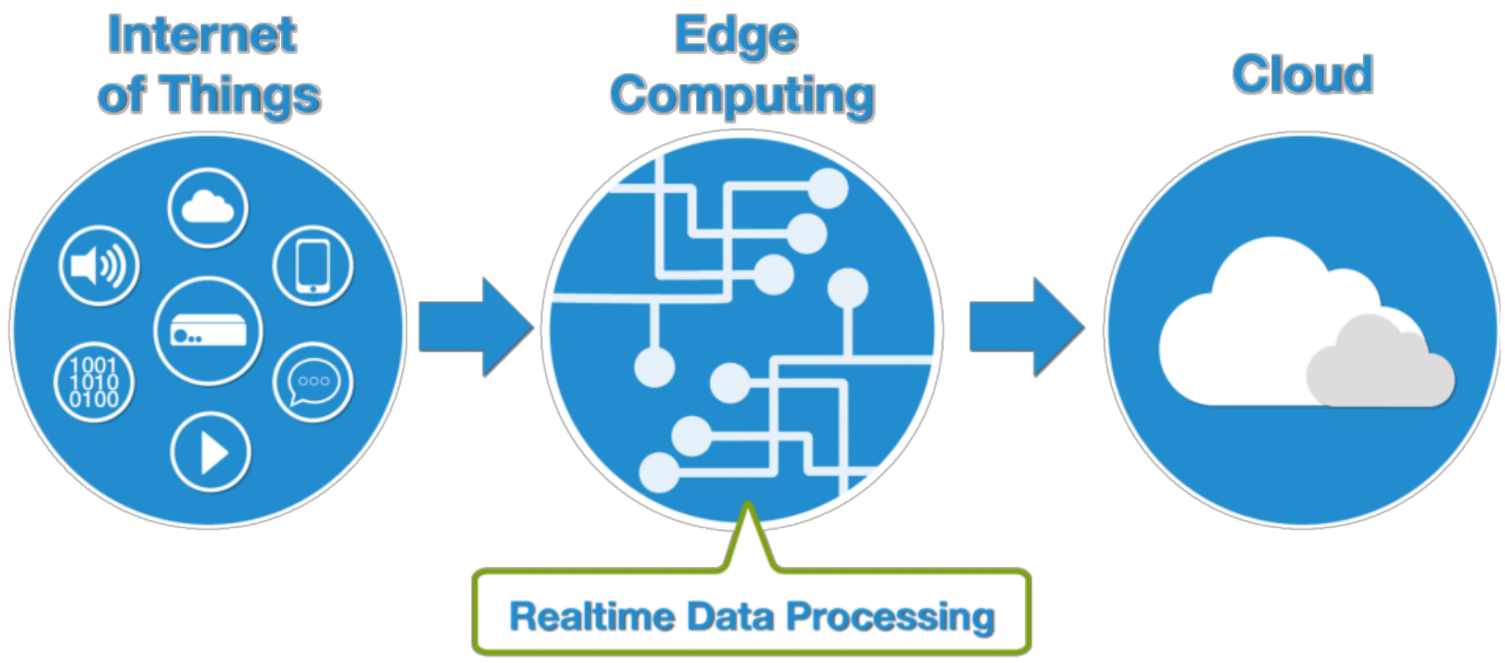


IONN: Incremental Offloading of Neural Network Computations from Mobile Devices to Edge

Hyuk-Jin Jeong et al. ACM SoCC ‘18

Ashkan Yousefpour

Computer Science
University of Texas at Dallas
CS7301-003 Fall 2018

November, 2018

Image borrowed form https://www.talari.com/glossary_faq/edge-computing/

Outline

- Introduction
- Motivation
- System Model
- DNN Partitioning
- IONN Architecture
- Evaluation Results

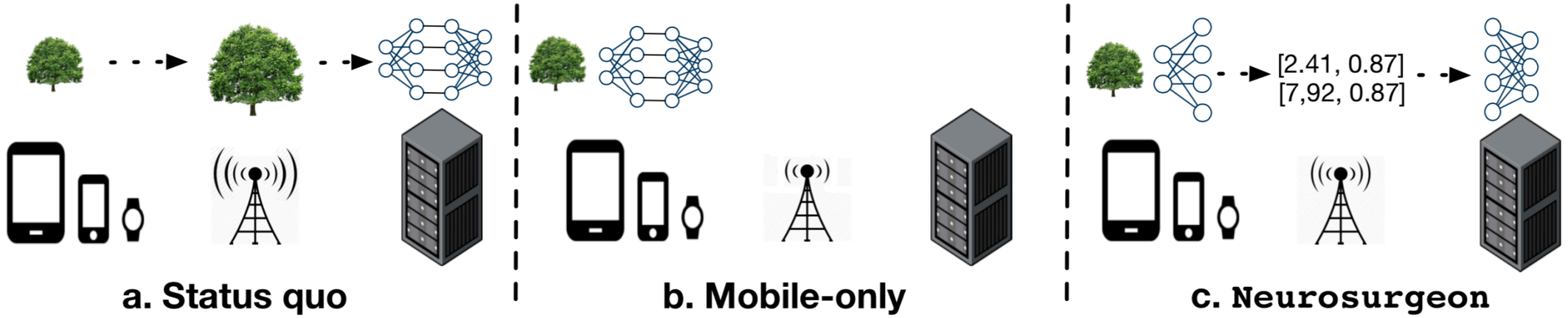
Introduction

- Deep Neural Network (DNN) models are compute intensive
  - On IoT Devices? Energy and latency
  - In the Cloud? Yes. Cloud ML APIs: Google, HP, AWS, etc.
    - Hybrid! Ok, install
- Pre-install in the cloud?
  - YES! But what about edge computing?
- Pre-install on edge servers?
  - NO! How about on-demand install?
    - NO! large upload delay

Introduction

- Idea: **incremental** offloading of DNN model
  - 1-partition >>>> 2-order >>>> 3-offload
- Edge server **incrementally** builds DNN
- Client can offload DNN query before the whole model is uploaded
- Server is faster than client

Prior Work



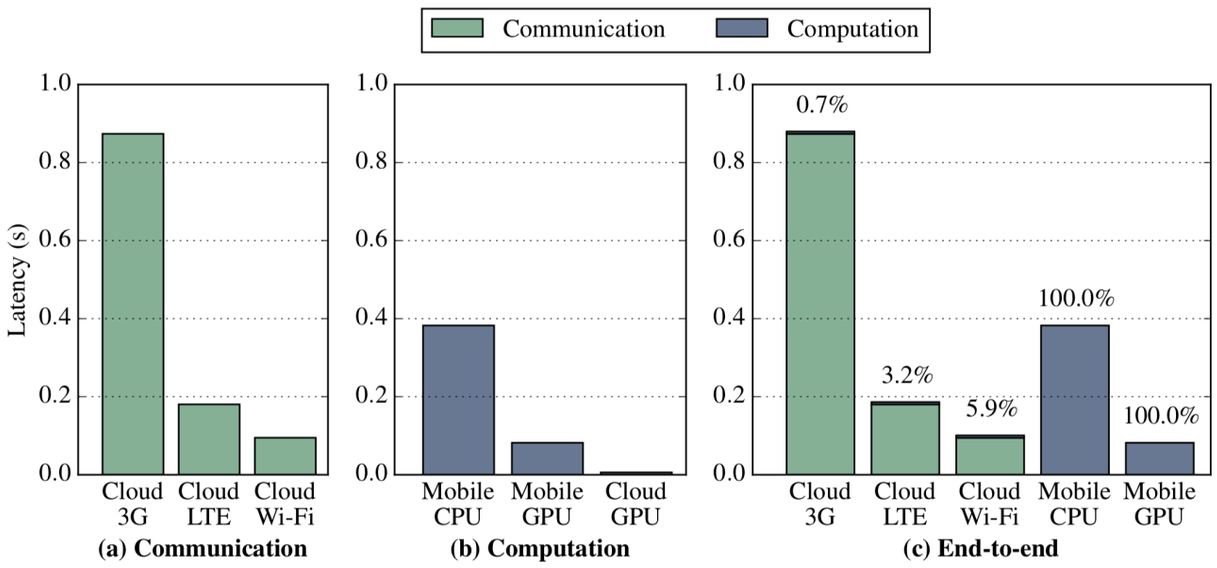


Figure 3: Latency breakdown for AlexNet (image classification). The cloud-only approach is often slower than mobile execution due to the high data transfer overhead.

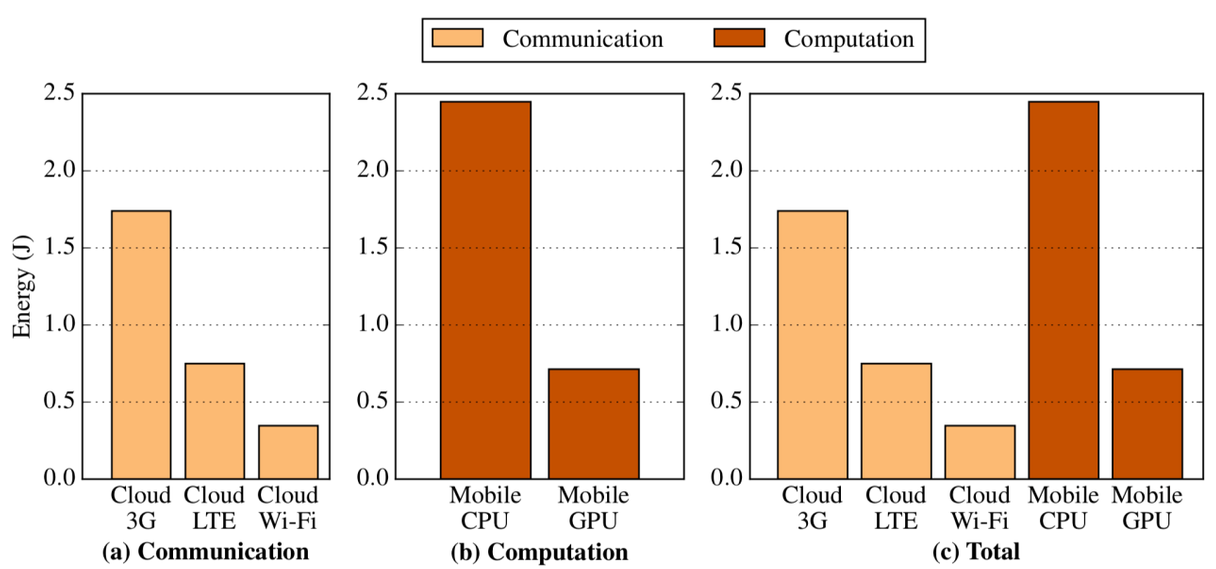
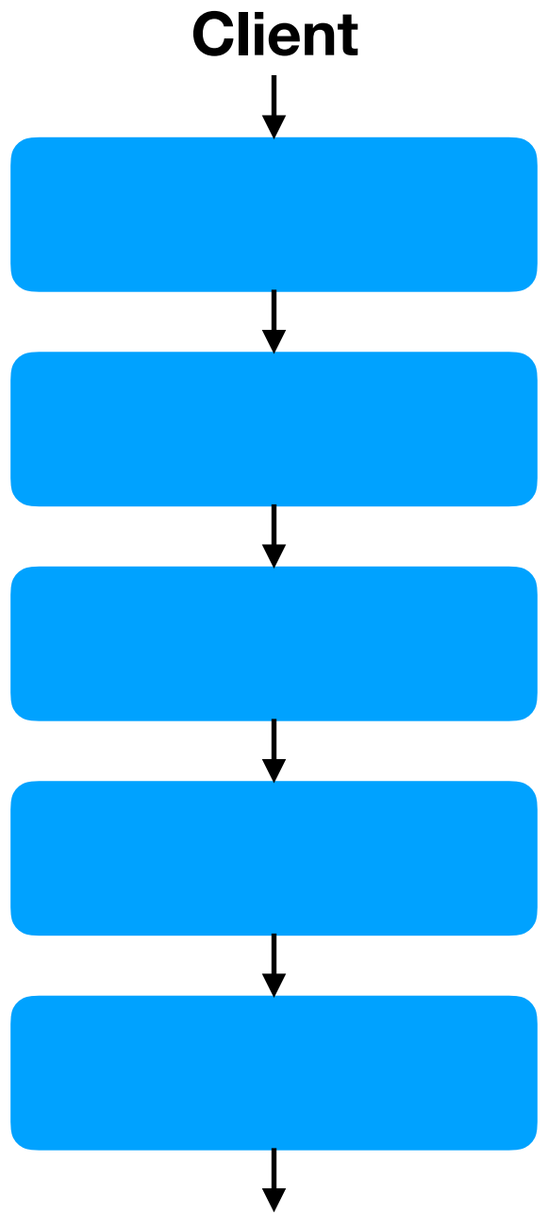


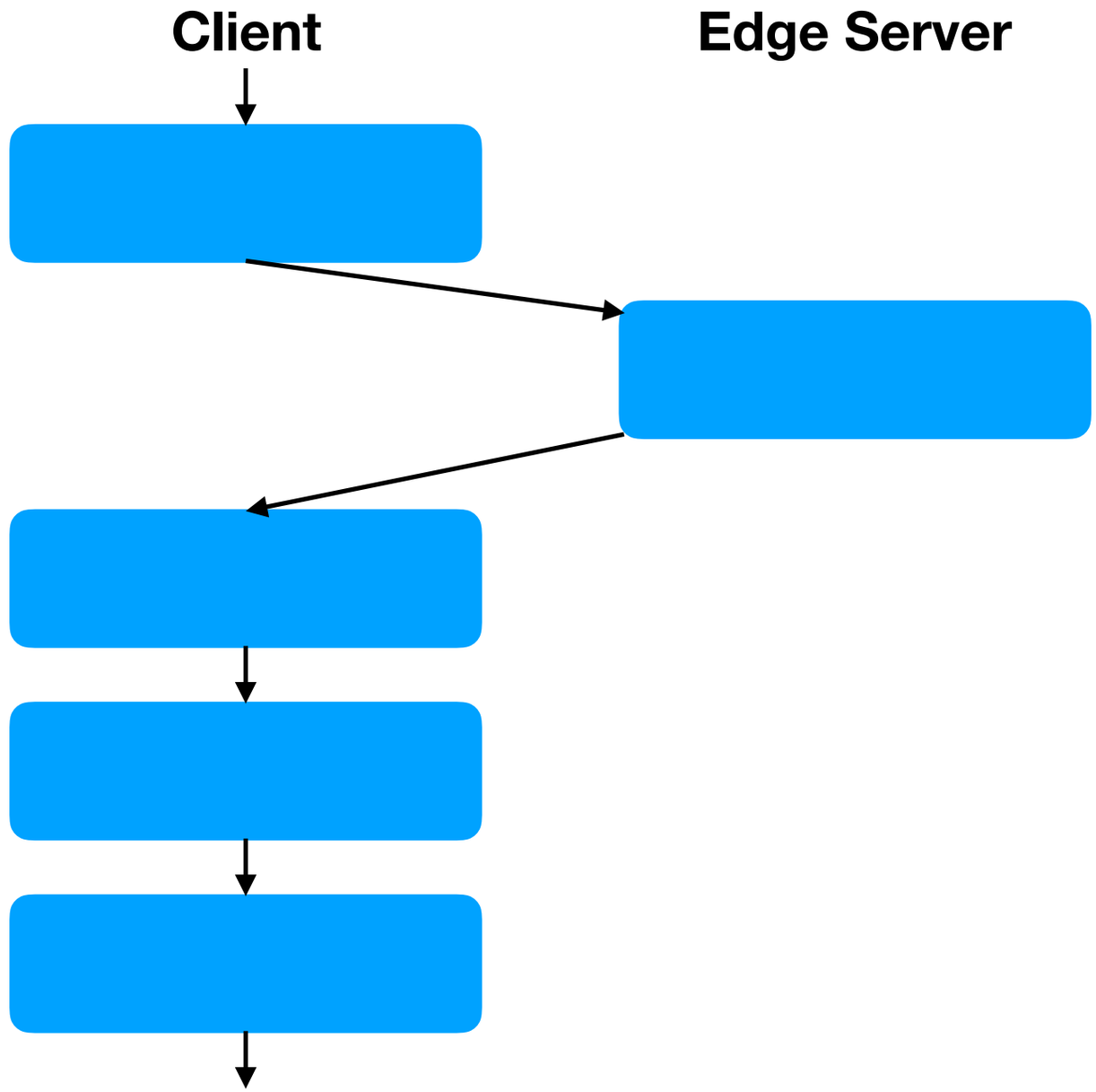
Figure 4: Mobile energy breakdown for AlexNet (image classification). Mobile device consumes more energy transferring data via LTE and 3G than computing locally on the GPU.

