Nose to Tail: Uncertainty Quantification for the Inverse Design of Heat Conductive Nano-materials

Antonio Varagnolo

avaragnolo3@gatech.edu

Abstract

The ability to design materials with tailored thermal properties is critical for applications in electronics, energy storage, and advanced manufacturing. At nanoscale, heat conduction deviates from classical Fourier's law and requires description via the Boltzmann Transport Equation (BTE). However, solving the BTE numerically is computationally expensive, making it impractical for large-scale inverse design. To address this, data-driven surrogate models have emerged as a promising alternative, significantly accelerating the design process. Despite their success, these models often lack uncertainty quantification (UQ), which is essential for reliable optimization in a data-scarce regime. We will implement Physics Enhanced Deep Surrogates, a scientific ML architecture that has showed promising result in optics. We investigate if active learning can enhance the surrogate data-efficiency and design accuracy. To propagate uncertainty in the inverse design process, we explore combinations of ensemble methods, neural network surrogates, and polynomial chaos expansion (PCE). These approaches allow us to assess confidence in the PDE surrogate predictions, build a more informative dataset, and guide optimization decisions under uncertainty.

1 Introduction

The ability to design materials with desired thermal properties has become increasingly important across a range of applications, from heat management in electronics [3] to energy harvesting and storage technologies [4]. In particular, controlling heat conduction at the nanoscale has become as a serious challenge. Classical relations such as Fourier's law fail to capture the essential phenomenons caused by phonon-related properties such as ballistic transport, phonon-boundary scattering, or size effects when the sizes become comparable to the mean free paths of phonons [11]. As a result, the Boltzmann Transport Equation (BTE), a more accurate description of the heat transport relationships, have been proposed. However, the numerical solution of the BTE remains often computationally challenging, particularly for complex geometries required in inverse design problems [7, 12].

In recent years, surrogate modeling has become an important tool for speeding up the simulation and design process in computational physics [8]. Machine learning-based surrogates trained on high-fidelity data produce good alternatives by approximating the mapping between design parameters and system properties at a lower cost [14]. However, purely data-driven models often lack generalizability, especially in situations where data is scarce both as a result of experiments or high-fidelity simulations being expensive. Previous attempts at creating effective data surrogates have comprised the use of Multi-fidelity DeepONet [11]. Altough the creation of multi-fidelity dataset allows for better data-efficiency, the proposed model still requires a great amount of training points. This partially defeats the purpose of even creating a surrogate since the data generated for the training dataset could have just been used to directly optimize with the BTE. Furthermore, the lack of uncertainty quantification (UQ) in standard machine learning models further limits their reliability in optimization tasks, where the cost of inaccurate predictions may be even higher [13].

To address these problems, Physics-Enhanced Deep Surrogates (PEDS) was introduced as a hybrid model that combines the strengths of data-driven methods and lower-cost physics models [15]. By incorporating low-fidelity physics-based solvers into the machine learning architecture, PEDS can maintain physical constraints and improve predictive accuracy, especially for the out-of-sample distribution. Previous works have demonstrated the effectiveness of PEDS to approximate PDEs such as diffusion-reaction equations or Maxwell's equations, where they achieved higher accuracy and data efficiency compared to purely data-driven surrogates [15].

Another important improvement for more robust designs of surrogate models is the inclusion of uncertainty quantification. Once we obtain an uncertainty measure for our surrogate, it can be useful in multiple ways in the inverse design pipeline, from training dataset generation to optimization accounting for model uncertainty, from *nose to tail*, not wasting it in any part. Various techniques have been proposed to quantify uncertainty in Deep Learning [10] and in SciML [16]. In our work, where high-fidelity simulations are expensive, UQ not only enhances model interpretability but also enables active learning strategies, where the algorithm selectively queries the most informative designs [8]. This active learning loop significantly reduces the computation needed to achieve the desired level of accuracy.

