On Using Deep Learning Proxies as Forward Models in Optimization Problems

Fatima Albreiki[*], **Nidhal Belayouni and Deepak K. Gupta**
AIQ, UAE
fatima.albreiki@mbzuai.ac.ae, {nbelayouni,dgupta}@aiqintelligence.ae

Abstract

Physics-based optimization problems are generally very time-consuming, especially due to the computational complexity associated with the forward model. Recent works have demonstrated that physics-modelling can be approximated with neural networks. However, there is always a certain degree of error associated with this learning, and we study this aspect in this paper. We demonstrate through experiments on popular mathematical benchmarks, that neural network approximations (NN-proxies) of such functions when plugged into the optimization framework, can lead to erroneous results. In particular, we study the behaviour of particle swarm optimization and genetic algorithm methods and analyze their stability when coupled with NN-proxies. The correctness of the approximate model depends on the extent of sampling conducted in the parameter space, and through numerical experiments, we demonstrate that caution needs to be taken when constructing this landscape with neural networks. Further, the NN-proxies are hard to train for higher dimensional functions, and we present our insights for 4D and 10D problems. The error is higher for such cases, and we demonstrate that it is sensitive to the choice of the sampling scheme used to build the NN-proxy. The code is available at `https://github.com/Fa-ti-ma/NN-proxy-in-optimization`.

1 Introduction

In the past few years, deep learning has led to several advancements in the different disciplines of science and engineering. Some popular examples include the development of a rigorous solution to the protein-fold problem [5] and enhancing computational fluid dynamics simulations [9], among others. One popular application of deep learning is to develop proxy models that can serve as an alternative for the computationally-intensive physics-based models. An interesting example is the deep-learning tomography method [2] which aims at by-passing the computationally demanding steps of the traditional seismic tomography processes.

The gain with deep learning can be even higher when such a proxy model is used as an alternative to a physics-based simulation that serves as a forward model of an optimization problem. Note that the optimization problem referred is different from the optimization problem solved for model training in deep learning. We refer here to the physics-based optimization problems where the goal is to optimize the parameters associated with the underlying physics of a certain problem. An example is the problem of near-surface velocity modeling of the earth, where the goal of the optimization is to identify the distribution of compression and shear velocities for a predefined model space, such that the resultant spectrum images of the earth, as acquired by sensors, can be optimally reconstructed [10]. This optimization involves solving the forward model and computing its gradient repeatedly for a number of iterations, and reducing the cost associated with the forward model can speed up the overall optimization problem by a large margin.

[*]Work done as part of the internship at AIQ.

Machine Learning and the Physical Sciences workshop, NeurIPS 2022.

When using deep learning proxy models, it is important to note that the parameter space of the proxy model might not be the same as the original forward model. While the physics-model is parameterized with a few control variables, the proxy NN-model is generally designed using millions of weights. The choice of parameters for the NN-proxy depends on several factors, such as the sampling method for picking the training data, extent of convergence of the model, *etc*. It is well known that the performance of the optimization methods depends significantly on the complexity of the parameter landscape, and a changed landscape due to the use of deep learning can also lead to completely different optimized solutions which are not necessarily optimal. Clearly, there are two different aspects that need to be studied when creating such a proxy model for the optimization process: the robustness of the deep learning approach as well as efficacy of the chosen optimization method.

Contributions. In this work, we present our first observations on how the choice of data sampling as well as the amount of data samples for training the NN-proxy can influence the performance of the overall optimization process. We use the popular benchmark functions from the field of global optimization (*Rosenbrock, Rastrigin, Ackley,* etc.) as alternative representations for the physics models used in NN-proxies. These mathematical functions have long served as benchmarks for the generic field of optimization, and the exact solutions are known. For optimization, we use particle swarm optimization (PSO) [6] method and genetic algorithm (GA) [4] and start with first analyzing the stability of these methods when the original parameter landscapes are replaced with those obtained from NN-proxies. Our investigation reveals that the chosen optimization methods become very sensitive to their respective initialization schemes when the NN-proxy is employed. Sampling scheme as well as the amount of training data plays an important role in the construction of good-NN proxies, and we study this aspect as well qualitative as well as quantitative assessment of the error in optimization. Lastly, we also study the influence of NN-proxy error on optimization in higher dimensions and report results for 4D and 10D functions.

