ACENET
accelerate discovery

Tensorflow and Keras

with

Daniel L. Silver, Ph.D.

Andy McIntyre, Ph.D.

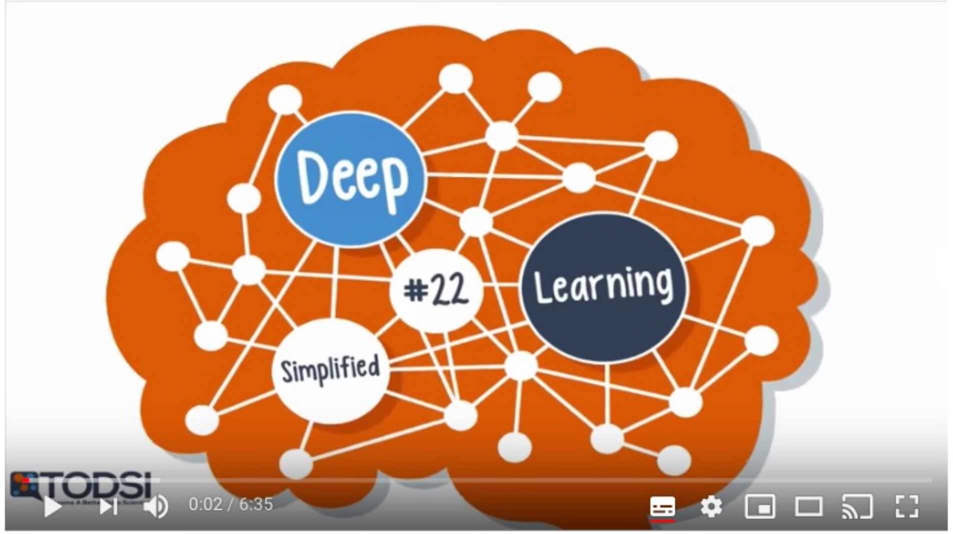
June 24, 2022

ACADIA INSTITUTE
FOR DATA ANALYTICS
digital harvest

What is tensorflow?

TensorFlow is:

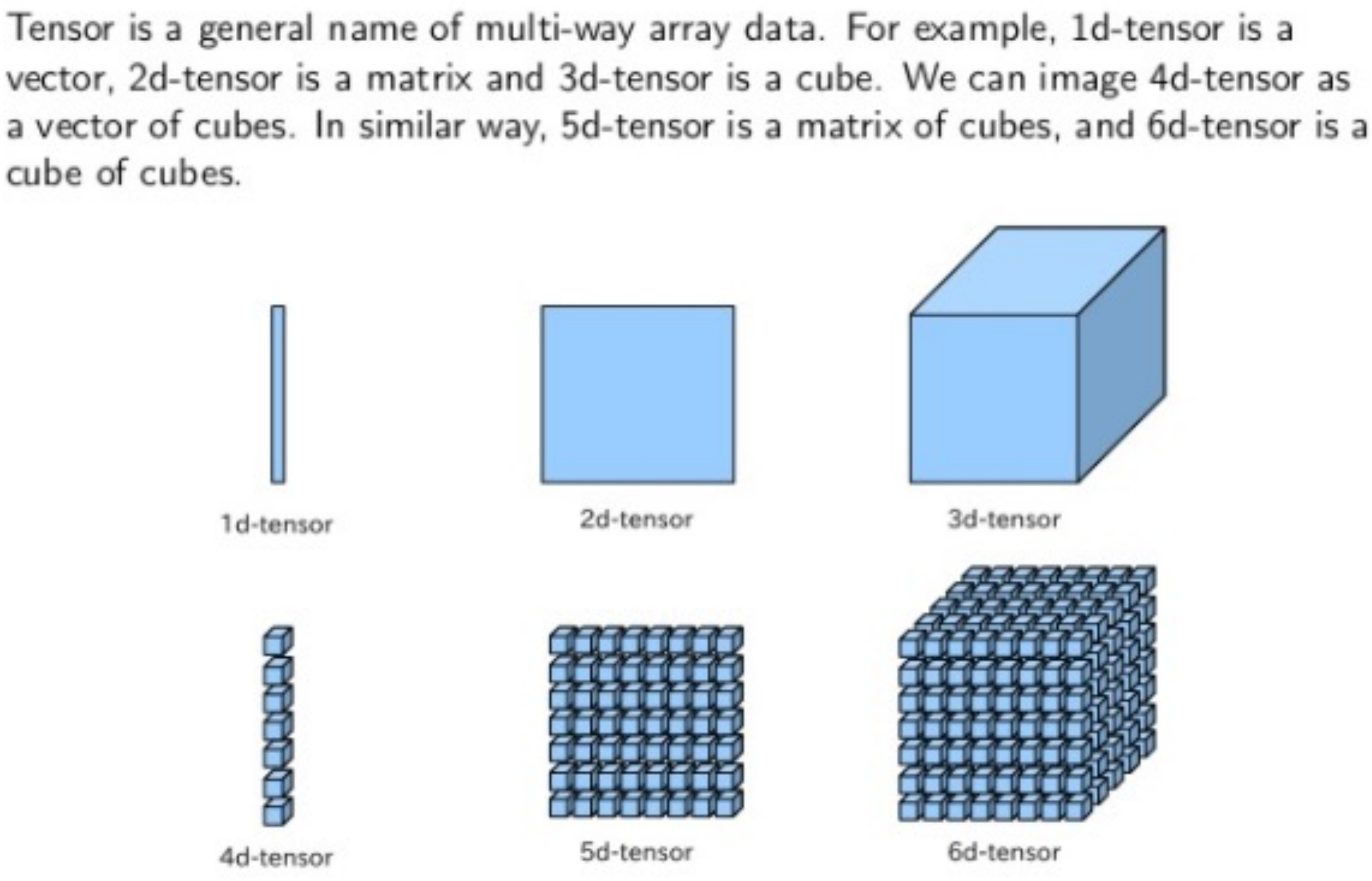
- an interface for expressing machine learning algorithms, and
- an implementation for executing such algorithms.



