49th International Conference on Parallel Processing - ICPP

17-20 August 2020, Edmonton, AB, Canada

Developing a Loss Prediction-based Asynchronous Stochastic Gradient Descent Algorithm for Distributed Training of Deep Neural Networks

Junyu Li[1], Ligang He[1*], Shenyuan Ren[2], Rui Mao[3]

[1] University of Warwick, Coventry, United Kingdom

[2] University of Oxford, Oxford, United Kingdom

[3] Shenzhen University, Shenzhen, China

[*] Corresponding author

Contents

Introduction

- Deep Neural Network (DNN) has shown significant results in image processing [1].
- Increasing trends in Deep Learning:
    - Size of training datasets
    - Architecture complexity of the neural networks
- Limitations in computing infrastructures:
    - Central Processing Unit (CPU): less computing capacities
    - Graphics Processing Unit (GPU): restricted memory size
    - Tensor Processing Unit (TPU): limited generalization
- Distributed training DNNs is a promising strategy, because:
    - Computing in parallel
    - Utilizing abundant computing resources on demand

Introduction

- An example of the distributed training system

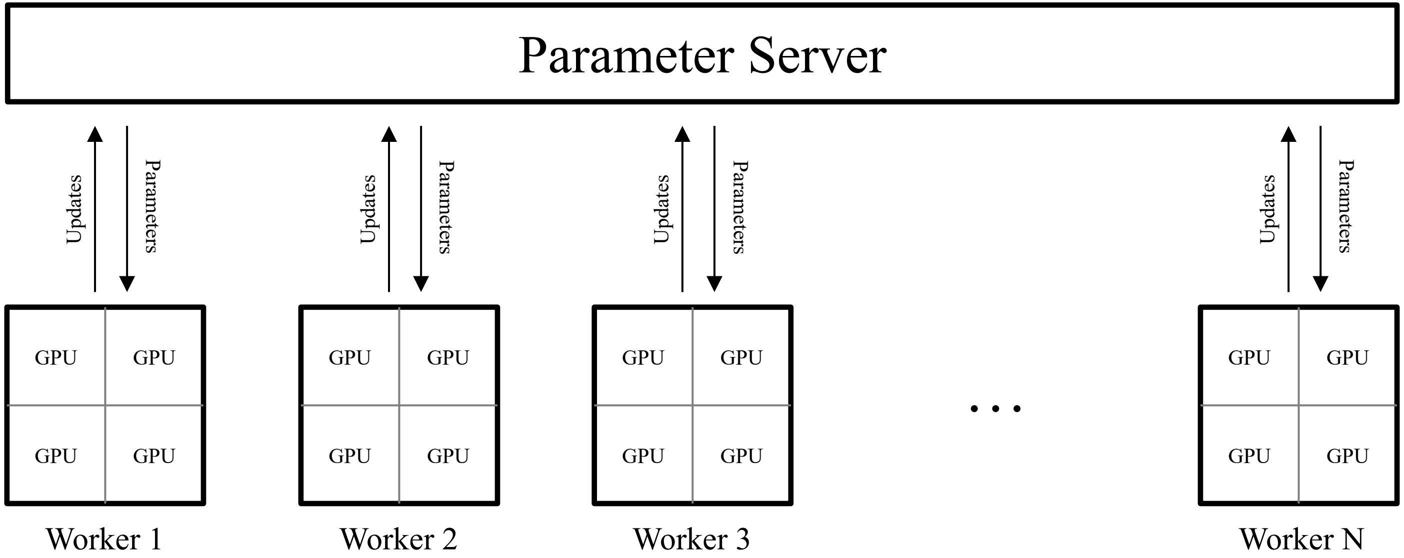


Figure 1. An example of distributed training system

Motivations

- SGD and popular SGD-based optimizers for distributed training:
