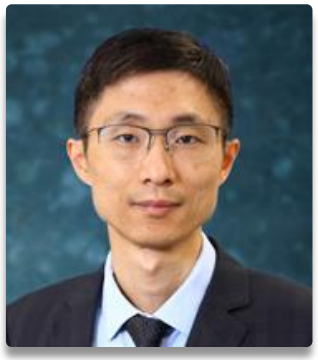
EdgeML: An AutoML Framework for Real-Time Deep Learning on the Edge

ZHIHE SHAO, KAI WANG, NEIWEN LING AND GOULIAN XING

About the Authors

Zhihe Zhao

Neiwen Ling



Guolian Xing

Kai Wang

- Zhihe Zhao
  - B.S in Computer Science from University of Liverpool
  - Ph.D Candidate at Duke University in Electrical Engineering
- Kai Wang
  - Ph.D Candidate at the Chinese University of Hong Kong
- Neiwen Ling
  - Ph.D Candidate at the Chinese University of Hong Kong
- Guoliang Xing
  - Professor in the Department of Information Engineering at the Chinese University of Hong Kong
  - Received his doctorate in 2006 from Washington University of St. Louis

What is Machine Learning

What is a Neural Network (high level)

TRY ME

- Data is input into the network and a prediction is the output
- Goal is to find weight parameters (W) that fit a training set that consist of inputs and their corresponding labels
- Intermediate activations are created in the different layers which lead to a prediction
- Predictions are not always correct, so errors are backpropagated into the model via weight gradients
- Many passes through input data set necessary for training
- DNN => Neural net with many hidden layers implemented

Quick Definitions

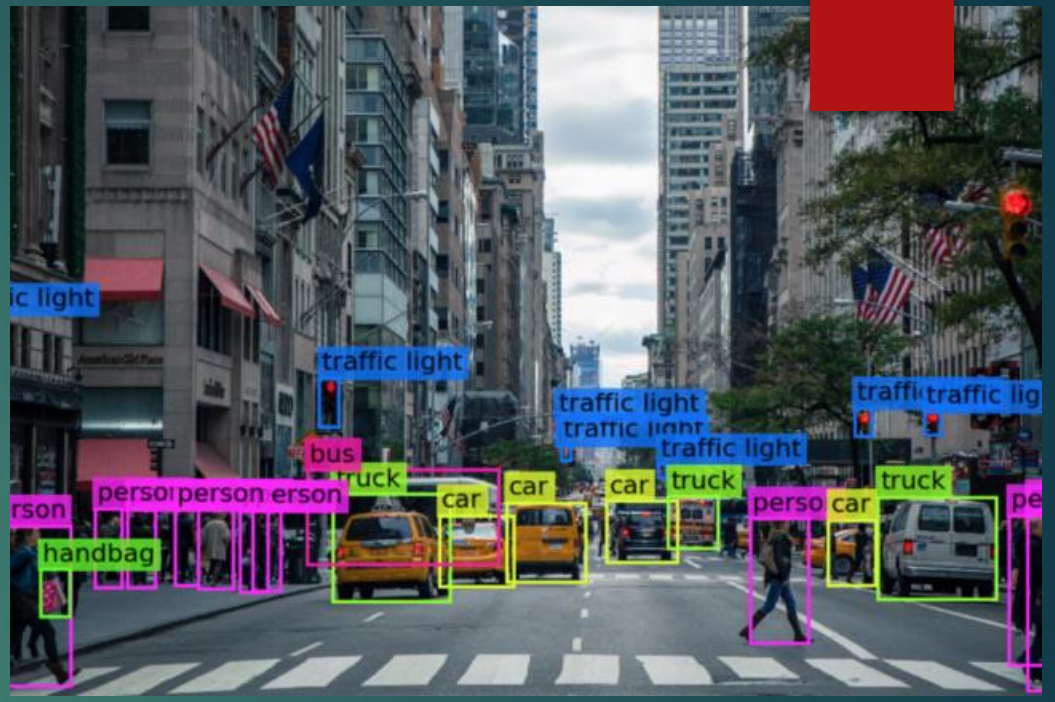
- Reinforcement Learning:
  - Systems take action in order to maximize the notion of a cumulative reward
  - Good action is rewarded, bad action is not
- Offline Learning vs. Online Learning:
  - Online learning is training as the data comes in while offline is working off of an existing data set
- AutoML:
  - “an emerging paradigm that aims to automate the pipeline of DNN design”
  - More generally I think of it is as the automatic application of machine learning onto other machine learning

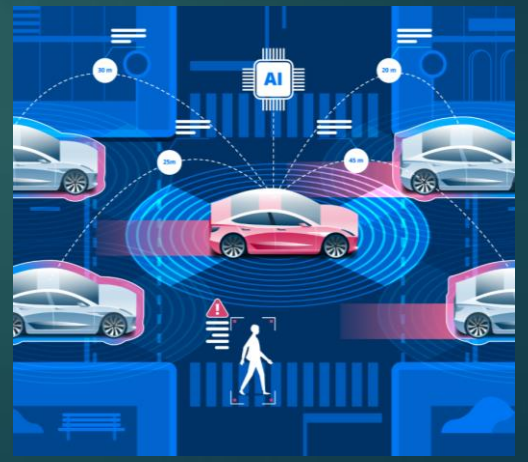
What is Q-Learning?

