End-Edge Cooperative Inference of Deep Learning Model Based on DNN Partition

Yuwei Wang

Institute of Computing Technology Chinese Academy of Sciences

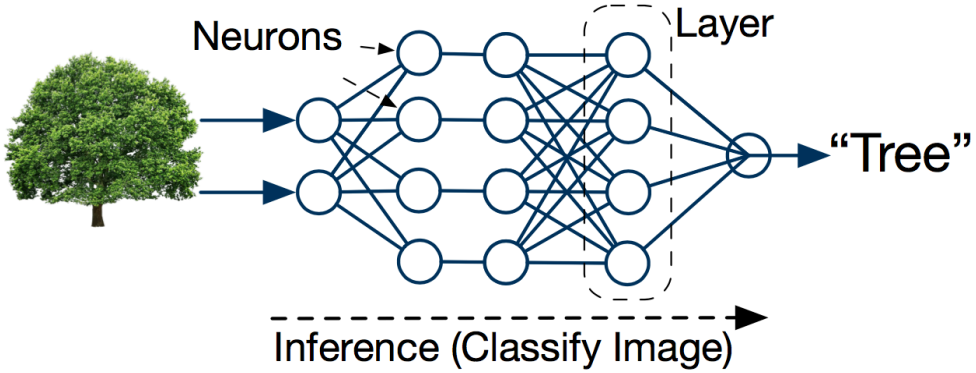
Dec 2020

Outline

- Background
- Related Works
- Research Objectives and Contents
- Our Solutions
- Experimental Result
- Progress of Scientific Research and Courses

Research Background (1/3)

- Deep Neural Network (DNN) models are gradually emerging
  - As smart phones are made more personalized and intellectualized , wearable devices and smart home devices are becoming more and more popular, the interaction mode among mobile devices is changing rapidly
  - Deep Neural Networks (DNN) is widely used in computer vision, speech and natural language processing due to its high precision



Deep learning model inferencing refers to the forward propagation process of deep learning model

Research Background (2/3)

- Currently, great majority of the deep learning model inference is executed in the cloud data center, and a small part is executed on the mobile devices
  - Issues of cloud data center
    - Huge amount of data must be uploaded to the cloud and transmission delay is too long, and it will lead to high risk of network congestion.
    - Massive data transmission leads to high energy costs.
    - High risk of user privacy data disclosure
  - Issues of mobile devices
    - The resource of mobile devices nowadays is still limited. Running DNN model independently will lead to high computation latency and high energy consumption.

Research Background (3/3)

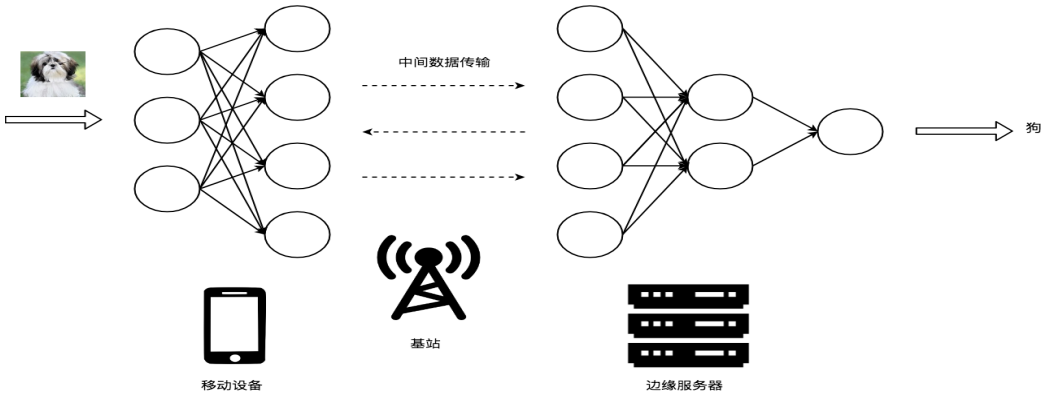
- Mobile edge computing (MEC) has brought new possibilities for the implementation of the DNN model inference

Advantages of end-edge collaboration for the implementation of the DNN model inference :

- Shorten the transmission distance effectively
- Reduce inference latency
- Protect privacy of devices
- Save core network resources

Important research issues during the end-edge collaborative inference process :

- How to automatically divide DNN computing tasks between end/edge devices and how to reduce the overall inference latency



Inference latency = End devices computation latency+ Data transmission latency + Edge devices computation latency

Outline

- Background
- Related Works
- Research Objectives and Contents
- Our Solutions
- Experimental Result
- Progress of Scientific Research and Courses

Related Works

- Estimation of computation latency in DNN partition

No prediction model

- JALAD[1] 、DADS[2]: The computation latency of each layer is measured in advance
- MoDNN[3]: Dividing the amount of computation of different DNN layers by the fixed computing capacity to obtain the computation latency

Computation latency prediction model

- Edge Intelligence[4] : Taking DNN layer as granularity, a linear regression model was established to predict the computation latency of each layer
- Neurosurgeon[5] : Use log or linear function to predict the computation latency of DNN in each layer

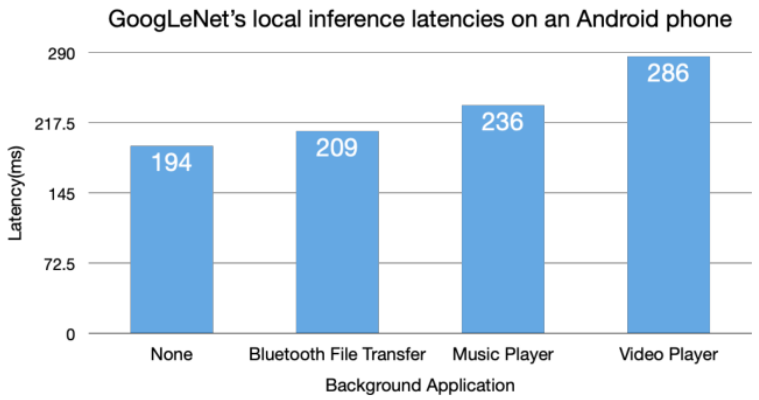


Figure 2: GoogLeNet's local inference latencies under different device loads.

The existing strategies do not consider the change of load of devices when estimating the computation latency. The estimation of the computation latency is not accurate when other tasks are running on the devices.

[1] Li H, Hu C, Jiang J, et al. JALAD: Joint Accuracy-And Latency-Aware Deep Structure Decoupling for Edge-Cloud Execution[C]. International Conference on Parallel and Distributed Systems, 2018: 671-678.

[2] Hu C, Bao W, Wang D, et al. Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge[C]. International Conference on Computer Communications, 2019: 1423-1431.

[3] Mao J, Chen X, Nixon K W, et al. MoDNN: Local distributed mobile computing system for Deep Neural Network[C]. design, automation, and test in europe, 2017: 1400-1405.

[4] Li E, Zhou Z, Chen X, et al. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy[C]. acm special interest group on data communication, 2018: 31-36

[5] Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[C]. architectural support for programming languages and operating systems, 2017, 45(1): 615-629.

Outline

- Background
- Related Works
- Research Objectives and Contents
- Our Solutions
- Experimental Result
- Progress of Scientific Research and Courses

Research Objectives and Contents

The aim of the research on deep learning model end-edge collaboration inference is to optimize the end-to-end inference latency and improve the collaboration efficiency of end-to-edge devices.

DNN partition algorithm

- Topology of DNN
- Calculation and data characteristics of each layer of DNN
- Gap between equipment capabilities and computing power
- Network status during runtime

Computation latency prediction model adapting the devices' load

- Different DNN layer types have different computation characteristics.
- The computation latency varies when different devices execute the same DNN layer. It is related to the hardware capability which executes DNN.
- It is related to the dynamic load of the devices which execute DNN

We have an ITU standard F.AI-DMPC under development on" Technical framework for Deep Neural Network model partition and collaborative execution" in ITU-T SG16 Q5

Outline

- Research Background and Significance
- Current Research Status
- Research Objectives and Contents
- Solutions
- Experimental Result
- Progress of Scientific Research and Courses

DNN Partition Algorithms(1/7)

- Definition of DNN partition problems

Computation latency of end devices

Data transmission latency

Computation latency of edge devices

$$\min \boxed{\sum_{i=1}^{N} m_i T_{mobile_i}} + \boxed{T_{computation}} + \boxed{\sum_{i=1}^{N} e_i T_{edge_i}}$$

$$T_{communication} = \sum_{i=1}^{N} m_i \max_{j=1\ldots N}(e_j T_{upload_{i,j}}) + \sum_{i=1}^{N} e_i \max_{j=1\ldots N}(m_j T_{download_{i,j}})$$

$$+ e_1 T_{upload_0} + e_N T_{download_N}$$

$$s.t.\ m_i + e_i = 1 \quad \forall i \in \{i, 2, \ldots, N\}$$

DNN Partition Algorithms(2/7)

- Characteristics of DNN partition problems
  - Decomposable
  - Have the optimal substructure property

  → Dynamic programming (DP) algorithms

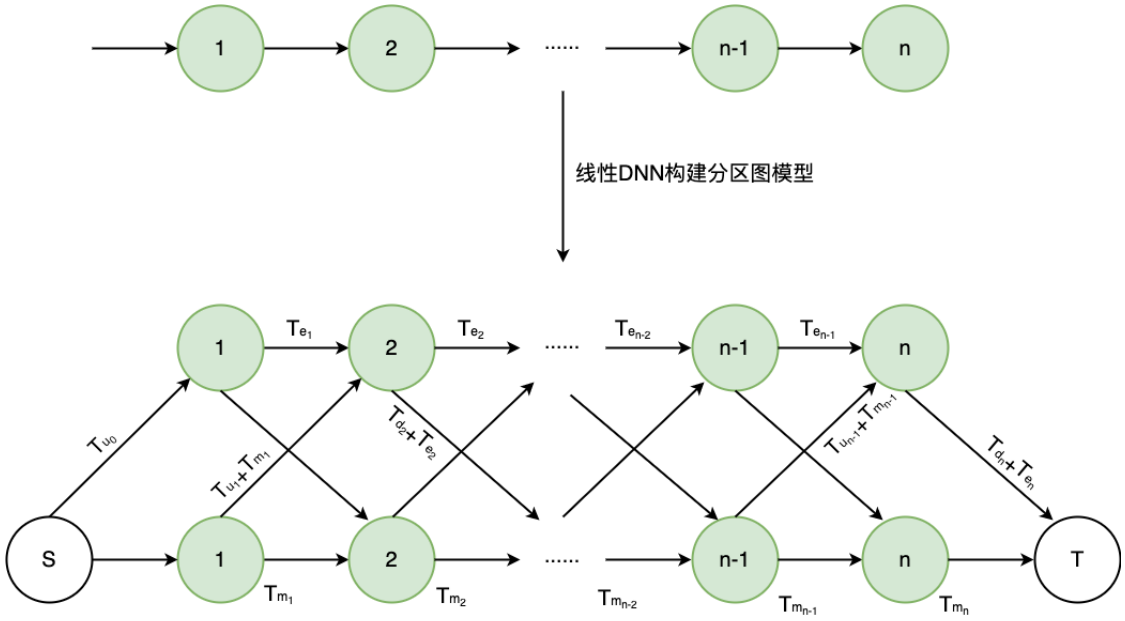
- Definition of DNN partition subproblems
  - Consider a DNN model with n network layers, they are labeled as 1,2,...,n according to the topological sorting.
  - First, we divide the original DNN partition problem into two sub-problems:
    - Sub-problem1： When the i^{th} DNN layer is executed on the end device, we aim to minimize the inference latency from the input layer to the i-th DNN layer. The optimal solution of this sub-problem is denoted as $OPT_{mobile}(i)$.
    - Sub-problem2： When the i^{th} DNN layer is executed on the edge device, we aim to minimize the inference latency from the input layer to the i^{th} DNN layer. The optimal solution of this sub-problem is denoted as $OPT_{edge}(i)$.
  - The optimal substructure properties of the problems can be proved by the “cut-paste” method.