2 Formal Problem Definition

When designing a nano-material geometry, we want to pick the location of pores that yields the desired effective thermal conductivity κ, a property of the material that describes its behavior as if it was macro-scale. Consider a 2D domain $\Omega = [-50,50] \times [-50,50]$ (nm) with periodic boundary conditions and an applied temperature gradient of 1K along the x-axis. The conductivity of the material is parametrized by a binary vector of size 25, corresponding to a grid of 5x5 embedded pores of size 10x10 (nm). This yields a total of approximately 33 million (2^{25}) possible configurations. The equation that governs heat conduction at nano-scale is the steady-state Boltzmann Transport Equation (BTE):

$$-v_\mu \cdot \nabla f_\mu(r) = \frac{f_\mu(r) - f_\mu^0(r)}{\tau_\mu} \qquad r \in \Omega = (-50,50) \times (-50,50) \tag{1}$$

With boundary conditions:

$$\begin{cases} f_\mu((x,-50)) = f_\mu((x,50)) + \left(\dfrac{\partial f_\mu}{\partial T}\right)\Delta T & \Delta T = 1\,\mathrm{K}, \qquad x \in (-50,50) \\ f_\mu((-50,y)) = f_\mu((50,y)) & y \in [-50,50] \end{cases} \tag{2}$$

The equilibrium distribution under relaxation time assumption can be written as a linear combination of other modes:

$$f_\mu^0(r) = \sum_{\mu'} \alpha_{\mu'} f_{\mu'}(r) \tag{3}$$

where

$$\alpha_{\mu'} = \frac{(c/\tau)_{\mu'}}{\sum_{\mu^*}(c/\tau)_{\mu^*}} \tag{4}$$

Our objective is to find the geometry p that yields a target effective thermal conductivity of $\hat{\kappa}$.

$$\begin{cases} J_y = \frac{\partial T}{\partial y} \\ \kappa = \int_{-50}^{50} J_y(0,y)dy \end{cases} \tag{5}$$

Where we chose to compute the integral along $x = 0$ as by convention.

3 Contributions

We claim the following early results and contributions:

1. The proposed surrogate model outperforms the MLP baseline, improving the model test set fractional error of at least 33%, and up to 59% on a dataset size of 100 data points.
2. The proposed uncertainty-based active learning framework allows to substantial computational savings at data generation time. At parity of validation set fractional error, our dataset building strategy allows us to save at least 37% and up-to 65% (350 training points vs 1000 training points) of generated data points. This is equivalent to around a week computation time on 4 CPUs.
3. Our current inverse design pipeline enables fast and super cheap repeated design from the 2nd design onwards with an average design error of around 5%.
4. We integrate uncertainty quantification directly into the design optimization process, implementing a stochastic optimization loss that balances target accuracy with model confidence.
5. Developed a new uncertainty quantification model to pair with Physics Enhanced Deep Surrogates.

4 Methods

4.1 PEDS for BTE

A physics-enhanced deep surrogate is a scientific machine learning model comprising of a computationally cheap low-fidelity solver paired with a neural network generator [15]. The neural network feeds a non-linear transformation of the design parameter space to the low fidelity solver that solves the equation and outputs the target property. The NN weights are learned contextually using backpropagation and adjoint simulation. The modified geometry can be a combination of the original geometry and of a learned residual, and it can enforce physically sound biases like symmetries. PEDS has previously shown great improvements in terms of accuracy and data efficiency compared to fully parametric surrogates in similar tasks, including surrogate modeling for diffusion and diffusion-reaction equations [15]. PEDS greatly fits the application requirements, as through its learnable parameter w it tunes its prediction to resemble the PE linear regime or the BTE non-linear regime effective conductivity.

