Speeding up Fourier Neural Operators via Mixed Precision

Anonymous Authors[1]

Abstract

The Fourier neural operator (FNO) is a powerful technique for learning surrogate maps for partial differential equation (PDE) solution operators. For many real-world applications, which often require high-resolution data points, training time and memory usage are significant bottlenecks. While there are mixed-precision training techniques for standard neural networks, those work for real-valued datatypes and therefore cannot be directly applied to FNO, which crucially operates in (complex-valued) Fourier space. On the other hand, since the Fourier transform is already an approximation (due to discretization error), we do not need to perform the operation at full precision. In this work, we *(i)* profile memory and runtime for FNO with full and mixed-precision training, *(ii)* conduct a study on the numerical stability of mixed-precision training of FNO, and *(iii)* devise a training routine which substantially decreases training time and memory usage (up to 27%), with little or no reduction in accuracy, on the Navier-Stokes and Darcy flow equations. Combined with the recently proposed tensorized FNO (Kossaifi et al., 2023), the resulting model has far better performance while also being significantly faster than the original FNO.

1. Introduction

Real-world problems in science and engineering often involve solving systems of partial differential equations (PDEs). These problems typically require large-scale, extremely high-resolution data. For example, meteorologists solve large systems of PDEs every day in order to forecast the weather with reasonable prediction uncertainties (Pathak et al., 2022; Nguyen et al., 2023; Lam et al., 2022). Traditional PDE solvers often require hundreds of compute hours to solve real-world problems, such as climate modeling or 3D fluid dynamics simulations (Jasak, 2009). These problems generally require extreme computational resources and high-memory GPUs.

On the other hand, *neural operators* (Li et al., 2020a; Kovachki et al., 2023; Kossaifi et al., 2023; Li et al., 2021b; 2020b) are a powerful *data-driven* technique for solving PDEs. Neural operators learn maps between function spaces, and they can be used to approximate the solution operator of a given PDE. By training on pairs of input and solution functions, the trained neural operator models are orders of magnitude faster than traditional PDE solvers. In particular, the Fourier neural operator (FNO) (Li et al., 2021b) has been immensely successful in solving PDE-based problems deemed intractable by conventional solvers (Liu et al., 2022; Li et al., 2021c; Gopakumar et al., 2023).

Despite their success, neural operators, including FNO, are still computation- and memory-intensive when faced with extremely high-resolution and large-scale problems. For example, when forecasting 2D Navier-Stokes equations in 1024×1024 resolution, a single training datapoint is 45MB. For standard deep learning models, there is a wealth of knowledge on mixed-precision training, in order to reduce training time and memory usage. However, these techniques are designed for real-valued datatypes and therefore do not directly translate to FNO, whose most expensive operations are complex-valued. Furthermore, learning systems of PDEs often inherently involves learning subtle patterns across a wide frequency spectrum, which can be challenging to learn at half precision, since its range of representation is significantly less than in full precision. On the other hand, the Fourier transform within FNO is already approximated by the discrete Fourier transform (because the training dataset is an approximation of the ground-truth continuous signal); since we already incur approximation error from the discretization, there is no need to run the discrete Fourier transform in high precision.

In this work, we devise a new mixed-precision training routine for FNO which significantly improves runtime and memory usage. We start by profiling the runtime of FNO in full and mixed precision, showing the potential speedups in mixed precision. However, directly running FNO in mixed precision leads to overflow and underflow errors caused by

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the workshop on Synergy of Scientific and Machine Learning Modeling. Do not distribute.

numerical instability, typically manifesting in the first few epochs. To address this issue, we study mixed-precision stabilizing techniques, such as using a pre-activation function before the Fourier transform. We also study the loss caused by aliasing, and we show it can be mitigated by truncating the frequency modes. Based on our study, we devise a new FNO training routine, which includes a `tanh` pre-activation and frequency mode truncation. We test our training routine on the Navier-Stokes and Darcy flow equations, resulting in up to a 27% improvement in runtime and memory with little or no reduction in accuracy.

2. Background and Related Work

Fourier Neural Operator. Many real-world scientific and engineering problems are based on solving partial differential equations (PDEs). Recently, many works have focused on machine learning-based methods to solve PDEs (Gupta et al., 2021; Bhatnagar et al., 2019; Lu et al., 2019; Adler & Öktem, 2017). However, the majority of these methods are based on standard neural networks and are therefore limited to a fixed input and/or output grid, although it is desirable in many applications to have a map between function spaces.

