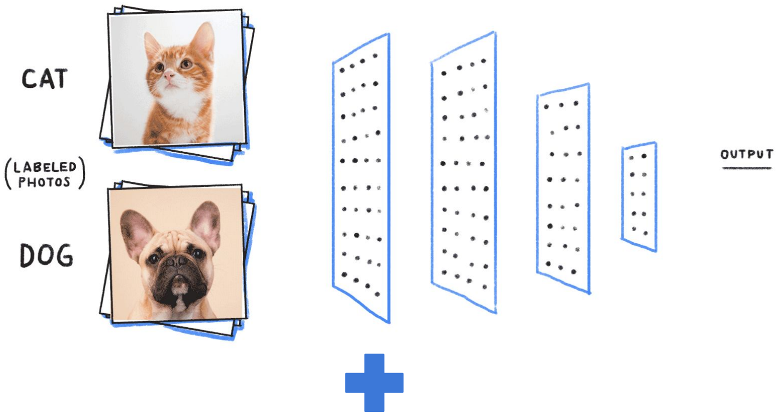
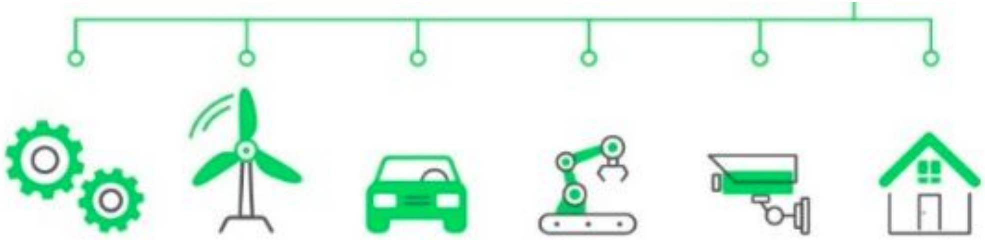
Distributing Deep Neural Networks with Containerized Partitions at the Edge

Li Zhou[1], Hao Wen[2], Radu Teodorescu[1], David H.C. Du[2]

Today's Problem: Move ML to the Edge





- ML: DNNs, CNNs
- Tasks: computer vision (image recognition, object detection), natural language processing, etc.

- IoT: edge devices, 5G
- Applications: smart homes, cities, autonomous cars, etc.

➢ Run ML on IoT devices

Figures are from internet sources

DNNs at the Edge

Cloud computing (model is executed by cloud-only, or edge-cloud collaboration)

- Privacy concerns (GDPR)
- Cloud availability, data transfer latency

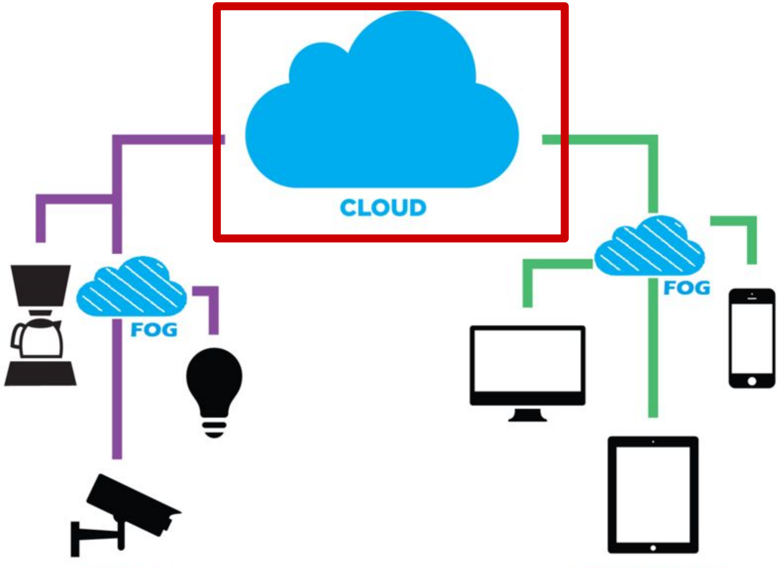
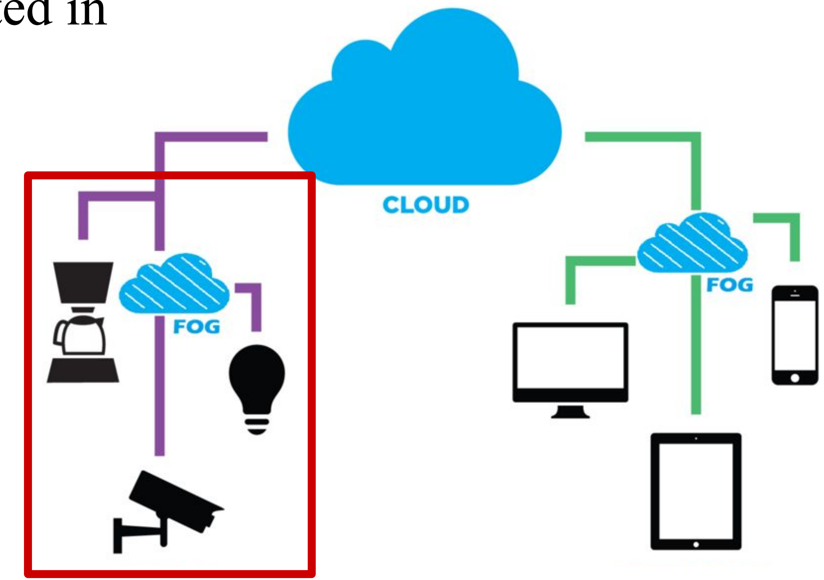