DNN Partition Algorithms(3/7)

- Chain structure
  - The DP state transition equations

$$OPT_{mobile}(node) = T_{m_{node}} + \min\{OPT_{mobile}(last\_node),\ OPT_{edge}(last\_node) + T_{d_{last\_node}}\} \quad (3.4)$$

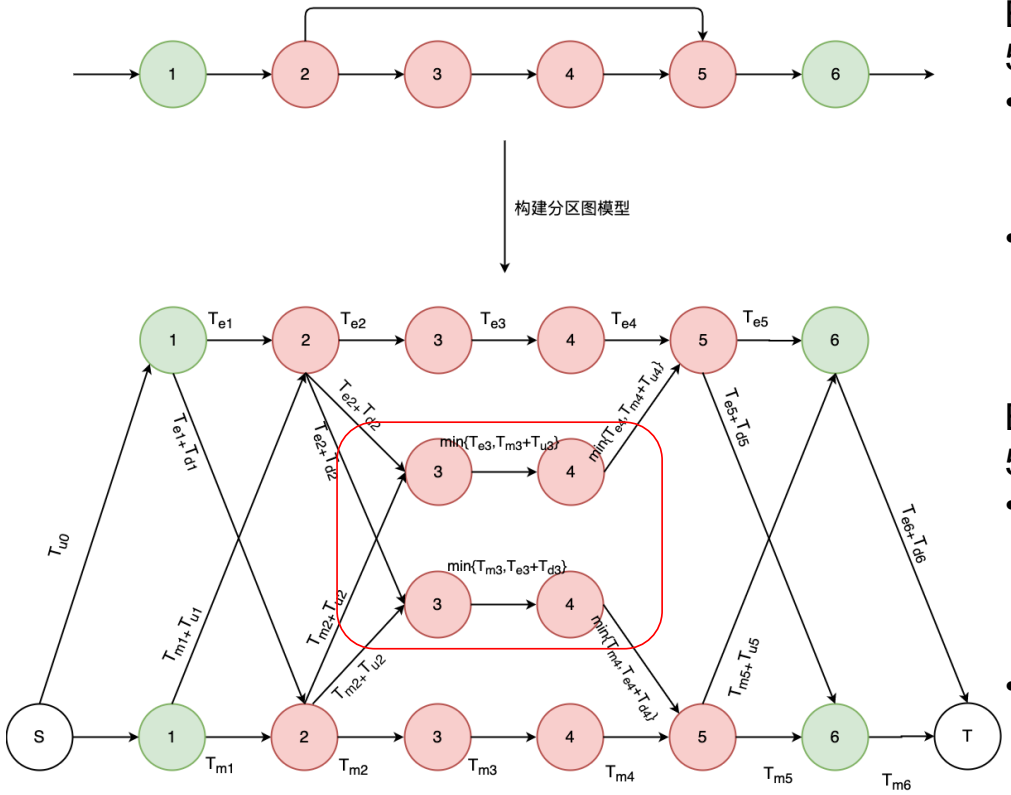
$$OPT_{edge}(node) = T_{e_{node}} + \min\{OPT_{edge}(last\_node),\ OPT_{mobile}(last\_node) + T_{u_{last\_node}}\} \quad (3.5)$$



- Record the source of optimal solution of subproblems
- Find the optimal solution by backtracking

DNN Partition Algorithms(4/7)

- State transition analysis- Simple branch structure 1
  - Only an additional edge dependent branching is added between the node of start of the branch and the node of the end of the branch