- symbolic ML dataflow framework that compiles to native CPU -or- fast GPU code
- offers a reduction in development time

- https://www.youtube.com/watch?v=bYeBL92v99Y

What is a tensor?



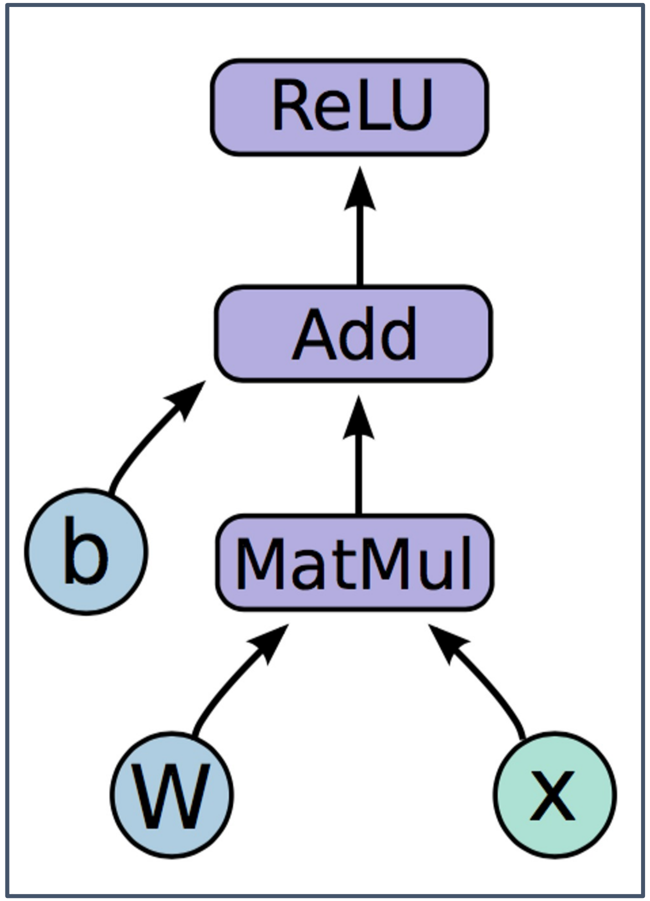
Tensor is a general name of multi-way array data. For example, 1d-tensor is a vector, 2d-tensor is a matrix and 3d-tensor is a cube. We can image 4d-tensor as a vector of cubes. In similar way, 5d-tensor is a matrix of cubes, and 6d-tensor is a cube of cubes.

Tensorflow - Programming model

Expresses a numeric computation as a **graph**.

