Open in Colab

In [2]:
```
!git clone https://github.com/bellevue-university/dsc650.git
```

```
Cloning into 'dsc650'...
remote: Enumerating objects: 120326, done.
remote: Counting objects: 100% (128/128), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 120326 (delta 57), reused 98 (delta 46), pack-reused 120198
Receiving objects: 100% (120326/120326), 360.60 MiB | 18.21 MiB/s, done.
Resolving deltas: 100% (7340/7340), done.
Updating files: 100% (114699/114699), done.
```

In [30]:
```
from datetime import datetime

with open('/content/dsc650/dsc650/assignments/assignment01/logs/keras-mnist.log', 'a')
  from tensorflow import keras
  from keras.datasets import mnist
  from keras.models import Sequential
  from keras.layers import Dense, Dropout
  from 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
  f.write(str(x_train.shape[0]))
  f.write(str(x_test.shape[0]))

  # 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))
f.write(str(history))
score = model.evaluate(x_test, y_test, verbose=0)
f.write (datetime.now().strftime('%Y-%m-%d %H:%M:%S') + "\n" )
f.write('Test loss :' +  str(score[0]) + "\n")
f.write('Test Accuracy :' + str(score[1]) + "\n")
```

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_27 (Dense)            (None, 512)               401920

 dropout_18 (Dropout)        (None, 512)               0

 dense_28 (Dense)            (None, 512)               262656

 dropout_19 (Dropout)        (None, 512)               0

 dense_29 (Dense)            (None, 10)                5130

=================================================================
Total params: 669,706
Trainable params: 669,706
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
469/469 [==============================] - 29s 4ms/step - loss: 0.2414 - accuracy: 0.
9254 - val_loss: 0.1061 - val_accuracy: 0.9680
Epoch 2/20
469/469 [==============================] - 2s 4ms/step - loss: 0.1014 - accuracy: 0.9
690 - val_loss: 0.0846 - val_accuracy: 0.9741
Epoch 3/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0738 - accuracy: 0.9
776 - val_loss: 0.0692 - val_accuracy: 0.9805
Epoch 4/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0608 - accuracy: 0.9
820 - val_loss: 0.0750 - val_accuracy: 0.9779
Epoch 5/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0497 - accuracy: 0.9
849 - val_loss: 0.0663 - val_accuracy: 0.9825
Epoch 6/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0426 - accuracy: 0.9
872 - val_loss: 0.0788 - val_accuracy: 0.9812
Epoch 7/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0386 - accuracy: 0.9
888 - val_loss: 0.0880 - val_accuracy: 0.9789
Epoch 8/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0337 - accuracy: 0.9
898 - val_loss: 0.0837 - val_accuracy: 0.9817
Epoch 9/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0295 - accuracy: 0.9
908 - val_loss: 0.0974 - val_accuracy: 0.9815
Epoch 10/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0284 - accuracy: 0.9
918 - val_loss: 0.0892 - val_accuracy: 0.9827
Epoch 11/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0253 - accuracy: 0.9
928 - val_loss: 0.0973 - val_accuracy: 0.9821
Epoch 12/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0236 - accuracy: 0.9
927 - val_loss: 0.1039 - val_accuracy: 0.9816
Epoch 13/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0227 - accuracy: 0.9
932 - val_loss: 0.0953 - val_accuracy: 0.9827
Epoch 14/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0222 - accuracy: 0.9
```

```
936 - val_loss: 0.0934 - val_accuracy: 0.9851
Epoch 15/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0216 - accuracy: 0.9
942 - val_loss: 0.1037 - val_accuracy: 0.9829
Epoch 16/20
469/469 [==============================] - 2s 5ms/step - loss: 0.0198 - accuracy: 0.9
944 - val_loss: 0.0998 - val_accuracy: 0.9846
Epoch 17/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0185 - accuracy: 0.9
948 - val_loss: 0.1135 - val_accuracy: 0.9842
Epoch 18/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0183 - accuracy: 0.9
951 - val_loss: 0.1177 - val_accuracy: 0.9829
Epoch 19/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0169 - accuracy: 0.9
955 - val_loss: 0.1177 - val_accuracy: 0.9843
Epoch 20/20
469/469 [==============================] - 2s 4ms/step - loss: 0.0159 - accuracy: 0.9
957 - val_loss: 0.1133 - val_accuracy: 0.9824
```

In [31]:
```
!pip install pyspark
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/p
ublic/simple/
Collecting pyspark
  Downloading pyspark-3.3.2.tar.gz (281.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 281.4/281.4 MB 5.7 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting py4j==0.10.9.5
  Downloading py4j-0.10.9.5-py2.py3-none-any.whl (199 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 199.7/199.7 KB 24.2 MB/s eta 0:00:00
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.3.2-py2.py3-none-any.whl size=2818240
28 sha256=28f8f1c14a74f7caef53ba87aeeb49ae74da0580bba5c1bcfd86b6de750ff0f5
  Stored in directory: /root/.cache/pip/wheels/6c/e3/9b/0525ce8a69478916513509d436935
11463c6468db0de237c86
Successfully built pyspark
Installing collected packages: py4j, pyspark
  Attempting uninstall: py4j
    Found existing installation: py4j 0.10.9.7
    Uninstalling py4j-0.10.9.7:
      Successfully uninstalled py4j-0.10.9.7
Successfully installed py4j-0.10.9.5 pyspark-3.3.2
```

In [32]:
```
from datetime import datetime
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()

    partitions =  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).reduce(
    print("Pi is roughly %f" % (4.0 * count / n))

    spark.stop()
    with open('/content/dsc650/dsc650/assignments/assignment01/logs/spark-pi.log', 'a'
      f.write (datetime.now().strftime('%Y-%m-%d %H:%M:%S') + "\n" )
      f.write ('Pi is :' +  str((4.0 * count / n)) + "\n")
```

```
Pi is roughly 3.139320
```

In [ ]: