Lecture 09: Distributed Machine Learning for Training and Inference

Some Notes

- Project proposals have been graded, and feedback has been posted.
- Lab 2 grades have been released.
- Lab 3 is due this weekend.
- We'll also hold a round of project meetings (~two weeks later) to check on each team's progress.

Final Presentation

- Plan for May 8th (Friday)
  - Each team present for 20mins+5mins QA.
  - A timer will be used.

Topics

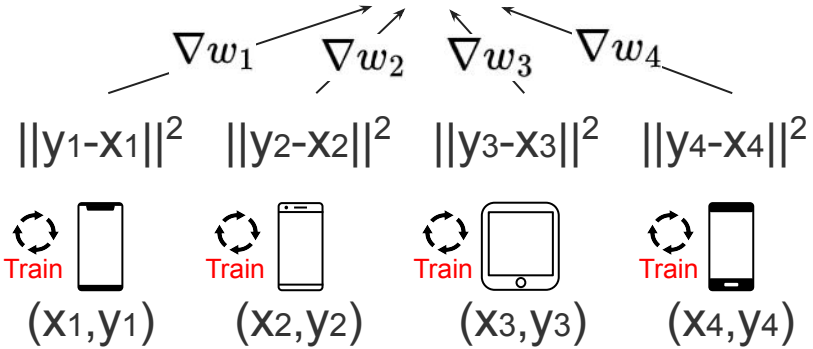
- Federated Learning
- Distributed DNN Training
- Distributed DNN Inference

Federated Learning

- Training data: (x_1,y_1), (x_2,y_2), (x_3,y_3), (x_4,y_4)

$$\sum_{i=1}^{4} ||y_i - F(x_i)||^2$$

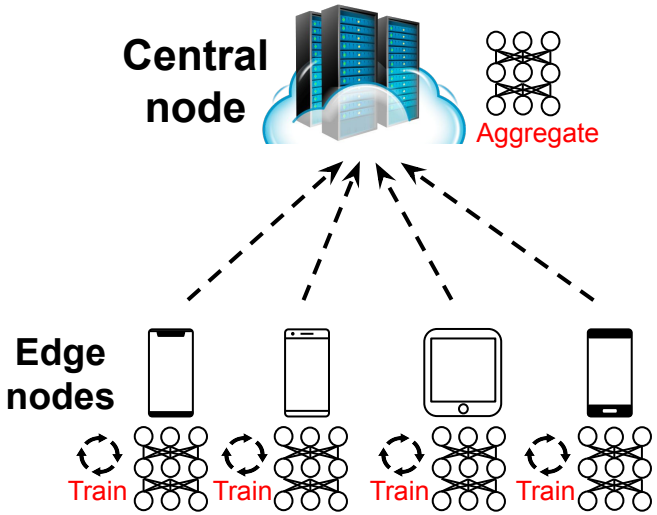
$$\nabla w = \frac{\nabla w_1 + \nabla w_2 + \nabla w_3 + \nabla w_4}{4}$$



- Non-iid training data distribution
- Heterogeneity among the edge devices
- Communication error

NYU SAI LAB

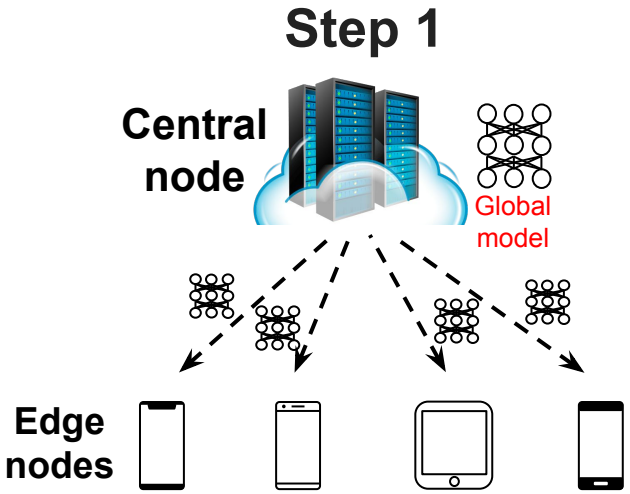
Federated Learning



- Federated learning is a machine learning technique that allows the training of models across multiple decentralized nodes holding local data samples, **without exchanging their data**.
- This approach enhances privacy, user can train the powerful DNN model **without sharing the dataset**.

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

FedAvg

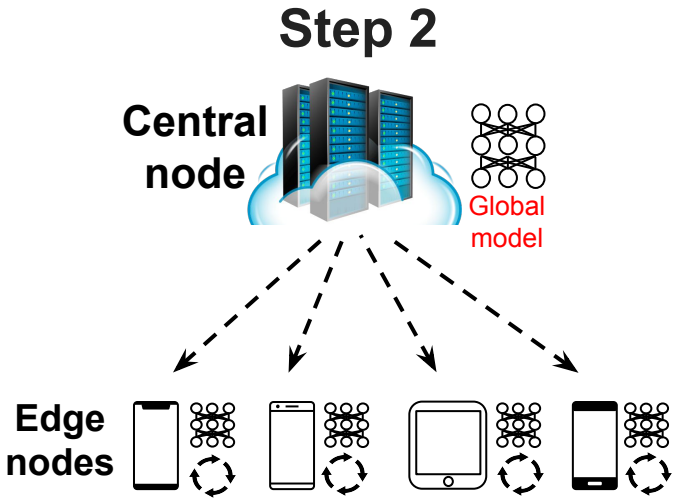


- A global model is initialized on the central node and sent to all participating nodes .

$$w_i = w_{global} \quad \text{For each i}$$

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

FedAvg

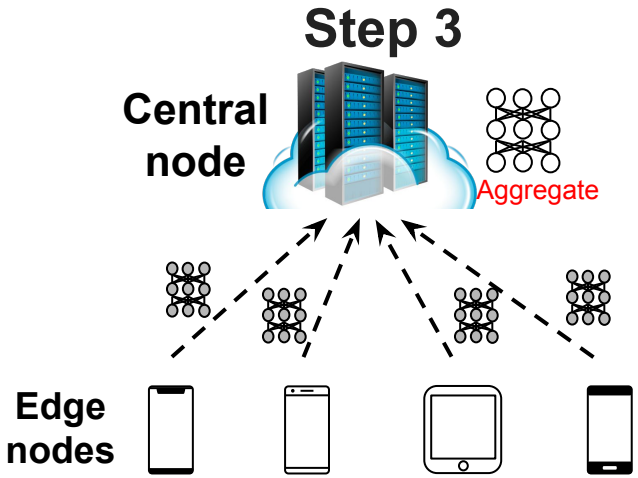


$$w_i' = \underset{w_i}{min} L(F(D_i), Y_i)$$

- Each node i trains the global model locally using its own data for a few epochs.
- The length of local training process may vary.

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

FedAvg

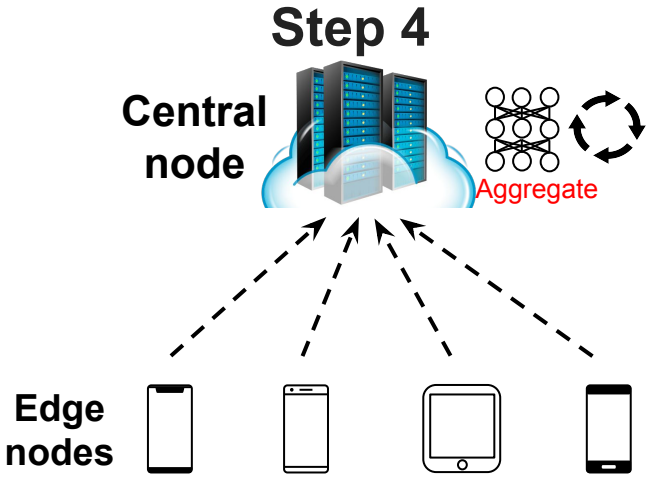


$$\Delta w_i = w - w_i$$

- Local updates are sent from each node to the central node.

McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

FedAvg



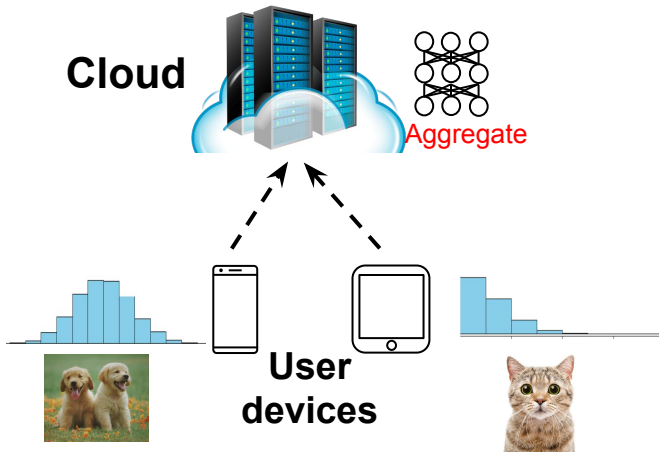
- The central node aggregates the local updates to update the global model.

$$w + \frac{1}{N} \sum_i \Delta w_i$$

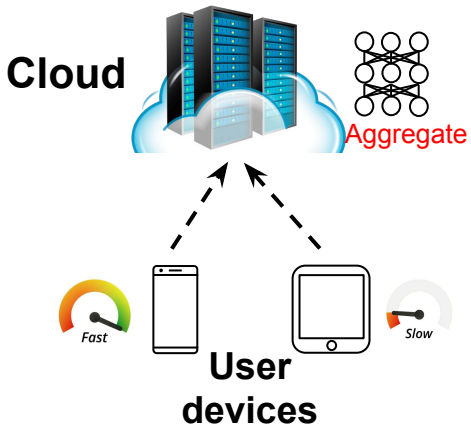
McMahan, Brendan, et al. "Communication-efficient learning of deep networks from decentralized data." *Artificial intelligence and statistics*. PMLR, 2017.

Federated Learning Problems: Non-IID

- However, in FL, the data distributed across different devices or clients is not drawn from the same statistical distribution.
- Unlike the scenario distributed training, where the training data are randomly distributed. For FL, the data stored in each device is highly biased.
- This may lead to significant accuracy degradation for the global model.

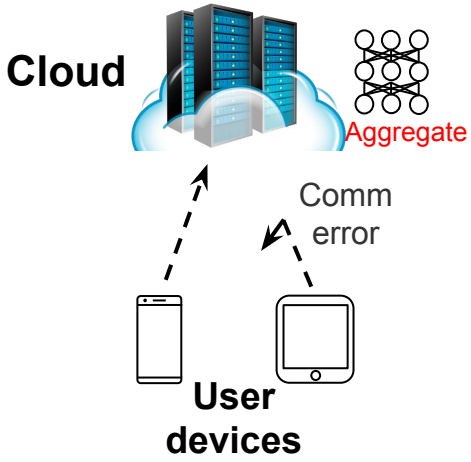


Federated Learning Problems: Heterogeneity



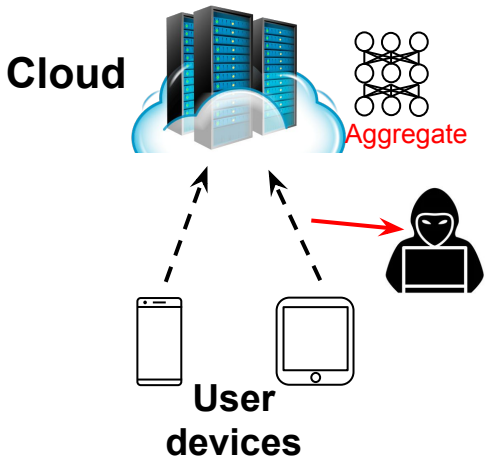
- Different edge device may have different processing speed.
- This will cause the total latency of each training round bottlenecked by the straggler, leading to a slow convergence of the training process.

Federated Learning Problems: Communication



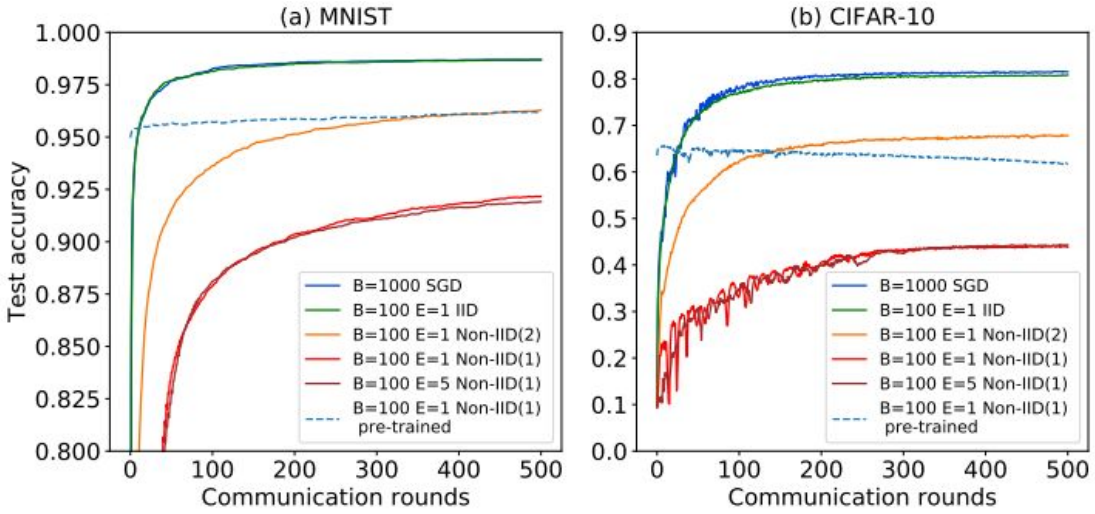
- The communication between edge devices and central cloud may incur transmission loss or error.
- This will impact the training latency and accuracy.

Federated Learning Problems: Privacy



- The attacker can leverage the transmitted gradient to reconstruct the original input training data.
- This will lead to privacy leakage.

Federated Learning with Non-iid Data

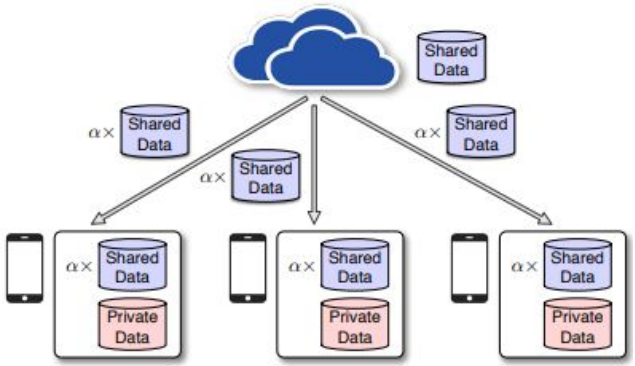


- The training sets are evenly partitioned into 10 clients.
- For IID setting, each client is randomly assigned a uniform distribution over 10 classes.
- For non-IID setting, the data is sorted by class and divided to create two extreme cases: (a) 1-class non-IID, where each client receives data partition from only a single class, and (b) 2-class non-IID, where the sorted data is divided into 20 partitions and each client is randomly assigned 2 partitions from 2 classes.

Zhao, Yue, et al. "Federated learning with non-iid data." *arXiv preprint arXiv:1806.00582* (2018).

Federated Learning with Non-iid Data

- We propose a data-sharing strategy to improve FedAvg with non-IID data by creating a small subset of data which is globally shared between all the edge devices.
- Experiments show that test accuracy can be increased by ~30% on CIFAR-10 dataset with only 5% globally shared data.



NYU SAI LAB

Zhao, Yue, et al. "Federated learning with non-iid data." *arXiv preprint arXiv:1806.00582* (2018).