Neural operators are a new technique that addresses this limitation by directly learning maps between function spaces (Li et al., 2021c; 2020a;b; Kovachki et al., 2023). The input functions to neural operators can be in any resolution or mesh, and the output function can be evaluated at any point in the domain; therefore, neural operators are *discretization invariant*. The Fourier neural operator (FNO) (Li et al., 2021b), inspired by the spectral method, is a highly successful neural operator (Wen et al., 2022a; Gopakumar et al., 2023; Li et al., 2021a; Renn et al., 2023; Wen et al., 2022b). We formally describe FNO below.

Given a dataset of pairs of initial conditions and solution functions $\{a_j, u_j\}_{j=1}^N$, which are consistent with an operator $\mathcal{G}(a_j) = u_j$ for all $1 \leq j \leq N$, the goal is to learn a *neural operator* $\mathcal{G}_\theta$ that approximates $\mathcal{G}$. The primary operation in FNO is the Fourier convolution operator, $(\mathcal{K}v_t)(x) = \mathcal{F}^{-1}(R \cdot T_K(\mathcal{F}v_t))(x), \forall x \in D$, where $\mathcal{F}$ and $\mathcal{F}^{-1}$ denote the Fourier transform and its inverse, R denotes a learnable transformation, T_K denotes a truncation operation, and v_t denotes the function at the current layer of the neural operator. In order to implement this operator on discrete data, we use the discrete Fast Fourier Transform (FFT) and its inverse (iFFT). Recent work (Kossaifi et al., 2023) made a number of improvements to the FNO architecture, including Canonical-Polyadic factorization of the weight tensors in Fourier space, which significantly improves performance while decreasing memory usage.

Mixed-Precision Training. Mixed-precision training of neural networks consists of reducing runtime and memory usage by representing input tensors and weights (and performing operations) at lower-than-standard precision. For example, PyTorch (Paszke et al., 2019) has a built-in mixed precision mode which places all operations at `float16` rather than `float32`, with the exception of reduction operations, weight updates, normalization operations, and specialized operations such as ones that are complex-valued.

Mixed-precision training has been well studied for standard neural nets (Zhao et al., 2022; De Sa et al., 2018; Micikevicius et al., 2017; Jia et al., 2018), but has not been studied for FNO. The most similar work to ours is FourCastNet (Pathak et al., 2022), a large-scale climate model that uses mixed precision with Adaptive Fourier Neural Operators (Guibas et al., 2021), however, it is a transformer-based architecture that makes use of 3 000 GPUs, so the technical challenges are different compared to this work.

3. Mixed-Precision FNO

In this section, we first discuss the potential speedups when using mixed-precision FNO, then we address the issue of numerical stability for mixed-precision FNO, and finally, we devise and run our final training pipeline.

Throughout this section, we run experiments on two datasets. The first dataset is the two-dimensional Navier-Stokes equation for a viscous, incompressible fluid on the unit torus. The goal is to predict 5 timesteps into the future (Kossaifi et al., 2023), and there are 10 000 training and 2 000 test samples. The second dataset is the steady-state two-dimensional Darcy flow equation. We learn the operator that maps the diffusion coefficient to the solution, and there are 5 000 training and 1 000 test samples. The resolution for both datasets is 128×128. See Kossaifi et al. (2023) for more details about these datasets.

3.1. Potential Speedup of Mixed-Precision FNO

We compare the runtime of FNO in full and mixed-precisions. First, we run mixed-precision via Automatic Mixed Precision (AMP)(Paszke et al., 2019). It places all operations into half precision (from `float32` to `float16`) with a few exceptions: reduction operations, such as computing the sum or mean of a tensor; weight updates; normalization layers; and operations involving complex numbers. The first three exceptions are due to the additional precision needed for operations that involve small numbers or small differences among numbers; the last one is due to the lack of support for half-precision complex datatype in PyTorch (Paszke et al., 2019). Therefore, we devise a custom half-precision implementation of the FNO block for complex-valued operations, which includes half-precision FFT, tensor contraction, and inverse FFT.

We find that AMP and half-precision-Fourier operations

Table 1. Comparison of different pre-activation functions used for numerical stability. `tanh` achieves the fastest runtime. Since our focus now is on numerical stability, we compare only the train loss (we compare the test losses in Section 3.3).