2 Deep Learning Proxy Models (NN-Proxies)

Building a deep learning proxy model, also referred as NN-proxy, implies learning the mapping of the physics-based models in a data-driven manner and using it as a forward model in an optimization problem. A better understanding of this can be obtained from Figure 1. Similar to the physics-model, the NN-proxy computes the desired output which is then evaluated using the objective function to decide if the model input parameters are to be updated or not. It is evident from the figure that a large error in the predictions made by the NN-proxy can drift the whole optimization process off track, and it is very important that the neural network model is sufficiently good before it can be plugged into the optimization process as a NN-proxy. We analyze this aspect through multiple experiments on popular mathematical benchmarks below.

Experimental setup. There are several factors that need to be understood when using a NN-proxy in a optimization process, and in this paper, we study a few important ones. In place of the original physics-models, we use popular mathematical benchmarks to study the efficacy of global optimization methods, and use neural networks to learn proxy representations of these functions. We use Rosenbrock, Rastrigin and Ackley functions [8, 7, 1] and experiment with particle swarm optimization (PSO) [6] and genetic algorithm (GA) [4] as the two optimization methods. The two chosen optimization methods are well established to solve complex multimodal global optimization methods, and we experiment here how they fair when the original models are replaced by their NN-proxies. We also consider how the error associated with the NN-proxy scales in higher dimensions, and for this purpose, we consider the cases of 4D and 10D. More details related to the chosen benchmark functions are described in Appendix A. Further, brief descriptions of the chosen optimization methods are presented in Appendix B.

Below we present our observations related to a series of experiments aimed at understanding the efficacy of NN-proxies.

Choice of the initial model for optimization. Optimization methods are generally sensitive to the choice of the initial values of the optimization parameters. Thus, it is of interest to investigate how this behaviour changes when the NN-proxies replace the actual forward models. Figures 2 and 3 show the results obtained for Rosenbrock and Rstrigin functions, respectively, using PSO and GA. For both the functions, we present the solution of PSO and GA on the actual function, and it is observed

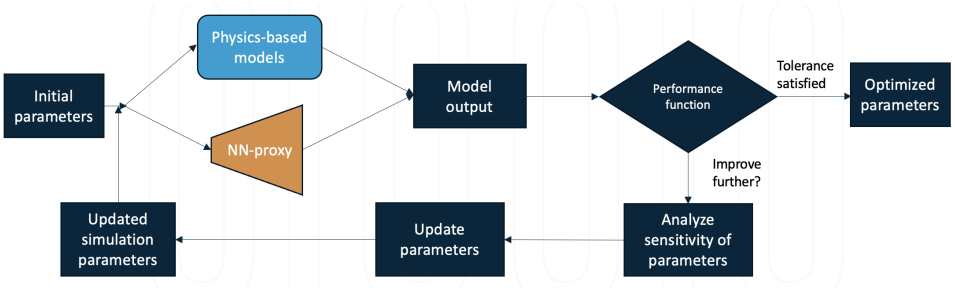


Figure 1: Schematic representation of a conventional optimization pipeline using conventional physics-based forward models as well as their alternative representation using NN-proxies.

