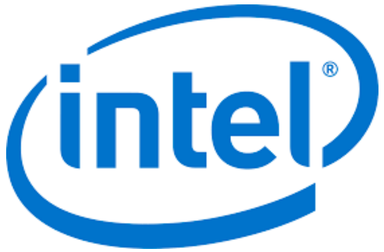


Distributed Learning of Neural Networks using Independent Subnet Training

Anastasios Kyrillidis
Rice CS

Joint work with: Binhang Yuan, Chen Dun, Cameron Wolfe,
Fangshuo Liao, Yuxin Tang, Jingkang Yang,
Santiago Segarra, Chris Jermaine

Introducing myself

Both CS and ECE background:

Undergraduate (5 year diploma): ECE (Tech. Univ. Of Crete – Greece)

M.Sc.: ECE (Tech. Univ. Of Crete – Greece)

Ph.D.: CS (EPFL – Switzerland)

Simons Foundation PostDoc: ECE working on CS (UT Austin)

Goldstine Fellowship PostDoc: CS (IBM)

Introducing myself

Both CS and ECE background:

Undergraduate (5 year diploma): ECE (Tech. Univ. Of Crete – Greece)

M.Sc.: ECE (Tech. Univ. Of Crete – Greece)

Ph.D.: CS (EPFL – Switzerland)

Simons Foundation PostDoc: ECE working on CS (UT Austin)

Goldstine Fellowship PostDoc: CS (IBM)

General research interests: Optimization for ML/AI, Algorithmic theory

Understanding when and why (continuous) algorithms work

Introducing myself

Both CS and ECE background:

Undergraduate (5 year diploma): ECE (Tech. Univ. Of Crete – Greece)

M.Sc.: ECE (Tech. Univ. Of Crete – Greece)

Ph.D.: CS (EPFL – Switzerland)

Simons Foundation PostDoc: ECE working on CS (UT Austin)

Goldstine Fellowship PostDoc: CS (IBM)

General research interests: Optimization for ML/AI, Algorithmic theory

Understanding when and why (continuous) algorithms work

Research Keywords (update annually or so):

Matrix factorization and NNs, acceleration in noncvx settings, sparsity in optimization, overparameterization, quantum verification, quantum algorithms, X-supervised learning, large-scale NN training, synchronous vs. asynchronous computation, pruning algorithms, streaming learning, ..

Motivation

- Scenario #1

Distributed computing scenario

> *"I want to train my neural network quickly, using a really big data set. I've prototyped my learning algorithm on TensorFlow, but a back-of-the-envelope calculation shows it will take me two months to train on a GPU using all of my data. I can't wait that long. I am willing to pay for more machines and get the training done quickly. But if I pay for twice the machines, I should get something close to half the training time."*

Motivation

- Scenario #1

Distributed computing scenario

> *"I want to train my neural network quickly, using a really big data set. I've prototyped my learning algorithm on TensorFlow, but a back-of-the-envelope calculation shows it will take me two months to train on a GPU using all of my data. I can't wait that long. I am willing to pay for more machines and get the training done quickly. But if I pay for twice the machines, I should get something close to half the training time."*

- Scenario #2

Decentralized computing/FL scenario

> This is the case of mobile ML applications. Systems, based on handheld devices and edge infrastructure, introduce further challenges, on top of the challenges in classical distributed learning with traditional compute nodes: *The heterogeneity of edge networks, the ephemeral nature of the mobile workers, the computational and energy restriction of handheld devices, and the communication bottlenecks in wireless networks are some impediments towards reliable distributed ML over edge networks.* Optimizing the NN training in terms of computation/communication/energy efficiency, while retaining competitive accuracy, has become a fundamental challenge.

TL;DR version of the talk

- Independent Subnet Training framework in neural networks

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly
  - New iteration starts, with completely new subnets

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly
  - New iteration starts, with completely new subnets

  - Key attributes of the proposed framework:
    - Fewer parameters sent over distributed network

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly
  - New iteration starts, with completely new subnets

  - Key attributes of the proposed framework:
    - Fewer parameters sent over distributed network
    - Fewer parameters updated per device

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly
  - New iteration starts, with completely new subnets

  - Key attributes of the proposed framework:
    - Fewer parameters sent over distributed network
    - Fewer parameters updated per device
    - (Some times) better final accuracies, (often) faster!

TL;DR version of the talk

- Independent Subnet Training framework in neural networks
  - Each subnet is trained on a different device for a short period of time
  - Updated subnets are transmitted to the parameter server for reassembly
  - New iteration starts, with completely new subnets

  - Key attributes of the proposed framework:
    - Fewer parameters sent over distributed network
    - Fewer parameters updated per device
    - **(Some times) better final accuracies, (often) faster!**

  - Current status: different NN architectures, provide theory close as possible to practice, FL, connections with LTH, and more.

Published at VLDB 2022

Distributed Learning of Neural Networks using Independent Subnet Training

Binhang Yuan
Rice University
by8@rice.edu

Cameron R. Wolfe
Rice University
crw13@rice.edu

Chen Dun
Rice University
cd46@rice.edu

Yuxin Tang
Rice University
yuxin.tang@rice.edu

Anastasios Kyrillidis
Rice University
anastasios@rice.edu

Chris Jermaine
Rice University
cmj4@rice.edu

https://arxiv.org/pdf/1910.02120.pdf

Published at UAI 2022

ResIST: Layer-Wise Decomposition of ResNets for Distributed Training

Chen Dun [*1] Cameron R. Wolfe [*1] Chris Jermaine [1] Anastasios Kyrillidis [1]

https://arxiv.org/pdf/2107.00961.pdf

ON THE CONVERGENCE OF SHALLOW NEURAL NETWORK TRAINING WITH RANDOMLY MASKED NEURONS

Fangshuo Liao, Anastasios Kyrillidis
Department of Computer Science
Rice University
Houston, TX 77005, USA
{Fangshuo.Liao,anastasios}@rice.edu

https://arxiv.org/pdf/2112.02668.pdf

Published at TMLR

GIST: Distributed Training for Large-Scale Graph Convolutional Networks

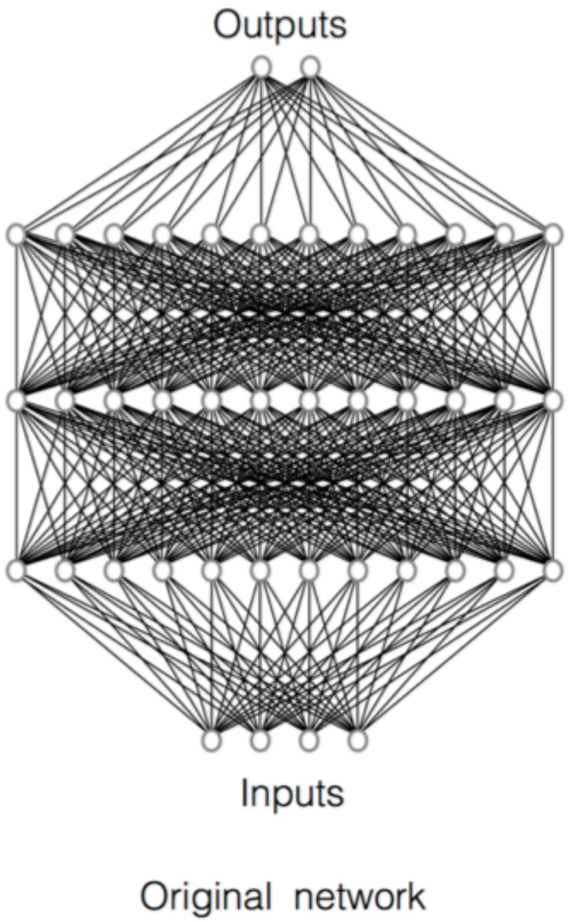
Cameron Wolfe [*1] Jingkang Yang [*2] Arindam Chowdhury [3] Chen Dun [1] Artun Bayer [3] Santiago Segarra [3]
Anastasios Kyrillidis [1]

https://arxiv.org/pdf/2102.10424.pdf

Under review

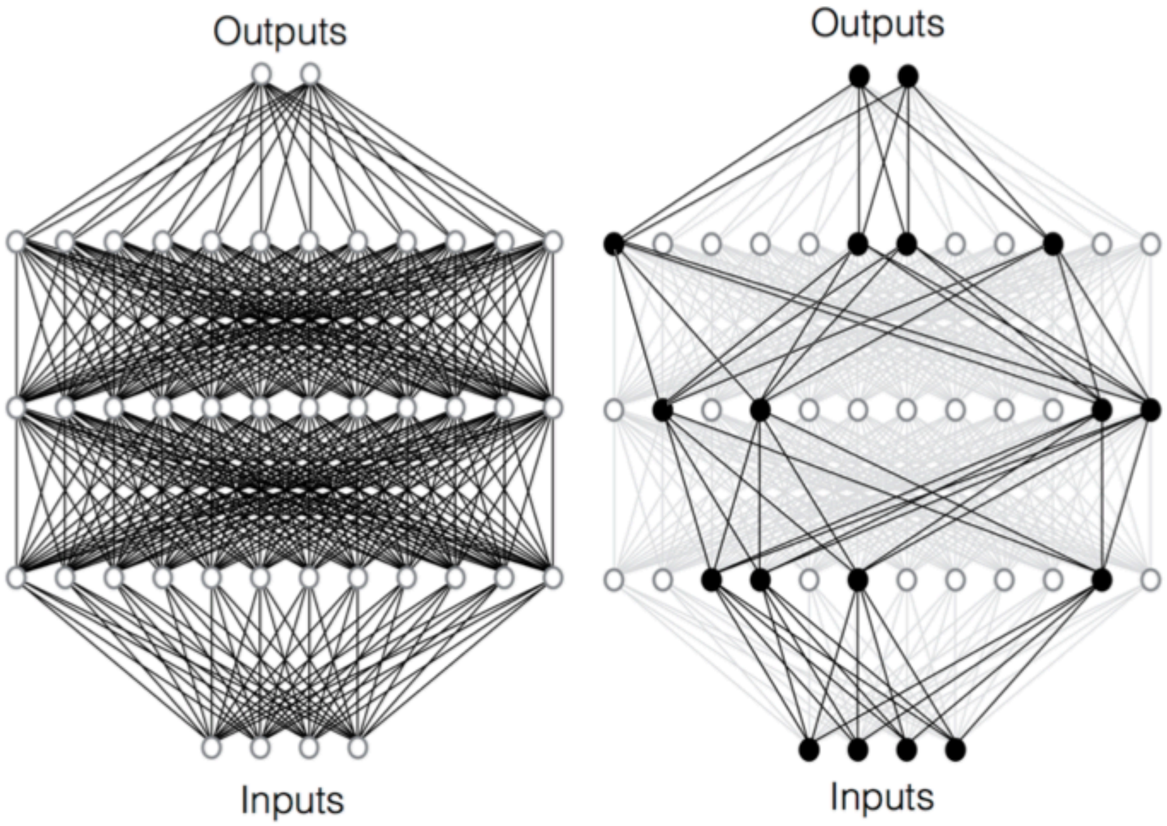
Independent Subnet Training

Independent Subnet Training: NN decomposition



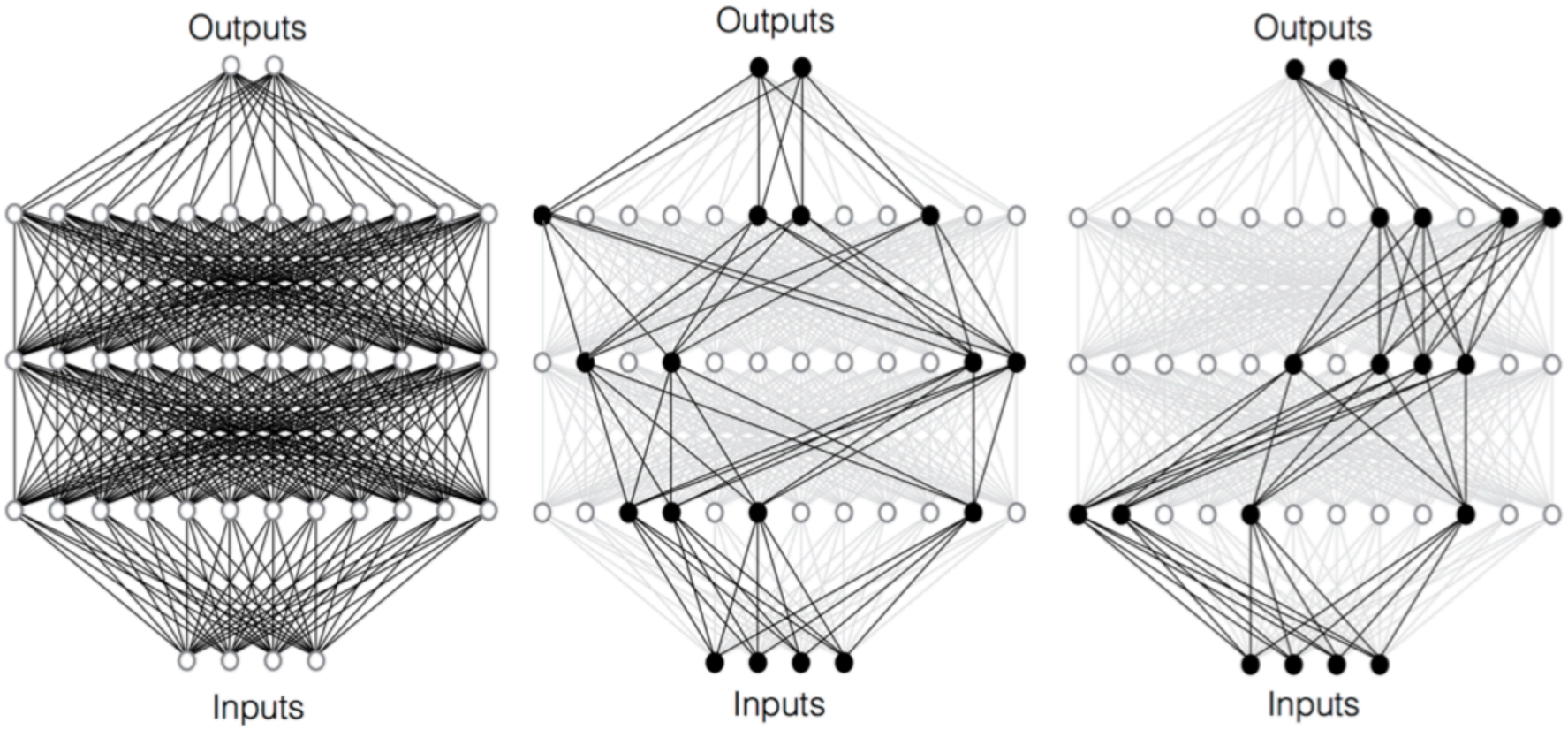
Original network

Independent Subnet Training: NN decomposition



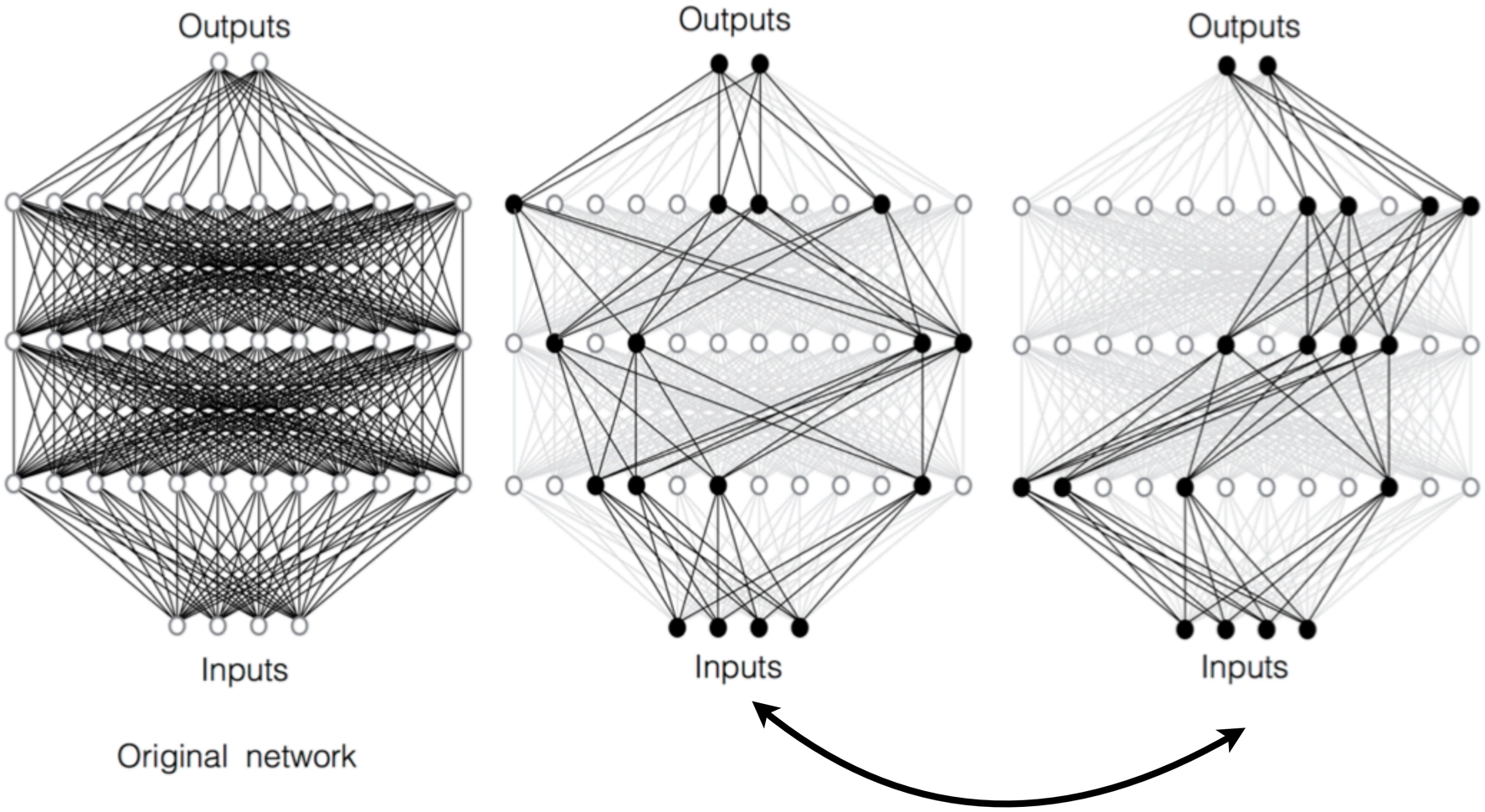
Original network

Independent Subnet Training: NN decomposition



Original network

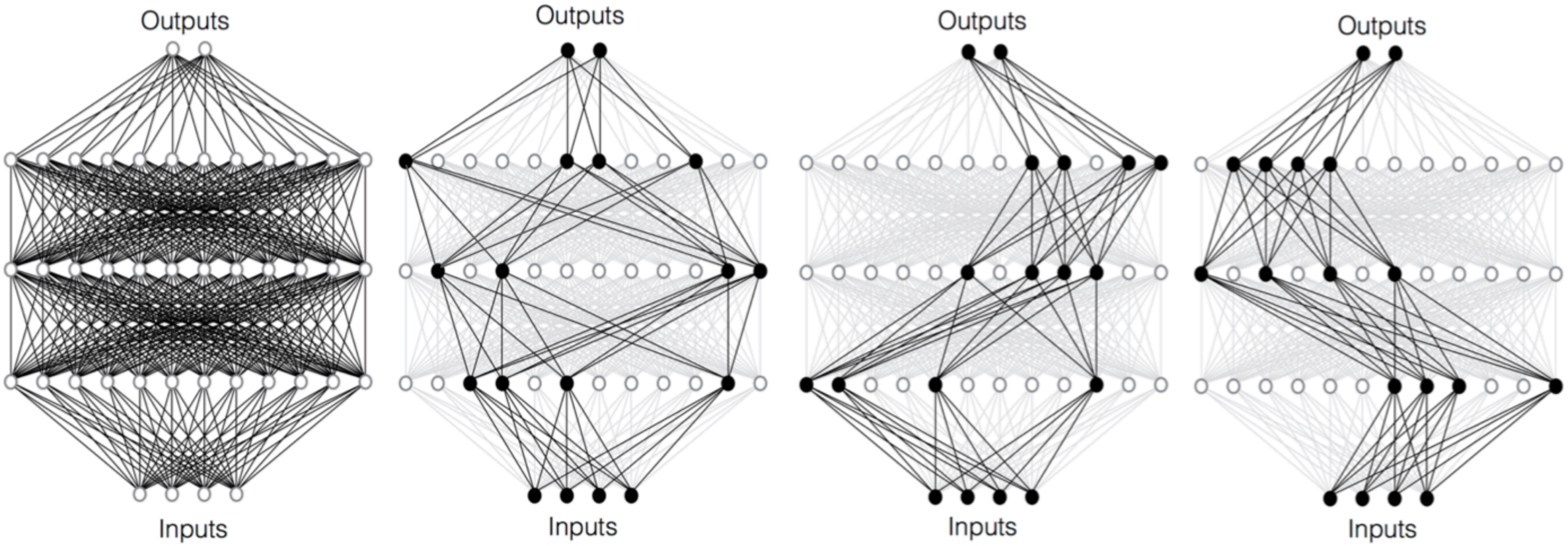
Independent Subnet Training: NN decomposition



No overlap between the weights

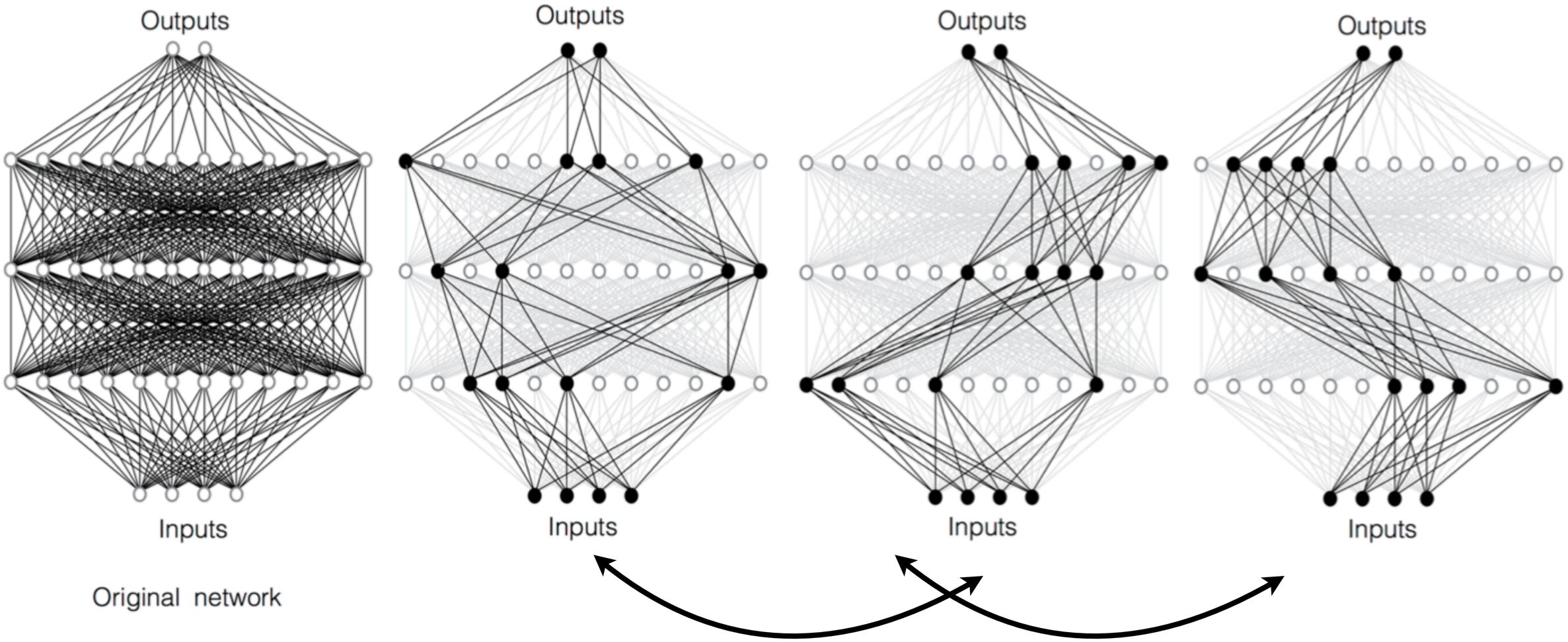
(note: this will change later)