- Graph nodes are **operations** which have any number of inputs and outputs
- Graph edges are **tensors** which flow between nodes

$$h_i = \mathrm{ReLU}(Wx + b)$$

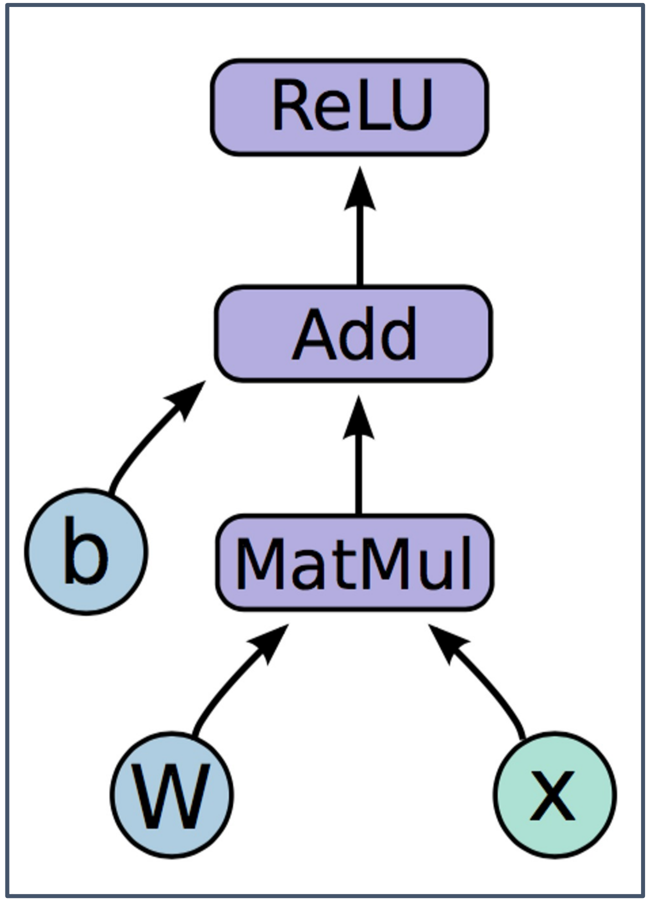


Tensorflow - Programming model

$$h_i = \mathrm{ReLU}(Wx + b)$$

Basic Flow:

1. Build a graph
   a. Contains parameter specifications, model architecture, optimization process, …
2. Define and initialize a session
3. Compile and run a session
   a. Compilation, optimization on CPU or GPU on different operating systems



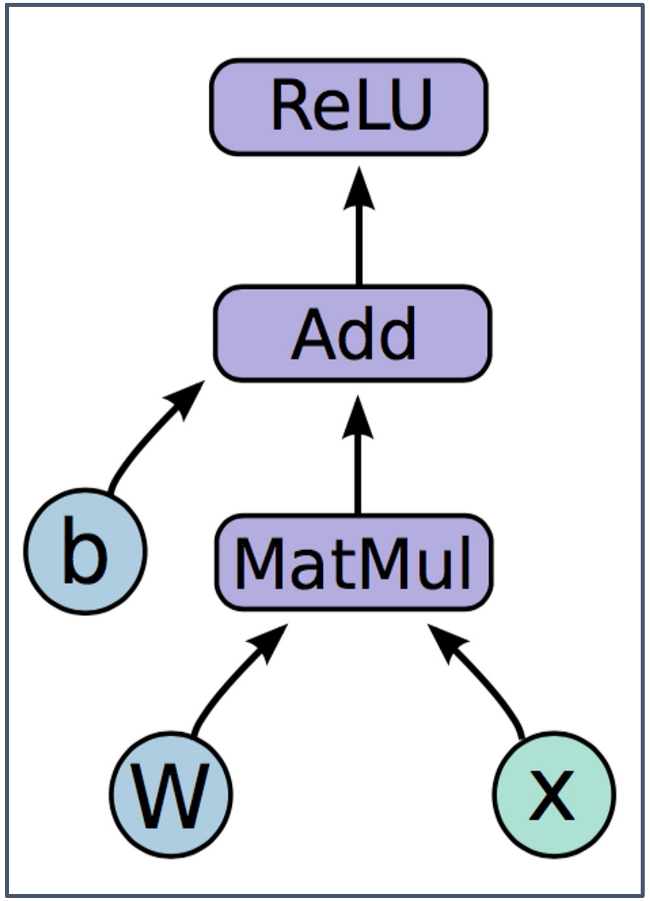
Tensorflow - Programming model

```
import tensorflow as tf
```

$$h_i = \mathrm{ReLU}(Wx + b)$$

1

```
b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784,
100),-1, 1))
x = tf.placeholder(tf.float32, (None, 784))
h_i = tf.nn.relu(tf.matmul(x, W) + b)
```



