Neural Network Framework

y = wx+b

CPU

Deep Learning Frameworks

- Scale ML code
- Compute Gradients!
- Standardize ML applications for sharing
- Interface with GPUs for parallel processing

Confusion Matrix

	Yes	No
Yes	TP	FN
No	FP	TN

2 x 2 matrix

TensorFlow, Keras, and PyTorch

TensorFlow is an end-to-end open source **platform** for machine learning. It has a comprehensive, flexible ecosystem to build and deploy ML powered applications.

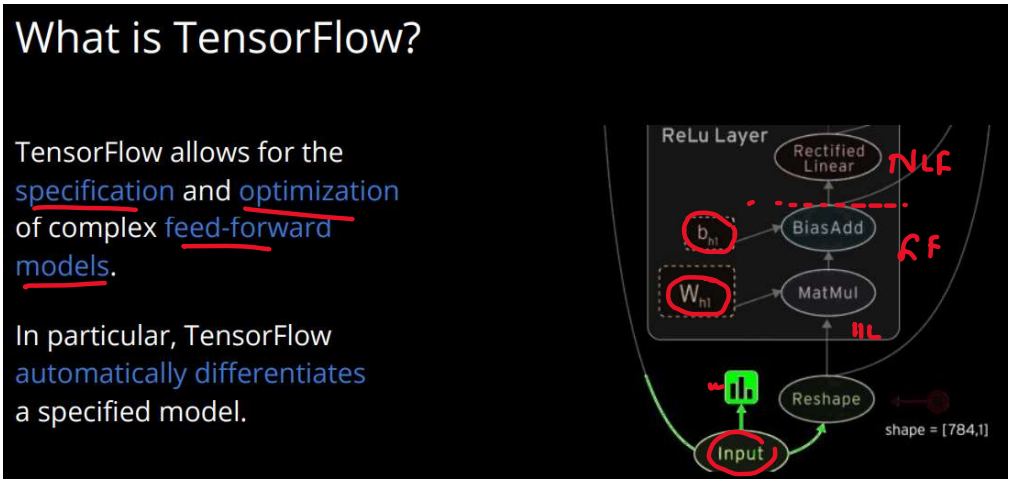
Keras is a high-level neural networks **API**, written in Python and capable of running on top of *TensorFlow*, *CNTK*, or *Theano*. It was developed with a focus on enabling fast experimentation.

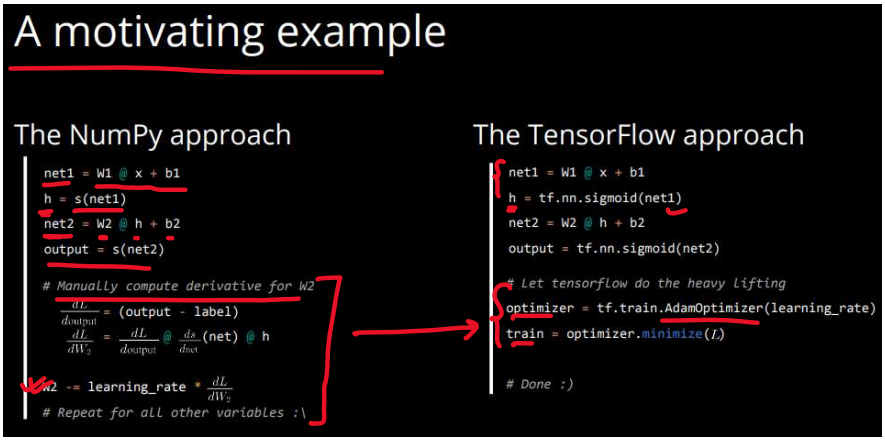
PyTorch is an open source machine learning **framework** that accelerates the path from research prototyping to production deployment.

A Brief History of TensorFlow

TensorFlow is an end-to-end FOSS (free and open source software) library for dataflow, differentiable programming. TensorFlow is one of the most popular program frameworks for building machine learning applications.

- Google Brain built **DistBelief** in 2011 for internal usage.
- TensorFlow 1.0.0 was released on Feb 11, 2017
- TensorFlow 2.0 was released in Jan 2018.





What is a Tensor in TensorFlow?

Rank – Unit of dimensionality described within the tensor

Shape – Number of rows and columns define together

- **TensorFlow** uses a tensor data structure to represent all data. A TensorFlow tensor has an n-dimensional array or list. A tensor has a static type, a rank, and a shape.

Name	Rank	Tensor
Scalar	0	[5]
Vector	1	[1 2 3]
Matrix	2	[[1 2 3 4], [5 6 7 8]]
Tensor	3	...

→ Row vector, Column Vector

Type – Data type assigned to Tensor's elements.

TensorFlow Data Types

Basic TensorFlow data types include:

- **int[8|16|32|64], float[16|32|64], double**
- **bool**
- **string**

with **tf.cast()**, the data types of variables could be converted.

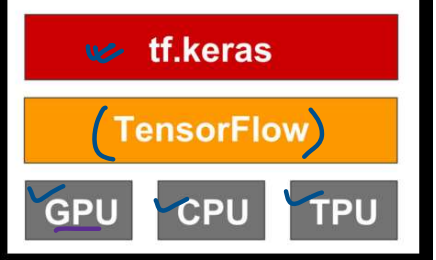
TensorFlow Variables

TensorFlow variables can represent shared, persistent state manipulated by your program. **Weights** and **biases** are usually stored in variables.