FedProx

Algorithm 2 `FedProx` (Proposed Framework)

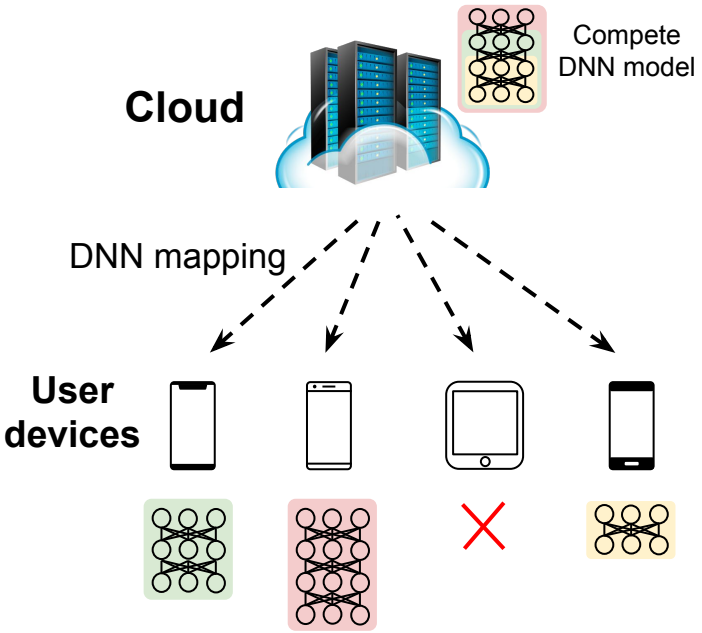
Input: $K, T, \mu, \gamma, w^0, N, p_k, k = 1, \cdots, N$
for $t = 0, \cdots, T-1$ **do**
  Server selects a subset S_t of K devices at random (each device k is chosen with probability p_k)
  Server sends w^t to all chosen devices
  Each chosen device $k \in S_t$ finds a w_k^{t+1} which is a γ_k^t-inexact minimizer of: $w_k^{t+1} \approx \arg\min_w \; h_k(w;\; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$
  Each device $k \in S_t$ sends w_k^{t+1} back to the server
  Server aggregates the w's as $w^{t+1} = \frac{1}{K}\sum_{k \in S_t} w_k^{t+1}$
end for

$$\min_w h_k(w;\; w^t) = F_k(w) + \frac{\mu}{2}\|w - w^t\|^2$$

- We add an extra term to minimize the l_2 distance between the initial weight w_t and the learned weight w.
- This loss ensures that the learnt w is not too different from the original w.

Li, Tian, et al. "Federated optimization in heterogeneous networks." *Proceedings of Machine learning and systems* 2 (2020): 429-450.

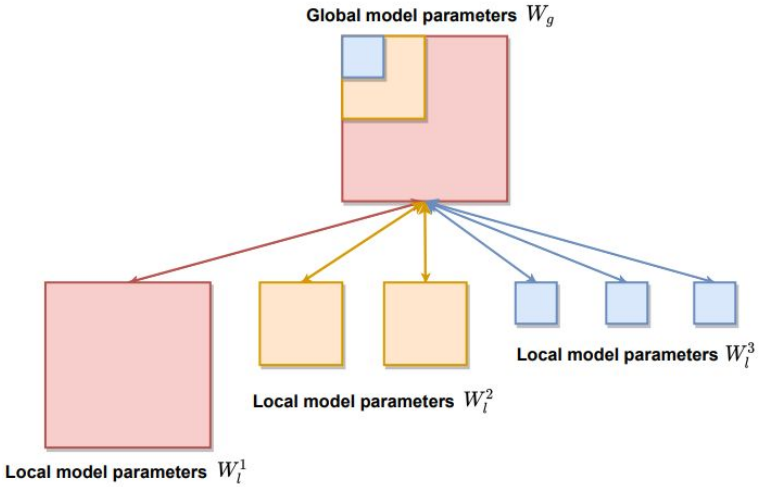
Federated Learning Problems: Heterogeneity



- End devices will have heterogeneous system configuration.
- HeteroFL partitions and assigns the DNN based on the processing power of each device.
- Each device only train a subset of the DNN model.

Diao, Enmao, Jie Ding, and Vahid Tarokh. "Heterofl: Computation and communication efficient federated learning for heterogeneous clients." *arXiv preprint arXiv:2010.01264* (2020).

HeteroFL



- Each edge device will be assigned with part of the neural network to perform local training based on its computational complexity.

NYU SAI LAB

Diao, Enmao, Jie Ding, and Vahid Tarokh. "Heterofl: Computation and communication efficient federated learning for heterogeneous clients." *arXiv preprint arXiv:2010.01264* (2020).

Federated Learning Problems: Communication

$$e(\mathbf{u}, \bar{\mathbf{u}}) = \frac{1}{N} \sum_{j=1}^{N} I(\mathrm{sgn}(u_j) = \mathrm{sgn}(\bar{u}_j))$$

- u_j denotes the sign of the model weight after local updates.
- Our solution dynamically identifies relevant local updates and excludes those irrelevant from being.
- Only the local device with high relevance will transmit their weight to the central server.

Luping, W.; Wei, W.; and Bo, L. 2019. Cmfl: Mitigating communication overhead for federated learning. In 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 954–964. IEEE.

FedMARL

Final model Accuray

Total Training Latency

Total Bandwidth

$$\max_{A} E\left[w_1 Acc(T) - w_2 \sum_{t \in T} H_t - w_3 \sum_{t \in T} B_t\right]$$

$$A = [a_n^t]$$

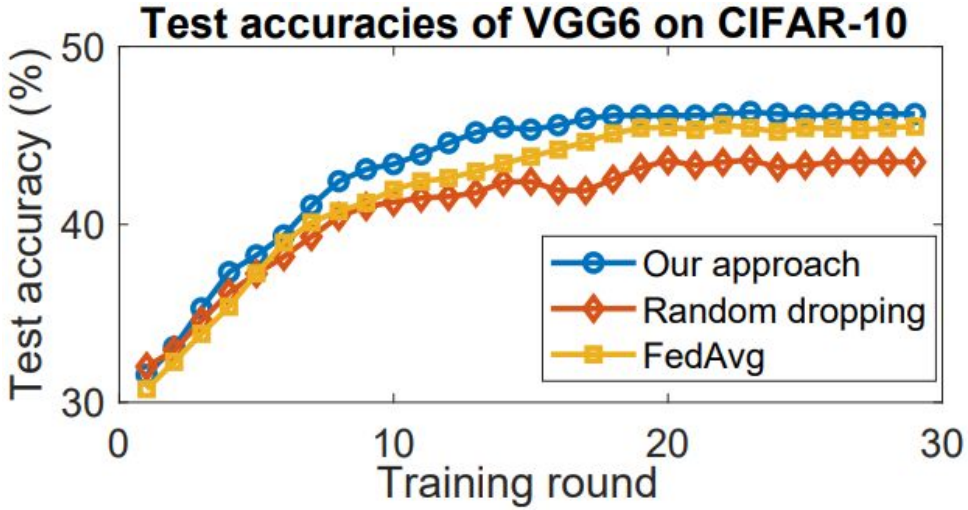
Client Selection

- Our objective is to maximize the accuracy of the global model while minimizing the total processing latency and communication cost.
- w1,w2,w3 are the importance of the objectives controlled by the FL application designers.
- The FL optimization problem is difficult to solve directly. We instead model the problem as a MARL problem.

NYU SAI LAB

Zhang, Sai Qian, Jieyu Lin, and Qi Zhang. "A multi-agent reinforcement learning approach for efficient client selection in federated learning." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 36. No. 8. 2022.

FedMARL



- Every random dropping is better than FedAvg.
- FedMarl is much better than random dropping and FedAvg.

NYU SAI LAB

Topics

- Federated Learning (Continue)
- Distributed DNN Training
- Distributed DNN Inference
- Speculative Decoding

Distributed DNN Training: Data Parallelism

- To train DNN in a distributed fashion, we need to batchify the training datasets.
- Assume a batch size of $b \in B$, x denotes a batch of training dataset.
- Let η represent the learning rate. w_t represents the weight at t.
- The distributed training process will be similar to the federated learning. Excepted that the data will be distributed in an independent and identically distributed (IID) fashion.

$$L(w) = \frac{1}{|X|} \sum_{x \in X} l(x, w)$$

Loss function

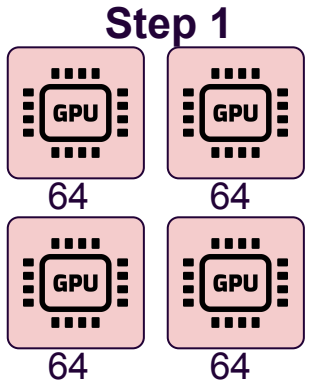
$$w_{t+1} = w_t - \eta \frac{1}{n} \sum_{x \in \mathcal{B}} \nabla l(x, w_t)$$

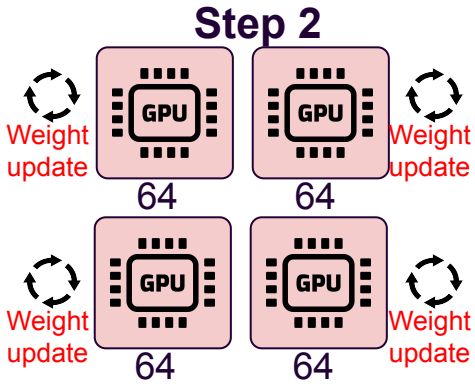
Weight update

NYU SAI LAB

Parameter Server

- A parameter server is a distributed system used to manage and synchronize the parameters (weights) of a machine learning model during training, especially in large-scale and distributed training scenarios.

