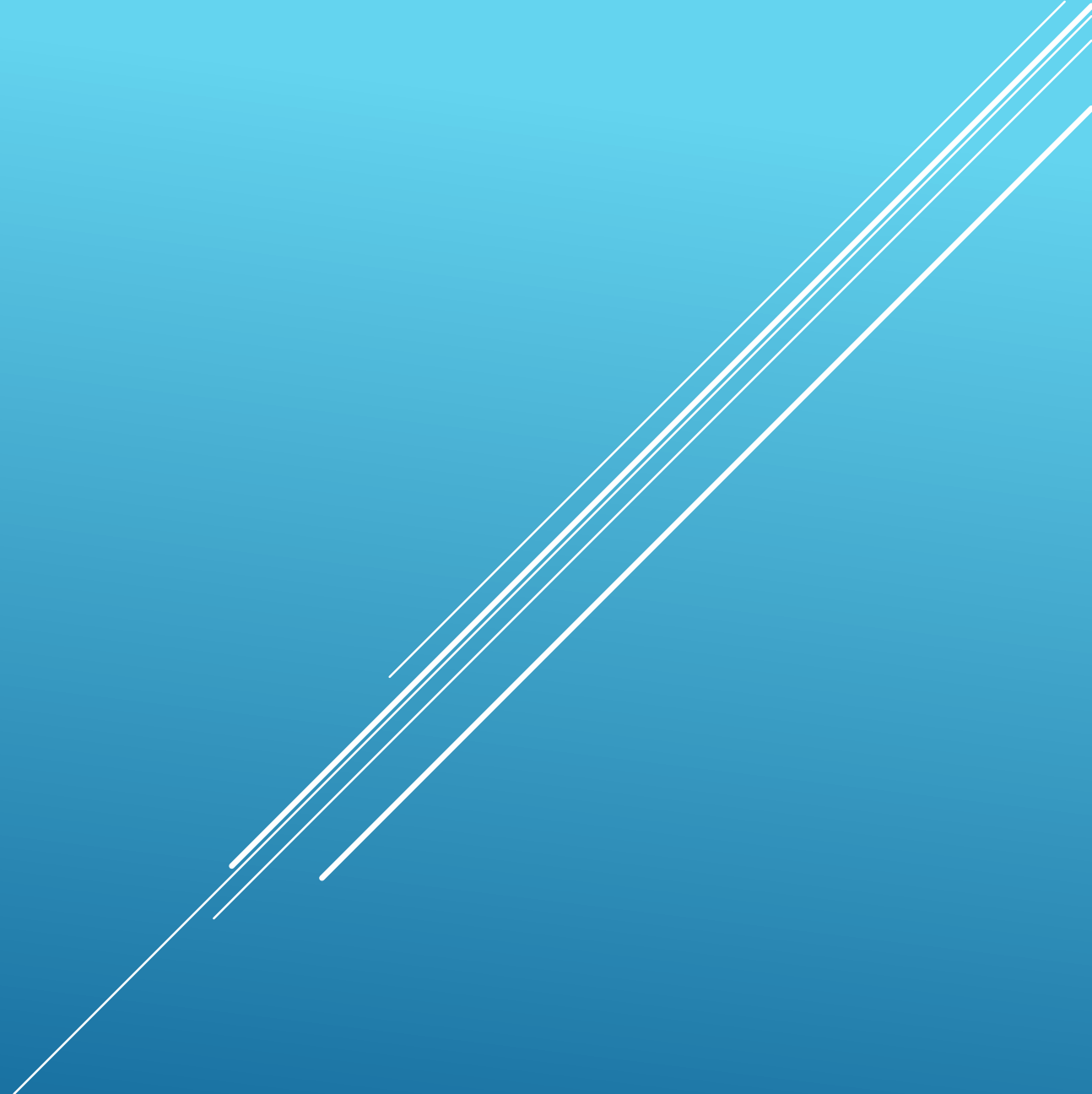


# DEEP LEARNING

TENSORFLOW



# OVERVIEW

## ► MNIST Dataset

- MNIST is a labeled set of images of handwritten digits.
- Google's AI tool TensorFlow is used in this project to train the traditional dataset MNIST where the model looks at images and predict what digits they are.

