redisconf19

Serving Deep Learning Models at Scale with RedisAI

Luca Antiga
[tensor]werk, *CEO*

presented by
redislabs
HOME OF REDIS

Agenda:

1. Deep learning basics

   Fundamentals to get going

2. RedisAI

   Architecture, API, operation

3. Hands-on

   Image recognition and text generation

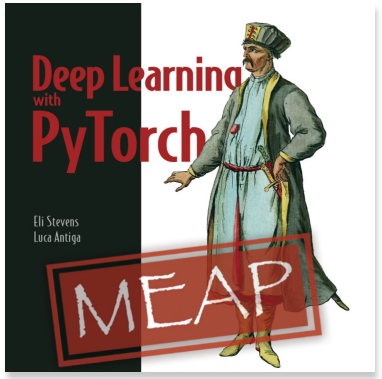
Who am I (@lantiga)

- Co-founder and CEO of **[tensor]werk**
  Infrastructure for data-defined software
- Co-founder of **Orobix**
  AI in healthcare/manufacturing/gaming/+
- **PyTorch** contributor in 2017-2018
- Co-author of **Deep Learning with PyTorch**, Manning (with **Eli Stevens**)

[tensor]werk

OROBIX
TOOLS WITH A MIND

PyTorch



Training virtual labs
graciously sponsored by:

appsembler

Virtual Software Labs
by appsembler

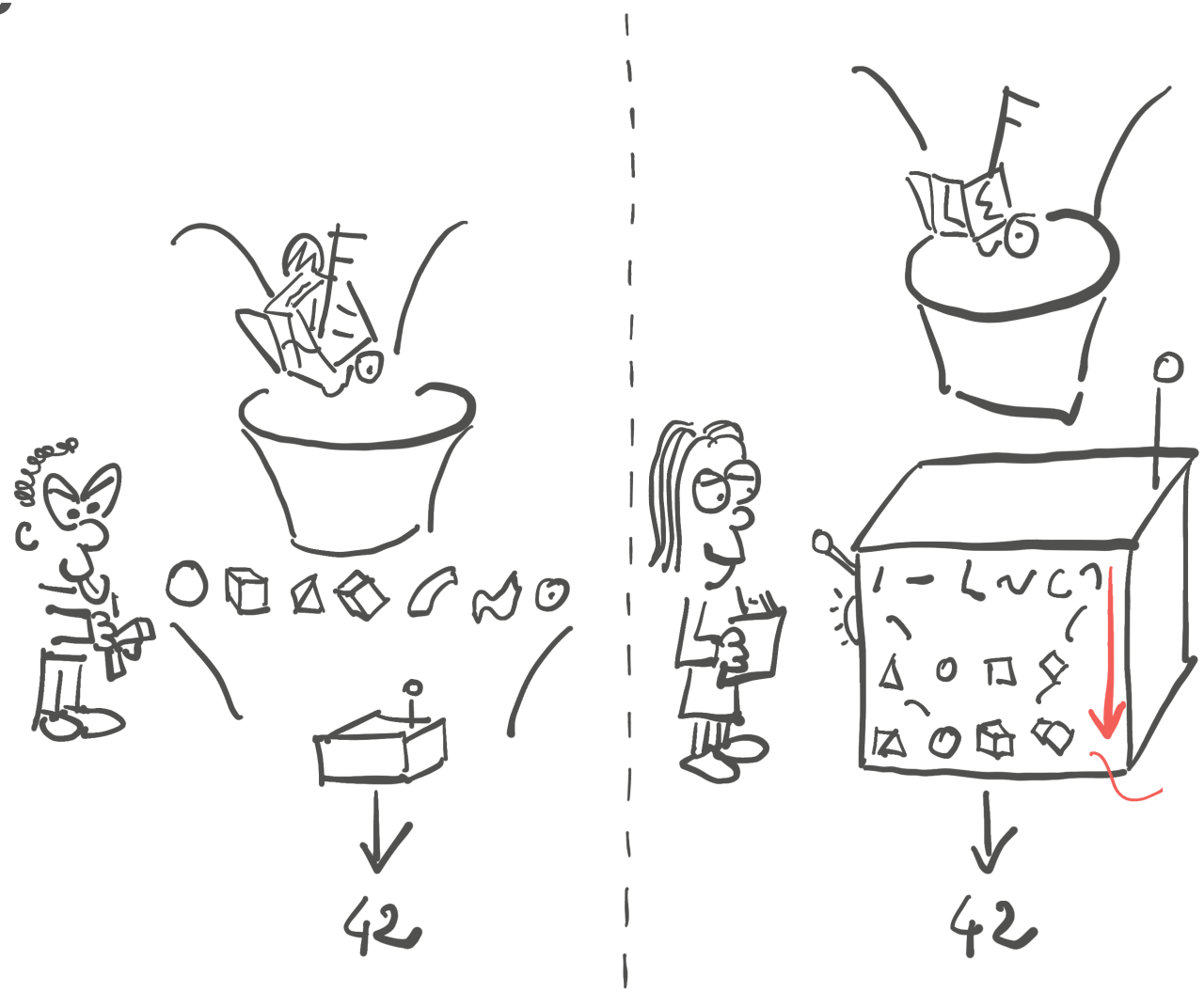
Start your training by going to:

http://bit.ly/day0-dashboard

Deep learning basics

The era of data-defined software

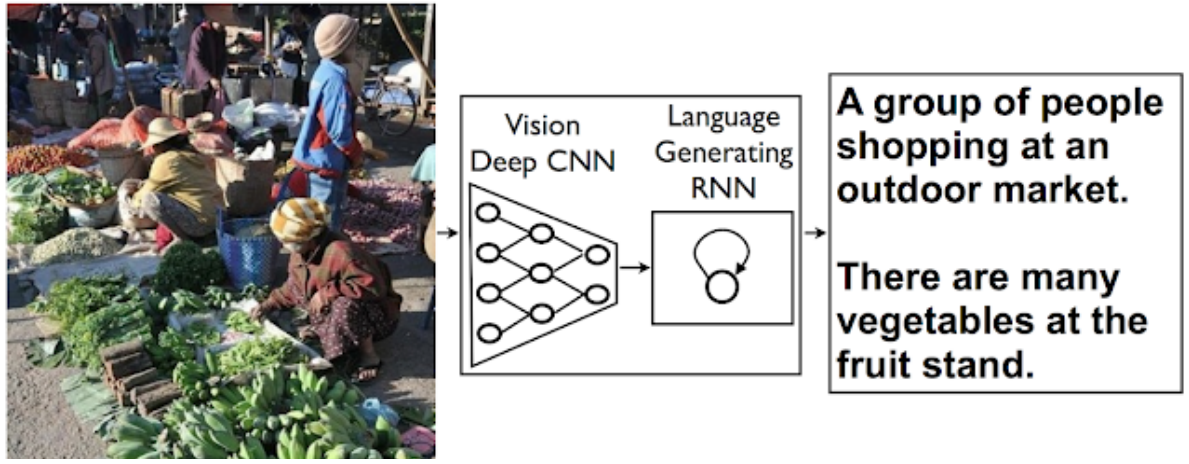
- Programs whose behavior is defined by data
- New era for software development
- Needs to leverage on best practices of software engineering
- From data scientists to developers

