Intro to Tensorflow

Siham Tabik

siham@ugr.es

Outline

1. Overview of the existing DL Software, structure and differences
2. Most popular DL software
3. Why Tensorflow?
4. How Tensorflow works?
5. Graph
6. Session
7. How to train a model?
8. How backpropagation is done in TF?
9. Build a CNN

Why study DL framework?

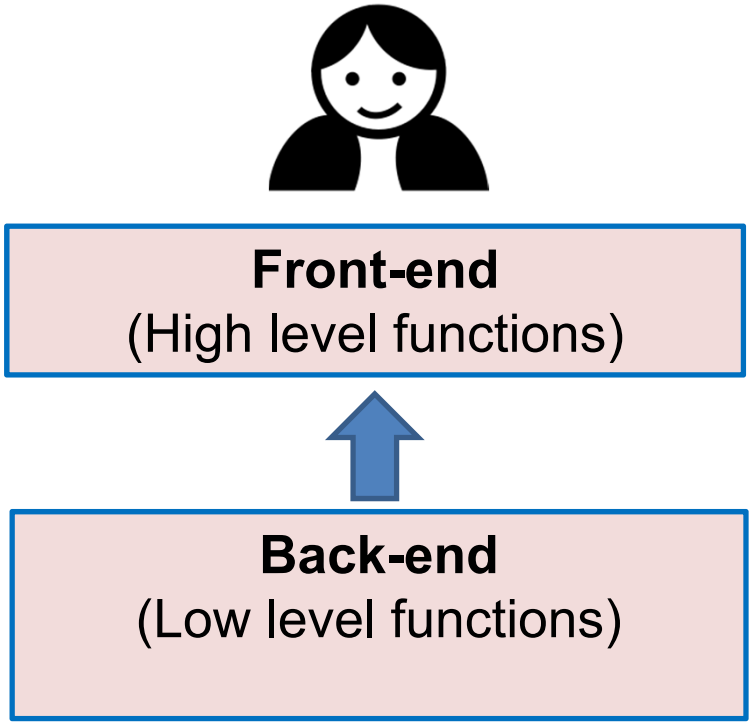
- Many achievements of DL and ML were developed thanks to DL software
- If you think about an application in your smartphone it is highly probable that it was developed thanks to these frameworks
- Allows researchers to run their models extremely quickly
- Make DL and ML much more accessible to practitioners than before
- DL frameworks helps scaling ML codes. Thanks to this Google and Facebook have scaled their users to billions
- They compute gradient automatically
- Building a new DL model is easy because you don’t have to start from scratch

How DL software is structured?

- Currently, there are more than 44 frameworks, libraries and programs

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software

- structured in Back-end (Interface) + Front-end



Which DL framework do I need?

Supported models: CNNs, Recurrent NN, FNN, …

Target architectures: multicore, single GPU, multi-GPU, distributed memory systems, cloud

Interface or API: python, matlab, c++, command line

Pre-trained models: e.g., VGG-16

Solvers: e.g., SGD, Adam, Adadelta,

Which one is the easiest?

