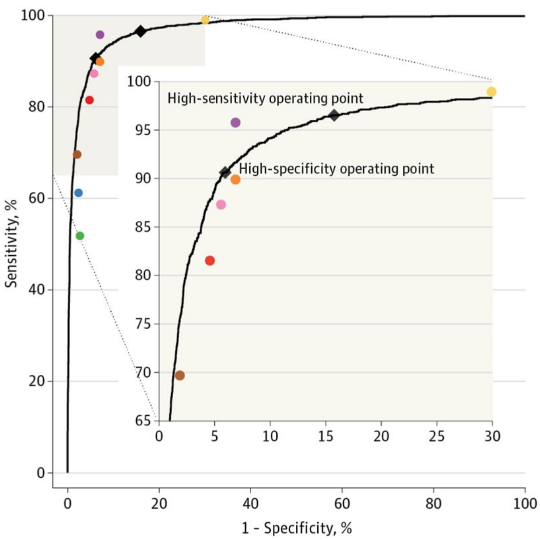
From PyTorch to TensorFlow

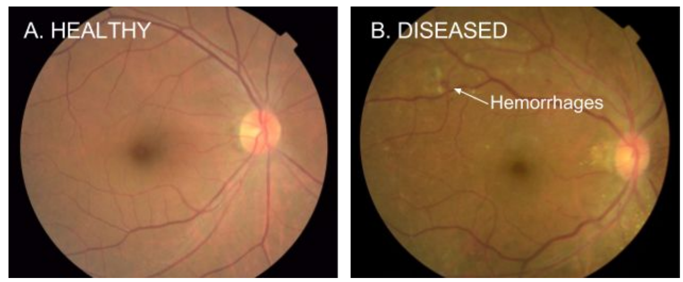
EE-559 Deep Learning, EPFL, 5/23/18
Andreas Steiner, Guest Lecture
https://fleuret.org/dlc

Agenda

Today	
Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ?

Diabetic Retinopathy





(Gulshan 2016)

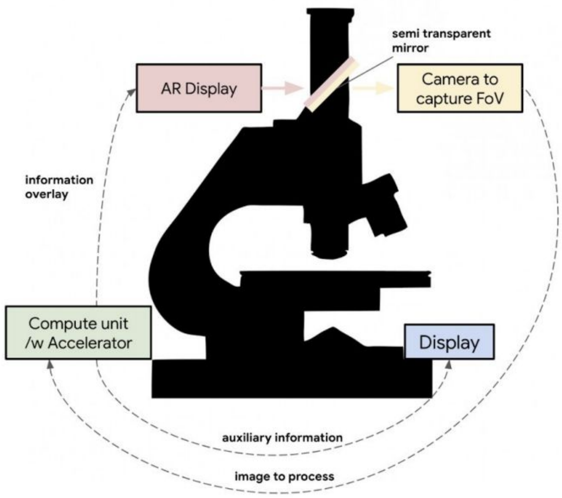
Pathology: Tumor Detection

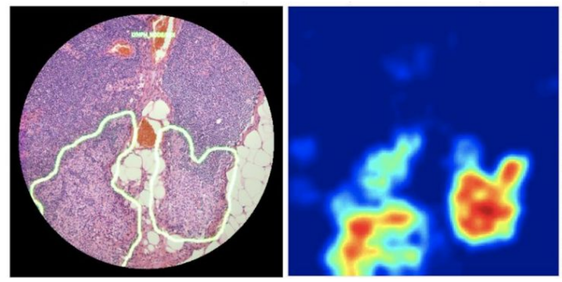
- Inception v3 network, sliding window
- 242 slides - 10^7 299x299 patches
- 82.7% sensitivity (@7 FP/slide) vs. 73.2% pathologist

100,000 pixels   label   prediction

(Liu 2017)

Pathology : Augmented Reality

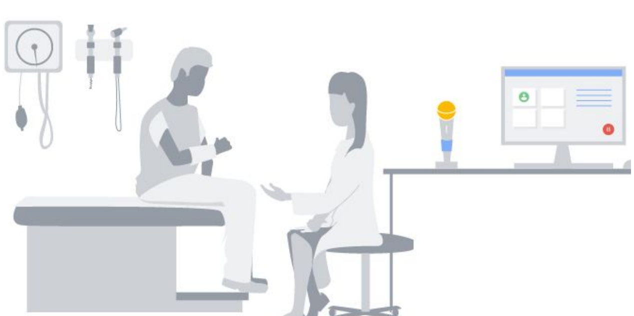




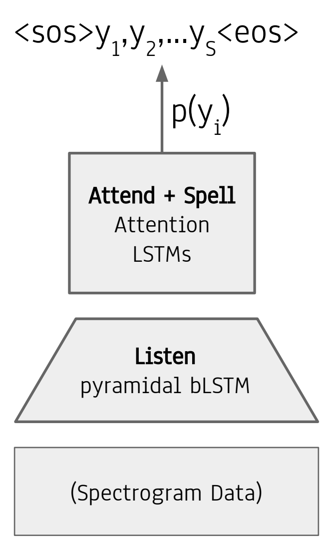
(Po-Hsuan 2018)

Transcribing Medical Records

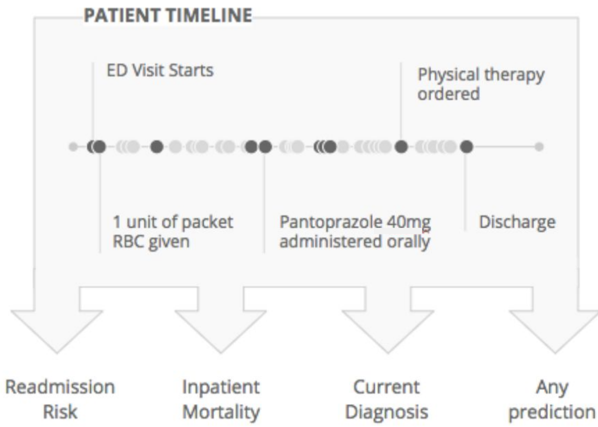
- ~6/11h workday spent on electronic health records
- 14,000 hours anonymized conversations
- Attention model : 18% word error rate



(Chiu 2017)
(Chan 2015)



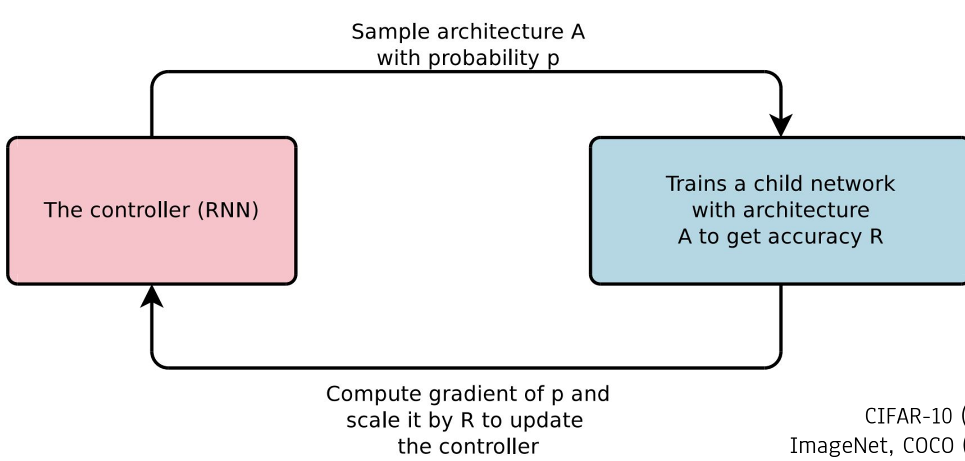
Medical Records



- Predictive modeling from EHR
- Deep learning from raw FHIR data
- 216k patients
  46,864M datapoints
- Outperforms state-of-the-art in all predictions

