Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge

Yiping Kang Johann Hauswald Cao Gao Austin Rovinski
Trevor Mudge Jason Mars Lingjia Tang

December 5th, 2019
Jiae Lee

Contents

Overview

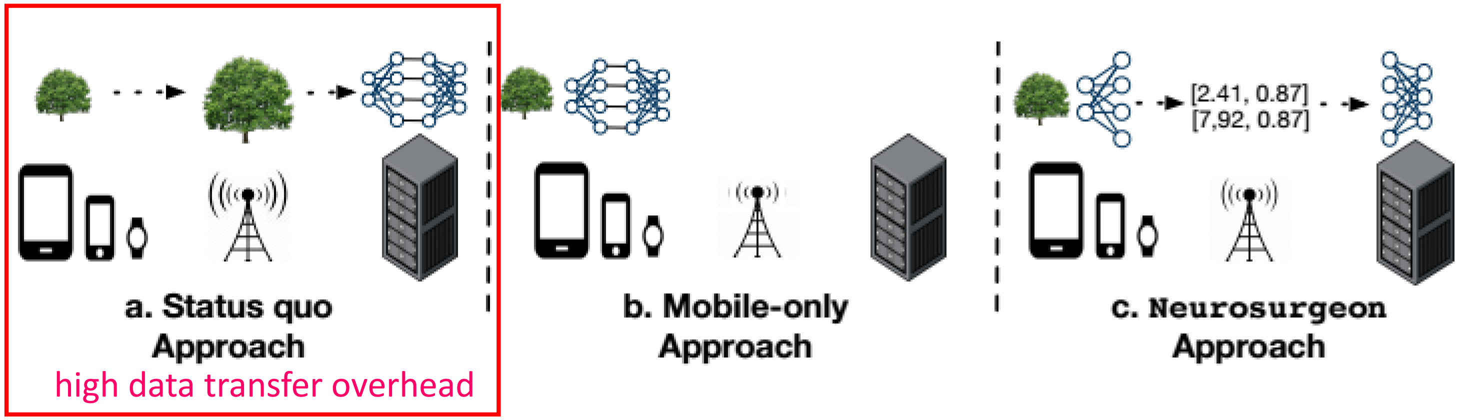
- As the computational resources in mobile devices become more powerful and energy efficient

- Whether this cloud-only processing is desirable moving forward

- What are the implications of pushing some or all of this compute to the mobile devices on the edge

Overview

- ❖ Queries generated from a user's mobile device are sent to the cloud for processing, as shown in Figure 1a
  - Large amounts of data (e.g., images, video and audio) are uploaded to the server via the wireless network, resulting in high latency and energy costs.
- ❖ **Data transfer**
  - The latency and energy bottleneck : high
  - Performance and energy efficiency of modern mobile hardware : better through powerful mobile SoC integration



- **a. Remote** : all computation remotely in the cloud
- **b. Local** : all computation on the mobile
- **c. Neurosurgeon** : partitions computation btw. The cloud & mobile device

The detailed contributions of this paper are as follows:

- ❖ In-depth **examination** of the status quo
- ❖ 1) **DNN compute** and 2) **data size characteristics study**
- ❖ DNN computation **partitioning** across the cloud & edge
- ❖ Neurosurgeon runtime system and layer performance **prediction models**

Cf. Uses 8 intelligent applications spanning computer vision, speech, and natural language domains

Cloud-only Processing: The Status Quo

- ❖Large overhead of in sending data over the wireless net.
- ❖ Used by cloud providers for intelligent applications
- ❖ perform all DNN processing in the cloud
- ❖Experimental setup
  - Caffe for the mobile and server platform.
  - OpenBLAS, NEONvectorized for the mobile CPU
    - matrix multiplication library and use the 4 cores available
  - cuDNN for both GPUs
    - Optimized NVIDIA library that accelerates key layers in Caffes
  - Caffe's CUDA implementations for rest of the layers.

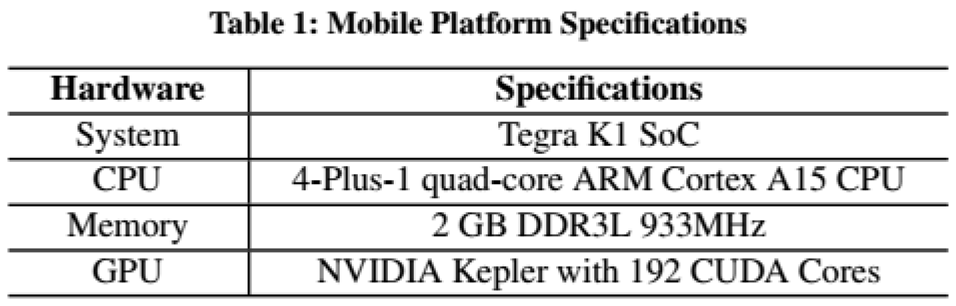


Table 1: Mobile Platform Specifications

Hardware	Specifications
System	Tegra K1 SoC
CPU	4-Plus-1 quad-core ARM Cortex A15 CPU
Memory	2 GB DDR3L 933MHz
GPU	NVIDIA Kepler with 192 CUDA Cores

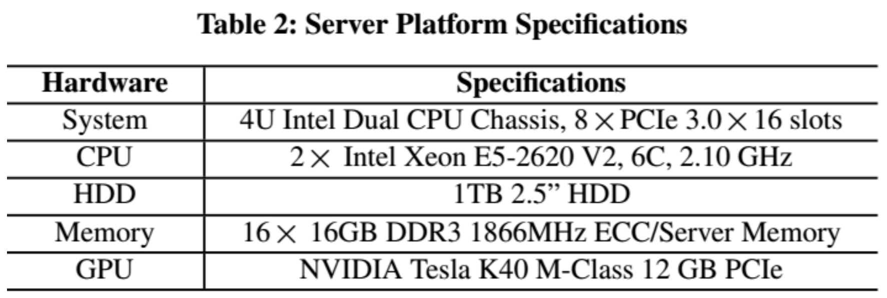


Table 2: Server Platform Specifications

Hardware	Specifications
System	4U Intel Dual CPU Chassis, 8 × PCIe 3.0 × 16 slots
CPU	2 × Intel Xeon E5-2620 V2, 6C, 2.10 GHz
HDD	1TB 2.5" HDD
Memory	16 × 16GB DDR3 1866MHz ECC/Server Memory
GPU	NVIDIA Tesla K40 M-Class 12 GB PCIe

