Novel Approaches to Discovery and Optimization in Physics: Symbolic Regression, Bayesian Optimization, and Topological Photonics

by

Samuel Kim

A.B., Harvard University (2015)
S.M., Massachusetts Institute of Technology (2019)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2023

Authored by: Samuel Kim
Department of Electrical Engineering and Computer Science
March 24, 2023

Certified by: Marin Soljačić
Professor of Physics
Thesis Supervisor

Accepted by: Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Novel Approaches to Discovery and Optimization in Physics: Symbolic Regression, Bayesian Optimization, and Topological Photonics

by

Samuel Kim

Submitted to the Department of Electrical Engineering and Computer Science
on March 24, 2023, in partial fulfillment of the
requirements for the degree of
DOCTOR OF PHILOSOPHY

Abstract

Computational tools including high-fidelity simulations, optimization algorithms, and more recently, machine learning, have become increasingly important in furthering scientific and engineering innovations as available computational power and computing methodologies have both advanced significantly. However, as our understanding of the world and the systems we study correspondingly increase in complexity, there is still a need for designing novel computational methods. In this thesis, I describe three major innovations in which I use optimization and machine learning to automate scientific discovery, optimization, and inverse design.

First, I propose a neural network-based method to perform symbolic regression and automatically learn the underlying equations from high-dimensional and complex datasets. The neural network-based model can integrate with other deep learning architectures, thus taking advantage of the powerful capabilities of deep learning for the task of scientific discovery. Second, I demonstrate the usage of Bayesian neural networks as a surrogate model in Bayesian optimization to enable global optimization of high-dimensional, non-convex problems including topology optimization of photonic crystals and chemical property optimization of molecules. On these complex tasks, my method is able to outperform more commonly used surrogate models and improve optimization in terms of both sampling efficiency and computational cost of training. Finally, I develop a framework for global optimization and automated discovery of 3D topological photonic crystals using a combination of a low-dimensional level-set parameterization and standard gradient-free optimization algorithms. My approach is able to discover novel 3D photonic crystals in several topology settings requiring no prior knowledge of topological candidates, thus indicating a path towards the automated discovery of novel topological photonic crystal designs.

Acknowledgments

First and foremost, I am extremely grateful to my advisor, Professor Marin Soljačić. Even before starting grad school, I was inspired by his audacious vision for the future of machine learning and his daringness in tackling new topics. At a time where it seemed liked the scientific community was still figuring out how machine learning could intersect with physics, Marin had big visions for using AI to solve science and offered me the opportunity to explore this intersection. I am grateful for his patience, guidance, and mentorship.

I have had the privilege of learning from numerous distinguished and accomplished researchers at MIT. Professor Steven Johnson has been a great source of knowledge and excitement in photonics and computational methods. His class on Mathematical Methods in Nanophotonics was one of my favorite classes at MIT, and he has always been excited to help me throughout my research. Dr. Thomas Christensen has been a fantastic mentor over the last year and a half of my Ph.D., teaching me everything I know about topological photonics and guiding me to be a better researcher. I am grateful for my thesis commitee, Professor Luca Daniel and Professor Isaac Chuang, who have given me valuable guidance on my thesis and pushed me to become a better researcher.

I am thankful to the entire Soljačić research group, who have all been a delight to hang out and work with, and in particular, the machine learning subgroup within the group–Peter Y. Lu, Charlotte Loh, Rumen Dangosvki, and Andrew Ma. Even when not collaborating directly, we have spent countless number of hours brainstorming, bouncing ideas off of each other, and sharing research papers, especially though the pandemic lockdown when we perhaps needed it the most. It has also been a pleasure mentoring and working with numerous undergraduate students through MIT's UROP program who have all been extremely talented and ambitious, including Srijon Mukherjee, Ilan Mitnikov, Michael Gilbert, Amber Li, Michael Zhang, Cem Tepe, and Anka Hu.

In terms of funding, I am most grateful for the National Defense Science and

Engineering Graduate (NDSEG) Fellowship, which fully supported me for three years, put trust in me as an aspiring researcher, and given me significant flexibility in my research direction.

Many thanks to my collaborators at Google, Jasper Snoek and Jamie Smith, who have been an amazing resource for their expertise in machine learning. I would also like to thank my former colleagues at the Johns Hopkins Applied Physics Laboratory (JHU APL)–including David Shrekenhamer, Jacob Alldredge, Joe Miragliotta, Ra'id Awadallah, and the rest of the Experimental and Computational Physics group–who have all been terrific mentors to me, introduced me to the world of electromagnetic metamaterials and computational electromagnetics, and supported me in my plans in pursuing graduate school.

Of course, I am deeply indebted to all my friends and family who have supported me through this journey. My blocking group from college–Erik Schluntz, Joey Kim, Andy Gonzalez, Nick Merrill, Ian Choi, Crystal Stowell, Tiffany Lim, Amanda Reilly, and Miranda Chang–have always been an anchor for me, even when we are spread out across the country. My roommates Tin (Stan) D. Nguyen, Arish Shah, and Dylan McCormick, my partner Amy Miao, and my family have all been a huge support and kept me sane.

Taekwondo has been a huge part of my life, and I would like to sincerely thank all of the taekwondo communities I have been a part of and all the people I have trained with. Thanks to the MIT Sport Taekwondo club, including the "old people" crew and the instructors–Stan Nguyen, Tahin Syed, Renee Zhao, Joy Zeng, Jialin Ding, Nick Hensel, Ashley Wang, and Isaac Fenta. Thanks also to the t.Bos competition team at C.W. Taekwondo Boston, including the freestyle crew–Andrew Hurd, Leah Rosenzweig, Kyra Chan, Kevin Lu, and Elizabeth Zou. Finally, thanks to Master Dan Chuang, has been so welcoming and supportive of my training throughout many ups and downs.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Computational sciences have enabled the study, analysis, and design of extremely complex systems in science and engineering. For example, Maxwell's equations, which may seem simple at the surface, are incredibly rich in physics and give rise to numerous phenomena in electromagnetics, optics, and photonics, not to mention the ubiquity in which it touches all of the other sub-fields of physics. Sophisticated simulation methods and optimization techniques have been able to harness the endless potential in Maxwell's equations to investigate novel phenomena and design new devices. On the other hand, deep learning and machine learning have become extremely powerful, easily outperforming humans on tasks previously thought to be impossible, and can potentially also make large impact on the sciences. While I have studied a large variety of seemingly disparate topics over the course of my PhD, they all share a common backbone of applying computational approaches to problems in physics. More specifically, my projects fall in three themes: (1) machine learning for scientific discovery, (2) optimization algorithms for problems in science and engineering, and (3) computational design in photonics.

In Chapter 2, I explore the use of machine learning to aid and even automate the process of scientific discovery. As experiments and datasets grow larger and more complex, there is a need to turn to computational algorithms to aid scientists in sifting through the data and glean insights. I propose the use of a neural network architecture that can discover equations from data in a process called symbolic regression. Symbolic regression is a powerful technique to discover analytic equations that describe data, which can lead to explainable models and the ability to predict unseen data. In contrast, neural networks have achieved amazing levels of accuracy on image recognition and natural language processing tasks, but they are often seen as black-box models that are difficult to interpret and typically extrapolate poorly. I demonstrate how we can integrate this architecture with various types of powerful deep learning architectures and train end-to-end through backpropagation to enable symbolic regression on high-dimensional data. I also propose several extensions of this architecture to enable symbolic regression on data described by parametric equations in which the underlying equation structure may remain constant but the coefficients may vary in arbitrarily complex ways. The neural network-based architecture for symbolic regression is able to extrapolate well outside of the training data set compared with standard neural networks while extracting meaningful equations that are interpretable by humans, paving the way to automate scientific exploration and discovery. Perhaps machine learning methods will eventually be able to make strides in many unsolved questions such as the mechanism of high-temperature superconductivity or a model for turbulent flow. This chapter is partially based on work that has been published in Ref. [99] and reported in Ref. [263].

In Chapter 3, I propose a method to combine deep learning and Bayesian optimization (BO) to tackle the optimization and inverse design of problems with high-dimensional structure. Bayesian optimization (BO) is a popular paradigm for global optimization of expensive black-box functions, but there are many domains where the function is not completely a black-box. The data may have some known structure (e.g. symmetries) and/or the data generation process may be a composite process that yields useful intermediate or auxiliary information in addition to the value of the optimization objective. However, surrogate models traditionally employed in BO, such as Gaussian Processes (GPs), scale poorly with dataset size and dimensionality and do not easily accommodate known structure. Instead, we use Bayesian neural networks, a class of scalable and flexible surrogate models with inductive biases, to extend BO to complex, structured problems with high dimensionality. We demonstrate

BO on a number of realistic problems in physics and chemistry, including topology optimization of photonic crystal materials using convolutional neural networks, and chemical property optimization of molecules using graph neural networks. On these complex tasks, we show that neural networks often outperform GPs as surrogate models for BO in terms of both sampling efficiency and computational cost. As this work is able to optimize extremely high-dimensional problems without the use of gradients or a hand-crafted low-dimensional parameterization, it indicates a path towards more flexible and automated inverse design with reduced requirements for domain expertise. This work has been published in Ref. [100].

Finally, in Chapter 4, I focus on the optimization and design of 3D topological photonic crystals. The combination of powerful gradient-based optimization methods and the flexibility of nanoscale fabrication techniques have motivated and enabled topology optimization of photonic crystals that can give rise to extremely complex geometries. However, photonic crystal design is still a difficult problem since the objective is non-differentiable and non-convex. Here I propose a method for global optimization of 3D topological photonic crystals using a combination of strategies to handle the difficult search space and a flexible level-set function to parameterize the photonic crystal with a handful of parameters while still allowing for complex geometries. Whereas most reported topological photonic crystals have been typically carefully designed by hand with significant physical insight, my method is able to automatically discover novel topological photonic crystals. Furthermore, we propose several different types of topological photonic crystals that significantly surpass the performance of any photonic crystal that has been reported so far. This work has been published in Ref. [101].

Chapter 2

Integration of Neural Network-Based Symbolic Regression in Deep Learning for Scientific Discovery

2.1 Introduction

Discovering the governing equations of nature is key to many scientific disciplines, as many complex phenomena can be reduced to general models that can be described in terms of relatively simple mathematical equations. For example, classical electrodynamics can be described by Maxwell's equations, harmonic oscillators by Hooke's law, and non-relativistic quantum mechanics by the Schrödinger equation. These models elucidate the underlying dynamics of a particular system and can provide general predictions over a very wide range of conditions. While scientists have often spent years developing insights to discover these equations, machine learning has become alluring in its potential to tackle and automate extremely complex tasks.

Modern machine learning techniques have become increasingly powerful for many tasks including image recognition and natural language processing. In recent years, the power of deep learning has expanded to creating photorealistic or artistic images from captions [185] and predicting a protein's 3D structure [95] more quickly and accurately than any other algorithm. However, the neural network-based architectures in these state-of-the-art techniques are black-box models that often make them difficult for use in scientific exploration. In order for machine learning to be widely applied to science, there needs to be interpretable models that can extract meaningful information from complex datasets and make useful predictions outside of the training dataset.

Symbolic regression is a type of regression analysis in machine learning that searches the space of mathematical expressions to find the best model that fits the data, and can thus fit a much wider range of datasets than other models such as linear regression. Assuming that the resulting mathematical expression correctly describes the underlying model for the data, it is easier to interpret and can extrapolate better than black-box models such as neural networks. Symbolic regression is typically carried out using techniques such as genetic programming, in which the structure of the mathematical expression is found using evolutionary algorithms to best fit the data, while ensuring that the equation is viable through various heuristics [107]. The equations are pieced together through basic building blocks known as primitive functions, which include constants and simple functions (e.g. addition, multiplication, sine). Schmidt and Lipson [199], one of the most popular earlier works in this direction, demonstrated how symbolic regression could discover equations of motions including Hamiltonians and Lagrangians for various physical systems from experimental data. However, due to the combinatorial nature of the problem, these genetic programming approaches do not scale well to high-dimensional problems and can be prone to overfitting, often requiring numerous hand-built heuristics and rules.

Alternative approaches to finding the underlying laws of data have been explored. For example, a method called SINDy combines sparsity with regression techniques and numerically-evaluated derivatives to find partial differential equations (PDEs) that describe dynamical systems [19, 193, 197].

There has been significant work on designing neural network architectures that are either more interpretable or contain inductive biases that make them more suitable for scientific exploration. Neural networks with unique activation functions that correspond to primitive functions have been proposed to perform symbolic regression [144, 195]. A deep learning architecture called the PDE-Net has been proposed to predict the dynamics of spatiotemporal systems and produce interpretable differential operators through constrained convolutional filters [125, 126]. Trask et al. [223] proposed a neural network module called the Neural Arithmetic Logic Unit (NALU) that introduces inductive biases towards simple arithmetic operations so that the architecture can extrapolate well on specific tasks. Neural network-based architectures have also been used to extract relevant and interpretable parameters from dynamical systems and use these parameters to predict the propagation of a similar system [269, 134]. Additionally, Chari et al. [23] use symbolic regression as a separate module to discover kinematic equations using parameters extracted from videos various types of motion.

There have also been numerous approaches at introducing the power of deep learning into symbolic regression to enable it for more complex tasks. For example, AI-Feynman checks for a number of physics-inspired invariances and symmetries using both hand-built rules and neural networks to simplify the data [226]. Neural network autoencoders have been combined with SINDy to enable equation discovery on high-dimensional dynamical systems [22]. PDE-Net 2.0 incorporates a symbolic network to discover PDEs using convolutional networks with constrained filters [126]. Lu et al. [135] incorporates a symbolic network with a neural network encoder to discover ODE and PDE systems from partial observations. Cranmer et al. [31] performs traditional symbolic regression on graph neural network weights in a 2-step process to discover the dynamics of many-body systems.

Here we present a neural network architecture for symbolic regression that is integrated with other deep learning architectures so that it can take advantage of powerful deep learning techniques while still producing interpretable and generalizable results. Our base neural network is similar to EQL network containing primitive functions as activation functions, which was developed in parallel to our work [144, 195]. Because this symbolic regression method can be trained through backpropagation, the entire system can be trained end-to-end without requiring multiple steps.

Sections 2.2-2.4 details how the EQL network can be integrated into deep learning including convolutional networks and recurrent networks to perform symbolic regression on high-dimensional and dynamical systems. This work has been published [99]. Ref. [28] further extended this for recursive programs, implicit functions, and image classification. Furthermore, Sections 2.5-2.8 details an extension to the EQL network to tackle parametric equations in which the coefficients may vary in complex, non-symbolic ways while the base structure of the equation remains constant. This has also been reported [263].

Source code for the original EQL experiments is made publicly available in Tensorflow[1] and PyTorch[2]. Code for the extensions to parametric equations is also available[3].

2.2 EQL Architecture

The symbolic regression neural network we use is similar to the Equation Learner (EQL) network proposed in [144, 195]. As shown in Figure 2-1, the EQL network is a fully-connected neural network where the i^{th} layer of the neural network is described by

$$\mathbf{g}_i = \mathbf{W}_i \mathbf{h}_{i-1} \tag{2.1}$$

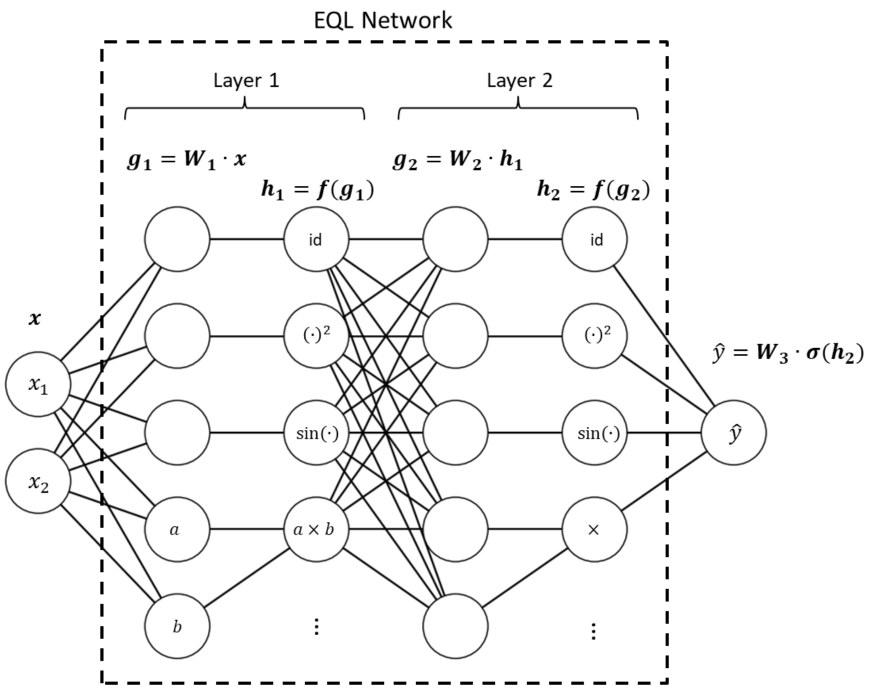
$$\mathbf{h}_i = f\left(\mathbf{g}_i\right) \tag{2.2}$$

where $\mathbf{W}_i$, $\mathbf{g}_i$, and $\mathbf{h}_i$ are the weight matrix, pre-activation units, and post-activation units of the i^{th} layer, and $\mathbf{h}_0 = \mathbf{x}$ is the input data. The final layer does not have an activation function, so for a network with L hidden layers, the output of the network y is described by

$$y = \mathbf{h}_{L+1} = \mathbf{W}_{L+1}\mathbf{h}_L.$$

The activation function $f(\mathbf{g})$, rather than being the usual choices in neural net-

[1]`https://github.com/samuelkim314/DeepSymReg`
[2]`https://github.com/samuelkim314/DeepSymRegTorch`
[3]`https://github.com/samuelkim314/topo-phc-opt`



EQL Network
Layer 1
Layer 2
$\boldsymbol{g_1 = W_1 \cdot x}$
$\boldsymbol{h_1 = f(g_1)}$
$\boldsymbol{g_2 = W_2 \cdot h_1}$
$\boldsymbol{h_2 = f(g_2)}$
$\boldsymbol{x}$
x_1
x_2
id
$(\cdot)^2$
$\sin(\cdot)$
$a \times b$
a
b
$\times$
$\hat{y} = \boldsymbol{W_3 \cdot \sigma(h_2)}$
$\hat{y}$

Figure 2-1: **Equation Learner (EQL) network architecture for symbolic regression.** The architecture resembles a fully-connected neural network, but with the activation functions replaced by the primitive functions. Here we show only 4 activation functions (identity or “id", square, sine, and multiplication) and 2 hidden layers for visual simplicity, but the network can include more functions or more hidden layers to fit a broader class of functions.

works such as ReLU or tanh, may consist of a separate function for each component of $\mathbf{g}$ and may include functions that take two or more arguments while producing one output (such as the multiplication function):

$$f(\mathbf{g}) = \begin{bmatrix} f_1(g_1) \\ f_2(g_2) \\ \vdots \\ f_{n_h}(g_{n_g-1}, g_{n_g}) \end{bmatrix} \tag{2.3}$$

Note that the additive bias term can be absorbed into $f(\mathbf{g})$ for convenience. These activation functions in (2.3) are analogous to the primitive functions (such as sine or the square function) in symbolic regression. Allowing functions to take more than one argument allows for multiplicative operations inside the network.

While the schematic in Figure 2-1 only shows 4 activation functions in each hidden layer for visual simplicity, $f(\mathbf{g})$ in 2.3 can include other functions including $\exp(g)$ and $\text{sigmoid}(g) = \frac{1}{1+e^{-g}}$. Additionally, we allow for activation functions to be duplicated within each layer (i.e., multiple components in g can use the same activation function). This reduces the system's sensitivity to random initializations and creates a smoother optimization landscape so that the network does not get stuck in local minima as easily. This also allows the EQL network to fit a broad range of functions. More details can be found in Appendix A.1.

By stacking multiple layers (i.e. $L \geq 2$), the EQL architecture can fit complex combinations and compositions of a variety of primitive functions. Note that L is analogous to the maximum tree depth in genetic programming approaches and sets the upper limit on the complexity of the resulting expression. While this model is not as general as conventional symbolic regression, it is powerful enough to represent most functions that are typically seen in science and engineering. More importantly, because the EQL network can be trained by backpropagation, it can be integrated with other neural network-based models for end-to-end training.

2.2.1 Related Work

Sine has been used as an activation function in neural networks [112, 213, 166, 271]. More generally, Fourier neural networks use a combination of sines and cosines with varying frequencies to mimic the behavior of a Fourier series [228]. These architectures were proposed as an alternative to the more popular sigmoidal activation functions, but are still generally difficult to train and do not match the state-of-the-art on real-world datasets.

Fletcher and Hinde [44] proposed a general activation function that was a linear combination of sine and sigmoid functions to improve representational power of a one-layer neural network. [54] use a mixture of sine, softplus, and identity functions for the activation functions, combined with regularization to shift the weights towards the linear units and simplify the model.

2.2.2 Sparsity

A key ingredient of making the results of symbolic regression interpretable is enforcing sparsity regularization such that the system finds the simplest possible equation that fits the data. The goal of sparsity regularization is to set as many weight parameters to 0 as possible such that those parameters are inactive and can be removed from the final expression. In genetic programming-based symbolic regression approaches, this is typically done by limiting the number of terms in the expression. Enforcing sparsity in neural networks is an active field of research as modern deep learning architectures using billions of parameters start to become computationally prohibitive, especially for edge computing devices [129, 152, 270]. Gale et al. [50] evaluates several recent developments in neural network sparsity techniques.

A straightforward and popular way of enforcing sparsity is adding to the loss function a regularization term that is a function of the neural network weight matrices:

$$L_q = \sum_{i=0}^{L+1} L_q\left(\mathbf{W}_i\right) \tag{2.4}$$

where i indexes the layer. L_q acts element-wise on the matrix as follows:

$$L_q(\mathbf{W}) = \sum_{j,k} L_q\left(w_{j,k}\right) = \sum_{j,k} |w_{j,k}|^q$$

where j, k indexes the elements in the weight matrix $\mathbf{W}$.

Setting $q = 0$ in (2.4) results in L_0 regularization, which penalizes weights for being non-zero regardless of the magnitude of the weights (in other words, it counts the number of non-zero weights) and thus drives the solution towards sparsity. However, L_0 regularization is equivalent to a combinatorics problem that is NP-hard, and is not compatible with gradient descent methods commonly used for optimizing neural networks [157]. A much more popular and well-known sparsity technique is L_1 regularization with $q = 1$, which was used in the original EQL network [144]. Although it does not push solutions towards sparsity as strongly as L_0 regularization, L_1 regularization is a convex optimization problem that can be solved using a wide range of optimization techniques including gradient descent to drive the weights towards 0.

Here, we modify sparsity regularization and implement two different versions of the EQL that use a smoothed $L_{0.5}$ regularization and a relaxed L_0 regularization, respectively. We now describe both of these regularization techniques and how they are compatible with gradient descent training methods. Our experiments and results in Sections 2.3–2.4 use the smoothed $L_{0.5}$ regularization, although we note that we have found similar results using the smoothed L_0 regularization. The parametric versions of the EQL network used in Sections 2.5–2.7 use the smoothed L_0 regularization as it lends itself to a compact technique for group sparsity.

Smoothed $L_{0.5}$ Regularization

Any setting of regularization with $q \neq 0$ penalizes the magnitude of the weights, even the commonly used L_1 which also drives the solution towards sparsity. Thus, $L_{0.5}$ has been proposed to enforce sparsity more strongly without penalizing the magnitude of the weights as much as L_1 [253, 252]. $L_{0.5}$ regularization is still compatible with gradient descent (although it is no longer convex) and has been applied to neural

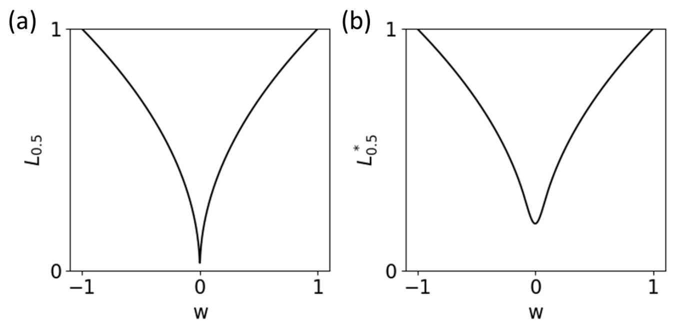


Figure 2-2: (a) $L_{0.5}$ and (b) $L^*_{0.5}$ regularization, as described in Equations (2.4) and (2.5), respectively. The threshold for the plot of Equation (2.5) is set to $a = 0.1$ for easy visualization, but we use a threshold of $a = 0.01$ in our experiments.

networks [42, 247]. Experimental studies suggest that $L_{0.5}$ regularization performs no worse than other L_q regularizers for $0 < q < 0.5$, implying that $L_{0.5}$ is optimal for enforcing sparsity [252]. Our experiments with $L_{0.3}$ and $L_{0.7}$ regularizers show no significant overall improvement compared to the $L_{0.5}$ regularizer, in agreement with this study. In addition, our experiments show that $L_{0.5}$ drives the solution towards sparsity more strongly than L_1, producing simpler expressions.

However, $L_{0.5}$ regularization has a singularity in the gradient as the weights go to 0, which can make training difficult for gradient descent-based methods. To avoid this, we use a smoothed version of $L_{0.5}$ proposed in [247], which we label as $L^*_{0.5}$. The $L^*_{0.5}$ regularizer uses a piecewise function to smooth out the function at small magnitudes:

$$L^*_{0.5}(w) = \begin{cases} |w|^{1/2} & |w| \geq a \\ \left(-\frac{w^4}{8a^3} + \frac{3w^2}{4a} + \frac{3a}{8}\right)^{1/2} & |w| < a \end{cases} \tag{2.5}$$

where $a \in \mathbb{R}^+$ is the transition point between the standard $L_{0.5}$ function and the smoothed function.

A plot of the $L_{0.5}$ and $L^*_{0.5}$ regularization are shown in Figure 2-2. The smoothed $L^*_{0.5}$ regularization avoids the extreme gradient values to improve training convergence. In our experiments, we set $a = 0.01$. When the EQL network is integrated with other deep learning architectures, the regularization is only applied to the weights of the EQL network.

Relaxed L_0 Regularization

While vanilla L_0 regularization is non-differentiable, recent works have explored training sparse neural networks with a relaxed version of L_0 regularization through stochastic gate variables, allowing this regularization to be compatible with backpropagation [129, 215]. We briefly review the details here, and refer the reader to Louizos et al. [129] for more details.

In relaxed L_0 regularization, the weights of the neural network are reparameterized as

$$\mathbf{W} = \tilde{\mathbf{W}} \odot \mathbf{z}$$

where $\mathbf{z}$ has the same dimensions as $\mathbf{W}$ and can be interpreted as a gate variable, and the multiplication is component-wise. Ideally each element of $\mathbf{z}$ is a binary "gate" such that $z \in \{0, 1\}$. However, this is not continuous (and thus non-differentiable), so we allow z to be a stochastic variable drawn from the hard concrete distribution:

$$\begin{gathered} u \sim \mathcal{U}(0,1) \\ s = \text{sigmoid}\left([\log u - \log(1-u) + \log \alpha] / \beta \right) \\ \bar{s} = s(\zeta - \gamma) + \gamma) \\ z = \min(1, \max(0, \bar{s})) \end{gathered}$$

where α is a trainable variable that describes the location of the hard concrete distribution, and β, ζ, γ are hyperparameters that describe the distribution. In the case of binary gates, the regularization penalty would simply be the sum of $\mathbf{z}$ (i.e., the number of non-zero elements in $\mathbf{W}$). However, in the case of the hard concrete distribution, we can calculate an analytical form for the expectation of the regularization penalty over the distribution parameters. The sparsity regularization loss is then

$$\mathcal{L}_R = \sum_j \text{sigmoid}\left(\log \alpha_j - \beta \log \frac{-\gamma}{\zeta} \right)$$

where j is indexing through all of the weight components. While ref. [129] applies group sparsity to the rows of the weight matrices with the goal of computational efficiency, we apply parameter sparsity (to individual elements) with the goal of simplifying the expression in symbolic regression.

The advantage of L_0 regularization is that it enforces sparsity without placing a penalty on the magnitude of the weights by placing a penalty on the expected number of non-zero weights. In the EQL network, this also allows us to train the system without needing a final stage where small weights are set to 0 and frozen. Additionally, for the case of learning parametric equations, it lends itself to a straightforward definition of group sparsity across time-steps as we will see in Section 2.6.1. In our experiments, we use the hyperparameters for the L_0 regularization suggested by ref. [129], although these can be optimized in future work.

While the reparameterization to achieve relaxed L_0 regularization requires us to double the number of trainable parameters in the neural network, the regularization is only applied to the EQL network, which is small compared to the rest of the architecture when integrating the EQL network with other deep learning modules.

We benchmark the EQL network using L_0 regularization with the aforementioned trial functions and list the results in Table A.1. The success rates appear to be as good or better than the network using $L_{0.5}$ regularization for most of the trial functions that we have picked. We have also integrated the EQL network using L_0 regularization into the MNIST arithmetic and kinematics architectures, and have found similar results as the EQL network using $L_{0.5}$ regularization.

2.2.3 Skip Connections

We can also add skip connections to the EQL network to introduce an inductive bias towards simpler equations while simultaneously enabling the discovery of more complex equations. The most well-known type of skip connections were introduced in ResNets, which take the output of a layer and add it to the layer ahead with the goal of allowing gradient information to efficiently propagate though many layers and enabling extremely deep architectures [70]. While these would be feasible to implement in the EQL network, they would serve to increase the complexity of the equation as information flows through the network. In contrast, we turn to the skip connections introduced by DenseNets which concatenates, rather than sums, the output of the previous layer with that of the next layer [79]. More specifically, we modify Equation 2.2 as:

$$\mathbf{h}^{(i)} = \left[f\left(\mathbf{g}^{(i)}\right); \mathbf{h}^{(i-1)} \right] \tag{2.6}$$

Skip connections introduce a slight inductive bias towards learning simpler functions, since functions can route "directly" to the output without needing to go through the identity primitive function of successive layers. Additionally, skip connections minimize instabilities during training that can arise as a result of gradients exploding as they pass through the primitive functions. Thus, skip connections allow us to train EQL networks with more layers, which in turn can learn more complex equations.

Note that in the experiments in Section 2.3, we do not use skip connections in the EQL network, although they can in principle be applied to facilitate training. The experiments on parametric equations in Section 2.7 do use skip connections.

2.3 Experiments

2.3.1 Symbolic Regression on Analytic Expressions

To validate the EQL network's ability to perform symbolic regression, we first test the EQL network on data generated by analytic expressions such as $\exp\left(-x^2\right)$ or $x_1^2 + \sin\left(2\pi x_2\right)$. The data is generated on the domain $x_i \in [-1, 1]$. Because of the network's sensitivity to random initialization of the weights, we run 20 trials for each function. We then count the number of times the network has converged to the correct answer ignoring small terms and slight variations in the coefficients from the true value. Additionally, equivalent answers (such as $\sin(4\pi + x)$ instead of $\sin(2\pi + x)$) are counted as correct. These results are shown in Appendix A.1.

The network only needs to be able to find the correct answer at least once over a reasonable number of trials, as one can construct a system that picks out the desired equation from the different trials using a combination of equation simplicity and extrapolation ability. We find that when we measure the extrapolation ability by measuring the equation error evaluated on the domain $x_i \in [-2, 2]$, this extrapolation error of the correct equation tends to be orders of magnitude lower than that of other equations that the network may find, making it simple to pick out the correct answer.