  - SGD: $\omega_{t+1} = \omega_t - \gamma \cdot \nabla_{\omega_t} \ell(f_{\omega_t}(x_i), y_i)$ Single machine
  - SSGD: $\omega_{t+1} = \omega_t - \gamma \cdot \frac{1}{M} \cdot \sum_{j=1}^{M} (\frac{1}{b} \sum_{i=1}^{b} \nabla_{\omega_t} \ell(f_{\omega_t}(x_{i,j}), y_{i,j}))$ Synchronous barrier
  - ASGD [3,4]: $\omega_{t+\tau+1} = \omega_{t+\tau} - \gamma \cdot g \frac{1}{b} \sum_{i=1}^{b} \nabla_{\omega_t} \ell(f_{\omega_t}(x_i), y_i)$ Delayed update
  - DC-ASGD [5]: $\omega_{t+\tau+1} = \omega_{t+\tau} - \gamma \cdot (g_m + \lambda_t g_m \otimes g_m \otimes (w_t - w_{bak}(m)))$ Remarkable work solving the delay issue in ASGD

Motivations

- The delay issue in ASGD

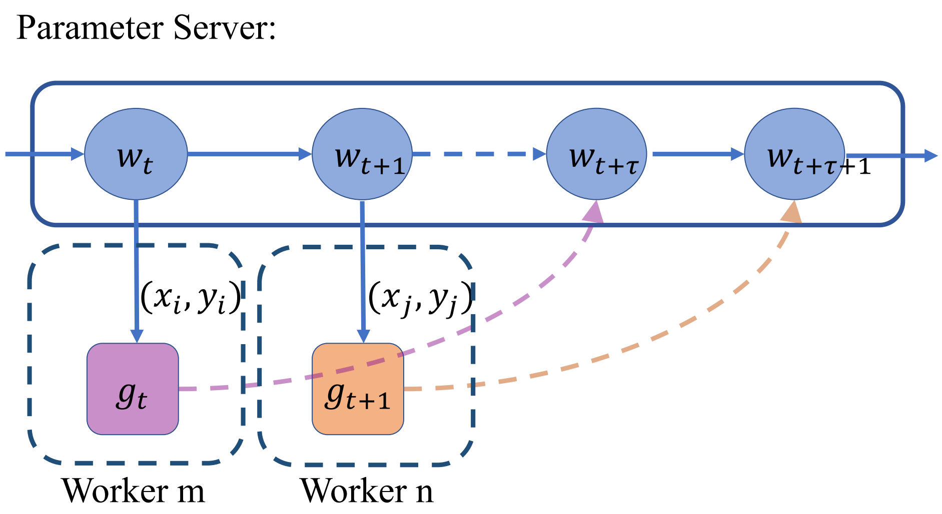


Figure 2. ASGD weight updating procedure

Motivations

- Performance of DC-ASGD with different numbers of workers

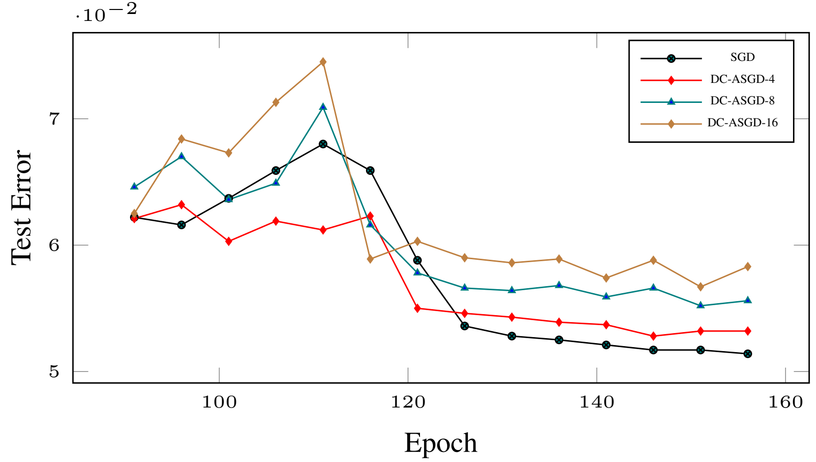


Figure 3. Performance of DC-ASGD training ResNet-18 on CIFAR-10 w.r.t. number of workers

Distributed training with loss compensation

- We propose LC-ASGD to address the delayed updating problem in ASGD.
- The trend of loss values during training is modelled as a time series by a **loss predictor**.
- We use a **step predictor** to model delayed steps for the loss predictor.
- We extend regular **batch normalization** to an **asynchronous** version to further improve the performance.