Figure is from internet sources

DNNs at the Edge

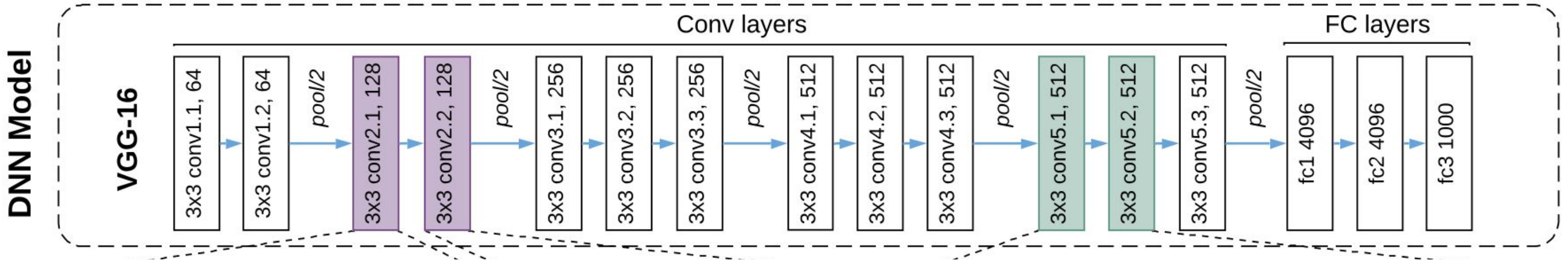
Edge computing (model is stored and executed in edge devices)

- Model compression: smaller model
- Customized accelerator: faster device
- Pipeline: throughput improved
- Parallelization: model **1) partition** and **2) parallelization**



➢ Goal: Efficient deployment of DNNs inference at the Edge with IoT devices

Model Partition Problem and Contribution



- How to partition the workload efficiently across devices?
- How to account for the higher communication latency in wirelessly connected devices?
- How to optimize execution across heterogeneous devices possessing different compute capabilities?

