NYU

Lecture 08:
Machine Learning System for Training and Inference

Some Notes

- Lab1 grade will post tomorrow.
- Lab3 will post this Tomorrow.
- Midterm coverage is released next week.
- Project proposal due next week.
  - Discussion board has built on Brightspace.

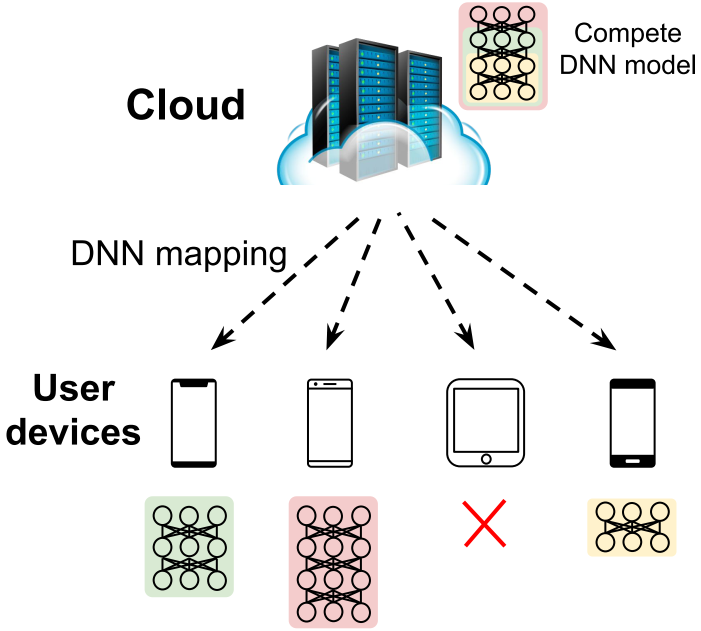
Recap

- Efficient training of DNNs
  - Efficient computing
  - Efficient storage
- Parameter efficient finetuning
- Federated Learning

Topics

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

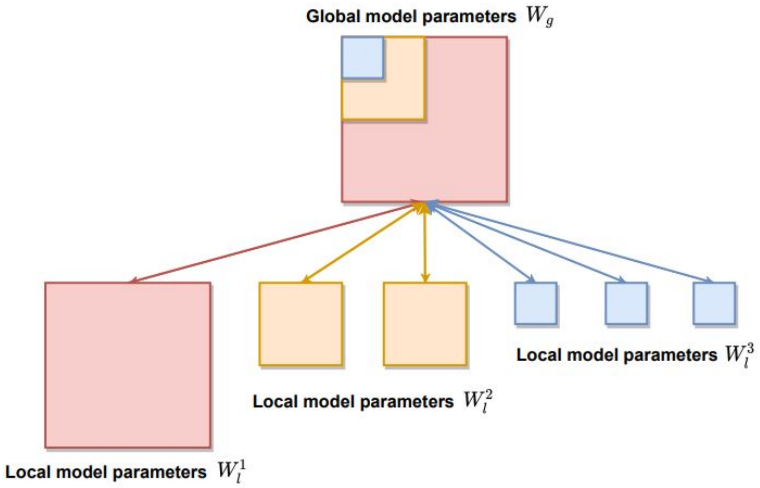
Federated Learning Problems: Heterogeneity



- 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.

NYU SAI LAB

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.

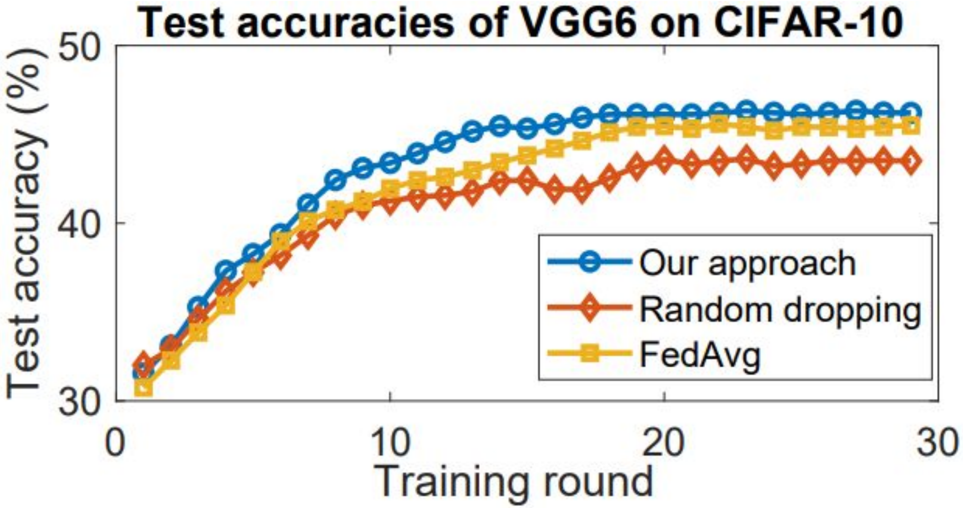
FedMARL

$$\max_{A} E\Big[w_1 \overset{\text{Final model Accuray}}{\boxed{Acc(T)}} - w_2 \overset{\text{Total Training Latency}}{\boxed{\sum_{t \in T} H_t}} - w_3 \overset{\text{Total Bandwidth}}{\boxed{\sum_{t \in T} B_t}}\Big]$$

$$A = [a_n^t] \quad \text{Client Selection}$$

- 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.

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∈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

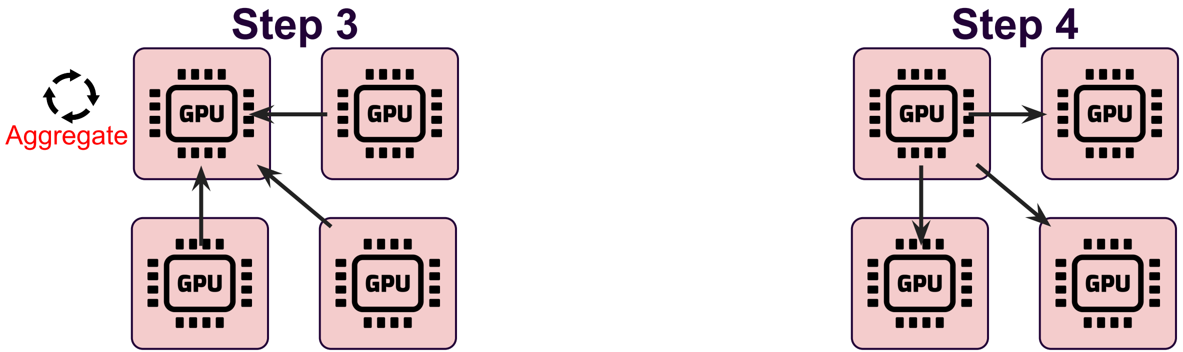
Parameter Server

- 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

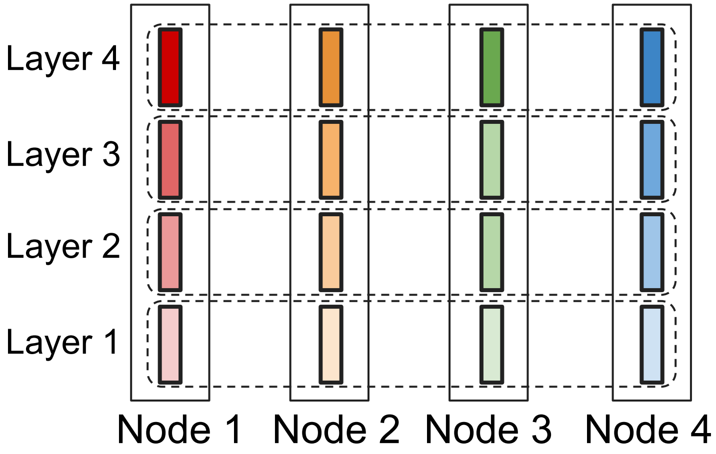


- Total amount of communication: 2(N-1)G.
- N is the number of nodes, G is the size of the weight gradient.
- If a worker node fails, other nodes can continue training without significant disruption. But PS scheme is not scalable, the central node can not handle all the servers, as the number of nodes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

All Reduce

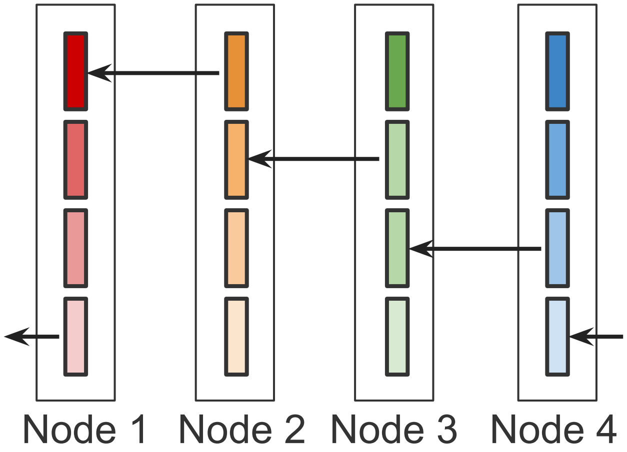
- All-reduce is a communication operation widely used in distributed deep neural network (DNN) training to synchronize and aggregate data across multiple computing nodes or devices.
- Detailed training steps:
  - **Forward Pass:** Each node (e.g., GPU) computes the forward pass of the neural network 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.

