Accelerate Cooperative Deep Inference via Layer-wise Processing Schedule Optimization

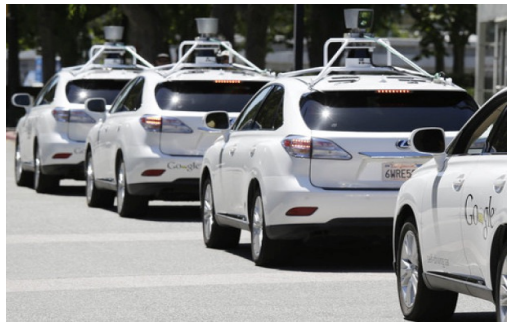
Ning Wang, Yunbin Duan, and Jie Wu

Background

- Internet-of-Things (IoT) devices are **pervasive**. We want to run Deep Learning (DL) applications **everywhere**! Not just in data center.

smartphone

drone

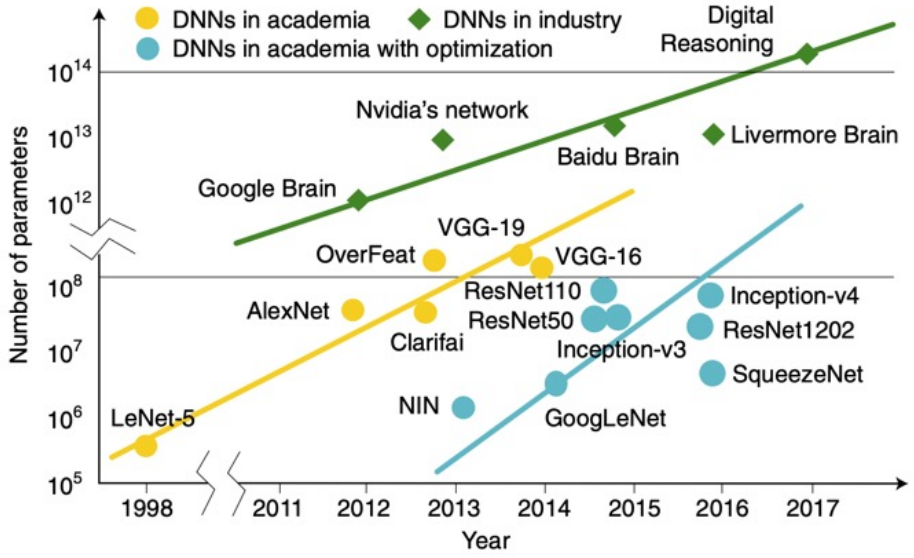


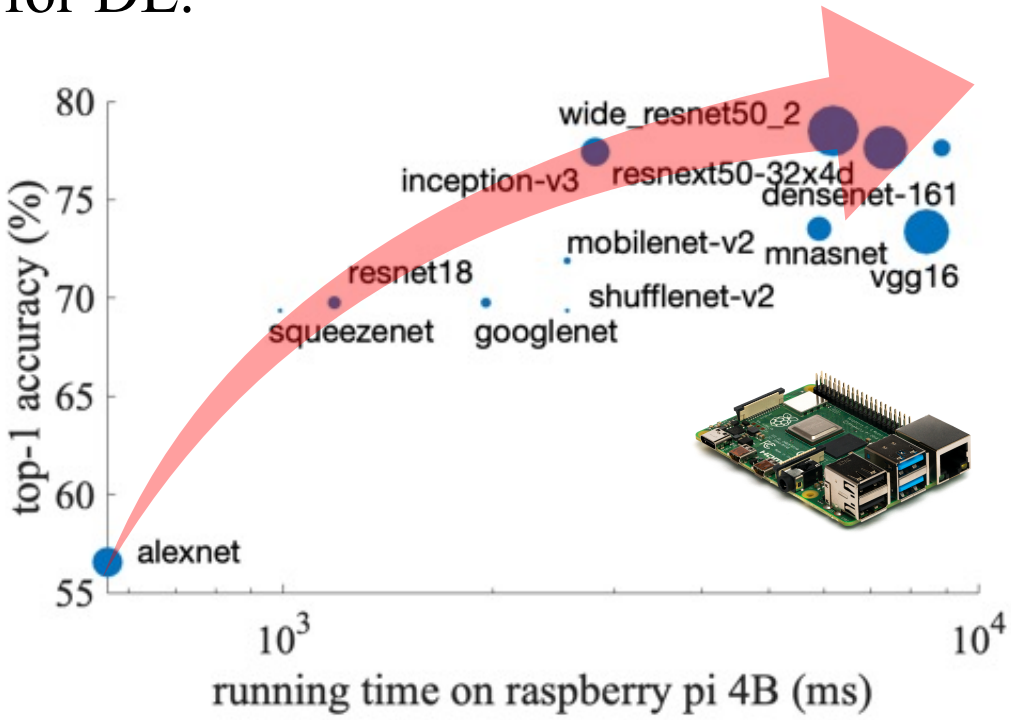
autonomous driving

VR/AR

Background

- Deep Neural Networks (DNN) complexity vs IoT speed.
  - IoT devices are not powerful enough for DL.

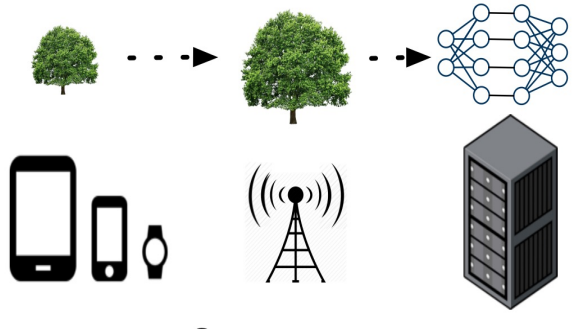




*Xu, Xiaowei, et al. "Scaling for edge inference of deep neural networks." Nature Electronics 1.4 (2018)

Objective

- **Goal:** Minimizing DL inference latency