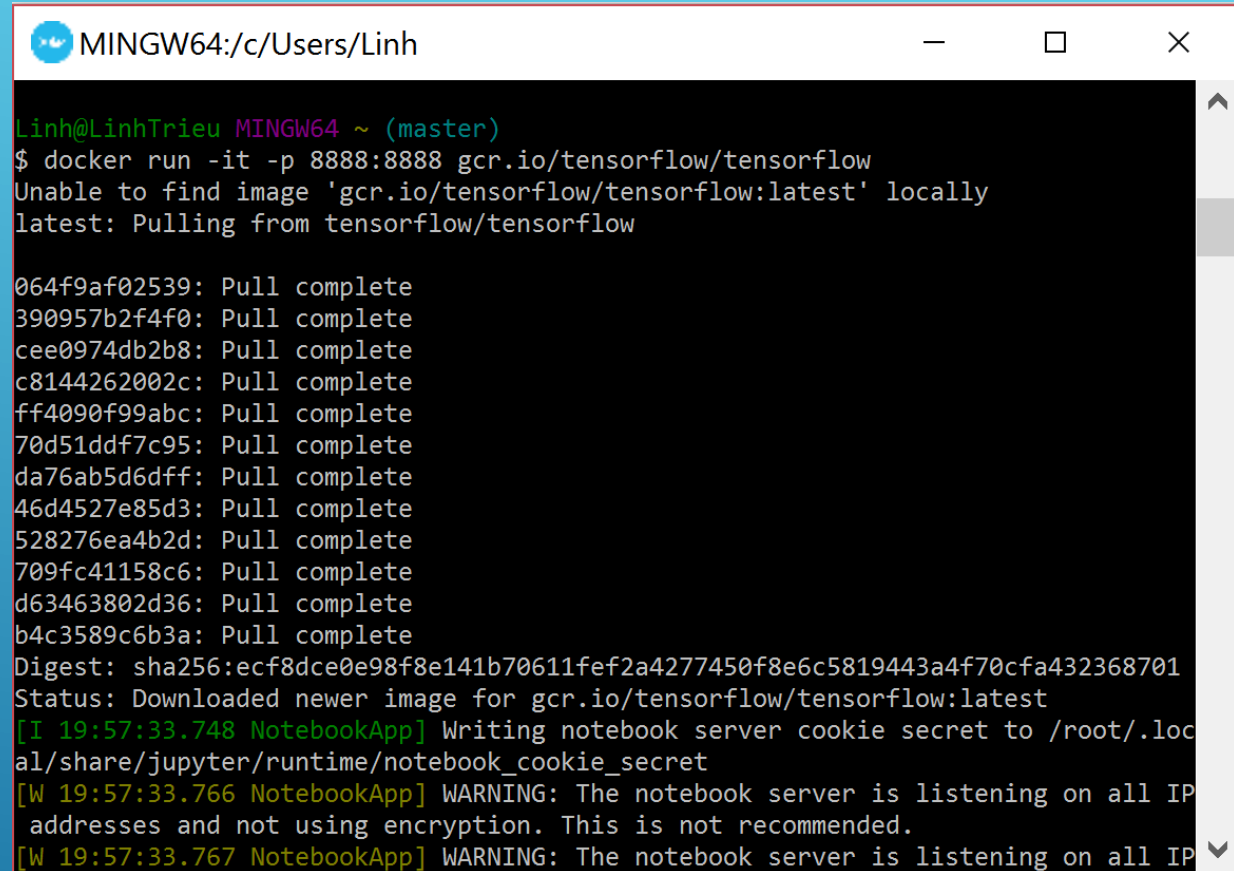
## ► TensorFlow

This presentation is not necessarily building the best model, but rather showing my first training model leveraging powerful tool TensorFlow directly on my computer.

Several white lines of varying lengths and angles are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.

