On the impact of larger batch size in the training of Physics Informed Neural Networks

Anonymous Author(s)
Affiliation
Address
email

Abstract

Physics Informed Neural Networks (PINNs) have demonstrated remarkable success in learning complex physical processes such as shocks and turbulence, but their applicability has been limited due to long training times. In this work, we explore the potential of large batch size training to save training time and improve final accuracy in PINNs. We show that conclusions about generalization gap brought by large batch size training on image classification tasks may not be compatible with PINNs. We conclude that larger batch sizes always beneficial to training PINNs.

1 Introduction

The advance of computation hardware technologies in the last decade has enabled researchers to explore the possibility of extremely large scale training of neural networks for image based tasks such as classification and segmentation. In the course of 3 years, from 2016 to 2019, the training time of the ResNet50 [1] architecture for ImageNet[2] classification task has been reduced from 29 hours[1] in the original paper to 1 minute [3], while maintaining comparable Top-1 testing accuracy, by using various techniques such as learning rate warm up and layer-wise gradient normalization[4, 5]. However, other studies[6, 7] suggest that training with extremely large batch size often suffer from poor generalization performance on the test set, possibly due to the discrepancy between the loss landscapes on the train data and test data. *Keskar et al.*[6] numerically show that the training loss landscape, linearly parameterized around the local minima, is often "flatter", and matches that of the test loss, when using smaller batch size. They, therefore, pose a conjecture that a larger batch size leads to "sharper" local minimum in the training loss landscape that results in poor generalization accuracy. In [8], the authors empirically validate on ImageNet the assumption that decreasing learning rate and increasing batch size have similar effects on training and suggest that under the current parallelization hardware framework, larger batch sizes should be preferable due to its superior efficiency.

While numerous works[6, 9, 8] have studied the effect of batch size on Convolutional Neural Networks (CNNs) and image-based classification tasks, no such studies have been carried out for physics-informed machine learning [10]. In this paper, we extend such analysis onto Physics Informed Neural Networks (PINNs)[11], a framework for learning representations of the solutions to non-linear partial differential equations (PDEs). Unlike the ImageNet classification task, where the train and test data are finite, the input data for PINNs is often sampled uniformly within the interior and on the boundary of the domain. Hence, we don't have a limitation on the amount of data we can use during training and each training iteration is carried out using a set of unique points. Additionally, PINNs are trained in a self-supervised manner where we the train the network to satisfy certain boundary conditions and an underlying PDE, which directly contrasts with image classification problems. Furthermore, in PINNs, the evaluation metric is often calculated on test points that are also evenly sampled across the

Submitted to the DLDE Workshop in the 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

domain, leading to the same train and test data distributions. This means that the generalization gap as described in [6] does not directly apply when dealing with PINNs.

Based on these observations, we conclude that the literature surrounding batch size on vision tasks may not generalize to PINNs and that this area demands more research. Thus, in this paper, we explore the impact of larger batch size under the scientific computing scenario with PINNs. The structure of the remaining of this paper is as follows: in Section 2, we briefly describe Physics Informed Neural Networks (PINNs); in Section 3 we carry out experiments demonstrate the benefit of utilizing larger batch-sizes for PINNs through two different PDEs, namely the Poisson equation and the Navier-Stokes equation; finally, in Section 4 we summarize our findings and outline possible future areas of research.

2 Physics Informed Neural Networks (PINNs)

We begin with a brief overview of physics-informed neural networks (PINNs)[12] which is a method for approximating the solution to differential equations using neural networks. Consider a PDE with a Dirichlet boundary condition

$$\mathcal{N}[u](\mathbf{x}) = f(\mathbf{x}), x \in \Omega \qquad\qquad u(\mathbf{x}) = g(\mathbf{x}), \mathbf{x} \in \partial\Omega, \tag{1}$$

where $\mathcal{N}$ is some nonlinear operator. To approximate this PDE using PINNs, we define a neural network u_θ and tune the parameters θ of the network to minimize the following loss function

$$\mathcal{L}(\theta) = \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}_{\partial\Omega}} \alpha(u_\theta(\mathbf{x}) - g(\mathbf{x}))^2 + \mathop{\mathbb{E}}_{\mathbf{x}\sim\mathcal{D}_{\Omega}} (\mathcal{N}[u_\theta](\mathbf{x}) - f(\mathbf{x}))^2. \tag{2}$$

