Orchestrating a brighter world NEC

Deep Learning with Vector Processor

Hans Peter Graf

June 21, 2018

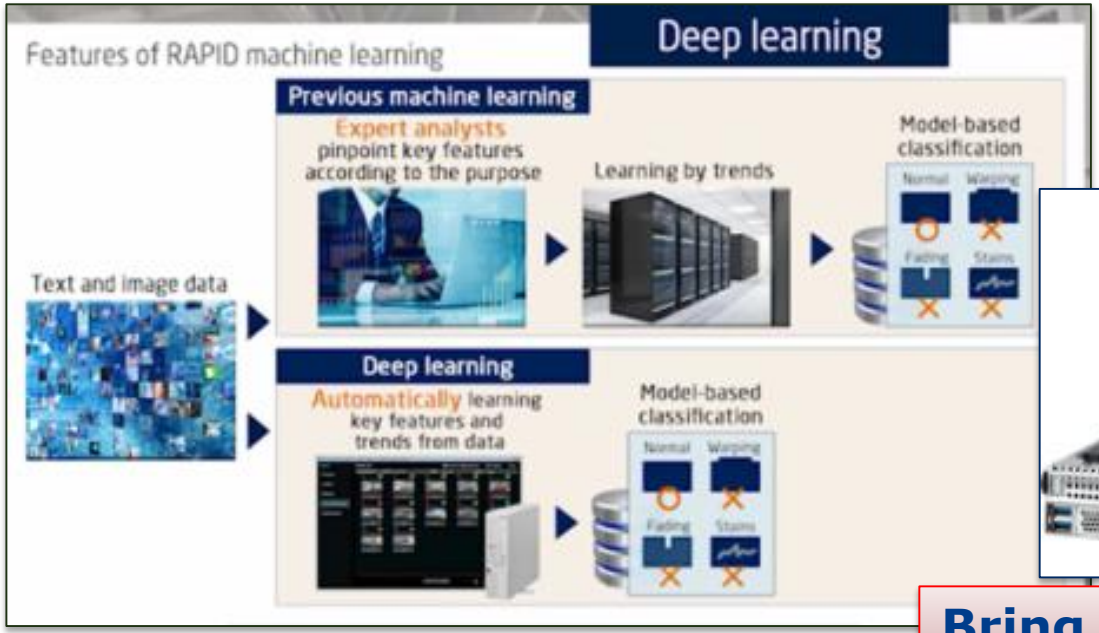
NEC Laboratories America

Relentless passion for innovation

NEC's Machine Learning Platform: RAPID

Application Templates: Inspection, Time series analysis, Surveillance, Risk Analysis

Next release: RAPID 2.2, July 2018



Fast, light weight, high precision

CPU, GPU, soon: VECTOR

Bring the algorithms to the data!

YouTube video: https://www.youtube.com/watch?v=sFfJngldFJs

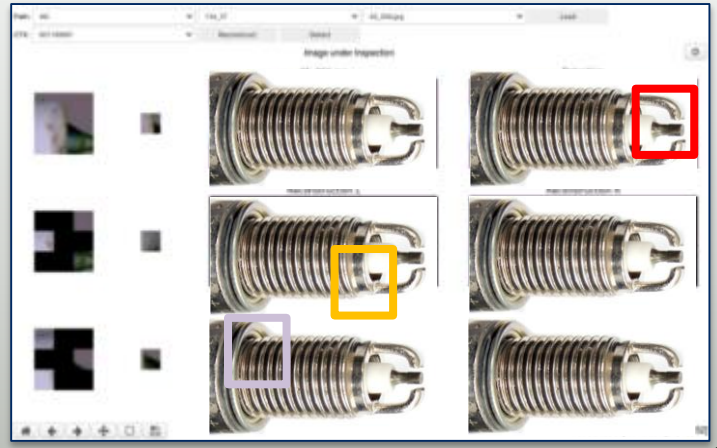
Commercial Applications

System for Unsupervised Defect Detection on Manufactured Objects

Introduced June 2017

http://jpn.nec.com/press/201706/20170621_04.html

This system can be applied for a wide range of inspection tasks for quality control of manufactured products.



