The Splendors and Miseries of Tensorflow

Oleksandr Khryplyvenko

sept. 2016

Ph.D. Student at IMMSP NASU

m3oucat@gmail.com

Briefly about myself and how I met tf

2002	2003	2004	2008	2014	2016
code	work	linux	master CS	ML	Ph.D. student

I use ML for (order matters):

Research

Work

For whom and what it covers

Imagine that you spent 2 years on intensive ML(more research, less - production)
Here's brief of these 2 years related to frameworks

I'll will:

- compare TF with other frameworks
- tell about pros and cons of TF
- do some mathematics TF is based on (so are other frameworks too)
- tell about installation & usage nuances
- show how to debug(with a demo)

There are lots of related pages on the internet,
but I'm telling here only about the things I've used.

TF & other ML frameworks

	TF	Theano	Torch	Caffe	CNTK
prog language	python/C++	python/C++	lua/C	C++/python	specific language
the way $\partial f(x)/\partial x$ calculated	symbolic	symbolic	automatic *, ***	no	automatic
cluster	yes	no	yes	yes**	yes
quality of doc, samples	excellent	good	good	poor	poor
community help	guarranteed (up to 1 day)	not guaranteed	middle	Not used	Not used
core/API code complexity	easy	cryptic	good	hard	Not used
I used	≈ 1 yr	6 months	6 months	<1 month	< 1 month

https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software *http://dmlc.ml/2016/09/30/build-your-own-tensorflow-with-nnvm-and-torch.html

**https://software.intel.com/en-us/articles/caffe-training-on-multi-node-distributed-memory-systems-based-on-intel-xeon-processor-e5

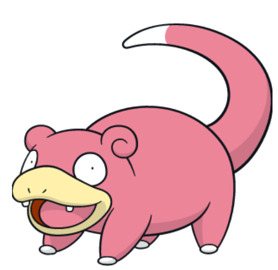
***https://github.com/twitter/torch-autograd

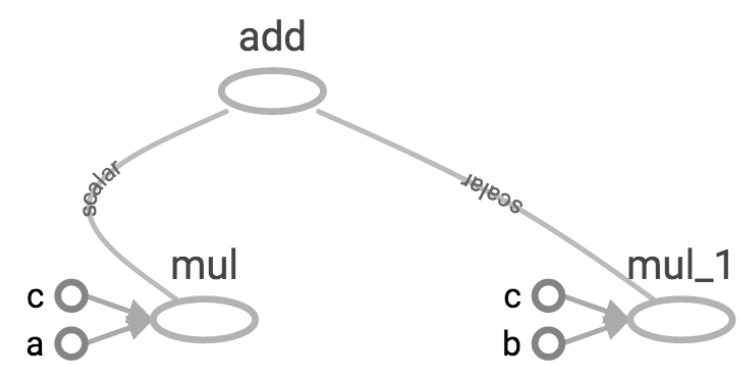
https://indico.io/blog/the-good-bad-ugly-of-tensorflow/

Baby-steps TF cons (immature)

- TF is not the fastest at the moment. But it's getting faster each release
- lots of reported & unreported issues. Be gentoo-way!
- syntax sugar-free. But it's getting better each release. (example - slices on vars)
- can't modify existing graph
- does not automatically simplify graph: ca + cb -> c(x+y)

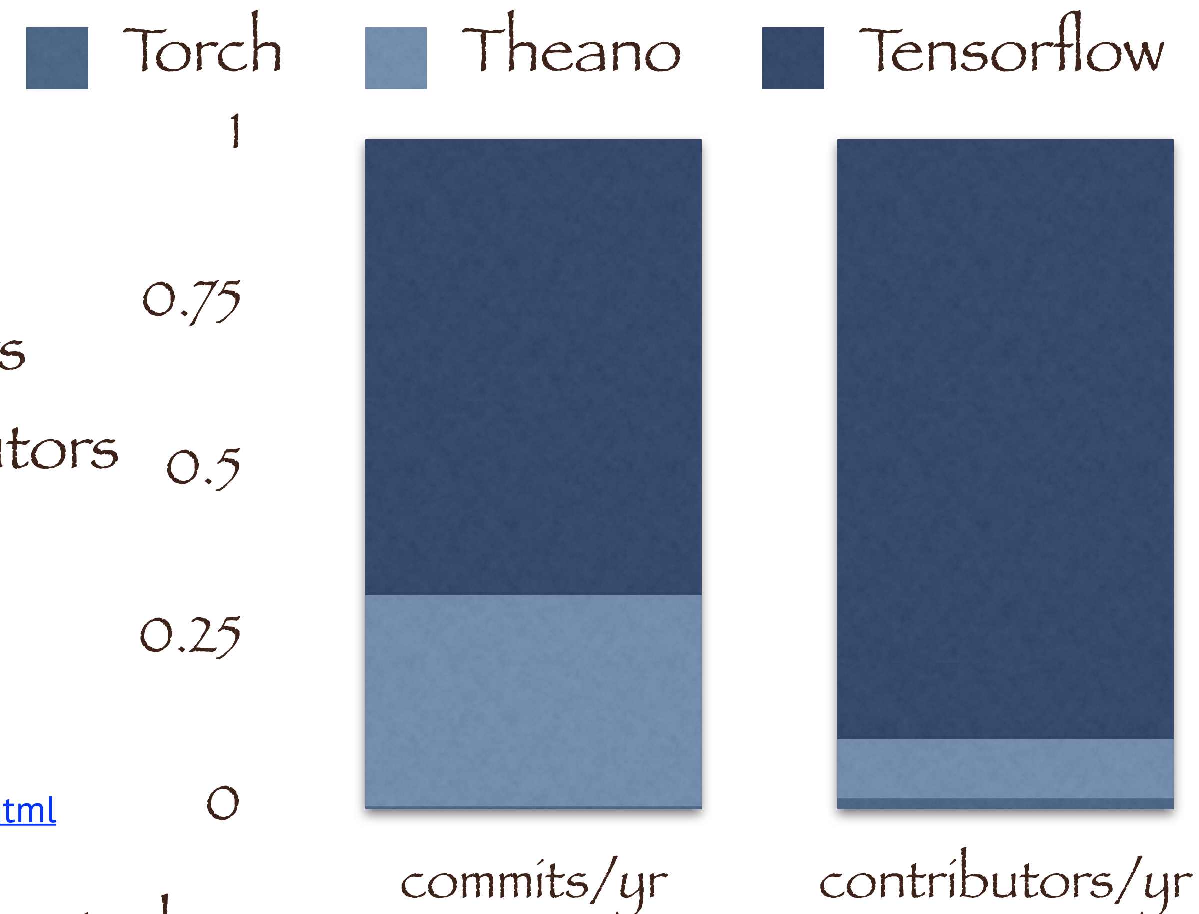
Resume: it's not always the choice for production yet





