Introduction to Keras

3.2 Keras

- Keras is a deep-learning framework for Python
  - It provides a convenient way to define and train almost any kind of deep-learning model
- Keras was initially developed for researchers
  - with the aim of enabling fast experimentation
- “I started Keras for my own use, pretty much.” - François Chollet.
- François is currently working as an AI Researcher at Google
  - he’s also at the core of Keras Development

“I definitely didn’t expect it to grow so big. I kind of expected it to make a bit of a splash in the small community of people using deep learning at the time, in March 2015 (a few thousands of people), but back then no one really expected deep learning would gather such interest over the following few years.”

3.2 Why Keras became popular?

In [3]:
```
# Training Data
train_X = numpy.asarray([3.3,4.4,5.5,6.71,6.93,4.168,9.779,6.182,7.59,2.167,
                         7.042,10.791,5.313,7.997,5.654,9.27,3.1])
train_Y = numpy.asarray([1.7,2.76,2.09,3.19,1.694,1.573,3.366,2.596,2.53,1.221,
                         2.827,3.465,1.65,2.904,2.42,2.94,1.3])
n_samples = train_X.shape[0]
```

In [4]:
```
# tf Graph Input
X = tf.placeholder("float")
Y = tf.placeholder("float")

# Set model weights
W = tf.Variable(rng.randn(), name="weight")
b = tf.Variable(rng.randn(), name="bias")
```

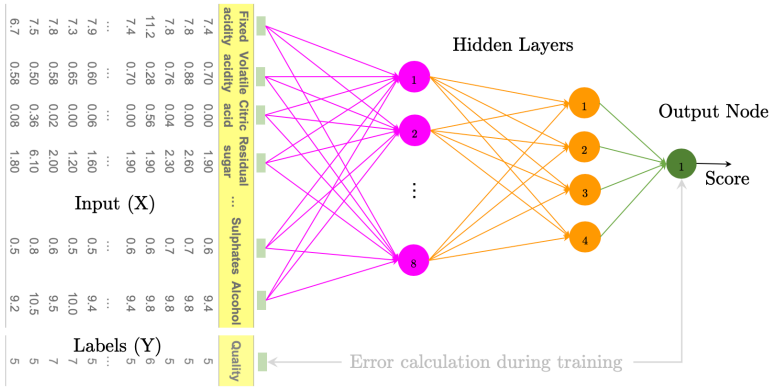
In [5]:
```
# Construct a linear model
pred = tf.add(tf.multiply(X, W), b)
```

In [6]:
```
# Mean squared error
cost = tf.reduce_sum(tf.pow(pred-Y, 2))/(2*n_samples)
# Gradient descent
optimizer = tf.train.GradientDescentOptimizer(learning_rate).minimize(cost)
```

In [8]:
```
# Initialize the variables (i.e. assign their default value)
init = tf.global_variables_initializer()
```

In [9]:
```
# Start training
with tf.Session() as sess:
    sess.run(init)

    # Fit all training data
    for epoch in range(training_epochs):
        for (x, y) in zip(train_X, train_Y):
            sess.run(optimizer, feed_dict={X: x, Y: y})

        #Display logs per epoch step
        if (epoch+1) % display_step == 0:
            c = sess.run(cost, feed_dict={X: train_X, Y:train_Y})
            print "Epoch:", '%04d' % (epoch+1), "cost=", "{:.9f}".format(c), \
                "W=", sess.run(W), "b=", sess.run(b)

    print "Optimization Finished!"
    training_cost = sess.run(cost, feed_dict={X: train_X, Y: train_Y})
    print "Training cost=", training_cost, "W=", sess.run(W), "b=", sess.run(b), '\n'
```