System for Spoof Detection at Tokyo Stock Exchange

Introduced March 2018

https://jpn.nec.com/press/201803/20180319_02.html

System identifies trading patterns that indicate illegal activities. This reduces drastically the amount of trades that have to be checked by experts.

売数量	値段	買数量
1,000	104	
1,000	103	
2,000	102	
3,000	101	
	100	3,000
	99	50,000
	98	1,000
	97	1,000

From Deep Learning to Reasoning

Deep Learning is pattern matching. Need abstract reasoning

Approach: Structured Network Learning

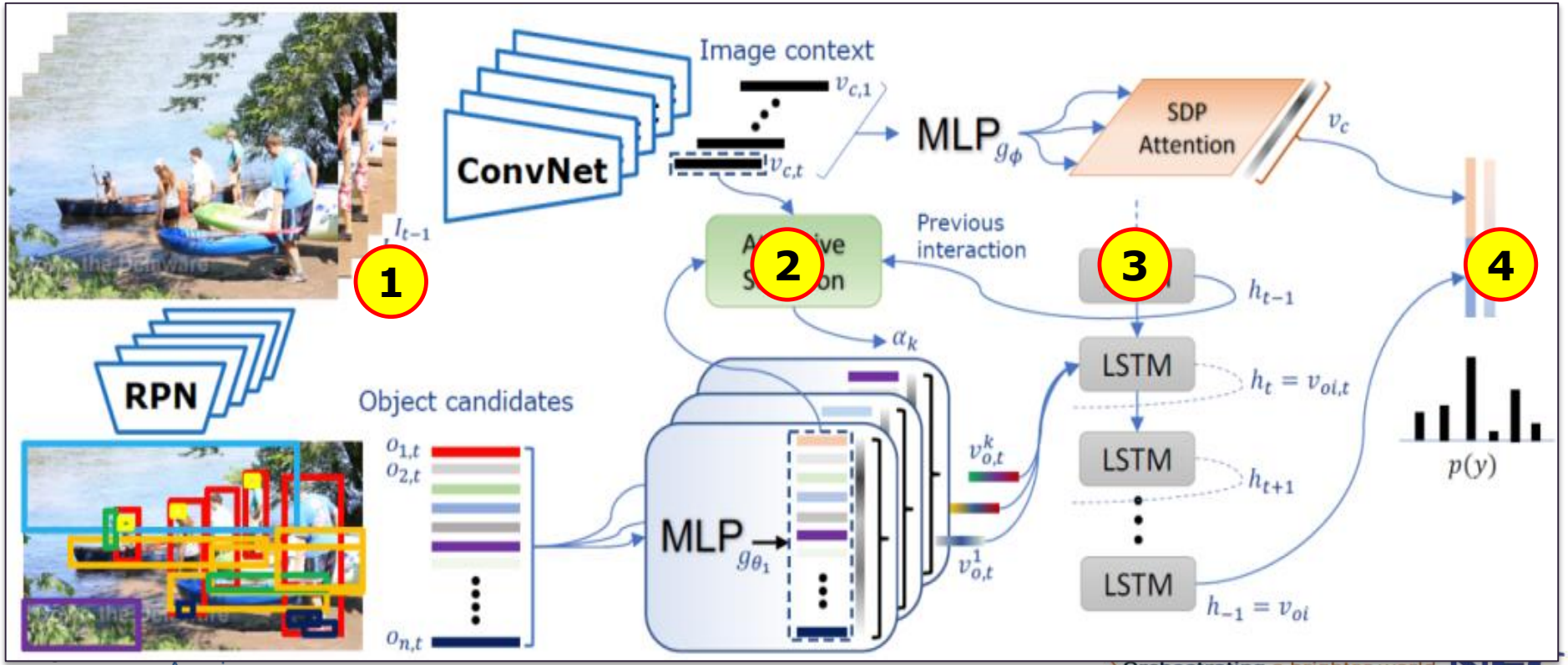
Analyze spatial and temporal relations between objects

A group of people get off of a yellow school bus with life rafts around their neck.
Relationships: [group of people, get off, yellow school bus], [group of people, with, life rafts]

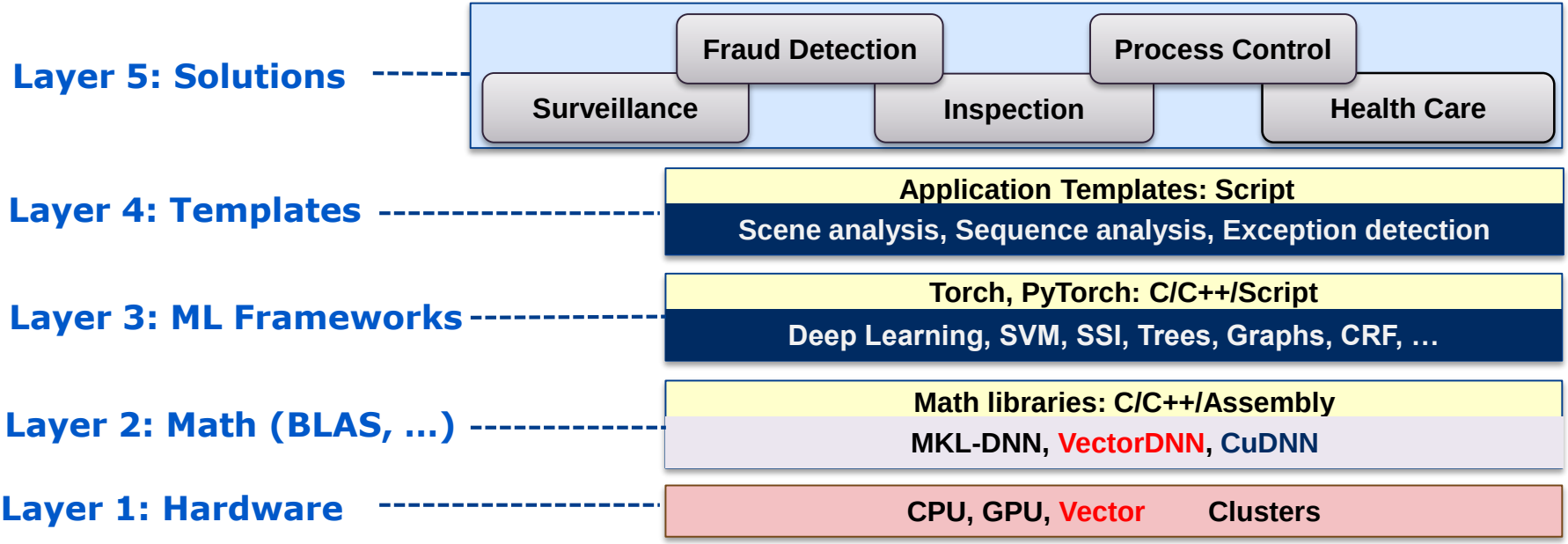
"Attend and Interact: Higher-Order Object Interactions for Video Understanding"; C. Ma; A. Kadav; I. Melvin; Z. Kira; G. AlRegib; H. Graf; CVPR 2018

Structured Network Learning

1. Object localization (CNN) → 2. Focus of attention (MLP, LSTM) →
3. Interactions of objects (spatial and temporal) → 4. Interpretation (MLP)



Software Architecture

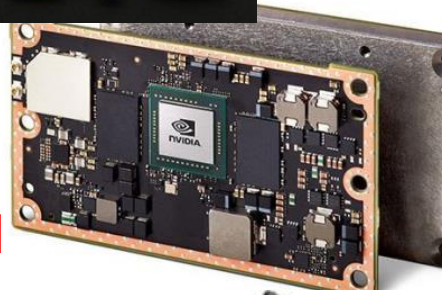


Other Critical components:

- Tools to analyze the conditions at the customer premise
- Automated evaluation of the data. Understand which Templates to use.
- Good user interface for easy deployment of solutions even by non-experts.

