**Spring 2024: CS5720  
 Neural Networks and Deep Learning - ICP-8**

**Yasaswini Majety (700747747)**

**Github Link: https://github.com/yasaswini8777/Neural-ICP8**

from keras.layers import Input, Dense

from keras.models import Model

from keras.datasets import fashion\_mnist

import numpy as np

encoding\_dim = 32 # 32 floats -> compression factor 24.5, assuming the input is 784 floats

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', metrics=['accuracy'])

(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:])))

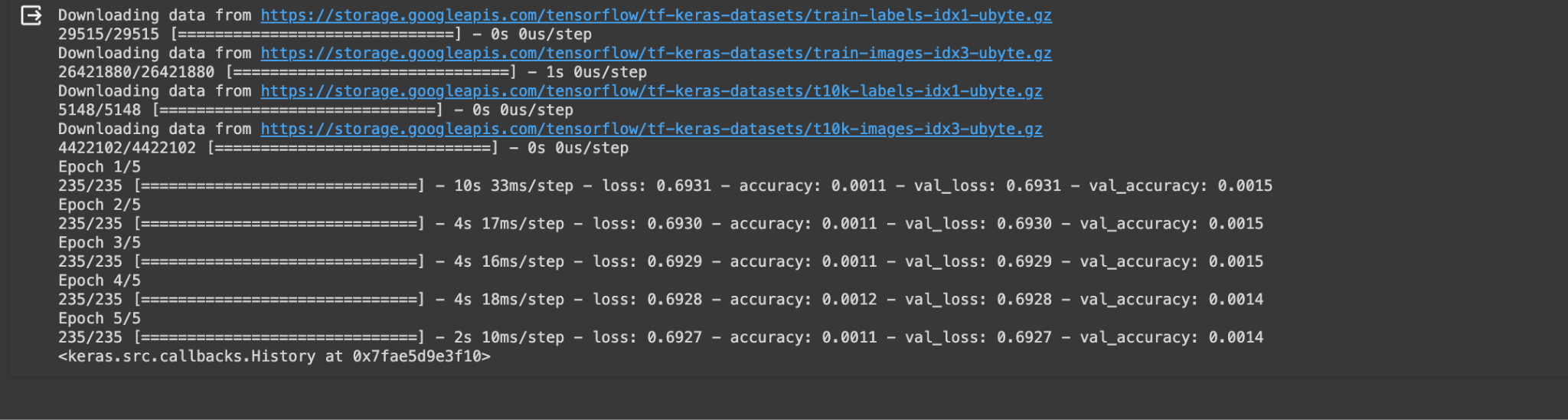
autoencoder.fit(x\_train, x\_train,

epochs=5,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test, x\_test))



from keras.layers import Input, Dense

from keras.models import Model

encoding\_dim = 32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

input\_img = Input(shape=(784,))

encoded1 = Dense(128, activation='relu')(input\_img)

encoded2 = Dense(encoding\_dim, activation='relu')(encoded1)

decoded1 = Dense(128, activation='relu')(encoded2)

decoded2 = Dense(784, activation='sigmoid')(decoded1)

autoencoder = Model(input\_img, decoded2)

encoder = Model(input\_img, encoded2)

encoded\_input = Input(shape=(encoding\_dim,))

decoder\_layer1 = autoencoder.layers[-2]

decoder\_layer2 = autoencoder.layers[-1]

decoder = Model(encoded\_input, decoder\_layer2(decoder\_layer1(encoded\_input)))

autoencoder.compile(optimizer='adadelta', loss='binary\_crossentropy',metrics ='accuracy')

from keras.datasets import mnist, fashion\_mnist

import numpy as np

(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:])))