- A reinforcement learning technique used for learning the optimal policy in a Markov Decision Process
- Markov Decision Process:
  - Provides a mathematical framework for modeling decision making in situations where outcomes are partly random and partly under the control of a decision make
- Model-Free Learning
  - Meaning it can derive an optimal policy directly from its interactions with the environment
- Can utilize the Monte Carlo technique:
  - Repeated random sampling to obtain numerical results

What is the topic

- There are an increasing number of data-intensive & time critical IoT environments
  - DNN needed to be deployed at the edge:
    - Reduce latency to prediction
    - Handle individualized data from their deployment
- Some example applications include:
  - Autonomous driving, embedded computer vision and VR





What is the problem?

- DNN deployed at the edge must deal with two big problems:
  1. Limited computational power (these are not large servers)
  2. Dynamically changing environments make "one-size fits all" solutions impractical
- What if we could use machine learning itself to assist in the machine learning?
  - The concept of a DNN that adapts to a dynamic runtime environment to optimize itself for certain situations

Other Solutions

- There have been many studies and AutoML solutions proposed in which the parameters of a DNN or another ML method can be tuned using machine learning (ex)
  - Model compression parameters
  - Number of layers for a network
  - Learning Rate
  - Etc.
- What if instead of using AutoML to tune parameters, we instead use AutoML to update the **execution strategy** of the DNN in real time?

EdgeML

An AutoML Framework which provides flexible & fine-grained DNN execution control utilizing:

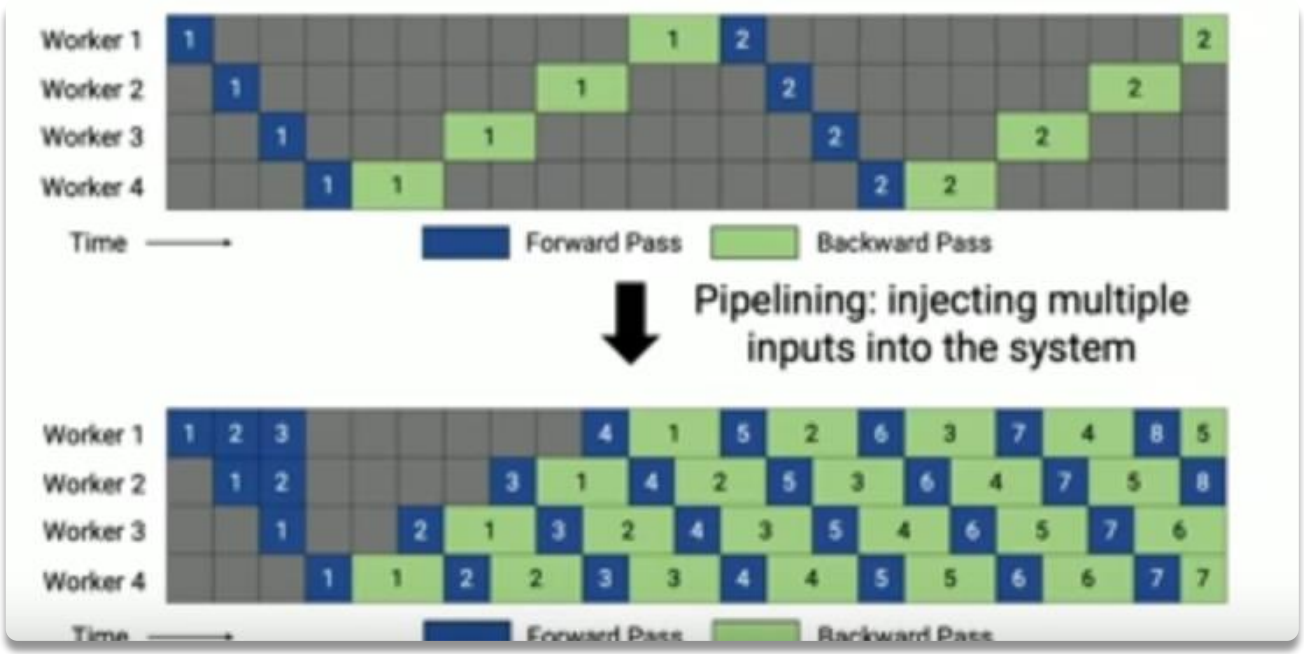
- Workload offloading mechanisms
- Progressive Neural Architecture

Goal:

- Achieve desirable latency-accuracy-energy system performance of edge platforms

What is Workload Offloading

- Could break up the inputs across the same DNN split among many machines
  - i.e. Many inputs, many machine solution
- Could introduce pipelining into the DNN where the backpropogation for one pass could be computed at the sime time as the forward propagation of the next pass



Credit: Generalized Pipeline DNN – Databricks

What is Workload Offloading?