Kang, Yiping, Johann Hauswald, Cao Gao, Austin Rovinski, Trevor Mudge, Jason Mars, and Lingjia Tang.
"Neurosurgeon: Collaborative intelligence between the cloud and mobile edge." *ACM SIGPLAN Notices* 52, no. 4 (2017): 615-629.

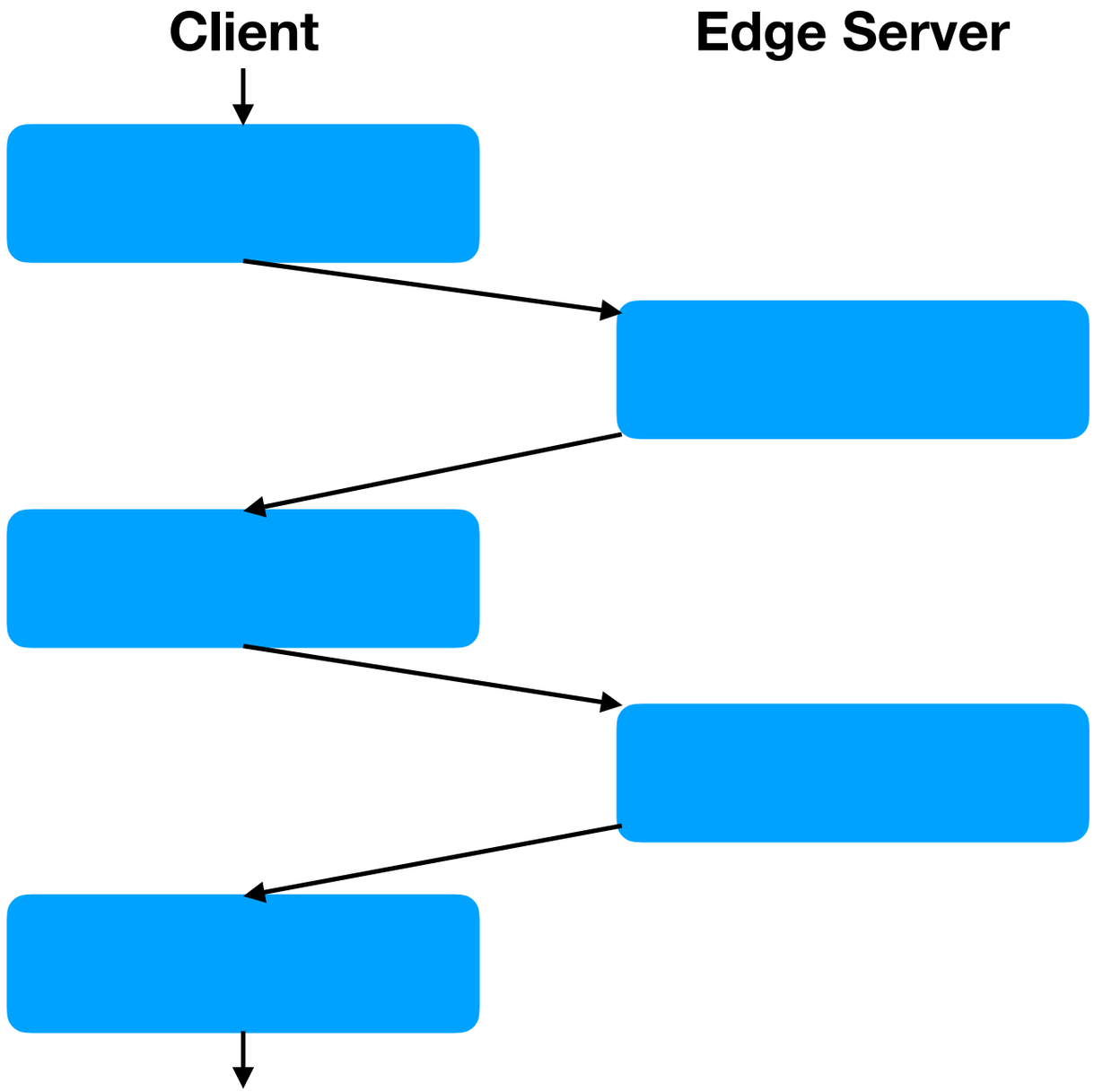
Client Server Collaboration



Client Server Collaboration



Client Server Collaboration



Motivation

- Smart glass cognitive assistance (eye-sight)

Subway station

Client

1.3 sec/query
(ARM CPU)