Examining the Mobile Edge

- Investigate **the capability of the mobile platform** to execute a traditionally cloud-only DNN workload.
  - Use AlexNet : SOTA CNN for image classification
    & **representative in server environments**
- Break down the latency of an AlexNet query
  - a single inference on a 152KB image in Figure 3
- Wireless Communication: Measure the b/w of 3G, LTE, & Wi-Fi on several mobile devices using TestMyNet

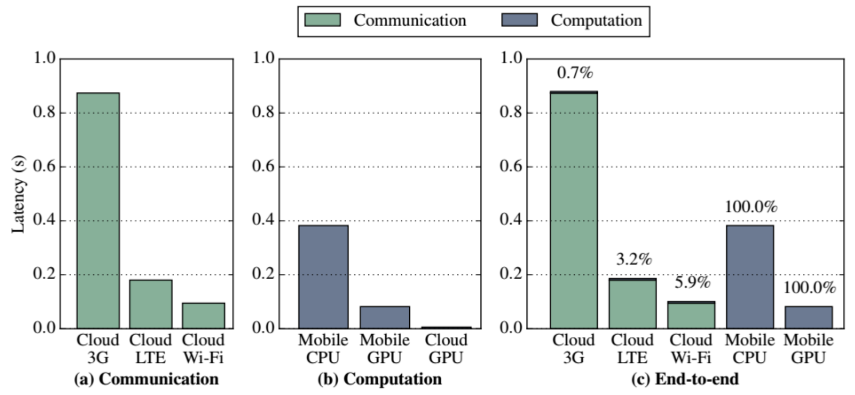


Figure 3: Latency breakdown for AlexNet (image classification). The cloud-only approach is often slower than mobile execution due to the high data transfer overhead.

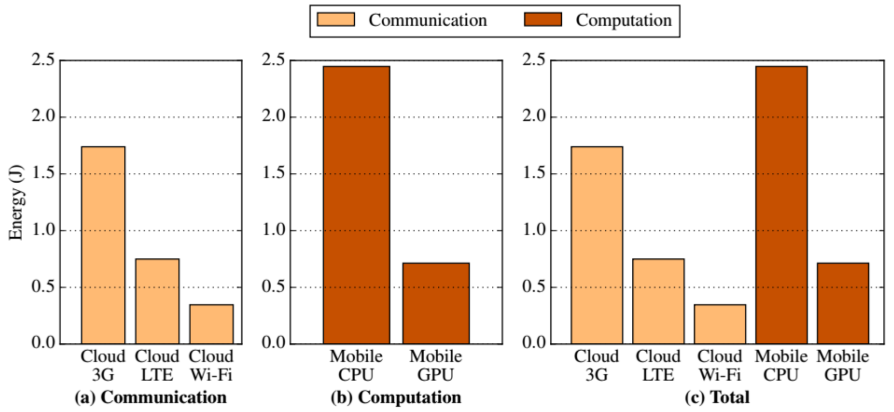


Figure 4: Mobile energy breakdown for AlexNet (image classification). Mobile device consumes more energy transferring data via LTE and 3G than computing locally on the GPU.

TestMy.net: Broadband Internet Speed Test

Per layer latency & Granuality

❖Investigate the data and computation **characteristics** of each layer in AlexNet

- Identify a better computation partitioning btw. mobile &cloud at the layer level

Wireless net. configuration

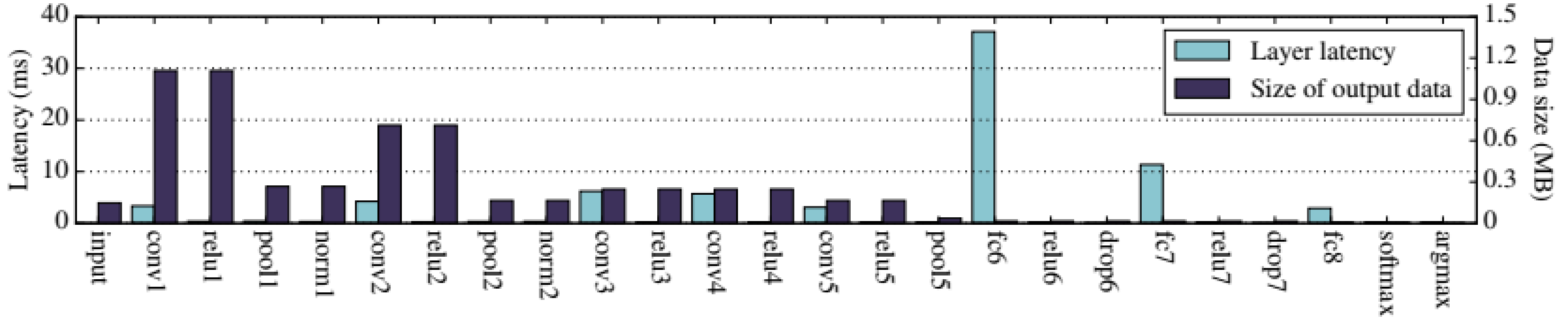
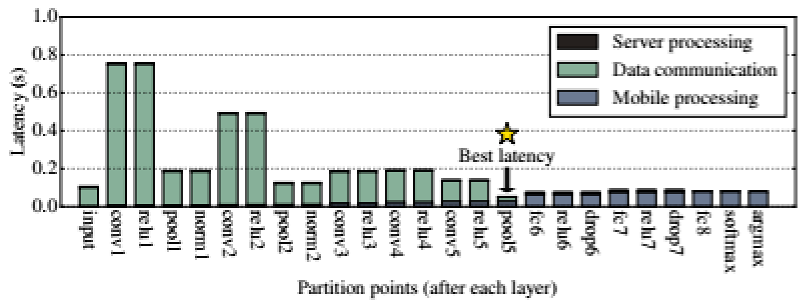
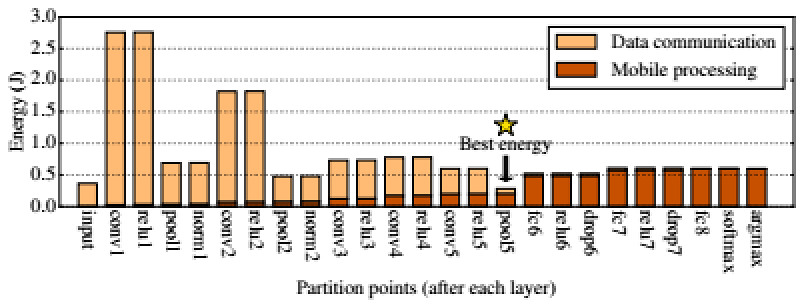


Figure 5: The per layer execution time (the light-colored left bar) and size of data (the dark-colored right bar) after each layer's execution (input for next layer) in AlexNet. Data size sharply increases then decreases while computation generally increases through the network's execution.