- This is **NOT** what is done in EdgeML
- EdgeML says that there is one edge device and there is one cloud
  - The cloud can compute much faster BUT it is at the whim of communication bandwidth limitations
- EdgeML's solution to workload offloading attempts to find the right balance of local computation on the edge and cloud offloading by dynamically varying the partition point in the DNN to adapt to the env

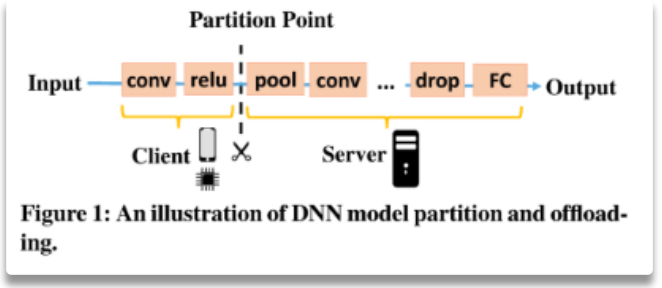
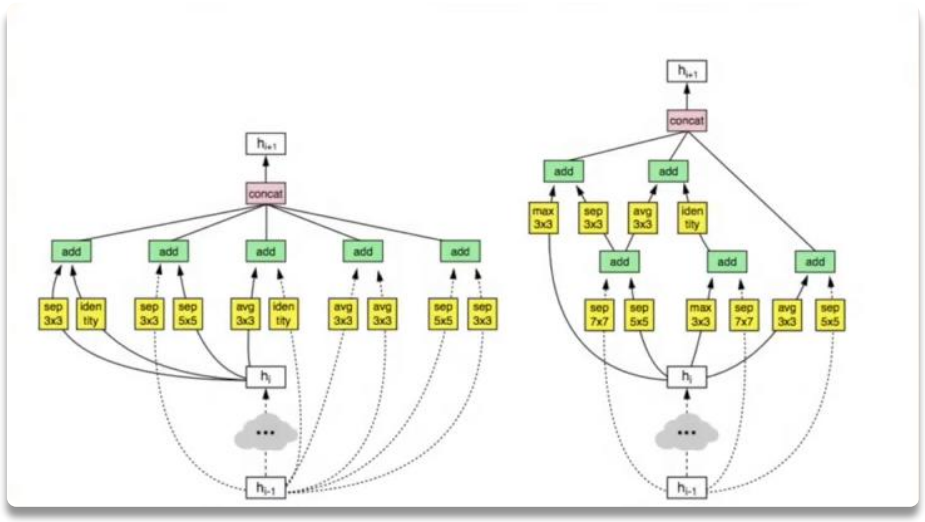


Figure 1: An illustration of DNN model partition and offloading.

What is Progressive Neural Architecture (PNA)



- Represents the ability to inject branches into a neural network where early exit or prediction is possible without having to go through all layers
- Exploits the fact that different data has different difficulty in characterization… Not all data needs to go through all layers of the NN
- (ex) Pixels in the middle right are strongly indicative of a “3” or a “5” from previous example

What can EdgeML Try and Vary?

- Where the DNN is partitioned for cloud offloading
- Where branches are inserted for PNA
  - And the threshold values of these branches (i.e. what confidence is needed for an early exit)
- Problem? Search Space can get very large (infinite) very quickly
  - i.e. The VGG16 DNN used has:
    - 17 possible partition points
    - 3 possible branch points (for PNA)
    - PNA branch thresholds continuously distributed between 0-1

Solution to infinite Search Space?

- EdgeML applies a reinforcement learning algorithm (RL Optimizer) to tune the execution policy based on the past performance of the previous strategy or strategies
- Goal of the RL Optimizer is to MAXIMIZE accuracy while limiting latency and energy usage
  - Bias towards accuracy w/ an emphasis on meeting the upper-bound requirements on latency and energy

$$\max \ Accuracy(I) \ \ s.t.$$

$$Latency(I) < L_{usr}, \ Energy(I) < E_{usr}$$

Offline Stage

- The DNN is first transformed into a Progressive Neural Architecture by inserting some base branches
- Possible model partitions are manually selected from the programmer
- Trained locally on some base data dependent on its deployment

