A Systematic Methodology for Analysis of Deep Learning Hardware and Software Platforms

Yu Emma Wang

Gu-Yeon Wei

David Brooks

The Deep Learning Landscape

ML Models	ResNet, MobileNet, GPT-3, Transformer, …
Frameworks	TensorFlow, PyTorch, …
Libraries	openBLAS, Eigen, MKL, …
Compilers	TVM, XLA, …
Processors	CPUs, GPUs, TPUs, …

Tons of flexibility in designing a system!

A Systematic Approach

- A design methodology which is:
  - Not based on trial and error → Design space may be huge & intractable
  - Not based on "experience" → Past experience may not apply to future systems
- Goals:
  - Qualitativeness → "Parameter A is very important"
  - Quantitativeness → "Parameter A is more important than B by x times"
  - Comparability → "System A is strictly/conditionally better than B"

What is the Problem

- Inefficiencies with current DL benchmarking systems (E.g. MLPerf)
  - Long development cycles
  - Short shelf life
  - Not generalizable to future systems
  - Not comprehensive enough to reveal the whole system responses
- Lack of comprehensive evaluation methodology

What are the key insights

- Performance-sensitive attributes from DL models can be parameterized
  - Producing a collection of inputs that cover a much larger design space
  - Turning a fixed data point to a whole spectrum of data points
  - Allowing interpolation & extrapolation of system behavior

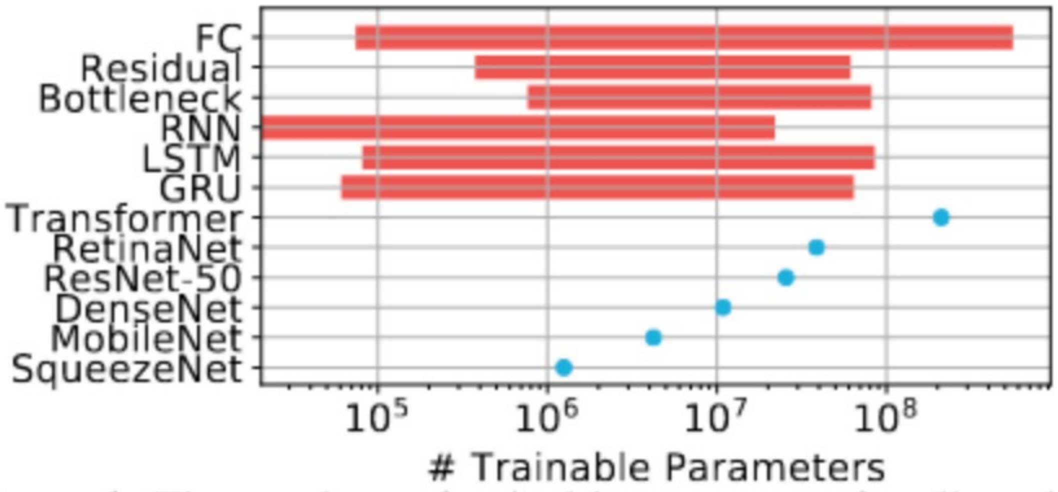
- Systematic analysis tools can:
  - Qualify, quantify and compare the DL design space in various dimensions

What is the solution

- Model generator
  - Model Parameterization

Model	Performance-Sensitive Parameters
DNN	No. of layers, nodes, input, output, and batch size
CNN	No. of kernels, size of kernels, input, output, and batch size
RNN	No. of layers, size of embedding, vocab, batch size