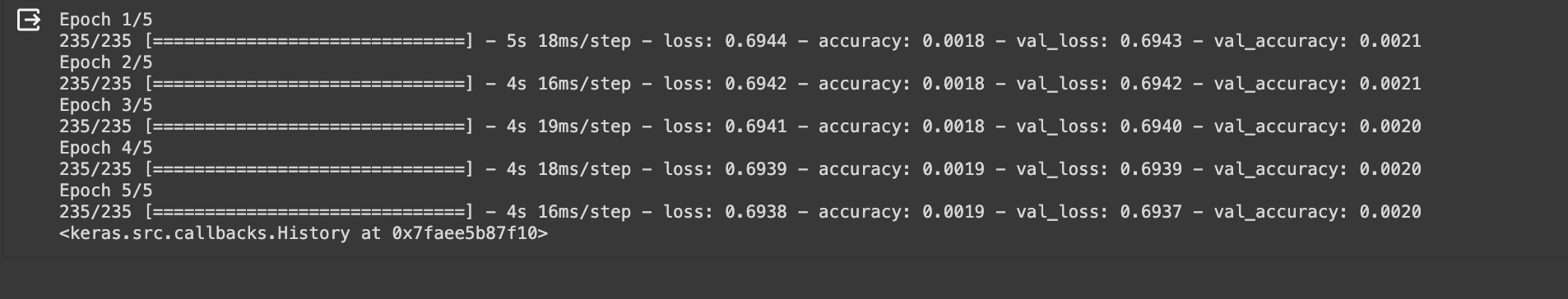
autoencoder.fit(x\_train, x\_train,

epochs=5,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test, x\_test))



import matplotlib.pyplot as plt

plt.showhe reconstructed images for the test set

reconstructed\_imgs = autoencoder.predict(x\_test)

# Choose a random image from the test set

n = 10 # index of the image to be plotted

plt.figure(figsize=(10, 5))

# Plot the original image

ax = plt.subplot(1, 2, 1)

plt.imshow(x\_test[n].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

ax.set\_title("Original Image")

# Plot the reconstructed image

ax = plt.subplot(1, 2, 2)

plt.imshow(reconstructed\_imgs[n].reshape(28, 28))

plt.gray()

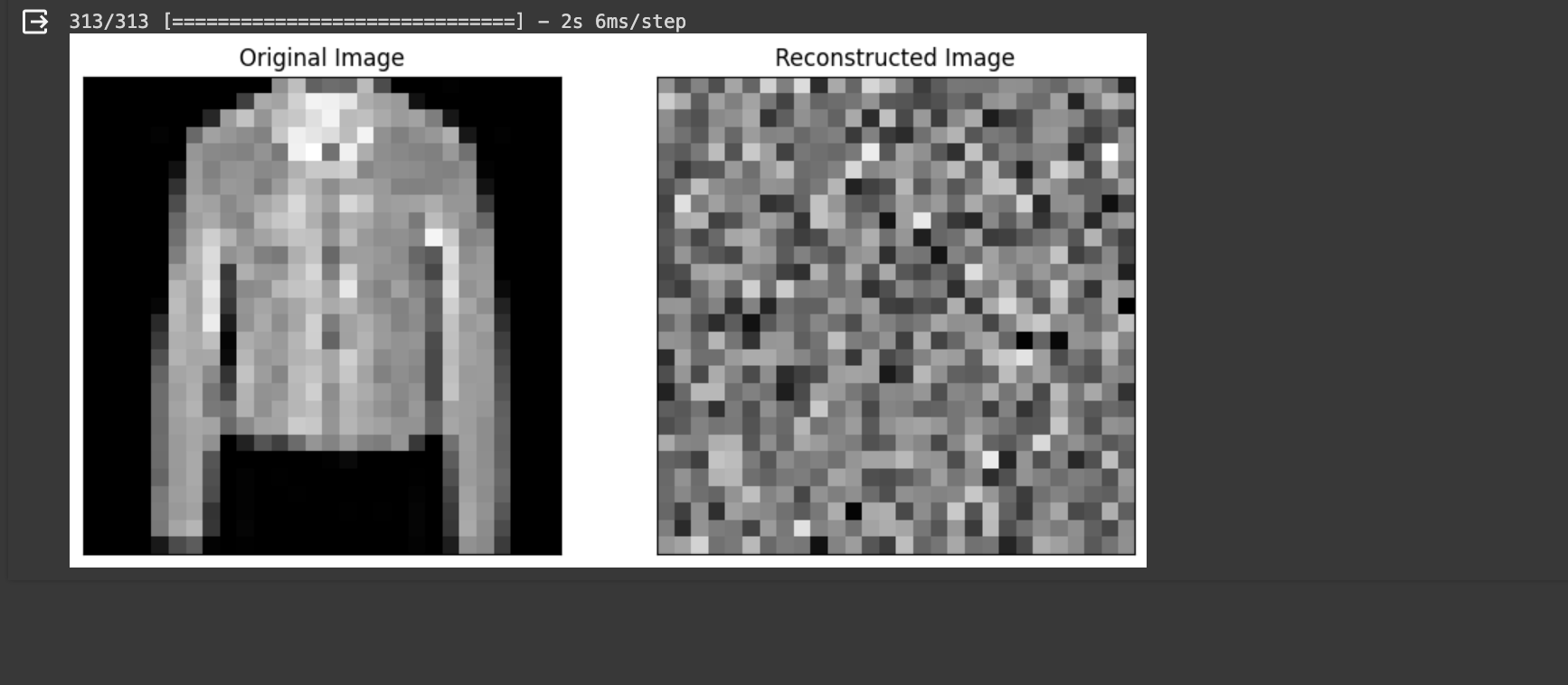
ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

