Low Precision Training of Deep Learning Models

Investigating the Impact of Training Precision on Training Time and Classification Accuracy

Romina Mahinpei
mahinpei@student.ubc.ca

Abstract

The adoption of lower precision arithmetic, such as single, half, and quarter precision, has gained traction in both the scientific computing and machine learning communities given its potential for increased speed and reduced resource consumption. While specialized low precision formats and techniques have been explored in machine learning, there remains a gap in evaluating the performance gains of general low precision schemes without relying on specialized formats and techniques. As such, this report investigates the impact of four general low precision schemes on the training time and classification accuracy of four different deep learning models from the domains of image and text classification. After implementing multi-precision and mixed-precision schemes of the selected models, we examine the performance trade-offs between speed and accuracy through our measurements on Google Colab's Tesla T4 GPU. Our findings provide insights into the suitability of general low precision training for deep learning tasks and offer suggestions and benchmarks for machine learning practitioners.

1 Introduction

Most numerical algorithms from the scientific computing community are implemented in double precision, corresponding to a 64-bit floating point number format. Although double precision offers more accuracy by using a larger number of bits to represent numbers, there has been recent interest in incorporating lower precision arithmetic, such as single, half, and even quarter precision, into numerical algorithms (Higham and Mary 2022). The majority of this interest originates from the speed, minimal energy usage, and reduced communication costs that lower precision offers relative to its higher precision counterpart (Higham and Mary 2022). However, the user of lower precision can decrease accuracy because such formats are only accurate to a limited number of digits, thus making them more vulnerable to numerical instabilities and problems like underflow and overflow. As such, to strike a balance between the speed of low precision arithmetic and the accuracy of high precision arithmetic, some studies have started to explore *multi-precision algorithms*, which allow an implementation to be executed in a user-specified precision, as well as *mixed precision algorithms*, which use two or more precisions within the same execution of an implementation (Higham and Mary 2022).

As with the scientific computing community, there have been similar efforts in using low precision arithmetic within machine learning (Wang et al. 2018; Sun, Choi, et al. 2019; Sun, Wang, et al. 2020). Most of these efforts have focused on implementing *specialized low precision formats and techniques* for training deep neural networks. As an example, Wang et al. (2018) developed a specialized 8-bit floating point format and technique to allow low precision computations during training without significant loss in the model's accuracy. Similarly, Sun, Choi, et al. (2019) proposed a hybrid 8-bit floating point format and a round-off residual scheme for training deep learning models in computer vision, speech, and NLP while highlighting a non-significant loss in accuracy.

Course project for UBC CPSC 440/550, April 2024.

Although these efforts hold much promise, there has been no evaluation of whether general low precision schemes without specialized formats and techniques can be used as general-purpose methods for speeding up the training of deep learning models without significantly reducing the quality of the results. The major contribution of our work is to fill this gap by investigating the impact of four general low precision schemes on the training time and classification accuracy of four different deep learning models. As such, we develop a multi-precision implementation (supporting training in either half, single, or double precision) and a mixed-precision implementation (supporting training in a combination of single and half precision) of four deep learning models from the domains of image and text classification. We then investigate the impact of the proposed low precision schemes on the training time and classification accuracy of our chosen models and provide our obtained low precision training times as benchmarks for machine learning practitioners.

The rest of this paper is structured as follows. We first summarize the existing literature on multi-precision and mixed precision algorithms within the two related fields of scientific computing and machine learning in Section 2. We then describe the multi-precision and mixed precision implementations of our selected models in Section 3 before discussing our experimental results in Section 4 and concluding with suggested directions for future work in Section 5.

2 Related Work

2.1 Low Precision Arithmetic in Scientific Computing

The increase in hardware support for low precision arithmetic over the last couple of decades has motivated a transition to multi-precision and mixed precision algorithms in an effort to make the best use of computational resources (Higham and Mary 2022). The reason for this is twofold. The first is that most current GPUs can complete more low precision operations per clock cycle than high precision operations (Buttari et al. 2008). The second is that more low precision data can be held in caches relative to its high precision counterpart, and the low precision data can be moved at a faster rate through the memory hierarchy given the reduced amount of data to be transferred (Buttari et al. 2008).

As such, recent efforts from the scientific computing community have been put towards exploring multi-precision and mixed precision variants of existing iterative algorithms for solving linear systems as outlined in (Higham and Mary 2022). One notable example includes the work done by Carson and Higham (2018), which proposes an iterative refinement algorithm based on the General Minimal Residual (GMRES) method that uses three precisions for solving nonsingular linear systems of the form $Ax = b$. In their work, Carson and Higham (2018) used single precision as their working precision, computed their LU factors in half precision, calculated the residual at each step in double precision, and ultimately showed that it is still possible for the system to be solved to full single-precision accuracy. Similar results were obtained by Higham and Pranesh (2021) when they used low precision Cholesky factors as preconditioners in GMRES and also in the Conjugate Gradient (CG) method.

2.2 Low Precision Arithmetic in Machine Learning

Similar to the field of scientific computing, low precision arithmetic has become widely used in the field of machine learning within the last decade. This rise has primarily been motivated by the experimental observations that algorithms can run faster with certain parts executed in low precision while having little to no deterioration in the quality of the results (Higham and Mary 2022). Although there is little theoretical understanding of these effects, it has been hypothesized that the success of low precision arithmetic in machine learning could be related to the fact that a surrogate problem is often being optimized in place of the true optimization problem (Higham and Mary 2022). Another argument is that low precision arithmetic can provide some degree of regularization that is beneficial to machine learning algorithms (Higham and Mary 2022).

In particular, low precision arithmetic has been used in deep learning because its potential speedup can help address the computational intensity inherent in deep learning models and thus make training more efficient. A notable example includes the work of Wang et al. (2018), which is one of the very first attempts at training a deep neural network in low precision. Wang et al. (2018) specifically devised a specialized 8-bit floating point format along with a technique, called chunk-based computation, to allow computations in low precision without loss in model accuracy. Similarly, Sun, Choi, et al. (2019) proposed a hybrid 8-bit floating point format along with a round-off residual scheme and showed successful training of a collection of deep learning models without accuracy degradation. In a similar manner, Sun, Wang, et al. (2020) developed specialized numerical representation formats and a technique called Gradient Scaling to enable 4-bit training of deep neural networks.

All the works mentioned above achieved successful low precision training using *specialized* floating point formats and techniques, which may not be widely accessible to practitioners within the machine learning community. To fill this gap, our work focuses on *general low precision schemes* that can easily be extended and applied to a wide range of models commonly used by practitioners.

3 Implementations

3.1 Low Precision Schemes

For each of the selected models described in the next section, we implemented multi-precision and mixed precision schemes in an interactive Python notebook using the TensorFlow library. This resulted in the following four general schemes that can be broadly applied for the training of any machine learning model:

- **Scheme 1**: Using the multi-precision implementation to train a model in double precision;
- **Scheme 2**: Using the multi-precision implementation to train a model in single precision;
- **Scheme 3**: Using the multi-precision implementation to train a model in half precision;
- **Scheme 4**: Using the mixed precision implementation to train a model with a combination of half and single precision.

The schemes using the multi-precision implementations achieve the user-specified precision by setting the data type of the operations (`dtype`) involved in the model to the appropriate value. For Schemes 1-3, this requires setting `dtype` to `tf.float64`, `tf.float32`, and `tf.float16` respectively. This then results in the layers' computations being performed in the specified precision and the layers' parameters being stored in the specified precision. On the other hand, Scheme 4 uses the mixed precision implementation, which explicitly sets the `dtype_policy` to `tf.mixed_float16`. This then results in the layers' computations being done in half precision while the layers' variables are stored in single precision.

3.2 Selected Models

To cover a range of classification models in our discussion, we built four different models from the two domains of image classification and text classification using TensorFlow. Our chosen models along with the abbreviations that we will use throughout the remainder of this paper include:

- **Model 1**: Neural network for image classification (NN-I);
- **Model 2**: Convolutional neural network for image classification (CNN-I);
- **Model 3**: Neural network for text classification (NN-T);
- **Model 4**: Recurrent neural network for text classification (RNN-T).

We maintain the multi-precision and mixed precision implementations of all four classification models as an open-source project for others to use for further investigations. We also outline the architecture of these models in Figure 1. Further note that the architectures of these models were based on TensorFlow's recommendations for the datasets described in Section 4.1. We did not further tune the recommended hyper-parameters and model architectures since our focus was on investigating the impact of different training precisions for a *given configuration and model architecture* and not on finding the optimal configuration and architecture.

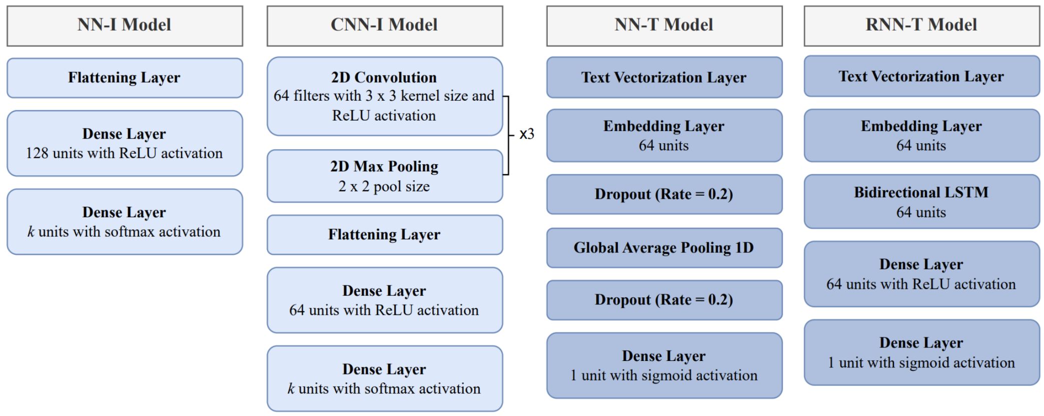

Figure 1: High-level architecture of our four chosen classification models. All the models used an Adam optimizer, and the image classification models (highlighted in light blue) used a categorical cross-entropy loss while the text classification models (highlighted in dark blue) used a binary entropy loss. Further note that the variable k in the image classification models denotes the number of classes.

4 Experiments

4.1 Experimental Setting

Our two image classification models (NN-I and CNN-I) were trained and tested on TensorFlow's Fashion-MNIST dataset consisting of 60,000 training examples and 10,000 test examples. Each example is a 28 by 28 grayscale image, associated with a label from 10 classes. Our two text classification models (NN-T and RNN-T) were trained and tested on TensorFlow's IMDB Reviews dataset consisting of 25,000 training examples and 25,000 test examples for binary sentiment classification. All of our models were trained on Google Colab's Tesla T4 GPU, which is based on the Turing architecture. To account for the non-determinism associated with GPU measurements, all of the reported training times were averaged across three independent training runs for each model.

4.2 Training Time Results

We provide the training time measurements from our experiments in Table 1, with the corresponding bar charts being provided in the first row of Figure 2.

We first observe that **Scheme 1 (multi-precision scheme using double precision) generally had the largest average training time**, with the only exception being in the case of the NN-T model. These results are as expected since fewer double precision operations can be performed on every clock cycle of our GPU relative to lower precision operations. As for the NN-T model, we observe that Scheme 3 (multi-precision scheme using half precision) was significantly slower than all other schemes. One potential explanation for this could be the numerical issues (e.g., underflow and overflow) that resulted from the use of half precision operations, which could have then led to slow convergence or instability during training and thus longer training times. Nonetheless, it is also interesting to note that the accuracy of Scheme 3 for the NN-T model was *not* significantly reduced relative to the other schemes. As such, further investigation is required to better understand the unexpected average training time for the NN-T model when using Scheme 3.

We then observe that **Scheme 2 (multi-precision scheme using single precision) consistently obtained the lowest average training time**. Although it was expected for Scheme 2 to have a faster training time than Scheme 1 (multi-precision scheme using double precision), it was rather surprising to see Scheme 2 *consistently* performed faster than Scheme 3 (multi-precision scheme using half precision) and Scheme 4 (mixed precision scheme using a combination of half and single precision), both of which involved lower precision operations. Comparing Scheme 2 to Scheme 3, the faster performance could be reasoned by the reduced potential for numerical stability, which could have led to faster convergence and shorter training times. Comparing Scheme 2 to Scheme 4, in addition to the

Table 1: Average training time across three runs for the four proposed low precision schemes on the selected classification models.

Model	Scheme 1 Average Training Time (s)	Scheme 2 Average Training Time (s)	Scheme 3 Average Training Time (s)	Scheme 4 Average Training Time (s)
NN-I	3.84E+01	3.08E+01	3.66E+01	3.41E+01
CNN-I	7.35E+01	3.95E+01	4.06E+01	5.91E+01
NN-T	6.30E+01	3.13E+01	2.71E+02	4.21E+01
RNN-T	2.03E+02	1.60E+02	1.81E+02	1.61E+02

Table 2: Classification accuracy on the test set evaluated across the four proposed low-precision schemes for the selected classification models.

