Tencent AI Lab

A RANDOM GOSSIP BMUF PROCESS FOR NEURAL LANGUAGE MODELING

Yiheng Huang* Jinchuan Tian* Lei Han* Guangsen Wang* Xingchen Song†† Dan Su* Dong Yu†

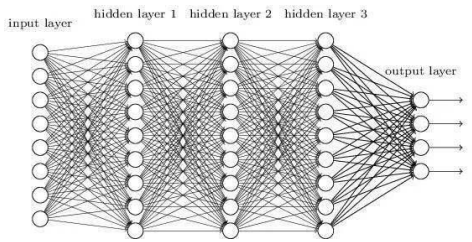
*Tencent AI Lab
††Department of Computer Science and Technology, Tsinghua University, Beijing, China
†Tencent AI Lab, Bellevue, WA, USA

Delivered by Dr. Yiheng Huang

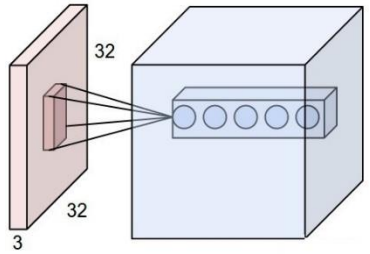
Introduction: Background

Tencent AI Lab

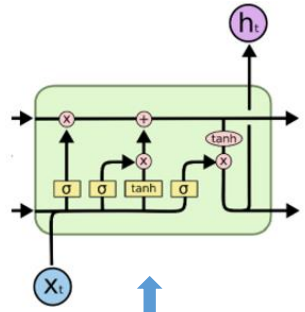
DNN



CNN



LSTM

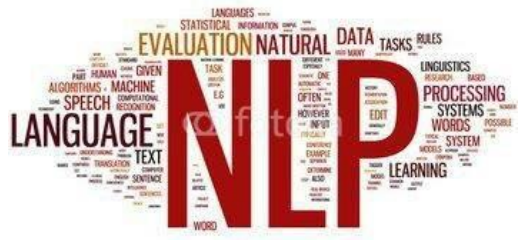


Machine Learning

Deep Learning

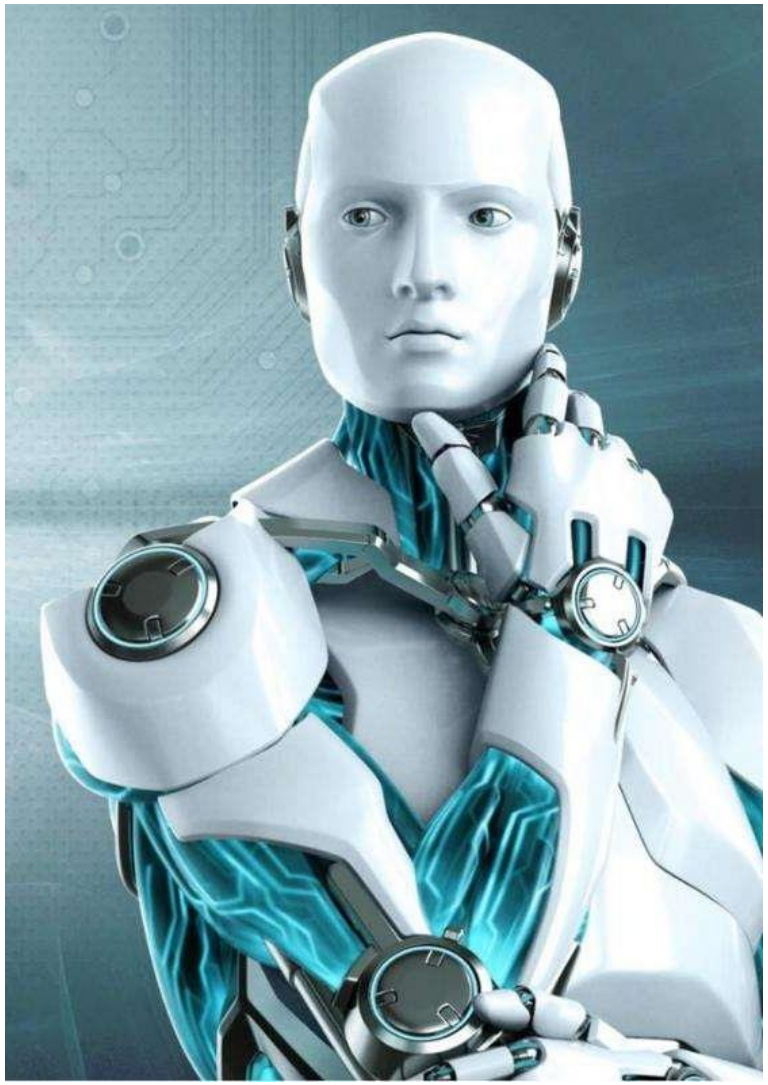
Computer Vision

Speech Recognition



Natural Language Processing

Introduction: Background



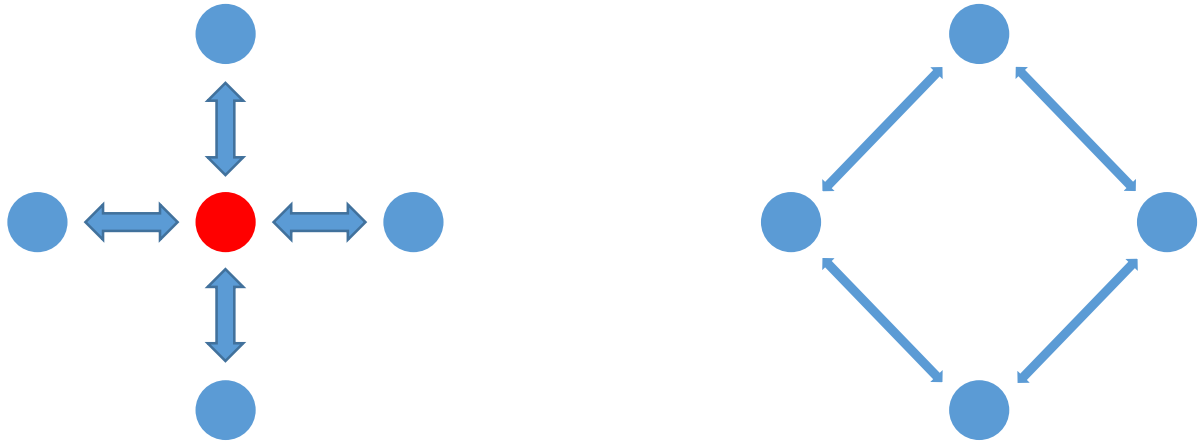
As data size and model complexity increase, one essential challenge is to leverage between scaling the learning procedure and handling big data

