Stochastic Gradient Quantization : Saving Bandwidth in Distributed Learning

Salim Zrouga, Dominik Frey, Emilien Guandalino
École Polytechnique Fédérale de Lausanne, Switzerland

***Abstract*—With the ever-increasing size of machine learning models, and with the development of scale-out architectures, distributed learning has become crucial to exploit the benefits of parallelism. Stochastic Gradient Descent (SGD) is naturally suited for this task as it can leverage the distribution of data. However, the necessary exchange of gradients among workers leads to bandwidth limitations becoming a new performance bottleneck. Quantized SGD (QSGD) addresses this issue through quantization of gradients before their exchange. This report aims to analyze the practical effects on convergence of such lossy compression schemes.**

I. Introduction

Distributed learning provides scalable and elastic compute power for model training. However as the number of nodes increases, communication can saturate bandwidth limitations and cause large overheads. QSGD [1] proposes gradient quantization to trade-off lower bandwidth usage for higher variance during training, and shows that models will still converge to local optima. The paper gives two algorithmic ideas : a stochastic gradient quantization scheme and an efficient encoding of quantized gradients. In this report, we will only be concerned about the first idea and we will analyze the effects of gradient quantization on convergence rates in practice.

We will be presenting the results of training of 3 popular models in different distributed settings and quantization rates. Remarkably, we found that in our applications we can reduce bandwidth usage by at least 4x without any percieved downsides, and up to 8x with minor increase on convergence time. In terms of network accuracy, quantized training converges towards each model's top accuracy, and even sometimes generalizes better on the test set.

II. Technical Background

Quantization is a signal processing technique in which continuous values are mapped to a discrete subset of the value range. This enables values to be encoded in a space efficient manner while preserving most of the original information. This quantization can be deterministic, and map to the closest discrete value, or stochastic and map to discrete values with probability proportional to their distance, see Fig. 1.

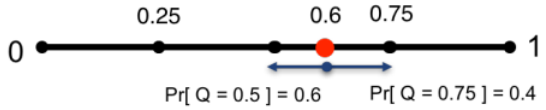

Fig. 1. Stochastic Quantization

QSGD defines a stochastic quantization scheme for s quantization levels of $v \in \mathbb{R}^n$, $v \neq 0$:

$$Q_s(v_i) = \|v\|_2 \cdot \text{sgn}(v_i) \cdot \xi_i(v, s)$$

where $\xi_i(v, s)$'s are independent random variables defined as follows : Let $0 \leq k < s$ be an integer such that $\frac{|v_i|}{\|v\|_2} \in [\frac{k}{s}, \frac{k+1}{s}]$. That is, $[\frac{k}{s}, \frac{k+1}{s}]$ is the quantization interval corresponding to $\frac{|v_i|}{\|v\|_2}$. Then,

$$\xi_i(v, s) = \begin{cases} \frac{k}{s} & \text{with probability } 1 - p\left(\frac{|v_i|}{\|v\|_2}, s\right), \\ \frac{k+1}{s} & \text{otherwise.} \end{cases}$$

With $p(a, s) = as - k$ for $a \in [0, 1]$. If $v = 0$, then we define $Q(v, s) = 0$.

Intuitively, k is the 0-based index to which the value will be mapped in the interval, for example 2 or 3 in Fig. 1. The $\xi_i(v, s)$'s are then the corresponding values at that index, 0.5 or 0.75 on the figure.

Once this quantization is done, we exchange gradients using a simple encoding, for which we estimate bandwidth usage in number of bits sent as such :
For a n dimension vector with $q = log_2(s)$ quantization bits,

$$\text{\# bits} = 32 + n + q * n$$

We send 32 bits for the norm, n bits for the sign vector, and the n quantization indices k of q bits each, from which $\xi_i(v, s)$ can be recovered. QSGD actually proposes a more clever encoding, but this one is simple enough and sufficient for our analysis.

III. Experiments

A. Model Definition

1) Dataset: We will conduct our experiments on an image classification task on the CIFAR-10 dataset. It contains 60,000 colored 32x32 pixel images, with 10 differents classes. We will be using 50,000 samples for training, with a 0.1 validation split (5,000 samples), and the remaining 10,000 samples are used for testing.

2) Models: We train 3 popular models for image classigication : AlexNet, ResNet18 and ResNet110. AlexNet is a large convolutional neural network with 58M parameters.

