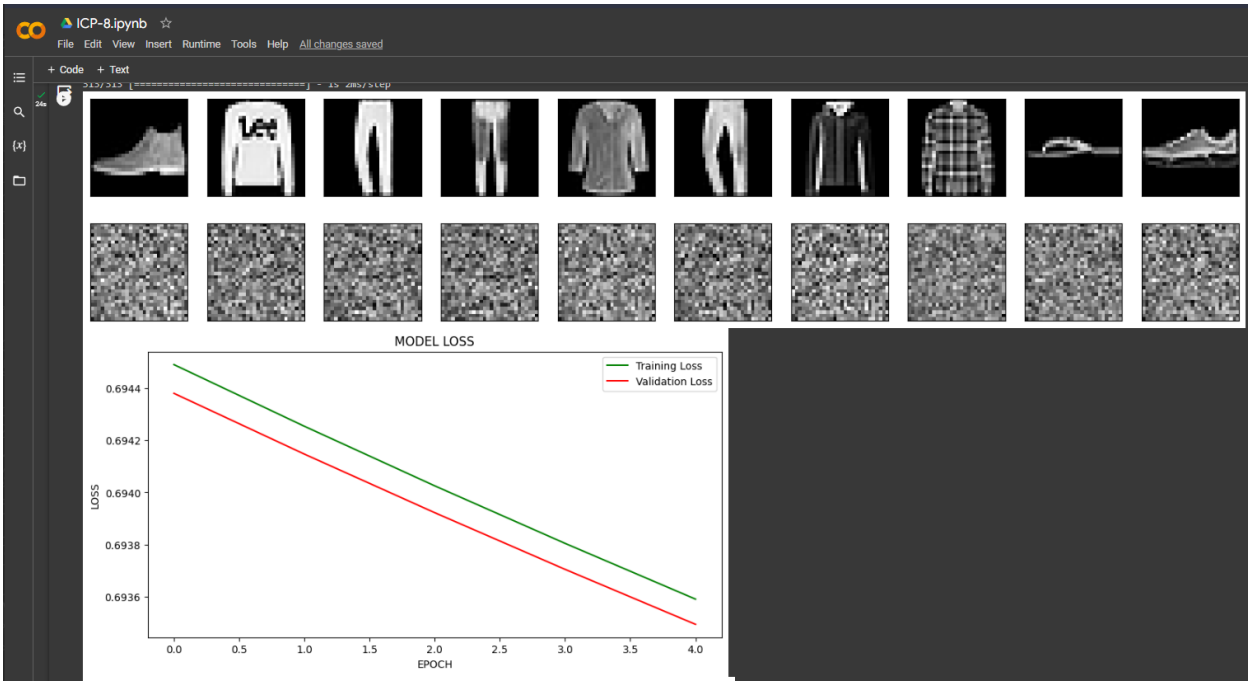
ICP-8.ipynb

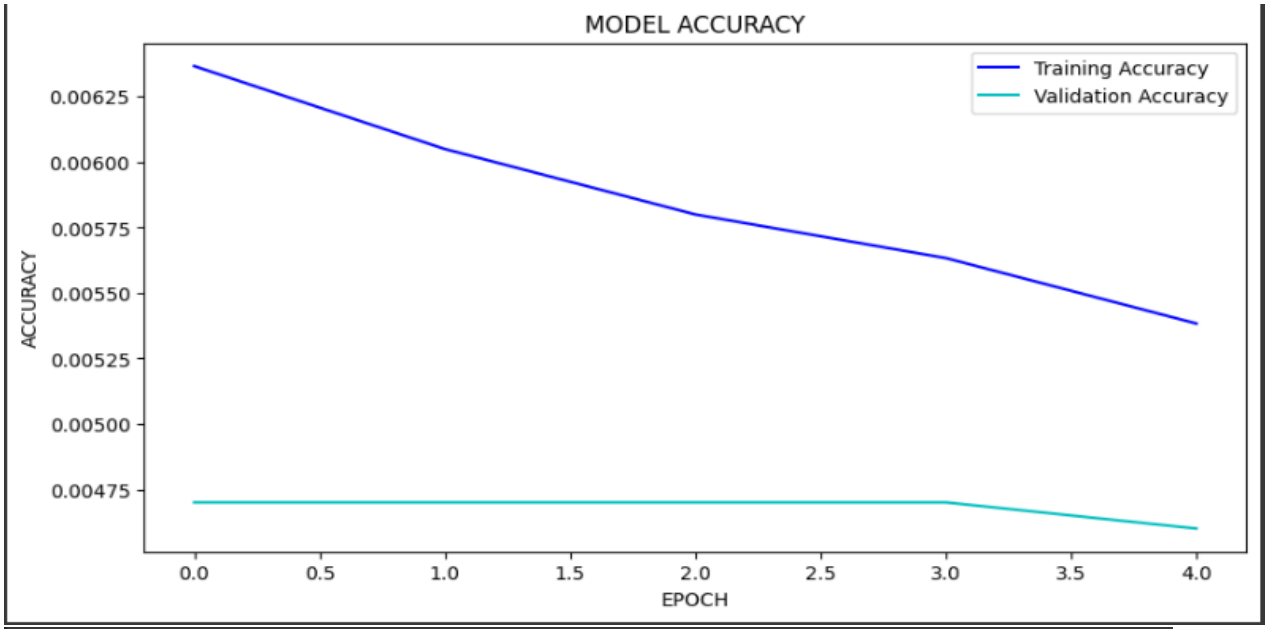
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
    # Original data
    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)

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

# Plotting the Loss
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], 'g-', label='Training Loss')
plt.plot(history.history['val_loss'], 'r-', label='Validation Loss')
plt.title('MODEL LOSS')
plt.xlabel('EPOCH')
plt.ylabel('LOSS')
plt.legend()
plt.show()

# Plotting the Accuracy
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], 'b-', label='Training Accuracy')
plt.plot(history.history['val_accuracy'], 'c-', label='Validation Accuracy')
plt.title('MODEL ACCURACY')
plt.xlabel('EPOCH')
plt.ylabel('ACCURACY')
plt.legend()
plt.show()
```

```
Epoch 1/5
235/235 [==============================] - 3s 11ms/step - loss: 0.6945 - accuracy: 0.0064 - val_loss: 0.6944 - val_accuracy: 0.0047
Epoch 2/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6943 - accuracy: 0.0060 - val_loss: 0.6941 - val_accuracy: 0.0047
Epoch 3/5
235/235 [==============================] - 2s 10ms/step - loss: 0.6940 - accuracy: 0.0058 - val_loss: 0.6939 - val_accuracy: 0.0047
Epoch 4/5
235/235 [==============================] - 3s 12ms/step - loss: 0.6938 - accuracy: 0.0056 - val_loss: 0.6937 - val_accuracy: 0.0047
Epoch 5/5
235/235 [==============================] - 4s 15ms/step - loss: 0.6936 - accuracy: 0.0054 - val_loss: 0.6935 - val_accuracy: 0.0046
313/313 [==============================] - 1s 2ms/step
```





ICP-8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

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

     # Loading the data
     (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:])))

     # Introducing the 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)

     # Model definition:

     # this is the size of our encoded representations
     encoding_dim = 32  # 32 floats -> compression of factor 24.5, assuming the input is 784 floats
     # this is our input placeholder
     input_img = Input(shape=(784,))
     # "encoded" is the encoded representation of the input
     encoded = Dense(encoding_dim, activation='relu')(input_img)
     # "decoded" is the lossy reconstruction of the input
     decoded = Dense(784, activation='sigmoid')(encoded)
     # this model maps an input to its reconstruction
     autoencoder = Model(input_img, decoded)
     # this model maps an input to its encoded representation
     autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy', metrics=['accuracy'])

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

     # Predictions on the test data
     decoded_imgs = autoencoder.predict(x_test_noisy)

     # Visualization of noisy and reconstructed images
     n = 10
     plt.figure(figsize=(20, 4))
     for i in range(n):
         # Noisy data
         ax = plt.subplot(2, n, i + 1)
         plt.imshow(x_test_noisy[i].reshape(28, 28))
         plt.gray()
         ax.get_xaxis().set_visible(False)
         ax.get_yaxis().set_visible(False)
```