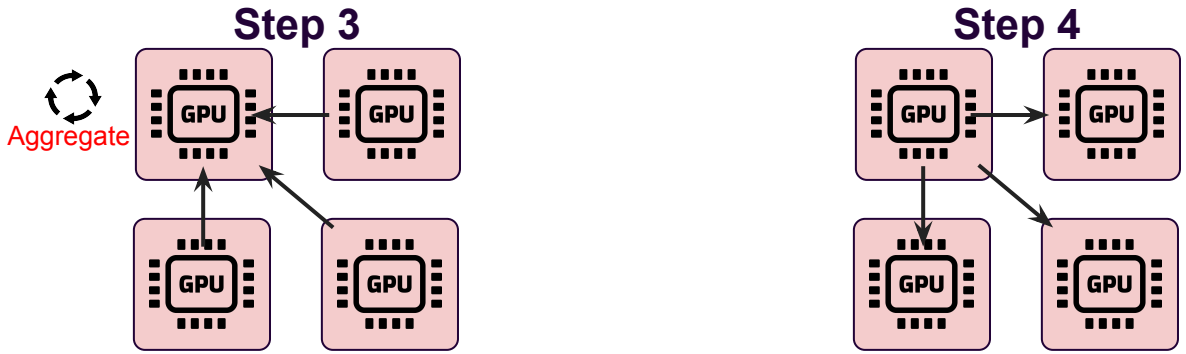




A total batch size of 256

Li, Mu, et al. "Scaling distributed machine learning with the parameter server." *11th USENIX Symposium on operating systems design and implementation (OSDI 14)*. 2014.

Parameter Server

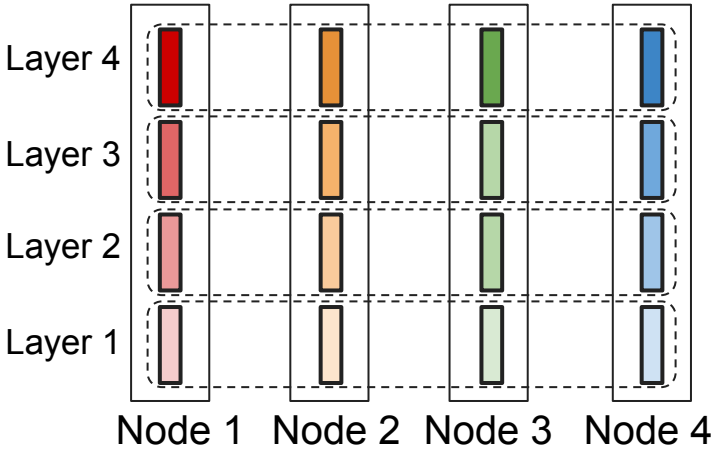


- Total amount of communication: 2(N-1)G.
- N is the number of nodes, G is the size of the weight gradient.
- If a worker node fails, other nodes can continue training without significant disruption. But PS scheme is not scalable, the central node can not handle all the servers, as the number of nodes increases.

All Reduce

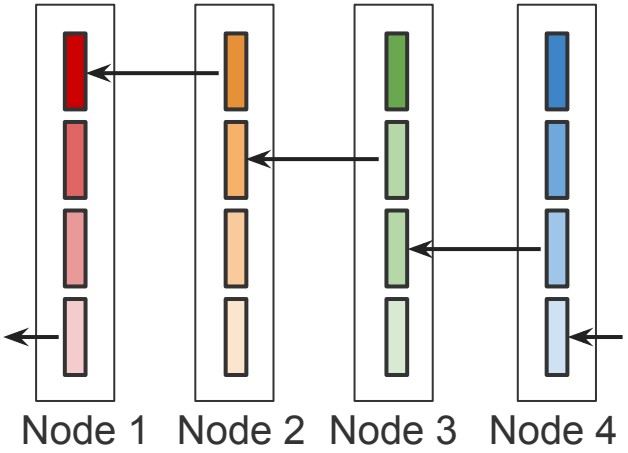
- All-reduce is a communication operation widely used in distributed deep neural network (DNN) training to synchronize and aggregate data across multiple computing nodes or devices.
- Detailed training steps:
  - **Forward Pass:** Each node (e.g., GPU) computes the forward pass of the neural network independently using its local mini-batch of data.
  - **Backward Pass:** Each node computes the gradients of the loss with respect to the model parameters.
  - **All-Reduce Step:** The gradients from all nodes are summed together using the all-reduce operation. This summed gradient is then broadcast to all nodes.
  - **Parameter Update:** Each node updates its local copy of the model parameters using the aggregated gradients.

Ring All-Reduce



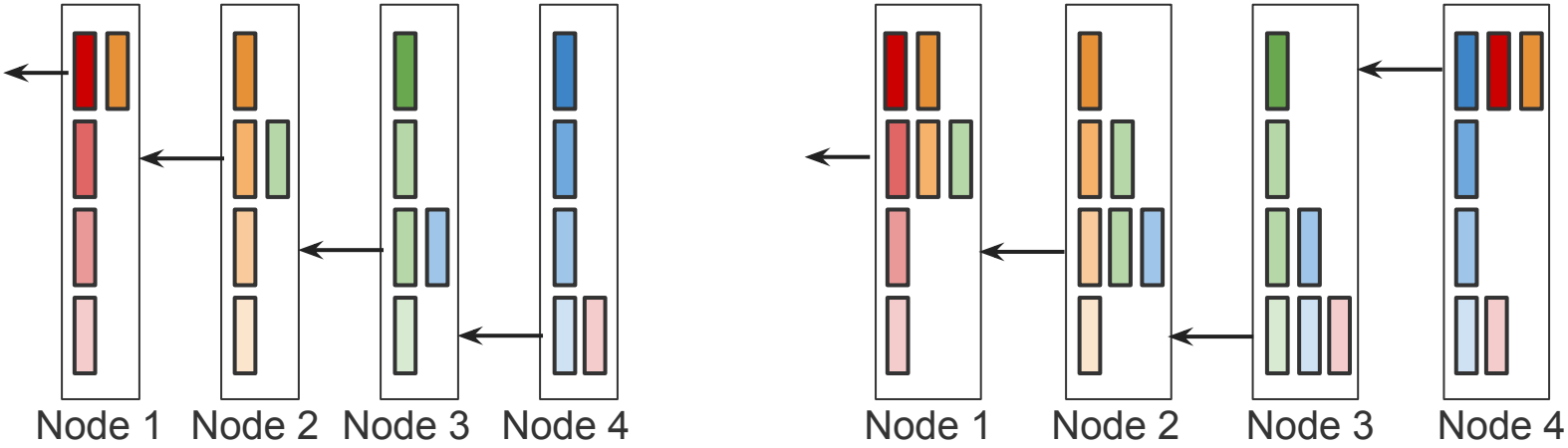
- Assume a neural network with four layers.
- Each node has been assigned with an equivalent amount of training dataset.

Ring All-Reduce

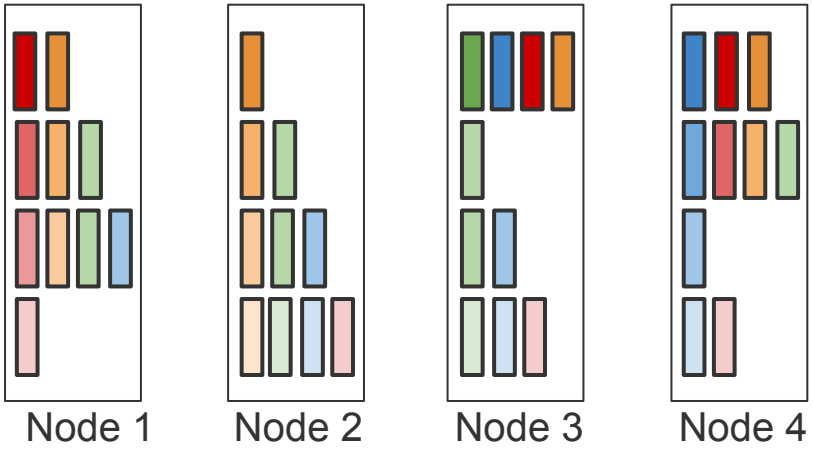


- Nodes are arranged in a ring topology, and each node passes a portion of its data to its neighbor in a circular fashion. This continues until all nodes have the complete reduced data.
- Each node has identical amounts of workload.