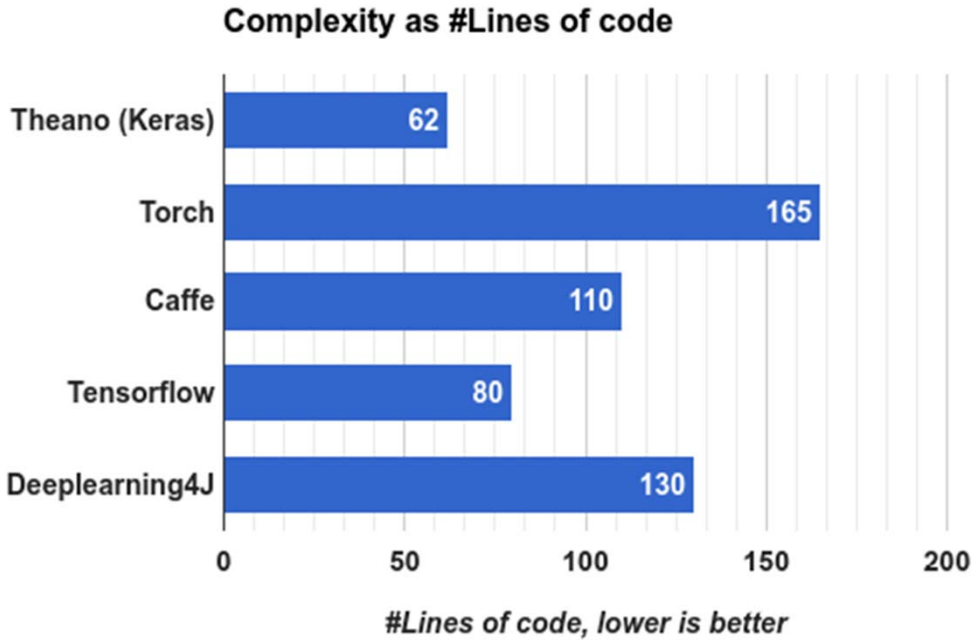
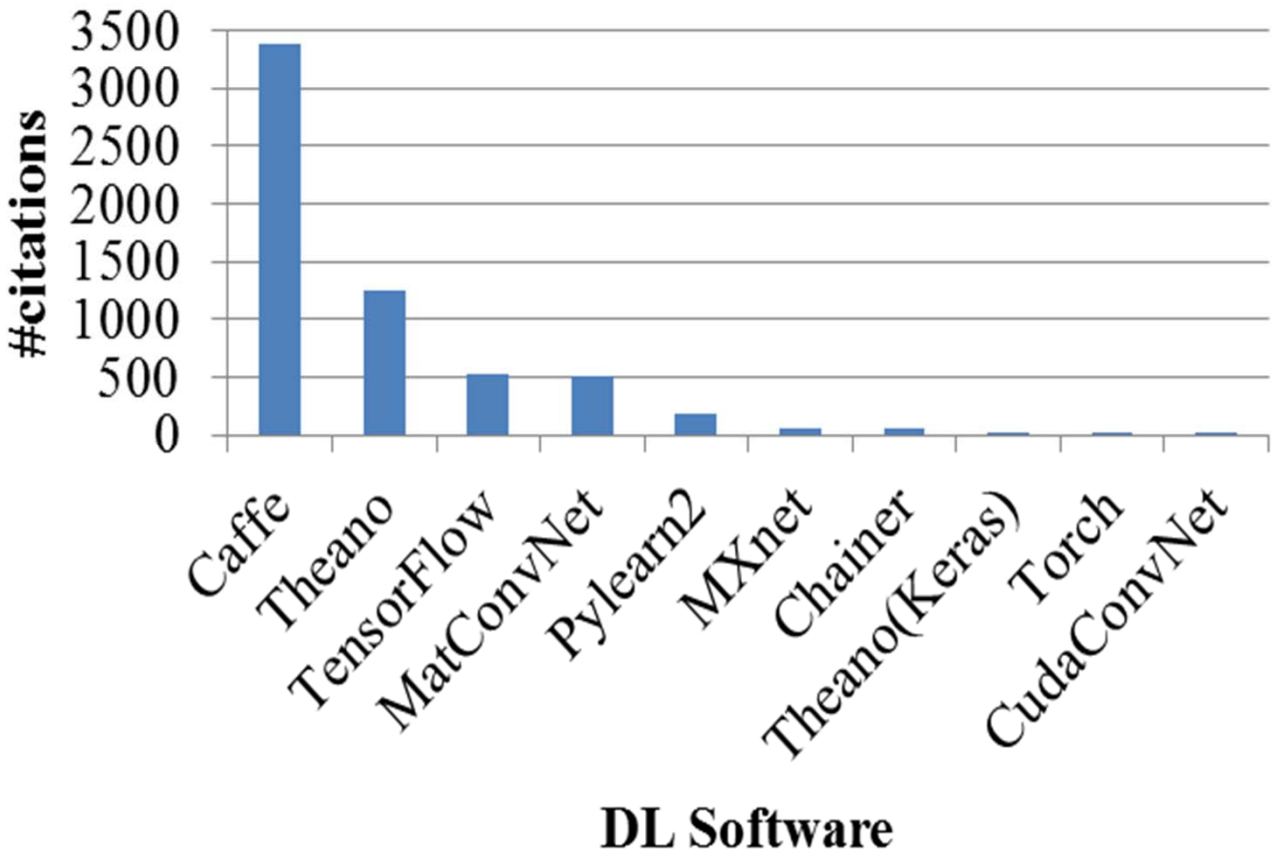


Fig.13 – Framework complexity, which measured as the number of lines of code needed to implement the algorithm (lower is better)