$\mathcal{D}_\Omega, \mathcal{D}_{\partial\Omega}$ are uniform distributions supported on the interior and the boundary of the domain Ω, and α is the weight for the boundary loss term used to balance the residual and the boundary losses. The weight α may be fixed or adaptive [13]. By minimizing the loss function above, the networks learns a solution to the desired PDE in (1).

3 Experiments on PINNs

For the experiments carried out in this section, we consider the following PDEs

Poisson Equation The $2-$dimensional Poisson PDE consists of finding $u : \Omega \to \mathbb{R}$ satisfying

$$u_{xx} + u_{yy} - f = 0, \tag{3}$$

where u_{xx}, u_{yy} are the second-order partial derivatives with respect to input x, y and $\Omega \subset \mathbb{R}^2$. For our experiments, we choose $\Omega = [-1, 1]^2$ and choose a model solution u and corresponding forcing function f as

$$u(x, y) = \sin(k\pi x)\sin(k\pi y) \qquad\qquad f(x, y) = -2k^2\pi^2 \sin(k\pi x)\sin(k\pi y) \tag{4}$$

Navier-Stokes Equation We consider a decaying turbulence simulation in a periodic square domain $\Omega = [0, 2\pi]^2$ with Reynolds number $\mathrm{Re} = 100$ and $T = 0.12$. We pose this problem using the velocity-vorticity formulation:

$$w_t + \boldsymbol{u} \cdot \nabla w = \frac{1}{\mathrm{Re}} \Delta w, \quad \text{in } [0, T] \times \Omega, \tag{5}$$

$$\nabla \cdot \boldsymbol{u} = 0, \quad \text{in } [0, T] \times \Omega, \tag{6}$$

$$w(0, x, y) = w_0(x, y), \quad \text{in } \Omega, \tag{7}$$

3.1 Sharpness of PINNs local minimum

First, utilizing the 2D Poisson Equation, we demonstrate that training with larger batch size leads to sharper local minimum. The specific architecture we utilize along with optimization hyper-parameters have been listed in Appendix B.4. We visualize the sharpness of the loss function qualitatively at the parameters obtained by the end of the training from singular values of the Hessian matrix of the loss function evaluated at the resulting parameters. Given the $\mathcal{O}(N^3)$ complexity in performing the SVD, we reduce the neural network to 3 hidden layers, each with 16 neurons, and choose $k = 4$ in

equation (4). Figure 1 shows the sorted singular values of parameters obtained from training with different batch size, as normalized by the $\mathbb{L}_2$ norm of the singular value vectors. The singular values obtained from larger batch sizes decay consistently faster than that obtained from smaller batch sizes, suggesting that sharper loss landscapes. While *Keskar et al.*[6] propose that sharper local minima exacerbates the discrepancy between the validation and training metrics, in the case of PINNs the continuous uniform sampling fills the gap and consequently yields improved errors at sharper local minima.

3.2 Learning rate scaling for PINNs

While the example above have shown the efficacy of increasing batch size, in this section we further demonstrate that an optimal learning rate choice can significantly improve prediction accuracy and accelerate training. While convex optimization suggests that less noise in the gradient estimate in each step allows the use of larger learning rates and consequently leads to faster convergence[14], it is not sufficiently studied *how* the learning rate should be scaled in proportion to the batch size increase in neural networks. Given the scaling scheme

$$\log \text{learning rate} \propto \kappa \log(\text{batch size}), \tag{8}$$

two commonly used schemes are linear scaling[8, 15], where $\kappa = 1$, and square root scaling[16], where $\kappa = 0.5$. In this experiment, we will use PINNs to solve the 2-dimensional Poisson problem as using $k = 12$ in equation (4) and a more challenging Navier-Stokes (NS) equation to explore the optimal learning rate scaling schemes.