(Rajkomar 2018)

AutoML



CIFAR-10 (Zoph 2016)
ImageNet, COCO (Zoph 2017)
Optimizers (Bello 2017)
Activation Functions (Ramachandran 2017)

Datasets

Youtube 8M
https://research.google.com/youtube8m/index.html
7M videos, 4716 classes

Youtube BB
https://research.google.com/youtube-bb/
240k videos, 11M human annotations

Atomic Visual Actions
https://research.google.com/ava/
58k segments, 210k action labels, 80 actions

AudioSet
https://research.google.com/audioset/
2.1M human-labeled 10s sound clips, 632 classes

Speech Commands
https://ai.googleblog.com/2017/08/launching-speech-commands-dataset.html
65k utterances, 30 words

OpenImages V4
https://storage.googleapis.com/openimages/web/
9.2M images, 30M human-verified labels, 20k classes
1.9M images, 15M bounding boxes, 600 classes

distill.pub

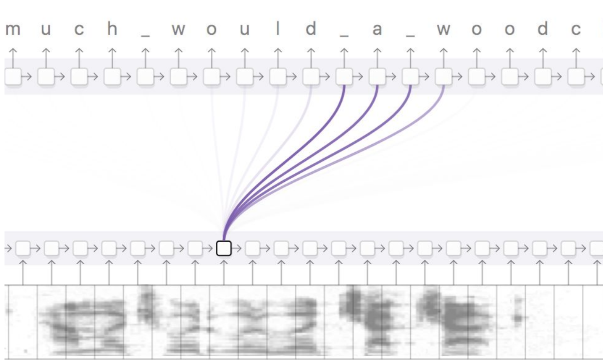
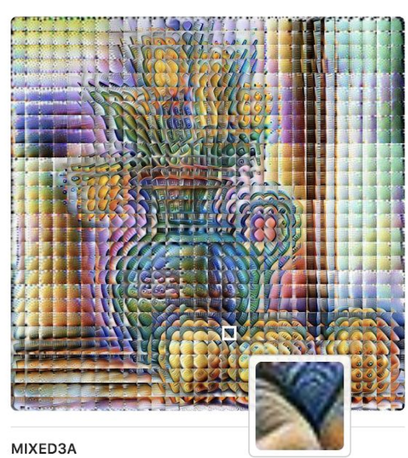


Figure derived from Chan, *et al.* 2015

https://distill.pub/2016/augmented-rnns/
(Olah 2016)



https://distill.pub/2018/building-blocks/
(Olah 2018)

AI Residency

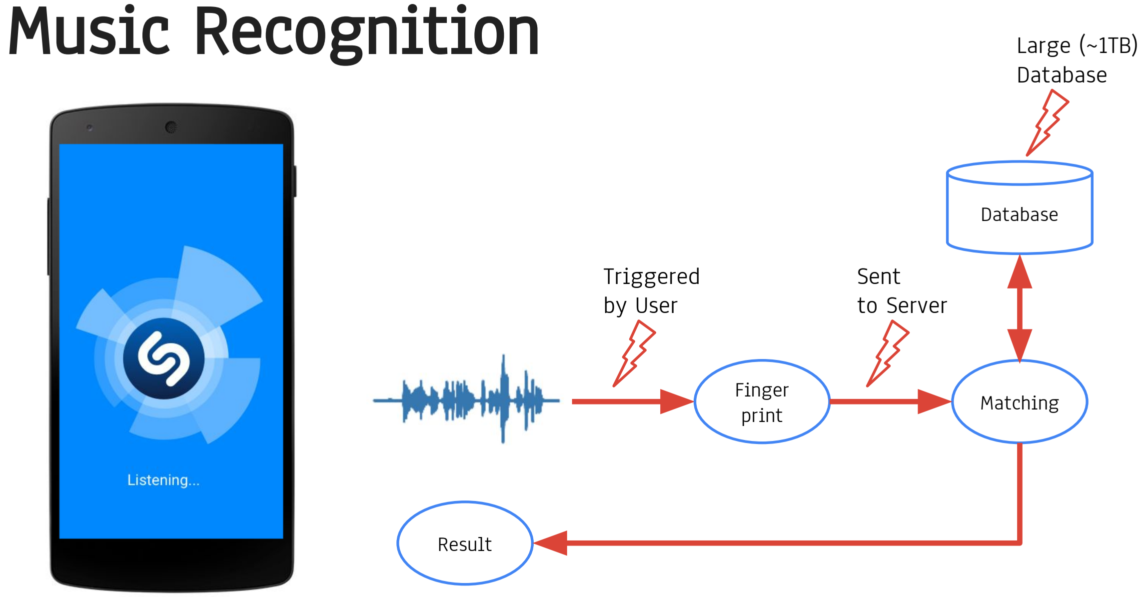
Ad g.co/airesidency

- 12-month research training role
- For whom?
  Candidates with BSc/MSc/PhD in STEM field
  … or any background with passion for ML.
- Applications for 2019 will open in Fall 2018
  g.co/airesidencysignup

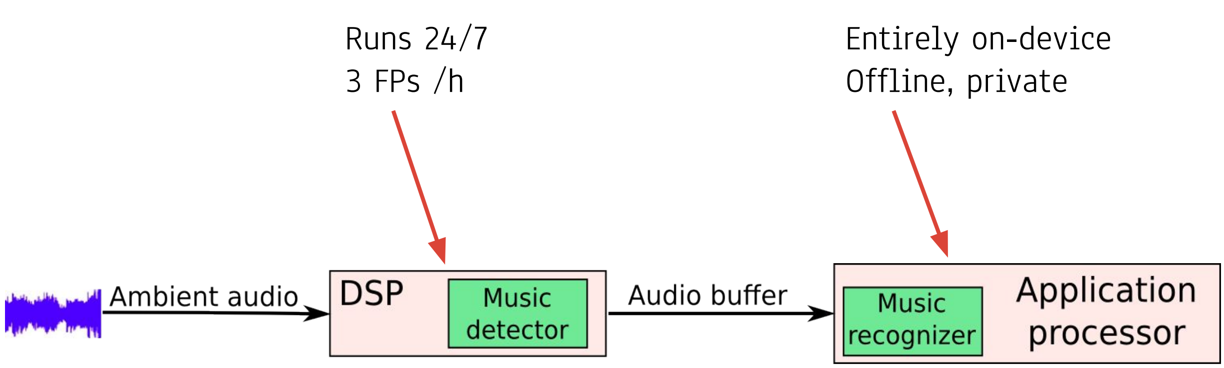
Agenda

Today	
Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ?