Native Tensorflow code

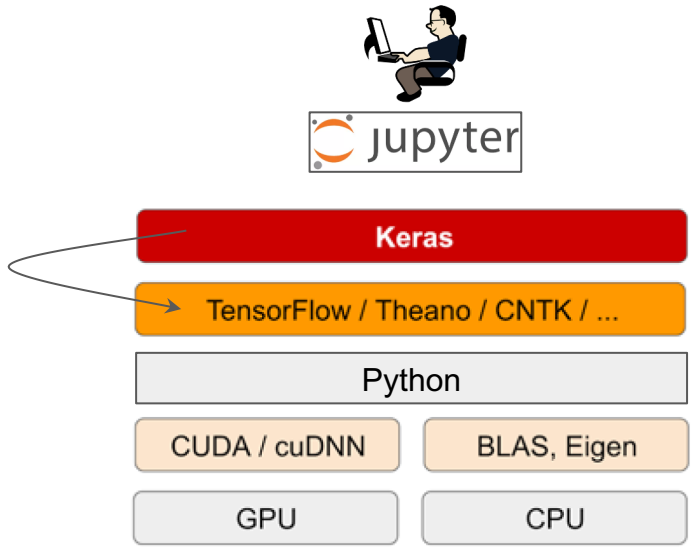


3.2 Features of Keras

- It allows the same code to run seamlessly on CPU or GPU
- Easy to quickly prototype deep-learning models
- It has built-in support for:
  - convolutional networks (for computer vision)
  - recurrent networks (for sequence processing)
  - and any combination of both
- It supports arbitrary network architectures
  - multi-input or multi-output models
  - layer sharing
  - model sharing, etc.
- Keras is appropriate for building essentially almost any deep-learning model
- It also supports multi-GPU training
- Keras can be freely used in commercial projects (permissive MIT license)

3.2.1 Keras, TensorFlow, & Jupyter Notebooks

- Keras doesn't handle low-level operations - tensor manipulation & differentiation
- Theano is developed by the MILA lab at Université de Montréal
- TensorFlow is developed by Google
- CNTK is developed by Microsoft
- Jupyter is developed by Project Jupyter
  - a nonprofit organization



Cuda

- CUDA is a parallel computing platform and API model created by NVIDIA
  - C/C++ programmers use 'CUDA C/C++'
  - Code is compiled with **nvcc**, Nvidia's C/C++ compiler
- It allows software developers and software engineers to use a CUDA-enabled graphics processing unit (GPU) for general purpose processing

```
void add( int *a, int *b, int *c ) {
    for (i=0; i < N; i++) {
        c[i] = a[i] + b[i];
    }
}
```

```
void add( int *a, int *b, int *c )
{
    int tid = 0;
    while (tid < N) {
        c[tid] = a[tid] + b[tid];
        tid += 2;
    }
}
```

- The cuBLAS library is an implementation of BLAS (Basic Linear Algebra Subprograms)
- It sits on top of the NVIDIA CUDA runtime
  - allows the user to access the computational resources of NVIDIA GPU

Cuda

```
void add( int *a, int *b, int *c ) {
    for (i=0; i < N; i++) {
        c[i] = a[i] + b[i];
    }
}
```

CPU CORE 1

```
void add( int *a, int *b, int *c )
{
    int tid = 0;
    while (tid < N) {
        c[tid] = a[tid] + b[tid];
        tid += 2;
    }
}
```

CPU CORE 2

```
void add( int *a, int *b, int *c )
{
    int tid = 1;
    while (tid < N) {
        c[tid] = a[tid] + b[tid];
        tid += 2;
    }
}
```

Keras is now an API in TensorFlow 2.0

Tensorflow 2.0 examples!

```
1 import tensorflow.keras
2 tensorflow.keras.__version__
```

```
'2.2.4-tf'
```

```
1 from tensorflow.keras.datasets import boston_housing
2 (train_data, train_targets), (test_data, test_targets) =  boston_housing.load_data()
```

Find the difference!

Examples in the book!

```
1 import keras
2 keras.__version__
```

```
'2.2.5'
```

```
1 from keras.datasets import boston_housing
2 (train_data, train_targets), (test_data, test_targets) =  boston_housing.load_data()
```

Colab currently offers both!

```
1 ! pip uninstall keras
```

```
1 import keras
2 keras.__version__
```

```
---------------------------
ModuleNotFoundError
```

3.3.2 Getting Keras Running

- Options:
  - Google Colab
  - Microsoft Azure
  - Install everything locally
    - For frequent/heavy usage and large datasets

- Local Installation
  - Ubuntu version for your hardware (problems with windows
  - CUDA driver version, CUDA version & CuDNN version
  - Python version & libraries' versions
  - Anaconda & Jupyter Notebook
  - Set up Tensorboard correctly

- Setting it up in servers can be daunting!!