Independent Subnet Training: NN decomposition



Original network

Independent Subnet Training: NN decomposition

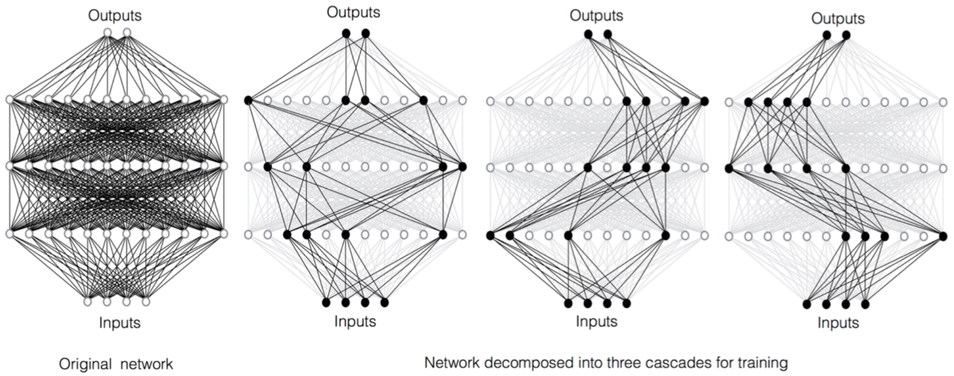


Union of neurons make original network

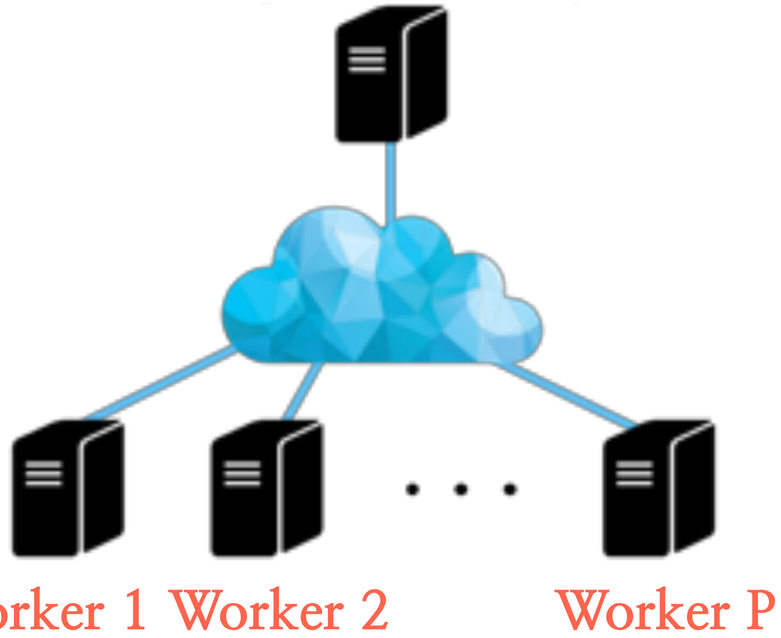
(Note: union of parameters do not make original network necessarily)

Independent Subnet Training: NN distr. training

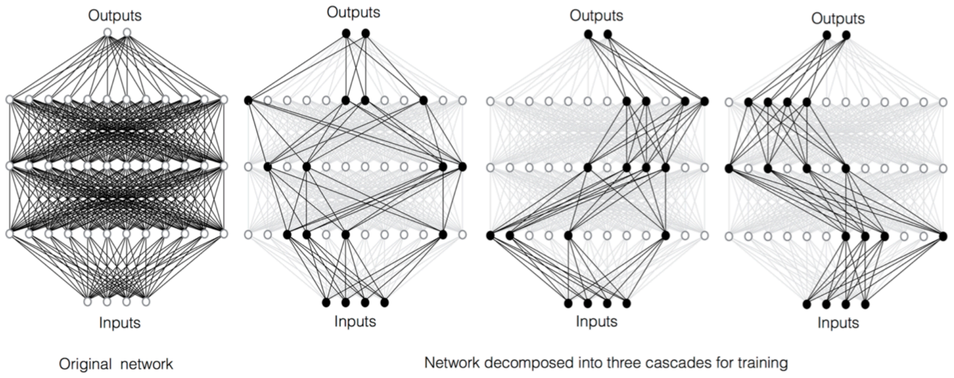
How to decompose a NN:

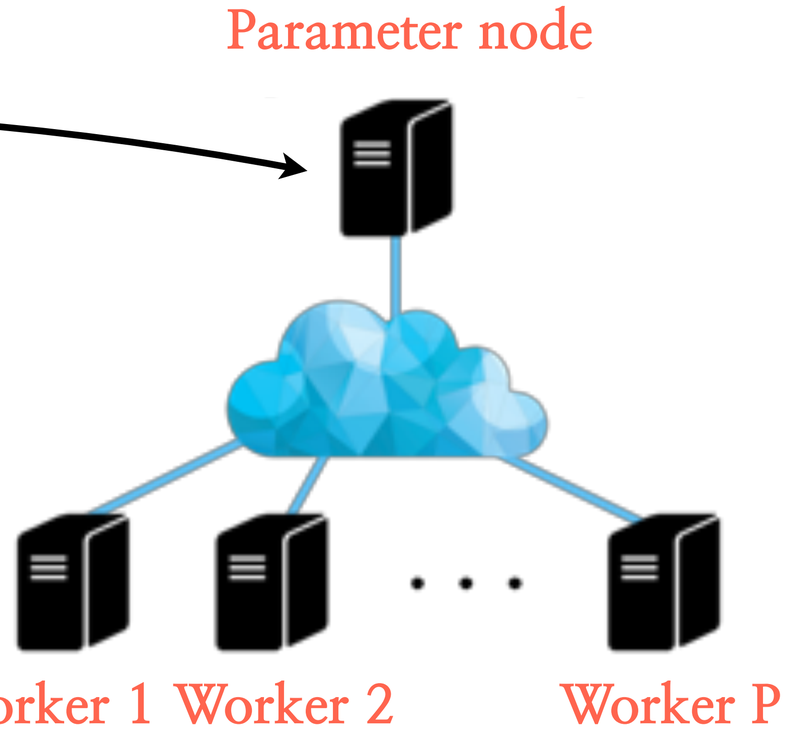


How to train NN in a distributed fashion:

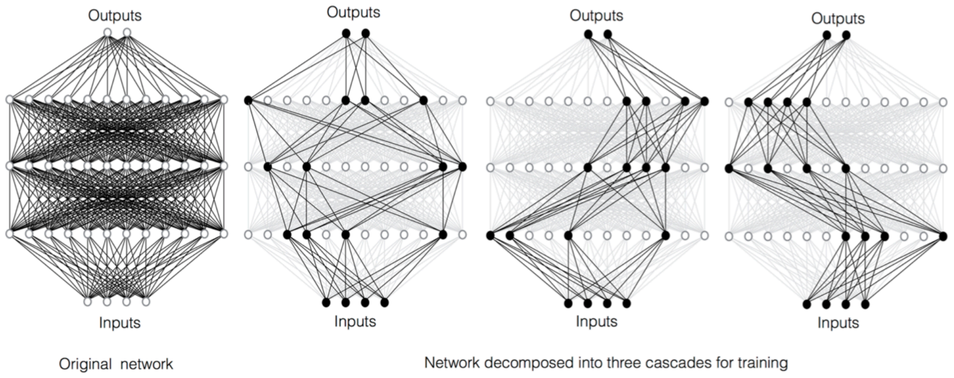


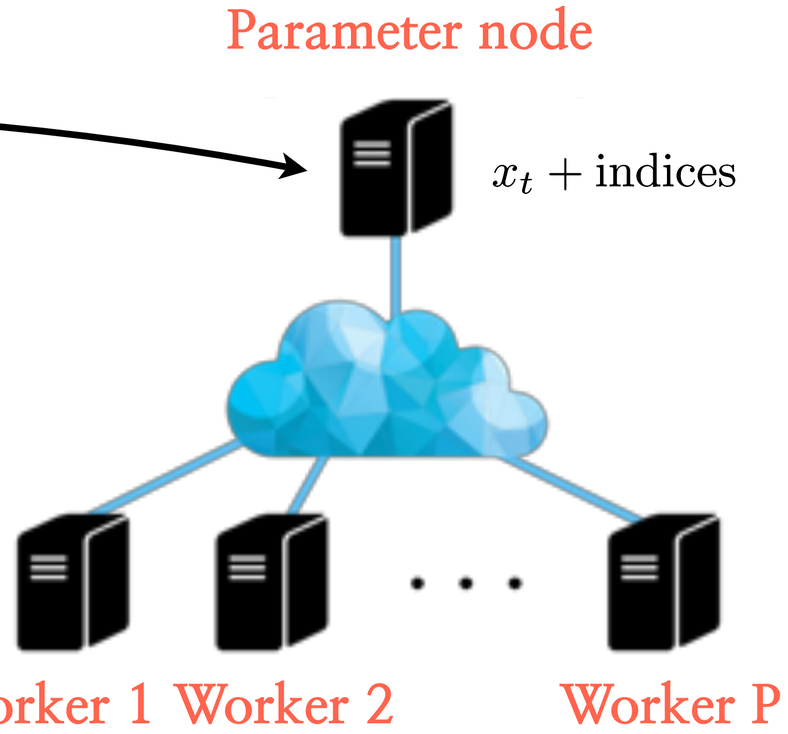
Independent Subnet Training: NN distr. training



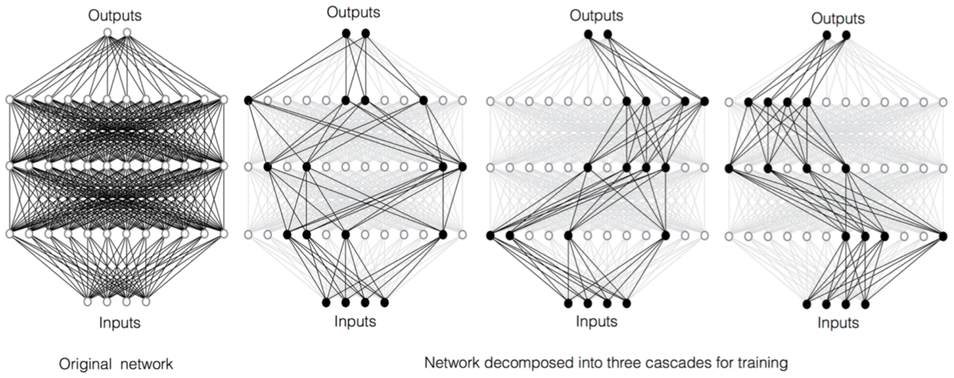


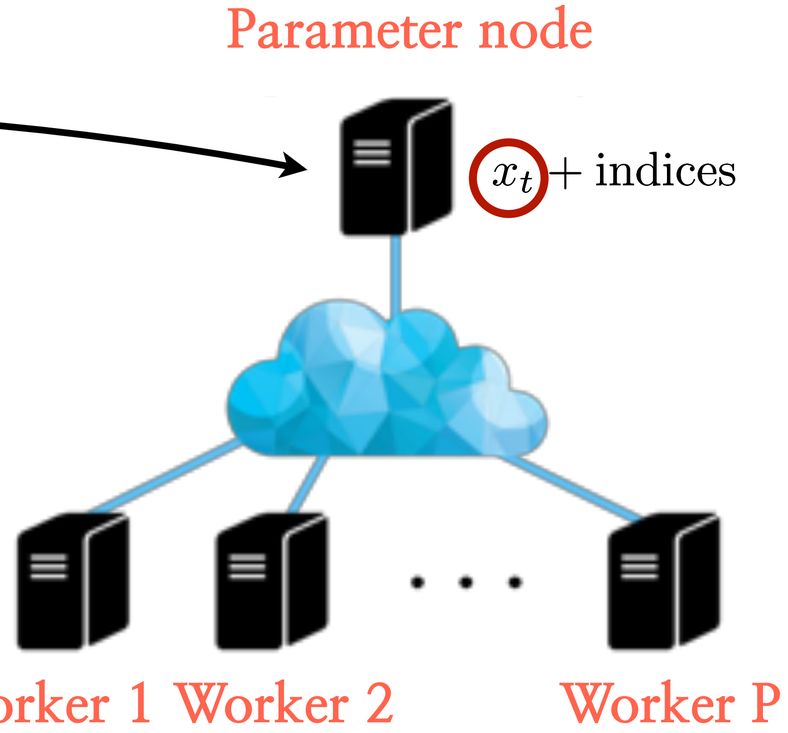
Independent Subnet Training: NN distr. training



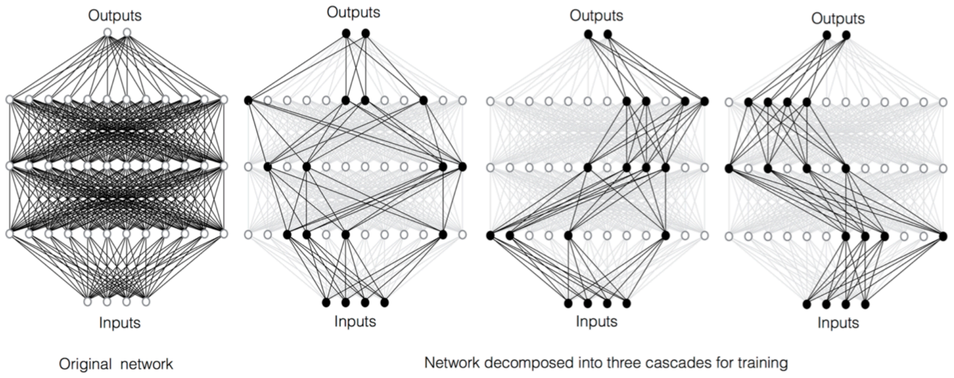


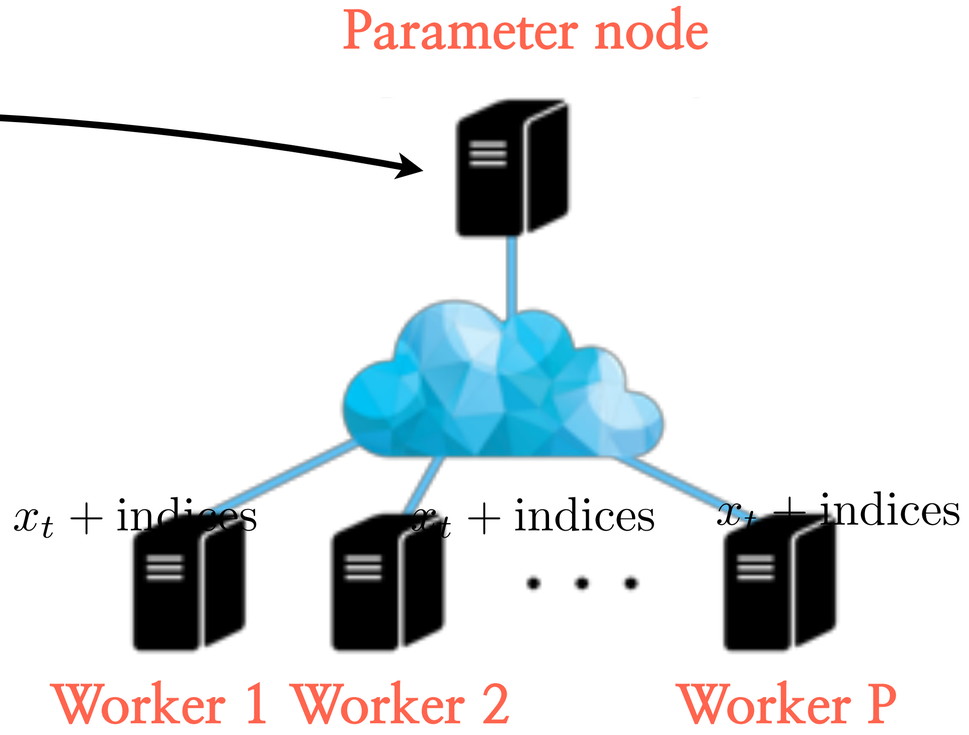
Independent Subnet Training: NN distr. training



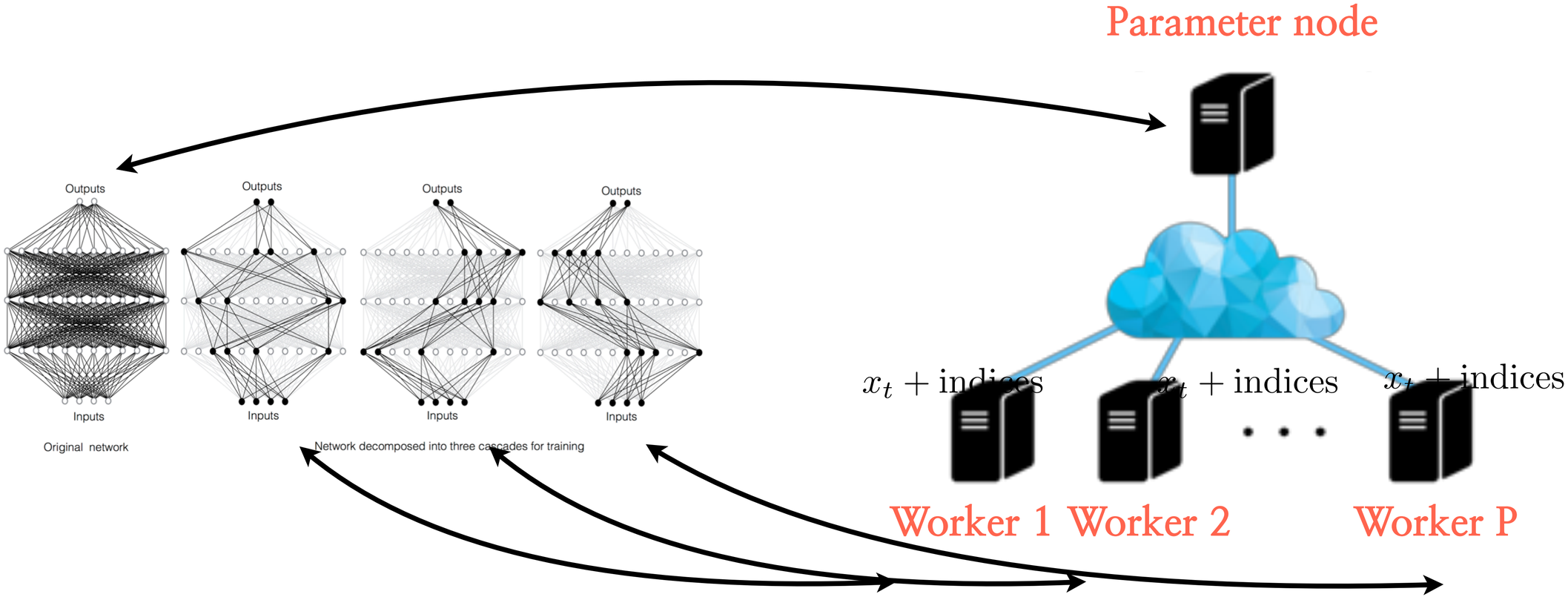


Independent Subnet Training: NN distr. training

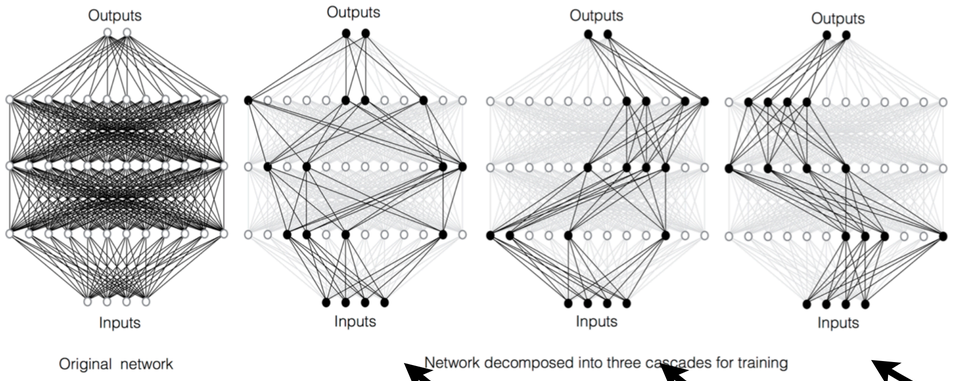




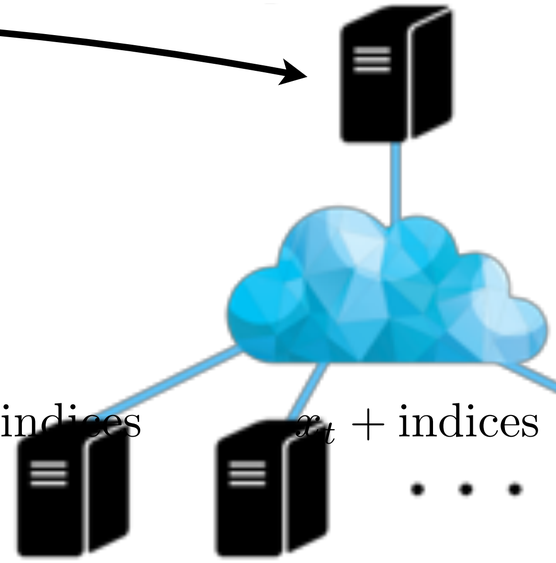
Independent Subnet Training: NN distr. training



Independent Subnet Training: NN distr. training

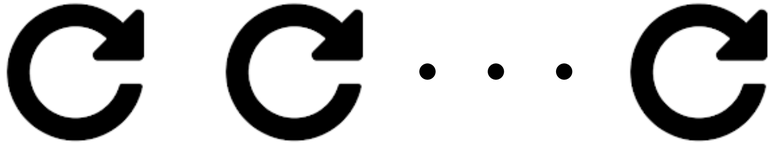


Parameter node



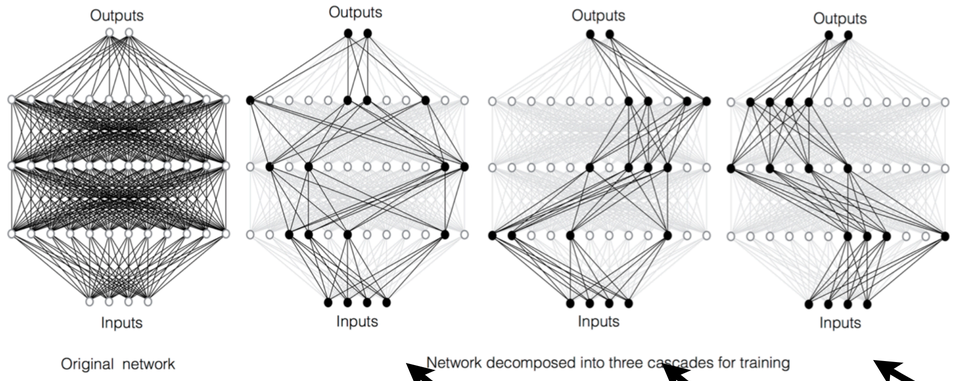
x_t + indices x_t + indices x_t + indices

Worker 1 Worker 2 Worker P



Local SGD updates for several iters

Independent Subnet Training: NN distr. training



One iteration!

Parameter node

x_t + indices

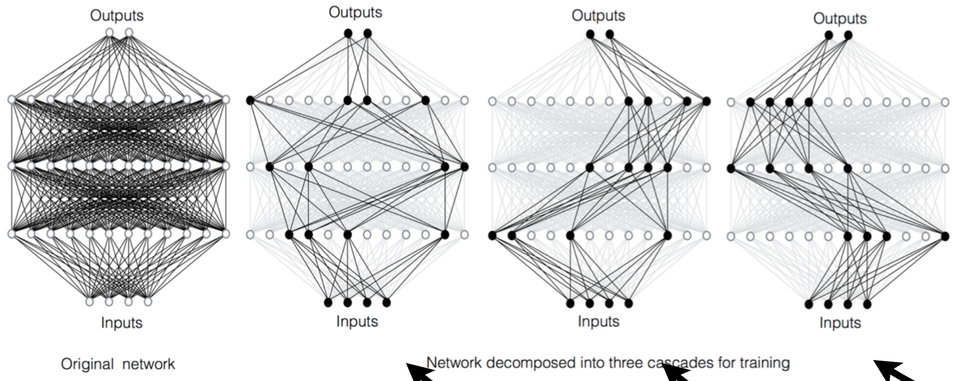
x_t + indices

x_t + indices

Worker 1 Worker 2 Worker P

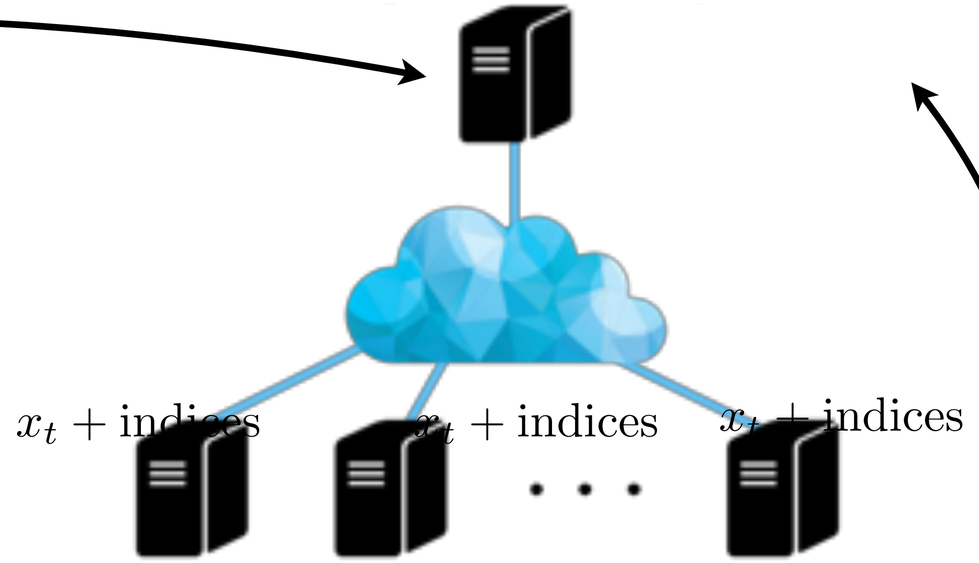
Local SGD updates for several iters

Independent Subnet Training: NN distr. training

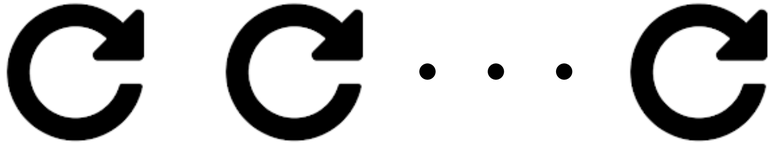


One iteration!