The network is still able to find the correct answer when 10% noise is added to the data. We also test an EQL network with 3 hidden layers which still finds the correct expression and is able to find even more complicated expressions such as $(x_1 + x_2x_3)^3$.

We also benchmark the EQL network using L_0 regularization with the aforementioned trial functions and list the results in Table A.1. The success rates appear to be as good or better than the network using $L_{0.5}$ regularization for most of the trial functions that we have picked.

2.3.2 MNIST Arithmetic

In the first experiment, we demonstrate the ability to combine symbolic regression and image recongition through an arithmetic task on MNIST digits. MNIST, a popular dataset for image recognition, can be notated as $\mathcal{D} = \{\chi, \psi\}$, where χ are 28×28 greyscale images of handwritten digits and $\psi \in \{0, 1, ..., 9\}$ is the integer-value label. Here, we wish to learn a simple arithmetic function, $y = \psi_1 + \psi_2$, with the corresponding images $\{\chi_1, \chi_2\}$ as inputs, and train the system end-to-end such that the system learns how to "add" two images together.

The deep learning architecture is shown in Figure 2-3. The input consists of two MNIST digits, $x = \{\chi_1, \chi_2\}$. During training, χ_i is randomly drawn from the MNIST training dataset. Each of $\{\chi_1, \chi_2\}$ are fed separately into an encoder to produce single-dimensional latent variables $\{z_1, z_2\}$ that are not constrained and can take on any real value, $z_i \in \mathbb{R}$. Alternatively, one can think of the architecture as having a separate encoder for each digit, where the two encoders share the same weights, as illustrated in Figure 2-3. The encoder consists of two convolutional layers with max pooling layers, followed by two fully-connected layers and a batch normalization layer at the output. More details on the encoder can be found in Appendix A.3. The latent

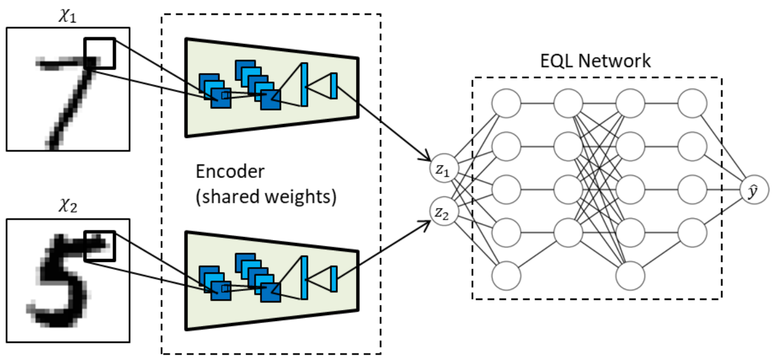


Figure 2-3: **Schematic of the MNIST addition architecture.** An encoder consisting of convolutional layers and fully-connected layers operate on each MNIST image and extract a single-dimensional latent variable. The two encoders share the same weights. The two latent variables are then fed into the EQL network. The entire system is fed end-to-end and without pre-training.

variables $\{z_1, z_2\}$ then feed into the EQL network. The EQL network has a single scalar output $\hat{y}$ which is trained on the true label $y = \psi_1 + \psi_2$.

The entire network is trained end-to-end using a mean-squared error (MSE) loss between the predicted label $\hat{y}$ and the true label y. In other words, the encoder is not trained separately from the EQL network. Note that the encoder closely resembles a simple convolutional neural network used for classifying MNIST digits except that it outputs a scalar value instead of logits that encode the digit. While there is no constraint on the properties of $z_{1,2}$, we expect it to map one-to-one to the true label $\psi_{1,2}$.

2.3.3 Dynamical System Analysis

In the next set of experiments, we apply the EQL network to analyzing physical time-varying systems. A potentially powerful application of deep learning in science exploration and discovery is discovering parameters in dynamical systems in an unsupervised setting and using these parameters to predict the propagation of similar systems. For example, [269] uses multilayer perceptrons to extract relevant properties from a system of bouncing balls (such as the mass of the balls or the spring constant of a force between the balls) and simultaneously predict the trajectory of a different

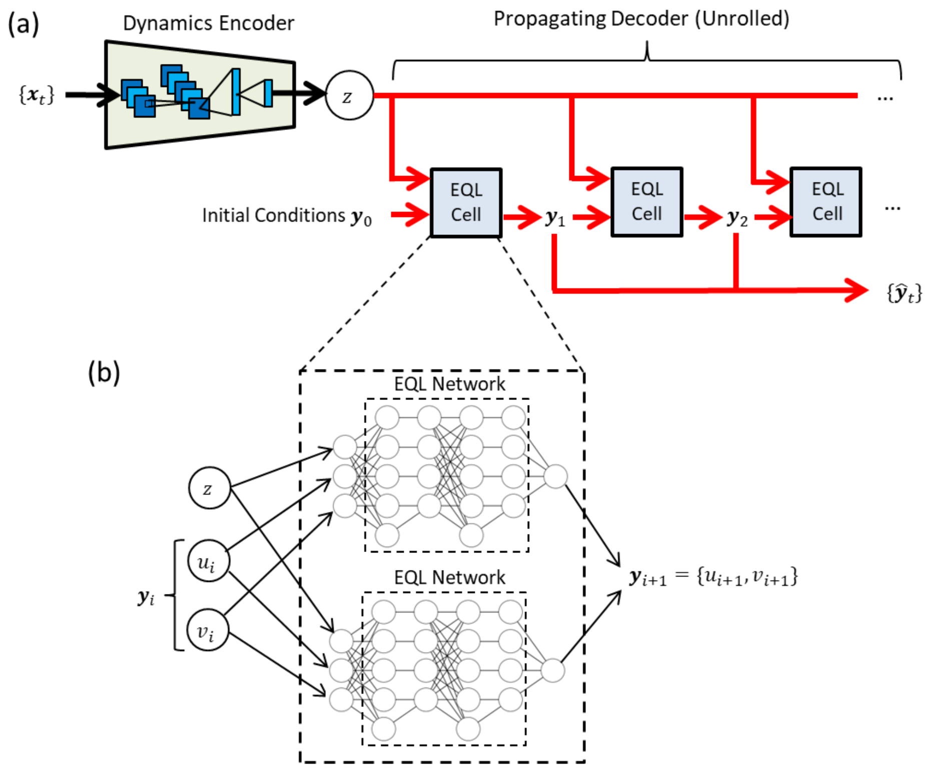


Figure 2-4: **Dynamical systems architecture.** (a) Architecture to learn the equations that propagate a dynamical system, including a dynamics encoder with convolutional layers and a propagating decoder with a recurrent architecture. (b) Each EQL cell in the propagating decoder consists of a separate EQL network for each dimension of $\mathbf{y}$ to be predicted. In our case, $\mathbf{y} = \{u, v\}$ where u is the position and v is velocity, so there are 2 EQL networks in each EQL cell.

set of objects. [134] accomplishes a similar goal but using a dynamics encoder (DE) with convolutional layers and a propagating decoder (PD) with deconvolutional layers to enable analysis and prediction of spatiotemporal systems such as those governed by PDEs. This DE-PD architecture is designed to analyze spatiotemporal systems that may have an uncontrolled dynamical parameter that varies among different instances of the dataset such as the diffusion constant in the diffusion equation. The parameters encoded in a latent variable are fed into the PD along with a set of initial conditions, which the PD propagates forward in time based on the extracted physical parameter and learned dynamics.

Here, we present a deep learning architecture shown in Figure 2-4 which is based on the DE-PD architecture. The DE takes in the full input series $\{\mathbf{x}_t\}_{t=0}^{T_x}$ over T_x time steps and outputs a single-dimensional latent variable z. Unlike the original DE-PD architecture presented in [134], the DE here is not a VAE. The DE here consists of several convolutional layers followed by fully-connected layers and a batch normalization layer. More details are given in Appendix A.3. The parameter z and a set of initial conditions $\mathbf{y}_0$ are fed into the PD which predicts the future time steps $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ based on the learned dynamics. The PD consists of an "EQL cell" in a recurrent structure, such that each step in the recurrent structure predicts a single time step forward. The EQL cell consists of separate EQL networks for each of feature, or dimension, in $\hat{\mathbf{y}}_t$.

The full architecture is trained end-to-end using a MSE loss between the predicted dynamics $\{\hat{\mathbf{y}}_t\}_{t=1}^{T_y}$ and the target series $\{\mathbf{y}_t\}_{t=1}^{T_y}$. Similar to the architecture in Section 2.3.2, the DE and PD are not trained separately, and there is no restriction or bias placed on the latent variable z. We explore two different physical systems (kinematics and simple harmonic oscillator) as described in the following sections.

Kinematics

Kinematics describes the motion of objects and is used in physics to study how objects move under an applied force. A schematic of a physical scenario described by 1-D kinematics is shown in Figure 2-5(a) in which an object on a frictionless surface has

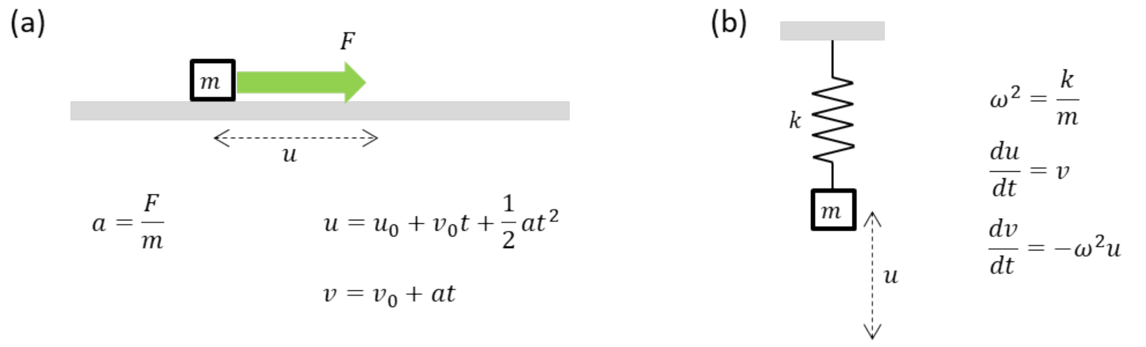


Figure 2-5: **Dynamical systems tasks.** (a) Kinematics describes the dynamics of an object where a force F is applied to a mass m. (b) Simple harmonic oscillator describes a mass m on a spring with spring constant k. In both cases, u is the displacement of the mass and v is the velocity.

a force applied to it where the direction of the force is parallel to the surface. The relevant parameter to describe the object's motion can be reduced to $a = \frac{F}{m}$ for a constant force F and object mass m. Given position u_i and velocity v_i at time step i, the object's state at time step $i+1$ are given by

$$\begin{aligned} u_{i+1} &= u_i + v_i \Delta t + \frac{1}{2} \Delta t^2 \\ v_{i+1} &= v_i + a \Delta t \end{aligned} \tag{2.7}$$

where Δt is the time step.

Acceleration a varies across different time series in the dataset. In our simulated dataset, we draw initial state and acceleration from uniform distributions:

$$u_0, v_0, a \sim \mathcal{U}(-1, 1)$$

We set $\Delta t = 1$. The initial parameters u_0, v_0 are fed into the PD, and z is expected to correlate with a.

Simple Harmonic Oscillator (SHO)

The second physical system we analyze is the simple harmonic oscillator (SHO), a ubiquitous model in physics that can describe a wide range of physical systems including springs, pendulums, quantum potentials, and electric circuits. In general, the dynamics of the SHO can be given by the coupled first-order ordinary differential equation (ODE)

$$\begin{aligned} \frac{du}{dt} &= v \\ \frac{dv}{dt} &= -\omega^2 u \end{aligned} \tag{2.8}$$

where u is the position, v is the velocity, and ω is the resonant frequency of the system. In the case of a spring as shown in Figure 2-5(b), $\omega = \sqrt{k/m}$ where k is the spring constant and m is the mass of the object on the end of the spring.

The SHO system can be numerically solved using a finite-difference approximation for the time derivatives. For example, the Euler method for integrating ODEs gives:

$$\begin{aligned} u_{i+1} &= u_i + v_i \Delta t \\ v_{i+1} &= v_i - \omega^2 u_i \Delta t \end{aligned} \tag{2.9}$$

In our experiments, we generate data with parameters drawn from uniform distributions:

$$\begin{aligned} u_0, v_0 &\sim \mathcal{U}(-1, 1) \\ \omega^2 &\sim \mathcal{U}(0.1, 1) \end{aligned}$$

The state variables u and v are measured at a time step of $\Delta t = 0.1$ to allow the system to find the finite-difference solution. Because of this small time step, we also need to propagate the solution for more time steps to find the right equation (otherwise the system learns the identity function). To avoid the recurrent architecture predictions exploding towards $\pm\infty$, we start the training by propagating only 1 time step, and add more time steps as the training continues. A similar strategy is used by [126] except that we are not restarting the training.

The initial parameters u_0, v_0 are fed into the PD, and z is expected to correlate with ω^2.

2.3.4 Particle System

Finally, we examine the system of interacting particles. We follow the setup provided by Cranmer et al. [31] in which multiple particles travel in an enclosed space with an interaction force between each pair of particles (i.e. spring force or Coulomb force). The system is parameterized by the properties of each particle (i.e. mass, charge) and the force between them (i.e. spring constant). Cranmer et al. [31] trained a graph neural network (GNN) to predict the instantaneous acceleration of each particle, with nodes corresponding to each particle. The GNN contained both edge models ϕ^e and node models ϕ^v. After training the GNN on the data, a traditional symbolic regression software was used on the edge models and node models to discover the force equation governing the system and the properties of each particle, respectively. Sparsity regularization is applied to the edge models to encourage interpretable equations.

In our experiments, we use the spring force dataset, where the potential is governed by the equation

$$U_{ij} = (r_{ij} - 1)^2 \tag{2.10}$$

where r is the distance between particles i and j. We also use the $1/r$ force dataset, where the potential is governed by the equation

$$U_{ij} = m_1 m_2 \log r_{ij}. \tag{2.11}$$

We use the same GNN architecture as the original work, but replace each of the edge and node models with EQL networks. While the original model required a 2-stage process to extract the governing equations, we expect our architecture to be able to discover the equations end-to-end in a single stage.

2.3.5 Training

The neural network is implemented in TensorFlow [3]. The network is trained using backpropagation with the RMSProp optimizer [222] and the following loss function:

$$\mathcal{L} = \frac{1}{N}\sum (y_i - \hat{y}_i)^2 + \lambda L_{0.5}^*$$

where N is the size of the training dataset and λ is a hyperparameter that balances the regularization versus the mean-squared error.

Similar to [144], we introduce a multi-phase training schedule. In an optional first phase, we train with a small value of λ, allowing for the parts of the network apart from the EQL to evolve freely and extract the latent parameters during training. In the second phase, λ is increased to a point where it forces the EQL network to become sparse. After this second phase, weights in the EQL network below a certain threshold α are set to 0 and frozen such that they stay 0, equivalent to fixing the L_0 norm. In the final phase of training, the system continues training without $L_{0.5}^*$ regularization (i.e. $\lambda = 0$) and with a reduced maximum learning rate in order to fine-tune the weights.

Specific details for each experiment are listed in Appendix A.3.

2.4 Results

2.4.1 MNIST Arithmetic

Figure 2-6(b) plots the latent variable z versus the true label ψ for each digit after the entire network has been trained. Note that while system is trained on digits drawn from the MNIST training dataset, we also evaluate the trained network's performance on digits drawn from the MNIST test dataset to confirm the encoder's generalizability. We see a strong linear correlation for both datasets, showing that the encoder has successfully learned a linear relation between z and ψ despite not having access to the digit label ψ. Also note that there is a scaling factor between z and ψ due to the

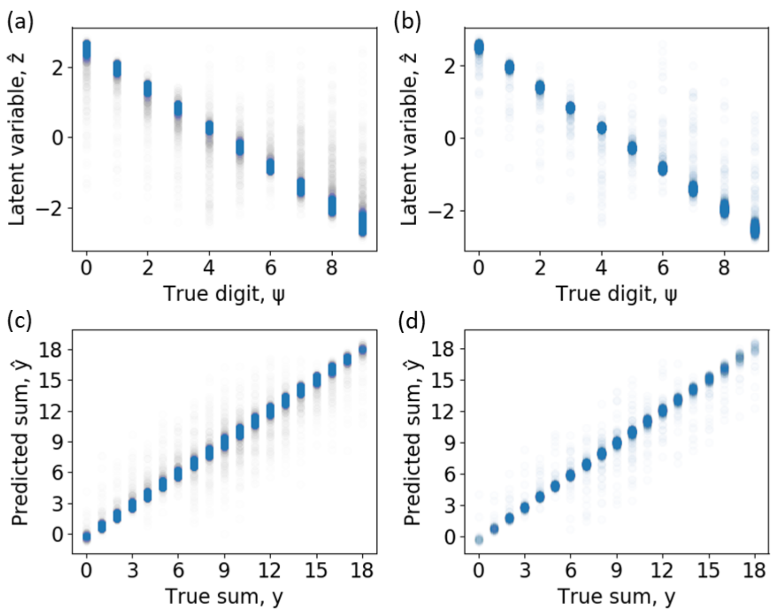


Figure 2-6: **MNIST arithmetic results.** The ability of the encoder to differentiate between digits as measured by the latent variable z versus the true digit ψ for digits χ drawn from the MNIST (a) training dataset and (b) test dataset. The correlation coefficients are -0.985 and -0.988, respectively. The ability of the entire architecture to fit the label y as measured by the predicted sum $\hat{y}$ versus the true sum y for digits χ drawn from the MNIST (c) training dataset and (d) MNIST test dataset.

Table 2.1: MNIST Arithmetic Expected and Extracted Equations

True	$y = \psi_1 + \psi_2$
Encoder	$\hat{y} = -1.788z_1 - 1.788z_2 + 9.04$
EQL	$\hat{y} = -1.809z_1 - 1.802z_2 + 9$

lack of constraint on z. A simple linear regression shows that the relation is

$$\psi = -1.788z + 4.519 \tag{2.12}$$

The extracted equation from the EQL network for this result is shown in Table 2.1. The "Encoder" equation is what we expect based on the encoder result in Equation 2.12. From these results, we conclude that the EQL network has successfully extracted the additive nature of the function. Plotted in Figure 2-6(c-d) are the predicted sums $\hat{y}$ versus the true sums y. The mean absolute errors of prediction for the model drawing digits from the MNIST training and test datasets are 0.307 and 0.315, respectively.

Table 2.2: MNIST Arithmetic Generalization Results

Source of w_i to form $x = \{w_1, w_2\}$	Network after the encoder	Accuracy [%] $y < 15$	Accuracy [%] $y \geq 15$
MNIST training dataset	EQL	92	87
	ReLU	93	0.8
MNIST test dataset	EQL	91	83
	ReLU	92	0.6

While the architecture is trained as a regression problem using an MSE loss, we can still report accuracies as if it is a classification task since the labels y are integers. To calculate accuracy, we first round the predicted sum $\hat{y}$ to the nearest integer and then compare it to the label y. The trained system achieves accuracies of 89.7% and 90.2% for digits drawn from the MNIST training and test datasets, respectively.

To demonstrate the generalization of this architecture to data outside of the training dataset, we train the system using a scheme where MNIST digit pairs χ_1, χ_2 are randomly sampled from the MNIST training dataset and used as a training data point if they follow the condition $\psi_1 + \psi_2 < 15$. Otherwise, the pair is discarded. In the test phase, MNIST digit pairs χ_1, χ_2 are randomly sampled from the MNIST training dataset and kept in the evaluation dataset if $\psi_1 + \psi_2 \geq 15$. Otherwise, the pair is discarded. For comparison, we also test the generalization of the encoder by following the above procedures but drawing MNIST digit pairs χ_1, χ_2 from the MNIST test dataset. Generalization results of the network are shown in Table 2.2. In this case, the EQL network has learned the equation $\hat{y} = -1.56z_1 - 1.56z_2 + 8.66$.

First, note the difference between the accuracy evaluated on pairs $y < 15$ and pairs $y \geq 15$. For the architecture with the EQL network, the accuracy drops by a few percentage points. However, for the architecture where the EQL network is replaced by the commonly used fully-connected network with ReLU activation functions (which we label as “ReLU"), the accuracy drops to below 1% showing that the results of the EQL is able to generalize reasonably well in a regime where the ReLU cannot generalize at all. Note that this is not a result of the encoder since the system sees all digits 0 through 9.

Second, the accuracy drops slightly when digits are drawn from the MNIST test dataset versus when the digits are drawn from the MNIST training dataset, as expected. We did not optimize the hyperparameters of the digit extraction network since the drop in accuracy is small, so the architecture could be optimized further if needed.

Finally, the accuracy drops slightly for pairs $y < 15$ when using the EQL versus the ReLU network. This is unsurprising since the larger size and symmetric activation functions of the ReLU network constrains the network less than the EQL and may make the optimization landscape smoother.

2.4.2 Kinematics

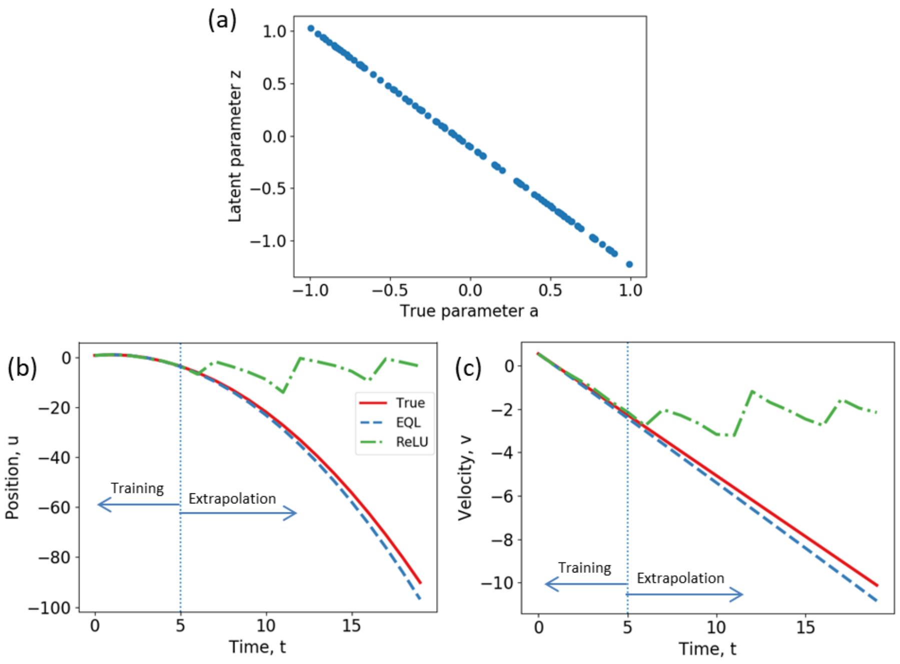


Figure 2-7: **Kinematics results.** (a) Latent parameter z of the dynamic encoder architecture after training plotted as a function of the true parameter a. We see a strong linear correlation. (b,c) Predicted propagation $\{\hat{\mathbf{y}}_i\} = \{\hat{u}_i, \hat{v}_i\}$ with the EQL cell and a conventional network using ReLU activations. "True" refers to the true propagation $\{\mathbf{y}_i\}$.

Figure 2-7(a) shows the extracted latent parameter z plotted as a function of the true parameter a. We see a linear correlation with correlation coefficient close to

-1, showing that the dynamics encoder has extracted the relevant parameter of the system. Again, there is a scaling relation between z and a, which we can extract through linear regression:

$$a = -0.884z - 0.091 \tag{2.13}$$

An example of the equations found by the EQL cell after training is shown in Table 2.3. The “DE" equations are what we expect based on the true equation and the extract relation in Equation 2.13. We can see that the EQL equations match closely with what we expect.

Table 2.3: Kinematics Expected and Extracted Equations

True	$u_{i+1} = u_i + v_i + \frac{1}{2}a$ $v_{i+1} = v_i + a$
DE	$\hat{u}_{i+1} = u_i + v_i - 0.442z - 0.045$ $\hat{v}_{i+1} = v_i - 0.884z - 0.091$
EQL	$\hat{u}_{i+1} = 1.002u_i + 1.002v_i - 0.475z$ $\hat{v}_{i+1} = 1.002v_i - 0.918z - 0.102$

The predicted propagation $\{\hat{\mathbf{y}}_i\}$ is plotted in Figure 2-7(b-c). “True" is the true solution that we want to fit, and “EQL" is the solution propagated by the EQL network. For comparison, we also train a neural network with a similar architecture to the one shown in Figure 2-4 but where the EQL cell is replaced by a standard fully-connected neural network with 2 hidden layers of 50 neurons each and ReLU activation functions (which we label as “ReLU"). While both networks match the true solution very closely in the training regime (left of the dotted line), the ReLU network quickly diverges from the true solution outside of the training regime. The EQL cell is able to match the solution reasonably well for several more time steps, showing how it can extrapolate beyond the training data.

2.4.3 SHO

The plot of the latent variable z as a function of the true parameter ω^2 after training on the SHO system is shown in Figure 2-8(a). Note that there is a strong linear correlation between z and ω^2 as opposed to between z and ω. This reflects the fact

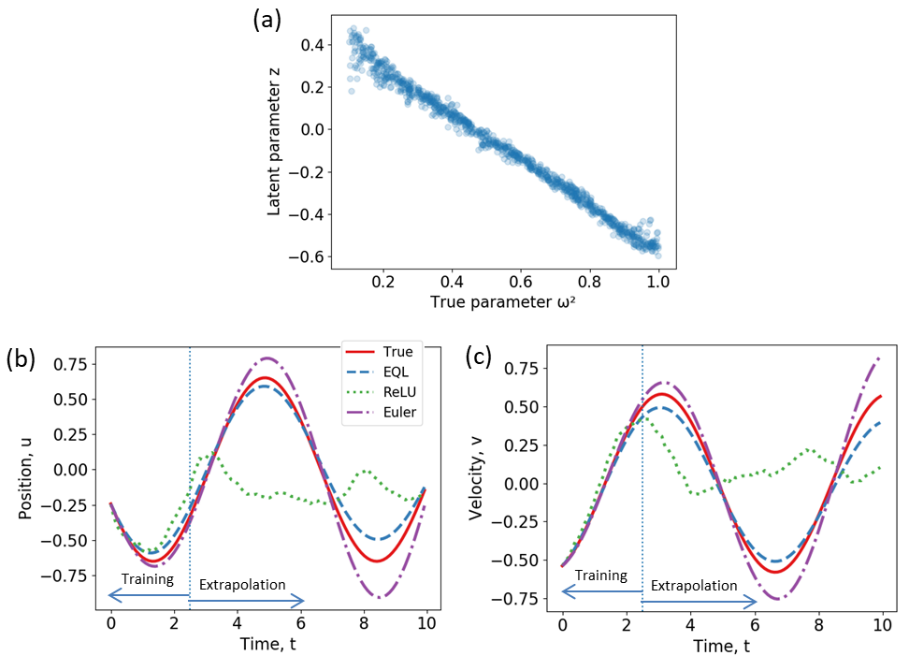


Figure 2-8: **Results of training on the SHO system.** (a) Latent parameter z of the dynamic encoder architecture after training plotted as a function of the true parameter ω^2. We see a good linear correlation. (b) Position u and (c) velocity v as a function of time for various models. "True" refers to the analytical solution. "EQL" refers to the propagation equation discovered by the EQL network. "ReLU" refers to propagation by a conventional neural network that uses ReLU activation functions. "Euler" refers to finite-difference solution using the Euler method.

Table 2.4: SHO Expected and Extracted Equations

True	$u_{i+1} = u_i + 0.1v_i$ $v_{i+1} = v_i - 0.1\omega^2 u_i$
DE	$\hat{u}_{i+1} = u_i + 0.1v_i$ $\hat{v}_{i+1} = v_i - 0.0464u_i + 0.0927u_i z$
DE, 2nd Order	$\hat{u}_{i+1} = u_i + 0.1v_i$ $\hat{v}_{i+1} = 0.998v_i - 0.0464u_i + 0.0927u_i z + 0.0046v_i z$
EQL	$\hat{u}_{i+1} = 0.994u_i + 0.0992v_i - 0.0031$ $\hat{v}_{i+1} = 0.995v_i - 0.0492d + 0.084u_i z + 0.0037v_i z + 0.0133z^2$

that using ω^2 requires fewer operations in the propagating equations than ω, the latter of which would require a squaring function. Additionally, the system was able to find that ω^2 is the simplest parameter to describe the system due to the sparsity regularization on the EQL cell. We see a strong linear correlation with a correlation coefficient of -0.995, showing that the dynamics encoder has successfully extracted the relevant parameter of the SHO system. A linear regression shows that the relation is:

$$\omega^2 = -0.927z + 0.464 \tag{2.14}$$

The equations extracted by the EQL cell are shown in Table 2.4. The "DE" equation is what we expect based on the dynamics encoder result in Equation 2.14. Immediately, we see that the expression for $\hat{u}_{i+1}$ and the first three terms of $\hat{v}_{i+1}$ match closely with the Euler method approximation using the latent variable relation extracted by the dynamics encoder.

An interesting point is that while we normally use the first-order approximation of the Euler method for integrating ODEs:

$$v_{i+1} = v_i + \Delta t \left.\frac{dv}{dt}\right|_{t=i} + \mathcal{O}(\Delta t^2)$$

it is possible to expand the approximation to find higher-order terms. If we expand the Euler method to its second-order approximation, we get:

$$\begin{aligned} v_{i+1} &= v_i + \Delta t \left.\frac{dv}{dt}\right|_{t=i} + \frac{1}{2}\Delta t^2 \left.\frac{d^2 v}{dt^2}\right|_{t=i} + \mathcal{O}(\Delta t^3) \\ &\approx v_i - \Delta t \omega^2 u_i - \frac{1}{2}\Delta t^2 \omega^2 v_i \end{aligned}$$

The expected equation based on the dynamics encoder result and assuming the 2nd order expansion is labeled as “DE, 2nd Order" in Table 2.4. It appears that the EQL network in this case has not only found the first-order Euler finite-difference method, it has also added on another small term that corresponds to second-order term in the Taylor expansion of v_{i+1}. The last term found by the EQL network, $0.0133z^2$ is likely from either cross-terms inside the network or a lack of convergence to exactly 0 and would likely disappear with another thresholding process.

The solution propagated through time is shown in Figure 2-8(b-c). As before, “ReLU" is the solution propagated by an architecture where the EQL network is replaced by a conventional neural network with 4 hidden layers of 50 units each and ReLU activation functions. For an additional comparison, we have also calculated the finite-difference solution using Euler’s method to integrate the true ODEs which is labeled as “Euler".

Within the training regime, all of the methods fit the true solution reasonably well. However, the conventional neural network with ReLU activation functions completely fails to extrapolate beyond the training regime. This is not particularly surprising as previous works report that neural networks generally struggle in learning periodic functions, and that periodic inductive biases are required to improve learning [271]. The Euler method and the EQL network are both able to extrapolate reasonably well beyond the training regime, although they both start to diverge from the true solution due to the large time step and the accumulated errors from numerical integration. A more accurate method such as the Runge-Kutta method almost exactly fits the analytical solution, which is not surprising due to its small error bound. However, it is more complex than the Euler method and would likely require a larger EQL network to find an expression similar to the Runge-Kutta method. Interestingly, the

Table 2.5: Particle System Results. Equations listed for the GNN and Graph EQL are only the first component of the edge model. c_i and b_i are arbitrary constants.