Running on Different Hardware Platforms

- **CPU:** Efficient parallelization
- **GPU:** Efficient multi-GPU parallelization
- **Embedded:** Cars, machining centers, …
  - **RaspberryPi**
  - **NVIDIA Jetson TX2**
- **Vector processor: Aurora with VectorDNN**
  - **Edge: Compact high-performance**
  - **Data Centers: Efficient parallelization**
- **FPGA**

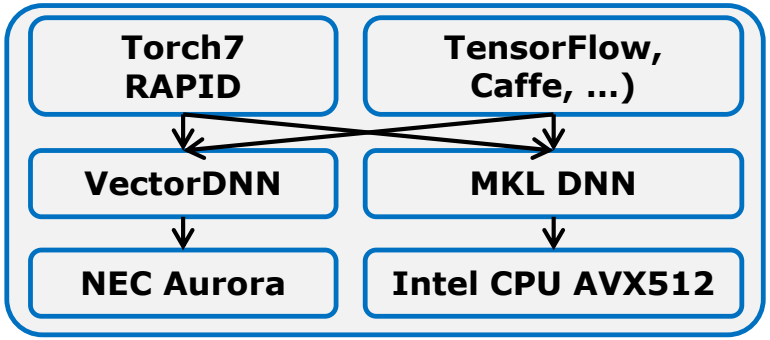


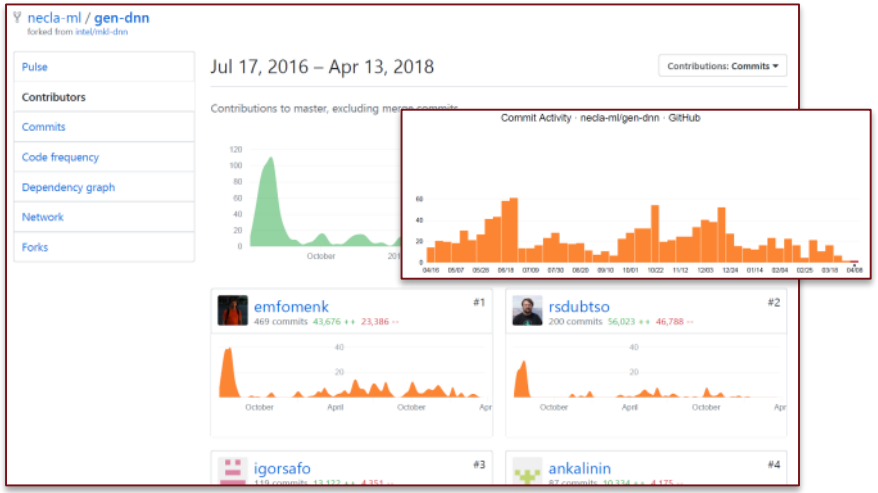
VectorDNN: API compatible with MKL-DNN

Goal: Make VectorDNN easy to integrate with many Data Analytics frameworks

Intel Library for Deep Learning is optimized for vector instructions (AVX)

Branch of MKL-DNN:
https://github.com/necla-ml/gen-dnn





Status:

- **VectorDNN is working, passes tests**
- **Torch compiles**
- **Performance optimizations are ongoing**