  - A collection of models that span across the whole design space

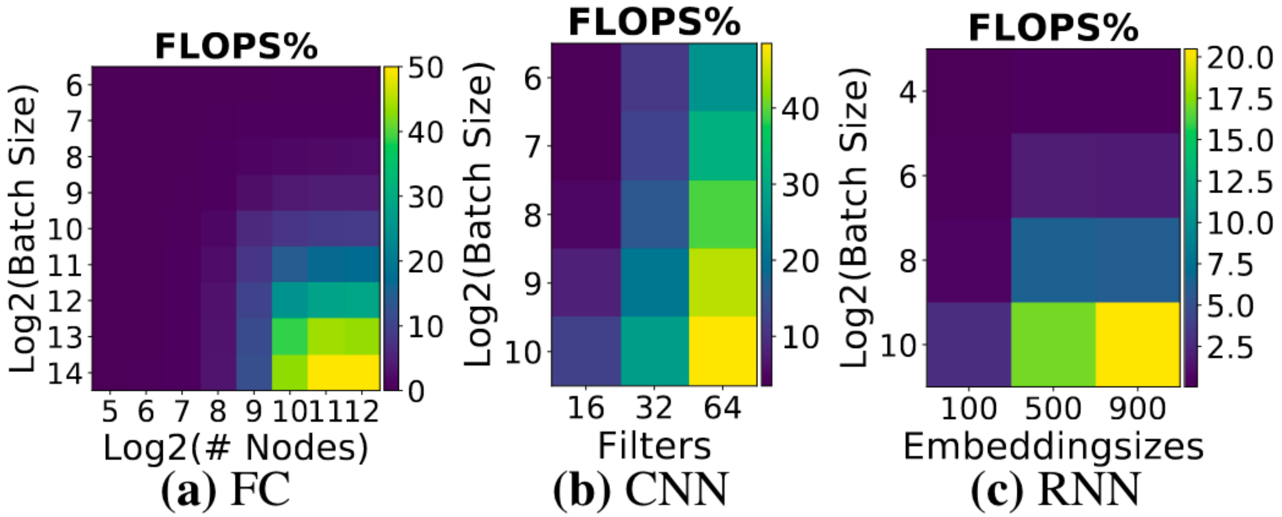


What is the solution

- Analysis toolbox
  - Heat Map
  - Linear Regression
  - Roofline Model
  - Box Plot

What is the solution

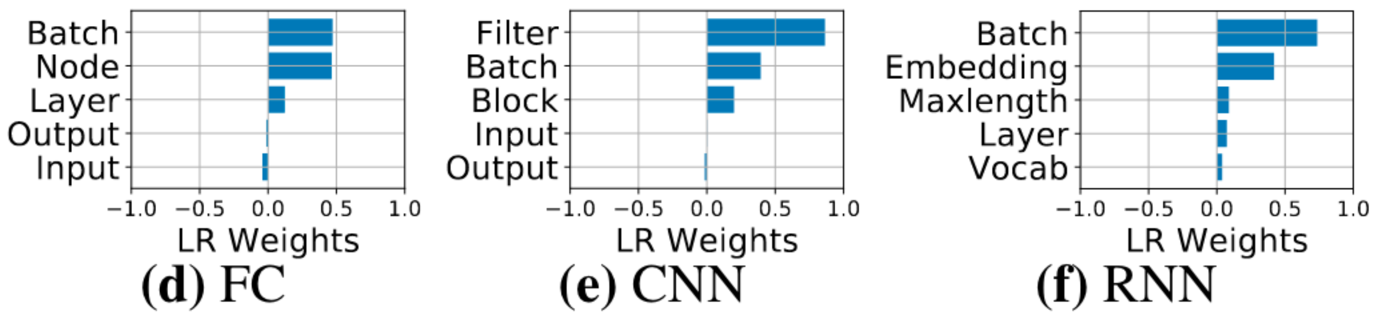
- Heat Map
  - Evaluate the contribution and interactions of two parameters in a system
- Example (TPU)
  - Parameters: Batch size and number of nodes
  - System metric: FLOPS utilization
  - Conclusion: TPU can exploit parallelism from both the batch sizes and model sizes



ParaDnn's FLOPS utilization

What is the solution

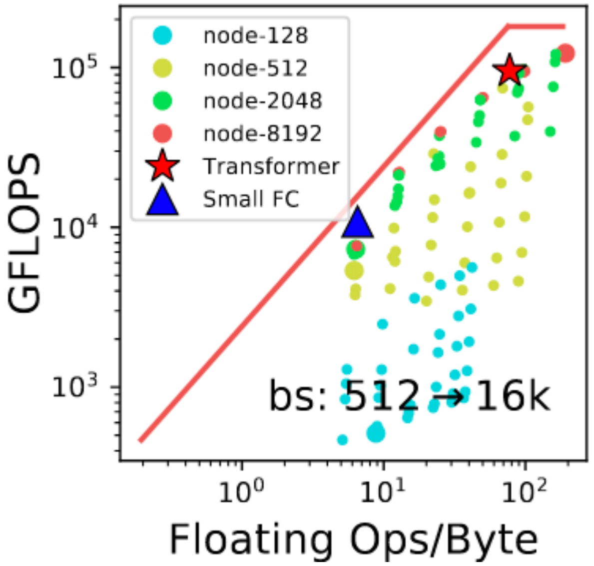
- Linear Regression
  - Evaluate the contribution of two or more parameters to a system
- Example (TPU)
  - Parameters: Batch size, number of nodes, layers, input size, output size
  - System metric: FLOPS utilization
  - Conclusion: TPU can exploit parallelism from both the batch sizes and model sizes