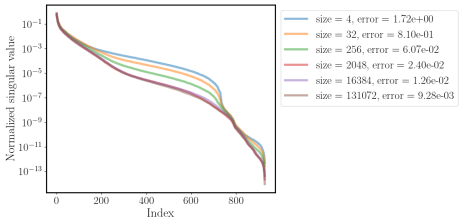


Figure 1: Sorted singular values, normalized by $\mathbb{L}_2$ norm of the singular value vectors. Each curve is evaluated on one set of parameters returned by an instance of training with various batch size, as indicated in the legend, along with its relative $\mathbb{L}_2$ error.

For both these cases, we perform a grid search to determine the optimal learning rate for a batch size. The specific architectures and hyperparameter choices we utilized for our experiments have been discussed in the Appendix. Figure 2 summarizes our findings through heat maps of the final mean ensemble errors for combinations of batch size and learning rates on the $2-$D Poisson and turbulence problem. Additionally, we plot the number of iterations required to achieve a particular error tolerance.

In summary, we infer the following from our results when utilizing a large batch size for training PINNs:

- Consistent improvements in the accuracy of the solution obtained.
- Higher tolerance to choice of learning rate; large batch sizes allows using larger learning rates without compromising the stability of training. However, we do note that beyond a certain threshold, higher learning rates would always lead to divergence of training.
- Quicker convergence to a desired error tolerance.
- Finally, the inferred slope κ for the scaling of optimal learning rate were 0.7636 and -0.05952 respectively for the $2-$D Poisson and turbulence. This shows that an optimal scaling scheme is likely dependent on properties of each specific problem and that a general learning rate scheme may not exist.

4 Discussion

We demonstrate that PINNs greatly benefit from higher batch sizes, which not only improve the accuracy but also speed up the training process. While the choice of optimal learning rate helps improve accuracy, we see that the improvement gained from learning rate tuning is not as impactful as that from increasing batch size. We are aware that PINNs still are significantly more expensive than classical numerical solvers. However, our work demonstrates that PINNs can become more competitive when worked with larger batch sizes. Given that they can achieve a required tolerance