Short intro to the models **TODO**

We will train our models on 1, 2, 4, and 8 nodes

3) Optimization: As mentioned earlier, SGD is well suited for distributed learning tasks as it can take advantage of the data parallelism. We therefore split our data accross nodes, perform computation per node, and then average out the gradients. Quantization of the gradients is added at each node level before global averaging. We will be presenting results for up to 2, 4 and 8 bit quantization.

4) Error correction: The compression of gradients introduces an approximation error called quantization error. We can mitigate this information loss during training with a process we will call error correction. Upon each iteration, we remember the quantization error and add it to the gradient on the next iteration before quantization. This preserves more information in expectation accross iterations.

5) Evaluation Method: For each model, we first train a baseline model and compare based on the following three metrics :

- *Accuracy :* Do we reach the same top accuracy as the baseline model ?
- *Convergence Time :* How many epochs do we train until we reach top baseline accuracy ?
- *Bandwidth saving estimates :* What ratio of bytes are we saving per iteration per node ? This is based on the basic encoding we defined in the technical background section and could be improved using more clever encondings.

B. Simulating Distributed Training

Distributed training is difficult to implement properly and requires to deal with multiple GPUs, message passing protocols, gradient encodings, and other tricky problems which are outside the scope of this project. We decided to simulate distributed training instead, using a single GPU which sequentially computes forward and backward passes, quantizes gradients, and averages them out on a single model.

A good point is that we are still able to see the negative effects of quantization on convergence rates, in terms of number of iterations. However, a major drawback is that since we are not actually broadcasting the gradients, we cannot truly measure improvements on communication costs. When using quantization for bandwidth reduction we essentially trade-off higher computation time for lower communication time. By balancing out this ratio, we decrease total training time, not in terms of iterations but in real time. This improvement cannot be measured quantitatively in our situation, only roughly estimated by looking at the number of bits saved per node.

C. Results

TODO
Stochastic quantization, Error correction
Explain parameters
number of epochs,
For each model, detail training parameters
Explications of the results here
Explications of the results here
Explications of the results here

Model	Accuracy	Convergence Time	Bandwidth Saving
AlexNet	global	sine like	localized
ResNet18	local	localized	sine like
wavelet	local	localized	sine like

TABLE I
RESULTS FOR 8 BIT QUANTIZATION

Model	Accuracy	Convergence Time	Bandwidth Saving
AlexNet	global	sine like	localized
ResNet18	local	localized	sine like
wavelet	local	localized	sine like

TABLE II
RESULTS FOR 4 BIT QUANTIZATION

D. Discussion

TODO
In our practical applications, is no percievable difference for larger than 8 bits, like 16 bits
Computation overhead
Too agressive quantization hurts accuracy
Communication vs Computation intensive
AlexNet x2 speedup

IV. SUMMARY

TODO
Present results
Limitations -¿ mpi protocols to support no communication costs
following structure:

Abstract
: Short description of the whole paper, to help the reader decide whether to read it.

Introduction
: Describe your problem and state your contributions.

Models and Methods
: Describe your idea and how it was implemented to solve the problem. Survey the related work, giving credit where credit is due.

Results
: Show evidence to support your claims made in the introduction.

Discussion
: Discuss the strengths and weaknesses of your ap-

Model	Accuracy	Convergence Time	Bandwidth Saving
AlexNet	global	sine like	localized
ResNet18	local	localized	sine like
wavelet	local	localized	sine like

TABLE III
RESULTS FOR 4 BIT QUANTIZATION

Fig. 2. Signal compression and denoising using the Fourier basis.

proach, based on the results. Point out the implications of your novel idea on the application concerned.

Summary

Summarize your contributions in light of the new results.

V. Tips for Good Writing

The ideas for good writing have come from

A. Getting Help

One should try to get a draft read by as many friendly people as possible. And remember to treat your test readers with respect. If they are unable to understand something in your paper, then it is highly likely that your reviewers will not understand it either. Therefore, do not be defensive about the criticisms you get, but use it as an opportunity to improve the paper. Before your submit your friends to the pain of reading your draft, please *use a spell checker*.

B. Abstract

The abstract should really be written last, along with the title of the paper. The four points that should be covered

1) State the problem.
2) Say why it is an interesting problem.
3) Say what your solution achieves.
4) Say what follows from your solution.

C. Figures and Tables

Fig. 3. Signal compression and denoising using the Daubechies wavelet basis.

