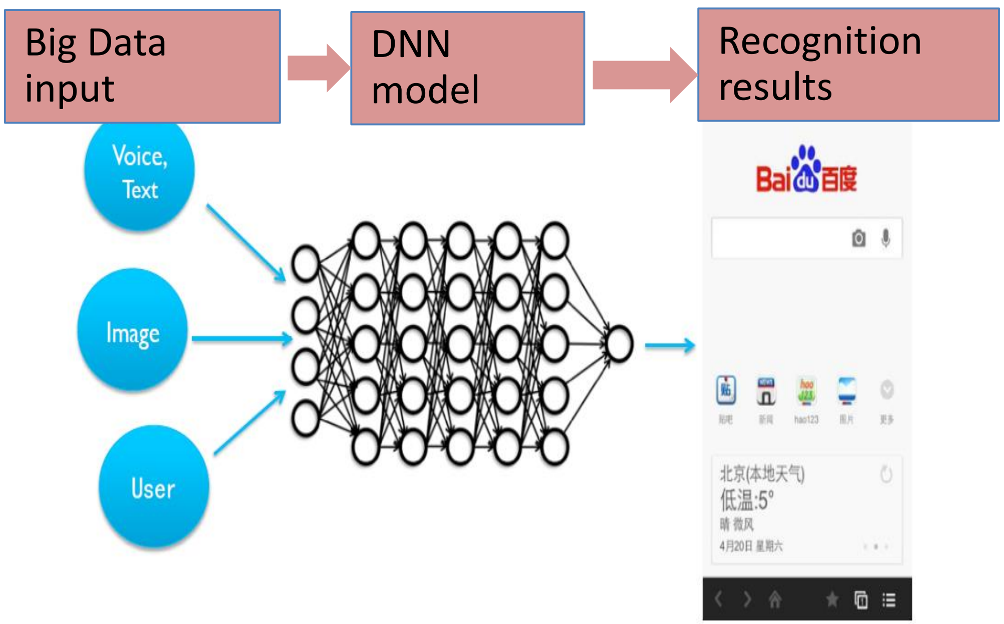
OpenCL caffe: Accelerating and enabling a cross platform machine learning framework

Junli Gu gujunli@gmail.com
Yibing Liu (Tsinghua University)
Yuan Gao
Maohua Zhu (UCSB)
Presented by Hugh Perkins (ASAPP)

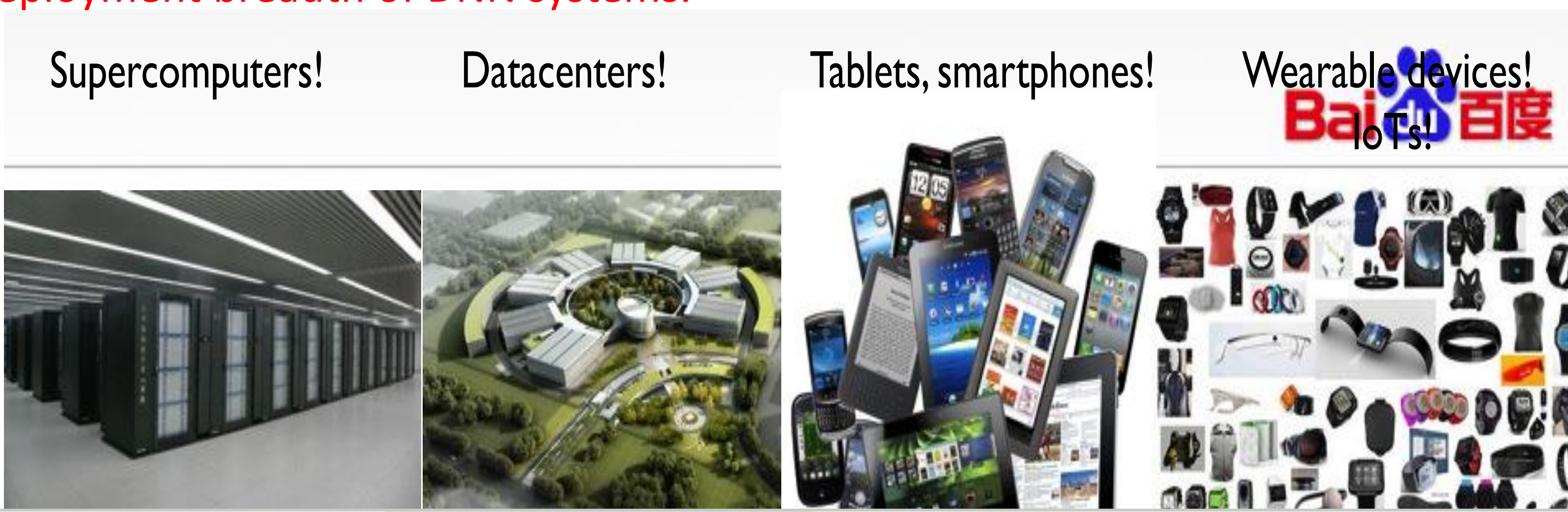
Deep learning brings challenges to system design

- Deep Learning: DNN model + Big Data
  - **Complex model**: millions to billions of parameters
  - **Big Data input**: OCR: 100M, Speech: 10B, CTR: 100B
- System is the final enabler
  - **Model training:** takes weeks on CPU + GPU clusters
  - **Model deployment**: trained model deployed for various application scenarios



Opportunities for OpenCL: cross platform DNN deployment

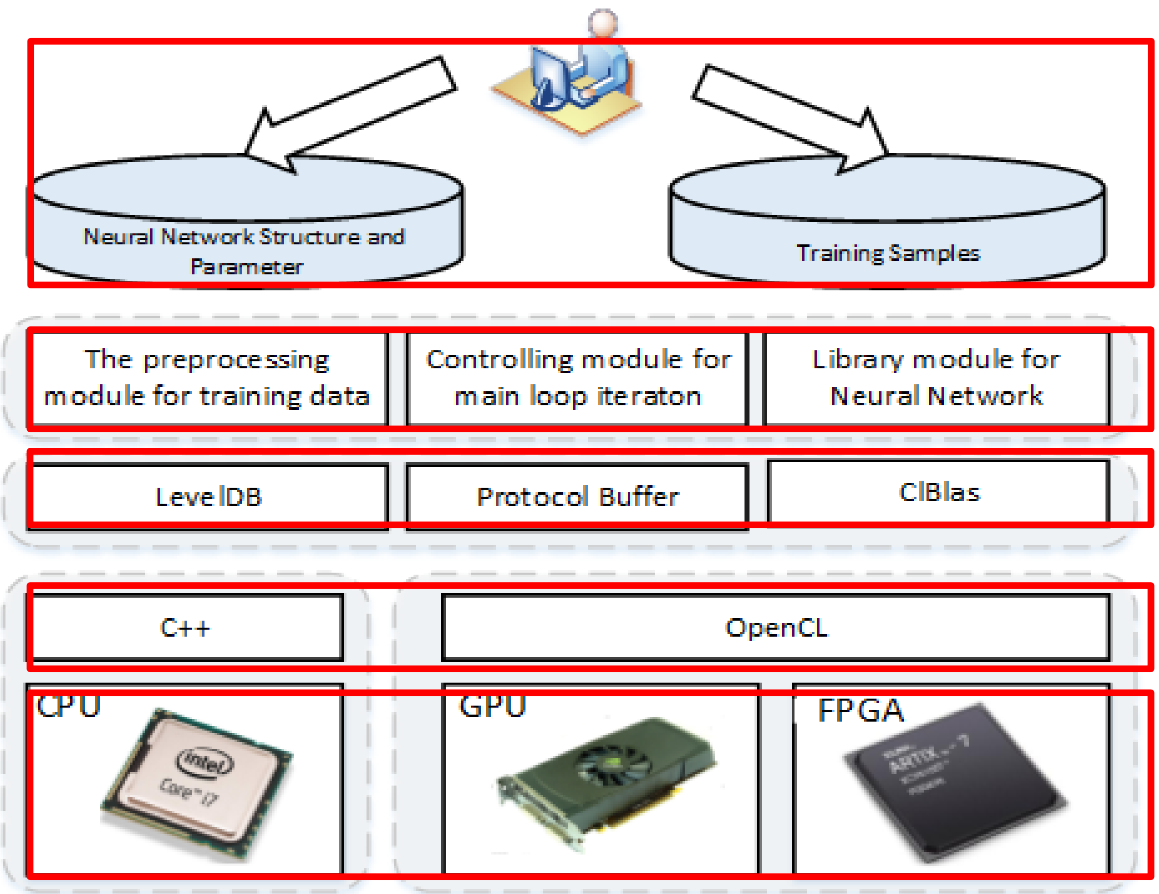
- **Current trend: DNN will be everywhere.**
- Cross platform compatibility is becoming a challenge for internet giants.
- However most DNN frameworks are based on CUDA: closed format, limiting the deployment breadth of DNN systems.



Functionality	**Offline DNN training**	**Deploy DNN on cloud**	**Deploy mobile DNN apps**	**Deploy on Wearable and IoTs**
H/W systems	**CPU + GPU cluster**	**CPU clusters or CPU+GPU clusters**	**ARM/GPU/SOC**	**ARM/Soc/FPGA**
Scale	**Small scale (hundreds)**	**100k-1M**	**700M**	**billions**

The goal of OpenCL caffe

- **Hierarchical framework** that serves as **machine learning OS**



- **Software level**
  - machine learning SDK and APIs
  - CNN, MLP, RNN, LSTM etc.
- **Hardware level**
  - hardware resources allocation and utilizations
  - optimized DNN and math libraries
- **Workload partition**
  - CPU: data processing and main loop iteration
  - GPU: major DNN kernel computation

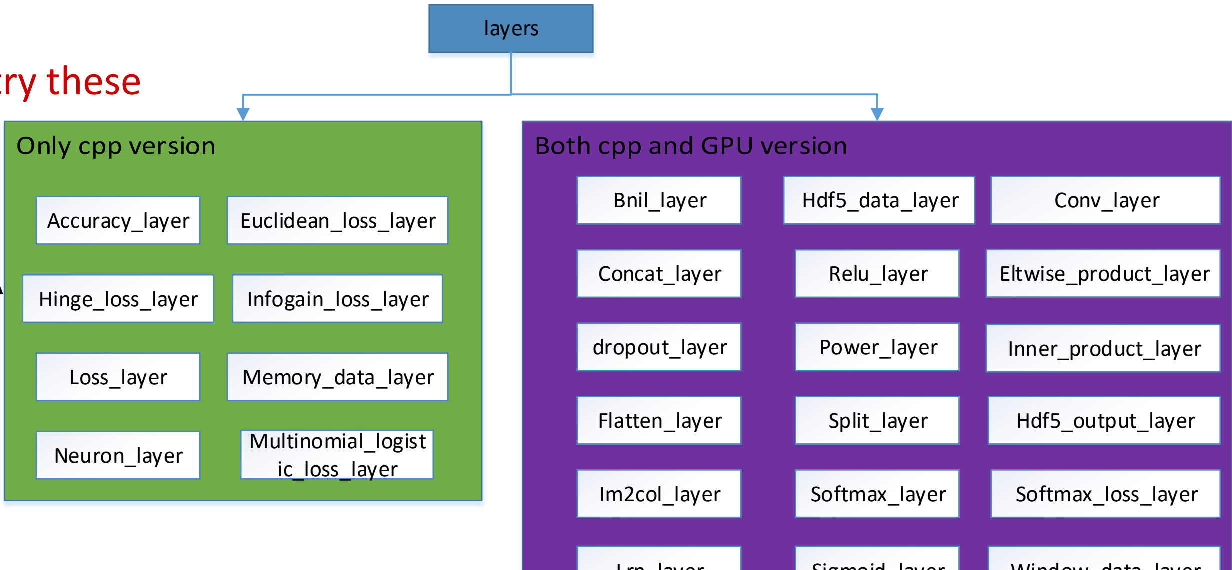
Original CUDA caffe from UC. Berkeley: https://github.com/BVLC/caffe

Two phase strategies

- Phase one: OpenCL backend porting and analysis
  - It is not a straightforward engineering porting, algorithm convergence might be destroyed
  - Re-architecture due to key difference between CUDA and OpenCL
- Phase two: OpenCL caffe performance optimizations
  - Given the algorithm correctness, improve the performance
  - Current BLAS libraries are not optimized for DNN computing, why and how to improve without modifying BLAS?

OpenCL Caffe Framework

- Hybrid CPU and GPU implementation
  - Each layer
- CAFFE is most popular in industry these days
  - Complexity:
  - ~70k lines of code
  - Originally designed for C++ & CUDA
- Seamless switch between CPU/GPU



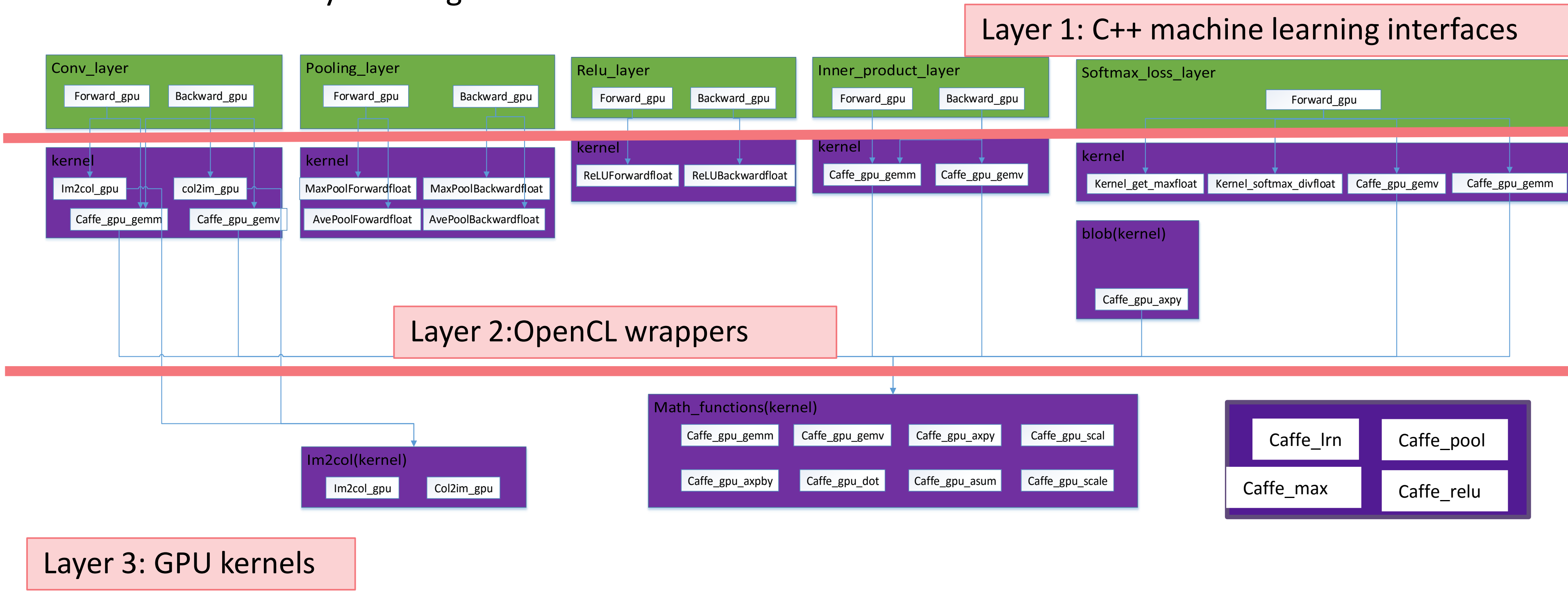
Prototype

Training

Deployment

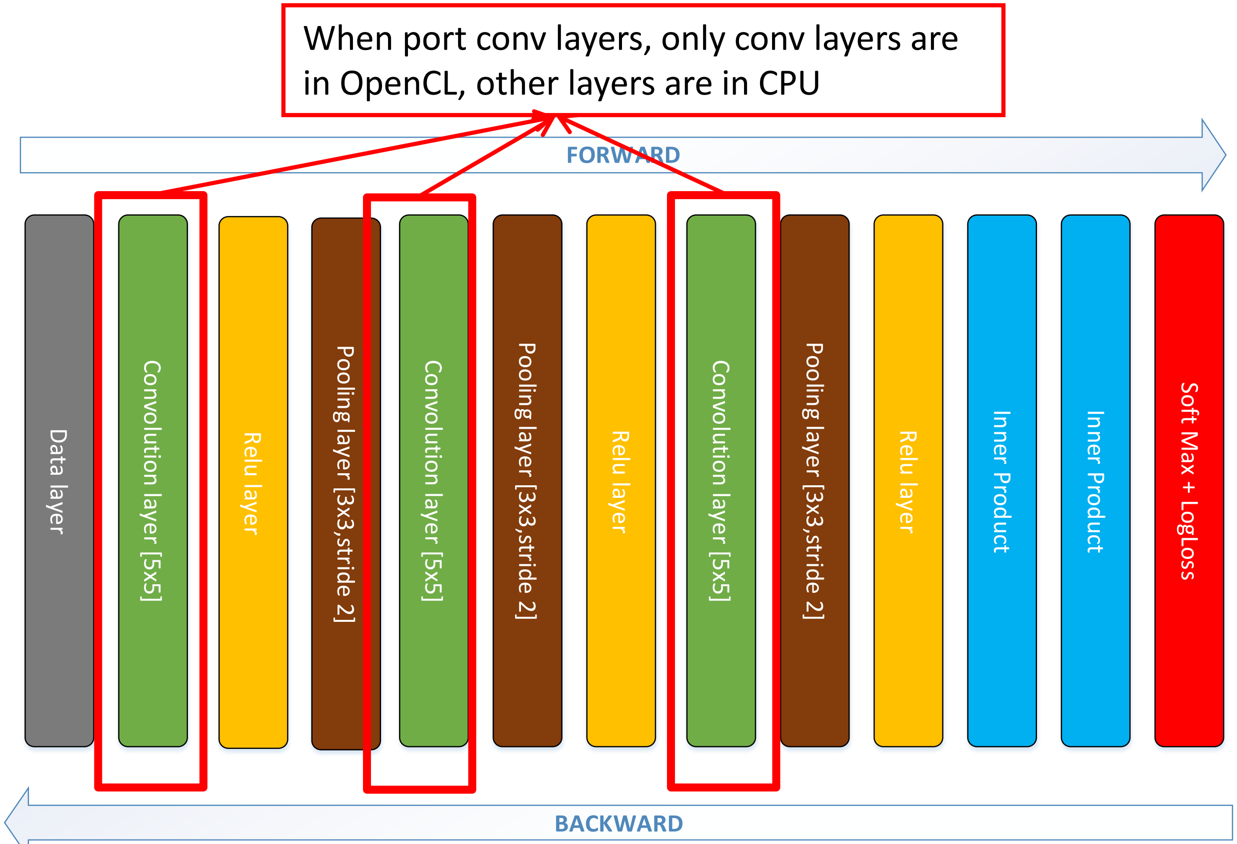
OpenCL porting challenges and re-architecturing

- Memory layout & data coherence
  - mutable data structures
  - Optimal buffer allocation for each layer
- Hide data transfer to the underlying hardware layers
- Added extra OpenCL wrapper layer compared to CUDA
  - Hide messy clSetArg etc stuff



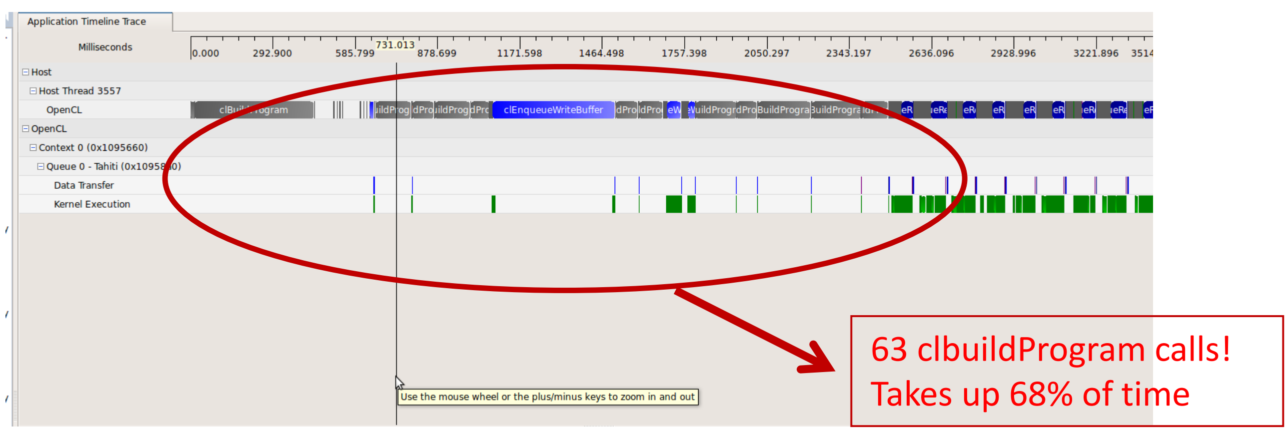
Layer wise porting to guarantee correctness

- DNN is a deep layered structure, algorithm convergence is fragile. Gradient check is well known challenge.
  - Local correctness: unit test
  - Global correctness: comparing the convergence curves with CPU/CUDA baseline



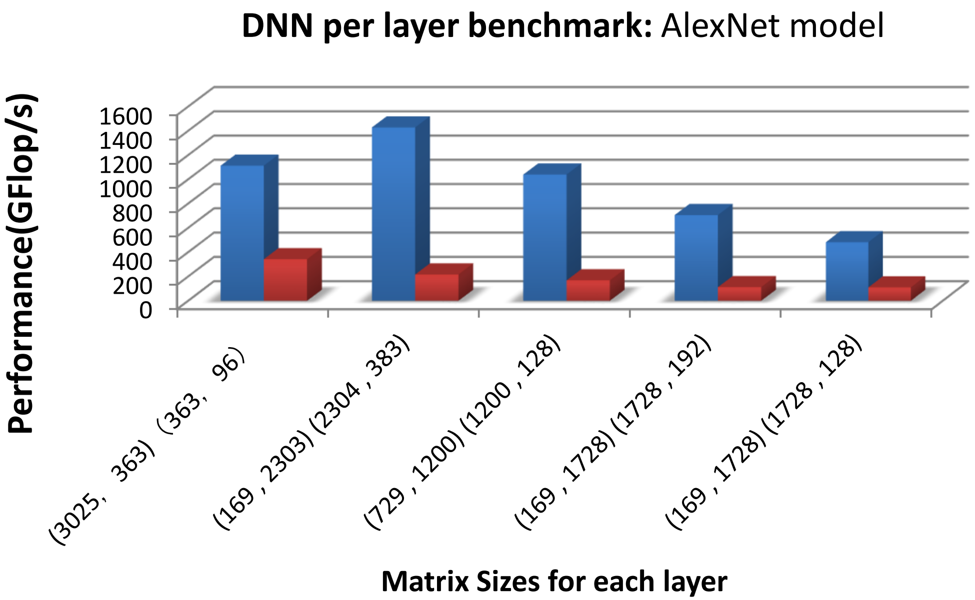
OpenCL backend bottleneck analysis

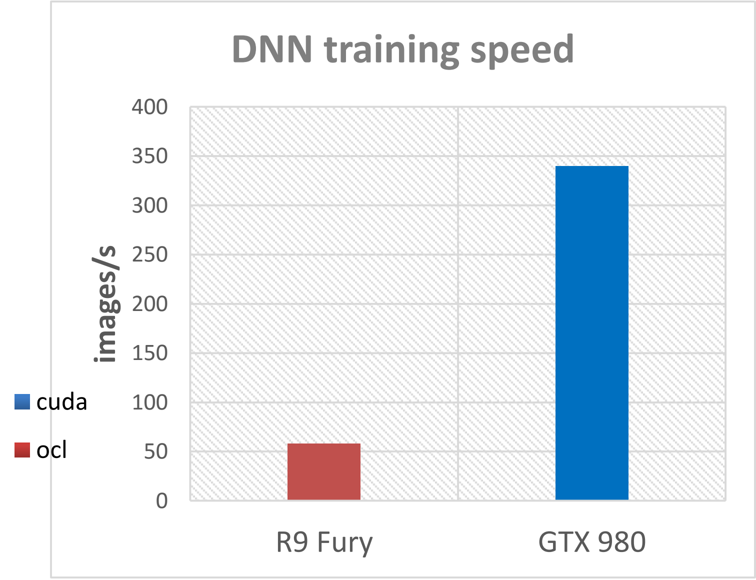
- OpenCL's online compilation frequently calls clBuildProgram
  - Too many DNN kernels to create!
- DNN falls into BLAS' poor performance area
  - Irregular tall and skinny matrix sizes from different layers
  - Bottleneck exists for all BLAS implementations, cuBLAS, clBLAS etc.
  - clBLAS is 3-5x slower than cuBLAS, the biggest performance gap to catch up



OpenCL backend bottleneck analysis

- OpenCL's online compilation frequently calls clBuildProgram
  - Too many DNN kernels to create!
- DNN falls into BLAS' poor performance area
  - Irregular tall and skinny matrix sizes from different layers
  - Bottleneck exists for all BLAS implementations, cuBLAS, clBLAS etc.
  - clBLAS is 3-5x slower than cuBLAS, the biggest performance gap to catch up





AMD R9 Fury vs. GTX980

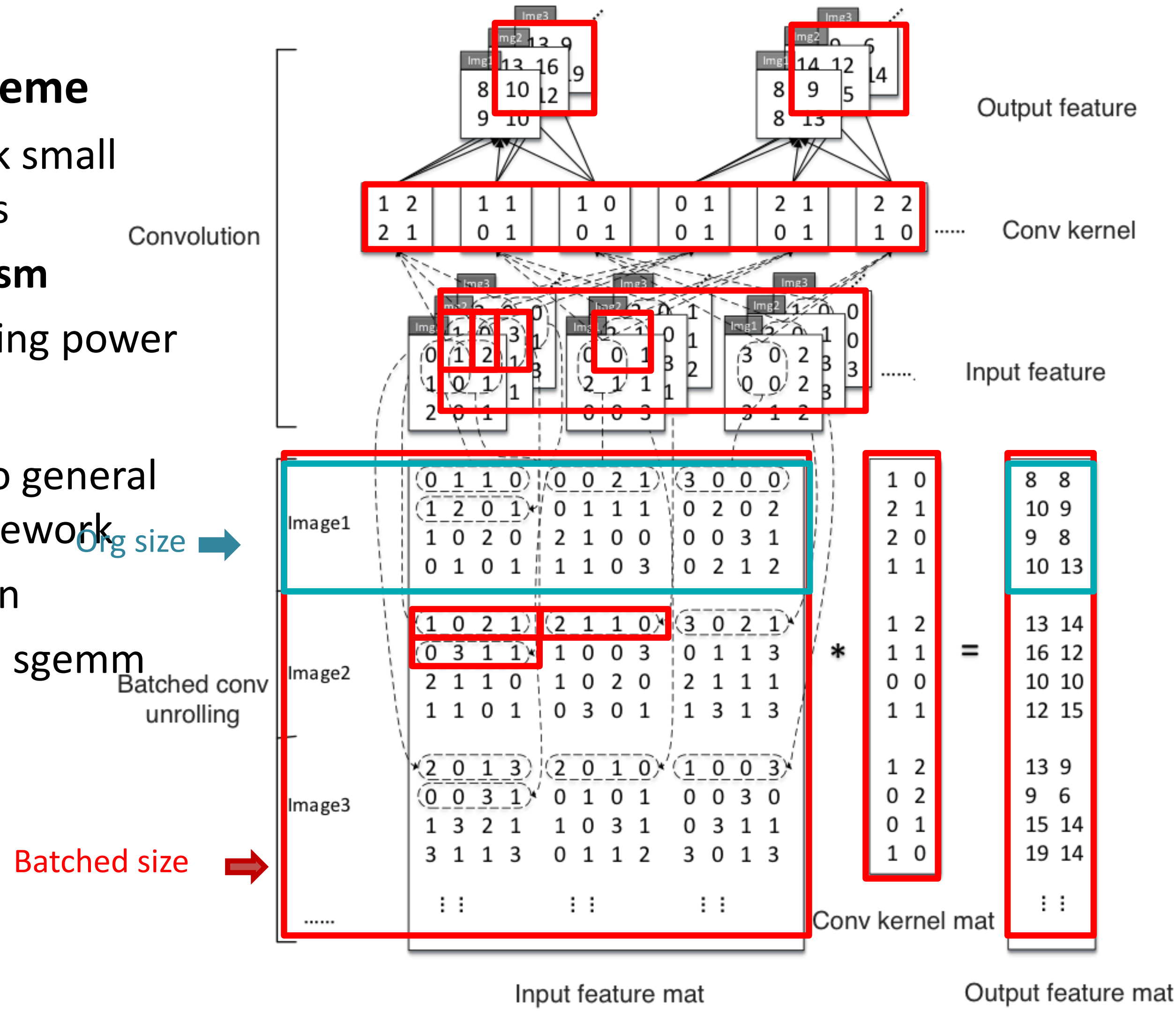
1. peak performance 7.2 vs. 4.6 TFLOPS
2. OpenCL caffe is 6x slower than cuda caffe

