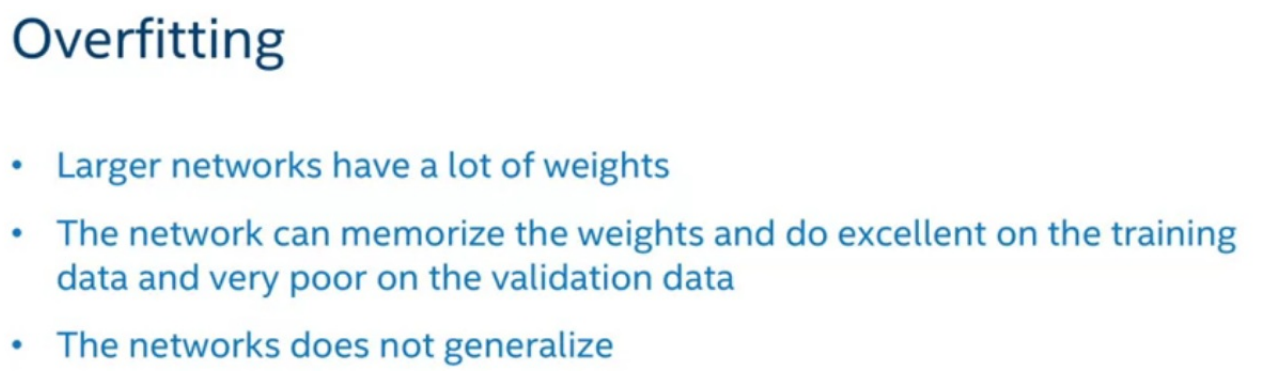
Practical Deep Learning — Intel Course (WEEK — 04) — Training Tips

medium.com/@aakashgoel12/practical-deep-learning-intel-course-week-04-training-tips-d2177782c6f8

July 14, 2020

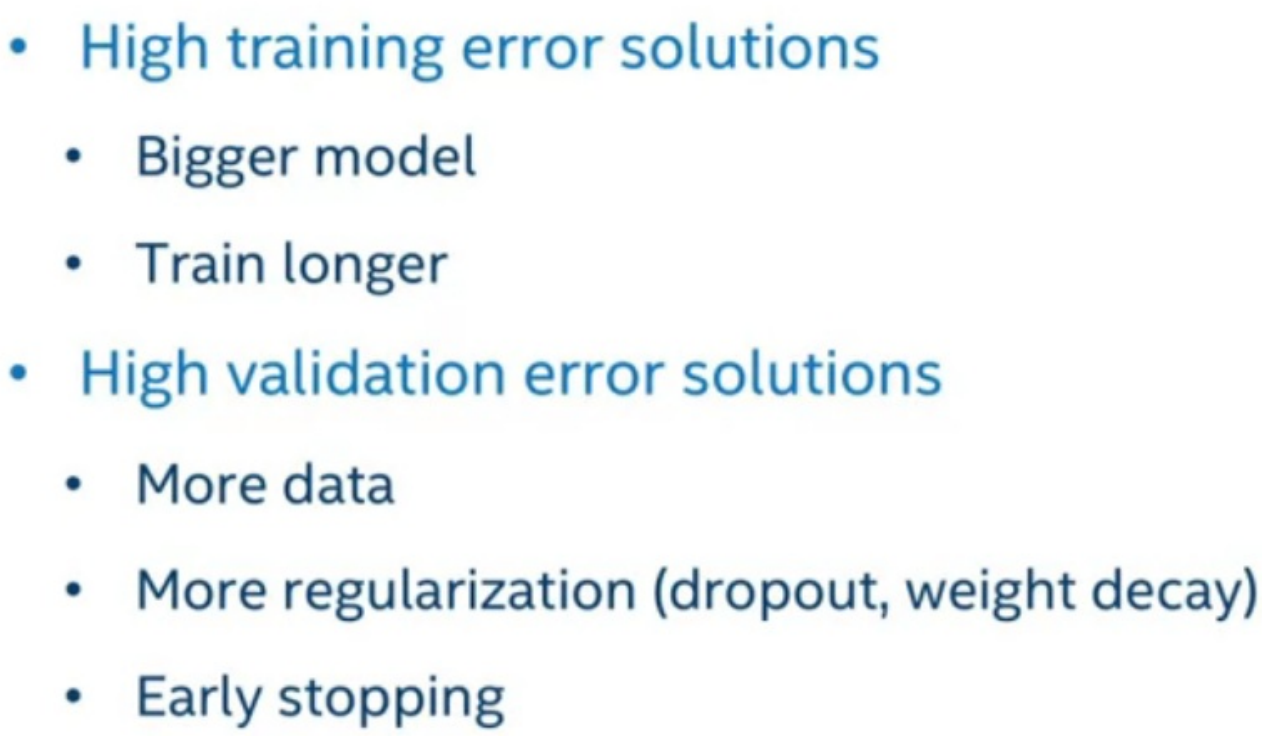
> Overfitting



> Data Augmentation

When training data isn't sufficient to learn generalized model

> Training & Validation Error



> Other Tips

- Use a large model to overfit and then fix ovefitting
- Val and Test data from the same distribution
- Break a task into various pieces
  - Text to speech – Baidu uses 4 networks
  - Autonomous driving – image to stirring wheel + pedal only works in simple scenarios