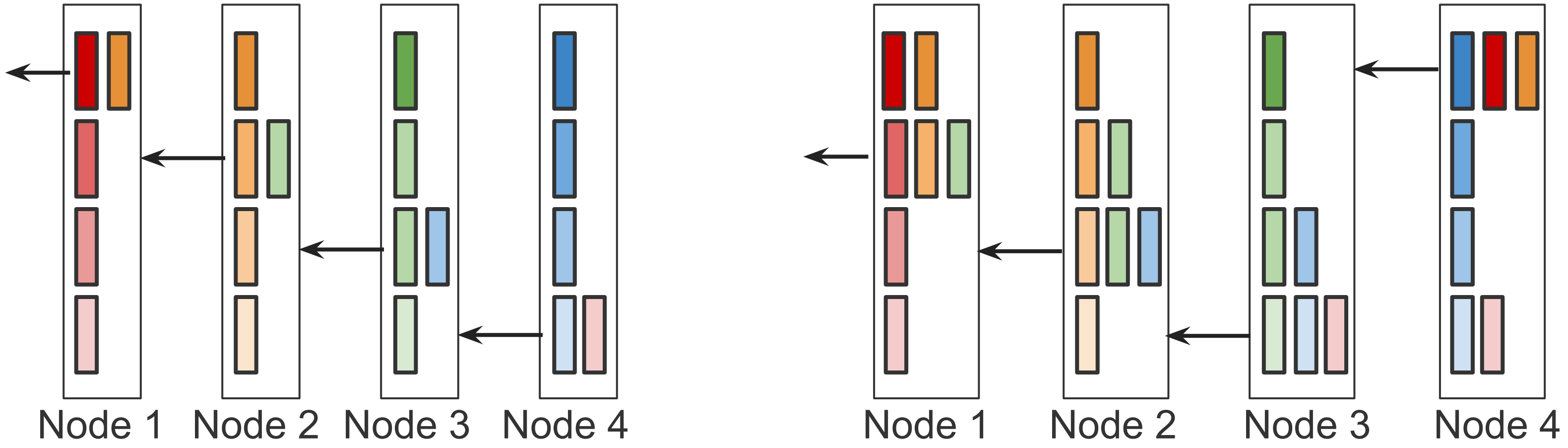
Ring All-Reduce



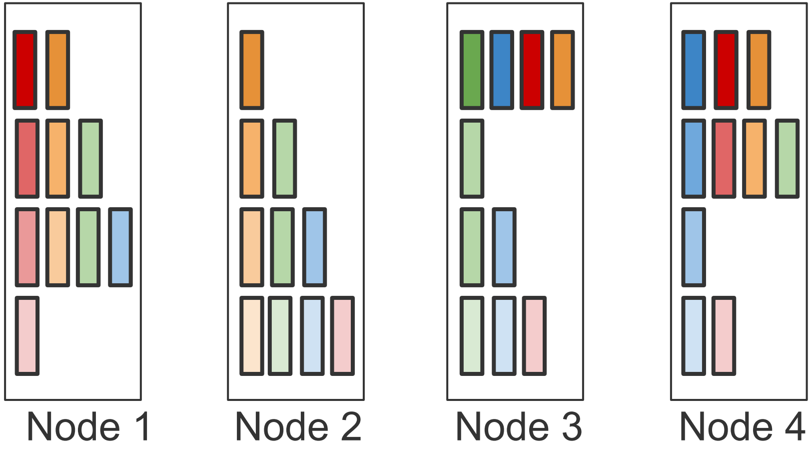
- Nodes are arranged in a ring topology, and each node passes a portion of its data to its neighbor in a circular fashion. This continues until all nodes have the complete reduced data.
- Each node has identical amounts of workload.

NYU SAI LAB

Ring All-Reduce

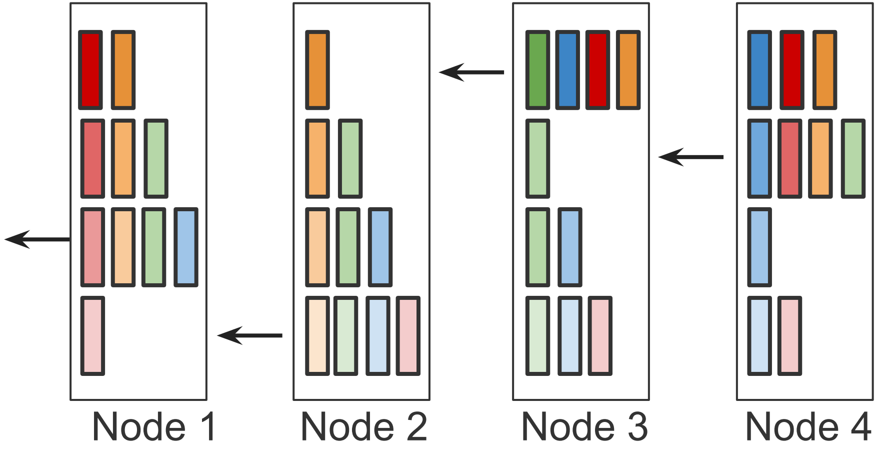


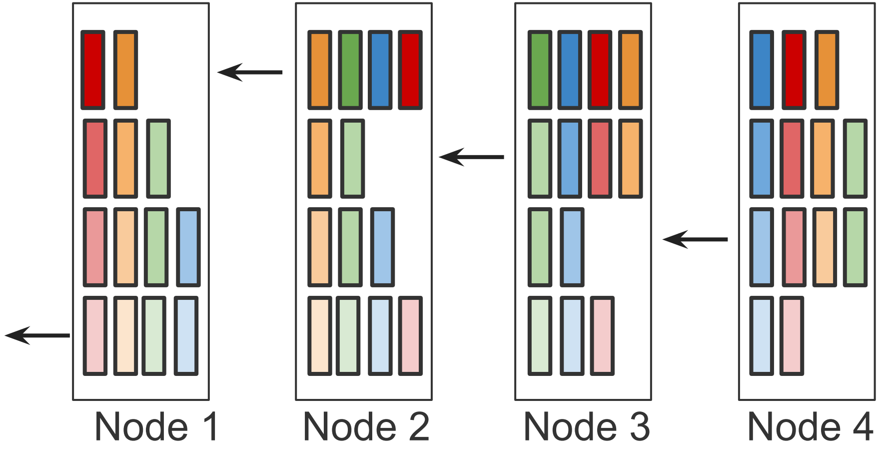
Ring All-Reduce



- The end of share-reduce phase.