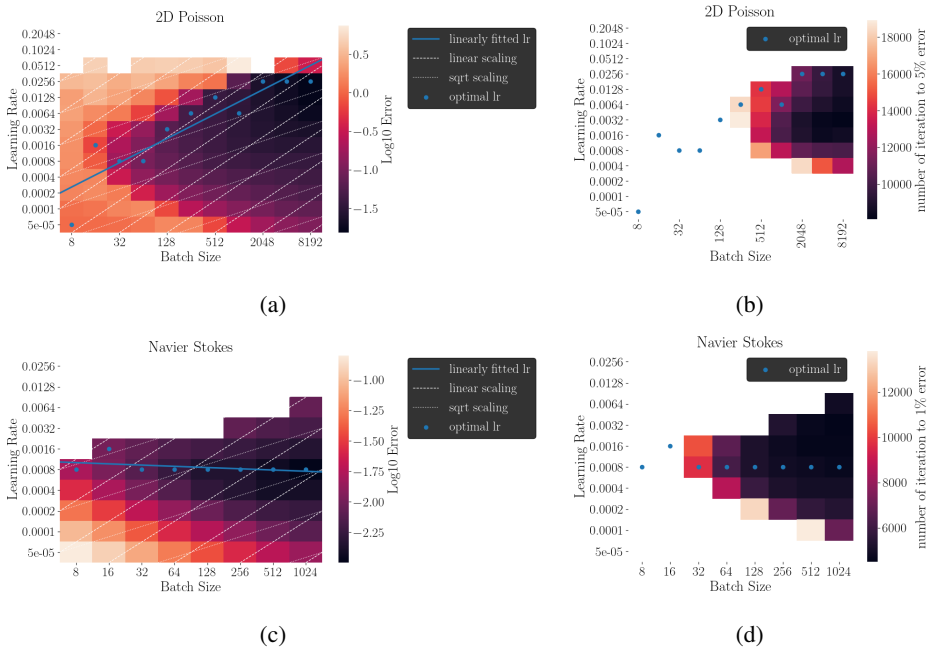


Figure 2: (*Left*) Final mean ensemble error. Ensembles where divergence in training occurs are marked in white. (*Right*) Number of iterations needed to reach certain error threshold. Ensembles whose mean error never reached the threshold are marked as white. The blue dots are the learning rate that returns best testing error given the batch size. The dashed and dotted parallel lines are scaling references for linear and square root scaling, respectively. The blue solid line is the linear regression fit for the optimal learning rate. (*Top*) Heat maps for 2-D Poisson problem. The error threshold is 0.05. The slope is 0.7636, and its 90% confidence interval is $[0.549647, 0.9776257]$. (*Bottom*) Heat maps for turbulence problem. The error threshold is 0.01 The slope is -0.05952, and its 90% confidence interval is $[-0.1638367, 0.04478911]$.

faster with higher batch sizes, a particularly effective route to this accelerate training would be employing data-parallelism to reduce the time per iteration.

Future lines of research regarding this topic include analysing the behaviour of batch size on more complicated PDEs. While this study does not consider higher dimensional PDEs, we conjecture that higher batch sizes could significantly help in that space as well, where utilizing classic numerical methods is often very computationally expensive or impossible in practice. Finally, while PINNs solve a single instance of a PDE, a possible future direction of study would involve the study of batch size in the context of operator learning architectures [17, 18, 19, 20, 21]. Using these architectures, it is possible to evaluate the solution of PDEs under different parameters almost instantaneously, after spending time training the model prior to deployment. Increasing batch sizes and reducing training time of operator learning frameworks has the potential of improving the practicality of these methods in both industry and academia.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[3] Sameer Kumar, Victor Bitorff, Dehao Chen, Chiachen Chou, Blake Hechtman, HyoukJoong Lee, Naveen Kumar, Peter Mattson, Shibo Wang, Tao Wang, Yuanzhong Xu, and Zongwei Zhou. Scale mlperf-0.6 models on google tpu-v3 pods, 2019. URL `https://arxiv.org/abs/1909.09756`.

[4] Yang You, Igor Gitman, and Boris Ginsburg. Scaling SGD batch size to 32k for imagenet training. *CoRR*, abs/1708.03888, 2017. URL `http://arxiv.org/abs/1708.03888`.

[5] Yang You, Jing Li, Jonathan Hseu, Xiaodan Song, James Demmel, and Cho-Jui Hsieh. Reducing BERT pre-training time from 3 days to 76 minutes. *CoRR*, abs/1904.00962, 2019. URL `http://arxiv.org/abs/1904.00962`.

[6] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima, 2016. URL `https://arxiv.org/abs/1609.04836`.

[7] Yang You, Yuhui Wang, Huan Zhang, Zhao Zhang, James Demmel, and Cho-Jui Hsieh. The limit of the batch size, 2020. URL `https://arxiv.org/abs/2006.08517`.

[8] Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. *CoRR*, abs/1711.00489, 2017. URL `http://arxiv.org/abs/1711.00489`.

[9] Dominic Masters and Carlo Luschi. Revisiting small batch training for deep neural networks. *CoRR*, abs/1804.07612, 2018. URL `http://arxiv.org/abs/1804.07612`.

[10] George Em Karniadakis, Ioannis G Kevrekidis, Lu Lu, Paris Perdikaris, Sifan Wang, and Liu Yang. Physics-informed machine learning. *Nature Reviews Physics*, 3(6):422–440, 2021.

[11] Maziar Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *The Journal of Machine Learning Research*, 19(1):932–955, 2018.

[12] Maziar Raissi, Paris Perdikaris, and George E Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[13] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective, 2020.

[14] Léon Bottou. Stochastic Gradient Descent Tricks. In Grégoire Montavon, Geneviève B. Orr, and Klaus-Robert Müller, editors, *Neural Networks: Tricks of the Trade: Second Edition*, Lecture Notes in Computer Science, pages 421–436. Springer, Berlin, Heidelberg, 2012. ISBN 9783642352898. doi: 10.1007/978-3-642-35289-8_25. URL `https://doi.org/10.1007/978-3-642-35289-8_25`.