Distributed Learning Algorithms are necessities in industrial practices

Introduction: Centralized vs. Decentralized

Tencent AI Lab

System Structure Classification



Centralized(left): Parameter-Server(red node) included vs. Decentralized(right): Parameter-Server excluded

Structure	Features	Drawbacks
Centralized	The parameter-server: • collects/distributes variables (or gradients) from all worker nodes • controls the learning process	• Communication speed with the parameter-server is the bottleneck of the training process
Decentralized	• Collect/Distribute variables (or gradients) by peer-to-peer communication	• Poor accuracy performance compared with Centralized structure

Introduction: Related Works

Tencent AI Lab

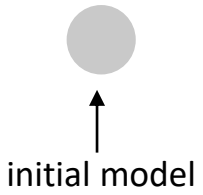
Related Works:

- Centralized methods:
  - Downpour SGD[8]:utilizes thousands of machines to train various deep machines with an asynchronous SGD (ASGD) procedure
  - Hogwild ASGD[9]; EASGD[10]
  - Model Average[12,13] and its derivatives like Blockwise Model Update Filtering(BMUF)[14]:
    - (1) models are updated independently on their own computing nodes
    - (2) the global average without/with momentum item is used to synchronize the models accordingly every a few iterations

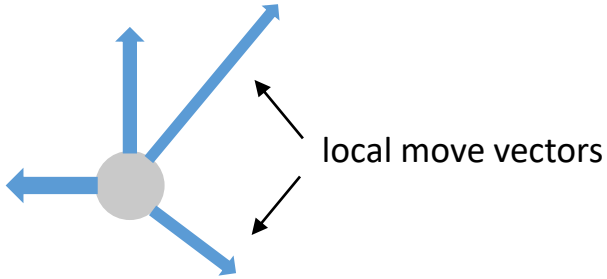
- Decentralized:
  - Gossip[19]: recursively, each node fetches the parameters of another node, and then compute the average between that and the local model to update itself.

Our method, gossip-BMUF, is mainly inspired by BMUF and gossip

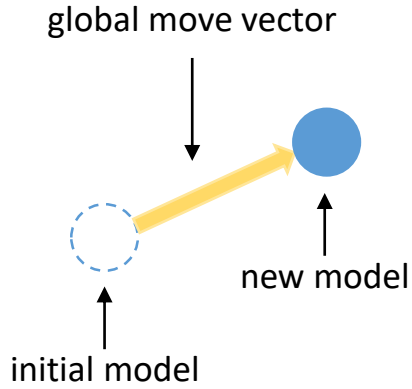
Model Average(MA)



①



②



③

Recursively, all nodes update as follow：

① Models are identical on all nodes after latest loop

② Each node updates its local model independently by local data, and then a local move vector(blue arrow) is accessible on each node

③ A global move vector(red arrow) is computed by averaging all local move vectors, and the initial model is updated by the global move vector to obtain the new model

Statistical Analysis of MA

Tencent AI Lab

Under mild assumptions, the MA estimator will asymptotically approach the optimal but with a bias term that is proportional to the number of nodes.

Assume $\boldsymbol{\theta^k}$ is the parameter for worker k, then $\boldsymbol{\theta_t} = [(\boldsymbol{\theta_t^1})^T, \cdots, (\boldsymbol{\theta_t^n})^T]^T$ is the concatenated parameter vector for all workers at time t. Also note $\boldsymbol{\theta_* 1} = [\boldsymbol{\theta_*^T}, \cdots, \boldsymbol{\theta_*^T}]^T$ is the optimal parameters to be estimated.

Theorem* *Let f be a m-strongly convex function with L-lipschitz gradients. Assume that we can sample gradients* $\boldsymbol{g} = \nabla f(\boldsymbol{\theta_t}; \boldsymbol{X_i}) + \boldsymbol{\xi_i}$. *with additive noise with zero mean* $\mathbb{E}[\boldsymbol{\xi_i}] = \mathbf{0}$ *and bounded variance* $\mathbb{E}[\boldsymbol{\xi_i^T \xi_i}] \leq \sigma^2$ *Then, running the MA algorithm, with constant step size* $0 < \alpha < \frac{2}{m+L}$, *the expected sum of squares convergence of the local parameters to the optimal is bounded by*

$$\mathbb{E}[|\boldsymbol{\theta_t} - \boldsymbol{\theta_* 1}|^2] \leq \left(1 - 2\alpha \cdot \frac{mL}{m+L}\right)^t |\boldsymbol{\theta_0} - \boldsymbol{\theta_* 1}|^2 + n\frac{m+L}{2mL}\alpha\sigma^2$$

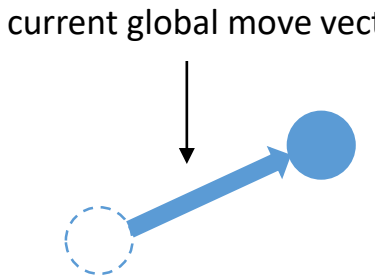
converge to zero as t grows

bias term, grows linearly with number of node: n

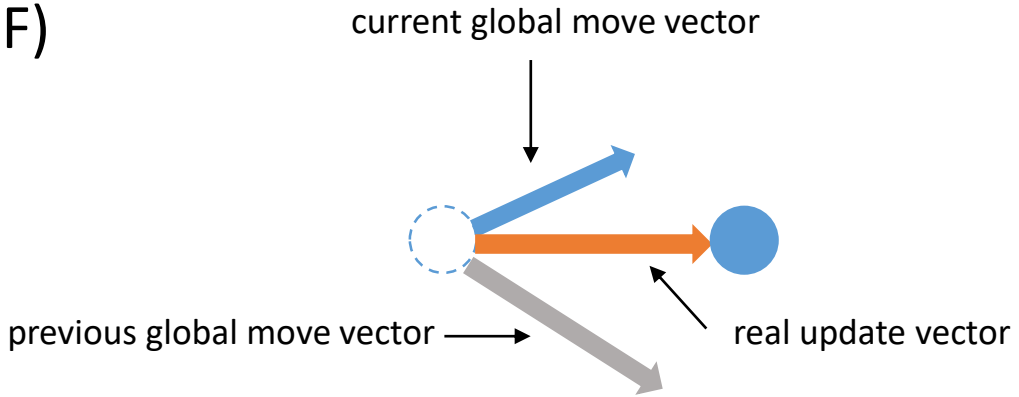
* Learn the proof in our paper

BMUF: a derivative of MA

Blockwise Model Update Filtering(BMUF)