Ring All-Reduce

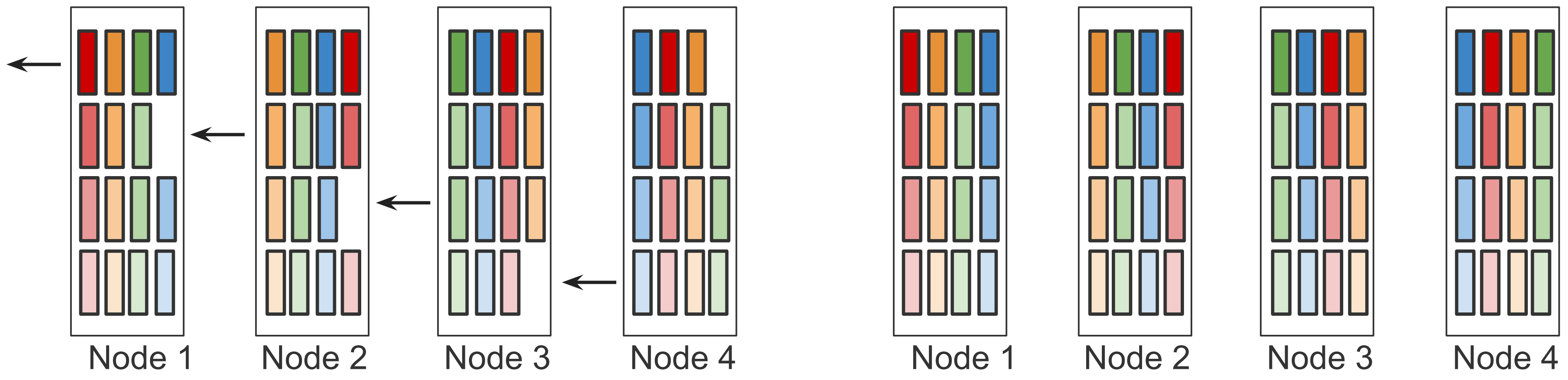




The end of reduce-only phase

NYU SAI LAB

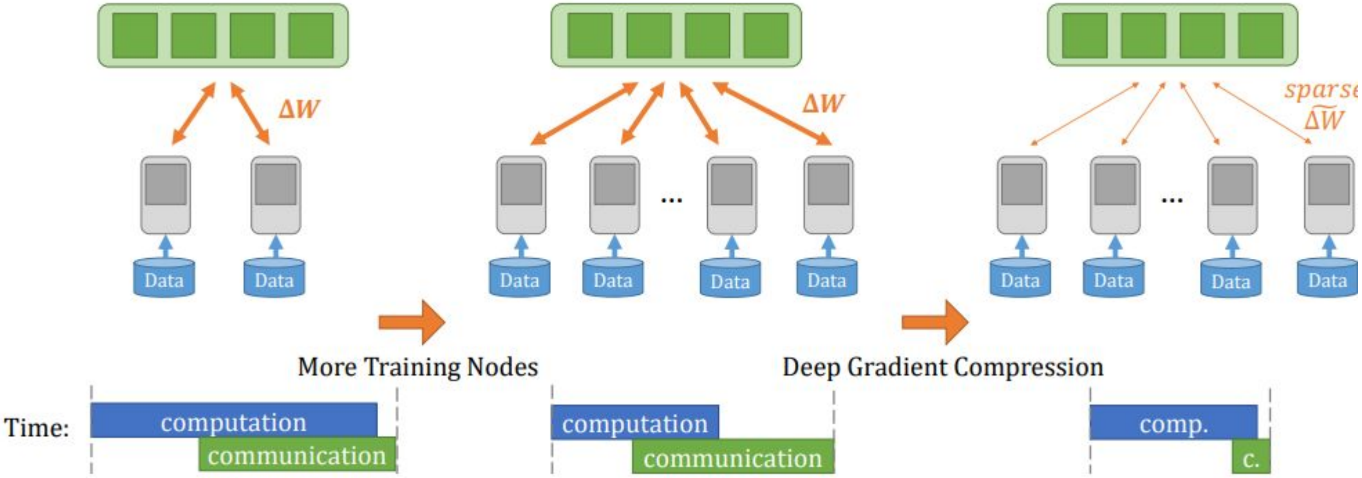
Ring All-Reduce



- 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.

NYU SAI LAB

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

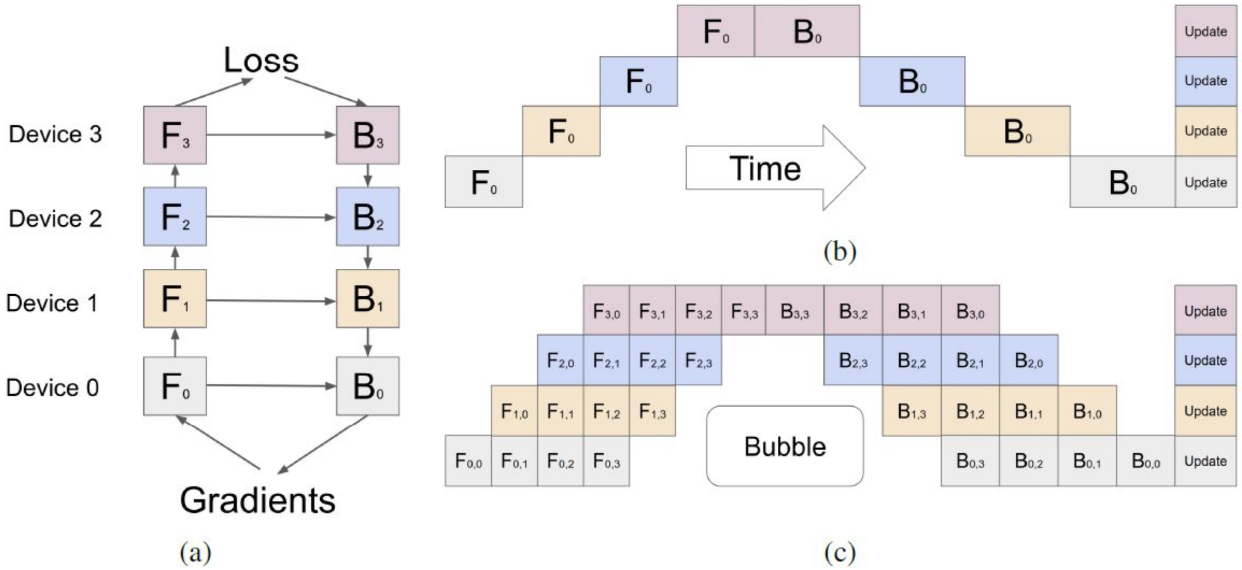
- The gradient is collected locally, only gradient with high magnitude are sent to the central server for model updating.
- Run-length encoding is utilized to compress the sparse gradient.

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

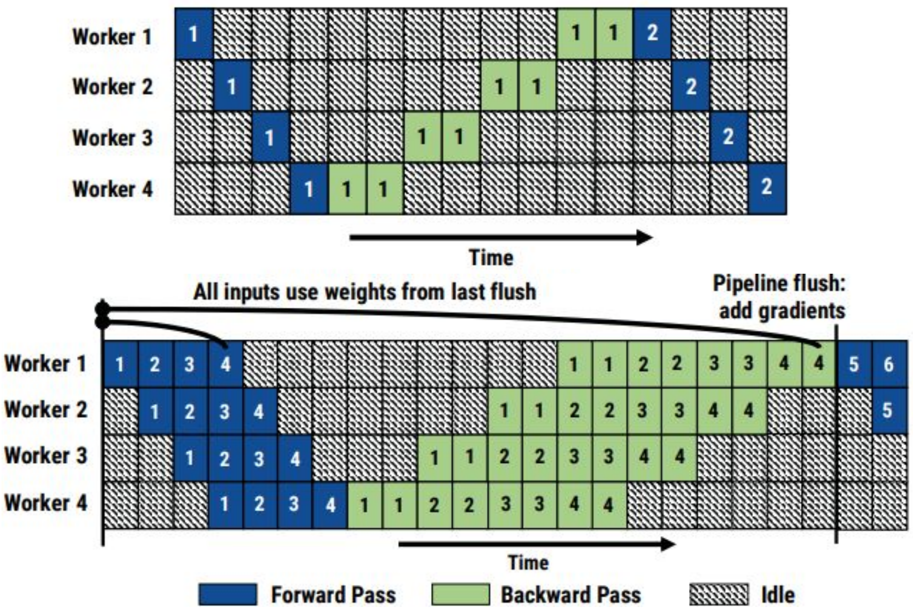
Distributed DNN Training: Model Parallelism

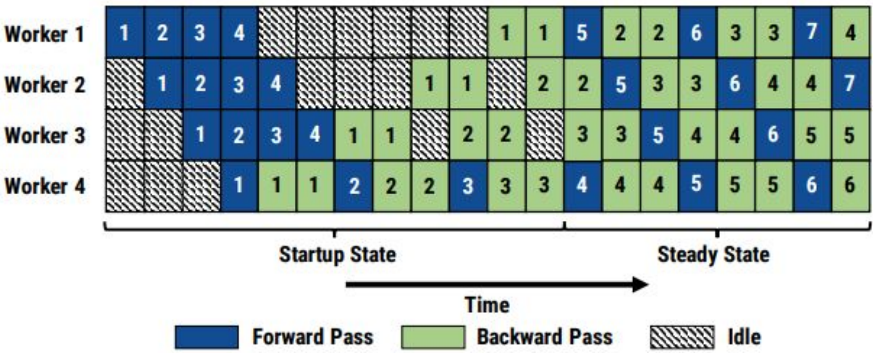


- The naive model parallelism strategy leads to severe under-utilization due to the sequential dependency of the network.
- GPipe first divides every mini-batch of size N into M equal micro-batches, enabling different accelerators to work on different micro-batches simultaneously.