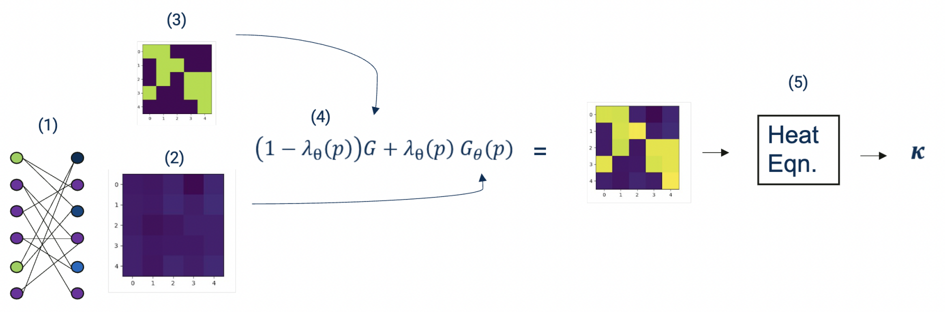

Figure 1: **PEDS Architecture**(1) Fully connected MLP with relu and hardtanh and layers width [25, 32, 32, 25], (2) Learnt 5x5 geometry, (3) Downsampled 5x5 geometry, (4) Linear combination coefficient learnt by a perceptron with sigmoid activation, (5) Low fidelity differentiable fourier solver

4.2 Solvers

We generate the ground truth steady-state equilibrium solution of the Boltzmann Transport Equation using OpenBTE [17]. The material dimension is 100x100 nm and is discretized with step size of 2 nm. The pores dimension is 10x10 nm. The most obvious choice for a low-fidelity approximation of the BTE is a simple Poisson equation (PE), or heat conduction equation. The PE is the macro-scale approximation of the BTE and it can be derived through simplification of the BTE. When the average conductivity approaches the bulk conductivity (i.e. in absence of pores), the BTE scalar temperature field approaches the PE solution. In this sense, the PE satifies the inclusion criteria as defined in [15]. We have implemented a direct solver and a Gauss Seidel solver but in the experiments we used the Fourier solver implemented in MatInverse, a library created by OpenBTE developers.

4.3 Adjoint Method

To efficiently compute the gradients of the PDE solution with respect to multiple design parameters, we exploit the adjoint simulation, also known as reverse-mode automatic differentiation or backpropagation, depending on your background. Instead of backpropagating through every operation or using forward numerical differentiation, it solves an auxiliary PDE — the adjoint equation. For self-adjoint operators, like our PE equation with periodic boundary conditions, the adjoint system is identical to the original. Thus, solving the adjoint is equivalent to one additional forward solve.

4.4 Uncertainty Quantification

Uncertainty quantification (UQ) and Scientific Machine Learning have always been very interconnected fields [16], and UQ plays a fundamental role in surrogate modeling when high-fidelity solvers are computationally expensive and experimental data is scarce. By modeling uncertainty, we can avoid overconfident predictions, prioritize highly-informative samples generation using Active Learning (AL), and make more robust decisions at optimization [Review papers]. This is particularly important in our setting, where one high-fidelity simulation (with a 1.5 nm grid) takes approximately 20 minutes on four CPU cores. UQ also enables Stochastic Optimization (SOPT) by accounting for confidence in the surrogate's output, rather than relying solely on point estimates. We implemented three models for UQ, each increasing in expressiveness and with varying computational complexities. The simplest UQ method is Deep Ensemble (UQ1), that consists of training multiple independent neural networks with different initializations and computes the empirical variance of their predictions [5]. This provides a simple measure of the uncertainty related to the model training increasing the cost linearly, but does not explicitly model the output as a distribution. The second method models the posterior distribution as a Gaussian, considering our surrogate point estimate as the mean $\kappa_\theta(x) \in \mathbb{R}^N$ and adding a NN for the log-variance $(\log \sigma^2)_\theta(x)$ (UQ2). We substitute the mean square error loss with a Gaussian negative log-likelihood loss, which enables us to learn mean and variance contextually.