ICP-8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

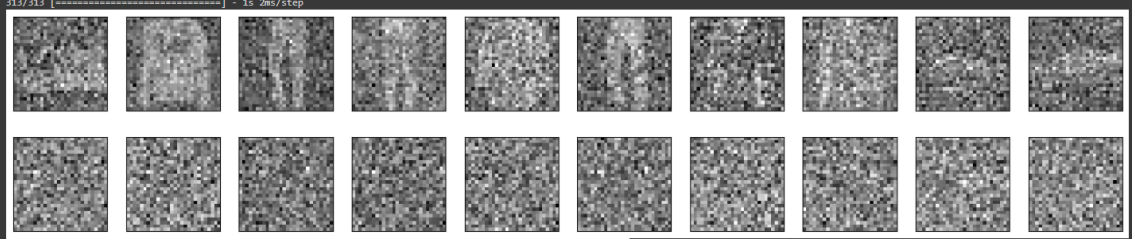
```
        ax.get_yaxis().set_visible(False)

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

# Plotting the Loss
plt.figure(figsize=(10, 5))
plt.plot(history.history['loss'], 'g-', label='Training Loss')
plt.plot(history.history['val_loss'], 'r-', label='Validation Loss')
plt.title('MODEL LOSS')
plt.xlabel('EPOCH')
plt.ylabel('LOSS')
plt.legend()
plt.show()

# Plotting accuracy
plt.figure(figsize=(10, 5))
plt.plot(history.history['accuracy'], 'b-', label='Training Accuracy')
plt.plot(history.history['val_accuracy'], 'c-', label='Validation Accuracy')
plt.title('MODEL ACCURACY')
plt.xlabel('EPOCH')
plt.ylabel('ACCURACY')
plt.legend()
plt.show()
```