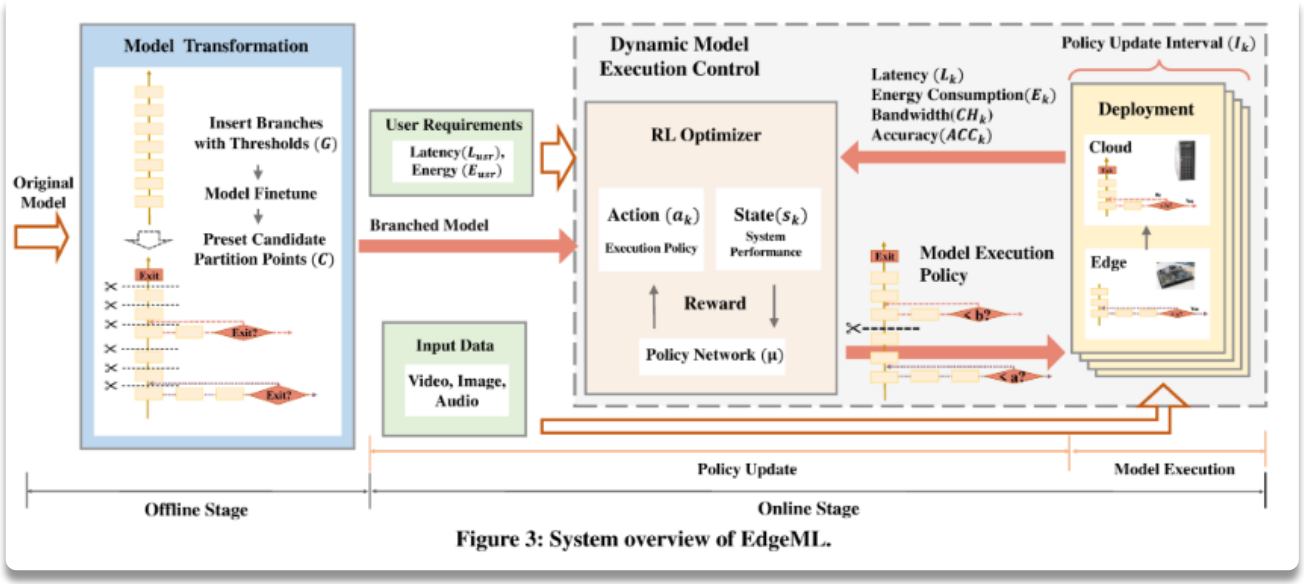


Figure 3: System overview of EdgeML.

Online Stage

- DNN Model executes based on some initial parameters
- Performance data passed back from system
- Data utilized by RL Optimizer to derive a new Action or Execution Policy
- Execution Policy re-applied to system and REPEAT

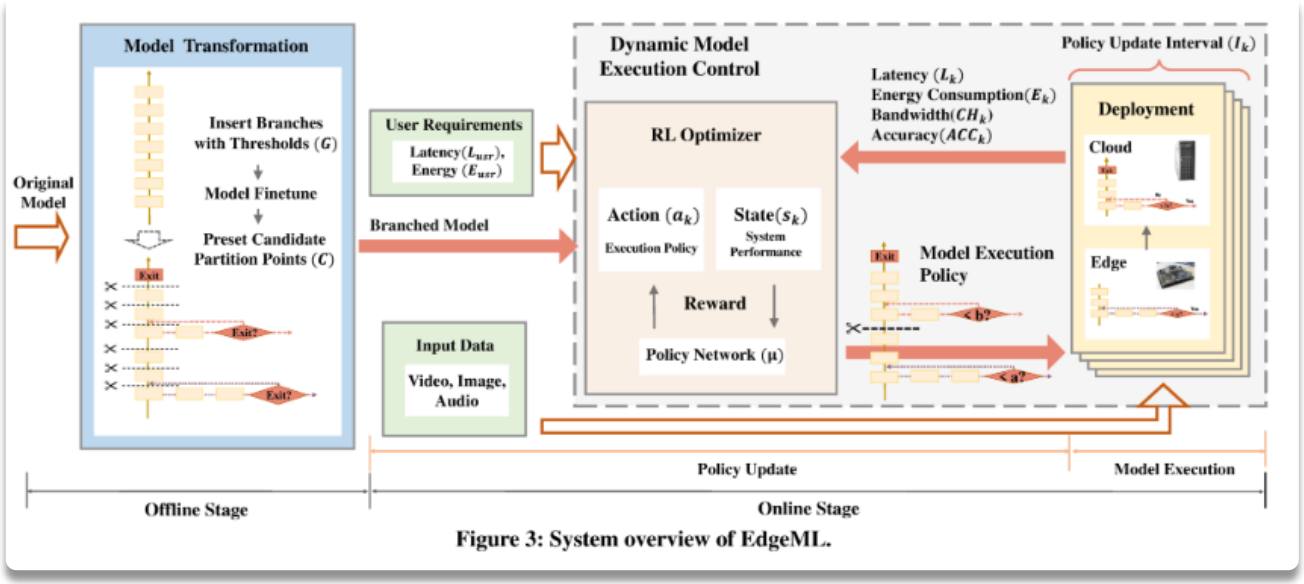


Figure 3: System overview of EdgeML.

Execution Policy

Partition Location for Edge Offloading

Branch Placement for PNA

Policy Execution Time set

- Because any policy is at the whim of changing communication bandwidth and input data, a reasonable time must be achieved before we re-visit changing the execution policy
- In their experiment it was a constant value based on the necessary latency to compute one epoch of image classification

Performance Markers

- Performance of an execution policy is passed back to the RL Optimizer to decide the next execution policy
- Performance markers include:
  - Latency
  - Energy Consumption
  - Bandwidth
  - Accuracy

RL Optimizer (high level)

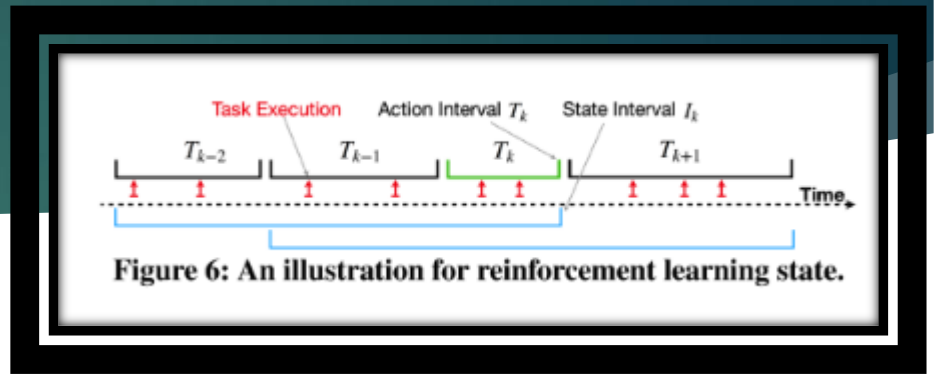


Figure 6: An illustration for reinforcement learning state.