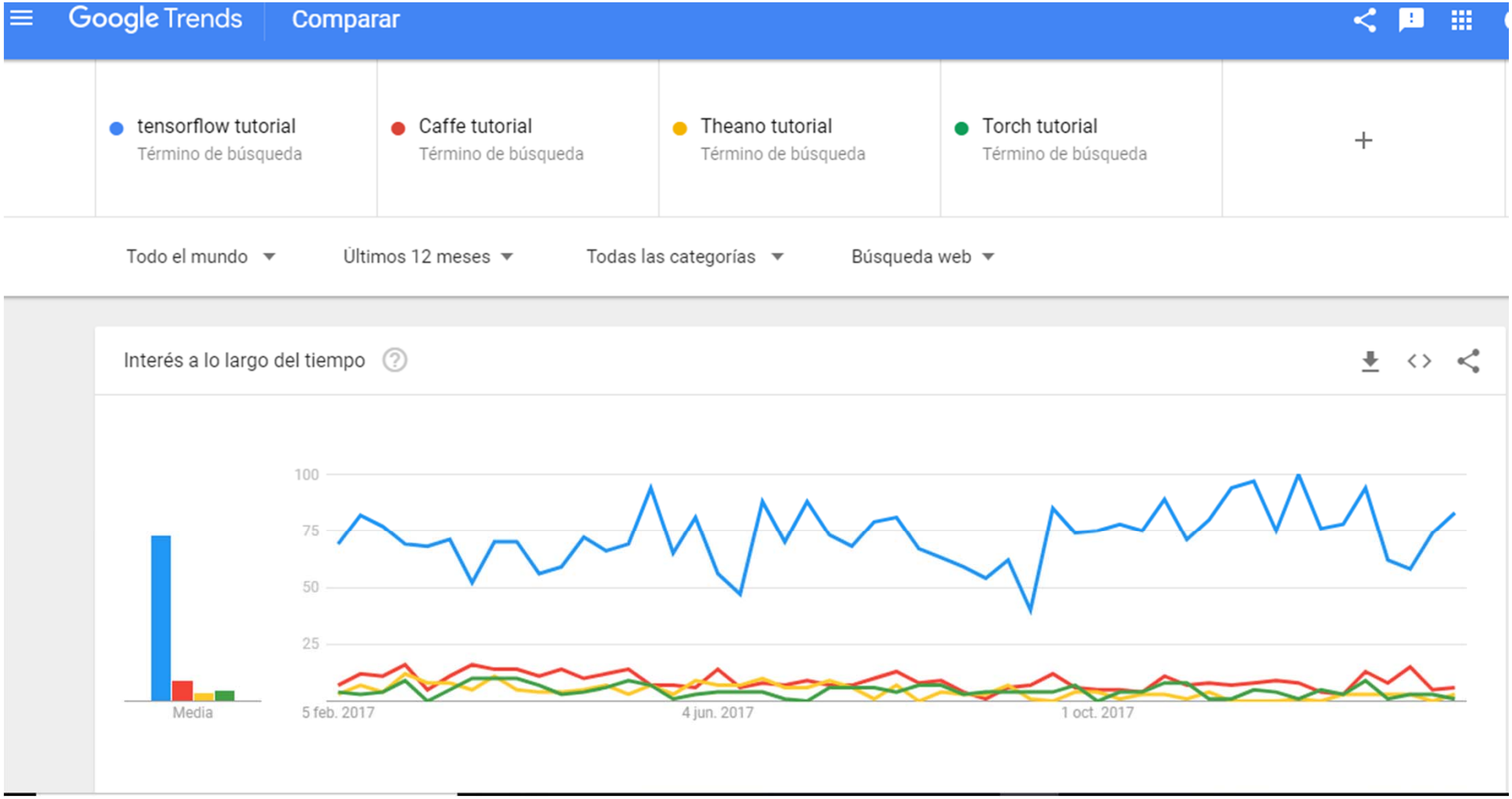
According to “Deep Learning with Theano, Torch, Caffe, Tensorflow, and Deeplearning4J: Which One is the Best in Speed and Accuracy? International Conference of Pattern Recognition and Information processing 2016.”

Top 10 according to Google scholar



First launch: theano(**2011**), Caffe(**2013**), tensorflow(Nov. **2015**)

Interest in DL frameworks

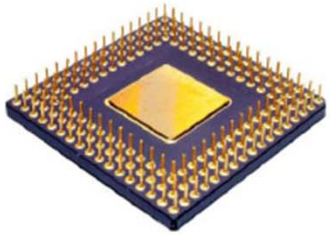


What is Tensorflow?

- Open source framework /library for **numerical computation** using **dataflow graphs**
- Originally developed by Google Brain team to conduct machine learning research
- With the word of G.B.team "*TF is an interface for expressing machine learning algorithms and an implementation for executing such algorithms*"

Why Tensorflow?

Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API



CPU

TPU

GPU

Android

iOS

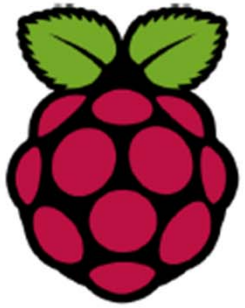
Raspberry Pi

Why Tensorflow?

Flexibility: from Raspberry Pi, Android, Windows, iOS, Linux to server farms





Why Tensorflow?

Python API:

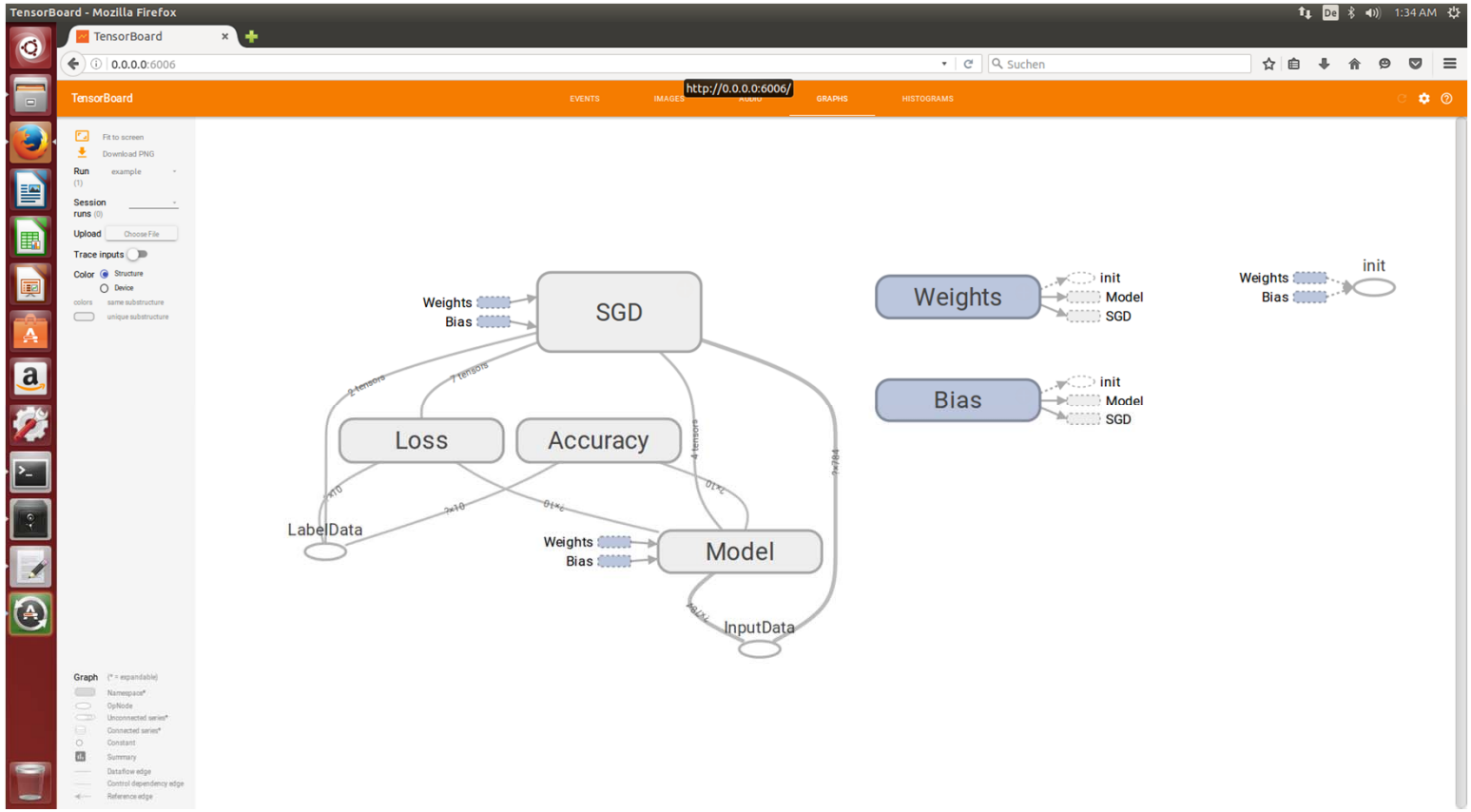
```
import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')

sess = tf.Session()
sess.run(hello) 'Hello, TensorFlow!'
sess.close()
```

Why Tensorflow?

- Gradient computation and backpropagation are hidden from the user
- Large community (+12,000 commits and +5000 TF-related repos since Nov. 2015)
- +570 contributers
- Used in ML classes in many universities: Berckley, Stanford, Toronto
- Awesome projects already using TensorFlow

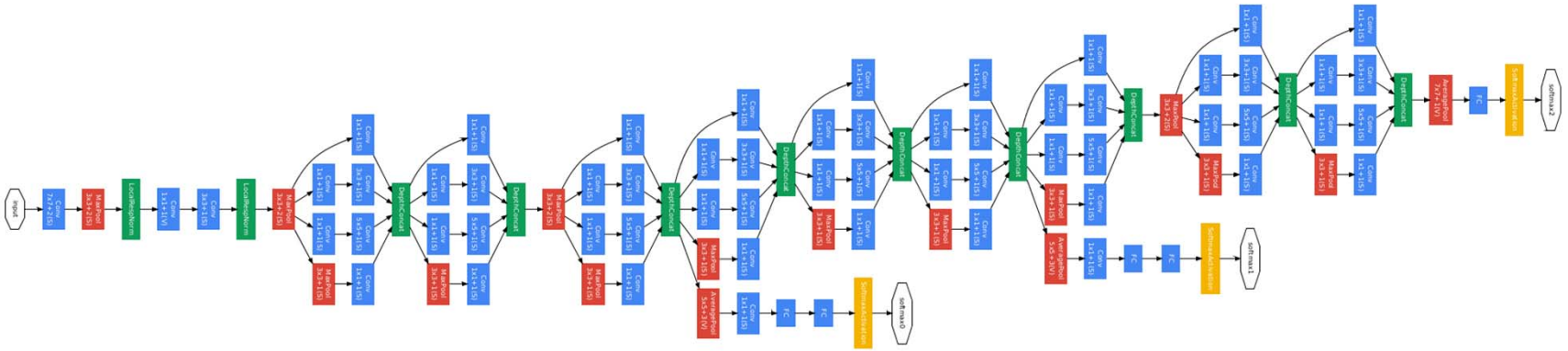
Why Tensorflow?

- Checkpoints (for managing experiments)
- Visualization (TensorBoard is da bomb)



Why Tensorflow?