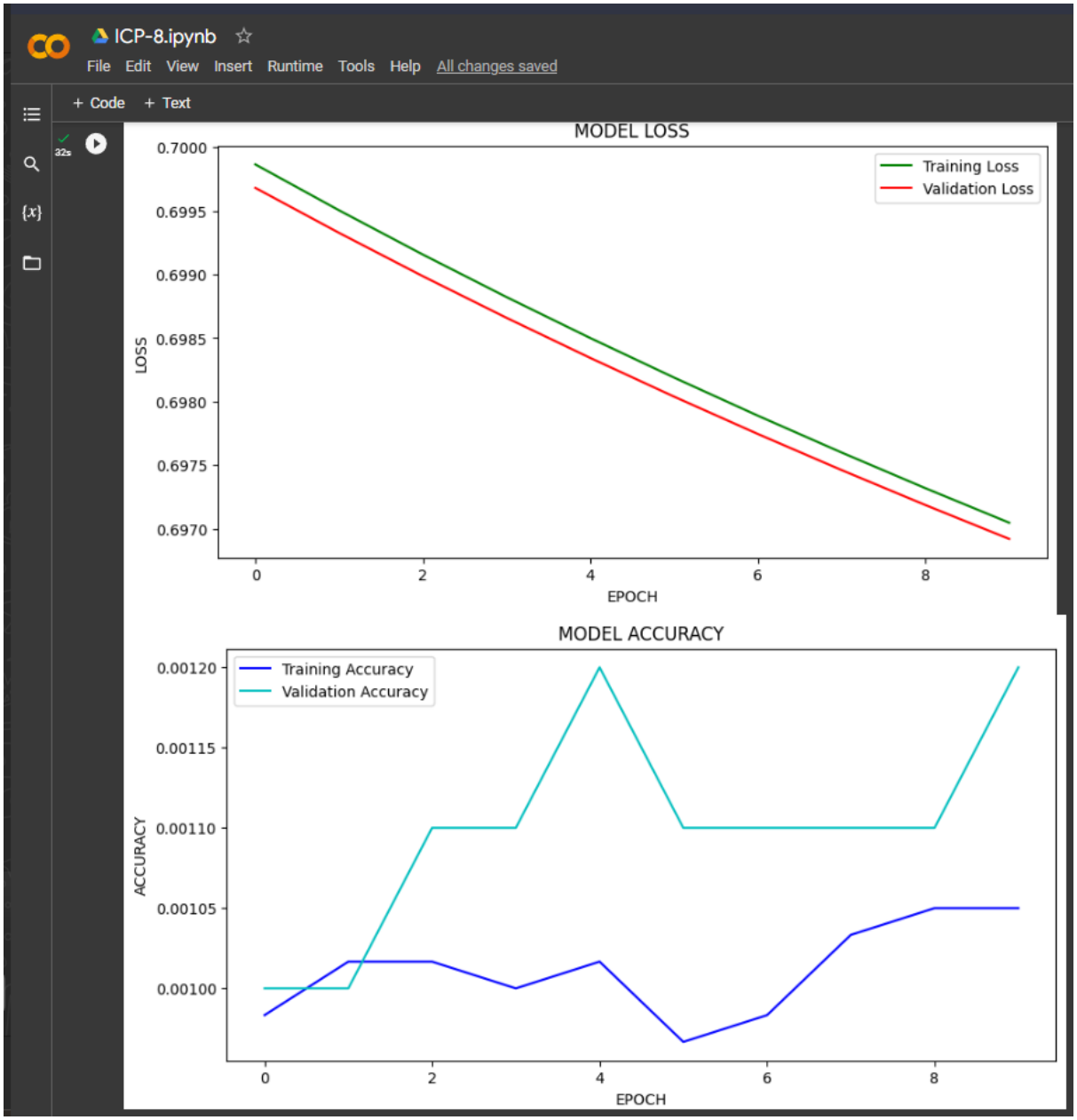
ICP-8.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
Epoch 1/10
235/235 [==============================] - 4s 12ms/step - loss: 0.6999 - accuracy: 9.8333e-04 - val_loss: 0.6997 - val_accuracy: 0.0010
Epoch 2/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6995 - accuracy: 0.0010 - val_loss: 0.6993 - val_accuracy: 0.0010
Epoch 3/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6992 - accuracy: 0.0010 - val_loss: 0.6990 - val_accuracy: 0.0011
Epoch 4/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6988 - accuracy: 0.0010 - val_loss: 0.6987 - val_accuracy: 0.0011
Epoch 5/10
235/235 [==============================] - 3s 12ms/step - loss: 0.6985 - accuracy: 0.0010 - val_loss: 0.6983 - val_accuracy: 0.0012
Epoch 6/10
235/235 [==============================] - 3s 14ms/step - loss: 0.6982 - accuracy: 9.6667e-04 - val_loss: 0.6980 - val_accuracy: 0.0011
Epoch 7/10
235/235 [==============================] - 2s 9ms/step - loss: 0.6979 - accuracy: 9.8333e-04 - val_loss: 0.6977 - val_accuracy: 0.0011
Epoch 8/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6976 - accuracy: 0.0010 - val_loss: 0.6975 - val_accuracy: 0.0011
Epoch 9/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6973 - accuracy: 0.0010 - val_loss: 0.6972 - val_accuracy: 0.0011
Epoch 10/10
235/235 [==============================] - 2s 10ms/step - loss: 0.6970 - accuracy: 0.0010 - val_loss: 0.6969 - val_accuracy: 0.0012
313/313 [==============================] - 1s 2ms/step
```





GitHub Repo Link: https://github.com/Krypton0626/Bigdata/tree/main/ICP%208

YouTube Video Link: https://youtu.be/RABwMlyqtMI