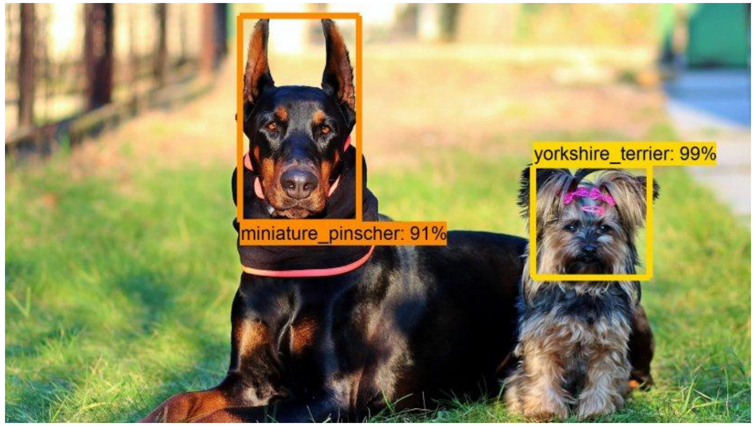
Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy

En Li, Zhi Zhou, Xu Chen

Sun Yat-Sen University
School of Data and Computer Science

The rise of artificial intelligence

- **Deep learning** is a popular technique that have been applied in many fields



Object Detection

Voice Recognition

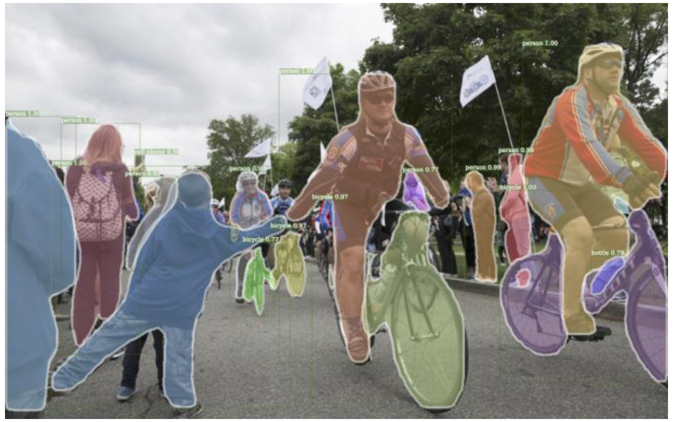
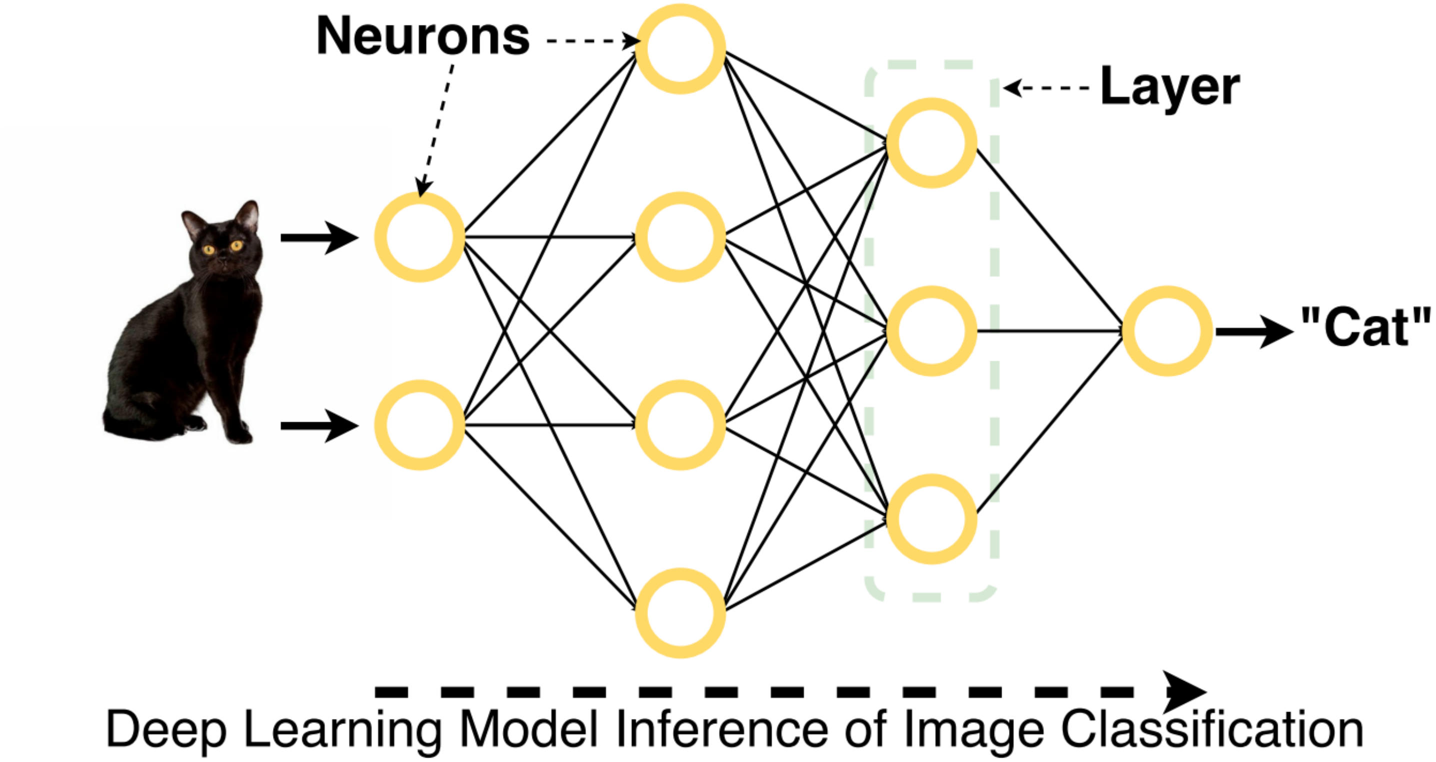


Image Semantic Segmentation

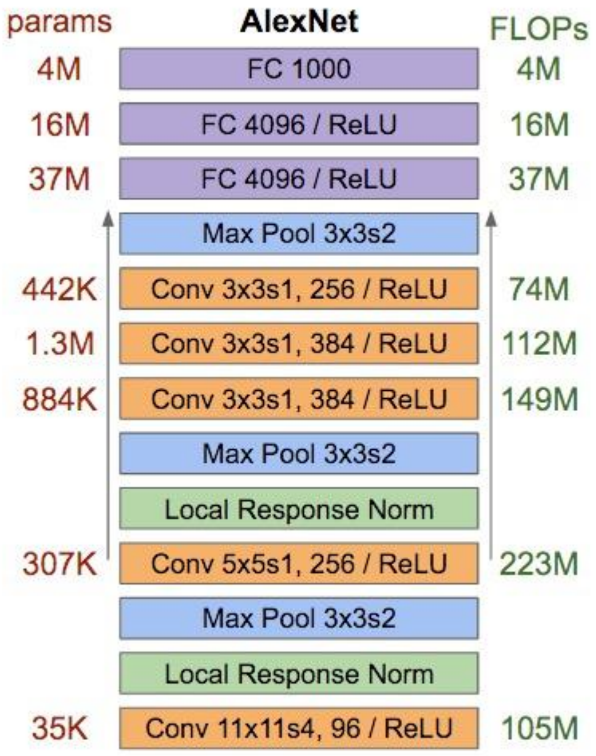
Why is deep learning successful

- Deep neural network is an important reason to promote the development of deep learning

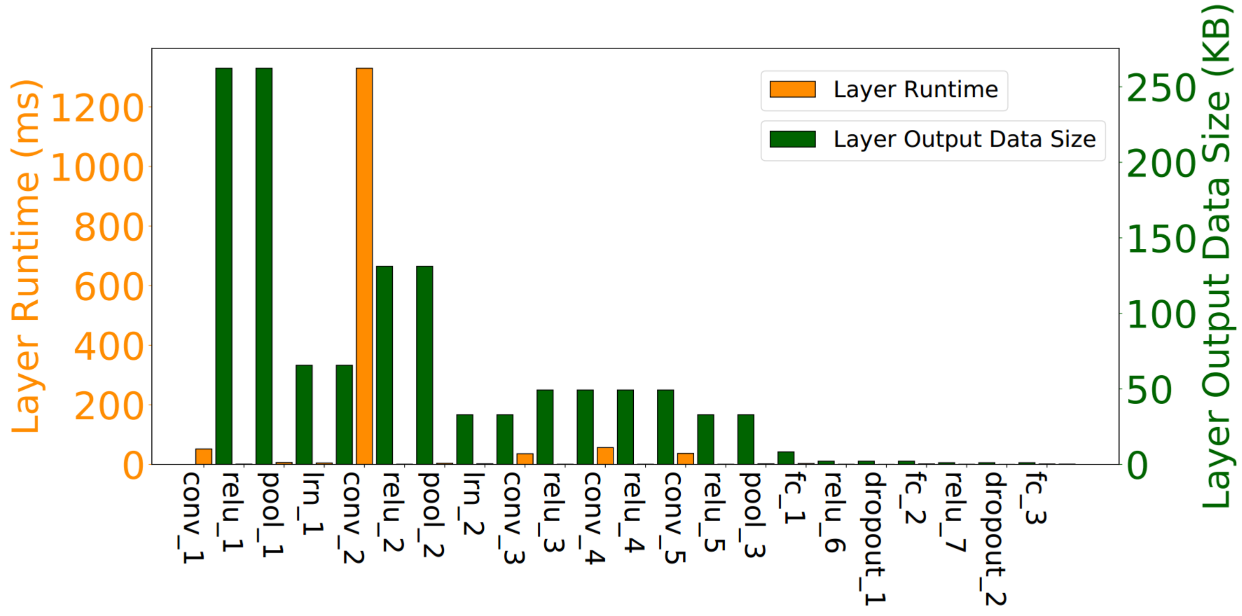


The headache of deep learning

- Deep Learning applications can **not be well supported** by today's mobile devices due to the large amount of computation.



AlexNet Params & Flops

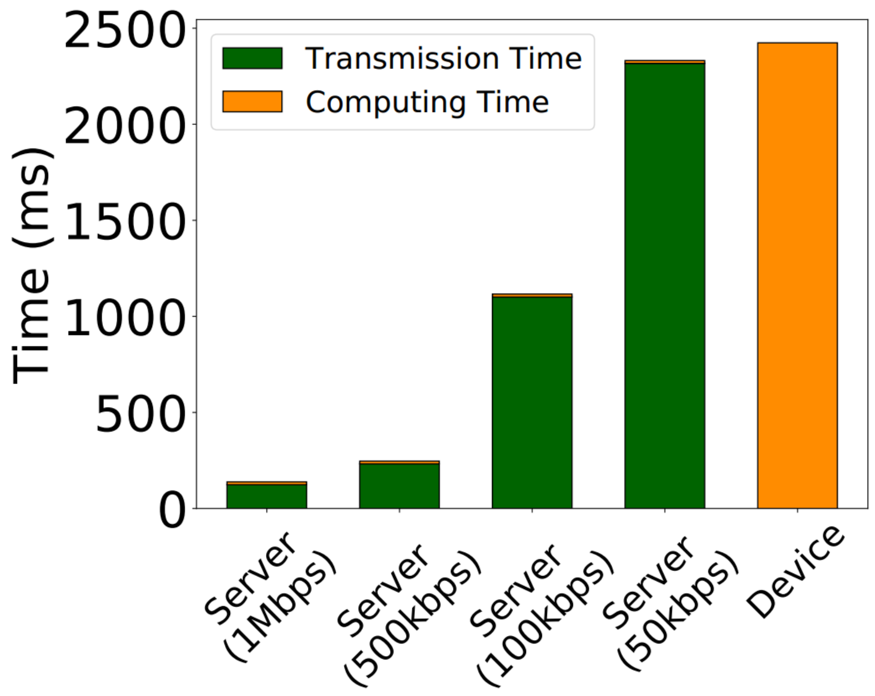


AlexNet Layer Latency on Raspberry Pi & Layer Output Data Size

What about Cloud Computing?

- Under a cloud-centric approach, large amounts of data are uploaded to the remote cloud, resulting in **high end-to-end latency** and **energy consumption**.

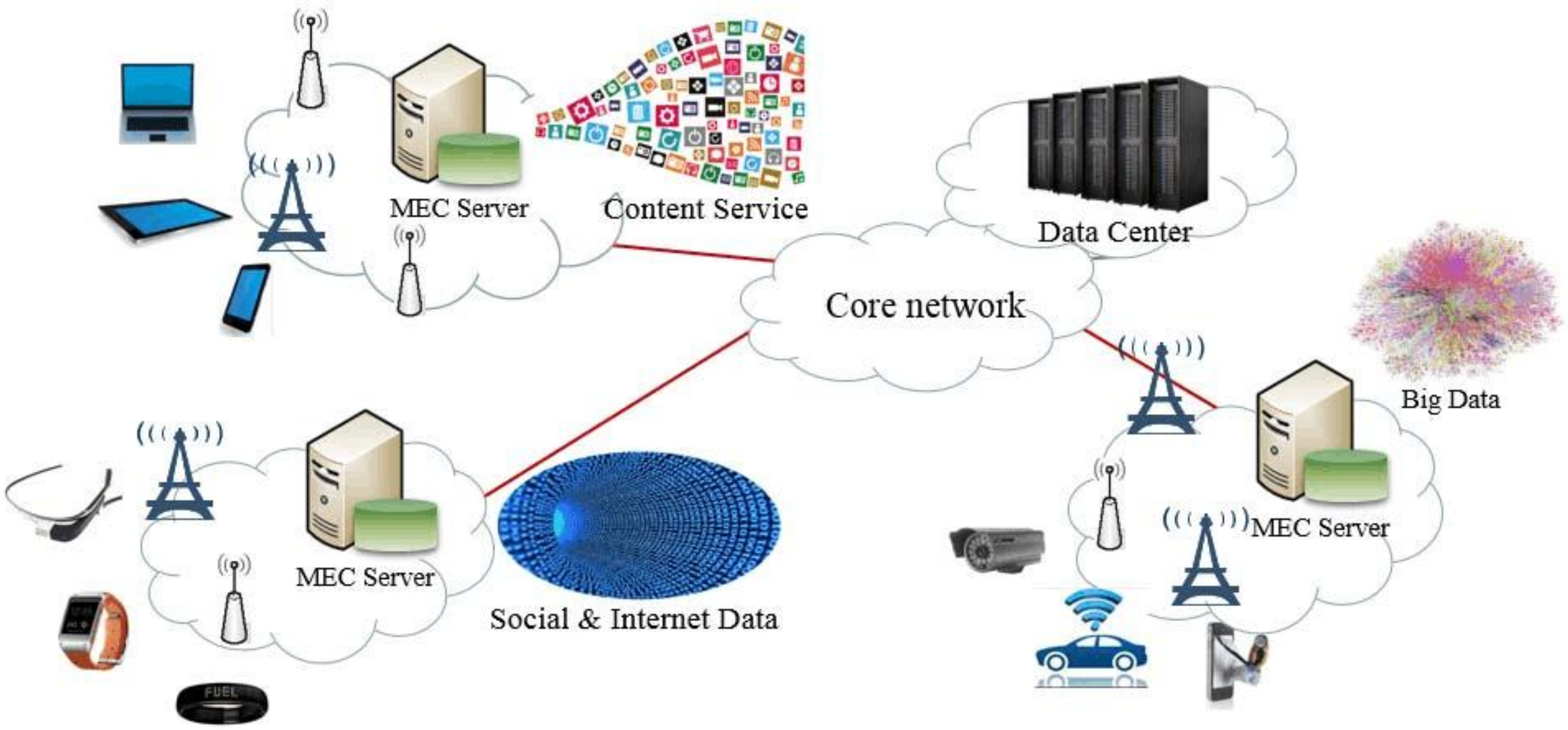
Cloud Computing Paradigm



AlexNet Performance under different bandwidth

Exploiting of Edge Computing

- By pushing the cloud capacities from the network core to the network edges (e.g. , base stations and Wi-Fi access points) in close to devices, edge computing enables low-latency and energy-efficient performance.