2

```
sess = tf.Session()
sess.run(tf.initialize_all_variables())
```

3

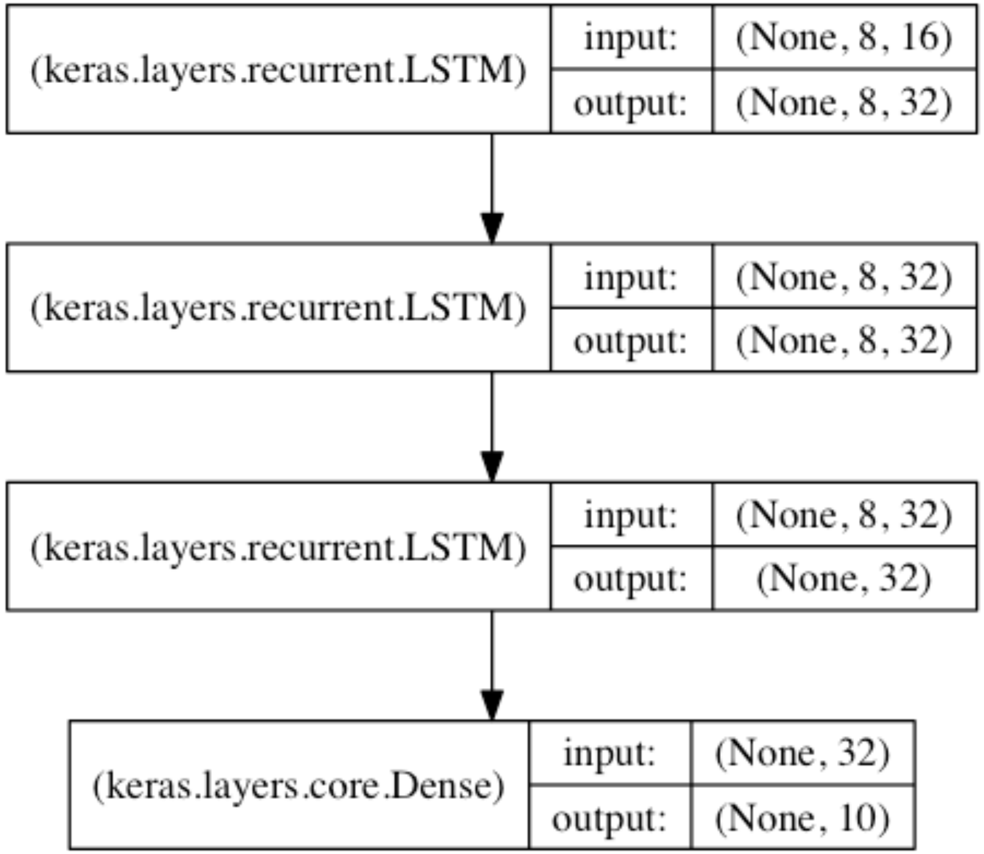
```
sess.run(h_i, {x: np.random.random(64,
784)})
```

What is Keras?

- A high-level API to build and train deep learning models.
- Used for fast prototyping, advanced research, and production, with three key advantages:
  - ***User friendly*** - simple, consistent interface optimized for common use cases. It provides clear and actionable feedback for user errors.
  - ***Modular and composable*** - Keras models are made by connecting configurable building blocks together, with few restrictions.
  - ***Easy to extend*** - *w*rite custom building blocks to express new ideas for research, or create new layers, loss functions, and develop new models
- http://Keras.io

What is Keras?