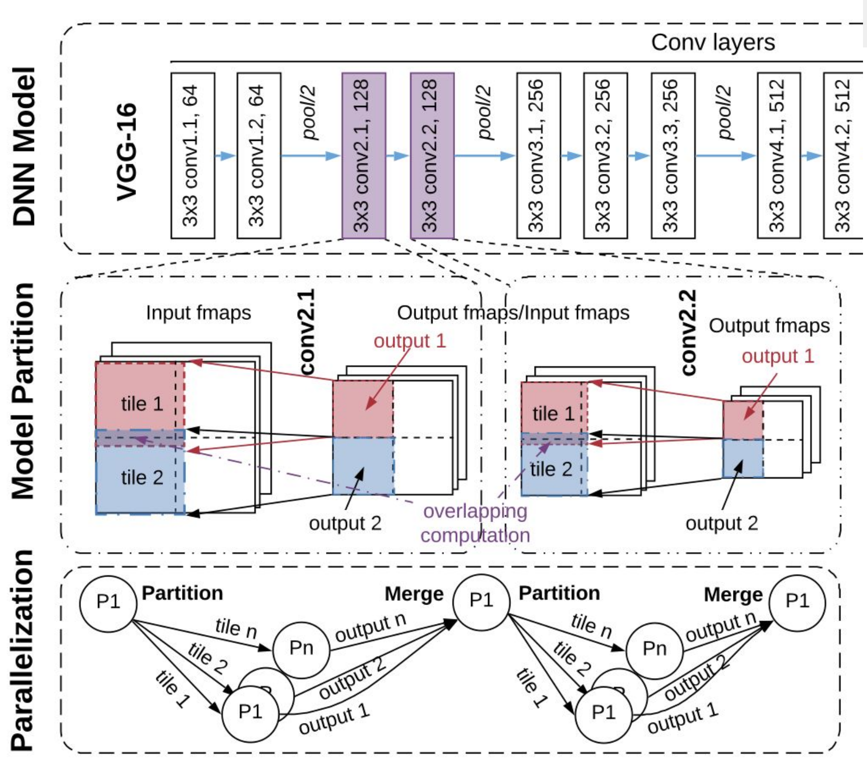
Design Overview

Hardware Profiling	Model Partition and Parallelization	Deployment
Linear regression **models** for ● DNN layers ● Network	Find out the optimal **partition points** for a given model based on current computational resources and network	**Distribute** and **execute** containerized partitions across edge devices through k8s

Layer-wise Parallelization

- Each layer is parallelized independently
- Each device compute a subset of output
- Partial outputs are aggregated, partitioned before the execution of the next layer