Model	Scheme 1 Accuracy	Scheme 2 Accuracy	Scheme 3 Accuracy	Scheme 4 Accuracy
NN-I	87.24%	87.07%	80.89%	87.28%
CNN-I	90.17%	90.26%	10.00%	90.01%
NN-T	83.42%	83.42%	77.47%	83.30%
RNN-T	85.87%	86.17%	77.53%	86.34%

reduced potential for numerical stability, the faster performance could be explained by the absence of additional overhead required for the conversion between half and single precision formats within a mixed precision scheme.

We also note that Scheme 4 (mixed precision scheme using a combination of half and single precision) generally performed better than Scheme 3 (multi-precision scheme using half precision). This could suggest that Scheme 4 is less susceptible to numerical issues than Scheme 3 given its combined usage of half and single precision as opposed to solely relying on half precision.

4.3 Test Set Classification Accuracy Results

We provide the test set classification accuracy results from our experiments in Table 2, with the corresponding bar charts being provided in the second row of Figure 2.

We first observe that **Scheme 2 (multi-precision scheme using single precision) and Scheme 4 (mixed precision scheme using a combination of half and single precision) often had slightly higher test set accuracy results relative to the other schemes**. One potential explanation for this could be that using low precision introduces some degree of *implicit regularization*. Considering that our use of low precision may have introduced some noise into the training process, this may have enabled our selected models to generalize better, overfit the training data less, and ultimately achieve higher accuracy on the test set. Nonetheless, further investigation is required on whether low precision training can indeed act as an implicit regularizer. Although implicit regularization has already been studied with respect to the stochastic gradient descent (SGD) algorithm (Sekhari, Sridharan, and Kale 2021; Zou et al. 2021), not much theory has been developed with respect to low precision training. We leave this as a future line of work worthy of a more thorough exploration.

It is also interesting to note that Scheme 3 (multi-precision scheme using half precision) obtained a significantly worse test set accuracy than all the other schemes for the CNN-I model. During the training of the CNN-I model using Scheme 3, the value of the loss function was observed to be `NaN` from the first epoch and onwards, with this most likely being caused by the vanishing and/or exploding gradients during the forward and backward training passes. These observed numerical instabilities likely interfered with the quality of the training for the CNN-I model using Scheme 3, thus resulting in a low test set accuracy.

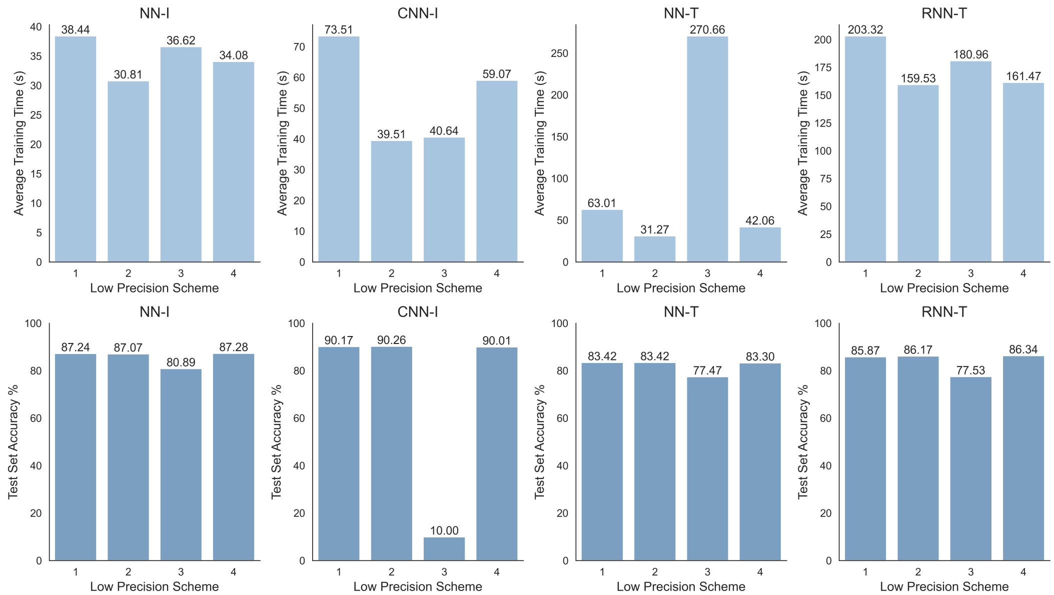

Figure 2: Bar charts displaying the average training time (first row) and test set classification accuracy results (second row) obtained by the selected classification models for each of our four low precision schemes.