Motivation

- Client: Board Odroid XU4 with (2.0GHz/1.5GHz 4 cores) and 2GB memory

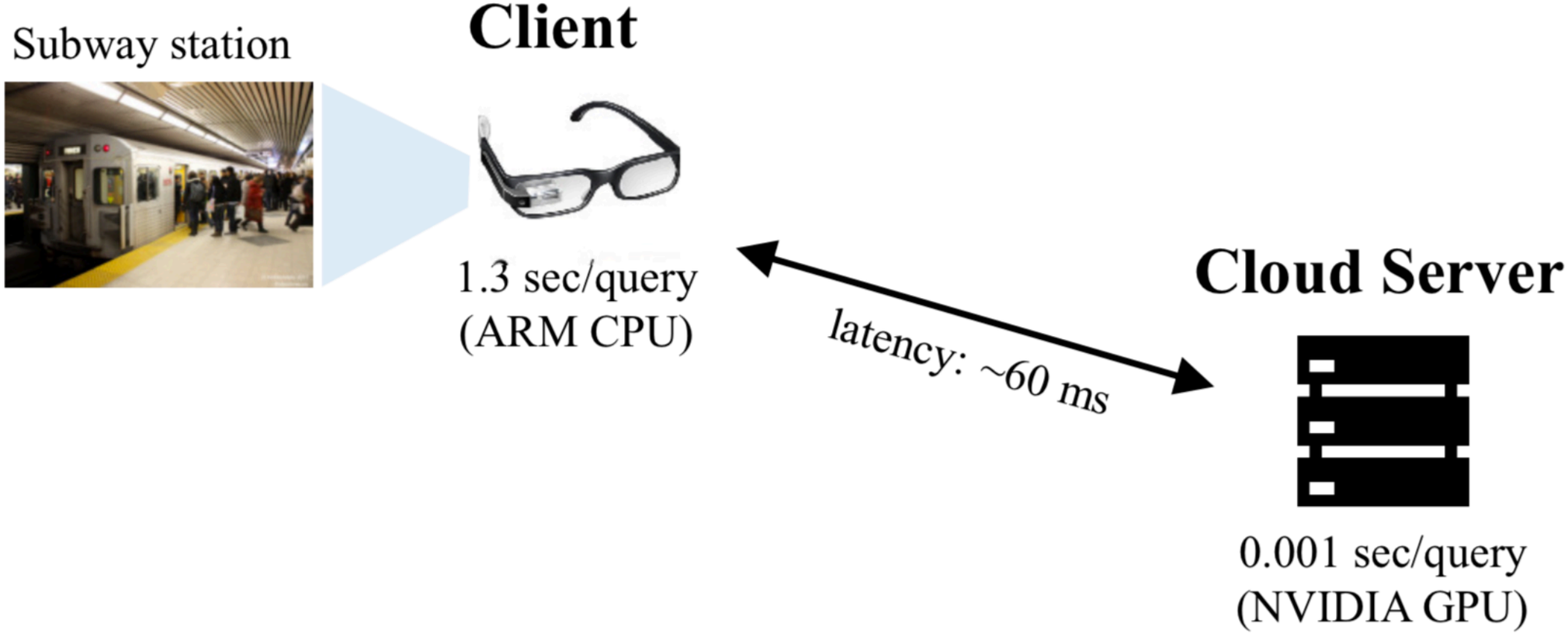
Subway station

Client

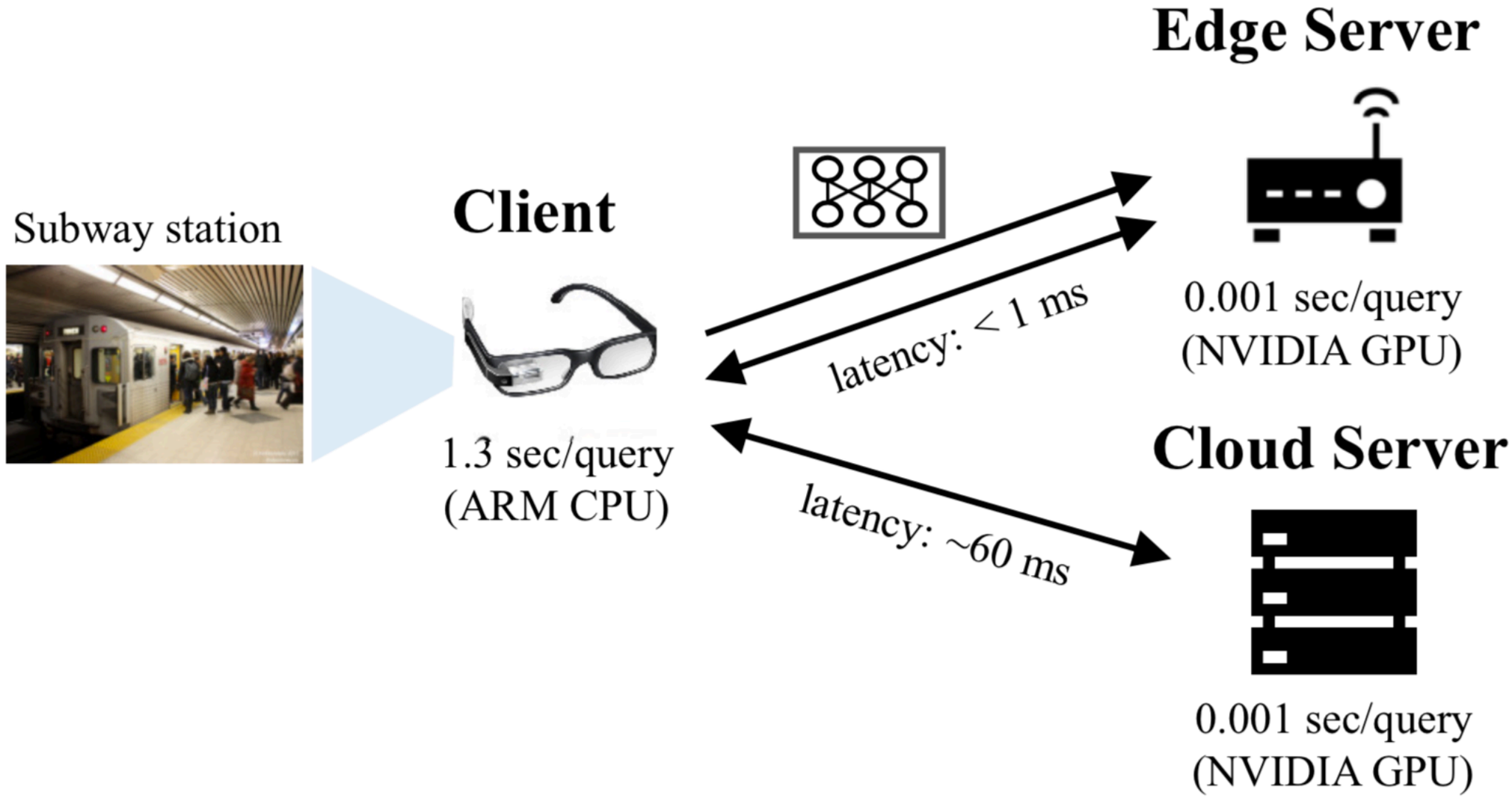
1.3 sec/query
(ARM CPU)

Motivation

- Server: x86 CPU (3.6GHz 4 cores), GTX 1080 Ti GPU, and 32GB memory

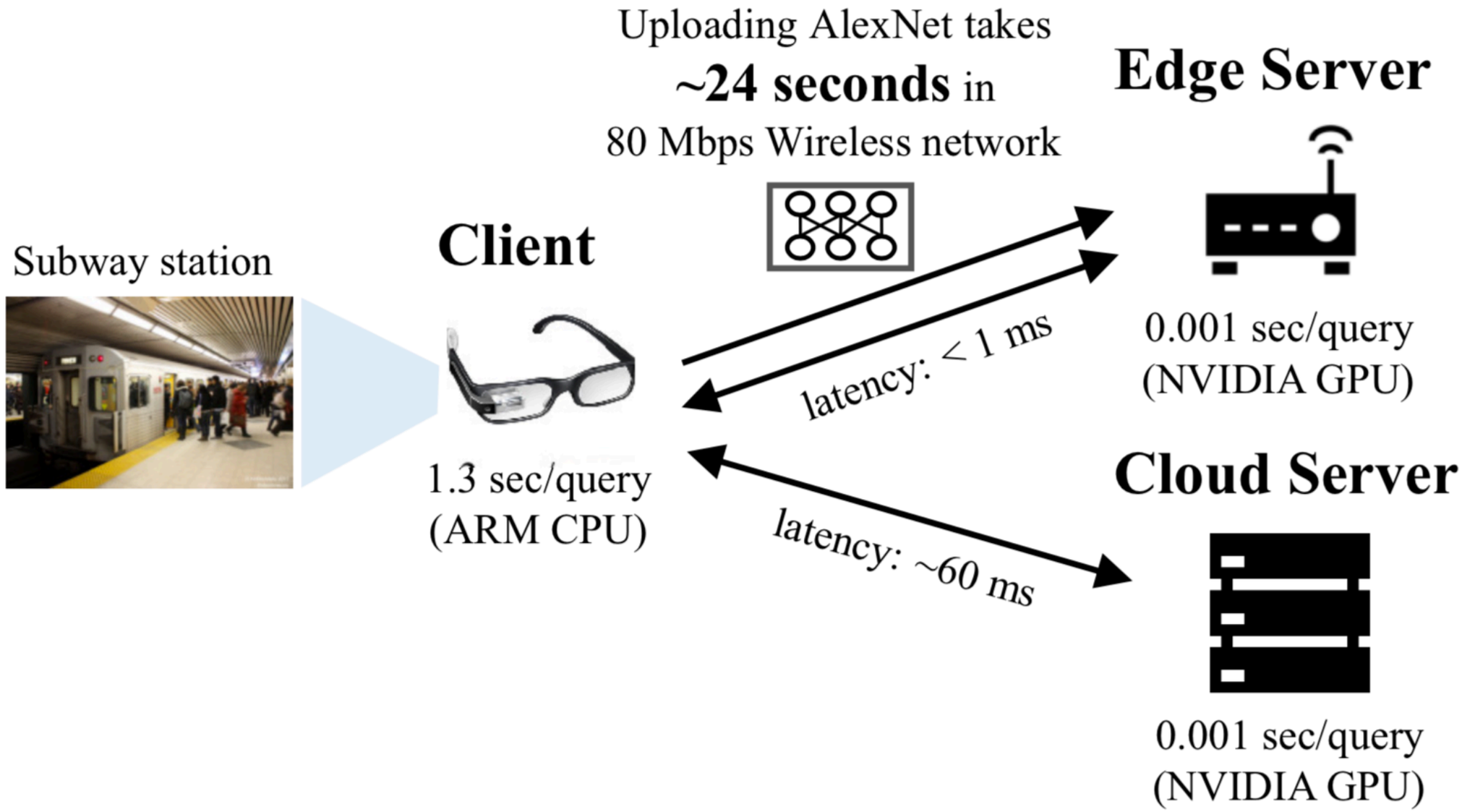


Motivation


Edge Server
Subway station
Client
0.001 sec/query
(NVIDIA GPU)
latency: < 1 ms
1.3 sec/query
(ARM CPU)
Cloud Server
latency: ~60 ms
0.001 sec/query
(NVIDIA GPU)