Branch source node 2 and branch sink node 5 are executed on the same device

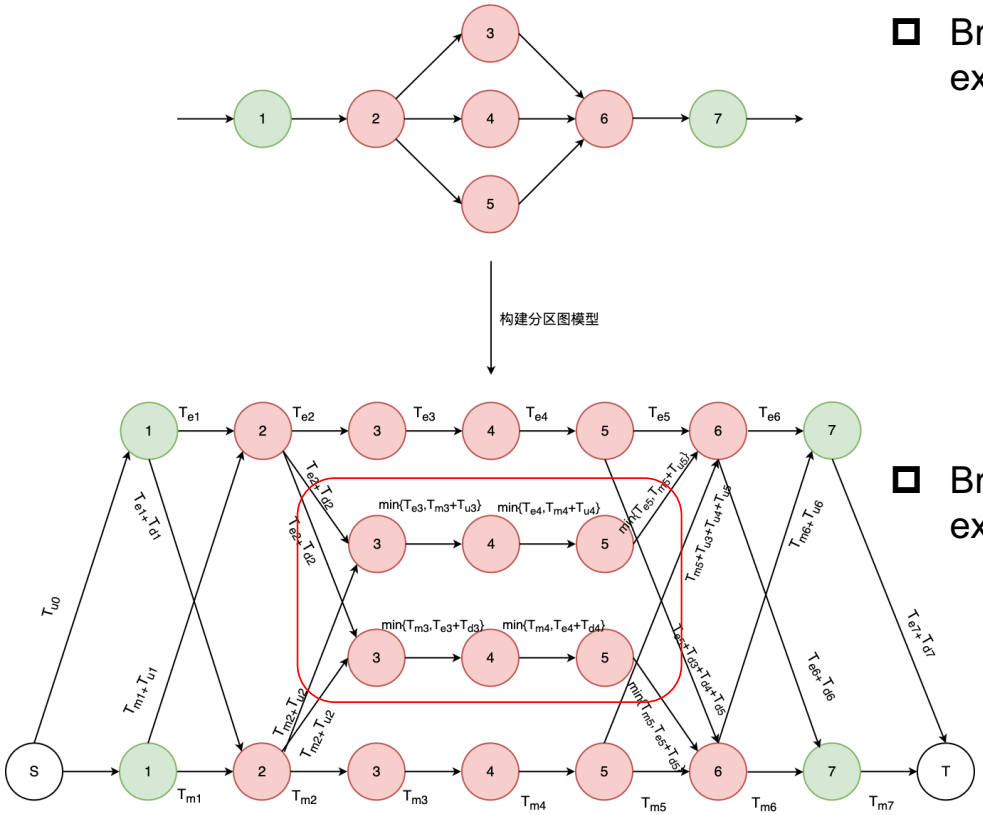
- Node 3 and 2 are executed on the same device, the output data of node 2 does not need to be transmitted
- Node 3 and 2 are executed on the different device, the output data of node 2 needs to be transmitted

Branch source node 2 and branch sink node 5 are executed on different devices

- Node 3 and 2 are executed on the same device, node 5 is executed on the other device, the output data of node 2 needs to be transmitted
- Node 3 and 2 are executed on the different device, node 5 is executed on the same device as node3, the output data of node 2 needs to be transmitted

DNN Partition Algorithms(5/7)

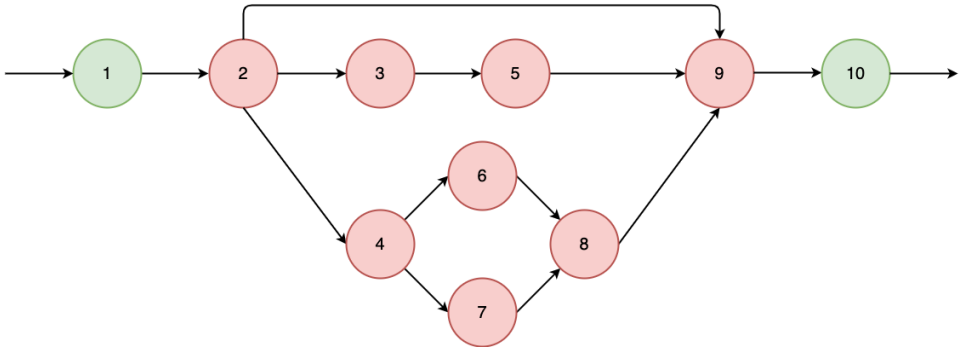
- State transition analysis- Simple branch structure 2
  - Only one branch intermediate node is added to each branch

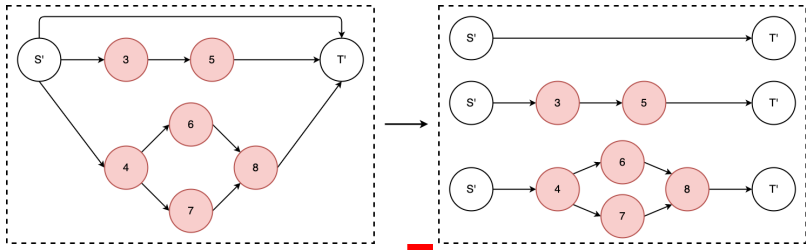


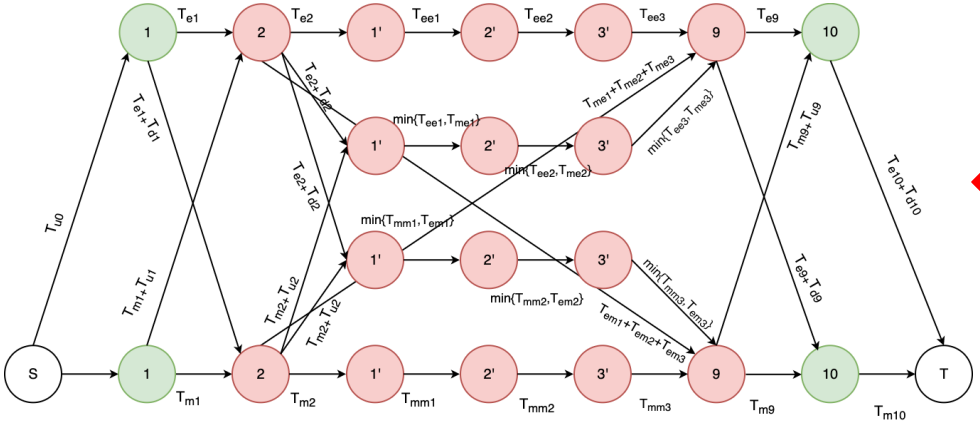
- Branch source node 2 and branch sink node 6 are executed on the same device:
  - Node 3, 4, 5, branch source node 2 and branch sink node 6 are executed on the same device，the output data of branch source node does not need to be transmitted
  - Node 3, 4, 5 are executed on the different device with branch source node 2 and branch sink node 6，the output data of branch source node needs to be transmitted
- Branch source node 2 and branch sink node 6 are executed on different devices:
  - Node 3, 4, 5, branch source node 2 and are executed on the same device，the output data of branch source node does not need to be transmitted
  - Node 3, 4, 5 are executed on different device with branch source node 2, the output data of branch source node needs to be transmitted

