Charles B.

What is TensorFlow first ?

TensorFlow

- Fast, flexible and scalable Google open source library for **Machine Learning** and **Deep Neural Networks** research

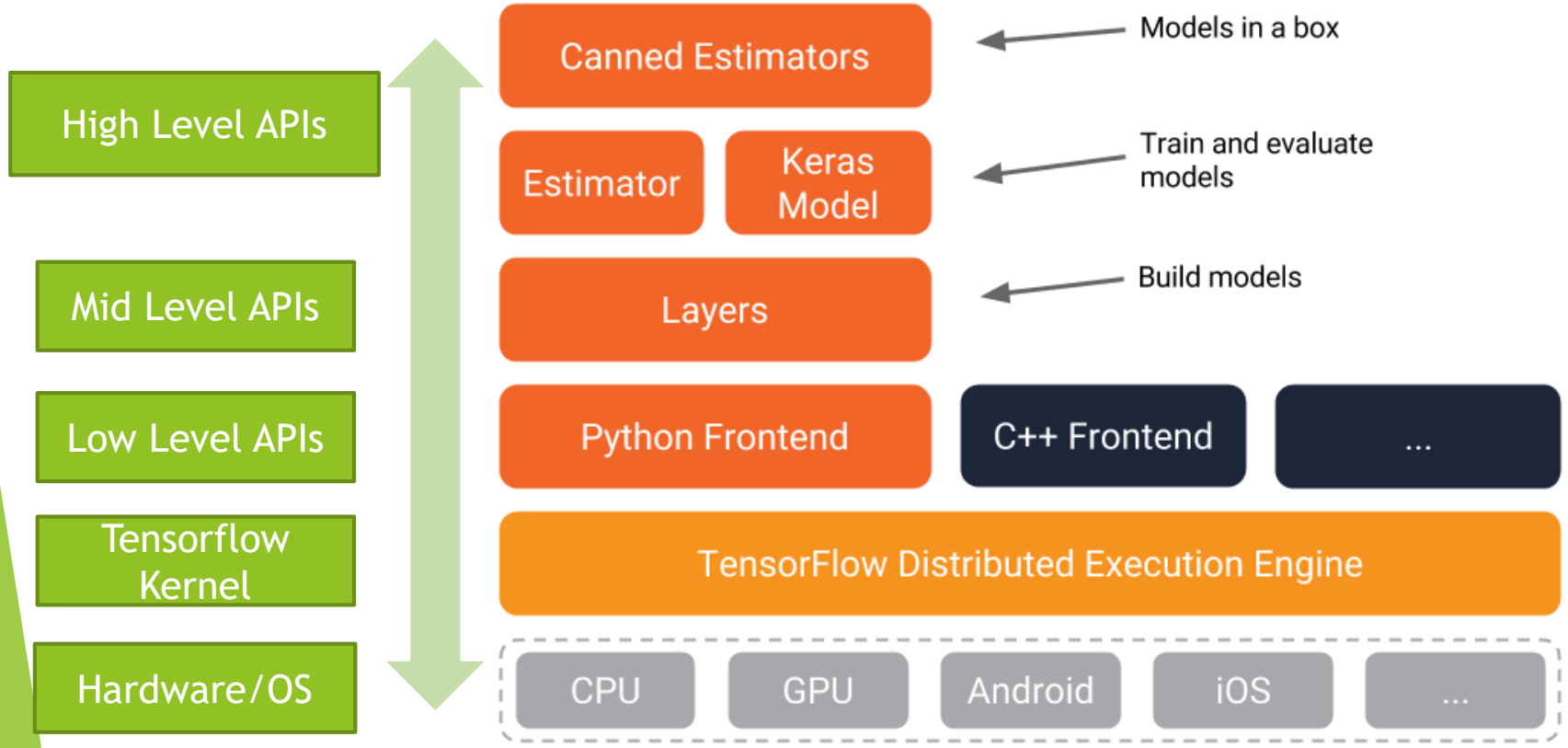
- Performs numerical computations in the form of a **Dataflow graph.** Tensorflow separates definition of computations from their execution.
  - The graph here is a data structure that describes the computation you want to perform

- **Cross-Platform** : it runs on nearly everything: **GPUs and CPUs**—including mobile and embedded platforms

Here is a demo if you are intereted in building a DNN fast through an application sandbox : http://playground.tensorflow.org