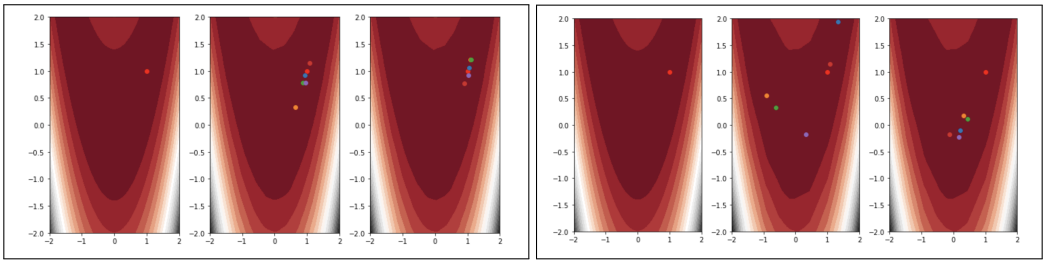


Figure 2: Example results for the Rosenbrock function for dense (left) and sparse (right) sampling of the function space. For each case, we show PSO and GA results on the true function (left), results of PSO on the NN-proxy (middle) and results of GA o the NN-proxy (right) for 5 random seeds.

that for almost all cases, the global minima is found. On the contrary, when the true function is replaced by its NN-proxy, both the optimization methods become very sensitive. For the case where dense sampling is used for creating the training dataset, the sensitivity to the initial values is low. However, for sparse sampling, the error is significantly high with the predicted solutions diverging significantly from the global minimum. Experimental details are described in Appendix C and the quantitative results related to this set of experiments as well as for Ackley function are presented in Table 1. From this experiment, it can be concluded that the behavior of an optimizer with the actual physics model can be very different from that when used with a NN-proxy. Moreover, most physics problems are complex and computationally expensive, and dense sampling is not the choice, and with sparse sampling, the results can be very erroneous. This is a matter of concern and deserves attention when using NN-proxies to represent physics of a system.

Effect of data sampling. Data sampling plays an important role in bulding a good NN-proxy. This is clearly reflected from the results shown in Figures 2 and 3 for 2D functions. We further study this and present a quantitative analysis in Table 1. Apart from the uniform dense and sparse samplings, we also include Gaussian sampling around the global minimum. This choice is meant to weakly reflect the scenario when a priori information is available on the location of the best solution. From

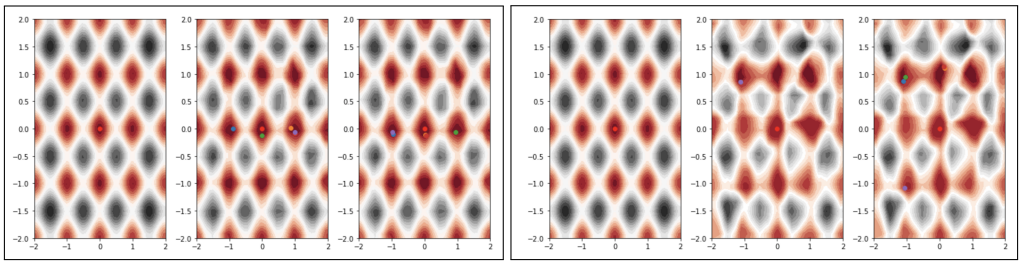


Figure 3: Example results for the Rastrigin function for dense (left) and sparse (right) sampling of the function space. For each case, we show PSO and GA results on the true function (left), results of PSO on the NN-proxy (middle) and results of GA o the NN-proxy (right) for 5 random seeds.

Table 1: Mean Euclidean distance and the standard deviation between the globally optimal point of the three mathematical benchmarks in 2D and the respective solutions obtained for the NN-proxies using PSO and GA. Three different sampling methods are used, and ground-truth refers to the case where PSO and GA are used on the true functions.