	Spring force	$1/r$ force
Expected (aligned)	$c_1\Delta x\left(1-\frac{1}{r}\right)+b_1$	$c_1 m_2\Delta x/r^2+b_1$
Expected (general)	$c_1\Delta x\left(1-\frac{1}{r}\right)+c_2\Delta y\left(1-\frac{1}{r}\right)+b_1$	$c_1 m_2\Delta x/r^2+c_2 m_2\Delta y/r^2+b_1$
GNN [31]	$0.60\Delta x\left(1-\frac{1}{r}\right)+1.36\Delta y\left(1-\frac{1}{r}\right)$	$\left(4.59m_2\Delta y-15.5m_2\Delta x\right)/r^2$
Graph EQL	$0.05\Delta x\left(1-\frac{1}{r}\right)$	$-0.02m_2\Delta x/r^2$

EQL network solution has a smaller error than the Euler solution, demonstrating that the EQL network was able to learn higher-order corrections to the first-order Euler method. This could possibly lead to discovery of more efficient integration schemes for differential equations that are difficult to solve through finite-difference methods.

2.4.4 Particle Systems

The results are shown in Table 2.5 for 2D versions of the datasets and with 4 particles For the extracted equations we only list the first component of the edge model (as the second component generally follows a similar form for the other spatial dimension, and the remaining components are zero). Because the original GNN is trained without any inductive bias, the learned force equations do not necessarily need to align with the Cartesian axes of the original data. Indeed, Cranmer et al. [31] report that the extracted equations are generally rotated versions of the expected equations, and use a rotation to back out the original equations. However, the Graph EQL network incorporates inductive biases inside the architecture in the form of the primitive activation functions combined with sparsity regularization in the EQL network. This discovers a simpler equation which corresponds to the unrotated version of the spring force, thus pointing to the advantage of the end-to-end nature of the Graph EQL network. We see similar results when scaling up the complexity of the $1/r$ force dataset, i.e. when we have 8 particles or when the dimensionality is 3, and for more complex datasets, i.e. a $1/r^2$ force.

Table 2.6: Learned equations by the EQL network on simple parametric equations.

True	EQL
$f_1 = tx^2 + 3\,\mathrm{sgn}(t)x$	$0.02x^2 - 0.14t + 1.86xt + 1.01x^2t + 0.03t^2 - 0.01x^2t^2$ $- 0.02t^3 - 0.01xt^3 + (0.09 - 1.80x + 0.01x^2)\sin(1.91t)$ $+ 0.77x\sin(3.71t) + 0.32\sin(3.22xt) - 0.10$
$f_2 = \sin(j(t)x)$	$0.720\sin(2.05x)$

2.5 Introduction to Parametric Equations

Next, we look at another type of complexity, which are datasets described by parametric equations in which the underlying equation structure may stay the same but coefficients may vary along one or more dimensions. PDEs are ubiquitous in describing the dynamics of many systems, but even the most simple settings can require varying coefficients. For example, solving for electromagnetic modes or electron wavefunction in a material requires solving Maxwell's equation with spatially-varying permittivity or the Schrödinger equation with varying potential, respectively. Parameters may be influenced or even controlled by external factors that are not captured in the data [268]. The nonlinear Schrödinger equation with varying coefficients has found applications in describing Bose-Einstein condensates [257]. Additionally, the varying coefficients may change in complex ways that are not easily expressible symbolically, which would result in standard symbolic regression tools failing to discover interpretable equations. Various approaches have been proposed to discover specifically parametric *PDEs*, including group sparsity combined with SINDy [192], genetic algorithms combined with averaging over local windows [250], and linear regression with kernel smoothing over adjacent coefficients [137].

In Sections 2.5-2.8 we generalize symbolic regression to parametric equations. The ability to learn parametric equations may greatly expand the scope of symbolic regression, especially in cases where the coefficients may vary over the dataset in arbitrarily complex ways that are difficult to express symbolically. Such behavior would greatly impede the performance of traditional symbolic regression approaches that attempt to find the simplest equation describing the dataset. To illustrate this, we train the EQL network on two simple parametric equations listed in Table 2.6.

The first function is a parabolic curve described as:

$$f_1(x,t) = tx^2 + 3\,\mathrm{sgn}(t)x$$

where sgn is the *sign* function (also known as the *signum* function):

$$\mathrm{sgn}(t) = \begin{cases} -1 & \text{if } t < 0 \\ 0 & \text{if } t = 0 \\ 1 & \text{if } t > 0 \end{cases} \cdot$$

The function notably contains a discontinuity at $t = 0$ and thus cannot be described in terms of smooth functions. The results of fitting the EQL network are shown in Figure 2-9(a) and Table 2.6. The EQL network learns an overly complicated equation with over a dozen terms that likely signify its attempt to fit the discontinuity. While it seems to fit reasonably well inside the training regime (in the range $-3 < x < 3$), it fails to extrapolate well since it has not learned the correct equation. (In principle, the function f_1 is simple enough such that an EQL network with sigmoidal activation functions could approximate it with reasonable accuracy, but we choose this example to illustrate some difficulties of symbolic regression.)

In the second example, we look at a sinusoidal curve where the frequency varies non-smoothly as a function of time:

$$f_2(x,t) = \sin(j(t)x)$$

where we have defined a "jagged" function:

$$j(t) = \begin{cases} 0.5t + 2.5 & \text{if } t < 0 \\ -0.5t + 2.5 & \text{if } 0 \le t < 1.5 \\ t + 0.25 & \text{otherwise} \end{cases} \cdot$$

The jagged function $j(t)$ is illustrated in Figure 2-9(d). As seen in Table 2.6, the

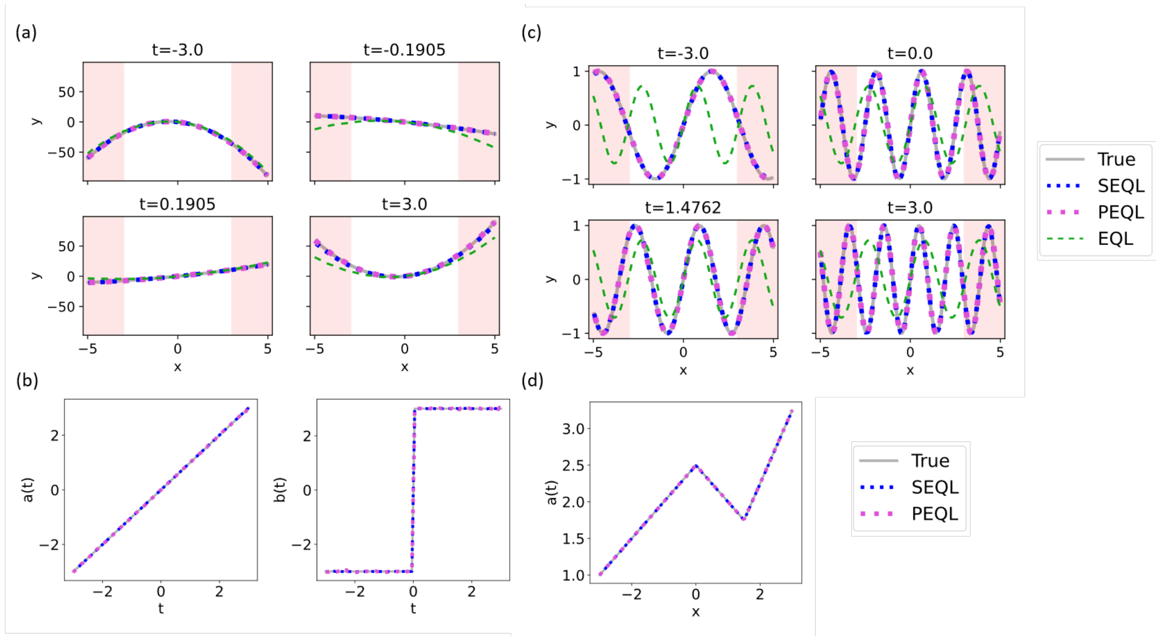


Figure 2-9: **Learning parametric equations.** (a, b) Learning the function f_1 which contains a discontinuity at $t = 0$. (c, d) Learning the function f_2 which corresponds to a sinusoid with a frequency that varies non-smoothly as a function of t. (a, c) Predictions after training the EQL, SEQL, and HEQL networks in the range $-3 < x < 3$ for various values of t. Values outside of this range (highlighted in red) are extrapolated. (b, d) Learned functions for the varying coefficents.

EQL network is unable to learn the parametric form of the sinusoidal frequency, and thus fits poorly.

In this section we extend the EQL network and enable neural network-based symbolic regression to parametric equations that may have varying coefficients. We propose two novel architectures, the stacked EQL network (SEQL) and the hyper EQL network (HEQL), which can each discover parametric equations with various advantages. We demonstrate our method on various analytic equations, PDEs, and a high-dimensional dataset consisting of images of particles.

In Section 2.6 we first propose several modifications to the EQL network that improve its training behavior. In Sections 2.6.1 and 2.6.2 we propose 2 variants of the EQL architecture that can discover parametric equations. Note that in our discussion and notation in this section, we assume that the coefficients are parameterized with respect to *time* as this provides a convenient intuition applicable to many systems. However, the parameterization could also be with respect to other quantities (e.g. space).

2.6 Parametric EQL Network Variants

In this work, we propose two variants of the EQL network to learn parametric equations: the Stacked EQL (SEQL) and the Hyper EQL (HEQL).

2.6.1 Stacked Architecture (SEQL)

The first extension we propose to analyze parametric equations is to train a separate EQL network for each time step, an architecture that we call the stacked EQL (SEQL) network. Suppose we have a dataset that is indexed by the time step j:

$$\mathcal{D} = \left\{ \left\{ x^{(i,j)}, y^{(i,j)} \right\}_{i=1}^{N^{(j)}} \right\}_{j=1}^{N_t} \tag{2.15}$$

where N_t is the number of time steps and $N^{(j)}$ is the number of data points in the jth time step (note that $N^{(j)}$ does not need to be constant across time steps). For layer

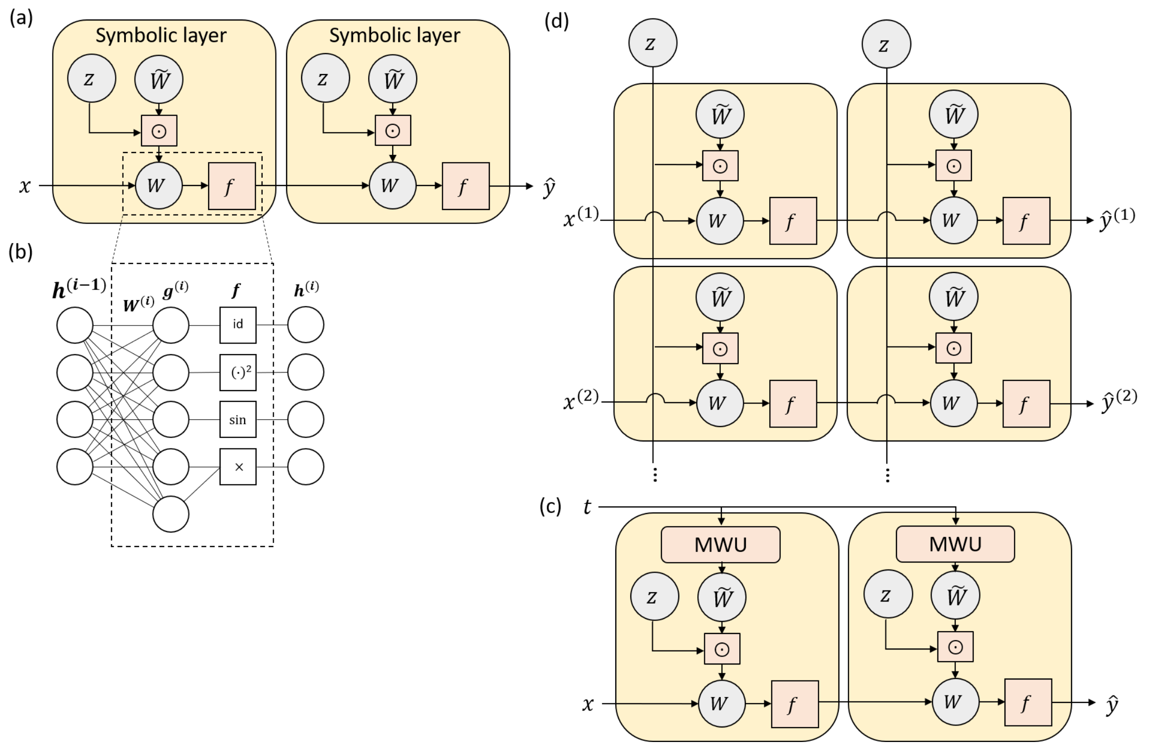


Figure 2-10: **Equation Learner (EQL) Architectures and Variants for Parametric Equations.** (a) Architecture of the base EQL network with relaxed L_0 regularization. The weights $\mathbf{W}$ are re-parameterized as an element-wise product of the gate variables $\mathbf{z}$ and the weight values $\tilde{\mathbf{W}}$. (b) The core of the symbolic layer, where the activation functions consist of the primitive functions for symbolic regression, where each element may contain a different primitive function and primitive functions may take multiple inputs. (c) Architecture of the stacked EQL (SEQL) network. Note that the indexing $x^{(j)}$ is for the time step. Each horizontal row represents an EQL network for each time step. The gate $\mathbf{z}$ is shared across time steps. (d) Architecture of the hyper EQL (HEQL) network. Note that in all schematics, the final (linear) layer is omitted for visual simplicity.

i of the SEQL network, we can construct N_t separate weight matrices, $\left\{\tilde{\mathbf{W}}^{(i,j)}\right\}_{j=1}^{N_t}$, such that Equations 2.2 are modified as:

$$\mathbf{g}^{(i,j)} = \mathbf{W}^{(i,j)}\mathbf{h}^{(i-1,j)} \tag{2.16}$$

$$\mathbf{h}^{(i,j)} = f\left(\mathbf{g}^{(i,j)}\right). \tag{2.17}$$

In other words, we parameterize a separate EQL network for each time step, as shown in Figure 2-10(c).

If we naïvely train N_t separate EQL networks, then it is possible that each network may learn a different equation in each time step. Additionally, each network would only see approximately $\frac{1}{N_t}$ of the total data, thus reducing data efficiency. To counteract this, we enforce that the different networks learn the same equation by implementing group sparsity through weight sharing of the gate variable $\mathbf{z}$. For the i^{th} layer of the j^{th} time step, we further modify Eq. 2.17 as:

$$\mathbf{h}^{(i,j)} = f\left(\left(\tilde{\mathbf{W}}^{(i,j)} \odot \mathbf{z}^{(i)}\right)\mathbf{h}^{(i-1,j)}\right) \tag{2.18}$$

Note that $\mathbf{z}$ is not parameterized with respect to the time step j. For an architecture with L hidden layers, there are $(L+1)\cdot N_t$ weight matrices and $L+1$ gate matrices.

Another modification we make to the architecture is weight regularization across time steps to introduce an inductive bias towards smoothness in the coefficients. We use L_2 regularization loss between adjacent time steps. Looking at just a single element $w_{k,l}$ of $\mathbf{W}$ in a single layer for notational simplicity, the inter-layer L_2 loss is simply

$$L_{S,k,l} = \sum_{j=1}^{N_t-1}\left(w_{k,l}^{(j+1)} - w_{k,l}^{(j)}\right)^2 \tag{2.19}$$

and the total inter-layer regularization loss is

$$\mathcal{L}_S = \sum_{i,k,l} L_{S,k,l}^{(i)} \tag{2.20}$$

where i indexes the layer. This regularization pushes coefficients in adjacent time-steps closer together and can more effectively counteract noisy datasets.

2.6.2 Hyper EQL (HEQL) Architecture

We also propose a second variant of the EQL network, the Hyper EQL (HEQL) network, in which the weights $\tilde{\mathbf{W}}$ are re-parameterized as a function of the varying coefficient, i.e., $\tilde{\mathbf{W}}(t)$. While a number of models can be used to parameterize the weights, we use a fully-connected neural network as it is a flexible model that can fit arbitrary functions and can be trained with backpropagation, allowing the entire system to be trained end-to-end. We call this fully-connected neural network the meta-weight unit (MWU). The architecture is shown in Figure 2-10(d).

This idea is similar to that of hypernetworks, in which a neural network is used to generate the weights of another neural network [64]. The general idea of using a network to parameterize or interact with the weights of another network has been most notably leveraged for meta-learning [6, 155, 186, 76], and has also been applied to a variety of other architectures, including the Neural ODE [24] and HyperPINN [34].

The HEQL has a separate MWU in each layer (including the linear output layer) which takes the parametric variable t as an input and outputs the weight matrix $\tilde{\mathbf{W}}^{(i)}(t)$ for that layer. The gate variables $\mathbf{z}$ are not modified and are thus not a function of t. As a result, all of the "time steps" share the same sparsity regularization, thus avoiding the need for any further modifications to implement group sparsity.

The advantage of this architecture compared to the SEQL is that the HEQL does not replicate the EQL network for each time step, thus saving on computational memory especially for large N_t. The architecture can also make predictions on a continuous domain of t and does not need require the data to align along a fixed grid in time, unlike the prior work on discovering parametric PDEs [192, 250, 137]. More specifically, rather than viewing the dataset as Equation 2.15, we have greater flexibility and can view the dataset as

$$\mathcal{D} = \left\{ x^{(i)}, y^{(i)}, t^{(i)} \right\}_{i=1}^{N} . \tag{2.21}$$

Table 2.7: Additional analytic parametric equations for benchmarking

Label	Equation
f_3	tx
f_4	$tx^2 + 3\sin(t)x$
f_5	$\sin\left(\frac{5+t}{2}x\right)$

Although we do not explicitly regularize the functional space of the parametric coefficients, neural networks tend to generalize well despite typically being overparameterized, which is a topic of significant interest [156, 121, 123, 83]. In practice, this means that the predictions of neural networks for regression tasks tend to be smooth, and so the function of the parametric coefficient will also tend to be smooth.

2.7 Results

We now look at several different problem settings with parametric quantities that can be analyzed by our system. For simplicity, we highlight some of the results here, and the remainder can be found in the appendix. Section 2.7.1 demonstrates some simple benchmarks to highlight the aspects of learning parametric equations. Section 2.7.2 shows results on PDE datasets taken from other works. Finally, section 2.7.3 presents results on 1D images of a spring system to demonstrate the ability to perform symbolic regression on higher-dimensional systems.

2.7.1 Analytic Expressions

To verify the ability of the SEQL and HEQL networks to discover parametric equations, we benchmark the networks on the analytical expressions discussed in Section 2.5 and listed in Table 2.6. We also benchmark additional expressions listed in Table 2.7. While we train the networks on data drawn from the domain $x \in [-3, 3]$, we evaluate the networks on a wider domain $x \in [-5, 5]$ to test extrapolation performance.

Figure 2-9 shows the results for learning f_1 and f_2 using the SEQL and HEQL networks. The true function and the predicted function are plotted for various values of t. In all cases, the SEQL/HEQL predictions are visually indiscernible from the true function in both the training regime and the test regime, demonstrating that the architectures are able to extrapolate. As in the case of the original EQL network, the learned equations can be extracted from the trained network by simply processing the learned weights with software for symbolic mathematics. In particular, we use SymPy, a Python package for symbolic mathematics, to simplify the resulting expression [149]. For clarity, we also omit negligible terms (i.e., those with coefficient magnitudes < 0.005) in the final expression.

Table 2.8: Results for training on parametric analytic expressions. Learned equations are extracted for various values of t.

	Training (Test) MSE				Learned equations	
Benchmark	SEQL	HEQL	t	True	SEQL	HEQL
			-2.619	$-2.619x^2 - 3x$	$-2.62x^2 - 3.00x - 0.05$	$-2.63x^2 - 3.02x - 0.03$
f_1	$\mathbf{2.37 \times 10^{-6}}$	3.79×10^{-6}	-1.095	$-1.095x^2 - 3x$	$-1.10x^2 - 3.00x - 0.02$	$-1.12x^2 - 3.02x - 0.01$
	($\mathbf{6.97 \times 10^{-6}}$)	(2.04×10^{-5})	0.381	$0.381x^2 + 3x$	$0.38x^2 + 3.00x + 0.01$	$0.39x^2 + 3.01x + 0.01$
			1.905	$1.905x^2 + 3x$	$1.91x^2 + 3.00x + 0.03$	$1.91x^2 + 3.02x + 0.03$
			-2.619	$\sin(1.191x)$	$0.999\sin(1.191x)$	$1.000\sin(1.190x)$
f_2	4.07×10^{-7}	$\mathbf{1.27 \times 10^{-8}}$	-1.095	$\sin(1.953x)$	$0.999\sin(1.952x)$	$1.000\sin(1.952x)$
	(8.98×10^{-7})	($\mathbf{1.22 \times 10^{-7}}$)	0.381	$\sin(2.310x)$	$0.997\sin(2.310x)$	$1.000\sin(2.309x)$
			1.905	$\sin(2.155x)$	$0.999\sin(2.155x)$	$1.000\sin(2.155x)$
			-2.619	$-2.62x$	$-2.62x$	$-2.62x - 0.02$
f_3	$\mathbf{2.81 \times 10^{-15}}$	2.53×10^{-6}	-1.095	$-1.10x$	$-1.10x$	$-1.10x$
	($\mathbf{5.28 \times 10^{-15}}$)	(4.43×10^{-6})	0.381	$0.38x$	$0.38x$	$0.38x$
			1.905	$1.90x$	$1.91x$	$1.91x$
			-2.619	$-2.62x^2 - 1.50x$	$-2.62x^2 - 1.51x - 0.08$	$-2.62x^2 - 1.50x + 0.01$
f_4	$\mathbf{3.09 \times 10^{-6}}$	5.97×10^{-6}	-1.095	$-1.10x^2 - 2.67x$	$-1.09x^2 - 2.68x - 0.08$	$-1.10x^2 - 2.66x + 0.02$
	($\mathbf{9.47 \times 10^{-6}}$)	(3.17×10^{-5})	0.381	$0.38x^2 + 1.12x$	$0.38x^2 + 1.12x + 0.01$	$0.38x^2 + 1.12x$
			1.905	$1.90x^2 + 2.83x$	$1.90x^2 + 2.85x + 0.06$	$1.90x^2 + 2.83x$
			-2.619	$\sin(1.19x)$	$\sin(1.19x)$	$\sin(1.19x)$
f_5	6.26×10^{-8}	$\mathbf{1.40 \times 10^{-8}}$	-1.095	$\sin(1.95x)$	$\sin(1.95x)$	$\sin(1.95x)$
	(1.21×10^{-7})	($\mathbf{3.63 \times 10^{-8}}$)	0.381	$\sin(2.69x)$	$\sin(2.69x)$	$\sin(2.69x)$
			1.905	$\sin(3.45x)$	$\sin(3.45x)$	$\sin(3.45x)$

Table 2.9: MSE after training on PDE datasets.

	u_t MSE			Coefficient MSE		
Benchmark	SINDy	SEQL	HEQL	SINDy	SEQL	HEQL
Advection-diffusion, with cross terms	$\mathbf{2.34 \times 10^{-7}}$	2.87×10^{-5}	1.99×10^{-5}	3.99×10^{-2}	2.99×10^{-3}	$\mathbf{1.44 \times 10^{-3}}$
Advection-diffusion, no cross terms	$\mathbf{2.34 \times 10^{-7}}$	2.12×10^{-5}	1.57×10^{-5}	1.73×10^{-2}	2.30×10^{-3}	$\mathbf{1.37 \times 10^{-3}}$
Burgers', with cross terms	$\mathbf{5.11 \times 10^{-8}}$	5.66×10^{-6}	5.54×10^{-6}	$\mathbf{3.95 \times 10^{-7}}$	8.81×10^{-6}	9.83×10^{-6}
Burgers', no cross terms	2.30×10^{-4}	$\mathbf{2.55 \times 10^{-7}}$	8.02×10^{-6}	1.73×10^{-3}	$\mathbf{8.41 \times 10^{-6}}$	4.44×10^{-5}

The extracted equations for learning the parametric parabolic function f_1 for various time steps are shown in Table 2.8. Upon inspection of the extracted equations over multiple time steps, we see that the architectures has successfully discovered the function $\hat{f}_1 = a(t)x^2 + b(t)x + \epsilon(t)$ where $a(t)$ and $b(t)$ are the varying coefficients and ϵ is a small number that can either be eliminated with further training or ignored upon inspection. The predicted parametric coefficients $a(t)$ and $b(t)$ match the true coefficients extremely closely, as seen in Figure 2-9(b). Note that the SEQL/HEQL networks are able to learn the discontinuous sgn function without any apparent smoothing at $t = 0$. Discontinuous coefficients would be difficult to learn using other methods for parametric equations that rely on local averaging [250] or smoothing [137].

For learning the sinusoidal function f_2, both the SEQL and HEQL networks have learned the equation $\hat{f}_2 = \sin(a(t)x)$ as seen in Table 2.8 where $a(x)$ is plotted in Figure 2-9(d). Again, the predictions match the true function extremely well across time steps and outside of the training regime. Although sinusoidal functions are typically difficult to learn through linear regression techniques, the SEQL and HEQL networks are able to learn this function across multiple spatial frequencies.

Note that because the varying coefficient is inside the sgn and sin functions for f_1 and f_2, respectively, other methods for learning parametric equations such as those proposed in Refs. [137] or [19] that rely on linear regression techniques would not be able to discover these types of equations. In contrast, the multi-layer architecture of the SEQL and HEQL networks allow for the varying coefficient to be inside nested functions, enabling discovery of much more complex parametric equations. Interestingly, there is no clear trend on whether the SEQL or HEQL tends to perform better.

2.7.2 PDEs

Next, we investigate learning partial differential equations (PDEs) with varying coefficients from data. Prior works in discovering parametric equations have focused on the setting of PDEs [192, 250, 137], as PDEs are ubiquitous in describing dynamics

Table 2.10: Learned equations on select x values for the advection-diffusion equation.

x	True	SEQL	HEQL
-4.375	$-0.89u - 0.79u_x + 0.10u_{xx}$	$-0.86u - 0.77u_x + 0.11u_{xx}$	$-0.86u - 0.76u_x + 0.11u_{xx}$
-1.875	$0.89u - 2.21u_x + 0.10u_{xx}$	$0.83u - 2.14u_x + 0.07u_{xx}$	$0.86u - 2.16u_x + 0.09u_{xx}$
0.625	$-0.89u - 0.79u_x + 0.10u_{xx}$	$-0.86u - 0.77u_x + 0.11u_{xx}$	$-0.86u - 0.77u_x + 0.11u_{xx}$
3.125	$0.89u - 2.21u_x + 0.10u_{xx}$	$0.83u - 2.14u_x + 0.07u_{xx}$	$0.88u - 2.17u_x + 0.10u_{xx}$

in a variety of fields. For ease of comparison, we benchmark our architectures on two of the datasets provided by Rudy et al. [192], the advection-diffusion equation and Burgers' equation with varying coefficients. In this setting, the partial differential terms (e.g. u_x, u_{xx}) are pre-computed from the dataset and concatenated with the input u.

We note that SINDy is not able to automatically calculate cross terms (e.g. uu_x) and so the cross terms were also pre-computed and fed into SINDy in the original work [192]. We label this approach as "with cross terms" in Table 2.9. In contrast, the SEQL and HEQL architectures are able to automatically discover cross terms as necessary, and so we also carry out experiments that omit the cross terms in the input, labelled "no cross terms."

Advection-Diffusion Equation

The advection-diffusion equation describes numerous physical transport systems and has been applied to describe the movement of pollutants, reservoir flow, heat, and semiconductors. We use an adaptation of the equation that includes a spatially-dependent velocity field, as in [192]:

$$u_t = f'(x)u + f(x)u_x + \epsilon u_{xx}. \tag{2.22}$$

where $f(x) = -1.5 + \cos\left(\frac{2\pi x}{5}\right)$ and $\epsilon = 0.1$. Note that the parametric quantities vary with respect to space rather than time.

Table 2.10 shows the equations that the SEQL and HEQL have learned after training for few instances of x. We see that both networks have learned an equation of the form $\hat{u}_t = \hat{f}'(x)u + \hat{f}(x)u_x + \hat{\epsilon}(x)u_{xx}$, and have thus successfully discovered the

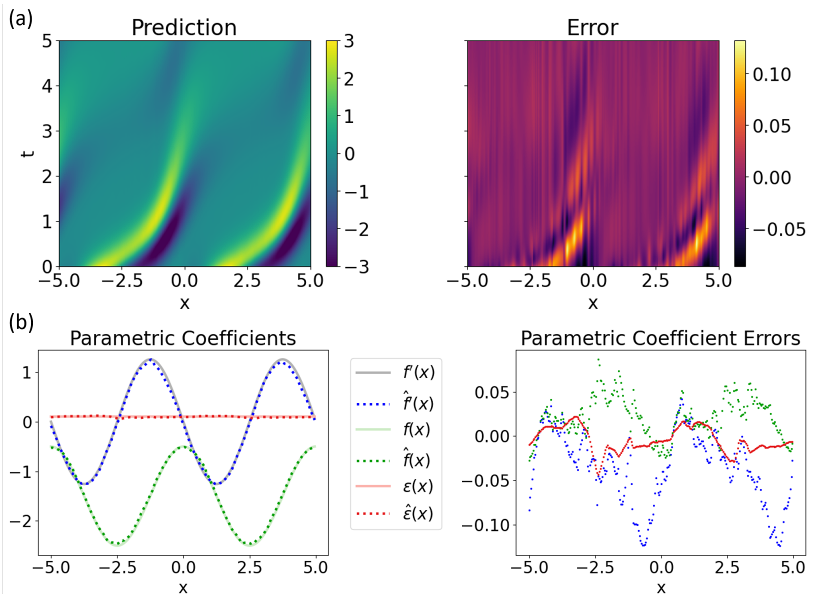


Figure 2-11: Results for learning the advection-diffusion equation using the HEQL network. (a) Prediction values and errors of u_t. (b) Predicted coefficient functions and prediction errors.

equation structure. The predicted $\hat{u}_t$ along with the learned parametric coefficients $(\hat{f}'(x), \hat{f}(x), \hat{\epsilon}(x))$ are shown in Figure 2-11 for the HEQL network. The predicted values match the actual values very closely. Again, note that the predicted coefficients by the fully-connected neural network are smooth as a function of x despite the lack of explicit regularization.

Burgers' Equation

Burgers' equation is an important differential equation originally proposed to model turbulent flow but has been applied to other processes such as traffic flow and boundary layer behavior. Here we analyze Burgers' equation with an oscillating coefficient for the non-linear term, as in [192]:

$$u_t = f(t)uu_x + \epsilon u_{xx}. \tag{2.23}$$

where $f(t) = -\left(1 + \frac{\sin(t)}{4}\right)$ and $\epsilon = 0.1$.