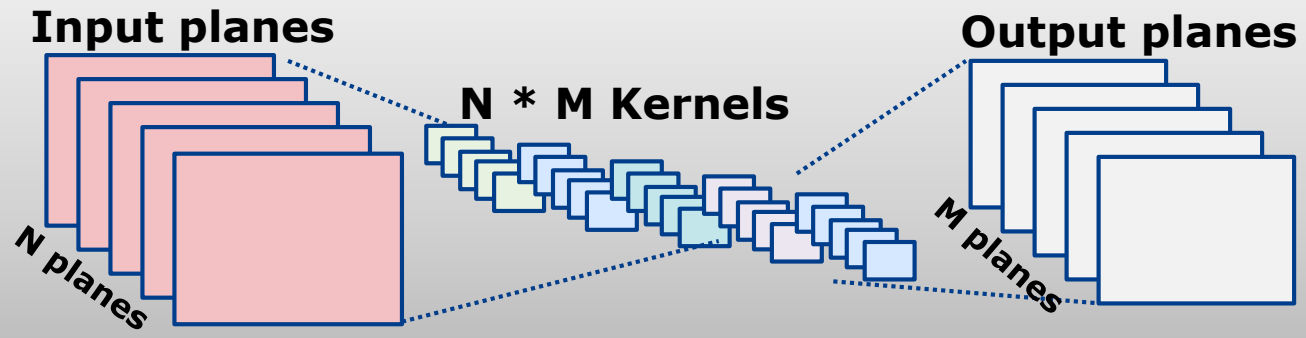
Gen-dnn activities on GitHub

NECLA also contributed bug fixes that are integrated in MKL-DNN

NEC Laboratories America
Relentless passion for innovation

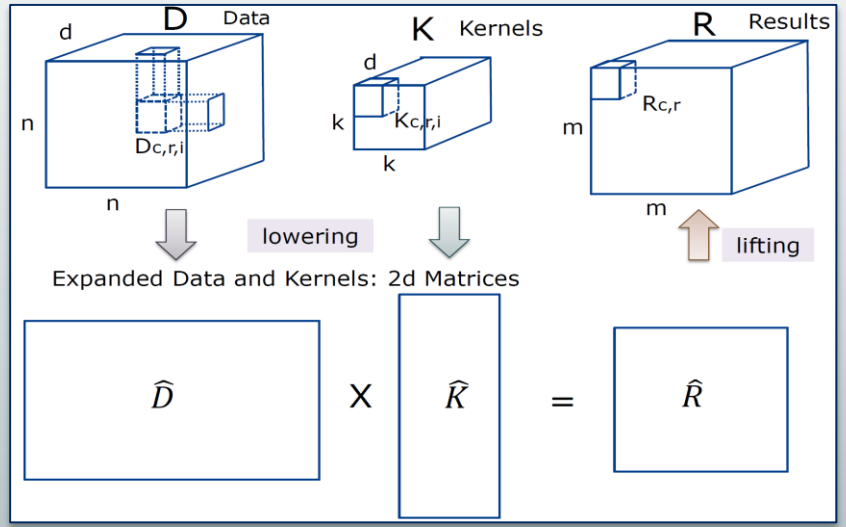
Orchestrating a brighter world NEC

Vectorization of Convolutions



Nested loops: 8 or more

Tiling: introduces more loop levels, but allows consistent implementations across all layers



Tested 25 ways of lowering and lifting

Example: im2col

- Expensive lowering, cheap lifting
- Often used; often not optimal, but usually decent performance without specific tuning. Makes use of 'gemm'

Tuning Gen-DNN convolutions

- There are many ways to speed-up operation of networks, but they become more and more specific; need to evaluate what is worth implementing.
- Speedups (this is ongoing)
  - Remove interleaving support → dense data & kernel layouts (esp. "nchw & goihw")
  - im2col techniques and loop transformations leading to gemm calls for inner loops.

Examples of features supported:

Convolution Feature	MKL-DNN	Description
Interleaving*	Yes	Specialized data layout (*) NEVER for Gen-DNN
strided	Yes	Every n'th image pixel
padding	Yes	Pad image border with zeros
groups	Yes	Kernel channels != image channels
dilation	Yes	Kernel applied to dilated image region
minibatch	Yes	Convolve multiple separate images at once
bias	Yes	Learn bias for convolution layer output
merging	Yes	Combine convolution with nonlinearity (Relu)
Forward	Yes	Forward with bias integrated
Backward	Yes	Backward wrt. data, weights, or both.

Comparisons

Compare with Skylake: Intel Xeon Gold 6126 @ 2.6 GHz (AVX512)

- Timings of im2col + gemm implementation (no JIT). Timings include only convolutional layers from each network forward and backward with respect to weights (no activation layers)
- Minibatch sizes: 64 for Resnet-50, 64 for VGG 11, 256 for Alexnet.
- Input sizes: 3x224x224 for Resnet-50, 3x100x100 VGG 11, 3x227x227 Alexnet.