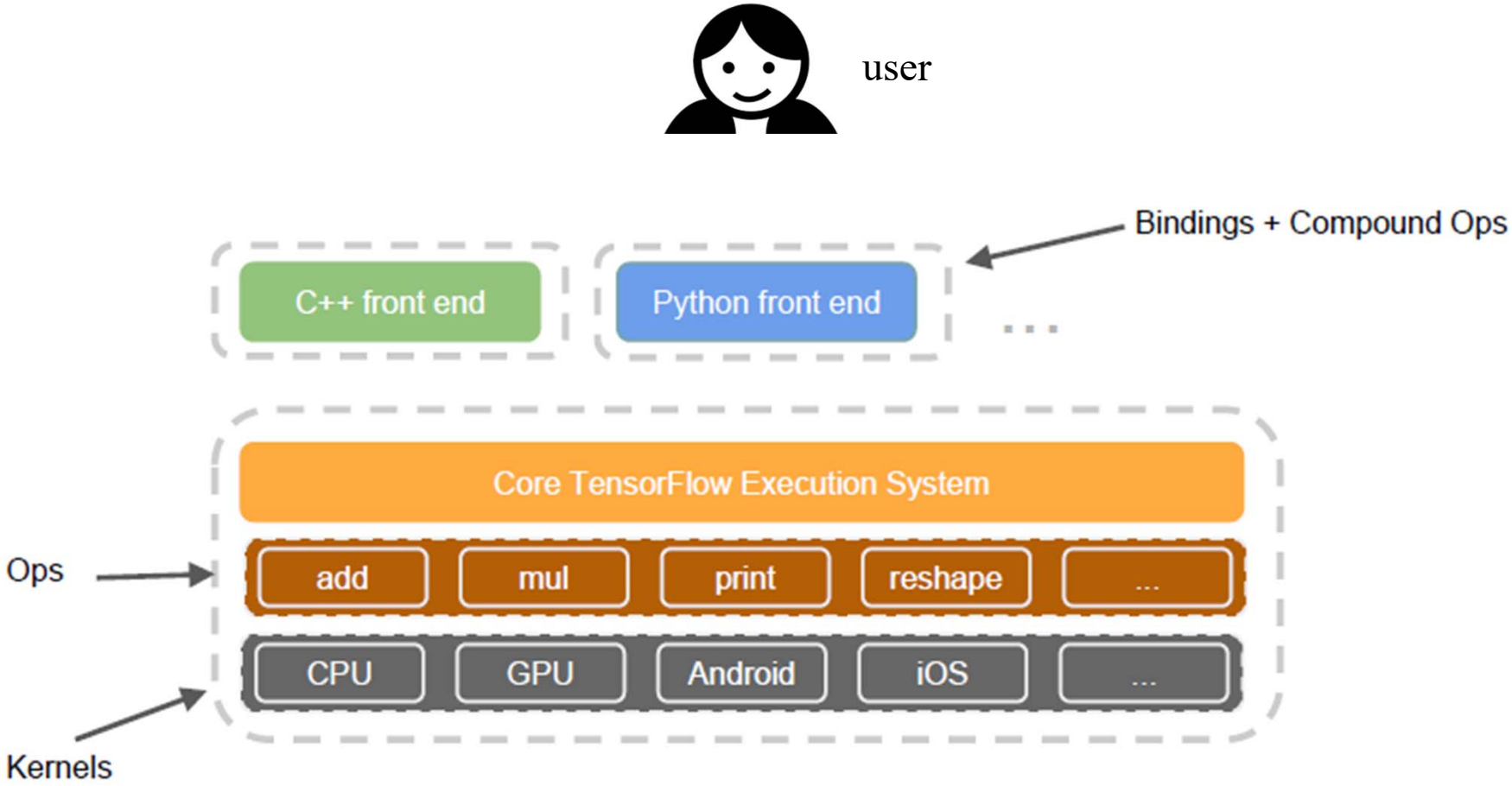
Complex models, e.g., Inception 2015



Why Tensorflow?

Companies using TensorFlow According to `WWW.tensorflow.org`

Tensorflow general architecture



How TF works?

TensorFlow separates definition of computations from their execution

- Graphs
- Sessions

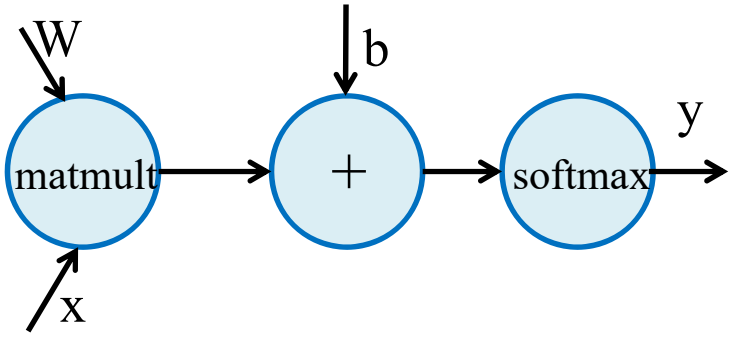
TF graph

Big idea: Express a numeric computation as a **graph**

- **Graph nodes** are **operations**, a function, which can have any number of inputs and outputs
- **Graph edges** are tensors which flows between nodes, n-dimensional arrays
  - Variables, the things that we want to tune to minimize the loss