TF pros, that won't be beaten

$$\frac{commits_k}{years_k} / \sum_{i=(torch, theano, tensorflow)} commits_i / years_i$$



- fundamental

  torch7: 1073 commits, 105 contributors

  theano: 23636 commits, 258 contributors

  tensorflow: 8603 commits, 430 contributors

  TF exists only a year. Theano - more than 6 yrs. Torch - 14 years

- parallelisation. It's simple.

  https://www.tensorflow.org/versions/r0.11/how_tos/distributed/index.html

- Virtually any architecture may be implemented
- Google dataset pretrained models (use or fine tune)

  https://research.googleblog.com/2016/03/train-your-own-image-classifier-with.html

  https://www.tensorflow.org/versions/r0.9/how_tos/image_retraining/index.html

- fast coding
- easy understandable and scalable code
- symbolic computation

Resume: It's THE choice for research/startup/perspective

Symbolic computations

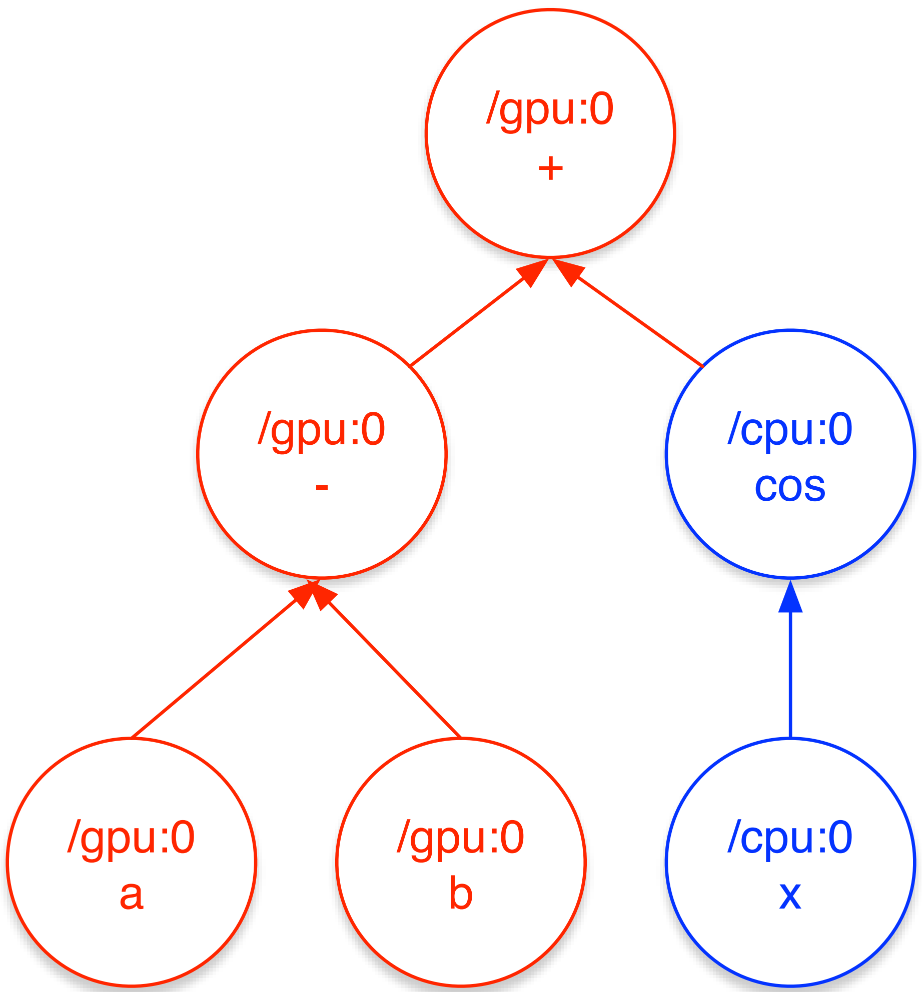
- You don't actually compute. You just say how to compute
- You can think of it as meta programming
- Symbolic computation shows how to get symbolic (common, analytical) solution
- by substituting numerical values to vars, you obtain partial numerical solutions