OpenCL caffe performance optimizations

- **Avoid OpenCL online compilation overheads**
  - Precompile and save the kernels
  - Works if hardware does not change
- **Boost data parallelism**
  - Batched manner data layout transformation
  - To bring DNN data size to better performance areas
- **Boost task parallelism**
  - Multiple command queues
  - Increase concurrent tasks

Batched data layout transformation optimization

- Batched **data layout scheme**
  - Design pipeline to pack small matrix into bigger ones
  - Increase **data parallelism**
  - Release GPU's computing power
- Notes
  - Optimization applies to general machine learning framework
  - When integrated within sgemm, called batched sgemm



Batched data layout transformation optimization

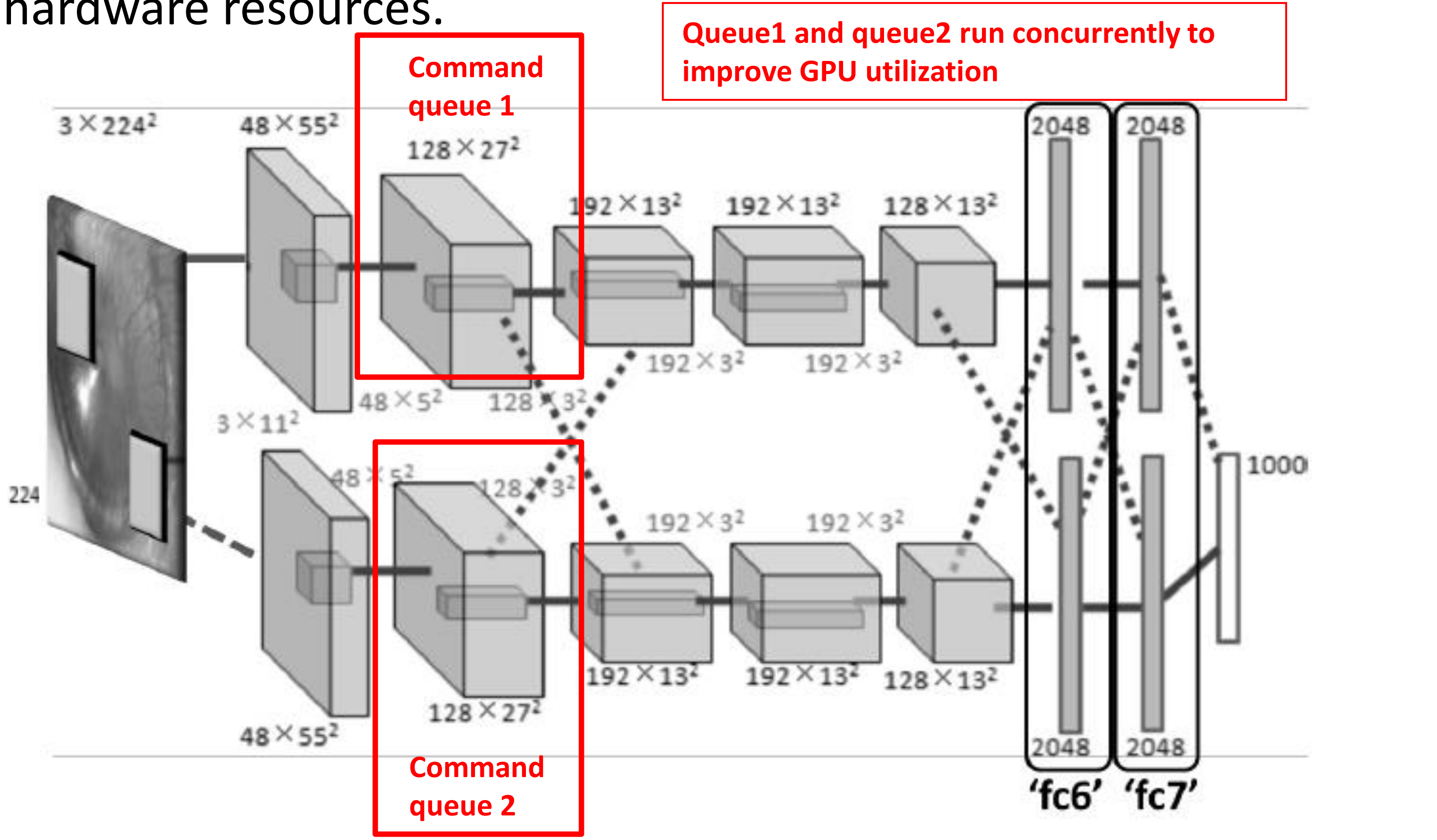
- **Batched transformation significantly unrolls the matrix size**
  - Bigger matrix, more regular
  - M, N,K can be aligned with 4/8/16/32 (BLAS preferred sizes)
  - Forward propogation, M scaled up; backward propogation, N,K scaled up (algorithm limitations)
- **Optimal batched number**
  - depending on H/W properties and input data size
  - 16 or 32 on AMD GPUs for ImageNet data set

