インプリ合宿（2017年2月）

TensorFlow演習

玉川大学
横山裕樹

TensorFlowの概要

- Googleが作った数値演算ライブラリ．
- つい最近（2/11）バージョン1.0がリリースされた．やっと安定？
- 遅延評価による直感的な演算定義，記号的微分の実現．
- 用意されている演算の粒度の小さいので，深層学習以外にも幅広く応用できる．

```
name: "LeNet"
layer {
 name: "mnist"
 type: "Data"
 top: "data"
 top: "label"
 include {
  phase: TRAIN
 }
 transform_param {
  scale: 0.00390625
 }
 data_param {
  source: "mnist_train_lmdb"
  batch_size: 64
  backend: LMDB
 }
}
layer {
 name: "conv1"
 type: "Convolution"
 bottom: "data"
 top: "conv1"
 param {
  lr_mult: 1
 }
 param {
  lr_mult: 2
 }
 convolution_param {
  num_output: 5
  kernel_size: 2
  stride: 1
  weight_filler {
   type: "xavier"
  }
  bias_filler {
   type: "constant"
  }
 }
}
layer {
 name: "pool1"
 type: "Pooling"
 bottom: "conv1"
 top: "pool1"
 pooling_param {
  pool: MAX
  kernel_size: 2
  stride: 2
 }
}
```

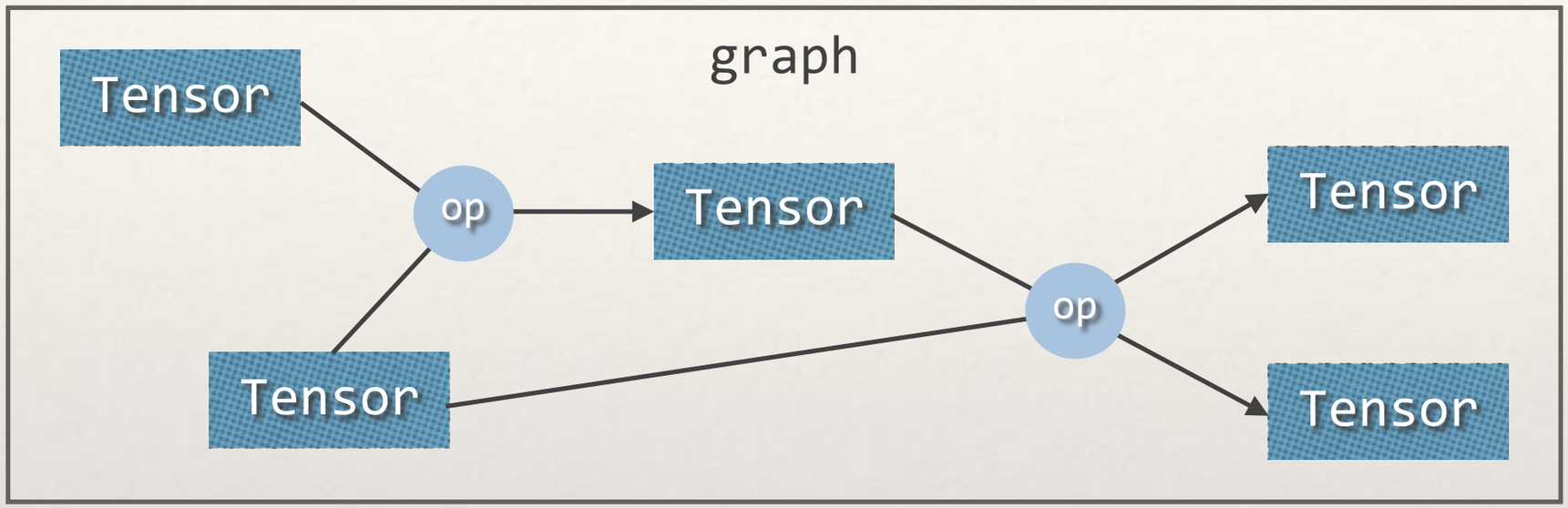
Caffeのprototxt

	Numpy	TensorFlow
Array	`np.ndarray` `np.zeros`	`tf.Tensor` `tf.zeros`
Types	`np.int64` `np.float32`	`tf.int64` `tf.float32`

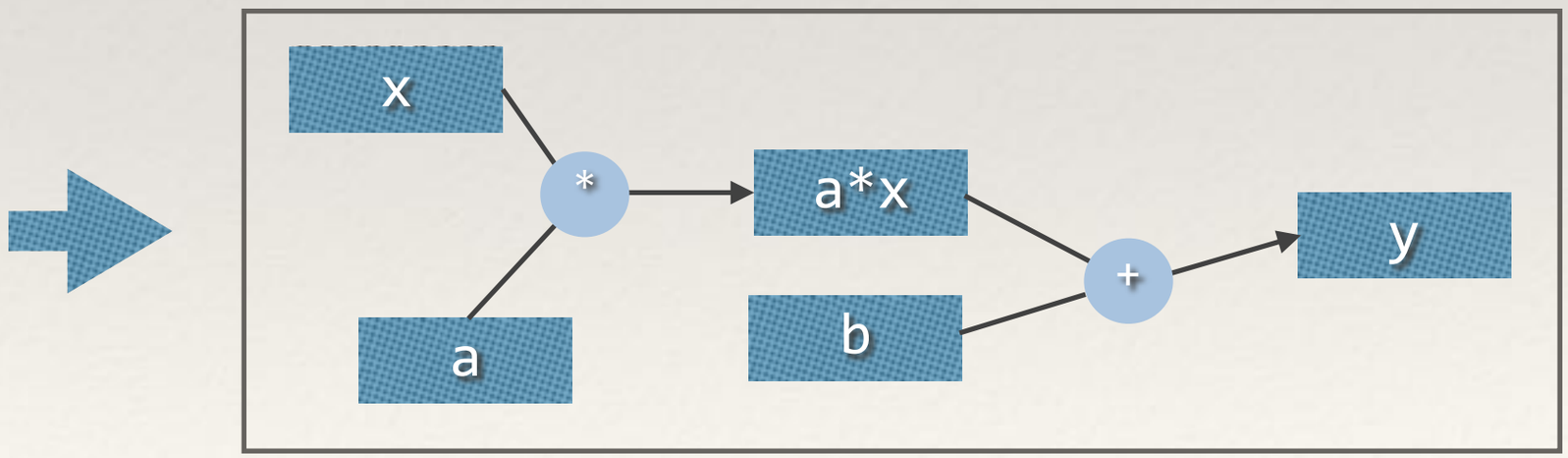
Operations	`np.sum`
	`np.reshape`
	`+ - * / > <`
		`tf.reduce_sum`
		`tf.reshape`
		`+ - * / > <`

Slicing	`x[:, begin:end]`	`x[:, begin:end]`

グラフと（そのノードとしての）テンソル




```
x = tf.placeholder(tf.float32)
a = tf.Variable(0)
b = tf.Variable(0)
y = a * x + b
```



テンソルいろいろ

- 定数

```
>>> tf.constant([[1,2],[3,4]])
<tf.Tensor 'Const_5:0' shape=(2, 2) dtype=int32>
```

- 式

```
>>> x = tf.constant(3.14)
>>> y = tf.constant(1.41)
>>> x+y
<tf.Tensor 'add:0' shape=() dtype=int32>
```

- 零行列，単位行列，乱数行列，etc.

```
>>> tf.eye(5)
<tf.Tensor 'eye_1/MatrixDiag:0' shape=(5, 5)
 dtype=float32>
```

- プレースホルダ（関数の仮引数のようなもの，詳しくは後述）

```
>>> tf.placeholder(tf.float32)
<tf.Tensor 'Placeholder:0' shape=<unknown> dtype=float32>
```

- 変数（その名の通り，変化させられるもの）

```
v = tf.Variable([0,1,2])
op1 = v.initializer
op2 = v.assign_add([1,2,3])
```

実行の主な流れ

グラフを作って

```
graph = tf.Graph()
with graph.as_default():
    x = tf.placeholder(tf.float32)
    t = tf.placeholder(tf.float32)
    w = tf.Variable(tf.truncated_normal(W_SHAPE))
    y = f(x, w)
    loss = tf.nn.l2_loss(y-t)
    opt = tf.train.GradientDescentOptimizer(LEARNING_RATE).minimize(loss)
    init = tf.global_variables_initializer()
```

セッションで走らせる

```
with tf.Session(graph=graph) as sess:
    init.run()
    for i in xrange(STEPS):
        _, loss_val = sess.run([opt, loss], feed_dict={x: INPUT, t: OUTPUT})
        print loss_val
```

out_list = sess.run(fetch_list, feed_dict)

fetch_list: 値を取得したいテンソル（もしくはその名前）のリスト

feed_dict: 値を与えたいテンソル（もしくはその名前）→その値（np.array）の辞書

out_list: fetch_listの各要素を評価した値（np.array）

変数の操作

- 勾配法

```
gd = GradientDescentOptimizer(learning_rate=0.1)
opt = gd.minimize()
```

- 勾配をいじる

```
gd = GradientDescentOptimizer(learning_rate=0.1)
grads_and_vars = gd.compute_gradients(loss, VAR_LIST)
grads_and_vars = [(DO_SOMETHING(grad), var) for grad, var in grads_and_vars]
opt = gd.apply_gradients(grads_and_vars)
```

- 直接操作する

・代入(=)

```
opt = var.assign(TENSOR)
```

・加算(+=)

```
opt = var.assign_add(TENSOR)
```

・減算(-=)

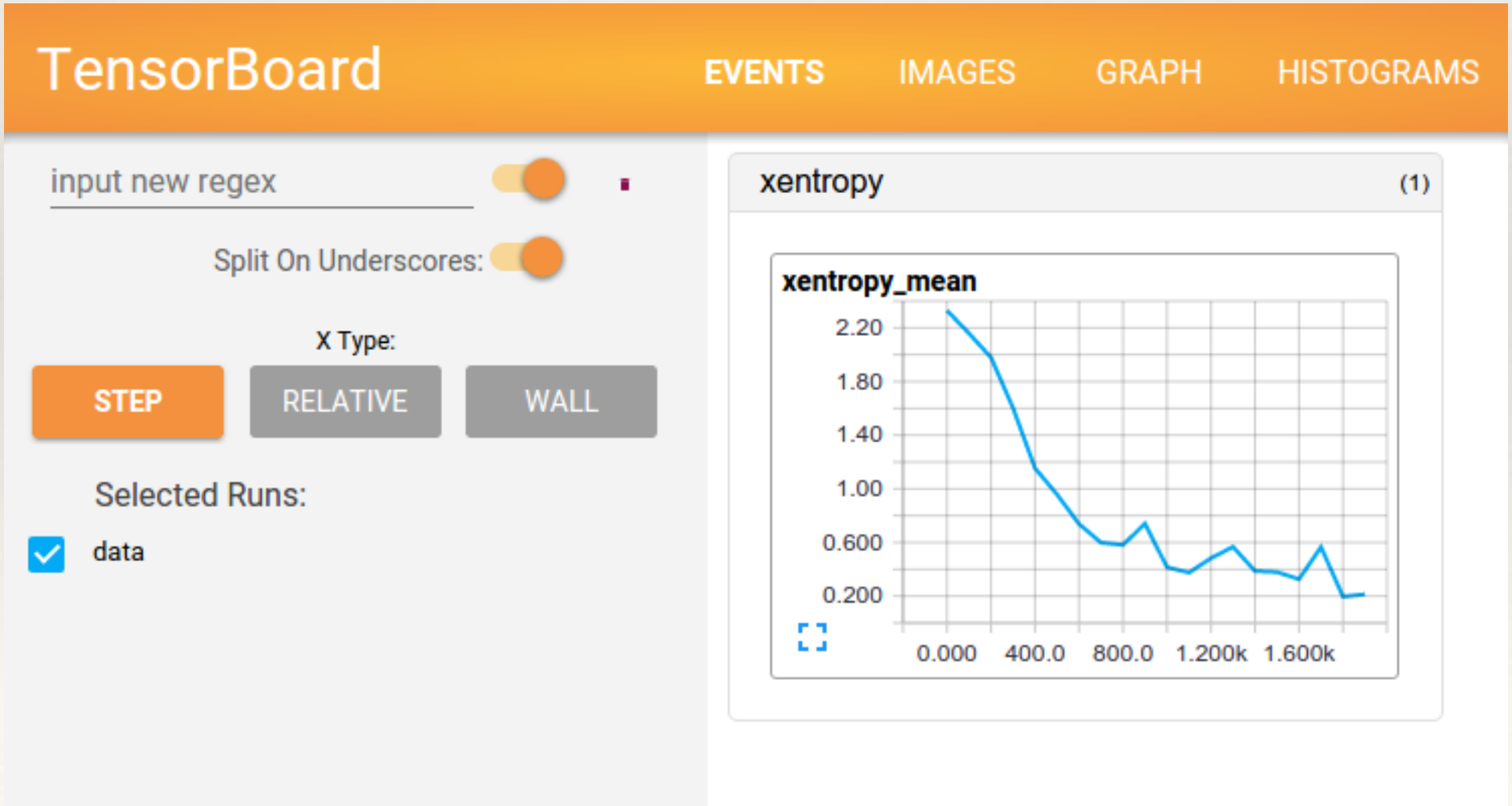
```
opt = var.assign_sub(TENSOR)
```

- 注：その場では実行されない．セッションで実行する．

```
with tf.Session() as sess:
    sess.run(opt)
```

Tensorboard

- 計算過程・結果の可視化ツール
- グラフの表示もできる
- HTTPサーバとして動作（外部から監視可能）



Tensorboardによる可視化の主な流れ

- 学習プログラム

グラフに
サマリーの定義を
追加

```
graph = tf.Graph()
with graph.as_default():
…
    loss = tf.nn.l2_loss(y-t)
    tf.summary.scalar(loss)
…
    merge = tf.summary.merge_all()

writer = tf.summary.FileWriter('/path/to/logfile', tf.get_default_graph())
```

セッションで
評価して
ファイルに書き込み

```
with tf.Session(graph=graph) as sess:
    init.run()
    for i in xrange(STEPS):
        _, loss_val, merge_val = sess.run([opt, loss, merge],
                                          feed_dict={x: INPUT, t: OUTPUT})
        writer.add_summary(merge_val, i)
```

- Tensorboard

```
$ tensorboard --logdir=/path/to/logfile
```

- ブラウザ

http://host:6006

tf.contrib.slim

- 公式にはあまり紹介されていない（まだ実験段階のため）がすごく便利なライブラリ．
- バージョン0.9から使用可能．Python上のフロントエンドなので，コードを持ってきていじれば0.8でも使える．
- 変数を作るなどの良くある処理をよしなにやってくれる．
- 一層につき一行ぐらいでニューラルネットのグラフを構築できる．

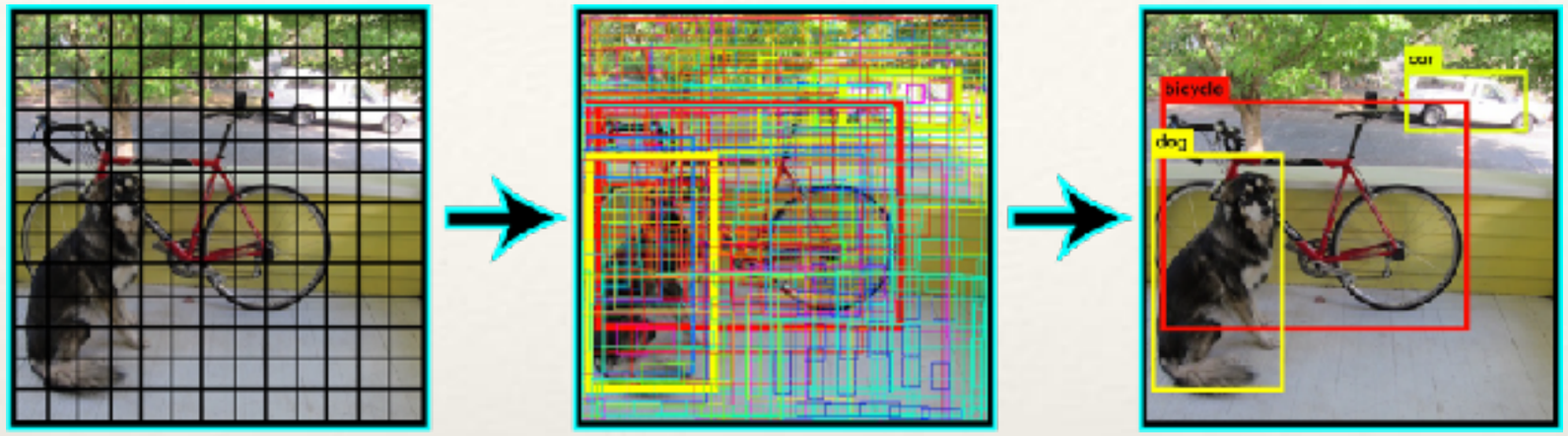
```
import tensorflow as tf
from tensorflow.contrib import slim

…

Outputs = slim.conv2d(inputs, 32, [3,3], 2)
```

```
…
weights = tf.Variable(…)
outputs = tf.nn.conv2d(inputs, weights, [1, 2, 2, 1])
…
outputs = tf.nn.batch_normalization(outputs, …)
…
```

YOLO v2
(https://pjreddie.com/darknet/yolo/)



- 物体の検出と認識を同時に行えるディープニューラルネット．
- CNNの最上位層の各ユニットがそれぞれ複数のBoundingBox（担当領域の相対座標）を認識（P(Object)とP(Class=i|Object)）する．
- 基本的にslim.conv2dとslim.max_pool2dで実装できるが，一箇所だけ枝分かれしてreorgと呼ばれる演算（tf.extract_image_patchesで実現可）がある．

1	2	3	4	5	6
7	8	9	10	11	12
13	14	15	16	17	18
19	20	21	22	23	24
25	26	27	28	29	30
31	32	33	34	35	36

reorg

[1,2, 7,8]	[3,4, 9,10]	[5,6, 11,12]
[13,14, 19,20]	[15,16, 21,22]	[17,18, 23,24]
[25,26, 31,32]	[27,28, 33,34]	[29,30, 35,36]