TensorFlow Hub

Tensorflow overview

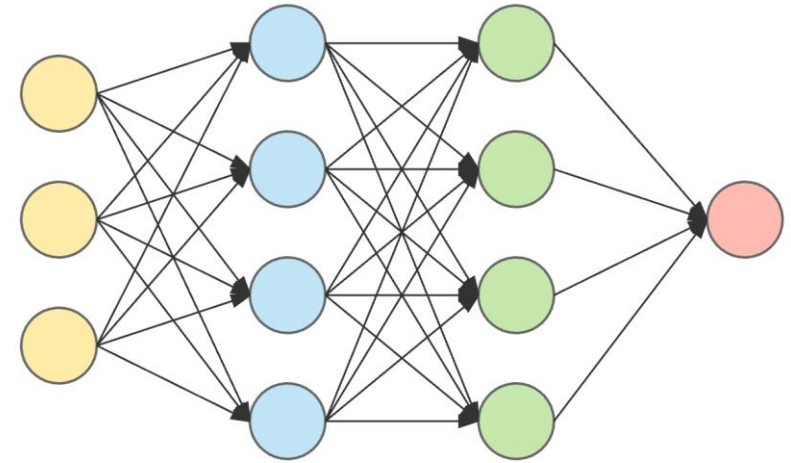


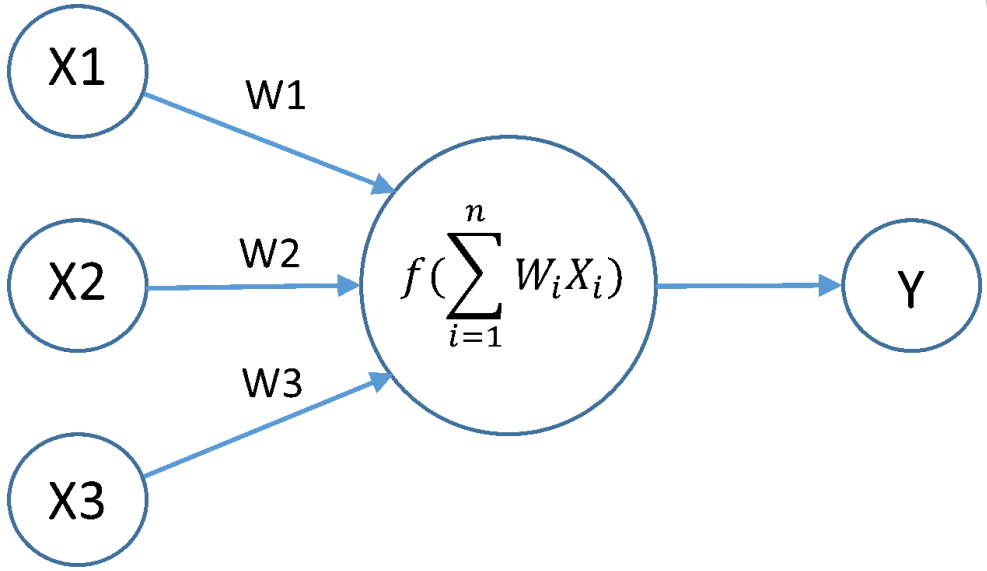
An **estimator** is a high level API Tensorflow module which encapsulate :

- Training
- Evaluation
- Prediction
- Export

TensorFlow Hub

Basic Neural Network Architecture





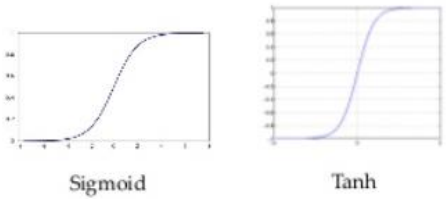
f

Non-Linear Activation Function

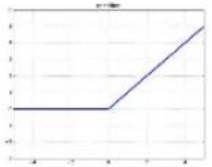
- Sigmoid: $S(t) = \frac{1}{1+e^{-t}}$.
- Tanh: $\tanh x = \frac{\sinh x}{\cosh x} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$
- Rectified Linear Unit (ReLU):

$$f(x) = \max(0, x)$$

Most popular activation function for DNN as of 2015, avoids saturation issues, makes learning faster