➢ Substantial comm overhead
➢ Benefits may be defeated

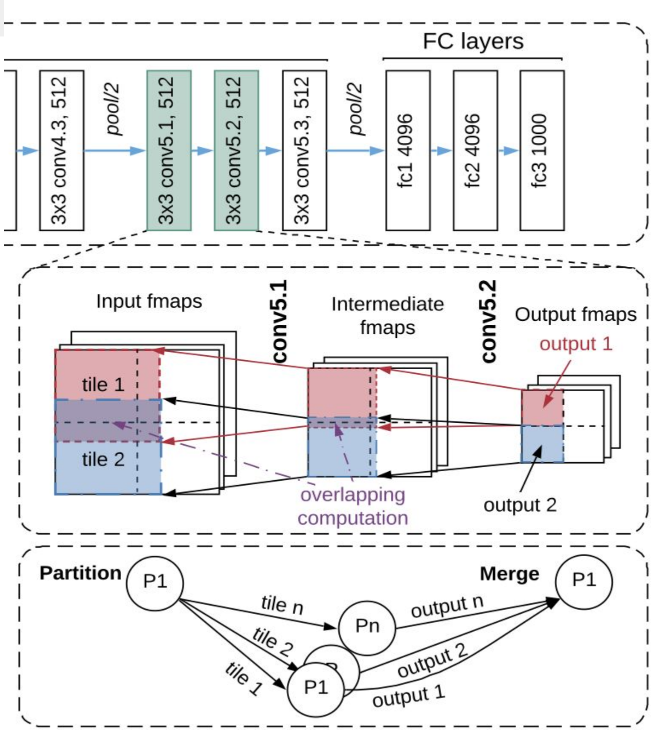


(a) Layer-wise

Fused-layer Parallelization

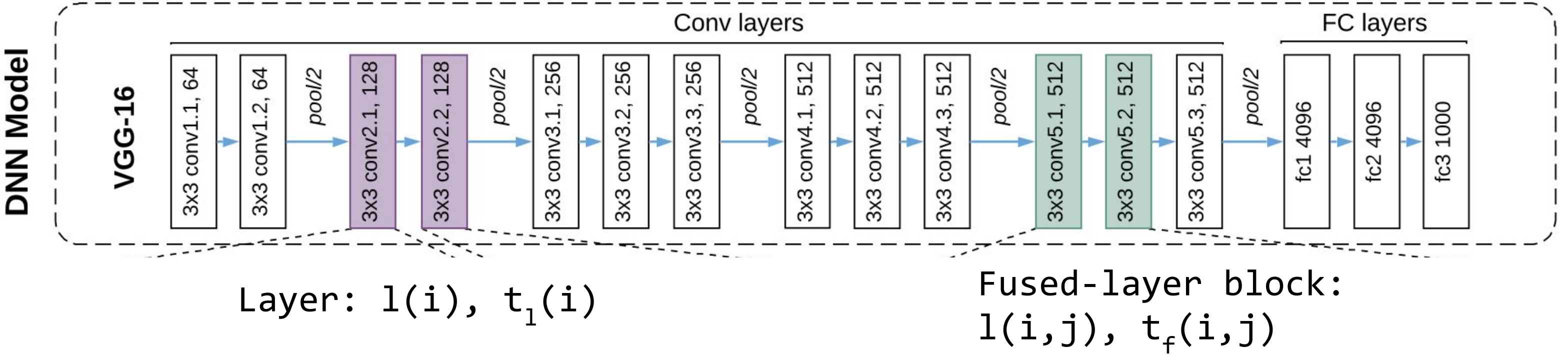
- Consecutive *conv* layers are first fused as a block
- Partitioning is performed layer-by-layer starting from the last layer in the fused block.
- Input elements are recursively calculated based on output
- Extend conv layer by $\lfloor f_i/2 \rfloor$ on height and width

➢ Less comm, more comp



(b) Fused-layer

Dynamic Programming based Search

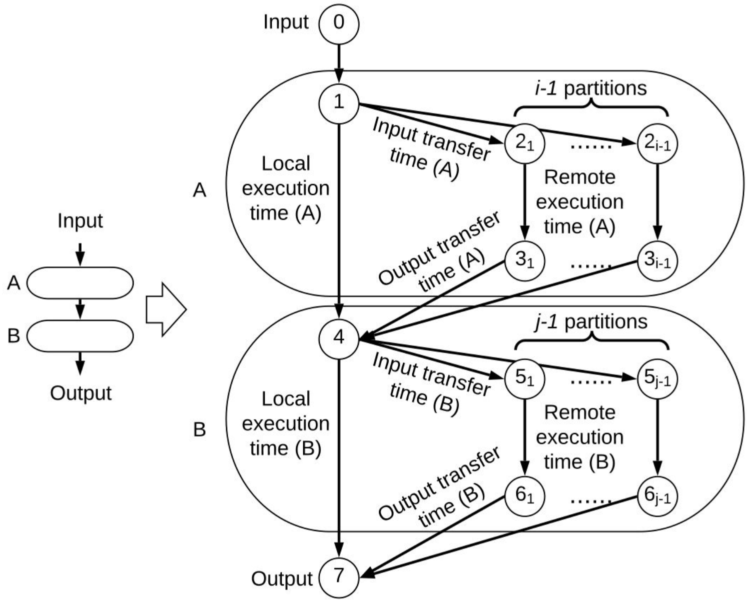


- Run a model G on a list of devices D with parallelization S
  - $G = \mathcal{G}^{lw} \cup \mathcal{G}^{fl}, \mathcal{S} = \mathcal{S}^{lw} \cup \mathcal{S}^{fl}$
  - Minimize: $\mathcal{T}(G, D, S) = \sum l_i \in \mathcal{G}^{lw}\ t_l(i) + \sum l_{(i,j)} \in \mathcal{G}^{fl}\ t_l(i, j)$
- We solve the problem with memorized DP, search all the possible combinations, to minimize $\mathcal{T}$