- Framework on top of TensorFlow
- Follows the principle of layers – can stack, split or merge for unique network architectures.
- Calculates the connection size between hidden layers based on each layers size.
- Allows GPU acceleration with minimal configuration.
- http://Keras.io



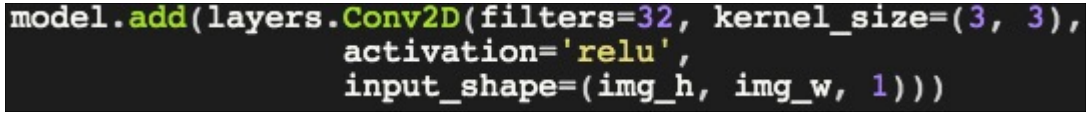
(keras.layers.recurrent.LSTM) input: (None, 8, 16) output: (None, 8, 32)

(keras.layers.recurrent.LSTM) input: (None, 8, 32) output: (None, 8, 32)

(keras.layers.recurrent.LSTM) input: (None, 8, 32) output: (None, 32)

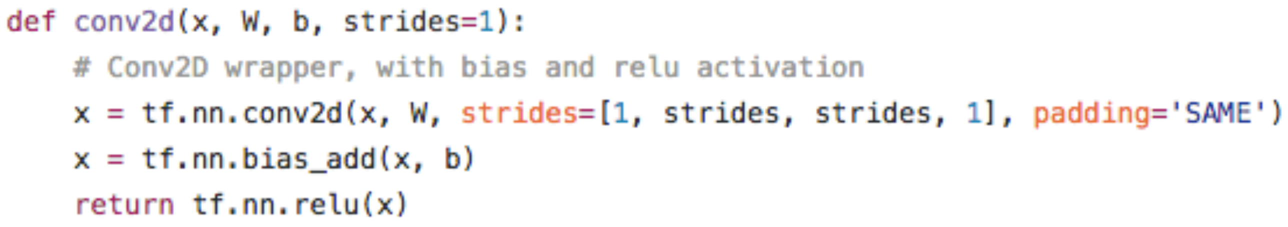
(keras.layers.core.Dense) input: (None, 32) output: (None, 10)

Keras

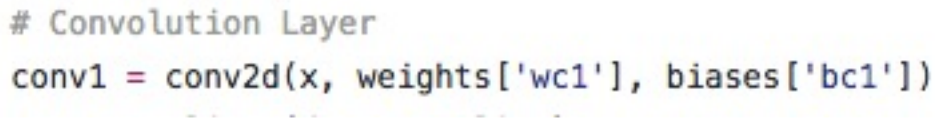


```
model.add(layers.Conv2D(filters=32, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=(img_h, img_w, 1)))
```

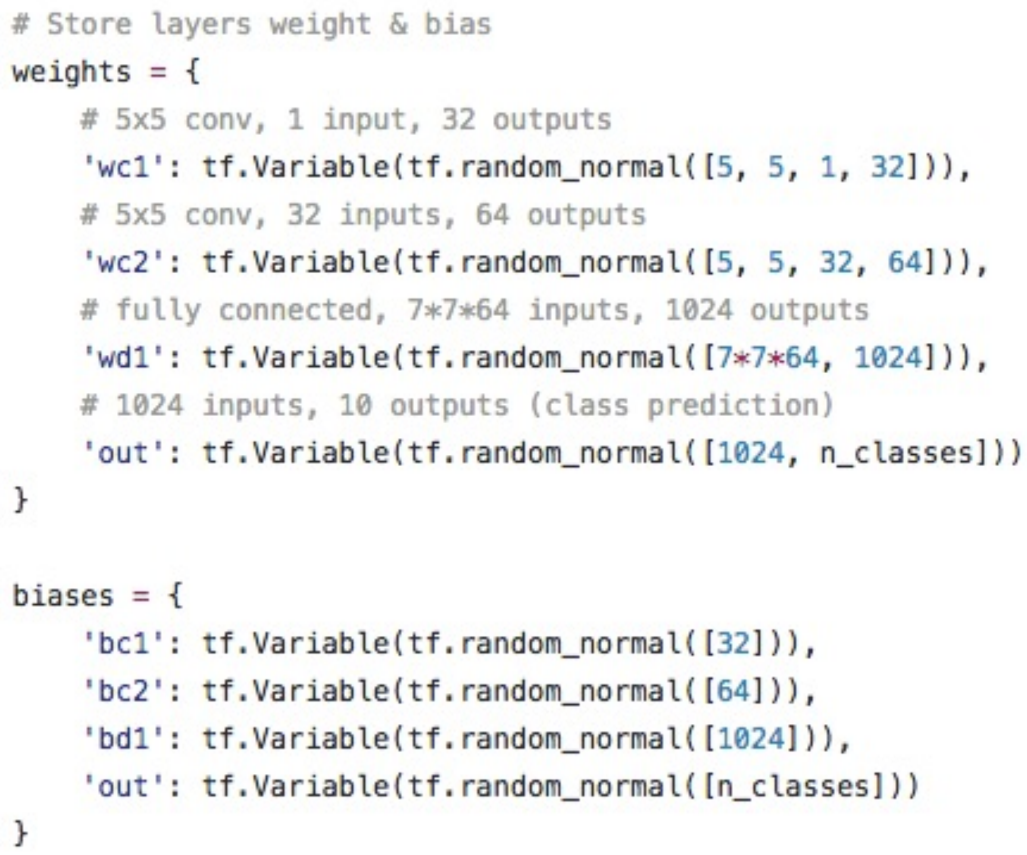
TensorFlow



```
def conv2d(x, W, b, strides=1):
    # Conv2D wrapper, with bias and relu activation
    x = tf.nn.conv2d(x, W, strides=[1, strides, strides, 1], padding='SAME')
    x = tf.nn.bias_add(x, b)
    return tf.nn.relu(x)
```



```
# Convolution Layer
conv1 = conv2d(x, weights['wc1'], biases['bc1'])
```