$$loss = \sum_i \beta(\log \sigma^2)_\theta(x_i) + (1-\beta)\frac{(\kappa - \kappa_\theta(x_i))^2}{\exp(\log \sigma^2)_\theta(x_i)}$$

where β is a parameter that can be tuned to improve training convergence, especially in the cases where data is limited and the model tends to be over-confident. In order to account both for the heteroskedastic error and the optimization error, we trained an ensemble of these surrogate-variance model pairs and computed the ensemble mean as the empirical mean $\hat{\kappa}$ and the variance as:

$$\hat{\sigma} = \sum_{i=0}^{M} \sigma_i^2 + \sum_{i=0}^{M} (\kappa_i^2 - \hat{\kappa}^2) \tag{6}$$

where M is the number of models in the ensemble. In the most expressive setup (UQ3), we assume the input design parameters (e.g. our geometry) follow a multivariate Gaussian distribution with a positive definite covariance $K(x) \in \mathbb{R}^{N\times N}$. This distribution is parameterized by a neural network that a vector of variances $\sigma^2 \in \mathbb{R}^N$ and a low-rank matrix $U \in \mathbb{R}^{r\times N}$, where $r \ll N$ is the number of retained modes. The covariance is then built as $K = U^\top \operatorname{diag}(\sigma^2)U$, capturing correlations in a low-dimensional subspace compared to N. This distribution is then fed to a non-intrusive Polynomial Chaos Expansion (PCE) with r modes, enabling uncertainty propagation through the PDE. The choice of PCE is good because it proved to be effective and relatively inexpensive for PDEs that require small number of modes, like our diffusion [1]. At training time we are just using the mean and the variance of the propagated distribution, as those are necessary to compute the loss and to implement our AL algorithm. In principle, this reduces to a Monte Carlo method at training. Once the model is trained though, we have access to the full posterior by solving for the coefficients of the expansion. For simplicity, the co-location points were chosen through the evaluation of hermite polynomials, standard for input gaussian distributions but also other approaches were proposed in the past. [1] [6]

4.5 Active Learning

PEDS has previously shown great data savings and better performance on hard designs when paired with an Active Learning framework [15]. We have set up a basic uncertainty-based active learning pipeline by use of our uncertainty measures inspired by [8]. We initialize a dataset with N training points and then upon meeting of a querying criteria, we propose M new geometries, compute the uncertainty on this training points and add to the dataset the K most uncertain points. This is in contrast with other sample proposal strategies that are based upon the diversity of the samples or a mixture of diversity and uncertainty-based methods [2][9]. Differently from the approach proposed by [8], the querying criteria was chosen to be a dynamic convergence: if the change in validation set fractional error is less than Q then we propose new points. In our case the measure of uncertainty is simply the predicted variance. This is consistent with the view that for AL purposes, differently from more complicated data assimilation problems that require more refined descriptions of the distribution to pick our points, we are just trying to find a rough approximation of the uncertainty [10].

4.6 Optimization under uncertainty

Not to waste any of the work we previously did in building a model that includes an UQ, we try to account for this also at optimization time. To optimize taking in account our rough estimate of model uncertainty we put it inside the loss to minimize:

$$\min_x Loss(x) = |k - k_\theta(x)| + \lambda \sigma_\theta(x) \tag{7}$$

When available, the error and average variance on the test set can be used to tune the parameter $\lambda \approx \varepsilon/\bar{\sigma}$. In the case of the ensemble of PEDS this value is around 1.00 (uncertainty of a estimate at a point is proportional to its fractional error on the test / training set). We tried both a genetic algorithm and a gradient-based ADAM optimizer with a continuous relaxation of the 25 binary parameters using a smoothed approximation of the Heaviside function :

$$\tilde{\xi}(\xi) = \frac{\tanh(\beta\eta) + \tanh(\beta(\xi - \eta))}{\tanh(\beta\eta) + \tanh(\beta(1-\eta))} \tag{8}$$

where ξ is the relaxed parameter, η is the threshold (typically $0.3 \leq \eta \leq 0.7$), and β controls the steepness. Initially, β is set small to allow smooth optimization, and then progressively increased (e.g., doubled each stage) to encourage convergence toward binary designs, following the continuation strategy in [15].