Fourier Precision	AMP (T/F)	Pre-Activation	Runtime per epoch	Train loss
Full	F	N/A	44.4	0.0471
Full	T	N/A	42.3	0.0454
Half	F	N/A	N/A	N/A
Half	T	`hard-clip`	37.1	0.0478
Half	T	2σ`-clip`	40.0	0.0473
Half	T	`tanh`	36.5	0.0489

result in 8.4% and 19.1% speedups, respectively, and together result in a 28.5% speedup, based on training time per epoch on a V100 GPU on the Navier Stokes dataset described above. We see a similar percentage reduction for memory usage. We also find that the speedup for AMP+half-precision-Fourier on the Darcy flow dataset is 17%. However, running in mixed precision results in numerical instability, especially for the forward and inverse FFTs. We find that the forward FFT causes overflow within only a few epochs. This results in NaN losses and the end of training. In the next section, we consider stabilizing techniques.

3.2. Stabilizing Mixed-Precision FFT

Mixed-precision training is prone to underflow and overflow because the dynamic range of half precision is significantly smaller. We empirically show that naïve methods such as scaling, normalization, and alternating half and full precision do not fix the numerical instability. In particular, a simple idea is to scale down all values by adding a fixed pointwise division operation before the FFT. However, this does not work: due to the presence of outliers, all values must be scaled down by significant amount (a factor 10^4). This forces all numbers into a very small range, which half precision cannot distinguish, preventing the model from converging. We also find that changing the normalization of the Fourier transform (essentially equivalent to scaling) does not fix the numerical instability.

We also find normalization simply scales by the variance of the batch or layer, which faces the same problems as above. And due to the stochastic yet consistent nature of the overflow errors, alternating between half and full precision also does not mitigate overflow.

Finally, we show that pre-activation is a very effective method for overcoming numerical instability. Pre-activation is the practice of adding a nonlinearity before a typical 'main' operation such as a convolution (He et al., 2016). Unlike scaling and normalization, pre-activation such as `tanh` forces all values within the range $[-1, 1]$ while being robust to outliers: the shape of `tanh` maintains the trends in the values near the mean.

Table 2. Comparison of the full-precision and mixed-precision (`AMP+HALF+TANH`) FNO with different frequency modes.

Precision	Modes	Runtime	Train loss	Test loss
Full	64	44.4	0.0451	0.0432
Full	32	35.8	0.0460	0.0447
Full	16	34.3	0.0670	0.0690
Mixed	64	36.5	0.0489	0.0505
Mixed	32	29.7	0.0486	0.0474
Mixed	16	28.3	0.0682	0.0689

We observe that only adding a pre-activation before the first FFT still results in overflow in the next FFT. Therefore, we add a pre-activation function before every forward FFT in the architecture. We run AMP+half-precision-Fourier with three pre-activation functions: `tanh`, `hard-clip`, and 2σ`-clip`. For this ablation study, we train each model for 100 epochs using the default hyperparameters from the FNO architecture (Li et al., 2021b; Kossaifi et al., 2023); see Table 1. We find that `tanh` is the most promising pre-activation overall: it is the fastest, its loss is not significantly different from the other pre-activations, and it is not lossy (unlike the two clipping pre-activations), and therefore more promising in the full training pipeline of 500 epochs.

Frequency Mode Truncation. Now we study frequency mode truncation in mixed precision. When solving systems of PDEs, typically the most important information in the input function is in the low frequency modes. The higher frequencies are typically noisier and must be truncated to prevent overfitting (Li et al., 2021b). We show that in half precision, with a lower range of representable numbers, it helps to truncate more frequency modes than in full precision. We run the full-precision FNO and the mixed-precision FNO from the previous section (`AMP+HALF+TANH`) across three frequency modes: 64, 32, and 16, and we otherwise keep the same experimental setup as in Section 3.2. Using 16 modes does not yield good results for either model, and we find that the decrease in accuracy when going from 64 to 32 modes is significantly less for mixed-precision FNO, compared to full-precision FNO. In Appendix A, we also run an experiment on synthetic data to demonstrate that the error caused by half precision is higher for higher frequencies, relative to the amplitude.