(a) AlexNet latency



(b) AlexNet energy consumption

Figure 6: End-to-end latency and mobile energy consumption when choosing different partition points. After the execution of every layer is considered a partition point. Each bar represents the total latency (a) or mobile energy (b) if the DNN is partitioned after the layer marked on the X-axis. The left-most bar represents cloud-only processing and the right-most bar represents mobile-only processing. The partition points for best latency and mobile energy are annotated.

Examining the latency & size of data On GPU

❖7a –7c : Similar characteristics as AlexNet

CV(computer vision): DNNs(VGG,FACE*DIGs) similar to AlexNet
ASR,POS, NER & CHK : consists of fully-connected layers & activation layers.

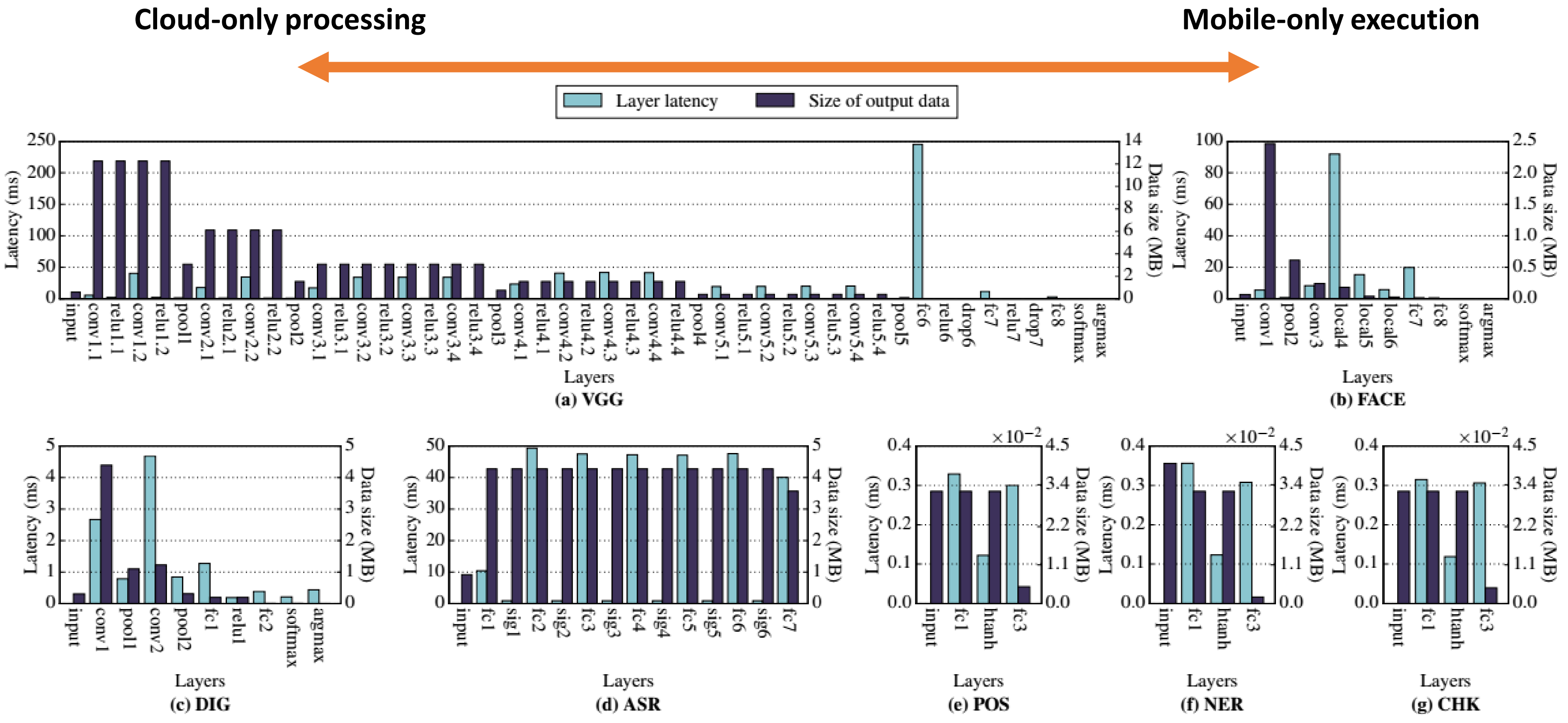


Figure 7: The per layer latency on the mobile GPU (left light-color bar) and size of data (right dark-color bar) after each layer's execution.

Examining : End-2-End latency

- Show the different partition points for best latency
- **8a - 8c** : show that different CV applications have different partition points for best latency