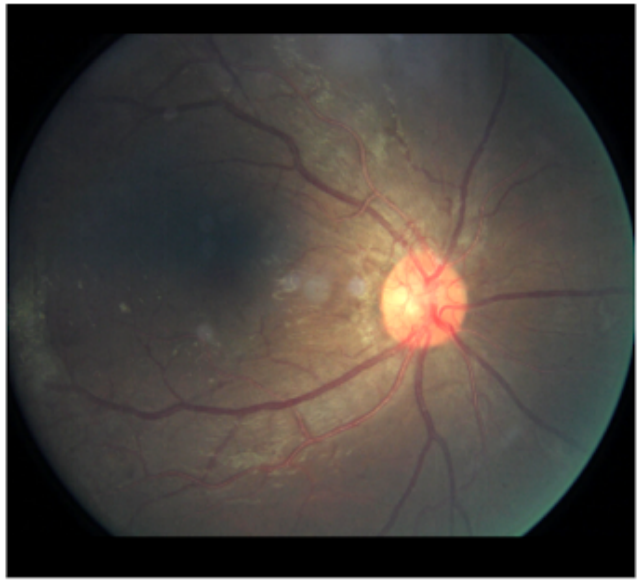


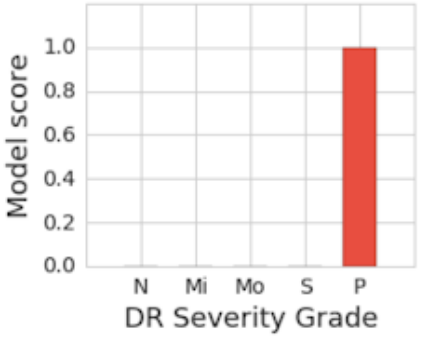
https://ai.googleblog.com

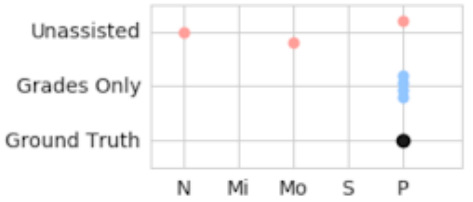
Deep learning

- Deep neural networks approximate functions based on examples
- Pervasive
- Solves problems that looked very hard a few years back
- Limit is the data, not the programmer (kinda)
- Not AGI, but a tool with superpowers