Symbolic: c = a + b given a=..., b=...
Numerical: 7 = 3 + 4

Benefits:
- simpler automatic differentiation
- easier parallelisation
- differentiation of graph produces graph, so you can get high order derivatives for no cost(PROFIT!!!)
  You say how to symbolically compute the gradient for an op when you make a new op in tf - single method @ops.RegisterGradient("MyOP")

Symbolic computations. TF sample

$(a-b) + \cos(x)$




```
import tensorflow as tf
import numpy as np

with tf.device('/cpu:0'):
    x = tf.constant(np.ones((100,100)))
    y = tf.cos(x)

with tf.device('/gpu:0'):
    a = tf.constant(np.zeros((100,100)))
    b = tf.constant(np.ones((100,100)))
    result = a-b+y

tf_session = tf.Session(
                 config=tf.ConfigProto(
                     log_device_placement=True
                 )
             )
writer = tf.train.SummaryWriter(
    "/tmp/trainlogs2",
    tf_session.graph
)

# then run
# tensorboard --logdir=/tmp/trainlogs2 in shell,
# go to the location suggested by tensorboard,
# `graphs` tab, click on each node/leaf,
# and check where it has been placed
```

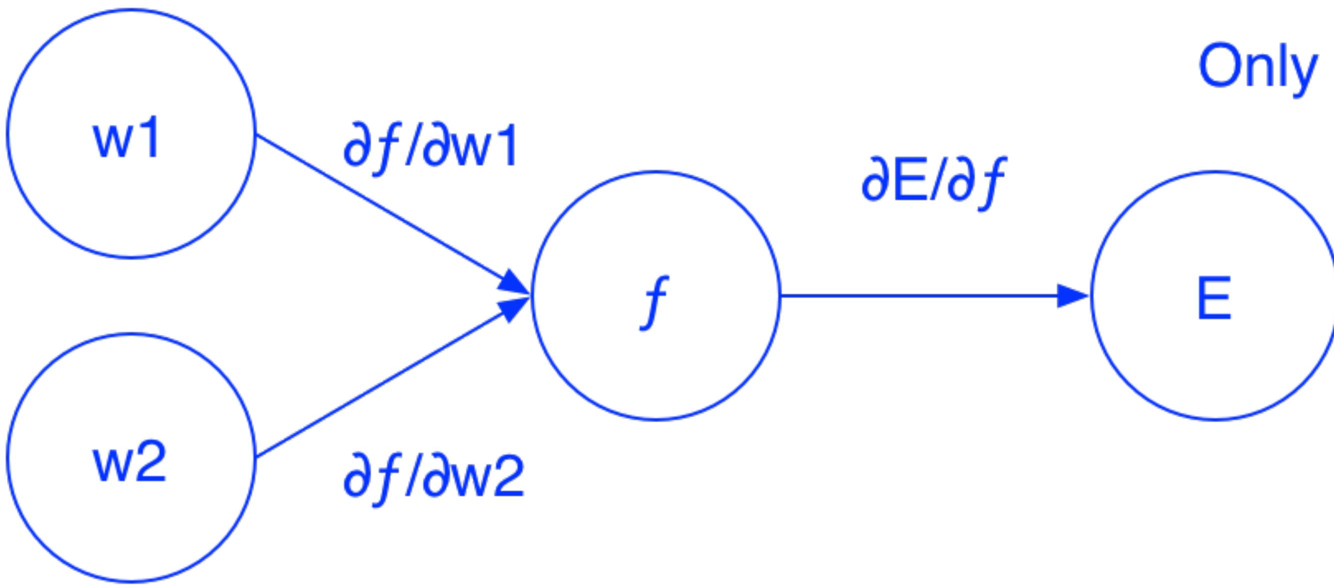
Automatic differentiation

Automatic differentiation is based on chain rule:

$$\frac{\partial E(f(w))}{\partial w} = \frac{\partial E(f(w))}{\partial f(w)} \times \frac{\partial f(w)}{\partial w}$$

But f(w) not depend directly on w,
it may depend on g(w)...

