Loss-driven sampling within hard-to-learn areas for simulation-based neural network training

Sofya Dymchenko
Univ. Grenoble Alpes, Inria,
CNRS, Grenoble INP, LIG
38000 Grenoble, France
sofya.dymchenko@inria.fr

Bruno Raffin
Univ. Grenoble Alpes, Inria,
CNRS, Grenoble INP, LIG
38000 Grenoble, France
bruno.raffin@inria.fr

Abstract

This paper focuses on active learning methods for training neural networks from synthetic input samples that can be generated on-demand. This includes Physics Informed Neural Networks (PINNs), simulation-based inference, deep surrogates and deep reinforcement learning. An adaptive process observes the training progress and steers the data generation with the goal of speeding up and increasing the quality of training. We propose a novel adaptive sampling method that concentrates samples close to the areas showing high loss values. Compared to the state-of-the-art R3 sampling our algorithm converges to a validation loss of 0.5 in 6000 iterations, while it takes 25000 iterations to reach a loss of 0.7 for the R3 algorithm when training a PINN with the Allen Cahn equation.

1 Introduction

The machine learning community has recently shown a growing interest in applying deep neural networks (DNN) to physical science [Lavin et al., 2021]. The novel approaches take different perspectives on the subject [Das and Tesfamariam, 2022]. Some design specific types of neural models and loss functions, such as graph-based networks [Meyer et al., 2021], physics-informed neural networks (PINNs) [Raissi et al., 2019]. Others use DNNs to learn probability distributions for experimental design [Foster et al., 2021], probabilistic programming [Baydin et al., 2019], and simulation-based inference [Cranmer et al., 2022]. Deep surrogates are combined with traditional numerical solvers and trained through different modalities on supercomputers as in [Meyer et al., 2022], [Brace et al., 2021] and [Ward et al., 2021]. In this paper we focus on adaptive sampling methods to optimize DNN training in the case where samples can be generated on-demand, usually through a solver code.

Classical DNN training methods work with fixed datasets that are repeatedly presented during training across multiple epochs. The ability to perform training using synthetic data, which can be generated on-demand, opens the way to different strategies to improve the training process. Synthetic data is generated from a set of input parameters sampled within a given bounded domain. The inputs can be directly used for DNN training without transformation as in data-free PINNs training, or be the initial conditions for autoregressive solvers, which produce data series. Adaptive sampling monitors training progress to guide the sampling process towards selecting the inputs that provide the most *effective* data. The associated computing cost should also be reduced so as not to slow down training.

The question of data selection and sample similarity originated in the context of training DNN with a finite dataset. It is tackled in various ways: measuring samples uncertainty by approximating training dynamics [Kye et al., 2022, Wang et al., 2022], calculating samples influence [K and Søgaard, 2021], selecting representative subset with use of gradients [Killamsetty et al., 2022, Fayyaz et al., 2022, Katharopoulos and Fleuret, 2019].

Machine Learning and the Physical Sciences Workshop, NeurIPS 2023.

For PINN's collocation set [Yang et al., 2022, Nabian et al., 2021] proposes re-weighting samples importance, [Wu et al., 2022] creates a training subset based on a distribution calculated as a normalized loss. However, these methods usually require additional computations or are not applicable to non-finite data. The recently published R3 sampling [Daw et al., 2023] advances the baseline by retaining points whose loss is higher than average and resampling the remaining ones uniformly for each iteration. This way the DNN is able to learn better the points with high loss while exploring new ones. Relying on the per-sample loss to quantify the data training effectiveness has the benefit of being a lightweight single value, which requires no extra computation or excessive memory.

In this paper, we propose a method where the main objective is to increase the sampling density in areas with high loss over the lifetime of the training process. Instead of retaining high-loss points, we sample new points in their Gaussian neighbourhoods according to loss. To avoid overfitting in these areas, we introduce a balance control value that evolves across iterations. It defines a ratio between the proposed loss-driven sampling and uniform sampling, which enables the exploration of new high-informative areas and the repetition of already learned examples. The main contributions of the paper are:

- a loss-driven sampling method called **Breed** (**B**alance **R**atio and **E**nhanc**E** **D**ensity);
- a concept of exploration-concentration balance control with ratio value;
- a novel benchmark designed for evaluating sampling strategies for simulation-based training;
- a comparison with the state-of-the-art R3 sampling on two tasks, which shows a significant performance improvement on both, the convergence speed and validation loss.

2 Proposed method

Let's first introduce notations. The sampling selects input points $\boldsymbol{x} \in \mathcal{X} \subseteq \mathbb{R}^d$. $\boldsymbol{x}$ can be used directly as a collocation point as for PINNs training, or go through a function $f(\boldsymbol{x}) = \boldsymbol{y}$, where $\boldsymbol{y} \in \mathcal{Y} \subseteq \mathbb{R}^d$ is the output, the tuple $(\boldsymbol{x}, \boldsymbol{y})$ being used for training. The trained neural network is denoted $f_\theta(\boldsymbol{x})$, where θ are neural network parameters. In the case of surrogate training, the goal is to have f_θ approximate f.

2.1 The exploration-concentration balance control value

To balance a training set, we introduce a concentrate-explore value $r^{(i)} \in [0, 1]$ as a **r**atio of points to sample non-uniformly by concentrating in areas of high loss. The remaining points are sampled uniformly in the domain for (1) having examples for the network to remember what was learned and (2) exploring areas with samples that might show high losses. The value $r^{(i)}$ changes over iterations i of the neural network, growing linearly from the starting value s_r to the ending value e_r for c_r iterations and then is constant at value e_r. The configuration of r value is called *scenario* and denoted as a triplet (s_r, e_r, c_r).

2.2 Gaussian neighbourhood sampling method

The sampling strategy is based on loss statistics from the neural network similar to [Wu et al., 2022]. The loss is normalized to construct a distribution that will provide a number of points to sample within sample neighbourhoods.

The algorithm is repeated for N **i**terations. i.e. $i = [N]$, where we denote $[N] = 0, \ldots, N-1$. The covariance $\boldsymbol{\Sigma} = I_d \cdot \sigma$ defines the radius of the spherical neighbourhood (fixed). The values $r^{(i)} \in [0, 1]$ to control concentrate-explore trade-off **r**atio are predefined (subsection 2.1). The initial training **s**et $\mathcal{S}^{(0)}$ is sampled uniformly and the **n**umber of **s**amples $|\mathcal{S}^{(i)}| = N_s$ is fixed.

At each iteration i, the neural network $f_{\theta^{(i)}}(x)$ provides a loss value $\mathcal{L}(\boldsymbol{x}_j^{(i)}; \theta^{(i)}) = l_j^{(i)} \geq 0$ per sample $\boldsymbol{x}_j^{(i)} \in \mathcal{S}^{(i)}$ for $j = [N_s]$. The values of vector $\boldsymbol{l}^{(i)}$ then are sum normalized to have distribution properties. The categorical distribution over the samples of $\mathcal{S}^{(i)}$ is constructed proportionally to the loss, i.e. $P(\boldsymbol{x}_j^{(i)}) = l_j'^{(i)}$. This distribution trialed n times is a multinomial distribution $P(n, \boldsymbol{l}'^{(i)})$. It models the number of points, called *children*, in the next training set to be sampled around each point, called *parent*, of the current training set. The sampling is run with replacements, so a parent

with high loss will have several children while a parent with low loss might not have any. It allows us to adaptively refine sampling density in areas with high loss. The name of the algorithm, **Breed**, self-explains this mechanic of breeding the most interesting points from the point of view of the training process.

The next training set $\mathcal{S}^{(i+1)}$ consists of two sets. The *concentration* set $\mathcal{S}_c^{(i+1)}$ is a set of points sampled in a loss-driven manner. Its size depends on $r^{(i)}$ value, i.e. $|\mathcal{S}_c^{(i+1)}| = N_c^{(i)} = \lfloor N_s \times r^{(i)} \rfloor$. To construct it, the number of children for each parent is sampled as $\{m_j\}_{j=[N_s]} \sim P(N_c^{(i)}, \boldsymbol{l}'^{(i)})$. Note that $\sum_{j=[N_s]} m_j = N_c^{(i)}$. A parent point acts as the centre of a Gaussian with $\boldsymbol{\Sigma}$ width, which we call a neighbourhood. We sample the children set from each neighbourhood, i.e.:

$$\mathcal{S}_c^{(i+1)} = \bigcup_{j=[N_s]} \mathcal{C}_\Sigma(x_j^{(i)}) = \bigcup_{j=[N_s]} \left\{ \boldsymbol{x}_k^{(i+1)} \sim \mathcal{N}(\boldsymbol{x}_j^{(i)}, \boldsymbol{\Sigma}), k = [m_j] \right\}. \tag{1}$$

The *uniform* set $\mathcal{S}_u^{(i+1)}$ is a remaining portion of points, i.e. $|\mathcal{S}_u^{(i+1)}| = N_u^{(i)} = N_s - N_c^{(i)}$, sampled uniformly to explore new data and to keep the presence of points with low loss as well, i.e.:

$$\mathcal{S}_u^{(i+1)} = \left\{ \boldsymbol{x}_k^{(i+1)} \sim \mathcal{U}(\mathcal{X}), k = [N_u^{(i)}] \right\}. \tag{2}$$

Finally, the composed training set is $\mathcal{S}^{(i+1)} := \mathcal{S}_c^{(i+1)} \cup \mathcal{S}_u^{(i+1)}$.

3 Experiments

We compare[1] Breed sampling to the baseline uniform dynamic sampling, which creates a training set for each iteration by selecting N_s uniformly distributed points in $\mathcal{X}$, and the R3 sampling [Daw et al., 2023] on two benchmark problems.

Benchmarks. The first problem is a new simulation-based benchmark called *pits gradient descent (PGD)*. The simulation is a gradient descent on a surface made of "bell pits" configured by the number, centres, weights, and widths of the bells (negative Gaussians curves). The input is a point $x_0 \in \mathcal{X} \subset \mathbb{R}^2$ and the output is the local minimum $x_f \in \mathcal{Y} = \mathcal{X} \subset \mathbb{R}^2$ corresponding to the starting point of the gradient descent. The task can be extended to a time-dependent variant and/or 3-dimensional variant with boundary conditions being a surface equation. The motivation to create this benchmark was to have flexibility in configuring different cases and an obvious visual clue about hard-to-learn areas, which are those with near-zero gradients.

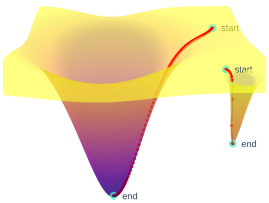
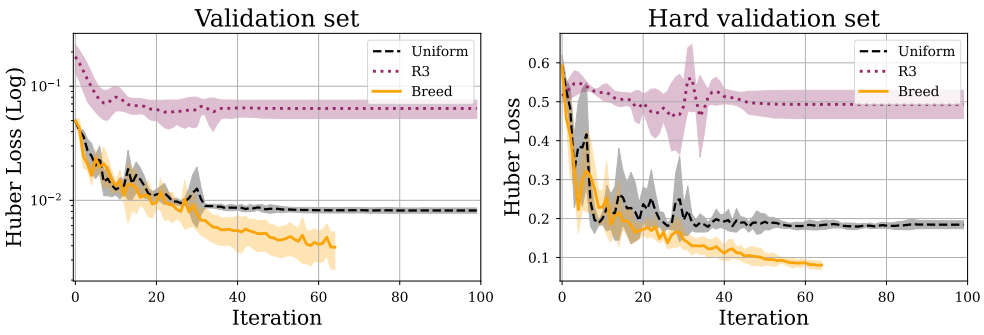


Figure 1: IPGD surface

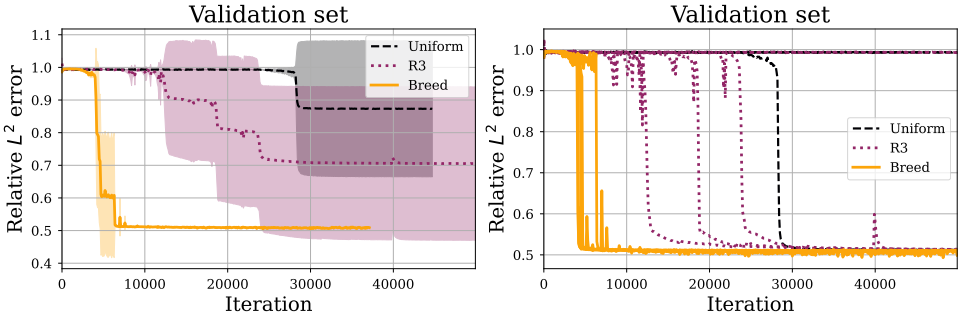
The case we consider in experiments is a surface with two pits located at points $(-0.5, -0.5)$ and $(1.5, 1.5)$ in the domain $\mathcal{X} \times \mathcal{Y} = [-2, 2]^2$, with corresponding weights $(0.895, 0.005)$, and widths $(0.4, 0.008)$. Because the widths and weights are extremely different, this configuration is referred to as *two imbalanced pits (IPGD)*, see Figure 1. The validation is done on two sets: the first one consists of 10k points uniformly sampled in the domain, and the second, *hard validation set*, consists of 10k points whose gradients are less than $4 \cdot 10^{-3}$. For a fair comparison, we designed a neural network with hyperparameters optimized for the best results with uniform sampling. The model is a 4-layered perceptron, Adam optimizer with a learning rate 10^{-3} scheduled by plateau scheduler with patience 5, weight decay $5 \cdot 10^{-5}$, and a Huber loss function. The maximum number of iterations is $N = 100$ and the number of samples is $N_s = 1000$. The r scenario is $(0.15, 0.7, 25)$, the width of neighborhoods is $\sigma = 0.005$.

The second benchmark is a PINN trained to solve the classical Allen Cahn partial differential equation. This benchmark is used in [Daw et al., 2023]. The model we use here has the exact same architecture and hyperparameters, and others are as follows: $N = 60k$, $N_s = 1000$, r scenario is $(0.15, 0.7, 5000)$, $\sigma = 0.001$.

[1] `https://gitlab.inria.fr/breed/breed/`



(a) Imbalanced pits GD



(b) Allen Cahn equation (averaged)

(c) Allen Cahn equation (not averaged)

Figure 2: Comparison of validation errors over training iterations for Breed, R3 and uniform sampling for (a) Imbalanced Pits Gradient Descent and (b-c) Allen Cahn PINN. The plots (a-b) are presented as mean and standard deviation computed over 5 random seeds, whereas in the plot (c) each line represents one run.

Results analysis. Results are presented in Figure 2 as errors over training iterations. For all experiments, Breed shows both faster convergence and lower errors compared to Uniform and R3 sampling. Notice that for Breed the error decrease happens near the c_r iteration, showing the importance of the exploration-concentration r scheme. In Figure 2a, R3 sampling performs worse than the baseline for both validation sets, while Breed reaches twice lower error than the baseline even for the hard validation set. In Figure 2b, the variation of error for R3 is explained by the instability of the method visible in Figure 2c. R3 converged for 3 out of the 5 trainings to the same error value. In opposite, Breed shows a stable convergence for all runs, while the uniform sampling converges only for 1 run.

4 Conclusion

We presented Breed, a novel adaptive sampling algorithm for training DNNs with synthetic data. Breed relies on the per-sample loss value to identify hard-to-train areas and combines a dual exploration-concentration scheme with Uniform sampling to discover potential hard areas and to remember trivial ones and Gaussian multinomial sampling to focus on hard areas. The experiments demonstrate overall better performance in quality, convergence speed, and stability compared to the baseline uniform and the state-of-the-art R3 sampling. Future work will focus on validating Breed with more benchmarks, including higher dimension problems and simulation-based scenarios with functions generating time series.

5 Acknowledgements

This work has been supported by the ENGAGE Inria-DFKI project.

References

[Baydin et al., 2019] Baydin, A. G., Shao, L., Bhimji, W., Heinrich, L., Meadows, L., Liu, J., Munk, A., Naderiparizi, S., Gram-Hansen, B., Louppe, G., Ma, M., Zhao, X., Torr, P., Lee, V., Cranmer, K., Prabhat, and Wood, F. (2019). Etalumis: Bringing probabilistic programming to scientific simulators at scale. Publisher: IEEE Computer Society.

[Brace et al., 2021] Brace, A., Yakushin, I., Ma, H., Trifan, A., Munson, T., Foster, I., Ramanathan, A., Lee, H., Turilli, M., and Jha, S. (2021). Coupling streaming AI and HPC ensembles to achieve 100-1000x faster biomolecular simulations.