Parameter node



Not till end of execution:
Main difference to ensemble methods
[ParallelSGD, Zinkevich et al., 2010]



Local SGD updates for several iters

Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.
$f(x) \leftarrow$ randomly initialized network.
for $t = 0, \ldots, T-1$ **do**
  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x),\ S)$.
  Distribute each $f_s(x_s)$ to a different worker.
  for $s = 1, \ldots, S$ **do**
    Train $f_s(x_s)$ for ℓ iterations using local SGD.
  end for
  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.
end for

Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.

$f(x) \leftarrow$ randomly initialized network.

for $t = 0, \ldots, T-1$ **do**

  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x),\ S)$.

  Distribute each $f_s(x_s)$ to a different worker.

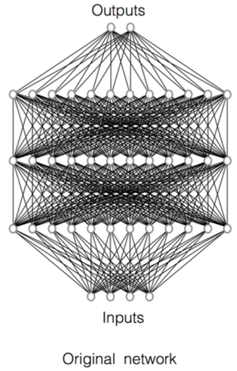
  for $s = 1, \ldots, S$ **do**

    Train $f_s(x_s)$ for ℓ iterations using local SGD.

  end for

  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.

end for



Original network

Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.

$f(x) \leftarrow$ randomly initialized network.

for $t = 0, \ldots, T-1$ **do**

  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x),\ S).$

  Distribute each $f_s(x_s)$ to a different worker.

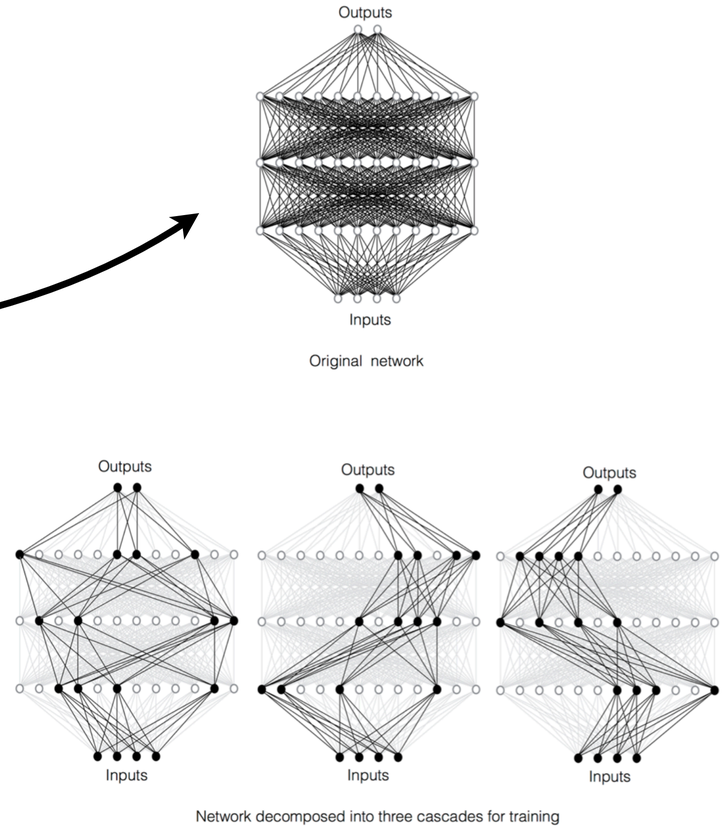
  for $s = 1, \ldots, S$ **do**

    Train $f_s(x_s)$ for ℓ iterations using local SGD.

  end for

  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right).$

end for



Independent Subnet Training: Pseudocode

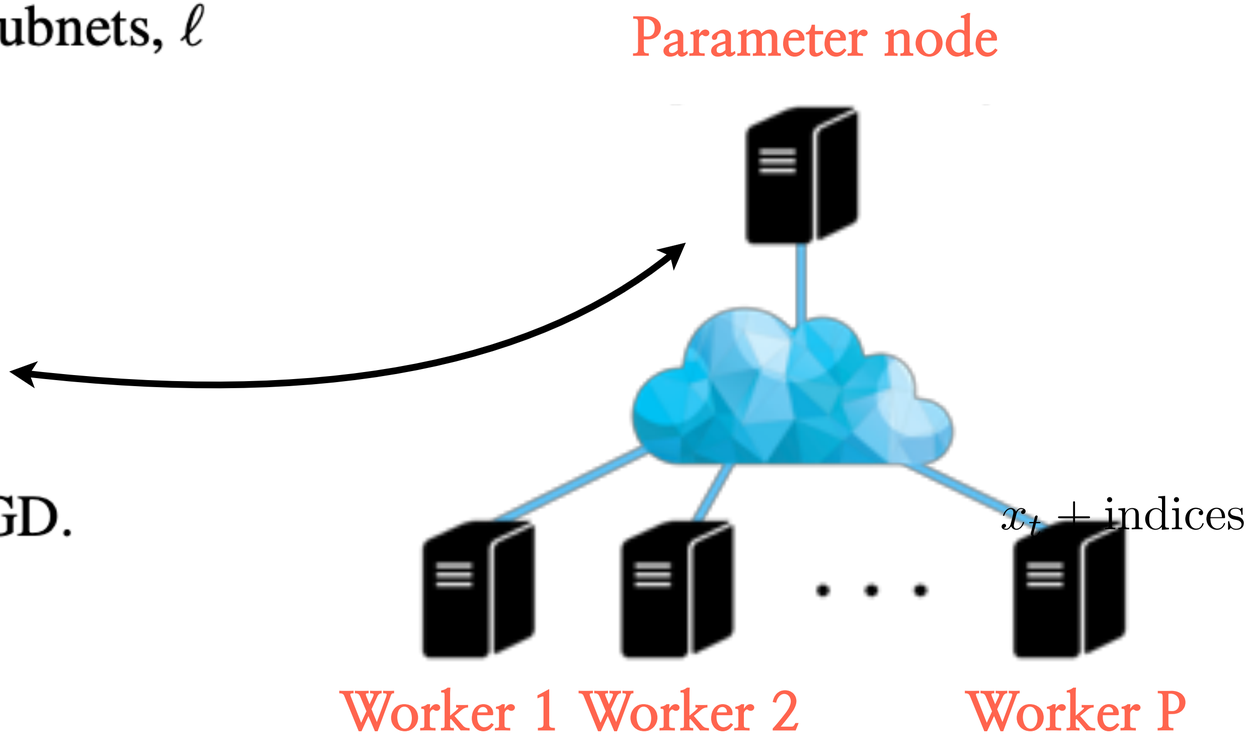
Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.
$f(x) \leftarrow$ randomly initialized network.
for $t = 0, \ldots, T-1$ **do**
  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x),\ S)$.
  Distribute each $f_s(x_s)$ to a different worker.
  for $s = 1, \ldots, S$ **do**
    Train $f_s(x_s)$ for ℓ iterations using local SGD.
  end for
  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.
end for

Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.
$f(x) \leftarrow$ randomly initialized network.
for $t = 0, \ldots, T-1$ **do**
  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x), S)$.
  Distribute each $f_s(x_s)$ to a different worker.
  for $s = 1, \ldots, S$ **do**
    Train $f_s(x_s)$ for ℓ iterations using local SGD.
  end for
  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.
end for



Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.

$f(x) \leftarrow$ randomly initialized network.

for $t = 0, \ldots, T-1$ **do**

  $\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x), S)$.

  Distribute each $f_s(x_s)$ to a different worker.

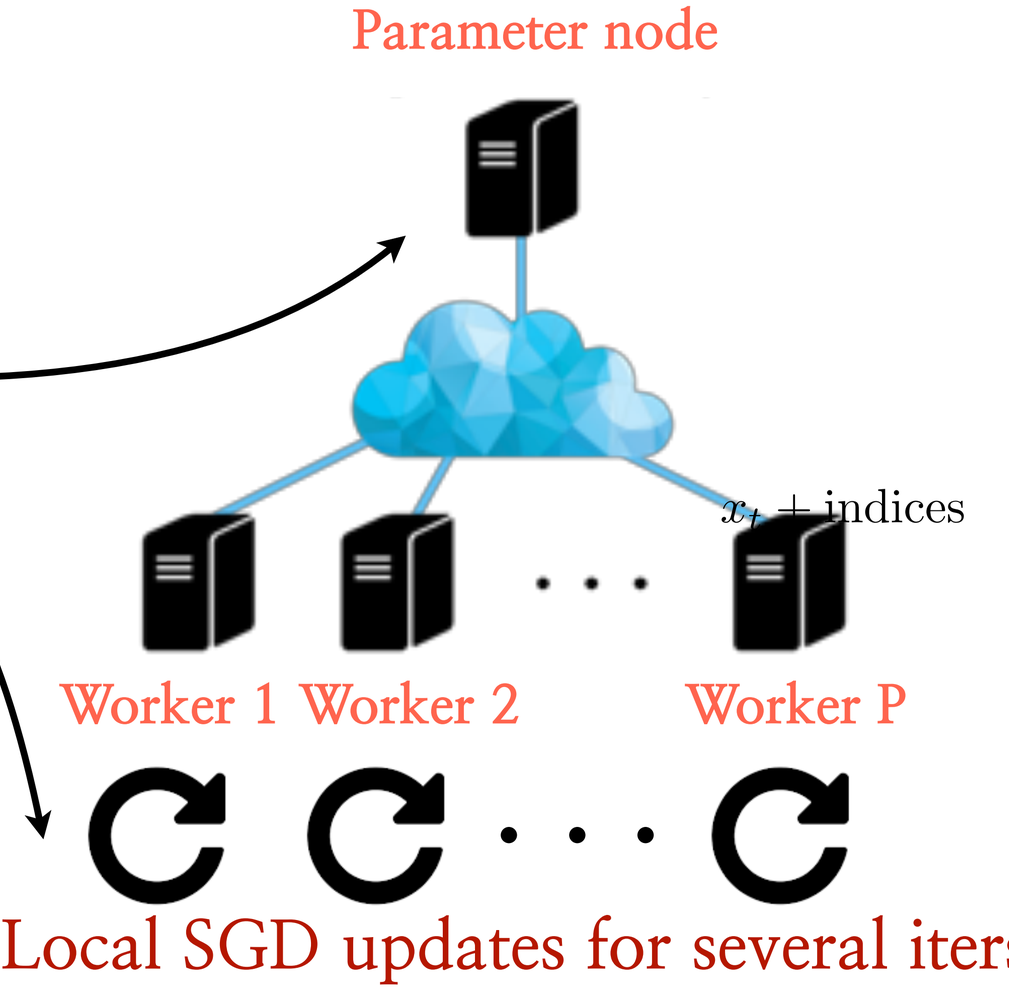
  for $s = 1, \ldots, S$ **do**

    Train $f_s(x_s)$ for ℓ iterations using local SGD.

  end for

  $f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.

end for



Independent Subnet Training: Pseudocode

Algorithm 1 IST Meta Algorithm

Parameters: T synchronization iterations, S subnets, ℓ local iterations, x weights.

$f(x) \leftarrow$ randomly initialized network.

for $t = 0, \ldots, T-1$ **do**

$\{f_s(x_s)\}_{s=1}^{S} = \texttt{subnets}(f(x), S)$.

Distribute each $f_s(x_s)$ to a different worker.

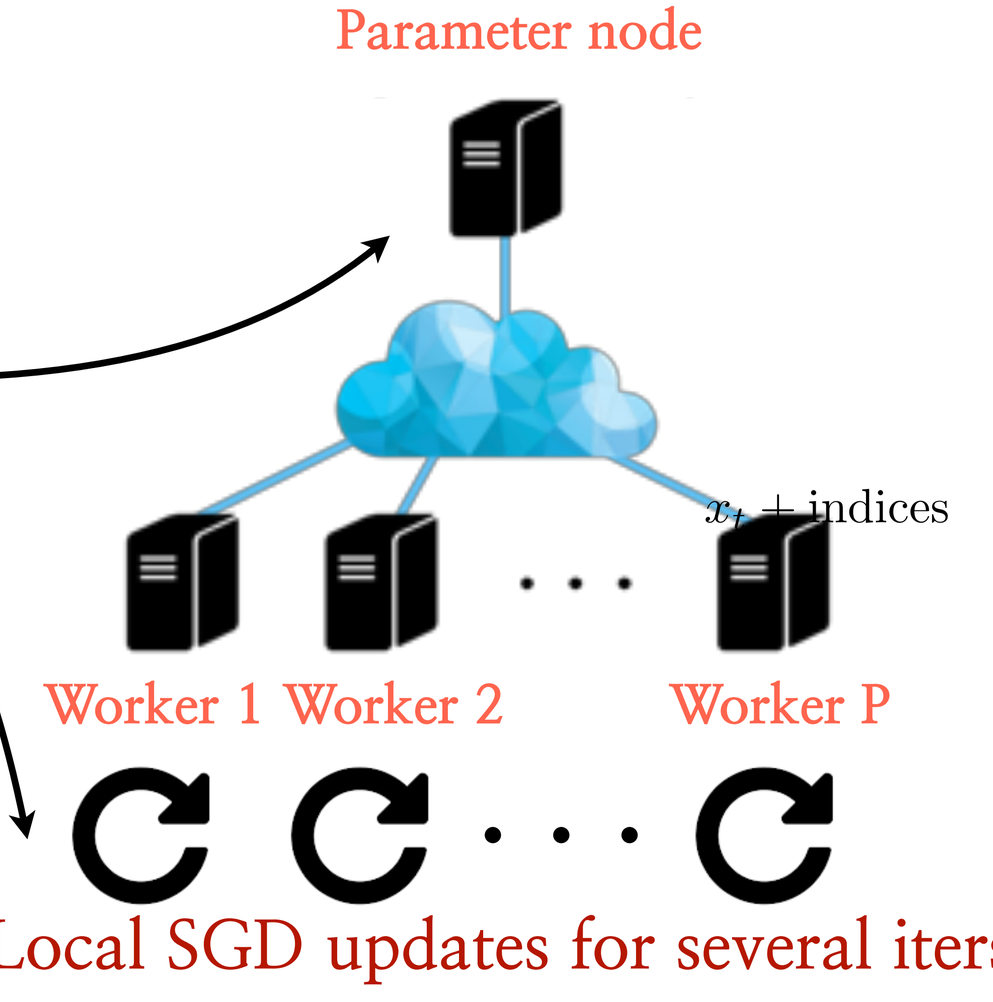
for $s = 1, \ldots, S$ **do**

Train $f_s(x_s)$ for ℓ iterations using local SGD.

end for

$f(x) = \texttt{aggregate}\left(\{f_s(x_s)\}_{s=1}^{S}\right)$.

end for



Is it that simple? No!

- Some of the questions that pop out:

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?
  - Can we use vanilla (local) SGD with common hyperparameters?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?
  - Can we use vanilla (local) SGD with common hyperparameters?
  - Is there a theory that supports this decomposition?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?
  - Can we use vanilla (local) SGD with common hyperparameters?
  - Is there a theory that supports this decomposition?
  - If it applies to other architectures, how do we split the CNN layers?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?
  - Can we use vanilla (local) SGD with common hyperparameters?
  - Is there a theory that supports this decomposition?
  - If it applies to other architectures, how do we split the CNN layers? What about the residual blocks? What if there is graph structure?

Is it that simple? No!

- Some of the questions that pop out:
  - Does it apply only for FC layers?
  - How shall we split the subnets? Can they overlap?
  - How to aggregate the subnets?
  - Can we use vanilla (local) SGD with common hyperparameters?
  - Is there a theory that supports this decomposition?
  - If it applies to other architectures, how do we split the CNN layers? What about the residual blocks? What if there is graph structure? What about Transformers?
  - ..

How does it compare to existing methodologies?

(Note: High level description)

Background on distributed learning

- “Data parallel”: Each compute site updates a complete copy of the neural network’s parameters on different portions of the data.

Background on distributed learning

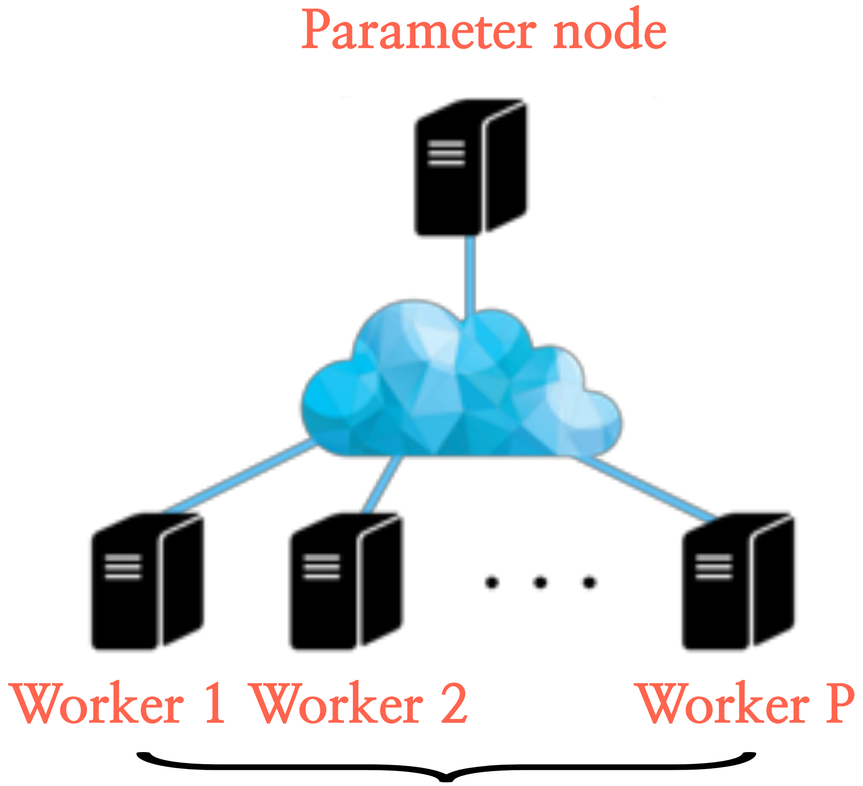
- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

Background on distributed learning

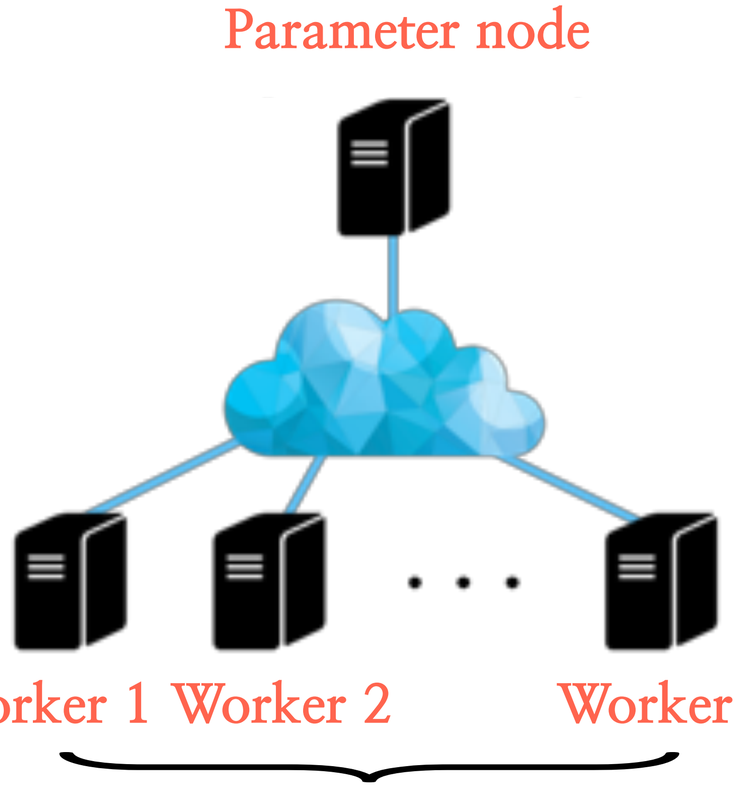
- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

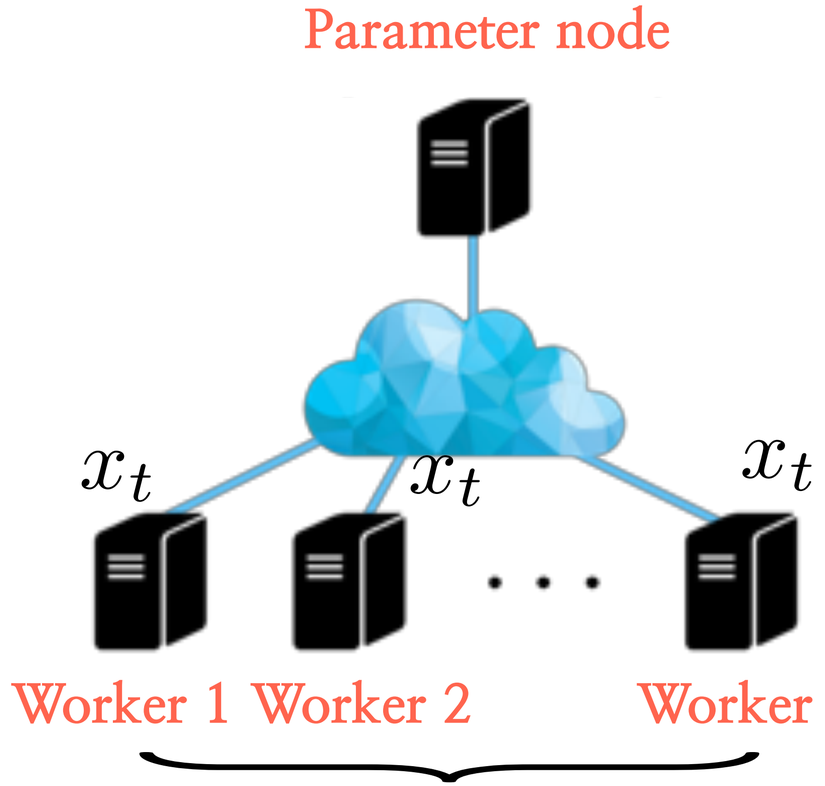


$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

i) Parameter node keeps and distributes model x_t at every cycle/iteration

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

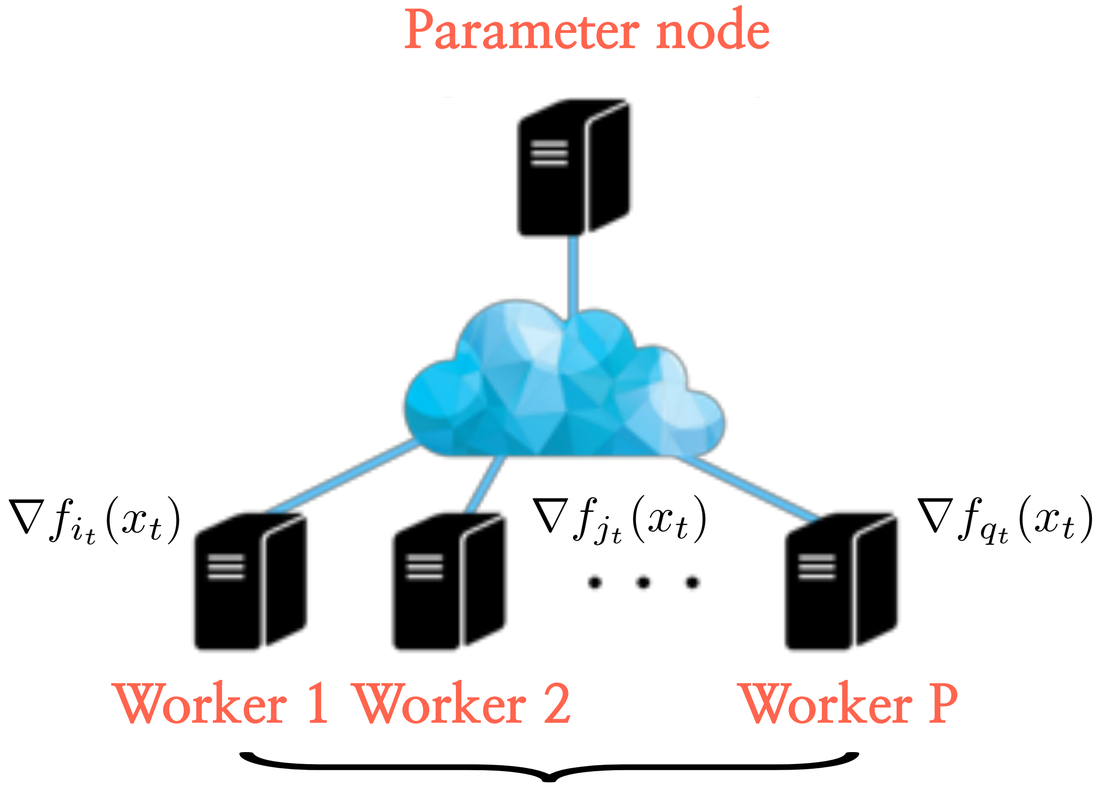


$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

i) Parameter node keeps and distributes model x_t at every cycle/iteration

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.