[15] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017. URL `https://arxiv.org/abs/1706.02677`.

[16] Alex Krizhevsky. One weird trick for parallelizing convolutional neural networks. *CoRR*, abs/1404.5997, 2014. URL `http://arxiv.org/abs/1404.5997`.

[17] Lu Lu, Pengzhan Jin, Guofei Pang, Zhongqiang Zhang, and George Em Karniadakis. Learning nonlinear operators via deeponet based on the universal approximation theorem of operators. *Nature Machine Intelligence*, 3(3):218–229, 2021.

[18] Sifan Wang, Hanwen Wang, and Paris Perdikaris. Learning the solution operator of parametric partial differential equations with physics-informed deeponets. *Science advances*, 7(40): eabi8605, 2021.

[19] Nikola Kovachki, Zongyi Li, Burigede Liu, Kamyar Azizzadenesheli, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Learning maps between function spaces. *arXiv preprint arXiv:2108.08481*, 2021.

[20] Georgios Kissas, Jacob H Seidman, Leonardo Ferreira Guilhoto, Victor M Preciado, George J Pappas, and Paris Perdikaris. Learning operators with coupled attention. *Journal of Machine Learning Research*, 23(215):1–63, 2022.

[21] Jacob H Seidman, Georgios Kissas, Paris Perdikaris, and George J Pappas. Nomad: Nonlinear manifold decoders for operator learning. *arXiv preprint arXiv:2206.03551*, 2022.

[22] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks, 2020.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[24] Sifan Wang, Shyam Sankaran, and Paris Perdikaris. Respecting causality is all you need for training physics-informed neural networks, 2022. URL `https://arxiv.org/abs/2203.07404`.

A Architecture

A.1 Multilayer Pecerptron (MLP)

A Multilayer Pecerptron (MLP) consists of compositions of layers, which are affine transformations followed by an activation function $\sigma : \mathbb{R} \to \mathbb{R}$. That is, a layer computes the following transformation from $\mathbb{R}^{d_{in}}$ to $\mathbb{R}^{d_{out}}$

$$x \mapsto \sigma(Wx + b),$$

where $W \in \mathbb{R}^{d_{in} \times d_{out}}$ and $b \in \mathbb{R}^{d_{out}}$ are trainable parameters and σ is applied element-wise.

The MLP architecture then consists of multiple iterations of the transformation above being carried out, each with independent trainable parameters.

A.2 Positional Encoding

The harmonic encoding given input vector $\boldsymbol{x}$ can be expressed as the concatenation of feature vectors

$$[\sin(2^k \pi \boldsymbol{x}), \cos(2^k \pi \boldsymbol{x})], k = -2, -1, 0, 1, 2. \tag{9}$$

A.3 Modified MLP

A modified MLP architecture for PINNs was described in [22] and found to consistently yield better results for PINNs. An L-layer architecture of the modified MLP is defined as follows

$$\boldsymbol{U} = \sigma(\boldsymbol{X}\boldsymbol{W}_1 + \boldsymbol{b}_1), \;\; \boldsymbol{V} = \sigma(\boldsymbol{X}\boldsymbol{W}_2 + \boldsymbol{b}_2), \tag{10}$$

$$\boldsymbol{H}^{(1)} = \sigma(\boldsymbol{X}\boldsymbol{W}^{(1)} + \boldsymbol{b}^{(1)}), \tag{11}$$

$$\boldsymbol{Z}^{(l)} = \sigma(\boldsymbol{H}^{(k)}\boldsymbol{W}^{(l+1)} + \boldsymbol{b}^{(l+1)}), \;\; l = 1, \ldots, L-1, \tag{12}$$

$$\boldsymbol{H}^{(l+1)} = (1 - \boldsymbol{Z}^{(l)}) \odot \boldsymbol{U} + \boldsymbol{Z}^{(l)} \odot \boldsymbol{V}, \;\; l = 1, \ldots, L-1, \tag{13}$$

$$\boldsymbol{u}_{\boldsymbol{\theta}}(\boldsymbol{X}) = \boldsymbol{H}^{(L)}\boldsymbol{W}^{(L+1)} + \boldsymbol{b}^{(L+1)}, \tag{14}$$

where σ denotes a nonlinear activation function, $\odot$ denotes a point-wise multiplication, and $\boldsymbol{X}$ denotes an batch of input coordinates.