Music Recognition

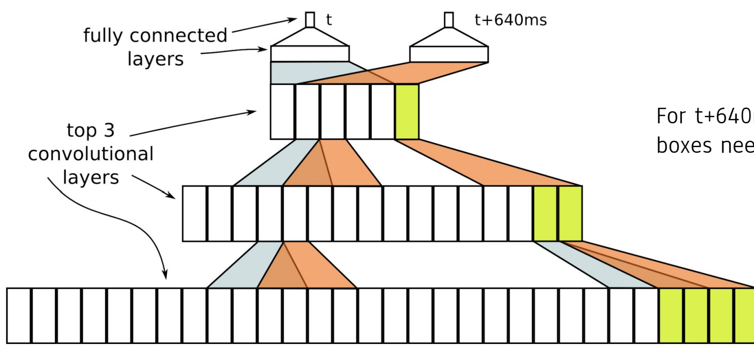


Music Recognition 2.0



(Agüera y Arcas 2017)

Music Detector



For t+640ms only green boxes need to computed.

- 6 convolutional layers (stride=2) - prediction every 0.64s
- 8k parameters - 10k memory footprint (quantization!)
- Trained on AudioSet data

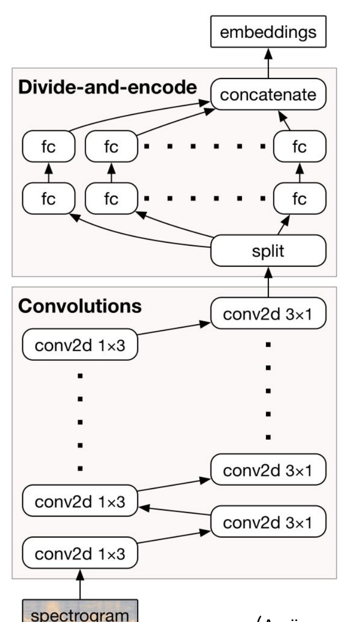
(Agüera y Arcas 2017)

Music Recognizer

Neural Network Fingerprinter

- Same input as Music Detector
- Triplet loss, 96-dim embeddings

Reduced size by:
- Separable convolutions
- FC : divide-and-conquer
- Quantization



(Agüera y Arcas 2017)

Now Playing (on Pixel 2)

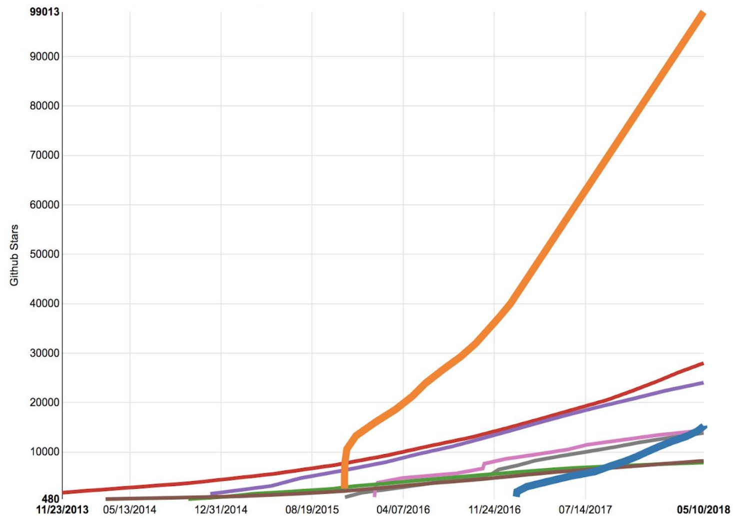
10k songs, <1% daily battery.

- On-device ML, private, always on.
- Requires tricks for reducing memory / power (cascading models, optimized models, hardware)

Agenda

Today	
Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ?

Stars on Github

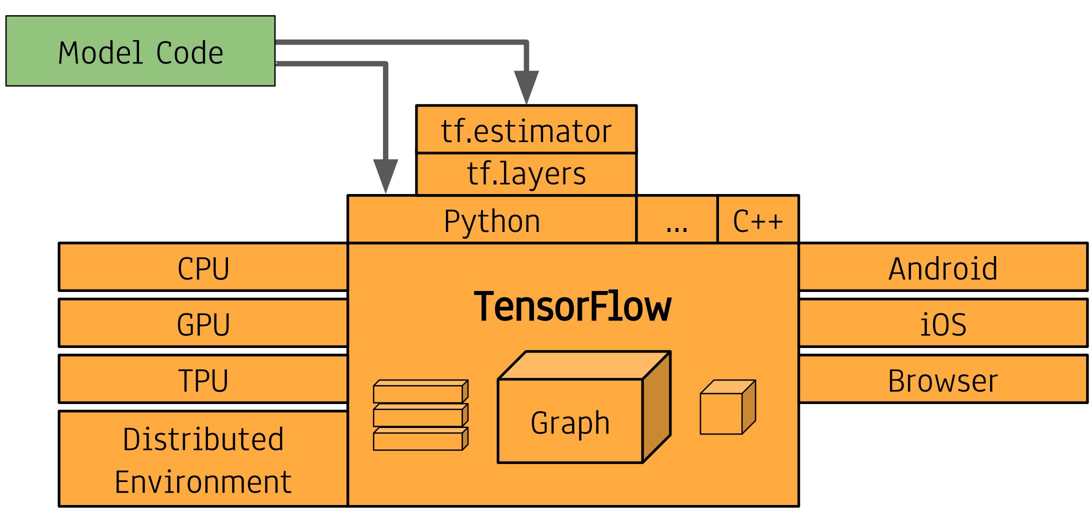


as of 5/21/18

7,896 torch/torch7
8,190 Theano/Theano
13,920 apache/incubator-mxnext
14,430 Microsoft/CNTK
24,150 BVLC/caffee
28,170 scikit-learn/scikit-learn
15,630 pytorch/pytorch
100,259 tensorflow/tensorflow

