Derek Lee Deep Learning Fall 2020

Professor Curro Assignment #3

import struct as st

import numpy as np

import tensorflow as tf

import kerastuner as kt

from sklearn.model\_selection import train\_test\_split

import matplotlib.pyplot as plt

import os

# Constants

NUM\_EPOCHS = 30

VALIDATION\_PERCENTAGE = 0.2

DROP\_RATE = 0.2

# Image specific parameters

IMG\_DIMS = ( 28, 28, )

NUM\_CLASSES = 10

HIDDEN\_SIZE = [ 512, 512 ]  # Number of output neurons for each hidden layer

# Data processing inspired by:

# https://stackoverflow.com/questions/39969045/parsing-yann-lecuns-mnist-idx-file-format

def imgRead( fileName ):

    with open( fileName, 'rb') as f:

        # Go to beginning of file and start reading

        f.seek(0)

        \_numMagic = st.unpack( '>I', f.read(4) )[0]                 # Read magic number

        numImg, numRows, numCols = st.unpack( '>III', f.read(12) )  # Read ints

        numTotalBytes = numImg \* numRows \* numCols \* 1  # Each pixel is 1 byte

        # Read the remaining data in the file

        img = np.asarray( st.unpack(

                    '>' + 'B'\*numTotalBytes, f.read( numTotalBytes )

                    ) ).reshape( ( numImg, numRows, numCols ) )

    return img

def labelRead( fileName ):

    with open( fileName, 'rb') as f:

        # Go to beginning of file and start reading

        f.seek(0)

        \_numMagic = st.unpack( '>I', f.read(4) )[0]     # Read magic number

        numImg = st.unpack( '>I', f.read(4) )[0]        # Read number of examples

        # Read the remaining data in the file

        labels = st.unpack( '>' + 'B'\*numImg, f.read( numImg ) )

    return labels

def model\_builder( hp ):

    model = tf.keras.models.Sequential()

    # Input Layer

    model.add( tf.keras.layers.Flatten( input\_shape = IMG\_DIMS ) )

    # Hidden Layers

    for numNeurons in HIDDEN\_SIZE:

        model.add( tf.keras.layers.Dense( numNeurons, activation = 'relu', kernel\_regularizer = 'l2' ) )

        model.add( tf.keras.layers.Dropout( hp.Choice( 'rate', values = [ 0.3, 0.25, 0.2 ] ), trainable = False ) )

    # Output Layer

    model.add( tf.keras.layers.Dense( NUM\_CLASSES, activation = 'softmax' ) )

    # Compile model

    optimizer = tf.keras.optimizers.SGD(

                        hp.Choice( 'learning\_rate', values = [ 0.01, 0.005 ] ) )

    model.compile( loss = "sparse\_categorical\_crossentropy",

                        optimizer = optimizer, metrics = "accuracy" )

    return model

def plotAccuracy( history ):

    # Plots accuracy over time

    plt.figure( figsize = (10,5) )

    plt.plot( history.history['accuracy'] )

    plt.plot( history.history['val\_accuracy'] )

    plt.title( 'Model Accuracy' )

    plt.xlabel( 'Epochs' )

    plt.ylabel( 'Accuracy', rotation = 'horizontal', ha = 'right' )

    plt.legend( [ 'Train', 'Valid' ], loc = 'upper left' )

    plt.show()

def main():

    # Load data

    img = imgRead( 'trainImages.idx3-ubyte' )

    label = labelRead( 'trainLabels.idx1-ubyte' )

    testImg = imgRead( 'testImages.idx3-ubyte' )

    testLabel = labelRead( 'testLabels.idx1-ubyte' )

    trainImg, validImg, trainLabel, validLabel = train\_test\_split( img, label, test\_size = 0.2 )

    # Normalize and convert to tensor

    trainImg = tf.convert\_to\_tensor( trainImg / 255, dtype=tf.float32 )

    trainLabel = tf.convert\_to\_tensor( trainLabel )

    validImg = tf.convert\_to\_tensor( validImg / 255, dtype=tf.float32 )

    validLabel = tf.convert\_to\_tensor( validLabel )

    testImg = tf.convert\_to\_tensor( testImg / 255, dtype=tf.float32 )

    testLabel = tf.convert\_to\_tensor( testLabel )

    # Test if images were properly loaded

    plt.figure()

    plt.imshow( trainImg[0,:,:], cmap = 'gray' )

    plt.show()

    # Test if labels were properly loaded

    print( trainLabel[0] )

    # Initialize model

    tuner = kt.Hyperband(

        model\_builder,

        objective = "val\_accuracy",

        max\_epochs = 10,

        directory = os.path.normpath( 'D:/' ),

        project\_name = 'tunedParams'

    )

    tuner.search(

        trainImg, trainLabel, epochs = 10,

        validation\_data = (validImg, validLabel)

    )

    best\_hps = tuner.get\_best\_hyperparameters( num\_trials = 2 )[0]

    # Train and Test

    print( "\nStarted Training\n" )

    classifier = tuner.hypermodel.build( best\_hps )

    history = classifier.fit(

                        trainImg, trainLabel, epochs = NUM\_EPOCHS,

                        validation\_data = ( validImg, validLabel ) )

    print( "\nFinished Training\n" )

    classifier.evaluate( testImg, testLabel )

    # Display accuracy

    plotAccuracy( history )

main()

Training and Testing were performed using the default batch size of 32. The learning rate of SGD, along with the drop rate for each dropout layer was learned through the Keras Tuner.

