

Sep 20, 22 19:39	hw3.py	Page 1/2
<pre> import os import numpy as np from matplotlib import pyplot as plt import matplotlib.image as mpimg from tensorflow.keras.utils import to_categorical from keras.models import Sequential from keras.layers import Dense, Activation, Dropout, Conv2D, MaxPooling2D from tensorflow.keras.optimizers import SGD import pandas as pd import tensorflow as tf from keras.layers.core import Flatten from sklearn.model_selection import train_test_split script_path = os.path.dirname(os.path.realpath(__file__)) """ function for loading csv and adjusting attributes for model use """ def load_data(path): df = pd.read_csv(path) df = df.values # load values np.random.shuffle(df) # shuffle dataset x = df[:, 1:].reshape(-1, 28, 28, 1) # reshape for model y = df[:, 0].astype(np.int32) y = tf.keras.utils.to_categorical(y, 10) # convert y_train to categorical by one-hot-encoding return x, y X_train, Y_train = load_data(script_path + "/mnist_train.csv") # loading X_test, Y_test = load_data(script_path + "/mnist_test.csv") X_train, X_valid, Y_train, Y_valid = train_test_split(X_train, Y_train, test_size=0.1) # splitting validation from train X = X_test X_train = X_train.astype("float32") # cast/convert type of array X_test = X_test.astype("float32") X_valid = X_valid.astype("float32") X_train /= 255 X_test /= 255 X_valid /= 255 # https://www.kaggle.com/code/cdeotte/how-to-choose-cnn-architecture-mnist/notebook # informed architecture choices model = Sequential() model.add(Conv2D(32, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same", input_shape=(28, 28, 1),)) model.add(</pre>		

Sep 20, 22 19:39	hw3.py	Page 2/2
<pre> Conv2D(64, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same", input_shape=(28, 28, 1),)) model.add(MaxPooling2D((2, 2))) model.add(Conv2D(64, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same")) model.add(Conv2D(64, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same")) model.add(MaxPooling2D((3, 3))) model.add(Conv2D(128, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same")) model.add(Conv2D(128, (3, 3), activation="relu", kernel_initializer="he_uniform", padding="same")) model.add(MaxPooling2D((3, 3))) # add flatten model.add(Flatten()) model.add(Dense(128, activation="relu", kernel_initializer="he_uniform")) model.add(Dense(10, activation="softmax")) model.compile(optimizer=SGD(learning_rate=0.01, momentum=0.9), loss="mse", metrics=["accuracy"]) # L2 # https://www.tensorflow.org/datasets/keras_example """ admit to changing hyperparameters based on performance mainly b/c i decided on using a more robust network, so i lowered the number of epochs for runtime """ history = model.fit(X_train, Y_train, epochs=2, batch_size=64, validation_data=(X_valid, Y_valid), verbose=1,) score = model.evaluate(X_test, Y_test, verbose=1) </pre>		

```
Epoch 1/2  
844/844 [=====] - 94s 110ms/step - loss: 0.0229 - accuracy: 0.8379 - val_loss: 0.0068 - val_accuracy: 0.9575  
Epoch 2/2  
844/844 [=====] - 90s 106ms/step - loss: 0.0051 - accuracy: 0.9671 - val_loss: 0.0041 - val_accuracy: 0.9742  
313/313 [=====] - 5s 17ms/step - loss: 0.0036 - accuracy: 0.9761
```