Motivation

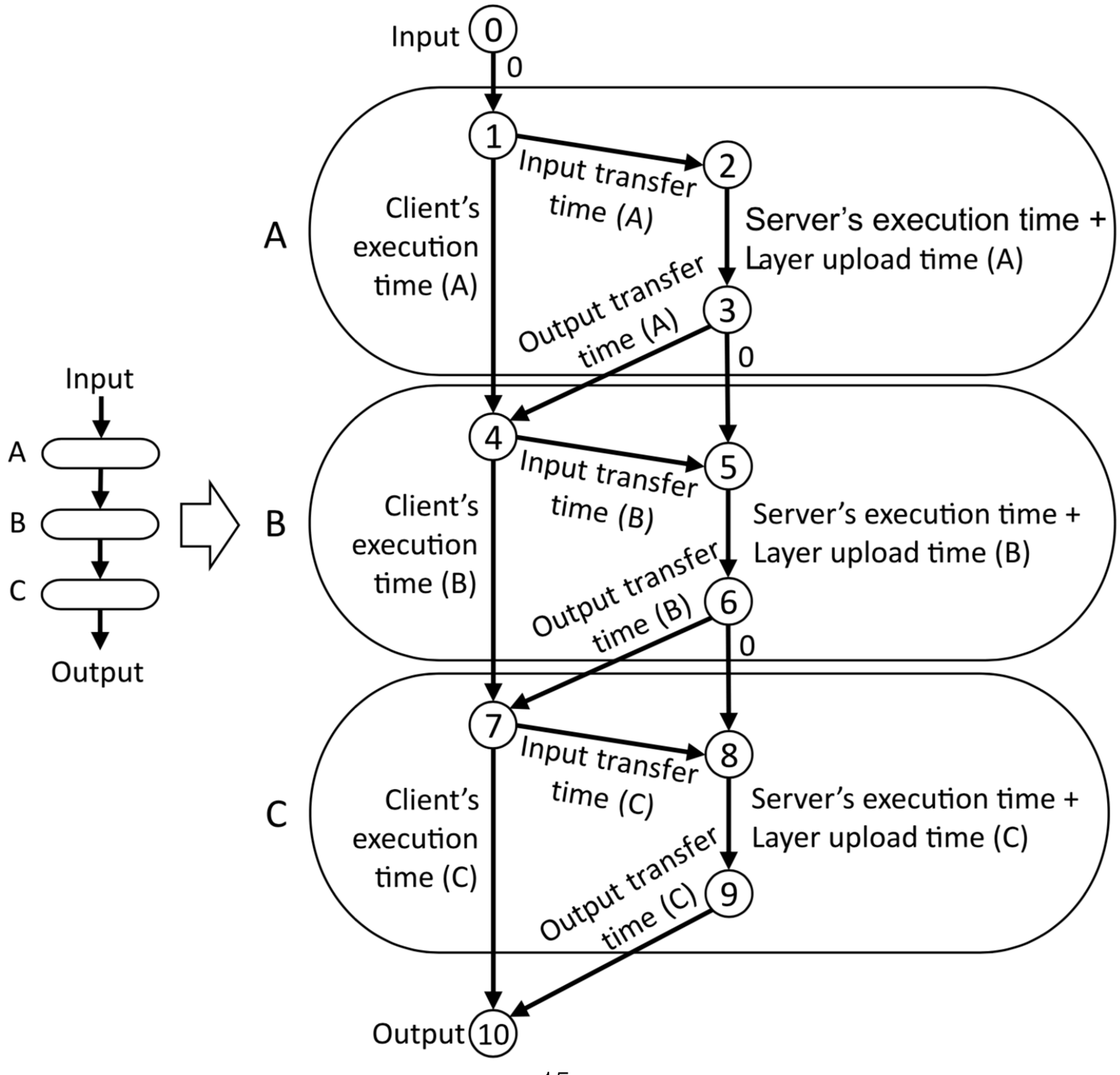
- **Incremental** offload! Assume IONN+VM is pre-installed



IONN: System Model

- Observation: Trained DNN can be saved in a file
- Can load a pre-trained DNN from the file and perform inference
- In the runtime phase, IONN creates an uploading plan
    - DNN partitions and their uploading order

NN Execution Graph



Partitioning Algorithm

- It is impossible to find an optimal solution unless we fully know the future occurrence of queries -> heuristic
  - 1- Prefer uploading DNN layers whose performance benefit is high and uploading overhead is low
  - 2- Do not send unnecessary DNN layers, that do not result in any performance increase,
- *Optimal state*: layer upload time = 0 (whole model is uploaded)
  - The solution is execution path with the best query performance with collaborative execution

Partitioning Algorithm

Algorithm 1 DNN Partitioning Algorithm

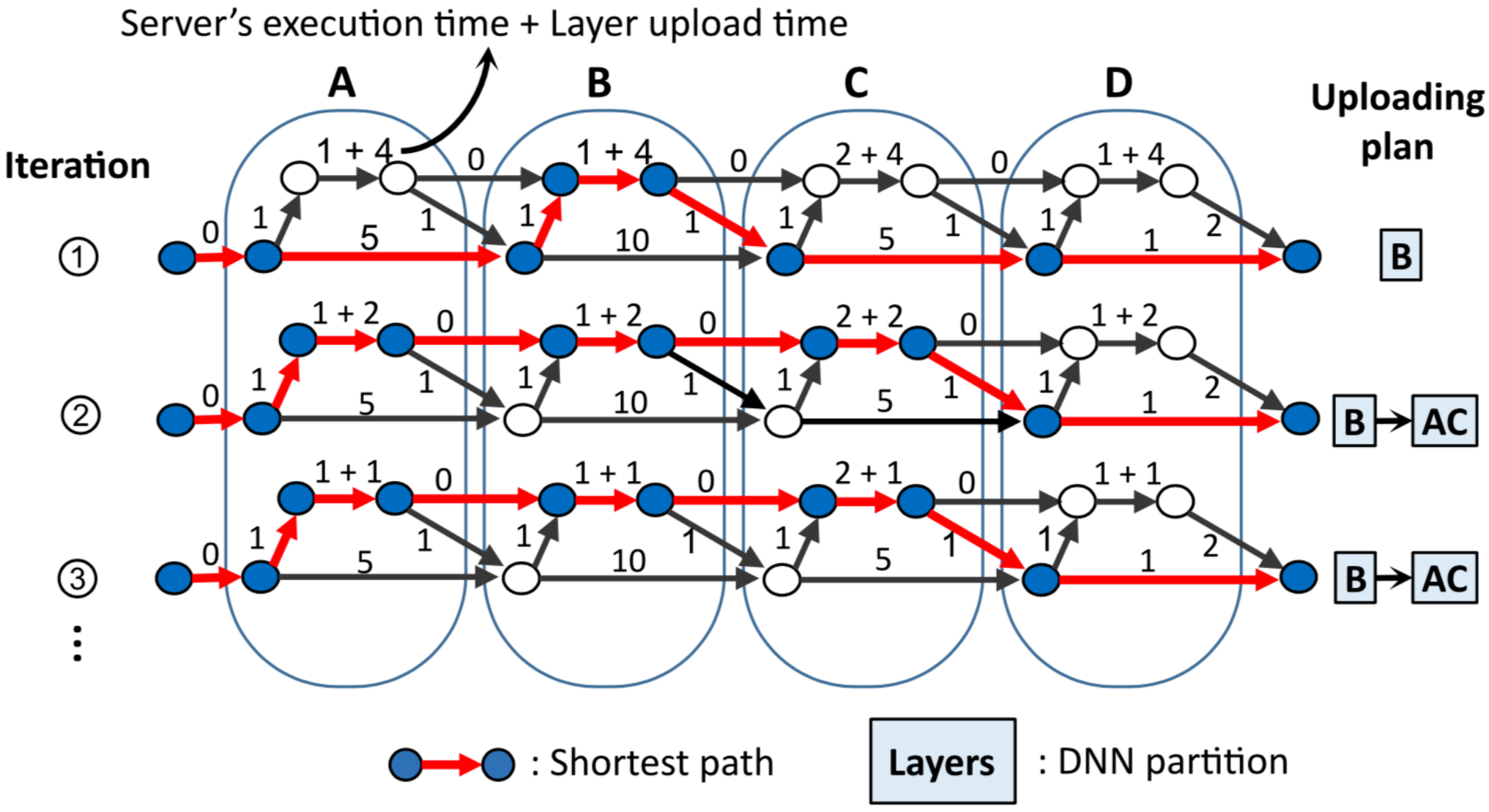
Input: DNN model description, DNN execution profile, prediction functions, network speed, K (positive number less than 1)
Output: Uploading plan (a list of DNN partitions)

1: **procedure** PARTITIONING
2: *partitions* ← [ ];
3: $n \leftarrow 0$;
4: Create NN execution graph using input parameters;
5: **while** $K^n \geq 0.01$ **do** ▷ Until layer upload time becomes ≈ 0
6: Search for the shortest path in the NN execution graph;
7: Create a DNN partition and add it to *partitions*;
8: Update the edge weights of the NN execution graph by multiplying K to layer upload time;
9: $n \leftarrow n + 1$;
10: **return** *partitions*