5 Results

5.1 Accuracy Comparison

In our first experiment we compared the two architectures and the ensemble training on a dataset of randomly sampled 1000 geometry-conductivity pairs. The validation set and test set are composed of 100 pairs also randomly sampled. We trained for 1000 epochs with an ADAM optimizer and a cosine schedule with maximum learning rate of $5 \cdot 10^{-3}$ and a minimum of $5 \cdot 10^{-4}$. PEDS architecture was chosen to be relatively small with 2 fully-connected hidden layers of size 32 and a resolution of 1 (5x5 grid). Consistently with previous work, bigger resolutions seem not to yield any improvement. The activation functions are relu beside the final one that is hardtanh outputting values from 1 to 157, the bulk conductivity of silice. Consequently, the initialization was chosen to be xavier normal, state-of-the-art for relu non-linearities. The learnable parameter $w_\theta(x)$ controlling the geometries linear combination is the output of a fully connected perceptron with a sigmoid activation and a kaiming initialization. The MLP baseline architecture was chosen to resemble PEDS', having the same number of parameters. The training was parallelized and performed on 4 chores. The training curves reported below are the average of 5 different runs performed with different initializations and datasets. In 4 we report test set mean square error and the fractional error while in figure 3 we show the average test loss for 5 trainings. The fractional error is computed as:

$$fracterror = \frac{|\kappa - \kappa_\theta|}{|\kappa|}$$

Models	MSE test	Fract Error (%)
MLP	5.53	7.26
PEDS	2.85	4.91
MLP Ensemble	8.93	9.29
PEDS Ensemble	2.81	4.12

Table 1: Average of 5 trainings with 5 different seeds

We then report the final design error results of the best MLP seed and the **worst** PEDS and ENSEMBLE seed using the gradient-based algorithm. The final design error is the sum of generalization error of the model and the optimization error of the optimizer. The MLP tends to focus only on the region of the output distribution going from 16 to 50 where we have the most samples. In this region they perform similarly while PEDS performs better on under-sampled regions and out-of-distribution samples.

Kappa Target	Model	Kappa Optimized	Generalization Error	Design Error
	MLP	17.08	0.37	0.42
12.01	PEDS	14.57	0.09	0.21
	ENSEMBLE	12.36	0.05	0.03
	MLP	13.25	0.12	0.12
15.00	PEDS	13.25	0.12	0.12
	ENSEMBLE	14.16	0.06	0.06
	MLP	19.07	0.05	0.05
20.00	PEDS	20.35	0.02	0.02
	ENSEMBLE	19.90	0.005	0.005
	MLP	27.78	0.07	0.07
30.00	PEDS	26.88	0.10	0.10
	ENSEMBLE	30.50	0.02	0.02
	MLP	50.39	0.12	0.12
44.98	PEDS	50.86	0.13	0.13
	ENSEMBLE	44.15	0.02	0.02
	MLP	83.30	0.46	0.38
59.99	PEDS	64.48	0.07	0.07
	ENSEMBLE	63.15	0.05	0.05

Table 2: Design breakdown for the best trained MLP and worst PEDS and Ensemble on selected target values

5.2 Data efficiency comparison

To test the data efficiency of our models we have tried the same experiment with progressively smaller training dataset sizes, keeping 100 validation and test points. In figure 2, we report the curve with the dataset size on the x-axis and the final test set error on the y-axis (an average of 2 seeds). PEDS achieves the same error or less with half the data points required by the MLP (more than X2 data points to reach 11% and 7 %, almost X4 to reach 6%). Consistently with literature, ensembling 4 models advantages PEDS more than MLPs.