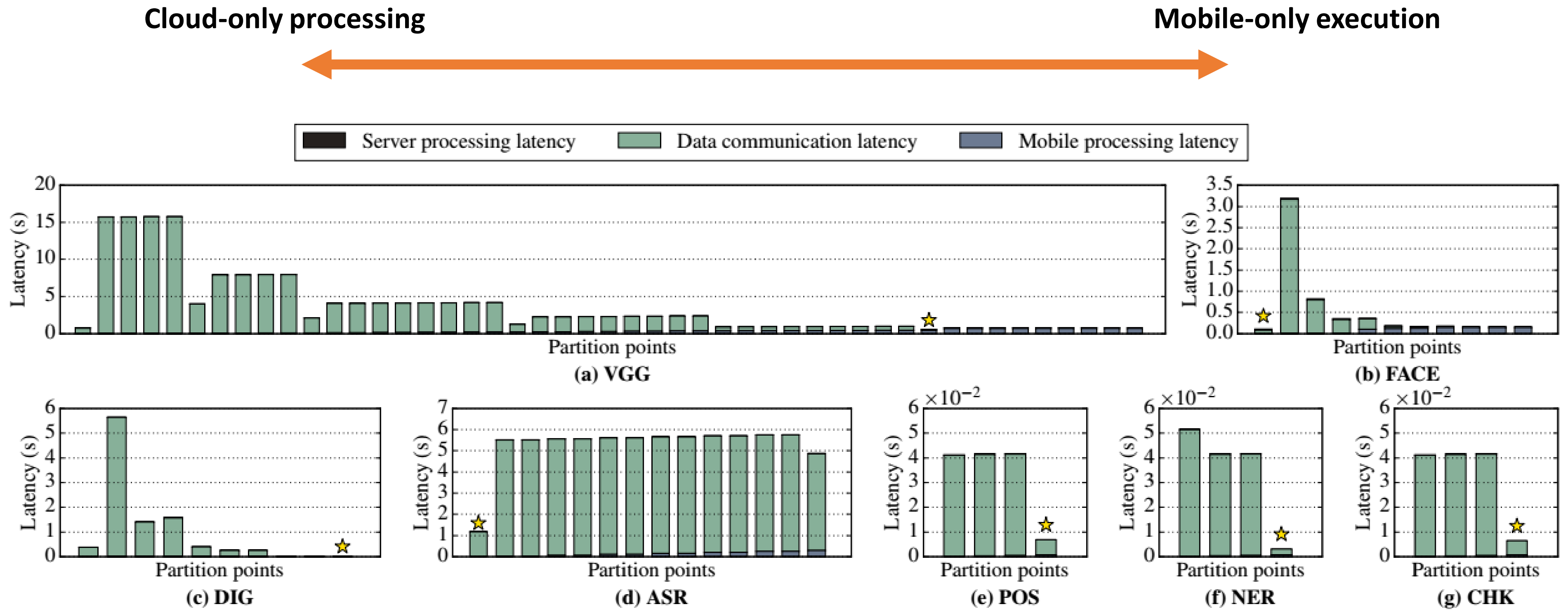


Figure 8: End-to-end latency when choosing different partition points. Each bar represents the end-to-end latency if the DNN is partitioned after each layer, where the left-most bar represents cloud-only processing (i.e., partitioning at the beginning) while the right-most bar represents mobile-only execution (i.e., partitioning at the end). The wireless network configuration is LTE. The partition points for best latency are each marked by ★.

Examining: Mob. Energy consumption

❖**9a – 9g** : show the different partition points for best energy for these DNNs.

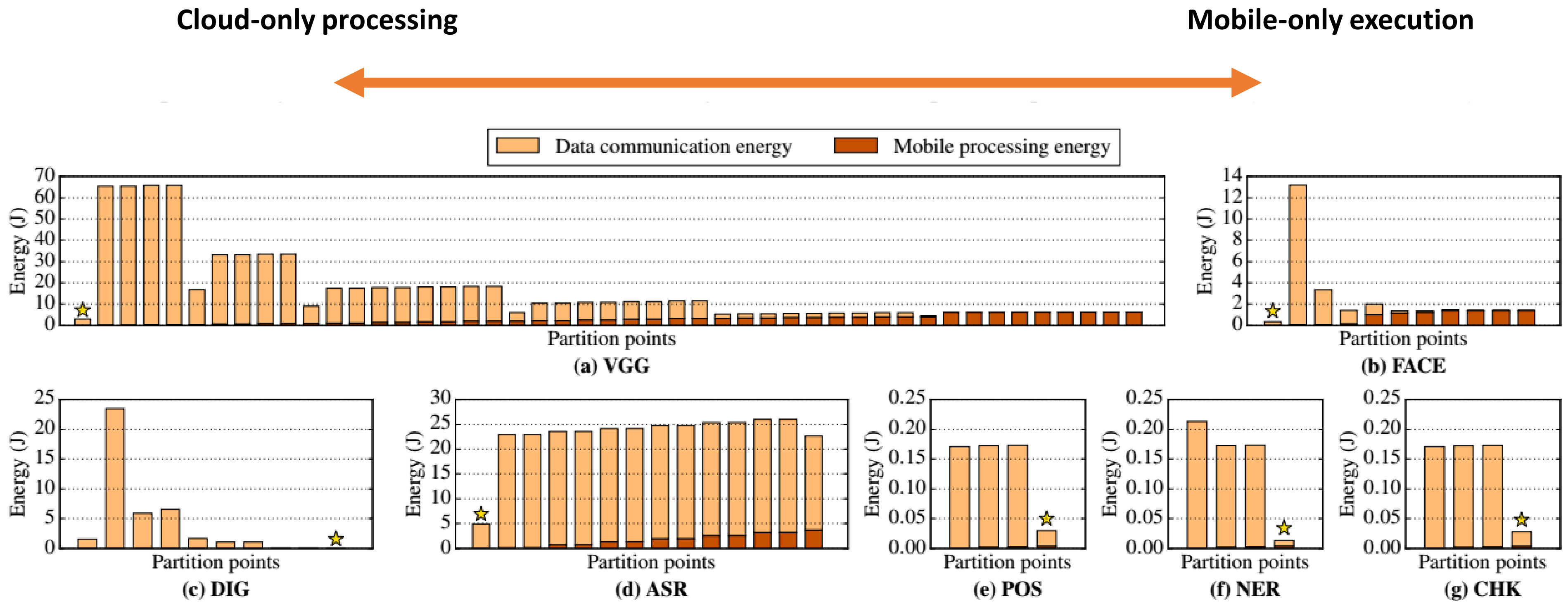


Figure 9: Mobile energy consumption when choosing different partition points. Each bar represents the mobile energy consumption if the DNN is partitioned after each layer, where the left-most bar represents cloud-only processing (i.e., partitioning at the beginning) while the right-most bar represents mobile-only execution (i.e., partitioning at the end). The wireless network configuration is LTE. The partition points for best energy are each marked by ★.

Neurosurgeon

- a lightweight scheduler to automatically partition DNN computation btw. mobile devices & datacenters at the granularity of NN layers
- Does not require per-application profiling
  - Adapts to various DNN arch., HW platforms, wireless net., & server load levels, intelligently partitioning computation for best mobile energy.
- Evaluate Neurosurgeon on a state-of-the-art mobile development platform
  - Improves end-to-end latency by 3.1 on avg. up to 40.7
  - Reduces mobile energy consumption 59.5% on average and up to 94.7%
  - Improves datacenter throughput by 1.5 on avg. and up to 6.7
- A runtime system spanning cloud and mobile platforms (between the mobile device and the datacenter.)
  - Automatically identifies the ideal partition points in DNNs
  - Orchestrates the distribution of computation
  - Partitions the DNN computation and takes advantage of the processing power of both the mobile and the cloud while reducing data transfer overhead. [Fig. 1c]