Only calculate $\partial f/\partial w1$, $\partial f/\partial w2$, $\partial E/\partial f$



then

$\partial E/\partial w1 = \partial E/\partial f \cdot \partial f/\partial w1$
$\partial E/\partial w2 = \partial E/\partial f \cdot \partial f/\partial w2$

to obtain ∂E/∂<free param> we keep multiplying up to(included) ∂…/∂<free param>

- Allows us to compute partial derivatives of objective function with respect to each free parameter in one pass.
- Efficient when # of objective functions is small

In TF it's much more convenient than in Torch or Theano

You can think of TF op = torch layer (in terms of automatic differentiation)

http://colah.github.io/posts/2015-08-Backprop/

Installation

Switch off UEFI safe boot (Linux, needed to installproprietary drivers)

Install Drivers (nvidia proprietary) (Linux)

Install CUDA

```
# dpkg -i <your downloaded cuda.deb>; apt-get update; apt-get install cuda
```

Install CUDNN (need nvidia developer account, takes you up to 1 day to get)

install tensorflow

- pip (trivial)
- from sources (you're getting the most recent fixes)

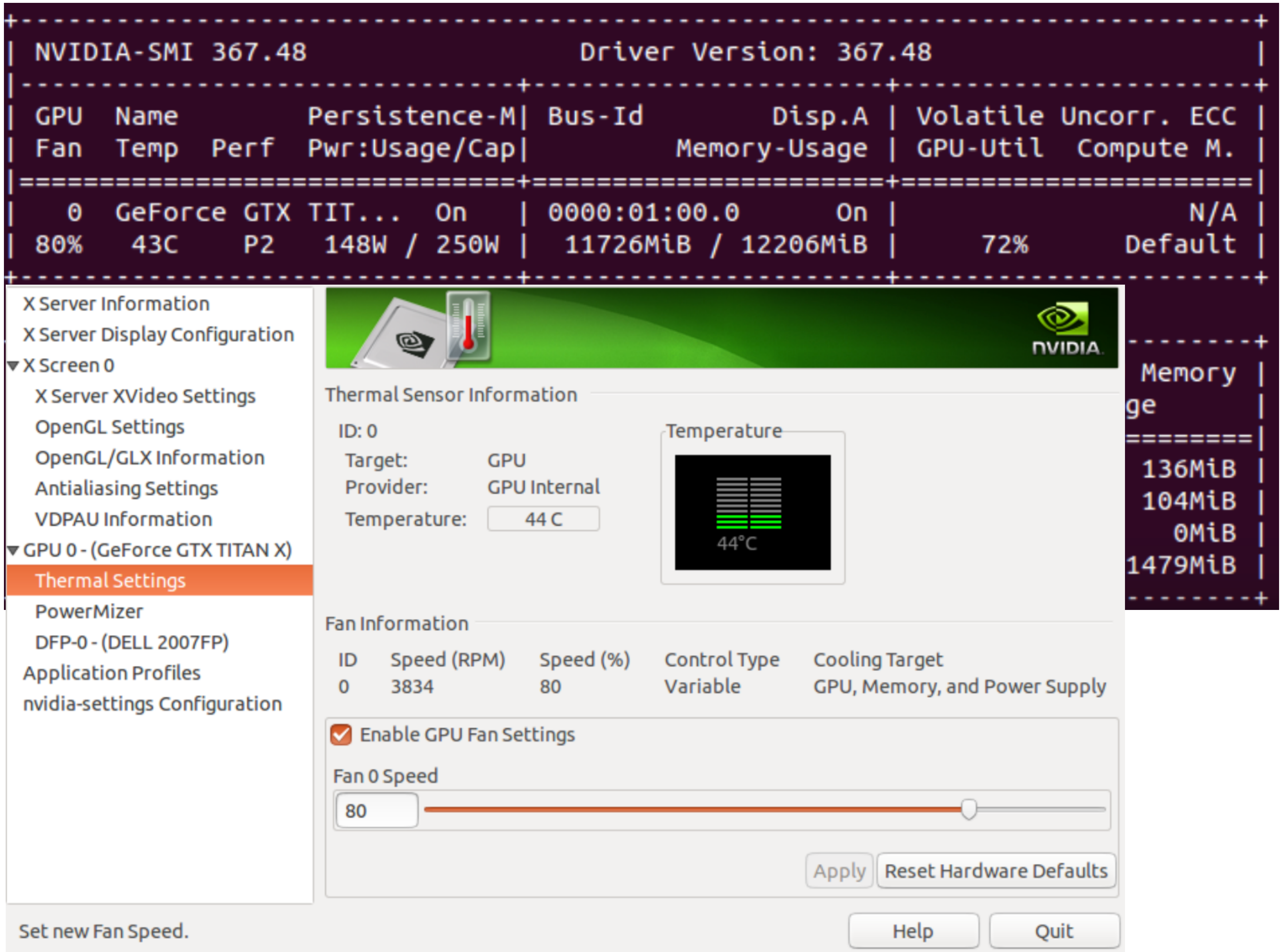
Usage tips

some video cards don't use custom fan speed

`# nvidia-xconfig –cool-bits=4`

then you can use cooling!

`# nvidia-settings -a [gpu:0]/GPUFanControlState=1 -a [fan:0]/GPUTargetFanSpeed=80`



Debugging. Why?

There are no programs without bugs. Period.

Your bugs:

- tensor shape mismatch
- OOM
- wrong calculation graphs
- gradients (numerical, BPTT stability)
- visual debug: agent behaviour

Developers' bugs:

- something works not as expected
- your code doesn't work after update

Debugging. Your bugs.