> **Additional resource**
>
> http://www.deeplearningbook.org/

> **Multi-Node Distributed Training**

- Allows us to efficiently parallelize deep N/W across multiple servers in order to minimize training time
- **Scaling Challenges** — When using SGD, using right batch size is very important. Small batch size → doesn't efficiently use computational resource, Larger batch size →similar issues as of gradient descent (Saddle Point, stuck in sharp minima).
- **Data Vs Model Parallelism —** With very deep networks, trained on large data sets efficiently parallelize n/w across multiple servers becomes essential in order to minimize training time.
- Model Parallelism — Split model weight among N nodes, with each node working on same mini batch. Algo sends same data to all nodes but each node responsible for estimating different parameters
- Data Parallelism — Same model in every node but feed different parts of data and every node independently tries to estimate same parameters, then they exchange their estimates using **parameter server or Allreduce method** . Such method good n/w with fewer weights like GoogleNet. Partition mini batch into N nodes, when back propagates, each worker computes a sum of the weight gradient for each subset of their batch. Weight gradients are then summed across workers, producing identical numerical result as one would find with a single node training methodology.

Which of the following characteristics are true of a data-parallel approach?

- [x] The algorithm distributes the data between various cores

  Correct

- [ ] Each core is responsible for estimating different parameter(s)

  Un-selected is correct

- [x] A data-parallel approach is useful when there are smaller number of nodes in the cluster

  Correct

- [ ] A data-parallel approach is useful when the number of parameters to be estimated is large

  Un-selected is correct

What are some possible issues with asynchronous stochastic gradient descent?

- [x] Asynchronous stochastic gradient descent requires more epochs to train

  Correct

- [ ] Asynchronous stochastic gradient descent does not overcome communication delays

  Un-selected is correct

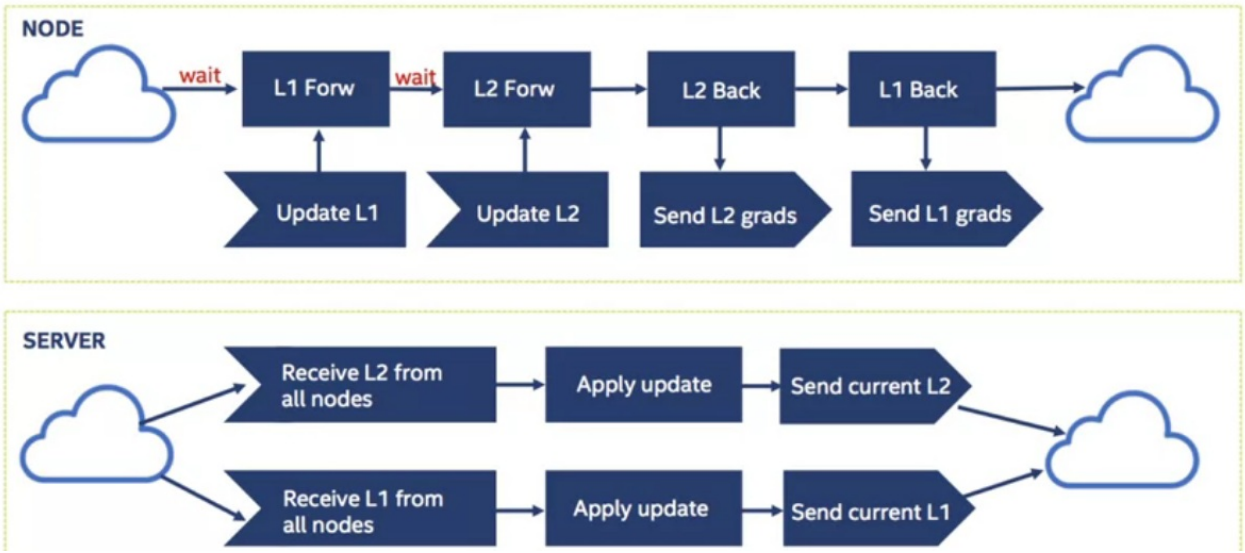
- [x] Asynchronous stochastic gradient descent requires more hypertuning of hyperparameters, including momentum and learning rate