Function	Rosenbrock		Rastrigin		Ackley	
Optimizer	PS	GA	PS	GA	PS	GA
Ground-Truth	0.0 ± 0.0	0.91 ± 0.53	0.0 ± 0.0	0.0 ± 0.0	0.0 0.0 ± 0.0	0.0 ± 0.0
Dense	0.34 ± 0.21	0.54 ± 0.69	0.75 ± 0.42	0.16 ± 0.22	0.03 ± 0.0	0.03 ± 0.0
Sparse	1.17 ± 0.62	1.37 ± 0.49	1.41 ± 0.04	1.24 ± 0.13	0.14 ± 0.08	0.14 ± 0.05
Gaussian	1.07 ± 0.54	1.15 ± 0.28	1.07 ± 0.54	1.15 ± 0.28	0.06 ± 0.03	0.07 ± 0.02

Table 2: Mean Euclidean distance and the standard deviation between the globally optimal point of the three mathematical benchmarks in 4D and the respective solutions obtained for the NN-proxies using PSO and GA. Here, three different sampling methods are used t build the training set for the NN-proxies.

Function	Rosenbrock		Rastrigin		Ackley	
Optimizer	PS	GA	PS	GA	PS	GA
Dense	0.60 ± 0.03	1.59 ± 0.74	0.47 ± 0.0	0.44 ± 0.02	0.76 ± 0.20	0.90 ± 0.20
Sparse	0.86 ± 0.06	1.56 ± 0.72	0.72 ± 0.0	0.70 ± 0.02	0.37 ± 0.03	0.27 ± 0.07
Gaussian	2.51 ± 0.57	2.49 ± 0.85	0.71 ± 0.0	0.67 ± 0.05	0.26 ± 0.07	0.27 ± 0.06

the obtained results, we see that for all cases of sampling, there is a significant amount of error in the prediction when using the NN-proxy. In general, we see that sparse sampling is not very effective and the optimizers are very sensitive to the landscape constructed from such data. For Rosenbrock, GA seems to be unstable even for the true function. In terms of comparing the two optimization methods, both seem to work better than each other for different scenarios and equally good overall. Out of the three functins, we see that the results of PSO and GA are very good on the NN-proxy of Ackley function. The reason could be that the globally optimal solution of this function differs significantly from the locally optimal solutions and the NN-proxy can represent it very well.

Stability in higher dimensions. For a better approximation of the actual physics, the mathematical benchmarks need to be studied in higher dimensions. In this regard, we study the performance of NN-proxy in 4D and 10D also, and the related results are reported in Tables 2 and 3, respectively. With the increased number of dimensions, the error grows as expected. An interesting observation is that for higher dimensions, the Gaussian sampling scheme, which is expected to better model the landscape arond the true solution, seems to perform inferior compared to sparse sampling. The anticipated reason behind this issue is that too many samples around the true solution limit the number of samples that can be used to approximate the landscape far from it. This means that the function cannot be constructed well for regions far from it. This can lead to landscapes comprising false good solutions for the NN-proxy. In this regard, a better choice seems to be to go with uniform sampling.

3 Conclusions

In this paper, we have studied how the outcome of a physics-based optimization process can be adversely influenced when approximating the underlying physics process with a neural network proxy. We have demonstrated through experiments on popular mathematical benchmarks, that neural network approximations (NN-proxies) of such functions can lead to erroneous results for different training

Table 3: Mean Euclidean distance and standard deviation between the true optimal solutions of the three mathematical benchmarks in 10D and the respective solutions obtained for the NN-proxies using PSO and GA.

Function	Rosenbrock		Rastrigin		Ackley	
Optimizer	PS	GA	PS	GA	PS	GA
Dense	2.62 ± 0.46	2.95 ± 0.51	3.69 ± 0.53	4.45 ± 1.06	0.64 ± 0.01	0.70 ± 0.13
Sparse	3.37 ± 0.74	3.71 ± 0.51	2.20 ± 0.29	2.36 ± 0.56	1.19 ± 0.0	1.05 ± 0.08
Gaussian	5.31 ± 1.22	5.67 ± 1.53	6.25 ± 0.43	7.9 ± 2.6	0.30 ± 0.0	0.30 ± 0.07

setups. The correctness of the approximate model depends on the extent of sampling conducted in the parameter space, and caution needs to be taken when constructing this landscape with neural networks. Further, the NN-proxies are hard to train for higher dimensional functions, and through experiments on 4D and 10D problems, we demonstrated that the error in the optimized solution can be very high. The results reported in this paper are our first observations on the effect of NN-proxies on the optimization process. Clearly, we need to develop a rigorous benchmarking criterion to evaluate any novel NN-proxy and understand its robustness and generalization before it can be plugged in an optimization framework to replace a certain physics process.