Shape mismatch, virtually all bugs:

```
import ipdb; ipdb.set_trace() # sometimes in catch
ipdb> session.graph.get_tensor_by_name('node_path').op.<press TAB in ipdb!!! ;)>
ipdb> <tensor/op>.get_shape()
```

Check devices:

```
tf_session = tf.Session(config=tf.ConfigProto(log_device_placement=True))
```

Check/simplify graph(tensorboard):

```
writer = tf.train.SummaryWriter("/tmp/trainlogs", self.tf_session.graph)
$ tensorboard --logdir=/tmp/trainlogs
```

Use variable scope! easier code, easier debug!

you can think of TF graph as a parallel program, accessible through tf.Session() object

OOM

- check size of your variables or
- change memory usage strategy:

```
config = tf.ConfigProto()
config.gpu_options.allow_growth = True
self.tf_session = tf.Session(config=config)
```

Check gradients numerically

Check if gradients vanish/explode over time(especially for RNNs)

Debugging. Your bugs. Advanced.

- when tensorboard failed due to large/inconsistent/whatsoever graph

Graph visualization failed: TypeError: undefined is not an object (evaluating 'rawNodes.length')

- or you're too lazy/need a quick glance

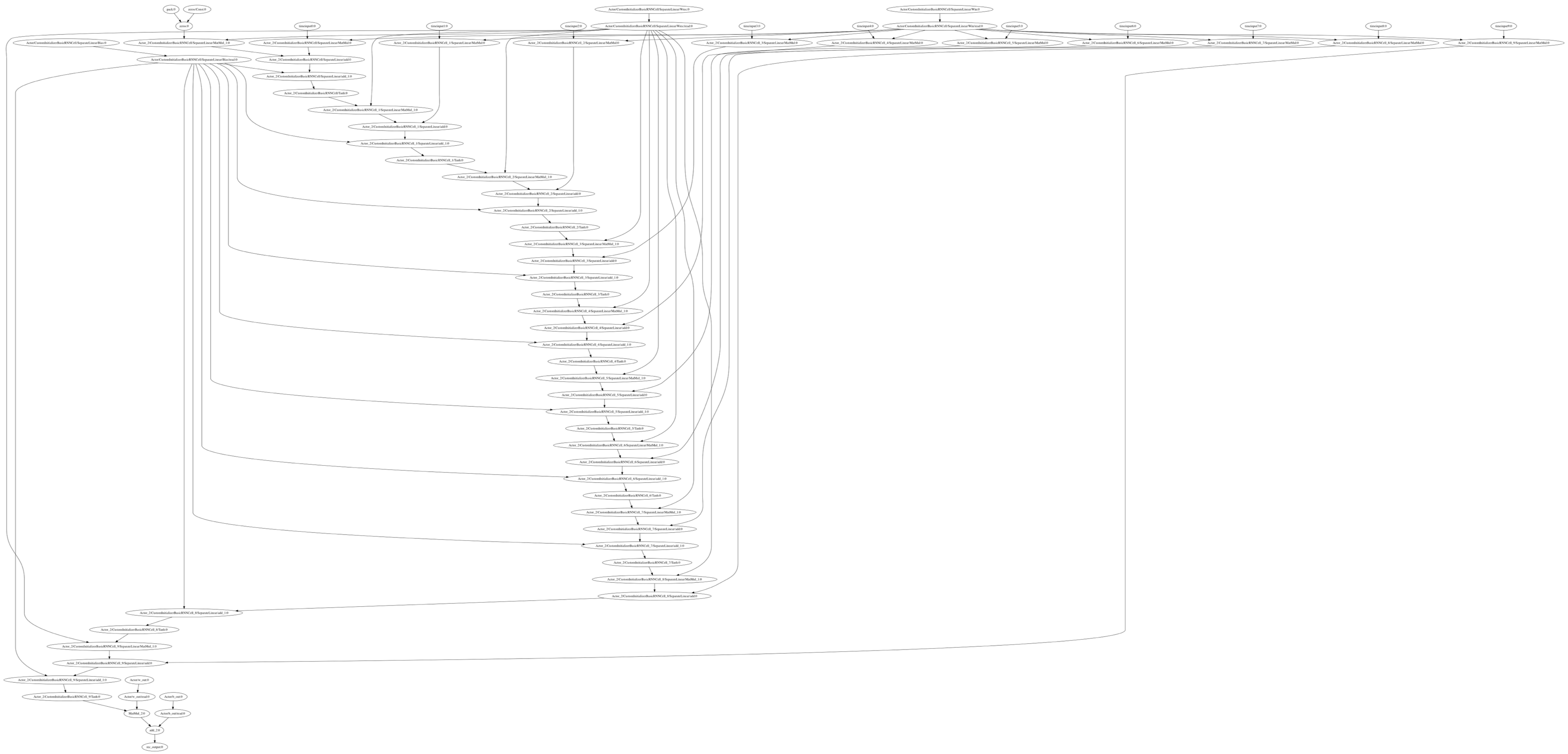
https://github.com/oleksandr-khryplyvenko/tf-graph-visualiser

Assuming, that tf_session is a tf.Session() object.

```
ipdb> node_to_display = tf_session.graph.get_tensor_by_name('softmax:0')
ipdb> from nodedisplay import draw
ipdb> draw(node_to_display, tf_session, 'inception_v3_net')
```

After this, you'll get $HOME/inception_v3_net.svg file

Debugging. Your bugs. Advanced.