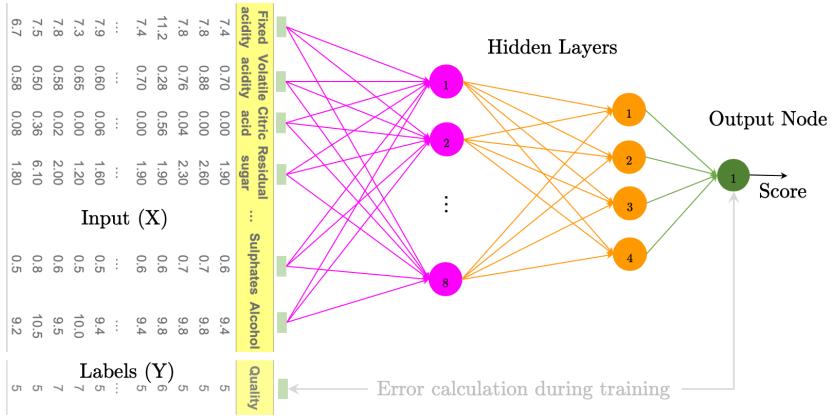
Will you ever need to write Tensorflow code?

- Probably not!
  - Unless you are working on something different from a typical machine learning problem
- Usually knowing how Tensorflow works is sufficient
  - But you may need Python & Numpy skills, and knowledge of Keras features
- I had to code my own Tensorflow function when I needed a specific huber loss

```
def huber_loss(y_true, y_pred, clip_delta = 15.0):
    error = y_true - y_pred
    cond  = tf.keras.backend.abs(error) < clip_delta
    squared_loss = 0.5 * tf.keras.backend.square(error)
    linear_loss  = clip_delta * (tf.keras.backend.abs(error) - 0.5 * clip_delta)
    return tf.where(cond, squared_loss, linear_loss)
```

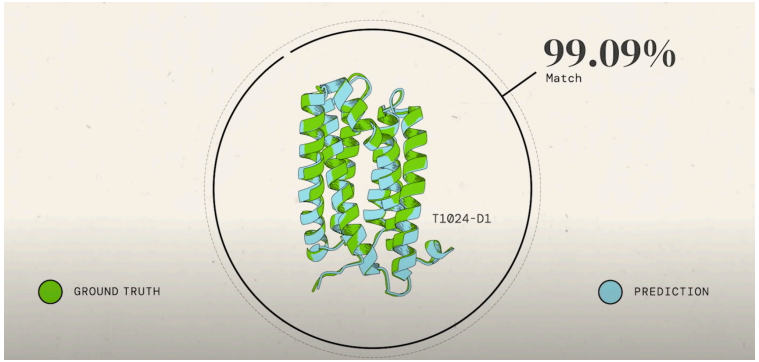
Sequential vs Functional API in Keras

```
from tensorflow.keras import Sequential, Model
from tensorflow.keras.layers import Dense, Input

seq_model = Sequential()
seq_model.add(Dense(8, input_shape = (8, ), activation = 'relu'))
seq_model.add(Dense(4, activation = 'relu'))
seq_model.add(Dense(1, activation = 'sigmoid'))
seq_model.summary()

layer1 = Input(shape = (8, ))
layer2 = Dense(8, activation='relu')(layer1)
layer3 = Dense(4, activation='relu')(layer2)
output = Dense(1, activation='sigmoid')(layer3)
func_model = Model(inputs = layer1, outputs = output)
func_model.summary()
```



Diversity of thought is holding back AI research

“There are some category of problems that require industry-scale training resources, for sure. But there is no shortage of problems where a single GPU is all you need to make significant progress. The main thing that’s holding back AI research right now is not a lack of hardware, it’s a lack of diversity of thought. If you have limited resources, don’t spend your time worrying about GPUs, rather, spend it worrying whether you’re working on the right problem and asking the right questions.” - François Chollet



AlphaFold: The making of a scientific breakthrough

https://youtu.be/gg7WjuFs8F4