AIDL: PROJECTS

Session 1 (2019-03-05): Introduction

Issey Masuda Mora

Education: Telecommunications Engineer

Currently working: Deep learning team lead at Vilynx

Technologies:

Hand-raised poll

- Programming skills?
- Working on software development? Or management?

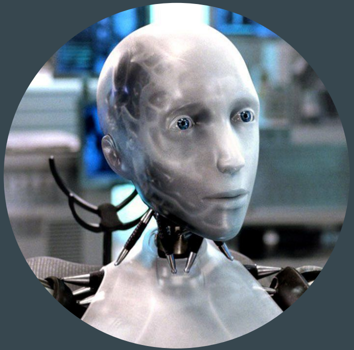
Subject goals

“Give them the tools to apply DL in the industry”

Expectations: what not to expect








Expectations

- Implement NN models
- Ready to prod
- Useful for current job
- Product MVP
- Solve real-world problems
- State of the art

Subject goals

Subject goals

Tangibles:

- Implement DL models
- Optimize resource usage / increase efficiency
- Optimize models
- Deploy to production

Intangibles:

- Add a DL implementation into the roadmap
- Acquire a data scarcity mindset
- It’s not all about DL!

Course overview

How is this all going to work?!?!?

Deep learning

6th of May

1st of July

MLP

Image classification

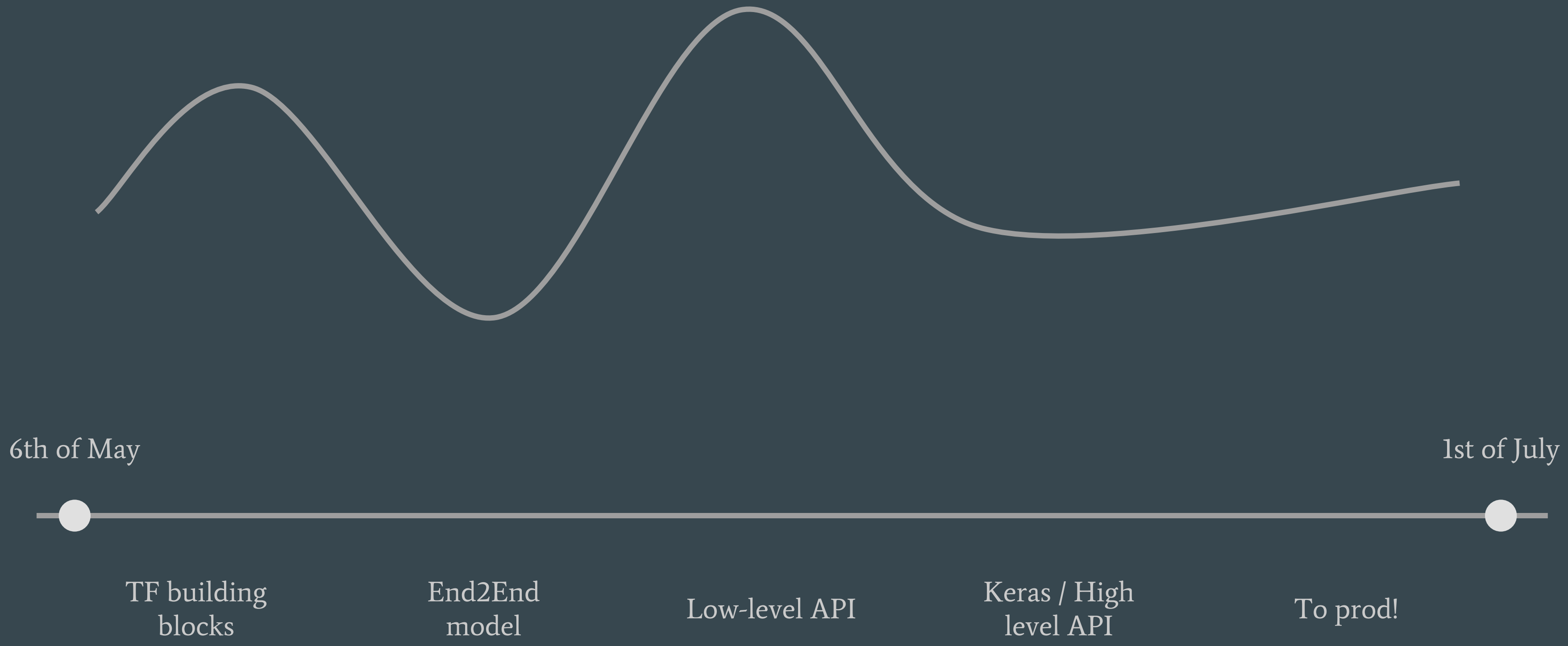
CNN

Object detection

RNN

Pro tip: GAN?