Existing effort of Edge Intelligence

Framework	Highlight
Neurosurgeon (ASPLOS 2017)	Deep learning model partitioning between cloud and mobile device, intermediate data offloading
Delivering Deep Learning to Mobile Devices via Offloading (SIGCOMM VR/AR Network 2017)	Offloading video input to edge server, according to network condition
DeepX (IPSN 2016)	Deep learning model are partitioned on different local processers
CoINF (arxiv 2017)	Deep learning model partitioning between smartphones and wearables

Existing effort focus on data offloading and local optimization

System Design

Our Goal

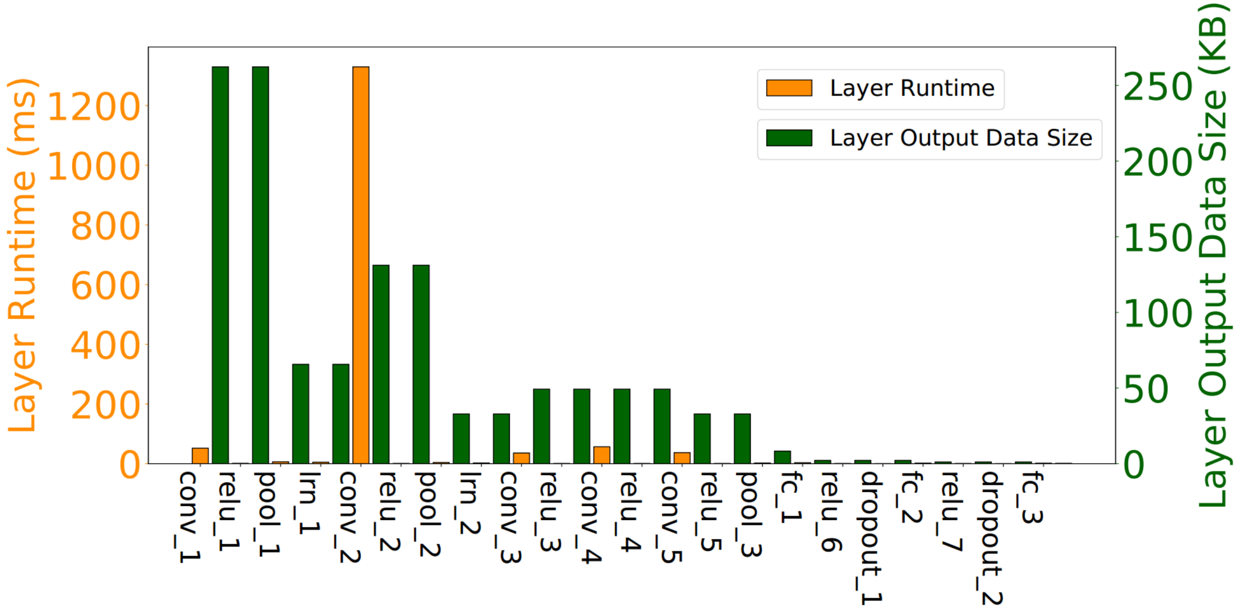
- With the collaboration between edge server and mobile device, we want to tune the latency of a deep learning model inference

Two Design Knobs

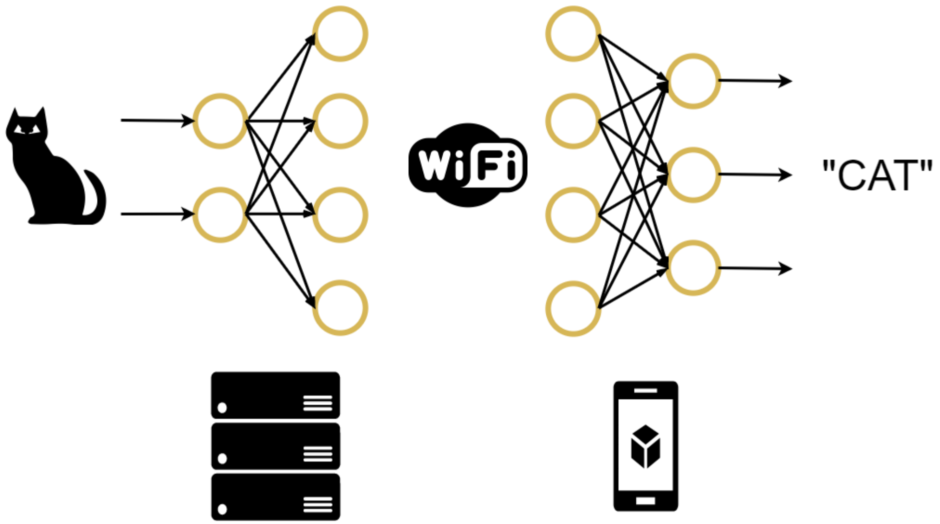
- Deep Learning Model Partition
- Deep Learning Model Right-sizing

Two Design Knobs

- Deep Learning Model Partition
- Deep Learning Model Right-sizing



AlexNet Layer Latency on Raspberry Pi & Layer Output Data Size

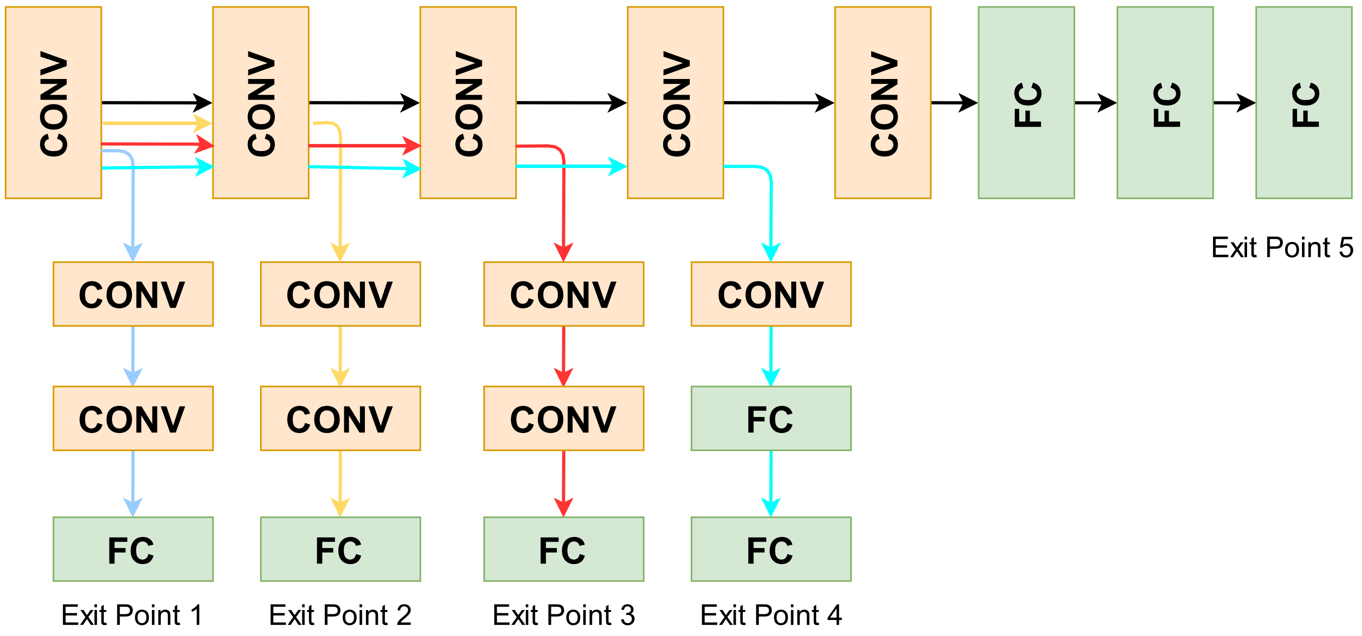


Deep Learning Model Partition[1]

[1] Kang, Yiping, et al. "Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge." *International Conference on ASPLOS* ACM, 2017:615-629.