- **Method:** Computation Offloading via edge computing.
  - IoT end device: environment awareness
  - edge server: accurate event inference



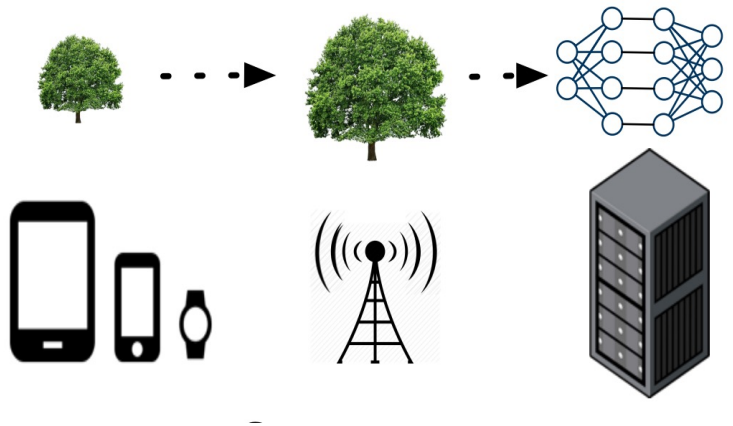
Computation Offloading

Vehicle to Everything (V2X) & 5G

https://www.autonomousvehicletech.com/articles/1129-cohda-announces-c-v2x-software-development-kit

Related Works

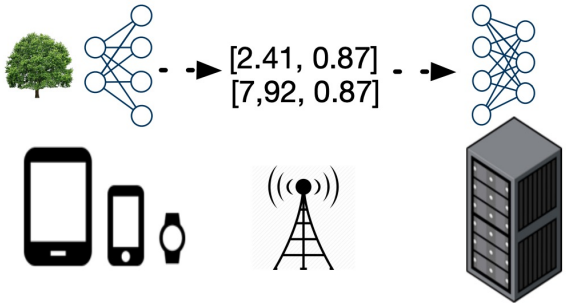
- Computation offloading introduces extra latency
  - Inference completion time = data transmission delay + data processing delay
  - The network may not be that fast and communication delay cannot be ignored*.



*Liu et al. A Comparison of Communication Mechanisms in Vehicular Edge Computing. In 3rd USENIX Workshop on Hot Topics in Edge Computing (HotEdge 20).