Goal

Goal of the RL Optimizer is to decend the State Space provided using the Action States

- Making decisions about how to define the action state based on performance of previous Execution Policies

State

State Space:

- Defined by the latency, energy consumption and data transmission rate

$$s_k = \{\mathbf{L}(I_k), \mathbf{E}(I_k), \mathbf{CH}(I_k)\}$$

Action

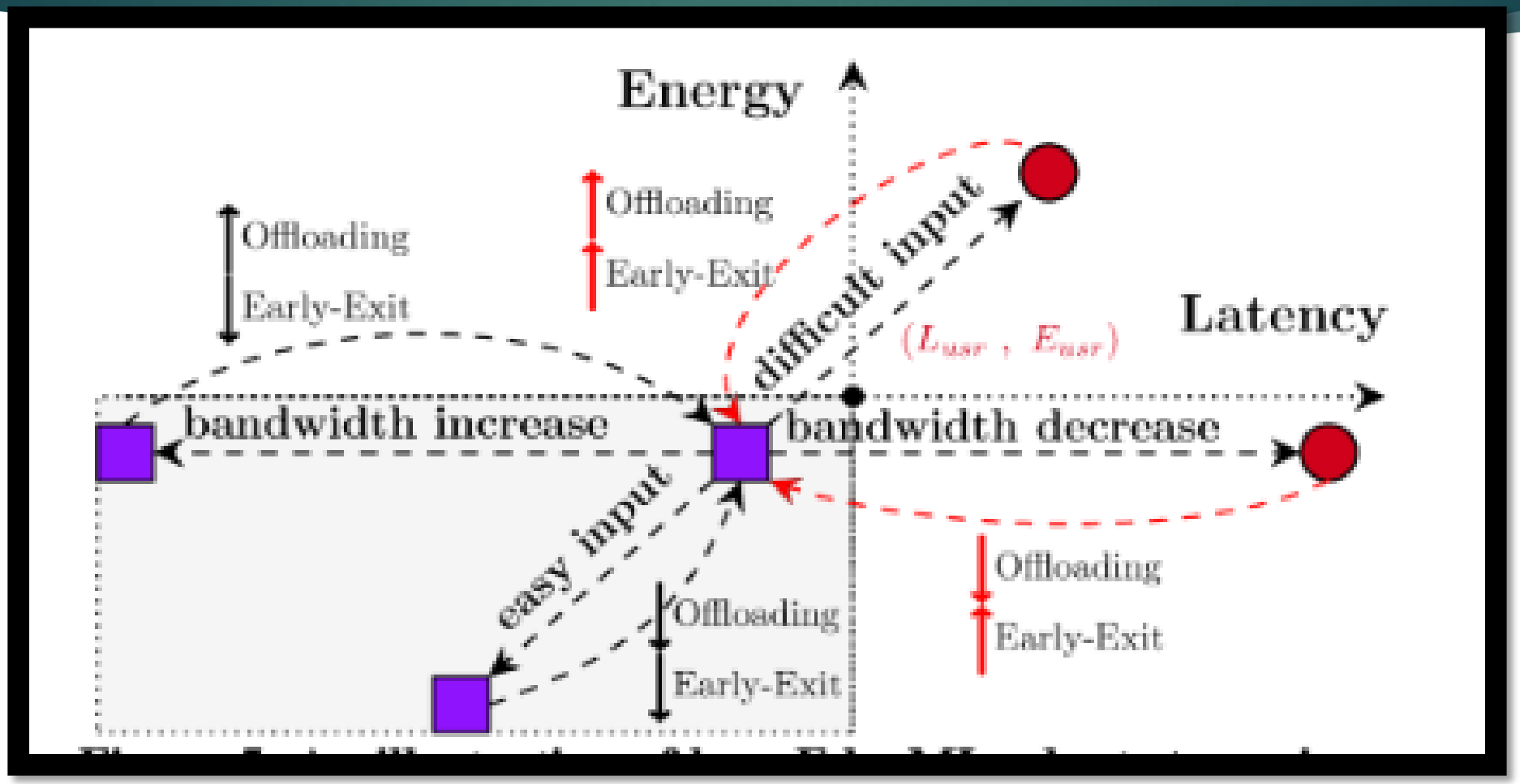
Action Space

- Defined by the Threshold values of all branches
- Model partition point
- Action interval (time before re-evaluating model)