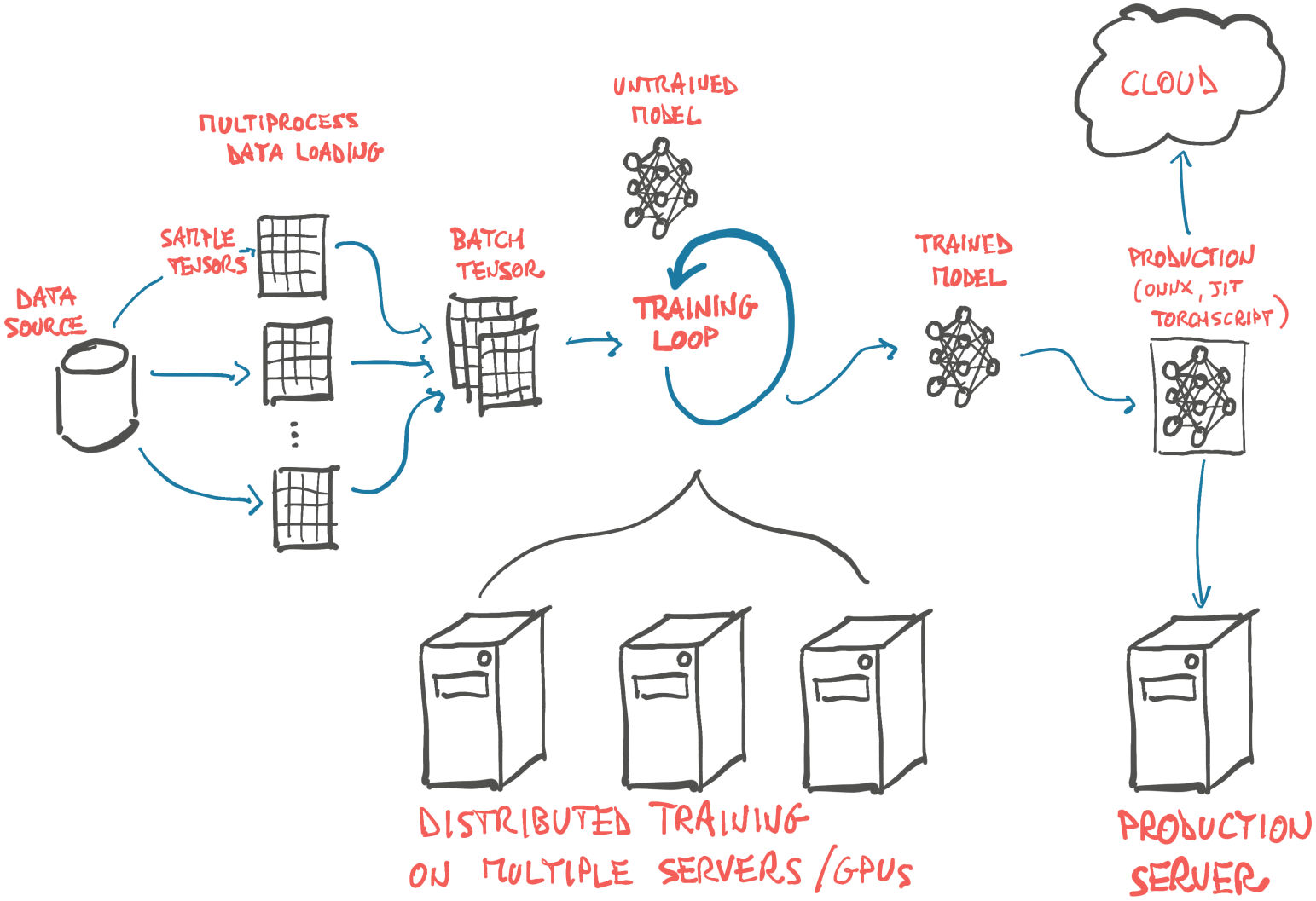






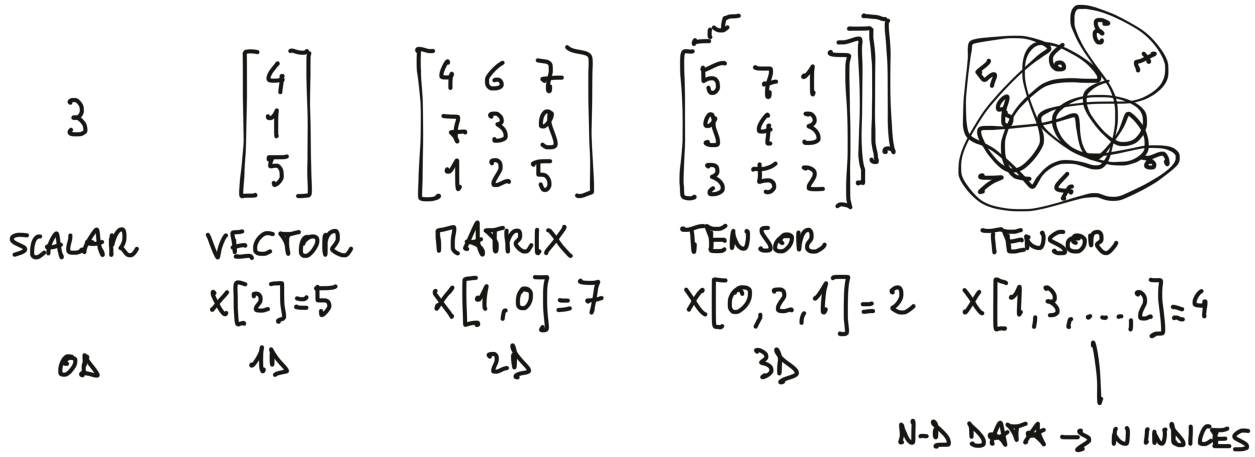
Deep learning frameworks

- TensorFlow
- PyTorch
- mxnet
- CNTK, Chainer,
- DyNet, DL4J,
- Flux, …



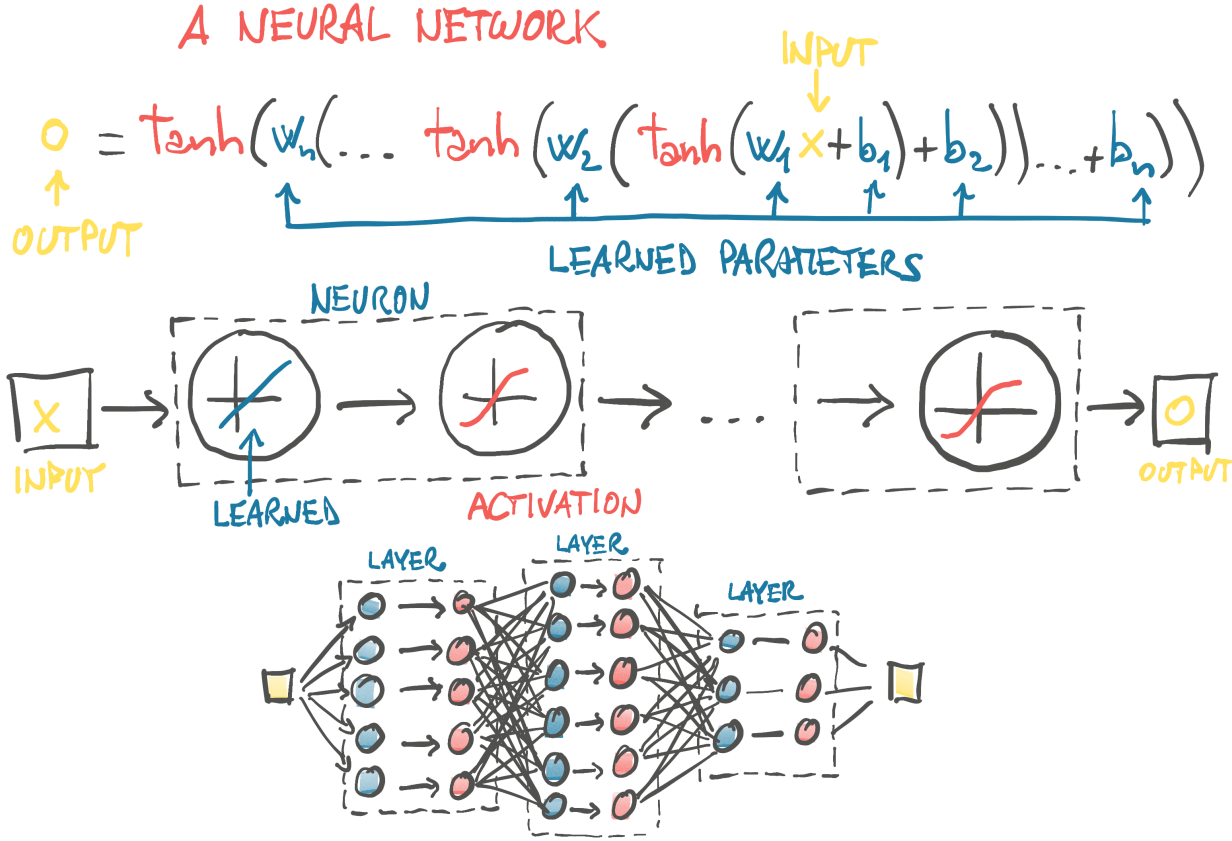
Deep learning 101 - Tensors

- The main data structure in deep learning
- Multidimensional arrays: arrays that can be indexed using more than one index
- Memory is still 1-dimensional
- It's just a matter of access and operator semantics