Two Design Knobs

- Deep Learning Model Partition
- Deep Learning Model Right-sizing



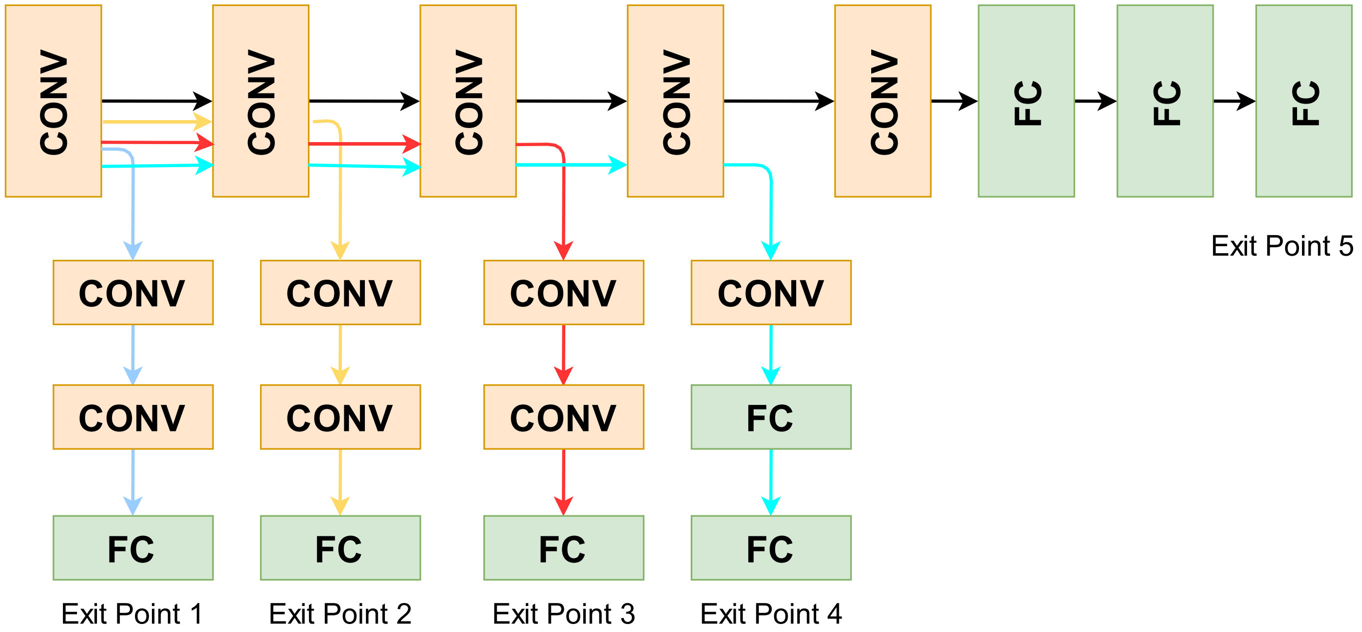
AlexNet with BranchyNet[2] Structure

[2] Teerapittayanon, Surat, B. Mcdanel, and H. T. Kung. "BranchyNet: Fast inference via early exiting from deep neural networks." ICPR IEEE, 2017:2464-2469.

A Tradeoff

A Tradeoff

- Early-exit naturally gives rise to the latency-accuracy tradeoff(i.e., early-exit harms the accuracy of the inference).



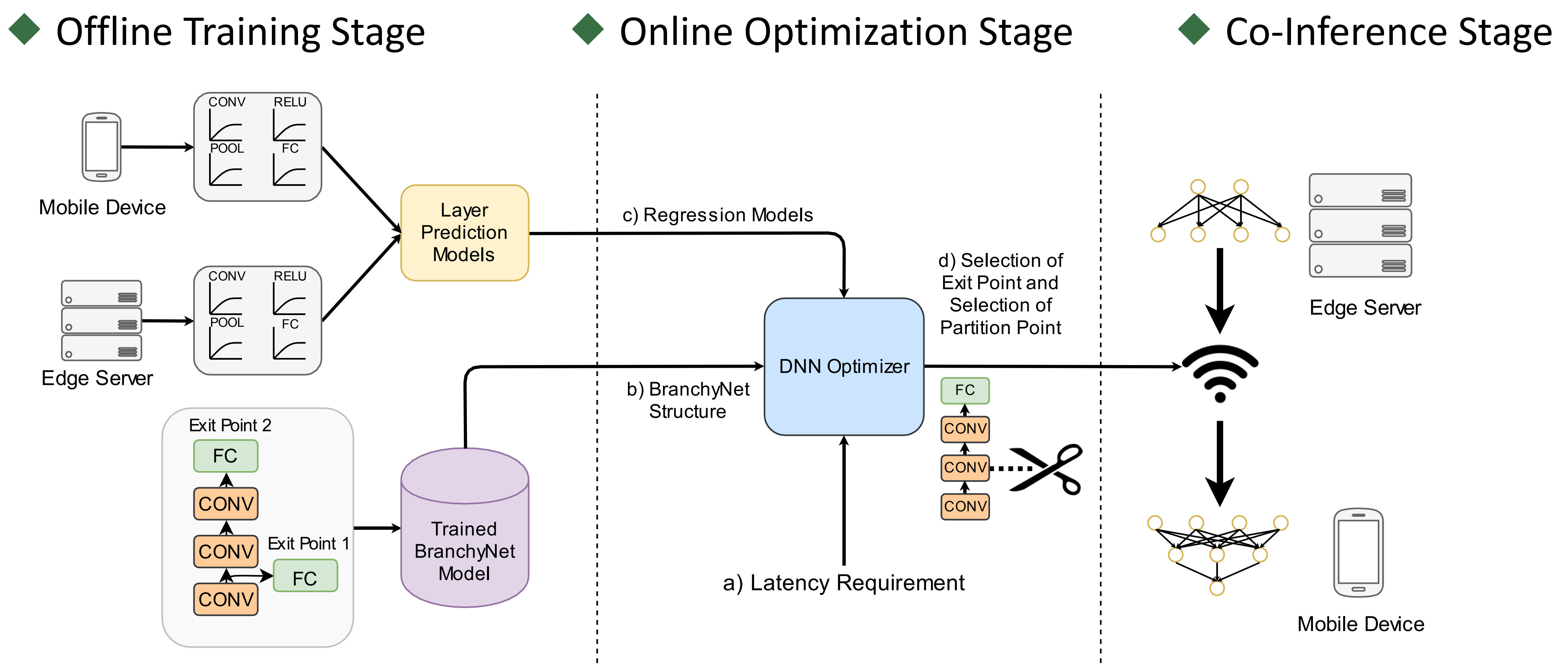
AlexNet with BranchyNet[2] Structure

[2] Teerapittayanon, Surat, B. Mcdanel, and H. T. Kung. "BranchyNet: Fast inference via early exiting from deep neural networks." ICPR IEEE, 2017:2464-2469.

Problem Definition

- For mission-critical applications that typically have a predefined latency requirement, our framework **maximizes the accuracy without violating the latency requirement**.