  Correct

- [x] Asynchronous stochastic gradient has not been shown to scale and retain accuracy on large models

  Correct

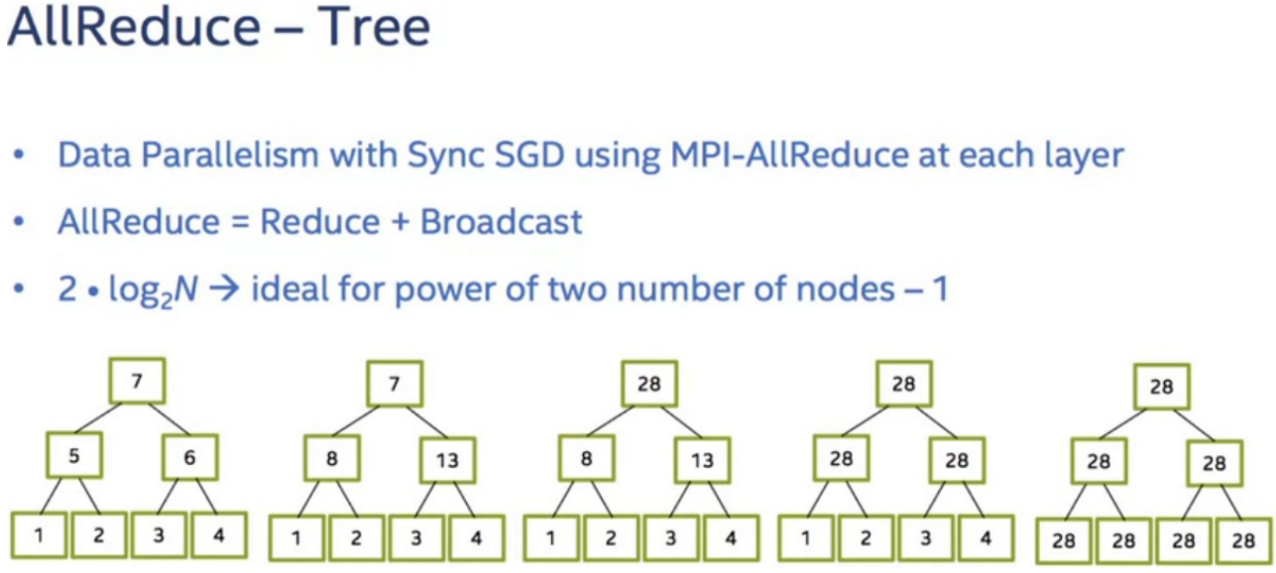
- Total batch size = 1024:
  - 1024 nodes each w/node batch size 1
    (too much communication and too little computation)
  - 16 nodes each w/node batch size 64
    (most communication is hidden in the computation)



> **Strategies for implementing gradient aggregation**

Parameter Server

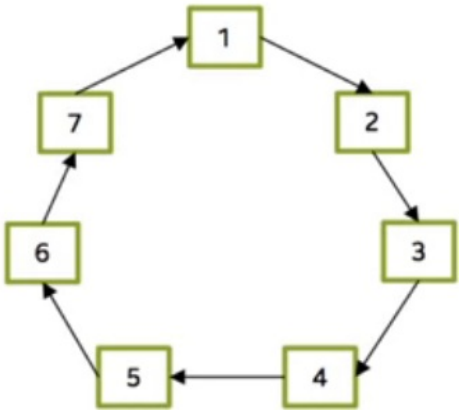
Reduction Tree — Separate reduce & broadcast step



Ring

All Reduce – ring

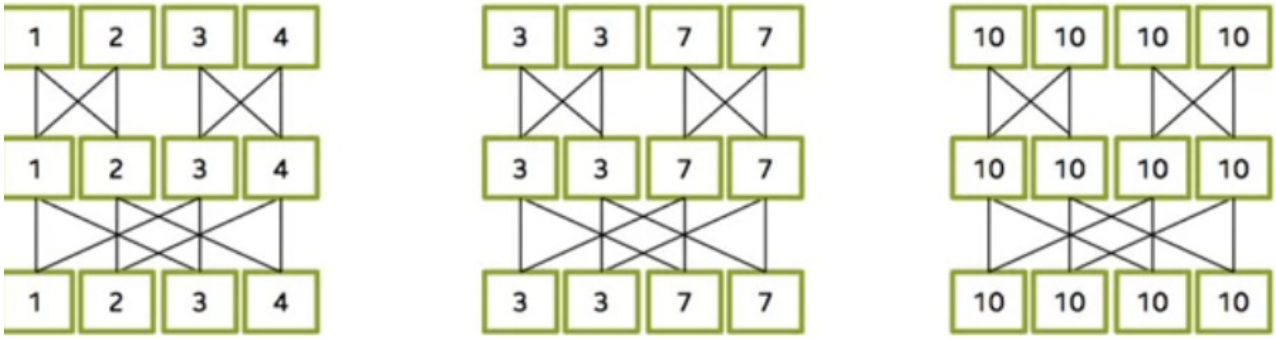
- Baidu uses this
- Efficient use of bandwidth



All reduce Butterfly — Complexity same as Trees but with half no of steps.