Network	Skylake1	Skylake8	Aurora1	Aurora8
Alexnet	5224	1399	2969	486
VGG 11	1752	542	895	260
Resnet 50	7862	1641	4797	1060

Time in ms

[Skylake1] Skylake OMP_NUM_THREADS=1
[Skylake8] Skylake OMP_NUM_THREADS=8, pragma omp
[aurora1] Aurora OMP_NUM_THREADS=1, blas_sequential
[aurora8] Aurora OMP_NUM_THREADS=8, blas_openmp, pragma omp

OpenMP for Gen-DNN

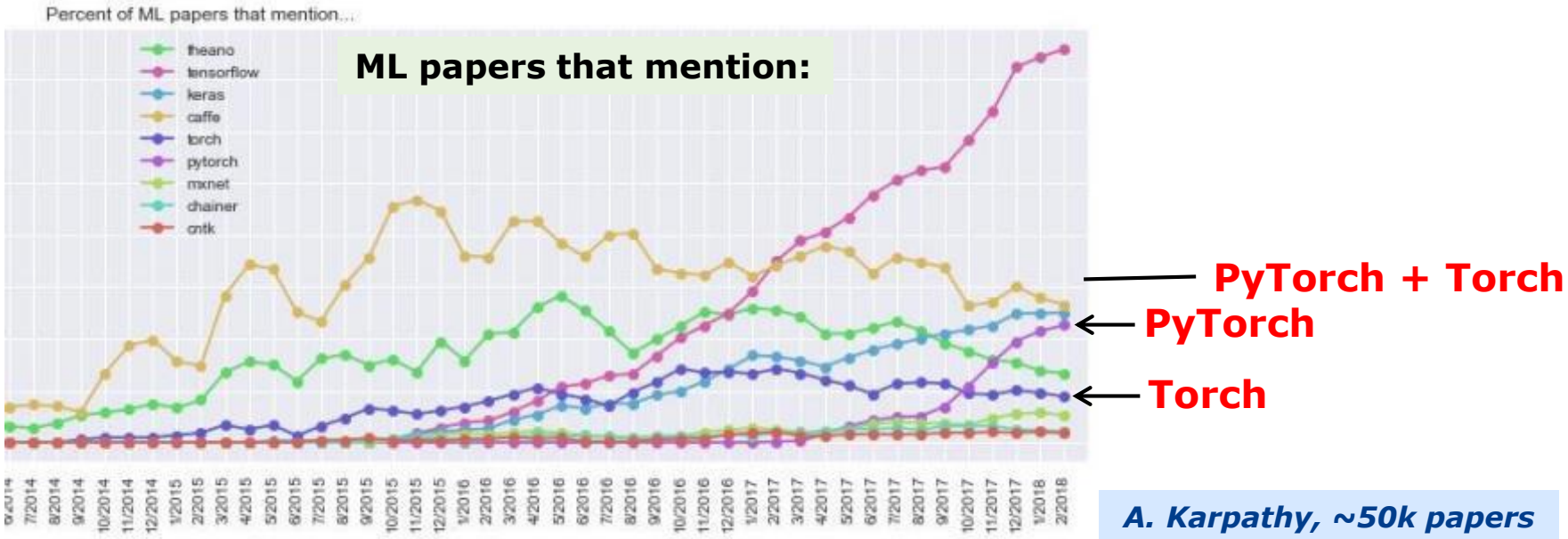
BLAS	Gen-DNN #pragma omp	Avg gigaflops	Max gigaflops
blas_sequential	disabled	225.2	474.6
blas_openmp	disabled	192.0	1334.9
blas_openmp	enabled	1062.5	2859.7

Results over Gen-DNN's "benchdnn" test suite of 204 convolutions

- Operation count is with respect to reference convolution (a faster implementation is used)

Implementation of Torch

- **Without vectorization, Torch and nn packages pass all tests. Optim, sys, paths also available. Uses compile-time Lua function bindings from C, instead of FFI**
- **With vectorization: Getting there**
  - **Torch passes all tests except one NaN problem**
  - **NN library has one test that segmentation faults (as of this week)**