MA



BMUF

In BMUF, the *momentum*-like process is introduced

We note *block momentum, block learning rate and synchronization period* by η, ξ and T respectively; for a model at time t and on rank k, $\boldsymbol{\theta^k}$, function $LocalMoveVector(t,k)$ stands for its local update information of that time and rank. Then BMUF optimizes the model from time t to t+T as follow:

$$\boldsymbol{\theta}_{\boldsymbol{t+T}}^{\boldsymbol{k}} \leftarrow \boldsymbol{\omega}_{\boldsymbol{g}} + \sum_{\boldsymbol{i=0}}^{\boldsymbol{T-1}} LocalMoveVector(t+i,k)$$

$$\overline{\boldsymbol{\theta}_{\boldsymbol{t+T}}} \leftarrow \left(\sum_{\boldsymbol{k=1}}^{\boldsymbol{n}} \boldsymbol{\theta}_{\boldsymbol{t+T}}^{\boldsymbol{k}}\right)/n$$

$$\boldsymbol{G} \leftarrow \overline{\boldsymbol{\theta}_{\boldsymbol{t+T}}} - \boldsymbol{\omega}_{\boldsymbol{g}}$$

$$\boldsymbol{\Delta} \leftarrow \eta\boldsymbol{\Delta} + \xi\boldsymbol{G} \;\; (0 \le \eta < 1, \xi > 0)$$

$$\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} + \boldsymbol{\Delta}$$

$$\begin{cases} \boldsymbol{\omega}_{\boldsymbol{g}} \leftarrow \boldsymbol{\omega} \;\; if\ Classic\ Block\ Momentum(CBM) \\ \boldsymbol{\omega}_{\boldsymbol{g}} \leftarrow \boldsymbol{\omega} + \eta\boldsymbol{\Delta} \;\; if\ Nesterov\ Block\ Momentum(NBM) \end{cases}$$

$$\boldsymbol{\theta}_{\boldsymbol{t+T}}^{\boldsymbol{k}} \leftarrow \boldsymbol{\omega}_{\boldsymbol{g}}$$

BMUF: from Centralized to Decentralized

Tencent AI Lab

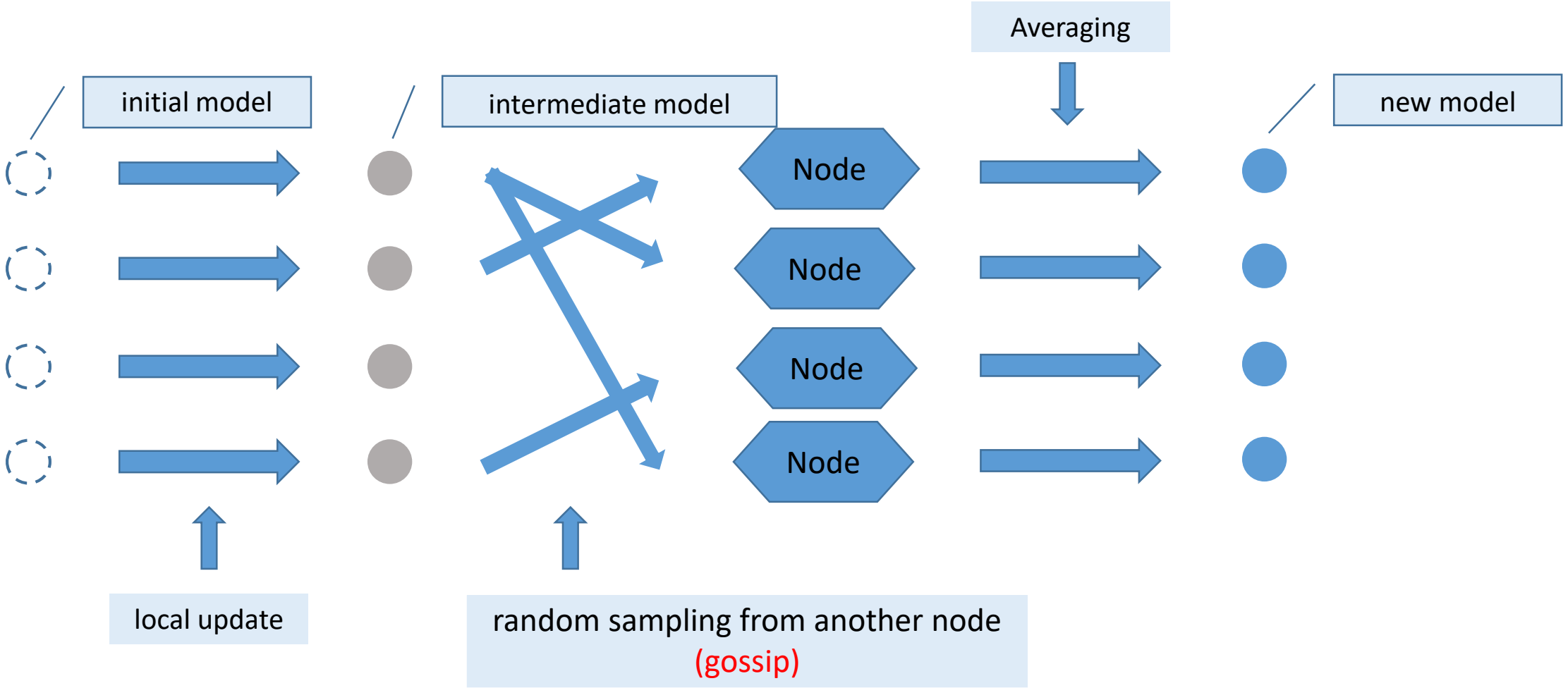
We further optimize BMUF by introducing gossip-like process

- BMUF is centralized and requires global communication: an *all-reduce* process. Problems occur since:
  - Communication latency is considerable when number of nodes grows

- gossip-BMUF, with *gossip* process introduced, is in decentralized style and benefits from:
  - Communication effort is significantly reduced as only local communications are needed