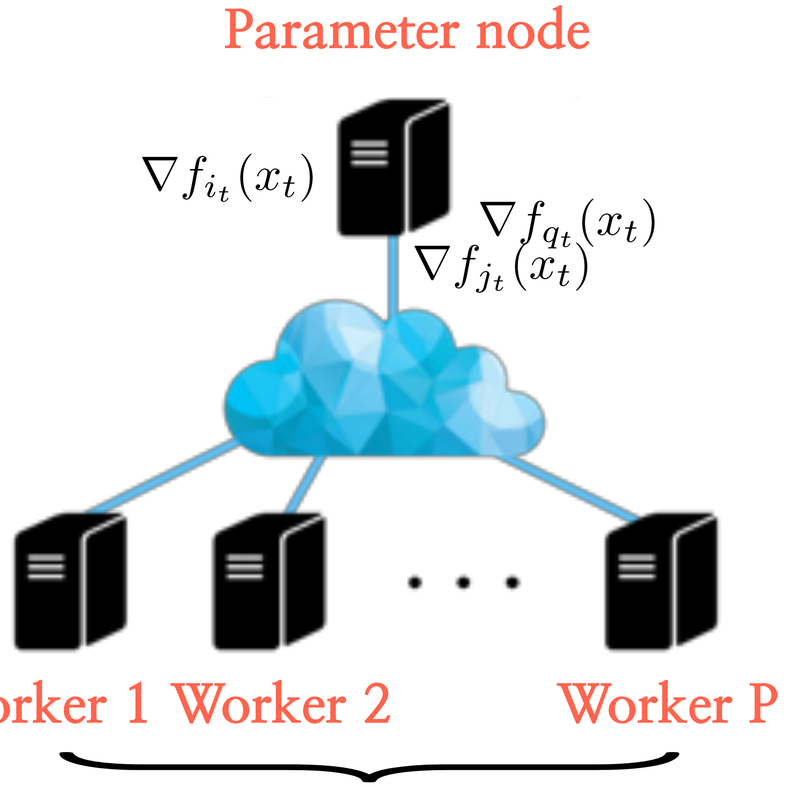
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

i) Parameter node keeps and distributes model x_t at every cycle/iteration

ii) Worker nodes compute part of the full gradient, based on the part of data they have

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.



$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

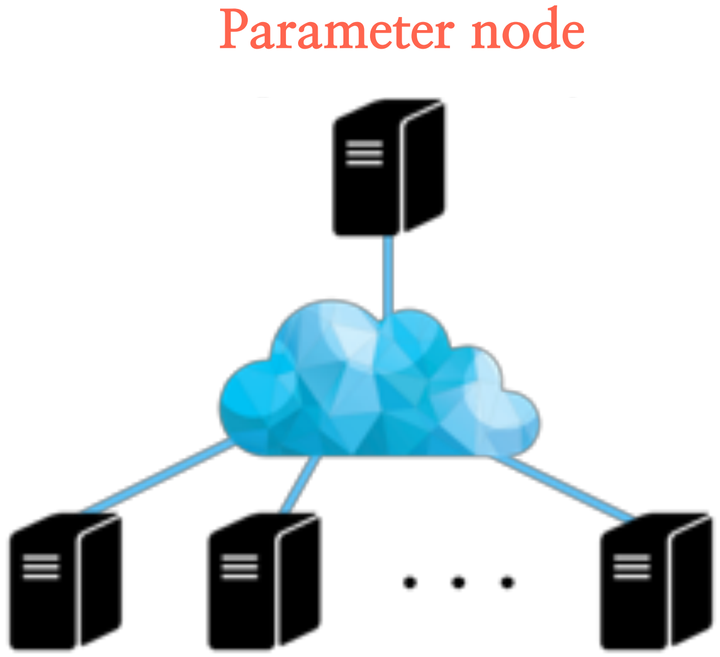
i) Parameter node keeps and distributes model x_t at every cycle/iteration

ii) Worker nodes compute part of the full gradient, based on the part of data they have

iii) Parameter node waits for **all gradient parts** to be collected to do the gradient step

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.



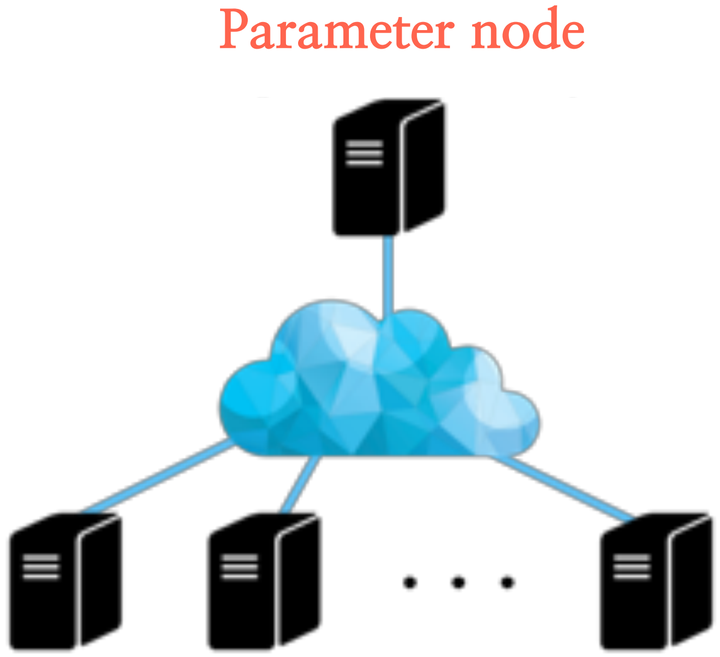
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature
  - Synchronous vs. Asynchronous

    [Large scale distributed DNNs, Dean et al., 2012, Revisiting distributed synchronous SGD, Chen et al., 2016, ..]

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

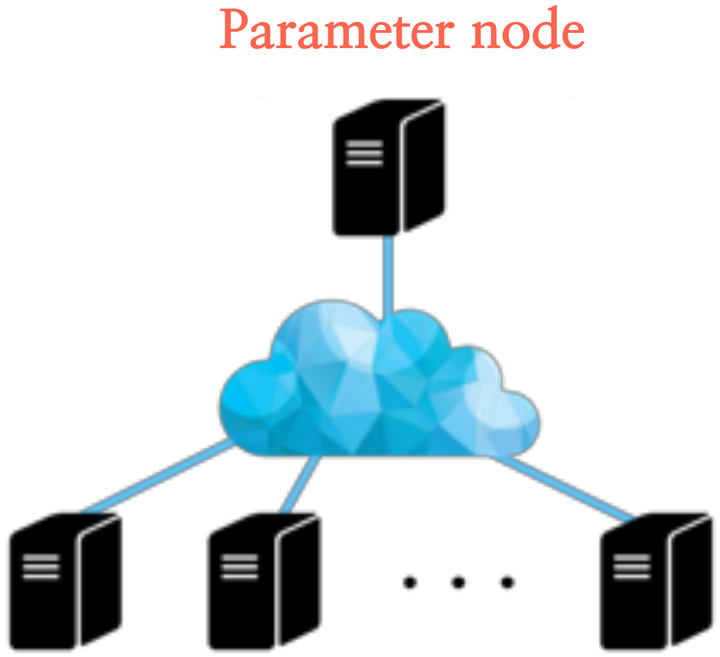


$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature
  - Synchronous vs. Asynchronous
    [Large scale distributed DNNs, Dean et al., 2012, Revisiting distributed synchronous SGD, Chen et al., 2016, ..]
  - Straggler mitigation
    [Large scale distributed DNNs, Dean et al., 2012, Gradient coding, Tandon et al., 2016, ..]

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.



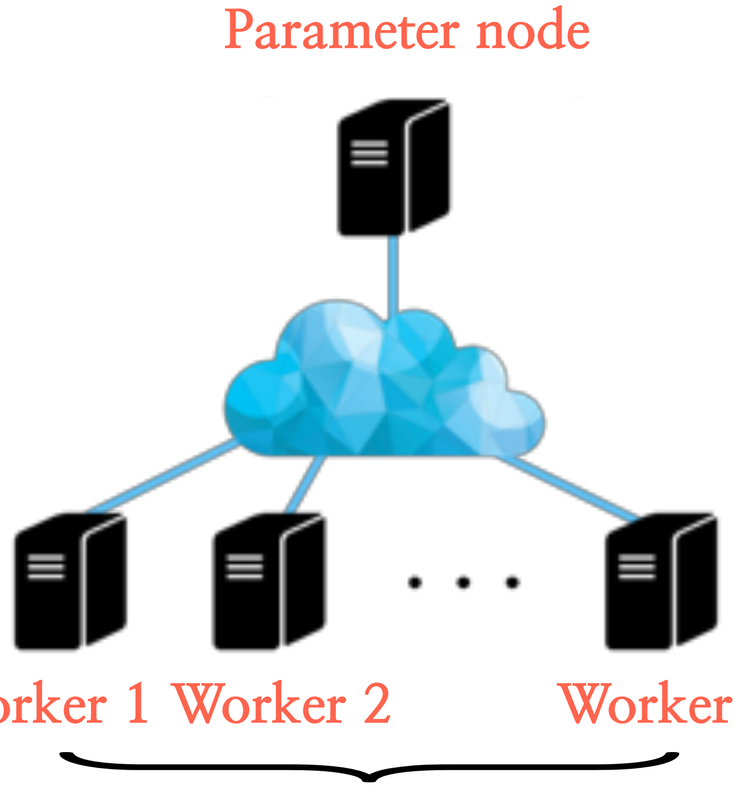
Each contains distinct partition of data

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature
  - Synchronous vs. Asynchronous
    [Large scale distributed DNNs, Dean et al., 2012,
    Revisiting distributed synchronous SGD, Chen et al., 2016, ..]
  - Straggler mitigation
    [Large scale distributed DNNs, Dean et al., 2012,
    Gradient coding, Tandon et al., 2016, ..]
  - Local SGD vs. Mini-batch
    [Local SGD converges fast and communicates little, Stich, 2019,
    Is local SGD better than mini batch SGD?, Woodworth et al., 2020, ..]

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.



$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

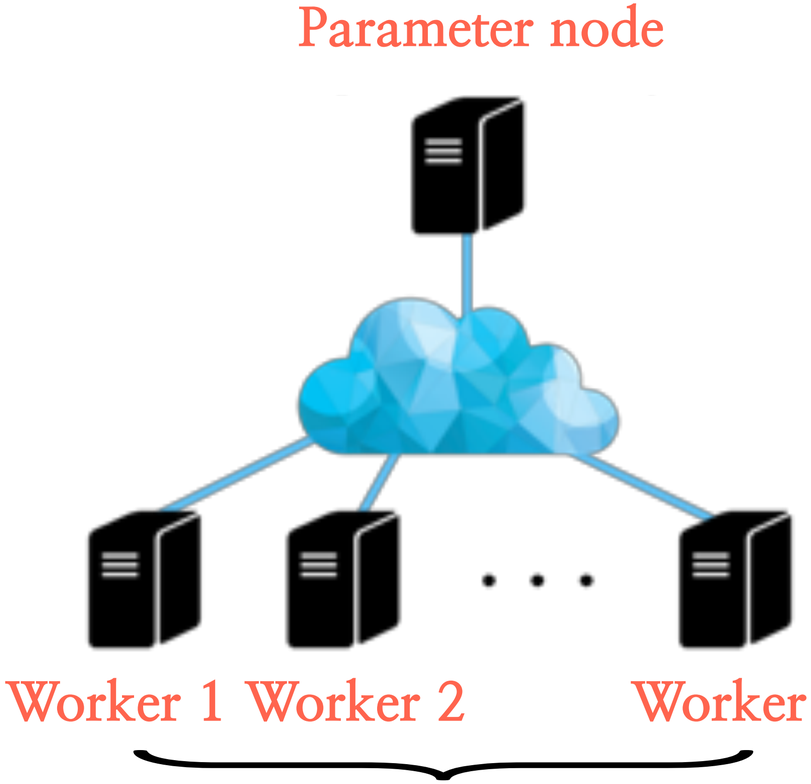
- Extensive Literature

Per iteration, the whole model is communicated and updated

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

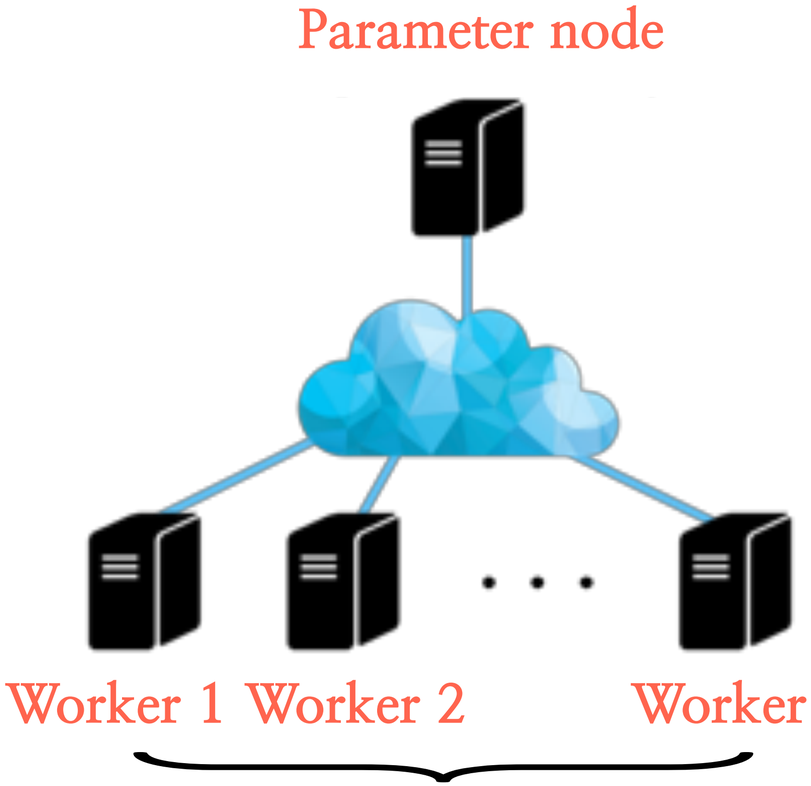


Example

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



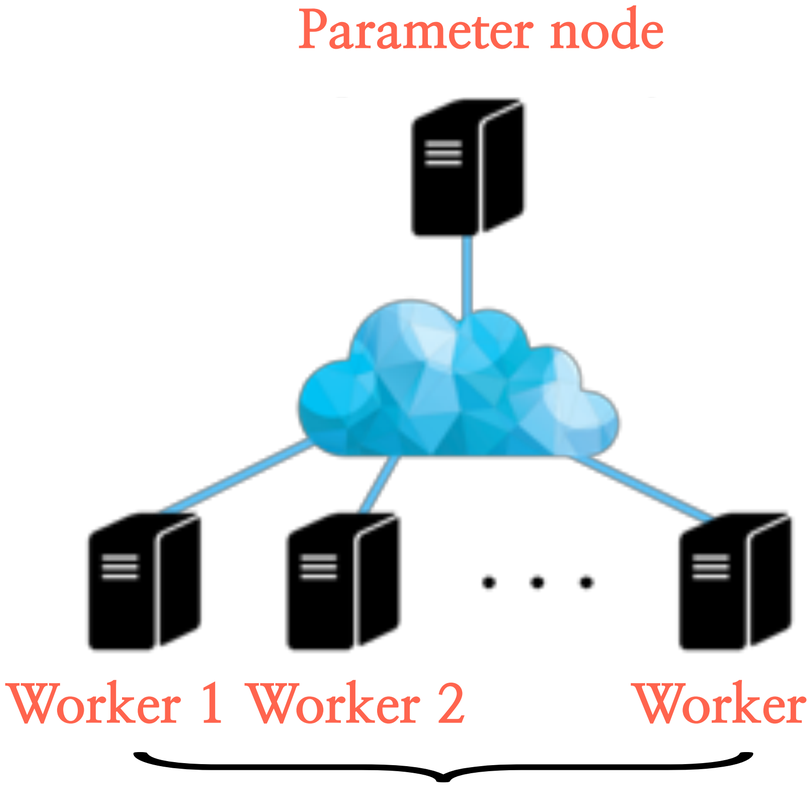
Example

Input dimension: 100,000

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Example

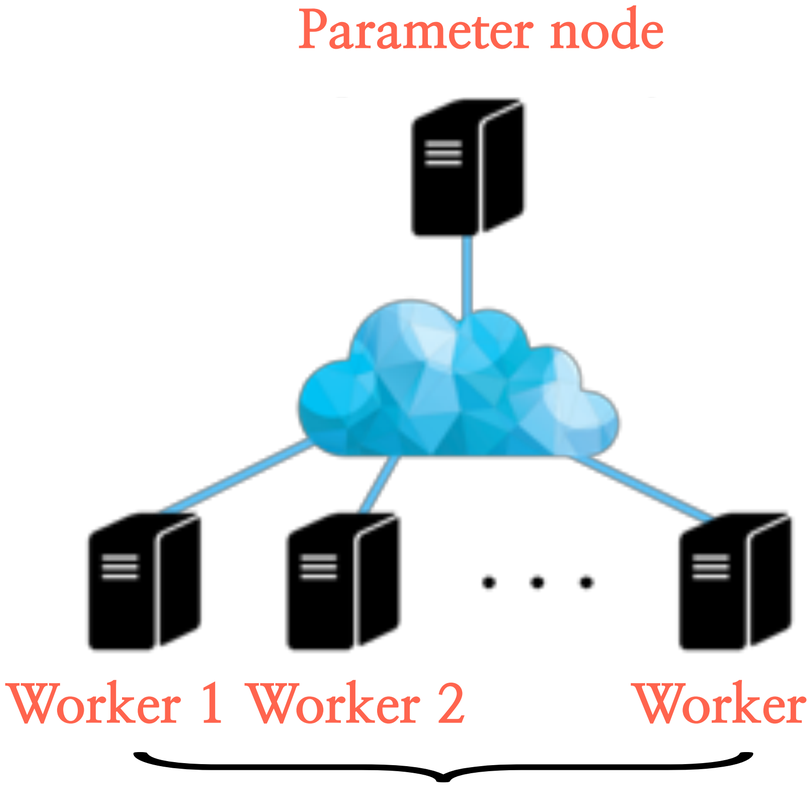
Input dimension: 100,000

Arch. : two hidden layers – 4096 neurons

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Example

Input dimension: 100,000

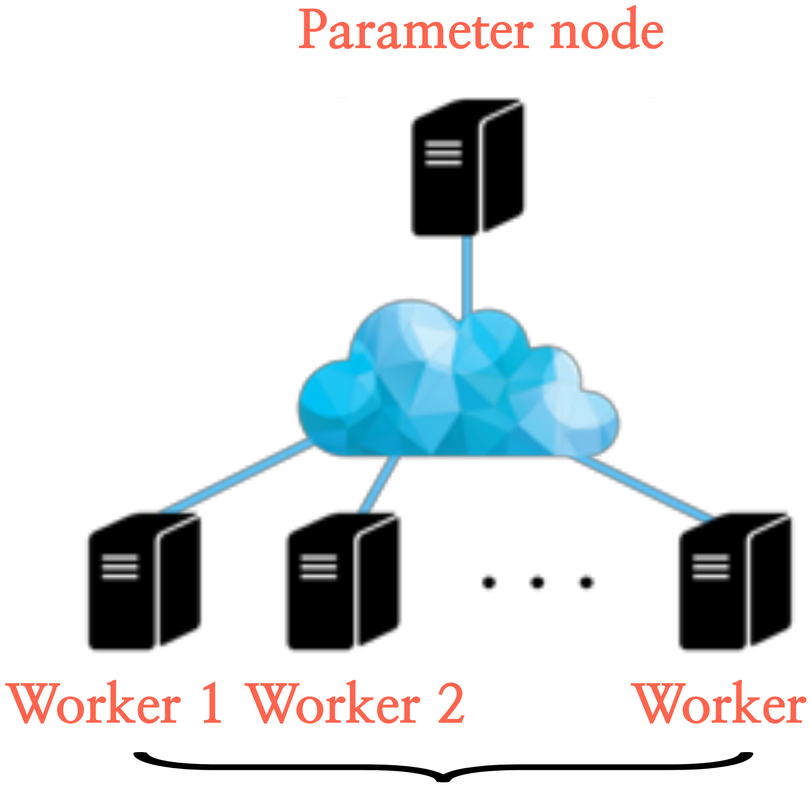
Arch. : two hidden layers – 4096 neurons

System : 64 devices

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Example

Input dimension: 100,000

Arch. : two hidden layers – 4096 neurons

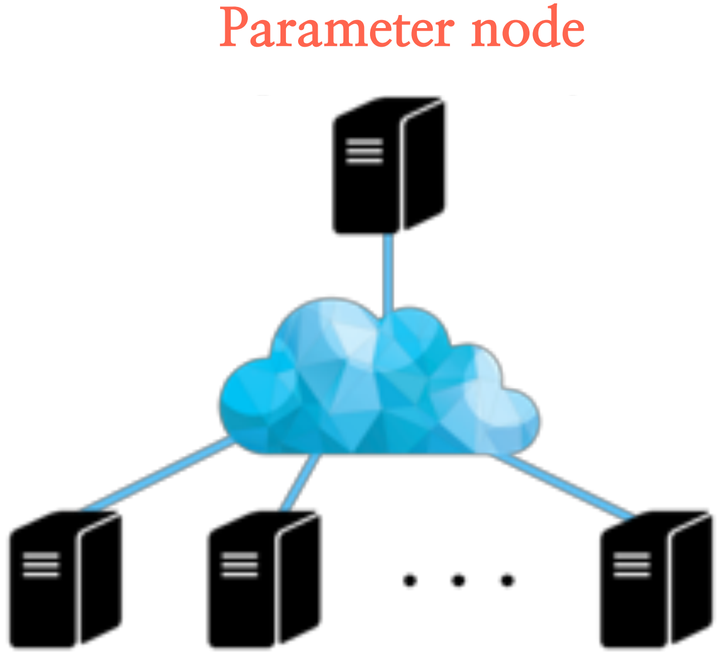
System : 64 devices

Training: 1024 batch size

Background on distributed learning

- "Data parallel": Each compute site updates a complete copy of the neural network's parameters on different portions of the data.

$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Example

Input dimension: 100,000

Arch. : two hidden layers – 4096 neurons

System : 64 devices

Training: 1024 batch size

Communication: 108GB per round!

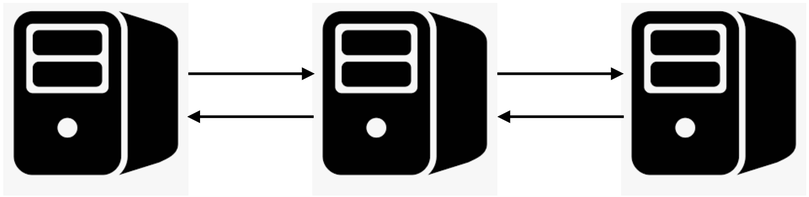
Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.

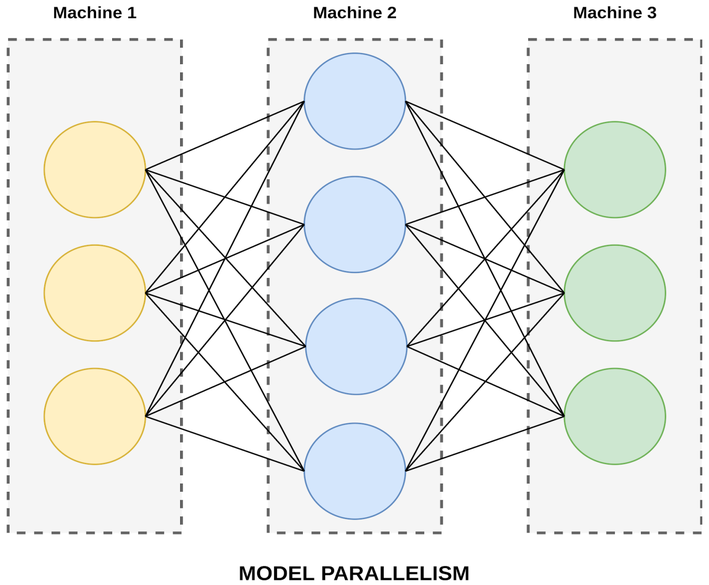
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

Background on distributed learning

- “Model parallel”: Different compute nodes are responsible for different parts of a unique neural network.