ML Frameworks

- **Torch is one of the most widely used Open Source platforms; used by thousands of developers in universities and industry (Facebook, Twitter, …).**
- **Lots of great tools; preferred by serious developers**

https://github.com/zer0n/deepframeworks

	Caffe	CNTK Microsoft	TensorFlow Google	Theano	Torch 7 NEC
Modeling capability	★★★	★★	★★★★	★★★★	★★★★★
Interfaces	★★★	★★	★★★★	★★★★	★★★★
Model Deployment	★★★★★	★★★★	★★★★	★★★	★★★
Performance single GPU			★★★★★	★★★	★★★★★
Architecture	★★★	TBD	★★★★★	★★★	★★★★★
Ecosystem	C++	C++	C++, Python	Python	Lua, Python

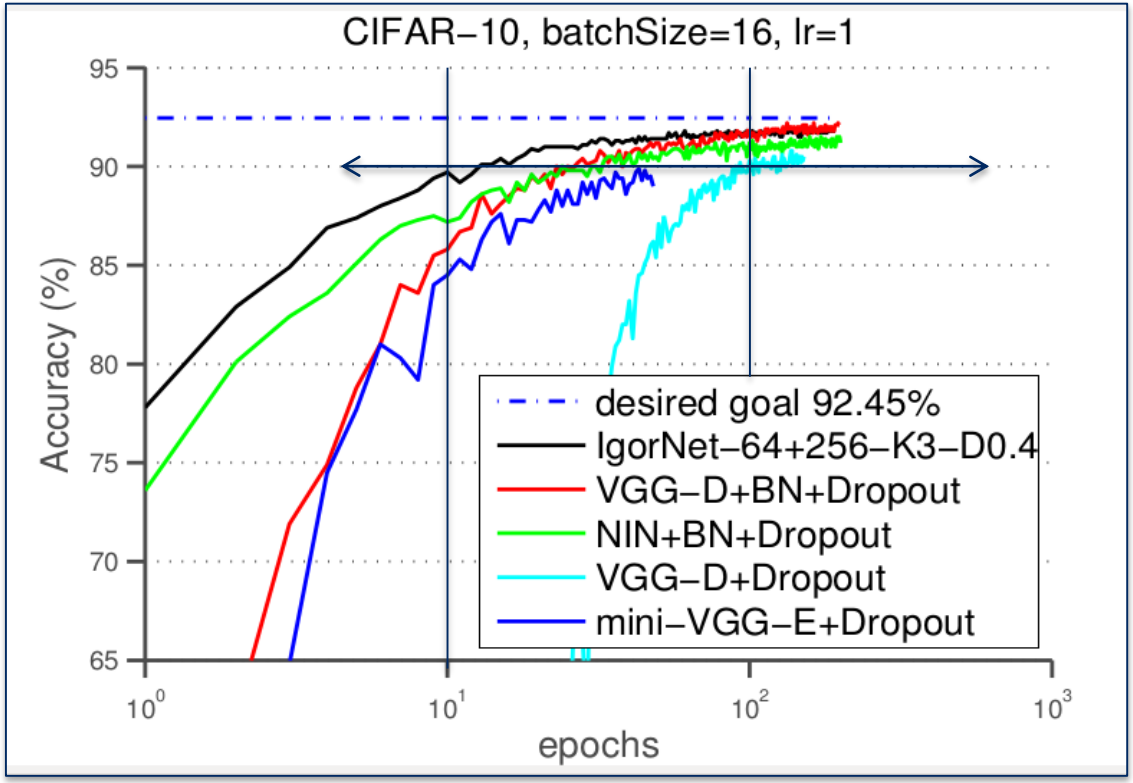
Benchmarking Deep Learning Networks

It is more than raw speed: Which network learns fastest?

Example of training with various networks:
Number of epochs to get to 90% accuracy

Compute time varies 7x to 17x **among various** networks that have best accuracies, depending on what features are activated

Key: The most efficient networks must run at high speed



NEC Laboratories America
Relentless passion for innovation
Orchestrating a brighter world NEC

Aurora: Conclusion

Deep Learning:

- Vector architecture is flexible and well suited for convolution layers as well as for the other types of layers (good SIMD problem)
- Any of the fastest algorithms can be implemented efficiently on the vector architecture (Winograd, Cook-Toom, …).
- A high efficiency of >75% or higher is achievable for most cases. This may require some tuning (JIT).
- Without JIT (im2col + gemm) efficiency more typically 30% - 70%.