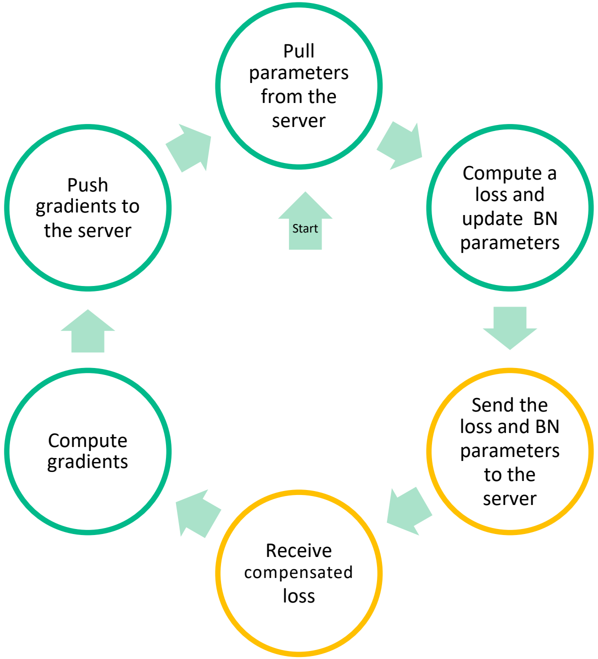
Distributed training with loss compensation

- Training processes of each worker

Algorithm 1 The computations performed by a worker, m

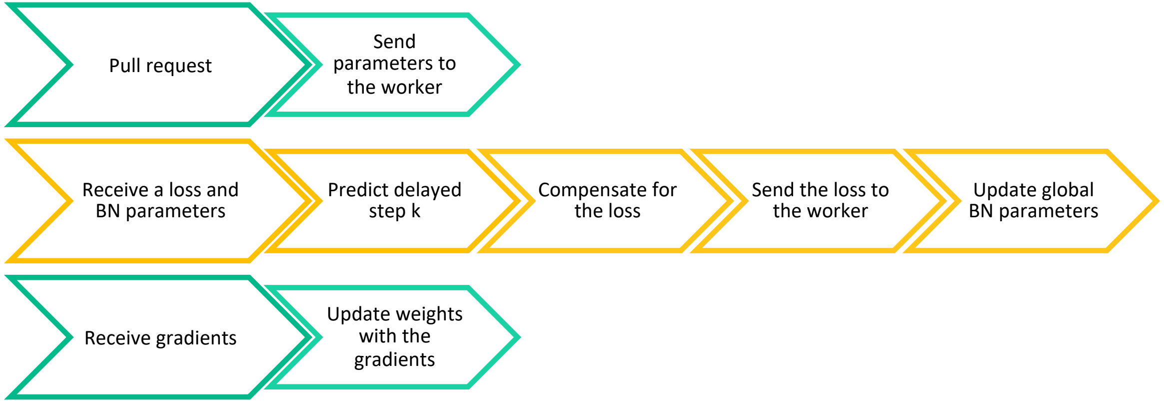
Initialize:
$state_m = \{loss : 0, mean : \{\ \}, var : \{\ \}, t_{comm} : 0, t_{comp} : 0\}$,
$z \in \{1, 2, ..., Z\}$, t_0, t_1