B Hyper Parameter and Architecture Choices for Numerical Experiments

B.1 Optimizer Choice

We train using Adam[23] and learning rate schedule consisting of a linear warmup phase of $1,000$ steps followed by an exponential decay schedule with rate 0.1 per $10,000$ transition steps. The total number of iterations is $20,000$. At the end of training, the learning rate is reduced to 0.01 of the peak value.

B.2 Singular Value Spectrum for 2D Poisson Problem

The weight on the boundary loss term in equation (2) is 1000 for the evaluation of singular value spectrum. We set the MLP following the positional encoding to be 3 hidden layers, each with 16 neurons.

B.3 Grid Search for 2D Poisson

We utilize a 3-layer MLP of width 256 with a positional encoding layer consisting of 5 frequencies, and utilize a boundary loss weight $\alpha = 100$ in Equation 2. The results presented were averaged over an ensemble size of 32.

B.4 Navier Stokes

The architecture we utilize is largely based on the architecture utilized in [24]: a modified MLP architecture consisting of 3 hidden layers of width 256. with harmonic expansion layer at the start to ensure that the network implicitly satisfies the periodic boundary conditions. Additionally, we parameterize the model using two separate networks as

$$u_{\theta_1,\theta_2}(x, y, t) = \bar{u}_{\theta_1}(x, y, t)t + \tilde{u}_{\theta_2}(x, y) \tag{15}$$

Before, we carry out our grid search, we train the network $\tilde{u}_{\theta_2}$ to capture the initial condition with high accuracy (0.01% relative $\mathcal{L}_2$ error) and keep this fixed for all the experiments we perform. Hence, we're only evolving θ_1 during the process of training our model and do not have losses for the initial condition competing during the process of training. However, we assign a weight of 500 to the continuity residual equation over the vorticity residual equation. Finally, we train the network utilizing the causality enforcing training process described in [24].

C Larger batch size helps overcome the spectral bias

We use a one dimensional Poisson problem to demonstrate that larger batch size helps to stabilize training.

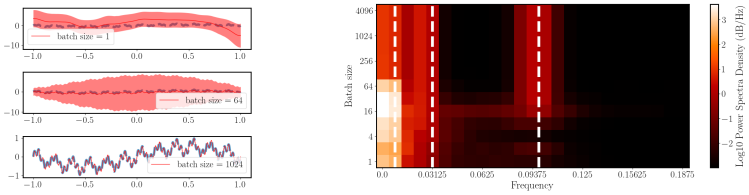
Poisson Equation The $1-$dimensional Poisson PDE consists of finding $u : \Omega \to \mathbb{R}$ satisfying

$$u_{xx} - f = 0, \tag{16}$$

where u_{xx} are the second-order partial derivatives with respect to input x, y and $\Omega \subset \mathbb{R}$. For our experiments, we choose $\Omega = [-1, 1]$. We choose the solution

$$u = \sum_{i}^{3} A_i \sin(\pi k_i x), A_1 = 0.5, A_2 = 0.3, A_3 = 0.2, k_1 = 1, k_2 = 12, k_3 = 48. \tag{17}$$

In this case the weight on the boundary loss term in equation (2) is 10, and we use the same architecture as in section B.3. Figure 3 shows that larger batch size stabilizes the training, reduce the uncertainty, and shift the predictive power spectrum towards higher frequency.



(a) (b)

Figure 3: (a) Final mean ensemble predictions trained with batch size $1, 64, 1024$ with $2-$std error bands, top to bottom. Ground truth function is represented in the dashed blue line. (b) Power spectrum of final mean ensemble prediction trained with various batch sizes. Vertical dashed lines are frequencies present in the target function. The intensities of these frequencies of target function are approximately $10^{0.80}, 10^{0.57}, 10^{0.17}$, left to right.