The validation set was created with an 80/20 split of the original training set. Training set had 48,000 examples and validation set had 12,000 examples.

Hidden layers use L2 regularization with the default regularization parameter of 0.01.

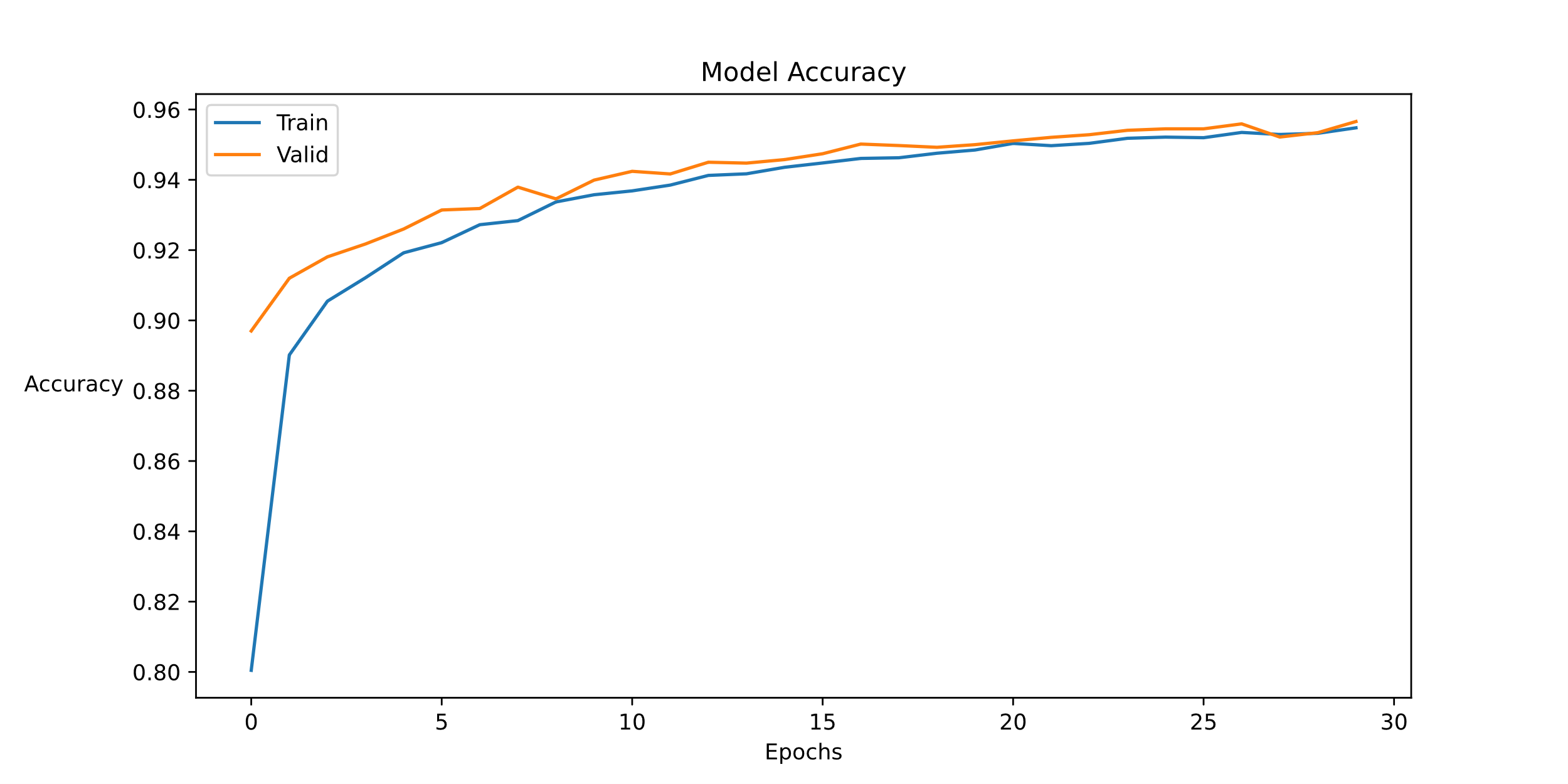


Figure 1: Example plot for the training and validation accuracy of the model during training

313/313 [==============================] - 0s 2ms/step - loss: 0.3282 - accuracy: 0.9598

Figure 2: Final accuracy on the test set