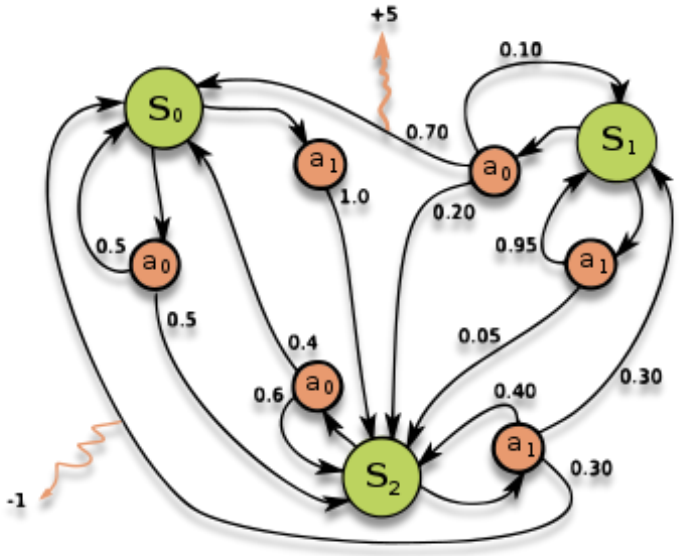
$$a_k = \{\mathbf{G}_k, C_k, T_k\}$$

Tk is the number of tasks executed during the state interval Ik

How RL Optimizer Reacts



Scalability to New Devices



- Every new Edge Device does not need to start from scratch with the RL-Optimizer training
- Could do a direct transfer of an RL Optimizer BUT:
  - Slow as it is a large transfer size
  - Bad idea as a new edge device will have new challenges
- Instead, we can transfer the Markov Decision Process and allow it to train itself
  - i.e. Send our performance experience, and the decisions we made at that time

Experiment

- Edge Device:
  - NIVIDIA Jetson TX2 and NIVIDIA Jetson Nano
- Cloud Infra:
  - Cluster of NIVIDIA GeForce GTX1070 and Intel I7 processor
- Developed EdgeML on top of tensor flow utilizing a 3-layer model of two representative built in DNN models:
  - VGG16
  - ResNet50
- Used Wondershaper to simulate changing bandwidth
- Used variable images for classification as a means of dynamically changing input

Results / Performance

- Model Performance Applicability:
  - How well does the model adapt to different DNN
- Bandwidth Applicability:
  - How well does the model adapt to changing bandwidth
- Input variation Impact:
  - How does the model handle changing inputs

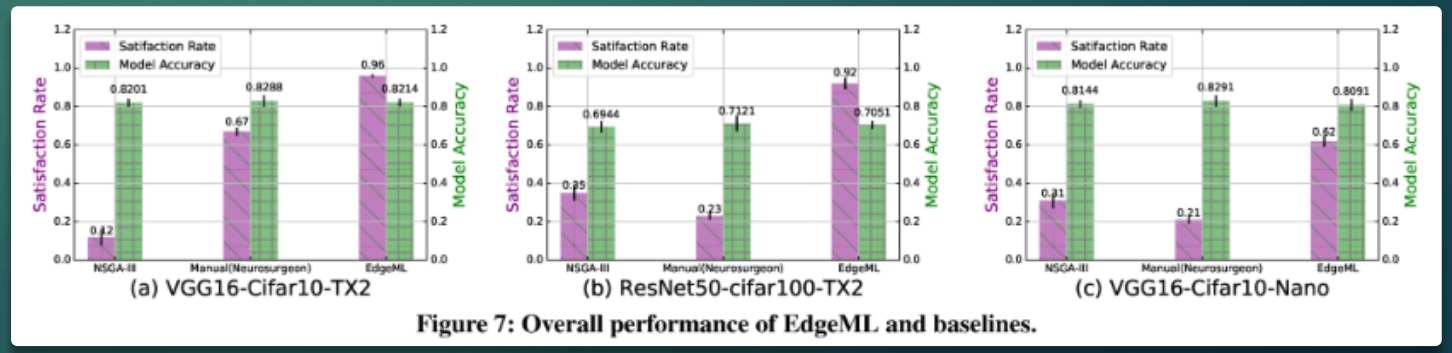
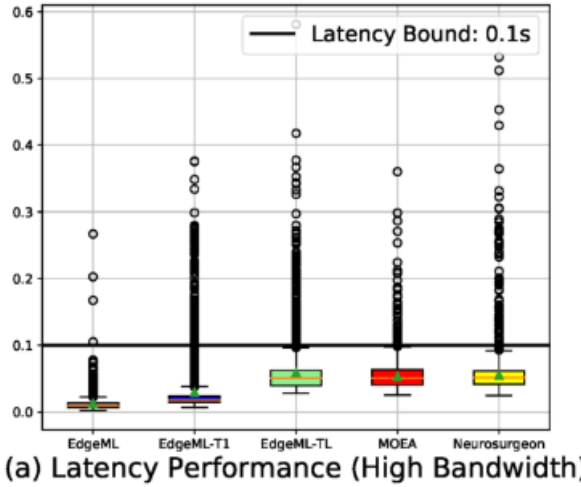


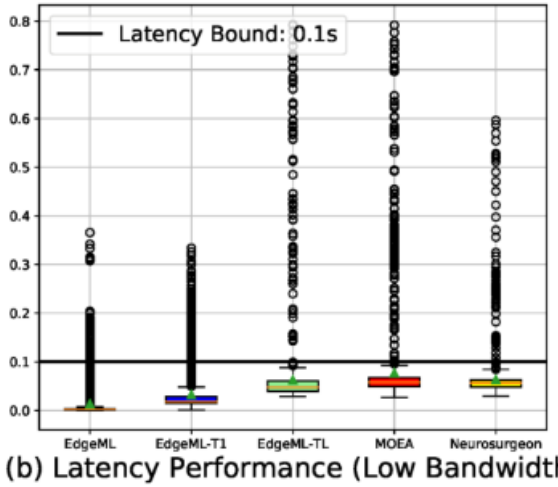
Figure 7: Overall performance of EdgeML and baselines.