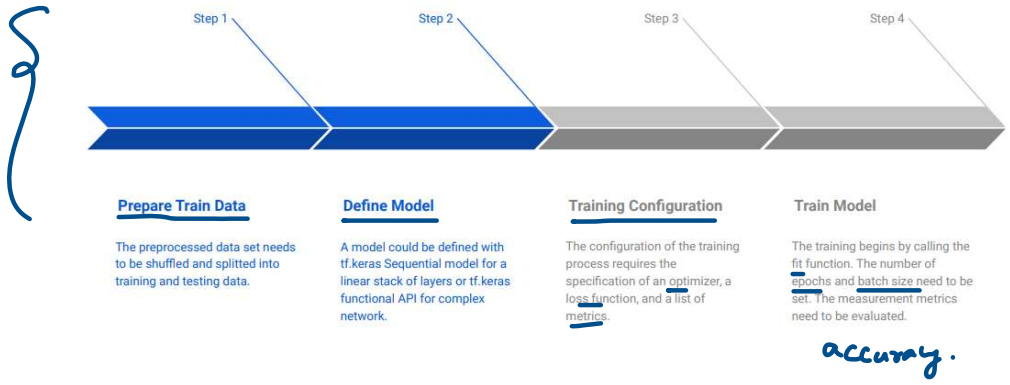
```
import tensorflow as tf

W = tf.Variable(tf.random.normal([2,2], stddev=0.1),
name = "W")
b = tf.Variable(tf.zeros(shape=(2)), name="b")
```

Keras is the official high-level API of TensorFlow

- tensorflow.keras (tf.keras) module
- Part of core TensorFlow since v1.4
- Full Keras API
- Better optimized for TF
- Better integration with TF-specific features



Machine Learning Workflow with tf.keras



What's special about Keras?

- A focus on user experience.
- Large adoption in the industry and research community.
- Multi-backend, multi-platform.
- Easy productization of models.

633 contributors

Google Microsoft NVIDIA aws

250,000
Keras developers

> 2x
Year-on-year growth

Industry Use

NETFLIX UBER Google

instacart HUAWEI NVIDIA

Square Expedia Zocdoc yelp

etc...

The Sequential API

```
import keras
from keras import layers

model = keras.Sequential()
model.add(layers.Dense(20, activation='relu', input_shape=(10,)))
model.add(layers.Dense(20, activation='relu'))
model.add(layers.Dense(10, activation='softmax'))

model.fit(x, y, epochs=10, batch_size=32)
```

The functional API

```
import keras
from keras import layers

inputs = keras.Input(shape=(10,))
x = layers.Dense(20, activation='relu')(x)
x = layers.Dense(20, activation='relu')(x)
outputs = layers.Dense(10, activation='softmax')(x)

model = keras.Model(inputs, outputs)
model.fit(x, y, epochs=10, batch_size=32)
```

Model Subclassing

```
import keras
from keras import layers

class MyModel(keras.Model):

    def __init__(self):
        super(MyModel, self).__init__()
        self.dense1 = layers.Dense(20, activation='relu')
        self.dense2 = layers.Dense(20, activation='relu')
        self.dense3 = layers.Dense(10, activation='softmax')

    def call(self, inputs):
        x = self.dense1(x)
        x = self.dense2(x)
        return self.dense3(x)

model = MyModel()
model.fit(x, y, epochs=10, batch_size=32)
```

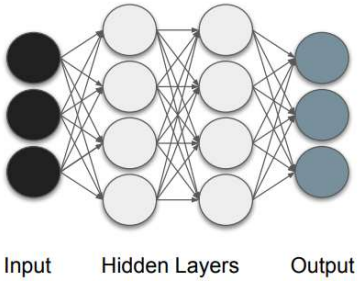
tf.keras Built-in Datasets

- tf.keras provides many popular reference datasets that could be used for demonstrating and testing deep neural network models. To name a few,
  - Boston Housing (regression)
  - CIFAR100 (classification of 100 image labels)
  - MNIST (classification of 10 digits)
  - Fashion-MNIST (classification of 10 fashion categories)
  - Reuters News (multiclass text classification)

Create a tf.keras Model

- Layers are the fundamental building blocks of **tf.keras** models.
- The **Sequential** model is a linear stack of layers.
- A **Sequential** model can be created with a list of layer instances to the constructor or added with the **.add()** method.
- The input shape/dimension of the first layer need to be set.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense,
Activation

model = Sequential([
    Dense(64, activation='relu', input_dim=20),
    Dense(10, activation='softmax')
])
```



Compile a tf.keras Model

The **compile** method of a Keras model configures the learning process before the model is trained. The following 3 arguments need to be set (the optimizer and loss function are required).

- An optimizer: **Adam**, **AdaGrad**, **SGD**, **RMSprop**, etc.
- A loss function: **mean_squared_error**, **mean_absolute_error**, **mean_squared_logarithmic_error**, **categorical_crossentropy**, **kullback_leibler_divergence**, etc.
- A list of measurement metrics: **accuracy**, **binary_accuracy**, **categorical_accuracy**, etc.