DNN Partition Algorithms(6/7)

- State transition analysis- Complex branch structure







For the newly generated k DNN, inference latency of 4 situations is calculated

- $S'$is executed on end devices , T' is executed on end devices , the shortest path length obtained is recorded as T_{mm_i};
- S' is executed on end devices , T' is executed on edge devices , the shortest path length obtained is recorded as T_{me_i};
- S' is executed on edge devices , T' is executed on end devices, the shortest path length obtained is recorded as T_{em_i};
- S' S' is executed on edge devices , T' S' is executed on edge devices, the shortest path length obtained is recorded as T_{ee_i};

DNN Partition Algorithms(7/7)

- Branching structure
  - DP state transition equations

$$OPT_{mobile}(branch\_end\_node) = \min\{$$

$$OPT_{mobile}(branch\_begin\_node) + \sum_{i=1}^{k} T_{mm_i},$$

$$OPT_{mobile}(branch\_begin\_node) + T_{u_{branch\_begin\_node}} + \sum_{i=1}^{k} \min\{T_{mm_i}, T_{em_i}\},$$

Branch source node is not transmitted

$$OPT_{edge}(branch\_begin\_node) + \sum_{i=1}^{k} T_{em_i},$$

$$OPT_{edge}(branch\_begin\_node) + T_{d_{branch\_begin\_node}} + \sum_{i=1}^{k} \min\{T_{mm_i}, T_{em_i}\}$$

Branch source node is not transmitted

$$\} + T_{m_{branch\_end\_node}}$$

$$OPT_{edge}(branch\_end\_node) = \min\{$$

$$OPT_{edge}(branch\_begin\_node) + \sum_{i=1}^{k} T_{ee_i},$$

$$OPT_{edge}(branch\_begin\_node) + T_{d_{branch\_begin\_node}} + \sum_{i=1}^{k} \min\{T_{ee_i}, T_{me_i}\},$$

$$OPT_{mobile}(branch\_begin\_node) + \sum_{i=1}^{k} T_{me_i},$$

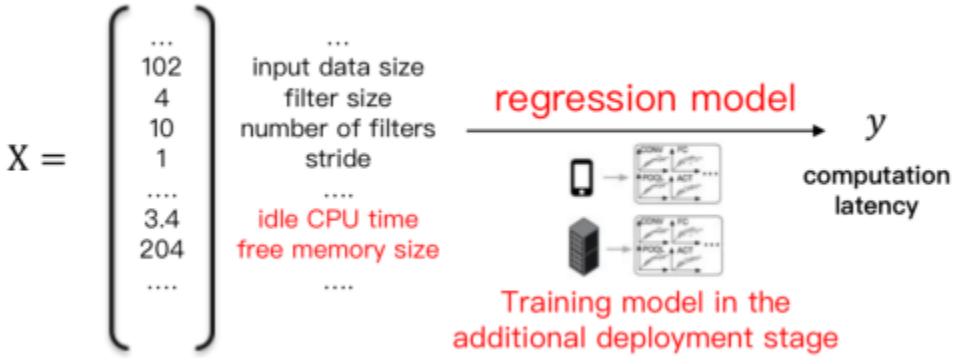
$$OPT_{mobile}(branch\_begin\_node) + T_{u_{branch\_begin\_node}} + \sum_{i=1}^{k} \min\{T_{ee_i}, T_{me_i}\}$$

$$\} + T_{e_{branch\_end\_node}}$$

Computation Latency Prediction Model (1/4)

- Self-adaption of variation of equipment load
  - Accurate prediction of DNN task computation execution time is very important for the accuracy of partition position selection
  - Different DNN network layers have different computing characteristics. It is necessary to establish a prediction model of layer granularity
  - Multiple intelligent applications may run on a device simultaneously, and the device load changes dynamically

The CPU and memory load of the device are considered when the computation latency is predicted



Computation Latency Prediction Model (2/4)

- Feature extraction & selection
  - Load-dependent dynamic characteristics are introduced

Name	Static Characteristics		Dynamic Characteristics
Our Solution	conv	Input Dimension; Convolution Kernel Quantities/Size/Step Size; Padding Type; Activation Type; Biased	Idle CPU time System CPU time CPU Utilization Free Memory Size Memory Usage
	pooling	Input Dimension; Output Dimension; Pool Type/Size/Step Size; Padding Type;	
	fc	Input Data Size; Number of Neurons; Activation Type; Biased	
	activation	Input Data Size; Activation Type	
	dropout	Input Data Size	
	softmax	Input Data Size; Output Data Size	
	deconv	Input Dimension; Convolution Kernel Quantities/Size/Step Size; Padding Type; Activation Type; Biased	
	lstm	Input Data Size; Number of Neurons;	
	rnn	Input Data Size; Number of Neurons;	
Neurosurgeon[1]	conv	Characteristic Number of Input Characteristic Graph; $(filter\ size/stride)^2 * (number\ of\ filters)$	N/A
	pooling	Input Characteristic Graph Size; Output Characteristic Graph Size	
	fc	Number of Input Neurons; Number of Output Neurons	
	bn	Number of Neurons	
	activation	Number of Neurons	
	softmax	Number of Input Neurons; Number of Output Neurons	
	argmax	Number of Input Neurons; Number of Output Neurons	
Edge Intelligence[2]	conv	Characteristic Number of Input Characteristic Graph; $(filter\ size/stride)^2 * (number\ of\ filters)$	
	pooling	Input Data Size; Output Data Size	N/A
	fc	Input Data Size; Output Data Size	
	bn	Input Data Size	
	softmax	Input Data Size	
	dropout	Input Data Size	