3.3. Final Training Pipeline

Based on the results from the previous sections, we compare full-precision FNO to mixed-precision FNO with `tanh` pre-activation, on the full training pipeline consisting of 500 epochs. We run this experiment in a variety of settings: on both NVIDIA V100 and RTX3090 TI GPUs, with both 32

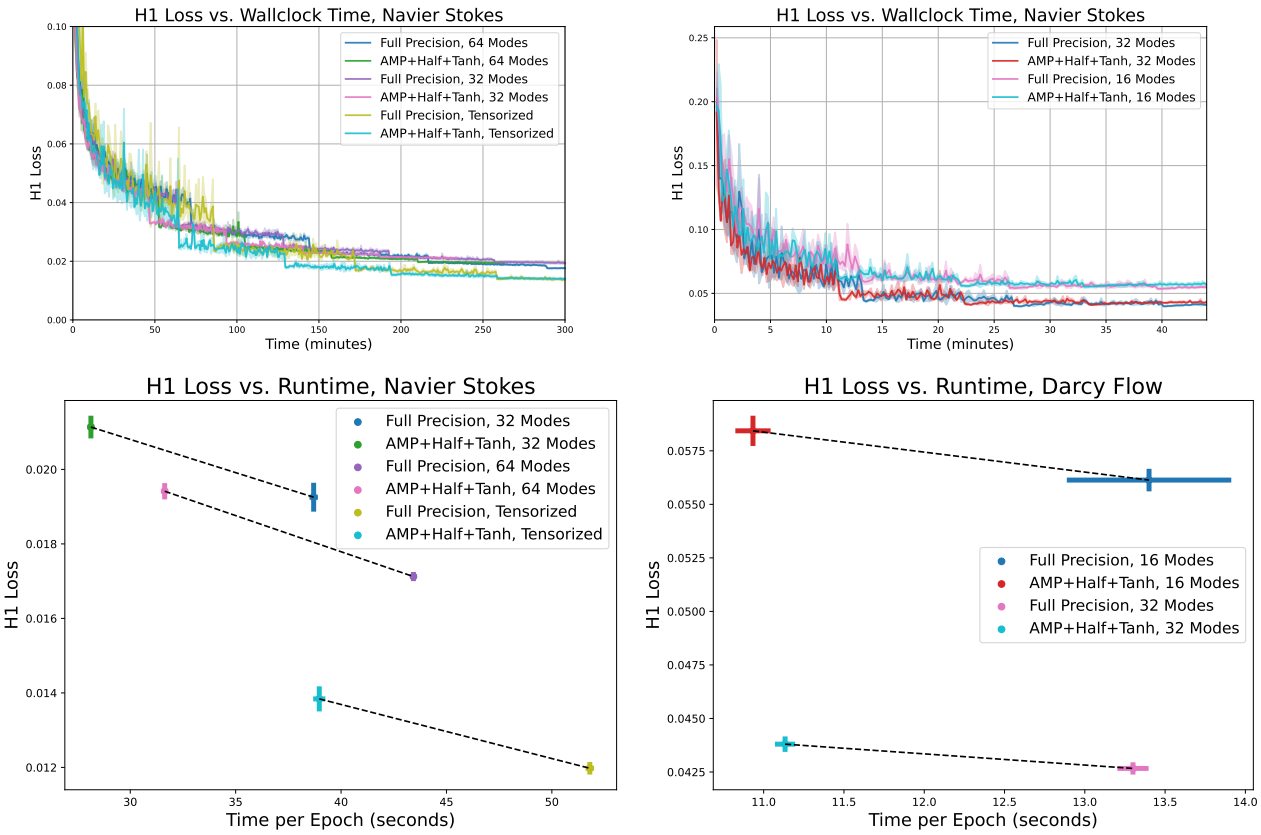

Figure 1. Test H1 error curves for full-precision FNO and `AMP+HALF+TANH` FNO, on the Navier Stokes (top left) and Darcy flow (top right) datasets. Pareto frontier for full-precision FNO and `AMP+HALF+TANH` FNO, each with both 32 and 64 frequency modes, on the Navier Stokes (bottom left) and Darcy flow (bottom right) datasets. For each experiment, the standard deviation is plotted from 3 trials.

and 64 frequency modes, and with and without Canonical-Polyadic factorization of the weight matrices (which was recently shown to substantially improve performance and memory usage of FNO (Kossaifi et al., 2023)). See Figure 1 (left) for results on the Navier Stokes dataset. For all of these settings, naïvely running mixed-precision results in NaN's after a few epochs, but adding `tanh` pre-activation allows the model to converge much faster and with nearly the same final performance. `AMP+HALF+TANH` achieves a 25-27% reduction in runtime per epoch on a V100 and 20-21% reduction in runtime per epoch on an RTX3090. The final loss after 500 epochs increases by 6-11%, but the *anytime performance* of `AMP+HALF+TANH` is better. We also note that the learning rate schedule and other hyperparameters were optimized for full-precision Navier Stokes in prior work, which suggests that additional hyperparameter tuning might lead to even better performance in mixed precision.