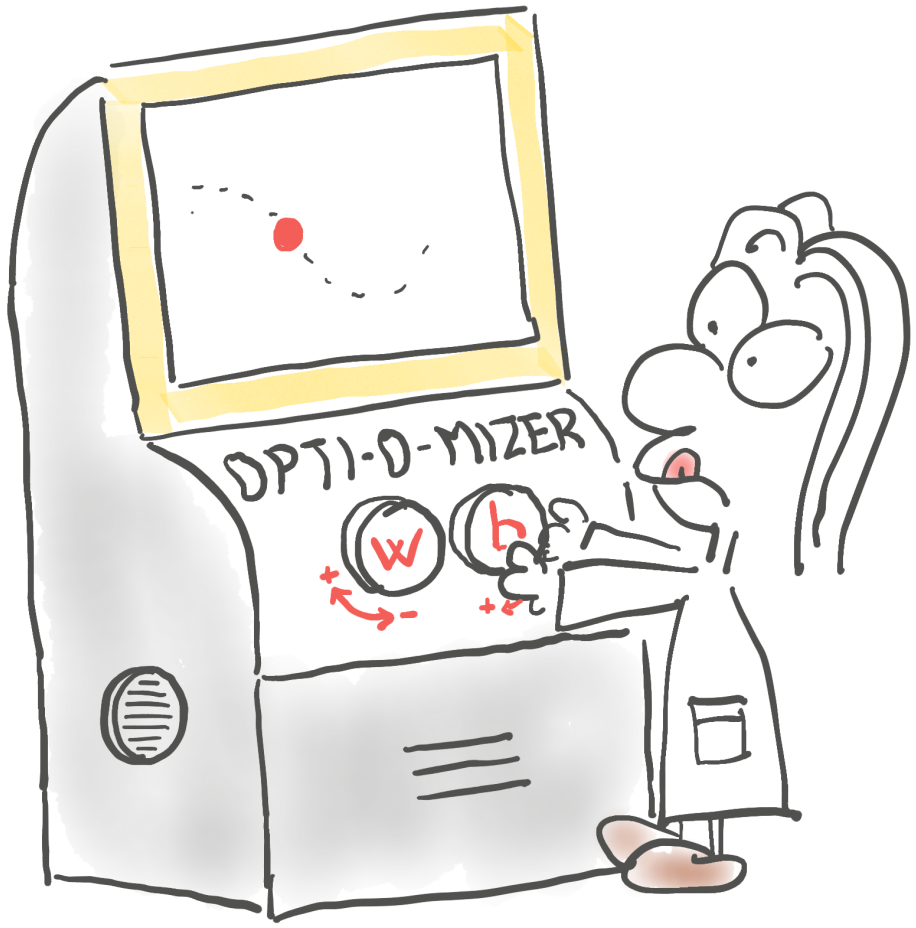
Deep learning 101 - Models

- Operations on tensors to produce other tensors
- Operations are layed out in a computation graph
- A graph is like a VM IR with an op-set limited to tensor ops
- E.g.:
  - linear transformations (w*x + b) where w, b are typically learned
  - + non-linearity: σ(w*x + b)
  - heavily nested



Deep learning 101 - Learning

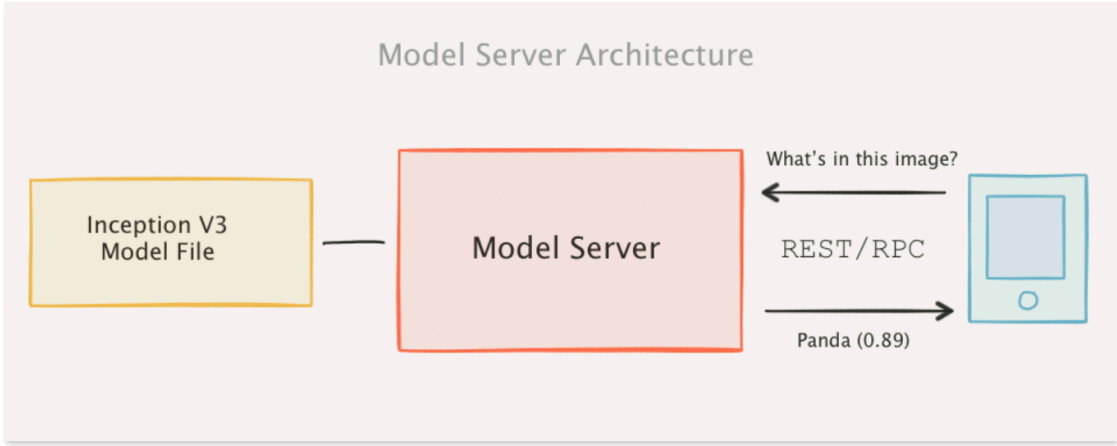
- Optimizing weights so to minimize a measure of error for a task (loss function)
- Once training is converged, we can rerun the computation graph with the learned weights on new data (inference)



So you’ve got your model trained

Production strategies

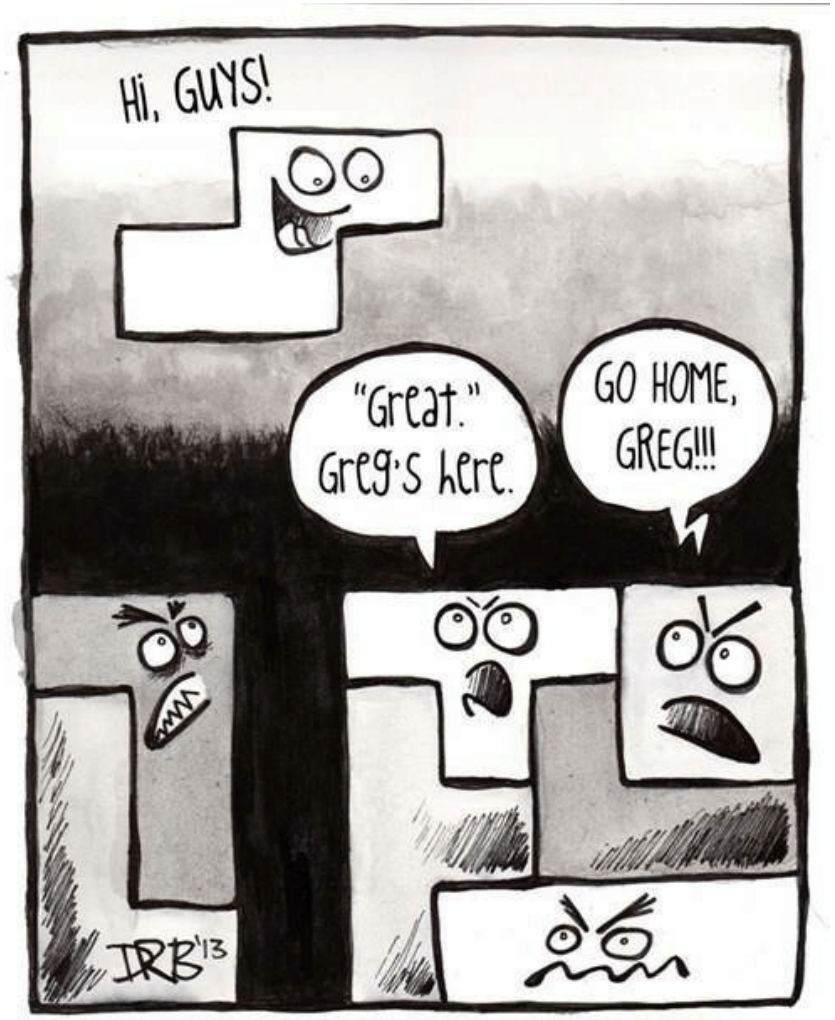
- Python code behind e.g. Flask
- Execution service from cloud provider
- Runtime
  - TensorFlow serving
  - Clipper
  - NVIDIA TensorRT inference server
  - MXNet Model Server
  - …
- Bespoke solutions (C++, …)



https://medium.com/@vikati/the-rise-of-the-model-servers-9395522b6c58

Production requirements

- Must **fit** the technology **stack**
- Not just about languages, but about **semantics**, **scalability**, **guarantees**
- Run **anywhere**, **any size**
- **Composable** building blocks
- Must try to **limit** the amount of **moving parts**
- And the amount of **moving data**
- Must make best **use of resources**