As before, both the SEQL and HEQL networks are able to accurately discover

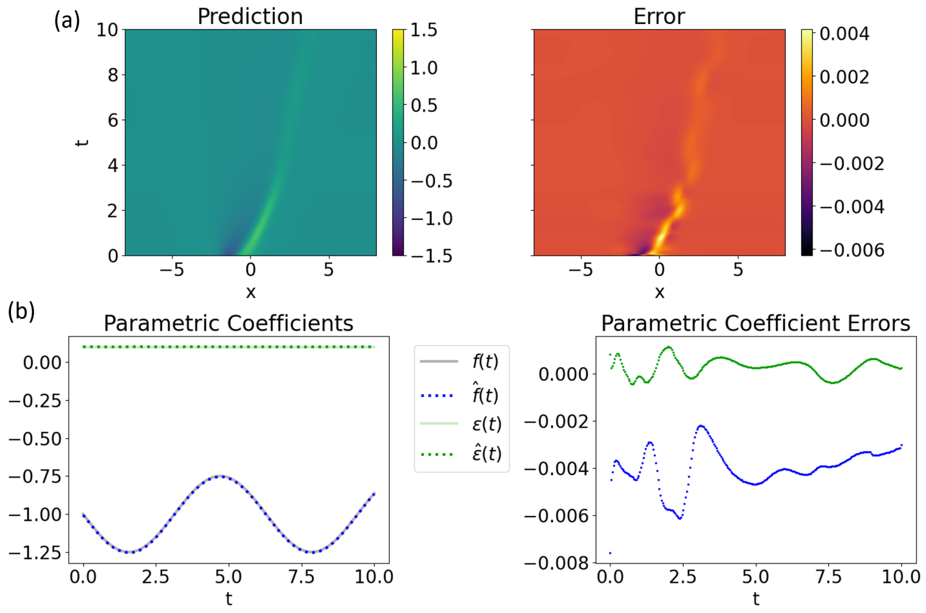


Figure 2-12: Results for learning Burgers' equation using the SEQL network. (a) Predicted vs. actual values of u_t. (b) Predicted coefficient functions and prediction errors.

Table 2.11: Learned equations for Burgers' equation.

x	True	SEQL	HEQL
0.627	$-1.15uu_x + 0.10u_{xx}$	$-1.15uu_x + 0.10u_{xx}$	$-1.16uu_x + 0.10u_{xx}$
3.137	$-1.00uu_x + 0.10u_{xx}$	$-1.00uu_x + 0.10u_{xx}$	$-1.01uu_x + 0.10u_{xx}$
5.647	$-0.85uu_x + 0.10u_{xx}$	$-0.86uu_x + 0.10u_{xx}$	$-0.85uu_x + 0.10u_{xx}$
8.157	$-1.24uu_x + 0.10u_{xx}$	$-1.24uu_x + 0.10u_{xx}$	$-1.25uu_x + 0.10u_{xx}$

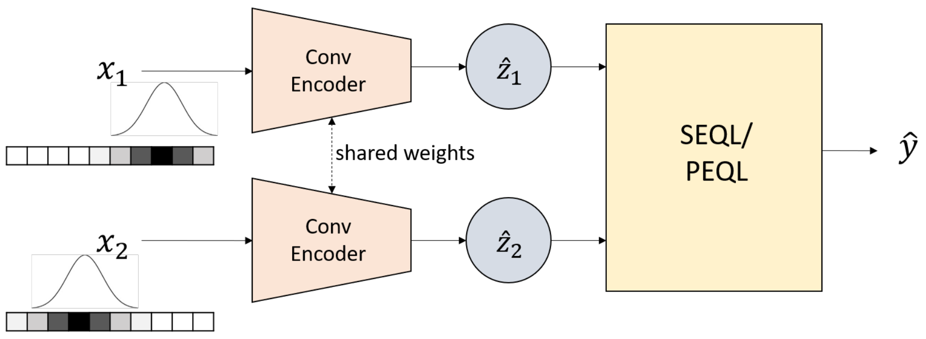


Figure 2-13: The combined architecture used for high-dimensional system tasks involving a convolutional encoder followed by an EQL network.

the correct equation as shown in Table 2.11. We see in Figure 2-12 that the SEQL network is able to accurately predict the function and the parametric coefficients. Note that while ref. [192] needs to pre-compute product terms of the individual spatial derivatives such as uu_x and $u_x^2 u_{xxx}^3$ and then perform a linear regression over these terms, the SEQL is able to learn non-linear relations on its own using the multiplication primitive function. So the SEQL is only given u, u_x, u_{xx}, $\ldots$ as inputs, but is able to learn the form of the nonlinear PDE.

2.7.3 Spring System

Finally, we demonstrate the ability of the parametric EQL networks to perform symbolic regression on structured, high-dimensional data by integrating with other deep learning architectures and training end-to-end.

We consider a dataset that consists of pairs of 1D images of point particles that interact through a spring-like force. The input data is a 1D grayscale image with 64 pixels which represents a 1D spatial domain $\psi \in [-4, 4]$. Each image contains a single particle, represented by a Gaussian with mean centered at its position ψ_i and a fixed variance of 0.1. We look at two different targets for symbolic regression: the spring force $F = -k(t)(\psi_2 - \psi_1)$ and the spring energy $E = \frac{k(t)}{2}(\psi_2 - \psi_1)^2$, where $k(t) = \frac{5-t}{2}$. The spring constant decreases over time, which we can imagine is representative of a spring degrading with use.

To approach this problem, we use the architecture shown in Figure 2-13. Each

Table 2.12: Learned equations of the SEQL on select t values for the spring force function $F(t, \psi_1, \psi_2) = -\frac{5-t}{2} \cdot (\psi_2 - \psi_1)$ in the latent space and transformed to the original parameter space.

t	True	Learned Latent	Learned Transformed
-2.619	$-3.81(\psi_2 - \psi_1)$	$-4.66\hat{z}_1 + 4.66\hat{z}_2$	$3.82\hat{\psi}_1 - 3.82\hat{\psi}_2$
-1.095	$-3.05(\psi_2 - \psi_1)$	$-3.72\hat{z}_1 + 3.72\hat{z}_2$	$3.05\hat{\psi}_1 - 3.05\hat{\psi}_2$
0.381	$-2.31(\psi_2 - \psi_1)$	$-2.82\hat{z}_1 + 2.82\hat{z}_2$	$2.31\hat{\psi}_1 - 2.31\hat{\psi}_2$
1.905	$-1.55(\psi_2 - \psi_1)$	$-1.89\hat{z}_1 + 1.89\hat{z}_2$	$1.55\hat{\psi}_1 - 1.55\hat{\psi}_2$

image is fed into a separate encoder, where the two encoders share the same weights. The encoder consists of 2 convolutional layers followed by 3 fully-connected layers and a batch normalization layer. The encoders each output a single-dimensional latent variable $\hat{z}_1, \hat{z}_2$, which are then fed into the parametric EQL network (which can be either the HEQL or the SEQL). The batch normalization layer serves to constrain the range of the latent variable so that the EQL network does not need to scale to arbitrarily-sized inputs when training end-to-end. The EQL network has a single scalar output, which is trained to match either the spring force or the spring energy. The entire network is trained end-to-end and is only shown the inputs and the output, but must learn an appropriate representation $\hat{z}_i$. While there are no constraints on the latent representation $\hat{z}_i$, we expect it to have a one-to-one mapping to the true position of the particle, ψ_i.

For all tests, 512 training data points with $\psi_1, \psi_2 \in [-3, 3]$ were sampled for each of 128 fixed values of $t \in [-3, 3]$. To evaluate the extrapolation ability of these architectures, training data points were restricted to pairs with $|\psi_2 - \psi_1| \leq 4$, while no such restriction was imposed on testing data. In addition, we compare against a baseline test of a model consisting of the same encoder architecture with a dense ReLU network replacing EQL network. We call this baseline the **ReLU network**.

Results for learning the spring force is shown in Figure 2-14. Both the SEQL network and the ReLU network successfully predict the force inside the training domain, but only the SEQL network is able to extrapolate outside of the training regime whereas the ReLU network completely fails to extrapolate. Additionally, the EQL network learns the governing equation as shown in Table 2.12, with the learned para-

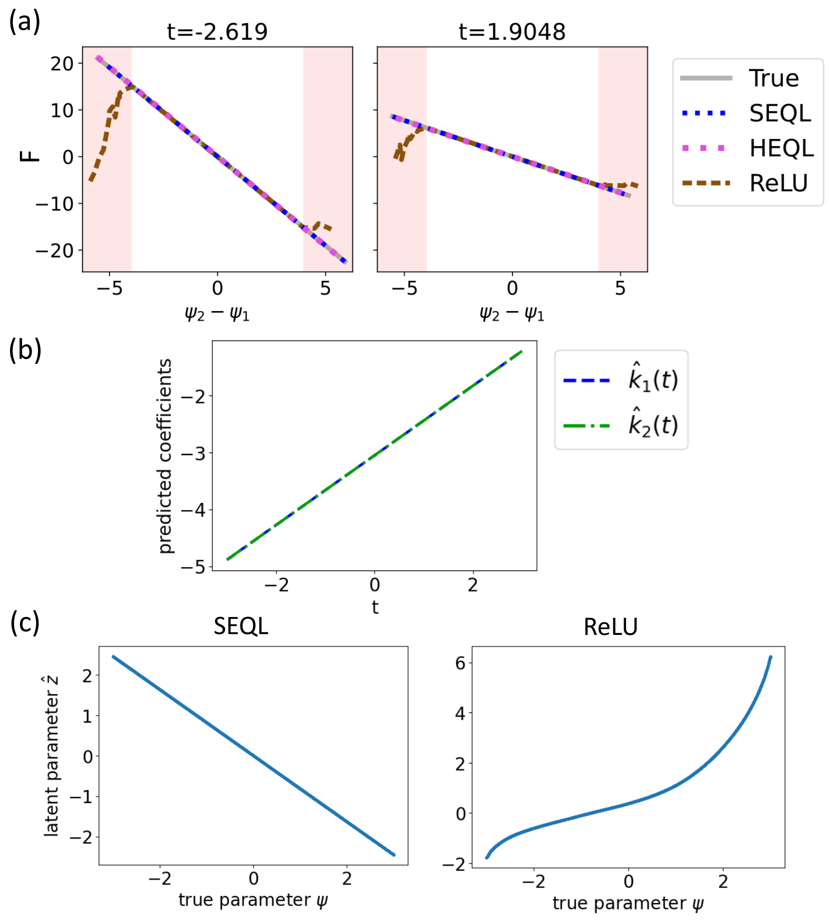


Figure 2-14: **Results for learning the spring force F.** (a) Predictions for select values of t. Outputs with $|\psi_2 - \psi_1| > 4$ (highlighted in red) are extrapolated. (b) Coefficient functions in the equation $\hat{F}(t, \hat{z}_1, \hat{z}_2) = \hat{k}_1(t) \cdot \hat{z}_1 - \hat{k}_2(t) \cdot \hat{z}_2$ learned by the SEQL network. (c) Latent variable encodings for the force function F learned by (left) the convolutional SEQL network and (right) the ReLU network.

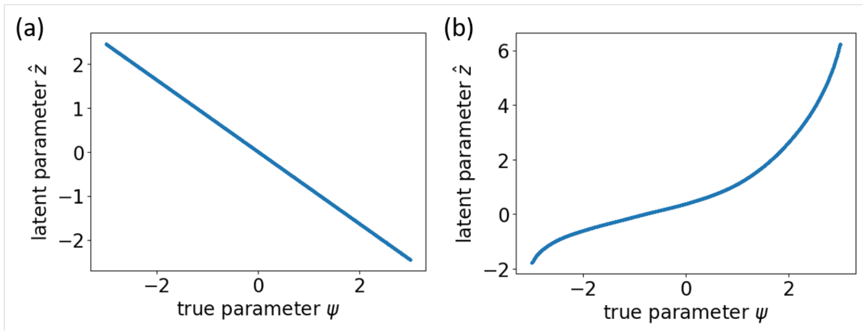


Figure 2-15: Latent variable encodings for the function $f(t, \psi_1, \psi_2) = -\frac{5-t}{2} \cdot (\psi_2 - \psi_1)$ learned by (a) the convolutional SEQL network and (b) the ReLU network.

Table 2.13: Learned equations of the HEQL on select t values for the function $E(t, \psi_1, \psi_2) = \frac{5-t}{4} \cdot (\psi_2 - \psi_1)^2$ in the latent space and transformed to the original parameter space.

t	True	Learned Latent	Learned Transformed
-2.619	$-1.90(\psi_2 - \psi_1)^2$	$6.59\hat{z}_1^2 + 6.59\hat{z}_2^2 - 13.18\hat{z}_1\hat{z}_2 + 0.02$	$1.91\hat{\psi}_1^2 + 1.91\hat{\psi}_2^2 - 3.82\hat{\psi}_1\hat{\psi}_2 + 0.02$
-1.095	$-1.52(\psi_2 - \psi_1)^2$	$5.27\hat{z}_1^2 + 5.27\hat{z}_2^2 - 10.55\hat{z}_1\hat{z}_2 + 0.01$	$1.53\hat{\psi}_1^2 + 1.53\hat{\psi}_2^2 - 3.06\hat{\psi}_1\hat{\psi}_2 + 0.01$
0.381	$-1.16(\psi_2 - \psi_1)^2$	$4.01\hat{z}_1^2 + 4.01\hat{z}_2^2 - 8.02\hat{z}_1\hat{z}_2 + 0.01$	$1.16\hat{\psi}_1^2 + 1.16\hat{\psi}_2^2 - 2.33\hat{\psi}_1\hat{\psi}_2 + 0.01$
1.905	$-0.77(\psi_2 - \psi_1)^2$	$2.64\hat{z}_1^2 + 2.64\hat{z}_2^2 - 5.28\hat{z}_1\hat{z}_2 + 0.01$	$0.77\hat{\psi}_1^2 + 0.77\hat{\psi}_2^2 - 1.53\hat{\psi}_1\hat{\psi}_2 + 0.01$

metric coefficient plotted in Figure 2-14(b). Note that the SEQL network learns the expression $\hat{F} = \hat{k}_1(t)\hat{z}_1 - \hat{k}_2(t)\hat{z}_2$, where we do not necessarily have $\hat{k}_1 = \hat{k}_2$. Upon inspection, however, we see that $\hat{k}_1(t) \approx \hat{k}_2(t)$ and so the SEQL network has discovered an approximately equal expression to what we expect.

Additionally, while the SEQL network discovers an equation in terms of $\hat{z}_{1,2}$, it also learns a linear mapping of the latent variable to the true position as shown in Figure 2-15(a). While there is no explicit constraint or regularization placed on the latent space, because the EQL network must learn to use the latent variable to form the equation, the end-to-end training of the architecture forces the mapping to be an analytical transformation of the original variable, which in this case is a linear mapping. In contrast, the latent variable mapping for the ReLU network is shown in Figure 2-15(b). While it is one-to-one, it is not linear since there is no bias to make the mapping linear. Using this linear mapping, we can perform a linear regression to find the approximate relationship between $\hat{z}$ and $\hat{\psi}$ and reconstruct the discovered equation in terms of $\hat{\psi}$, which is shown in the right-most column of Table 2.12.

We see similar results for the spring potential data, this time using the HEQL network, in Figures 2-16 and Table 2.13. Again, the HEQL network is able to extrapolate outside of the training regime whereas the ReLU network fails to extrapolate. Note that in this case, the HEQL learns the equation $\hat{E}(t, \hat{z}_1, \hat{z}_2) = \hat{k}_1(t)\hat{z}_1^2 + \hat{k}_2(t)\hat{z}_2^2 - 2\hat{k}_3(t)\hat{z}_1\hat{z}_2 + \epsilon(t)$ where $\hat{k}_1 \approx \hat{k}_2 \approx \hat{k}_3$ and ϵ is small. Thus, the HEQL network has discovered the correct equation.

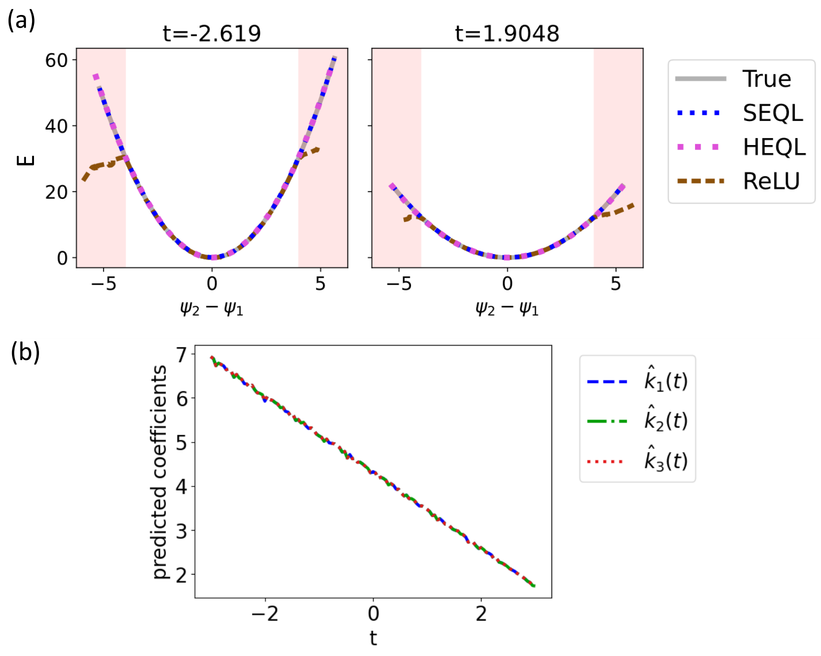


Figure 2-16: **Results for learning the spring energy E.** (a) Predictions for select values of t. Outputs with $|\psi_2 - \psi_1| > 4$ (highlighted in red) are extrapolated. (b) Coefficient functions in the equation $f(t, \hat{z}_1, \hat{z}_2) = \hat{k}_1(t) \cdot \hat{z}_1^2 + \hat{k}_2(t) \cdot \hat{z}_2^2 - 2\hat{k}_3(t) \cdot \hat{z}_1\hat{z}_2$ learned by the HEQL network.

2.8 Parametric Equations Discussion

We note that in our experiments we used analytic expressions for the varying coefficients for simplicity. However, our method is not constrained to these types of expressions, and the parametric coefficient can more generally be any arbitrary function. Thus, our method can be applied to systems that we know are partially governed by an analytic equation, but partially governed by some other mechanism that may be too complex or noisy to capture. This is similar in spirit to methods for solving PDEs that replace part of the equation with a neural network, often to correct for discretization errors [170, 105].

Comparing the two architectures, for a moderate number of time steps (e.g. $N_t < 512$) the SEQL has fewer parameters than the HEQL; despite this, however, the HEQL trains on each minibatch 3.7× faster than the SEQL on the analytic equations for our settings of hyper-parameters and network sizes. This is likely because the limiting factor is the computation of the activation functions, which must be processed separately for each component of h (whereas in a conventional neural network the use of a single activation function is able to take advantage of vectorization optimizations). For a larger number of time steps, (e.g. $N_t > 512$), the HEQL is more memory-efficient as well since the SEQL parameters scale linearly with the number of time steps. Thus, the HEQL is able to scale to larger datasets.

In terms of the data format, prior methods rely on gridded data [192, 250] while both the SEQL and the HEQL allow a variable grid along the varying dimension. The HEQL architecture takes this flexibility a step further in that it is able to interpolate in time and make predictions at arbitrary time points, whereas the stacked architecture is fixed to certain time points. On the other hand, we find that the stacked architecture is less sensitive to the random initialization and converges more quickly to the solution. Thus, we have a tradeoff between performance and flexibility. One possible direction for future work to bridge this gap is to introduce different learning rate schedules for the EQL network and the MWU in the HEQL architecture, as the EQL network typically requires large learning rates to escape local minima and converge, whereas large learning rates may be detrimental to the MWU.

As mentioned in Section 2.7, the SEQL and HEQL architectures are also more flexible than previous approaches in the types of equations that can be discovered. For example, the previous approaches rely on variants of linear regression, and are thus not able to discover varying coefficients that are inside other functions such as $\sin(f(t)x)$. Additionally, our approach is able to automatically discover cross terms whereas the SINDy framework relies on these terms being precomputed.

The parametric architectures presented here can be viewed as implementing functional regularization. Functional regularization, which imposes regularization on the learned function rather than on the parameters, is attractive as it is much more intuitive and can lead to more natural methods for tasks such as continual learning [12, 165]. It has been has been explored in neural networks through regularizing the predictions on batch of the data [12] and through defining the prior over functions rather than weights in the case of Bayesian neural networks [216, 191]. In the case of the EQL network, the coefficients of the resulting equation are typically very simple functions (oftentimes the identity function) of the weights themselves. This means that in practice, the L_2 smoothing regularization in the stacked EQL network architecture often implicitly applies to the *function space*, even though we are explicitly applying the regularization in the *weight space*. In the case of the parametric EQL architecture, the output of fully-connected neural networks will tend to be smooth due to modern training methods such as stochastic gradient descent (which is a deep topic of great interest in the literature), and so the MWU itself acts as a regularization on the function space of the EQL network.

2.9 Conclusion

We have shown how we can integrate symbolic regression with deep learning architectures and train the entire system end-to-end to take advantage of the powerful deep learning training techniques that have been developed in recent years. Namely, we show that we can learn arithmetic on MNIST digits where the system must learn to identify the images in an image recognition task while simultanesouly extracting the mathematical expression that relates the digits to the answer. Additionally, we show that we can simultaneously extract an unknown parameter from a dynamical system and extract the propagation equations. In the SHO system, the results suggest that we can discover new techniques for integrating ODEs, potentially paving the way for improved integrators, such as integrators for stiff ODEs that may be difficult to solve with numerical methods. Finally, we have proposed two different variants of the EQL network—the stacked architecture (SEQL) and the hyper architecture (HEQL)—to enable neural network-based symbolic regression of parametric systems where coefficients may vary. To this end, we demonstrated our system on parametric analytic equations and PDEs, as well as a dataset encoded as images. Altogether, these methods have the potential to combine the power of deep learning and symbolic regression to enable scientific discovery on complex and high-dimensional datasets.

One direction for future work is to study the role of random initializations and make the system less sensitive to random initializations. As seen by the benchmark results of the EQL network in Appendix A.1, the EQL network is not always able to find the correct mathematical expression. This is because there are a number of local minima in the EQL network that the network can get stuck in, and gradient-based optimization methods are only guaranteed to find local minima rather than global minima. Local minima are not typically a concern for neural networks because the local minima are typically close enough in performance to the global minimum [25]. However, for the EQL network, we often want to find the true global minimum. In this work, we have alleviated this issue by increasing stochasticity through large learning rates and by decreasing the sensitivity to random initializations by duplicating activation functions. Additionally, we run multiple trials and find the best results, either manually or through an automated system [144, 195]. In future work, it may be possible to find the true global minimum without resorting to multiple trials as it has been shown that over-parameterized neural networks with certain types of activation functions are able to reach the global minimum through gradient descent in linear time regardless of the random initialization [39].

Another path to improving convergence of the EQL network is to consider transformations of the primitive functions. For example, Ziyin et al. [271] use what they call a Snake function, defined as $x + \frac{1}{a}\sin^2(ax)$ where a is a learnable parameter, to learn periodic functions, as it maintains monotonicity while improving learning of periodic functions. The Snake function is also designed to work with commonly used initialization schemes such as Kaiming init, which assume a mean and variance of the preactivation values. Ramachandran et al. [183] performed an exhaustive search over combinations of many commonly used functions, and found that activations such as $\cos(x) - x$ and $\text{sinc}(x) + x$ performed extremely well on CIFAR datasets, pointing towards their nice convergence properties.

One limitation of the EQL network (and of symbolic regression through gradient-based methods in general) is the learning of functions with non-differentiable points, such as the division operator. Since rational functions are widely prevalent in science and engineering equations, a future direction should explore a robust way to learn these types of functions. For example, Padé Activation Units (PAU) use rational functions as activation units with the coefficients to be learned during training, although the proposing work initialized the coefficients to approximate commonly used activation functions (e.g., ReLU and Swish) to enable more flexible representation of these monotonic activations [154]. The neural arithmetic logic unit (NALU) offers a way to learn multiplication and division operators by reparameterizing the hidden units using logarithms [223] and follow-up work improve the stability and limitations of the original NALU [198]. Finally, Costa et al. [28] combine the EQL network with evolutionary strategies to enable learning of non-differentiable functions and programs.

Other directions for future work include expanding the types of deep learning architectures that the EQL network can integrate with. For example, supporting spatio-temporal systems can lead to PDE discovery. The spatial derivatives could be calculated using known finite-difference approximations or learnable kernels [125]. Additionally, the encoder can be expanded to capture a wider variety of data such as videos [23], audio signals, and text. These capabilities will allow deep learning to be applied in scientific exploration and discovery.

Chapter 3

Bayesian Optimization and Deep Learning for Scientific Problems with High-Dimensional Structure

3.1 Introduction

Bayesian optimization (BO) is a methodology well-suited for global (as opposed to local) optimization of expensive, black-box (e.g. derivative-free) functions and has been successfully applied to a wide range of problems in science and engineering [227, 60, 106] as well as hyperparameter tuning of machine learning models [210, 217, 104, 225, 189]. BO works by iteratively deciding the next data point to label in order to maximize sampling efficiency and minimize the number of data points required to optimize a function, which is critical in many contexts where experiments or simulations can be costly or time-consuming.

However, in many domains, the system is not a complete black box. For example, certain types of high-dimensional input spaces such as images or molecules have some known structure, symmetries and invariances. In addition, the function may be decomposed into other functions; rather than directly outputting the value of the objective, the data collection process may provide intermediate or auxiliary information from which the objective function can be cheaply computed. For example, a scientific experiment or simulation may produce a high-dimensional observation or multiple measurements simultaneously, such as the optical scattering spectrum of a nanoparticle over a range of wavelengths, or multiple quantum chemistry properties of a molecule from a single density functional theory (DFT) calculation. All of these physically-informed insights into the system are potentially useful and important factors for designing surrogate models through inductive biases, but they are often not fully exploited in existing methods and applications.

BO relies on specifying a surrogate model which captures a distribution over potential functions to incorporate uncertainty in its predictions. These surrogate models are typically Gaussian Processes (GPs), as the posterior distribution of GPs can be expressed analytically. However, (1) inference in GPs scales cubically in time with the number of observations and output dimensionality, limiting their use to smaller datasets or to problems with low output dimensionality without the use of kernel approximations, and (2) GPs operate most naturally over continuous low-dimensional input spaces, so kernels for high-dimensional data with complex structure must be carefully formulated by hand for each new domain. Thus, encoding inductive biases can be challenging.

Neural networks (NNs) and Bayesian neural networks (BNNs) have been proposed as an alternative to GPs due to their scalability and flexibility [211, 214]. Alternatively, neural networks have also been used to create continuous latent spaces so that BO with vanilla GPs can be more easily applied [110, 224]. The ability to incorporate a variety of constraints, symmetries, and inductive biases into BNN architectures offers the potential for BO to be applied to more complex tasks with structured data.

This work demonstrates the use of deep learning to enable BO for complex, real-world scientific datasets, without the need for pre-trained models. In particular:

- We take advantage of auxiliary or intermediate information to improve BO for tasks with high-dimensional observations.

- We demonstrate BO on complex input spaces including images and molecules using convolutional and graph neural networks, respectively.

- We apply BO to several realistic scientific datasets, including the optical scattering of a nanoparticle, topology optimization of a photonic crystal material, and chemical property optimization of molecules from the QM9 dataset.

We show that neural networks are often able to significantly outperform GPs as surrogate models on these problems, and we believe that these strong results will also generalize to other contexts and enable BO to be applied to a wider range of problems. We note that while our methods are based on existing methods, we use a novel combination of these methods to tailor existing BO frameworks to real-world, complex applications.

3.2 Related Work

Various methods have been formulated to scale GPs to larger problems. For example, Bruinsma et al. [18] proposes a framework for multi-output GPs that scale linearly with m, where m is the dimensionality of a low-dimensional sub-space of the data. Maddox et al. [139] uses multi-task GPs to perform BO over problems with large output dimensionalities. Additionally, GPs have been demonstrated on extremely large datasets through the use of GPUs and intelligent preconditioners [52, 235] or through the use of various approximations [181, 239, 120, 140].

Another approach to scaling BO to larger problems is by combining it with other methods such that the surrogate model does not need to train on the entire dataset. For example, TuRBO uses a collection of independent probabilistic models in different trust regions, iteratively deciding in which trust region to perform BO and thus reducing the problem to a set of local optimizations [41]. Methods such as LA-MCTS build upon TuRBO and dynamically learn the partition function separating different regions [236].

GPs have been extended to complex problem settings to enable BO on a wider variety of problems. Astudillo and Frazier [9] decompose synthetic problems as a composition of other functions, and take advantage of the additional structure to improve BO. However, the multi-output GP they use scales poorly with output dimensionality, and so this approach is limited to simpler problems. This work has also been extended [10, 139]. GP kernels have also been formulated for complex input spaces including convolutional kernels [229, 159, 245] and graph kernels [202, 231]. The graph kernels have been used to apply BO to neural architecture search (NAS) where the architecture and connectivity of a neural network itself can be optimized [189].

Deep learning has been used as a scalable and flexible surrogate model for BO. In particular, Snoek et al. [211] uses neural networks as an adaptive basis function for Bayesian linear regression, which allows BO to scale to large datasets. This approach also enables BO in more complex settings including transfer learning of the adaptive basis across multiple tasks, and modeling of auxiliary signals to improve performance [171]. Additionally, Bayesian neural networks (BNNs) that use Hamiltonian Monte Carlo to sample the posterior have been used for single-task and multi-task BO for hyperparameter optimization [214].

Another popular approach for BO on high-dimensional spaces is latent-space approaches. Here, an autoencoder such as a VAE is trained on a dataset to create a continuous latent space that represents the data. From here, a more conventional optimization algorithm, such as BO using GPs, can be used to optimize over the continuous latent space. This approach has been applied to complex tasks such as arithmetic expression optimization and chemical design [110, 58, 60, 224, 36]. Note that these approaches focus on both data generation and optimization simultaneously, whereas our work focuses on just the optimization process.

Random forests have also been used for iterative optimization such as sequential model-based algorithm configuration (SMAC) as they do not face scaling challenges [82]. Tree-structured Parzen Estimators (TPE) are also a popular choice for hyperparameter tuning [13]. However, these approaches still face the same issues with encoding complex, structured inputs such as images and graphs.

Deep learning has also been applied to improve tasks other than BO. For example, active learning is a similar scheme to BO that, instead of optimizing an objective function, aims to optimize the predictive ability of a model with as few data points as possible. The inductive biases of neural networks has enabled active learning on a variety of high-dimensional data including images [49], language [205], and partial differential equations [262]. BNNs have also been applied to the contextual bandits problem, where the model chooses between discrete actions to maximize expected reward [14, 187].

3.3 Bayesian Optimization

3.3.1 Prerequisites

Now, we briefly introduce the BO methodology; more details can be found in the literature [17, 200, 53]. We formulate our optimization task as a maximization problem in which we wish to find the input $\mathbf{x}^* \in \mathcal{X}$ that maximizes some function f such that $\mathbf{x}^* = \arg\max_{\mathbf{x}} f(\mathbf{x})$. The input x is most simply a real-valued continuous vector, but can be generalized to categorical variables, images, or even discrete objects such as molecules. The function f returns the value of the objective $y = f(x)$ (which we also refer to as the "label" of x), and can represent some performance metric that we wish to maximize. In general f can be a noisy function.

A key ingredient in BO is the surrogate model that produces a distribution of predictions as opposed to a single point estimate for the prediction. Such surrogate models are ideally Bayesian models, but in practice, a variety of approximate Bayesian models or even frequentist (i.e. empirical) distributions have been used. In iteration N, a Bayesian surrogate model $\mathcal{M}$ is trained on a labeled dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_n, y_n)\}_{n=1}^N$. An acquisition function α then uses $\mathcal{M}$ to suggest the next data point $\mathbf{x}_{N+1} \in \mathcal{X}$ to label, where

$$\mathbf{x}_{N+1} = \arg\max_{\mathbf{x}\in\mathcal{X}} \alpha\left(\mathbf{x}; \mathcal{M}, \mathcal{D}_{\text{train}}\right). \quad (3.1)$$

The new data is evaluated to get $y_{N+1} = f(\mathbf{x}_{N+1})$, and $(\mathbf{x}_{N+1}, y_{N+1})$ is added to

$\mathcal{D}_{\text{train}}$.