Neurosurgeon

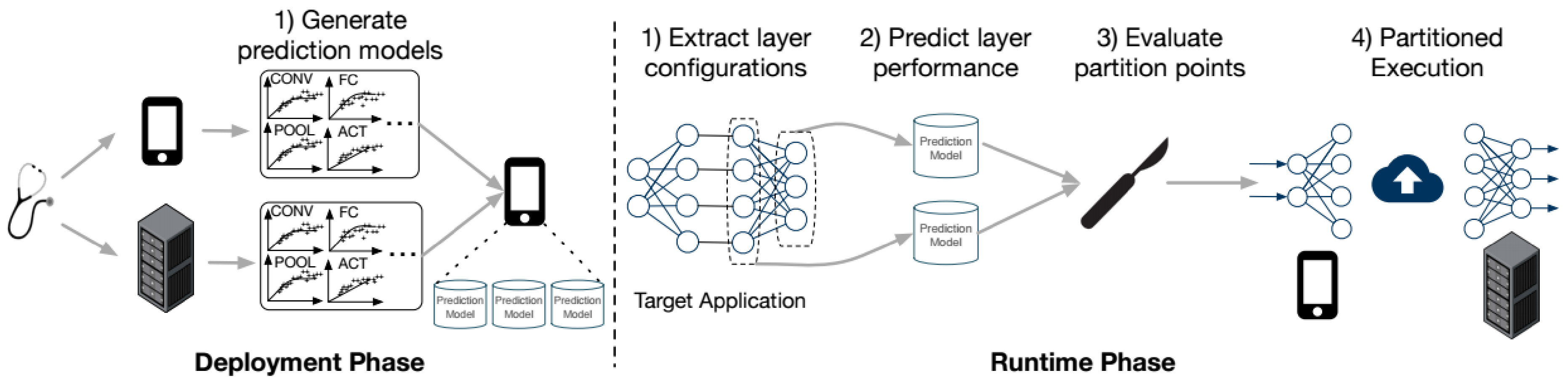

1) Generate prediction models
CONV
FC
POOL
ACT
Prediction Model
Deployment Phase
1) Extract layer configurations
2) Predict layer performance
3) Evaluate partition points
4) Partitioned Execution
Target Application
Runtime Phase

Figure 10: Overview of Neurosurgeon. At deployment, Neurosurgeon generates prediction models for each layer type. During runtime, Neurosurgeon predicts each layer's latency/energy cost based on the layer's type and configuration, and selects the best partition point based on various dynamic factors.

Profiles the mobile device & the server to generate performance prediction models for the spectrum of DNN layer types (enumerated in Section 4.1)

DNN based intelligent application on the mobile device, Neurosurgeon dynamically decides the best partition point for the DNN.

Performance Prediction Model

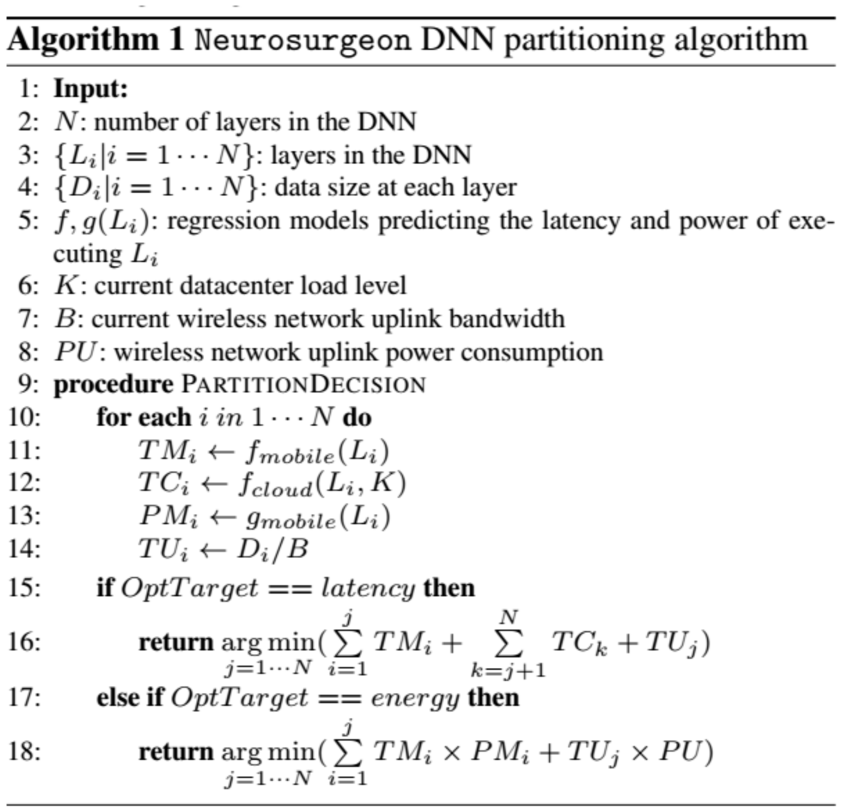
Use **GFLOPS** (Giga Floating Point Operations per Second) as performance matric
Based on layer type, use a l ogarithmic or linear function as the regression function

Dynamic DNN partitioning

- ❖Neurosurgeon dynamically selects the best DNN partition points
- ❖2-step
  - Step 1: Analysis of the Target DNN
  - Step 2: Partition Point Selection

Line 11-12: Neurosurgeon extracts each layer's type and configuration (Li) and uses the regression models to predict the latency of executing layer Li on mobile (TMi) and cloud (T Ci), while taking into consideration of current datacenter load level (K)
Line 13 : estimates the power of executing layer L_i on the mobile device (PM_i)
Line 14 : calculates the wireless data transfer latency(TU_i) based on the latest wireless net. BW

Line 16 &18 : evaluate the performance when partitioning at each candidate point & select the point for either best end-to-end latency or best mobile energy consumption (Becuz **simplicity** of the regression models
→ **lightweight and efficient**)



Algorithm 1 Neurosurgeon DNN partitioning algorithm

```
1: Input:
2: $N$: number of layers in the DNN
3: $\{L_i | i = 1 \cdots N\}$: layers in the DNN
4: $\{D_i | i = 1 \cdots N\}$: data size at each layer
5: $f, g(L_i)$: regression models predicting the latency and power of exe-
   cuting $L_i$
6: $K$: current datacenter load level
7: $B$: current wireless network uplink bandwidth
8: $PU$: wireless network uplink power consumption
9: procedure PARTITIONDECISION
10:    for each $i$ in $1 \cdots N$ do
11:        $TM_i \leftarrow f_{mobile}(L_i)$
12:        $TC_i \leftarrow f_{cloud}(L_i, K)$
13:        $PM_i \leftarrow g_{mobile}(L_i)$
14:        $TU_i \leftarrow D_i / B$
15:    if $OptTarget == latency$ then
16:        return $\arg\min_{j=1 \cdots N}(\sum_{i=1}^{j} TM_i + \sum_{k=j+1}^{N} TC_k + TU_j)$
17:    else if $OptTarget == energy$ then
18:        return $\arg\min_{j=1 \cdots N}(\sum_{i=1}^{j} TM_i \times PM_i + TU_j \times PU)$
```