Use examples and illustrations to clarify ideas and results. For example, by comparing Figure 2 and Figure 3, we can see the two different situations where Fourier and wavelet basis perform well.

D. Models and Methods

The models and methods section should describe what was done to answer the research question, describe how it was done, justify the experimental design, and explain how the results were analyzed.

The model refers to the underlying mathematical model or structure which you use to describe your problem, or that your solution is based on. The methods on the other hand, are the algorithms used to solve the problem. In some cases, the suggested method directly solves the problem, without having it stated in terms of an underlying model. Generally though it is a better practice to have the model figured out and stated clearly, rather than presenting a method without specifying the model. In this case, the method can be more easily evaluated in the task of fitting the given data to the underlying model.

The methods part of this section, is not a step-by-step, directive, protocol as you might see in your lab manual, but detailed enough such that an interested reader can reproduce your work

The methods section of a research paper provides the information by which a study's validity is judged. Therefore, it requires a clear and precise description of how an experiment was done, and the rationale for why specific experimental procedures were chosen. It is usually helpful to structure the methods section by

1) Layout the model you used to describe the problem or the solution.
2) Describing the algorithms used in the study, briefly including details such as hyperparameter values (e.g. thresholds), and preprocessing steps (e.g. normalizing the data to have mean value of zero).
3) Explaining how the materials were prepared, for example the images used and their resolution.
4) Describing the research protocol, for example which examples were used for estimating the parameters (training) and which were used for computing performance.
5) Explaining how measurements were made and what calculations were performed. Do not reproduce the full source code in the paper, but explain the key steps.

E. Results

Organize the results section based on the sequence of table and figures you include. Prepare the tables and figures as soon as all the data are analyzed and arrange them in the sequence that best presents your findings in a logical way. A good strategy is to note, on a draft of each table or figure, the one or two key results you want to address in the text portion of the results. The information from the figures is summarized in Table **??**.

When reporting computational or measurement results, always report the mean (average value) along with a measure of variability (standard deviation(s) or standard error of the mean).

VI. Tips for Good Software

There is a lot of literature (for example reader happy:

- Have a `README` file that (at least) describes what your software does, and which commands to run to obtain results. Also mention anything special that needs to be set up, such as toolboxes[1].
- A list of authors and contributors can be included in a file called `AUTHORS`, acknowledging any help that you may have obtained. For small projects, this information is often also included in the `README`.
- Use meaningful filenames, and not `temp1.py`, `temp2.py`.
- Document your code. Each file should at least have a short description about its reason for existence. Non obvious steps in the code should be commented. Functions arguments and return values should be described.
- Describe how the results presented in your paper can be reproduced.

[1]For those who are particularly interested, other common structures can be found at http://en.wikipedia.org/wiki/README and http://www.gnu.org/software/womb/gnits/.

A. LaTeX Primer

LaTeX is one of the most commonly used document preparation systems for scientific journals and conferences. It is based on the idea that authors should be able to focus on the content of what they are writing without being distracted by its visual presentation. The source of this file can be used as a starting point for how to use the different commands in LaTeX. We are using an IEEE style for this course.

1) Installation: There are various different packages available for processing LaTeX documents. See our webpage for more links for getting started.

2) Compiling LaTeX: Your directory should contain at least 4 files, in addition to image files. Images should ideally be `.pdf` format (or `.png`).

3) Equations: There are three types of equations available: inline equations, for example $y = mx + c$, which appear in the text, unnumbered equations

$$y = mx + c,$$

which are presented on a line on its own, and numbered equations

$$y = mx + c \tag{1}$$

which you can refer to at a later point (Equation (1)).

4) Tables and Figures: Tables and figures are "floating" objects, which means that the text can flow around it. Note that `figure*` and `table*` cause the corresponding figure or table to span both columns.

VII. Summary

The aim of a scientific paper is to convey the idea or discovery of the researcher to the minds of the readers. The associated software package provides the relevant details, which are often only briefly explained in the paper, such that the research can be reproduced. To write good papers, identify your key idea, make your contributions explicit, and use examples and illustrations to describe the problems and solutions.

Acknowledgements

Thanks to the authors of QSGD for amazing paper. Thanks to the provided code which we adapted and enabled us to quickly start working on the interesting stuff. Thanks to our corrector for taking the time to read our work.

References

[1] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/6c340f25839e6acdc73414517203f5f0-Paper.pdf