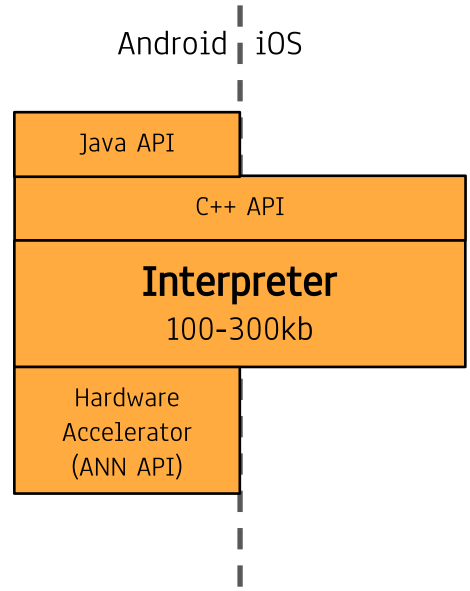
http://timqian.com/star-history

Dealing With Complexity

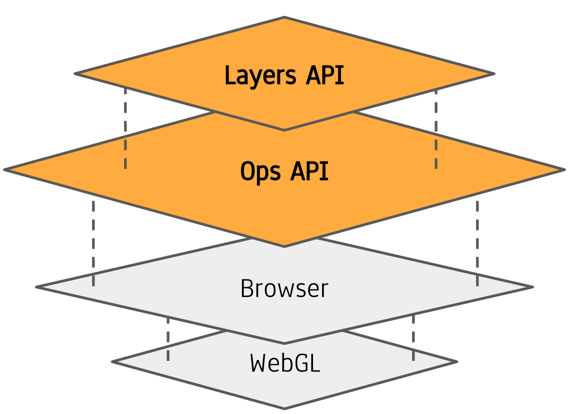


TensorFlow Lite

- Lightweight
- Low latency
- Operators (float/quantized) tuned for mobile performance
- Convert existing models to .tflite format



TensorFlow.js



- Supports 90+ Ops, 32+ layers
- Train / run in browser
- Import SavedModel/Keras models
- ~1.5-2x slower than Python/AVX

Try it out:
https://js.tensorflow.org

TensorFlow-Hub

```
import tensorflow_hub as hub

embed = hub.Module("https://tfhub.dev/google/"
    "universal-sentence-encoder/1")

embedding = embed([
    "The quick brown fox jumps over the lazy dog."])
```

(Cer 2018)

- Module : self-contained piece of TF graph including weights and assets.
- Canonical URLs for versioned modules.
- For direct application, transfer learning, ...
- **Beware** of model bias !

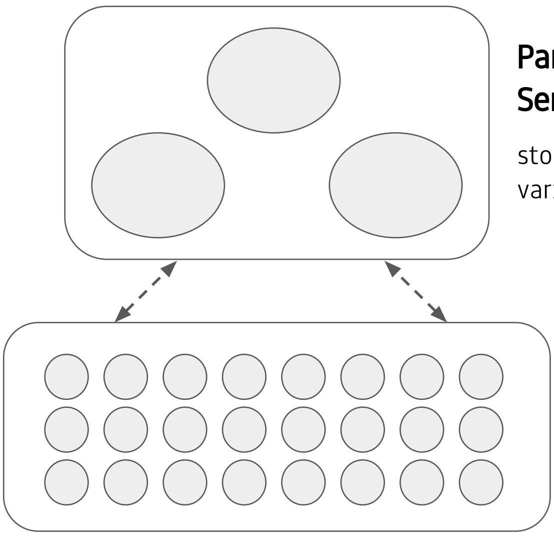
Many more models under github.com/tensorflow/models

Tensor Processing Units (TPUs)

(Jouppi 2017)

180 TFLOPS, 64 GB HBM memory, 2400 GB/s BW – g.co/tpusignup

Distributed TensorFlow



Parameter Servers

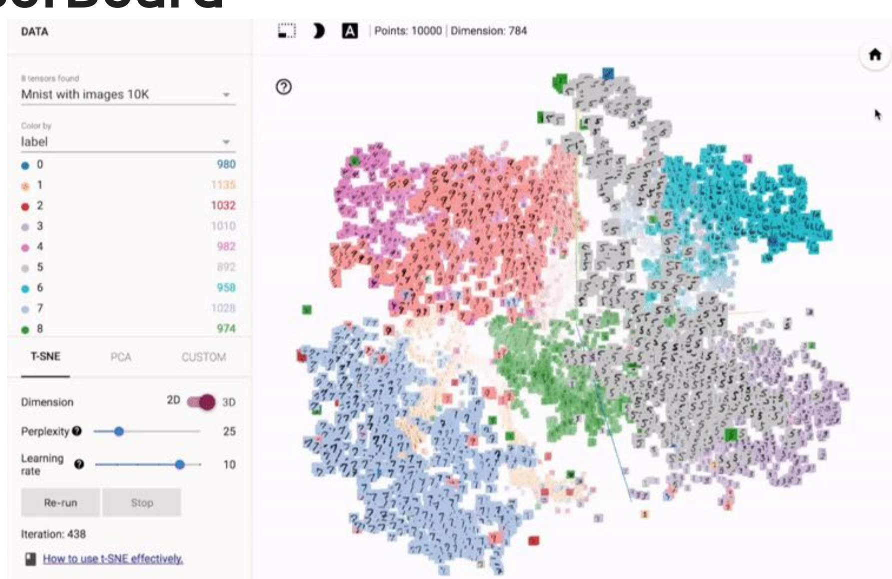
store/update variables

Workers

do computation

- Lot of flexibility.
- Communication within process / over network.
- Graph can be split or replicated.
- Easy to use with higher level API.

TensorBoard



Agenda

Today

Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ?

Low-Level TensorFlow - Overview

- The Graph
- The Session
- Shapes
- Variables

Code

```
import tensorflow as tf
print(tf.__version__)
```

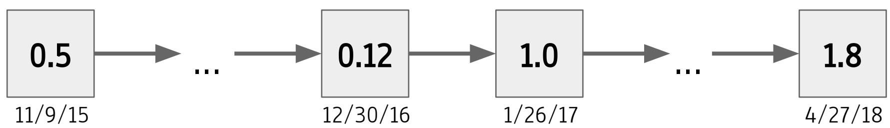
```
1.7.0
```

Output

Find all code from slides here:
https://goo.gl/EKVKWz

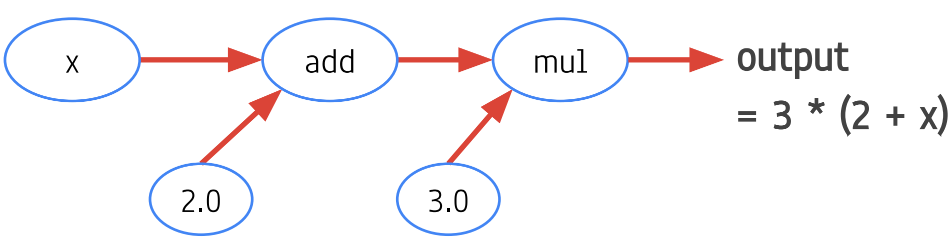
TensorFlow Versions

Semantic Versioning : MAJOR.MINOR.PATCH
https://github.com/tensorflow/tensorflow/releases
https://github.com/tensorflow/tensorflow/blob/master/RELEASE.md



The Graph*

Ops - Nodes in the graph. Perform computation on Tensors
Tensors - Edges in the graph. Input/output of Ops