Ring All-Reduce

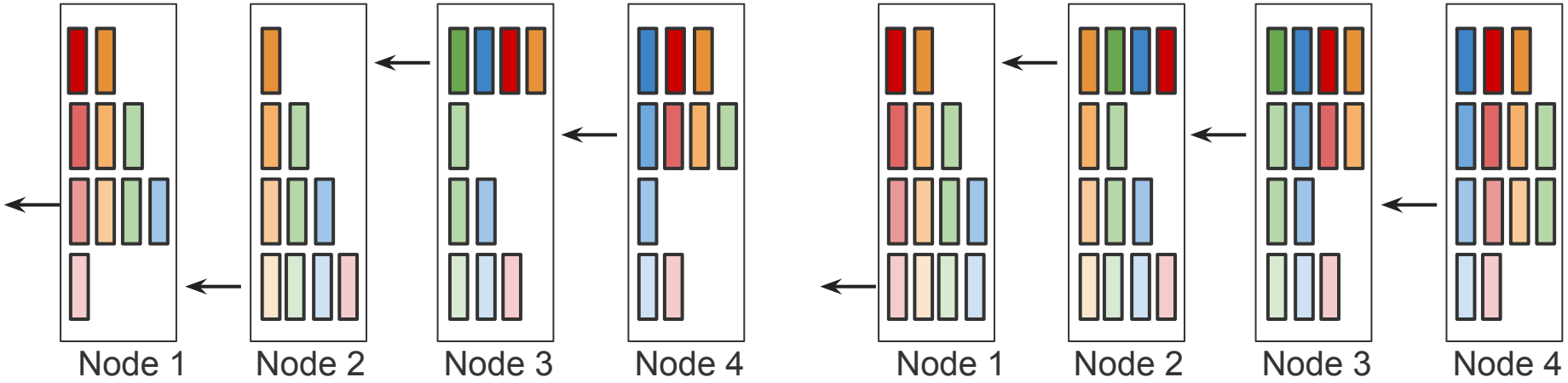


Ring All-Reduce



- The end of share-reduce phase.

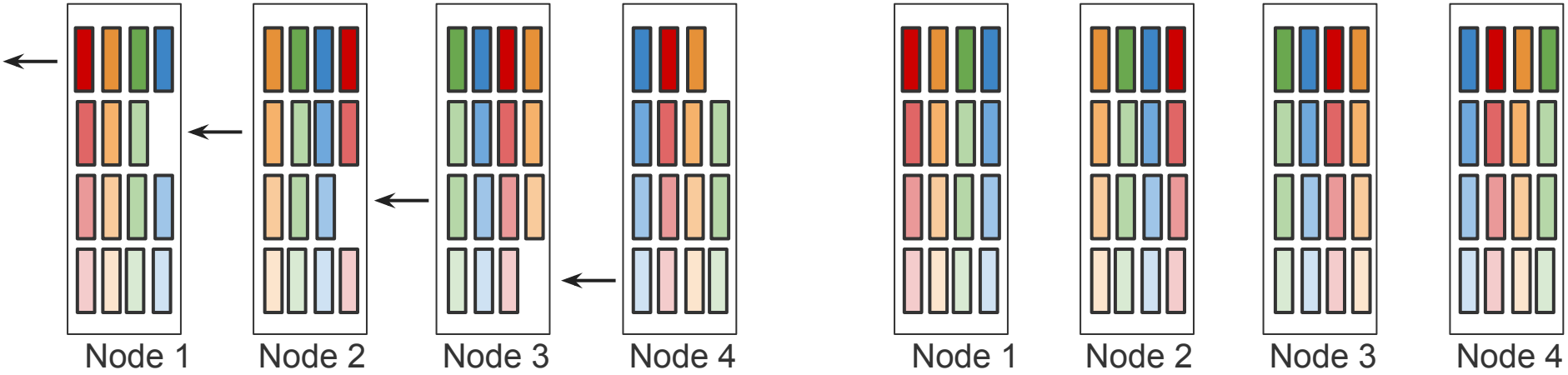
Ring All-Reduce



The end of reduce-only phase

NYU SAI LAB

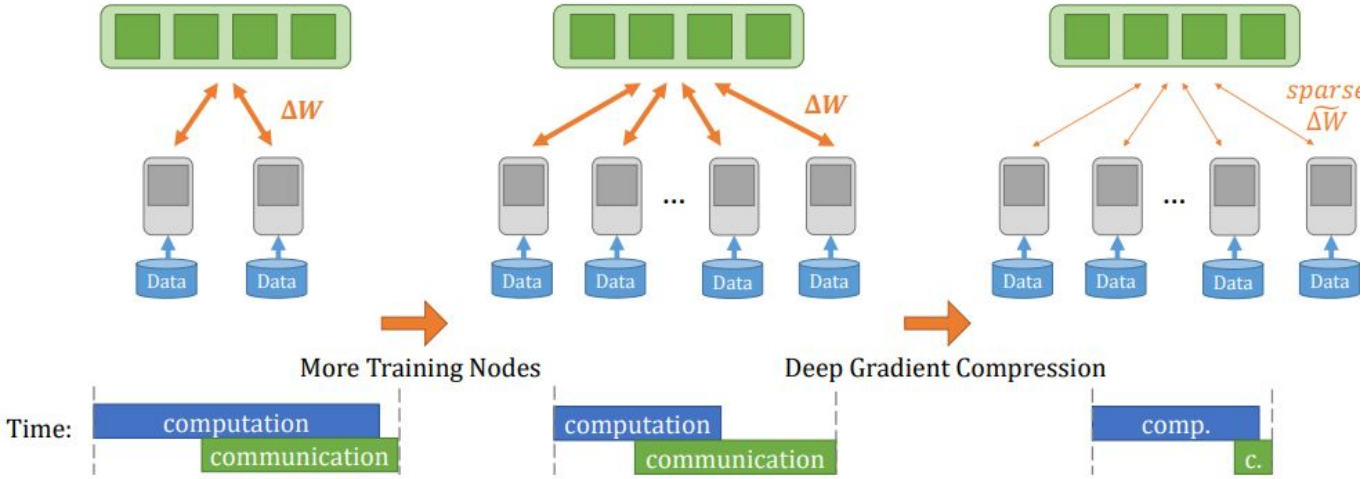
Ring All-Reduce



- Total amount of communication: 2(N-1)G.
- N is the number of nodes, G is the size of the weight gradient.

NYU SAI LAB

Communication Reduction for Distributed Training



- We reduce the communication bandwidth by sending only the important gradients (magnitude > thres).
- The accumulated weight gradient of each layer is transmitted only when its value is larger than a threshold.

Lin, Yujun, et al. "Deep gradient compression: Reducing the communication bandwidth for distributed training." *arXiv preprint arXiv:1712.01887* (2017).

Communication Reduction for Distributed Training

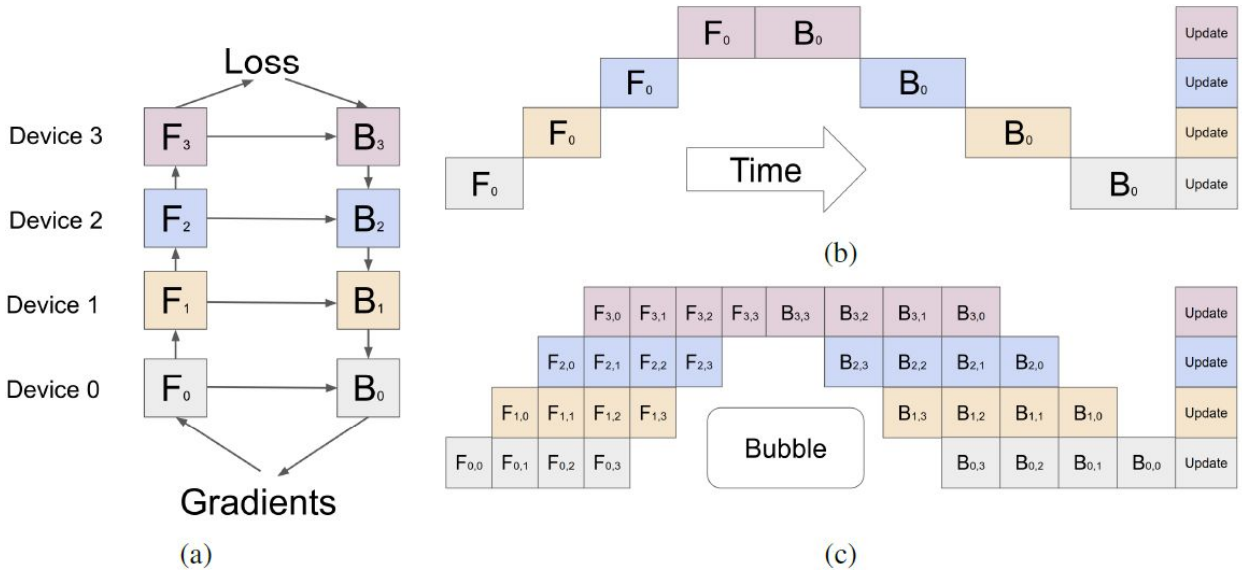
- The gradient is collected locally, only gradient with high magnitude are sent to the central server for model updating.
- Run-length encoding is utilized to compress the sparse gradient.