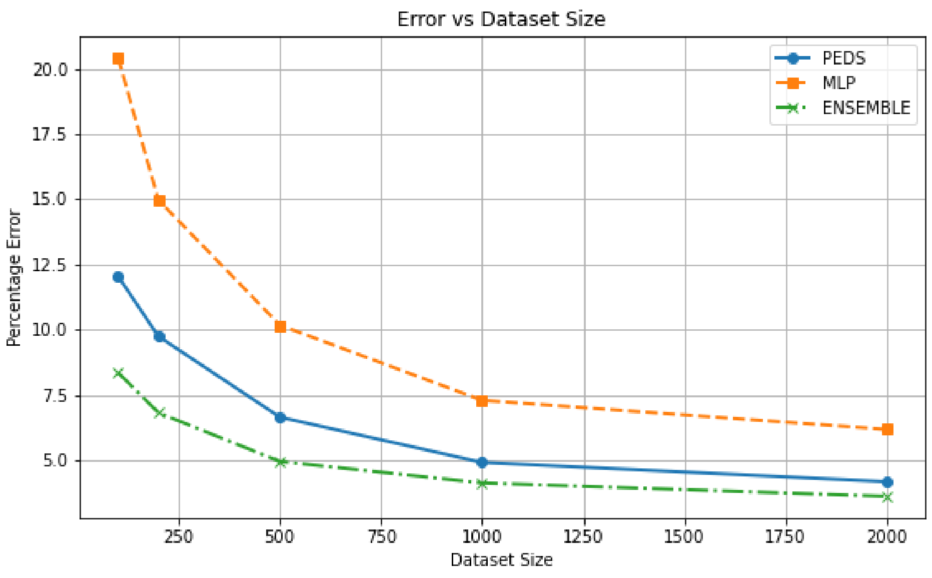

Figure 2: Fractional Errors on different dataset sizes (average of 2 trainings)

To quantify uncertainty we used a Deep Ensemble of 4 models (UQ1). The AL were chosen to be $N = 40$, $K = 200$, $M = 25$, $Q = 0.025\%$. The use of active learning not only allows us to save computing by evaluating less times the expensive high-fidelity solver but it also makes our generalization more robust (see 5) by oversampling regions of the space in which our model performs poorly. AL oversamples the right tail of our geometries distribution, corresponding to higher effective kappas. This is because our model fist learns the basic modes of the posterior of the non-linear BTE, then tries to improve also bigger kappas that are in between the BTE and PE regime. AL yields good results in terms, reaching the same fractional error with a reduction of at least 30% in data generated and of 45% for our base case (650 training points instead of 1000). Performances on designs is reported in a table in the appendix 5.

Target Fractional Error (%)	Data No AL	Data AL	Save (%)
8.21	100	65	37
6.81	200	125	38
4.91	500	350	30
4.12	1000	650	45

Table 3: Data Efficiency of PEDS with UQ1 using AL

Now we try to check if increasing the complexity of uncertainty estimation, we can obtain significant improvement in our two UQ-related tasks. Below we report AL results using the two first uncertainty models. The hidden layer sizes for our UQ2 network outputting the log variance are [32, 32], the activations are relu and initialization is xavier.

Target Fractional Error (%)	Data No AL	Data AL UQ1	Save UQ1 (%)	Data AL UQ2	Save UQ2 (%)
8.21	100	65	40	65	40
6.81	200	125	37	100	50
4.91	500	350	30	250	50
4.12	1000	650	45	350	65

Table 4: Data Efficiency due to AL

The second UQ method outperforms the first in the active learning task. This is partially in contrast with the literature that associates marginal improvement to more expressive and computationally expense UQ models. The improvement can be attributed to the modeling of more than one type of uncertainty, including not only the optimization error but also an heteroskedastic variance, and reflecting the fact that some designs are indeed harder to guess, highlighting more non linearity.

5.3 Time Considerations

Our main objective is to create a surrogate that can enable accurate time-efficient design. In cost-benefit terms, our proposed pipeline for design tries to find a good trade-off between the high fixed costs of generating data and training the surrogate, and the high variable computational costs of optimizing directly with the ground truth solver. Coherently with the economic metaphor, the fundamental success metric is the break-even number of designs for which choosing our strategy yields time savings. This number can be found using the following simple equation:

$$K * N_{opt} * T_{BTE} = K * N_{opt} * T_{PEDS} + T_{BTE} * M + T_{train} \tag{9}$$

where K is the number of designs, N_{opt} is the number of evaluations needed to optimize a specific effective conductivity, M is the number of points generated to train the model and $T_{BTE}, T_{PEDS}, T_{train}$ are respectively the BTE evaluation time, surrogate evaluation time and the training fixed time. We could also account for the small percentage error of our model but, depending on the application, it is often negligible compared to the average fabrication error.

