Data Parallelism in Training Sparse Neural Networks

Namhoon Lee¹, Philip Torr¹, Martin Jaggi²

¹University of Oxford, ²EPFL

ICLR 2020  Workshop on PML4DC
Motivation

Compressing neural networks can save a large amount of memory and computational cost.
Compressing neural networks can save a large amount of memory and computational cost.

Network pruning is an effective methodology to compress large neural networks.

Han et al. 2015
Compressing neural networks can save a large amount of memory and computational cost.

Network pruning is an effective methodology to compress large neural networks, but typically requires training steps (Han et al., 2015, Liu et al., 2019, Frankle et al., 2019).
Compressing neural networks can save a large amount of memory and computational cost.

Network pruning is an effective methodology to compress large neural networks, but typically requires training steps (Han et al., 2015, Liu et al., 2019, Frankle et al., 2019).

Pruning can be done at initialization prior to training (Lee et al., 2019, Wang et al., 2020).
Motivation

Compressing neural networks can save a large amount of memory and computational cost.

Network pruning is an effective methodology to compress large neural networks, but typically requires training steps (Han et al., 2015, Liu et al., 2019, Frankle et al., 2019).

Pruning can be done at initialization prior to training (Lee et al., 2019, Wang et al., 2020).

Little has been studied about the training aspects of sparse neural networks (Evci et al., 2019, Lee et al. 2020).
Motivation

Compressing neural networks can save a large amount of memory and computational cost.

Network pruning is an effective methodology to compress large neural networks, but typically requires training steps (Han et al., 2015, Liu et al., 2019, Frankle et al., 2019).

Pruning can be done at initialization prior to training (Lee et al., 2019, Wang et al., 2020).

Little has been studied about the training aspects of sparse neural networks (Evci et al., 2019, Lee et al. 2020).

Our focus ⇒ Data Parallelism on Sparse Networks.
Data parallelism?

It refers to distributing training data to multiple processors and computing gradient in parallel, so as to accelerate training.

The amount of data parallelism is equivalent to the batch size for optimization on a single node.

A centralized, synchronous, parallel computing system.

*It can be a higher-order derivative.
Data parallelism?

It refers to distributing training data to multiple processors and computing gradient in parallel, so as to accelerate training.

The amount of data parallelism is equivalent to the batch size for optimization on a single node.

Understanding the effect of batch size is crucial and an active research topic (Hoffer et al., 2017, Smith et al., 2018, Shallue et al., 2019).

*It can be a higher-order derivative.
Data parallelism?

A centralized, synchronous, parallel computing system.

It refers to distributing training data to multiple processors and computing gradient in parallel, so as to accelerate training.

The amount of data parallelism is equivalent to the batch size for optimization on a single node.

Understanding the effect of batch size is crucial and an active research topic (Hoffer et al., 2017, Smith et al., 2018, Shallue et al., 2019).

Sparse networks can enjoy a reduced memory and communication cost in distributed settings.

*It can be a higher-order derivative.
Steps-to-result

It refers to the lowest number of training steps required to reach a goal out-of-sample error.
Steps-to-result

It refers to the lowest number of training steps required to reach a goal out-of-sample error.

We measure steps-to-result for all combinations of

- workload (data set, model, optimization algorithm)
- batch size (from 1 to 16384)
- sparsity level (from 0% to 90%)

Errors are measured on the entire validation set, at every fixed interval during training.

Our experiments are largely motivated by and closely follow experiments in Shallue et al., 2019.
Steps-to-result

It refers to the lowest number of training steps required to reach a goal out-of-sample error.

We measure steps-to-result for all combinations of

- workload (data set, model, optimization algorithm)
- batch size (from 1 to 16384)
- sparsity level (from 0% to 90%)

Errors are measured on the entire validation set, at every fixed interval during training.

Our experiments are largely motivated by and closely follow experiments in Shallue et al., 2019.

Metaparameters

They refer to parameters whose values are set before the learning begins, such as network size for model, or learning rate for optimization.
Steps-to-result

It refers to the lowest number of training steps required to reach a goal out-of-sample error.

We measure steps-to-result for all combinations of

- workload (data set, model, optimization algorithm)
- batch size (from 1 to 16384)
- sparsity level (from 0% to 90%)

Errors are measured on the entire validation set, at every fixed interval during training.

Our experiments are largely motivated by and closely follow experiments in Shallue et al., 2019.

Metaparameters

They refer to parameters whose values are set before the learning begins, such as network size for model, or learning rate for optimization.

We tune all optimization metaparameters to avoid any assumptions on the optimal metaparameters as a function of batch size or sparsity level.

The optimal metaparameters are selected based on quasi-random search that yield best performance on a validation set.

We perform the search under a budget of trials, while taking into account a predefined search space for each metaparameter.
Data parallelism in training sparse neural networks

Universal scaling pattern across different sparsity:

- perfect scaling
- diminishing returns
- maximal data parallelism
Data parallelism in training sparse neural networks

Universal scaling pattern across different sparsity:

- perfect scaling
- diminishing returns
- maximal data parallelism

Same patterns are observed for different optimizers:

- SGD
- Momentum
- Nesterov
Putting different sparsity together

The higher sparsity, the longer it takes to train.

$\rightarrow$ General difficulty of training sparse networks.
Putting different sparsity together

The higher sparsity, the longer it takes to train. → **General difficulty of training sparse networks.**

The regions of diminishing returns and maximal data parallelism appear at a similar point. → **The effects of data parallelism on sparse network is comparable to the dense case.**
Putting different sparsity together

The higher sparsity, the longer it takes to train.
→ **General difficulty of training sparse networks.**

The regions of diminishing returns and maximal data parallelism appear at a similar point.
→ **The effects of data parallelism on sparse network is comparable to the dense case.**

A bigger critical batch size is achieved with highly sparse networks when using a momentum based SGD.
→ **Resources can be used more effectively.**
Continuing results

Momentum based optimizers are better at exploiting large batch for all sparsity levels.

The data parallelism on sparse networks hold across different workloads.

Our results on sparse networks were unknown and is difficulty to estimate a priori.

More results can be found in the paper.

Comparing SGD, Momentum, and Nesterov optimizers.

CIFAR-10, ResNet-8, Nesterov with a linear learning rate decay.
Summary

- A universal scaling pattern for training sparse neural networks is observed across different workloads.

- Despite the general difficulty of training sparse neural networks, data parallelism on them remains no worse than that on dense networks.

- When training using a momentum based SGD, the critical batch size is often bigger for highly sparse networks than for dense networks.

- Our results render a positive impact on the community, by potentially helping practitioners to utilize resources more effectively.