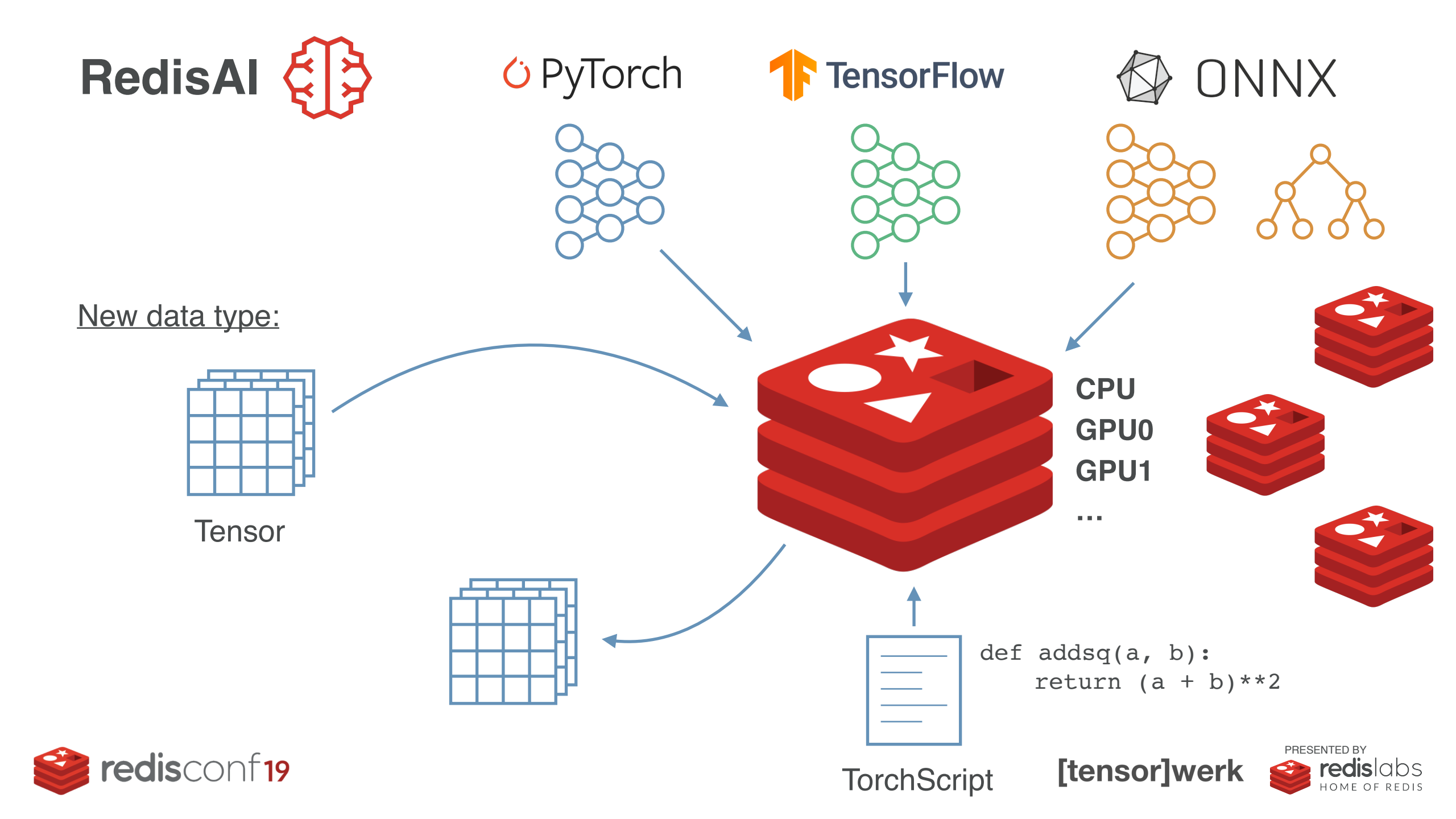


RedisAI

What it is

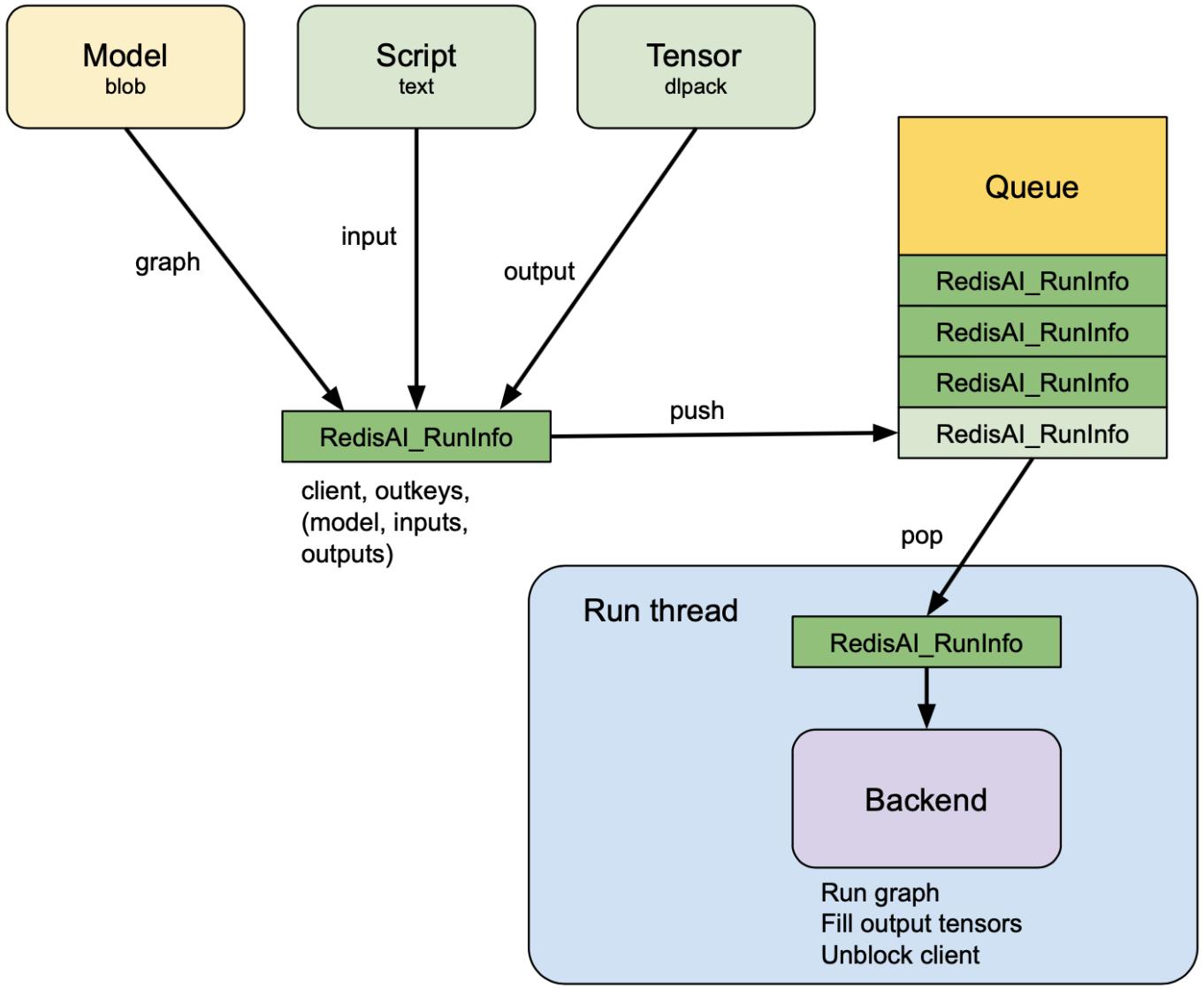
- A Redis module providing
- Tensors as a data type
- and Deep Learning model execution
- on CPU and GPU