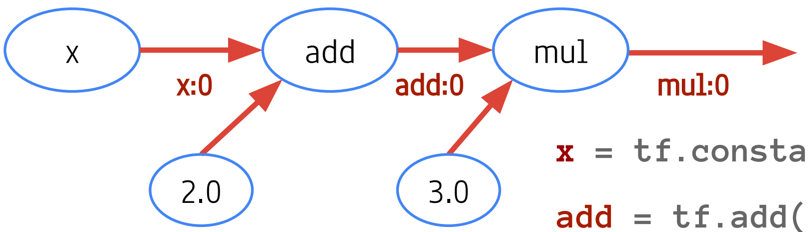


* always present, but might be invisible

The Graph

Ops - Nodes in the graph. Perform computation on Tensors

Tensors - Edges in the graph. Input/output of Ops

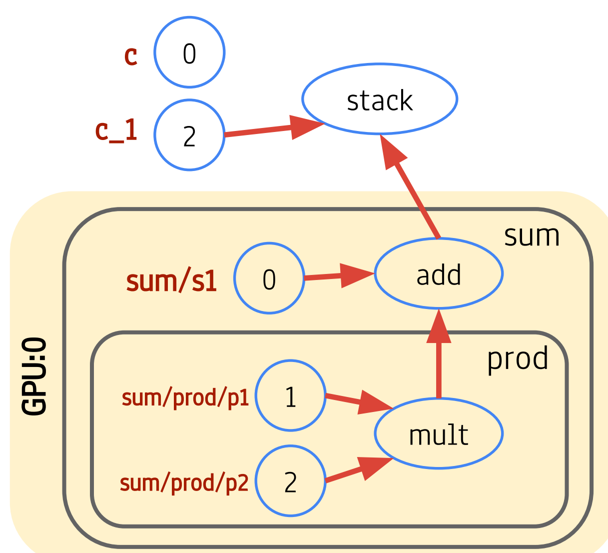



```
x = tf.constant(1.0, name="x")
add = tf.add(x, 2.0, name="add")
mul = tf.multiply(add, 3.0,
                  name="mul")
```

Naming, Placement, Scope

```
c = tf.constant(0, name="c")
c = tf.constant(2, name="c")
with tf.device("/device:GPU:0"):
  with tf.name_scope("sum"):
    s1 = tf.constant(0, name="s1")
    with tf.name_scope("prod"):
      p1 = tf.constant(1, name="p1")
      p2 = tf.constant(2, name="p2")
      p12 = tf.multiply(p1, p2,
                        name="mult")
    sp = tf.add(s1, p12, name="add")

stack = tf.stack([sp, c],
                 name="stack")
```



Conversion, Overloading, Broadcasting

```
a = tf.constant([1, 1])
c = a + 2
```

Conversion
`2 -> tf.constant(2)`

Broadcasting
`2 -> [2, 2]`

Overloading
`a + 2 -> tf.add(a, 2)`

```
a = tf.constant([1, 1])
c = tf.add(a, tf.constant([2, 2]))
```

The Graph – Inspection

(helper function practical session)

graph.**get_operations**()

```
[<tf.Operation 'Const' type=Const>,
 <tf.Operation 'add/y' type=Const>,
 <tf.Operation 'add' type=Add>]
```

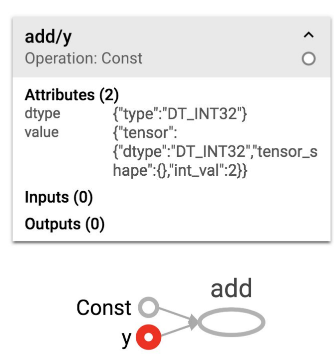
graph.**get_operation_by_name**('Const')

```
<tf.Operation 'Const' type=Const>
```

graph.**get_tensor_by_name**('Const:**0**')

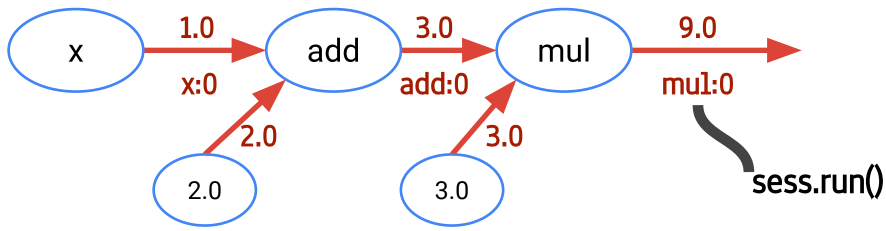
```
<tf.Tensor 'Const:0' shape=(2,) dtype=int32>
```

show_graph(graph)



The Session - Makes the Tensors Flow

The Session **executes the Graph**, similar to the JVM running Java bytecode.



build graph...

```
x = tf.constant(1.0, name="x")
add = tf.add(x, 2.0, name="add")
mul = tf.multiply(add, 3.0, name="mul")
```

...run graph

```
with tf.Session() as sess:
  mul_, = sess.run([mul])
  print(mul_)
```

(Default) Graph, (Interactive) Session

```
tf.reset_default_graph()
x = tf.constant(3.1415)
sess = tf.InteractiveSession()
print(x.eval())
```

Start with a clean slate.

x.eval() is shortcut for sess.run([x])

```
graph = tf.Graph()
with graph.as_default():
  x = tf.constant(2.718)
with tf.Session(graph=graph) as sess:
  print(x.eval())
```

New Ops always attached to default graph.

Provide graph as argument.

```
with tf.Graph().as_default(),\
     tf.Session() as sess:
  x = tf.constant(1.414)
  print(sess.run([x]))
```

Slides often only show this code.

Placeholders, Feeding

```
x = tf.placeholder(tf.float32, shape=[2], name="x")
norm = tf.norm(x)

norm.eval(feed_dict={x: [3, 4]})
```

```
5.0
```

```
sess.run([x, norm], feed_dict={"x:0": [3, 4]})
```

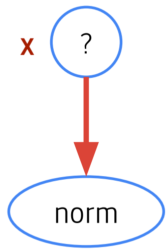
```
[array([3., 4.], dtype=float32), 5.0]
```

```
sess.run([x, norm])
```

```
InvalidArgumentError ... must feed a value ... tensor 'x'
```

```
sess.run([x, norm], feed_dict={x: [3, 4], norm: 0})
```

```
[array([3., 4.], dtype=float32), array(0., dtype=float32)]
```



Graph/Session Quiz

```
with tf.Graph().as_default() as graph:
  x = tf.constant(1)
  x2 = tf.multiply(x, x)

with tf.Session() as sess:
  print(sess.run(x2))
```

```
ValueError: Tensor ... is not an element of this graph.
```

```
with tf.Graph().as_default() as graph:
  x = tf.constant(1)
  x2 = tf.multiply(x, x)

with tf.Session(graph=graph) as sess:
  print(sess.run(x2))
```

```
with tf.Graph().as_default():
  x = tf.constant(1)
  x2 = tf.multiply(x, x)

  with tf.Session() as sess:
    print(sess.run(x2))
```

Providing Functions, Not Tensors

```
inputs = get_inputs_tensor()
labels = get_labels_tensor()

model.train(inputs, labels)
```

⇨ These tensors are already attached to a graph.

```
def input_fn():
  inputs = get_inputs_tensor()
  labels = get_labels_tensor()
  return inputs_labels

model.train(input_fn)
```

⇨ model.train() can decide which graph tensors are attached to.