4 Broader Impact Statement

This work focuses on understanding the error associated with the use of deep learning to replace the complex physics-based forward models used in optimization processes. While the initial concept in the paper is demonstrated on mathematical benchmarks, we hope that the observations and any countermeasures will be transferable on real world optimization problems involving physical simulations. Example of such problems includes proxy model for eigen value analysis in a compliant mechanism problem of topology and design optimization, among others. Overall, there are several problems where the results of this paper and the follow up will be useful.

Further, we do not see any ethical concerns of negative societal impact of this work.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [No] The work is still in the early phase, for now, limitations cannot be stated.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

References

[1] D.H Ackley. *A Connectionist Machine for Genetic Hillclimbing*, volume SECS28 of *The Kluwer International Series in Engineering and Computer Science.* Kluwer Academic Publishers, Boston, 1987.

[2] Mauricio Araya-Polo, Joseph Jennings, Amir Adler, and Taylor Dahlke. Deep-learning tomography. *The Leading Edge*, 37(1):58–66, 2018.

[3] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[4] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.

[5] J. Jumper, R. Evans, and A. et al. Pritzel. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021.

[6] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.

[7] Leonard Andreevič Rastrigin. Systems of extremal control. *Nauka*, 1974.

[8] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3):175–184, 01 1960.

[9] R. Vinuesa and S. L. Brunton. Enhancing computational fluid dynamics with machine learning. *Nature Computational Science*, 2:358–366, 2022.

[10] P. Zwartjes, D. Gupta, and T. Gupta. Near surface velocity estimation using surface waves and deep learning. *Proceedings, EAGE*, pages 1–5, 2022.

A Benchmark Functions

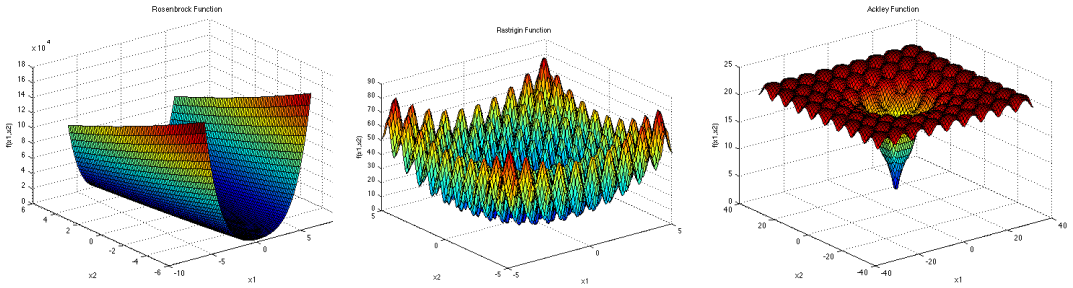
We use popular mathematical benchmark functions from the field of global optimization to train our NN-proxies. Our choice of functions are Rosenbrock, Rastrigin and Ackley. Figure 4 shows the functions plots.

Rosenbrock [8] is usually evaluated on the hypercube $x_i \in [-5, 10],\ \forall\ i = 1, \ldots, d$. We have restricted to the hypercube $x_i \in [-2.048, 2.048],\ \forall\ i = 1, \ldots, d$ and use it in our study for the sake of simplification. The mathematical formulation is stated in Eq. 1. The global minimum for this function is at $\mathbf{x} = [1, .., 1]$ and $f(x_i) = 0$.

Rastrigin [7] is usually evaluated on the hypercube $x_i \in [-5.12, 5.12],\ \forall\ i = 1, \ldots, d$, and the mathematical formulation for this function is stated in Eq. 2). The global minimum is at $\mathbf{x} = [0, .., 0]$ with $f(x_i) = 0$.