3.3.2 Acquisition Function

An important consideration within BO is how to choose the next data point $\mathbf{x}_{N+1} \in \mathcal{X}$ given the model $\mathcal{M}$ and labelled dataset $\mathcal{D}_{\text{train}}$. This is parameterized through the "acquisition function" α, which we maximize to get the next data point to label as shown in Equation 3.1.

We choose the expected improvement (EI) acquisition function α_{EI} [93]. When the posterior predictive distribution of the surrogate model $\mathcal{M}$ is a normal distribution $\mathcal{N}(\mu(\mathbf{x}), \sigma^2(\mathbf{x}))$, EI can be expressed analytically as

$$\alpha_{\text{EI}}(\mathbf{x}) = \sigma(\mathbf{x}) \left[\gamma(\mathbf{x})\Phi(\gamma(\mathbf{x})) + \phi(\gamma(\mathbf{x}))\right], \tag{3.2}$$

where $\gamma(\mathbf{x}) = (\mu(\mathbf{x}) - y_{\text{best}})/\sigma(\mathbf{x})$, $y_{\text{best}} = \max(\{y_n\}_{n=1}^{N})$ is the best value of the objective function so far, and ϕ and Φ are the PDF and CDF of the standard normal $\mathcal{N}(0, 1)$, respectively. For surrogate models that do not give an analytical form for the posterior predictive distribution, we sample from the posterior N_{MC} times and use a Monte Carlo (MC) approximation of EI:

$$\alpha_{\text{EI-MC}}(\mathbf{x}) = \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} \max\left(\mu^{(i)}(\mathbf{x}) - y_{\text{best}}, 0\right). \tag{3.3}$$

where $\mu^{(i)}$ is a prediction sampled from the posterior of $\mathcal{M}$ [246]. While works such as [111] fit the output of the surrogate model to a Gaussian to use Eq. 3.2 for acquisition, this is not valid when the model prediction for y is not Gaussian, which is generally the case for composite functions (see Section 3.3.4).

EI has the advantage over other acquisition functions in that the MC approximation (1) remains differentiable to facilitate optimization of the acquisition function in the inner loop (i.e. the MC approximation of upper confidence bound (UCB) is not differentiable and can result in ties) and (2) is inexpensive (i.e. naive Thompson sampling for ensembles would require re-training a model from scratch in each iteration).

3.3.3 Continued Training with Learning Rate Annealing

One challenge is that training a surrogate model on $\mathcal{D}_{\text{train}}$ from scratch in every optimization loop adds a large computational cost that limits the applicability of BO, especially since neural networks are ideally trained for a long time until convergence. To minimize the training time of BNNs in each optimization loop, we use the model that has been trained in the Nth optimization loop iteration as the initialization (also known as a "warm start") for the $(N+1)$th iteration, rather than training from a random initialization. In particular, we use the cosine annealing learning rate proposed in Loshchilov and Hutter [127] which starts with a large learning rate and drops the learning rate to 0. For more details, refer to Section C.3 in the Appendix.

3.3.4 Auxiliary Information

Typically we assume f is a black box function, so we train $\mathcal{M}\colon \mathcal{X} \to \mathcal{Y}$ to model f. Here we consider the case where the experiment or observation may provide some intermediate or auxiliary information $\mathbf{z} \in \mathcal{Z}$, such that f can be decomposed as

$$f(\mathbf{x}) = h(g(\mathbf{x})), \tag{3.4}$$

where $g\colon \mathcal{X} \to \mathcal{Z}$ is the expensive labeling process, and $h\colon \mathcal{Z} \to \mathcal{Y}$ is a known objective function that can be cheaply computed. Note that this is also known as "composite functions" [9, 10]. In this case, we train $\mathcal{M}\colon \mathcal{X} \to \mathcal{Z}$ to model g, and the approximate EI acquisition function becomes

$$\alpha_{\text{EI-MC-aux}}(\mathbf{x}) = \frac{1}{N_{\text{MC}}} \sum_{i=1}^{N_{\text{MC}}} \max\left(h\left(\mu^{(i)}(\mathbf{x})\right) - y_{\text{best}}, 0\right). \tag{3.5}$$

which can be seen as a Monte Carlo version of the acquisition function presented in Astudillo and Frazier [9]. We denote models trained using auxiliary information with the suffix "-aux." Because h is not necessarily linear, $h\left(u^{(i)}(\mathbf{x})\right)$ is not in general

Gaussian even if $u^{(i)}$ itself may be, which makes the MC approximation convenient or even necessary.

3.4 Surrogate Models

Bayesian models are able to capture uncertainty associated with both the data and the model parameters in the form of probability distributions. To do this, there is a *prior* probability distribution $P(\theta)$ placed upon the model parameters, and the *posterior* belief of the model parameters can be calculated using Bayes' theorem upon observing new data. Fully Bayesian neural networks have been studied in small architectures, but are impractical for realistically-sized neural networks as the nonlinearities between layers render the posterior intractable, thus requiring the use of MCMC methods to sample the posterior. In the last decade, however, there have been numerous proposals for approximate Bayesian neural networks that are able to capture some of the Bayesian properties and produce a predictive probability distribution. In this work, we compare several different options for the BNN surrogate model. In addition, we compare against other non-BNN baselines. We list some of the more notable models here, and model details and results can be found in Section C.4.1 of the Appendix.

Ensembles combine multiple models into one model to improve predictive performance by averaging the results of the single models Ensembles of neural networks have been reported to be more robust than other BNNs [163], and we use "ENSEMBLE" to denote an ensemble of neural networks with identical architectures but different random initializations, which provide enough variation for the individual models to give different predictions. Using the individual models can be interpreted as sampling from a posterior distribution, and so we use Eq. 3.5 for acquisition. Our ensemble size is $N_{\mathrm{MC}} = 10$.

Other BNNS. We also compare to variational BNNs including Bayes by Backprop (BBB) [14] and Multiplicative Normalizing Flows (MNF) [128]; BOHAMIANN [214]; and NEURALLINEAR [211]. For BBB, we also experiment with KL annealing, denoted by "-Anneal."

GP Baselines. GPs are largely defined by their kernel (also called "covariance functions") which determines the prior and posterior distribution, how different data points relate to each other, and the type of data the GP can operate on. In this work, we will use "GP" to refer to a specific, standard specification that uses a Matérn 5/2 kernel, a popular kernel that operates over real-valued continuous spaces. To operate on images, we use a convolutional kernel, labeled as "ConvGP", which is implemented using the infinite-width limit of a convolutional neural network [159]. Finally, to operate directly on graphs, we use the Weisfeiler-Lehman (WL) kernel as implemented by [189], which we label as "GraphGP". The WL kernel is able to operate on undirected graphs containing node and edge features making it appropriate for chemical molecule graphs, and was used by Ru et al. [189] to optimize neural network architectures in a method they call NAS-BOWL. Additionally, we compare against "GP-aux" which use multi-output GPs for problems with auxiliary information (also known as composite functions) [9]. In the Appendix, we also look at GPs that use infinite-width and infinite-ensemble neural network limits as the kernel [159] as well as TuRBO which combines GP-based BO with trust regions [41].

VAE-GP uses a VAE trained ahead of time on an unlabelled dataset representative of $\mathcal{X}$. This allows us to encode complex input spaces, such as chemical molecules, into a continuous latent space over which conventional GP-based BO methods can be applied, even enabling generation and discovery of novel molecules that were not contained in the original dataset. Here, we modified the implementation provided by [224] in which they use a junction tree VAE (JTVAE) to encode chemical molecules [86]. More details can be found in the Appendix.

Other Baselines. We compare against two variations of Bayesian optimization, TuRBO [41] and TPE [13]. We also compare against several global optimization algorithms that do not use surrogate models and are cheap to run, including LIPO [142], DIRECT-L [48], and CMA-ES.

We emphasize that ensembles and variational methods can easily scale up to high-dimensional outputs with minimal increase in computational cost by simply

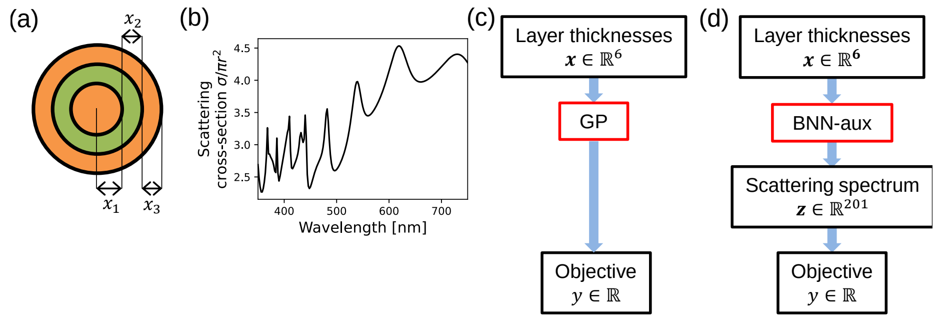

(a)
x_2
x_1
x_3
(b)
Scattering cross-section $\sigma/\pi r^2$
Wavelength [nm]
(c)
Layer thicknesses $\boldsymbol{x} \in \mathbb{R}^6$
GP
Objective $y \in \mathbb{R}$
(d)
Layer thicknesses $\boldsymbol{x} \in \mathbb{R}^6$
BNN-aux
Scattering spectrum $\boldsymbol{z} \in \mathbb{R}^{201}$
Objective $y \in \mathbb{R}$

Figure 3-1: (a) A cross-section of a three-layer nanoparticle parameterized by the layer thicknesses. (b) An example of the scattering cross-section spectrum of a six-layer nanoparticle. (c) Whereas GPs are trained to directly predict the objective function, (d) multi-output BNNs can be trained with auxiliary information, which here is the scattering spectrum.

changing the output layer size. Neural Linear and GPs scale cubically with output dimensionality (without the use of covariance approximations), making them difficult to train on high-dimensional auxiliary or intermediate information.

3.5 Results

We now look at three real-world scientific optimization tasks all of which provide intermediate or auxiliary information that can be leveraged. In the latter two tasks, the structure of the data also becomes important and hence BNNs with various inductive biases significantly outperform GPs and other baselines. For simplicity, we only highlight results from select architectures (see Appendix for full results along with dataset and hyperparameter details). All BO results are averaged over multiple trials, and the shaded area in the plots represents $\pm$ one standard error over the trials.

3.5.1 Multilayer Nanoparticle

We first consider the simple problem of light scattering from a multilayer nanoparticle, which has a wide variety of applications that demand a tailored optical response [56] including biological imaging [196], improved solar cell efficiency [74, 204], and catalytic materials [219]. In particular, the nanoparticle we consider consists of a lossless silica core and 5 spherical shells of alternating TiO_2 and silica. The nanoparticle is

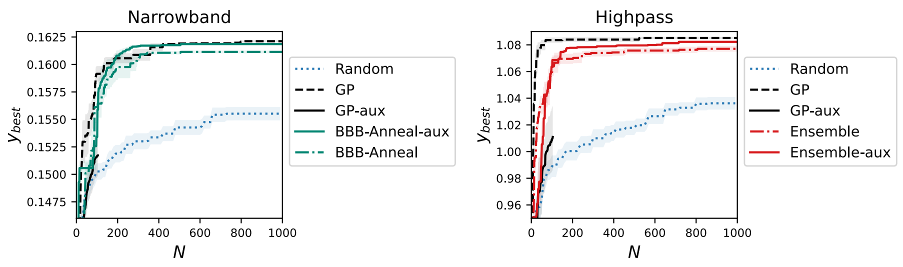


Figure 3-2: BO results for two different objective functions for the nanoparticle scattering problem. Training with auxiliary information (where $\mathcal{M}$ is trained to predict $\mathbf{z}$) is denoted with “-aux”. Adding auxiliary information to BNNs significantly improves performance.

parameterized by the core radius and layer thicknesses as shown in Figure 3-1(a), which we restrict to the range 30 nm to 70 nm. Because the size of the nanoparticle is on the order of the wavelength of light, its optical properties can be tuned by the number and thicknesses of the layers. The scattering spectrum can be calculated semi-analytically, as detailed in Section C.1.1 of the Appendix.

We wish to optimize the scattering cross-section spectrum over a range of visible wavelengths, an example of which is shown in Figure 3-1(b). In particular, we compare two different objective functions: the narrowband objective that aims to maximize scattering in the small wavelength range 600 nm to 640 nm and minimize it elsewhere, and the highpass objective that aims to maximize scattering above 600 nm and minimize it elsewhere. While conventional GPs train using the objective function as the label directly, BNNs with auxiliary information can be trained to predict the full scattering spectrum, i.e. the auxiliary information $\mathbf{z} \in \mathbb{R}^{201}$, which is then used to calculate the objective function, as shown in Figure 3-1(c,d).

BO results are shown in Figure 3-2. Adding auxiliary information significantly improves BO performance for BNNs. Additionally, they are competitive with GPs, making BNNs a viable approach for scaling BO to large datasets. In Appendix C.5, we see similar trends for other types of BNNs. Due to poor scaling of multi-output GPs with respect to output dimensionality, we are only able to run GP-AUX for a small number of iterations in a reasonable time. Within these few iterations, GP-AUX performs poorly, only slightly better than random sampling. We also see in the

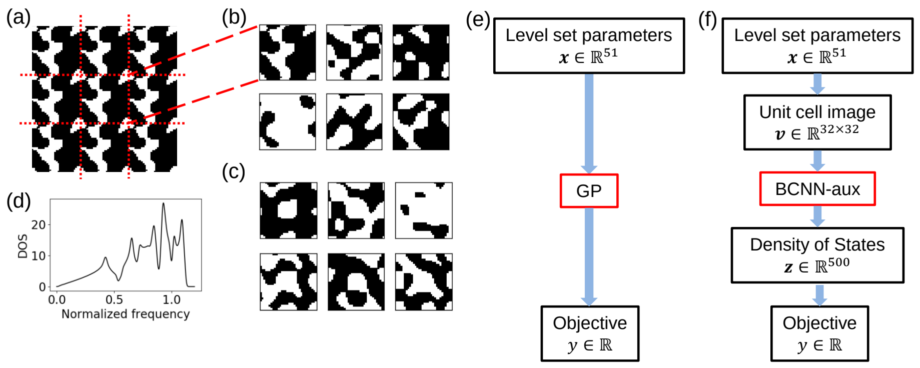


Figure 3-3: (a) A 2D photonic crystal (PC). The black and white regions represent different materials, and the periodic unit cells are outlined in red. Examples of PC unit cells drawn from the (b) PC-A distribution and (c) the PC-B distributions. The PC-A data distribution is translation invariant, whereas unit cells drawn from the PC-B distribution all have white regions in the middle of the unit cell, so the distribution is not translation invariant. (d) Example of a PC density of states (DOS). (e, f) Comparison of the process flow for training the surrogate model in the case of (e) GPs and (f) Bayesian Convolutional NNs (BCNN). The BCNN can train directly on the images to take advantage of the structure and symmetries in the data, and predict the multi-dimensional DOS.

Appendix that BO with either GPs or BNNs are comparable with, or outperform other global optimization algorithms, including DIRECT-L and CMA-ES.

3.5.2 Photonic Crystal Topology

Next we look at a more complex, high-dimensional domain that contains symmetries not easily exploitable by GPs. Photonic crystals (PCs) are nanostructured materials that are engineered to exhibit exotic optical properties not found in bulk materials, including photonic band gaps, negative index of refraction, and angular selective transparency [89, 254, 87, 201]. As advanced fabrication techniques are enabling smaller and smaller feature sizes, there has been growing interest in inverse design and topology optimization to design even more sophisticated PCs [84, 148] for applications in photonic integrated circuits, flat lenses, and sensors [173, 117].

Here we consider 2D PCs consisting of periodic unit cells represented by a 32×32 pixel image, as shown in Figure 3-3(a), with white and black regions representing vacuum (or air) and silicon, respectively. Because optimizing over raw pixel values may lead to pixel-sized features or intermediate pixel values that cannot be fabricated, we have parameterized the PCs with a level-set function $\phi\colon \mathcal{X} \to \mathcal{V}$ that converts a 51-dimensional feature vector $\mathbf{x} = [c_1, c_2, ..., c_{50}, \Delta] \in \mathbb{R}^{51}$ representing the level-set parameters into an image $\mathbf{v} \in \mathbb{R}^{32\times32}$ that represents the PC. More details can be found in Section C.1.2 in the Appendix.

We test BO on two different data distributions, which are shown in Figure 3-3(b,c). In the PC-A distribution, $\mathbf{x}$ spans $c_i \in [-1, 1]\,, \Delta \in [-3, 3]$. In the PC-B distribution, we arbitrarily restrict the domain to $c_i \in [0, 1]$. The PC-A data distribution is translation invariant, meaning that any PC with a translational shift will also be in the data distribution. However, the PC-B data distribution is not translation invariant, as shown by the white regions in the center of all the examples in Figure 3-3(c).

The optical properties of PCs can be characterized by their photonic density of states (DOS), e.g. see Figure 3-3(d). We choose an objective function that aims to minimize the DOS in a certain frequency range while maximizing it everywhere else, which corresponds to opening up a photonic band gap in said frequency range. As shown in Figure 3-3(e,f), we train GPs directly on the level-set parameters $\mathcal{X}$, whereas we train the Bayesian convolutional NNs (BCNNs) on the more natural unit cell image space $\mathcal{V}$. BCNNs can also be trained to predict the full DOS as auxiliary information $\mathbf{z} \in \mathbb{R}^{500}$.

The BO results, seen in Figure 3-4(a), show that BCNNs outperform GPs by a significant margin on both datasets, which is due to both the auxiliary information and the inductive bias of the convolutional layers, as shown in Figure 3-4(b). Because the behavior of PCs is determined by their topology rather than individual pixel values or level-set parameters, BCNNs are much better suited to analyze this dataset compared to GPs. Additionally, BCNNs can be made much more data-efficient since they directly encode translation invariance and thus learn the behavior of a whole class of translated images from a single image. Because GP-AUX is extremely expensive compared to GP (500$\times$ longer on this dataset), we are only able to run GP-AUX for a small number of iterations, where it performs comparably to random sampling.

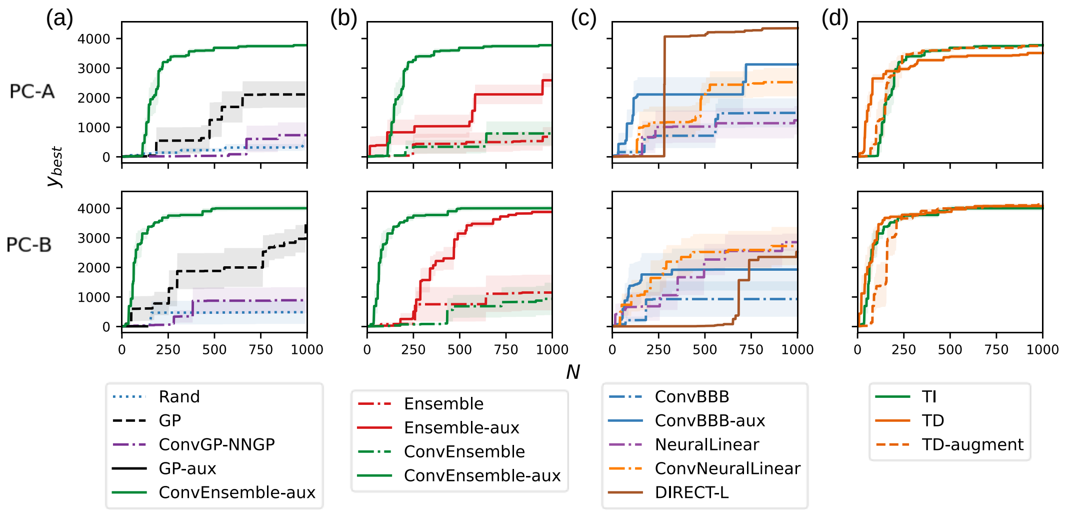


Figure 3-4: Three sets of comparisons for BO results on the (top row) PC-A and (bottom row) PC-B datasets. (a) BNNs with inductive biases outperform all other GP baselines and the random baseline. Note that GP-AUX is comparable to random sampling. (b) The inductive bias of convolutional layers and the addition of auxiliary information significantly improve performance of BCNNs. (c) Additional comparisons. (d) Data augmentation boosts performance if the augmentations reflect a symmetry present in the dataset but not enforced by the model architecture. "TI" refers to a translation invariant BCNN architecture, whereas "TD" refers to a translation dependent architecture. "-augment" signifies that data augmentation of the photonic crystal image is applied, which includes periodic translations, flips, and rotations.

We also compare to GPs using a convolutional kernel ("ConvGP-NNGP") in Figure 3-4(a). ConvGP-NNGP only performs slightly better than random sampling, which is likely due to a lack of auxiliary information and inflexibility to learn the most suitable representation for this dataset.

For our main experiments with BCNNs, we use an architecture that respects translation invariance. To demonstrate the effect of another commonly used deep learning training technique, we also experiment with incorporating translation invariance into a translation dependent (i.e. *not* translation invariant) architecture using a data augmentation scheme in which each image is randomly translated, flipped, and rotated during training. We expect data augmentation to improve performance when the data distribution exhibits the corresponding symmetries: in this case, we focus on translation invariance. As shown in Figure 3-4(c), we indeed find that data augmentation improves the BO performance of the translation dependent architecture when trained on the translation invariant PC-A dataset, even matching the performance of a translation invariant architecture on PC-A. However, on the translation dependent PC-B dataset, data augmentation initially hurts the BO performance of the translation dependent architecture because the model is unable to quickly specialize to the more compact distribution of PC-B, putting its BO performance more on par with models trained on PC-A. These results show that techniques used to improve generalization performance (such as data augmentation or invariant architectures) for training deep learning architectures can also be applied to BO surrogate models and, when used appropriately, directly translate into improved BO performance. Note that data augmentation would not be feasible for GPs without a hand-crafted kernel as the increased size of the dataset would cause inference to become computationally intractable.

3.5.3 Organic Molecule Quantum Chemistry

Finally, we optimize the chemical properties of molecules. Chemical optimization is of significant interest in both academia and industry with applications in drug design and materials optimization [80]. This is a difficult problem where computational approaches such as density functional theory (DFT) can take days for simple molecules and are intractable for larger molecules; synthesis is expensive and time-consuming, and the space of synthesizable molecules is large and complex. There have been many approaches for molecular optimization that largely revolve around finding a continuous latent space of molecules [58] or hand-crafting kernels to operate on molecules [106].

Here we focus on the QM9 dataset [190, 184], which consists of 133,885 small organic molecules along with their geometric, electronic, and thermodynamics quantities that have been calculated with DFT. Instead of optimizing over a continuous space, we draw from the fixed pool of available molecules and iteratively select the next molecule to add to $\mathcal{D}_{\text{train}}$. This is a problem setting especially common to materials design where databases are incomplete and the space of experimentally-feasible materials is small.

We use a Bayesian graph neural network (BGNN) for our surrogate model, as GNNs have become popular for chemistry applications due to the natural encoding of a molecule as a graph with atoms and bonds as nodes and edges, respectively. For baselines that operate over continuous spaces (i.e. GPs and simple neural networks), we use the Smooth Overlap of Atomic Positions (SOAP) descriptor to produce a fixed-length feature vector for each molecule, as shown in Figure 3-5(a) [33, 72].

We compare two different optimization objectives derived from the QM9 dataset: the isotropic polarizability α and $(\alpha - \epsilon_{\text{gap}})$ where ϵ_{gap} is the HOMO-LUMO energy gap. Other objectives are included in Appendix C.5.4. Because many of the chemical properties in the QM9 dataset can be collectively computed by a single DFT or molecular dynamics calculation, we can treat a group of labels from QM9 as auxiliary information $\mathbf{z}$ and train our BGNN to predict this entire group simultaneously. The objective function h then simply picks out the property of interest.

As shown in Figure 3-5(c), GraphGP and the BGNN variants significantly outperform GPs, showing that the inductive bias in the graph structure leads to a much more natural representation of the molecule and its properties. In the case of maximizing the polarizability α, including the auxiliary information improves BO per-

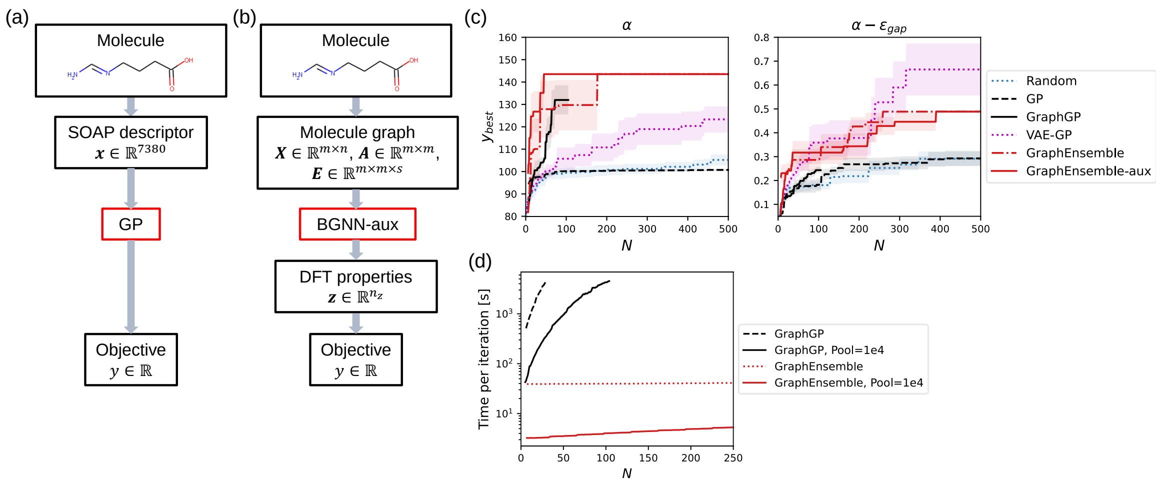


Figure 3-5: Quantum chemistry task and results. (a) The GP is trained on the SOAP descriptor, which is precomputed for each molecule. (b) The BGNN operates directly on a graph representation of the molecule, where atoms and bonds are represented by nodes and edges, respectively. The BGNN can be trained on multiple properties given in the QM9 dataset. (c) BO results for various properties. Note that GraphEnsemble is a type of BGNN. (d) Time per BO iteration on the GPU. (Note the logarithmic scale on the y-axis.) GraphGP takes orders of magnitudes longer than BGNNs for moderate N.

formance, showing signs of positive transfer. However, it does not have a significant impact on the other objectives, which may be due to the small size of the available auxiliary information (only a handful of chemical properties from the QM dataset) as compared with the nanoparticle and photonic crystal tasks. In a more realistic online setting, we would have significantly more physically-informative information available from a DFT calculation, e.g. we could easily compute the electronic density of states (the electronic analogue of the auxiliary information used in the photonics task).

As seen in Figure 3-5(d), we also note that the GraphGP is relatively computationally expensive (15× longer than GPs for small N and 800× longer than BGNNs for $N = 100$) and so we are only able to run it for a limited N in a reasonable time frame. We see that BGNNs perform comparably or better than GraphGPs despite incurring a fraction of the computational cost.

VAE-GP uses a modified version of the latent-space optimization method implementation provided by Tripp et al. [224]. Rather than optimizing over a continuous latent space of the VAE, we feed the data pool through the VAE encoder to find their latent space representation, and then apply the acquisition function to the latent points to pick out the best unlabeled point to sample. We keep as many hyper-parameters the same as the original implementation as possible, with the exception of the weighted re-training which we forgo since we have a fixed data pool that was used to train the VAE. This setup is similar to GraphNeuralLinear in that a deep learning architecture is used to encode the molecule as a continuous vector, although GraphNeuralLinear is only trained on the labelled data. The results for this experiment show that VAE-GP performs worse than BNNs on two of the three objective functions we tested and slightly better on one objective. We also note that the performance of VAE-GP depends very heavily on the pre-training of the VAE, as choosing different hyper-parameters or even a different random seed can significantly deteriorate performance (see Figure C-10 in the Appendix).

3.6 Discussion

Introducing physics-informed priors (in the form of inductive biases) into the model are critical for their performance. Well-known inductive biases in deep learning include convolutional and graph neural networks for images and graph structures, respectively, which significantly improve BO performance. Another inductive bias that we introduce is the addition of auxiliary information present in composite functions, which significantly improves the performance of BO for the nanoparticle and photonic crystal tasks. We conjecture that the additional information forces the BNN to learn a more consistent physical model of the system since it must learn features that are shared across the multi-dimensional auxiliary information, thus enabling the BNN to generalize better. For example, the scattering spectrum of the multilayer particle consists of multiple resonances (sharp peaks), the width and location of which are determined by the material properties and layer thicknesses. The BNN could potentially learn these more abstract features, and thus, the deeper physics, to help it interpolate more efficiently, akin to data augmentation [172]. Auxiliary information can also be interpreted as a form of data augmentation. Indeed, tracking the prediction error on a validation set shows that models with auxiliary information tend to have a lower loss than those without (see Appendix C.5). It is also possible that the loss landscape for the auxiliary information is smoother than that of the objective function and that the auxiliary information acts as an implicit regularization that improves generalization performance.

Interestingly, GP-AUX performs extremely poorly on the nanoparticle and photonic crystal tasks. One possible reason is that we are only able to run GP-AUX for a few iterations, and it is not uncommon for GP-based BO to require some critical number of iterations to reach convergence especially in the case of high-dimensional systems where the size of the covariance matrix scales with the square of the dimensionality. It may also be possible that GP-AUX only works on certain types of decompositions of functions and cannot be applied broadly to all composite functions, as the inductive biases in GPs are often hard-coded.

There is an interesting connection between how well BNNs are able to capture and explore a multi-modal posterior distribution and their performance in BO. For example, we have noticed that larger batch sizes tend to significantly hurt BO performance. On the one hand, larger batch sizes may be resulting in poorer generalization as the model finds sharper local minima in the loss landscape. Another explanation is that the stochasticity inherent in smaller batch sizes allows the BNN to more easily explore the posterior distribution, which is known to be highly multi-modal [45]. Indeed, BO often underperforms for very small dataset sizes N but quickly catches up as N increases, indicating that batch size is an important hyperparameter which must be balanced with computational cost.

All our results use continued training (or warm restart) to minimize training costs. We note that re-initializing $\mathcal{M}$ and training from scratch in every iteration performs better than continued training on some tasks (results in the Appendix), which points to how BNNs may not sufficiently represent a multi-modal posterior distribution or that continued training may skew the training distribution that the BNN sees. Future work will consider using stochastic training approaches such as SG-MCMC methods for exploring posterior distributions [241, 264] as well as other continual learning techniques to further minimize training costs, especially for larger datasets [167].

When comparing BNN architectures, we find that ensembles tend to consistently perform among the best, which is supported by previous literature showing that ensembles capture uncertainty much better than variational methods [163, 63] especially in multi-modal loss landscapes [45]. Ensembles are also attractive because they require no additional hyperparameters and they are simple to implement. Although training costs increase linearly with the size of the ensemble, this can be easily parallelized on modern computing infrastructures. Furthermore, recent work that aims to model efficient ensembles that minimize computational cost could be an interesting future direction [69, 242]. NeuralLinear variants are also quite powerful and cheap, making them very promising for tasks without high-dimensional auxiliary information. Integrating Neural Linear with multi-output GPs is an interesting direction for future work. The other BNNs either require extensive hyper-parameter tuning or perform poorly, making them difficult to use in practice. Additional discussion can be found in Appendix C.5.5.