Much bigger for real nets

Debugging. Artillery.

If you suspect bug in Master(unlikely but possible):

```
$ cd tensorflow; git pull origin master
```

Then rebuild pip package & reinstall.

If something breaks, use google. Very often you just need to reinstall some package tf depends on.

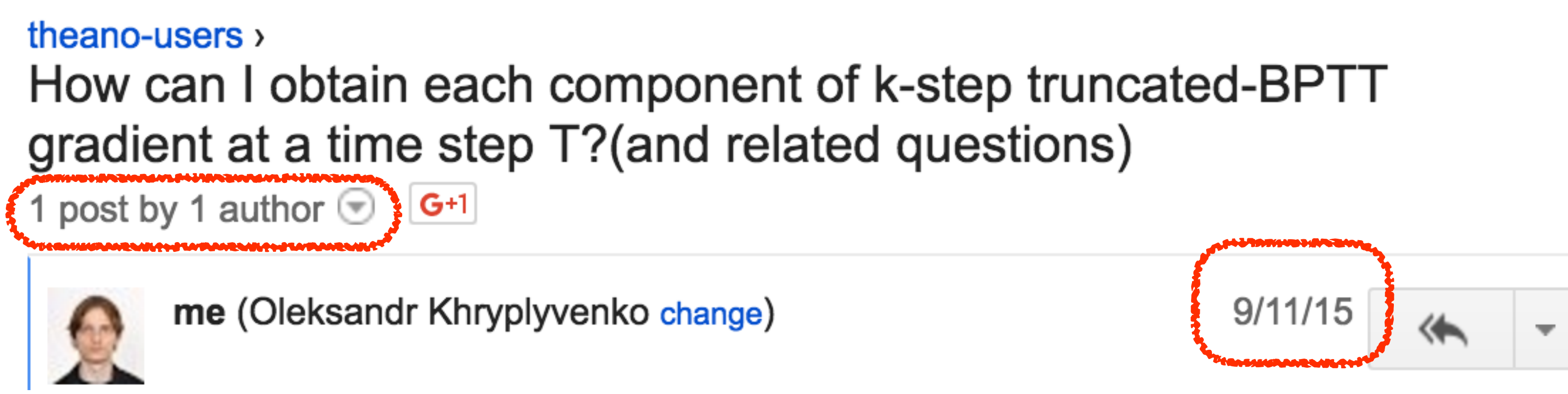
If this hasn't helped, try to solve/hotfix this problem on your own.
The code is pretty simple, up to platform specific prototypes.
Hasn't helped? Post a bug. And rollback meanwhile, if possible.

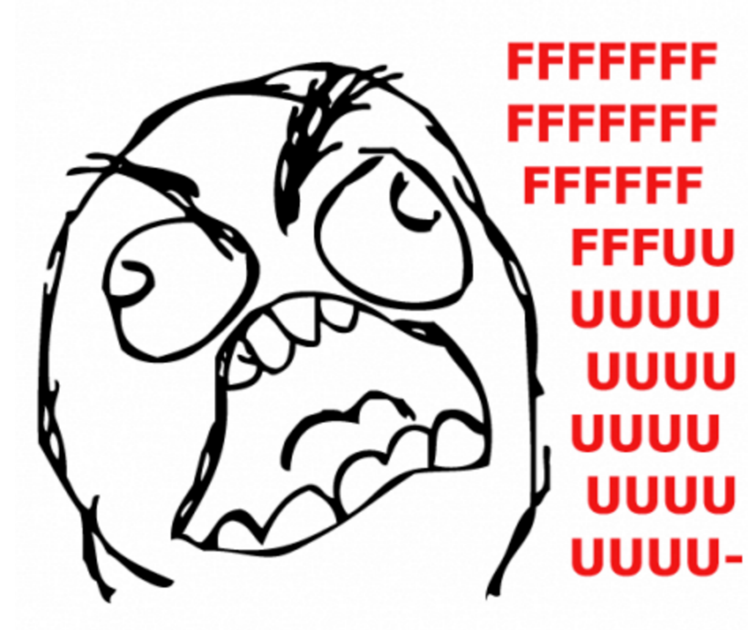
Debugging. Advanced. RNN stability.