Naïve Gossip Process

The *gossip* process described in [19] acts as follow

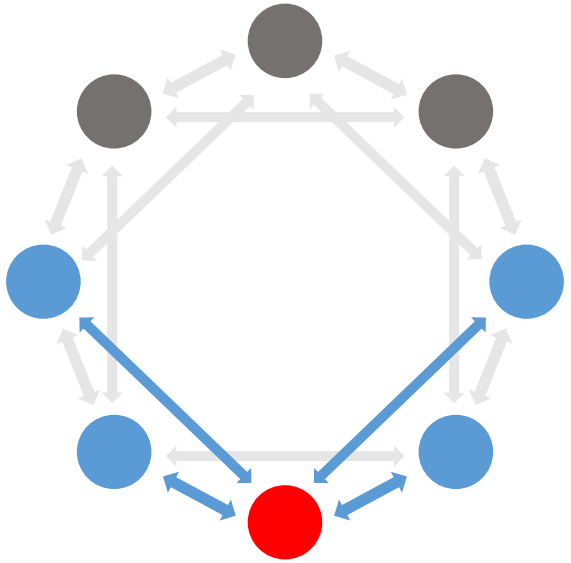


Gossip Process: Our Modified Version

Tencent AI Lab

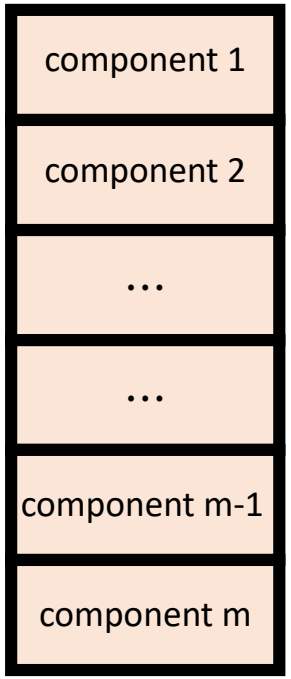
Our randomness in *gossip* process, however, is more complex

Ring Topology: each node only sample its neighbors instead of all other nodes



For each node(like the red one), only *p* nodes on its right and left are connected and could be sampled, called neighbors(the blue ones). *p* is considered *symmetric degree* (Above: 8-node ring topology with 2 symmetric degree)

Model Split: split the model into multiple components



For models of each rank , the model $\boldsymbol{\theta}$ is split into *m* components: $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^T, \cdots, \boldsymbol{\theta}_m^T]^T$

Gossip-BMUF: Pseudo-Code

Tencent AI Lab

- With a n-node, p-symmetric topology, the whole model parameter $\boldsymbol{\theta}$ is split into m components: $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^{\mathbf{T}}, \cdots, \boldsymbol{\theta}_{\mathbf{m}}^{\mathbf{T}}]^{\mathbf{T}}$, and the training dataset $\mathbf{D}$ is evenly split into n partitions: $\mathbf{D} = \cup\, \mathbf{D}_{\mathbf{k=1 \cdots n}}$
- To implement Gossip-BMUF, hyper-parameter setting and additional variable initialization are necessary for each worker k:
  - **Require:** initial model $\boldsymbol{\theta}_0$
  - **Require:** components of the model $\boldsymbol{\theta} = [\boldsymbol{\theta}_1^{\mathbf{T}}, \cdots, \boldsymbol{\theta}_{\mathbf{m}}^{\mathbf{T}}]^{\mathbf{T}}$
  - **Require:** slots $\boldsymbol{\Delta}^{\mathbf{k}}$, $\boldsymbol{\omega}^{\mathbf{k}}$, $\mathbf{G}^{\mathbf{k}}$, with the same shape as $\boldsymbol{\theta}$
  - **Require:** training data with labels $\mathbf{D}$
  - **Require:** synchronous period $\mathrm{H_i}$ for each component $\boldsymbol{\theta}_{\mathbf{i}}$
  - **Require:** number of gossip neighbors $q \leq 2p$
  - **Require:** momentum η and block learning rate ζ
  - **Require:** learning rate α_t