As seen in Appendix C.5.4, VAE-GP performs worse than our method on two of the chemistry objectives and better on one objective. While latent-space optimization methods are often applied to domains where one wants to simultaneously generate data and optimize over the data distribution, these methods can also be applied to the cases in this work, where a data pool (e.g. QM9 dataset for the chemistry task) or separate data generation process (e.g. level-set process for the photonic crystal task) is already available. In these cases, the VAE is not used as a generative model, but rather as a way to learn appropriate representations. While latent-space approaches are able to take advantage of well-developed and widely available optimization algorithms, they also require unsupervised pre-training on a sizeable dataset and a suitable autoencoder model with the necessary inductive biases. Such models are available in chemistry where there have been significant development, but are more limited in other domains such as photonics. On the other hand, our method is able to incorporate the data structure or domain knowledge in an end-to-end manner during training, although future work is needed to more carefully evaluate how much of an advantage this is and whether it depends on specific dataset or domain characteristics. For settings where we do not need a generative model, it would also be interesting to replace the autoencoder with a self-supervised model [71, 124] or semi-supervised model [103] to create a suitable latent space.

3.7 Conclusion

We have demonstrated global optimization on multiple tasks using a combination of deep learning and BO. In particular, we have shown how BNNs can be used as surrogate models in BO, which enables the scaling of BO to large datasets and provides the flexibility to incorporate a wide variety of constraints, data augmentation techniques, and inductive biases. We have demonstrated that integrating domain-knowledge on the structure and symmetries of the data into the surrogate model as well as exploiting intermediate or auxiliary information significantly improves BO performance, all of which can be interpreted as physics-informed priors. Intuitively, providing the BNN surrogate model with all available information allows the BNN to learn a more faithful physical model of the system of interest, thus enhancing the performance of BO. Finally, we have applied BO to real-world, high-dimensional scientific datasets, and our results show that BNNs can outperform our best-effort GPs, even with strong domain-dependent structure encoded in the covariance functions. We note that our method is not necessarily tied to any particular application domain, and can lower the barrier of entry for design and optimization.

Future work will investigate more complex BNN architectures with stronger inductive biases. For example, output constraints can be placed through unsupervised learning [97] or by variationally fitting a BNN prior [259]. Custom architectures have also been proposed for partial differential equations [182, 134], many-body systems [31], and generalized symmetries [81], which will enable effective BO on a wider range of tasks. The methods and experiments presented here enable BO to be effectively applied in a wider variety of settings. There are also variants of BO including

TuRBO which perform extremely well on our tasks, and so future work will also include incorporating BNNs into these variants.

We have made our datasets and code publicly available[1].

[1]`https://github.com/samuelkim314/DeepBO`

Chapter 4

Automated Discovery and Optimization of 3D Topological Photonic Crystals

4.1 Introduction

The past few decades have seen tremendous advances in optimization and inverse design techniques for nanophotonic components and, in particular, photonic crystals (PhCs) [85, 153, 115]. These advances have been spurred partly by the increase in computing capacity and methodological innovations, and partly by emerging micro- and nanofabrication capabilities that have significantly expanded the overall design space for device structures. A central and persisting effort has focused on the optimization of PhCs with large photonic bandgaps, with approaches including gradient-based [38, 29, 96], semi-definite programming (SDP) [146, 148], and gradient-free [57, 256] techniques. Systematic explorations through pre-defined templates have revealed insights into the connection between PhC structure and their associated bandgap [141, 21]. In more recent years, topological PhCs have emerged as a particularly exciting research direction, and with the more recent introduction of topological band theory to photonics [131, 164, 218], the PhC's topology has also emerged as a quantity of interest for design. Here, we revisit the question of optimizing photonic bandgaps in this new context, seeking the automated discovery and optimization of large, topologically nontrivial bandgaps and well-isolated, robust topological degeneracies.

Topological PhCs are attractive due to their promised robustness for photonic devices and the plethora of associated exotic optical phenomena [131, 164, 218]. In contrast to their electronic counterparts, topological PhCs offer the exciting prospect of a flexible and tunable design space, with structural morphology subject only to the limits of fabrication constraints. This design freedom has not only endowed photonics with the underlying technological promise of topological protection, but also positioned it as a promising platform for the fundamental exploration of spinless topological physics. Exemplifying this, the first experimental realization of a Chern insulator without Landau levels—the quantum anomalous Hall effect [65, 180]—was achieved in a gyromagnetic 2D PhC [238]. Similarly, the first experimental observation of a Weyl point was achieved in a 3D high-index PhC [132], simultaneously with its observation in TaAs [251, 138]. Gapless photonic topology has since grown to encompass a wide variety of phases, including unconventional Weyl points [94, 203, 265], Dirac points [234, 62, 233], and nodal lines [130, 255, 51, 248, 258, 168, 232, 169].

These advances in photonic topology have been achieved largely by using analogy, physical intuition, and symmetry requirements. Similarly, optimization of associated designs have depended mainly on parameter sweeps [249] and trial-and-error. Although recent studies have applied topology optimization to topological PhCs [27, 116, 160], these efforts have all relied on optimization of a continuous proxy objective to implicitly encourage topology-associated behavior (e.g., power flow [27], directionality of Purcell enhancement [160], or density of states [116]) in lieu of explicitly enforcing a discrete topological constraint. In addition to requiring a different formulation of this proxy for each topological effect, a significant downside of this approach is the frequent need for an initial candidate for optimization which is already topological [27, 160] to avoid designs that maximize the proxy objective but lack the intended band topology. There are two significant challenges to explicitly enforcing band topology. First, they are inherently discrete and discontinuous, rendering commonly-used optimization algorithms that assume differentiability nominally inapplicable. Second, the conventional evaluation of topological invariants from band holonomy [4] (i.e., parallel transport of wave functions, involving notions of band connections and curvatures) is computationally costly since they require simulating the full Brillouin zone (BZ) (e.g., 2D Chern number) or a dense subset of it (e.g., Berry phase or 3D Chern vector).

In this work, we propose to use a combination of gradient-free optimization algorithms, a symmetry-constrained level-set parameterization of the structure, and a computationally efficient symmetry-based evaluation of band topology to automate the discovery and optimization of novel topological PhCs. An overview of the optimization process' elements is shown in Fig. 4-1, the flow of which is summarized in Fig. 4-1(a). By using a level-set function expressed as a symmetry-constrained Fourier sum, we can parameterize even highly complex geometries using only a low-dimensional parameter space [Fig. 4-1(b)], which in turn makes both global and local gradient-free optimization algorithms feasible. To efficiently evaluate the band topological constraints, we incorporate the recently introduced frameworks of topological quantum chemistry [16] and symmetry indicators [176, 108], which have also been applied to PhCs [35, 26]. Furthermore, we use a global optimization algorithm in conjunction with a stochastic local optimization stage to escape local optima during optimization while also handling the non-continuous objective function stemming from the topology of the PhC. We focus on 3D topological PhCs and present novel examples in three settings: (i) Γ-enforced topological nodal lines, (ii) ideal (i.e., frequency-isolated) Weyl points in the interior of the BZ, and (iii) photonic Chern insulators with chiral surface states. In the last setting, we find a photonic Chern insulator with the largest known complete bandgap, and achieve this without explicitly relying on the supercell modulation technique that underlies earlier designs. While we focus on a handful of concrete topological properties, our method can in principle be applied to any symmetry-identifiable band topology.

For convenience, commonly used abbreviations and notation throughout this chapter are listed in Appendix D.

4.2 Methods

4.2.1 Photonic Crystal Parameterization

A key choice in the optimization of any system is the parameterization, i.e., the degrees of freedom. In the context of photonic crystals (PhCs), this entails a parameterization of the dielectric function $\varepsilon(\mathbf{r})$ across the unit cell. Historically, a widely employed approach has involved restricting the PhC structure to simple "basic" geometric shapes (e.g., circles, squares, spheres, or cubes of dielectric material) requiring very few geometric parameters (e.g., radius or width), which in turn enables optimization using gradient-free techniques or even plain parameter sweeps [88]. Such approaches have been surprisingly effective, achieving for example (trivial) complete three-dimensional bandgaps of nearly 35% with an index contrast of 4 in a diamond lattice of spherical holes [75]. At the opposite end of the spectrum, topology optimization (referring to the optimization of the geometric and topological shape of a *structure*; not to be confused with topology of *bands*) is an extremely flexible approach that divides the design region into a grid of pixels or voxels, each representing a continuous design parameter. The associated dimensionality of the design space can be very high, ranging from thousands to millions. To make this optimization tractable, gradient-based algorithms [38, 29] or subspace methods combined with semi-definite programming (SDP) [146, 148] have been used, enabling optimization of PhC bandgaps in both 2D and 3D.

To represent the permittivity $\varepsilon(\mathbf{r})$ over the coordinates $\mathbf{r}$ of the unit cell, we adopt a level-set parameterization. Level-set functions are significantly more flexible than combinations of simple shapes while making the problem more tractable than voxel-based discretizations for gradient-free optimization algorithms. Concretely, we introduce a level-set function $\phi(\mathbf{r})$ whose intersection with a level-set offset Δ determines

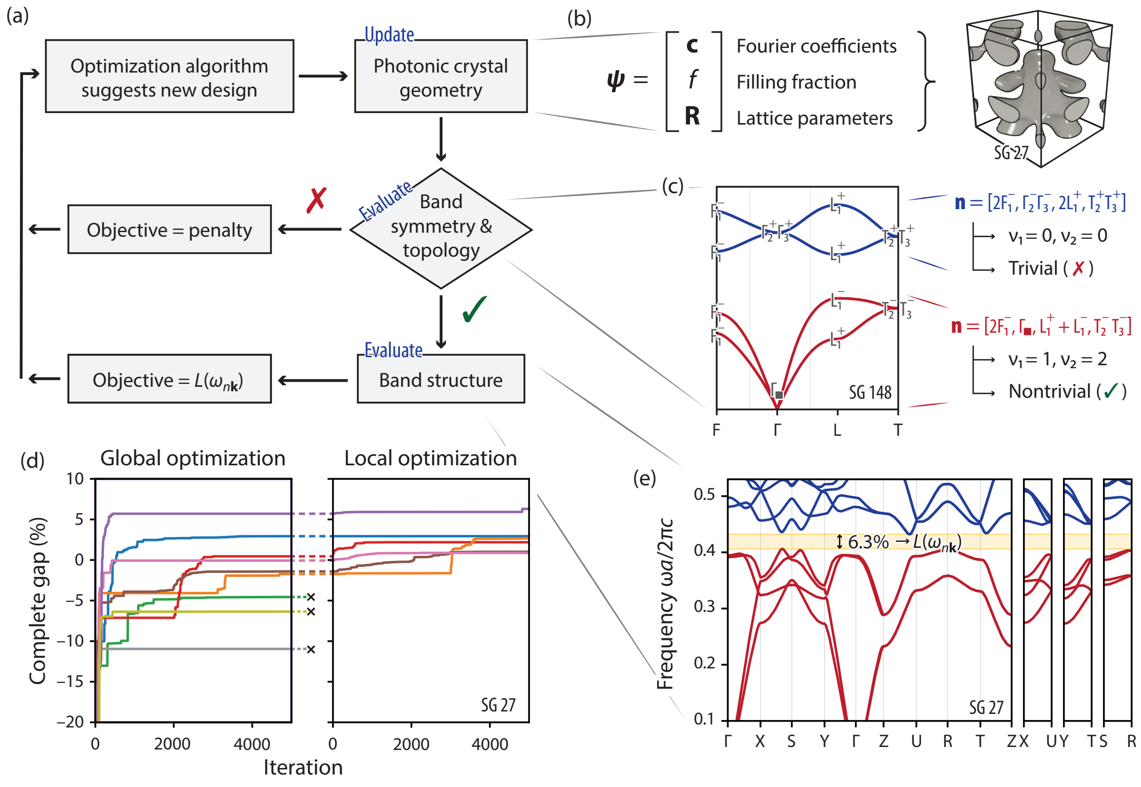


Figure 4-1: **Overview of topological photonic crystal (PhC) optimization.** (a) Flowchart of the optimization process in each iteration. Optimization continues until a convergence criterion or reaching a maximum number of iterations. (b) The structure of the PhC is parameterized by a continuous vector consisting of the geometry coefficients (the Fourier sum coefficients), the filling fraction, and the unit cell lattice parameters. (c) Symmetry-based tools can be used to calculate the band connectivity and topology from the high-symmetry (HS) **k**-points, providing a computationally efficient evaluation of the topology constraint (here, exemplified for a hypothetical PhC in space group (SG) 148). (e) If the PhC is found to have the desired topological indicator as computed from the band symmetries, then the bandstructure is calculated along the HS **k**-lines (shown in purple in the inset). This example shows a complete bandgap highlighted in yellow. (d) Optimization performance of complete bandgaps in SG 27 over multiple trials, as measured by the best value of the objective found so far as a function of iteration. Optimization consists of a 2-step process: global optimization with multiple trials, followed by local optimization using the best candidates from global optimization. Colors represent different trials from random initializations.

the boundary between regions of permittivity ε_1 and ε_2:

$$\varepsilon(\mathbf{r}) = \begin{cases} \varepsilon_2 & \operatorname{Re}\phi(\mathbf{r}) > \Delta \\ \varepsilon_1 & \operatorname{Re}\phi(\mathbf{r}) \leq \Delta \end{cases}, \tag{4.1}$$

Going forward, we will take $\varepsilon_1 = 1$, corresponding to vacuum or air, and denote ε_2 simply by ε. The advantages of a level-set function is that it allows a relatively low-dimensional parameterization while retaining a versatile geometric design space. The technique has been widely employed in photonic optimization, with level-set functions parameterized by spherical harmonics [150], Hamilton-Jacobi formulations [96], and eigenfunctions of a correlation function [256].

Here, exploiting the periodic nature of PhCs, we parameterize the level-set function $\phi(\mathbf{r})$ by a finite Fourier summation of plane waves with spatial frequencies at the reciprocal lattice vectors $\{\mathbf{G}\} = \{n_1\mathbf{G}_1 + n_2\mathbf{G}_2 + n_3\mathbf{G}_3 \mid n_i \in \mathbb{Z}\}$, i.e., by:

$$\phi(\mathbf{r}; \mathbf{c}_{\mathbf{G}}) = \sum_{\{\mathbf{G}\}} c_{\mathbf{G}} \mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}}, \tag{4.2}$$

with complex expansion coefficients $\mathbf{c} = \{c_{\mathbf{G}}\}$. The Fourier parameterization has a number of advantages: (i) it automatically incorporates lattice periodicity, (ii) imposing an upper cutoff $G_{\max}$ on the norm of included reciprocal lattice vectors translates to limiting the spatial variation of the permittivity profile (i.e., small feature sizes) and approximates feature-size constraints, and (iii) symmetry constraints can be straightforwardly incorporated which additionally translates to a reduction in the number of free expansion coefficients.

The third point—symmetry-constraints—is crucial in our context, given the close connection between symmetry and band topology that we aim to exploit. We briefly summarize how symmetry constraints can be imposed on the Fourier coefficients, following Ref. 26. Concretely, the permittivity $\varepsilon(\mathbf{r})$ must be invariant under every symmetry operation g in the space group (SG) $\mathcal{G}$, such that $g\varepsilon(\mathbf{r}) = \varepsilon(g^{-1}\mathbf{r}) = \varepsilon(\mathbf{r})$—or equivalently $\phi(g^{-1}\mathbf{r}) = \phi(\mathbf{r})$ for every $g \in \mathcal{G}$. This imposes the coefficient constraints $c_{\mathbf{G}}\mathrm{e}^{-\mathrm{i}g\mathbf{G}\cdot\mathbf{w}} = c_{g\mathbf{G}}$, with g expressed in Seitz notation $g = \{\mathbf{W}|\mathbf{w}\}$ with rotation part $\mathbf{W}$ and translation part $\mathbf{w}$. These constraints impose a set of interrelations among the Fourier coefficients, linking $c_{\mathbf{G}}$ of symmetry-related reciprocal lattice vectors $\mathrm{star}(\mathbf{G}) = \{g\mathbf{G} \mid g \in \mathcal{G}\}$, i.e., linking coefficients associated with distinct stars of $\mathbf{G}$. Overall, this reduces the unconstrained Fourier summation in Eq. (4.2) to a symmetry-constrained sum:

$$\phi(\mathbf{r}; \mathbf{c}_{\mathrm{star}(\mathbf{G})}) = \sum_{\{\mathrm{star}(\mathbf{G})\}} c_{\mathrm{star}(\mathbf{G})} \sum_{\mathbf{G}\in\mathrm{star}(\mathbf{G})} c_{\mathbf{G}}^{\mathrm{star}(\mathbf{G})} \mathrm{e}^{\mathrm{i}\mathbf{G}\cdot\mathbf{r}}. \tag{4.3}$$

Here, $c_{\mathbf{G}}^{\mathrm{star}(\mathbf{G})}$ denotes fixed, symmetry-determined coefficients among reciprocal lattice vectors from the same star while $c_{\mathrm{star}(\mathbf{G})}$ denotes the remaining free coefficients, each associated with a specific star. We only need to optimize the free parameters $\mathbf{c}_{\mathrm{star}(\mathbf{G})}$. Note that the number of parameters thus depends on the SG; roughly, more symmetries (larger group order) translate to fewer free parameters. Generally, these parameters are complex; in the presence of inversion symmetry, however, they can be restricted to be real parameters.

In practice, we limit the included stars of $\mathbf{G}$ to those that have a representative element with components $n_i \in \{0, \pm1, \pm2\}$. In addition, we parameterize the level-set boundary using the filling fraction f rather than the level-set offset Δ, since the mapping between Δ and f is monotonic and the filling fraction is more intuitive to interpret. This choice is also motivated by the empirical observation that a filling fraction parameterization leads to better optimization performance (likely because uniform sampling of Δ and $c_{\mathrm{star}(\mathbf{G})}$ produces structures with normally distributed filling fractions of small variance and mean close to 0.5, which is an undesirable bias). We bound the values of f and $c_{\mathrm{star}(\mathbf{G})}$ to $f \in [0, 1]$ and $c_{\mathrm{star}(\mathbf{G})} \in [-1, 1]$. Altogether, the symmetry-constrained level-set Fourier parameterization of Eq. (4.3) allows a great deal of shape design freedom, as illustrated e.g., in Fig. 4-1(b), while requiring only very few parameters (in the range 11–69 for the problems considered here).

We also include the unit cell's lattice vectors $\{\mathbf{R}_1, \mathbf{R}_2, \mathbf{R}_3\}$ as part of the overall structure specification, which can be parameterized (up to an immaterial overall rotation) by their lengths $\{a, b, c\}$ and mutual angles $\{\alpha = \angle(\mathbf{R}_2, \mathbf{R}_3), \beta = \angle(\mathbf{R}_3, \mathbf{R}_1), \gamma = \angle(\mathbf{R}_1, \mathbf{R}_2)\}$. We exploit the scale-invariance of the dispersion-free Maxwell's equations to eliminate one of these parameters, setting $a = 1$ without loss of generality. The remaining 5 parameters are typically constrained further by the choice of space group (SG): e.g., SGs in the cubic crystal system (195–230) have no free lattice parameters whereas SGs in the monoclinic crystal system (3–15) have 3 free parameters (b, c, and β). [8] As a practical matter, we restrict the free lattice vector lengths and angles to $b, c \in [0.75, 1.25]$ and $\alpha, \beta, \gamma \in [90°, 150°]$ during optimization. Intuitively, we expect that a nearly isotropic BZ will tend to host larger HS bandgaps; empirically, we observe that restricting b, c to near unity indeed improves optimization convergence.

Unless otherwise specified, we adopt a scalar, fixed permittivity of $\varepsilon = 16$, roughly representative e.g., of silicon at visible frequencies or certain ceramic-filled plastics at microwave frequencies [132]. The permittivity could in principle be included as an additional optimization parameter; in practice, however, bandgap optimization tends to automatically favor increased permittivity, rendering its optimization trivial.

4.2.2 Objective

In the topological settings explored in this work, we wish to maximize the bandgap along high-symmetry (HS) **k**-lines in the Brillouin zone (BZ). However, bandgap optimization is a notably difficult problem as the bandgap objective is not fully differentiable (especially at points of band crossing) and is a highly non-convex problem with numerous local optima [29, 146]. Earlier works were only able to optimize the bandgap if the structure used as an initial point for optimization already had at least an incomplete gap [38, 29]. To address this limitation, Cox and Dobson [29] combined gradients with evolutionary algorithms to better search the entire parameter space, while Men et al. [146] transformed the objective and constraints to improve the differentiability of the problem. However, these methods are still prone to getting trapped in local maxima. We note that using the SDP and subspace method developed by Men et al. [148] to maximize the bandgap in SG 13, we were not able to find a single example with a complete HS bandgap over dozens of trials. Furthermore, in optimizing topological PhCs we introduce additional constraints in the form of the topological indicator of the bands of interest. Topology is inherently a discrete quantity and so the constraints are non-continuous, making this problem difficult to address using existing approaches.

To address these challenges, we propose an optimization framework shown in Fig. 4-1(a) which uses gradient-free optimization algorithms. This optimization framework allows us to (1) simultaneously optimize the lattice parameters as we are interested in SGs that do not have cubic lattices, (2) easily search the global space which is highly non-convex and has numerous local maxima, (3) handle non-differentiable objectives, and (4) incorporate non-continuous constraints corresponding to the topological indicators of the bands.

In particular, we choose the objective function $L(\omega_{n\mathbf{k}})$—i.e., the quantity we wish to maximize—as the relative bandgap:

$$\Delta\omega_n = 2\frac{\min_{\mathbf{k}} \omega_{n+1}(\mathbf{k}) - \max_{\mathbf{k}} \omega_n(\mathbf{k})}{\min_{\mathbf{k}} \omega_{n+1}(\mathbf{k}) + \max_{\mathbf{k}} \omega_n(\mathbf{k})} \tag{4.4}$$

where $\omega_n(\mathbf{k})$ is the nth band from the bottom. The objective function $L(\omega_{n\mathbf{k}})$ is in general a function of $\mathbf{k}$ across the entire BZ. However, to make optimization of 3D PhCs computationally efficient, we restrict the domain of $\mathbf{k}$ to the high-symmetry (HS) $\mathbf{k}$-lines as shown in Fig. 4-1(e), discretized into approximately 300 equally spaced $\mathbf{k}$-points. The HS $\mathbf{k}$-lines typically lie along the edges of the BZ, and empirically the extrema of the bands typically lie along the HS $\mathbf{k}$-lines, allowing us to make this simplification [88, 148, 145].

For some applications, such as Γ-enforced topology, n is fixed. However, for other applications where we are agnostic to the particular band and simply wish to maximize *a* gap, we search over multiple bands by using the objective function:

$$\Delta\omega = \max_{n} \Delta\omega_n \tag{4.5}$$

where n can vary from 1 up to an upper limit we set based on computational constraints.

In the context of topological PhCs, we want to constrain bands 1 through n (where n may be fixed or variable) to have a particular topological index. To enforce the topology of the bands, we check the topological index in each iteration as shown in Fig. 4-1(c), as symmetry indicators allow us to quickly calculate the index from only the special HS **k**-points (the details of which are explained in the section on "PhC Simulation"). If the bands are indeed topological, we then calculate the entire band structure along the HS **k**-lines from which we set $L(\omega_{n\mathbf{k}}) = \Delta\omega_n$ or $L(\omega_{n\mathbf{k}}) = \Delta\omega$, as appropriate. Otherwise, we assign a penalty term of -2 to the objective, as this is the minimum possible value of the relative bandgap. Incorporating the constraint into the objective function allows us to use a variety of optimization algorithms that are not necessarily designed for nonlinear constraints. The overall process in each iteration is summarized in Fig. 4-1(a)

4.2.3 Optimization

As shown in Fig. 4-1(d), the optimization framework consists of 2 stages: a global optimization stage to search the entire parameter space, and a local optimization stage that takes the best candidates from the first stage and fine-tunes them within a small region in the parameter space.

The first stage of optimization uses a randomized and locally-biased variant of the DIviding RECTangles (DIRECT) algorithm (which we will refer to as DIRECT-L-RAND) as implemented by NLopt [91, 48]. DIRECT is a gradient-free global optimization algorithm that systematically divides the parameter space into smaller and smaller hyperrectangles, and is thus able to handle highly non-convex functions. The locally-biased variant of DIRECT focuses more on local search rather than global search, and the randomized variant uses some randomization in how it decides which dimension to split the hyperrectangle. The randomized variant allows us to run the algorithm over multiple trials to take advantage of high-performance computing cluster resources. We also use a random initial point to further introduce diversity

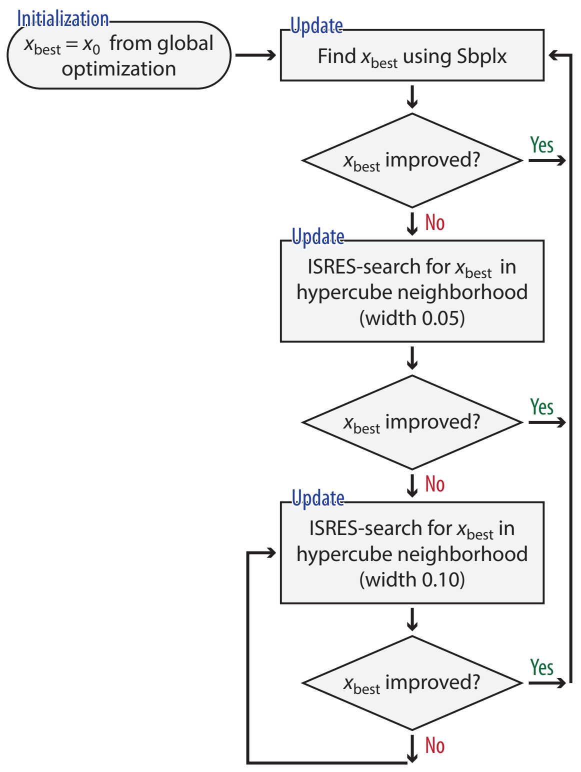

Initialization
$x_{best} = x_0$ from global optimization
Update
Find x_{best} using Sbplx
x_{best} improved?
Yes
No
Update
ISRES-search for x_{best} in hypercube neighborhood (width 0.05)
x_{best} improved?
Yes
No
Update
ISRES-search for x_{best} in hypercube neighborhood (width 0.10)
x_{best} improved?
Yes
No

Figure 4-2: **Flowchart of the local optimization algorithm.** Due to the non-differentiable objective function, Sbplx is subject to premature convergence. To ameliorate this, we periodically switch to ISRES to escape local maxima and continue optimization. Each rectangular node represents a sub-stage of 500 iterations, and the framework evaluates whether the objective has improved in that sub-stage to determine the algorithm for the next sub-stage.

over different trials. In principle, the non-randomized variants, DIRECT or DIRECT-L, can be used in place of DIRECT-L-RAND, although they are insensitive to the initial point and different trials will generally converge to the same solution.

As shown in Fig. 4-3(a), the global optimization stage not only is able to quickly find candidates with the desired topological index, but also produces a variety of possible candidates from which we can start the local search. The PhC geometries as well as their band structures are quite diverse, demonstrating the effectiveness of stochastic global exploration.

The second stage is a local optimization to refine the best candidates found from the first stage. We primarily use the Sbplx algorithm, a variant of Nelder-Mead, as implemented by the NLopt library [91, 188]. Nelder-Mead is a popular gradient-free local optimization algorithm that builds an $d+1$-dimensional simplex (where d is the dimension of the problem) to approximate the objective function and find a new point (which is subsequently used to adjust the simplex). Sbplx (a re-implementation of the Subplex algorithm) decomposes the problem into low-dimensional subspaces so that the simplex method can search more efficiently and robustly. A challenge with such methods is that they are not guaranteed to converge for non-convex or discontinuous functions (although in practice they seem to work reasonably well [188]). Indeed, the non-differentiability stemming from the discrete nature of topology combined with the non-convexity of our problem leads to frequent failure modes in the optimization algorithm, including getting stuck in non-feasible regions of the parameter space, decreasing the step size to the point where it no longer improves the objective, or even diverging and jumping to a point far away from the initial point. Thus, to avoid the local optimization prematurely converging, we periodically restart the search algorithm starting from the best point found so far to reset the trust region and step size. We also occasionally switch to Improved Stochastic Ranking Evolution Strategy (ISRES) [194, 91] to escape local maxima. ISRES is a stochastic evolutionary-based global optimization algorithm that uses a combination of a mutation rule and differential variation, and can incorporate nonlinear constraints (although we do not explicitly incorporate constraints in our problem). The bounds of ISRES are set to be a small hypercube around the best point found so far so that it behaves like a stochastic local optimization.

More specifically, the local optimization stage is broken up into sub-stages of 500 iterations each. The first sub-stage uses the Sbplx algorithm. If a sub-stage using Sbplx does not improve the objective, then the next sub-stage switches to ISRES. If a sub-stage using ISRES does improve the objective, the next sub-stage switches back to Sbplx. A detailed flowchart of the local optimization process is shown in Fig. 4-2.

As seen in Fig. 4-3(b), the local optimization stage is effective in quickly and significantly increasing the bandgap. The vast difference in performance between global and local optimization stages indicates the difficulty of the bandgap problem and suggests that the set of feasible parameters that support complete HS bandgaps is small relative to the parameter space. The diversity of the results from global optimization and the fact that the different trials do not converge to the same structure during local optimization also demonstrates that the bandgap problem likely has many local optima, many of which do not have complete HS bandgaps. Thus, multiple trials with different random initializations are necessary to find the globally optimal structure.

Our framework is similar to multi-start local optimization algorithms which apply local optimization algorithms multiple times from different points to find the global maxima in non-convex problems. The difference is that we apply multiple trials of stochastic global optimization to pick the starting points for local optimization rather than randomly sampling the starting points. This is necessary due to (1) the extremely large number of local maxima in the parameter space and (2) the topology constraints. Only a small fraction of randomly sampled points obey the desired topology constraints (typically $< 10\%$ and often $< 1\%$), so the global search is necessary to find valid starting points with the desired topology. Thus, unlike prior work that optimizes topological PhCs from a known topological structure [249, 160], our method is able to automatically discover new topological PhCs.

4.2.4 PhC Simulation

We use the MIT Photonics Bands (MPB) software to solve for the eigenmodes of the PhC unit cell and calculate the band structure and associated symmetry eigenvalues [92]. Specifically, we use a $16 \times 16 \times 16$ spatial resolution during optimization and a $32 \times 32 \times 32$ resolution when calculating the full BZ. For the symmetry-based evaluation of band topology, we use the Julia packages Crystalline.jl [26], which implements the methodology of topological quantum chemistry [16] and symmetry indicators [108, 174]; and MPBUtils.jl, which implements additional MPB-specific functionality [2]. The choice of HS $\mathbf{k}$-paths is not unique for a particular SG; we use the SeeK paths as defined by Hinuma et al. [73] and as implemented in Brillouin.jl [1].