Partitioned execution

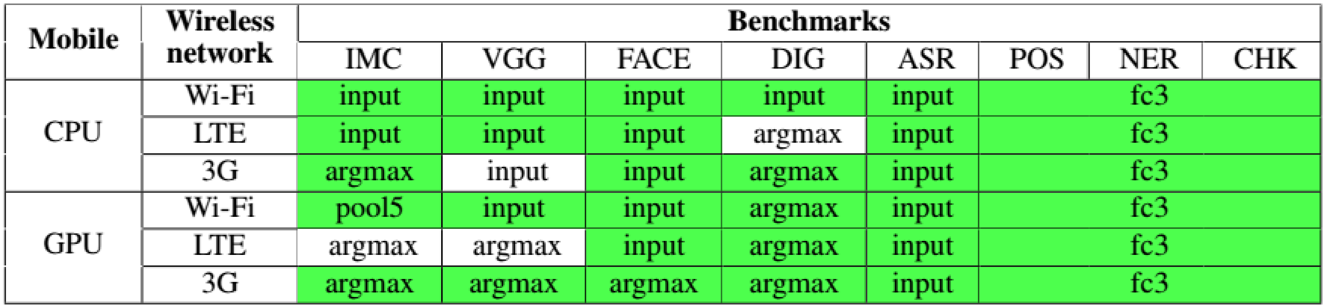
- Prototyping Neurosurgeon by creating modified instances of Caffe

- NSmobile(mobile-side) & NSserver(server-side)

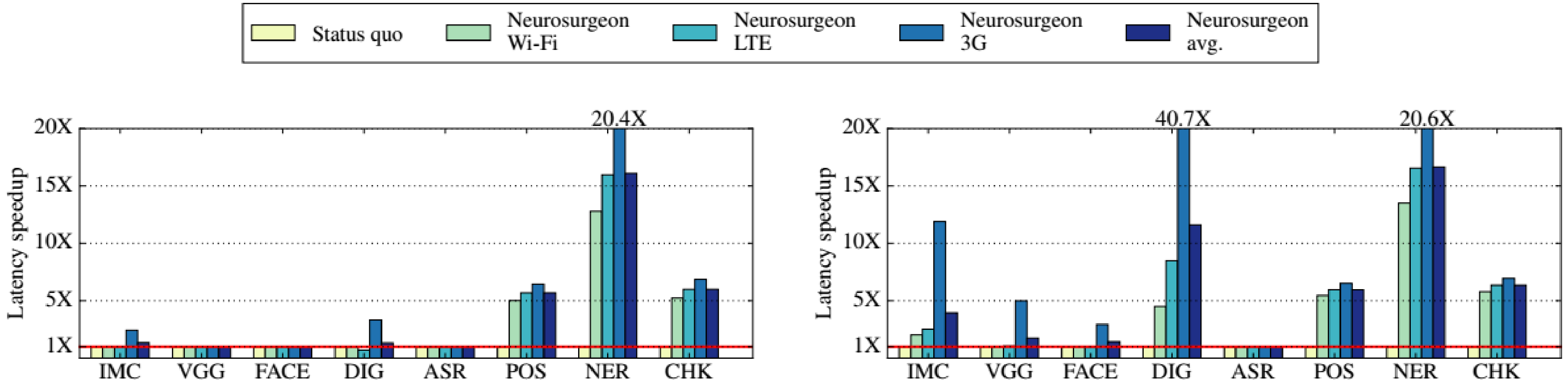
- Client-server interface using Thrift
  - Open source flexible RPC interface for inter-process communication
  - More precisely, Thrift is a **SW library** and set of **code-generation tools** developed at Facebook
    to expedite development and implementation of **efficient** and **scalable backend services**

Results: Optimal/sub optimal partition choice

Table 4: Neurosurgeon's partition point selections for best end-to-end latency. Green block indicates Neurosurgeon makes the optimal partition choice and white block means a suboptimal partition point is picked. On average, Neurosurgeon achieves within 98.5% of the optimal performance.



Mobile	Wireless network	Benchmarks							
		IMC	VGG	FACE	DIG	ASR	POS	NER	CHK
CPU	Wi-Fi	input	input	input	input	input		fc3	
	LTE	input	input	input	argmax	input		fc3	
	3G	argmax	input	input	argmax	input		fc3	
GPU	Wi-Fi	pool5	input	input	argmax	input		fc3	
	LTE	argmax	argmax	input	argmax	input		fc3	
	3G	argmax	argmax	argmax	argmax	input		fc3	



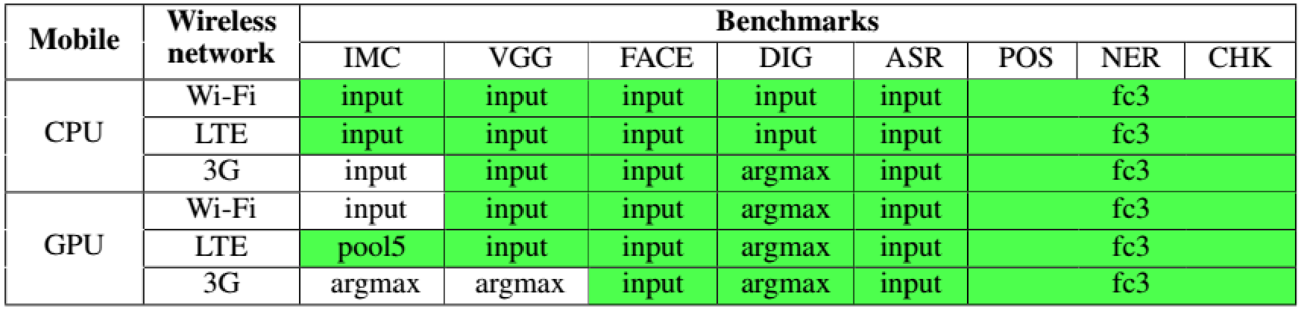
(a) Neurosurgeon using the mobile CPU

(b) Neurosurgeon using the mobile GPU

Figure 11: Latency speedup achieved by Neurosurgeon normalized to status quo approach (executing entire DNN in the cloud). Results for three wireless networks (Wi-Fi, LTE and 3G) and mobile CPU and GPU are shown here. Neurosurgeon improves the end-to-end DNN inference latency by 3.1× on average (geometric mean) and up to 40.7×.