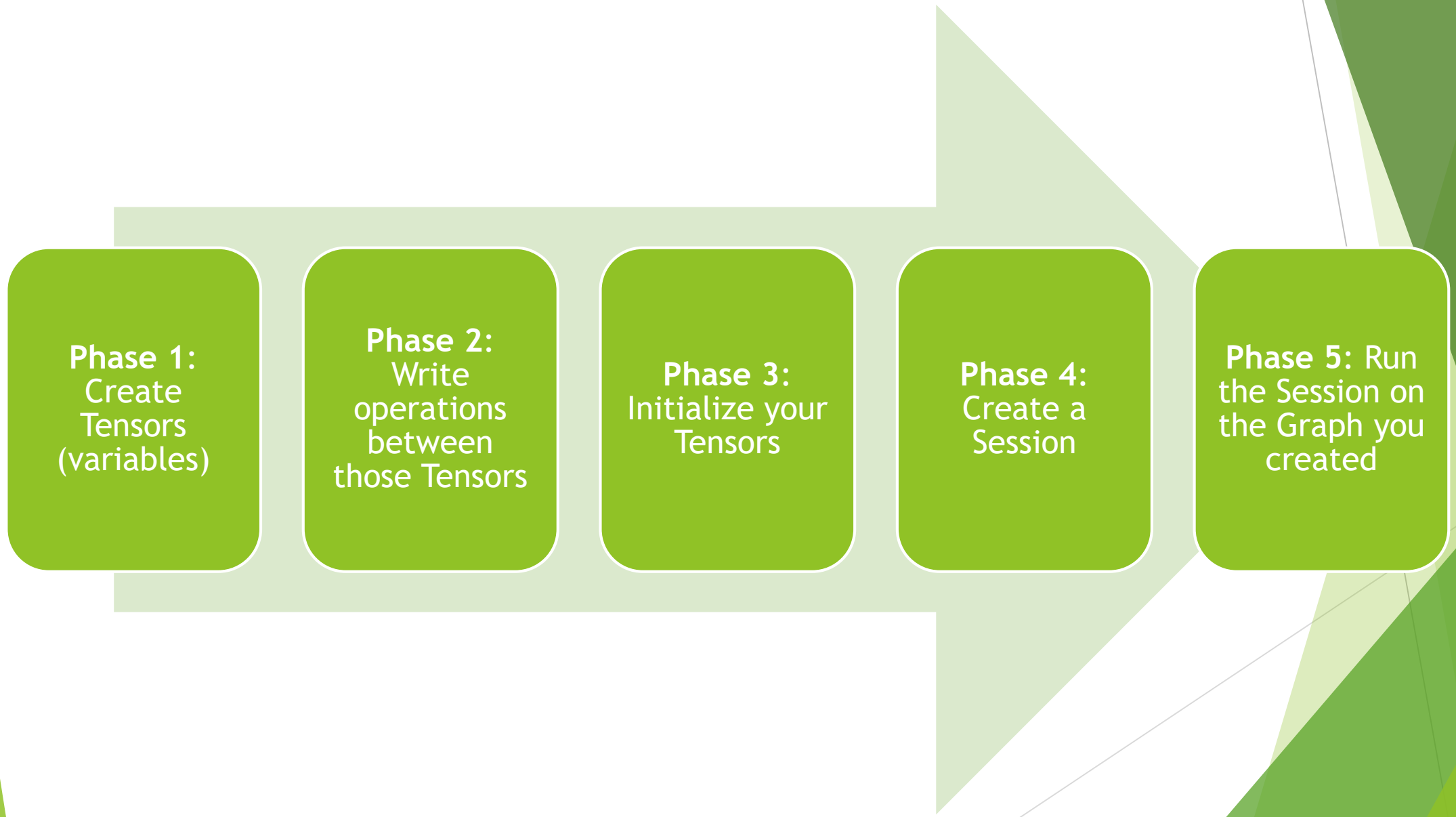

Sigmoid

Tanh


ReLU

TensorFlow Hub

Data flow



TensorFlow Hub

Tensorflow Tensors

- Computations in Tensorflow are made inside a graph with **Tensors,** the main data structure.
- A **Tensor** is a generalization of vectors and matrices to potentially higher dimensions. Internally, TensorFlow represents tensors as **n-dimensional arrays** of base datatypes
- A Tensor has a **data type** (float32, int32 or string) and a **shape** (dimension).

Rank	Math entity
0	Scalar (magnitude only)
1	Vector (magnitude and direction)
2	Matrix (table of numbers)
3	3-Tensor (cube of numbers)
n	n-Tensor (you get the idea)

```
import tensorflow as tf
```

```
floating = tf.Variable(3.14159265359, tf.float64)
```

```
first_primes = tf.Variable([2, 3, 5, 7, 11], tf.int32)
```