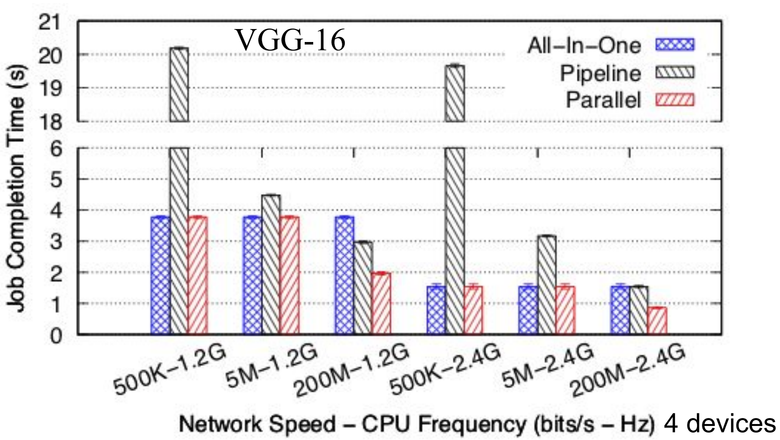
Deployment

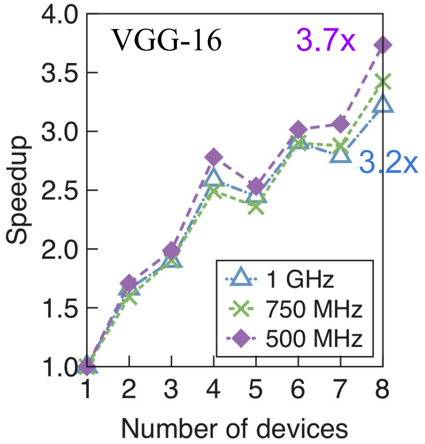
- Docker container + Kubernetes
- Small scheduling unit (flexibility): each partition runs in a pod
- Reduce config complexity: all pods are running based on the same Docker image

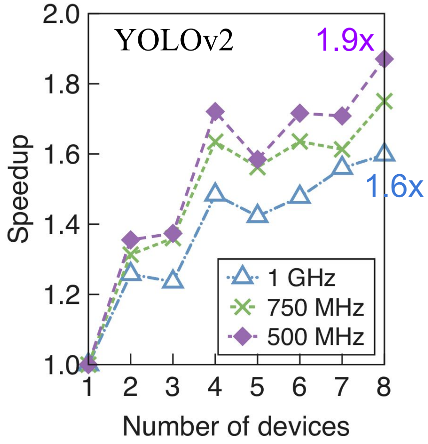
- **Pipeline**: partitions belonging to different blocks form a pipeline (one device for each stage in evaluation).
- **Parallel**: partitions belonging to the same block run in parallel



Latency Improvement





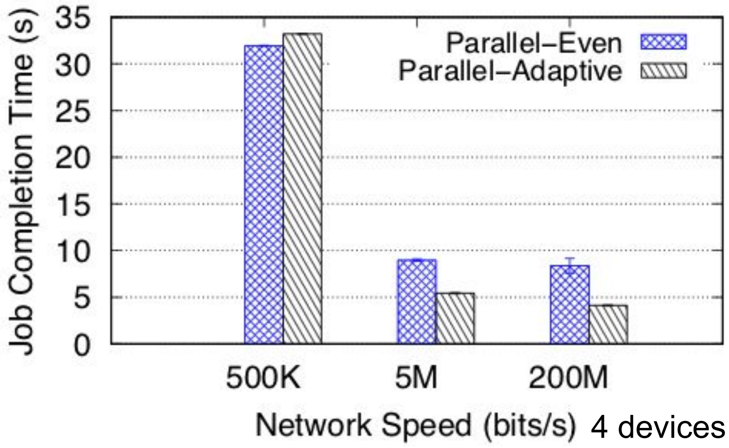


Speedup is relative to 1 device at 1GHz/750MHz/500MHz in 100Mbps WIFI

- Limited by network
- Speedup: VGG-16: 1.6x ~ 3.7x, YOLOv2: 1.3x ~ 1.9x
- Implication: more improvements are observed under faster network, and less compute capability

Heterogeneity

- The input portion sizes are adjusted to the devices computing capability, e.g., one device runs at 1/2 compute capability of other devices
- Slow devices may be dropped
- We achieve a reduction of 51% and 39% in execution time respectively in *medium* and *fast* network.



4 devices

Summary

- This work proposes a collaborative convolutional neural network (CNN) acceleration framework for Internet-of-things (IoT)
- The framework leverages spatial partitioning through fusion of the convolution layers
- A dynamic programming-based search algorithm has been proposed to decide the optimal partition and parallelization for a DNN model
- Achieves 1.3x ~ 3.7x speedup with 8 edge devices for two popular CNN models

Discussion

- Feedback
  - Granularity for DNN containerizations: *model* vs *partition*
  - Impact of 5G for ML at the edge
- Controversial points
  - Efforts on *parallelizing model* vs *customized accelerator or model compression*
- Open issues
  - Efficient resource management and scheduling
  - QoS (Quality of service), Fault-tolerance
- Fall apart
  - Optimal solution falls into using 1 device under *faster* devices, *slower* network
  - Complex model like DenseNet

Thank you!