The Shapes*

`1`	**scalar** rank=0, shape=()
`[1, 2]`	**vector** rank=1 shape=(2,)
`[[1, 2, 3],` ` [2, 4, 6]]`	**matrix** rank=2 shape=(2, 3)

`[[[1, 2],`
`  [3, 4]],`
` [[5, 6],`
`  [7, 8]]]`
	tensor
	rank=3
	shape=(2, 2, 2)

```
x = tf.random_normal(shape=(2, 2))
x.shape
```

```
TensorShape([Dimension(2), Dimension(2)])
```

```
x.shape.as_list()
```

```
[2, 2]
```

```
tf.reshape(x, (1, -1)).shape
```

```
TensorShape([Dimension(1), Dimension(4)])
```

* the cause of, and solution to, all TensorFlow crashes!

Partially Known Shapes

```
x = tf.placeholder(tf.float32, [None, 4])
x.shape
```

```
TensorShape([Dimension(None), Dimension(4)])
```

```
x2 = tf.reshape(x, (2, -1))
x2.eval({x: [range(4)]})
```

```
array([[0., 1.], [2., 3.]], dtype=float32)
```

```
batch_size = tf.shape(x)[0]
x2b = tf.reshape(x, [batch_size, 2, -1])
x2b.eval({x: [range(4), range(0, 8, 2)]})
```

```
array([[[0., 1.], [2., 3.]],
       [[0., 2.], [4., 6.]]], dtype=float32)
```

⇨ Dimension will only be fully defined at runtime. (Often used for batch dimension.)

⇨ The shape information can be accessed **as a Tensor**.

Sparse Tensors

```
sparse = tf.SparseTensor(indices=[[0, 0], [1, 2]], values=[1, 2], dense_shape=[3, 4])
sparse.eval()
```

```
SparseTensorValue(indices=array([[0, 0],
       [1, 2]]), values=array([1, 2], dtype=int32),
dense_shape=array([3, 4]))
```

```
tf.sparse_tensor_to_dense(sparse).eval()
```

```
array([[1, 0, 0, 0],
       [0, 0, 2, 0],
       [0, 0, 0, 0]], dtype=int32)
```

[0,0] [1,2]

⇨ Efficient encoding of "mostly zero" tensors (one-hot, linear models)

⇨ Variable-length features

⇨ Most Ops only for dense tensors

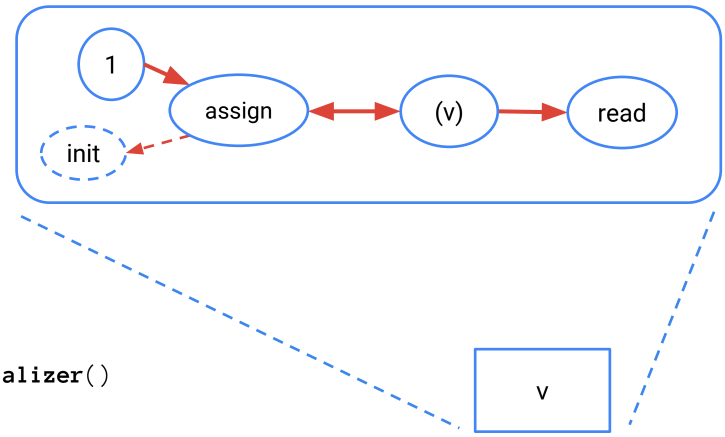
Variables

```
v = tf.Variable(1, name="v")
print(v.eval())
```

```
FailedPreconditionError ...
```

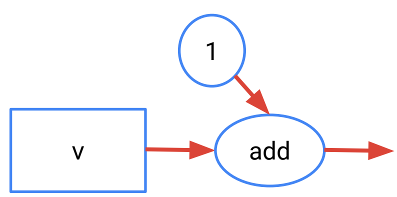
```
init_op = tf.global_variables_initializer()
sess.run(init_op)
print(v.eval())
```

```
1
```

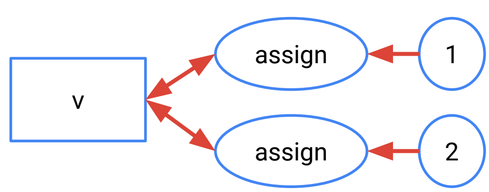


"v++"

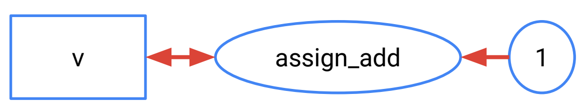
```
v = tf.Variable(0, name="v")
tf.global_variables_initializer().run()
v = v + 1
```



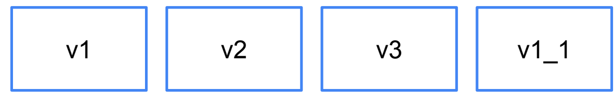

```
v = tf.Variable(0, name="v")
tf.global_variables_initializer().run()
sess.run([tf.assign(v, v.eval() + 1)])
sess.run([tf.assign(v, v.eval() + 1)])
```




```
v = tf.Variable(0, name="v")
tf.global_variables_initializer().run()
inc_v = tf.assign_add(v, 1)
sess.run([inc_v])
sess.run([inc_v])
```



Naming, Sharing

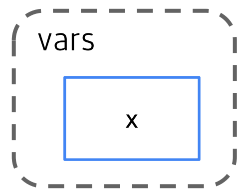



```
v1 = tf.Variable(0, name="v1")
v2 = tf.get_variable("v2", initializer=0)
v3 = tf.get_variable("v3", shape=(3, 3), initializer=tf.random_normal_initializer())
v1 = tf.Variable(0, name="v1")
v2 = tf.get_variable("v2", initializer=0)
```

```
ValueError: Variable v2 already exists
```

```
with tf.variable_scope("vars"):
  x = tf.get_variable("x", initializer=0.)

with tf.variable_scope("vars", reuse=True):
  x = tf.get_variable("x", initializer=0.)
```



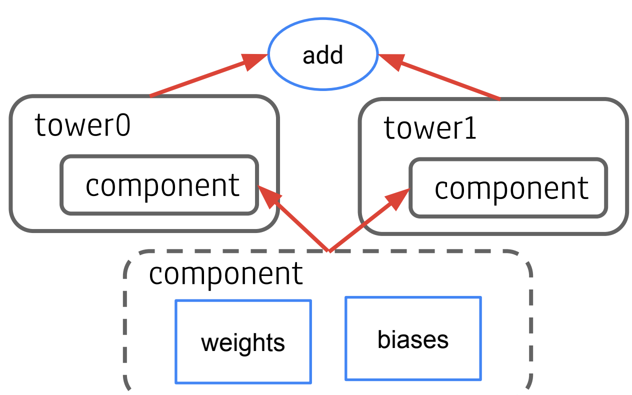
Sharing Example

```
def component(x, n, initializer=tf.random_normal_initializer):
  with tf.variable_scope("component", reuse=tf.AUTO_REUSE):
    weights = tf.get_variable("weights", shape=(n, n), initializer=initializer)
    biases = tf.get_variable("biases", shape=(n,), initializer=initializer)
    return tf.matmul(weights, x) + biases

with tf.name_scope("tower0"):
    x = tf.random_normal([5, 1])
    y0 = component(x, n=5)

with tf.name_scope("tower1"):
    x = tf.random_normal([5, 1])
    y1 = component(x, n=5)

y = y0 + y1
```