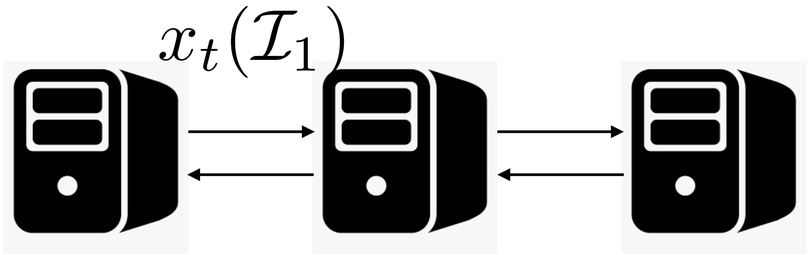


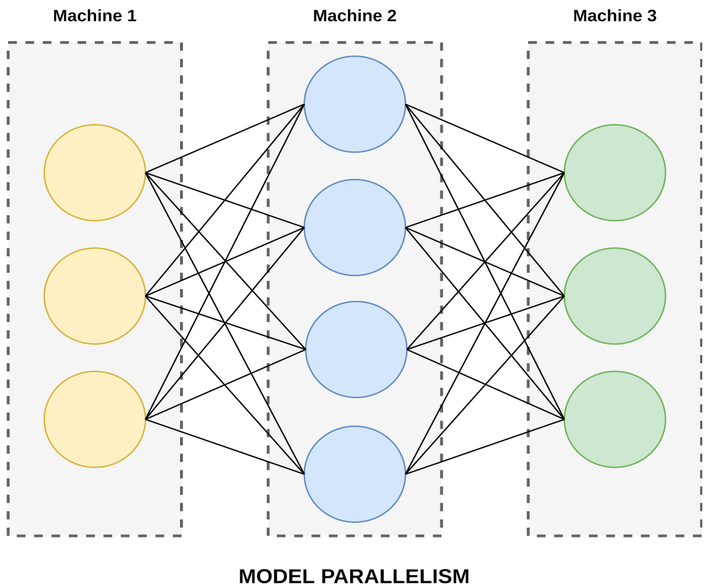
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$



Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.



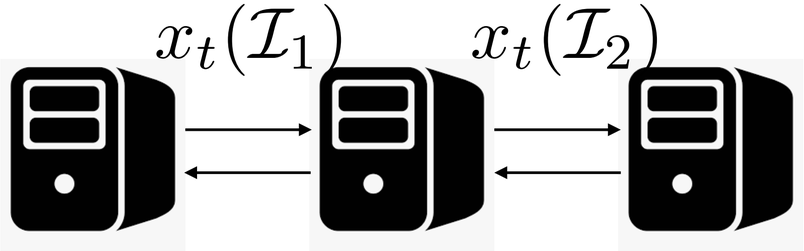


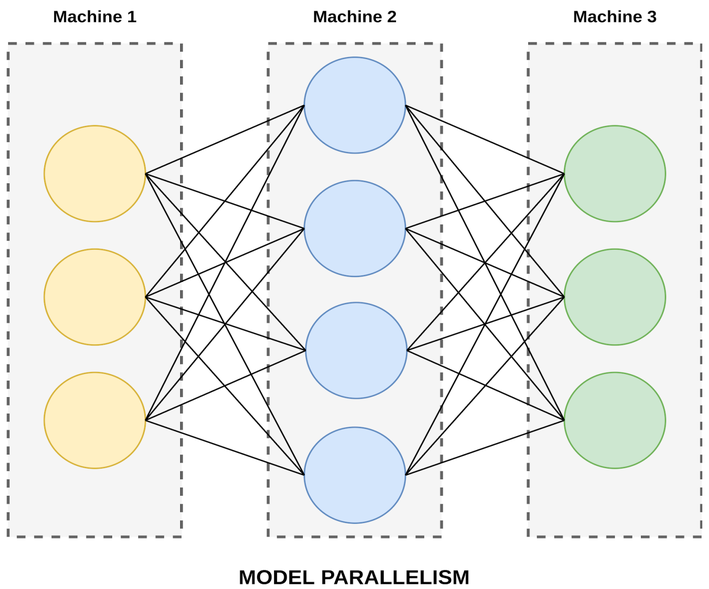
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

$i)$ Each device propagates forward parts of x_t that is responsible for.

Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.



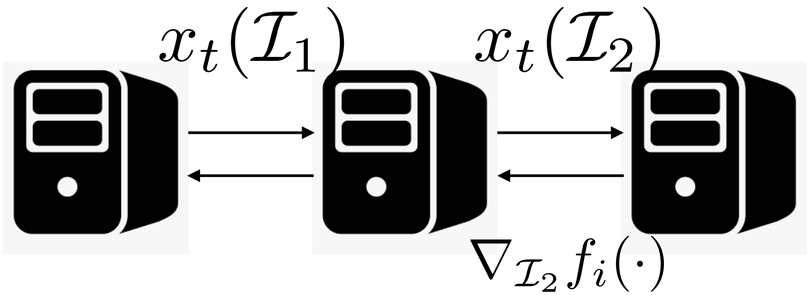


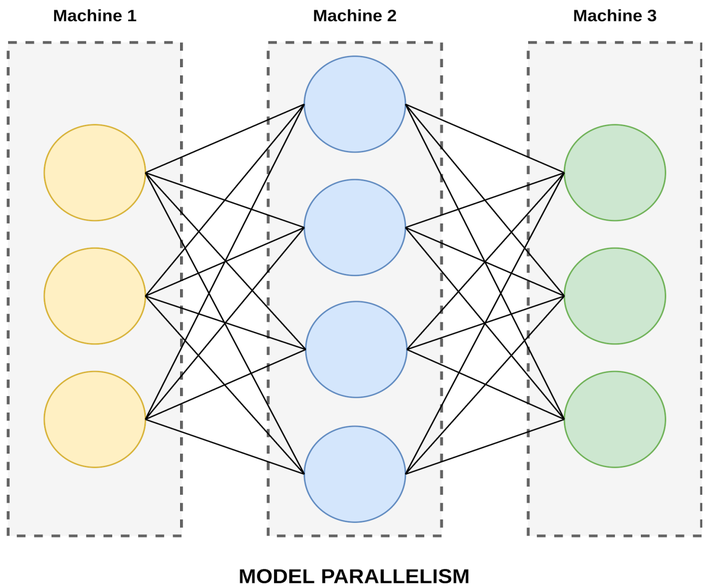
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

$i)$ Each device propagates forward parts of x_t that is responsible for.

Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.





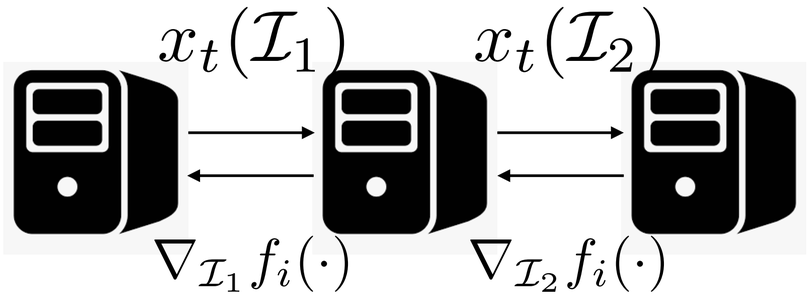
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

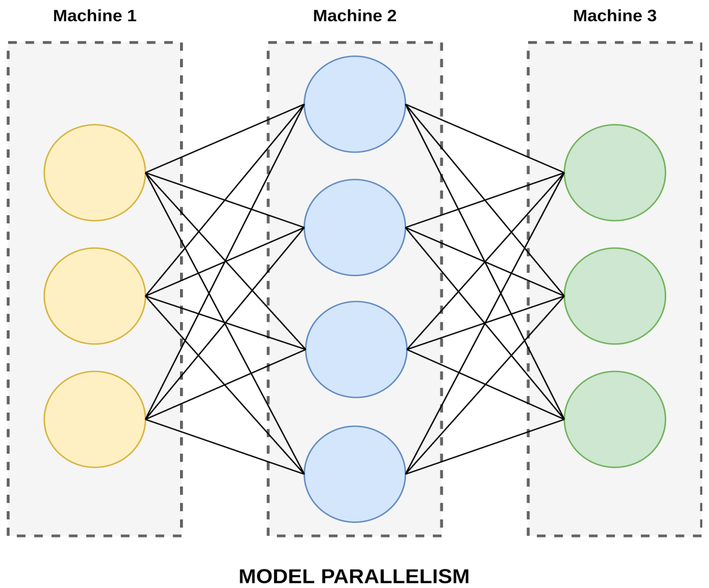
i) Each device propagates forward parts of x_t that is responsible for.

ii) Each device backpropagates parts of the gradient.

Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.





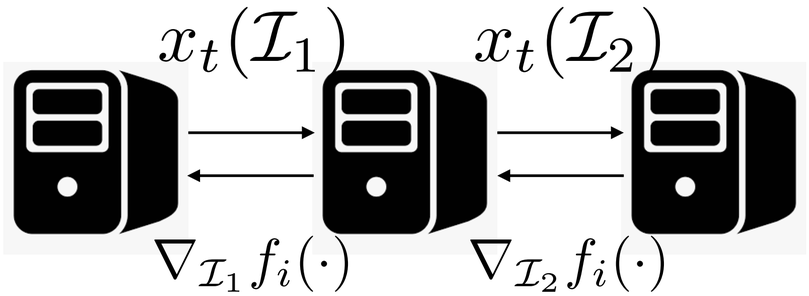
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

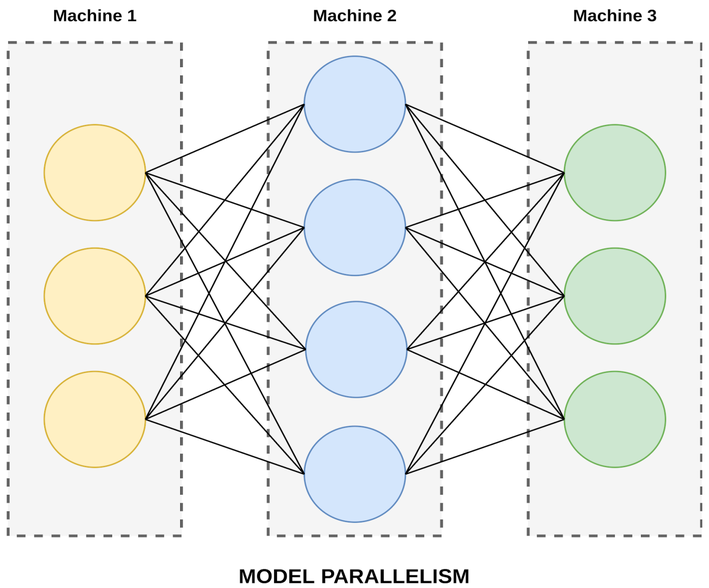
$i)$ Each device propagates forward parts of x_t that is responsible for.

$ii)$ Each device backpropagates parts of the gradient.

Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.



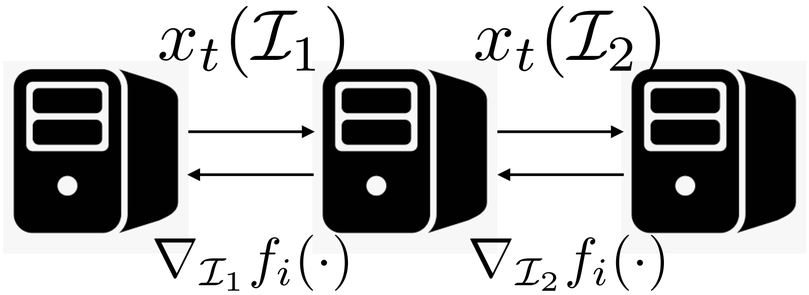


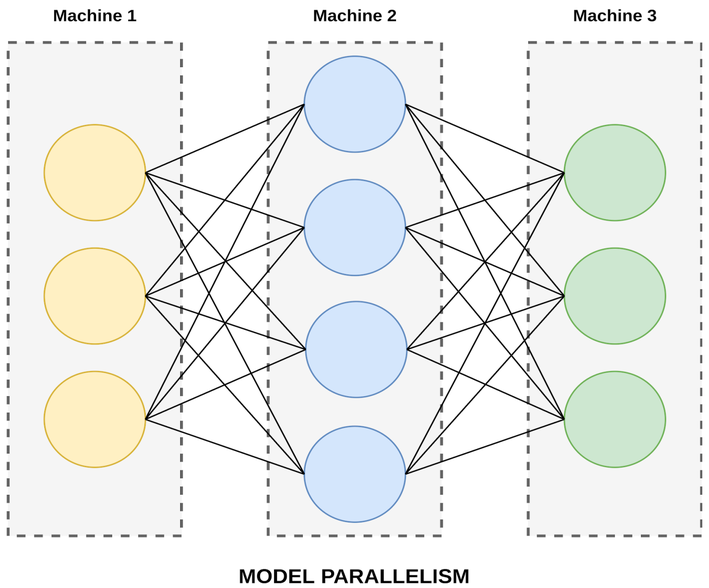
$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature
  - Pipeline Parallel
    [GPipe, Huang et al., 2018,
    The rest of the GPipe family, ..]

Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.



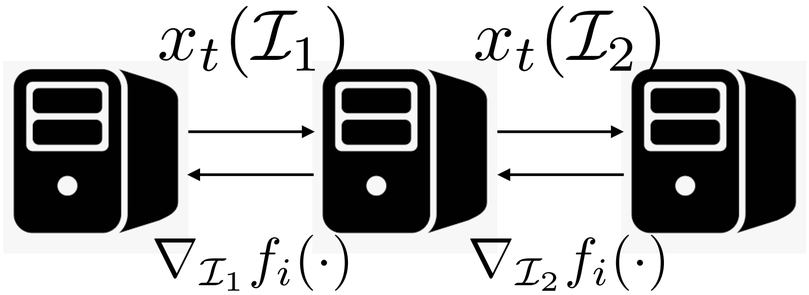


$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature
  - Pipeline Parallel
    [GPipe, Huang et al., 2018,
    The rest of the GPipe family, ..]
  - One step further: ZeRO parallelism
    [ZeRO, Rajbhandari et al., 2019; DeepSpeed, Rasley et al., 2020,
    ZeRO-Offload, Ren et al, 2021; ZeRO-inifity, Rajbhandari et al., 2021]

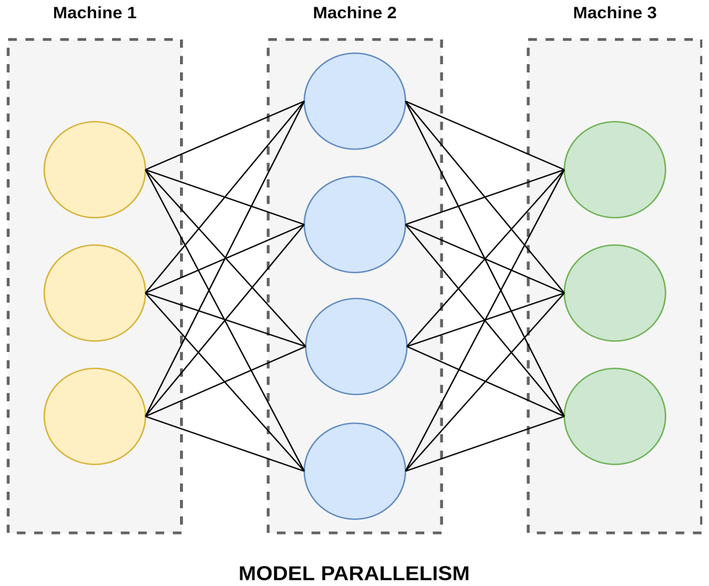
Background on distributed learning

- "Model parallel": Different compute nodes are responsible for different parts of a unique neural network.



$$x_{t+1} = x_t - \eta \sum_{i=1}^{n} \nabla f_i(x_t)$$

- Extensive Literature



For model parallel, data is shared/not private; there are cases where distributed computing is not a choice (Cross SILO FL, cross-device FL)

Independent Subnet Training: In practice

(Note: FC layers for now)

Independent Subnet Training: In practice

- Google Speech Commands: 2-/3- layer NN
  - 4096 and 8192 neurons per layer, respectively.
  - Predict 35 labeled keywords from a 4096 feature vectors.
  - Setup: 2, 4, 8 m5.2xlarge CPU cluster.

Independent Subnet Training: In practice

- Google Speech Commands: 2-/3- layer NN
  - 4096 and 8192 neurons per layer, respectively.
  - Predict 35 labeled keywords from a 4096 feature vectors.
  - Setup: 2, 4, 8 m5.2xlarge CPU cluster.

- VGG12 on CIFAR100 and full ImageNet:
  - Full ImageNet includes 14,197,122 images belongs to 21,841 categories.
  - Setup: 2, 4, 8 p3.2xlarge GPU cluster.

Independent Subnet Training: In practice

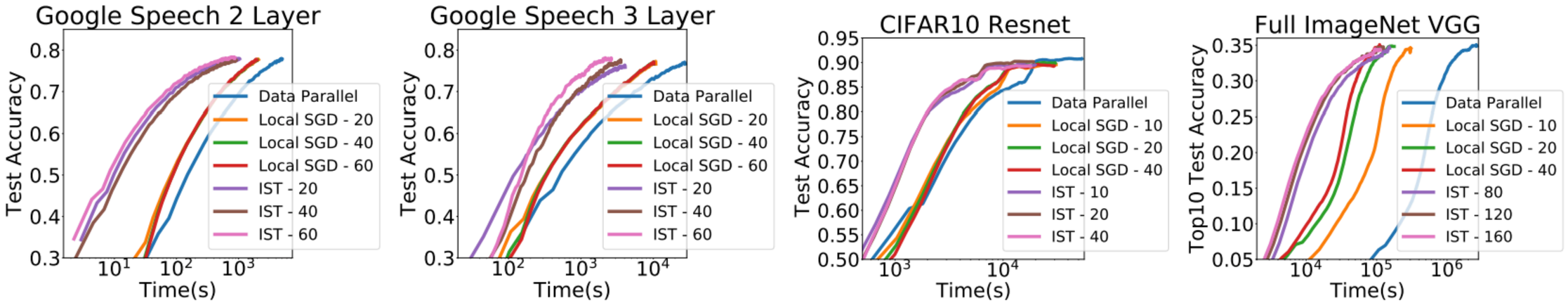
- Google Speech Commands: 2-/3- layer NN
  - 4096 and 8192 neurons per layer, respectively.
  - Predict 35 labeled keywords from a 4096 feature vectors.
  - Setup: 2, 4, 8 m5.2xlarge CPU cluster.

- VGG12 on CIFAR100 and full ImageNet:
  - Full ImageNet includes 14,197,122 images belongs to 21,841 categories.
  - Setup: 2, 4, 8 p3.2xlarge GPU cluster.

- Amazon-670k XML: 2-layer NN
  - Feature dimension: 135,909, label dimension: 670,091.
  - Setup: 8 p3.2xlarge GPU cluster.

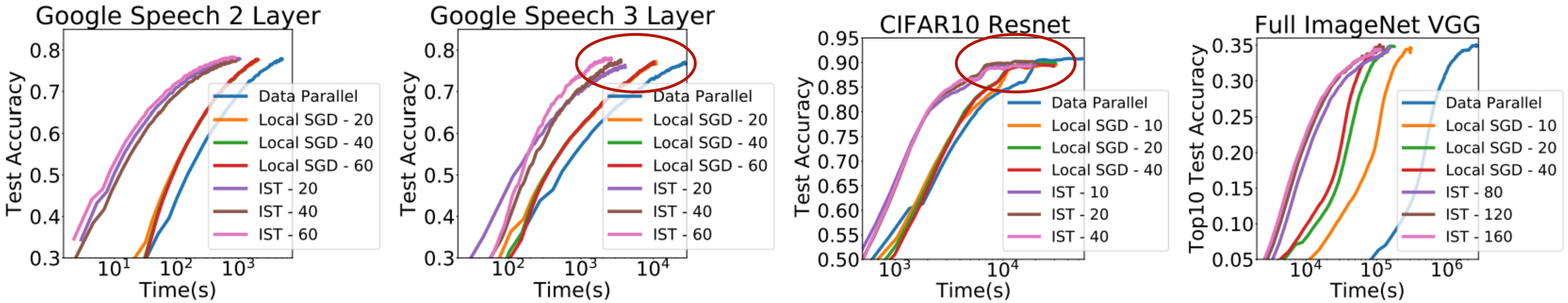
Independent Subnet Training: Scalability

Independent Subnet Training: Scalability



2-/3- layer on 8 GPUs; ResNet18 on 4 GPUs; VGG12 on 8 GPUs

Independent Subnet Training: Scalability



2-/3- layer on 8 GPUs; ResNet18 on 4 GPUs; VGG12 on 8 GPUs

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

Google Speech 2 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	118	269	450	68	130	235	35	28	**24**
0.75	759	1708	2417	444	742	1110	231	**167**	192

Google Speech 3 Layer									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.63	376	1228	1922	182	586	1115	**76**	141	300
0.75	4534	9340	14886	2032	4107	6539	812	**664**	1161

CIFAR10 Resnet18									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.85	21775	13689	6890	18769	12744	7020	15093	7852	**5241**
0.90	54002	38430	17853	36891	22198	**12157**	33345	16798	13425

Full ImageNet VGG12									
	Data Parallel			Local SGD			IST		
Accuracy	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node	2 Node	4 Node	8 Node
0.20	108040	278542	504805	6900	14698	30441	**3629**	4379	5954
0.26	225911	393279	637188	15053	22055	39439	**6189**	7711	10622

Table 1: The time (in seconds) to reach various levels of accuracy.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD
Speech 2 layer	0.7938	0.7998
Speech 3 layer	0.7950	0.7992
CIFAR100 VGG	0.5787	0.5878
Full Imagenet VGG	0.3688	0.3685

Table 3: Final accuracy on each benchmark.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

	Data Parallel			IST		
Dim	P@1	P@3	P@5	P@1	P@3	P@5
512	0.3861	0.3454	0.3164	0.3962	0.3604	0.3313

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

	Data Parallel			IST		
Dim	P@1	P@3	P@5	P@1	P@3	P@5
512	0.3861	0.3454	0.3164	0.3962	0.3604	0.3313
1024	Fail			0.4089	0.3685	0.3392
1536	Fail			0.4320	0.3907	0.3614
2048	Fail			0.4365	0.3936	0.3637
2560	Fail			**0.4384**	**0.3944**	**0.3658**

Table 2: Precision @1, @3, @5 on the Amazon 670k benchmark.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

	Data Parallel			IST		
Dim	P@1	P@3	P@5	P@1	P@3	P@5
512	0.3861	0.3454	0.3164	0.3962	0.3604	0.3313
1024	Fail			0.4089	0.3685	0.3392
1536	Fail			0.4320	0.3907	0.3614
2048	Fail			0.4365	0.3936	0.3637
2560	Fail			**0.4384**	**0.3944**	**0.3658**

Table 2: Precision @1, @3, @5 on the Amazon 670k benchmark.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

	Data Parallel			IST		
Dim	P@1	P@3	P@5	P@1	P@3	P@5
512	0.3861	0.3454	0.3164	0.3962	0.3604	0.3313
1024		Fail		0.4089	0.3685	0.3392
1536		Fail		0.4320	0.3907	0.3614
2048		Fail		0.4365	0.3936	0.3637
2560		Fail		**0.4384**	**0.3944**	**0.3658**

Table 2: Precision @1, @3, @5 on the Amazon 670k benchmark.

Independent Subnet Training: Scalability

	Data Parallel	Local SGD	IST
Speech 2 layer	0.7938	0.7998	**0.8153**
Speech 3 layer	0.7950	0.7992	**0.8327**
CIFAR100 VGG	0.5787	0.5878	**0.6228**
Full Imagenet VGG	0.3688	0.3685	**0.3802**

Table 3: Final accuracy on each benchmark.

	Data Parallel			IST		
Dim	P@1	P@3	P@5	P@1	P@3	P@5
512	0.3861	0.3454	0.3164	0.3962	0.3604	0.3313
1024		Fail		0.4089	0.3685	0.3392
1536		Fail		0.4320	0.3907	0.3614
2048		Fail		0.4365	0.3936	0.3637
2560		Fail		**0.4384**	**0.3944**	**0.3658**

Table 2: Precision @1, @3, @5 on the Amazon 670k benchmark.