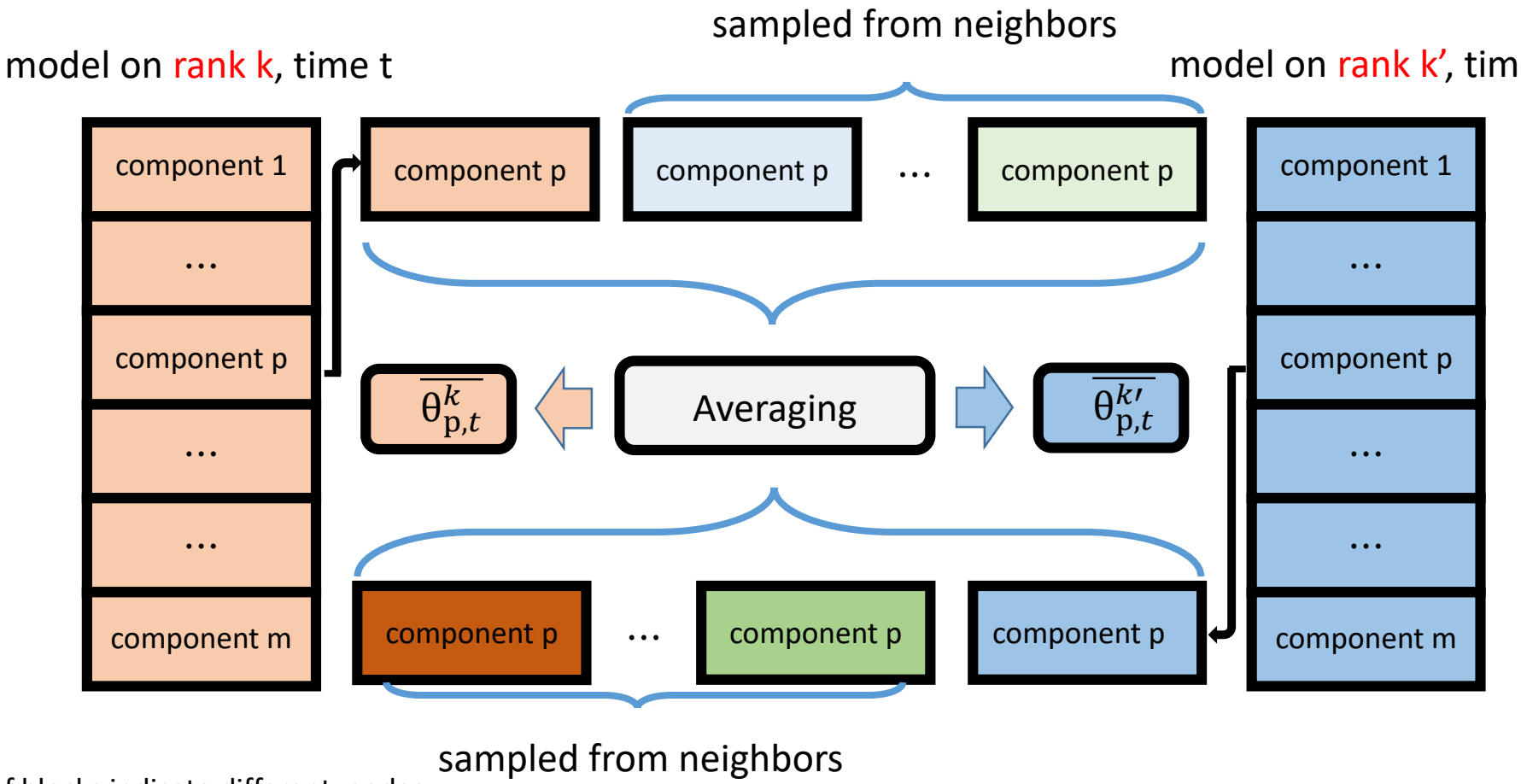
Gossip-BMUF: Pseudo-Code

gossip-BMUF training Process on worker k

1. $\boldsymbol{\theta_0^k \leftarrow \theta_0,\ \omega^k \leftarrow \theta_0,\ \Delta^k \leftarrow 0,\ G^k \leftarrow 0}$
2. For for $\mathrm{i} = 1, \cdots, \mathrm{T}$ do
3. sample a mini-batch from $\mathbf{D_t^k}$ and calculate the gradients $\boldsymbol{g_t^k}$
4. for $\mathrm{i} = 1, \cdots,$ m parallel do
5. $\boldsymbol{\theta_{i,t}^k \leftarrow \theta_{i,t-1}^k - \alpha_t \cdot g_{i,t}^k}$
6. if t mod $\mathrm{H_i}$ == 0 then
7. randomly choose q neighbors $\mathrm{k_1} \cdots, \mathrm{k_q}$
8. $\boldsymbol{\bar{\theta}_{i,t}^k \leftarrow \frac{1}{q+1}\left(\theta_{i,t}^k + \sum_{1 \le j \le q} \theta_{i,t}^{kj}\right)}$
9. if Gossip-BMUF then
10. $\boldsymbol{G_i^k \leftarrow \bar{\theta}_{i,t}^k - \theta_{i,t-H_i}^k}$
11. $\boldsymbol{\Delta_i^k \leftarrow \eta \Delta_i^k + \zeta G_i^k}$
12. $\boldsymbol{\omega_i^k \leftarrow \omega_i^k + \Delta_i^k}$
13. $\boldsymbol{\theta_{i,t}^k \leftarrow \omega_i^k + \eta \Delta_i^k}$ **(Nesterov)**
14. else if Gossip-MA then
15. $\boldsymbol{\theta_{i,t}^k \leftarrow \bar{\theta}_{i,t}^k}$
16. end if
17. end if
18. end for
19. end for
20. return $\boldsymbol{\theta = \frac{1}{n}\sum_{k=1}^{n} \theta_T^k}$

Gossip Process: Our Modified Version

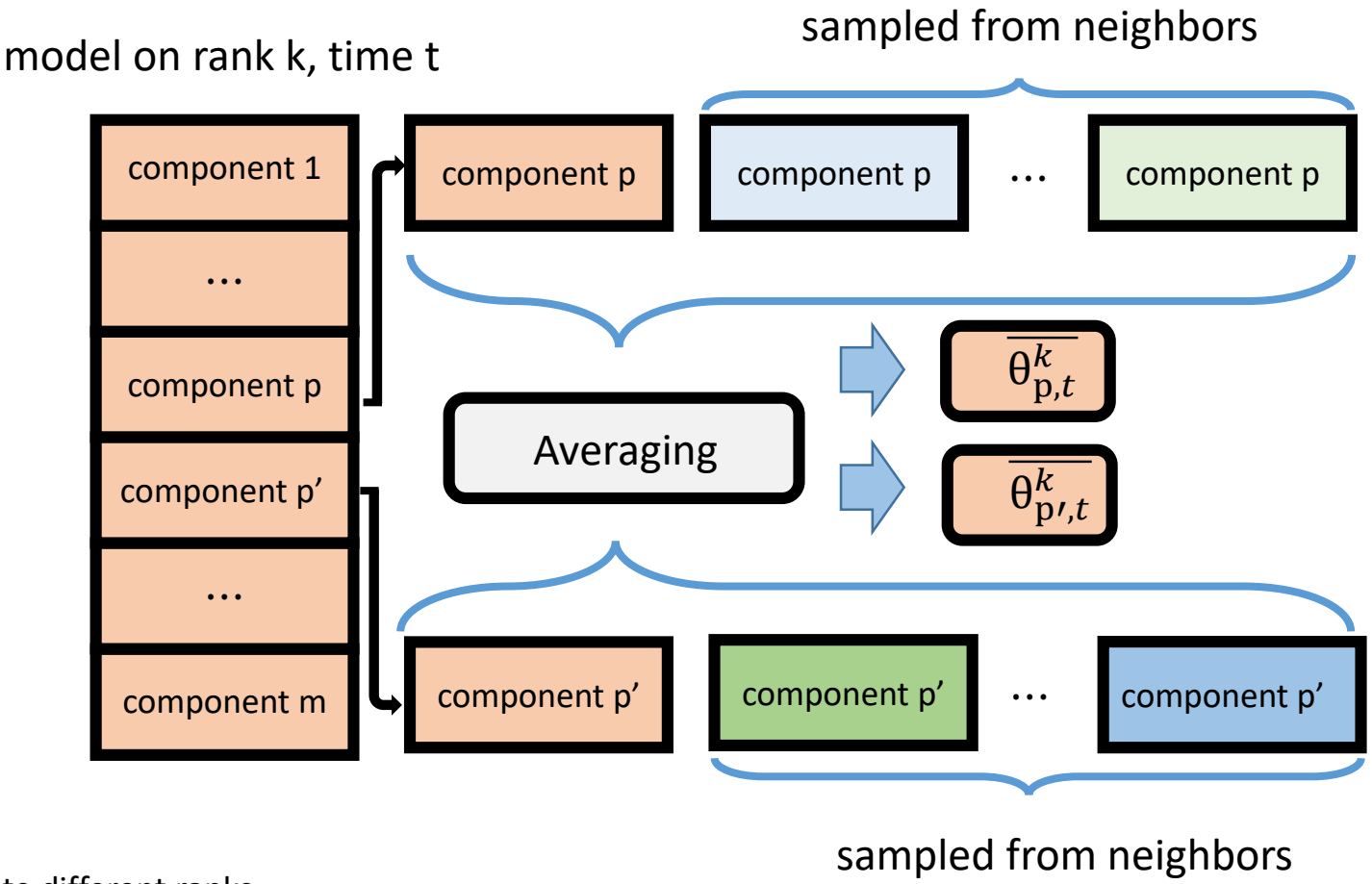
- A *Gossip* process is independently random:
  - on different nodes;
  - for different components of the model;
  - in different iterations where synchronization conducts