ParaDnn's sensitivity to hyperparameters

What is the solution

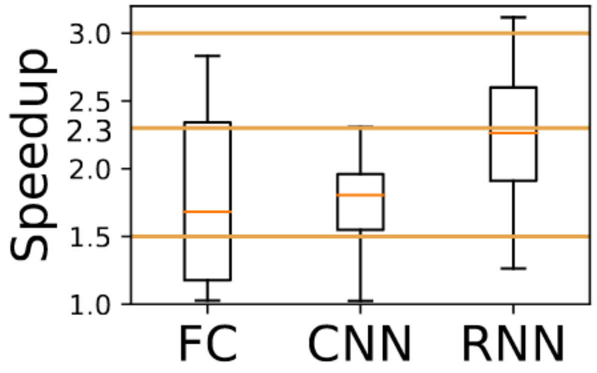
- Roofline Model
  - Characterize full system response to memory scaling region and compute scaling region
  - Points close to the slope are the memory bound operations
  - Points close to the roof are the compute bound operations
- Example (TPU)
  - Conclusion:
    - At low node size:
      - compute units are under utilized
    - At high node size:
      - Higher batch size hits compute ceiling
      - lower batch size hits memory ceiling



TPU rooflines for FCs

What is the solution

- Box Plot
  - Summarizes the variability in systems
- Example (TPU)
  - Conclusion:
    - TPUv3 has a more drastic improvement on RNNs



Speedups of TPUv3 over TPUv2

What is the takeaway message

- Due to system complexity in both the software and the hardware, formal tools are required to reason about the pros and cons of different systems
- Shouldn’t blindly apply “conventional wisdom”

Will this paper win the test of time award

- No.
- The analysis tools are a reproduction of work from other domains
- The analysis tools do not reveal more insights than we already know

Why should this paper not have appeared in top conferences

- Same reason as the previous question
- Fail to demonstrate what we can do more with the tools to reveal insights that were otherwise non-trivial to discover

Walle: An End-to-End, General-Purpose, and Large-Scale Production System for Device-Cloud Collaborative Machine Learning

Chengfei Lv
Zhejiang University & Alibaba Group

Chaoyue Niu*
Shanghai Jiao Tong University & Alibaba Group

Renjie Gu, Xiaotang Jiang, Zhaode Wang, Bin Liu, Ziqi Wu, Qiulin Yao, Congyu Huang, Panos Huang, Tao Huang, Hui Shu, Jinde Song, Bin Zou, Peng Lan, Guohuan Xu
Alibaba Group

Fei Wu
Zhejiang University

Shaojie Tang
University of Texas at Dallas

Fan Wu, Guihai Chen
Shanghai Jiao Tong University

Redefining Device-Cloud Interaction

- In Alibaba's E-Commerce streaming service:
  - Identify products – CV
  - Voice search, keyword triggers – NLP
  - Feeds based on users' browsing behavior – Recommendation

ML tasks are prevalent in mobile services

Redefining Device-Cloud Interaction

- ML for mobile services have stringent timing requirements:
  - CV – 30ms
  - NLP – 100 ~ 500ms
  - Feed – 500ms upon page refresh

ML mobile services are performance critical!

What is the Problem?

- Mainstream device-cloud interaction has the following drawbacks
  - High latency
  - High cost & heavy load
  - Data security & privacy
- Implementation challenges

Stage	Problems
Data pre-processing	▪ User data are sparse and intermittent ▪ Privacy concerns

ML Inference	▪ Cloud-side:
	  ▪ Consumes much computation
	▪ Device-side:
	  ▪ requires porting diverse ML operations

Server-side book-keeping	▪ Managing multiple services, business scenarios efficiently ▪ Syncing local and remote user states

What is the Insight?

