

Neural Networks - I



TensorFlow can be tough to learn :(

Keras



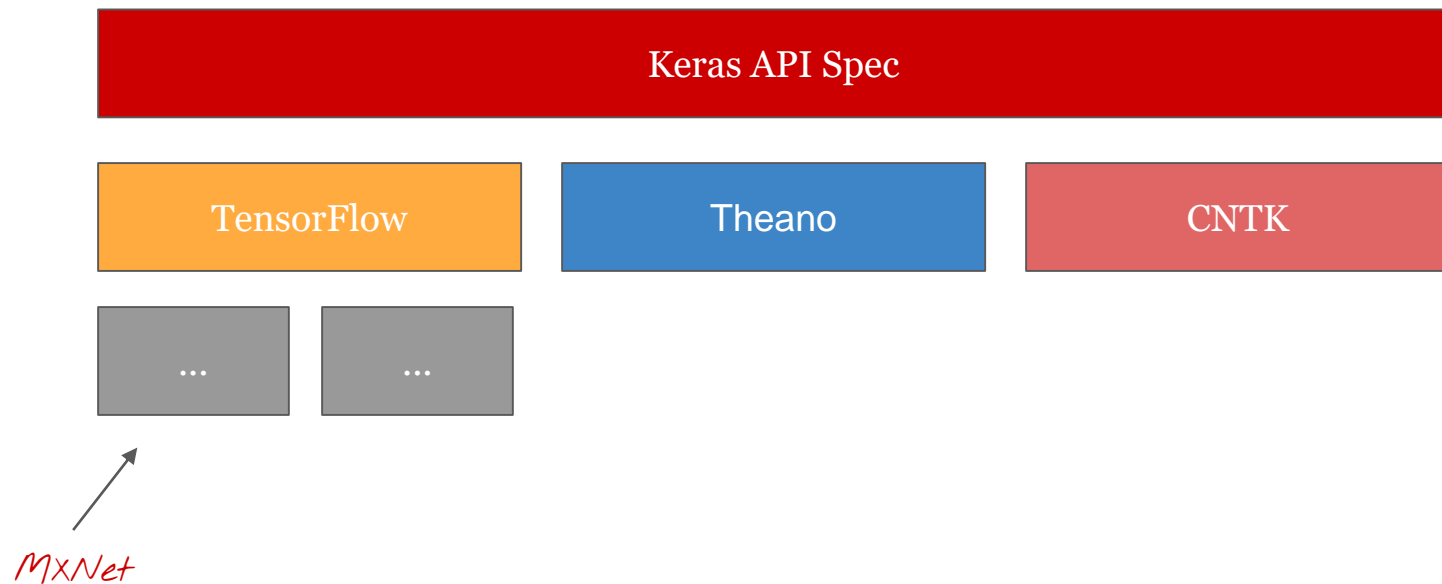
Simple



Minimal Code



Powerful





How do we simplify TensorFlow?

Keras is now part of TensorFlow codebase

How can Keras help

Using Keras as part of TensorFlow

Building model in TensorFlow Keras


```
#Declare Weights and Bias
```

```
w = tf.zeros(shape=(13,1))
```

```
b = tf.zeros(shape=(1))
```

```
# y = wx + b
```

```
def prediction(x, w, b):
```

```
    xw_matmul = tf.matmul(x, w)
```

```
    y = tf.add(xw_matmul, b)
```

```
return y
```

```
model = tf.keras.Sequential()
```

```
model.add(tf.keras.layers.Dense(1, input_shape=(13,)))
```



```
#Define Loss
```

```
def loss(y_actual, y_predicted):
```

```
    diff = y_actual - y_predicted
```

```
    sqr = tf.square(diff)
```

```
    avg = tf.reduce_mean(sqr)
```

```
    return avg
```

```
#Define Gradient Descent function
```

```
def train(x, y_actual, w, b, learning_rate=0.01):
```

```
    #Record math ops on 'tape' to calculate loss  
    with tf.GradientTape() as t:
```

```
        t.watch([w,b])
```

```
        current_prediction = prediction(x, w, b)
```

```
        current_loss = loss(y_actual, current_prediction)
```

```
    #Calculate Gradients for Loss w.r.t Weights and Bias  
    dw, db = t.gradient(current_loss,[w, b])
```

```
    #Update Weights and Bias
```

```
    w = w - learning_rate*dw
```

```
    b = b - learning_rate*db
```

```
    return w, b
```

```
model.compile(optimizer='sgd', loss='mse')
```



Training the Model



```
#Training the model for 100 iterations

for i in range(100):

    w, b = train(train_x, train_y, w, b)
    print('Current Loss on iteration', i, loss(train_y, prediction(train_x, w,
b)).numpy())
```



```
model.fit(train_x, train_y, epochs=100)
```

Where is my model saved?

Saving Model

```
model.save(<file_name>)
```

Install h5py using pip

Exercise

Build Boston Housing Predictor in TensorFlow Keras