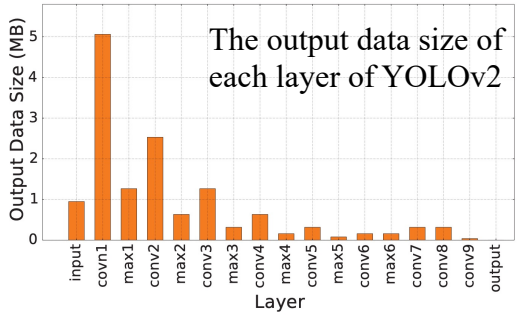
Related Works

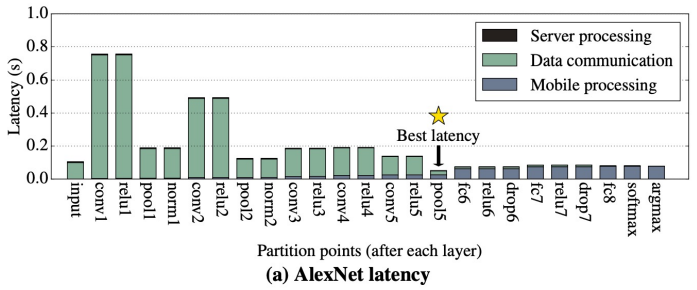
- Cooperative Deep Inference
  - Stage 1: local computation at IoT device
  - Stage 2: intermediate result transmission
  - Stage 3: remote server computation

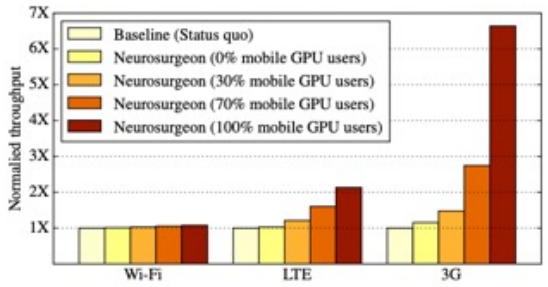


Collaborative deep inference

- Rationale
  - Intermediate DNN layers output is significantly smaller than that of raw input data



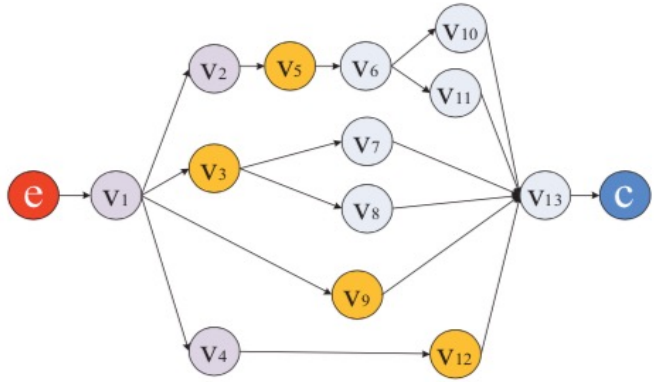




Kang, Yiping, et al. "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge." ACM SIGARCH 2017.

Challenges

- DNN Computation Model
  - abstracted as a **Directed acyclic graph (DAG)*** denoted as G(V, E), where a vertex $v_i \in V$ represents a layer and a link $e_{ij} = (v_i , v_j) \in E$ represents the processing dependency relationship between two layers.



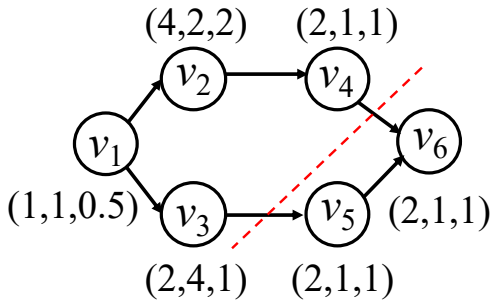
Inception

- Cooperative Deep Inference Optimization
  - Task assignment (i.e., how to partition a DNN)
    - abstracted as a cut in DAG
  - Scheduling (i.e., how to process vertices)
    - **Our major contribution!**

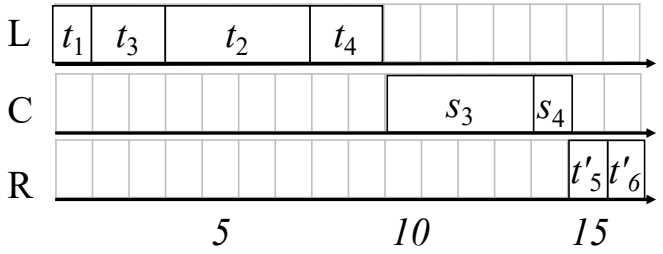
*Recurrent Neural Networks (RNN) models such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) are out of the scope of this paper.