Computation Latency Prediction Model (3/4)

- ## Dynamic Characteristic Gain
  - Neurosurgeon[1] Model：Linear Regression Model | Logarithmic Regression Model
  - Edge Intelligence[2] Model： Linear Regression Model
  - Evaluating Indicator ： Goodness of fit

$$R2= 1-\frac{\sum_i (\hat{y}_i - y_i)^2}{\sum_i (\bar{y}_i - y_i)^2}$$

Layer Type	Linear Regression Model			Logistic Regression Model		
	Static Feature Set	Dynamic Feature Set	Increase Ratio	Static Feature Set	Dynamic Feature Set	Increase Ratio
conv	0.4458	0.5718	28.26%	0.6687	0.7136	6.71%
pooling	0.9604	0.9776	1.79%	0.7239	0.8939	23.48%
fc	0.6461	0.8417	30.27%	0.8321	0.9306	11.84%
bn	0.9535	0.9874	3.56%	0.6269	0.8625	37.58%
activation	0.7395	0.8889	20.20 %	0.6939	0.8583	23.69%
softmax	0.9712	0.9836	1.27%	0.7262	0.9061	24.77%
argmax	0.9535	0.9751	2.27%	0.6747	0.8159	20.93%

[1] Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[C]. architectural support for programming languages and operating systems, 2017, 45(1): 615-629.
[2] Li E, Zhou Z, Chen X, et al. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy[C]. acm special interest group on data communication, 2018: 31-36

Computation Latency Prediction Model (4/4)

- Model Selection
  - Higher model complexity does not indicates better results (over-fitting)

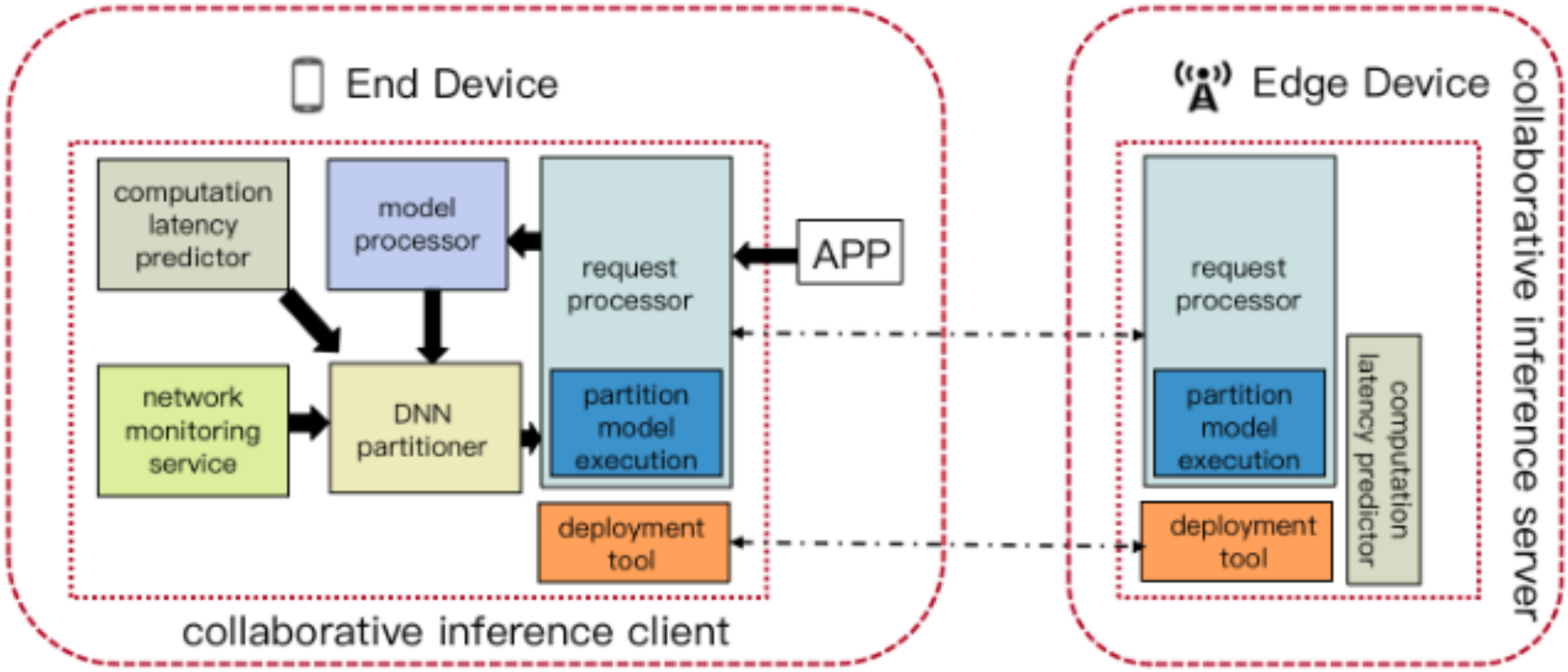
Layer Type	logarithmic	linear	lasso	ridge	kernel ridge	polynomial	GBDT
conv	0.7136	0.5718	0.5733	0.5718	0.5936	**0.9758**	0.7579
pooling	0.8939	0.9776	0.9758	0.9776	0.9909	**0.9962**	0.8595
fc	0.9306	0.8417	0.8411	0.8416	0.9638	**0.9661**	0.7303
bn	0.8625	0.9874	0.9864	0.9873	0.9978	**0.9982**	0.8613
activation	0.8583	0.8889	0.8888	0.8889	0.8936	**0.8991**	0.7151
softmax	0.9061	**0.9836**	0.9835	0.9836	0.9801	0.9749	0.7547
argmax	0.8159	0.9751	0.9749	0.9752	0.9762	**0.9776**	0.7306