* Different colors of blocks indicate different nodes

Gossip Process: Our Modified Version

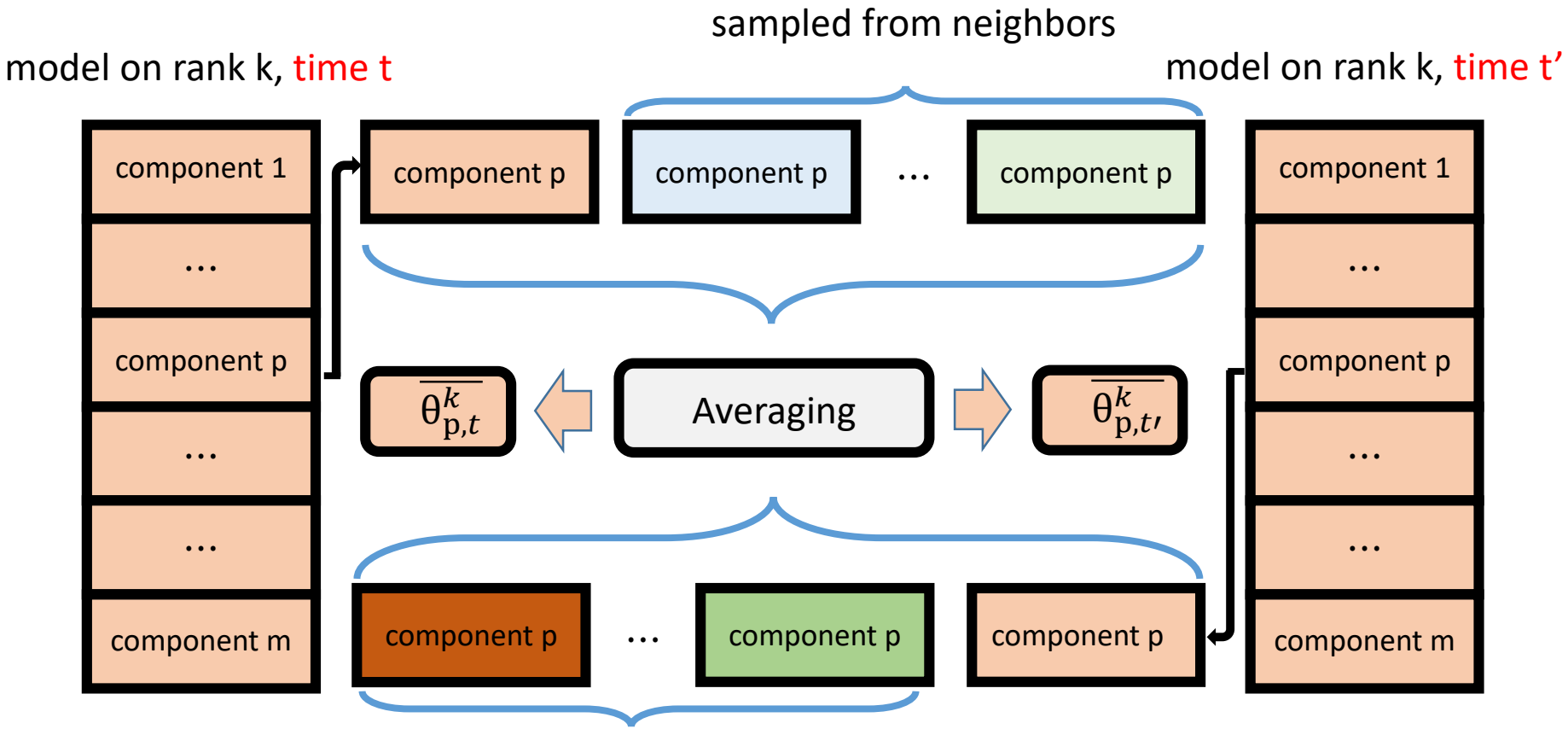
- A *Gossip* process is independently random:
  - on different nodes;
  - for different components of the model;
  - in different iterations where synchronization conducts



* Different colors of blocks indicate different ranks

Gossip Process: Our Modified Version

- A *Gossip* process is independently random:
  - on different nodes;
  - for different components of the model;
  - in different iterations where synchronization conducts



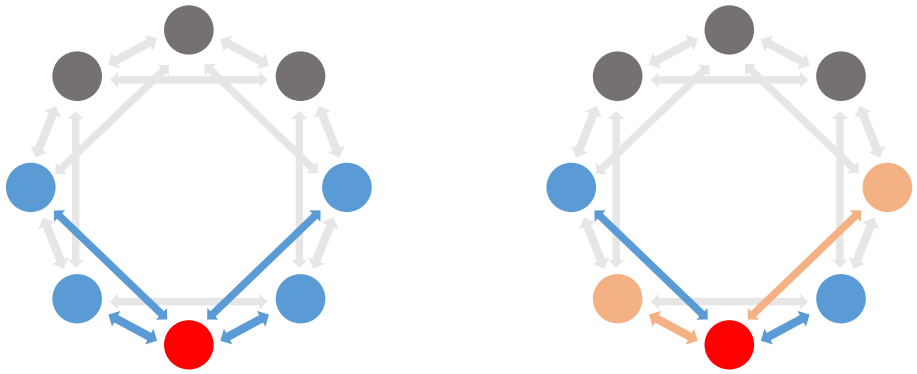
* Different colors of blocks indicate different ranks

Gossip-BMUF: Additional

Tencent AI Lab

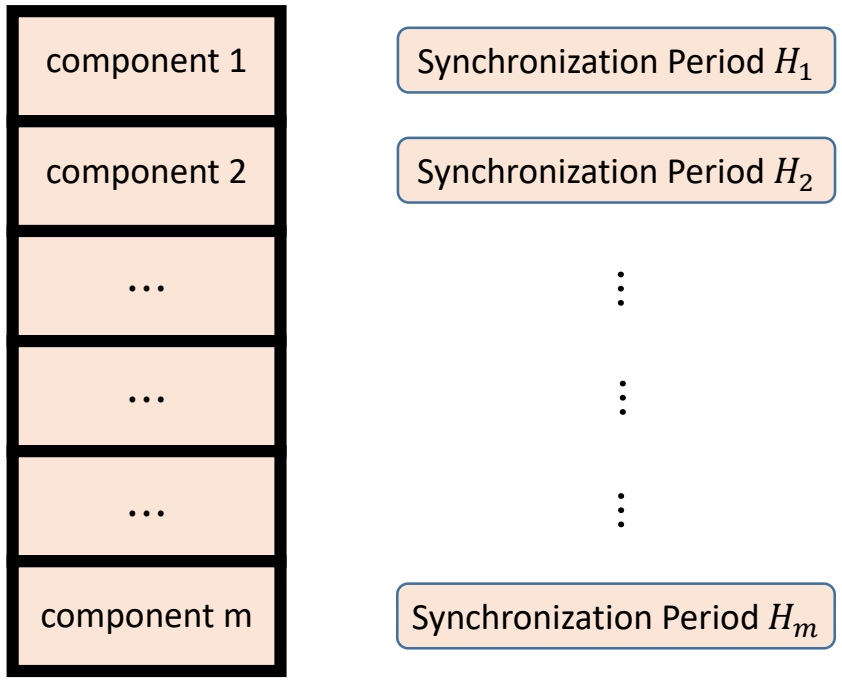
Two things to highlight in Gossip-BMUF

Full Sampling: Gossip process would degrade if all neighbors are sampled



Gossip process would degraded to *local* process if all neighbors are sampled, as no randomness is introduced. (Above: local process(left), gossip process(right). Blue node means being sampled)