Independent Subnet Training: Some theory

(Note: not quite there yet)

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t$$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t \xrightarrow{\text{Decompose}} \begin{cases} x_t^{(1)} \\ x_t^{(2)} \\ \vdots \\ x_t^{(P)} \end{cases}$$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t \xrightarrow{\text{Decompose}} \begin{cases} x_t^{(1)} \xrightarrow{\text{Send to workers}} x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} \longrightarrow x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots \qquad\qquad\qquad \vdots \\ x_t^{(P)} \longrightarrow x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{cases}$$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t \xrightarrow{\text{Decompose}} \left\{ \begin{array}{lcl} x_t^{(1)} & \xrightarrow{\text{Send to workers}} & x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} & \longrightarrow & x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots & & \vdots \\ x_t^{(P)} & \longrightarrow & x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{array} \right\} \xrightarrow{\text{Reassembly}} x_{t+1}$$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t \xrightarrow{\text{Decompose}} \left\{ \begin{array}{lll} x_t^{(1)} & \xrightarrow{\text{Send to workers}} & x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} & \longrightarrow & x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots & & \vdots \\ x_t^{(P)} & \longrightarrow & x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{array} \right\} \xrightarrow{\text{Reassembly}} x_{t+1}$$

Different than $\nabla f_i(x_t^{(P)})$

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

$$x_t \xrightarrow{\text{Decompose}} \left\{ \begin{array}{ccc} x_t^{(1)} & \xrightarrow{\text{Send to workers}} & x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} & \longrightarrow & x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots & & \vdots \\ x_t^{(P)} & \longrightarrow & x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{array} \right\} \xrightarrow{\text{Reassembly}} x_{t+1}$$

Different than $\nabla f_i(x_t^{(P)})$

- Not pure stochastic gradient descent

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Send to workers

$$x_t \xrightarrow{\text{Decompose}} \left\{ \begin{array}{lll} x_t^{(1)} & \longrightarrow & x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} & \longrightarrow & x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots & & \vdots \\ x_t^{(P)} & \longrightarrow & x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{array} \right\} \xrightarrow{\text{Reassembly}} x_{t+1}$$

Different than $\nabla f_i(x_t^{(P)})$

- Not pure stochastic gradient descent
- Not pure stochastic coordinate descent

Independent Subnet Training: Iteration decomposition

$$\min_x \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

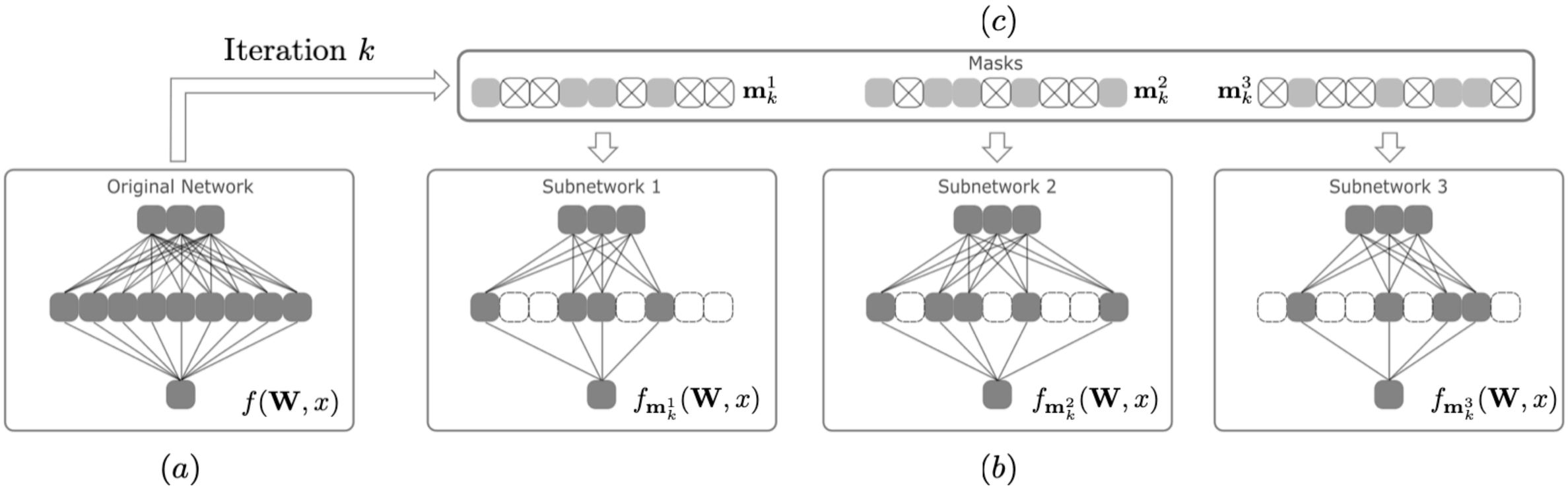
Send to workers

$$x_t \xrightarrow{\text{Decompose}} \left\{ \begin{array}{lll} x_t^{(1)} & \longrightarrow & x_{t+1}^{(1)} = x_t^{(1)} - \eta g_1(x_t^{(1)}) \\ x_t^{(2)} & \longrightarrow & x_{t+1}^{(2)} = x_t^{(2)} - \eta g_2(x_t^{(2)}) \\ \vdots & & \vdots \\ x_t^{(P)} & \longrightarrow & x_{t+1}^{(P)} = x_t^{(P)} - \eta g_P(x_t^{(P)}) \end{array} \right\} \xrightarrow{\text{Reassembly}} x_{t+1}$$

Different than $\nabla f_i(x_t^{(P)})$

- Not pure stochastic gradient descent
- Not pure stochastic coordinate descent
- Not exactly quantized stochastic coordinate descent

Independent Subnet Training: Single hidden FC layer



- Task: Regression

$$L_k = \|\mathbf{y} - f(\mathbf{W}_k, \mathbf{X})\|_2^2$$

- Model: 1-hidden layer

$$f(\mathbf{W}, \mathbf{x}) := f(\mathbf{W}, \mathbf{a}, \mathbf{x}) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} a_r \sigma(\langle \mathbf{w}_r, \mathbf{x} \rangle)$$

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Mild assumptions on data properties + subnetwork random selection
- Some interesting quantities:

  p : number of workers

  ξ : dropout rate

  $\theta = 1-(1-\xi)^p$: denotes probability a neuron is selected by a subnet
- Sampling scheme: Allows neurons to be selected by more than one subnet

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Mild assumptions on data properties + subnetwork random selection
- Some interesting quantities:

  p : number of workers

  ξ : dropout rate

  $\theta = 1-(1-\xi)^p$: denotes probability a neuron is selected by a subnet
- Sampling scheme: Allows neurons to be selected by more than one subnet

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Mild assumptions on data properties + subnetwork random selection
- Some interesting quantities:

  p : number of workers

  ξ : dropout rate

  $\theta = 1-(1-\xi)^p$: denotes probability a neuron is selected by a subnet
- Sampling scheme: Allows neurons to be selected by more than one subnet

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n, p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1 - \delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y} - \mathbf{u}_k\|_2^2\right] \leq \left(1 - \frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y} - \mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta - \xi^2}{p} + \left(1 - \frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Mild assumptions on data properties + subnetwork random selection
- Some interesting quantities:

  p : number of workers

  ξ : dropout rate

  $\theta = 1 - (1-\xi)^p$: denotes probability a neuron is selected by a subnet
- Sampling scheme: Allows neurons to be selected by more than one subnet

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Mild assumptions on data properties + subnetwork random selection
- Some interesting quantities:

  p : number of workers

  ξ : dropout rate

  $\theta = 1-(1-\xi)^p$: denotes probability a neuron is selected by a subnet
- Sampling scheme: Allows neurons to be selected by more than one subnet

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Interesting take aways:
  - Increasing the number of subnets/workers improve the convergence rate, and the overparameterization requirement (through θ)

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Interesting take aways:
  - Increasing the number of subnets/workers improve the convergence rate, and the overparameterization requirement (through θ)

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Interesting take aways:
  - Increasing the number of subnets/workers improve the convergence rate, and the overparameterization requirement (through θ)
  - Increasing the number of subnets/workers improve the error region convergence, when dropout rate is fixed.

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Interesting take aways:
  - Increasing the number of subnets/workers improve the convergence rate, and the overparameterization requirement (through θ)
  - Increasing the number of subnets/workers improve the error region convergence, when dropout rate is fixed.

Independent Subnet Training: Single hidden FC layer

Corollary 3. *Let assumptions (1), (2), and (3) hold. Fix the number of dropout iterations to K, the step size to $\eta = O\left(\lambda_0 / n\tau \max\{n,p\}\right)$, and let the number of hidden neurons satisfy $m = \Theta\left(n^5 K / \xi\theta\lambda_0^4\delta\right)$. Then the IST algorithm on a two-layer ReLU neural network converges with probability at least $1-\delta$, according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(1-\frac{1}{4}\eta\theta\tau\lambda_0\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{(1-\xi)^2}{nK} + \frac{\theta-\xi^2}{p} + \left(1-\frac{1}{\tau}\right)\theta^2(1-\xi)\right)$$

- Interesting take aways:
  - Increasing the number of subnets/workers improve the convergence rate, and the overparameterization requirement (through θ)
  - Increasing the number of subnets/workers improve the error region convergence, when dropout rate is fixed.
  - The first and third terms in error region comes from dropout theory

Independent Subnet Training: Single hidden FC layer

Theorem 4. *Let assumptions (1) and (4) hold. Then* $\lambda_0 > 0$. *Moreover, let* $\lambda_{\max}$ *denote the maximum eigenvalue of* $\mathbf{H}^{\infty}$. *Fix the number of global iterations to* K *and the number of local iterations to* τ. *Let the number of hidden neurons be* $m = \Omega\left(\frac{n^5\tau^2 K\lambda_{\max}}{\lambda_0^6\delta}\right)$, *and choose the initialization scale* $\kappa = \sqrt{n\lambda_{\max}}\lambda_0^{-1}$. *Let* $\gamma = \left(1-p^{-1}\right)^{\frac{1}{3}}$. *Then, Algorithm (1) with a constant step-size* $\eta = O\left(\frac{\lambda_0}{n^2}\min\left\{\frac{p}{\gamma^2\tau}, 1\right\}\right)$ *converges with probability at least* $1-\delta$, *according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1-\frac{\eta\lambda_0}{2}\right)^{\tau}\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{\gamma\tau n\kappa^2\lambda_{\max}}{\lambda_0^2}\right).$$

- When neurons are not shared among subnets
- When $p = 1,\ \gamma = 0$ i.e., IST becomes full training conv. rate guarantee
- When $p \uparrow$ i.e., convergence rate becomes slower for fixed overparameterization (reasonable)

Independent Subnet Training: Single hidden FC layer

Theorem 4. *Let assumptions (1) and (4) hold. Then* $\lambda_0 > 0$. *Moreover, let* $\lambda_{\max}$ *denote the maximum eigenvalue of* $\mathbf{H}^{\infty}$. *Fix the number of global iterations to* K *and the number of local iterations to* τ. *Let the number of hidden neurons be* $m = \Omega\left(\frac{n^5\tau^2 K\lambda_{\max}}{\lambda_0^6\delta}\right)$, *and choose the initialization scale* $\kappa = \sqrt{n\lambda_{\max}}\lambda_0^{-1}$. *Let* $\gamma = \left(1-p^{-1}\right)^{\frac{1}{3}}$. *Then, Algorithm (1) with a constant step-size* $\eta = O\left(\frac{\lambda_0}{n^2}\min\left\{\frac{p}{\gamma^2\tau}, 1\right\}\right)$ *converges with probability at least* $1-\delta$, *according to:*

$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1-\frac{\eta\lambda_0}{2}\right)^{\tau}\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{\gamma\tau n\kappa^2\lambda_{\max}}{\lambda_0^2}\right).$$

- When neurons are not shared among subnets
- When $p = 1,\ \gamma = 0$ i.e., IST becomes full training conv. rate guarantee
- When $p \uparrow$ i.e., convergence rate becomes slower for fixed overparameterization (reasonable)

Independent Subnet Training: Single hidden FC layer

Theorem 4. *Let assumptions (1) and (4) hold. Then* $\lambda_0 > 0$. *Moreover, let* $\lambda_{\max}$ *denote the maximum eigenvalue of* $\mathbf{H}^{\infty}$. *Fix the number of global iterations to* K *and the number of local iterations to* τ. *Let the number of hidden neurons be* $m = \Omega\left(\frac{n^5\tau^2 K\lambda_{\max}}{\lambda_0^6\delta}\right)$, *and choose the initialization scale* $\kappa = \sqrt{n\lambda_{\max}}\lambda_0^{-1}$. *Let* $\gamma = \left(1-p^{-1}\right)^{\frac{1}{3}}$. *Then, Algorithm (1) with a constant step-size* $\eta = O\left(\frac{\lambda_0}{n^2}\min\left\{\frac{p}{\gamma^2\tau}, 1\right\}\right)$ *converges with probability at least* $1-\delta$, *according to:*

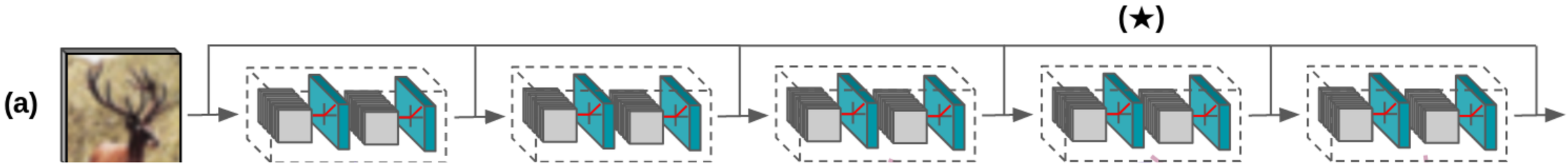
$$\mathbb{E}_{[\mathbf{M}_{k-1}]}\left[\|\mathbf{y}-\mathbf{u}_k\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1-\frac{\eta\lambda_0}{2}\right)^{\tau}\right)^k \|\mathbf{y}-\mathbf{u}_0\|_2^2 + O\left(\frac{\gamma\tau n\kappa^2\lambda_{\max}}{\lambda_0^2}\right).$$

- When neurons are not shared among subnets
- When $p = 1,\ \gamma = 0$ i.e., IST becomes full training conv. rate guarantee
- When $p \uparrow$ i.e., convergence rate becomes slower for fixed overparameterization (reasonable)

Does it work only for FC layers?

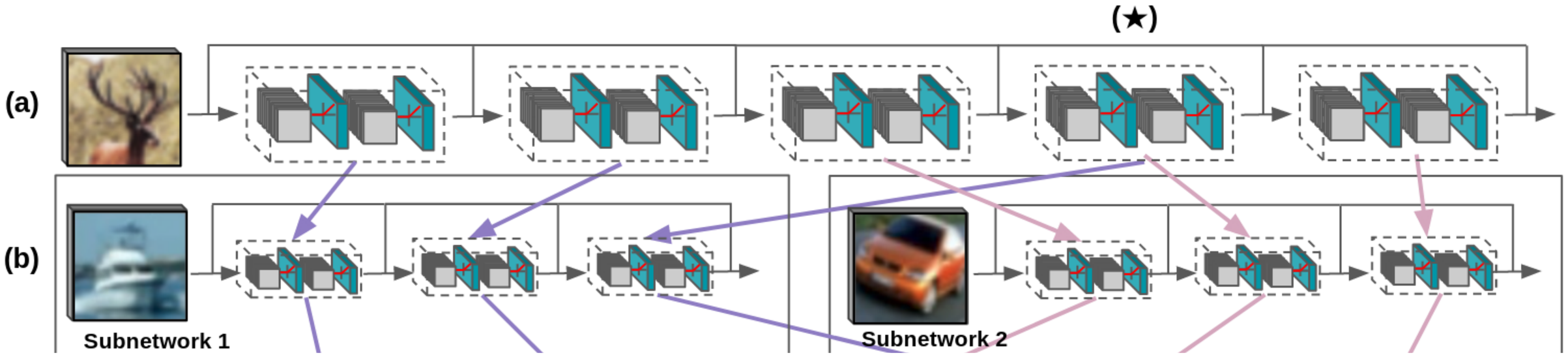
IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]



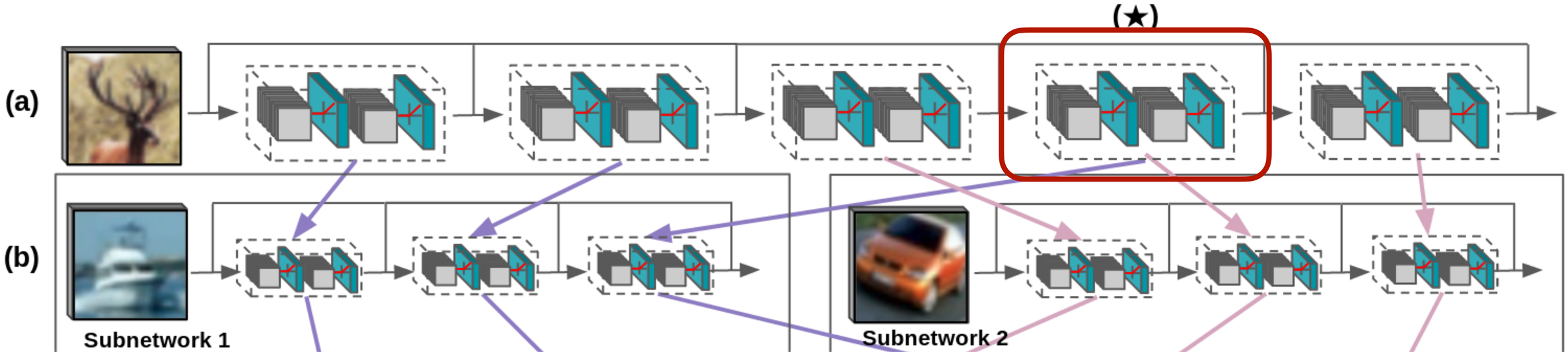
IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]



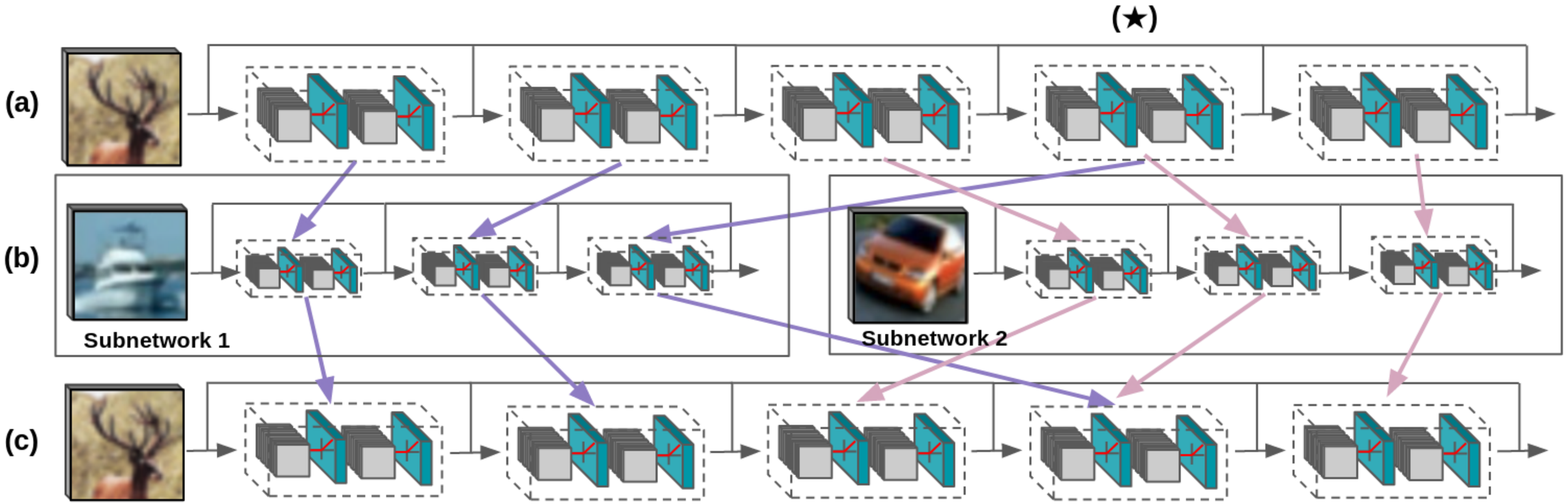
IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]



IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]



IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Seems simple.. but it is not!

Using ResNet101

Modification	**Naive Model**						**ResIST**
Share strided layers		✓	✓	✓	✓	✓	✓
Only Partition Section 3			✓	✓	✓	✓	✓
Scale Activations				✓	✓	✓	✓
Pre-Act ResNet					✓	✓	✓
Minimum Depth						✓	✓
Tune Local Iterations							✓
2 Sub-ResNet Acc.	66.3%	68.3%	84.3%	91.5%	92.0%	-	92.0%
8 Sub-ResNet Acc.	-	-	-	-	86.5%	89.9%	91.3%

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Seems simple.. but it is not!

Using ResNet101

Modification	Naive Model						ResIST
Share strided layers		✓	✓	✓	✓	✓	✓
Only Partition Section 3			✓	✓	✓	✓	✓
Scale Activations				✓	✓	✓	✓
Pre-Act ResNet					✓	✓	✓
Minimum Depth						✓	✓
Tune Local Iterations							✓
2 Sub-ResNet Acc.	66.3%	68.3%	84.3%	91.5%	92.0%	-	92.0%
8 Sub-ResNet Acc.	-	-	-	-	86.5%	89.9%	91.3%

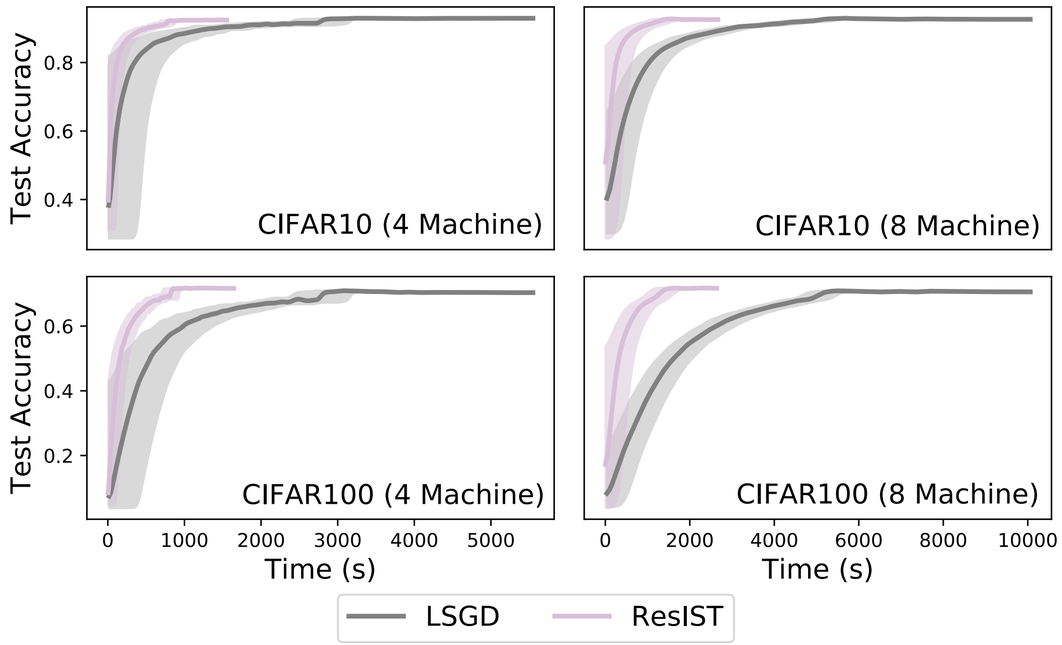
IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Seems simple.. but it is not!

Modification	Naive Model						ResIST
Share strided layers		✓	✓	✓	✓	✓	✓
Only Partition Section 3			✓	✓	✓	✓	✓
Scale Activations				✓	✓	✓	✓
Pre-Act ResNet					✓	✓	✓
Minimum Depth						✓	✓
Tune Local Iterations							✓
2 Sub-ResNet Acc.	66.3%	68.3%	84.3%	91.5%	92.0%	-	92.0%
8 Sub-ResNet Acc.	-	-	-	-	86.5%	89.9%	91.3%

Using ResNet101



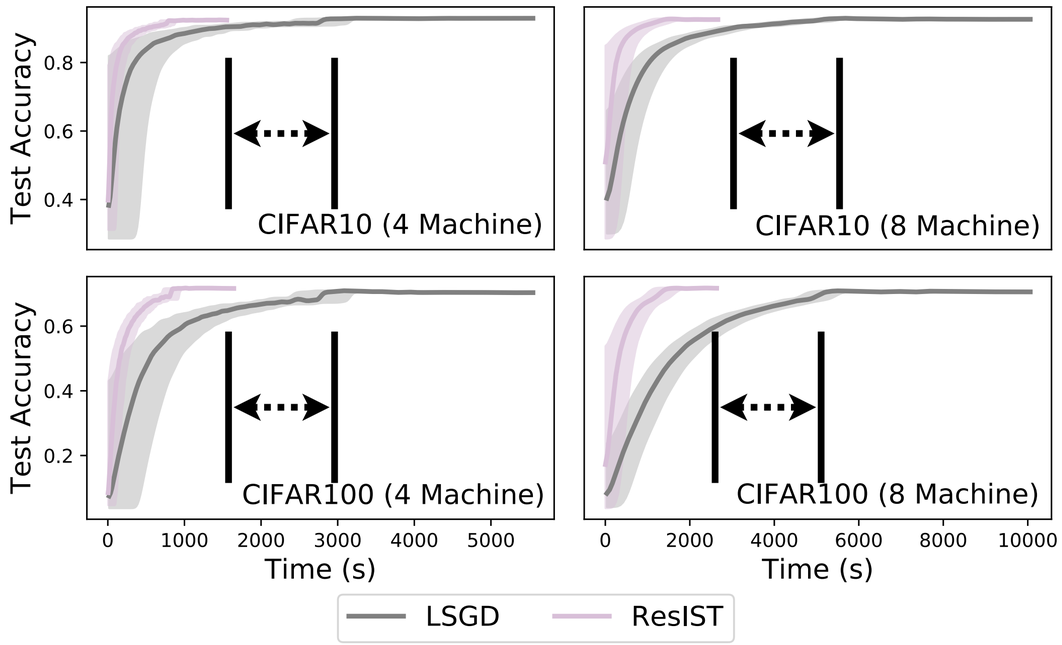
IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Seems simple.. but it is not!