Comparisons

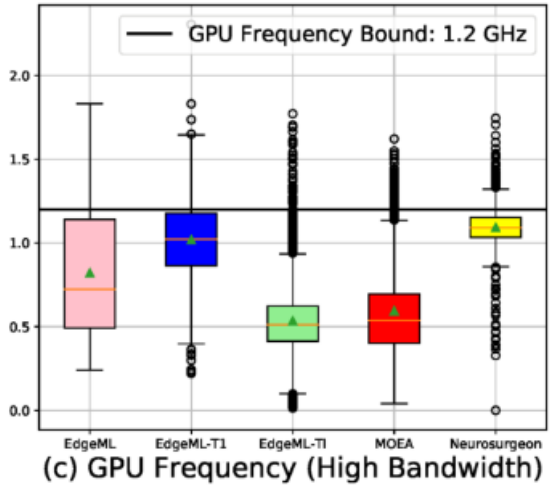
- EdgeML
  - Base Version
- EdgeML-T1
  - Threshold values of all branches rounded down to match that of first branch
- EdgeML-TL
  - Threshold values of all branches rounded up to match that of last branch
- Neurosurgeon
  - Off the shelf DNN offloading mechanism
- MOEA
  - A genetic algorithm to replace the RL Optimizer



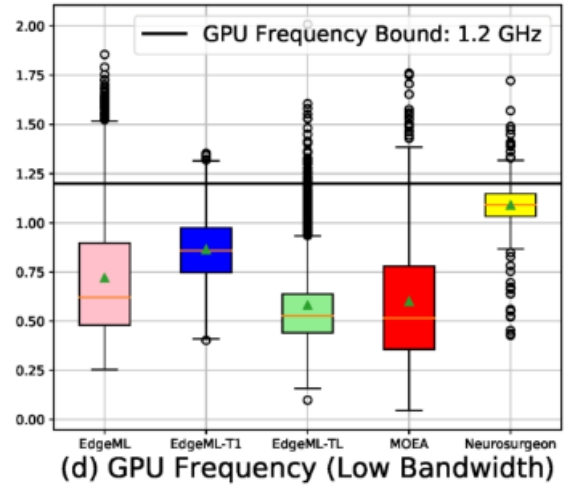
(a) Latency Performance (High Bandwidth)



(b) Latency Performance (Low Bandwidth)

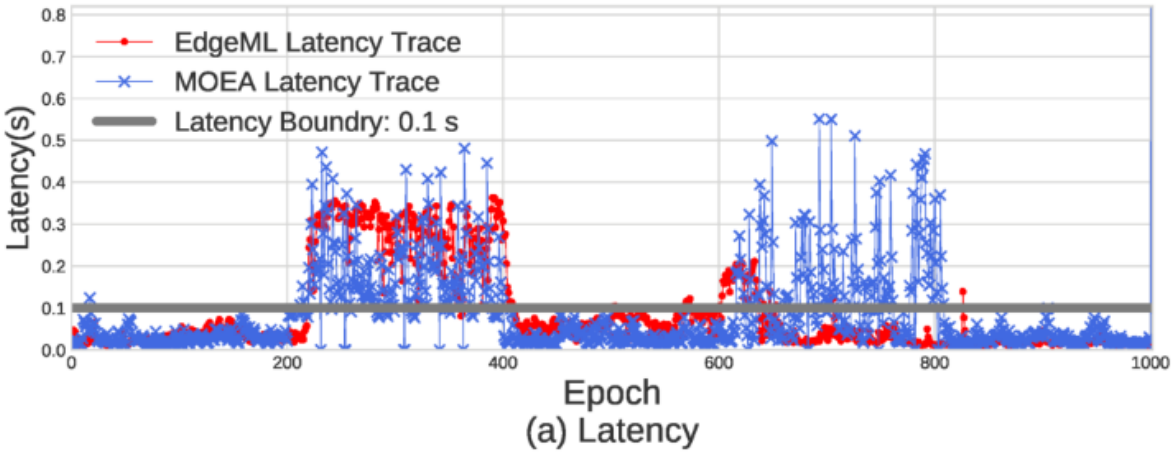


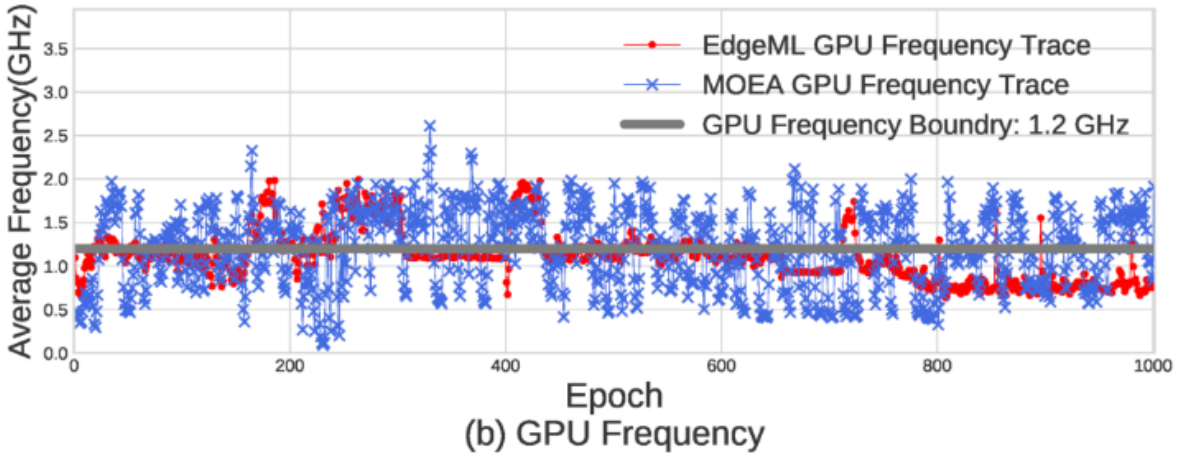
(c) GPU Frequency (High Bandwidth)