ax.set\_title("Reconstructed Image")

plt.show()

Output:



from keras.layers import Input, Dense

from keras.models import Model

encoding\_dim = 32 # 32 floats -> compression of factor 24.5, assuming the input is 784 floats

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',metrics ='accuracy')

from keras.datasets import fashion\_mnist

import numpy as np

(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:])))

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)

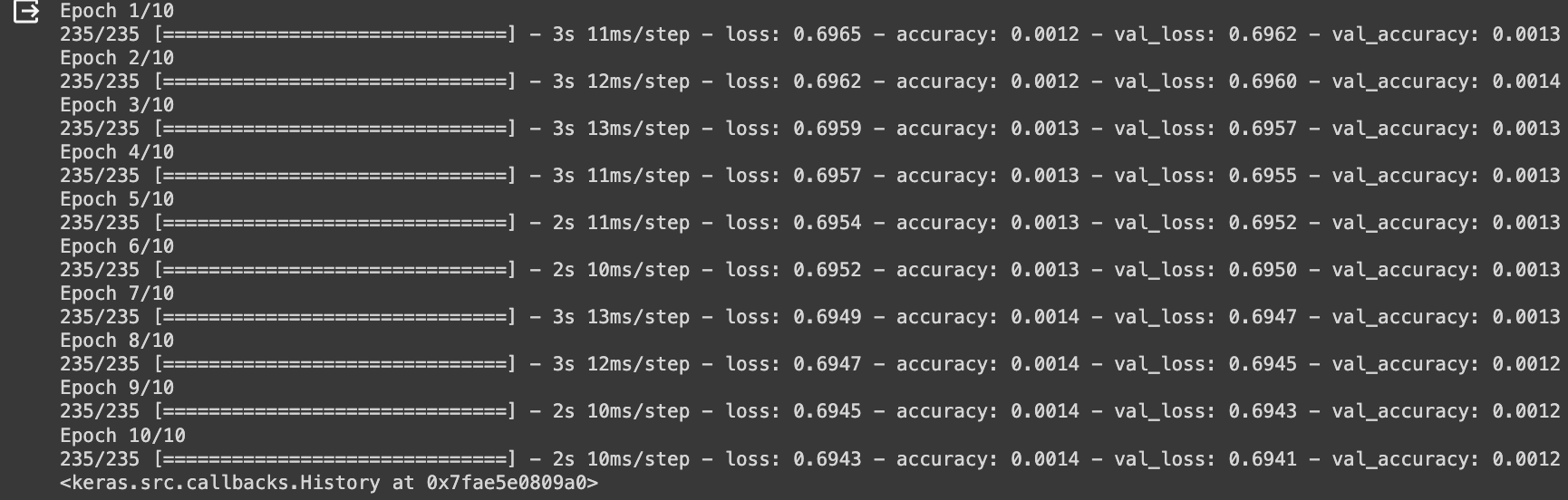
autoencoder.fit(x\_train\_noisy, x\_train,