- It turns Redis into a **full-fledged deep learning runtime**
- While still being **Redis**


RedisAI
PyTorch
TensorFlow
ONNX
New data type:
Tensor
CPU
GPU0
GPU1
...
def addsq(a, b):
return (a + b)**2
TorchScript

Architecture

- Tensors: framework-agnostic
- Queue + processing thread
  - Backpressure
  - Redis stays responsive
- Models are kept hot in memory
- Client blocks



Where to get it

- redisai.io
- github.com/RedisAI/RedisAI

```
$ docker run -p 6379:6379 -it --rm redisai/redisai
```

Where to get it

- redisai.io
- github.com/RedisAI/RedisAI

```
$ git clone git@github.com:RedisAI/RedisAI.git
$ bash get_deps.sh
$ mkdir build && cd build
$ cmake -DDEPS_PATH=../deps/install .. && make
$ cd ..
$ redis-server --loadmodule build/redisai.so
```

Where to get it

- redisai.io
- github.com/RedisAI/RedisAI

```
$ redis-server &
$ redis-cli
> MODULE LOAD /usr/local/lib/redisai.so
```

or

```
$ vim redis.conf
  loadmodule /usr/local/lib/redisai.so
```

API: Tensor

- AI.TENSORSET
- AI.TENSORGET

```
AI.TENSORSET foo FLOAT 2 2 VALUES 1 2 3 4
AI.TENSORSET foo FLOAT 2 2 BLOB < buffer.raw

AI.TENSORGET foo BLOB
AI.TENSORGET foo VALUES
AI.TENSORGET foo META
```

API: Tensor

- based on dlpack https://github.com/dmlc/dlpack
- framework-independent

```
typedef struct {
  void* data;
  DLContext ctx;
  int ndim;
  DLDataType dtype;
  int64_t* shape;
  int64_t* strides;
  uint64_t byte_offset;
} DLTensor;
```

API: Model

- AI.MODELSET

```
AI.MODELSET resnet18 TORCH GPU < foo.pt
```

```
AI.MODELSET resnet18 TF CPU INPUTS in1 OUTPUTS linear4 < foo.pt
```

To load TF graph, use Netron:

https://github.com/lutzroeder/netron

https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

API: Model

- AI.MODELRUN

```
AI.MODELRUN resnet18 INPUTS foo OUTPUTS bar
```

https://www.codeproject.com/Articles/1248963/Deep-Learning-using-Python-plus-Keras-Chapter-Re

Exporting models

- TensorFlow (+ Keras): freeze graph

```
import tensorflow as tf

var_converter = tf.compat.v1.graph_util.convert_variables_to_constants

with tf.Session() as sess:
    sess.run([tf.global_variables_initializer()])
    frozen_graph = var_converter(sess, sess.graph_def, ['output'])

tf.train.write_graph(frozen_graph, '.', 'resnet50.pb', as_text=False)
```

https://github.com/RedisAI/redisai-examples/blob/master/models/imagenet/tensorflow/model_saver.py

Exporting models

- PyTorch: JIT model

```
import torch

batch = torch.randn((1, 3, 224, 224))

traced_model = torch.jit.trace(model, batch)

torch.jit.save(traced_model, 'resnet50.pt')
```

https://github.com/RedisAI/redisai-examples/blob/master/models/imagenet/pytorch/model_saver.py

API: Script

- AI.SCRIPTSET
- AI.SCRIPTRUN

addtwo.txt

```
def addtwo(a, b):
    return a + b
```

```
AI.SCRIPTSET myadd2 GPU < addtwo.txt
```

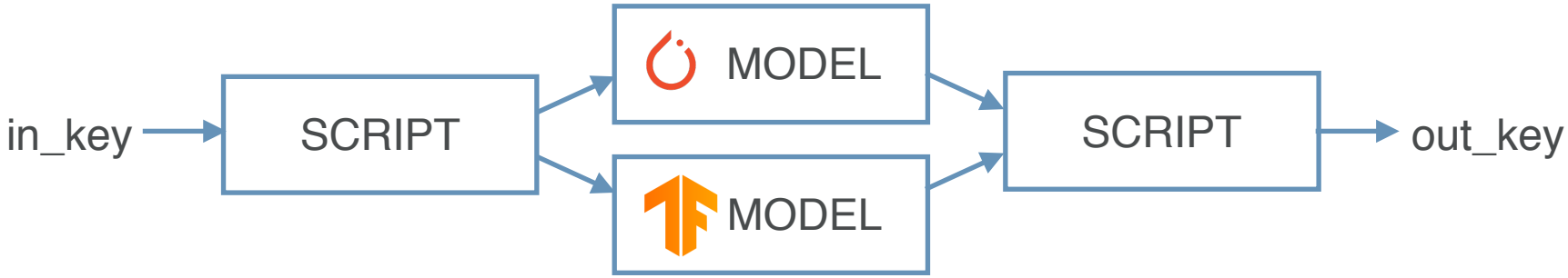
```
AI.SCRIPTRUN myadd2 addtwo INPUTS foo OUTPUTS bar
```

Scripts?

- SCRIPT is a TorchScript interpreter
- Python-like syntax for **tensor ops**
- on **CPU** and **GPU**
- **Vast library** of tensor operations
- Allows to prescribe computations directly (without exporting from a Python env, etc)
- Pre-proc, post-proc (but not only)

TORCHSCRIPT

Ref: https://pytorch.org/docs/stable/jit.html



Hands-on

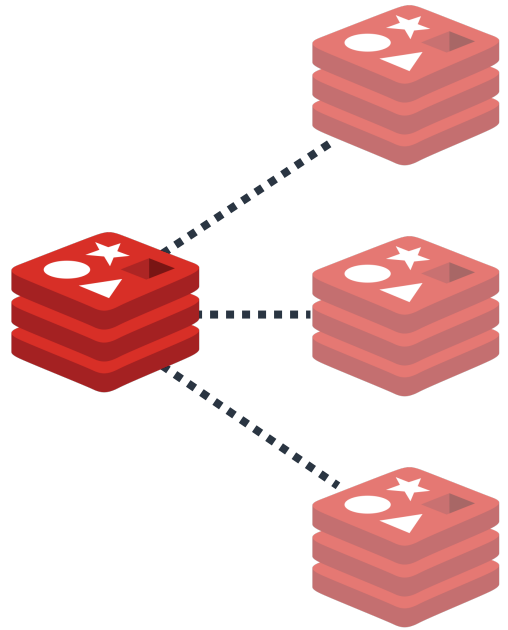
- Create two 3x3 tensors
- Create a SCRIPT that performs element-wise multiply
- Set it
- Run it