Layers	Original M, N, K	Unrolled M', N', K'	speedup
conv1	3025, 96, 363	48400, 96, 363	11
conv2	729, 128, 1200	11664, 128, 1200	12
conv3	169, 384, 2034	2704, 384, 2034	10
conv4	169, 192, 1728	2704, 192, 1728	9
conv5	169, 128, 1728	2704, 128, 1728	16

This is matrix size for forward propagation

Boost task parallelism

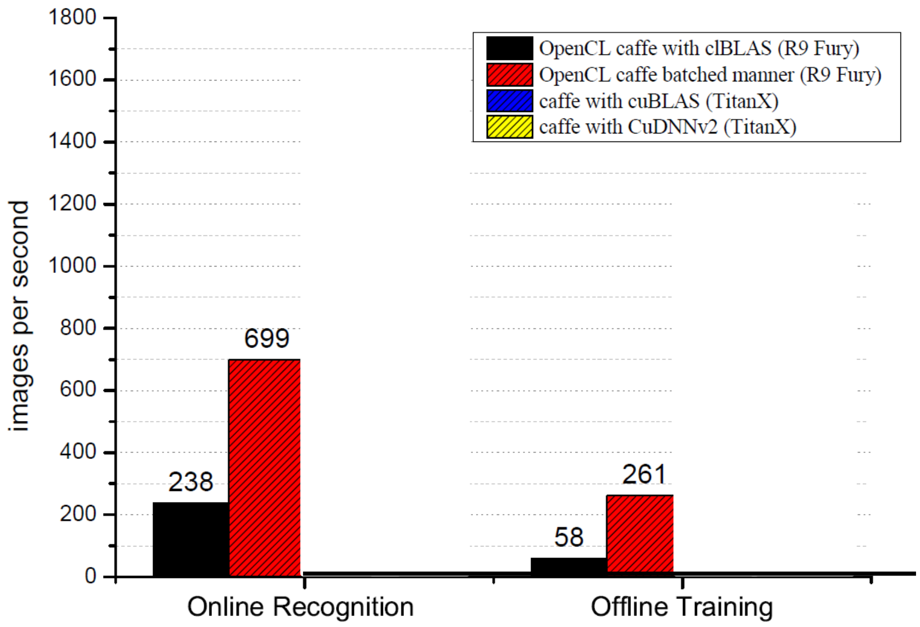
- The nature of workload imbalance among DNN layers
- Luckily, we can make use of **model parallelism**
- Performance improvement depends on layer structure, data size and hardware resources.



Performance evaluation

– OpenCL batched vs clBLAS
  - 4.5x speedup without modifying clBLAS
– OpenCL vs CUDA caffe (apple to apple)
  - Similar performance
– OpenCL vs cuDNN v2
  - 2x gap
  - Potential to catch with low

-level hardware optimization



Conclusions

- OpenCL caffe
  - To enable a cross platform DNN framework
- Optimize towards competitive performance
  - Data parallelism: batched manner data layout transformation
  - Task parallelism: make use of model parallelsim
  - 4.5x speedup on top of clBLAS library
- Existing challenges of OpenCL in cross-platform
  - Differences of various hardware manufacture extensions
  - Queueing efficiency, command queue synchronization overheads, runtime efficiency
  - Low level hardware optimizaiton tool chain for highly optimized machine learning libraries

OpenCL Caffe is at: https://github.com/gujunli/OpenCL-caffe