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
- Speculative Decoding

NYU SAI LAB

BranchyNet

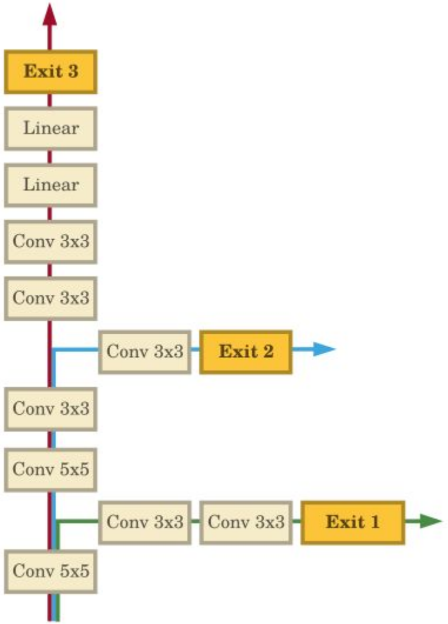
- Data samples are not equal in their recognition difficulties.
- For the easy samples, they only needs to be processed with a few layers before generating the correct results.

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: $\text{entropy}(\boldsymbol{y}) = \sum_{c \in \mathcal{C}} y_c \log y_c,$

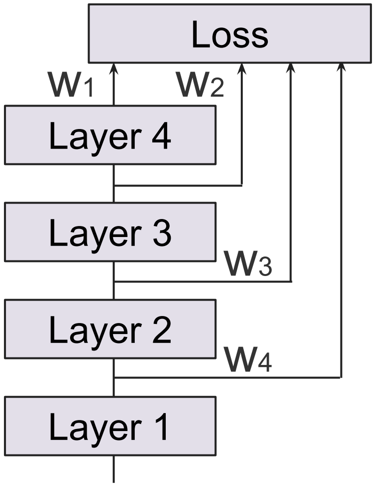


Teerapittayanon, Surat, Bradley McDanel, and Hsiang-Tsung Kung. "Branchynet: Fast inference via early exiting from deep neural networks." *2016 23rd international conference on pattern recognition (ICPR)*. IEEE, 2016.

BranchyNet

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

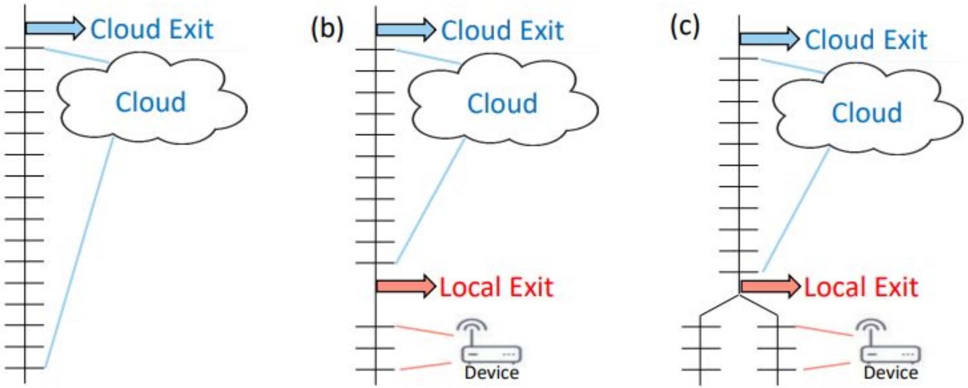
$$L_{\text{branchynet}}(\hat{\boldsymbol{y}}, \boldsymbol{y}; \theta) = \sum_{n=1}^{N} w_n L(\hat{\boldsymbol{y}}_{\text{exit}_n}, \boldsymbol{y}; \theta)$$



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.

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

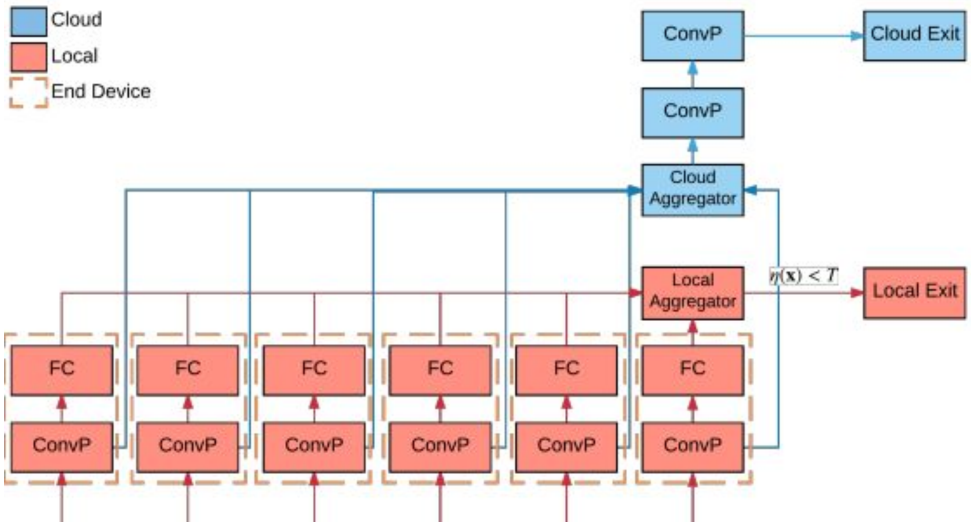


(a) Cloud-based DDN
(b) DDNN over cloud and device
(c) DDNN over cloud and geographically distributed devices

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

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



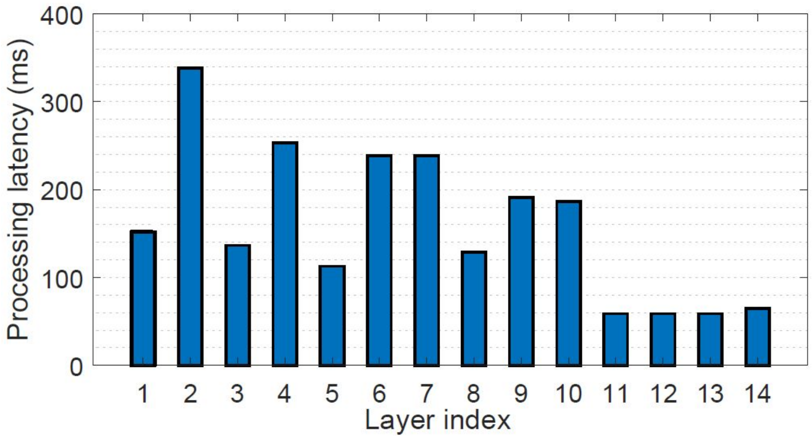
- Each edge device is implemented with a local DNN for local inference.
- The results from each local DNN is first aggregated locally.
- If the local exit is not confident, the activation output after the last convolutional layer from each end device is sent to the cloud aggregator for further processing.