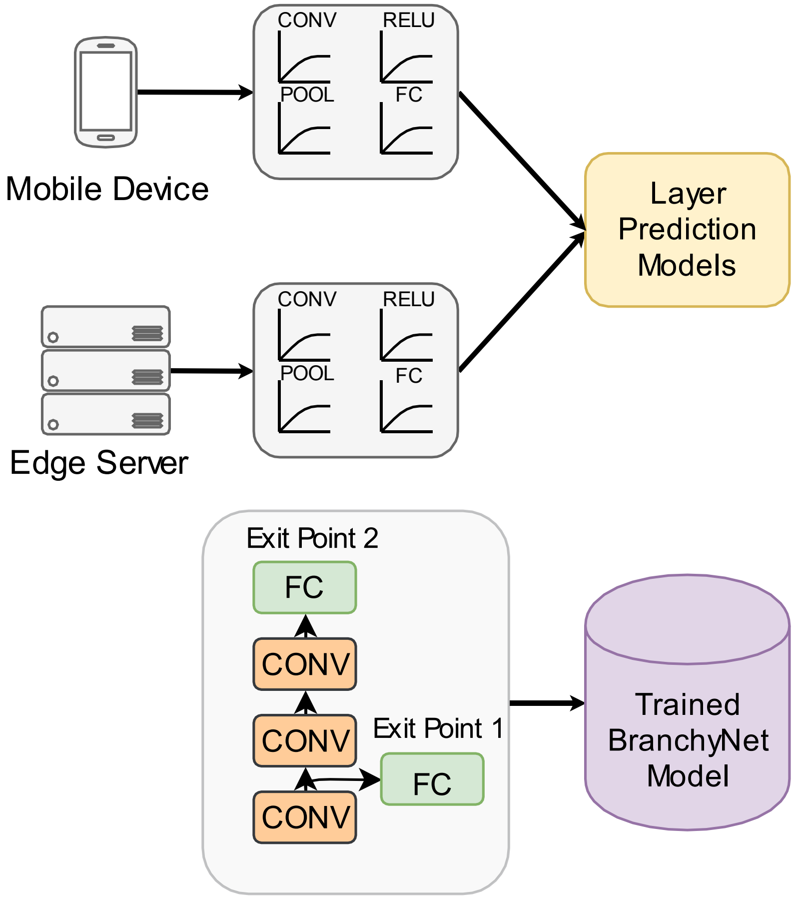
System Overview



System Overview

- Offline Training Stage
- Online Optimization Stage
- Co-Inference Stage

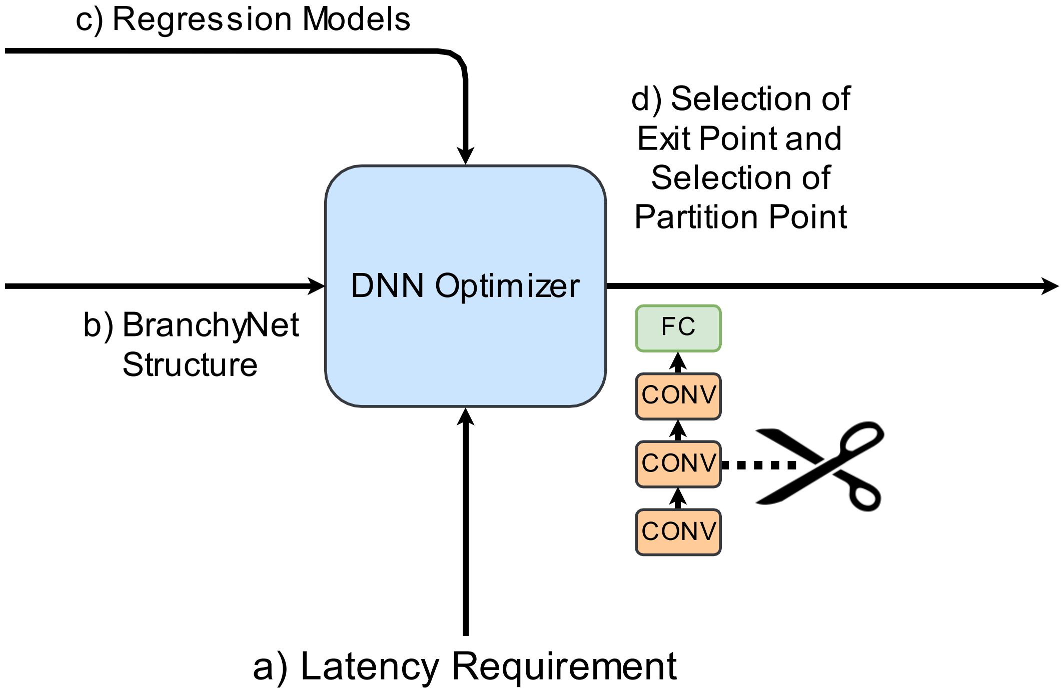
  - Training regression models for layer runtime prediction
  - Training AlexNet with BranchyNet structure



System Overview

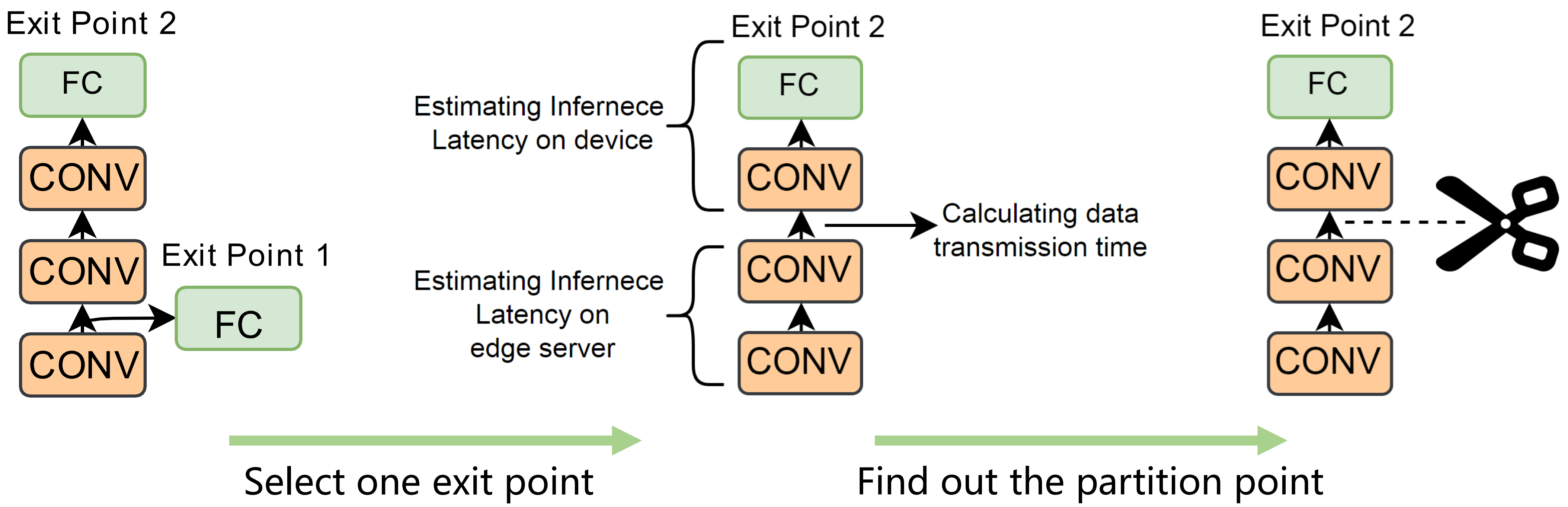
◆ Offline Training Stage ◆ **Online Optimization Stage** ◆ Co-Inference Stage