Objectives

- Background
- Related Works
- Research Objectives and Contents
- Our Solutions
- Experimental Result
- Progress of Scientific Research and Courses

End-edge Collaboration Inference Framework

- Overall Architecture



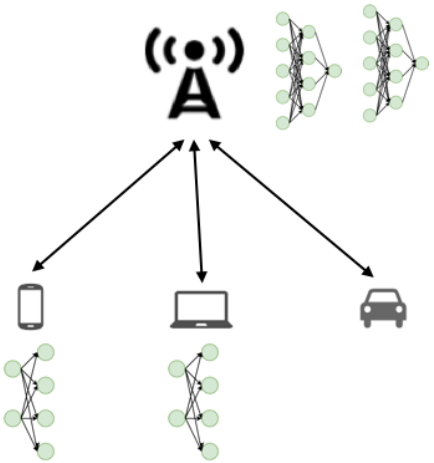
Experimental Settings (1/2)

- Device Configurations

Device	Type	Operating System	CPU	Memory
edge device	MacBook Pro	macOS	64-bit 4-core 2.2GHz	16GB
end device	Huawei glory 9X	Android	2.27GHz(2+6 core)	4GB

- Network Settings

Network speed type	3G	4G	WiFi
download speed (Mbps)	4.6	29.1	54.97
upload speed (Mbps)	1.1	8.8	18.88



[1]OpenSignal.com. 2020.Mobile Network Experience:USA. https://www.opensignal.com/reports/2020/01/usa-/mobile-network-experience
[2] Eshratifar A E, Abrishami M S, Pedram M, et al. JointDNN: An Efficient Training and Inference Engine for Intelligent Mobile Cloud Computing Services[J]. arXiv: Distributed, Parallel, and Cluster Computing, 2018.

Experimental Settings(2/2)

- DNN Setting

Topological Structure	Model Name	Type	Topological Structure	Model Name	Type
Chain DNN	AlexNet	CNN Classification	Non-chain DNN	GoogLeNet	CNN Classification
				ResNet	CNN Classification
	VGG16	CNN Classification		Inception-v3-Net	CNN Classification
				Inception-v4-Net	CNN Classification
	Seq2Seq	RNN Generative		SE-Res-Net	CNN Classification
				MobileNet	CNN Classification
				DeepSpeech	CNN Classification
	DQN	Reinforcement Learning		Chair	CNN Generative
				Pix2Pix	GAN
				VAE	Auto Encoder

Experiment 1 : Validation of Computation Latency Prediction Model

- Baseline : Neurosurgeon
  - Feature set : Static features
  - Regression model: Logarithmic regression model | Linear regression model
- The scheme in this research
  - Feature set : Static features + Dynamic features
  - Regression model: Polynomial regression model | Linear regression model

Layer Type	Neurosurgeon	Our Model	Total Improvement
conv	0.6687	0.9758	45.92%
pooling	0.9604	0.9962	3.73%
fc	0.8321	0.9661	16.10%
bn	0.9535	0.9982	4.69%
activation	0.7395	0.8991	21.58%
softmax	0.9712	0.9836	1.28%
argmax	0.9535	0.9776	2.53%

Experiment 2: Validation of DNN Partition Algorithm (1/3)

- Benchmarks

- Benchmark 1 : The partition scheme proposed in Neurosurgeon [1] is the earliest DNN partition scheme and only supports chained DNNs.

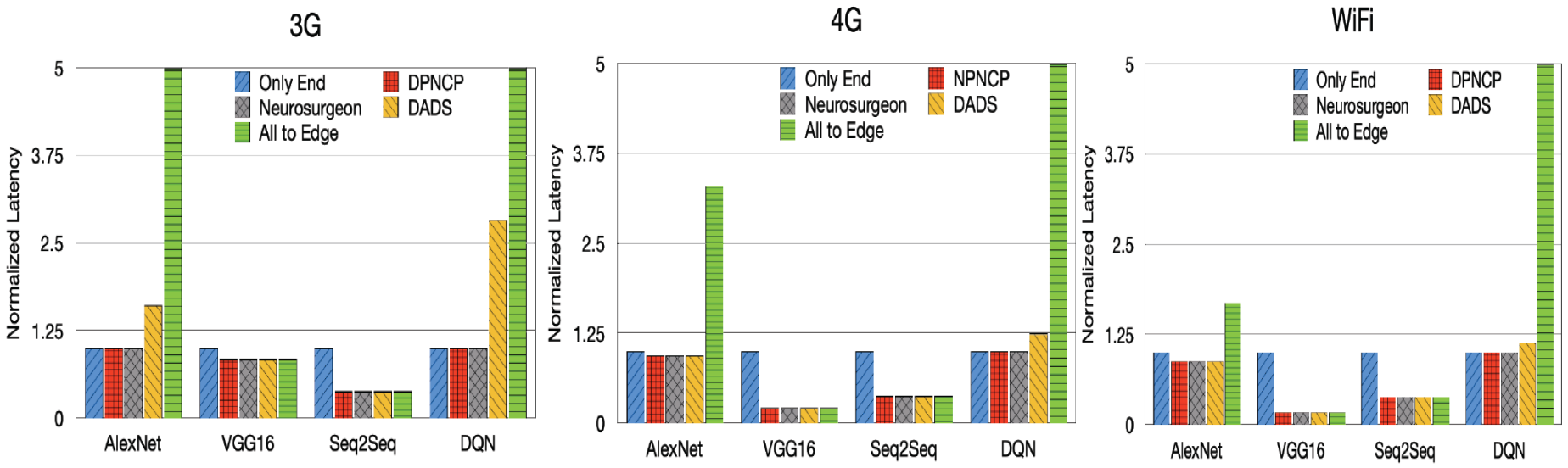
- Benchmark 2 : The partition scheme proposed in DADS [2] is the only work that partitions the branch structure in non-chain DNNs.