TF graphs

$$y = softmax(Wx + b)$$

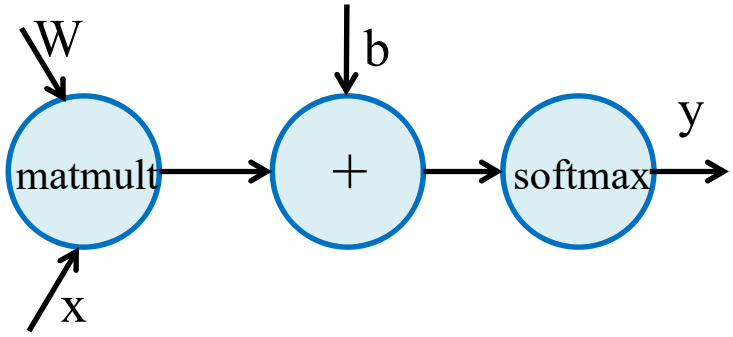


What is a tensor?

Edges, are called tensors**,** multi-dimensional arrays

- **Variables:** TF objects that hold and updates values, e.g., parameters of the net
- **Placeholders:** values whose value is fed in at execution time, e.g., input, labels
- A tensor is characterized by its rank, shape and type
  - Rank 0 is a scalar, 1 is a vector and 2 is a matrix
  - Shape: number of rows and cols
  - Type: data type, e.g., tf.float32

$$h = softmax(Wx + b)$$

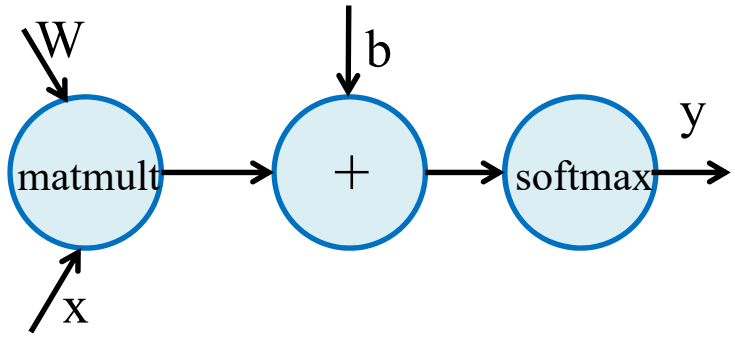


What is a node?

$$y = softmax(Wx + b)$$

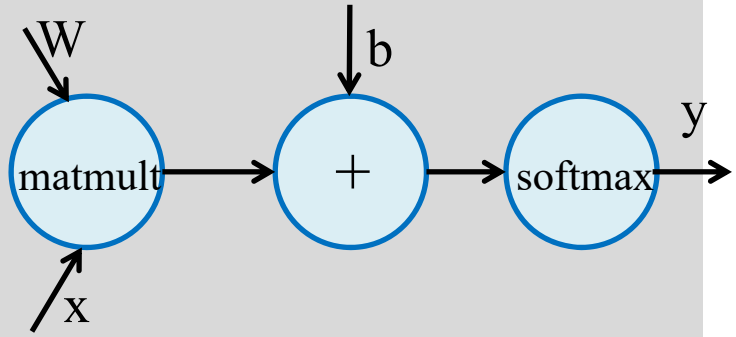
Nodes: Mathematical operations

- **MatMult:** Multiply two matrices
- **Add:** add elementwise
- **Softmax:** calculate probability distribution



How to define a graph in TF?

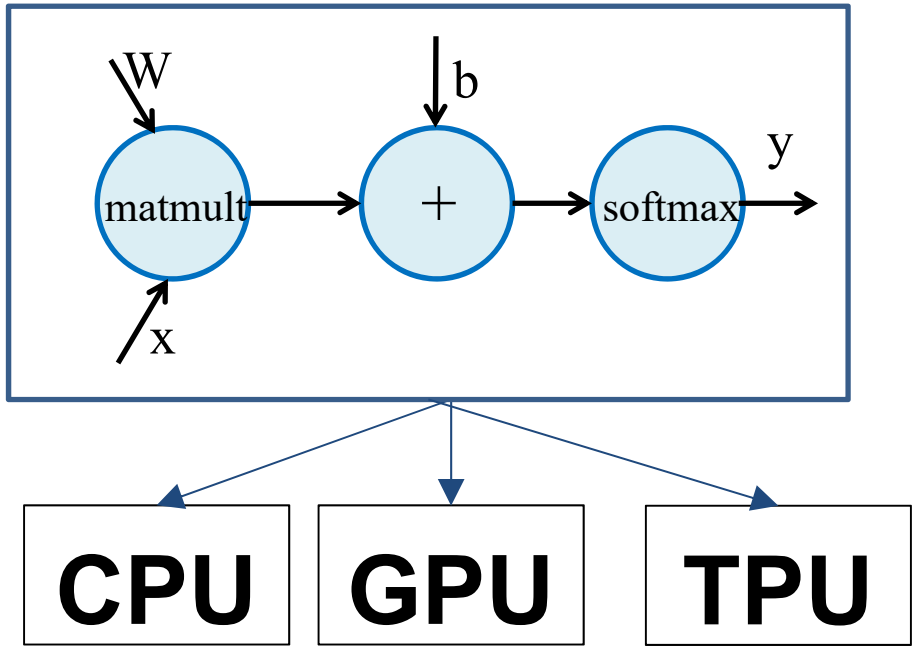
```
import tensorflow as tf

# Create two variables.
w= tf.Variable(tf.random_uniform([784, 10], -1,1))
b= tf.Variable(tf.zeros([10]))

# Create placeholder.
x= tf.placeholder(tf.float32, (None,784))

#Build flowgraph.
y=tf.nn.softmax(tf.matmul(x,w)+b)
```



But where is the graph?