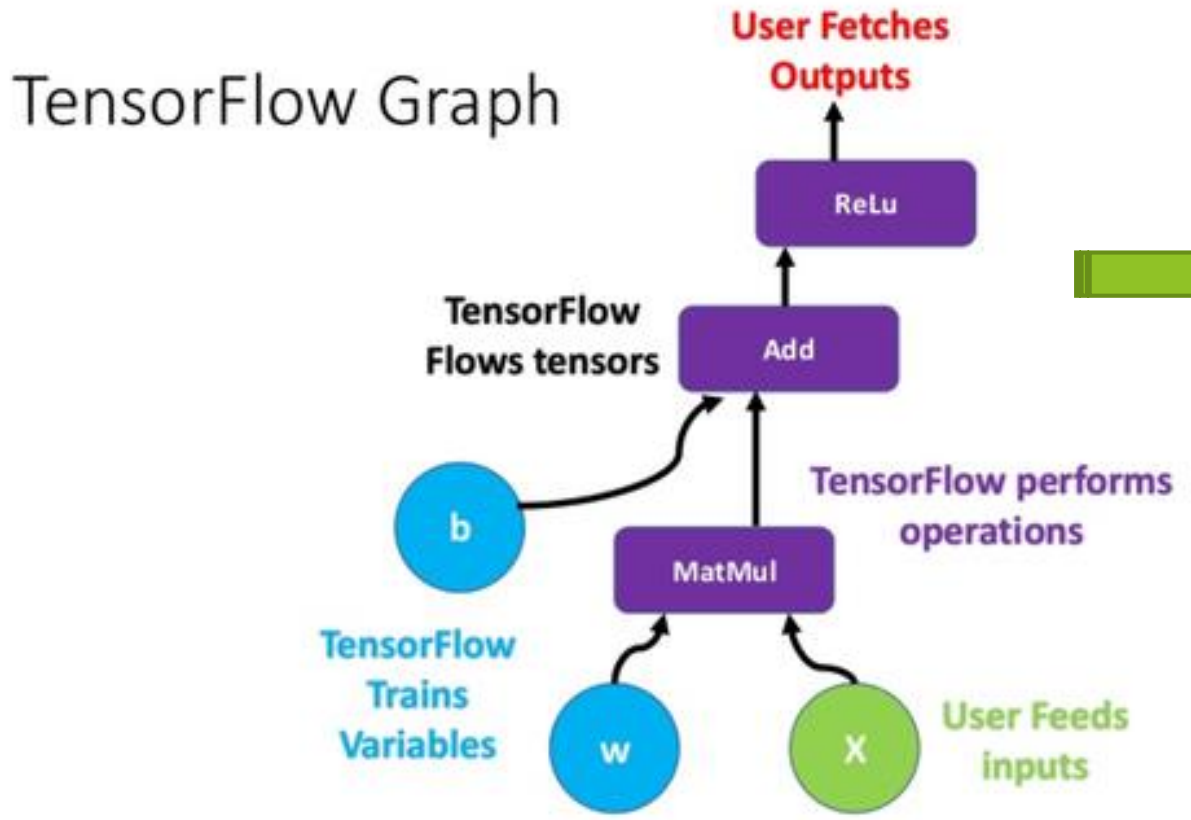
```
squarish_squares = tf.Variable([ [4, 9], [16, 25] ], tf.int32)
```

```
my_image = tf.zeros([10, 299, 299, 3])  # batch x height x width x color
```

TensorFlow Hub

Tensorflow Graph

A **graph** is a series of TensorFlow operations arranged into a graph of nodes. In another words, it means a graph is just an arrangement of nodes that represent the operations in your model.




```
Z1 = tf.add(tf.matmul(W1,X),b1)
A1 = tf.nn.relu(Z1)
```

TensorFlow Hub

Running a Session in the current Graph

```
init = tf.global_variables_initializer()

with tf.Session() as session:
    session.run(init)
    print(session.run(A1))
```

Initializes the variables (when run later in the session)

Session() object takes the current graph by default, tf.get_default_graph. But we could have precised another one with *graph* argument

Working with several graphs

```
g1 = tf.Graph()
g2 = tf.Graph()
with g1.as_default():
    #...add nodes to graph g1
with g2.as_default():
    #...add nodes to graph g2

sess1 = tf.Session(graph = g1)
sess2 = tf.Session(graph = g2)
```

TensorFlow Hub

What is TensorFlow-Hub ?

- TF-Hub is a library for the **publication**, discovery and **consumption** of state-of-art Machine and Deep Learning Models.
- It is similar to some Deep Learning frameworks such as Caffe, Keras, etc. but is actually easier for everyone to **publish** and **host models**.
- It is part of the Tensorflow framework

TensorFlow Hub

Why TensorFlow-Hub ?

We need a **library** for reusable machine learning modules.

A **module** is a self-contained piece of a TensorFlow graph, along with its weights and assets, that can be reused across different tasks in a process known as **Transfer Learning**.

Weights are the parameters of the pretrained model that we want to use for our own model. They can be either retrained or just fixed.

It is a very useful technique for people who doesn't have time or skills to **collect a huge amount of data** or doesn't have enough **hardware power** to train their model.

Sharing a **pretrained model** makes it possible for a developer to customize it for their domain.