- In addition to these two benchmarks, the results of two simple strategies: 1) only the end device executes, and 2) all DNN calculation tasks are offloaded to the edge are also reported.

[1]Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge[C]. architectural support for programming languages and operating systems, 2017, 45(1): 615-629.

[2] Hu C, Bao W, Wang D, et al. Dynamic Adaptive DNN Surgery for Inference Acceleration on the Edge[C]. international conference on computer communications, 2019: 1423-1431.

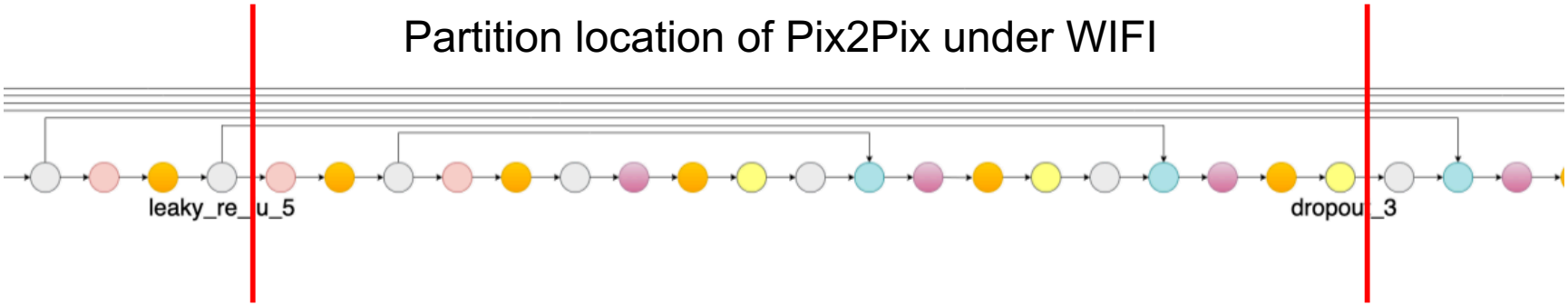
Experiment 2: Validation of DNN Partition Algorithm (2/3)

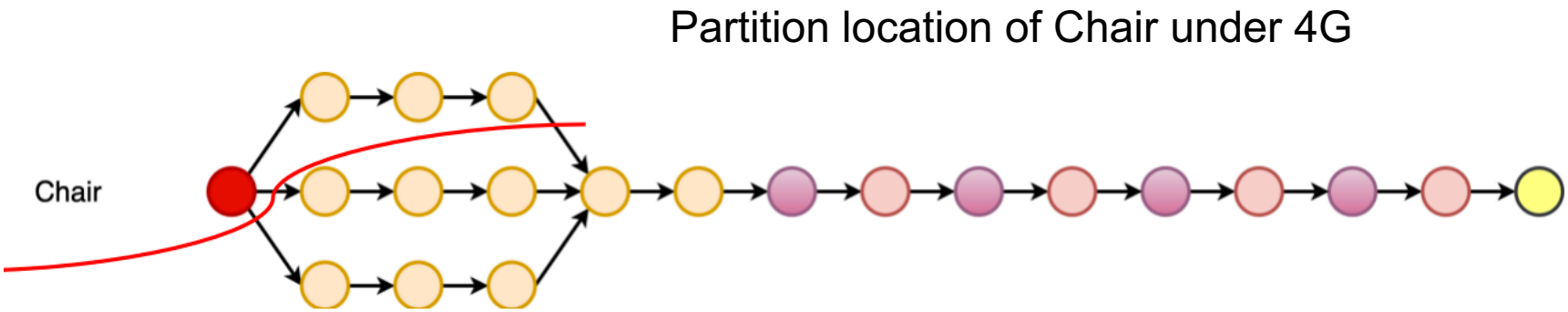


- Different DNNs have different computation characteristics and data characteristics. The performance of the simple strategies that only the end device executes or offloads all DNN computing tasks to the edge varies significantly under different network conditions.
- For the chain DNNs, the scheme in this research and the Neurosurgeon scheme can select the optimal partition position.

Experiment 2: Validation of DNN Partition Algorithm (3/3)

- Internal Cutting of Branch Structure





If the entire branch structure is simply partitioned as a whole, then the optimal solution of end-to-end inference latency will be missed in these two scenarios.

Summary

- An efficient dynamic partition algorithm for DNN will be proposed. The partition problem of DNN will be solved by utilizing dynamic programming method, and the overall latency of single inference of DNN model will be optimized.
- The characteristics related to device load changes such as idle CPU time and memory utilization are added to the DNN layer computation latency prediction model for the first time. At the same time, many common regression models are evaluated through a large number of experiments, which can accurately predict the computation latency of DNN in the real-world scenarios of device load change
- An end-to-edge collaborative inference framework based on DNN partition is designed and executed. Experimental results show that the proposed DNN partition solution can adaptively adjust the partition strategies of DNN model according to its own characteristics and running environment, and significantly reduce the end-to-end inference latency.

Thanks !