Results

Table 5: `Neurosurgeon` partition point selections for best mobile energy consumption. Green block indicates `Neurosurgeon` makes the optimal partition choice and white block means a suboptimal partition point is picked. On average, `Neurosurgeon` achieves a mobile energy reduction within 98.8% of the optimal reduction.



Mobile	Wireless network	Benchmarks							
		IMC	VGG	FACE	DIG	ASR	POS	NER	CHK
CPU	Wi-Fi	input	input	input	input	input		fc3	
	LTE	input	input	input	input	input		fc3	
	3G	input	input	input	argmax	input		fc3	
GPU	Wi-Fi	input	input	input	argmax	input		fc3	
	LTE	pool5	input	input	argmax	input		fc3	
	3G	argmax	argmax	input	argmax	input		fc3	

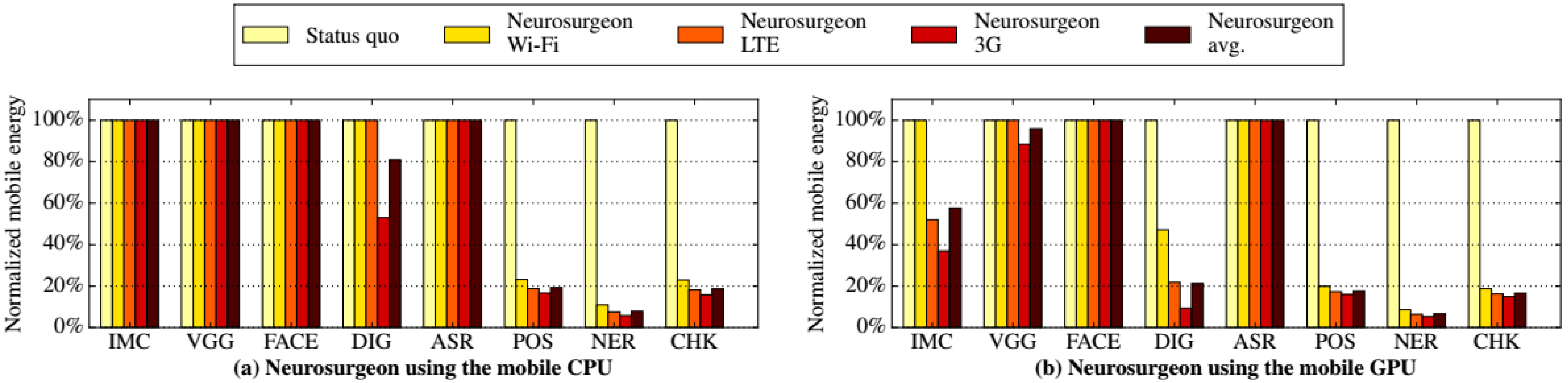


Figure 12: Mobile energy consumption achieved by `Neurosurgeon` normalized to status quo approach (executing entire DNN in the cloud). Results for three wireless networks (Wi-Fi, LTE and 3G) and mobile CPU and GPU are shown here. `Neurosurgeon` reduces the mobile energy consumption by 59.5% on average (geometric mean) and up to 94.7%.

Results :Comparing other frameworks

MAUI?
a system that enables fine-grained energy-aware offload of mobile code to the infrastructure

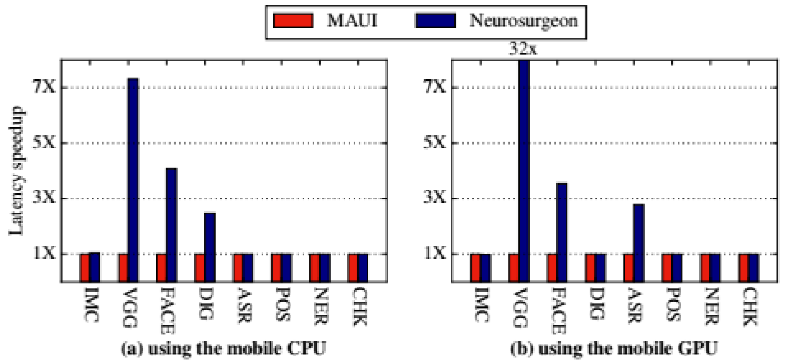
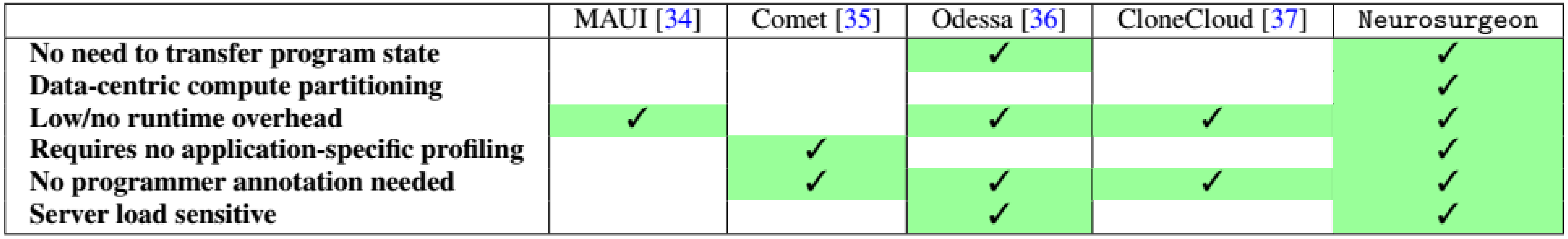


Figure 13: Latency speedup achieved by Neurosurgeon vs. MAUI [34]. For MAUI, we assume the optimal programmer annotation that achieves minimal program state transfer. Neurosurgeon outperforms MAUI by up to 32× and 1.9× on average.