# DOCKER

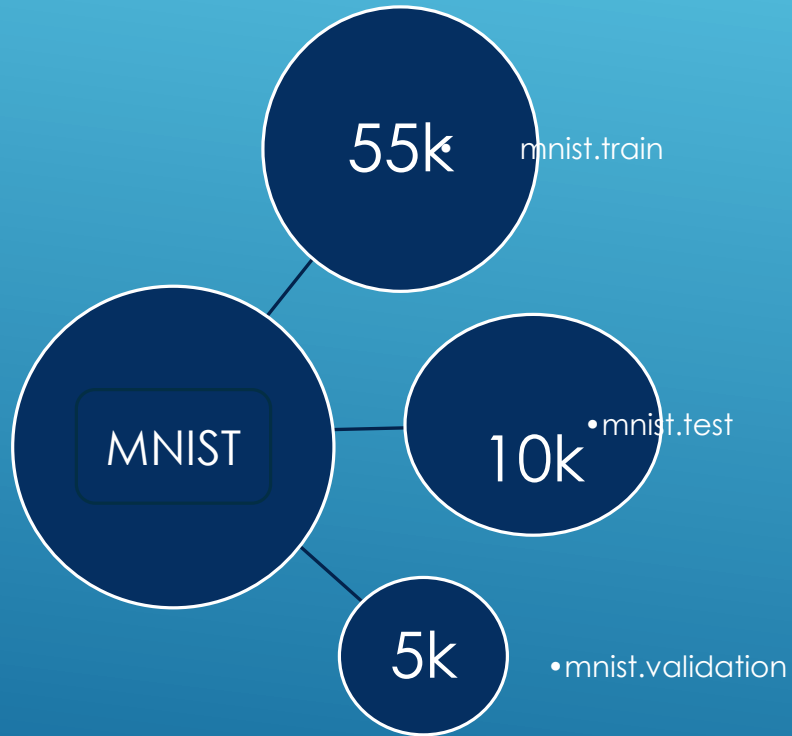
On Windows environment, I installed Docker, launched docker container, and ran Ipython Notebook



```
MINGW64:/c/Users/Linh
Linh@LinhTrieu MINGW64 ~ (master)
$ docker run -it -p 8888:8888 gcr.io/tensorflow/tensorflow
Unable to find image 'gcr.io/tensorflow/tensorflow:latest' locally
latest: Pulling from tensorflow/tensorflow

064f9af02539: Pull complete
390957b2f4f0: Pull complete
cee0974db2b8: Pull complete
c8144262002c: Pull complete
ff4090f99abc: Pull complete
70d51ddf7c95: Pull complete
da76ab5d6dff: Pull complete
46d4527e85d3: Pull complete
528276ea4b2d: Pull complete
709fc41158c6: Pull complete
d63463802d36: Pull complete
b4c3589c6b3a: Pull complete
Digest: sha256:ecf8dce0e98f8e141b70611fef2a4277450f8e6c5819443a4f70cfa432368701
Status: Downloaded newer image for gcr.io/tensorflow/tensorflow:latest
[I 19:57:33.748 NotebookApp] Writing notebook server cookie secret to /root/.local/share/jupyter/runtime/notebook_cookie_secret
[W 19:57:33.766 NotebookApp] WARNING: The notebook server is listening on all IP addresses and not using encryption. This is not recommended.
[W 19:57:33.767 NotebookApp] WARNING: The notebook server is listening on all IP
```

# LOAD DATASET MNIST



- Imported tensorflow into ipynb
- Loaded dataset as mnist

```
In [1]: from __future__ import absolute_import
        from __future__ import division
        from __future__ import print_function
        import argparse
```

```
In [54]: from tensorflow.examples.tutorials.mnist import input_data
        mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)
```

```
import tensorflow as tf
```

```
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting MNIST_data/train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

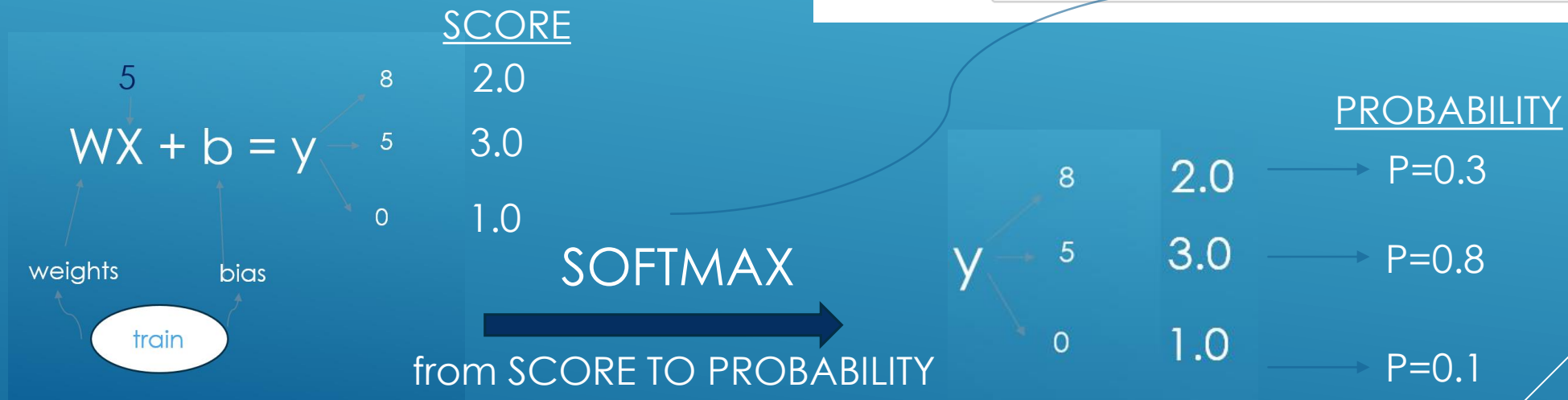
# SOFTMAX

- Flattened array into a vector of  $28 \times 28 = 784$  numbers
- Each image has 28 pixels by 28 pixels.

- Declared variables to the model for recognizing digits by looking at every pixel in the image