How do we run the graph?

So far we have defined a **graph**.
We can deploy this graph with a**session**: a binding to a particular execution context (e.g. CPU, GPU)



How to run the graph?

```
# Launch the graph in a session.
with tf.Session() as sess:
      sess.run(tf.initialize_all_variables())
      sess.run(y,{x: np.random.random(100, 784)})
```

Next we will see how to **train** the **model**

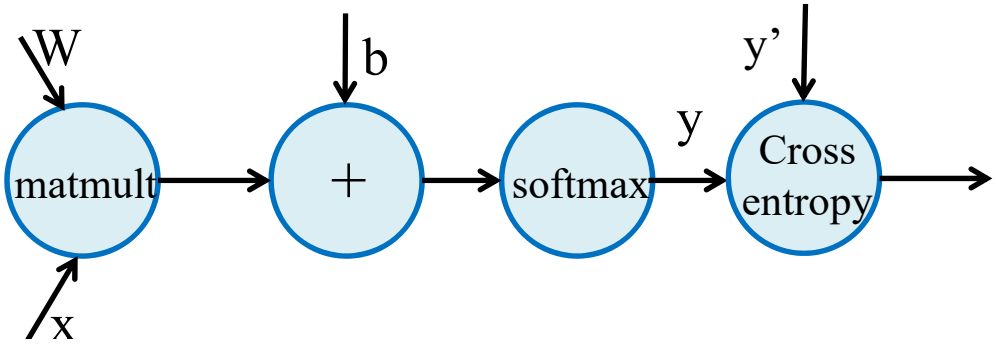
Ho do we define the loss?

Use placeholder for **labels** (groundtruth): y'

Build loss node using **labels** and **prediction** y

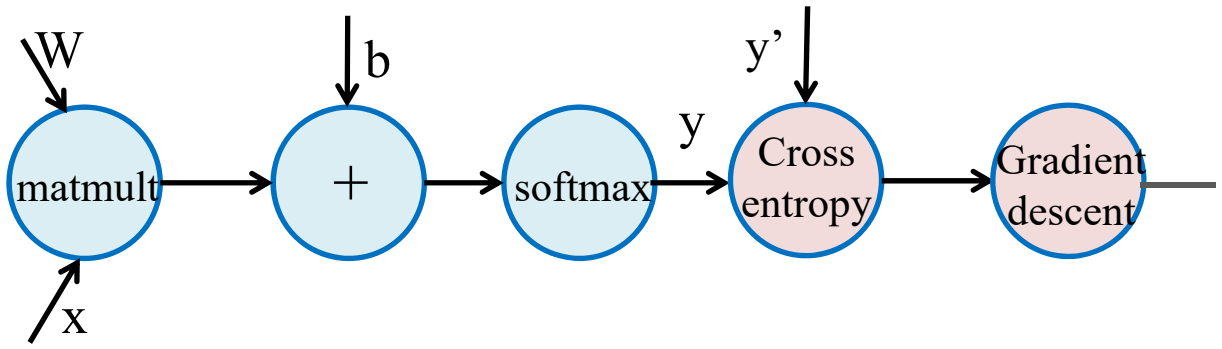
Cross_entropy= $-reduce\_mean(\sum_{k=0}^{n} y' \log(y))$

```
#output of the NN: the predition
y = tf.nn.softmax(tf.matmul(x,w)+b)
#the correct answer.
y'= tf.placeholder(tf.float32,[100,10])
#create the cross entropy node.
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y'* tf.log(y),
reduction_indices=[1]))
```



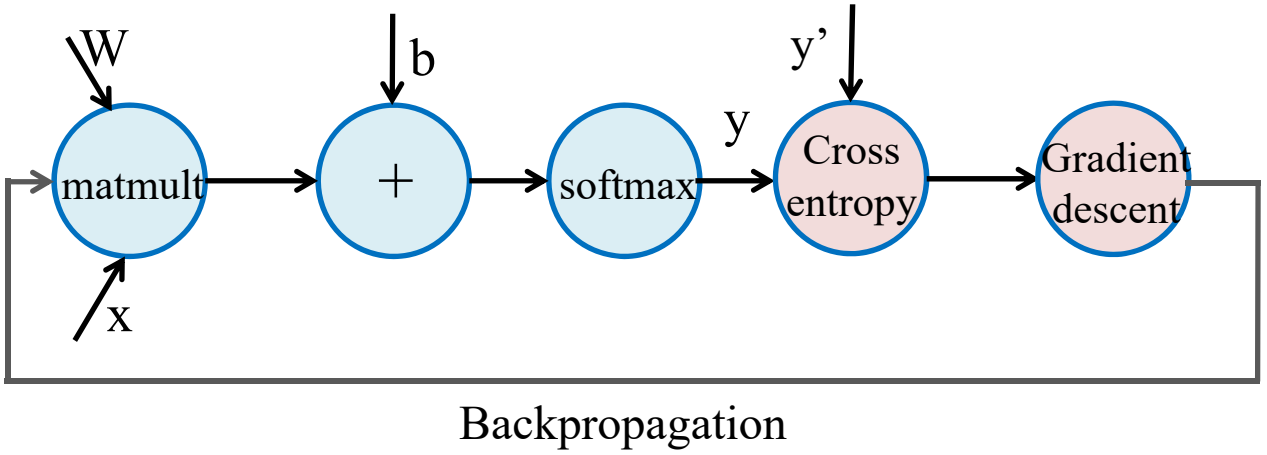
Ho do we compute the gradients?

```
#tf.train.GradientDescentOptimizer is an Optimizer object

#adds optimization operation to computation graph

tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy)
```