In our case these numbers are $N_{opt} = 300$, $M = 350$, $T_{BTE} = 1200s$, $T_{PEDS} = 0.001s$, $T_{train} = 2400s$, yielding $K \geq 1.17 \approx 2$.

5.4 Future Work (In progress)

In the following weeks we will be working on the following:

1. Fine-tune the Stochastic Optimization algorithm and get quantitative results on its performance;
2. Get a better quantitative result on the final designs;
3. Test results on UQ3;
4. We plan to investigate if we can generalize this results of PEDS+AL+SOPT on more multi-scale problems. Is this pipeline a natural choice for problems with such a dependence on scale?
5. It also generates the field that can be used as a starting point.

6 Code and Data availability

The code is available at this github repository, specifically at the experiments branch. The Fourier solver used for final trainings is part of Matinverse, a package not yet released. Comparable results can be reproduced using the Gauss-Seidel solver or direct solver. The data is available.

References

[1] Dongbin Xiu and George Em Karniadakis. *Modeling Uncertainty in Steady State Diffusion Problems via Generalized Polynomial Chaos*. Tech. rep. Defense Technical Information Center, 2002. DOI: 10.21236/ada460658.

[2] Burr Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012. DOI: 10.1007/978-3-031-01560-1.

[3] David G Cahill et al. "Nanoscale thermal transport. II. 2003–2012". In: *Applied Physics Reviews* 1.1 (2014), p. 011305.

[4] Zhong Lin Wang, Zong-Hong Wang, and Ya Yang. "All-in-one energy harvesting and storage devices". In: *Journal of Materials Chemistry A* 4.38 (2016), pp. 14686–14704. DOI: 10.1039/C6TA01229A.

[5] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles". In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017, pp. 6402–6413. URL: https://proceedings.neurips.cc/paper/2017/file/9ef2ed4b7fd2c810847ffa5fa85bce38-Paper.pdf.

[6] Mohammad Hadigol and Alireza Doostan. "Least squares polynomial chaos expansion: A review of sampling strategies". In: *Computer Methods in Applied Mechanics and Engineering* 332 (2018), pp. 382–407. DOI: 10.1016/j.cma.2017.12.019.

[7] A. Zunger. "Inverse Design in Search of Materials with Target Functionalities". In: *Nature Reviews Chemistry* 2.4 (2018). DOI: 10.1038/s41570-018-0121.

[8] R. Pestourie et al. "Active Learning of Deep Surrogates for PDEs: Application to Metasurface Design". In: *NPJ Computational Materials* 6.1 (2020). DOI: 10.1038/s41524-020-00431-2.

[9] Changjian Shui et al. "Deep Active Learning: Unified and Principled Method for Query and Training". In: *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*. Ed. by Silvia Chiappa and Roberto Calandra. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 1308–1318. URL: https://proceedings.mlr.press/v108/shui20a.html.

[10] Moloud Abdar et al. "A review of uncertainty quantification in Deep learning: Techniques, applications and challenges". In: *Information Fusion* 76 (2021), pp. 243–297. DOI: 10.1016/j.inffus.2021.05.008.

[11] L. Lu et al. "Multifidelity Deep Neural Operators for Efficient Learning of Partial Differential Equations with Application to Fast Inverse Design of Nanoscale Heat Transport". In: *Physical Review Research* 4.2 (2022). DOI: 10.1103/physrevresearch.4.023210.

[12] G. Romano and S. G. Johnson. “Inverse Design in Nanoscale Heat Transport via Interpolating Interfacial Phonon Transmission”. In: *Structural and Multidisciplinary Optimization* 65.10 (2022). DOI: `10.1007/s00158-022-03392-w`.