Ackley [1] is usually evaluated on the hypercube $x_i \in [-32.768, 32.768]\ \forall\ i = 1, \ldots, d$, it may be restricted to a smaller domain. For the mathematical representation of this function, see Eq. 3. The global minimum is at $x_i = [0, ..., 0]$ with $f(x_i) = 0$.

$$f(x) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] \tag{1}$$



(a) Rosenbrock (b) Rastrigin (c) Ackley

Figure 4: Three benchmark functions: Rosenbrock, Rastrigin and Ackley.

$$f(x) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)] \tag{2}$$

$$f(x) = -20exp(-0.2\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}) - exp(\frac{1}{d}\sum_{i=1}^{d} cos(2\pi x_i)) + 20 + e \tag{3}$$

B Optimization Methods

B.1 Particle Swarm Optimization

Particle Swarm Optimization was proposed by Kennedy and Eberhart [6]. It is inspired by the idea that a school of fish or a flock of birds that moves in a group 'can profit from the experience of all other members'. A brief overview of the underlying mechanism is described below.

Algorithm Details. Let there be P particles, where we denote the position of particle i at iteration t as $X^i(t)$. This position can then be stated as $X^i(t) = (x^i(t), y^i(t))$ and velocity as $V^i(t) = (v_x^i(t), v_y^i(t))$. At the next iteration, the position is updated as: $X^i(t+1) = X^i(t) + V^i(t+1)$ and the velocity is updated as:

$$V^i(t+1) = wV^i(t) + c_1 r_1(pbest^i \breve{} X^i(t)) + c_2 r_2(gbest \breve{} X^i(t)) \tag{4}$$

Where r_1 and r_2 are random numbers between 0 and 1. Constants w, c_1 and c_2 are inertia, learning rate for individual ability and social influence respectively. $pbest^i$ is the position that gives the best cost explored by particle i and $gbest$ is the best explored solution by all swarms.

B.2 Genetic Algorithm

The genetic algorithm [4] is a search heuristic that is inspired by Darwin's theory of natural evolution in which the fittest of individuals are the ones who survive. This fitness is measured by a fitness function. At every iteration, individuals with high fitness have more chance to be selected for reproduction to produce off springs for the next generation. The algorithm terminates if the population has converged i.e, offspring are not different from the previous generation or number of generations has been reached. For details related to implementation, see for example [3].

C Experiments: Additional Details

In this section we provide details about our experimental settings including the proxy models' architecture and training details.

We employ three different sampling mechanisms. For the dense sampling strategy, we use 10,000 samples for all cases to build the NN-proxies and these samples are equally spaced in the hyperspace.

For sparse case, we use 25% of the data that we use in the dense case. For the case of Gaussian sampling, we use the same number of points as the dense case, but the distribution follows a multidimensional Gaussian distribution.

In Dense case we use 10K points for training our models. In Sparse case we use 25% of the data. In Gaussian case, we use 10K centred around the functions' global minimum.

For Rosenbrock proxy model in 2D case, we employ a 3 layer multilayer perceptron (MLP) with hidden units of $\{15, 50, 15\}$ and RelU activation function. We train for 100 epochs with learning rate of 0.001 and Adam optimizer. In 4D case, we use a 4 layer MLP with hidden units of $\{15, 50, 15, 10\}$ and RelU activation. We train with the same number of epochs and similar learning rate and optimizer as 2D case. In 10D case, we use the same model setting as 4D case and train for 500 epochs with learning rate of 0.001 and Adam optimizer. All experiments were run on a NVIDIA RTX 6000 graphics card.

For Rastrigin and Ackley proxy models in 2D, 4D and 10D cases, we use 6 layer MLP with hidden units of $\{20, 50, 120, 70, 20, 10\}$. We use Relu activation and train for 500 epochs. Adam optimizer is used with learning rate of 0.001.