Pull w_t from the parameter server at timestamp t_0
Receive the weights w_t at timestamp t_1
Record the pulling time cost $state_m[t_{comm}] = t_1 - t_0$
Compute loss $\ell_m = \ell(f_{w_t}(x_i),\ y_i)$
Record the local loss $state_m[loss] = \ell_m$
Store mean μ_z in each BN layer bn_z into $state_m[mean]$
Store variance σ_z in each BN layer bn_z into $state_m[var]$
Push all recordings $state_m$ to the parameter server
Receive loss compensation ℓ_{delay} from the parameter server at timestamp t_2
Compute gradient $g_m = \nabla_{w_t}(\ell_m + \lambda \cdot \ell_{delay})$, finishing at timestamp t_3
Record computational time cost $state_m[t_{comp}] = t_3$ - t_2
Push the gradients g_m to the parameter server



Distributed training with loss compensation

- Training processes of the parameter server



Algorithm 2 LC-ASGD: parameter server

Input: learning rate γ

Initialize: $t = 0$, $E_{bn_z} = 0$, $Var_{bn_z} = 1$, w_0 is initialized randomly, $iter = [\ ]$, $m \in \{1, 2, ..., M\}$, $z \in \{1, 2, ..., Z\}$

repeat

- **if** receive $state_m$ **then**
  - Append m to $iter$
  - Predict step $k_m = stepPredictor(m, state_m[t_{comm}, state_m[t_{comp}],\ iter)$
  - Predict loss ℓ_{delay} for the next k_m steps by $lossPred(state_m[loss], k)$
  - Send ℓ_{delay} to worker m
  - Update $E_z = (1 - d) * E_z + d * state_m[mean_z]$
  - Update $Var_z = (1 - d) * Var_z + d * state_m[var_z]$
- **else if** receive g_m **then**
  - $w_{t+1} = w_t - \gamma \cdot g_m$
  - $t = t + 1$
- **else if** receive pull request from worker m **then**
  - Send w_t to worker m
- **end if**

until *forever*

Distributed training with loss compensation

- Loss predictor

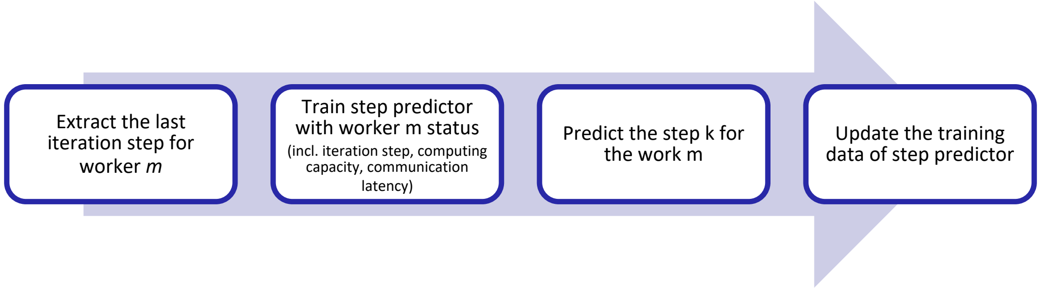
Algorithm 3 LC-ASGD: loss predictor

Input: loss ℓ_m (the loss received from worker m), step k_m
Initialize: ℓ_t (the latest loss of the network)

Train *lossPred* with $(data = \ell_t,\ label = \ell_m)$
$predictions = lossPred(data = \ell_m,\ future = k)$
$\ell_{delay} = sum(predictions)$
$\ell_t = \ell_m$

Return: ℓ_{delay}

- Step predictor



Algorithm 4 LC-ASGD: step predictor

Input: worker rank m, t_{comm}, t_{comp}, iteration recording *iter*
Initialize: $step_m = 0$, t^m_{comm}, t^m_{comp}, $m \in \{1, 2, \ldots, M\}$

Extract the last iteration $step_t$ of worker m from *iter*
Train *stepPred* with
$(data = \{step_m, t^m_{comm}, t^m_{comp}\},\ label = step_t)$
$k_m = stepPred(data = \{step_t, t_{comm}, t_{comp}\},\ future = 1)$
$t^m_{comm}, t^m_{comp}, step_m = t_{comm}, t_{comp}, step_t$