TensorFlow Hub

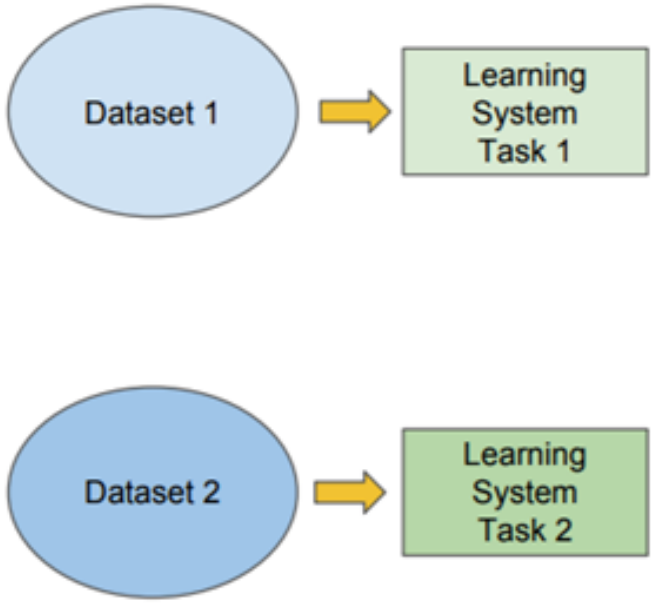
« Transfer Learning will be the next driver of ML success ! »