Algorithm 1 Gradient Sparsification on node k

Input: dataset χ
Input: minibatch size b per node
Input: the number of nodes N
Input: optimization function SGD
Input: init parameters $w = \{w[0], w[1], \cdots, w[M]\}$

$G^k \leftarrow 0$
for $t = 0, 1, \cdots$ **do**
  $G_t^k \leftarrow G_{t-1}^k$
  for $i = 1, \cdots, b$ **do**
    Sample data x from χ
    $G_t^k \leftarrow G_t^k + \frac{1}{Nb} \nabla f(x; w_t)$
  end for
  for $j = 0, \cdots, M$ **do**
    Select threshold: $thr \leftarrow s\%$ of $\left|G_t^k[j]\right|$
    $Mask \leftarrow \left|G_t^k[j]\right| > thr$
    $\widetilde{G}_t^k[j] \leftarrow G_t^k[j] \odot Mask$
    $G_t^k[j] \leftarrow G_t^k[j] \odot \neg Mask$
  end for
  All-reduce $G_t^k : G_t \leftarrow \sum_{k=1}^{N} encode(\widetilde{G}_t^k)$
  $w_{t+1} \leftarrow SGD(w_t, G_t)$
end for

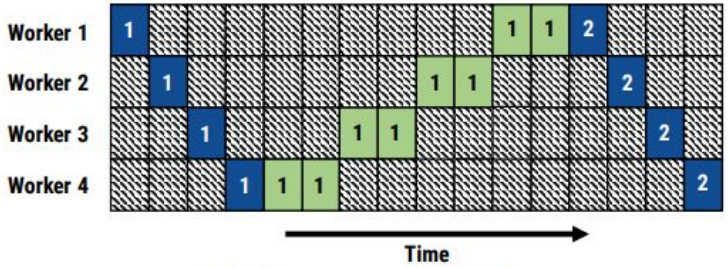
Distributed DNN Training: Model Parallelism

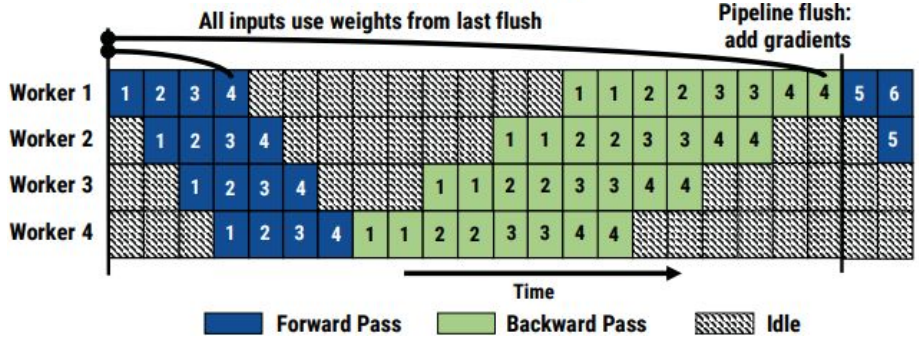


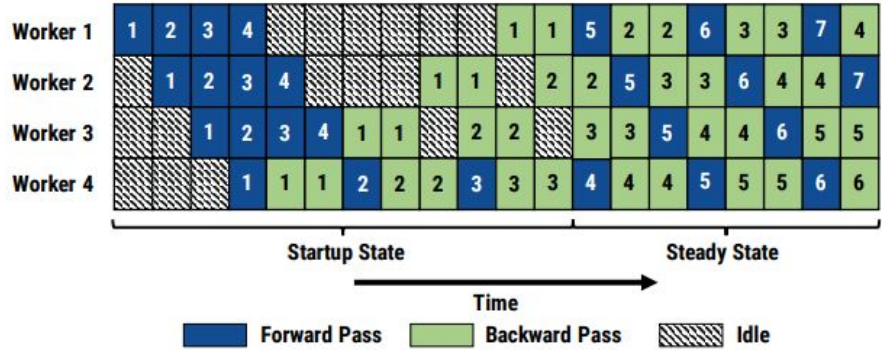
- The naive model parallelism strategy leads to severe under-utilization due to the sequential dependency of the network.
- GPipe first divides every mini-batch of size N into M equal micro-batches, enabling different accelerators to work on different micro-batches simultaneously.

Huang, Yanping, et al. "Gpipe: Efficient training of giant neural networks using pipeline parallelism." *Advances in neural information processing systems* 32 (2019).

Model Parallelism: PipeDream







- In this paper, we propose PipeDream, a system that uses Pipeline to enable faster DNN training by combining intra-batch parallelism with inter-batch parallelization.

Narayanan, Deepak, et al. "PipeDream: Generalized pipeline parallelism for DNN training." *Proceedings of the 27th ACM symposium on operating systems principles*. 2019.

Topics

- Federated Learning (Continue)
- Distributed DNN Training
- Distributed DNN Inference

BranchyNet

- Data samples are not equal in their recognition difficulties.
- For the easy samples, they only needs to be processed with a few layers before generating the correct results.

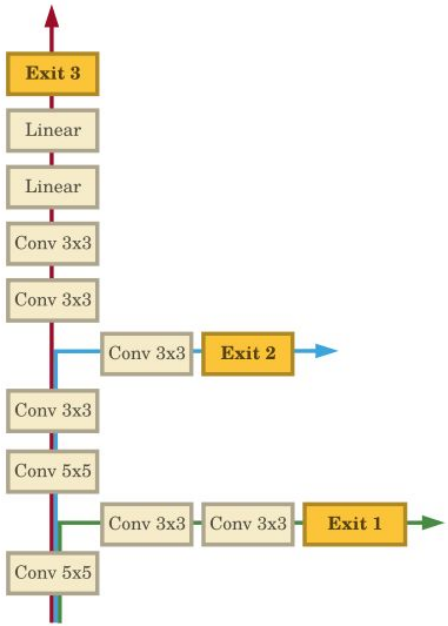
NYU SAI LAB

Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks." *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016.

BranchyNet

- During Inference, a confidence score is computed at each exit point, if greater than a predefined threshold, then the output is computed locally, leading to a faster inference.

- The confidence score is defined as: $\text{entropy}(\boldsymbol{y}) = \sum_{c \in \mathcal{C}} y_c \log y_c,$



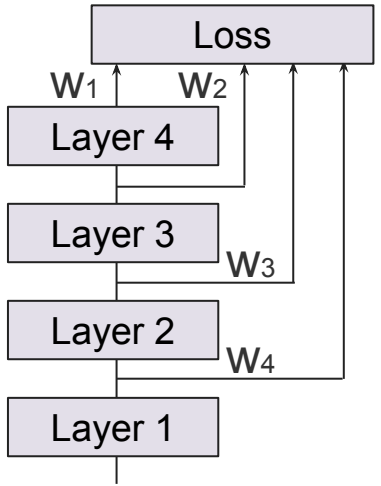
NYU SAI LAB

Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks." *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016.