Tensorflow


6th of May
1st of July
TF building blocks
End2End model
Low-level API
Keras / High level API
To prod!

Classes

Classes

What they should be:

- Guide + DIY + review solution → what did we learn?
- Align each one with the subject goals

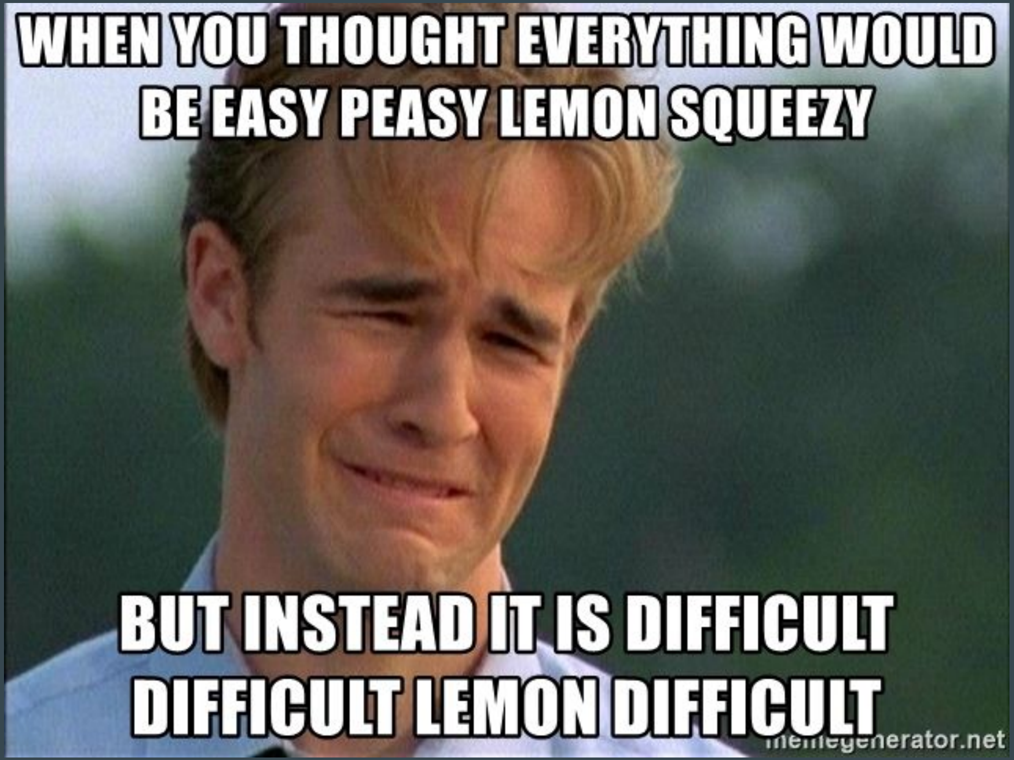
What they should not be:

- Medium blog post
- Tensorflow tutorial
- TF API reference summary

Classes

~~Jupyter & Codalab~~

Old & good python



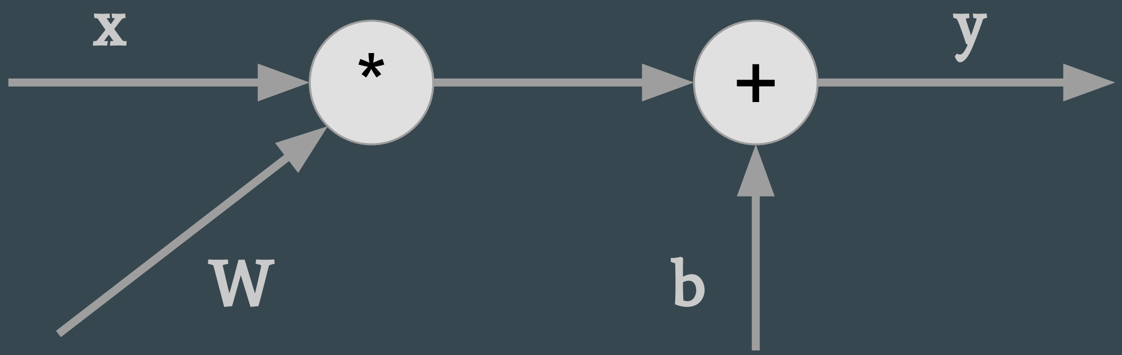
Begin!

