Edge Computing for Real-Time ML/Cognition

Break-Out Session

NSF PI Meeting 11/04/2019

Session co-leads: Wei Gao and Mahadev Satyanarayanan

Attendees & Contributors (55)

- Trisha Andrew
- Moussa Ayyash
- Vaneet Aggarwal
- Aruna Balasubramanian
- Laura Biven
- Abhishek Chandra
- Chen Chen
- Samir Das
- Sujit Dey
- Fahad Dogar
- Flavio Esposito
- Rui Dai
- Deepak Ganesan
- Javad Ghaderi
- Soudeh Ghorbani
- Jiaqi Gong
- Jie Gu
- Tian Guo
- Indranil Gupta
- Mohammad Hajiesmaili
- Kyle Hale
- Bo Ji
- Bala Kalyanasundaram
- Sandip Kundu
- Lifeng Lai
- Palden Lama
- Bin Li
- Benyuan Liu
- Jia (Kevin) Liu
- Xin Liu
- Xiaojun Lin
- Hui Lu
- Lena Mashayekhy
- Matt Mutka
- Tamer Nadeem
- Miao Pan
- Lu Peng
- Barath Raghavan
- K. K. Ramakrishnan
- Danda Rawat
- Eric Rozner
- Surosh Singh
- Aatmesh Shrivastava
- Massimo Tornatore
- Shivaram Venkataraman
- Pu Wang
- Liqiang Wang
- Xin Wang
- Dalei Wu
- Fei Xie
- Jiang (Linda) Xie
- Dejun Yang
- Li Yang
- Ming Zhao
- Ziliang Zong

Research Topics (in descending order of votes)

1. Distributed learning for DNNs

2. Inferencing vs. training at the edge

3. Just-in-time learning

4. Resource management

5. Quality of Experience (QoE)

6. Security & privacy
   - How to protect from distributed learning?
   - Will protection lead to performance loss?

7. Splitting DNN pipelines

8. Heterogeneity
   - How to tackle with the varying split between the edge and end devices?

9. Hardware acceleration
   - What HW acceleration is needed?
   - How to reconcile accelerators?

10. MIMD vs. SIMD
    - How to balance? How to handle SIMD for multi-tenant edge?

Additional Thoughts

Discovery and maintenance of edge (Tier-2) and context

Evaluation metrics and key goodness indicators (KGI)

Custom-designed DNNs for edge devices (model compression also included)

Collaborative inferencing among Tier-3 devices

Correctness and Debugging of complex adaptive Tier-3/Tier-2 architectures

Opportunistic service discovery, offloading and data collection

Distributed Learning for DNNs

Why is this important?

- Sources of incoming data are distributed
- Because distribution intrinsically masks provenance of training data

Why is this hard? What are the major challenges?

- How to label the high volume of data produced by the massive amount of edge devices?
- Heterogeneity on modularities and characteristics of data
- Design and verification of edge systems are specific to certain ML models, but are much slower than how ML models are mutating.
- Non-uniform distribution of data and workloads at the edge may overload some edge servers and prolong the response latency
- New malicious attacks have been developed to breach privacy in federated learning

Resulting Research Problems:

- Edge system solutions that are independent from specific ML models
- Appropriate abstractions of ML models to facilitate generic edge system design
- Distributed data collection with preprocessing that avoids redundancy and minimizes data size
- Redundancy in resource provisioning to avoid overloading and excessive response delay

Inferencing vs. Training at the Edge

Why is this important?

- Continuous training at the edge timely updates a pre-loaded generic model to be more context specific
- Learning more at the edge protects data privacy

Why is this hard? What are the major challenges?

- Training is expensive; resources are limited at the edge, especially the computing capacity and power
- Heterogeneity of edge data in volumes, rates and complexity makes training even more expensive
- It is hard to synchronize the training processes, especially their convergence, among distributed edge servers
- Data ownership or uncertain willingness of sharing data may lead to incomplete or biased data for training

Resulting Research Problems:

- Training algorithms should be resource-aware (e.g., lightweight and energy-efficient)
- Training at the edge should be application specific to balance between accuracy and complexity
- Training in a hierarchical manner could possibly help synchronize among distributed training processes
- Develop reconfigurable hardware that can flexibly adapt to both training and inferencing tasks

Resource Management

Why is this important?

- ML pipelines will be split between the edge and end devices
- Data produced at the edge could have heterogeneous characteristics, which result in varying ML computations with intermittent peaks

Why is this hard? What are the major challenges?

- Hardware limitation at the edge adds complexity to virtualization on ML tasks.
- Global changes in context and priorities call for agile resource re-allocation
- Multi-tenancy edge

Resulting Research Problems:

- Computing, storage and network bandwidth resources should be all taken into consideration for resource management
- Better virtualization methods are needed, due to the involvement of heterogeneous data and hardware
- Resource management should be highly agile to allow multiple real-time ML tasks to co-exist
- Algorithms that better factorize ML tasks could improve the efficiency of resource management. Exploiting the commonality among edge tenants could help such factorization.

Quality of Experience (QoE)

Why is this important?

- Real-time ML at the edge is often human-in-the-loop (i.e., cyber-human systems)
- There is a big gap between the existing QoS metrics and the humans' perceived QoE
- New QoE models and metrics are needed

Why is this hard? What are the major challenges?

- Humans' perception about edge system performance is subjective and different across different applications
- How to timely, precisely and efficiently collect human feedback about QoE?

Resulting Research Problems:

- QoE metrics should be more comprehensive and incorporate more aspects, such as humans' delay sensitivity and privacy measurements
- Segment the space of edge applications at a fine granularity based on their characteristics and requirements
- Humans should be in the loop, and edge systems should be personalized to achieve better QoE with timely human feedback