BranchyNet

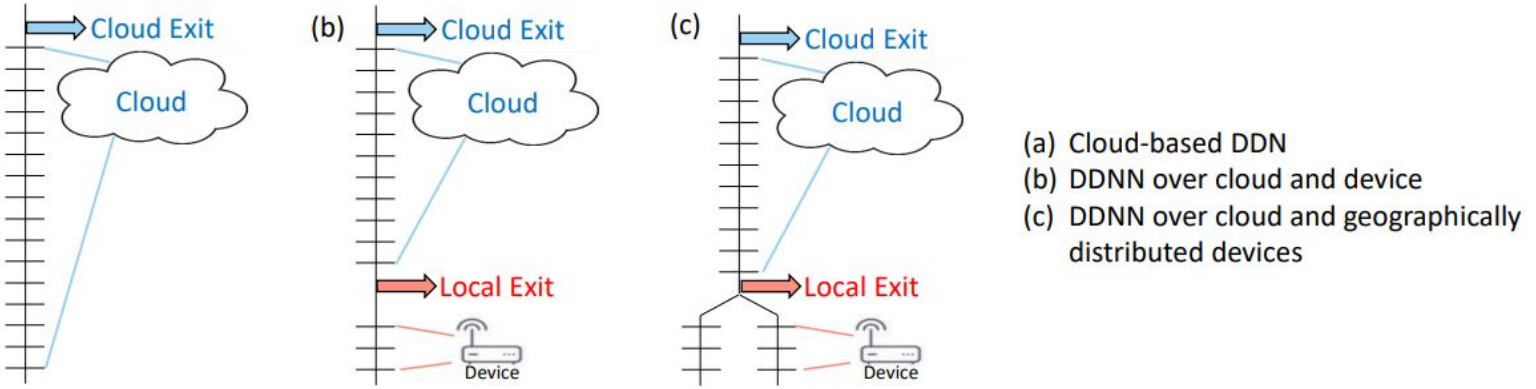
- To train the Branchy-style DNN, we can sum the cross-entropy loss at each local exit points, and train them jointly.

$$L_{\text{branchynet}}(\hat{\boldsymbol{y}}, \boldsymbol{y}; \theta) = \sum_{n=1}^{N} w_n L(\hat{\boldsymbol{y}}_{\text{exit}_n}, \boldsymbol{y}; \theta)$$



Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks." *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016.

Distributed Deep Neural Networks over the Cloud, the Edge and End Devices

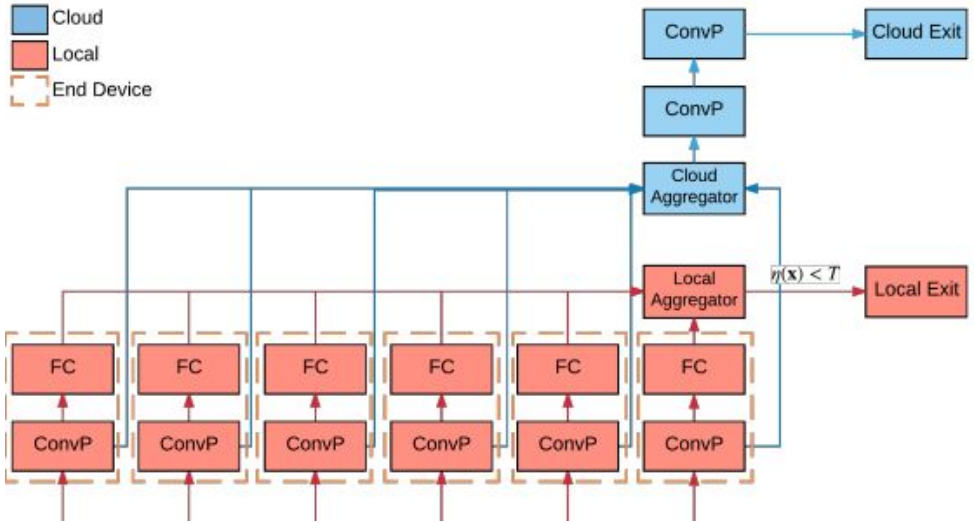


- We propose distributed deep neural networks (DDNNs) over distributed computing hierarchies, consisting of the cloud, the edge (fog) and end devices.

NYU SAI LAB

Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Distributed deep neural networks over the cloud, the edge and end devices." *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017.
Kang, Yiping, et al. "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge." *ACM SIGARCH Computer Architecture News* 45.1 (2017): 615-629.

DDNN

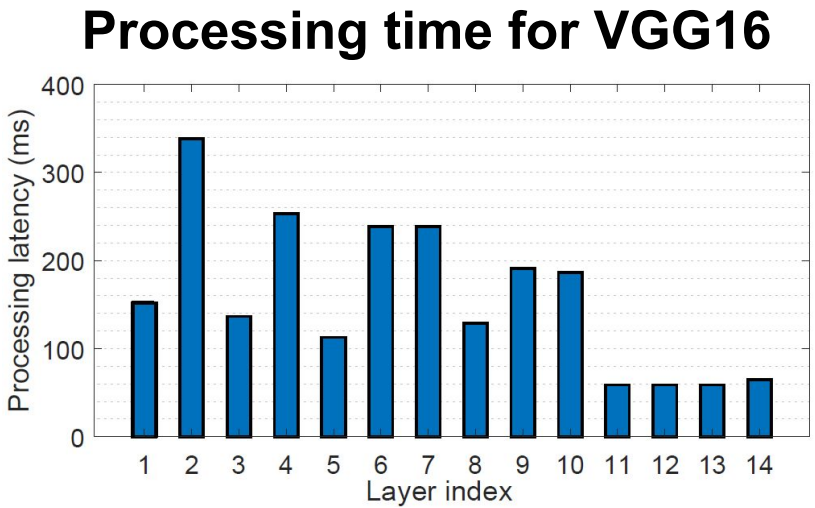


- Each edge device is implemented with a local DNN for local inference.
- The results from each local DNN is first aggregated locally.
- If the local exit is not confident, the activation output after the last convolutional layer from each end device is sent to the cloud aggregator for further processing.

NYU SAI LAB

Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Distributed deep neural networks over the cloud, the edge and end devices." *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*. IEEE, 2017.
Kang, Yiping, et al. "Neurosurgeon: Collaborative intelligence between the cloud and mobile edge." *ACM SIGARCH Computer Architecture News* 45.1 (2017): 615-629.

ADCNN

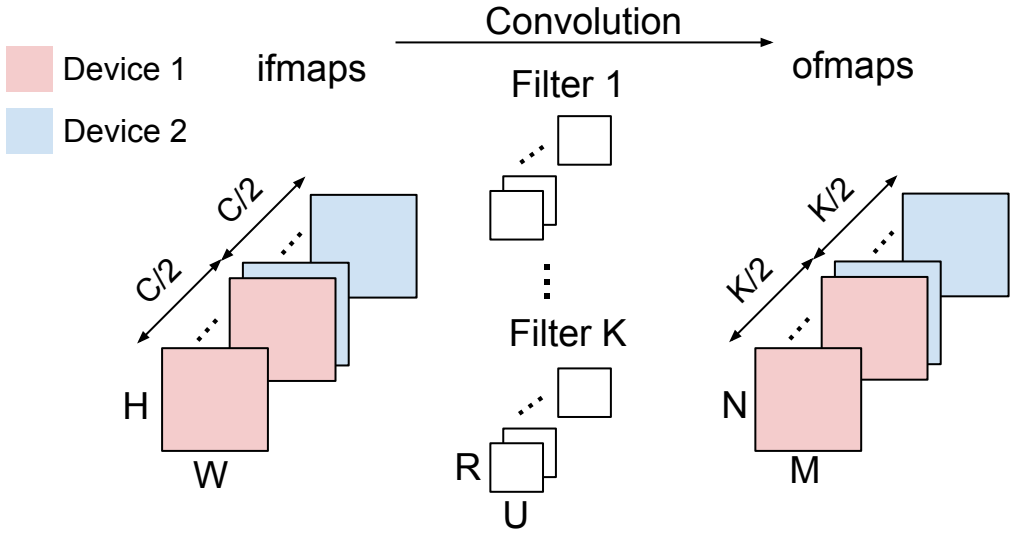


- Earlier layers take much longer to process than the later layers.

NYU SAI LAB

Zhang, Sai Qian, Jieyu Lin, and Qi Zhang. "Adaptive distributed convolutional neural network inference at the network edge with ADCNN." *Proceedings of the 49th International Conference on Parallel Processing*. 2020.