We briefly summarize the salient ideas of the symmetry-based evaluation of band topology (see, e.g., Refs. 20 and 176 for existing reviews of the general methodology and Ref. 26 for its adaptation to PhCs). Given a selection of bands $\{n\}$, we first

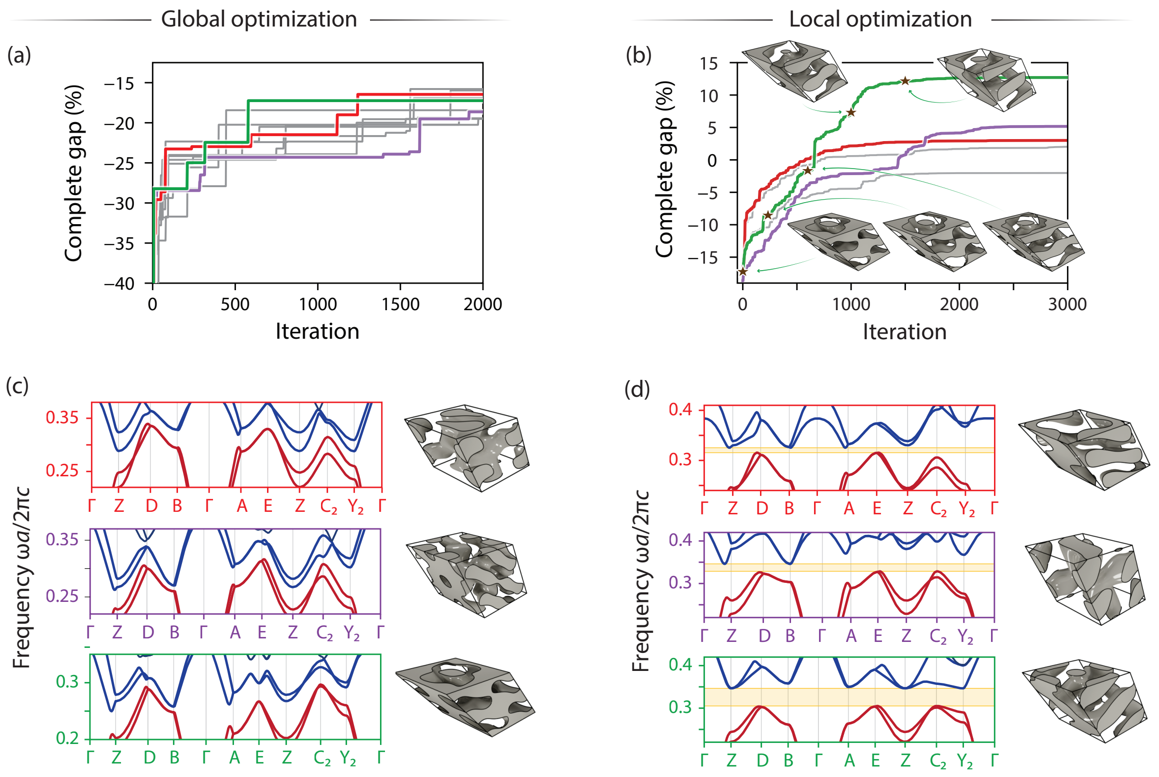


Figure 4-3: **Global and local optimization of PhCs in SG 13 with Γ-enforced topology.** (a–b) Evolution of the the complete HS gap, i.e., the optimization objective, during the (a) global and (b) local optimization stages. Different curves correspond to different trials from random initializations. Of the 10 trials shown in (a), the 5 best trials are used as initialization for the local optimization stage. The three trials with best eventual local optimization result are color-highlighted (red, purple, and green). The unit cell evolution during local optimization for the green trial is shown at select iterations (stars). A detailed movie of the unit cell evolution is included in the Supporting Information. (c–d) Band structures and unit cells for the color-highlighted (red, purple, and green) optimization curves at the final iteration of global (c) and local (d) optimization.

compute the band symmetry eigenvalues $\langle \mathbf{E}_{n\mathbf{k}} | g \mathbf{D}_{n\mathbf{k}} \rangle$ at the special **k**-points of the BZ, using the Bloch eigenmodes of the **E**- and **D**-field, over the symmetry operations g of each **k**-point's little group (i.e., the operations that leaves the **k**-point invariant, modulo reciprocal lattice vectors). From the set of symmetry eigenvalues, we next compute the combination of irreducible representations (irreps) that the bands transform as at each **k**-point. The overall information is aggregated into an integer-valued symmetry vector:

$$\mathbf{n} = [n_{\mathbf{k}_1}^1, n_{\mathbf{k}_1}^2, \ldots, n_{\mathbf{k}_N}^1, n_{\mathbf{k}_N}^2, \ldots, \mu], \tag{4.6}$$

where $n_{\mathbf{k}_i}^\alpha$ denotes the number of bands in the selection that transform as the αth irrep $D_{\mathbf{k}_i}^\alpha$ of the little group at $\mathbf{k}_i$, and μ denotes the total number of bands in the selection.

For a given PhC, we build up a set of symmetry vectors iteratively, starting from the lowest bands. For each candidate symmetry vector, we test its consistency against the set of compatibility relations imposed by the SG—i.e., constraints on how band symmetries can connect across the Brillouin zone [15, 98]—and if incompatible, additional bands are included until a compatible band grouping is found. Each such compatible symmetry vector signals a set of bands which are (pointwise) gapped from all other bands along all HS **k**-lines. The associated assignment of topology to each such symmetry vector can be achieved using the formalism of symmetry indicators [176, 108] and topological quantum chemistry [16]. In brief, for each SG, a choice of a set of trivial generators $\{\mathbf{a}_i\}$ and a set of nontrivial generators $\{\mathbf{b}_j\}_{j=1}^{d^{\mathrm{BS}}}$ exists such that any compatible vector can be expressed as their integer combination according to [212]:

$$\mathbf{n} = \sum_i c_i \mathbf{a}_i + \sum_j \nu_j \mathbf{b}_j, \tag{4.7}$$

where $c_i \in \mathbb{Z}$ are integer coefficients and $\nu_j \in \mathbb{Z}_{\lambda_j}$ are coefficients from the ring of integers modulo λ_j, i.e., from $\mathbb{Z}_{\lambda_j} = \{0, 1, \ldots, \lambda_j - 1\}$. Here, λ_j are the minimal integers that allow an integer-coefficient expansion of $\lambda_j \mathbf{b}_j$ in $\{\mathbf{a}_i\}$ (i.e., $\lambda_j \mathbf{b}_j$ is trivial from the perspective of symmetry). The symmetry indicator, which characterizes the symmetry-identifiable topology, is precisely the expansion coefficients ν_i: that is, we associate with every $\mathbf{n}$ the symmetry indicator $\boldsymbol{\nu} = (\nu_1, \ldots, \nu_{d^{\mathrm{BS}}})$ which is an element of the indicator group $\mathbb{Z}_{\lambda_1} \times \ldots \times \mathbb{Z}_{d^{\mathrm{BS}}}$. Nontrivial band topology corresponds to nonzero $\boldsymbol{\nu}$ and symmetry-identifiable topology consequently exists only in SGs with indicator group different from $\mathbb{Z}_1$. In practice, we compute the compatibility relation tests as well as the symmetry indicators using a procedure described elsewhere [40, 26]. A sketch of the application of the framework to a hypothetical PhC in SG 148 (R$\overline{3}$) with indicator group $\mathbb{Z}_2 \times \mathbb{Z}_4$ is shown in Fig. 4-1(c).

We emphasize that the computational efficiency of this symmetry-based approach is crucial to our optimization technique. With it, we can evaluate both the band connectivity and topology using just a handful of special $\mathbf{k}$-points (6–8 for the SGs considered here); without it, this would require simulating a densely discretized subset of the BZ, rendering the evaluation of the penalty term computationally prohibitive.

4.3 Results and discussion

This section presents a number of optimized PhCs with nontrivial topology, including Γ-enforced nodal lines, Weyl points, and Chern insulating gaps.

4.3.1 Γ-enforced topological nodal lines

We start by considering the optimization of PhCs exhibiting so-called Γ-enforced topological nodal lines, a recently introduced type of photonic band topology that can be inferred solely from the number of bands below the first gap along the high-symmetry (HS) $\mathbf{k}$-lines of the BZ [26]. In the presence of time-reversal symmetry, this band topology can exist in six SGs (13, 48, 49, 50, 68, and 86) and arises as a direct result of a uniquely photonic zero-frequency polarization singularity at the Γ-point. It is associated with an apparent bandgap between bands 2 and 3 along HS lines of the BZ which, however, vanishes along nodal lines in the interior of the BZ. Conceptually, it is analogous to filling-enforced topology [175], i.e., topology that can be inferred from band connectivity alone. Here, we focus on optimizing the gap along the HS $\mathbf{k}$-lines (which we denote as the "HS bandgap") with the aim of obtaining

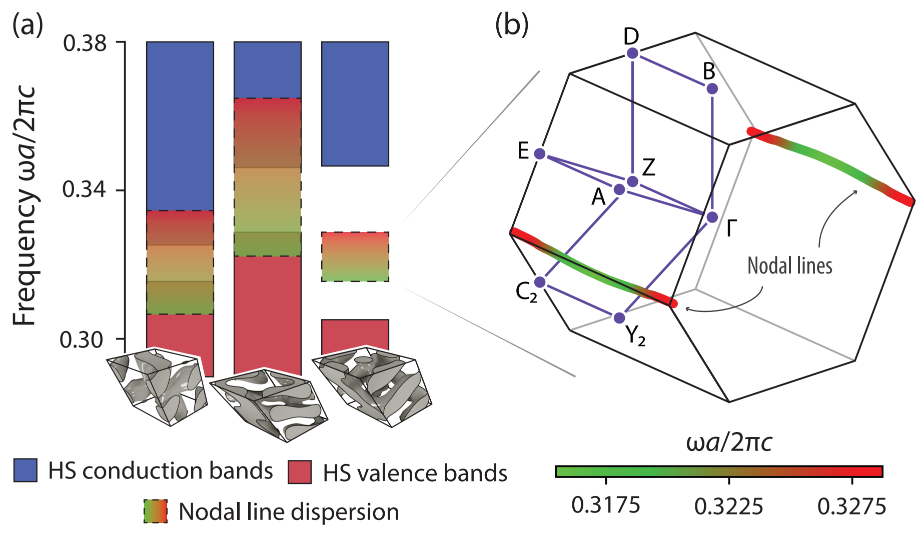


Figure 4-4: **Nodal line location and dispersion in optimized candidates of SG 13.** (a) Nodal line frequency dispersion compared to the HS bands. The right-most candidate exhibits a maximally-isolated nodal line. (b) The nodal lines (color-coded by frequency) in the BZ (outlined in black) run along the k_y direction. Purple lines represent the HS **k**-lines over which the bandgap is maximized

"ideal" frequency-isolated nodal lines. We explore each of the six SGs that support Γ-enforced nodal lines. In all cases, Γ-enforced topology occurs only between bands 2 and 3, so we use the objective in Eq. (4.4) with $n = 2$.

Running our optimization algorithm for each of these six SGs, across 10–20 global trials and 5000 subsequent local iterations for the top 3–5 global candidates, we find Γ-enforced topological designs in every SG. However, only in SG 13 (P2/c; generated by inversion and a glide operation $\{2_{010}|0, 0, \frac{1}{2}\}$ in the monoclinic crystal system) are we able to find full bandgaps across the HS **k**-path. Notably, SG 13 is a (maximal) subgroup of the remaining SGs; the inability to obtain HS bandgaps in these supergroups suggests that their additional symmetry constraints act counter to gap formation. Additionally, as shown in Fig. 4-3, the optimization framework is able to find full HS bandgaps in nearly half of the trials.

The best candidate we have found in SG 13 is presented in the bottom-most panel of Fig. 4-3(d), which contains a 12.7% HS gap between bands 2 and 3. Snapshots of the evolution of the structure is shown in Fig. 4-3(b), and a more detailed movie is included in the Supporting Information. We can see that the structure converges to what resembles a double gyroid structure, a well-known design for complete bandgaps [130]. The symmetry indicator of the bottom 2 bands is $(\nu_1, \nu_2) = (1, 1)$, indicating the presence of a pair of nodal lines, each protected by a π-Berry phase [212]. Performing a full BZ computation over a dense **k**-grid, we see a pair of nodal lines running along the k_y axis in Fig. 4-4(b), consistent with the predictions by Song et al. [212]. Unsurprisingly, the nodal lines are far away in the BZ from the HS **k**-lines along which we maximized the bandgap, since the band crossings that form nodal lines are antithetical to the optimization objective. In other words, optimizing the HS bandgap pushes the nodal line away from the HS **k**-lines.

The nodal line is frequency dispersive, exhibiting a relative 4.2% variation across its span. While there exist isofrequency contours around the nodal line at the nodal line frequencies due to its dispersion, there do not exist any nodal line frequency pockets elsewhere in the BZ, making the nodal line well isolated in frequency. Ideally, we would like to also minimize the frequency dispersion of the nodal line [255], as it is necessary to realize the characteristic linear DOS dispersion of nodal lines. However, this would require formulating an objective that is a function of the entire BZ rather than just the HS **k**-lines, which is computationally prohibitive; we leave such efforts to future work.

Because Γ-enforced topology is diagnosable from the connectivity alone (i.e., if the appropriate bands are connected, they must be topological), optimization methods such as the SDP solver [148] that were not formulated specifically for topological PhCs can be applied for this setting without requiring additional constraints. However, starting from random initializations, the SDP solver was not able to find any complete HS gaps in SGs 13, 48, and 68 over multiple trials and over a range of lattice bases. The difficulty in finding bandgaps points towards the nonconvexity of the problem and the necessity of global optimization methods.

Optimization under variable permittivity To study the minimum index required to support an ideal nodal line, we take the best candidate (which has a 12.7% HS gap) and calculate the band structure as a function of index for the optimized

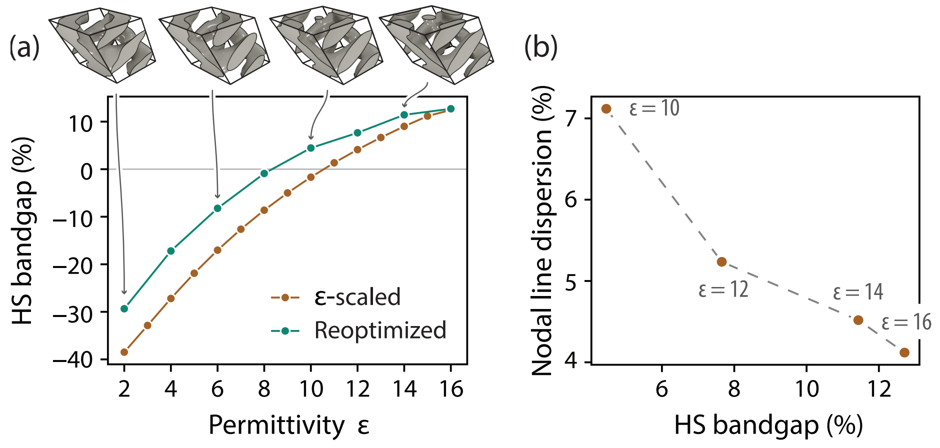


Figure 4-5: **Permittivity dependence of nodal line properties.** (a) We compare the HS gap attainable by simply scaling the permittivity of an original design (brown lines) versus the HS gap attainable by re-optimizing the structure at the new target permittivity (green lines), using the original structure as initialization point. The starting design is that associated with Fig. 4-4(b), originally optimized at $\varepsilon = 16$. Substantial improvements are attainable by re-optimizing. (b) Nodal line frequency dispersion as a function of the HS bandgap for the re-optimized structures, showing a monotonic decrease of nodal line dispersion with HS gap size.

structure in Fig. 4-5. As expected, the HS bandgap increases monotonically with the index contrast. The same structure has a HS bandgap of 0 at a relative permittivity of $\varepsilon \approx 10.6$. Alternatively, if we take the same structure and re-optimize it using our local optimization algorithm in Fig. 4-2, we can further improve the bandgap at a fixed permittivity, decreasing the minimum permittivity required for a complete HS bandgap to $\varepsilon \approx 8.3$. Looking at the nodal line for the optimized structures at different permittivities in Fig. 4-5(b), the nodal line frequency dispersion scales inversely with the attainable HS bandgap. In other words, the nodal line spans a greater fraction of the bandgap in frequency as permittivity decreases, remaining isolated inside the bandgap down to $\varepsilon = 12$.

Many previous proposals for PhCs containing nodal lines either require metallic materials or very high index-contrast dielectrics [130, 168, 169]. Our work paves the way for demonstrating nodal lines in PhCs with lower index, potentially enabling experimental realization in a wider range of frequencies.

4.3.2 Weyl points

Next, we pursue designs of PhCs with frequency-isolated Weyl points, also known as ideal Weyl points. Weyl points can arise in three different regions of the BZ: (i) at HS **k**-points, stabilized by spatial or time-reversal symmetries [143, 261], (ii) along HS **k**-lines, corresponding to the intersection of bands transforming as different irreps [240, 266], and (iii) at generic **k**-points in the interior of BZ, stabilized by a nontrivial symmetry indicator [212]. To date, proposed designs of frequency-isolated ideal photonic Weyl points fall in the first two classes. Here, we seek designs in the latter class, i.e., Weyl points at generic momenta. As before, we use the bandgap metric along HS **k**-lines as a proxy for isolating the Weyl points from other bulk bands. Additionally, because we are agnostic to the number of bands below the gap, we allow the objective $L(\omega_{n\mathbf{k}})$ to look over multiple bands, as in Eq. (4.5).

Space group constraints Song et al. [212] showed that for spinless, time-reversal invariant particles in non-centrosymmetric crystals (i.e., lacking inversion centers), all symmetry-indicated nontrivial band topology is associated with Weyl points in the interior of the BZ. We therefore initially restrict our attention to the 12 non-centrosymmetric SGs with nontrivial indicator group. However, only a subset of these groups are suitable for finding well-isolated Weyl points since the Nielsen–Nimomiya theorem requires Weyl points to occur in pairs of opposite chirality, which in general do not need to have the same frequency [230]. However, the existence of additional symmetries, e.g., mirrors or rotations, can map opposite-chirality Weyl points onto each other, thereby pinning them to the same frequency and allowing ideal single-frequency Weyl points. Consulting the results of Ref. 212 for each SG individually, we find that only six SGs (27, 37, 81, 82, 103, and 184) can host ideal, symmetry-indicated Weyl points. We perform global and local optimizations for PhCs in each of these SGs, and find designs with HS bandgaps in SGs 27 (Pcc2), 37 (Ccc2), 81 (P$\overline{4}$), and 82 (I$\overline{4}$). SGs 27 and 37 are centering-variants, both generated by a 2-fold rotation and an orthogonal glide plane; similarly, 81 and 82 are centering-variants, both generated by 4-fold rotoinversion.

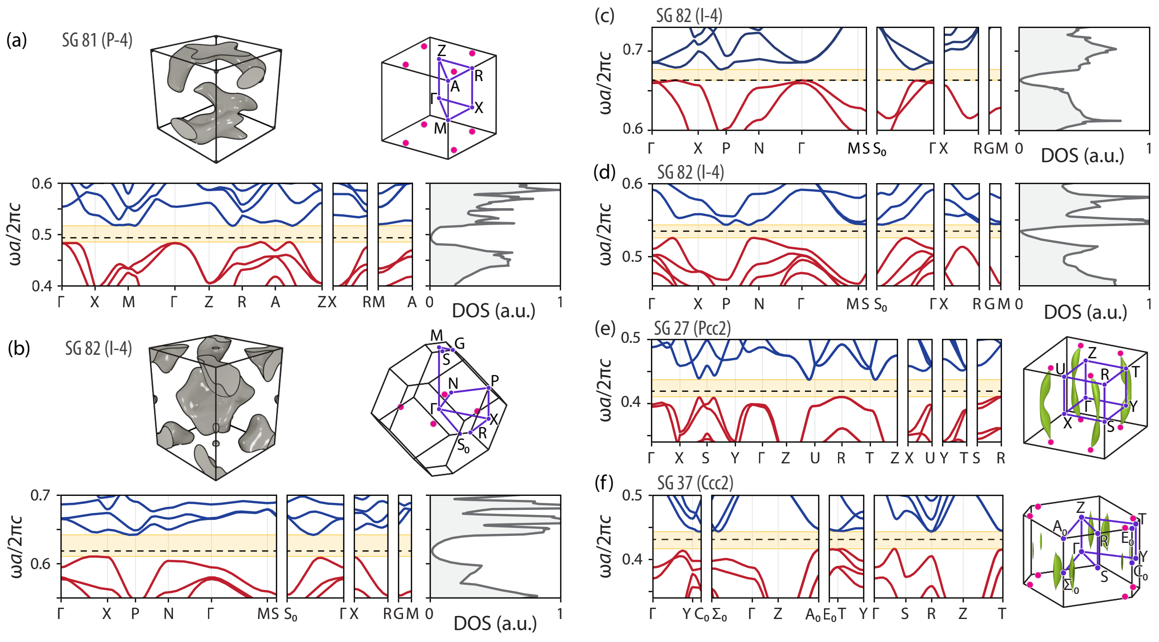


Figure 4-6: **Weyl point optimization in noncentrosymmetric SGs.** Selected results for optimized PhCs designs with Weyl points in SGs 27, 37, 81, and 82, visualized by their dispersions along HS **k**-lines (red lines, nontrivial valence bands; blue lines, conduction bands; yellow shading, HS gap; dashed black line, Weyl point frequency), density of states (DOS), and Weyl point locations in the BZ. (a–b) Optimized PhC Weyl point designs in SGs 81 and 82: the unit cells of each design are shown (for SG 82, we show the conventional unit cell, not the primitive). The BZs feature 4 ideal Weyl points in the $k_z = \pi/c$ plane (with equivalent copies in the $k_z = -\pi/c$ plane) in SG 81 and in the $k_z = 0$ plane in SG 82, without intersecting Fermi pockets. The density of states (DOS) exhibits a parabolic frequency dependence at the Weyl point frequency. (c,d) Optimized results in SG 82 with HS bandgaps, but non-ideal Weyl points. (c) The Weyl point frequency lies on the valence HS band edge, preventing frequency-isolation. (d) Although the Weyl point frequency lies in the center of the HS bandgap, the DOS is asymmetric and non-parabolic, due to conduction band minima right above the Weyl point frequency that do not intersect the considered HS **k**-lines. (e-f) Optimized PhC designs in SGs 27 and 37. In both cases, the Weyl points overlap spectrally with large Fermi pockets that extend over the interior of the BZ without intersecting any HS **k**-line.

Ideal Weyl point designs We show results for a range of optimized designs from these SGs in Fig. 4-6. In SGs 81 and 82 we find several designs with ideal Weyl points and without any intersecting trivial bulk bands, i.e., without any "Fermi pockets". Figure 4-6(a) highlights one such design from SG 81. The band structure has a 6.3% HS gap between bands 4 and 5, with Weyl points closing the gap in the interior of the BZ. The ideality of the Weyl points is also clearly exhibited in the density of states (DOS), calculated using the tetrahedron method implementation from Ref. 118, which displays the characteristic parabolic dependence $\text{DOS}(\omega) \propto (\omega - \omega_0)^2$ associated with ideal Weyl points at frequency ω_0. To determine the Weyl point locations, we perform a full BZ dispersion calculation and identify the Weyl points with the **k**-points where the conduction–valence frequency difference vanishes. We observe 4 Weyl points in the BZ at $k_z = \pi/c$ (8 are shown; note, however, that the $k_z = \pm\pi/c$ planes are equivalent), consistent with expectations for a symmetry indicator $(\nu_1, \nu_2) = (0, 1)$ [212]. Similar to the nodal line optimization, the Weyl points are pushed away from the HS **k**-lines in the $k_z = \pi$ plane, since the Weyl points represent a band-closing point which, if near a HS **k**-line, would act counter to the HS gap metric.

We have also found an example of a well-isolated, ideal Weyl point in SG 82, as shown in Fig. 4-6(b). The bandstructure displays a gap of 5.0% between bands 6 and 7, and the DOS drops to 0 at the Weyl point frequency as expected. In this SG, the Weyl points are fixed to the $k_z = 0$ plane in the BZ. Again, the Weyl points are pushed away from the HS **k**-lines by the optimization process.

Limitations of the high-symmetry bandgap metric While our optimization procedure has discovered several designs for ideal Weyl points, the HS bandgap metric is not infallible. Concretely, even when large complete HS bandgaps are achieved, the corresponding Weyl points may not necessarily be truly frequency-isolated. We identify two related failure modes. First, the success of the HS bandgap metric hinges on an assumption of minimally varying dispersions between HS lines, i.e., on the absence of non-Weyl-related local optima of the dispersion at generic **k**-points.

Although this is the tacit assumption underlying all the visualizations of the band structure along HS lines, it is well-known that this assumption can be violated [68, 32, 145]. Second, and relatedly, the frequency-location of the Weyl points relative to the HS bandgap is not pinned; specifically, although the *existence* of Weyl points is guaranteed by topology, their frequency *location* is not guaranteed to fall near the center of the HS bandgap.

Figure 4-6(c-f) summarizes a few examples of these failure modes, where despite obtaining a large HS gap, the optimization procedure does not yield truly ideal Weyl points. Figure 4-6(c,d) shows the dispersions and DOS for two designs in SG 82 with bandgaps of 2.0% and 3.3%, respectively. In Fig. 4-6(c) the Weyl point lies on the edge of the HS gap, illustrating the second failure mode mentioned above. Despite this, the DOS still exhibits an approximately parabolic dependence near the Weyl point and vanishes exactly at the Weyl point frequency. Even when the Weyl point occurs near the center of the HS gap, it may still be non-ideal, as illustrated by the design candidate considered in Fig. 4-6(d): the DOS is both asymmetrical and distorted from the expected parabolic shape due to the conduction band dropping close to, but not intersecting with, the Weyl point frequency elsewhere in the BZ, in illustration of the first failure mode mentioned earlier.

A more pronounced type of this failure mode, i.e., spectral intersection with Fermi pockets despite an otherwise “centered” Weyl point, was widespread in the optimized designs found in SG 27 and 37. Exemplifying this, Fig. 4-6(e) shows an optimized PhC in SG 27 with a HS gap of 6.3%, and Fig. 4-6(f) shows a PhC in SG 37 with a HS gap of 6.2%. In both settings, the Weyl points are pinned to the $k_z = \pi$ plane (and, equivalently, $k_z = -\pi$ plane), and the Weyl point frequency lies in the middle of the HS gap. Despite this, neither case exhibits ideal Weyl points; instead, the Weyl points are occluded by Fermi pockets that reside in the interior of the BZ, i.e., there exist local optima of the bands inside the BZ. We visualize these Fermi pockets in Fig. 4-6(e,f) by plotting the isofrequency contours at the Weyl point frequency ω_0.

Overall, the HS bandgap metric serves as a meaningful and largely effective proxy for optimizing Weyl points, as evidenced by our findings of ideal Weyl point designs

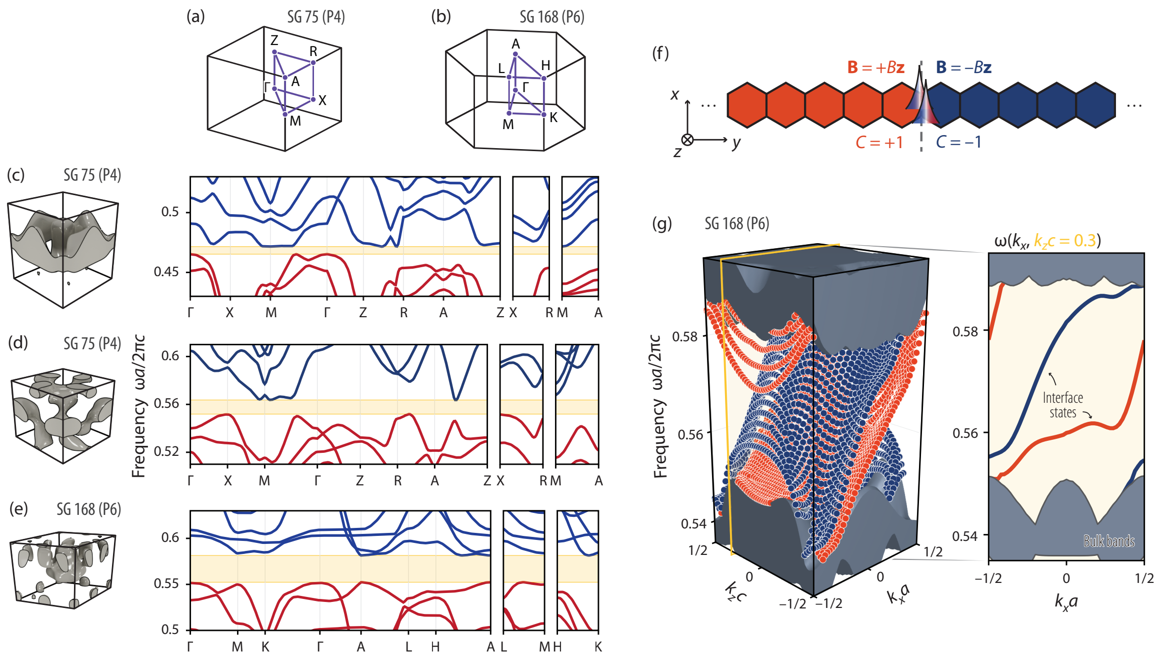


Figure 4-7: **Optimization of photonic Chern insulators.** BZs and HS **k**-lines for (a) SG 75 and (b) SG 168, respectively. PhC structure and band structure (complete gap highlighted in yellow) for (c) SG 75, $C = 1$, (d) SG 75, $C = 2$, and (e) SG 168, $C = 1$. (f) Supercell calculation for the SG 168 candidate in (e) to calculate surface states. The supercell consists of $1 \times 16 \times 1$ unit cells. At the interface, the sign of the applied applied magnetic field $\mathbf{B}$ is flipped: i.e., the orange units cells have $\mathbf{B} = +B\hat{\mathbf{z}}$ ($C = +1$) and the blue unit cells have $\mathbf{B} = -B\hat{\mathbf{z}}$ ($C = -1$). (g) Surface state dispersion at the interface, extracted from supercell calculation (projected bulk states indicated in gray).

in SGs 81 and 82. Overcoming the limitations of the HS gap metric is in principle possible, e.g., by replacing it by a full BZ gap metric (at considerable computational cost) or by supplementing the HS paths with additional paths, either from the outset or adaptively. The DOS, which can be computed efficiently using the tetrahedron or Gilat–Raubenheimer methods [118], could potentially also serve as an effective proxy.

4.3.3 Chern insulators

As a final showcase, we pursue designs of time-reversal broken 3D PhCs with nonzero Chern numbers, i.e., photonic 3D Chern insulators. There have been numerous proposals [180, 237, 7, 208, 122] and associated experiments [238, 209, 177, 260, 178] of photonic 2D Chern insulators. The 3D generalization, however, has only recently seen more development with only three distinct designs known [133, 37, 119]. Two of these designs require supercell modulations [133, 37], which are undesirable from a fabrication perspective, while the third requires non-dielectric (metallic) materials [119]. In all cases, the proposed designs feature relatively small ($\lesssim 2\%$) bandgaps. 3D Chern insulators are characterized by a Chern vector $\mathbf{C} = (C_1, C_2, C_3)$ rather than a single Chern number, with components C_i giving the Chern number of a $\mathbf{k}$-slice orthogonal to $\mathbf{G}_i$ [230]. Since the associated bands are assumed to be fully gapped, the slice-Chern numbers C_i are invariant with respect to the slice position k_i.

Nontrivial Chern phases require breaking time-reversal symmetry. To achieve this in the PhC context, we consider a gyroelectric material under a $\hat{\mathbf{z}}$-oriented external magnetic field with a permittivity tensor of the form [130]:

$$\boldsymbol{\varepsilon}(g) = \begin{bmatrix} \varepsilon_\perp & \mathrm{i}g & 0 \\ -\mathrm{i}g & \varepsilon_\perp & 0 \\ 0 & 0 & \varepsilon_\parallel \end{bmatrix}, \tag{4.8}$$

where $\varepsilon_\perp = (\varepsilon_\parallel^2 + g^2)^{1/2}$. We take $\varepsilon_\parallel = 16$ and $g = 12$ corresponding to a dimensionless effective magnetic field intensity of $|\mathbf{B}| = g/\varepsilon_\parallel = 0.75$. Note that the determinant of $\boldsymbol{\varepsilon}(g)$ is independent of g and $|\mathbf{B}|$, ensuring that the overall band structure is not shifted as a whole by the applied magnetic field [130].