Return: k_m

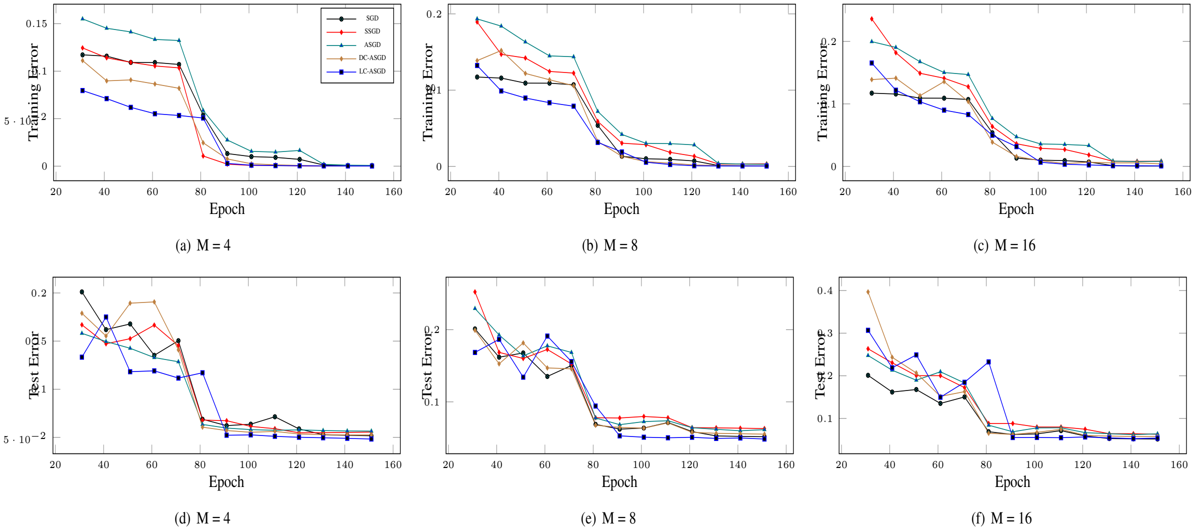
Experiments

- ✓ We evaluated the proposed LC-ASGD on CIFAR-10 and ImageNet benchmark datasets.
- ✓ The experiments were carried out on a GPU cluster equipped with NVIDIA Tesla V100 GPUs.
- ✓ The hyper-parameters followed the settings in the original works [5,6].
- ✓ The results demonstrate that LC-ASGD delivers significant results outperforming other distributed training algorithms.

Experiments

WARWICK

- Learning curves of ResNet-18 with Async-BN on CIFAR-10 dataset



(a) M = 4 (b) M = 8 (c) M = 16

(d) M = 4 (e) M = 8 (f) M = 16

Figure 4. Error rates of the global model ResNet-18 with Async-BN as the training progresses on CIFAR-10

Experiments

- Evaluation results of ResNet-18 on CIFAR-10 dataset

Table 3. Training performance of ResNet-18 on CIFAR-10.

# Workers	Algorithm	CIFAR-10 BN		CIFAR-10 Async-BN	
		Test Error (%)	Perf. Deg. (%)	Test Error (%)	Perf. Deg. (%)
1	SGD	5.15	Baseline	5.15	Baseline
4	SSGD	5.67	10.10	5.57	8.16
	ASGD	5.73	11.26	5.65	9.71
	DC-ASGD	5.33	3.50	5.22	1.36
	LC-ASGD	**4.98**	**-3.3**	**4.87**	**-5.44**
8	SSGD	6.19	20.19	6.01	16.70
	ASGD	6.38	23.88	6.27	21.75
	DC-ASGD	5.72	11.07	5.58	8.35
	LC-ASGD	**5.11**	**-0.78**	**4.96**	**-3.69**
16	SSGD	6.41	24.47	6.20	20.39
	ASGD	6.59	27.96	6.41	24.47
	DC-ASGD	6.05	17.48	5.83	13.20
	LC-ASGD	**5.76**	**11.84**	**5.52**	**7.18**