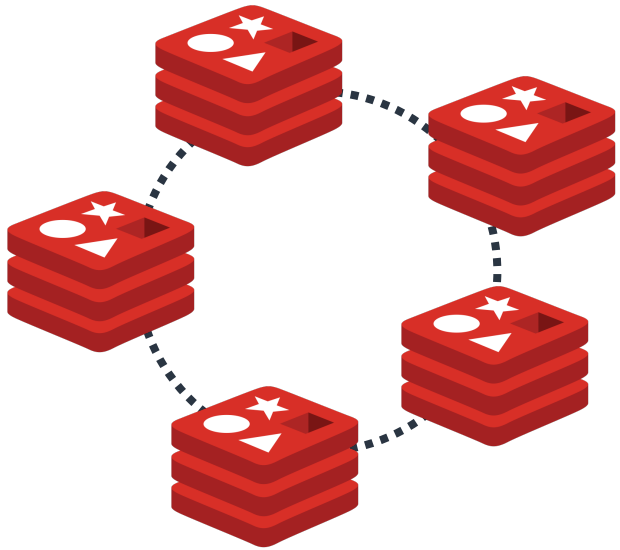
http://bit.ly/day0-dashboard

Replication

- **Master-replica** supported for all data types
- Right now, run cmds replicated too (post-conf: replication of results of computations where appropriate)

- **Cluster** supported, caveat: sharding models and scripts
- For the moment, use *hash tags*
  `{foo}resnet18 {foo}input1234`

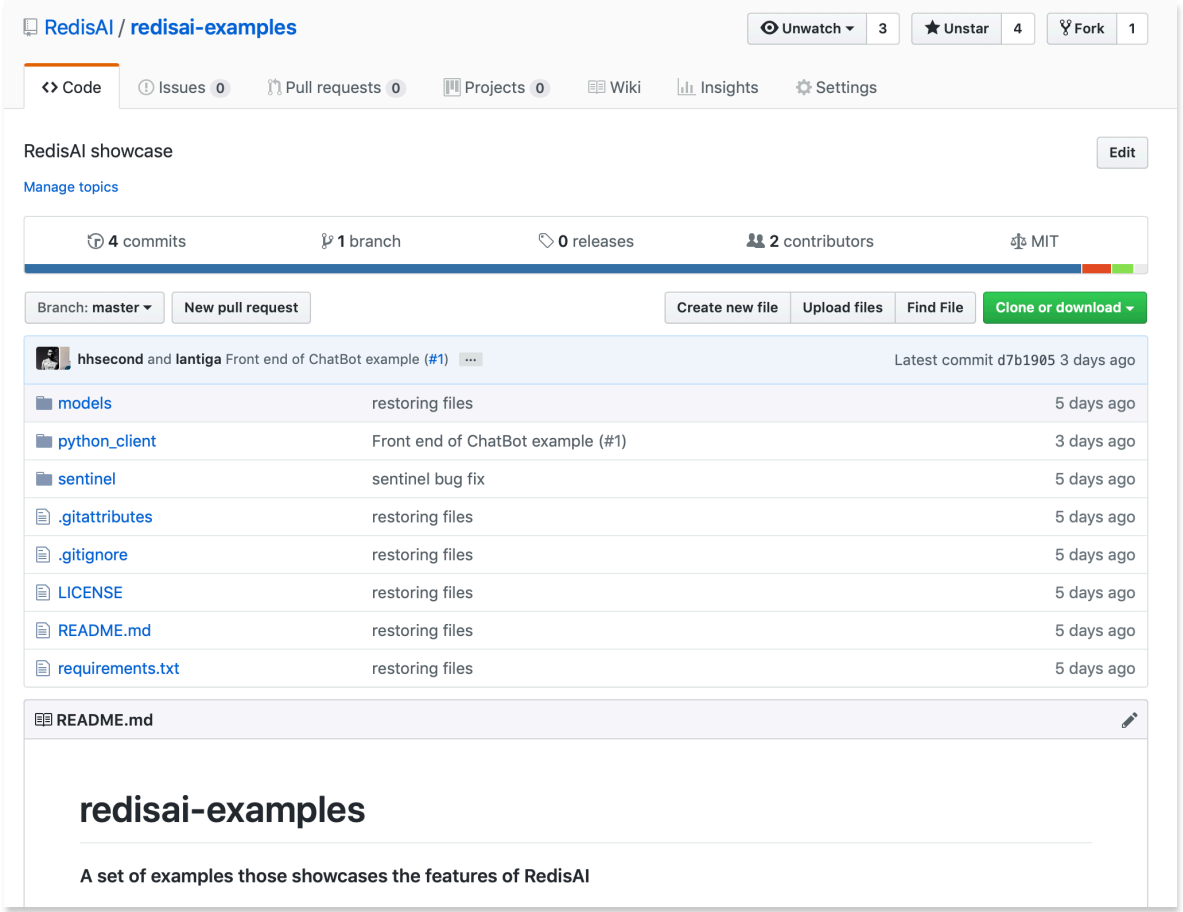




Examples

- https://github.com/RedisAI/redisai-examples
  - Imagenet
  - CharRNN
  - Chatbot
  - Sentinel

Set up by Sherin Thomas (**@hhsecond**)