[Cranmer et al., 2022] Cranmer, K., Brehmer, J., and Louppe, G. (2022). The frontier of simulation-based inference.

[Das and Tesfamariam, 2022] Das, S. and Tesfamariam, S. (2022). State-of-the-art review of design of experiments for physics-informed deep learning. Number: arXiv:2202.06416.

[Daw et al., 2023] Daw, A., Bu, J., Wang, S., Perdikaris, P., and Karpatne, A. (2023). Mitigating propagation failures in physics-informed neural networks using retain-resample-release (r3) sampling. In *Proceedings of the 40th International Conference on Machine Learning*, pages 7264–7302. PMLR. ISSN: 2640-3498.

[Fayyaz et al., 2022] Fayyaz, M., Aghazadeh, E., Modarressi, A., Pilehvar, M. T., Yaghoobzadeh, Y., and Kahou, S. E. (2022). BERT on a data diet: Finding important examples by gradient-based pruning. In *NeurIPS*.

[Foster et al., 2021] Foster, A., Ivanova, D. R., Malik, I., and Rainforth, T. (2021). Deep adaptive design: Amortizing sequential bayesian experimental design.

[K and Søgaard, 2021] K, K. and Søgaard, A. (2021). Revisiting methods for finding influential examples.

[Katharopoulos and Fleuret, 2019] Katharopoulos, A. and Fleuret, F. (2019). Not all samples are created equal: Deep learning with importance sampling.

[Killamsetty et al., 2022] Killamsetty, K., Abhishek, G. S., Ramakrishnan, G., Evfimievski, A. V., Popa, L., and Iyer, R. (2022). AUTOMATA : Gradient based data subset selection for compute-efficient hyper-parameter tuning. In *Advances in Neural Information Processing Systems*.

[Kye et al., 2022] Kye, S. M., Choi, K., and Chang, B. (2022). TiDAL: Learning training dynamics for active learning. Publisher: arXiv Version Number: 1.

[Lavin et al., 2021] Lavin, A., Zenil, H., Paige, B., Krakauer, D., Gottschlich, J., Mattson, T., Anandkumar, A., Choudry, S., Rocki, K., Baydin, A. G., Prunkl, C., Paige, B., Isayev, O., Peterson, E., McMahon, P. L., Macke, J., Cranmer, K., Zhang, J., Wainwright, H., Hanuka, A., Veloso, M., Assefa, S., Zheng, S., and Pfeffer, A. (2021). Simulation intelligence: Towards a new generation of scientific methods.

[Meyer et al., 2021] Meyer, L., Pottier, L., Ribes, A., and Raffin, B. (2021). Deep surrogate for direct time fluid dynamics. pages 1–7.

[Meyer et al., 2022] Meyer, L., Ribés, A., and Raffin, B. (2022). Simulation-based parallel training.

[Nabian et al., 2021] Nabian, M. A., Gladstone, R. J., and Meidani, H. (2021). Efficient training of physics-informed neural networks via importance sampling. 36(8):962–977.

[Raissi et al., 2019] Raissi, M., Perdikaris, P., and Karniadakis, G. E. (2019). Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. 378:686–707.

[Wang et al., 2022] Wang, H., Huang, W., Wu, Z., Tong, H., Margenot, A. J., and He, J. (2022). Deep active learning by leveraging training dynamics.

[Ward et al., 2021] Ward, L., Sivaraman, G., Pauloski, J. G., Babuji, Y., Chard, R., Dandu, N., Redfern, P. C., Assary, R. S., Chard, K., Curtiss, L. A., Thakur, R., and Foster, I. (2021). Colmena: Scalable machine-learning-based steering of ensemble simulations for high performance computing. pages 9–20.

[Wu et al., 2022] Wu, C., Zhu, M., Tan, Q., Kartha, Y., and Lu, L. (2022). A comprehensive study of non-adaptive and residual-based adaptive sampling for physics-informed neural networks.

[Yang et al., 2022] Yang, Z., Qiu, Z., and Fu, D. (2022). DMIS: Dynamic mesh-based importance sampling for training physics-informed neural networks.