Stage	Insights

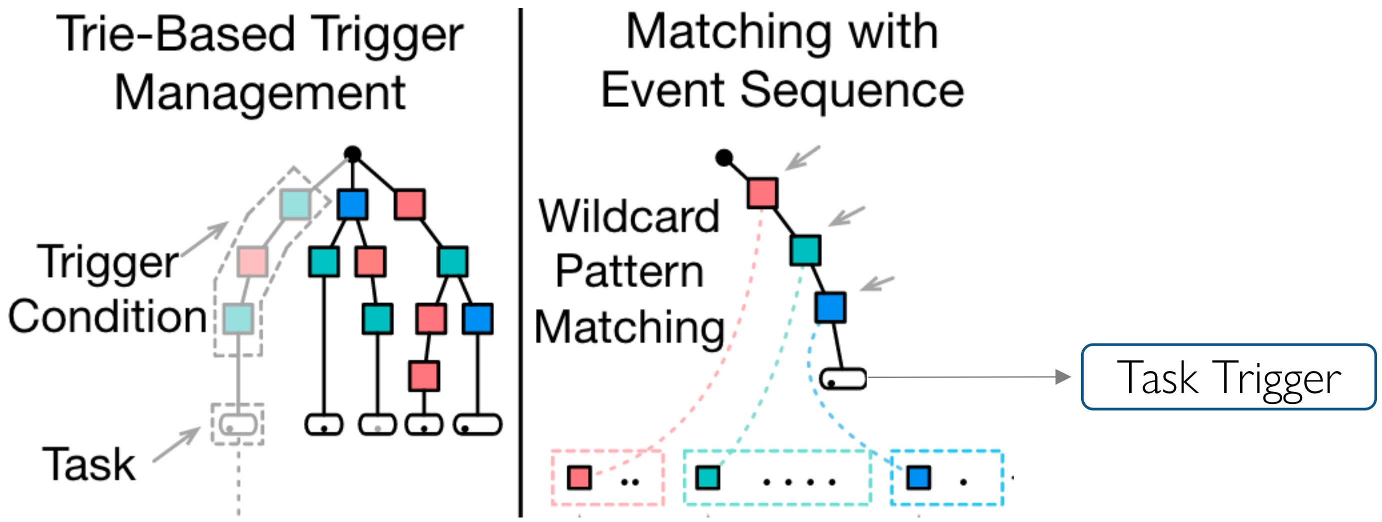
Data pre-processing	▪ Device-side processing:
	  ▪ Distributed computation
	  ▪ Good data locality
	  ▪ Privacy preserving

ML Inference	▪ Device-side processing:
	  ▪ Decompose into a set of atomic operations
	  ▪ Consistent interface

Server-side book-keeping	▪ Managed hierarchically ▪ Coarse-grain syncing

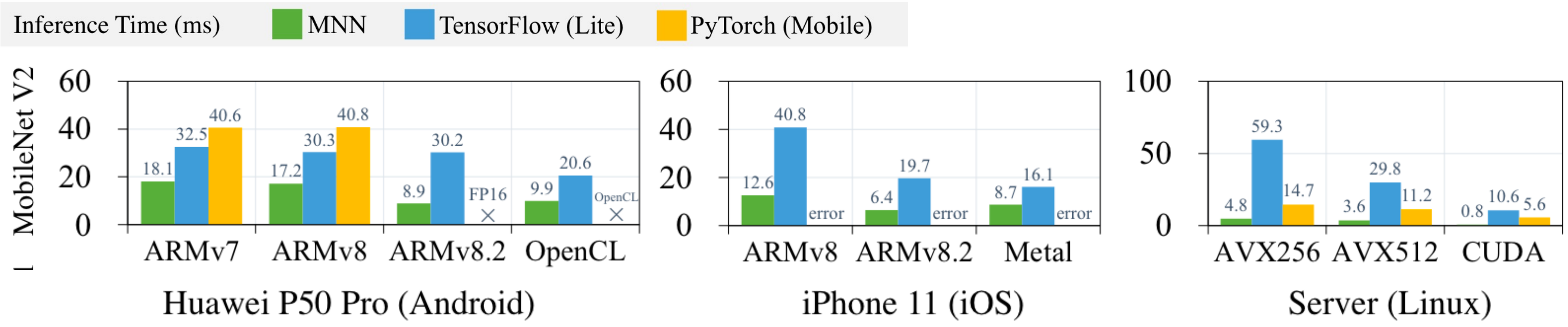
What is the Solution?

- Data preprocessing:
  - Device tracks and manages user actions
    - Store event sequences in data structures
    - Multiple trigger conditions detections in time series



What is the Solution?

- ML inference:
  - ML operation decomposition
    - Computation
    - Transformation
    - Control-flow
    - Raster
  - Compatible with high-level API calls from Python
  - Mathematical formulation to find the best modes of computation



What is the Solution?

- Book-keeping
  - Business scenarios, services, tasks, versions can be organized hierarchically and managed using git
  - Syncing only needs to be done upon new events, rather than every user action
    - Piggy-backing local device states together with another network request

What is the takeaway message

- There is large untapped power from user devices which can process a considerable amount of computation
  - offloading server workload
  - reduce communication cost

Will this paper win the test of time award?

- No.
- Software may scale at a faster speed than hardware. It is unclear 10 years later what will be the computation requirement and if it will still be feasible.
- Power analysis is not discussed, which is a fundamental aspect in edge processing

Why should this paper not have appeared in top conferences?

- Power analysis is a huge part of device side offloading. This aspect is not discussed.
- Some optimizations seem redundant or lack quantifiable results. E.g. the performance is attributed to mathematical optimizations. But there is no convincing analysis.