```
In [56]: # Create the model
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

- Model function:



# ONE-HOT ENCODING

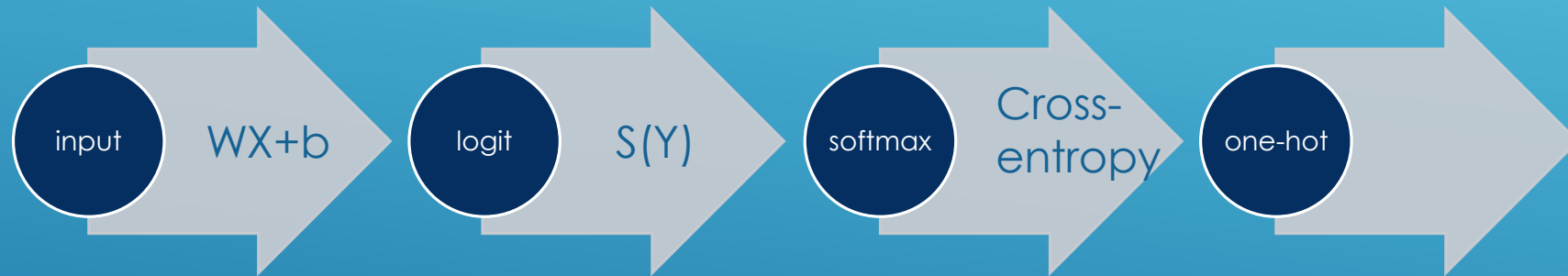
- Set the probability of the correct class ( $P=0.8$ ) be close to 1
- For all others, be close to 0



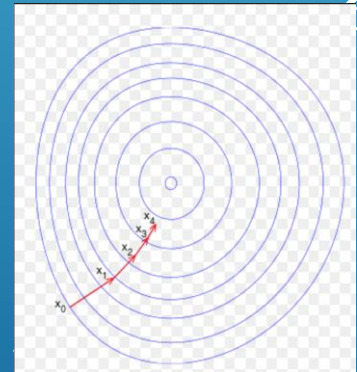
# CROSS-ENTROPY

```
In [57]: # Define loss and optimizer
y_ = tf.placeholder(tf.float32, [None, 10])
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

So far, model looks like this



- The step above is called Multinomial Logistic Classification
- Cross-entropy function: distance  $D(S(WX+b), L)$
- Loss is the average of cross-entropy, which is lots of sums and matrices
- Gradient descent algorithm is chosen to find the least loss with learning rate at 0.5



# SCHOLASTIC GRADIENT DESCENT

```
In [59]: init = tf.initialize_all_variables()
sess = tf.Session()
sess.run(init)
for i in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

- As loss function is very huge and dependent on all single element in the training set, it uses lots of compute to train the data.
- Instead, batch included 100 random data points was used, many times, to increase efficiency.
- TensorFlow session was initialized



# MODEL EVALUATION

```
In [60]: correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))  
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))  
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))  
  
0.9092
```

First time: 90.9% accuracy

```
In [63]: correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))  
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))  
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))  
  
0.9189
```

Second time: 91.9% accuracy

```
In [79]: correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))  
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))  
print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))  
  
0.9224
```

Third time: 92.2% accuracy

# REFERENCE

This project follows instructions on [www.tensorflow.com](http://www.tensorflow.com)

Linh Trieu  
Advanced Business Intelligence  
The University of Texas at Dallas

A series of white lines of varying lengths and orientations are positioned in the bottom right corner of the slide, creating a modern, abstract graphic element.