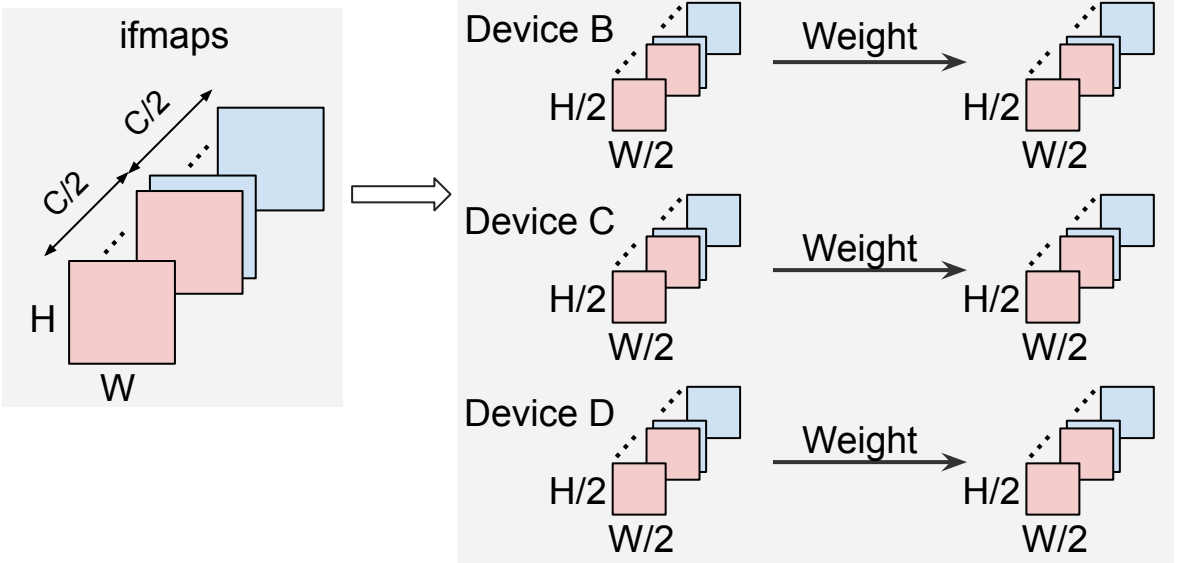
ADCNN



- In channelwise partition, each node needs to exchange their partially accumulated output feature maps to produce final output feature maps, which leads to a significant communication overhead.

Zhang, Sai Qian, Jieyu Lin, and Qi Zhang. "Adaptive distributed convolutional neural network inference at the network edge with ADCNN." *Proceedings of the 49th International Conference on Parallel Processing*. 2020.

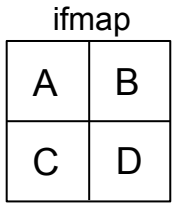
ADCNN



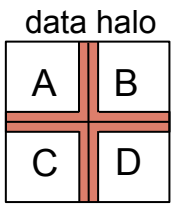
- The input will partitioned in spatial dimension and distribute over multiple devices.
- The weight will duplicate and save on each device.

Zhang, Sai Qian, Jieyu Lin, and Qi Zhang. "Adaptive distributed convolutional neural network inference at the network edge with ADCNN." *Proceedings of the 49th International Conference on Parallel Processing*. 2020.

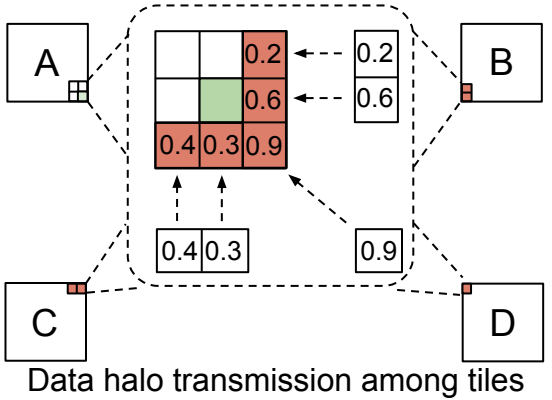
ADCNN



(a)



(b)

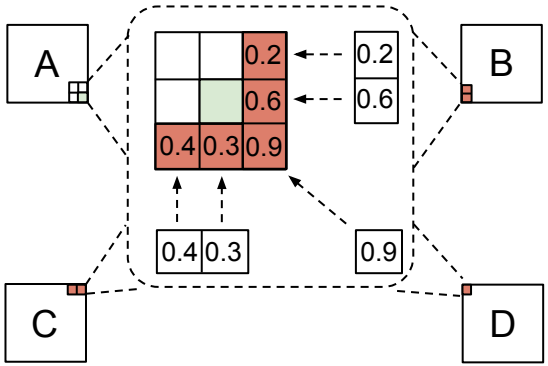


Data halo transmission among tiles

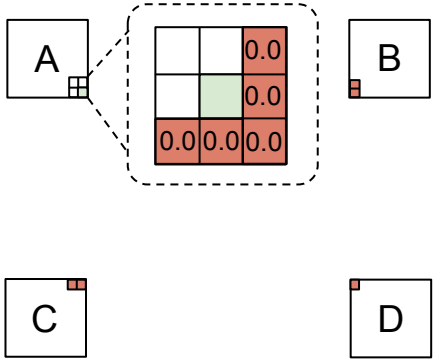
(c)

- In spatial partition, each tile needs to transmit their data halo in order to compute the correct result.

ADCNN



Normal Spatial Partition



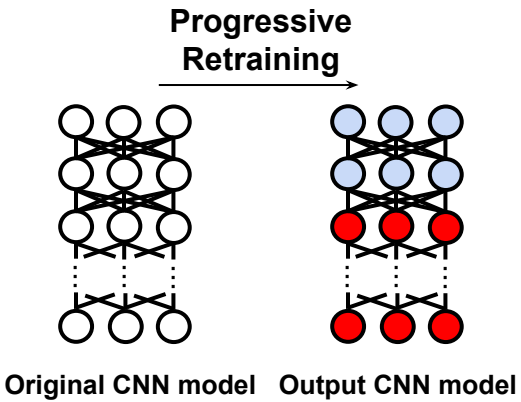
Fully Decomposable Spatial Partition (FDSP)

- The cross-tile information transfer can be eliminated by padding the edge pixels with zeros.

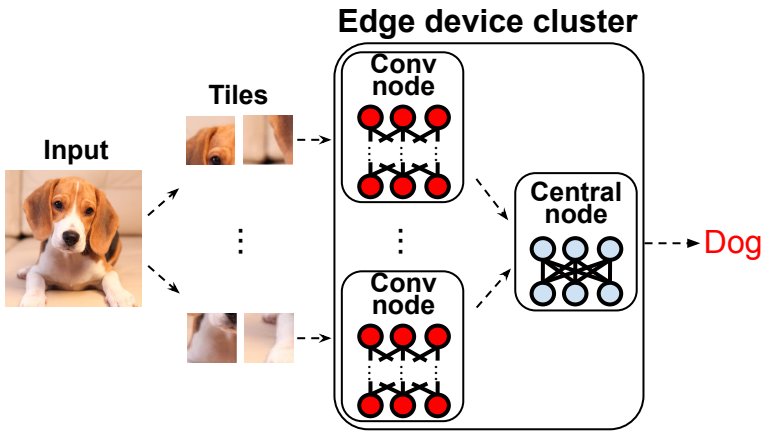
NYU SAI LAB

ADCNN

Step 1

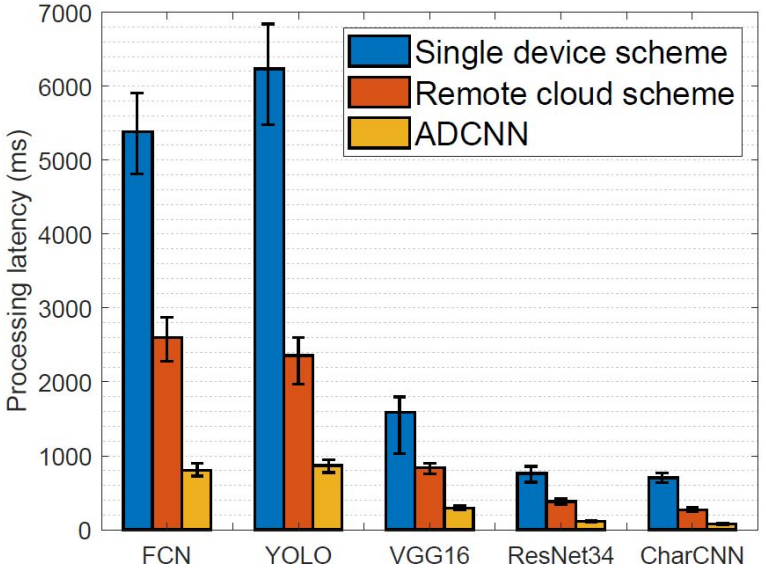


Step 2



Evaluation Results

- We implement ADCNN system with nine identical Raspberry Pi devices which simulate the edge devices. Among these nine devices, eight are used as Conv nodes, and the rest one is used as the Central node.
- Baselines:
  - Single device scheme
  - Remote cloud scheme
- ADCNN decreases the average processing latency by 6.68x and 4.42x, respectively.



NYU SAI LAB

Presentation

Hangyi, Jiawei, Yue
Smoothquant: Accurate and efficient post-training quantization for large language models
Amogh
LoRA: Low-rank adaptation of large language models
Ruizhe, Yunchi, Tianqi
QuaRot: Outlier-Free 4-Bit Inference in Rotated LLMs