5 Conclusion

In this report, we implemented four general low precision schemes for four deep learning models from the domains of image and text classification. We then made performance measurements using Google Colab's Tesla T4 GPU in order to investigate the impact of low precision trading on training time and classification accuracy, with an emphasis on the performance gains that could be obtained by machine learning practitioners through a switch to multi-precision or mixed precision schemes. Our findings indicate that **single precision training generally offers the fastest training time and highest accuracy** compared to our other multi-precision schemes (training in half or double precision) and mixed precision scheme (training using a combination of half and double precision). We especially caution against half precision training due to potential numerical issues but suggest mixed precision as a viable alternative, particularly if hardware optimization for half precision operations is present.

As for future work, we plan to collect measurements for the same schemes and models investigated in this report on different GPU architectures. This will allow us to better understand whether the relatively low performances of Schemes 3 and 4 are related to numerical instabilities arising from the use of half precision or whether it is due to the architecture not being highly optimized for half precision operations. We also plan to investigate whether converting the training data to lower precision has any impact on the training time and classification accuracy. We expect that storing the training data in low precision will lead to faster training times as a result of the faster movement of data through the memory hierarchy and the reduced amount of data to be transferred, but we are hoping to have experimental results to verify this claim. Lastly, we plan to explore the theoretical reasons behind the improved classification accuracy of low precision training, with a focus on whether lower precision training can indeed be viewed as an implicit form of regularization.

References

Buttari, Alfredo, Jack Dongarra, Jakub Kurzak, Piotr Luszczek, and Stanimir Tomov (2008). "Using Mixed Precision for Sparse Matrix Computations to Enhance the Performance while Achieving 64-bit Accuracy." 34.4. ISSN: 0098-3500. DOI: 10.1145/1377596.1377597.

Carson, Erin and Nicholas J. Higham (2018). "Accelerating the Solution of Linear Systems by Iterative Refinement in Three Precisions." *SIAM Journal on Scientific Computing* 40.2, A817–A847. DOI: 10.1137/17M1140819.

Higham, Nicholas J. and Theo Mary (2022). "Mixed precision algorithms in numerical linear algebra." *Acta Numerica* 31, pp. 347–414. DOI: 10.1017/S0962492922000022.

Higham, Nicholas J. and Srikara Pranesh (2021). "Exploiting Lower Precision Arithmetic in Solving Symmetric Positive Definite Linear Systems and Least Squares Problems." *SIAM Journal on Scientific Computing* 43.1, A258–A277. DOI: 10.1137/19M1298263.

Sekhari, Ayush, Karthik Sridharan, and Satyen Kale (2021). "SGD: The Role of Implicit Regularization, Batch-size and Multiple-epochs." *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 27422–27433.

Sun, Xiao, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi (Viji) Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan (2019). "Hybrid 8-bit Floating Point (HFP8) Training and Inference for Deep Neural Networks." *Advances in Neural Information Processing Systems*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett. Vol. 32. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/65fc9fb4897a89789352e211ca2d398f-Paper.pdf.

Sun, Xiao, Naigang Wang, Chia-Yu Chen, Jiamin Ni, Ankur Agrawal, Xiaodong Cui, Swagath Venkataramani, Kaoutar El Maghraoui, Vijayalakshmi (Viji) Srinivasan, and Kailash Gopalakrishnan (2020). "Ultra-Low Precision 4-bit Training of Deep Neural Networks." *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., pp. 1796–1807. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/13b919438259814cd5be8cb45877d577-Paper.pdf.

Wang, Naigang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan (2018). "Training Deep Neural Networks with 8-bit Floating Point Numbers." *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/335d3d1cd7ef05ec77714a215134914c-Paper.pdf.

Zou, Difan, Jingfeng Wu, Vladimir Braverman, Quanquan Gu, Dean P Foster, and Sham Kakade (2021). "The Benefits of Implicit Regularization from SGD in Least Squares Problems." *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 5456–5468. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/2b6bb5354a56ce256116b6b307a1ea10-Paper.pdf.