**Assignment2**

*#installations*

**from** sklearn.preprocessing **import** LabelBinarizer

**from** sklearn.metrics **import** classification\_report

**from** tensorflow.keras.models **import** Sequential

**from** tensorflow.keras.layers **import** Dense

**from** tensorflow.keras.optimizers **import** SGD

**from** tensorflow.keras.datasets **import** mnist

**from** tensorflow.keras **import** backend **as** K

**import** matplotlib.pyplot **as** plt

**import** numpy **as** np

*#grabbing the mnist dataset*

((X\_train, Y\_train), (X\_test, Y\_test)) **=** mnist**.**load\_data()

X\_train **=** X\_train**.**reshape((X\_train**.**shape[0], 28 **\*** 28 **\*** 1))

X\_test **=** X\_test**.**reshape((X\_test**.**shape[0], 28 **\*** 28 **\*** 1))

X\_train **=** X\_train**.**astype("float32") **/** 255.0

X\_test **=** X\_test**.**astype("float32") **/** 255.0

lb **=** LabelBinarizer()

Y\_train **=** lb**.**fit\_transform(Y\_train)

Y\_test **=** lb**.**transform(Y\_test)

model **=** Sequential()

model**.**add(Dense(128, input\_shape**=**(784,), activation**=**"sigmoid"))

model**.**add(Dense(64, activation**=**"sigmoid"))

model**.**add(Dense(10, activation**=**"softmax"))

sgd **=** SGD(0.01)

epochs**=**10

model**.**compile(loss**=**"categorical\_crossentropy", optimizer**=**sgd,metrics**=**["accuracy"])

H **=** model**.**fit(X\_train, Y\_train, validation\_data**=**(X\_test, Y\_test),epochs**=**epochs, batch\_size**=**128)

*#making the predictions*

predictions **=** model**.**predict(X\_test, batch\_size**=**128)

print(classification\_report(Y\_test**.**argmax(axis**=**1),predictions**.**argmax(axis**=**1),target\_names**=**[str(x) **for** x **in** lb**.**classes\_]))

*#plotting the training loss and accuracy*

plt**.**style**.**use("ggplot")

plt**.**figure()

plt**.**plot(np**.**arange(0, epochs), H**.**history["loss"], label**=**"train\_loss")

plt**.**plot(np**.**arange(0, epochs), H**.**history["val\_loss"], label**=**"val\_loss")

plt**.**plot(np**.**arange(0, epochs), H**.**history["accuracy"], label**=**"train\_acc")

plt**.**plot(np**.**arange(0, epochs), H**.**history["val\_accuracy"], label**=**"val\_acc")

plt**.**title("Training Loss and Accuracy")

plt**.**xlabel("Epoch #")

plt**.**ylabel("Loss/Accuracy")

plt**.**legend()