Synchronization Period: different components may have different synchronization period



Experiments and Results: Experiment Setting

Tencent AI Lab

Platform

- 2 identical MPI-based HPC machine learning platforms with 16 Nvidia Tesla M40 in total
- Implemented by TensorFlow equipped with Horovod*

Task and Dataset:

- Language Modeling Task
- wiki-text-103: 0.1B tokens; Gutenberg: 0.13B tokens

Model Configuration

- A standard, 1-layer LSTM structure and a projection layer is chosen, with *adaptive-softmax* trick adopted.
- Set block learning rate and momentum to 1.0 and 0.9 respectively
- The whole model is split into groups:
  - Embeddings are evenly split into 8 shards;
  - LSTM weights form one group;
  - LSTMP weights form another group;
  - Head weights in *adaptive-softmax* form one group;
  - Weights of each tails in *adaptive-softmax* form their individual groups.
- Learn more details about our language model setting and optimization strategy in our paper

* An internal modified version of Horovod that contains the implementation of Gossip process

Experiments and Results: Explanation of Experiments

Tencent AI Lab

Explanation of Experiments:

- We use Parentheses "()" to note the synchronization period:
  - (8) means: all variables are synchronized every 8 iterations
  - (16, 128) means: all variables except the embedding tables are synchronized every 16 iterations, while the embedding tables are synchronized every 128 iterations.

- We use curly braces "{}" to note the integer pairs of symmetric degree and number of gossip neighbors like {p, q}
  - For gossip-BMUF / gossip-MA, we choose {1, 1}, {2, 2}, {3, 2} for 4 / 8 / 16 GPUs respectively.
  - For local-BMUF / local-MA, we choose {1, 2}, {2, 4}, {3, 6} for 4 / 8 / 16 GPUs respectively.
  - We empirically recommend to choose $p = \log_2(n) - 1$ where n is the number of nodes

Experiments and Results: Experimental Results on wiki-text-103

Tencent AI Lab

wiki-text-103 Results

Method-period	GPUs	Test ppl	Period	GPUs	Test ppl
MA-(8)	4	55.1	(16,128)	8	64.1
BMUF-NBM-(8)	4	50.7	(16,128)	8	50.3
local-BMUF-(8)	4	51.0	(16,128)	8	50.4
gossip-BMUF-(8)	4	48.4	(16,128)	8	49.0
MA-(16,128)	4	54.3	(16,128)	16	80.5
BMUF-NBM-(16,128)	4	51.1	(16,128)	16	50.5
local-BMUF-(16,128)	4	51.5	(16,128)	16	50.6
gossip-BMUF-(16,128)	4	48.5	(16,128)	16	49.3
single-GPU baseline	49.3				

Method-GPUs	period	Speedups
Gossip-BMUF-4	(16,128)	3.42X*
Gossip-BMUF-8	(16,128)	6.32X*
BMUF-4	(16,128)	3.20X
BMUF-8	(16,128)	5.47X

* These results are obtained by our further optimization, differing from initial results in our paper

Experiments and Results: Experimental Results on Gutenberg

Tencent AI Lab

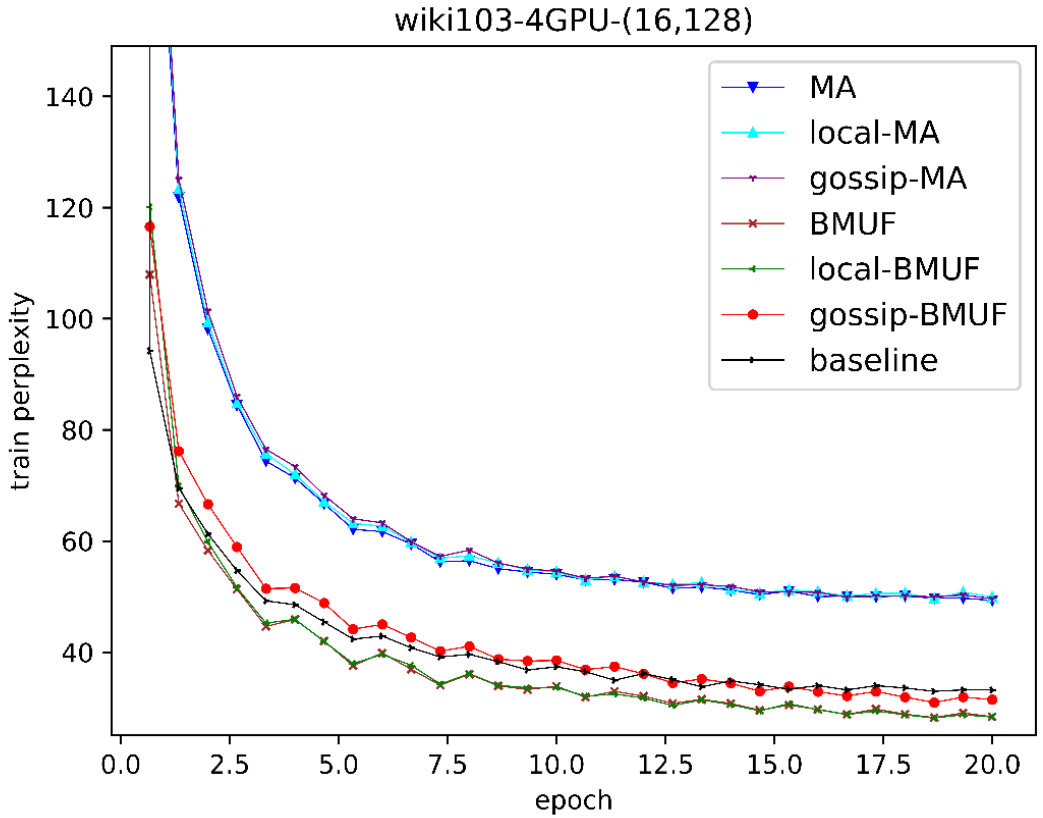
Gutenberg Results

method-period	GPUs	test ppl
MA-(16,128)	4/8/16	158.0/178.9/211.1
BMUF-NBM-(16,128)	4/8/16	156.3/152.6/146.3
local-BMUF-(16,128)	4/8/16	156.1/153.8/148.0
gossip-BMUF-(16,128)	4/8/16	149.4/148.4/146.1
single-GPU baseline	144.6	