Why Is Scheduling Important?

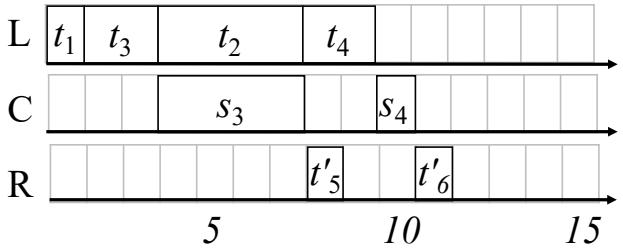
- A Toy Example
  - The local processing time, transmission time, and remote processing time is denoted as a tuple.



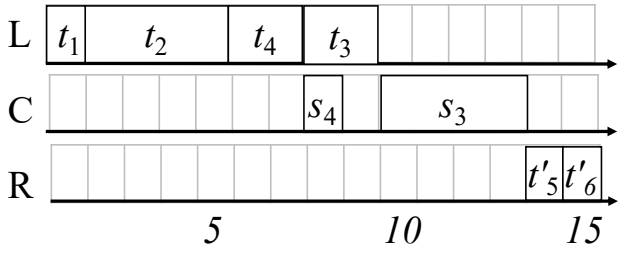
(a) Toy DNN



(b) No schedule



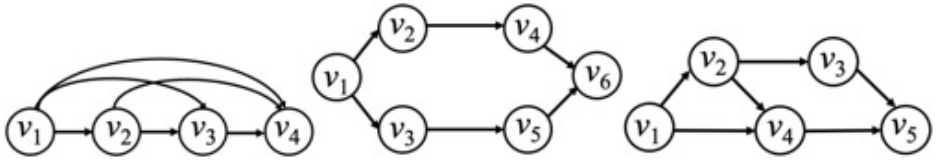
(c) Schedule 1



(d) Schedule 2

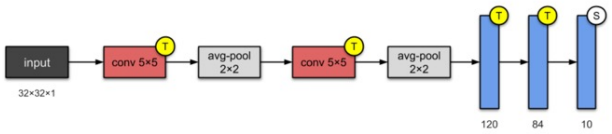
Solutions in Three Different DNNs

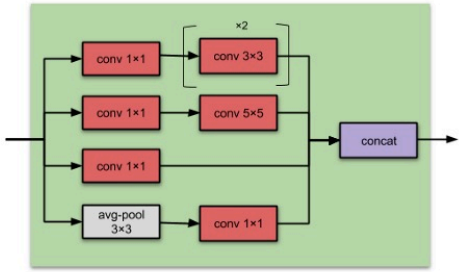
- We summarize state-of-the-art DNNs into three categories.
  - line, multi-path, and general DAG

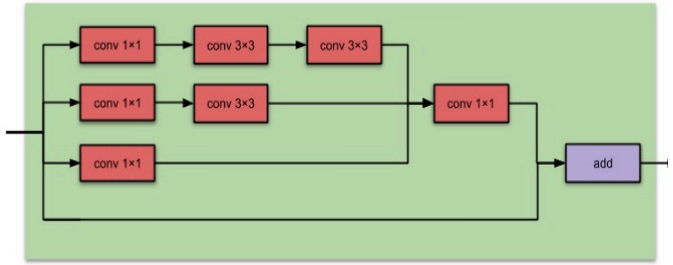


(a) line (b) multi-path (c) DAG

- Examples
  - LeNet, Inception and Inception-ResNet







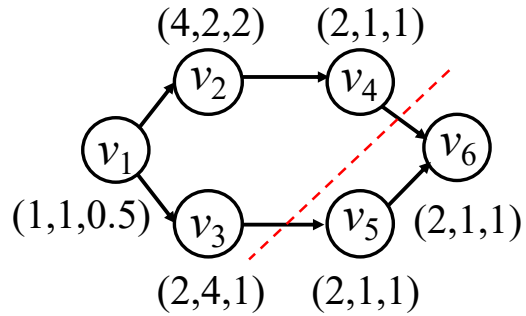
https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