We also test our method on the Darcy flow dataset described in Section 3. Similar to the Navier Stokes dataset, we find that simply running the FFT in half precision leads to numerical instability, but running `AMP+HALF+TANH` gives an 18% reduction in runtime with no increase in loss. See Figure 1 (right). This confirms that our findings from Section 3.2 can generalize to other datasets.

4. Conclusions and Future Work

In this work, we studied the numerical stability of half-precision training for FNO, and we devised a new training routine which results in a significant improvement in runtime and memory usage. Specifically, we showed that using `tanh` pre-activation before the Fourier transform mitigates numerical instability. We also showed that the range of half precision is too small to learn high frequency modes, and therefore, reducing the learnable frequency modes also helps performance. We show that with these modifications, running FNO in half precision results in up to a 27% reduction in runtime and memory, with little to no decrease in accuracy, on the Navier Stokes and Darcy flow equations. Running FNO on real-world applications that require super resolution is an exciting next step, because half-precision FNO makes it possible to train on datapoints that are 27% larger, with the same batch size.

Broader Societal Impact Statement

We do not foresee any strongly negative impacts, in terms of the broader societal impacts of this work. In fact, this work may help to reduce the carbon footprint of scientific computing experiments.

Recently, the overall daily amount of computation used for machine learning has been increasing exponentially (Schwartz et al., 2020). Relatedly, model sizes are also increasing fast (OpenAI, 2018). Leveraging the power of mixed-precision computing for neural operators can make it possible to design better models for important tasks such as climate modeling (Pathak et al., 2022) and carbon storage (Wen et al., 2022b).

References

Adler, J. and Öktem, O. Solving ill-posed inverse problems using iterative deep neural networks. *Inverse Problems*, 33(12):124007, 2017.

Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., and Kaushik, S. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64:525–545, 2019.

De Sa, C., Leszczynski, M., Zhang, J., Marzoev, A., Aberger, C. R., Olukotun, K., and Ré, C. High-accuracy low-precision training. *arXiv preprint arXiv:1803.03383*, 2018.

Gopakumar, V., Pamela, S., Zanisi, L., Li, Z., Anandkumar, A., and Team, M. Fourier neural operator for plasma modelling. *arXiv preprint arXiv:2302.06542*, 2023.

Guibas, J., Mardani, M., Li, Z., Tao, A., Anandkumar, A., and Catanzaro, B. Adaptive fourier neural operators: Efficient token mixers for transformers. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Gupta, G., Xiao, X., and Bogdan, P. Multiwavelet-based operator learning for differential equations. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 34:24048–24062, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pp. 630–645. Springer, 2016.

Jasak, H. Openfoam: open source cfd in research and industry. *International Journal of Naval Architecture and Ocean Engineering*, 1(2):89–94, 2009.

Jia, X., Song, S., He, W., Wang, Y., Rong, H., Zhou, F., Xie, L., Guo, Z., Yang, Y., Yu, L., et al. Highly scalable deep learning training system with mixed-precision: Training imagenet in four minutes. *arXiv preprint arXiv:1807.11205*, 2018.

Kossaifi, J., Kovachki, N. B., Azizzadenesheli, K., and Anandkumar, A. Multi-grid tensorized fourier neural operator for high resolution PDEs, 2023. URL https://openreview.net/forum?id=po-oqRst4Xm.

Kovachki, N., Li, Z., Liu, B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Learning maps between function spaces with applications to pdes. *Journal of Machine Learning Research*, 24(89):1–97, 2023. URL http://jmlr.org/papers/v24/21-1524.html.

Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Pritzel, A., Ravuri, S., Ewalds, T., Alet, F., Eaton-Rosen, Z., et al. Graphcast: Learning skillful medium-range global weather forecasting. *arXiv preprint arXiv:2212.12794*, 2022.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020a.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Stuart, A., Bhattacharya, K., and Anandkumar, A. Multipole graph neural operator for parametric partial differential equations. *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 33:6755–6766, 2020b.

Li, Z., Kovachki, N., Azizzadenesheli, K., Liu, B., Bhattacharya, K., Stuart, A., and Anandkumar, A. Learning dissipative dynamics in chaotic systems. *arXiv preprint arXiv:2106.06898*, 2021a.