Modification	Naive Model						ResIST
Share strided layers		✓	✓	✓	✓	✓	✓
Only Partition Section 3			✓	✓	✓	✓	✓
Scale Activations				✓	✓	✓	✓
Pre-Act ResNet					✓	✓	✓
Minimum Depth						✓	✓
Tune Local Iterations							✓
2 Sub-ResNet Acc.	66.3%	68.3%	84.3%	91.5%	92.0%	-	92.0%
8 Sub-ResNet Acc.	-	-	-	-	86.5%	89.9%	91.3%

Using ResNet101



IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Speedup in absolute numbers

Table 3. Total training time in seconds of models trained with both local SGD and `ResIST` using four machines.

Dataset	Local SGD	`ResIST`	Speedup
CIFAR10	5486 ± 7.05	1532 ± 0.83	$\times$ **3.60**
CIFAR100	5528 ± 65.90	1545 ± 1.27	$\times$ **3.58**
PascalVOC	16840 ± 0.11	11264 ± 49.38	$\times$ **1.49**

Table 4. Total training time in seconds of models trained with both local SGD and `ResIST` using eight machines.

Dataset	Local SGD	`ResIST`	Speedup
CIFAR10	10072 ± 5.12	2671 ± 3.25	$\times$ **3.77**
CIFAR100	10058 ± 8.71	2639 ± 3.89	$\times$ **3.81**

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Speedup in absolute numbers

Table 3. Total training time in seconds of models trained with both local SGD and `ResIST` using four machines.

Dataset	Local SGD	`ResIST`	Speedup
CIFAR10	5486 ± 7.05	1532 ± 0.83	$\times$ **3.60**
CIFAR100	5528 ± 65.90	1545 ± 1.27	$\times$ **3.58**
PascalVOC	16840 ± 0.11	11264 ± 49.38	$\times$ **1.49**

Table 4. Total training time in seconds of models trained with both local SGD and `ResIST` using eight machines.

Dataset	Local SGD	`ResIST`	Speedup
CIFAR10	10072 ± 5.12	2671 ± 3.25	$\times$ **3.77**
CIFAR100	10058 ± 8.71	2639 ± 3.89	$\times$ **3.81**

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Similar behavior regardless of depth

Table 6. Test accuracy on CIFAR10 (C10) and CIFAR100 (C100) for deeper architectures trained with `ResIST` and local SGD (LSGD). All tests were performed with 100 local iterations between synchronization rounds. All models were trained for 80 epochs.

Dataset	# Machines	Method	ResNet152 Time	ResNet152 Test Acc.	ResNet152 Speedup	ResNet200 Time	ResNet200 Test Acc.	ResNet200 Speedup
C10	2	LSGD	3512s	92.27% ± 0.003		4575s	92.31% ± 0.001	
		`ResIST`	2215s	92.01% ± 0.002	× **1.58**	2380s	92.10% ± 0.001	× **1.92**
	4	LSGD	3598s	91.39% ± 0.001		4357s	91.35% ± 0.000	
		`ResIST`	1054s	90.67% ± 0.001	× **3.41**	1161s	90.27% ± 0.001	× **3.75**
C100	2	LSGD	3528s	70.50% ± 0.003		4639s	71.05% ± 0.005	
		`ResIST`	2291s	70.32% ± 0.005	× **1.53**	2202s	70.71% ± 0.002	× **2.10**
	4	LSGD	3518s	68.39% ± 0.004		4391s	69.05% ± 0.003	
		`ResIST`	1164s	67.27% ± 0.003	× **3.02**	1195s	67.62% ± 0.001	× **3.67**

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Similar behavior regardless of depth

Table 6. Test accuracy on CIFAR10 (C10) and CIFAR100 (C100) for deeper architectures trained with `ResIST` and local SGD (LSGD). All tests were performed with 100 local iterations between synchronization rounds. All models were trained for 80 epochs.

			ResNet152			ResNet200		
Dataset	# Machines	Method	Time	Test Acc.	Speedup	Time	Test Acc.	Speedup
C10	2	LSGD	3512s	92.27% ± 0.003		4575s	92.31% ± 0.001	
		`ResIST`	2215s	92.01% ± 0.002	× **1.58**	2380s	92.10% ± 0.001	× **1.92**
	4	LSGD	3598s	91.39% ± 0.001		4357s	91.35% ± 0.000	
		`ResIST`	1054s	90.67% ± 0.001	× **3.41**	1161s	90.27% ± 0.001	× **3.75**
C100	2	LSGD	3528s	70.50% ± 0.003		4639s	71.05% ± 0.005	
		`ResIST`	2291s	70.32% ± 0.005	× **1.53**	2202s	70.71% ± 0.002	× **2.10**
	4	LSGD	3518s	68.39% ± 0.004		4391s	69.05% ± 0.003	
		`ResIST`	1164s	67.27% ± 0.003	× **3.02**	1195s	67.62% ± 0.001	× **3.67**

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- Similar behavior regardless of depth

Table 6. Test accuracy on CIFAR10 (C10) and CIFAR100 (C100) for deeper architectures trained with `ResIST` and local SGD (LSGD). All tests were performed with 100 local iterations between synchronization rounds. All models were trained for 80 epochs.

			ResNet152			ResNet200		
Dataset	# Machines	Method	Time	Test Acc.	Speedup	Time	Test Acc.	Speedup
C10	2	LSGD	3512s	92.27% ± 0.003		4575s	92.31% ± 0.001	
		`ResIST`	2215s	92.01% ± 0.002	× **1.58**	2380s	92.10% ± 0.001	× **1.92**
	4	LSGD	3598s	91.39% ± 0.001		4357s	91.35% ± 0.000	
		`ResIST`	1054s	90.67% ± 0.001	× **3.41**	1161s	90.27% ± 0.001	× **3.75**
C100	2	LSGD	3528s	70.50% ± 0.003		4639s	71.05% ± 0.005	
		`ResIST`	2291s	70.32% ± 0.005	× **1.53**	2202s	70.71% ± 0.002	× **2.10**
	4	LSGD	3518s	68.39% ± 0.004		4391s	69.05% ± 0.003	
		`ResIST`	1164s	67.27% ± 0.003	× **3.02**	1195s	67.62% ± 0.001	× **3.67**

Highlight: ResIST is theoretically proven with more complicated analysis than IST

(Skipped but happy to discuss – First of its kind)

IST on Residual Networks: ResIST

[Dun, Wolfe, Jermaine and Kyrillidis, 2021]

- What about theory?

Theorem B.1 (Convergence Rate of Gradient Descent for `ResIST`). *Assume there are S workers, ℓ local and T global steps. Assume the depth of the whole ResNet is H. Assume for all data indices $i \in [n]$, the data input satisfies $\|\mathbf{x}_i\|_2 = 1$, the data output satisfies $|y_i| = O(1)$, and the number of hidden nodes per layer satisfies $m =$*

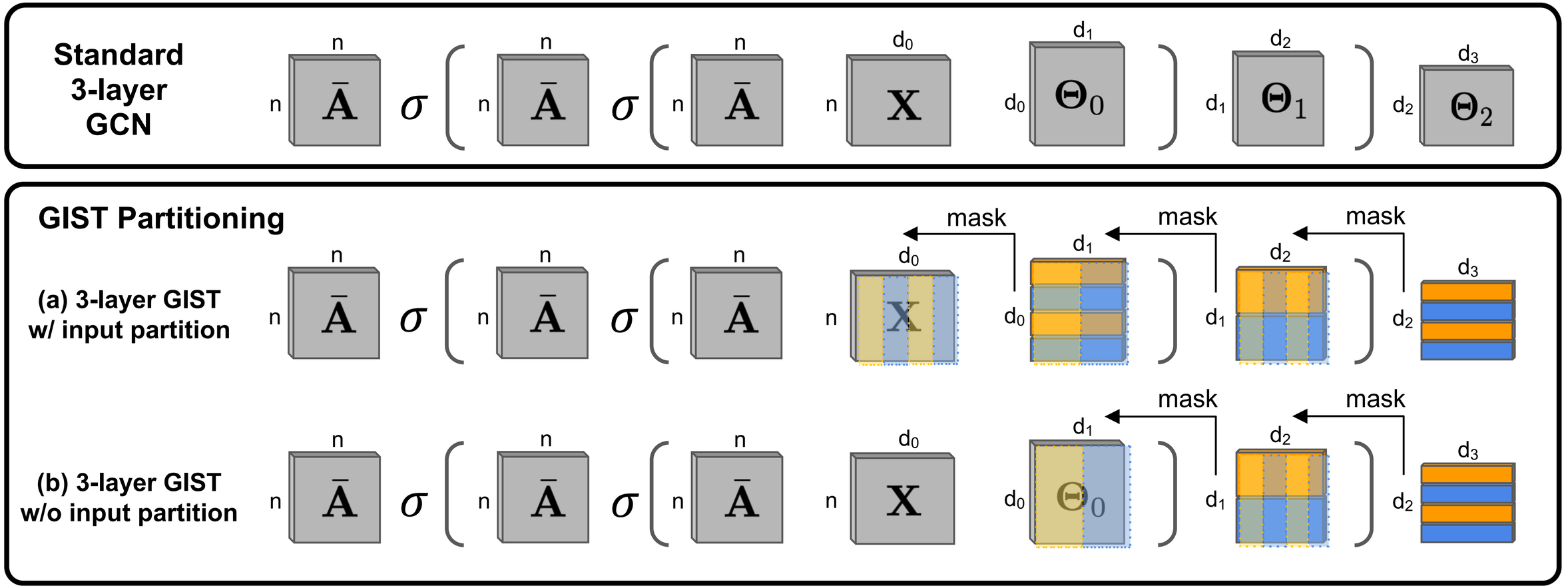
$$\Omega\bigg(\max\bigg\{ \frac{n^4}{\lambda_{\min}^4\big(\mathbf{K}^{(H)}\big)H^6}, \frac{n^2}{\lambda_{\min}^2(\mathbf{K}^{(H)})H^2}, \frac{n}{\delta}, \frac{n^2\log\big(\frac{Hn}{\delta}\big)}{\lambda_{\min}^2\big(\mathbf{K}^{(H)}\big)} \bigg\}\bigg).$$

Set the step size $\eta = O\bigg(\frac{\lambda_{\min}\big(\mathbf{K}^{(H)}\big)H^2}{n^2\ell^2 S}\bigg)$ in gradient descent in local training iteration, and follow the procedure as in Algorithm 1. Let the squared-norm loss be $L(\theta(t)) := \frac{1}{2}\|\mathbf{y} - f(\theta(t))\|_2^2$, per t global synchronization round, $t = 1, 2, \ldots T$; here, $\mathbf{y}$ corresponds to the data "labels", and $\theta(t)$ and $f(\theta(t))$ represent the parameters and the output of the whole ResNet, respectively, after t-global rounds of `ResIST`. Here, θ includes weights $\mathbf{W}^{(h)}$ at depth h and the last layer's weights $\mathbf{a}$. Then, with probability at least $1 - \delta$ over the random initialization, we have:

$$L(\theta(t)) \leq \bigg(1 - \frac{\eta\ell\lambda_{\min}\big(\mathbf{K}^{(H)}\big)}{2}\bigg)^t \cdot L(\theta(0)).$$

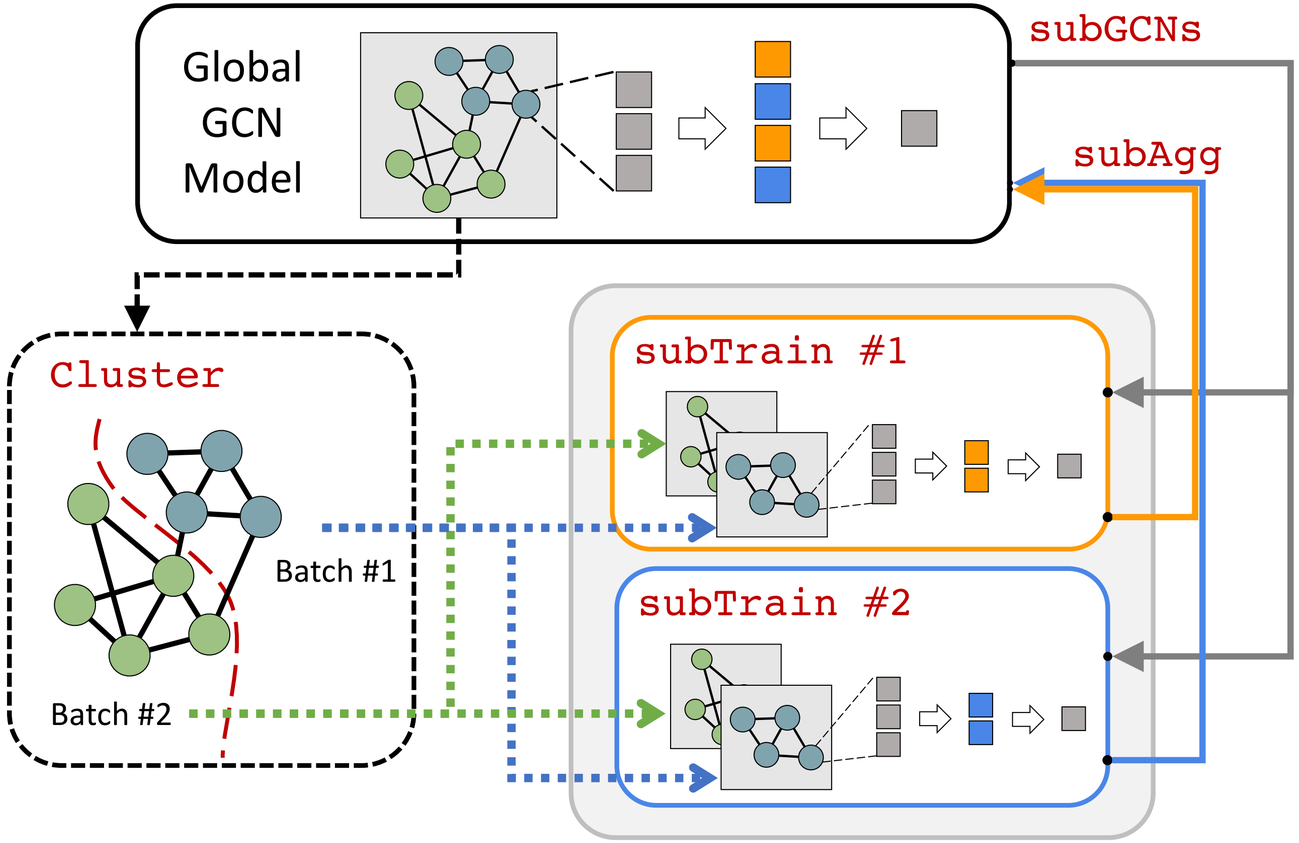
IST on Graph Convolutional Networks: GIST

[Wolfe, Yang, Chowdhury, Dun, Bayer, Segarra, and Kyrillidis, 2021]



IST on Graph Convolutional Networks: GIST

[Wolfe, Yang, Chowdhury, Dun, Bayer, Segarra, and Kyrillidis, 2021]



IST on Graph Convolutional Networks: GIST

[Wolfe, Yang, Chowdhury, Dun, Bayer, Segarra, and Kyrillidis, 2021]

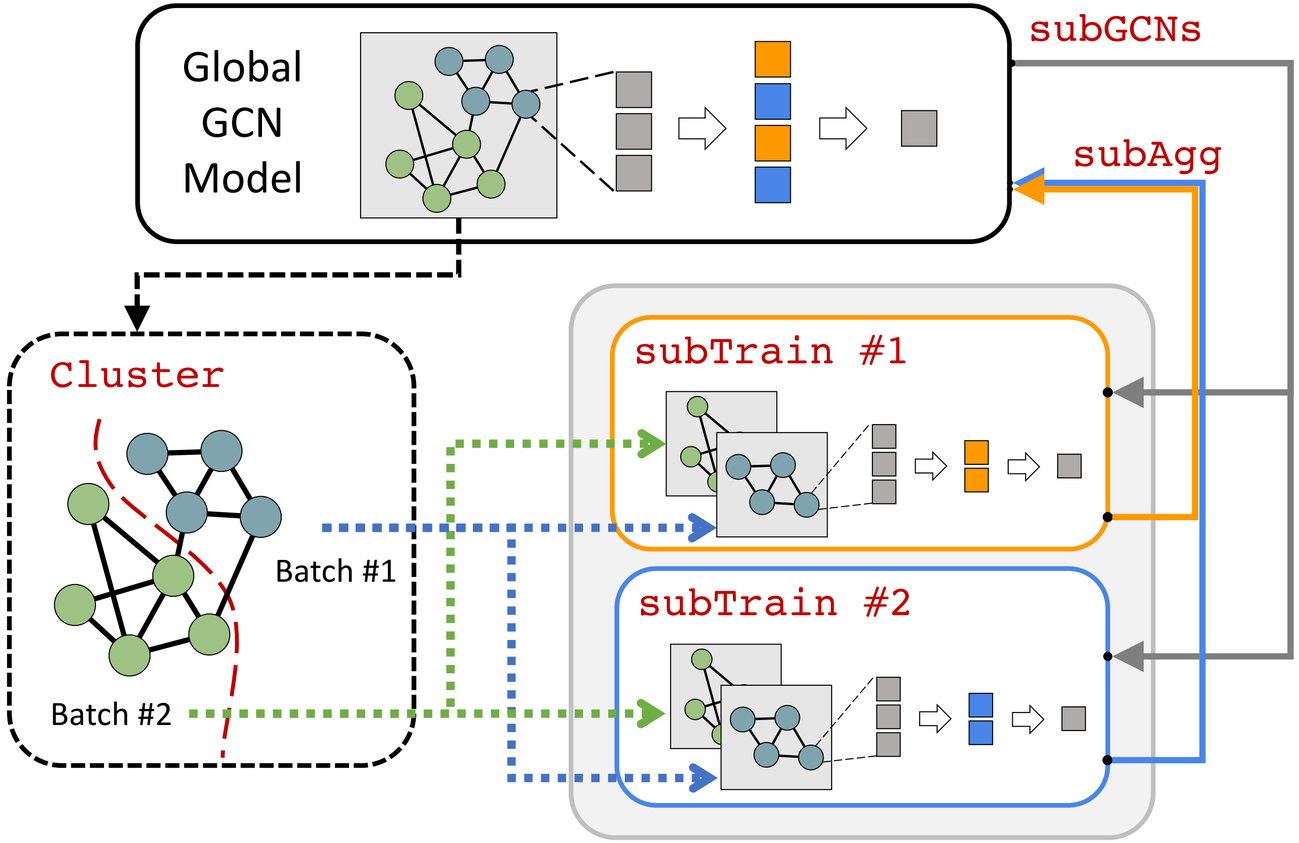


Table 4. Performance of `GIST` on the Amazon2M dataset with different hidden dimensions. Experiments marked with a "-" are excluded because training took more than 12 hours.

L	# Sub-GCNs	F1 Score (Time)	
		$d_i = 400$	$d_i = 4096$
2	Baseline	89.90 (6519.73s)	91.25 (18624.2s)
	2	88.36 (4485.33s)	90.70 (6109.88s)
	4	86.33 (4009.24s)	89.49 (4074.85s)
	8	84.73 (4061.81s)	88.86 (4003.87s)
3	Baseline	90.36 (8367.09s)	91.51 (34278.6s)
	2	88.59 (5615.34s)	91.12 (7626.45s)
	4	86.46 (4917.64s)	89.21 (5103.69s)
	8	84.76 (4936.97s)	86.97 (4812.80s)
4	Baseline	90.40 (10808.1s)	-
	2	88.56 (6443.53s)	91.02 (9966.33s)
	4	87.53 (5674.26s)	89.07 (5953.43s)
	8	85.32 (5601.51s)	87.53 (5596.43s)

IST on Graph Convolutional Networks: GIST

[Wolfe, Yang, Chowdhury, Dun, Bayer, Segarra, and Kyrillidis, 2021]

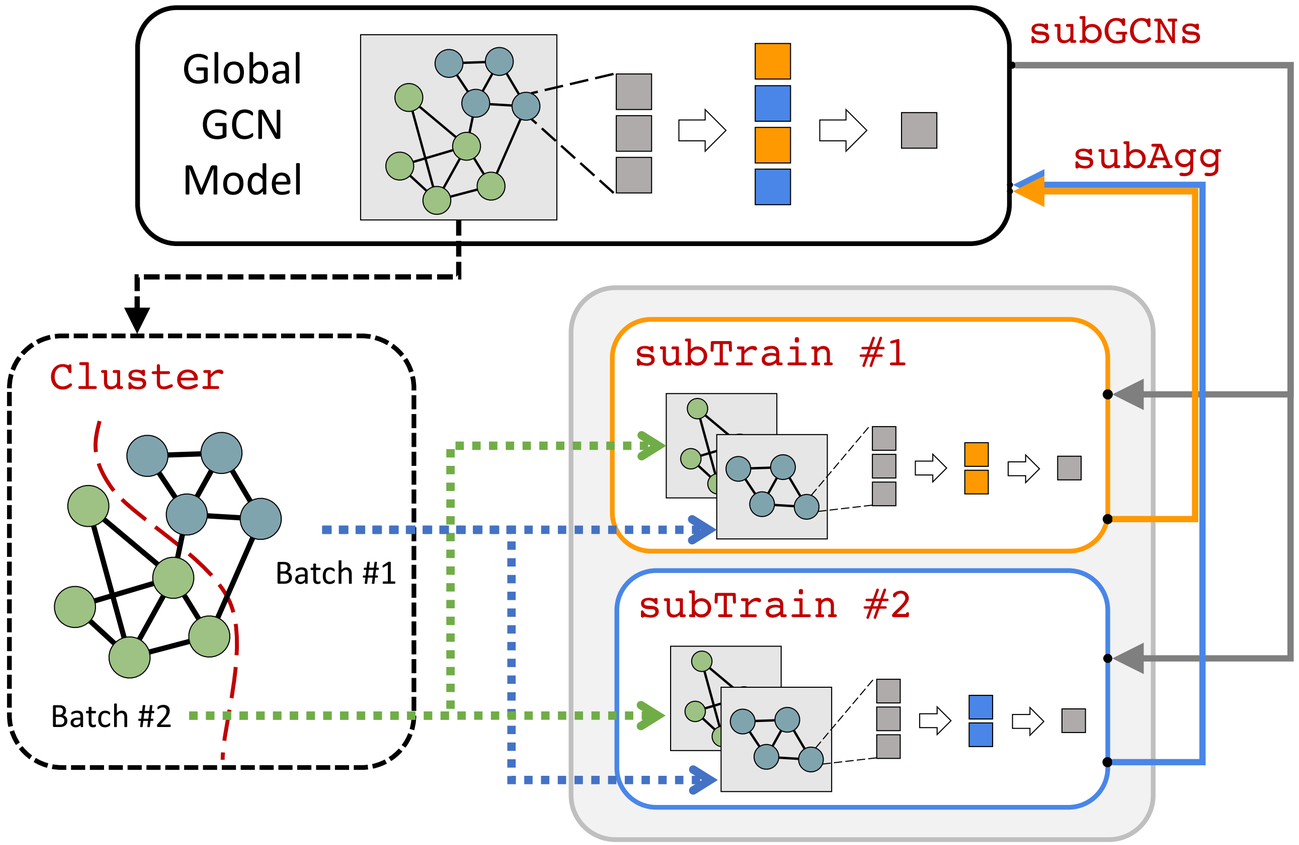


Table 4. Performance of `GIST` on the Amazon2M dataset with different hidden dimensions. Experiments marked with a "-" are excluded because training took more than 12 hours.