➢ Searching for exit point and partition point



System Overview

- Offline Training Stage
- Online Optimization Stage
- Co-Inference Stage



Experimental Setup

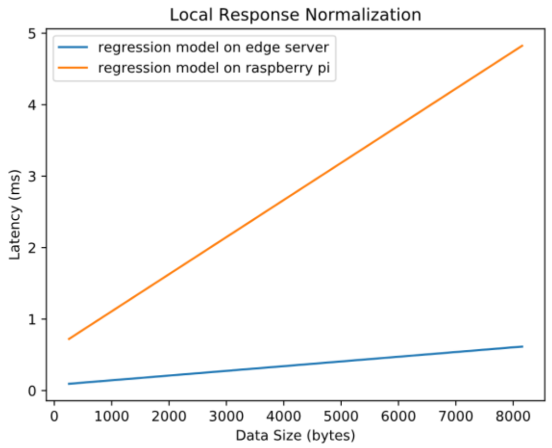
- Deep Learning Model
  - AlexNet with five exit point (built on Chainer deep learning framework)
  - Dataset: Cifar-10
  - Trained on a server with 4 Tesla P100 GPU
- Local Device: Raspberry Pi 3b
- Edge Server: A desktop PC with a quad-core Intel processor at 3.4 GHz with 8 GB of RAM

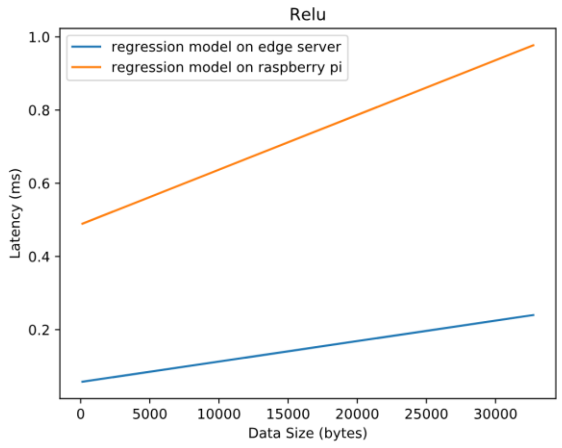
Experiments

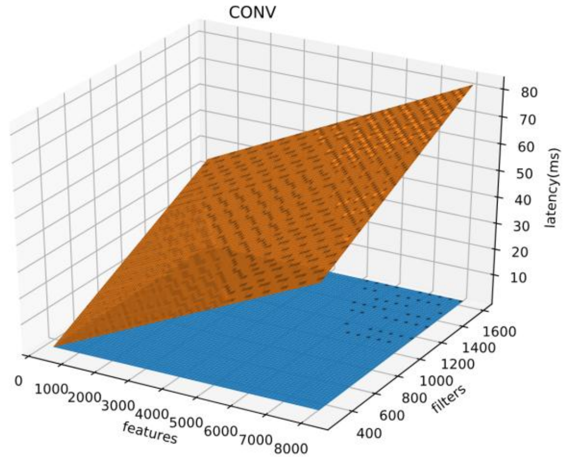
Regression Model

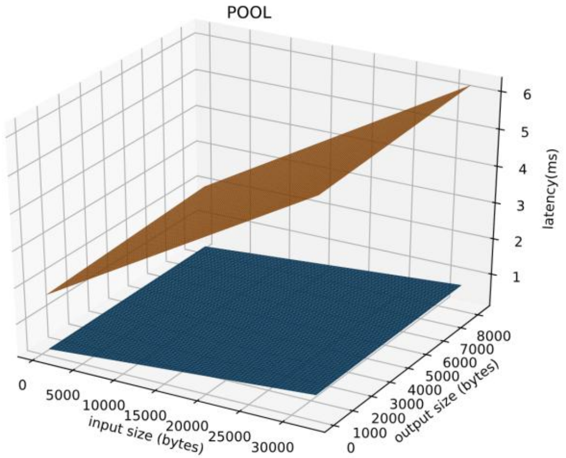
Table 1: The independent variables of regression models

Layer Type	Independent Variable
Convolution	amount of input feature maps, (filter size/stride)^2*(num of filters)
Relu	input data size
Pooling	input data size, output data size
Local Response Normalization	input data size
Dropout	input data size
Fully-Connected	input data size, output data size
Model Loading	model size









Experiments

Regression Model

Table 2: Regression Models

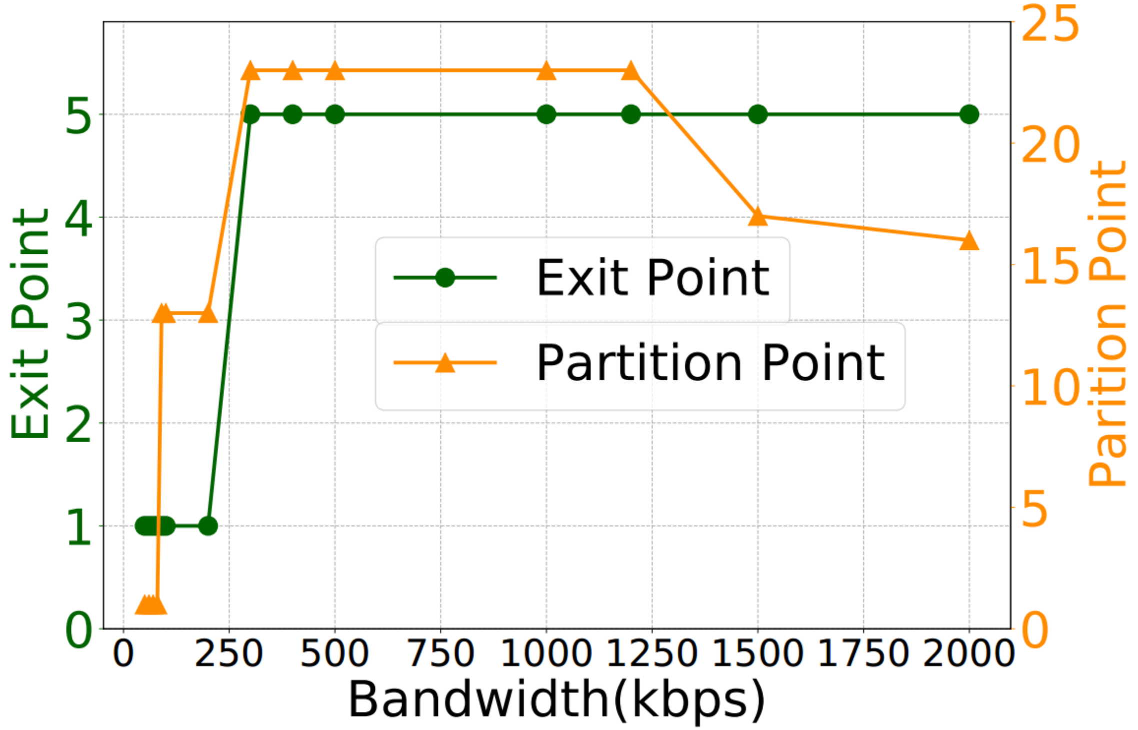
Layer	Edge Server Model	Mobile Device Model
Convolution	y = 6.03e-5 * x1 + 1.24e-4 * x2 + 1.89e-1	y = 6.13e-3 * x1 + 2.67e-2 * x2 - 9.909
Relu	y = 5.6e-6 * x + 5.69e-2	y = 1.5e-5 * x + 4.88e-1
Pooling	y = 1.63e-5 * x1 + 4.07e-6 * x2 + 2.11e-1	y = 1.33e-4 * x1 + 3.31e-5 * x2 + 1.657
Local Response Normalization	y = 6.59e-5 * x + 7.80e-2	y = 5.19e-4 * x+ 5.89e-1
Dropout	y = 5.23e-6 * x+ 4.64e-3	y = 2.34e-6 * x+ 0.0525
Fully-Connected	y = 1.07e-4 * x1 - 1.83e-4 * x2 + 0.164	y = 9.18e-4 * x1 + 3.99e-3 * x2 + 1.169
Model Loading	y = 1.33e-6 * x + 2.182	y = 4.49e-6 * x + 842.136