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

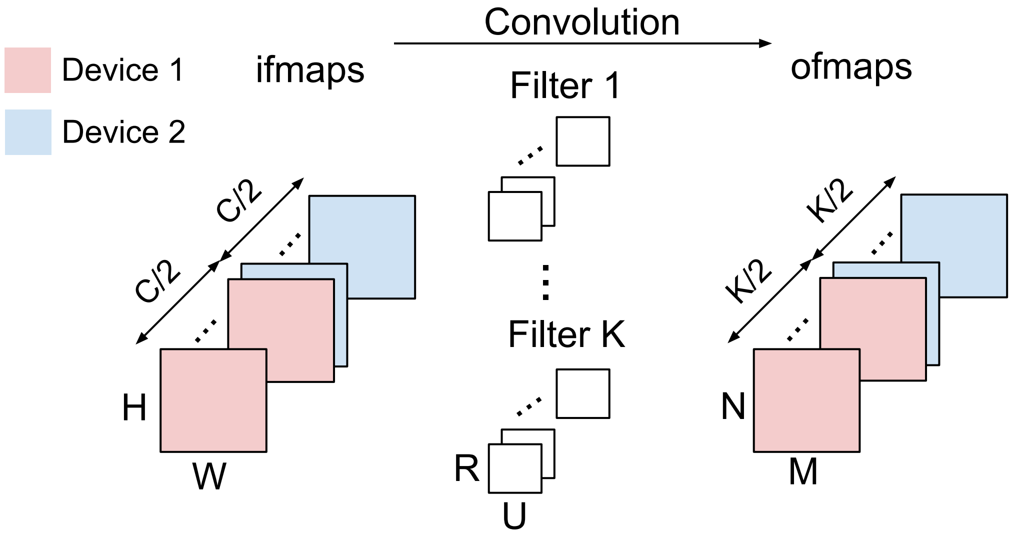
Processing time for VGG16



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

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

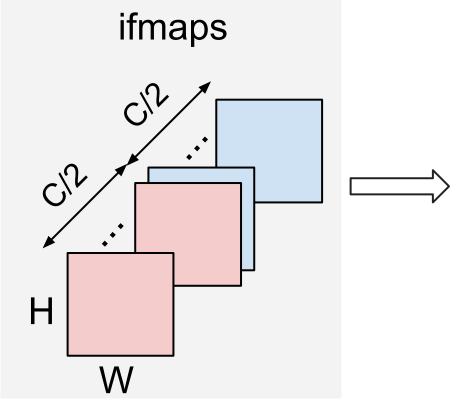


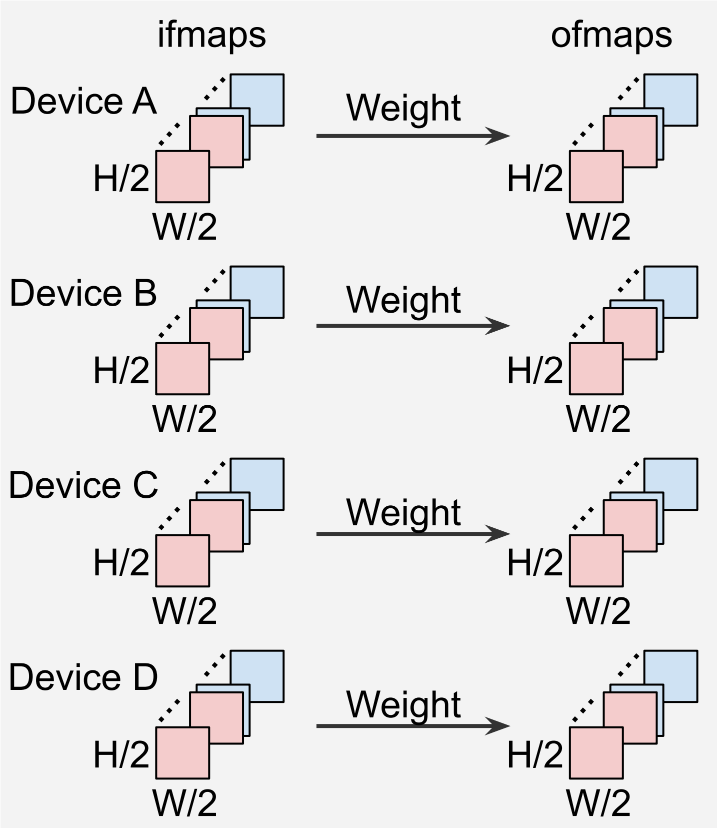
- In channelwise partition, each node needs to exchange their partially accumulated output feature maps to produce final output feature maps, which leads to a significant communication overhead.

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

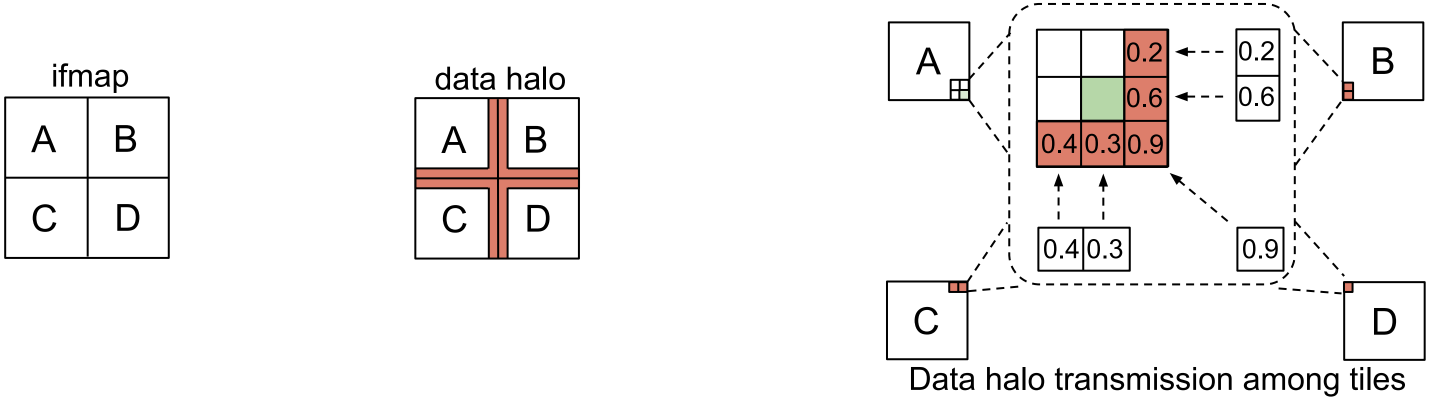




- The input will partitioned in spatial dimension and distribute over multiple devices.
- The weight will duplicate and save on each device.

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.

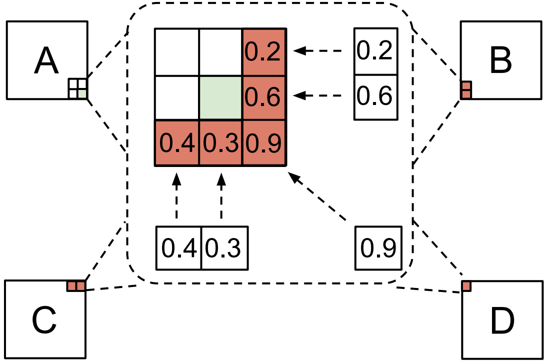
ADCNN



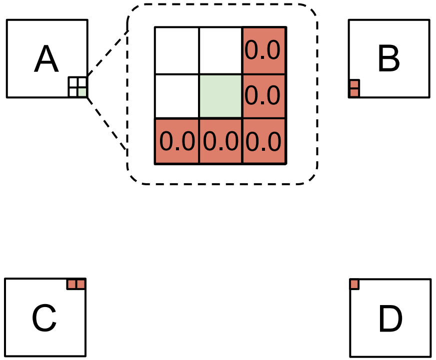
(a) (b) (c)

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

ADCNN



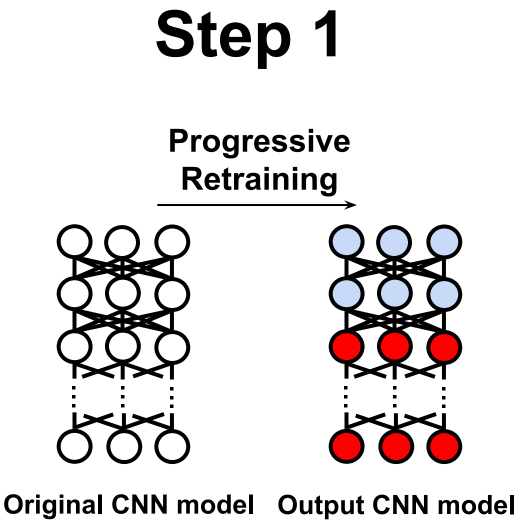
Normal Spatial Partition



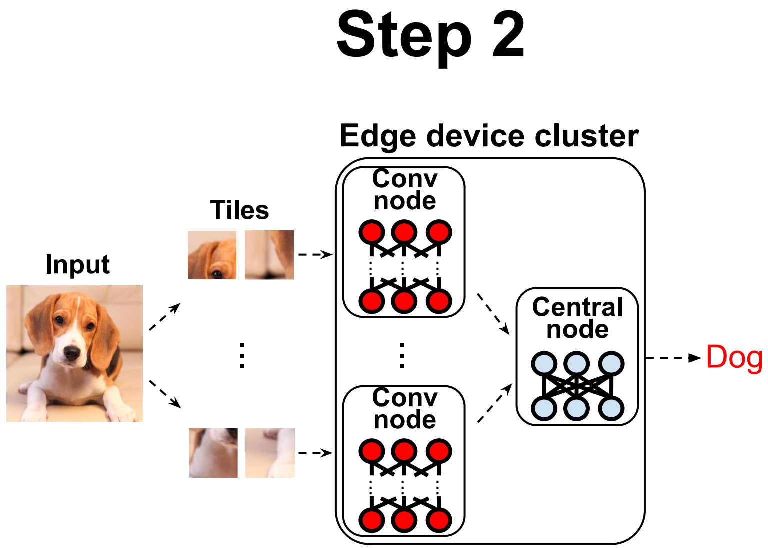
Fully Decomposable Spatial Partition (FDSP)

