In [1]:

```
## Rebecca Lewis
## DSC 650
## Assignment 1.1
```

In [1]:

```
'''Trains a simple deep NN on the MNIST dataset.

Gets to 98.40% test accuracy after 20 epochs
(there is *a lot* of margin for parameter tuning).
2 seconds per epoch on a K520 GPU.
'''

from tensorflow import keras
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.optimizers import RMSprop

batch_size = 128
num_classes = 10
epochs = 20

# the data, split between train and test sets
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255
print(x_train.shape[0], 'train samples')
print(x_test.shape[0], 'test samples')

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)

model = Sequential()
model.add(Dense(512, activation='relu', input_shape=(784,)))
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

model.compile(loss='categorical_crossentropy',
              optimizer=RMSprop(),
              metrics=['accuracy'])

history = model.fit(x_train, y_train,
                    batch_size=batch_size,
                    epochs=epochs,
                    verbose=1,
                    validation_data=(x_test, y_test))
score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-data
sets/mnist.npz (https://storage.googleapis.com/tensorflow/tf-keras-datasets/m
nist.npz)
11493376/11490434 [==============================] - 1s 0us/step
60000 train samples
10000 test samples
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 512)               401920    
_________________________________________________________________
dropout (Dropout)            (None, 512)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               262656    
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130      
=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 5s 10ms/step - loss: 0.2471 - accu
racy: 0.9237 - val_loss: 0.1100 - val_accuracy: 0.9650
Epoch 2/20
469/469 [==============================] - 5s 10ms/step - loss: 0.1022 - accu
racy: 0.9689 - val_loss: 0.0836 - val_accuracy: 0.9728
Epoch 3/20
469/469 [==============================] - 4s 10ms/step - loss: 0.0742 - accu
racy: 0.9774 - val_loss: 0.0774 - val_accuracy: 0.9773
Epoch 4/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0609 - accur
acy: 0.9813 - val_loss: 0.0752 - val_accuracy: 0.9786
Epoch 5/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0512 - accur
acy: 0.9838 - val_loss: 0.0715 - val_accuracy: 0.9803
Epoch 6/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0423 - accur
acy: 0.9876 - val_loss: 0.0830 - val_accuracy: 0.9794
Epoch 7/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0395 - accur
acy: 0.9883 - val_loss: 0.0844 - val_accuracy: 0.9800
Epoch 8/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0325 - accur
acy: 0.9902 - val_loss: 0.0946 - val_accuracy: 0.9812
Epoch 9/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0299 - accur
acy: 0.9909 - val_loss: 0.0821 - val_accuracy: 0.9819
Epoch 10/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0274 - accur
acy: 0.9914 - val_loss: 0.0821 - val_accuracy: 0.9832
Epoch 11/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0250 - accur
acy: 0.9926 - val_loss: 0.0939 - val_accuracy: 0.9823
```

```
Epoch 12/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0264 - accur
acy: 0.9930 - val_loss: 0.0878 - val_accuracy: 0.9834
Epoch 13/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0224 - accur
acy: 0.9935 - val_loss: 0.0903 - val_accuracy: 0.9849
Epoch 14/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0227 - accur
acy: 0.9933 - val_loss: 0.0980 - val_accuracy: 0.9832
Epoch 15/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0190 - accur
acy: 0.9944 - val_loss: 0.1142 - val_accuracy: 0.9827
Epoch 16/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0210 - accur
acy: 0.9944 - val_loss: 0.0944 - val_accuracy: 0.9837
Epoch 17/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0166 - accur
acy: 0.9949 - val_loss: 0.1246 - val_accuracy: 0.9834
Epoch 18/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0170 - accur
acy: 0.9948 - val_loss: 0.1206 - val_accuracy: 0.9827
Epoch 19/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0152 - accur
acy: 0.9956 - val_loss: 0.1246 - val_accuracy: 0.9830
Epoch 20/20
469/469 [==============================] - 4s 9ms/step - loss: 0.0179 - accur
acy: 0.9953 - val_loss: 0.1278 - val_accuracy: 0.9841
Test loss: 0.12781260907649994
Test accuracy: 0.9840999841690063
```

In [9]:

```
#
# Licensed to the Apache Software Foundation (ASF) under one or more
# contributor license agreements.  See the NOTICE file distributed with
# this work for additional information regarding copyright ownership.
# The ASF licenses this file to You under the Apache License, Version 2.0
# (the "License"); you may not use this file except in compliance with
# the License.  You may obtain a copy of the License at
#
#    http://www.apache.org/licenses/LICENSE-2.0
#
# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

import sys
from random import random
from operator import add

from pyspark.sql import SparkSession


if __name__ == "__main__":
    """
        Usage: pi [partitions]
    """
    spark = SparkSession\
        .builder\
        .appName("PythonPi")\
        .getOrCreate()

    #receiving a base 10 error with original code, added a base of 16 but was get
    #for a negative value.  Applied the absolute value and was able to run succes
    partitions = abs(int(sys.argv[1], base=16)) if len(sys.argv) > 1 else 2
    n = 100000 * partitions

    def f(_):
        x = random() * 2 - 1
        y = random() * 2 - 1
        return 1 if x ** 2 + y ** 2 <= 1 else 0

    count = spark.sparkContext.parallelize(range(1, n + 1), partitions).map(f).re
    print("Pi is roughly %f" % (4.0 * count / n))

    spark.stop()
```

```
Pi is roughly 3.138987
```

In [7]:

```
print (sys.argv)
```

```
['/opt/conda/lib/python3.8/site-packages/ipykernel_launcher.py', '-f', '/home/j
ovyan/.local/share/jupyter/runtime/kernel-5dabe6bd-6159-4d02-af0e-b7446b39bf31.
json']
```

In [ ]: