

Einführung in



Sören Wegener

Data-Science Meetup Kassel

24. Oktober 2017

TensorFlow, the TensorFlow logo and any related marks are trademarks of Google Inc.

Was ist TensorFlow?

- Machine Learning Library für Python
- Entstanden aus Googles *DistBelief*
- Im Einsatz bei z.B. Google Suche¹, Google Mail¹, Snapchat²

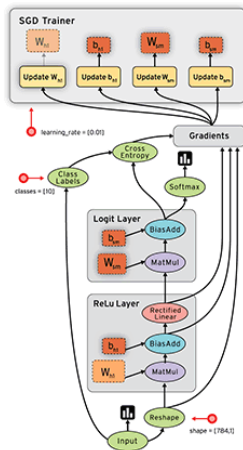
¹<https://tensorflow.org/about/uses>

²<https://www.tensorflow.org/>

Python?

Das ist doch viel zu langsam!

Computation Graph



Zwei APIs

TensorFlow Core

- Umsetzung von Modellen

Higher Level APIs

- Abstrakte Probleme lösen

Minimales Beispiel

```
1  import tensorflow as tf
2
3  a = tf.constant(3) # Tensor, rank 0
4  b = tf.constant(14, dtype='int32')
5  c = tf.multiply(a, b, name='awesome_multiplication')
6  # oder: c = a * b
7
8  sess = tf.Session()
9  result = sess.run(c)
```

TensorBoard

Logfiles schreiben:

```
tf.summary.FileWriter('./logdir/slide6', sess.graph)
```

In der Shell starten:

```
$ tensorboard --logdir=logdir
```

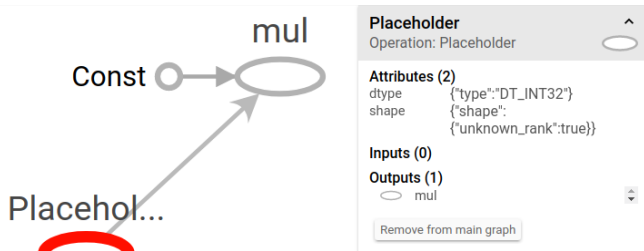
TensorBoard

The screenshot displays the TensorBoard web interface. The top navigation bar is orange and contains the text "TensorBoard", "GRAPHS", and "INACTIVE". On the left sidebar, there are several controls: "Fit to screen" (icon), "Download PNG" (icon), "Run (2)" with a dropdown menu showing "slide6", "Session runs (0)", "Upload" with a "Choose File" button, a "Trace inputs" toggle switch, and a "Color" section with radio buttons for "Structure" (selected), "Device", "XLA Cluster", "Compute time", "Memory", and "TPU Compatibility". Below these are two "colors" options: "same substructure" and "unique substructure". At the bottom of the sidebar is an "Expand legend" link. The central area shows a computational graph with two input nodes labeled "Const" and "Const_1" connected by arrows to an output node labeled "awesome...". The output node is highlighted with a red circle. On the right sidebar, the details for the "awesome_multiplication" operation are shown, including "Operation: Mul", "Attributes (1)" with a dictionary {"type": "DT_INT32"}, "Inputs (2)" listing "Const" and "Const_1" as scalar inputs, and "Outputs (0)". An "Add to main graph" button is at the bottom of this panel.

Placeholder

```
1 a = tf.placeholder(dtype='int32')
2 b = tf.constant(10)
3 c = a * b
4 sess = tf.Session()
5 result = sess.run(c, feed_dict={a: 30})
```

Placeholder

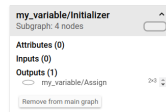
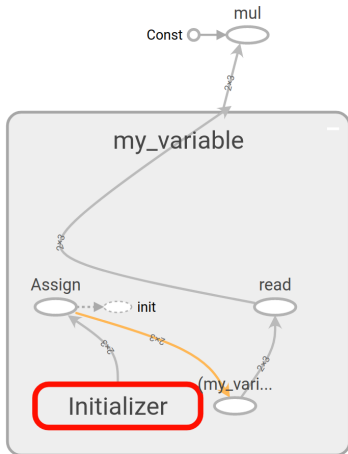


Variablen

```
1  v = tf.get_variable('my_variable', shape=[2, 3])
2  b = tf.constant(10, dtype='float32')
3
3  c = v * b
4
4  sess = tf.Session()
5  sess.run(tf.global_variables_initializer())
6  result = sess.run(c)
7
7  print(result)

[[-7.83156967  5.84567547 -5.6848073 ]
 [ 2.27666855 -8.07361794 -4.07426929]]
```

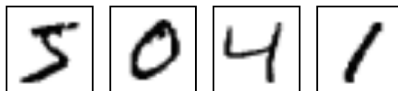
Variablen



TensorFlow am Beispiel von MNIST

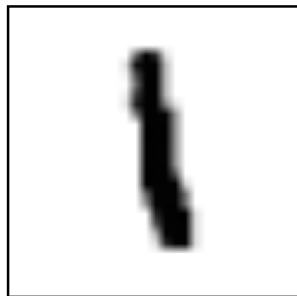
MNIST?

- Handgeschriebene Ziffern, 28x28 Pixel



Tutorial und Bildmaterial von https://www.tensorflow.org/get_started/mnist/beginners

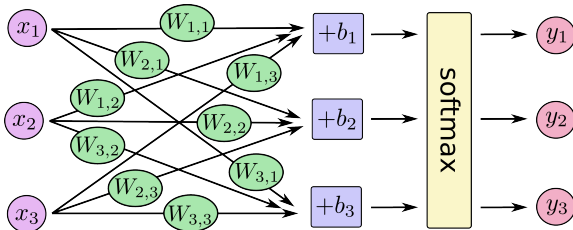
Darstellung als 784-elementiges Array



12

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	.6	.8	0	0	0	0	0	0
0	0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	0	.7	1	0	0	0	0	0	0
0	0	0	0	0	0	0	.5	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.4	0	0	0	0	0
0	0	0	0	0	0	0	0	1	.7	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	.9	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	.3	1	.1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Das Modell

 $x_1 \dots x_{784}, b_1 \dots b_{10}, y_1 \dots y_{10}$ 

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

Das Modell

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax} \left(\begin{bmatrix} W_{1,1} & W_{1,2} & W_{1,3} \\ W_{2,1} & W_{2,2} & W_{2,3} \\ W_{3,1} & W_{3,2} & W_{3,3} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} \right)$$

```
1 x = tf.placeholder(tf.float32, [None, 784], name='images')
2 y_ = tf.placeholder(tf.float32, [None, 10], name='labels')
3 W = tf.Variable(tf.zeros([784, 10]), name='weights')
4 b = tf.Variable(tf.zeros([10]), name='bias')
5 y = tf.matmul(x, W) + b
```

Der Fehler

- Kreuzentropie als Fehlermaß (*loss-Funktion*)

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \neq \begin{bmatrix} 0.2 \\ 0.02 \\ 0.01 \\ 0.01 \\ 0.01 \\ 0.03 \\ 0.01 \\ 0.01 \\ 0.2 \\ 0.5 \end{bmatrix}$$

Der Fehler

```
5  y = tf.matmul(x, W) + b
6
7  cross_entropy = tf.reduce_mean(
8      tf.nn.softmax_cross_entropy_with_logits(labels=y_,
9                                                  logits=y)
10 )
11 optimizer = tf.train.GradientDescentOptimizer(
12     0.5
13 ).minimize(cross_entropy)
```

Trainieren...

```
14 sess = tf.Session()
15 sess.run(tf.global_variables_initializer())
16 for i in range(1000):
17     batch_xs, batch_ys = mnist.train.next_batch(100)
18     sess.run(optimizer, feed_dict={
19         x: batch_xs,
20         y_: batch_ys
21     })
```

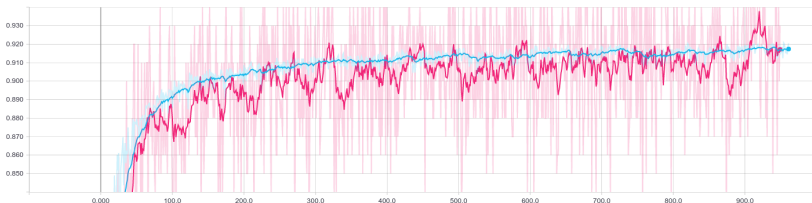
... und testen

```
22 correct_prediction = tf.equal(tf.argmax(y, 1),
23                               tf.argmax(y_, 1))
24 accuracy = tf.reduce_mean(tf.cast(correct_prediction,
25                                   tf.float32))

26 sess.run(accuracy, feed_dict={
27     x: mnist.test.images,
28     y_: mnist.test.labels
29 })
```

Trainingsverlauf

- 92% Accuracy
- > 99% mit anderen Modellen möglich



Boah ey, Mathematik

Immer Mathematik, überall Mathematik, ey!

Bilderkennung

```
1  import tensorflow as tf
2  import numpy as np
3  from pprint import pprint
4
5  image = tf.contrib.keras.preprocessing.image
6  vgg16 = tf.contrib.keras.applications.vgg16
7  model = vgg16.VGG16()
8
9  img = image.load_img('images/cat.jpg',
                        target_size=(224, 224))
10
11 x = image.img_to_array(img)
12 x = np.expand_dims(x, axis=0)
13 x = vgg16.preprocess_input(x)
14
15 predictions = model.predict(x)
16 pprint(vgg16.decode_predictions(predictions))
```


Bildererkennung



```
[[('n02123159', 'tiger_cat', 0.43690097),  
 ('n02124075', 'Egyptian_cat', 0.32366198),  
 ('n02123045', 'tabby', 0.1447085),  
 ('n02127052', 'lynx', 0.019589322),  
 ('n07930864', 'cup', 0.0077141393)]]
```

[https://en.wiktionary.org/wiki/cat#
/media/File:Cat03.jpg](https://en.wiktionary.org/wiki/cat#/media/File:Cat03.jpg)



Intelligent
Embedded Systems

https://www.ies.uni-kassel.de/Soft_Computing



UDACITY

<https://www.udacity.com/course/deep-learning--ud730>

Q&A

Folien und Codebeispiele in Jupyter:

<https://github.com/fino-digital/kassel-data-science-meetup/tree/master/2017-10-24/tensorflow>