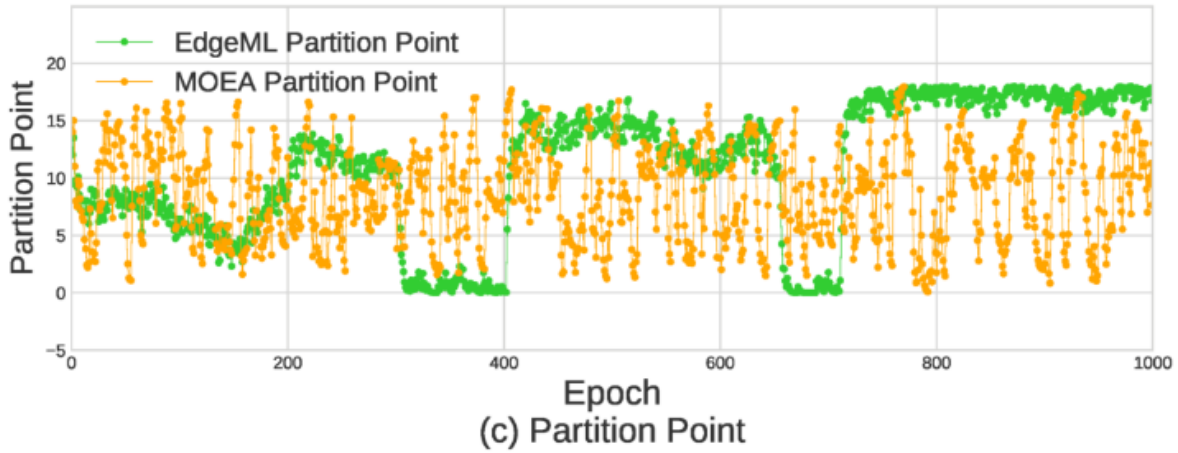


(d) GPU Frequency (Low Bandwidth)

Edge ML has better latency, but MOEA dominates in energy consumption







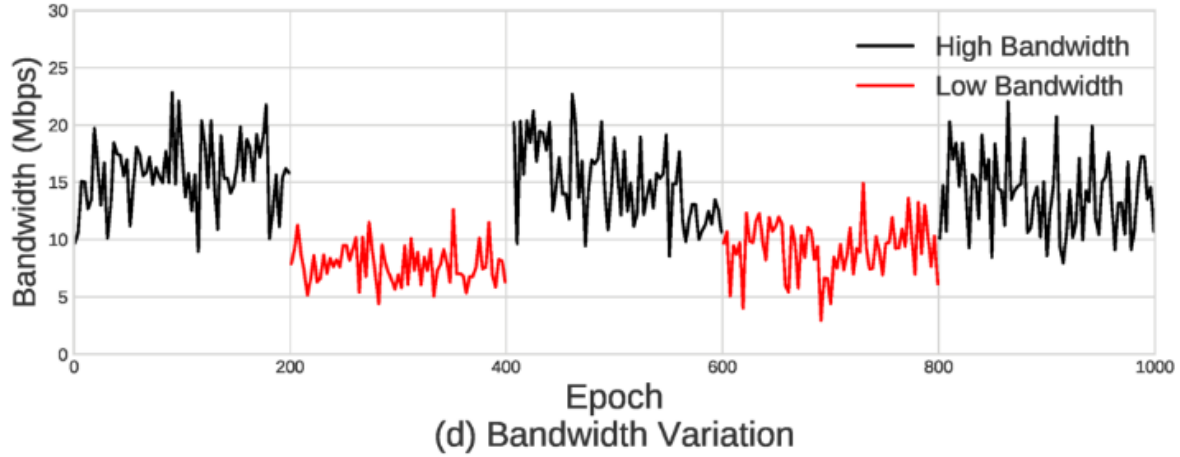


Figure 10: EdgeML system performance and generated actions versus time

EdgeML fluctuates less to changing environments and can more quickly adapt

Overall Performance

- 8x improvement in meeting user satisfaction
  - i.e. meeting energy and latency requirements
- Sacrifices less than 0.2% accuracy
- The overhead of RL Optimizer only adds 2% computation which is easily surmounted by the faster prediction times

Question

- Their fluctuations in the environment were fairly drastic, would it have been more realistic to handle a slow change in environment?
  - Like a slow losing of light or bandwidth slowly decreasing as more students enter a classroom on the wifi
- What do you all think about the applicability of the EdgeML model on general scenarios?
  - Do you think this is a drag and drop solution to DNN or is this more tailored towards the problems that the designed?