Goal: get and visualise gradients for BPTT

https://groups.google.com/forum/?hl=en#!topic/theano-users/lTVpc4XD8C8
https://stackoverflow.com/questions/32553374/how-can-i-get-not-only-an-unrolled-for-k-steps-truncated-bptt-grad-in-theano-sc

Theano:


theano-users ›
How can I obtain each component of k-step truncated-BPTT gradient at a time step T?(and related questions)
1 post by 1 author G+1
me (Oleksandr Khryplyvenko change) 9/11/15



Debugging. Advanced. RNN stability.

```
def _linear(args, output_size, bias, bias_start=0.0, scope=None):
    """Linear map: sum_i(args[i] * W[i]), where W[i] is a variable."""

    shapes = [a.get_shape().as_list() for a in args]
    for shape in shapes:
        total_arg_size += shape[1]

    with vs.variable_scope(scope or "Linear"):
        matrix = vs.get_variable("Matrix", [total_arg_size, output_size])
        res = math_ops.matmul(array_ops.concat(1, args), matrix)
        bias_term = vs.get_variable(
        "Bias", [output_size], initializer=init_ops.constant_initializer(bias_start))

    return res + bias_term


class BasicRNNCell(RNNCell):
    def __call__(self, inputs, state, scope=None):
      """Most basic RNN: output = new_state = activation(W * input + U * state + B)."""
      with vs.variable_scope(scope or type(self).__name__):  # "BasicRNNCell"
        output = self._activation(_linear([inputs, state], self._num_units, True))
      return output, output
```

Surprise! Only predefined initialisers,
matrix is glued

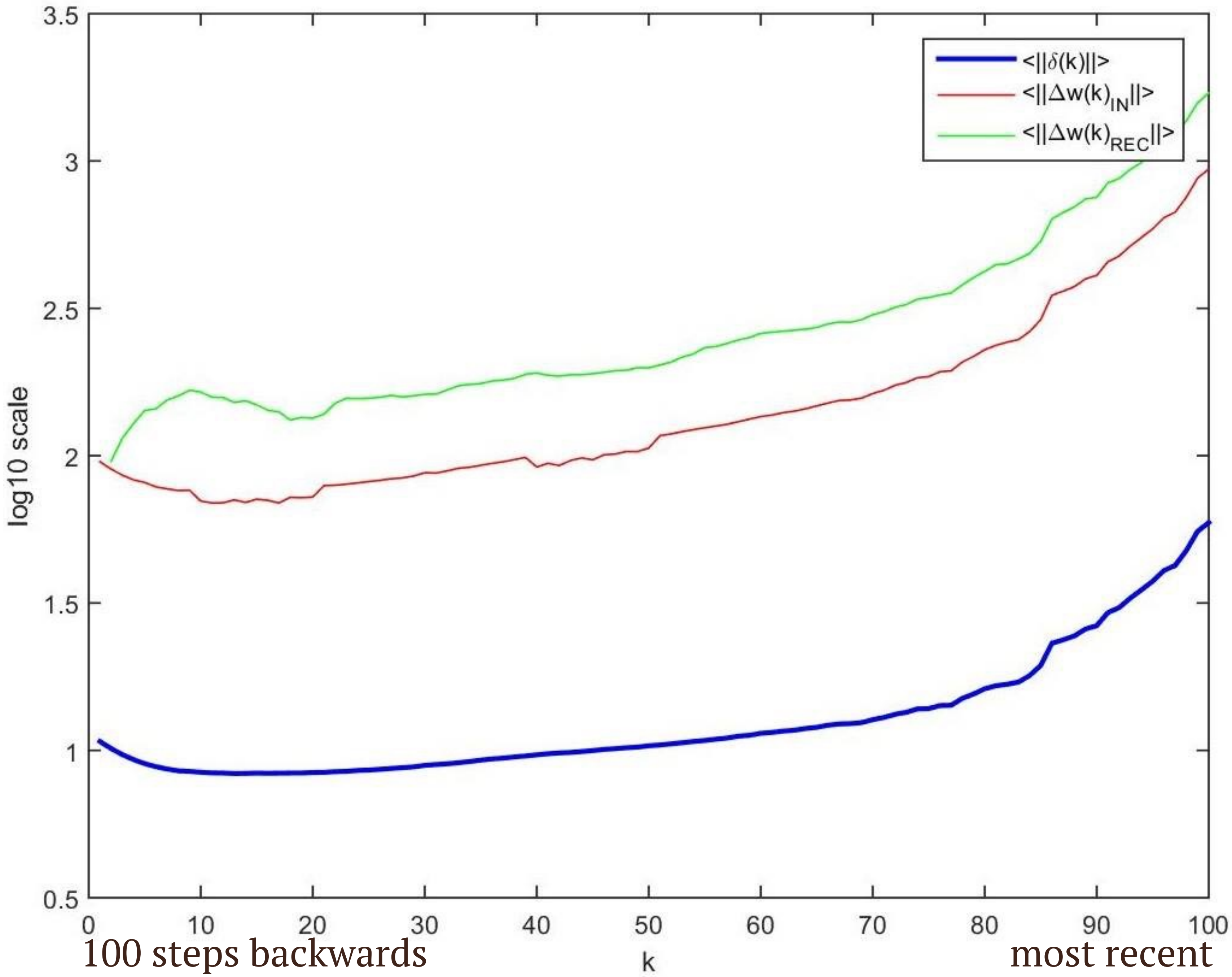
Debugging. Advanced. RNN stability.