Hands-on

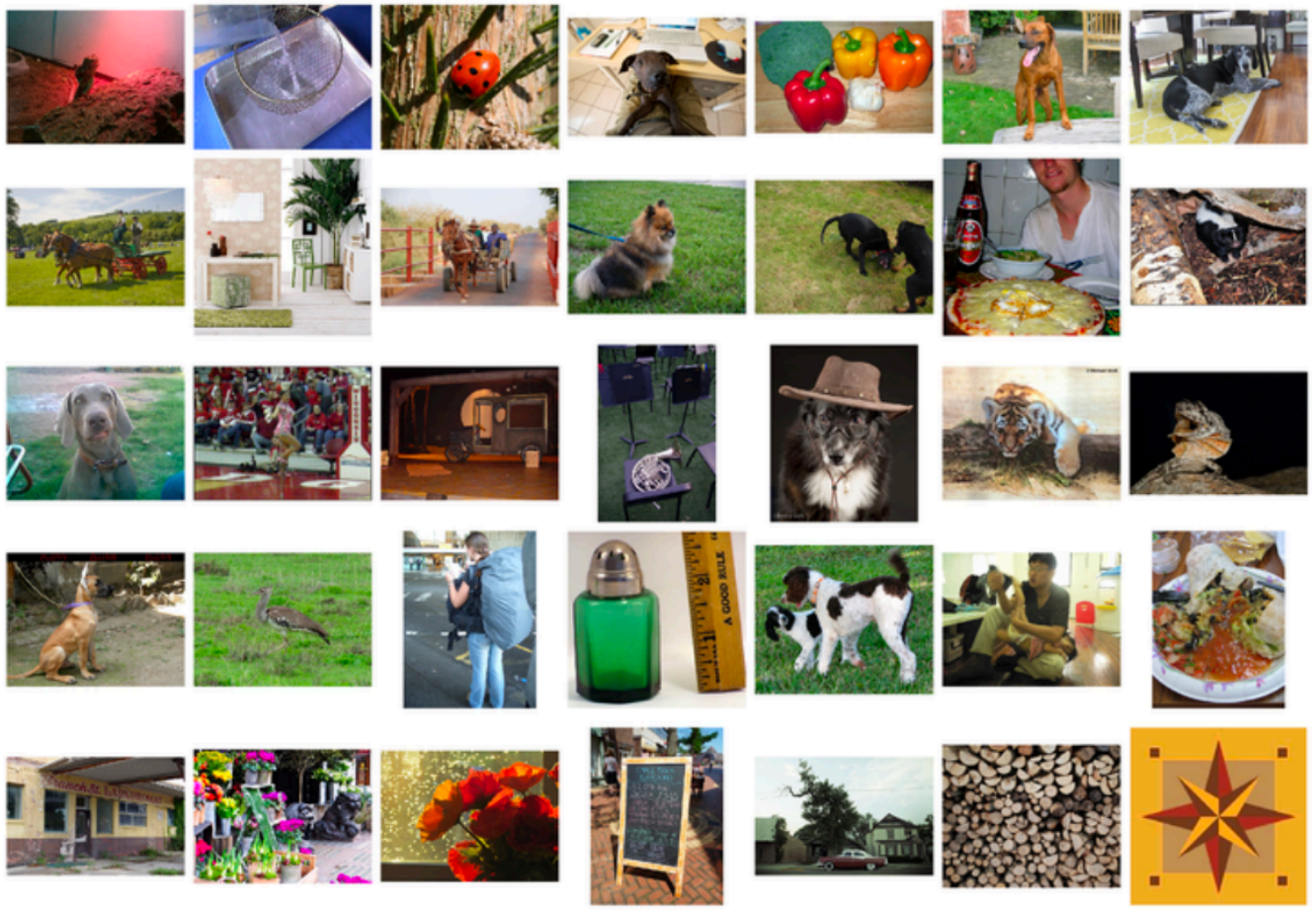
Hands-on

- We'll run **two** hands-on demos
  - Image recognition (with TensorFlow and PyTorch)
  - Text generation (with PyTorch)
- Using **NodeJS** as the client language and **pre-trained** models

```
wget https://s3.amazonaws.com/redisai/redisai-examples.tar.gz
tar -zxvf redisai-examples.tar.gz
cd redisai-examples/js_client
npm install jimp ioredis
```

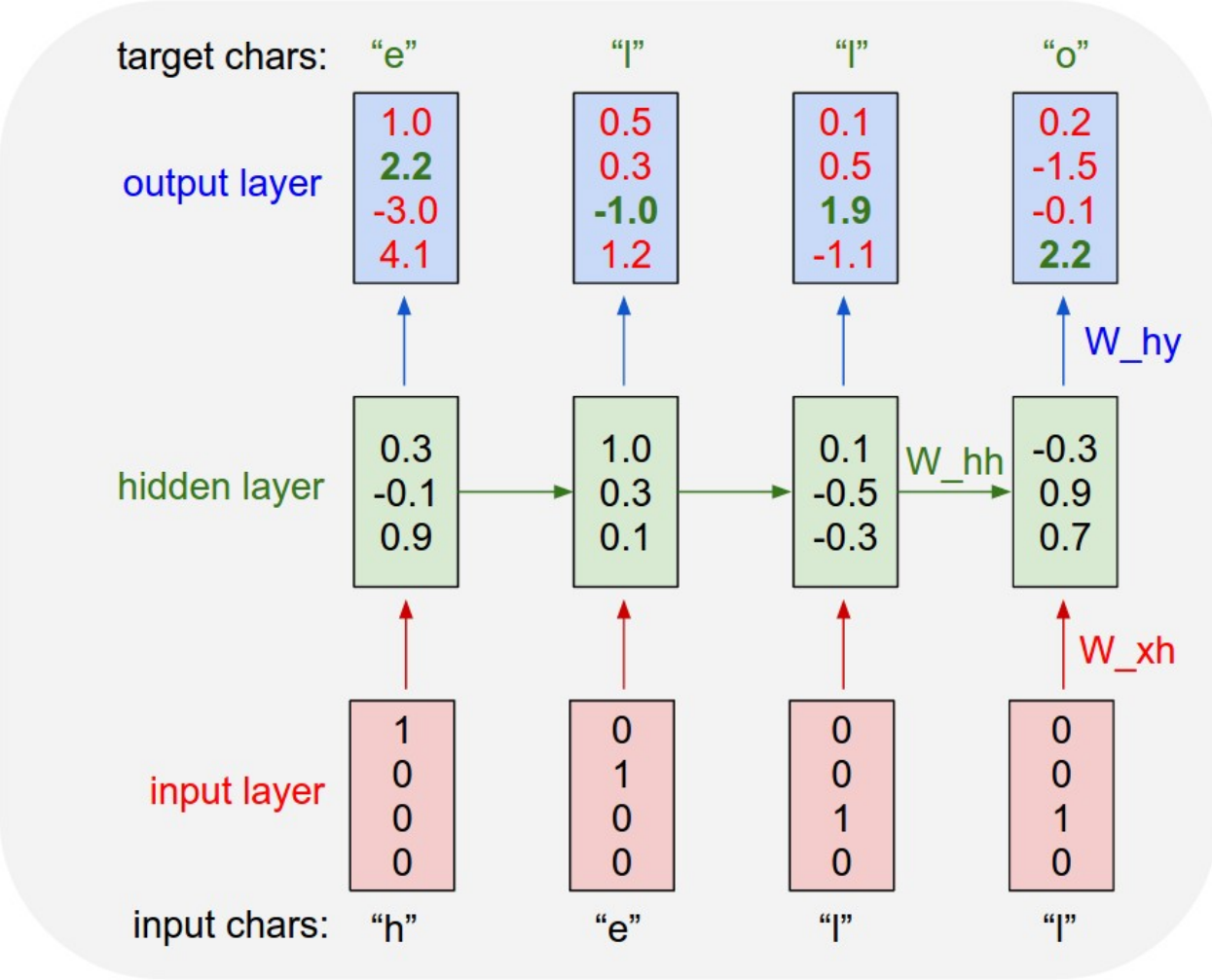
ImageNet

- Image classification task (1000 classes)
- Images 224x224



CharRNN

- Text generation starting from a seed
- No grammar, text is generated character by character



http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Acknowledgements

Acknowledgements

- **[tensor]werk**
  - Sherin Thomas, Rick Izzo, Pietro Rota
- **RedisLabs**
  - Guy Korland, Itamar Haber, Pieter Cailliau, Meir Shpilraien, Mark Nunberg, Ariel Madar

luca@tensorwerk.com

redisconf19

Thank you!

TS.RANGE key FROM_TIMESTAMP
TO_TIMESTAMP [aggregationType]
[bucketSizeSeconds]

presented by

redislabs
HOME OF REDIS

redis