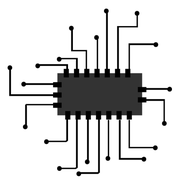
Prototype Modular Framework for Deep Learning Performance Testing

Aleksandr Drozd and Satoshi Matsuoka
Tokyo Institute of Technology

Motivation 1: emerging technologies



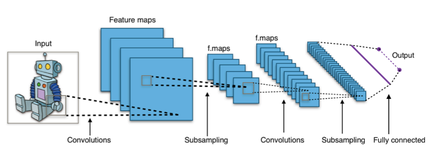
- multi- and many-core processors
- FPGAs and ASICs
- interconnects

Motivation 2: plethora of DL frameworks

Caffe, Chainer, DL4J Deeplearning4j, KERAS, Microsoft CNTK, MatConvNet, MINERVA, mxnet, Purine, TensorFlow, theano, torch

+ the whole stack of underlying libraries: from BLAS to deep learning primitives to network communication

Motivation 3: many neural net architectures



- convolutional
- recurrent
- residual
- attention-based
- with memory
- other exotic beasts

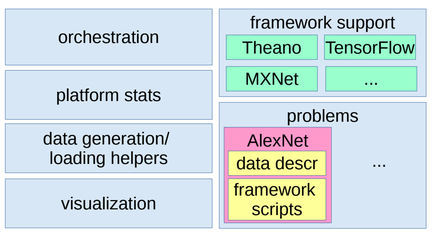
Other issues

- Vendor benchmarks are not always trustworthy
- Other factors like modeling capability are equally important to performance.

Solution: semi-automatic benchmarking

- across hardware platforms
- deep learning frameworks
- network architectures
- underlying software stacks

Implementation: modularity



- support for new frameworks/ problems can be added by corresponding plugins
- framework takes care of orchestrating execution, collecting runtime metrics and detailed platform characteristics.
- maximized code reuse and minimized chances of implementation errrors

Implementation: consistent output format

- human- and machine-readable JSON
- detailed platform information included
- basic analysis / visualization tools provided as well

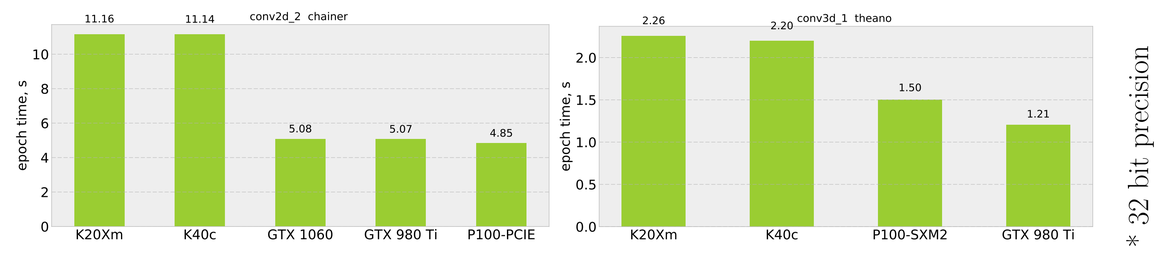
```
{
"batch_size": 8,
"framework": "Keras-2.0.2/theano_0.9.0",
"device": "Tesla P100-PCIE-16GB",
"hostname": "roma5.m.gsic.titech.ac.jp",
"linux": "Linux-3.10.0-327.18.2.el7.x86_64",
"nb_gpus": 1,
"problem": "conv2d_2",
"shape_x_train": [2048, 1, 128, 128],
"time": 37.598225240285196
}
```

simplified example of output file

Where to get:

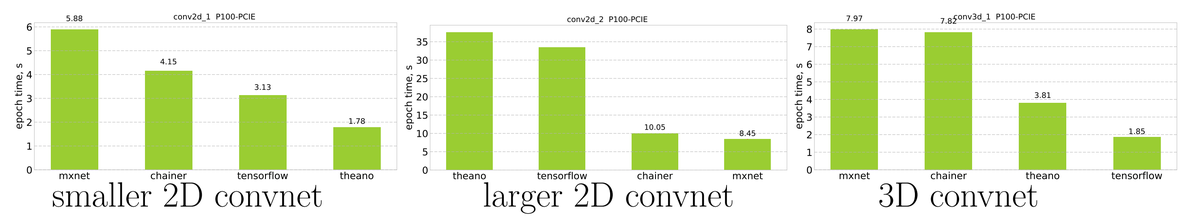
`http://blackbird.pw/performance`

Performance across devices



* 32 bit precision

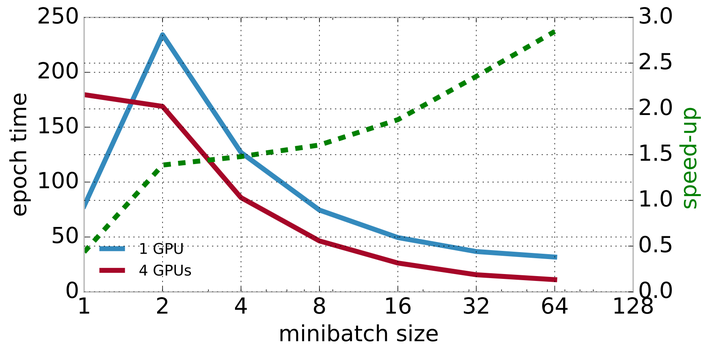
Performance across frameworks



smaller 2D convnet | larger 2D convnet | 3D convnet

Don't draw quick conclusions based on just single benchmark. For almost any framework we could find regimes where it performs better or worse than others. Rather then benchmarking frameworks against each other it seems to to be more sensible to talk about "state of implemented-ness / optimized-ness" with respect to particular kernels/topologies for each framework.

More benchmarks



- scaling
- impact of various hyper-parameters
- achieved accuracy

Conclusions

- (Relative) performance of individual frameworks vary drastically depending on network architectures. No absolute winners.
- Complicated network implementations (e.g. attention-based) tend to run with low efficiency.
- Framework performance tends to improving with each software release.
- Performance dynamic across devices is somewhat consistent.
- Already used to spot performance anomalies in TF -> submit issues -> get bugs fixed.

Future work

Currently the framework is in the early prototype stage, a lot of things have to be improved or redesigned. Particular focus points are:

- better coverage: from DeepBench primitives to popular neural net architectures.
- better support for distributed experiments, including integration with workload management systems

Acknowledgements

This work was partially supported by JST CREST Grant Numbers JPMJCR1303 and JPMJCR1687, and performed under the auspices of Real-world Big-Data Computation Open Innovation Laboratory, Japan