Don't care. In TF, you can always take


```
def separate_linear(args, argnames, output_size, bias, bias_start=0.0, scope=None, initializers=None):
    with tf.variable_scope(scope or "SeparateLinear"):
      arg, shape, matrixname, initializer = args[0], shapes[0], argnames[0], initializers[0]
      matrix = tf.get_variable(matrixname, [shape[1], output_size], initializer=initializer)
      res = tf.matmul(arg, matrix)

      for arg, shape, matrixname, initializer in zip(args, shapes, argnames, initializers)[1:]:
          matrix = tf.get_variable(matrixname, [shape[1], output_size], initializer=initializer)
          res += tf.matmul(arg, matrix)

      if bias:
          res += tf.get_variable("Bias", [output_size], initializer=tf.constant_initializer(bias_start))

  return res


class CustomInitializerBasicRNNCell(tf.nn.rnn_cell.BasicRNNCell):
    def __call__(self, inputs, state, scope=None):
        """Most basic RNN: output = new_state = tanh(W * input + U * state + B)."""
        with tf.variable_scope(scope or type(self).__name__):
            output = tf.tanh(
                separate_linear.separate_linear(
                    [inputs, state],
                    ["Win", "Wrec"],
                    self._num_units,
                    True,
                    initializers=[
                        tf.random_uniform_initializer(minval=-tf.sqrt…, maxval= ), # Input matrix
                        tf.random_uniform_initializer()                            # Recurrent matrix
                    ]
                )
            )
        return output, output
```

Debugging. Advanced. RNN stability.

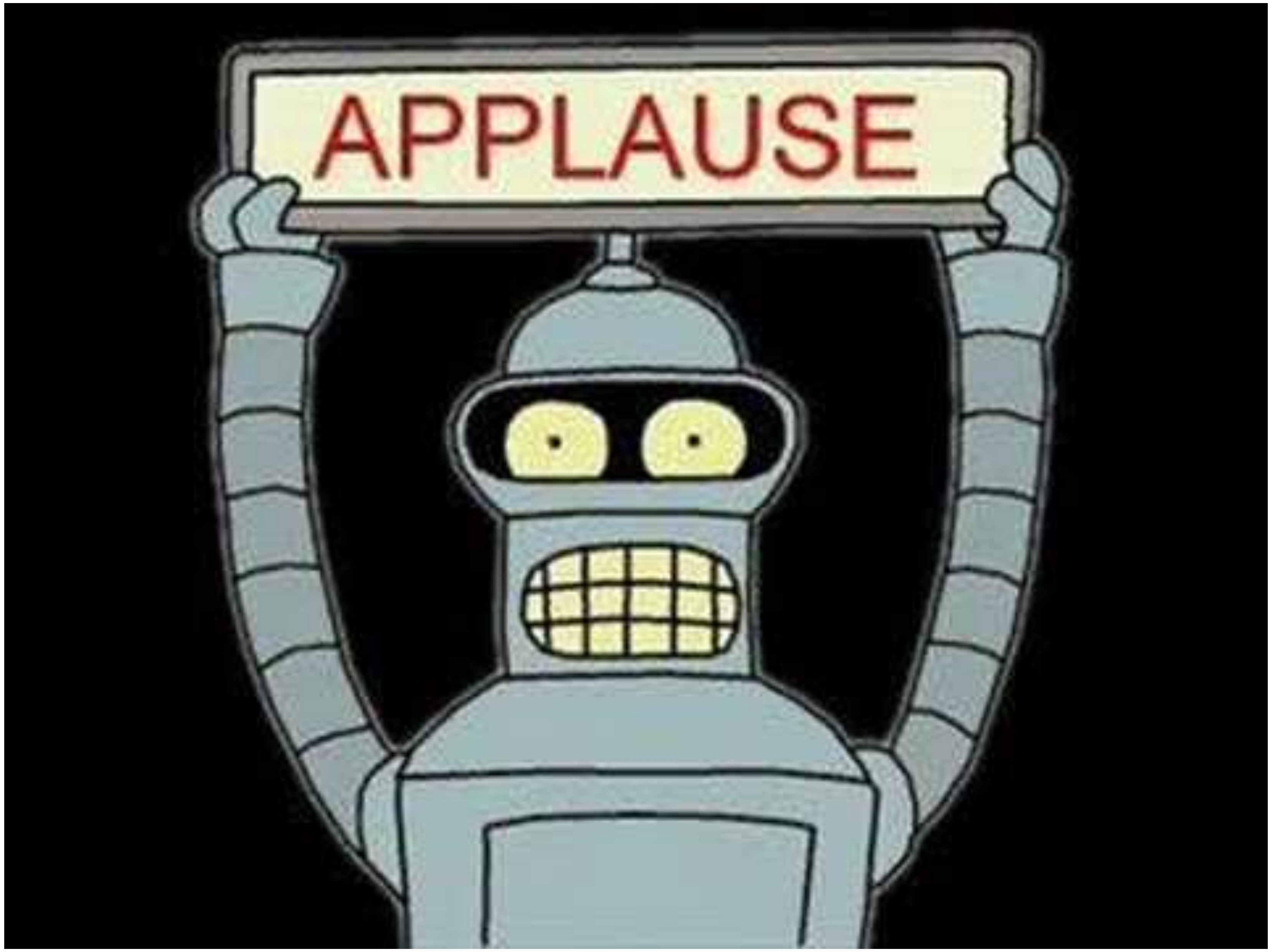
Now we have separate Win, Wrec, custom-initialized



100 steps backwards

most recent

© Artem Chernodub


APPLAUSE