Experiments

- Learning curves of ResNet-50 with Async-BN on ImageNet dataset

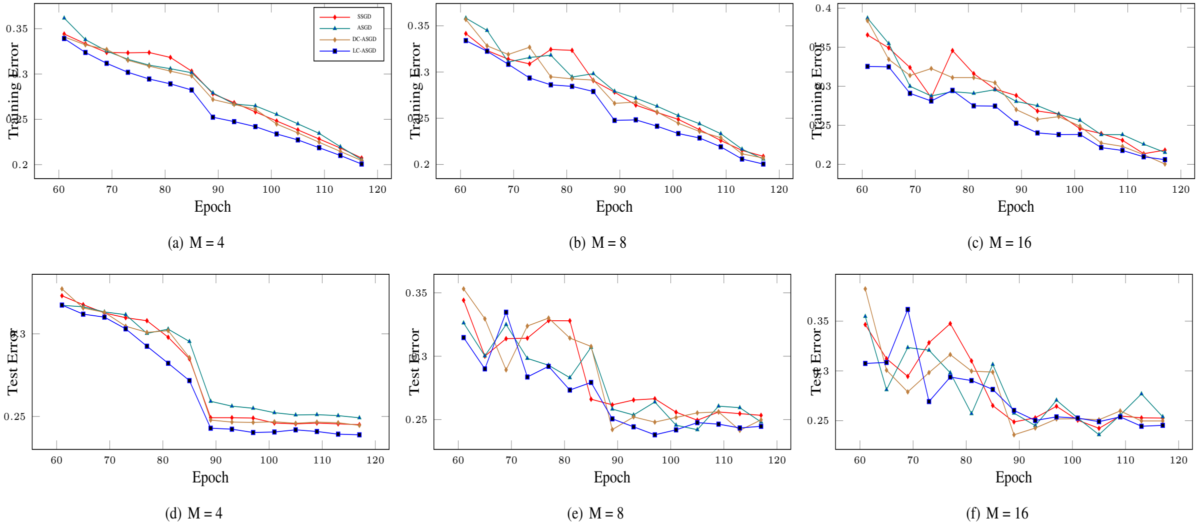


Figure 5. Error rates of the global model ResNet-50 with Async-BN as the training progresses on ImageNet

Experiments

- Evaluation results of ResNet-50 on ImageNet dataset

Table 4. Training performance of ResNet-50 on ImageNet.

# Workers	Algorithm	ImageNet BN		ImageNet Async-BN	
		Test Error (%)	Perf. Deg. (%)	Test Error (%)	Perf. Deg. (%)
4	SSGD	24.61	Baseline	24.49	Baseline
	ASGD	24.99	1.54	24.90	1.67
	DC-ASGD	24.53	-0.33	24.46	-0.12
	LC-ASGD	**23.91**	**-2.84**	**23.86**	**-2.57**
8	SSGD	25.24	2.56	25.11	2.53
	ASGD	25.71	4.47	25.64	4.70
	DC-ASGD	25.98	5.57	24.89	1.63
	LC-ASGD	**24.17**	**-1.79**	**24.07**	**-1.71**
16	SSGD	25.80	4.84	25.62	4.61
	ASGD	25.96	5.49	25.81	5.39
	DC-ASGD	25.41	3.25	25.23	3.02
	LC-ASGD	**24.99**	**1.54**	**24.82**	**1.35**