Experiments

Result

- Selection under different bandwidths

The higher bandwidth leads to higher accuracy

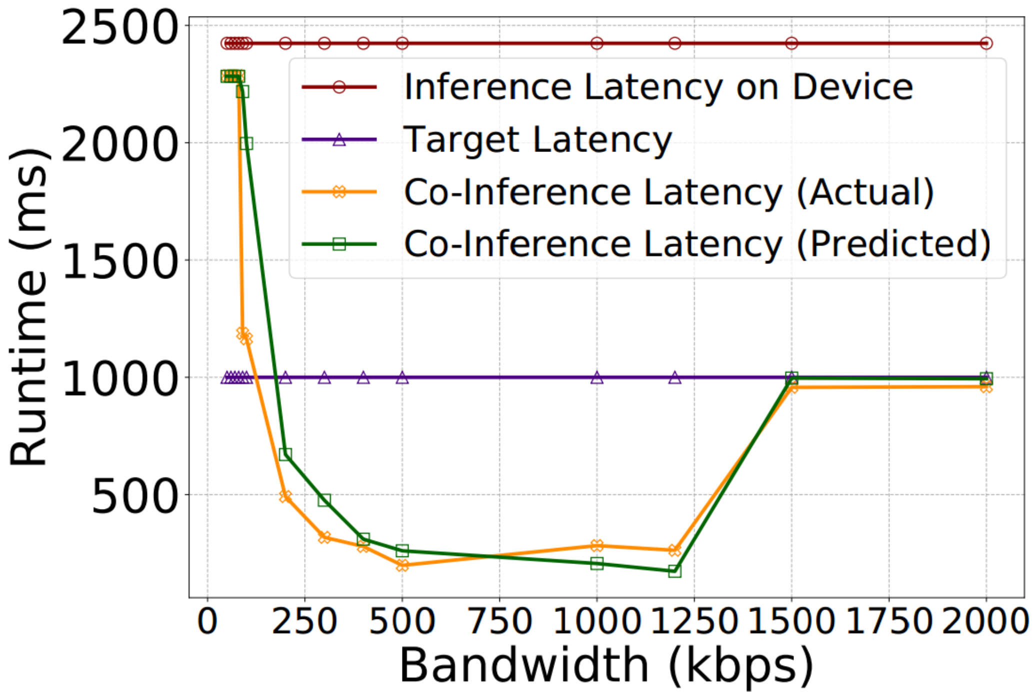


(a) Selection under different bandwidths

Experiments

- Inference Latency under different bandwidths

Our proposed regression-based latency approach can well estimate the actual deep learning model runtime latency.

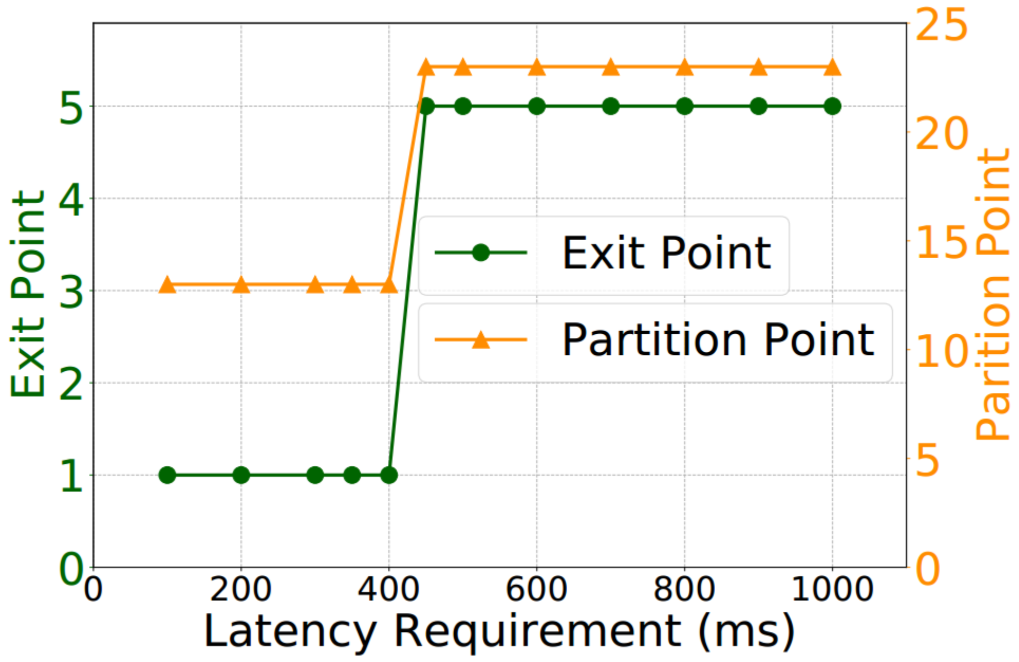


(b) Model runtime under different bandwidths

Experiments

- Selection under different latency requirements

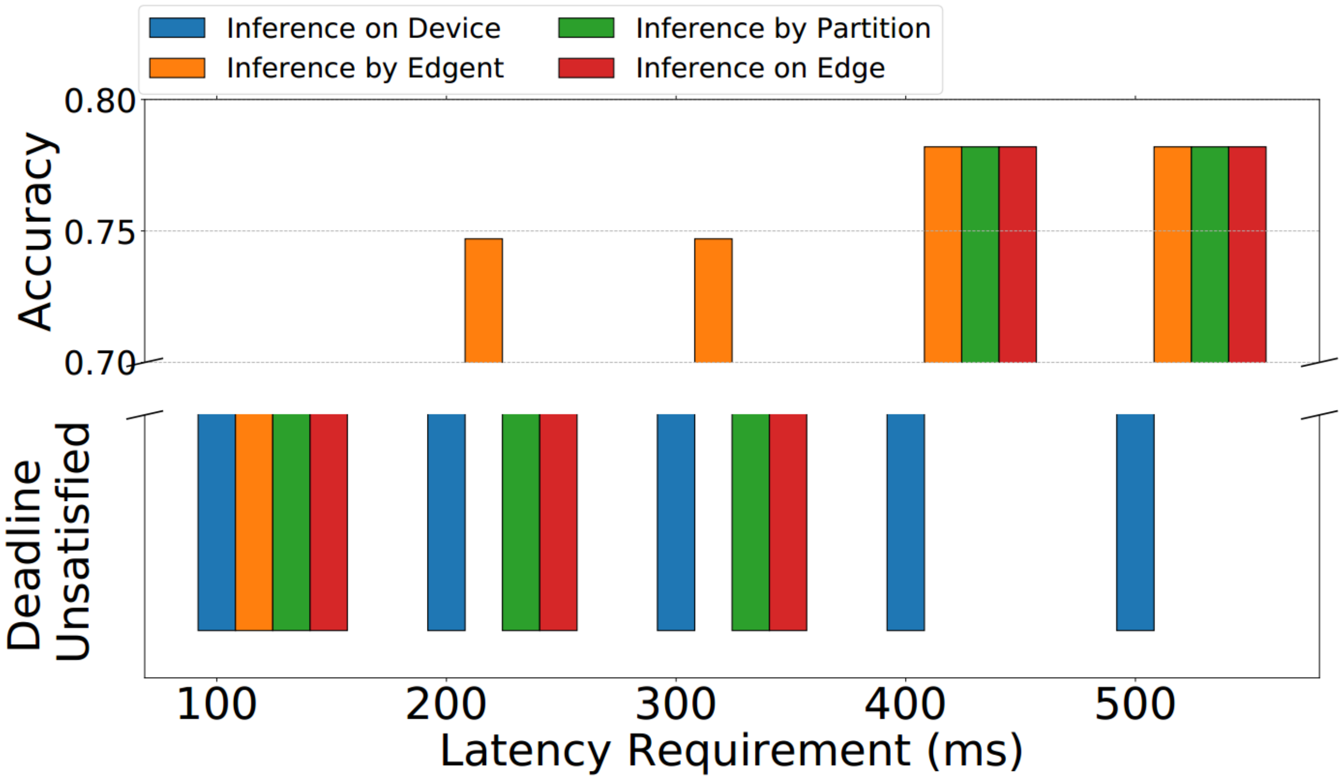
A larger latency goal gives more room for accuracy improvement