Data analytics is more than deep learning → key: Flexibility

- Large memory and high data bandwidth makes vector processor attractive for large scale data analytics.
- Can integrate a wide range of algorithms with deep learning. Extensive library to support such functions.
  - e.g. collaborative filtering: Sparse matrix factorization, SVD.

Appendix

Data Analytics Eco System: PyTorch, ONNX

Dominating open source frame works.

PYTORCH

PyTorch is most popular open source platform now for development.
But for commercial products often Lua is preferred! We can't be stuck with Python!
Main differentiator of RAPID is the performance and flexibility

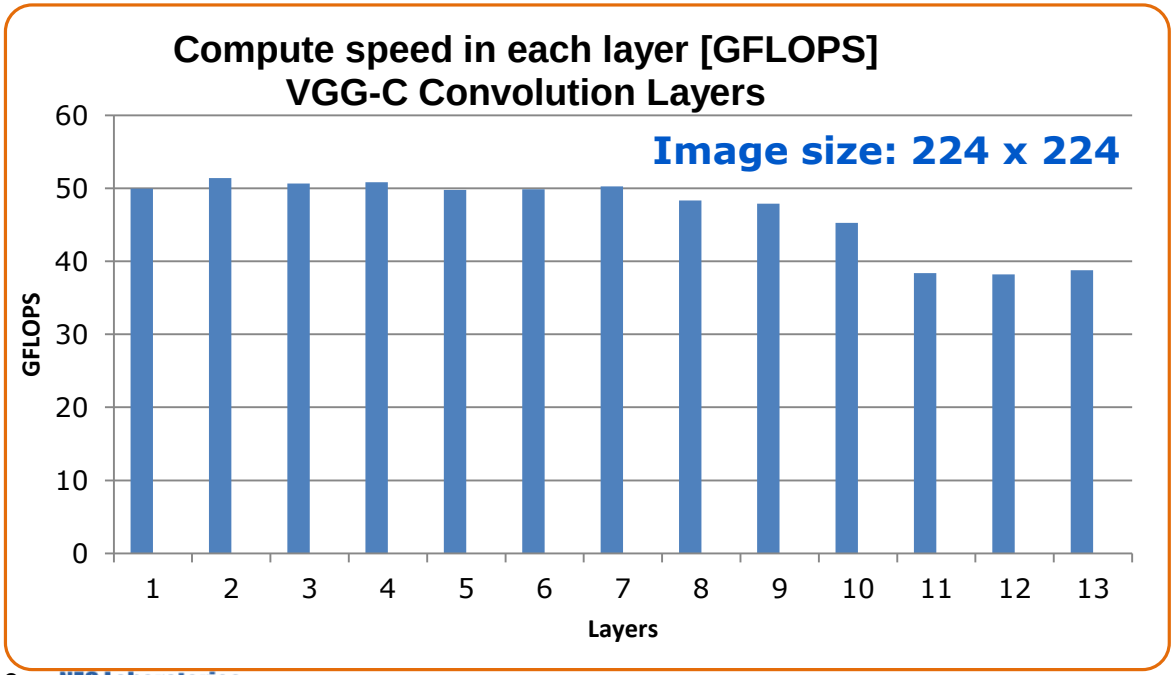
ONNX: Open Neural Network Exchange Format

Can more easily move models between state-of-the-art tools and choose the combination that is best.

PyTorch ONNX exporter:
***trace-based* exporter**, executes model once and exports the operators which were run. If model is dynamic, the export won't be accurate. Examine the model trace and make sure the traced operators look reasonable.

Tests of different ways of unrolling

We tested over 25 way of unrolling the nested loops of convolutions.
Best way of unrolling differs for SX and AVX
Obtain consistently very good performance: here 76.5% efficiency on SX-ACE core (64 GFLOPS max)



- Large networks achieve very high efficiency across all layers.
- Tiled algorithms make also smaller networks efficient.