Observations:

- gossip-BMUF consistently outperform other methods, and even achieve better results than single-GPU baseline on wiki-text-103 dataset
- The performance degradation of MA is very significant when the number of GPUs is large
- Local-BMUF has a slightly worse performance than BMUF-NBM, which indicates that the randomly selected neighbors are the key success in gossip-BMUF

Experiments and Results: Training Curves of wiki-text-103



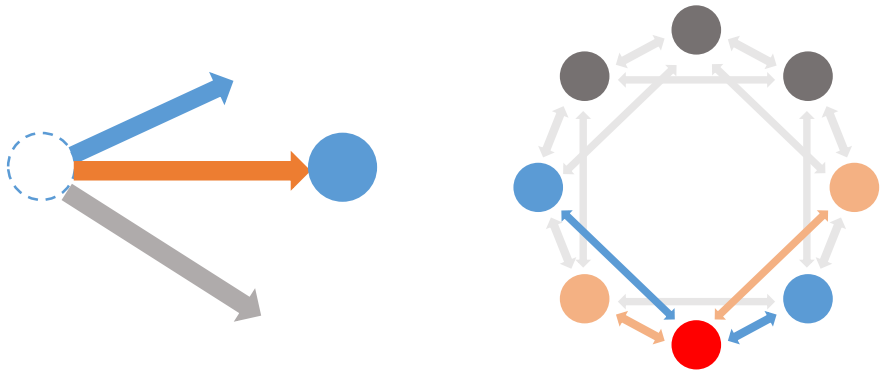
Observation:

- The training curves of gossip-BMUF is very similar to that of single-GPU baseline
- The training curves of BMUF-NBM fluctuates more fiercely than that of gossip-BMUF
- The training curves of BMUF and local-BMUF indicate that the over-fitting might already happen

Conclusion

Tencent AI Lab

combine gossip process(modified) with BMUF: gossip-BMUF



Gossip-BMUF: randomly samples neighbors and act in a BMUF-like process
(Above: BMUF(left), gossip(right))

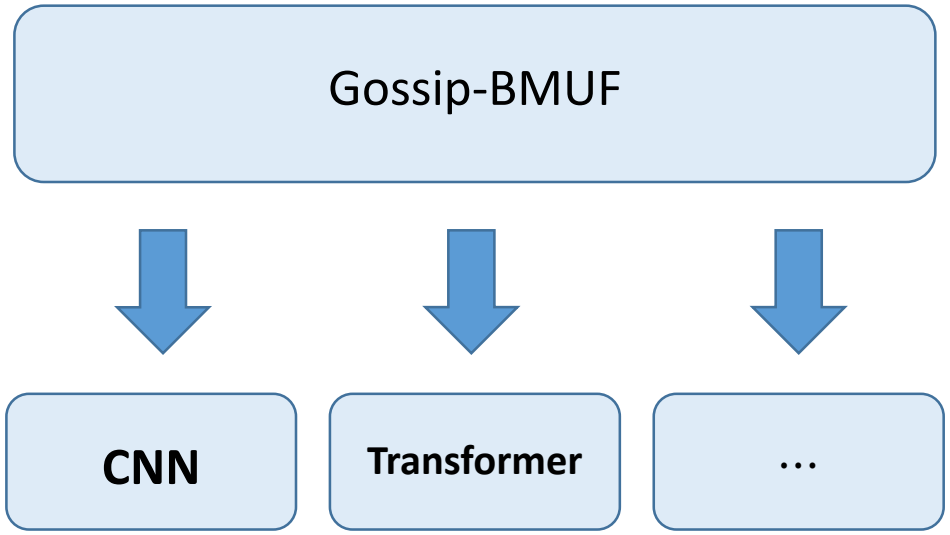
Features of gossip-BMUF

- Decentralized style:
  - Adopt more flexible topology
- Robustness:
  - The training can continue even several nodes get stuck
- Accuracy:
  - Consistently better performance than other competitors on two benchmarks
- Speedup:
  - Considerable speed improvement, as no global communication needed
- Randomness:
  - The random collection of neighbor components are the key success of our algorithm
- Over-fitting reduction:
  - Best approximation of single-GPU baseline and less over-fitting observed compared with BMUF

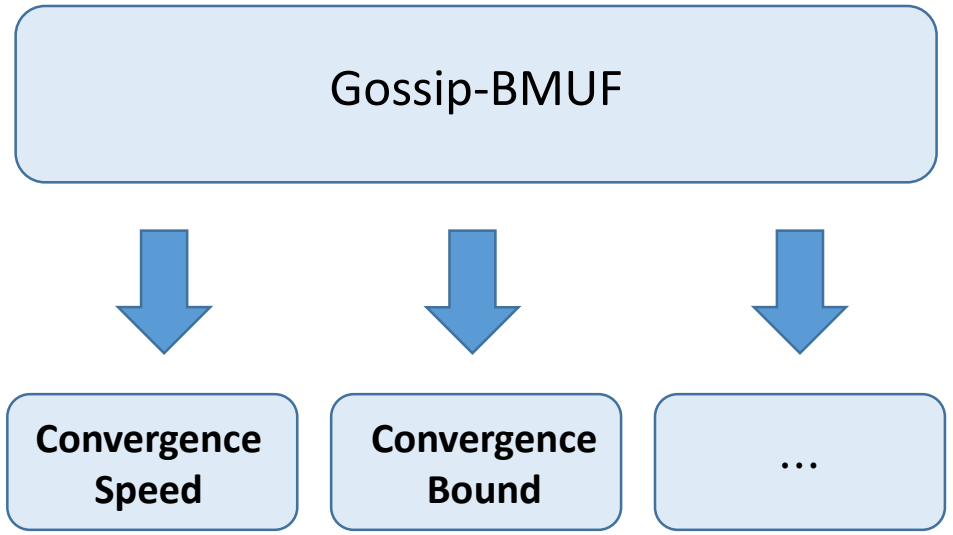
Future Work Perspective

Tencent AI Lab

Experimentally evaluate our method on other deep machines



Statistical Analysis on gossip-BMUF



Thanks for Listening!