Multi-Path DNNs

❖ Problem Hardness

❑ Line/Single-path

➢ Straightforward solution, even without a given cut



❑ Multi-path (**Theorem 1: NP-hard**)

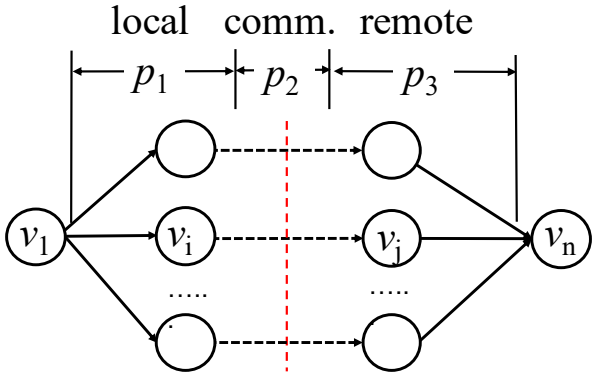
➢ Path: a sequence of layers which have sequential dependency relationship (except input and output vertices)

➢ Non-overlaps among paths (e.g., v_2-v_4, v_3-v_5)

❑ **Theorem 2:** In multi-path DNNs, the optimal schedule can be achieved via the non-preemptive path-based schedule.

Multi-Path DNNs

- Extended Johnson (EJ) Algorithm
  - Path $p(i)$ in three stages $p_1(i)$, $p_2(i)$, $p_3(i)$
  - Dividing paths into H and L (Linear)
    - E.g., H = {1}, L = {3, 4, 2}

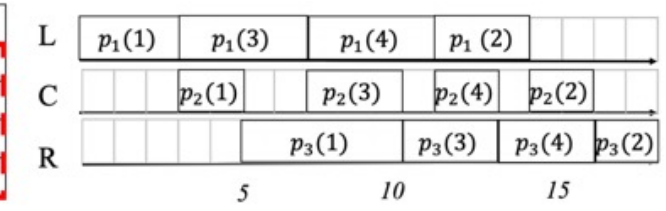


Algorithm 1 Extended Johnson (EJ) Algorithm

Input: $G(V, E)$, X, t_i and t'_i, $\forall v_i$
Output: The offloading schedule σ

1: $H \leftarrow L \leftarrow \emptyset$
2: **for** $i = 1$ to m **do**
3: **if** $p_1(i) + p_2(i) \leq p_2(i) + p_3(i)$ **then**
4: $H = H \cup p(i)$
5: **else**
6: $L = L \cup p(i)$
7: Sort H increasingly based on $p_1(i) + p_2(i)$
8: Sort L decreasingly based on $p_2(i) + p_3(i)$
9: Concatenate H and L to obtain σ

Path	$p_1(i)$	$p_2(i)$	$p_3(i)$
$i = 1$	3	2	5
$i = 2$	3	2	2
$i = 3$	4	3	3
$i = 4$	4	2	3



(a) EJ Algorithm

Multi-Path DNNs

- Extended Johnson (EJ) Algorithm
  - Theorem 3*: If stage 2 is dominated by either stage 1 or 3, $\max\{\min p_1(i), \min p_3(i)\} \geq \max p_2(i)$, EJ is optimal.

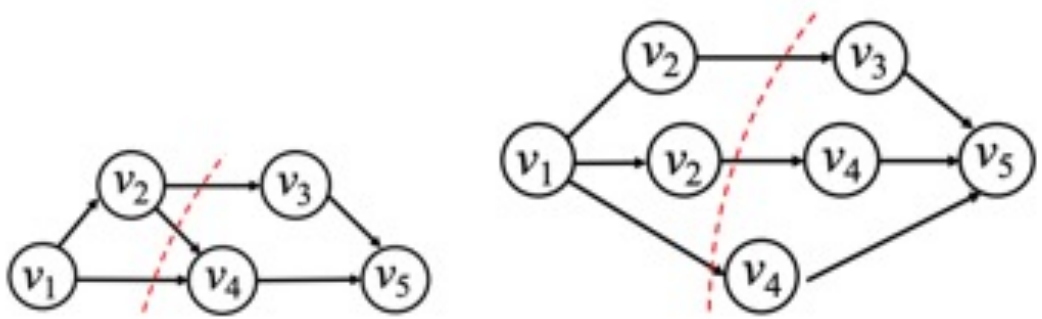
Path	$p_1(i)$	$p_2(i)$	$p_3(i)$
$i = 1$	3	2	5
$i = 2$	3	2	2
$i = 3$	4	3	3
$i = 4$	4	2	3