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

ADCNN



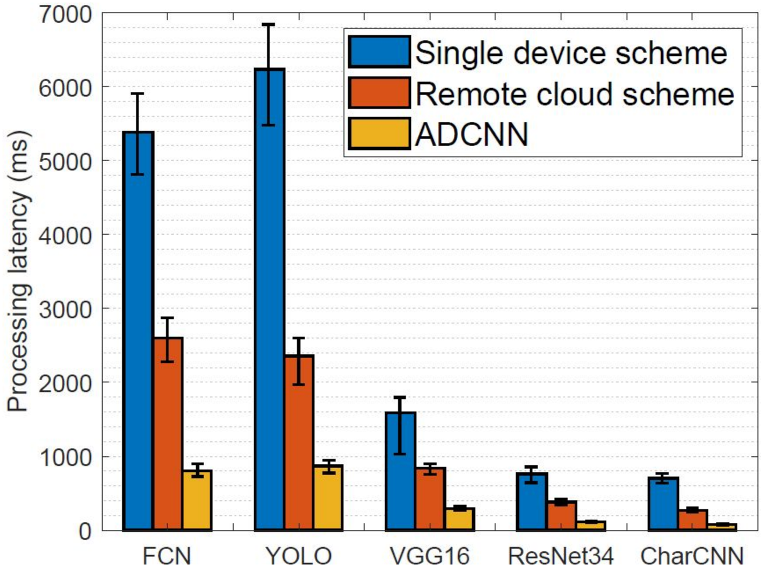
Step 1
Progressive
Retraining
Original CNN model
Output CNN model



Step 2
Edge device cluster
Tiles
Input
Conv
node
Conv
node
Central
node
Dog

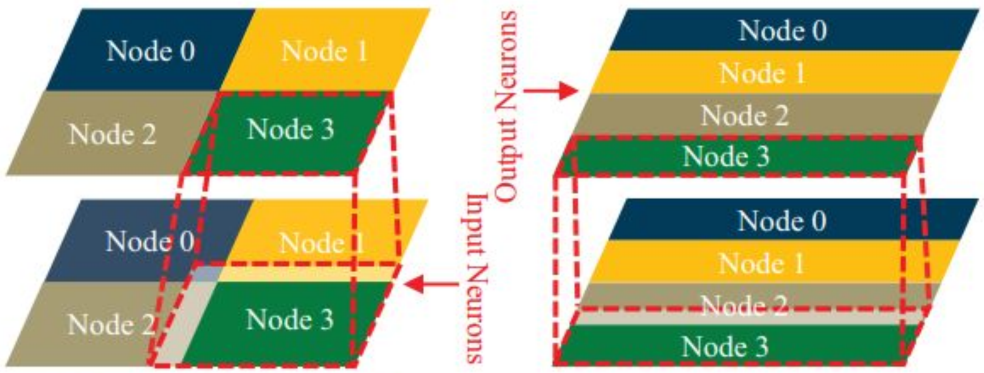
Evaluation Results

- 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

MoDNN



(a) Neurons in 2D-grids of 4 nodes. (b) BODP for 4 worker nodes.

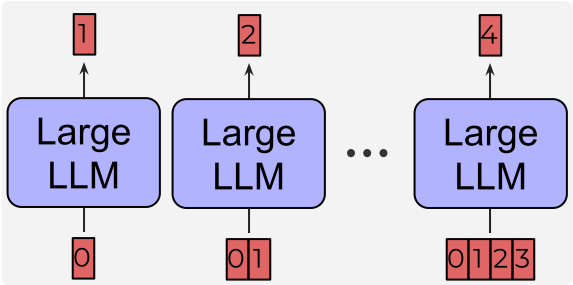
Mao, Jiachen, et al. "Modnn: Local distributed mobile computing system for deep neural network." *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 2017.

Topics

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

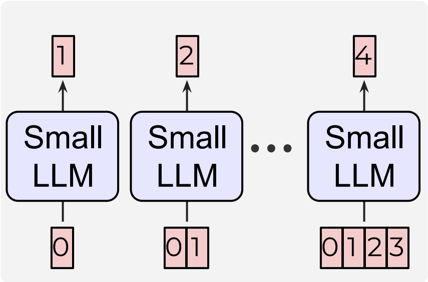
NYU SAI LAB

Speculative Decoding



Accurate but slow

$T_{tot} = NT_{p,1}$

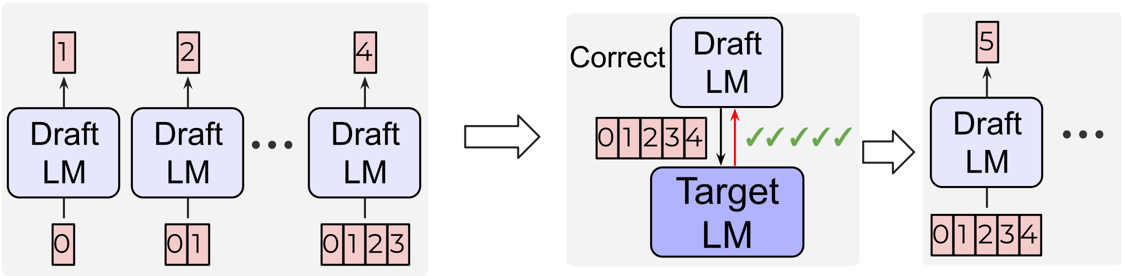


Fast but inaccurate

$T_{tot} = NT_{p,2}$

- Speculative decoding enables lossless token generation with low latency.

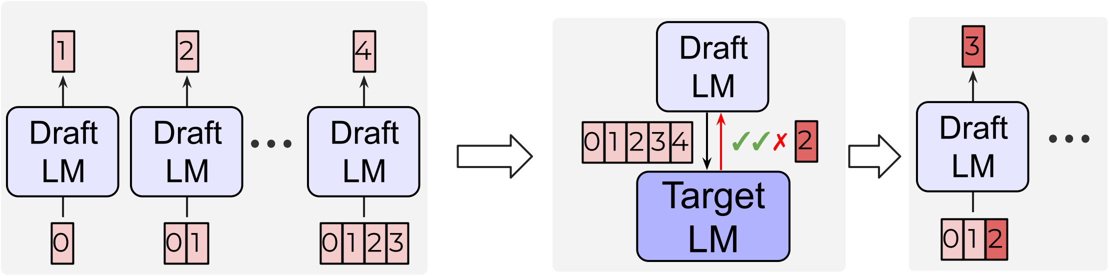
Speculative Decoding



$NT_{p,2}$ T_{val}

$$T_{tot} = NT_{p,2} + T_{val} < NT_{p,1}$$

Speculative Decoding



- If the amount of tokens that pass the verification is too low, then it is possible that speculative decoding is slower than autoregressive baseline.