epochs=10,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test\_noisy, x\_test\_noisy))



import matplotlib.pyplot as plt

# Get the reconstructed images for the test set

reconstructed\_imgs = autoencoder.predict(x\_test\_noisy)

# Choose a random image from the test set

n = 10 # index of the image to be plotted

plt.figure(figsize=(10, 5))

# Plot the original noisy image

ax = plt.subplot(1, 2, 1)

plt.imshow(x\_test\_noisy[n].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

ax.set\_title("Noisy Image")

# Plot the reconstructed image

ax = plt.subplot(1, 2, 2)

plt.imshow(reconstructed\_imgs[n].reshape(28, 28))

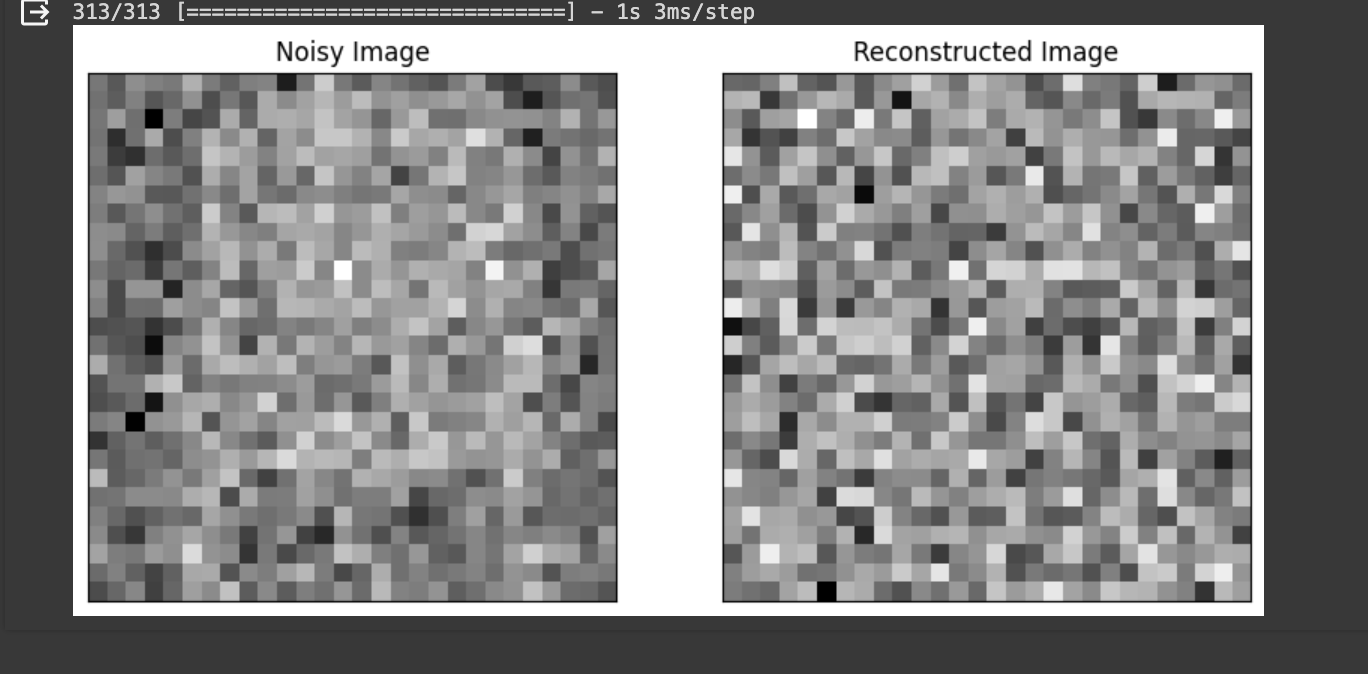
plt.gray()