L	# Sub-GCNs	F1 Score (Time)	
		$d_i = 400$	$d_i = 4096$
2	Baseline	89.90 (6519.73s)	91.25 (18624.2s)
	2	88.36 (4485.33s)	90.70 (6109.88s)
	4	86.33 (4009.24s)	89.49 (4074.85s)
	8	84.73 (4061.81s)	88.86 (4003.87s)
3	Baseline	90.36 (8367.09s)	91.51 (34278.6s)
	2	88.59 (5615.34s)	91.12 (7626.45s)
	4	86.46 (4917.64s)	89.21 (5103.69s)
	8	84.76 (4936.97s)	86.97 (4812.80s)
4	Baseline	90.40 (10808.1s)	-
	2	88.56 (6443.53s)	91.02 (9966.33s)
	4	87.53 (5674.26s)	89.07 (5953.43s)
	8	85.32 (5601.51s)	87.53 (5596.43s)

Highlight: GIST allows deeper and wider GCNs using commercial GPUs
(Trained GCN with width 32,768 nodes)

Highlight: GIST is also theoretically proven

IST on Graph Convolutional Networks: GIST

[Wolfe, Yang, Chowdhury, Dun, Bayer, Segarra, and Kyrillidis, 2021]

- What about theory?

Theorem 2. *Suppose assumptions 2-4, and property 2 hold. Moreover, suppose in each global iteration the masks are generated from a categorical distribution with uniform mean* $1/m$. *Fix the number of global iterations to* T *and local iterations to* ζ. *If the number of hidden neurons satisfies* $d_1 = \Omega\left(\frac{n^3\zeta^2T^2}{\delta^2\gamma(1-\gamma)^2\lambda_0^4}\left(n + \frac{d}{m^2}\|\bar{\mathbf{A}}^2\|_{1,1}\right)\right)$, *then procedure (3) with constant step size* $\eta = O\left(\frac{\lambda_0}{n^2\|\bar{\mathbf{A}}^2\|_{1,1}}\right)$ *converges according to*

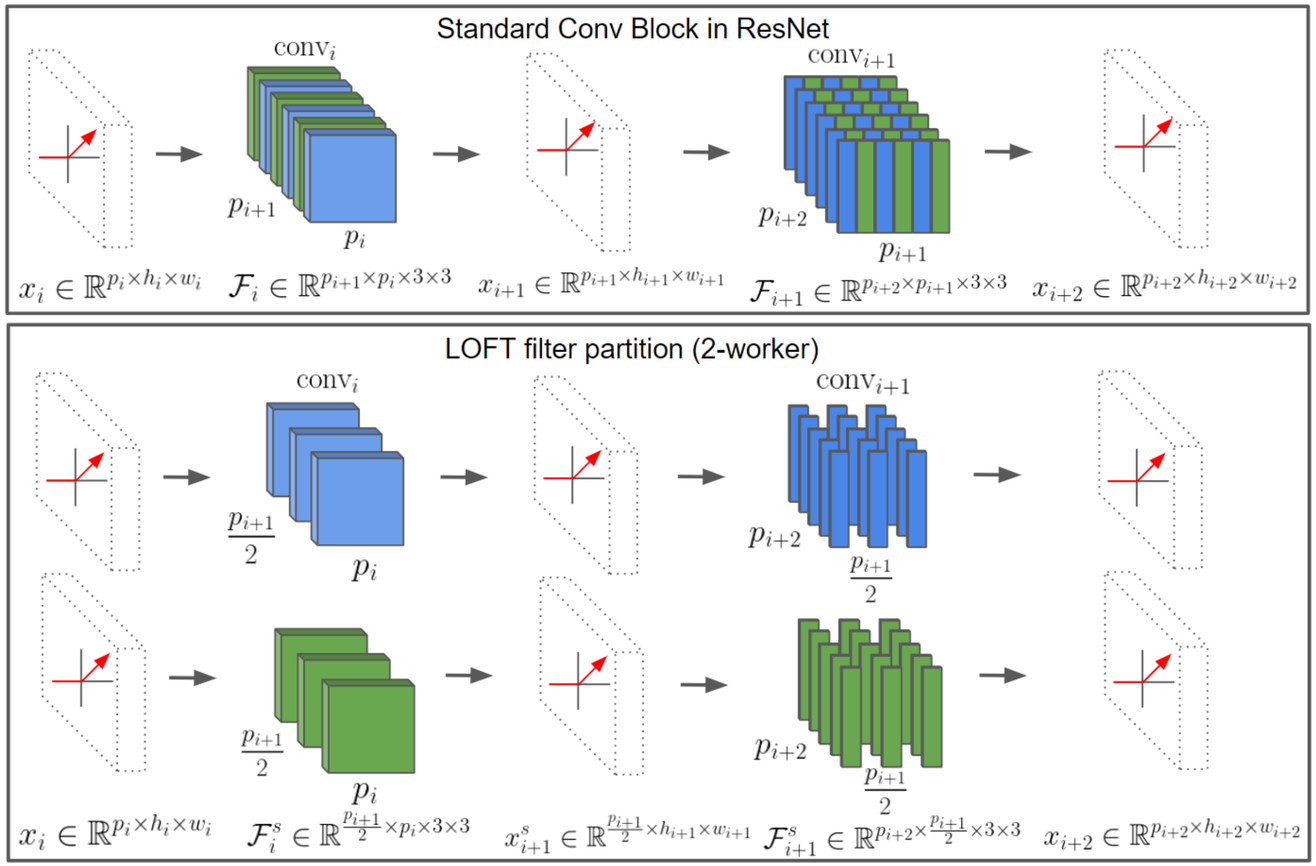
$$\mathbb{E}_{[\mathcal{M}_{t-1}],\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y} - \hat{\mathbf{y}}(t)\|_2^2\right] \leq \left(\gamma + (1-\gamma)\left(1 - \frac{\eta\lambda_0}{2}\right)^{\zeta}\right)^t \mathbb{E}_{\boldsymbol{\Theta}_0,\mathbf{a}}\left[\|\mathbf{y} - \hat{\mathbf{y}}(0)\|_2^2\right] + O\left(\frac{(m-1)^2\zeta\|\bar{\mathbf{A}}^2\|_{1,1}nd}{\gamma m^2 d_1}\right)$$

with probability at least $1 - \delta$.

IST as efficient pretraining for LTH: LOFT

[Wang, Dun, Wolfe and Kyrillidis, 2022]

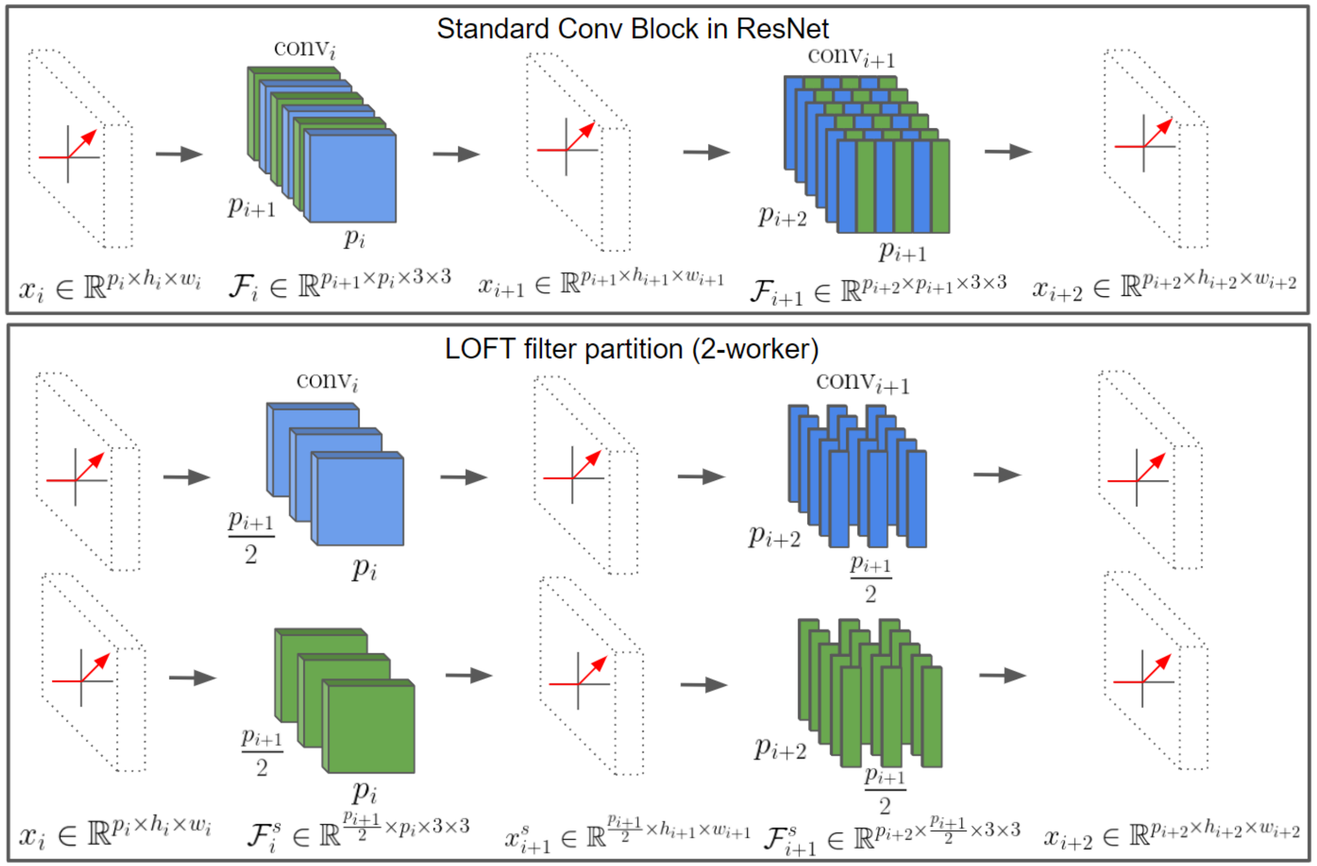
IST on CNNs



IST as efficient pretraining for LTH: LOFT

[Wang, Dun, Wolfe and Kyrillidis, 2022]

IST on CNNs

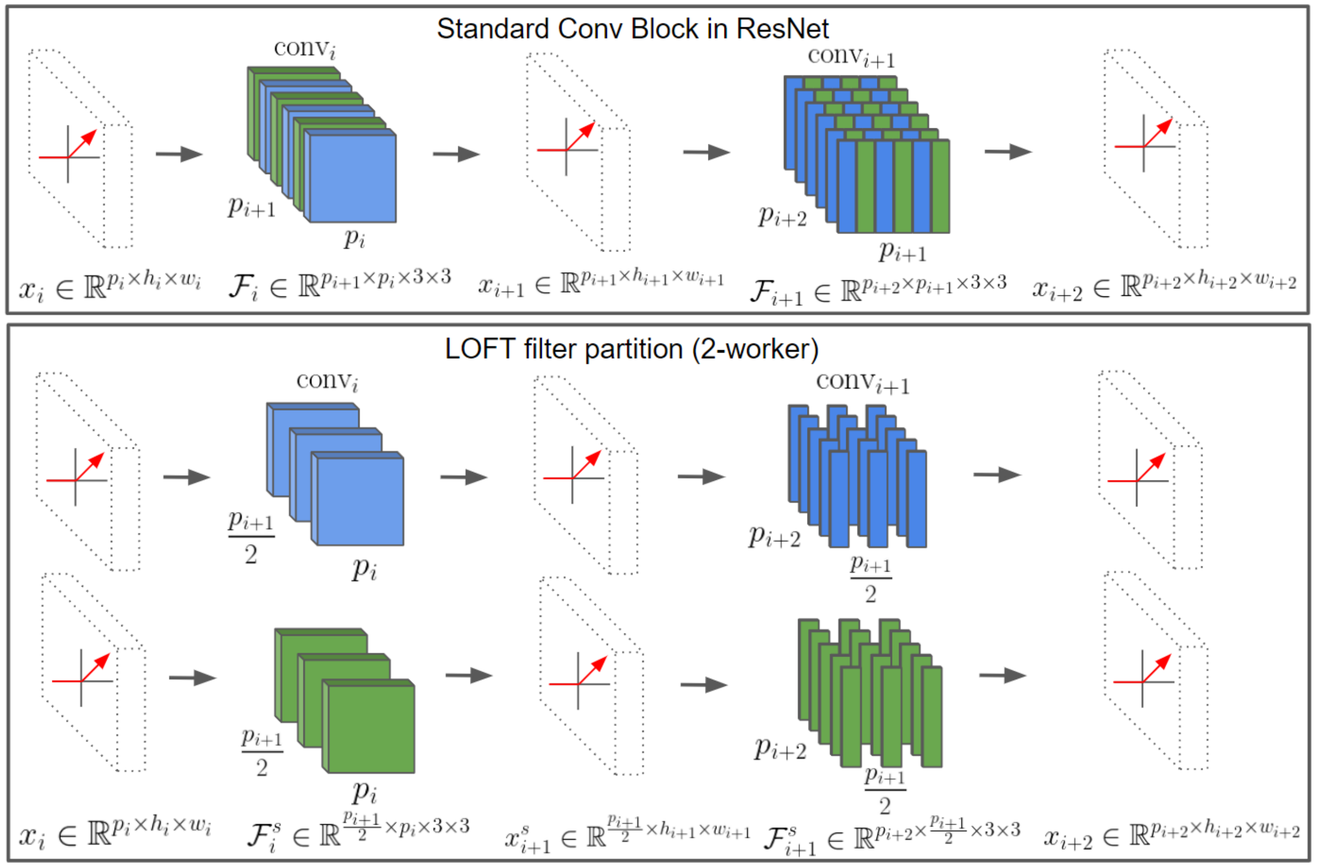


- Idea: can IST be combined with pruning techniques to ease distributed computing? Can IST identify "lottery tickets"?

IST as efficient pretraining for LTH: LOFT

[Wang, Dun, Wolfe and Kyrillidis, 2022]

IST on CNNs



- Idea: can IST be combined with pruning techniques to ease distributed computing? Can IST identify "lottery tickets"?
- Empirically: the answer is yes! IST allows pretraining of large-models without ever ..training the full model.
- Further, IST "preserves" lottery tickets for pruning

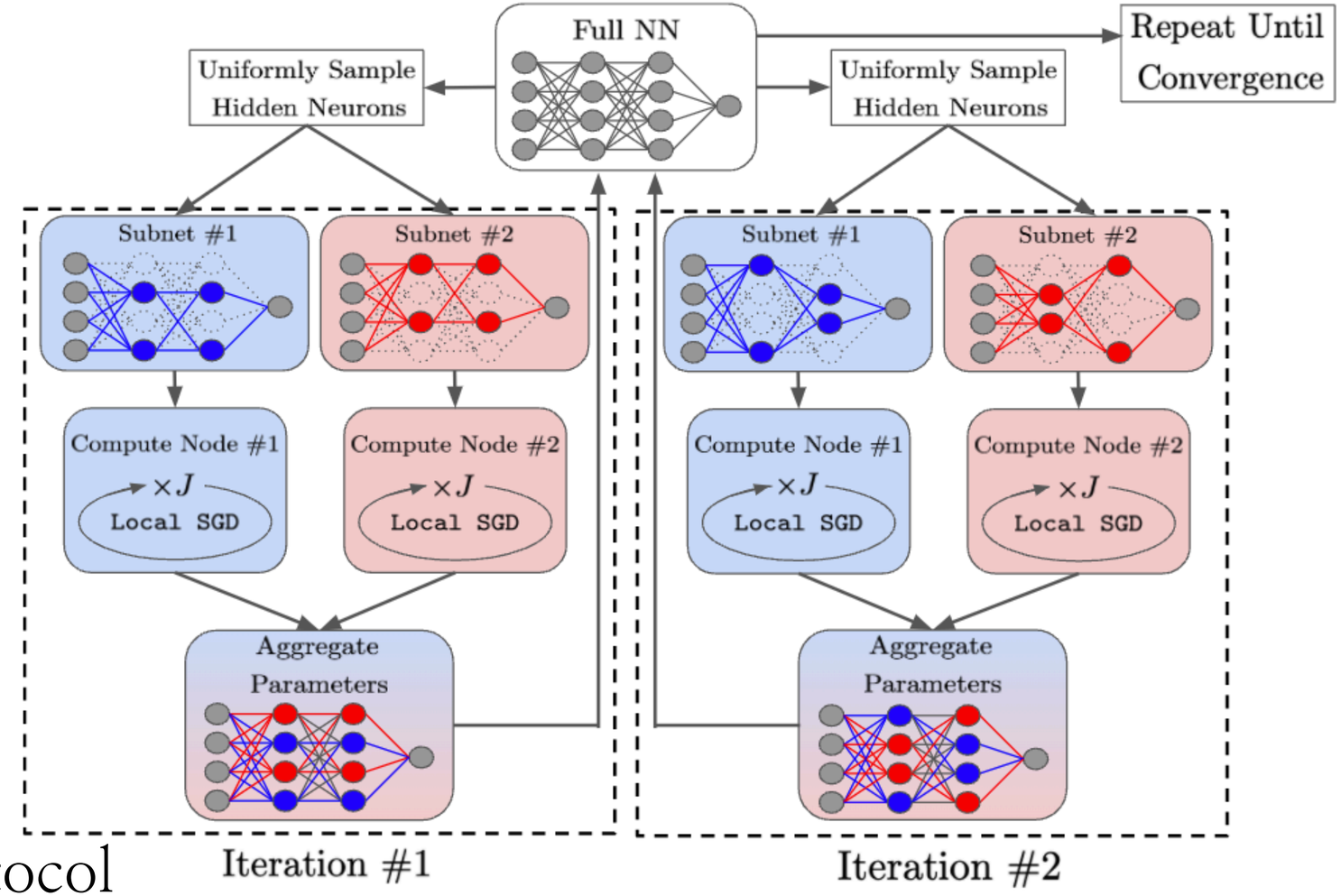
Landscape of IST-related papers

- Idea of splitting the model into smaller ones (as part of MM) – Google [Caldas, Konecny, McMahan, Talwalkar, 2018]

  Remark: IST is different since each worker receives a different model to aggregate

- One concurrent work from a FL perspective (to reduce comm/comp) [Jiang, Wang, Valls, Ko, Lee, Leung, Tassiulas, 2019]
- Several works after IST:
  - Helios [Xu, Yu, Xiong and Chen, 2019]
  - HeteroFL [Diao, Ding, and Tarokh, 2020]
  - FjORD [Horvath, Laskaridis, Almeida, Leontiadis, Venieris and Lane, 2021]
  - PVT by Google [Yang, Guliani, Beaufays and Motta, 2021]
  - Masked NNs [Mohtashami, Jaggi, Stich, 2021]
  - General theory [Zhou, Lan, Venkataramani and Ding, 2022]
  - Federated Dropout [Wen, Jean, and Huang, 2022]
  - Federated Pruning (Google) [Lin et al., 2022]

Take home message



- A.. different distributed protocol
- Potential impact on communication, compute requirements
- We need a clearer view of large-scale models with hundreds of workers
- Unifies well existing models with theory

Concluding with questions

- Statistical/Optimization guarantees why this technique works: **On going work**

Concluding with questions

- Statistical/Optimization guarantees why this technique works: **On going work**
- Extensions to other architectures: CNNs, ResNets, Transformers (**on going work**)

Concluding with questions

- Statistical/Optimization guarantees why this technique works: **On going work**
- Extensions to other architectures: CNNs, ResNets, Transformers **(on going work)**

Concluding with questions

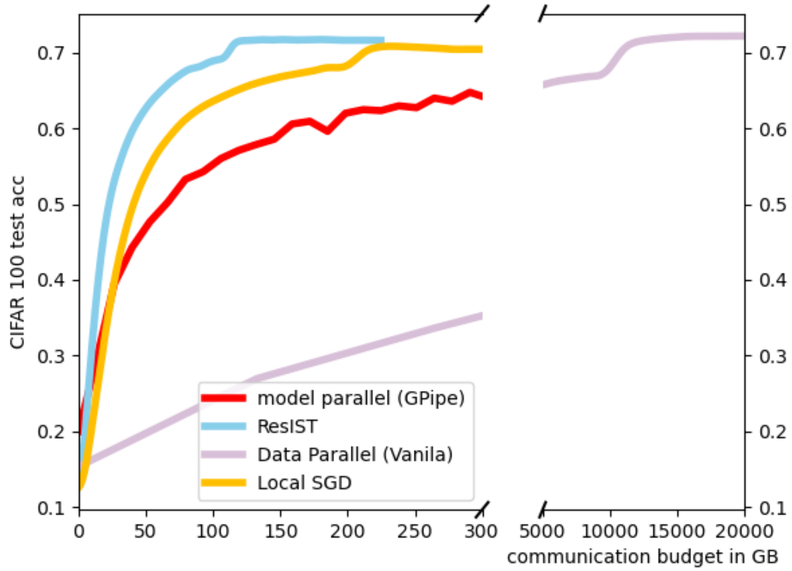
- Statistical/Optimization guarantees why this technique works: **On going work**
- Extensions to other architectures: CNNs, ResNets, Transformers **(on going work)**
- Federated Learning: Does IST help in this setting? Does it help with non-IID data? **(on going work)**

Concluding with questions

- Statistical/Optimization guarantees why this technique works: **On going work**
- Extensions to other architectures: CNNs, ResNets, Transformers **(on going work)**
- Federated Learning: Does IST help in this setting? Does it help with non-IID data? **(on going work)**
- Does IST conflict with quantization techniques? No.

Table 7. Performance of combining quantization with `ResIST` on CIFAR10 and CIFAR100 (denoted as C10 and C100, respectively).

Dataset	8 bit	7 bit	6 bit	5 bit	4 bit
C10	92.14	92.26	91.91	91.35	76.33
C100	71.38	72.15	71.37	68.29	40.48

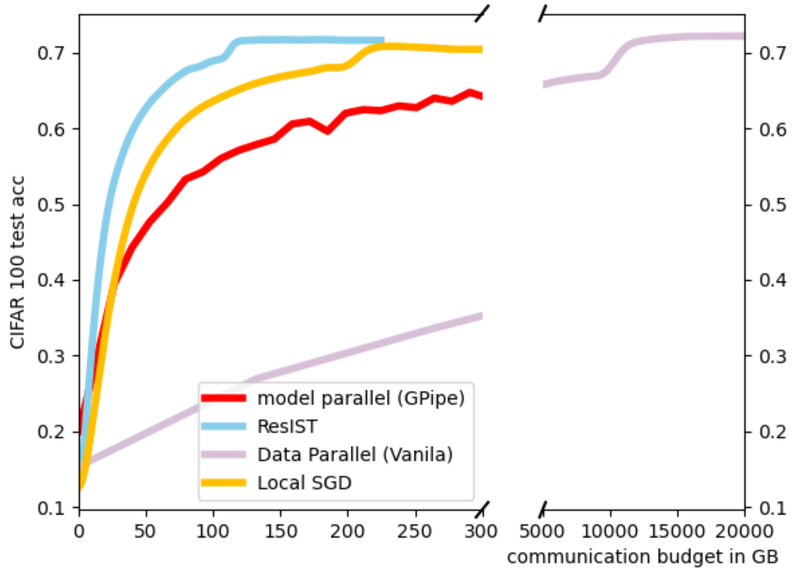


Concluding with questions

- Statistical/Optimization guarantees why this technique works: **On going work**
- Extensions to other architectures: CNNs, ResNets, Transformers **(on going work)**
- Federated Learning: Does IST help in this setting? Does it help with non-IID data? **(on going work)**
- Does IST conflict with quantization techniques? No.

Table 7. Performance of combining quantization with `ResIST` on CIFAR10 and CIFAR100 (denoted as C10 and C100, respectively).

Dataset	8 bit	7 bit	6 bit	5 bit	4 bit
C10	92.14	92.26	91.91	91.35	76.33
C100	71.38	72.15	71.37	68.29	40.48



- Implementation with smartphones: Real federated learning

Published at VLDB 2022

Distributed Learning of Neural Networks using Independent Subnet Training

Binhang Yuan
Rice University
by8@rice.edu

Cameron R. Wolfe
Rice University
crw13@rice.edu

Chen Dun
Rice University
cd46@rice.edu

Yuxin Tang
Rice University
yuxin.tang@rice.edu

Anastasios Kyrillidis
Rice University
anastasios@rice.edu

Chris Jermaine
Rice University
cmj4@rice.edu

https://arxiv.org/pdf/1910.02120.pdf

Published at UAI 2022

ResIST: Layer-Wise Decomposition of ResNets for Distributed Training

Chen Dun [*1] Cameron R. Wolfe [*1] Chris Jermaine [1] Anastasios Kyrillidis [1]

https://arxiv.org/pdf/2107.00961.pdf

ON THE CONVERGENCE OF SHALLOW NEURAL NETWORK TRAINING WITH RANDOMLY MASKED NEURONS

Fangshuo Liao, Anastasios Kyrillidis
Department of Computer Science
Rice University
Houston, TX 77005, USA
{Fangshuo.Liao,anastasios}@rice.edu

https://arxiv.org/pdf/2112.02668.pdf

Published at TMLR

GIST: Distributed Training for Large-Scale Graph Convolutional Networks

Cameron Wolfe [*1] Jingkang Yang [*2] Arindam Chowdhury [3] Chen Dun [1] Artun Bayer [3] Santiago Segarra [3]
Anastasios Kyrillidis [1]

https://arxiv.org/pdf/2102.10424.pdf

Under review

Thank you!