All Reduce – butterfly

- $\log_2 N$ steps
- Ideal for power of two number of nodes



> Asynchronous SGD

- Overcomes communication delays but...
- Requires more hypertuning of hyperparameters (momentum, learning rate)
- Require more epochs to train
- Does not match single node performance → harder to debug
- Has not been shown to scale and retain accuracy on large models
- Uses parameter server, e.g., zeroMQ

> Additional resources

FireCaffe: near-linear acceleration of deep neural network training on compute clusters: https://arxiv.org/abs/1511.00175

We present FireCaffe, which successfully scales deep neural network training across a cluster of GPUs. We also present a number of best practices to aid in comparing advancements in methods for scaling and accelerating the training of deep neural networks. The key consideration here is to reduce communication overhead wherever possible, while not degrading the accuracy of the DNN models that we train. When training GoogLeNet and Network-in-Network on ImageNet, we achieve a 47x and 39x speedup, respectively, when training on a cluster of 128 GPUs.

Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour: https://arxiv.org/abs/1706.02677

We empirically show that on the ImageNet dataset large minibatches cause optimization difficulties, but when these are addressed the trained networks exhibit good generalization. Specifically, we show no loss of accuracy when training with large minibatch sizes up to 8192 images. To achieve this result, we adopt a linear scaling rule for adjusting learning rates as a function of minibatch size and develop a new warmup scheme that overcomes optimization challenges early in training. With these simple techniques, our Caffe2-based system trains ResNet-50 with a minibatch size of 8192 on 256 GPUs in one hour, while matching small minibatch accuracy.

Large Batch Training of Convolutional Networks: https://arxiv.org/abs/1708.03888

We propose a new training algorithm based on Layer-wise Adaptive Rate Scaling (LARS). Using LARS, we scaled Alexnet up to a batch size of 8K, and Resnet-50 to a batch size of 32K without loss in accuracy.

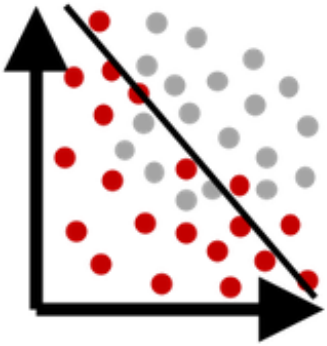
ImageNet Training in Minutes: https://arxiv.org/abs/1709.05011

We finish the 100-epoch ImageNet training with AlexNet in 11 minutes on 1024 CPUs. We finish the 90-epoch ImageNet training with ResNet-50 in 20 minutes on 2048 KNLs without losing accuracy. State-of-the-art ImageNet training speed with ResNet-50 is 74.9% top-1 test accuracy in 15 minutes. We got 74.9% top-1 test accuracy in 64 epochs, which only needs 14 minutes.

Can refer LinkedIn Guy work in Semantic

QUIZ

1. Which of the following terms best describes the following model in relation to the data it models?



- ○ Overfitting
- ◉ Underfitting
- ○ Good fit

2. If a model has high **training** error, which of the following solutions should be employed?

- [x] Bigger model
- [ ] More data
- [x] Train longer
- [ ] More regularization, such as dropout and weight decay

3. Which of the following characteristics are true of a data-parallel approach?

- [x] The algorithm distributes the data between various cores
- [ ] Each core is responsible for estimating different parameter(s)
- [x] A data-parallel approach is useful when there are smaller number of nodes in the cluster
- [ ] A data-parallel approach is useful when the number of parameters to be estimated is large

4. You are training a model on multiple nodes and have set your total batch size to be 1024. You are now trying to decide node batch sizes and the number of nodes to use. Which of the following options is correctly matched with a reasonable justification as to their respective benefits or problems?

- [ ] 2 nodes each with a node batch size of 512: while each node is not computing very much, there is too much communication between nodes, making this inefficient
- [x] 16 nodes each with a node batch size of 64: the communication between nodes is effectively hidden in the computation
- [x] 512 nodes each with a node batch size of 2: there is too much communication and too little computation, making this quite inefficient
- [ ] 2034 nodes each with a node batch size of 1: this would theoretically be ideal, but a batch size of 1 is not possible given hardware constraints