ax.get\_xaxis().set\_visible(False)

ax.get\_yaxis().set\_visible(False)

ax.set\_title("Reconstructed Image")

plt.show()



import matplotlib.pyplot as plt

plt.show the autoencoder

history = autoencoder.fit(x\_train\_noisy, x\_train,

epochs=10,

batch\_size=256,

shuffle=True,

validation\_data=(x\_test\_noisy, x\_test\_noisy))

plt.plot(history.history['loss'], label='train')

plt.plot(history.history['val\_loss'], label='test')

plt.title('Model Loss')

plt.ylabel('Loss')

plt.xlabel('Epoch')

plt.legend()

plt.show()

plt.plot(history.history['accuracy'], label='train')

plt.plot(history.history['val\_accuracy'], label='test')

plt.title('Model Accuracy')

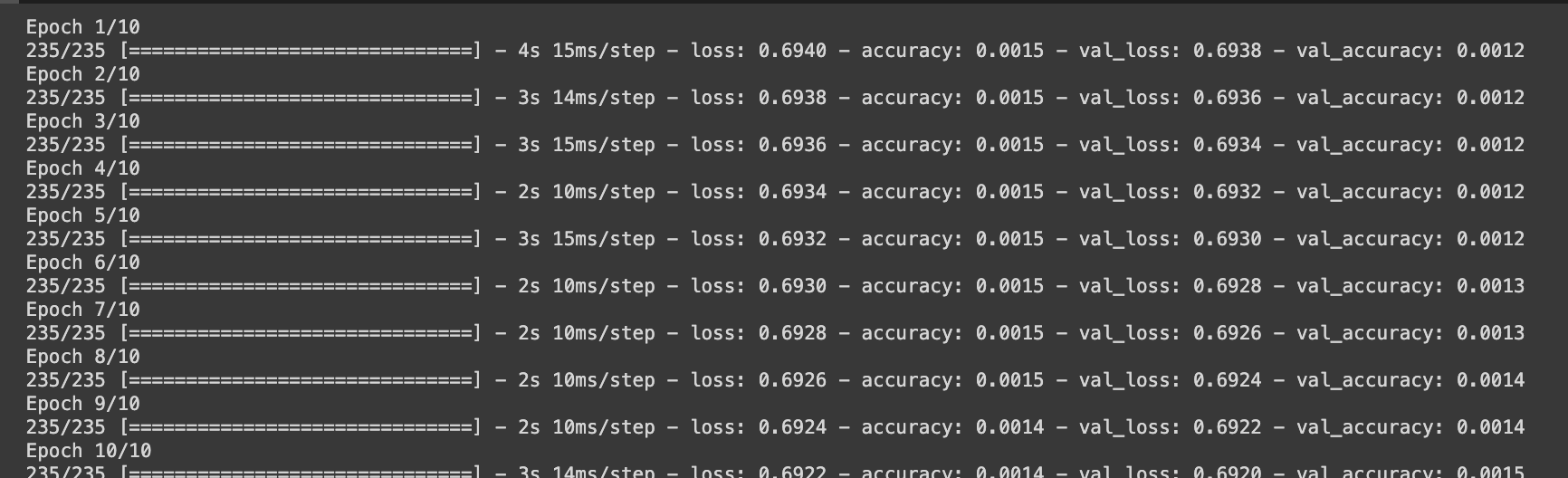
plt.ylabel('Accuracy')

plt.xlabel('Epoch')

plt.legend()

plt.show()

Output:



Test data using Matplotlib:

