Deep Learning Lecture 2018

Exercise Session 1 - Meta-introduction to Tensorflow

Florian Schmidt

Data Analytics Lab

September 24, 2018

Organization

Exercise Sessions

- Identical exercises at 15:00 and 16:00
- Weakly exercises – mostly theoretical. Optional, not graded, but very much recommended
- Practical project starting later in the semester
- Sign up at piazza.com and enrol in ETH's Deep Learning lecture
- Questions: During the lecture, exercise and at piazza, **not via email**.

Today: Tensorflow

1. How to learn Tensorflow (TF)
2. What TF-paradigms exist?
3. Best practices

Deep Learning Lecture, the practical part

- In the project!
- Rarely in the exercises.
- Pseudo-code in the exam, if at all.

Frameworks?

- Tensorflow, (Py-)Torch, Keras (part of Tensorflow)
- Tensorflow is recommended and a bit 'supported'

Resources

- Google *tensorflow XXX tutorial*. Seriously.
- http://tensorflow.org/get_started/get_started
- http://tensorflow.org/tutorials/
- Use http://stackoverflow.com and http://github.com/tensorflow/tensorflow/issues before asking us

Some recommendations and best practices

http://docs.google.com/document/d/1Zus92s1f0kjh7O5AbVRlW0plx3FPyP06y_uPKhfBPxA/edit?usp=sharing

Tensorflow

Development

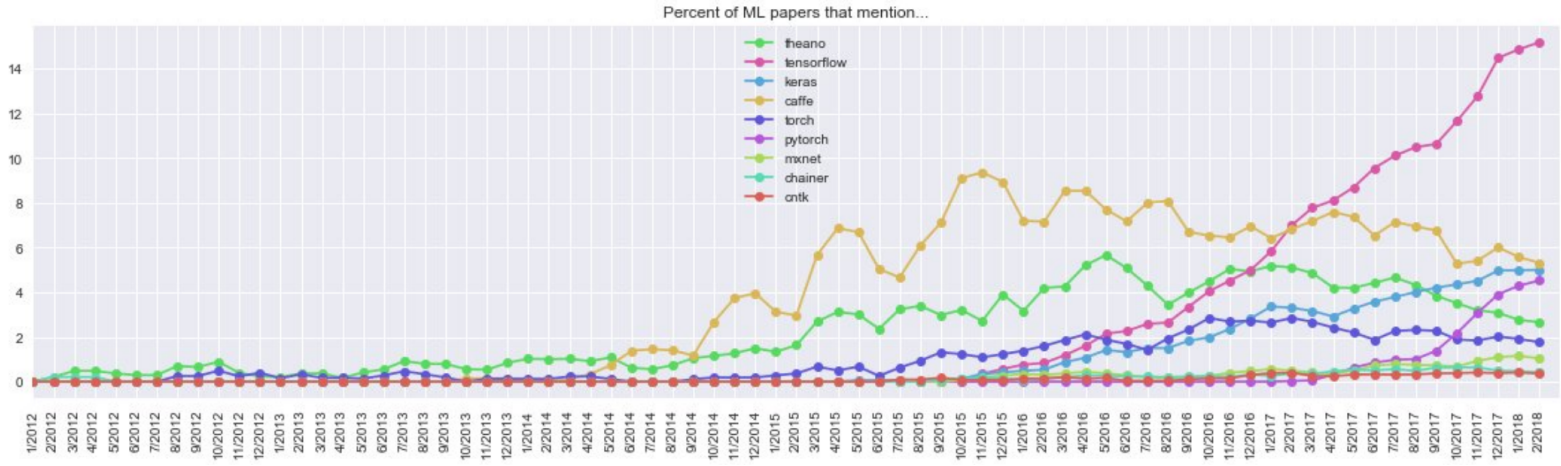
- Developed at Google
- Current version 1.11.0
- Focus on large scale learning and production

Architecture

- All relevant computations in C++
- User interaction in Python (numerical objects in numpy)
- Transparent CPU/GPU device layer
- Distributed mode available
- Usually graph-based compute model (*eager mode* as alternative)

Tensorflow relative to other frameworks

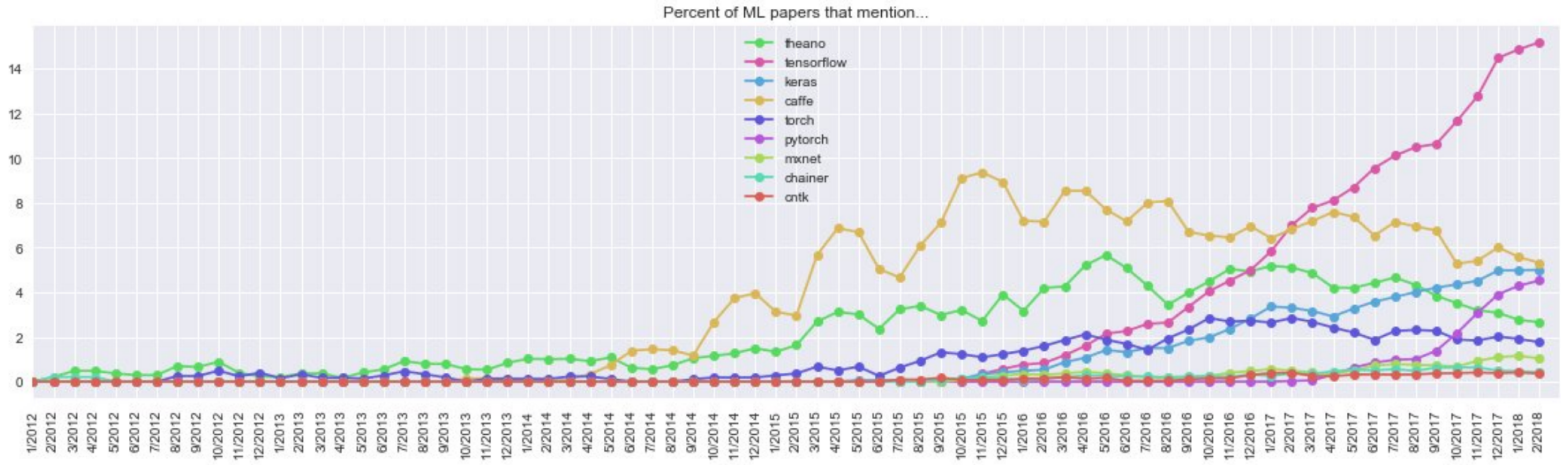
In academia



By Andrej Karpathy http://karpathy.github.io/

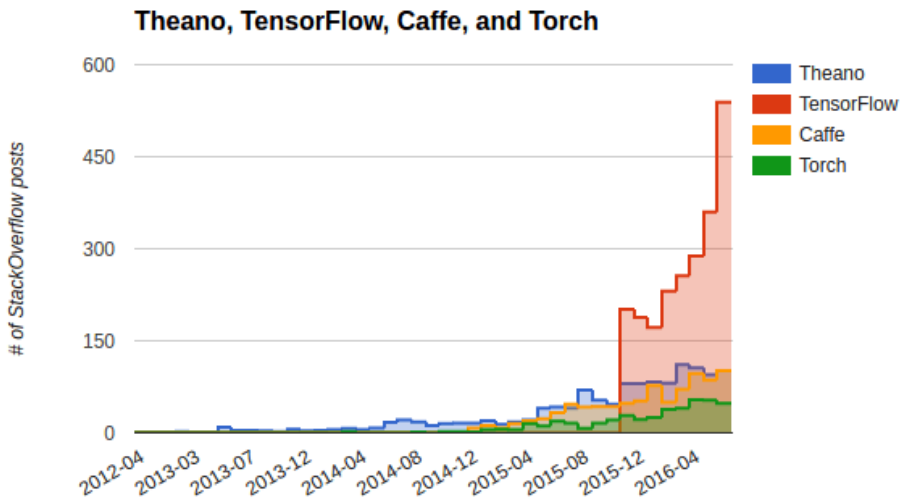
Tensorflow relative to other frameworks

In academia



By Andrej Karpathy http://karpathy.github.io/

On stackexchange

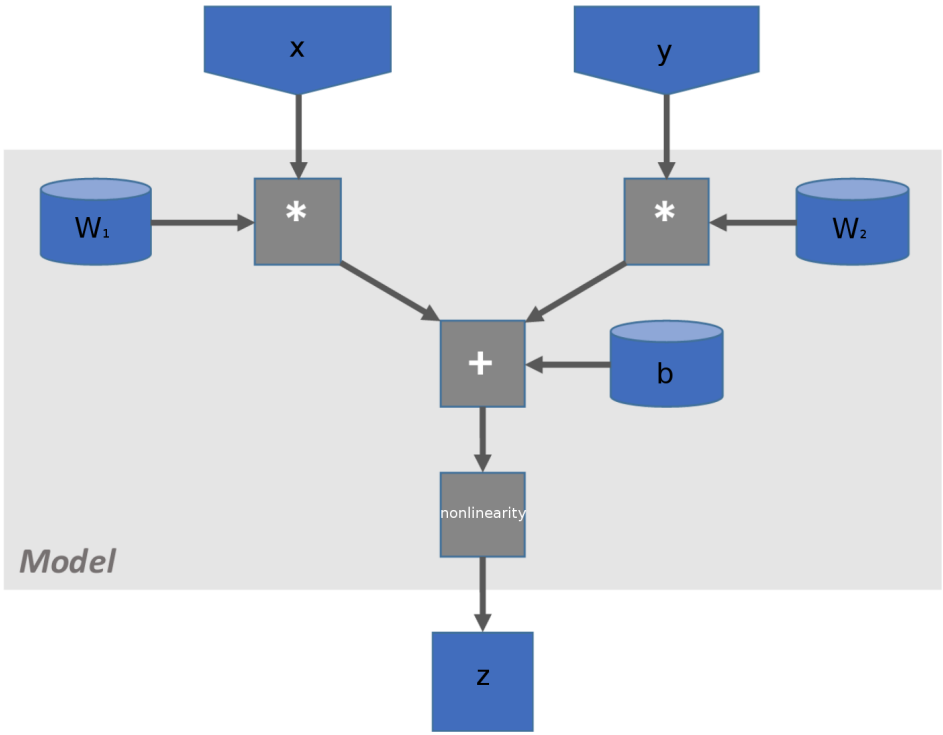


Computations as Graphs

$$z = \sigma(W_1 x + W_2 y + b)$$

Computations as Graphs

$$z = \sigma(W_1 x + W_2 y + b)$$



Tensorflow

- Every shape, gray and blue is a Tensor
- W_1, W_2, b are Variables

```
In [13]: import numpy as np

In [14]: W = np.ones([2,2])

In [15]: x = np.asarray([[0.1],[-0.2]])

In [16]: y = np.matmul(W,x)

In [17]: y = np.exp(y)

In [18]: y += 0.1

In [19]: y[0]
Out[19]: array([1.00483742])

In [20]:
```

```
In [19]: import tensorflow as tf

In [20]: W = tf.ones([2,2])

In [21]: x = tf.constant([[0.1],[-0.2]])

In [22]: y = tf.matmul(W,x)

In [23]: y = tf.exp(y)

In [24]: y += 0.1

In [25]: y[0]
Out[25]: <tf.Tensor 'strided_slice_4:0' shape=(1,) dtype=float32>

In [26]:
```

```
In [13]: import numpy as np

In [14]: W = np.ones([2,2])

In [15]: x = np.asarray([[0.1],[-0.2]])

In [16]: y = np.matmul(W,x)

In [17]: y = np.exp(y)

In [18]: y += 0.1

In [19]: y[0]
Out[19]: array([1.00483742])

In [20]:
```

```
In [19]: import tensorflow as tf

In [20]: W = tf.ones([2,2])

In [21]: x = tf.constant([[0.1],[-0.2]])

In [22]: y = tf.matmul(W,x)

In [23]: y = tf.exp(y)

In [24]: y += 0.1

In [25]: y[0]
Out[25]: <tf.Tensor 'strided_slice_4:0' shape=(1,) dtype=float32>

In [26]:
```

Numpy vs. Tensorflow

- Matrix operations
- Element-wise operations
- Broadcasting

Pumping data into the graph

Data incoming from

1. Constants (`tf.constant`, `tf.ones...`)
2. Variables `tf.get_variable` (bad but possible: `tf.Variable`)
3. Placeholders `tf.placeholder`
4. `tf.data` Framework

Data output by running a `tf.Session`

```
In [33]: with tf.Session() as session:
    ...:     session.run(y[0])
    ...:

In [34]: with tf.Session() as session:
    ...:     out = session.run(y[0])
    ...:     print(out)
    ...:
    ...:
[1.0048374]
```

Common mistakes

```
In [15]: W = tf.get_variable("W", [2,2], tf.float32)

In [16]: c = tf.constant([[0.1], [0.2]])

In [17]: result = tf.matmul(W,c)

In [18]: print(result[0])
Tensor("strided_slice:0", shape=(1,), dtype=float32)

In [19]:
```

Common mistakes

```
In [15]: W = tf.get_variable("W", [2,2], tf.float32)

In [16]: c = tf.constant([[0.1], [0.2]])

In [17]: result = tf.matmul(W,c)

In [18]: print(result[0])
Tensor("strided_slice:0", shape=(1,), dtype=float32)

In [19]:
```

```
In [19]: if (result[0] > 0.0):
    ...:     print("works")
    ...:
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-c8c751c2f661> in <module>()
----> 1 if (result[0] > 0.0):
      2     print("works")
      3

/home/schmiflo/.local/lib/python3.6/site-packages/tensorflow/python/framework/ops.py in __bool__(self)
    636       `TypeError`.
    637     """
--> 638     raise TypeError("Using a `tf.Tensor` as a Python `bool` is not allowed. "
```

Common mistakes

```
In [15]: W = tf.get_variable("W", [2,2], tf.float32)

In [16]: c = tf.constant([[0.1], [0.2]])

In [17]: result = tf.matmul(W,c)

In [18]: print(result[0])
Tensor("strided_slice:0", shape=(1,), dtype=float32)

In [19]:
```

```
In [19]: if (result[0] > 0.0):
    ...:     print("works")
    ...:
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-c8c751c2f661> in <module>()
----> 1 if (result[0] > 0.0):
      2     print("works")
      3

/home/schmiflo/.local/lib/python3.6/site-packages/tensorflow/python/framework/ops.py in __bool__(self)
    636       `TypeError`.
    637     """
--> 638     raise TypeError("Using a `tf.Tensor` as a Python `bool` is not allowed. "
```

```
In [21]: test_greater = tf.greater(result[0], 0.0)

In [22]:
```

General Advice and Choices to make

1. Use `tf.summarys` and tensorboard https://tensorflow.org/guide/summaries_and_tensorboard
2. **Avoid learning from old code**. Like from last year ;-)
3. Understand variable sharing https://tensorflow.org/guide/variables
4. Load data using `tf.data` https://tensorflow.org/guide/datasets
5. To Keras or not? Application dependent...What do **you** want to learn/put on your CV?
6. Yes, there is a debugger https://tensorflow.org/guide/debugger
7. Use pytorch instead?
8. Eager mode = no graph! Will still be buggy https:/tensorflow.org/guide/eager

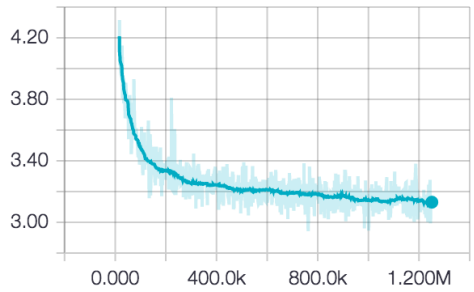
See http://docs.google.com/document/d/1Zus92s1f0kjh7O5AbVRlW0plx3FPyP06y_uPKhfBPxA/edit?usp=sharing

Tensorboard

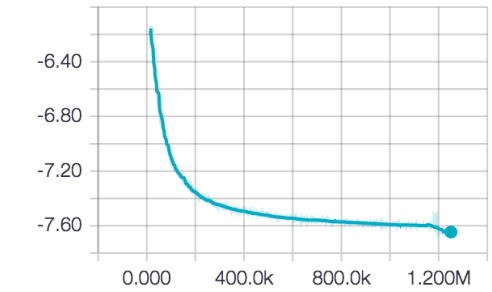
model

86

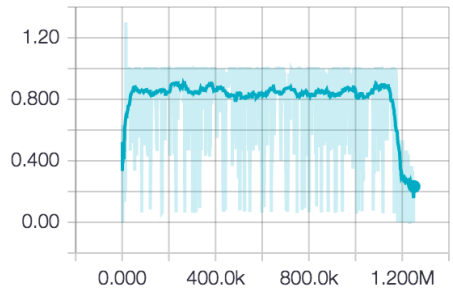
model/bits_per_dim



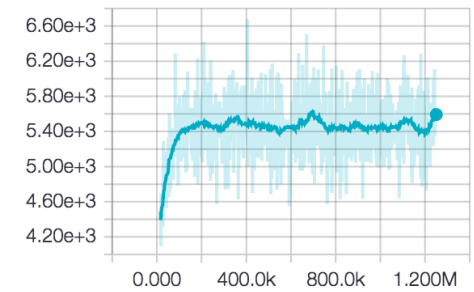
model/dec_log_stdv



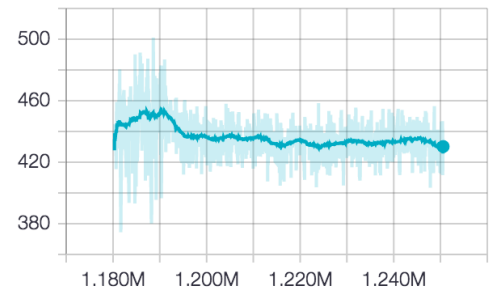
model/global_step/sec



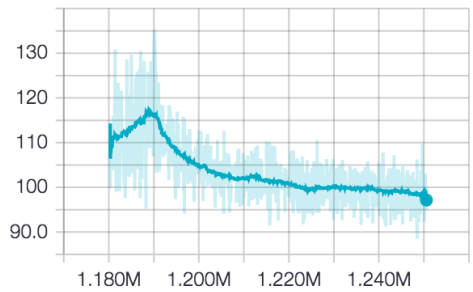
model/kl_cost



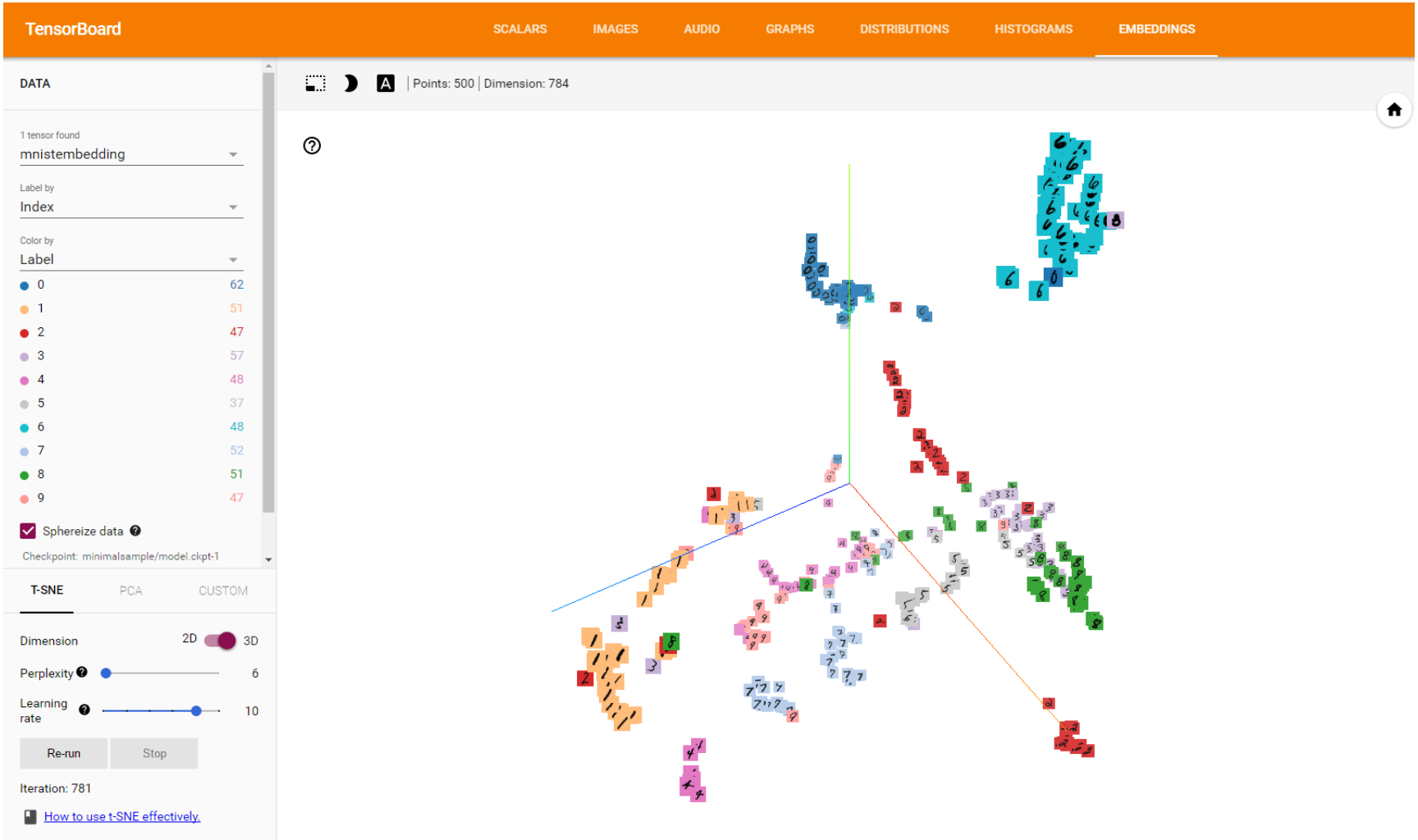
model/kl_cost_00_00



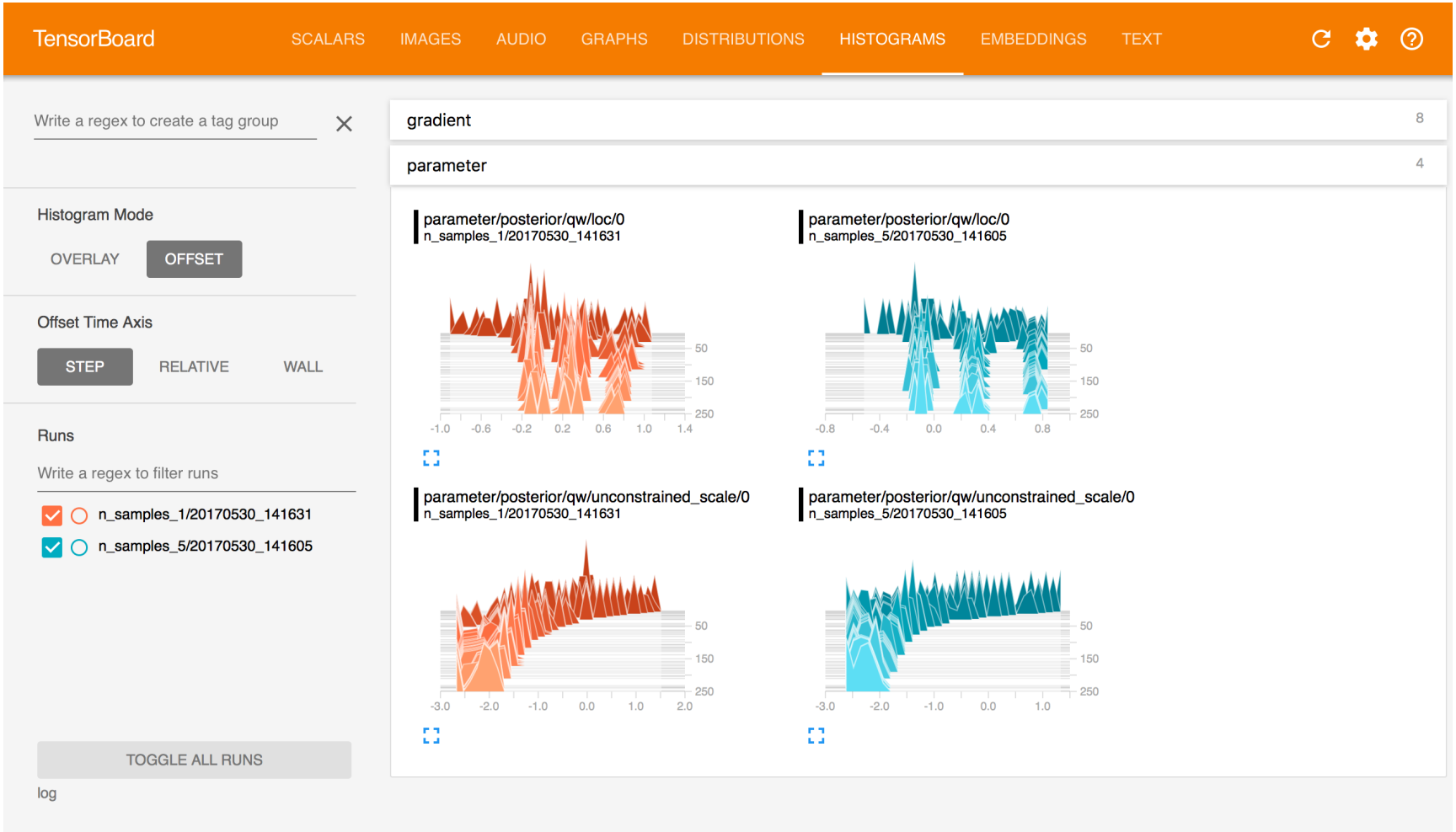
model/kl_cost_00_01



Tensorboard



Tensorboard



Example: Autoencoder

The Problem

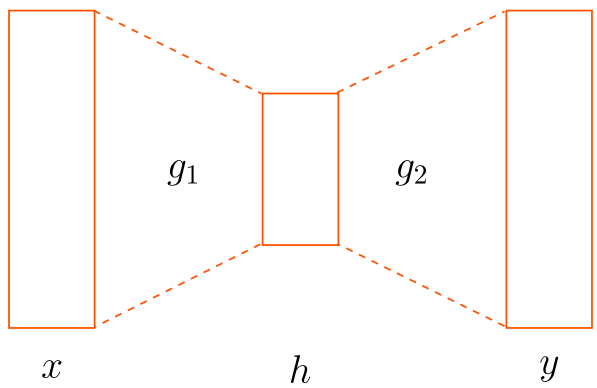
Given data	$\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
and $\tilde{d} < d$, find	$g_1 : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ and $g_2 : \mathbb{R}^{\tilde{d}} \to \mathbb{R}^d$
so that for	$h_i = g_1(x_i) \in R^{\tilde{d}}$
the reconstruction error	$\sum_i \|x_i - g_2(h_i)\|_2^2$ is small

Example: Autoencoder

The Problem

Given data	$\mathcal{X} = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$
and $\tilde{d} < d$, find	$g_1 : \mathbb{R}^d \to \mathbb{R}^{\tilde{d}}$ and $g_2 : \mathbb{R}^{\tilde{d}} \to \mathbb{R}^d$
so that for	$h_i = g_1(x_i) \in R^{\tilde{d}}$
the reconstruction error	$\sum_i \|x_i - g_2(h_i)\|_2^2$ is small

Neural network approach with single hidden layer



$h = g_1(x) = \sigma\left(\mathbf{W}_{\text{enc}} x + b_{\text{enc}}\right)$ with $\mathbf{W}_{\text{enc}} \in \mathbb{R}^{\tilde{d} \times d}$, $b_{\text{enc}} \in \mathbb{R}^{\tilde{d}}$

Example: Autoencoder in TF

```
d_data = 100
d_hidden = 30

# Construct Graph
x = tf.placeholder(tf.float32, [None, d_data], name="input")

# Hidden Layer Variables

W_enc = tf.get_variable("W_enc", [d_data, d_hidden], tf.float32)
b_enc = tf.get_variable("bias_enc", [1, d_hidden], tf.float32)

W_dec = tf.get_variable("W_dec", [d_hidden, d_data], tf.float32)
b_dec = tf.get_variable("bias_dec", [1, d_data], tf.float32)

# Hidden layer graph
h = tf.matmul(x, W_enc) + b_enc

# Output and reconstruction loss
y = tf.matmul(h, W_dec) + b_dec
loss = tf.sqrt(tf.reduce_sum(tf.square(x - y)))

# Optimizer
opt = tf.train.GradientDescentOptimizer(0.01)
update_step = opt.minimize(loss)
```

Now run the model

```
with tf.Session() as session:
  batch_size = 64
  datapoint = np.random.rand([batch_size,d_data]) # For now...
  feed_dict = {x : datapoint}
  hidden_rep, _ = session.run([h, update_step], feed_dict = feed_dict)
```

Remark: Don't do it like that...

Too verbose.... Instead use

- `tf.layers.dense`
- `tf.contrib.layers.fully_connected`

Will create variables internally $\rightarrow$ understand sharing before!

The graph in tensorboard