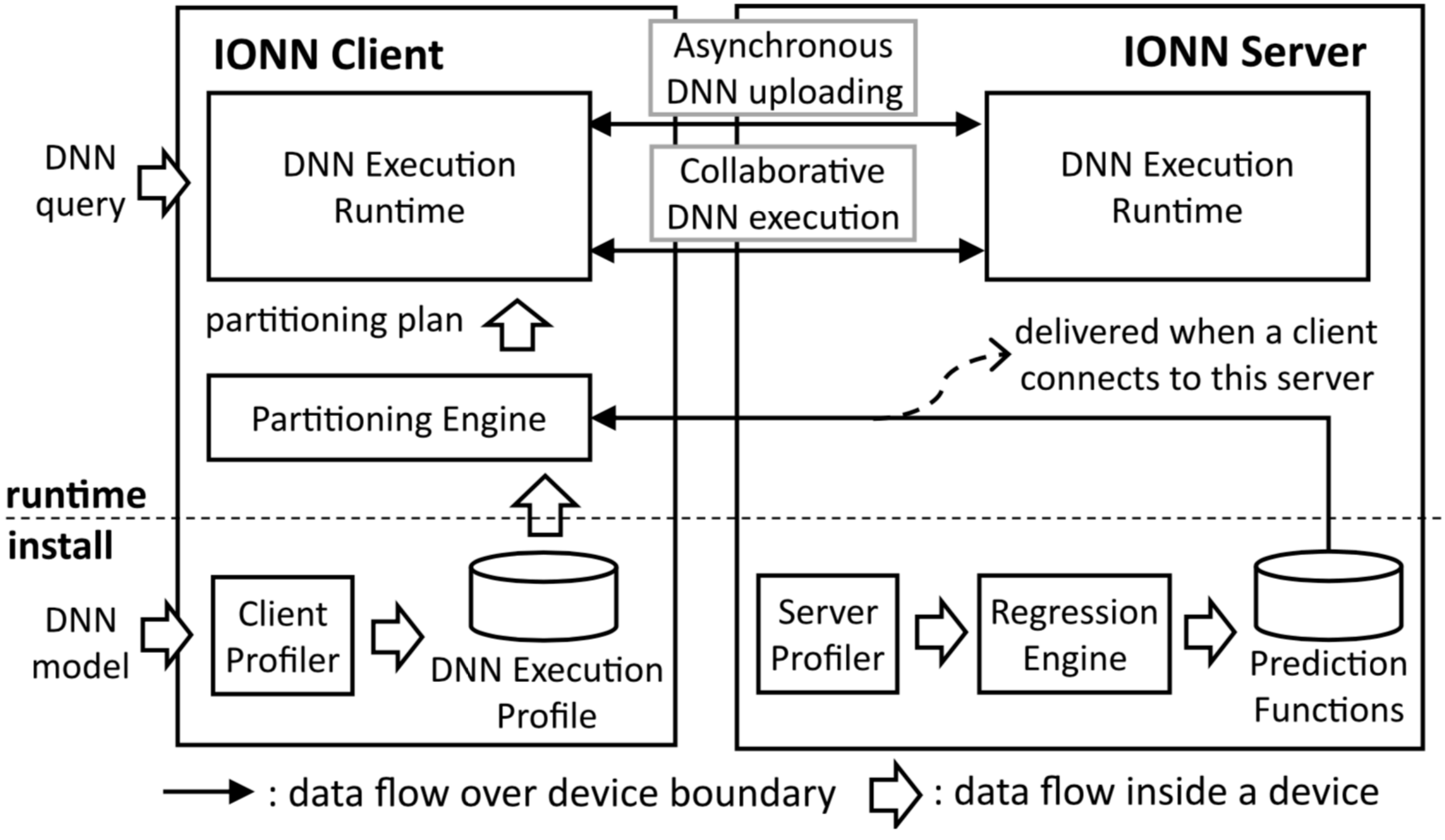
- If the value of K is small, will let the partitioning algorithm finish faster, making a few, large DNN partitions.

- A large K will lead to many iterations and create many, small DNN partitions.

Partitioning Example



IONN Architecture



Async. Upload and Execution

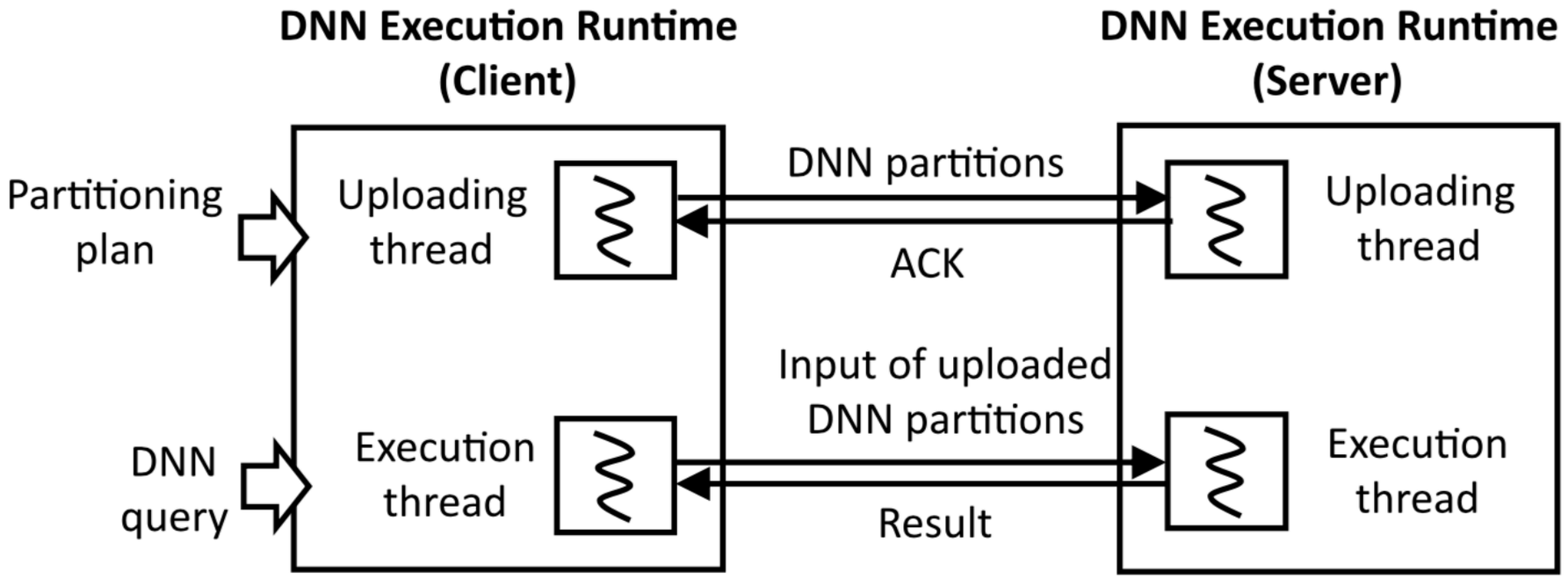
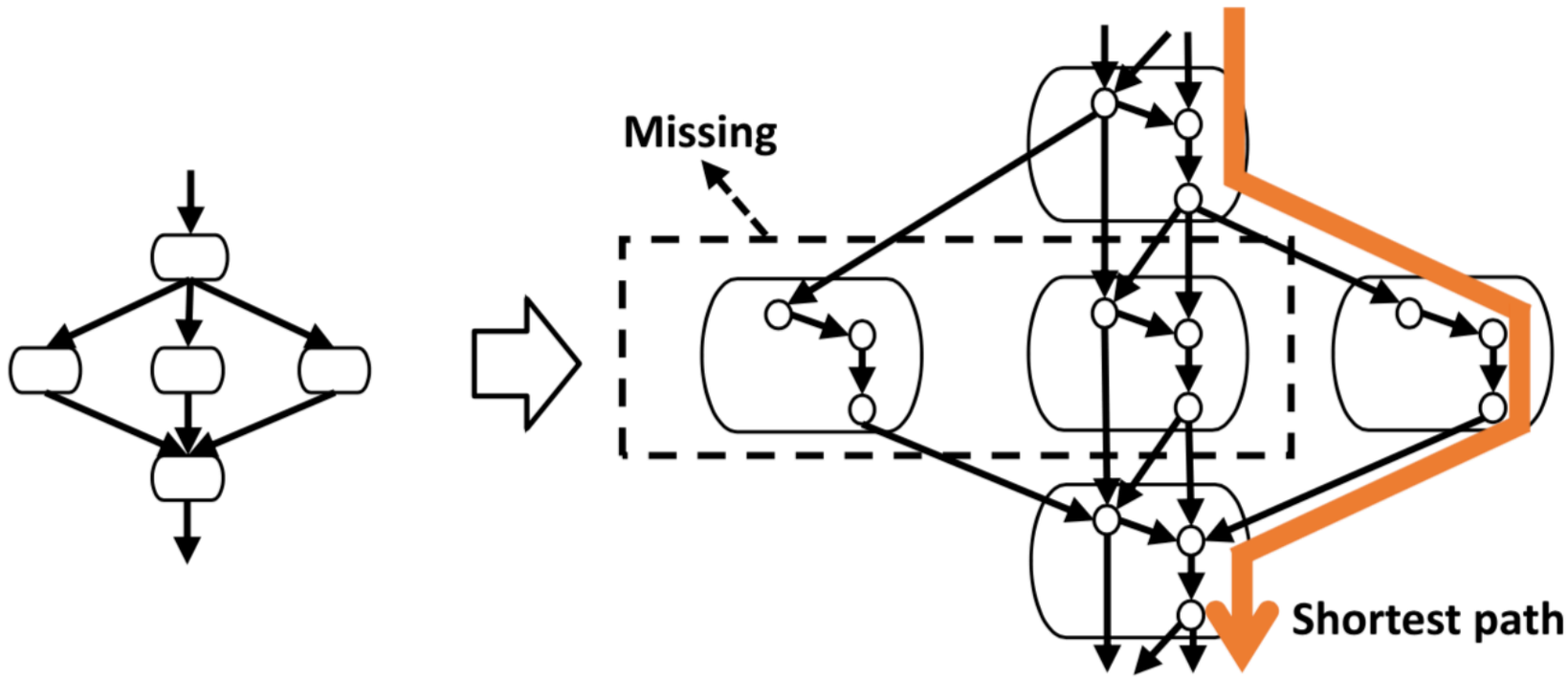
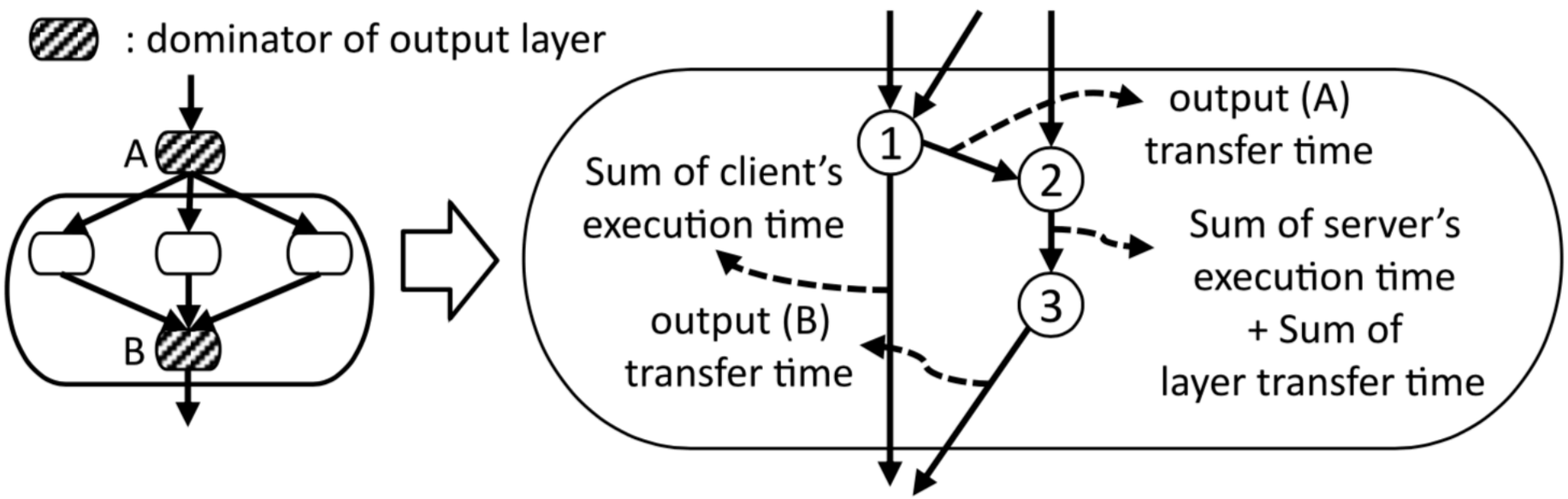