Li, Z., Kovachki, N. B., Azizzadenesheli, K., Bhattacharya, K., Stuart, A., Anandkumar, A., et al. Fourier neural operator for parametric partial differential equations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Li, Z., Zheng, H., Kovachki, N., Jin, D., Chen, H., Liu, B., Azizzadenesheli, K., and Anandkumar, A. Physics-informed neural operator for learning partial differential equations. *arXiv preprint arXiv:2111.03794*, 2021c.

Liu, B., Kovachki, N., Li, Z., Azizzadenesheli, K., Anandkumar, A., Stuart, A. M., and Bhattacharya, K. A learning-based multiscale method and its application to inelastic impact problems. *Journal of the Mechanics and Physics of Solids*, 158:104668, 2022.

Lu, L., Jin, P., and Karniadakis, G. E. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017.

Nguyen, T., Brandstetter, J., Kapoor, A., Gupta, J. K., and Grover, A. Climax: A foundation model for weather and climate. *arXiv preprint arXiv:2301.10343*, 2023.

OpenAI. Ai and compute. https://openai.com/blog/ai-and-compute/, 2018.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. Pytorch: An imperative style, high-performance deep learning library. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022.

Renn, P. I., Wang, C., Lale, S., Li, Z., Anandkumar, A., and Gharib, M. Forecasting subcritical cylinder wakes with fourier neural operators. *arXiv preprint arXiv:2301.08290*, 2023.

Schwartz, R., Dodge, J., Smith, N. A., and Etzioni, O. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

Wen, G., Li, Z., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. U-fno—an enhanced fourier neural operator-based deep-learning model for multiphase flow. *Advances in Water Resources*, 163:104180, 2022a.

Wen, G., Li, Z., Long, Q., Azizzadenesheli, K., Anandkumar, A., and Benson, S. M. Accelerating carbon capture and storage modeling using fourier neural operators. *arXiv preprint arXiv:2210.17051*, 2022b.

Zhao, J., Dai, S., Venkatesan, R., Zimmer, B., Ali, M., Liu, M.-Y., Khailany, B., Dally, W. J., and Anandkumar, A. Lns-madam: Low-precision training in logarithmic number system using multiplicative weight update. *IEEE Transactions on Computers*, 71(12):3179–3190, 2022.

A. Additional Details from Section 3

We run an experiment on synthetic data to demonstrate that the error caused by half precision is higher for higher frequencies, relative to the amplitude.

We conduct a synthetic experiment as follows. We create a signal based on sine and cosine waves with frequencies from 1 to 10, with randomly drawn, exponentially decaying amplitudes. Then we plot the Fourier spectrum in full and half precision, as well as the absolute error of the half-precision spectrum as a percentage of the true amplitudes. See Figure 2; we find that the percentage error exponentially increases.

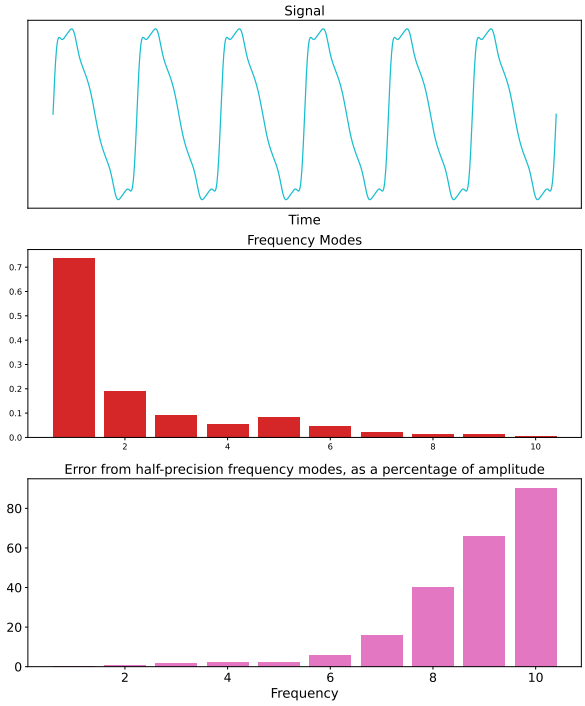

Figure 2. Synthetic signal (top), its frequency modes (middle), and the error due to half precision, as a percentage of the amplitude (bottom). The percentage error increases for higher frequencies.