Table 6: Comparing Neurosurgeon to popular computation offloading/partition frameworks



	MAUI [34]	Comet [35]	Odessa [36]	CloneCloud [37]	Neurosurgeon
No need to transfer program state			✓		✓
Data-centric compute partitioning					✓
Low/no runtime overhead	✓		✓	✓	✓
Requires no application-specific profiling		✓			✓
No programmer annotation needed		✓	✓	✓	✓
Server load sensitive			✓		✓

Results : Test variance

- ❖ Network variation(Wireless net.)
  - Resilience
  - In T-mobile
  - CPU-only model

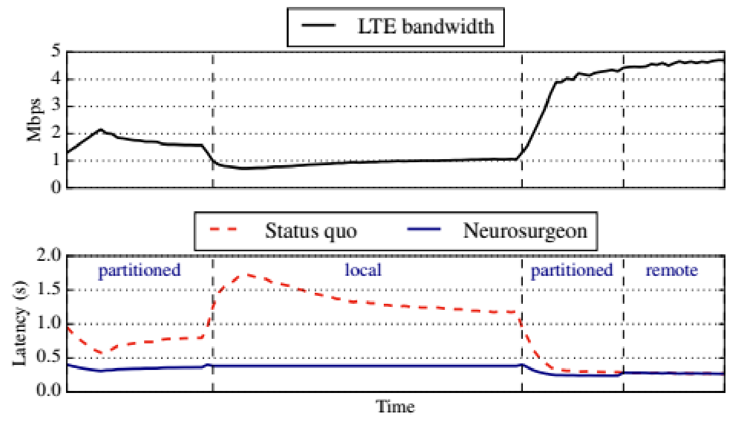


Figure 14: The top graph shows bandwidth variance using a LTE network. The bottom graph shows the latency of AlexNet (IMC) of the status quo and Neurosurgeon. Neurosurgeon's decisions are annotated on the bottom graph. Neurosurgeon provides consistent latency by adjusting its partitioned execution based on the available bandwidth.

- ❖ Server load variation(Datacenter)
  - Diurnal load pattern

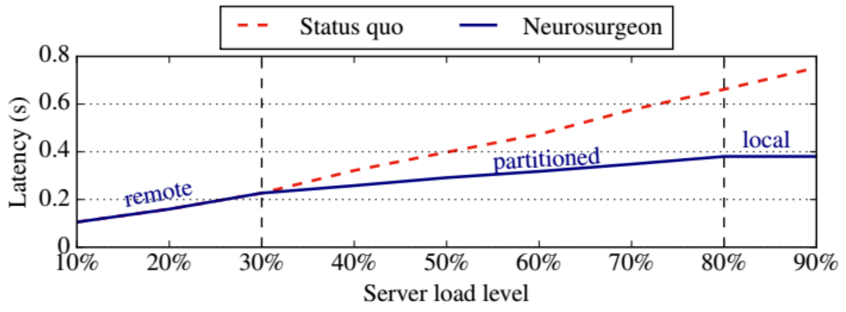


Figure 15: Neurosurgeon adjusts its partitioned execution as the result of varying datacenter load.

Results : Test Throuput

- Neurosurgeon onloads part or all of the computation from the cloud to mobile devices to improve end-to-end latency and reduce mobile energy consumption.
- Use BigHouse

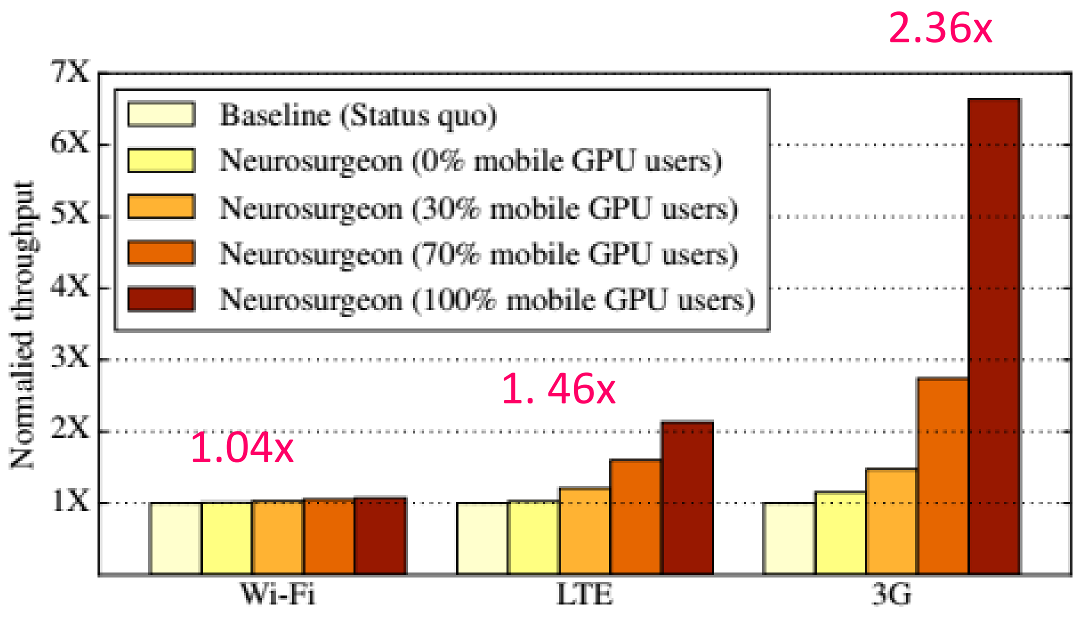


Figure 16: Datacenter throughput improvement achieved by Neurosurgeon over the status quo approach. Higher throughput improvement is achieved by Neurosurgeon for cellular networks (LTE and 3G) and as more mobile devices are equipped with GPUs.

Conclusion

- Deep Neural Networks have been traditionally executed in the cloud.
- In this work,
  - Examine the efficacy of this status quo approach of cloud-only processing
  - Show that it is not always optimal to transfer the input data to the server and remotely execute the DNN.
  - Investigate the compute and data characteristics of 8 DNN architectures spanning computer vision, speech, and natural language processing applications
  - Show the trade-off of partitioning computation at different points within the neural network.
- With these insights
  - Develop Neurosurgeon, which can automatically partition DNN between the mobile device and cloud at the granularity of neural network layers.
- **Neurosurgeon**
  - Adapts to various DNN architectures, (hardware platforms, wireless connections, and server load levels),
  - Chooses the partition point for best la tendancy and best mobile energy consumption.
  - Achieves on average 3.1⇥ and up to 40.7⇥ latency speedup,
  - Reduces mobile energy consumption by on average 59.5% and up to 94.7%
  - Improves datacenter throughput by on average 1.5⇥ and up to 6.7⇥.

Wireless network

	WiFi **(Wireless Lan, WLAN)**	**LTE (4G LTE,** **4th generation long term evolution)**	**3G** **(3rd generation())**
Standard	WiFi IEEE 802.11b high rate	3GPP release9	
Speeds	11Mbps	**100Mbps**	128Kbps~2Mbps

Tech.	Various..!	OFDMA(for DL)
		DFTS-OFDM(for UL)
			WCDMA

Coverage	500m (in home or office with access point)		
Bandwidth (Frequency)	2.4GHz	25MHz	2GHz
Contents	Large graphic, video, audio at least connected 5 PCs		Video, Comics, news, internet broadcasting, VOD