NYU SAI LAB

Speculative Decoding

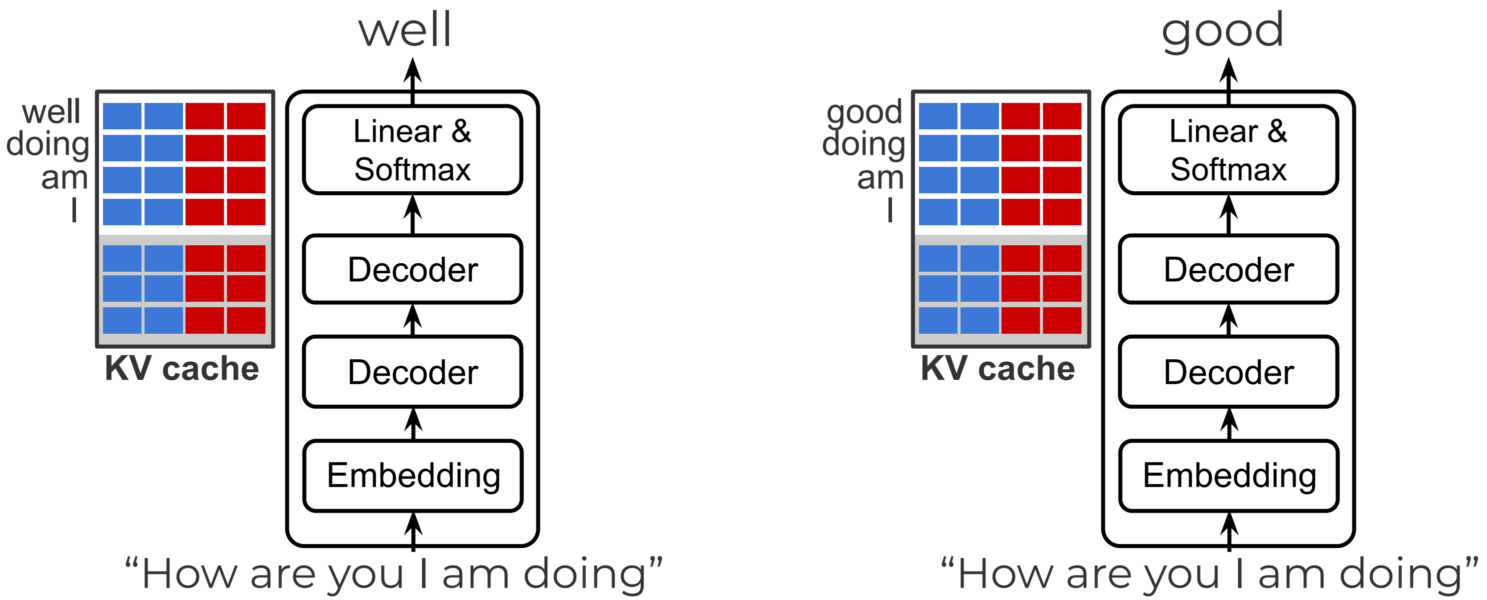
- Speculative decoding does not save computation, but greatly reduce the memory traffic by reducing the number of memory reads, further reducing the **overall latency**.

Algorithm 1 SpeculativeDecodingStep

Inputs: $M_p, M_q, prefix$.
▷ Sample γ guesses $x_{1,\ldots,\gamma}$ from M_q autoregressively.
for $i = 1$ **to** γ **do**
  $q_i(x) \leftarrow M_q(prefix + [x_1, \ldots, x_{i-1}])$
  $x_i \sim q_i(x)$
end for
▷ Run M_p in parallel.
$p_1(x), \ldots, p_{\gamma+1}(x) \leftarrow$
  $M_p(prefix), \ldots, M_p(prefix + [x_1, \ldots, x_\gamma])$
▷ Determine the number of accepted guesses n.
$r_1 \sim U(0,1), \ldots, r_\gamma \sim U(0,1)$
$n \leftarrow \min(\{i - 1 \mid 1 \le i \le \gamma, r_i > \frac{p_i(x)}{q_i(x)}\} \cup \{\gamma\})$
▷ Adjust the distribution from M_p if needed.
$p'(x) \leftarrow p_{n+1}(x)$
if $n < \gamma$ **then**
  $p'(x) \leftarrow norm(max(0, p_{n+1}(x) - q_{n+1}(x)))$
end if
▷ Return one token from M_p, and n tokens from M_q.
$t \sim p'(x)$
return $prefix + [x_1, \ldots, x_n, t]$

NYU SAI LAB

LLM Decoding



- We can simply select the token with the highest score. But better results are achieved if the model considers other words as well. So a better strategy is to sample a word from the entire list using the score as the probability of selecting that word.

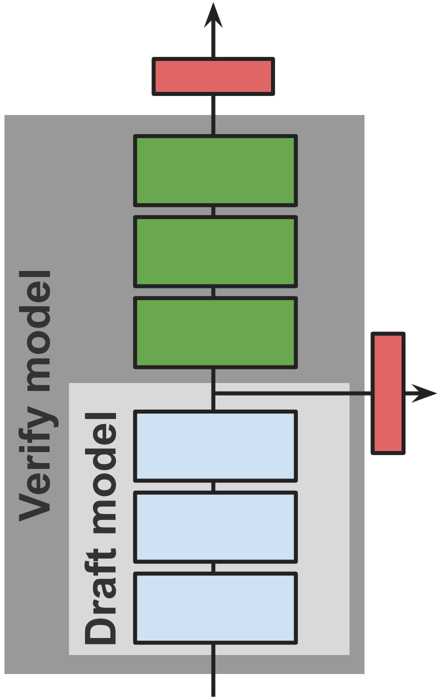
NYU SAI LAB

Speculative Decoding

- To increase the diversity of the LLM output, a better strategy is to sample a word from the entire list using the score as the probability of selecting that word.
- Let $p(x)$, $q(x)$ denote the probability density function specified by the target and draft LLM
- To sample $x \sim p(x)$, we instead sample $x \sim q(x)$, keeping it if $q(x) \leq p(x)$, and in case $q(x) > p(x)$ we reject the sample with probability $1 - p(x)/q(x)$ and sample x again from an adjusted distribution $p'(x) = norm(max(0, p(x) - q(x)))$ instead.

Leviathan, Yaniv, Matan Kalman, and Yossi Matias. "Fast inference from transformers via speculative decoding." *International Conference on Machine Learning*. PMLR, 2023.

Self-Speculative Decoding

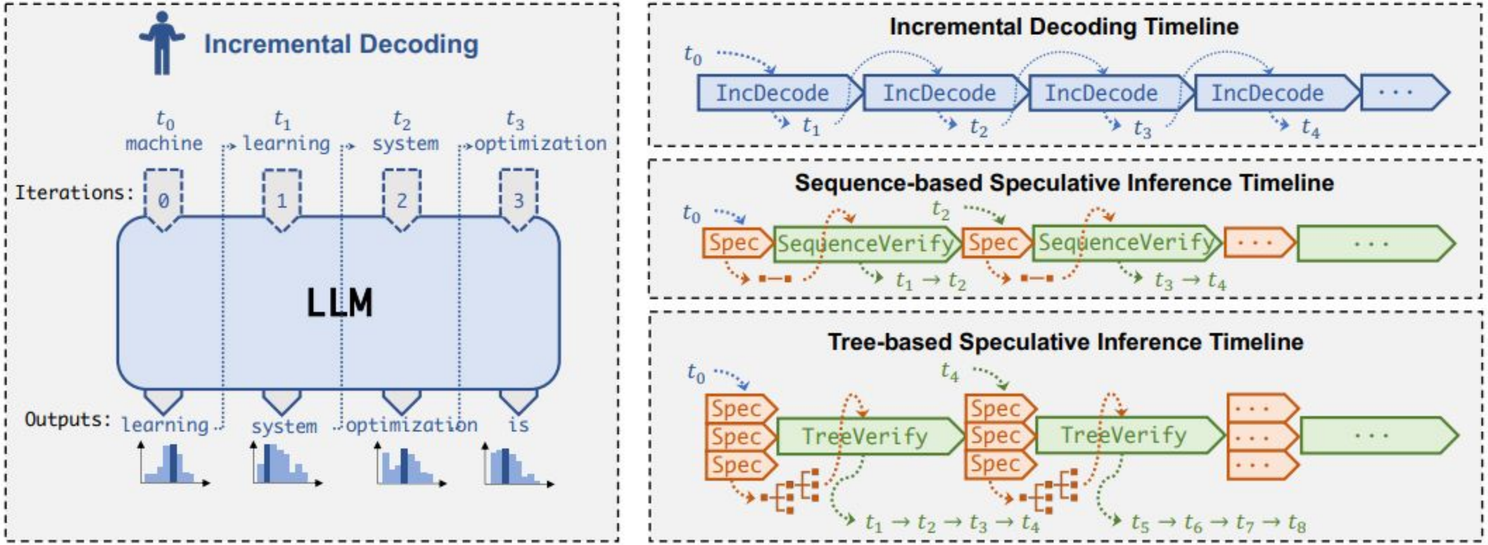


- Self-Speculative decoding the draft model is a subnetwork of the verify model. All the intermediate results from the draft model are reusable.
- No additional network needs to be trained, except a simple classification layer.