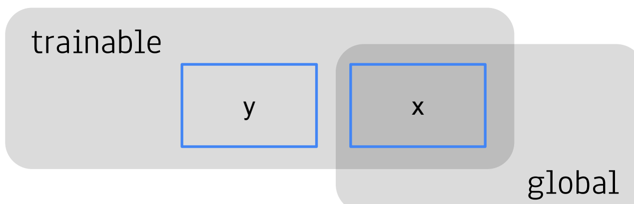
Collections

```
x = tf.get_variable("x", (), tf.float32)
y = tf.get_variable("y", (), tf.float32, collections=())
z = tf.get_variable("z", (), tf.float32, collections=(), trainable=False)
graph.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES)
```

```
[<tf.Variable 'x:0' shape=() dtype=float32_ref>,
 <tf.Variable 'y:0' shape=() dtype=float32_ref>]
```

```
graph.get_collection(tf.GraphKeys.GLOBAL_VARIABLES)
```

```
[<tf.Variable 'x:0' shape=() dtype=float32_ref>]
```



Save / Restore State

```
with tf.Graph().as_default() as graph, tf.Session() as sess:
  v = tf.get_variable("v", initializer=0.)
  ...
  saver = tf.train.Saver()
  save_path = saver.save(sess, "/tmp/model.ckpt")

with tf.Session(graph=graph) as sess:
  saver.restore(sess, "/tmp/model.ckpt")
  print(v.eval())
```

Checkpoint

```
/tmp/model.ckpt.data-*
/tmp/model.ckpt.index
/tmp/model.ckpt.meta
```

```
from tensorflow.python.tools import inspect_checkpoint as chkp
chkp.print_tensors_in_checkpoint_file(
    "/tmp/model.ckpt", tensor_name='', all_tensors=True, all_tensor_names=True)
```

```
tensor_name:  v
1.0
```

Agenda

Today	
Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ?

Imports, ...

```
import tensorflow as tf
print(tf.__version__)
from tensorflow.python.client import device_lib
device_lib.list_local_devices()
```

```
1.7.0
[name: "/device:CPU:0"
 device_type: "CPU"
 ...
 name: "/device:GPU:0"
 device_type: "GPU"
 ...
 physical_device_desc: "device: 0, name: Tesla K80, ..."]
 ...
```

..., Data, ...

```
from tensorflow.examples.tutorials.mnist import input_data
mnist = input_data.read_data_sets("MNIST_data/", one_hot=True)

mnist.train.next_batch(2)
```

```
...
Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
...
(array([[0., 0., 0., ..., 0., 0., 0.],
        [0., 0., 0., ..., 0., 0., 0.]], dtype=float32),
 array([[0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
        [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.]]))
```

..., Defining the Model, ...

```
graph = tf.Graph()

with graph.as_default():
  x = tf.placeholder(tf.float32, [None, 784])
  y_ = tf.placeholder(tf.float32, [None, 10])

  W = tf.Variable(tf.zeros([784, 10]))
  b = tf.Variable(tf.zeros([10]))

  logits = tf.matmul(x, W) + b
  y = tf.nn.softmax(logits)

  cross_entropy = tf.reduce_mean(-tf.reduce_sum(y_ * tf.log(y), axis=1))
  optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
  train_step = optimizer.minimize(cross_entropy)
```

..., Train + Evaluate

..., defining the model, ...

```
x = tf.placeholder(...)
y_ = tf.placeholder(...)
...
y = tf.nn.softmax(logits)
```

```
with tf.Session(graph=graph) as sess:
  sess.run(tf.global_variables_initializer())

  # train
  for _ in range(1100):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})

  # evaluate
  correct_prediction = tf.equal(tf.argmax(y,1), tf.argmax(y_,1))
  accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
  print(sess.run(accuracy, feed_dict={x: mnist.test.images, y_: mnist.test.labels}))
```

```
0.9183
```

Agenda

Today	
Intro	**Low-Level TF**
ML research	
ML to App @ZRH	Putting It Together
TF Goodness	TF vs. PyTorch ? ⇦

TensorFlow vs. PyTorch ?

	TensorFlow	PyTorch
Adoption	Huge	Rapidly growing
Graph	static, separate	dynamic, pythonic
Visualization	TensorBoard	Matplotlib etc
Deployment	TensorFlow Serving TensorFlow Lite, tensorflow.js	Caffe2
Parallelization	Distributed training device placement, variable scope	torch.nn.DataParallel
Abstractions	tf.keras, tf.layers, tf.estimator, tf.slim, …	torch.nn.Module
	BIG	**NEW**

tf.keras

```
from keras.models import Sequential
from keras.layers import Dense

model = Sequential([
    Dense(10, activation='softmax', input_dim=784)])

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

model.fit(mnist.train.images, mnist.train.labels, epochs=2)
model.evaluate(mnist.test.images, mnist.test.labels)
```

TensorFlow Eager

- No Session - No Graph.
- All tensors are executed "eager"ly.
- tf.GradientTape records gradients for tf.eager.Variable

Develop?

✓ Experimentation.

✓ Debugging, introspection.

✓ Natural control-flow.

✓ Very similar to PyTorch.

Deploy?

✘ Not mature yet.

✘ No graph-based optimization.

✘ No graph-based distribution / replication.

TensorFlow Eager - Putting it Together

```
W = tf.contrib.eager.Variable(tf.zeros([784, 10]))
b = tf.contrib.eager.Variable(tf.zeros([10]))

def get_preds(x):
  return tf.nn.softmax(tf.matmul(x, W) + b)

for _ in range(1100):
  batch_xs, batch_ys = mnist.train.next_batch(100)
  with tf.contrib.eager.GradientTape() as tape:
    preds = get_preds(batch_xs)
    loss = tf.reduce_mean(-tf.reduce_sum(batch_ys * tf.log(preds), axis=1))
  dW, db = tape.gradient(loss, [W, b])
  W.assign_sub(dW * 0.5)
  b.assign_sub(db * 0.5)

accuracy = lambda x,y: (x.argmax(axis=1) == y.argmax(axis=1)).mean()
accuracy(get_preds(mnist.test.images).numpy(),  mnist.test.labels)
```