Experiments

- Performance of the loss predictor and the step predictor

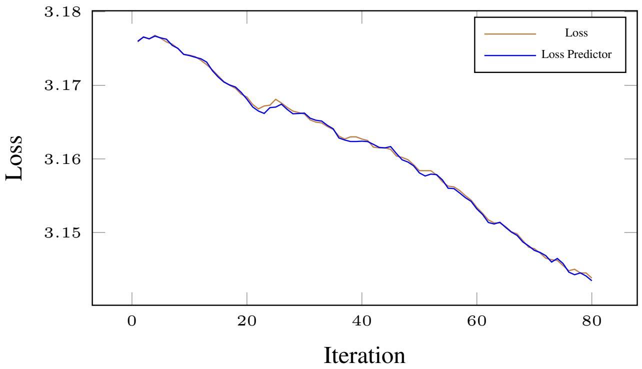


Figure 6. Performance of the loss predictor for ResNet-50 w.r.t. number of iterations on ImageNet training with 16 workers

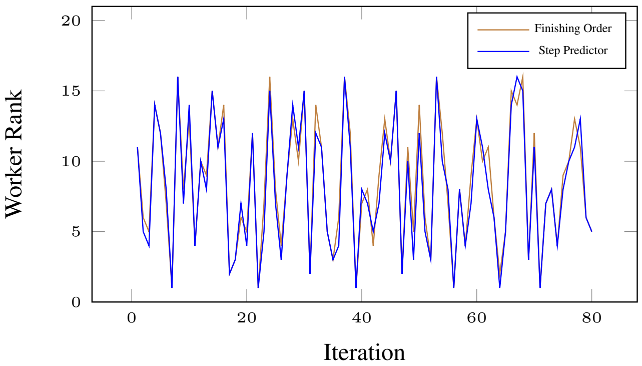


Figure 7. Performance of the step predictor for ResNet-50 w.r.t. number of iterations on ImageNet training with 16 workers

Experiments

- Overhead analysis of training on CIFAR-10 and ImageNet

Table 5. Average time of a training iteration on CIFAR-10.

# Workers	4	8	16
Loss Pred. (ms)	1.28	1.29	1.30
Step Pred. (ms)	1.37	1.43	1.48
Total Training (ms)	32.23	32.84	34.64
Overhead (%)	8.22	8.28	8.03

Table 6. Average time of a training iteration on ImageNet.

# Workers	4	8	16
Loss Pred. (ms)	1.27	1.29	1.33
Step Pred. (ms)	1.36	1.45	1.50
Total Training (ms)	183.23	185.68	188.71
Overhead (%)	1.44	1.48	1.50

Conclusion

- In this work, we discussed:
  - The issue of synchronization barrier in SSGD
  - The delayed gradient updating in ASGD
  - The limitation of DC-ASGD
- We proposed a novel distributed training algorithm with following components:
  - Workers
  - Parameter server with asynchronous batch normalization
  - Loss predictor
  - Step predictor
- Experiment results show that our LC-ASGD delivers outstanding accuracy compared with other algorithms.

References

1. LeCun, Y., Bengio, Y. and Hinton, G., 2015. Deep learning. *Nature*, 521(7553), pp.436-444.
2. Sovrasov, V., 2020. *Sovrasov/Flops-Counter.Pytorch*. [online] GitHub. Available at: <https://github.com/sovrasov/flops-counter.pytorch> [Accessed 6 August 2020].
3. Dean, J., Corrado, G., Monga, R., Chen, K., Devin, M., Mao, M., Ranzato, M.A., Senior, A., Tucker, P., Yang, K. and Le, Q.V., 2012. Large scale distributed deep networks. *In Advances in neural information processing systems* (pp. 1223-1231).
4. Srinivasan, A., Jain, A. and Barekatain, P., 2018. An analysis of the delayed gradients problem in asynchronous sgd.
5. Zheng, S., Meng, Q., Wang, T., Chen, W., Yu, N., Ma, Z.M. and Liu, T.Y., 2017, July. Asynchronous stochastic gradient descent with delay compensation. *In International Conference on Machine Learning* (pp. 4120-4129).
6. He, K., Zhang, X., Ren, S. and Sun, J., 2016. Deep residual learning for image recognition. *In Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

Thank you!