Figure 3: Asynchronous DNN uploading and collaborative DNN execution in DNN Execution Runtime.

Multiple Paths in DNN



(a) Problematic conversion of a DNN with multiple paths (some edges are omitted)



(b) Building NN execution graph as if there is no multiple paths

Results: Incremental Upload Overhead

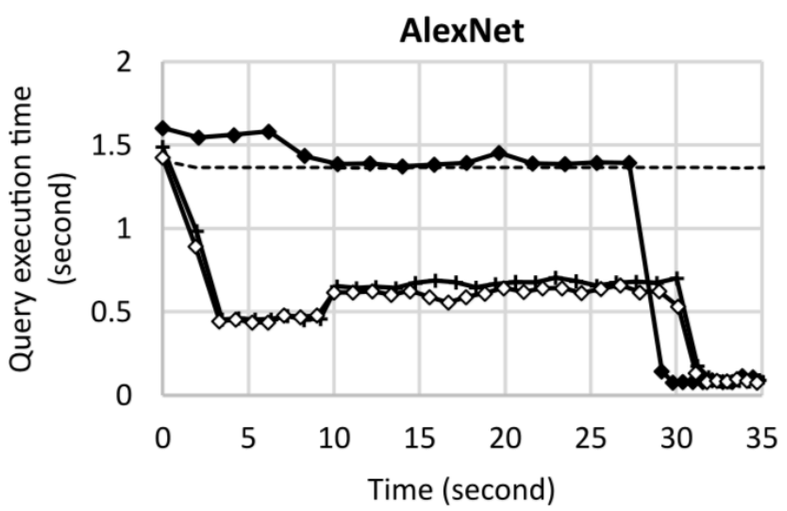
Name	Size (MB)	Number of layers	Reference
AlexNet	233	24	[22]
Inception	129	312	[37]
ResNet	98	245	[15]
GoogleNet	27	152	[36]
MobileNet	16	110	[16]

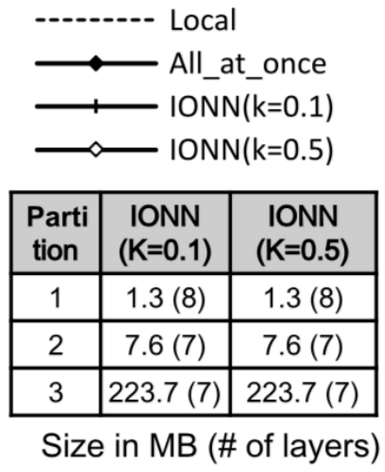
Table 1: DNNs For Evaluation

Name	All_at_once	IONN (K=0.1)	IONN (K=0.5)
AlexNet	28.7	30.3	30.7
Inception	16.4	16.7	16.8
ResNet	12.1	12.6	13.0
GoogleNet	3.9	3.9	4.4
MobileNet	2.3	2.5	2.8

Table 2: Uploading Completion Time (second)

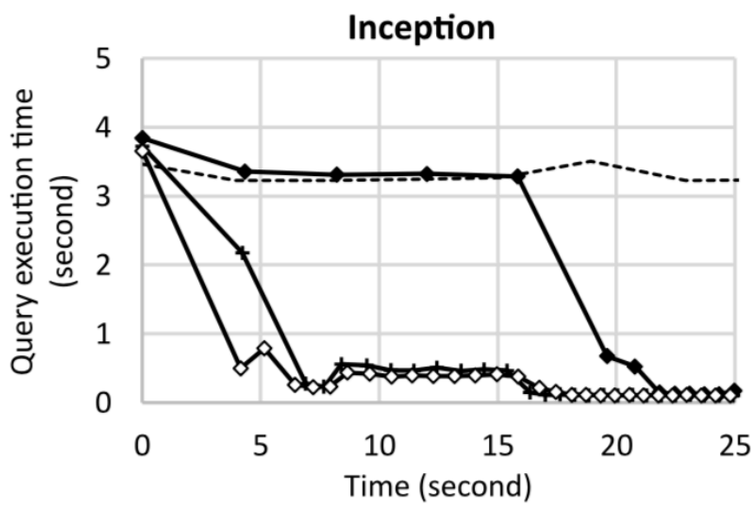
Results: Execution Time





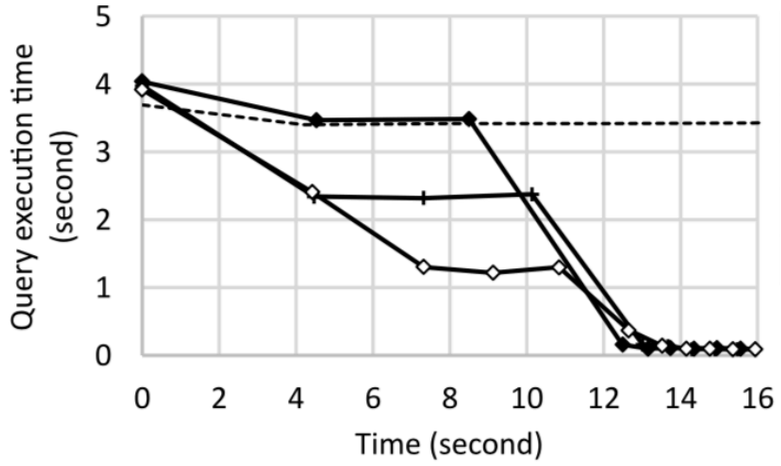
Partition	IONN (K=0.1)	IONN (K=0.5)
1	1.3 (8)	1.3 (8)
2	7.6 (7)	7.6 (7)
3	223.7 (7)	223.7 (7)