Andrew Ng

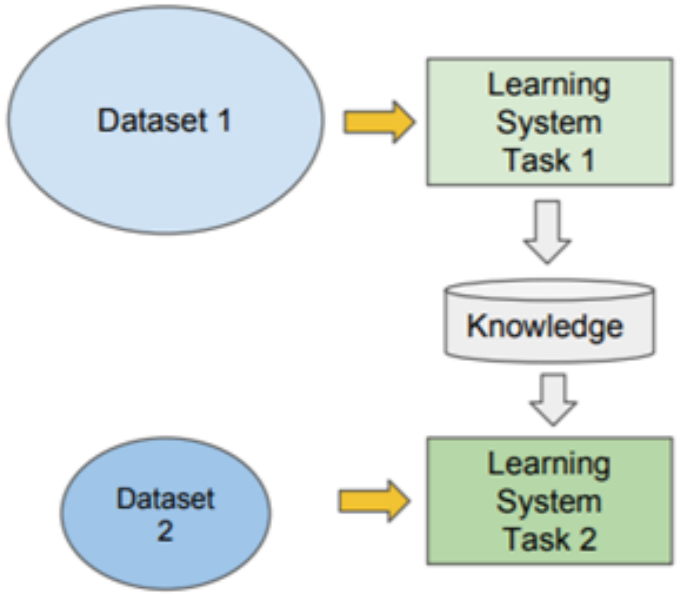
TensorFlow Hub

Traditional ML vs Transfer Learning

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks



- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data



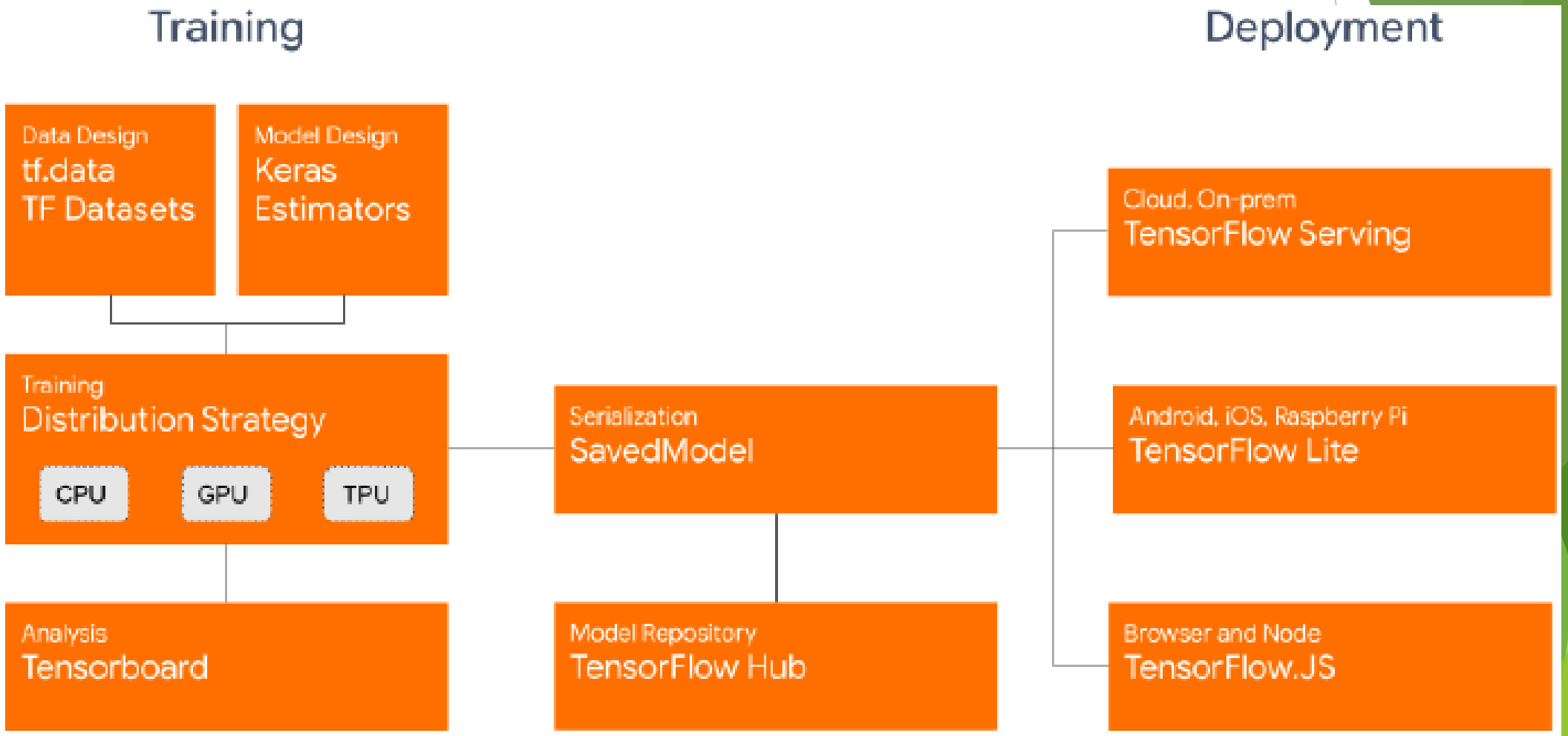
TensorFlow Hub

Transfer Learning in a nutshell

- Train a model with a smaller dataset
- Improve generalization
- Speed up model training

TensorFlow Hub

TF pipeline



TensorFlow Hub

Tensorflow-hub Modules

Image

Here you'll find models that enable:

- Image augmentation
- Image classification
- Feature Vector Extraction

And more.

Text

Here you'll find models that enable:

- Text embedding

Video

Here you'll find models that enable:

- Video Classification

TensorFlow Hub

Quick demonstration

Use of a pre-trained word embedding on random sentences.

Word Embedding is a technique which transforms a word into a vector of real numbers, making possible to cluster vectors which are close to each other (words having a similar meaning/sentiment). It is also helpful to reduce the dimensions of words representation, making training easier through a Machine/Deep Learning model.

```
!pip install "tensorflow_hub>=0.5.0"
!pip install "tensorflow>=2.0.0"

import tensorflow as tf
import tensorflow_hub as hub

module_url = "https://tfhub.dev/google/tf2-preview/nnlm-en-dim128/1"
embed = hub.KerasLayer(module_url)
embeddings = embed(["A long sentence.", "single-word",
                    "http://example.com"])
print(embeddings.shape)  #(3,128)
```

TensorFlow Hub

Let’s inspect this module in details

← tf2-preview/nnlm-en-dim128

Problem domain

Embedding

Publisher

Google

Architecture

Nnlm

Data set

News

TF2 SavedModel

This is a SavedModel in TensorFlow 2 format. Using it requires TensorFlow 2 (or 1.15) and TensorFlow Hub 0.5.0 or newer.

Overview

This module is in the **SavedModel 2.0** format and was created to help preview TF2.0 functionalities. It is based on https://tfhub.dev/google/nnlm-en-dim128/1.

Text embedding based on feed-forward Neural-Net Language Models[1] with pre-built OOV. Maps from text to 128-dimensional embedding vectors.

Details

Based on NNLM with three hidden layers.

Input

The module takes **a batch of sentences in a 1-D tensor of strings** as input.

Preprocessing

The module preprocesses its input by **splitting on spaces**.

Out of vocabulary tokens

Small fraction of the least frequent tokens and embeddings (~2.5%) are **replaced by hash buckets**. Each hash bucket is initialized using the remaining embedding vectors that hash to the same bucket.