NYU SAI LAB

Zhang, Jun, et al. "Draft & verify: Lossless large language model acceleration via self-speculative decoding." *arXiv preprint arXiv:2309.08168* (2023).
Elhoushi, Mostafa, et al. "Layer skip: Enabling early exit inference and self-speculative decoding." *arXiv preprint arXiv:2404.16710* (2024).

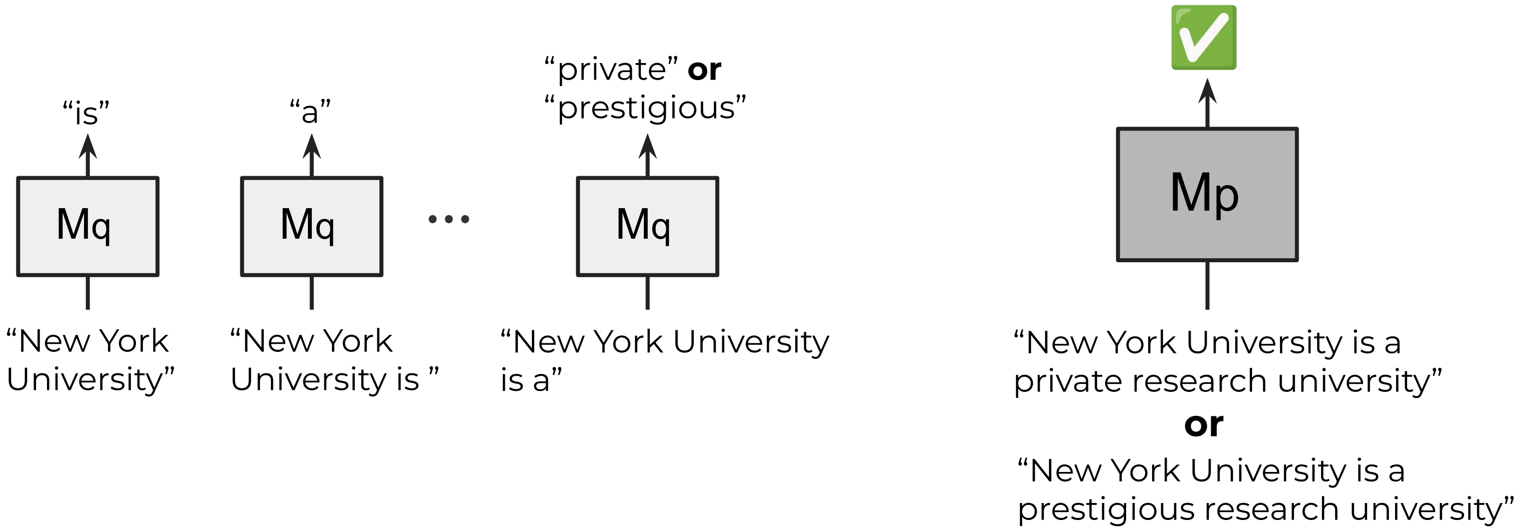
SpecInfer



(a) Incremental decoding.　　**(b)** Timeline Comparison.

Miao, Xupeng, et al. "SpecInfer: Accelerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification." *arXiv preprint arXiv:2305.09781* (2023).

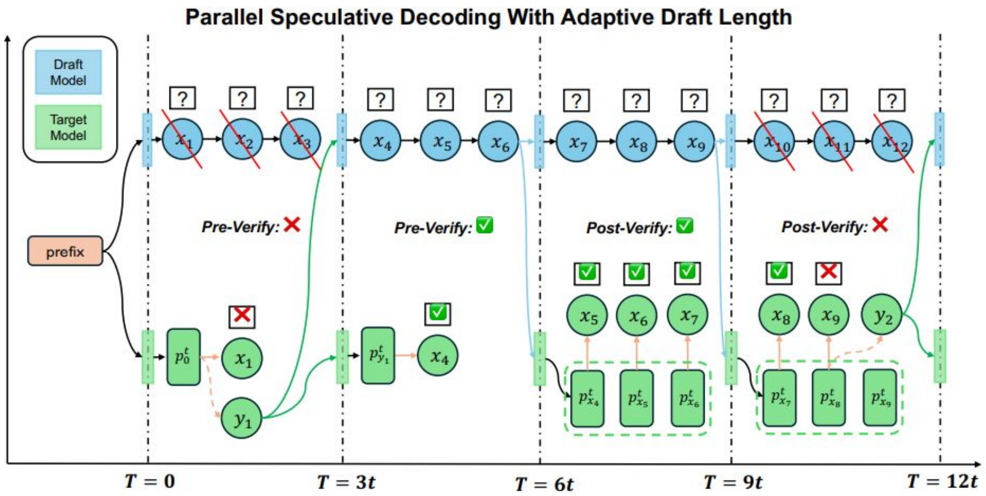
SpecInfer



NYU SAI LAB

Miao, Xupeng, et al. "SpecInfer: Accelerating Generative Large Language Model Serving with Tree-based Speculative Inference and Verification." *arXiv preprint arXiv:2305.09781* (2023).

Parallel Speculative Decoding

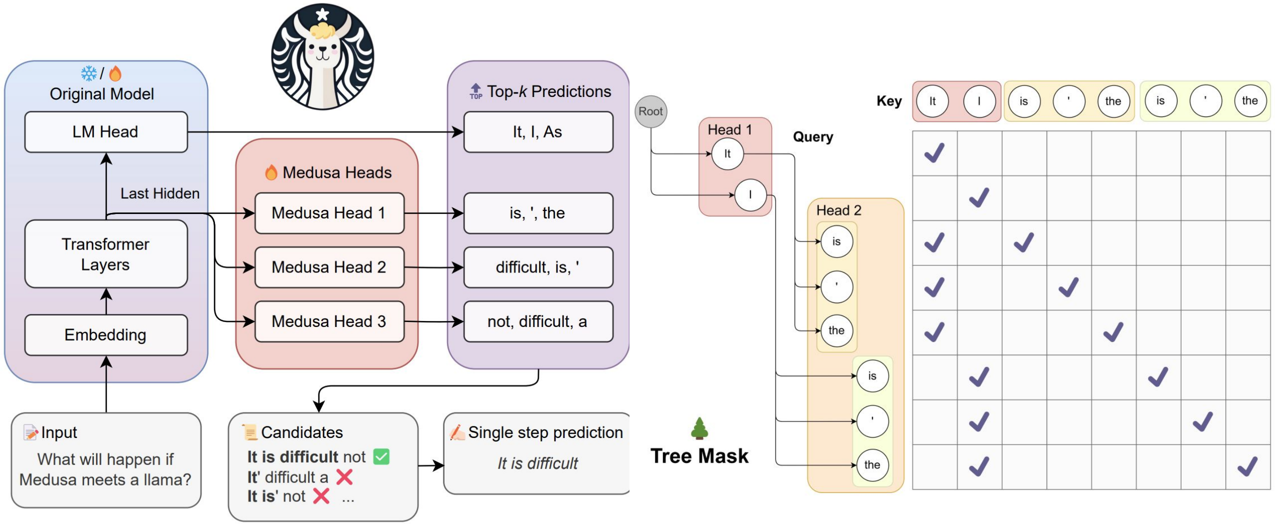


- PEARL is a parallel inference framework based on speculative decoding which utilizes pre-verify and post-verify to achieve adaptive draft length.
- The draft model continues to decode during the verification stage.
- If the verification fails, the windows size will become 1 in the next cycle.

NYU SAI LAB

Liu, Tianyu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. "Parallel speculative decoding with adaptive draft length." arXiv preprint arXiv:2408.11850 (2024).

Medusa



- Adding extra decoding heads to predict multiple subsequent tokens in parallel.

Cai, Tianle, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. "Medusa: Simple llm inference acceleration framework with multiple decoding heads." arXiv preprint arXiv:2401.10774 (2024).

Presentation

- Federated optimization in heterogeneous networks (Rujuta)
- TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning (Akram)
- Modnn: Local distributed mobile computing system for deep neural network (Archit)
- MEDUSA: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads (Aryan)
- Kangaroo: Lossless Self-Speculative Decoding via Double Early Exiting (Roshan Nayak)