How backpropagation is done?

Automatically!



The pseudo-code in TF

```
import tensorflow as tf

# Create two variables.
w= tf.Variable(tf.random_uniform([784, 10], -1,1))
b= tf.Variable(tf.zeros([10]))

# Create placeholder.
x= tf.placeholder(tf.float32, (None,784))

#Build the flowgraph of softmax(..)
y=tf.nn.softmax(tf.matmul(x,w)+b)

#output of the NN: the prediction
y = tf.nn.softmax(tf.matmul(x,w)+b)
#the correct answer.
y'= tf.placeholder(tf.float32,[100,10])
#create the cross entropy node in the graph.
cross_entropy = tf.reduce_mean(-tf.reduce_sum(y'* tf.log(y), reduction_indices=[1]))
#adds optimization operation to computation graph
Optimizer=tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy)

# Launch the graph in a session.
with tf.Session() as sess:
            sess.run(tf.initialize_all_variables())
            for epoch in range(training_epochs):
            sess.run(optimizer,feed_dict={X:batch_x,y'=batch_y})
```

Building a CNN

In addition to what we have seen before

- `tf.nn.conv2d()`
- `tf.nn.relu()`
- `tf.nn.max_pool()`
- `tf.nn.dropout()`

Building a CNN

- tf.nn.conv2d(input,filter,strides,

  padding,use_cudnn_on_gpu=True,

  data_format='NHWC',dilations=[1,1,1,1],name=None)

Input:

  input[batch, in_height, in_width, in_channels]

  filter[filter_height * filter_width * in_channels, output_channels]

Output:

Building a CNN

- `tf.nn.conv2d(input,filter,strides,`
  `padding,use_cudnn_on_gpu=True,`
  `data_format='NHWC',dilations=[1,1,1,1],name=None)`

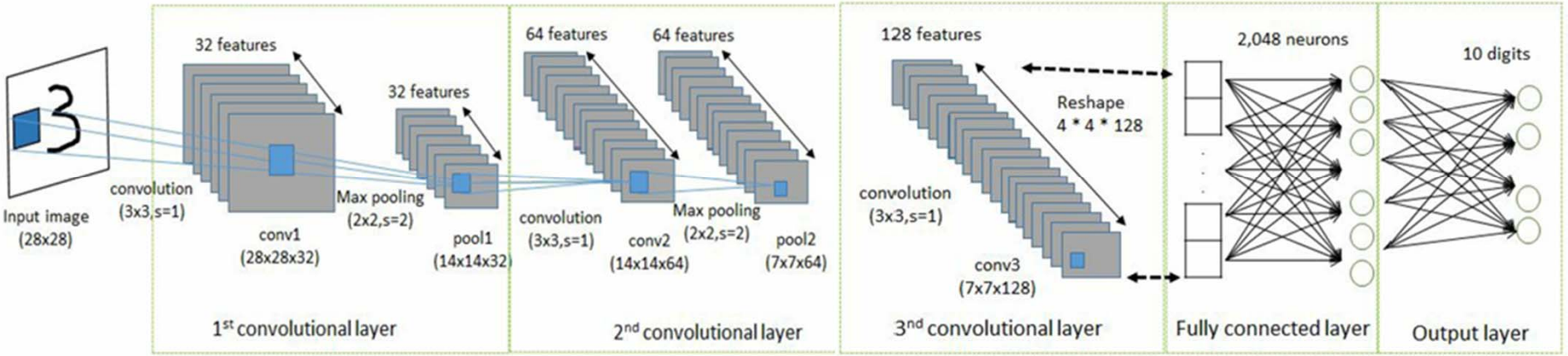
input[batch, in_height, in_width, in_channels]

filter[filter_height * filter_width * in_channels, output_channels]

- `tf.nn.relu(features,name=None)`
- `tf.nn.max_pool(value,ksize,`
  `strides,padding,`
  `data_format='NHWC',name=None)`
- `tf.nn.dropout(x,keep_prob,noise_shape=None,`
  `seed=None,name=None)`

Building my first CNN

- Implement this Lenet-like CNN
- Explore the data-augumentation techniques that could improve the accuracy of your model
- Explore the possibility of using transfer-learning



- Material: https://sihamtabik.github.io/tutorial.html