Size in MB (# of layers)



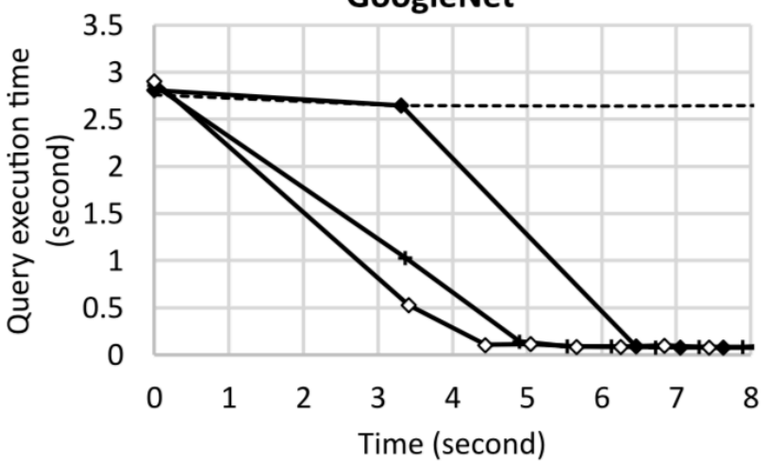
Partition	IONN (K=0.1)	IONN (K=0.5)
1	5.5 (76)	5.5 (76)
2	22.3 (233)	3.3 (23)
3	85.4 (1)	11.9 (124)
4	-	5.4 (23)
5	-	1.7 (63)
6	-	85.4 (1)

Size in MB (# of layers)



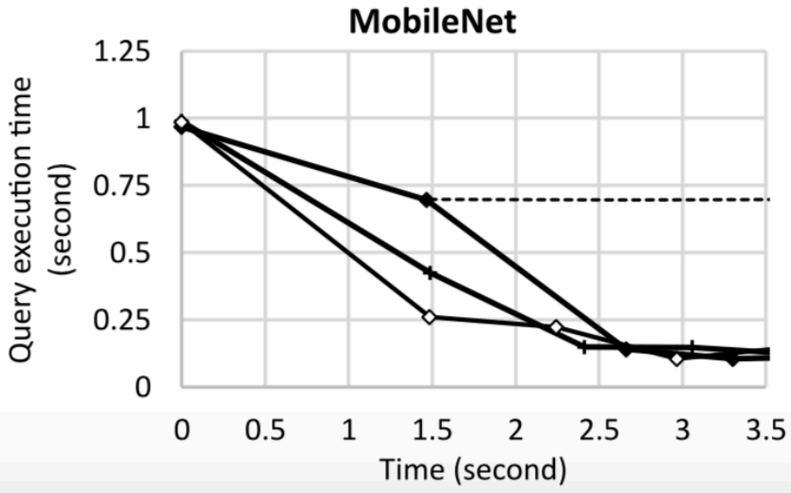
Partition	IONN (K=0.1)	IONN (K=0.5)
1	5.6 (108)	5.6 (108)
2	84.3 (134)	27.2 (87)
3	7.8 (1)	57.2 (47)
4	-	7.8 (1)

Size in MB (# of layers)



Partition	IONN (K=0.1)	IONN (K=0.5)
1	2.6 (41)	2.6 (41)
2	20.2 (107)	10.7 (76)
3	3.9 (2)	9.5 (31)
4	-	3.9 (2)

Size in MB (# of layers)



Partition	IONN (K=0.1)	IONN (K=0.5)
1	0.5 (45)	0.5 (45)
2	11.8 (64)	5.7 (48)
3	3.9 (1)	6.1 (16)
4	-	3.9 (1)

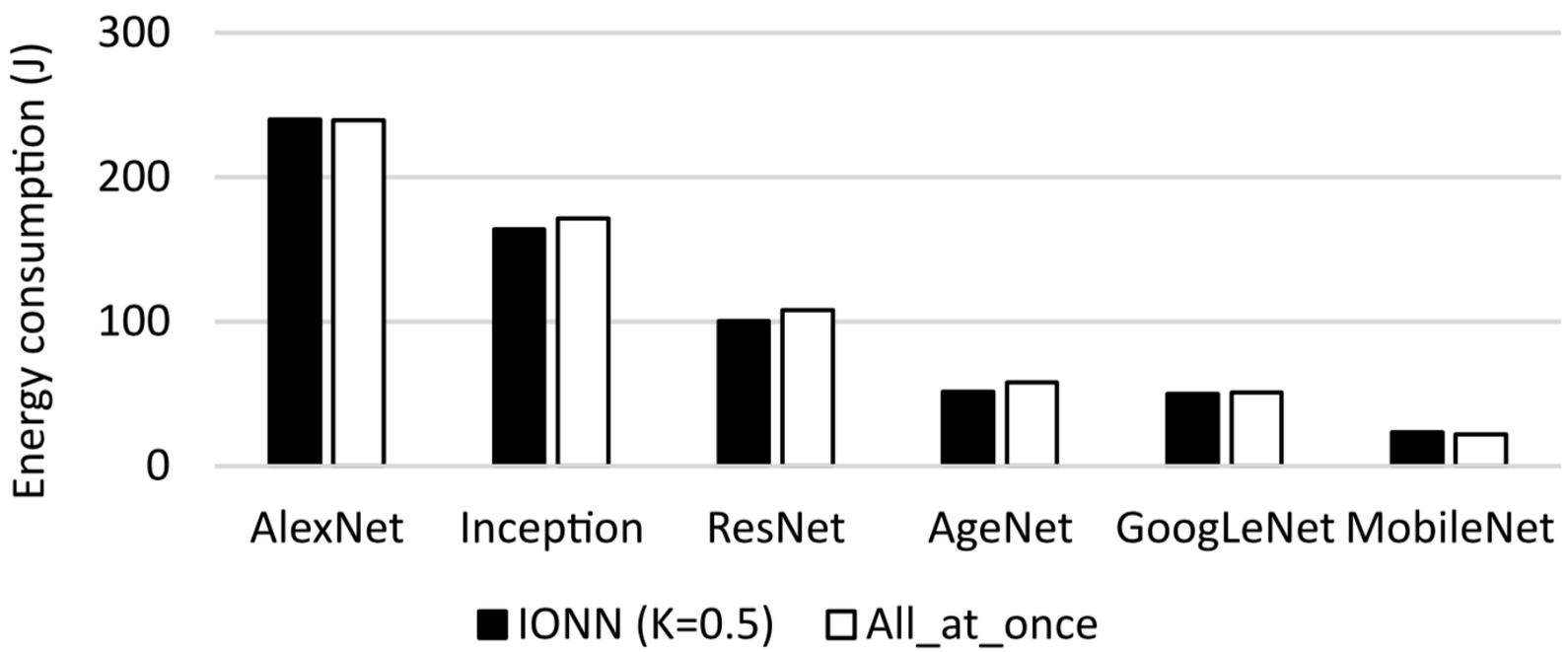
Size in MB (# of layers)

Results: Prediction Function Accuracy

Layer Type	R^2	RMSE (ms)	Layer Type	R^2	RMSE (ms)
Conv	0.428	0.025	FC	0.997	1.291
ReLU	0.999	0.001	Softmax	1.000	0.256
Pooling	0.853	0.002	BatchNorm	0.953	0.004
LRN	1.000	0.009	Scale	0.953	0.002
Concat	1.000	0.018	Eltwise	0.991	0.002

Table 3: R^2 and RMSE of Prediction Functions

Results: Throughput vs. Energy



Executed queries	AlexNet	Inception	ResNet	AgeNet	GoogLeNet	MobileNet
IONN	31	26	6	11	7	3
All_at_once	20	9	5	8	3	3

Figure 8: Execution time of DNN queries and the size of each DNN partition in our benchmark DNNs.

Thank you!