[13] M. Swaminathan et al. “Bayesian Learning for Uncertainty Quantification, Optimization, and Inverse Design”. In: *IEEE Transactions on Microwave Theory and Techniques* 70.11 (2022), pp. 4620–4634. DOI: `10.1109/tmtt.2022.3206455`.

[14] Jared Willard et al. “Integrating Scientific Knowledge with Machine Learning for Engineering and Environmental Systems”. In: *ACM Computing Surveys* 55.4 (2022), pp. 1–38. DOI: `10.1145/3514228`.

[15] Raphaël Pestourie et al. “Physics-Enhanced Deep Surrogates for PDEs”. In: *Nature Machine Intelligence* (2023). Currently in press. DOI: `10.48550/arXiv.2111.05841`.

[16] Alexandros F Psaros et al. “Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons”. In: *Journal of Computational Physics* 477 (2023), p. 111902. DOI: `10.1016/j.jcp.2022.111902`.

[17] Giuseppe Romano and contributors. *OpenBTE: An open-source solver for the Boltzmann Transport Equation*. `https://github.com/romanodev/OpenBTE`. Accessed: 2025-04-27. 2024.

7 Appendix

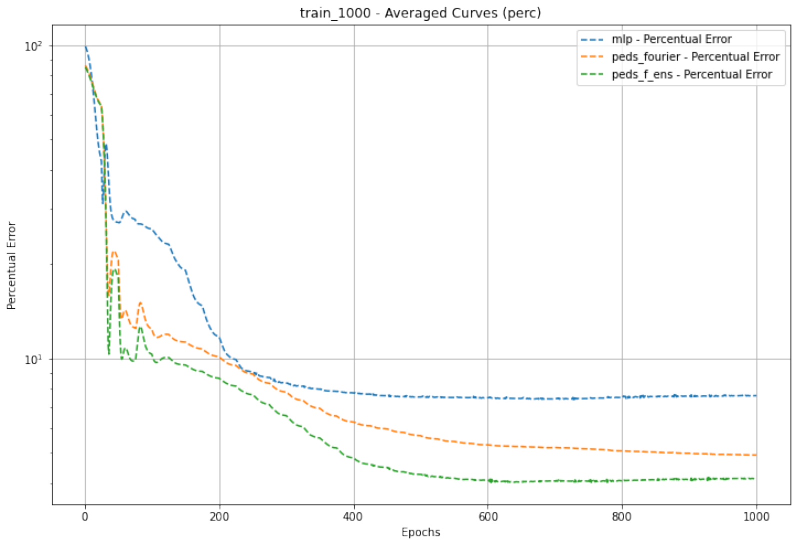

Figure 3: Fractional Errors over 1000 epochs

Kappa Target	Model	Kappa Optimized Variance Predicted	Generalization Error	Design Error
12.0	ENS	12.36 0.16	0.05	0.03
	ENS+AL	12.93 1.61	0.07	0.03
	ENS+AL+OPT	14.18 0.61	0.03	0.18
15.0	ENS	14.16 0.54	0.06	0.06
	ENS+AL	16.89 1.61	0.12	0.12
	ENS+AL+OPT	14.12 0.19	0.06	0.06
20.0	ENS	19.90 0.34	0.005	0.005
	ENS+AL	19.96 1.36	0.002	0.002
	ENS+AL+OPT	17.80 0.14	0.11	0.11
30.0	ENS	30.50 0.56	0.02	0.02
	ENS+AL	29.71 2.49	0.01	0.01
	ENS+AL+OPT	29.68 0.20	0.01	0.01
45.0	ENS	44.15 1.18	0.02	0.02
	ENS+AL	43.91 1.25	0.02	0.02
	ENS+AL+OPT	45.76 0.62	0.02	0.02
60.0	ENS	63.15 13.73	0.05	0.05
	ENS+AL	56.48 5.95	0.06	0.06
	ENS+AL+OPT	58.43 0.32	0.03	0.03

Table 5: Design breakdown for each model on selected target values, using UQ1