Inventors: Zehra Sura, Parijat Dube, Tong Chen

1. Invention title

Data Driven Mixed Precision Learning for Neural Networks

1. Invention description - Background
2. Problem statement

Training and using neural networks for deep learning problems is time consuming and requires a lot of compute resources. Further, designing and tuning neural networks is an iterative process based on trial-and-error, and this makes it even more imperative to speed up training.

1. Prior art

Reduced precision computing is a technique for faster and more efficient training or inference using neural networks. It has great potential in terms of time and power/energy consumption, and possibly in terms of memory requirements as well. Hardware (such as Intel Nervana chips, NVIDIA Volta GPUs with Tensorcores, Google TPU, and others) and software (e.g. Nervana NEON library, CUDA 9 APIs supporting multiple precisions, and libraries supporting flexible numerical formats) is available to enable the use of reduced precision computing in neural networks.

Besides using reduced precision, prior work has also used mixed precision computing, where certain operations (e.g. the multiplication operations) are performed in reduced precision, and other operations (e.g. accumulation) are performed in higher precision. This is primarily done to counter the negative effect of reduced precision, viz. loss of precision leading to reduced accuracy of trained neural networks.

1. Limitations of prior art

All prior work uses reduced or mixed precision in an identical manner for processing each input data item, which can be either inefficient or have an adverse effect on the accuracy of the trained model. Precision is not customized based on contents of the data item and domain knowledge of the learning problem.

1. Invention description – Summary
2. Main idea

We propose a new method for training and using neural networks to exploit reduced precision computing. The original neural network is replicated into multiple instances, with the instances differing from each other in the precision used for representing and computing the parameters of the network. A pre-processing step is added for the input data, where the input data is analyzed based on the content of the data and the context of the learning problem. The pre-processing step determines the best precision to be used for processing each data item. This determination is then used to route input data items to appropriate instances of the neural network for processing.

1. Advantages

Our method discriminately takes advantage of reduced precision computing for processing select parts of the input data set, which saves time, power/energy, and enables efficient use of available computing resources, without adversely affecting accuracy.

1. Invention description – Details

Reduced precision for training and using neural networks has shown a lot of potential, and it is explicitly supported in commercially available hardware, such as Intel Nervana and NVIDIA Volta GPUs with Tensorcores. However, in many cases, particularly in the training phase, reduced precision leads to a loss in accuracy. Current approaches to overcome this loss include using mixed precision (different precisions for different operations or functions in the overall neural network), finessing the design of the neural network, or reverting to computations using the original precision. Computing platforms generally support multiple precisions, with lower precision computations being more efficient than higher precision computations.

We make the observation that the precision requirement is not uniform for all items in the input data in a particular problem domain. For example, when using images as input data, images of open landscape or scenic vistas have very different properties for the purpose of learning, as compared to images of one or more specific objects. In some cases, lower precision may hurt accuracy, but in other cases, lower precision can facilitate better learning. In general, the relationship between precision used in learning and accuracy of the trained models is unpredictable. However, knowing the context of the learning problem and the contents of the input data, statistical metrics can be devised to determine the precision level that best fits the data item for the purpose of learning from it.

As an example, consider the learning problem of image classification. A possible metric for this is a weighted combination of:

1. a measure of repetitiveness in the image: this can be approximated by the percentage compression ratio that can be achieved using a standard algorithm or compression utility (zip, jpeg).
2. a measure of the smoothness in the image: this can be approximated by computing, for each image point, the sum of distances (differences) of that point and all its neighbors; then taking the mean and standard deviation of the computed values for all points in the image.

In general, depending on what is to be learned from the input images, there is a rich set of image processing algorithms that can be used to extract/emphasize specific properties in the images. Also, related work includes the field of compressive sensing, which explores theoretical basis and metrics for conditions under which under-sampled signals will still convey sufficient information. Appropriate metrics can be applied during a pre-processing step to determine the best precision level for each data input.

A machine learning model (e.g. a neural network) can be trained to determine the precision level that best fits each input data item, as an alternative method of implementing the pre-processing step using pre-defined metrics.

Our method works by replicating the original neural network, with the instances differing from each other in the precision used for representing and computing the parameters of the network. The number of instances depends on the different precisions/numerical formats efficiently supported by the computing platform. For each instance, the number of learners or batch size supported by the instance may also differ depending on available hardware resources or application requirements.

For each input data item, the best level of precision to use for it is determined during pre-processing, and this information is used to route the input data to one of the neural network instances for processing. The instance that uses the same level of precision is chosen, or if such an instance does not exist, then the instance supporting the next higher level of precision is chosen. Optionally, the routing of input data items to an instance can be dynamically determined, based on current processing load of each instance.

During training, each instance of the neural network is independently trained. For inference, the application may use the same set of instances as in the training step. Alternatively, inference may use a computing platform that supports different precision levels, and therefore it may use only a subset of the instances that were trained. Since the replicated instances of the neural network have the same structure, they can be combined into fewer neural networks if need be, using weighted combinations of parameters from the individual instances being merged. The weighting can depend on the precision level of the instance and the number of input data items processed through the instance during training.

Training and possibly inference are typically performed on batches of input data. In our method, we decide which neural network instance to use based on properties of individual input data items. Therefore, when forming batches, properties of each item within a batch need to be considered. Batches can be created statically or dynamically. Further, batches can be constrained to contain data items with the same precision level as the best fit for each item in the batch (homogeneous batches). Or, batches can contain data items with varying precision levels for the best fit (heterogeneous batches). For heterogeneous batches, the choice of neural network instance to use can be based on the maximum precision level for any data item in the batch, or it can be based on the precision level that occurs most often in the batch.

1. Novelty
2. A method for training and using neural networks to exploit reduced precision computing where the original neural network is replicated into multiple instances, with the instances differing from each other in the precision used for representing and computing the parameters of the network, and a data pre-processing step is used to route input data items to appropriate instances for processing.
3. The number of instances is determined based on the levels of precision and numerical formats supported by the targeted computing platform.
4. The instances can differ in batch size or number of learners used.
5. The data pre-processing step can use known metrics, or a machine learning model (such as a trained neural network), to determine the best level of precision for each input data item.
6. The routing of data to appropriate neural network instances can be determined statically based on the results of the pre-processing step, or dynamically by also taking into account the current processing load at each instance.
7. For inference, the trained neural network instances can be combined into fewer neural networks.
8. The combination of neural network instances can be based on a weighted combination of parameters from the instances being merged, where the weighting may depend on the precision level of the individual instances and the number of input data items processed through the individual instances.
9. Batches can be statically or dynamically formed, and batches can be homogeneous or heterogeneous in the precision levels of data items in the batch.
10. For heterogeneous batches, the choice of neural network instance to use can be based on the maximum precision level for any data item in the batch, or it can be based on the precision level that occurs most often in the batch.
11. Background/Related Work
12. Flexpoint: An Adaptive Numerical Format for Efficient Training of Deep Neural Networks, Urs Koster, et. al., Advances in Neural Information Processing Systems 30 (NIPS), 2017

# In-Datacenter Performance Analysis of a Tensor Processing Unit, Norman P. Jouppi, et. al., International Symposium on Computer Architecture (ISCA), 2017

1. NVIDIA Tensorcores, <https://devblogs.nvidia.com/tag/tensor-cores/>
2. Mixed Precision Training of Convolutional Neural Networks using Integer Operations, Dipankar Das, et. al., ICLR 2018
3. Deep Learning with Dynamic Computation Graphs, Moshe Looks, et. al., ICLR 2017
4. Compressive sensing: https://en.wikipedia.org/wiki/Compressed\_sensing