(c) Selection under different latency requirements

Experiments

- Comparison with other methods



The inference accuracy comparison under different latency requirement

Key Take-Aways

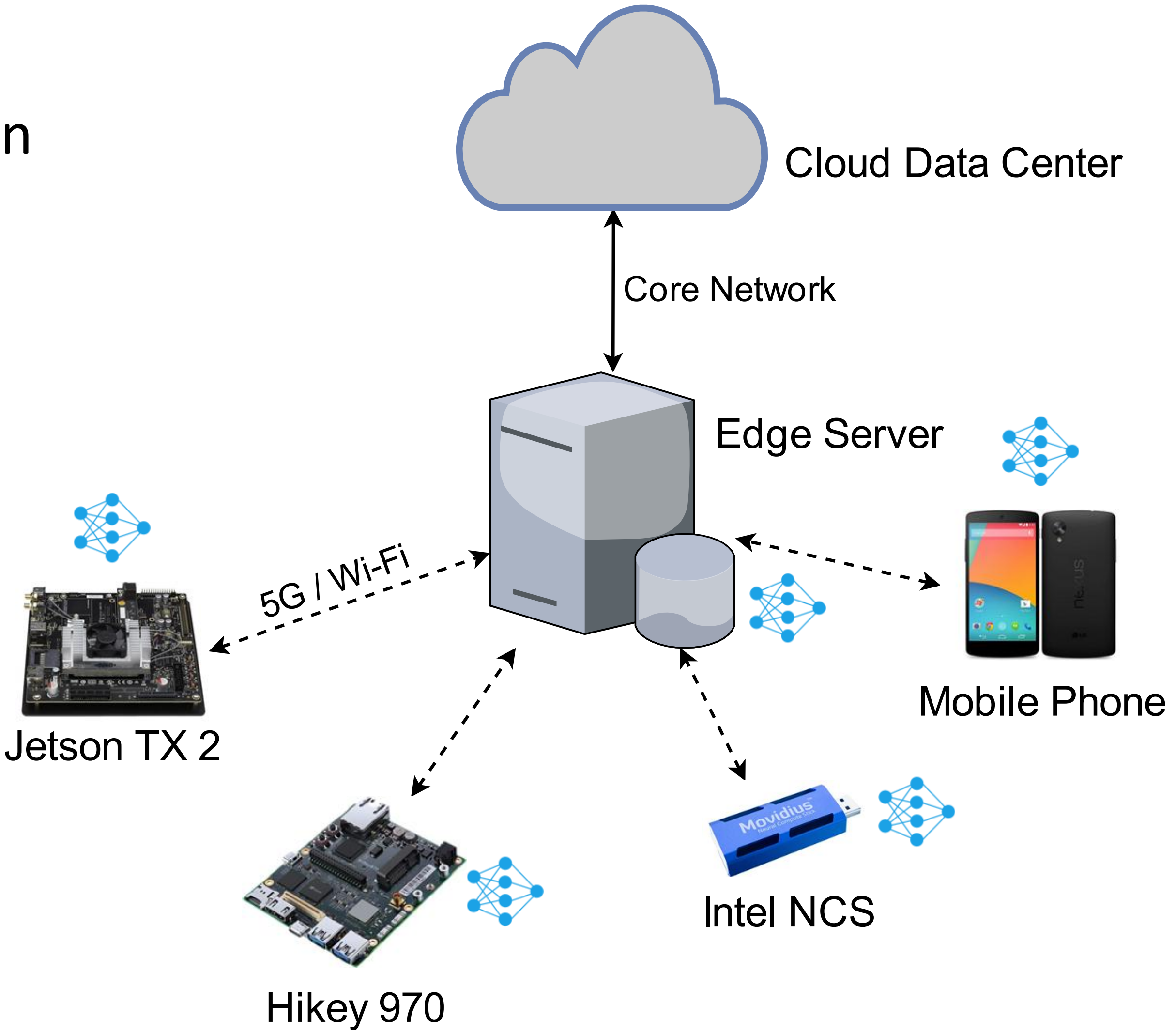
On demand accelerating deep learning model inference through device-edge synergy

Deep Learning Model Partition
Deep Learning Model Right-sizing

Implementation and evaluations demonstrate effectiveness of our framework

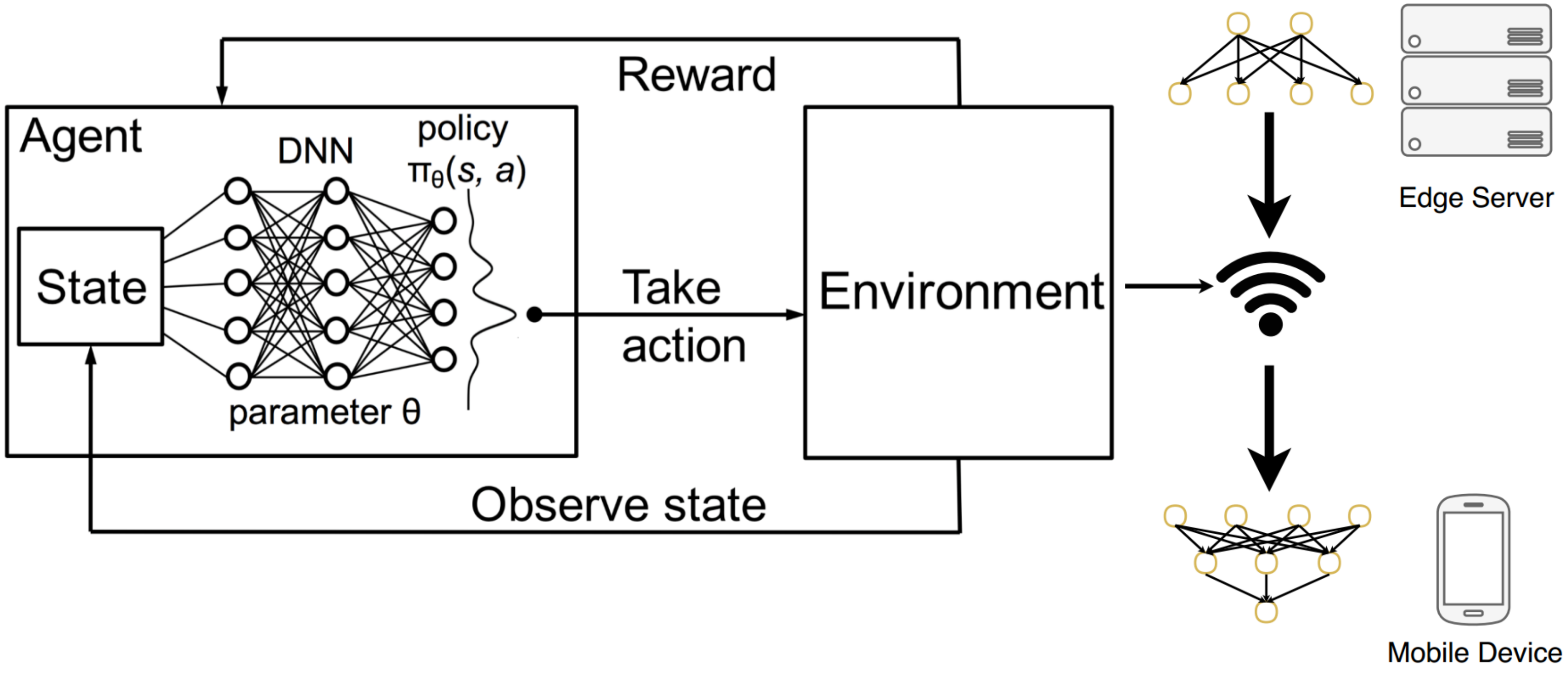
Future Work

- More Devices
- Energy Consumption



Future Work

- Deep Reinforcement Learning Technique



Deep Reinforcement Learning for Model Partition

Thank you

✉ **Contact:**
lien5@mail2.sysu.edu.cn
zhouzhi9@mail.sysu.edu.cn
chenxu35@mail.sysu.edu.cn