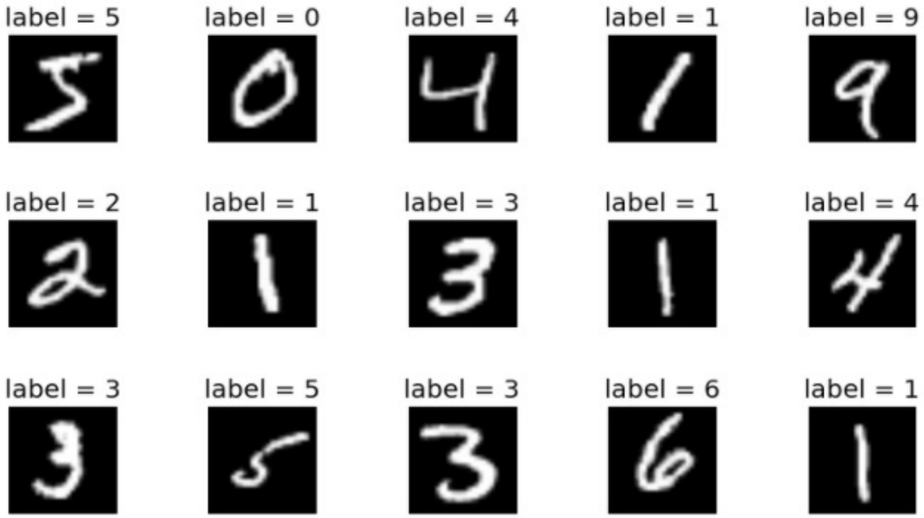
```
# Store layers weight & bias
weights = {
    # 5x5 conv, 1 input, 32 outputs
    'wc1': tf.Variable(tf.random_normal([5, 5, 1, 32])),
    # 5x5 conv, 32 inputs, 64 outputs
    'wc2': tf.Variable(tf.random_normal([5, 5, 32, 64])),
    # fully connected, 7*7*64 inputs, 1024 outputs
    'wd1': tf.Variable(tf.random_normal([7*7*64, 1024])),
    # 1024 inputs, 10 outputs (class prediction)
    'out': tf.Variable(tf.random_normal([1024, n_classes]))
}

biases = {
    'bc1': tf.Variable(tf.random_normal([32])),
    'bc2': tf.Variable(tf.random_normal([64])),
    'bd1': tf.Variable(tf.random_normal([1024])),
    'out': tf.Variable(tf.random_normal([n_classes]))
}
```

TUTORIAL #1(c)

- Let’s try learning a more challenging problem using Tensorflow and Keras (tf.keras_mnist_sigmoid_val.ipynb)

Problem: Classify the MNIST data set examples.



References

- https://www.tensorflow.org/tutorials
- https://www.tensorflow.org/api_docs/python/tf/keras
- https://opensource.google/projects/tensorflow-playground

- https://www.datacamp.com/courses/deep-learning-in-python
- https://developers.google.com/machine-learning/crash-course/ml-intro
- https://www.coursera.org/specializations/deep-learning