Computational graph

What is it?

Nodes: operations

Edges: data / tensors

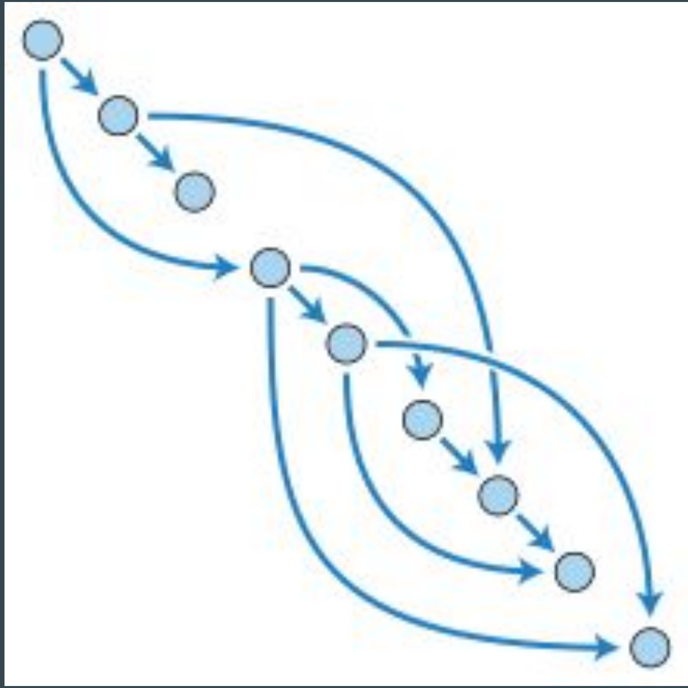


Example

Linear regression: y = Wx + b

Directed Acyclic Graph (DAG)

- Edged are directed from one node to another
- Nodes have a topological ordering
- No closed loops!



Static computational graphs

Define-and-Run methodology.

Pros:

- Speed: can spend a long time optimizing the graph
- Memory: can predict and allocate all mem ahead of time

Cons:

- Inflexible: once compiled, it cannot be modified at run time
- Debugging: graph representation doesn't match code
- Static: more difficult to support flexible sized inputs

Slide credit: Kevin McGuinness (adapted from the original one)

Dynamic computational graphs

Define-by-Run strategy

Pros:

- Can change structure of NN at runtime (add layers, change shape, etc.)
- Support flexible sized inputs
- Easier to debug
- More natural to code

Cons:

- Compile at runtime, can be slower
- Dynamic batching more difficult

Slide credit: Kevin McGuinness (adapted from the original one)

Deep learning frameworks

torch

Keras

theano

DEEPLEARNING4J

Caffe2

TensorLayer

TensorFlow

PYTORCH

Chainer

Lasagne

Caffe

Microsoft CNTK

Sonnet

mxnet

Framework basic comparison

Tensorflow:

- Production ready
- Multiple languages supported (Python, C, Java, JavaScript, Go)
- Multi device (Android, iOS, Web browser)
- Mature with a big community
- Tensorboard
- Keras
- Static graph

... Tensorflow 2.0 coming soon! → dynamic graph

Pytorch:

- Pythonic
- Easy to learn
- Framework rather than library
- Integration with Caffe2
- Dynamic graph

... Pytorch 1.0 finally released! → production ready

Surrounding technologies

Git

- Course repository (for the coding sessions)
- Use a personal branch to keep track of your progress
- Master branch will be updated each week with the solutions of previous sessions & required material for the current one

Virtualenv & virtualenvwrapper

- Isolate python environment

Docker

- Development reproducibility
- Containerize production environment
- TF Serving

NVIDIA

CUDA: “is a parallel computing platform and application programming interface (API) model created by Nvidia”