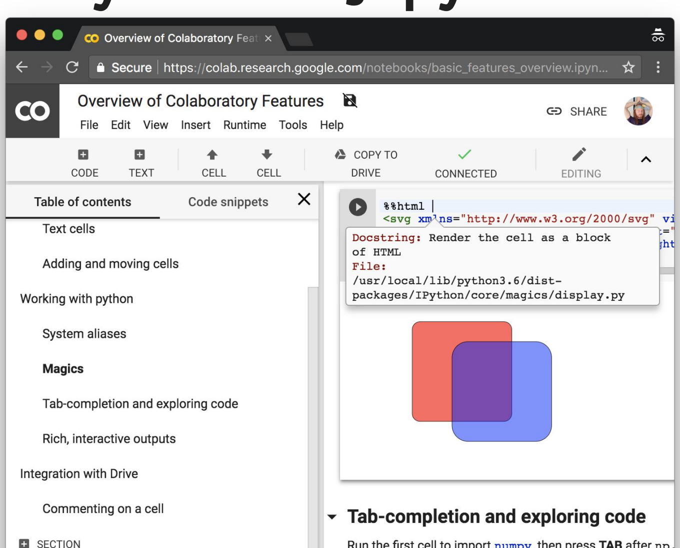
Define model

Train

Evaluate

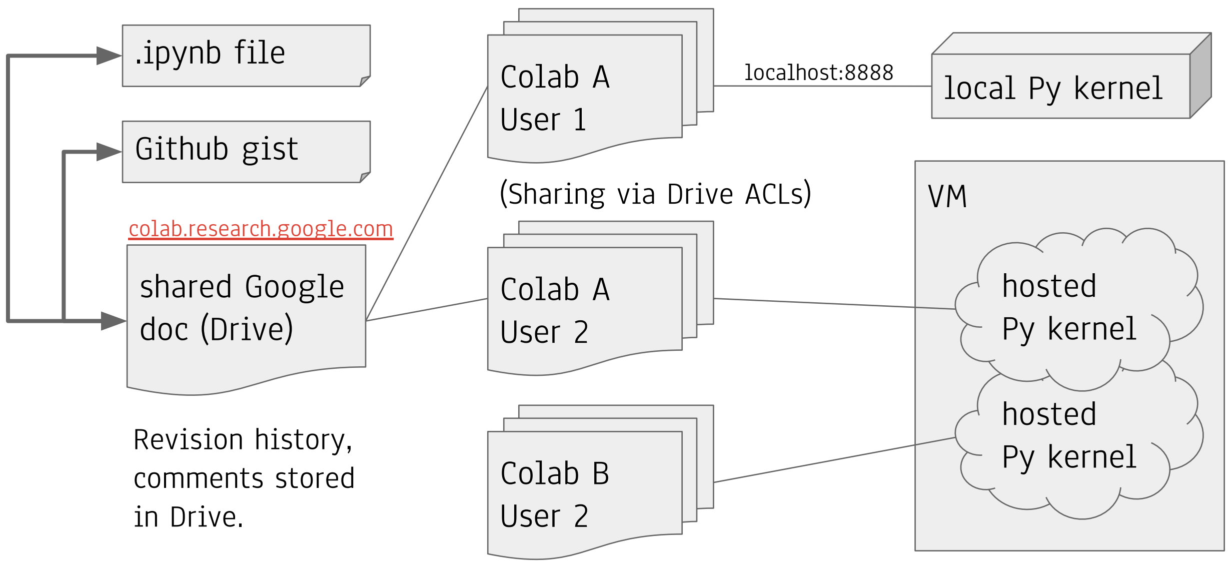
Practical Session, Colab

IPython ... Jupyter ... Colab



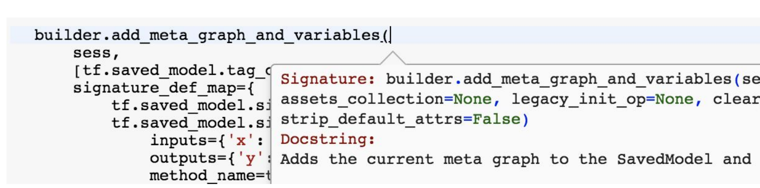
- TOC, folding
- Popover help
- Forms
- Revision history
- Document comments
- Snippets

Colab : About Kernels



Finding Documentation

1. Colab auto-completion



2. https://tensorflow.org
   a. Programmer's guide
   b. Tutorials
   c. API docs
3. Internet search : Stack Overflow, ...
4. Github codesearch : Search for usage patterns.

Thanks!

https://goo.gl/aSLGAA

References 1/3

(Agüera y Arcas 2017) Now Playing: Continuous low-power music recognition
https://arxiv.org/abs/1711.10958

(Bello 2017) Neural Optimizer Search with Reinforcement Learning
https://arxiv.org/abs/1709.07417

(Cer 2018) Universal Sentence Encoder
https://arxiv.org/abs/1803.11175

(Chiu 2016) Speech recognition for medical conversations
https://arxiv.org/abs/1711.07274

(Chan 2015) Listen, Attend and Spell
https://arxiv.org/abs/1508.01211

(Gulshan 2016) Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs
https://jamanetwork.com/journals/jama/fullarticle/2588763

References 2/3

(Jouppi 2017) In-Datacenter Performance Analysis of a Tensor Processing Unit
https://arxiv.org/abs/1704.04760

(Liu 2017) Detecting Cancer Metastases on Gigapixel Pathology Images
https://arxiv.org/abs/1703.02442

(Po-Hsuan 2018) An Augmented Reality Microscope for Real-time Automated Detection of Cancer
Under Review

(Poplin 2016) Creating a universal SNP and small indel variant caller with deep neural networks
https://www.biorxiv.org/content/early/2016/12/21/092890

(Rajkomar 2018) Scalable and accurate deep learning for electronic health records
https://arxiv.org/abs/1801.07860

(Ramachandran 2017) Searching for Activation Functions
https://arxiv.org/abs/1710.05941

References 3/3

(Shen 2017) Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions
https://arxiv.org/abs/1712.05884

(Van den Oord 2016) WaveNet: A Generative Model for Raw Audio
https://arxiv.org/abs/1609.03499

(Zoph 2016) Neural Architecture Search with Reinforcement Learning
https://arxiv.org/abs/1611.01578

(Zoph 2017) Learning Transferable Architectures for Scalable Image Recognition
https://arxiv.org/abs/1707.07012

Links 1/2

Introduction to TensorFlow Lite
https://www.tensorflow.org/mobile/tflite/

Machine Learning in JavaScript (TensorFlow Dev Summit 2018)
https://youtu.be/YB-kfeNIPCE

ImageNet is the new MNIST
https://supercomputersfordl2017.github.io/Presentations/ImageNetNewMNIST.pdf

Distributed TensorFlow
https://www.tensorflow.org/deploy/distributed

Distributed TensorFlow (TensorFlow Dev Summit 2017)
https://youtu.be/la_M6bCV91M

Graphs and Sessions
https://www.tensorflow.org/programmers_guide/graphs

Links 2/2

Sparse Tensors
https://www.tensorflow.org/versions/master/api_guides/python/sparse_ops

TensorFlow Linear Model Tutorial
https://www.tensorflow.org/versions/master/tutorials/wide

Variables
https://www.tensorflow.org/programmers_guide/variables

Variable Ops
https://www.tensorflow.org/versions/master/api_guides/python/state_ops#Variables

Sharing Variables
https://www.tensorflow.org/versions/master/programmers_guide/variable_scope

Eager Execution
https://www.tensorflow.org/programmers_guide/eager