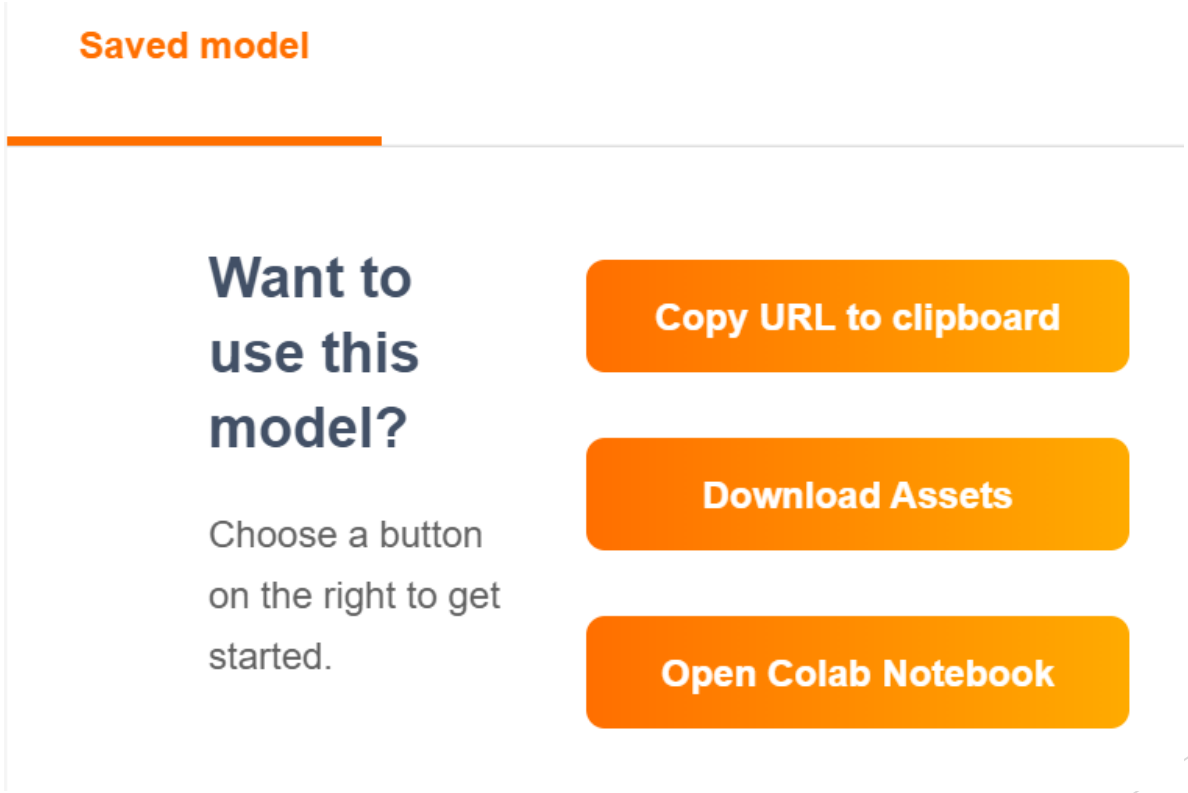
Sentence embeddings

Word embeddings are combined into sentence embedding using the `sqrtn` combiner (see tf.nn.embedding_lookup_sparse).

TensorFlow Hub

How to use modules in Tensorflow-Hub ?

- Go to Tensorflow-Hub website (https://tfhub.dev) and explore all available modules according to your needs.
- When you have chosen a module, use the corresponding url in your code or directly use the assets (model's weights). Look at the documentation and also examples to understand how to use it.



TensorFlow Hub

Example: Build a sentiment classifier

- We will use a **TF-Hub text embedding module** to train a simple **sentiment classifier** with a good accuracy. We will then analyze the predictions to make sure our model is reasonable.

- We will try to solve the *Large Movie Review Dataset v1.0* task from Mass et al. The dataset consists of IMDB (Internet Movie Database) movie reviews labeled by positivity from 1 to 10. The task is to label the reviews as **negative** or **positive**.

TensorFlow Hub

Data preprocessing

```
import tensorflow as tf
import tensorflow_hub as hub
import matplotlib.pyplot as plt
import numpy as np
import os
import pandas as pd
import re
import seaborn as sns
```

```
train_df, test_df = download_and_load_datasets()
train_df.head()
```

	sentence	sentiment	polarity
0	After what was considered to be the official D...	10	1
1	This movie was like "The Disney Channel after ...	1	0
2	Although time has revealed how some of the eff...	8	1
3	A lot has been said about Shinjuku Triad Socie...	10	1
4	In September 2003 36-year-old Jonny Kennedy di...	8	1

```
# Load all files from a directory in a DataFrame.
def load_directory_data(directory):
  data = {}
  data["sentence"] = []
  data["sentiment"] = []
  for file_path in os.listdir(directory):
    with tf.gfile.GFile(os.path.join(directory, file_path), "r") as f:
      data["sentence"].append(f.read())
      data["sentiment"].append(re.match("\d+_(\d+)\.txt", file_path).group(1))
  return pd.DataFrame.from_dict(data)

# Merge positive and negative examples, add a polarity column and shuffle.
def load_dataset(directory):
  pos_df = load_directory_data(os.path.join(directory, "pos"))
  neg_df = load_directory_data(os.path.join(directory, "neg"))
  pos_df["polarity"] = 1
  neg_df["polarity"] = 0
  return pd.concat([pos_df, neg_df]).sample(frac=1).reset_index(drop=True)

# Download and process the dataset files.
def download_and_load_datasets(force_download=False):
  dataset = tf.keras.utils.get_file(
      fname="aclImdb.tar.gz",
      origin="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
      extract=True)

  train_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                       "aclImdb", "train"))
  test_df = load_dataset(os.path.join(os.path.dirname(dataset),
                                      "aclImdb", "test"))

  return train_df, test_df
```