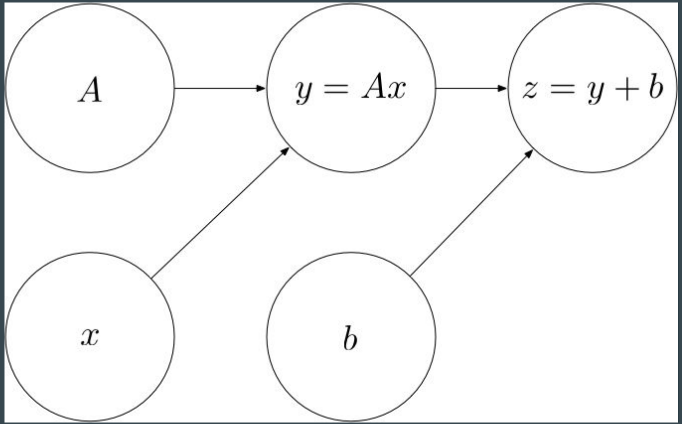
cuDNN: “is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers”

TF preview

Computational graph & Dataflow

Programs are represented as directed graphs with data flowing through them where:

- **Nodes**: Operations of the program → **tf.Operation**
- **Edges**: Data flowing through the graph → **tf.Tensor**

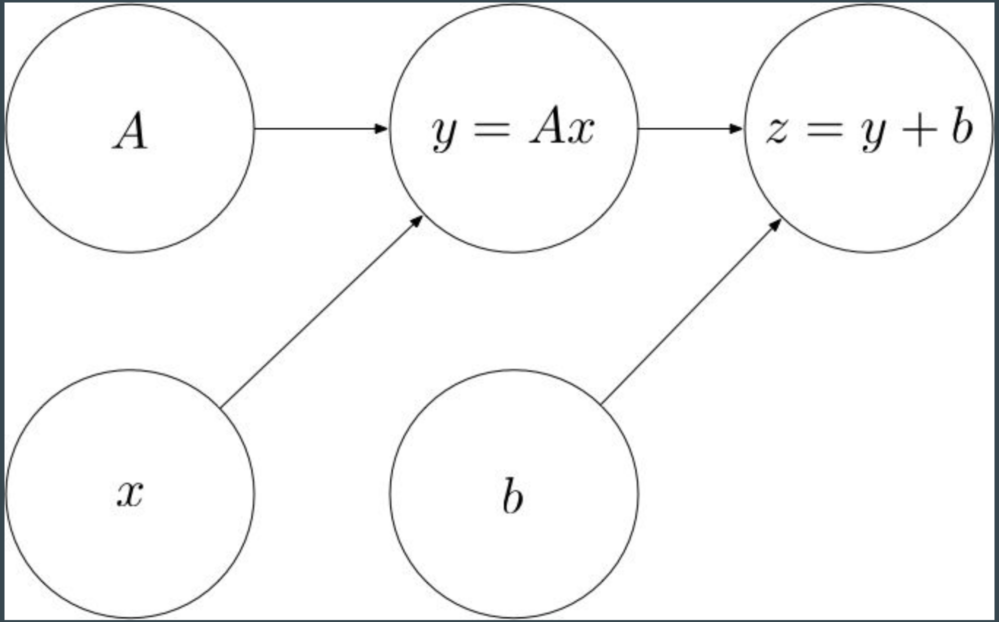


From symbolic to real numbers

Two phases:

- **Definition phase**: build the computational graph → **tf.Graph**
- **Execution phase**: interact with the graph feeding data and fetching results.
  Run a subgraph of the original graph & change its state → **tf.Session**

Micro example I: the graph



Micro example II: the code

```
import tensorflow as tf
import random

graph = tf.Graph()
with graph.as_default():
     x = tf.placeholder(tf.float32, shape=[], name='x')
     A = tf.get_variable('A', shape=[], dtype=tf.float32,
initializer=tf.initializers.random_normal())
     b = tf.random_normal(shape=[], dtype=tf.float32, name='b')
     y = A * x
     z = y + b

with tf.Session(graph=graph) as sess:
     sess.run(tf.global_variables_initializer())
     result = sess.run(z, feed_dict={x: random.random()})
```

Fetching

Feeding

Definition phase

Execution phase

Demo time

Questions?