  - Theorem 4^: If Theorem 3 fails, EJ still achieves an approximation ratio of 5/3.

*Chen et al, A new heuristic for three-machine flow shop scheduling, OR, 1996.
^Framinan et al, A review and classification of heuristics for permutation flow shop scheduling with makespan objectives, JORS, 2004.

DAG DNNs

- Graph Conversion Algorithm
  - Convert DAG DNNs to multi-path DNNs
  - Replicate nodes via **join** and **fork operations** until it becomes a multi-path DNN.
    - Replicated nodes only execute once (the first time)



(a) before Conversion, G (b) after Conversion, G'

Experiments

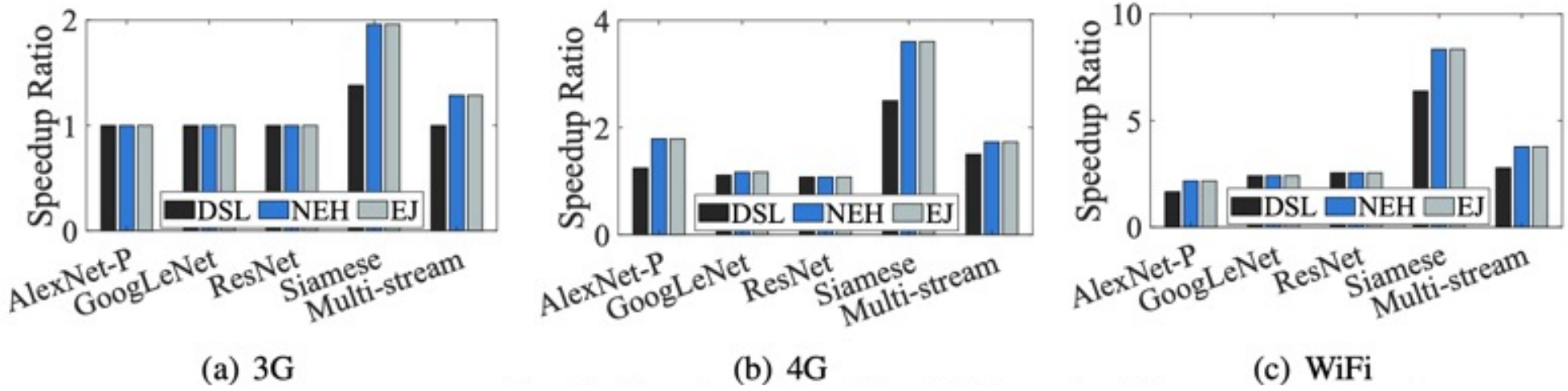
- **Testbed:**
  - IoT device: Raspberry Pi 4 model B
  - Server: A desktop in our lab which has a six-core CPU (i7-8700) @ 3.20GHz, a GTX 1080 GPU, and 32 GB RAM
- **Experiment results:**
  - LO algorithm: run on the Raspberry Pi; RO algorithm: run DNN on the server

Model	LO	3G		4G		WiFi	
		RO	Our	RO	Our	RO	Our
AlexNet-P	406	4422	**406**	877	**255**	337	**187**
GoogLeNet	848	4475	**848**	879	**728**	352	**352**
ResNet18	871	4463	**871**	946	**810**	341	**341**
Siamese	3919	8783	**1998**	1702	**1087**	613	**469**
Multi-stream	1198	13146	**931**	2513	**692**	891	**317**

Experiments

❖Experiment results:

- ❑ DSL: best existing work (no scheduling optimization)



Q&A

- Thanks!

- Contact Information
  - wangn@rowan.edu