TensorFlow Hub

```
# Training input on the whole training set with no limit on training epochs.
train_input_fn = tf.estimator.inputs.pandas_input_fn(
    train_df, train_df["polarity"], num_epochs=None, shuffle=True)

# Prediction on the whole training set.
predict_train_input_fn = tf.estimator.inputs.pandas_input_fn(
    train_df, train_df["polarity"], shuffle=False)
# Prediction on the test set.
predict_test_input_fn = tf.estimator.inputs.pandas_input_fn(
    test_df, test_df["polarity"], shuffle=False)
```

Input functions

```
embedded_text_feature_column = hub.text_embedding_column(
    key="sentence",
    module_spec="https://tfhub.dev/google/nnlm-en-dim128/1")
```

Text embedding module

Build our model

```
estimator = tf.estimator.DNNClassifier(
    hidden_units=[500, 100],
    feature_columns=[embedded_text_feature_column],
    n_classes=2,
    optimizer=tf.train.AdagradOptimizer(learning_rate=0.003))
```

Estimator (first layer: 500 units, second layer: 100 units)

TensorFlow Hub

```
estimator.train(input_fn=train_input_fn, steps=1000)
```

Model's training

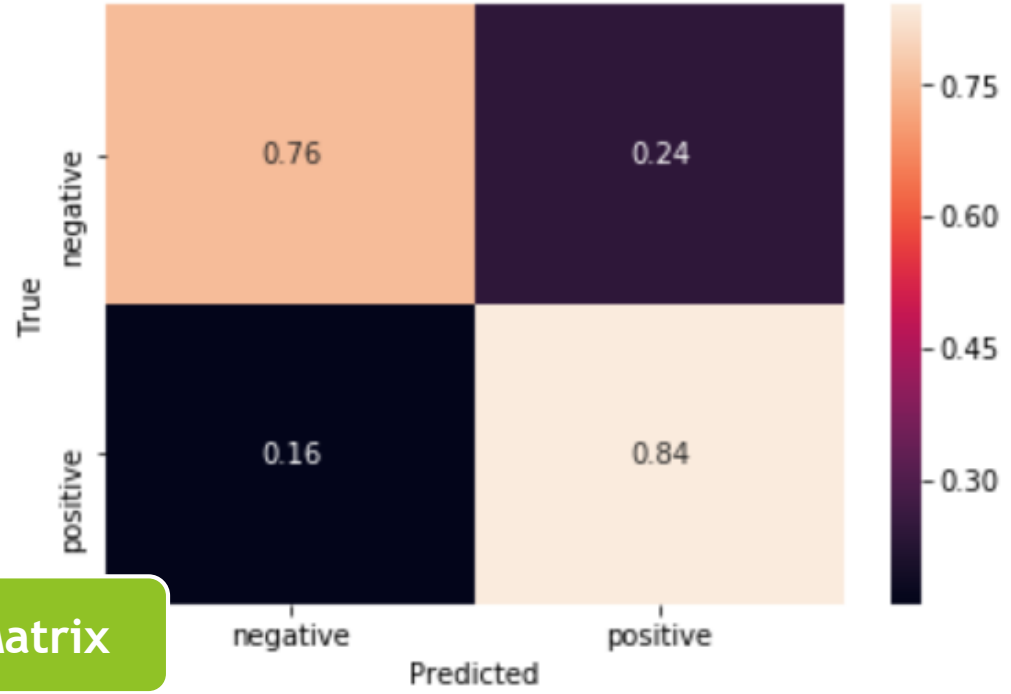
```
train_eval_result = estimator.evaluate(input_fn=predict_train_input_fn)
test_eval_result = estimator.evaluate(input_fn=predict_test_input_fn)

print("Training set accuracy: {accuracy}".format(**train_eval_result))
print("Test set accuracy: {accuracy}".format(**test_eval_result))
```

Predictions on train/test sets

```
Training set accuracy: 0.7990000247955322
Test set accuracy: 0.7924399971961975
```

Accuracies



Confusion Matrix

TensorFlow Hub

Additionnal Ressources

https://opensource.com/article/17/11/intro-tensorflow

https://medium.com/analytics-vidhya/reusing-a-pre-trained-deep-learning-model-on-a-new-task-transfer-learning-1c0a25a92dfb

https://medium.com/tensorflow/introducing-tensorflow-hub-a-library-for-reusable-machine-learning-modules-in-tensorflow-cdee41fa18f9

https://www.tensorflow.org/hub/tutorials/text_classification_with_tf_hub