Although the slice-Chern numbers can be computed by discretization [47], the computational cost is prohibitive for optimization purposes. Instead, we focus our attention to PhCs in SGs 75 (P4) and 168 (P6), where the slice-Chern number C_z can be computed from the bands' n-fold rotation-eigenvalues modulo n [43], or equivalently, from the symmetry indicators $\nu_1^{(75)} \in \mathbb{Z}_4$ and $\nu_1^{(168)} \in \mathbb{Z}_6$. Furthermore, under suitable convention choices, the symmetry indicators map directly to C_z, i.e., $C_z^{(75)} = \nu_1^{(75)} \pmod 4$ and $C_z^{(168)} = \nu_1^{(168)} \pmod 6$.[1] The remaining components, i.e.,

[1]This mapping involves a fixed convention for the choice of nontrivial generators in Eq. (4.7) [212]. Specifically, we take $\mathbf{b}_1^{(75)} = -\hat{\mathbf{e}}_{\Gamma_2} + \hat{\mathbf{e}}_{\Gamma_4} - \hat{\mathbf{e}}_{Z_2} + \hat{\mathbf{e}}_{Z_4}$ and $\mathbf{b}_1^{(168)} = +\hat{\mathbf{e}}_{H_2} - \hat{\mathbf{e}}_{H_3} + \hat{\mathbf{e}}_{K_2} - \hat{\mathbf{e}}_{K_3} - \hat{\mathbf{e}}_{L_1} + \hat{\mathbf{e}}_{L_2} - \hat{\mathbf{e}}_{M_1} + \hat{\mathbf{e}}_{M_2}$. Equivalently, the symmetry indicators are given by $\nu_1^{(75)} = 2n_{M_2} + n_{M_3} - n_{M_4} - 2n_{\Gamma_1} - n_{\Gamma_3} + n_{\Gamma_4} - 2n_{X_1} \bmod 4$ and $\nu_1^{(168)} = n_{\Gamma_1} - 2n_{\Gamma_2} + 5n_{\Gamma_3} + 2n_{\Gamma_4} + 3n_{\Gamma_5} + 2n_{H_1} - 2n_{H_2} - 3n_{L_1} \bmod 6$,

$C_{x,y}$, necessarily vanish since the Chern vector must transform as a pseudovector under the elements of the space group [43].

Figure 4-7 summarizes the results of applying our technique to the optimization of 3D Chern insulating phases with target Chern vectors $\mathbf{C} = (0, 0, C_z)$ for $C_z = 1$ and $C_z = 2$. For SG 75, we find PhC designs with $C_z = 1$ and a HS gap of 1.5% between bands 4 and 5 [Fig. 4-7(c)]. Interestingly, the optimization finds a design with larger HS gap of 2.1% between bands 6 and 7 for a target Chern number of $C_z = 2$ [Fig. 4-7(d)]. Conversely, in SG 168, we are unable to find a $C_z = 2$ design with a complete HS gap, but find a PhC design with a large 5.1% $C_z = 1$ HS gap between bands 7 and 8 [Fig. 4-7(e)].

In more detail, we consider a (010) interface with the applied magnetic field's sign flipped at the interface [Fig. 4-7(f)]. This flip preserves the band structure and, more specifically, the gap; but negates the associated Chern number, yielding a gap Chern number of $\Delta C_z = 1 - (-1) = 2$. By the bulk–boundary correspondence principle, two chiral surface states must therefore traverse the gap. To verify this, we computed the surface band structure across the remaining good wave numbers (k_x, k_z) using a supercell of $8 + 8$ unit cells along y, as shown in Fig. 4-7(g). A pair of chiral surface states traverse the gap, consistent with expectations, and effectively behave as a pair of 1D edge states at each k_x value. Additionally, we observe that the complete HS gap metric is highly effective in this example, in fact equaling the full BZ gap. For computational reasons, our y-supercell was implemented as periodic; as a result, a set of redundant oppositely-traversing dispersion copies, residing at a second interface, were manually removed from the band structure visualization for clarity.

4.4 Conclusion

We have presented a combined global and local optimization framework which exploits several off-the-shelf optimization algorithms to effectively handle the non-continuous and non-differentiable objective functions that arise in the bandgap optimization of

with irrep labels given in CDML notation [30].

topological PhCs. We incorporate a level-set function expressed as a symmetry-constrained Fourier sum with a relatively low parameter-space dimensionality that makes optimization by gradient-free methods tractable, while simultaneously instating a symmetry-constrained search space. We have applied our method in several distinct symmetry settings, hosting several distinct types of nontrivial band topology. In doing so, we have discovered several novel topological photonic designs featuring nodal lines, ideal Weyl points, and nontrivial 3D Chern insulators. To our knowledge, these are the first proposed structures with nodal lines and Weyl points in the interior of the BZ, a design task that would be computationally intractable without the use of the symmetry-based topological band analysis tools that we use here. As an example of the applicability and potential of our framework, our best Chern insulator design has the largest known bandgap of any 3D photonic Chern insulator, and achieves this without explicitly incorporating the supermodulation techniques employed in previous designs.

As we also discuss in the Results section, the HS bandgap maximization objective serves as an effective proxy for frequency-isolation of nodal lines and Weyl points. Nevertheless, this proxy can lead to non-ideal degeneracies that that overlap spectrally with other "trivial" bulk band features. Future work could focus on further optimization of the frequency isolation, especially in the local optimization stage given the computational cost of considering the full BZ. For example, an objective function for Weyl points could optimize the shape of the DOS directly such that it realizes the characteristic parabolic frequency-dependence over a desired range to exclude the possibility of Fermi pockets at or near the Weyl point frequency. Analogously, a modified objective function for nodal lines could incorporate terms that minimize frequency dispersion across the line while simultaneously maximizing the bandgap to nearby Fermi pockets.

While we explore just three different kinds of band topology, the method we present is general and can be applied to any other topological feature that can be analyzed from band symmetry. For example, the optimization scheme is not restricted to nodal lines and Weyl points in the interior of the BZ, but could be easily extended to nodal lines crossing or Weyl points lying on HS **k**-lines [267], simply by adjusting the objective function or symmetry setting accordingly. More generally, other topological features of potential interest include the quantum spin Hall effect, higher-order topological insulators (e.g., hinge states and corner states) [11, 162], and symmetry-protected degeneracies more broadly [261]. Furthermore, our exploration of optimization techniques can provide insight into bandgap optimization of non-topological (i.e., trivial) 3D and 2D PhCs, as well as of other quasiparticles, such as in engineered phononic crystals.

The Fourier level-set parameterization and its incorporation of a maximal (spatial) frequency cutoff has the advantage that it generally avoids rapid structural variations and fine details. Nevertheless, such constraints could potentially be incorporated with higher fidelity and stricter tolerances by exploiting standard techniques in topology optimization [206, 113]. In the context of PhCs, robust optimization techniques using min-max formulations for the objective function have been used to ensure satisfactory performance in the case of over- or underetching during fabrication [147, 148]. In photonics design more broadly, density filters and projection steps can be applied periodically throughout optimization to remove small features and gaps [66]. A separate but related concern of particular importance for 3D structures is the incorporation of connectivity constraints to avoid floating structures or holes. We note that several of our proposed Weyl point and Chern insulator structures exhibit floating features, reflecting the absence of a connectivity enforcement in the Fourier level-set parameterization. While enforcement of connectivity constraints remains relatively underexplored in the 3D PhC context—-perhaps in reflection of earlier topology optimizations of trivial bandgaps having automatically tended towards connected designs [148]—such constraints can in principle be explicitly incorporated in our framework [114]. On the other hand, it is well-established that PhCs with unconnected geometric features can host large (trivial) bandgaps [141, 21] and such disconnected structures may be realizable with emerging fabrication techniques such as implosion fabrication, wherein dielectric or metallic structures are embedded in a low-index hydrogel scaffold [161, 151].

In principle, any global or local optimization algorithm can be used in our framework. Although we have explored several choices for each stage, we have not exhaustively compared different algorithms, and further improvements may well be achievable by simply using more optimal algorithms, e.g., algorithms that explicitly handle non-continuous constraints. Similarly, gradient-based optimization algorithms could be used in the local optimization stage to speed up convergence. As a particularly exciting outlook, we note opportunities to exploit Bayesian optimization, a model-based algorithm typically used for global optimization of expensive non-convex functions. Recently, Bayesian optimization using Bayesian neural networks was demonstrated to outperform several gradient-free global optimization algorithms in the design of PhCs [100]. Bayesian optimization can also be extended to take advantage of gradient information [5] and non-continuous constraints [55] to improve convergence.

In principle, any global or local optimization algorithms can be used in our method, allowing the flexibility to easily re-implement this method with different packages or languages. Additionally, we have not exhaustively compared different algorithms, so there may be room for improvement using more optimal algorithms or algorithms that can explicitly handle non-continuous constraints. For example, gradient-based optimization algorithms can be used in the local optimization stage to speed up convergence while the global optimization stage avoids getting trapped in local optima. In another example, Bayesian optimization (BO) is a model-based algorithm typically used for optimization of expensive non-convex functions. BO using Bayesian neural networks to operate on an image representation of the PhC directly has been shown to outperform many other gradient-free global optimization algorithms [100]. BO can be extended to take advantage of gradients [5], function bounds [158] and constraints [55]. Because we can control the runtime vs accuracy tradeoff of our simulations (i.e., simulating only HS **k**-points, **k**-lines, or the entire BZ), multi-fidelity Bayesian optimization methods would also be applicable to maximize computational utility [46].

Code is publicly available[2].

[2]`https://github.com/samuelkim314/topo-phc-opt`

Chapter 5

Conclusion

In this thesis, I have proposed several deep learning- and optimization-based tools for automating the process of scientific discovery and inverse design with applications in physics, chemistry, and engineering.

The work in this thesis on the EQuation Learner (EQL) neural network architecture and its variants–the Stacked EQL (SEQL) and the Hyper EQL (HEQL) for parametric equations–can enable symbolic regression and the process of automatically discovering governing equations from data on high-dimensional and complex datasets. While here we present experiments on datasets such as images, dynamic systems, and parametric equations, there are many more types of structure and complexity that can be integrated into symbolic regression. For example, AI-Feynman 2.0 tries discover invariants and symmetries present to map the data onto a lower-dimensional space that can be more easily handled by symbolic regression [226]. Siamese neural networks [244] and manifold learning [136] have been used to discover invariants and conserved quantities . PDE-Net uses a constrained convolutional neural network architecture to automatically discover the differential terms for the governing PDE, rather than needing hand-crafted finite-difference approximations [125]. Additionally, while we demonstrate the use of simple convolutional encoders to automatically discover latent parameters in the dynamics datasets, there are a number of much more powerful encoder architectures and even alternate approaches such as self-supervised learning (e.g. contrastive learning) for more complex datasets. Many of these extensions can also be incorporated into the EQL network architecture, in part due to the general flexibility and expressiveness of deep learning.

Bayesian optimization is a powerful tool for global optimization and inverse design in a variety of fields in science and engineering, and the work in this thesis paves a path towards incorporating much of the intuition and knowledge that we have gained into the optimization process in the form of inductive biases in the surrogate models, which we have shown serves to improve optimization performance. While there have been many works on using deep learning for optimization and inverse design in the context of photonics, most of these typically rely on generating a dataset and training a machine learning model on the data in advance of optimization. which largely limits the scope of these approaches to use cases where one may need to repeatedly perform optimization in the same design space many times. In contrast, our work is able to automatically discover designs without pre-computed datasets and is thus more broadly applicable. Furthermore, our approach can potentially take advantage of the rapid advances in deep learning research such as transfer learning or self-supervised learning for better initializations of the neural network weights. Uncertainty calibration or nested hyper-parameter tuning could potentially be incorporated during optimization as an inner loop or periodically to improve training dynamics. Additionally, we hope to apply our method to more realistic use cases such as optimization topological photonic crystals, as discussed in Chapter 4.

Finally, we propose a framework for the automated discovery and global optimization of topological photonic crystals, which is conventionally a difficult but exciting goal to achieve. We hope to apply the framework to a number of other systems, including 2D photonic crystals and other types of topology. Additionally, we hope to incorporate fabrication constraints and eventually discover novel structures that can be fabricated and experimentally measured.

References

[1] Brillouin.jl (v0.5.10). https://github.com/thchr/Brillouin.jl. Accessed 1 Feb. 2023.

[2] MPBUtils.jl (v0.1.11). https://github.com/thchr/MPBUtils.jl. Accessed 1 Feb. 2023.

[3] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL http://tensorflow.org/. Software available from tensorflow.org.

[4] A. Alexandradinata. *Spatially-protected Topology and Group Cohomology in Band Insulators.* PhD thesis, Princeton University, 2015. URL http://arks.princeton.edu/ark:/88435/dsp01kp78gj767.

[5] Sebastian E Ament and Carla P Gomes. Scalable first-order Bayesian optimization via structured automatic differentiation. In *International Conference on Machine Learning*, pages 500–516. PMLR, 2022. doi:10.48550/arXiv.2206.08366.

[6] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[7] Xianyu Ao, Zhifang Lin, and C. T. Chan. One-way edge mode in a magneto-optical honeycomb photonic crystal. *Phys. Rev. B*, 80:033105, 2009. doi:10.1103/PhysRevB.80.033105.

[8] M. I. Aroyo, editor. *International Tables for Crystallography Volume A: Space-group Symmetry.* Wiley, 6 edition, 2016. doi:10.1107/97809553602060000114.

[9] Raul Astudillo and Peter Frazier. Bayesian optimization of composite functions. In *International Conference on Machine Learning*, pages 354–363. PMLR, 2019.

[10] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. Botorch: a framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[11] Wladimir A Benalcazar, B Andrei Bernevig, and Taylor L Hughes. Electric multipole moments, topological multipole moment pumping, and chiral hinge states in crystalline insulators. *Phys. Rev. B*, 96(24):245115, 2017. doi:10.1103/PhysRevB.96.245115.

[12] Ari S Benjamin, David Rolnick, and Konrad Kording. Measuring and regularizing networks in function space. *arXiv preprint arXiv:1805.08289*, 2018.

[13] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pages 115–123. PMLR, 2013.

[14] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[15] L. P. Bouckaert, R. Smoluchowski, and E. Wigner. Theory of Brillouin zones and symmetry properties of wave functions in crystals. *Phys. Rev.*, 50:58, 1936. doi:10.1103/PhysRev.50.58.

[16] Barry Bradlyn, L. Elcoro, Jennifer Cano, M. G. Vergniory, Zhijun Wang, C. Felser, M. I. Aroyo, and B. Andrei Bernevig. Topological quantum chemistry. *Nature*, 547(7663):298–305, 2017. doi:10.1038/nature23268.

[17] Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[18] Wessel Bruinsma, Eric Perim, William Tebbutt, Scott Hosking, Arno Solin, and Richard Turner. Scalable exact inference in multi-output gaussian processes. In *International Conference on Machine Learning*, pages 1190–1201. PMLR, 2020.

[19] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences of the United States of America*, 113(15):3932–7, apr 2016. ISSN 1091-6490. doi:10.1073/pnas.1517384113.

[20] Jennifer Cano and Barry Bradlyn. Band representations and topological quantum chemistry. *Annu. Rev. Condens. Matter Phys.*, 12:225–246, 2021. doi:10.1146/annurev-conmatphys-041720-124134.

[21] Rose K Cersonsky, James Antonaglia, Bradley D Dice, and Sharon C Glotzer. The diversity of three-dimensional photonic crystals. *Nature communications*, 12(1):1–7, 2021. doi:10.1038/s41467-021-22809-6.

[22] Kathleen Champion, Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Data-driven discovery of coordinates and governing equations. *Proceedings of the National Academy of Sciences*, 116(45):22445–22451, 2019.

[23] Pradyumna Chari, Chinmay Talegaonkar, Yunhao Ba, and Achuta Kadambi. Visual physics: Discovering physical laws from videos. *arXiv preprint arXiv:1911.11893*, 2019.

[24] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[25] Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surfaces of multilayer networks. In *Journal of Machine Learning Research*, volume 38, pages 192–204. Microtome Publishing, 2015.

[26] Thomas Christensen, Hoi Chun Po, John D Joannopoulos, and Marin Soljačić. Location and topology of the fundamental gap in photonic crystals. *Phys. Rev. X*, 12(2):021066, 2022. doi:10.1103/PhysRevX.12.021066.

[27] Rasmus E. Christiansen, Fengwen Wang, Ole Sigmund, and Søren Stobbe. Designing photonic topological insulators with quantum-spin-Hall edge states using topology optimization. *Nanophotonics*, 8(8):1363–1369, 2019. doi:10.1515/nanoph-2019-0057.

[28] Allan Costa, Rumen Dangovski, Owen Dugan, Samuel Kim, Pawan Goyal, Marin Soljačić, and Joseph Jacobson. Fast neural models for symbolic regression at scale. *arXiv preprint arXiv:2007.10784*, 2020.

[29] Steven J Cox and David C Dobson. Band structure optimization of two-dimensional photonic crystals in H-polarization. *J. Comput. Phys.*, 158(2): 214–224, 2000. doi:10.1006/jcph.1999.6415.

[30] A. P. Cracknell, B. L. Davies, S. C. Miller, and W. F. Love. *Kronecker Product Tables. General Introduction and Tables of Irreducible Representations of Space Groups.*, volume 1. New York: IFI/Plenum, 1979.

[31] Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33:17429–17442, 2020.

[32] R. V. Craster, T. Antonakakis, M. Makwana, and S. Guenneau. Dangers of using the edges of the Brillouin zone. *Phys. Rev. B*, 86:115130, 2012. doi:10.1103/PhysRevB.86.115130.

[33] Sandip De, Albert P Bartók, Gábor Csányi, and Michele Ceriotti. Comparing molecules and solids across structural and alchemical space. *Physical Chemistry Chemical Physics*, 18(20):13754–13769, 2016.

[34] Filipe de Avila Belbute-Peres, Yi-fan Chen, and Fei Sha. Hyperpinn: Learning parameterized differential equations with physics-informed hypernetworks. In *The Symbiosis of Deep Learning and Differential Equations*, 2021.

[35] María Blanco de Paz, Maia G. Vergniory, Dario Bercioux, Aitzol García-Etxarri, and Barry Bradlyn. Engineering fragile topology in photonic crystals: Topological quantum chemistry of light. *Phys. Rev. Research*, 1:032005, 2019. doi:10.1103/PhysRevResearch.1.032005.

[36] Aryan Deshwal and Jana Doppa. Combining latent space and structured kernels for bayesian optimization over combinatorial spaces. *Advances in Neural Information Processing Systems*, 34, 2021.

[37] Chiara Devescovi, Mikel García-Díez, Iñigo Robredo, María Blanco de Paz, Jon Lasa-Alonso, Barry Bradlyn, Juan L Mañes, Maia G Vergniory, and Aitzol García-Etxarri. Cubic 3D Chern photonic insulators with orientable large Chern vectors. *Nat. Commun.*, 12(1):1–12, 2021. doi:10.1038/s41467-021-27168-w.

[38] David C Dobson and Steven J Cox. Maximizing band gaps in two-dimensional photonic crystals. *SIAM J. Appl. Math.*, 59(6):2108–2120, 1999. URL `https://www.jstor.org/stable/118418`.

[39] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1675–1685. PMLR, 09–15 Jun 2019. URL `https://proceedings.mlr.press/v97/du19c.html`.

[40] Luis Elcoro, Zhida Song, and B. Andrei Bernevig. Application of induction procedure and Smith decomposition in calculation and topological classification of electronic band structures in the 230 space groups. *Phys. Rev. B*, 102:035110, Jul 2020. doi:10.1103/PhysRevB.102.035110.

[41] David Eriksson, Michael Pearce, Jacob Gardner, Ryan D Turner, and Matthias Poloczek. Scalable global optimization via local bayesian optimization. *Advances in neural information processing systems*, 32, 2019.

[42] Qinwei Fan, Jacek M. Zurada, and Wei Wu. Convergence of online gradient method for feedforward neural networks with smoothing L1/2 regularization penalty. *Neurocomputing*, 131:208–216, may 2014. ISSN 0925-2312. doi:10.1016/J.NEUCOM.2013.10.023. URL `https://www.sciencedirect.com/science/article/pii/S0925231213010825`.

[43] Chen Fang, Matthew J Gilbert, and B Andrei Bernevig. Bulk topological invariants in noninteracting point group symmetric insulators. *Phys. Rev. B*, 86 (11):115112, 2012. doi:10.1103/PhysRevB.86.115112.

[44] GP Fletcher and CJ Hinde. Learning the activation function for the neurons in neural networks. In *International Conference on Artificial Neural Networks*, pages 611–614. Springer, 1994. doi:10.1007/978-1-4471-2097-1_143.

[45] Stanislav Fort, Huiyi Hu, and Balaji Lakshminarayanan. Deep ensembles: A loss landscape perspective. *arXiv preprint arXiv:1912.02757*, 2019.

[46] Peter I Frazier. A tutorial on Bayesian optimization. *arXiv:1807.02811*, 2018. URL `https://doi.org/10.48550/arXiv.1807.02811`.

[47] Takahiro Fukui, Yasuhiro Hatsugai, and Hiroshi Suzuki. Chern numbers in discretized Brillouin zone: Efficient method of computing (spin) Hall conductances. *J. Phys. Soc. Jpn.*, 74(6):1674–1677, 2005. doi:10.1143/jpsj.74.1674.

[48] Joerg M Gablonsky and Carl T Kelley. A locally-biased form of the DIRECT algorithm. *J. Glob. Optim.*, 21(1):27–37, 2001. doi:10.1023/A:1017930332101.

[49] Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. *arXiv preprint arXiv:1703.02910*, 2017.

[50] Trevor Gale, Erich Elsen, and Sara Hooker. The state of sparsity in deep neural networks. *arXiv preprint arXiv:1902.09574*, 2019.

[51] Wenlong Gao, Biao Yang, Ben Tremain, Hongchao Liu, Qinghua Guo, Lingbo Xia, Alastair P Hibbins, and Shuang Zhang. Experimental observation of photonic nodal line degeneracies in metacrystals. *Nat. Commun.*, 9(1):1–7, 2018. doi:10.1038/s41467-018-03407-5.

[52] Jacob Gardner, Geoff Pleiss, Kilian Q Weinberger, David Bindel, and Andrew G Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. *Advances in neural information processing systems*, 31, 2018.

[53] Roman Garnett. *Bayesian Optimization.* Cambridge University Press, 2022. in preparation.

[54] Michael S Gashler and Stephen C Ashmore. Modeling time series data with deep fourier neural networks. *Neurocomputing*, 188:3–11, 2016. doi:10.1016/j.neucom.2015.01.108.

[55] Michael Adam Gelbart. *Constrained Bayesian optimization and applications.* PhD thesis, Harvard University, 2015.

[56] Rajib Ghosh Chaudhuri and Santanu Paria. Core/shell nanoparticles: classes, properties, synthesis mechanisms, characterization, and applications. *Chemical reviews*, 112(4):2373–2433, 2012.

[57] Joel Goh, Ilya Fushman, Dirk Englund, and Jelena Vučković. Genetic optimization of photonic bandgap structures. *Opt. Express*, 15(13):8218, 2007. doi:10.1364/oe.15.008218.

[58] Rafael Gómez-Bombarelli, Jennifer N Wei, David Duvenaud, José Miguel Hernández-Lobato, Benjamín Sánchez-Lengeling, Dennis Sheberla, Jorge Aguilera-Iparraguirre, Timothy D Hirzel, Ryan P Adams, and Alán Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4(2):268–276, 2018.

[59] Daniele Grattarola and Cesare Alippi. Graph neural networks in tensorflow and keras with spektral. *arXiv preprint arXiv:2006.12138*, 2020.

[60] Ryan-Rhys Griffiths and José Miguel Hernández-Lobato. Constrained bayesian optimization for automatic chemical design using variational autoencoders. *Chemical science*, 11(2):577–586, 2020.

[61] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International conference on machine learning*, pages 1321–1330. PMLR, 2017.

[62] Qinghua Guo, Oubo You, Biao Yang, James B Sellman, Edward Blythe, Hongchao Liu, Yuanjiang Xiang, Jensen Li, Dianyuan Fan, Jing Chen, et al. Observation of three-dimensional photonic Dirac points and spin-polarized surface arcs. *Phys. Rev. Lett.*, 122(20):203903, 2019. doi:10.1103/PhysRevLett.122.203903.

[63] Fredrik K Gustafsson, Martin Danelljan, and Thomas B Schon. Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 318–319, 2020.

[64] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

[65] F. D. M. Haldane. Model for a quantum Hall effect without Landau levels: Condensed-matter realization of the "parity anomaly". *Phys. Rev. Lett.*, 61: 2015–2018, 1988. doi:10.1103/PhysRevLett.61.2015.

[66] Alec M Hammond, Ardavan Oskooi, Mo Chen, Zin Lin, Steven G Johnson, and Stephen E Ralph. High-performance hybrid time/frequency-domain topology optimization for large-scale photonics inverse design. *Opt. Express*, 30(3):4467–4491, 2022. doi:10.1364/OE.442074.

[67] Nikolaus Hansen, Youhei Akimoto, and Petr Baudis. CMA-ES/pycma on Github. Zenodo, DOI:10.5281/zenodo.2559634, February 2019. URL `https://doi.org/10.5281/zenodo.2559634`.

[68] J. M. Harrison, P. Kuchment, A. Sobolev, and B. Winn. On occurrence of spectral edges for periodic operators inside the Brillouin zone. *J. Phys. A: Math. Theor.*, 40:7597, 2007. doi:10.1088/1751-8113/40/27/011.

[69] Marton Havasi, Rodolphe Jenatton, Stanislav Fort, Jeremiah Zhe Liu, Jasper Snoek, Balaji Lakshminarayanan, Andrew M Dai, and Dustin Tran. Training independent subnetworks for robust prediction. *arXiv preprint arXiv:2010.06610*, 2020.

[70] 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*, pages 770–778, 2016.

[71] Dan Hendrycks, Mantas Mazeika, Saurav Kadavath, and Dawn Song. Using self-supervised learning can improve model robustness and uncertainty. *Advances in neural information processing systems*, 32, 2019.

[72] Lauri Himanen, Marc O. J. Jäger, Eiaki V. Morooka, Filippo Federici Canova, Yashasvi S. Ranawat, David Z. Gao, Patrick Rinke, and Adam S. Foster. DScribe: Library of descriptors for machine learning in materials science. *Computer Physics Communications*, 247:106949, 2020. ISSN 0010-4655. doi:10.1016/j.cpc.2019.106949. URL `https://doi.org/10.1016/j.cpc.2019.106949`.

[73] Yoyo Hinuma, Giovanni Pizzi, Yu Kumagai, Fumiyasu Oba, and Isao Tanaka. Band structure diagram paths based on crystallography. *Comput. Mater. Sci.*, 128:140–184, 2017. doi:10.1016/j.commatsci.2016.10.015.

[74] Chung-I Ho, Dan-Ju Yeh, Vin-Cent Su, Chieh-Hung Yang, Po-Chuan Yang, Ming-Yi Pu, Chieh-Hsiung Kuan, I-Chun Cheng, and Si-Chen Lee. Plasmonic multilayer nanoparticles enhanced photocurrent in thin film hydrogenated amorphous silicon solar cells. *Journal of Applied Physics*, 112(2): 023113, 2012.

[75] KM Ho, Che Ting Chan, and Costas M Soukoulis. Existence of a photonic gap in periodic dielectric structures. *Phys. Rev. Lett.*, 65(25):3152, 1990. doi:10.1103/PhysRevLett.65.3152. URL `https://link.aps.org/doi/10.1103/PhysRevLett.65.3152`.

[76] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.

[77] Jilin Hu, Jianbing Shen, Bin Yang, and Ling Shao. Infinitely wide graph convolutional networks: semi-supervised learning via gaussian processes. *arXiv preprint arXiv:2002.12168*, 2020.

[78] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E Hopcroft, and Kilian Q Weinberger. Snapshot ensembles: Train 1, get m for free. *arXiv preprint arXiv:1704.00109*, 2017.

[79] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[80] James P Hughes, Stephen Rees, S Barrett Kalindjian, and Karen L Philpott. Principles of early drug discovery. *British journal of pharmacology*, 162(6): 1239–1249, 2011.

[81] Michael Hutchinson, Charline Le Lan, Sheheryar Zaidi, Emilien Dupont, Yee Whye Teh, and Hyunjik Kim. Lietransformer: Equivariant self-attention for lie groups. *arXiv preprint arXiv:2012.10885*, 2020.

[82] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In *International conference on learning and intelligent optimization*, pages 507–523. Springer, 2011.

[83] Daniel Jakubovitz, Raja Giryes, and Miguel RD Rodrigues. Generalization error in deep learning. In *Compressed sensing and its applications*, pages 153–193. Springer, 2019.

[84] Jakob Søndergaard Jensen and Ole Sigmund. Topology optimization for nano-photonics. *Laser & Photonics Reviews*, 5(2):308–321, 2011.

[85] J.S. Jensen and O. Sigmund. Topology optimization for nano-photonics. *Laser Photonics Rev.*, 5(2):308–321, 2010. doi:10.1002/lpor.201000014.

[86] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph generation. In *International conference on machine learning*, pages 2323–2332. PMLR, 2018.

[87] John D. Joannopoulos, Steven G. Johnson, Joshua N. Winn, and Robert D. Meade. *Photonic Crystals: Molding the Flow of Light (Second Edition)*. Princeton University Press, 2 edition, 2008. ISBN 0691124566.

[88] John D Joannopoulos, Steven G Johnson, Joshua N Winn, and Robert D Meade. *Photonic Crystals: Molding the Flow of Light.* Princeton University Press, 2 edition, 2008. ISBN 0691124566.

[89] Sajeev John. Strong localization of photons in certain disordered dielectric superlattices. *Physical review letters*, 58(23):2486, 1987.

[90] Steven G. Johnson. The nlopt nonlinear-optimization package, 2010. URL `http://github.com/stevengj/nlopt`.

[91] Steven G Johnson. The NLopt nonlinear-optimization package, 2014. URL https://github.com/stevengj/nlopt.

[92] Steven G Johnson and John D Joannopoulos. Block-iterative frequency-domain methods for maxwell's equations in a planewave basis. *Opt. Express*, 8(3):173–190, 2001. doi:10.1364/OE.8.000173.

[93] Donald R. Jones, Matthias Schonlau, and William J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13(4):455–492, Dec 1998. ISSN 1573-2916. doi:10.1023/A:1008306431147. URL https://doi.org/10.1023/A:1008306431147.

[94] Christina Jörg, Sachin Vaidya, Jiho Noh, Alexander Cerjan, Shyam Augustine, Georg von Freymann, and Mikael C Rechtsman. Observation of quadratic (charge-2) Weyl point splitting in near-infrared photonic crystals. *Laser Photonics Rev.*, 16(1):2100452, 2022. doi:10.1002/lpor.202100452.

[95] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[96] Chiu Y Kao, Stanley Osher, and Eli Yablonovitch. Maximizing band gaps in two-dimensional photonic crystals by using level set methods. *Appl. Phys. B: Lasers Opt.*, 81(2):235–244, 2005. doi:10.1007/s00340-005-1877-3.

[97] Anuj Karpatne, William Watkins, Jordan Read, and Vipin Kumar. Physics-guided neural networks (pgnn): An application in lake temperature modeling. *arXiv preprint arXiv:1710.11431*, 2017.

[98] E. Kaxirax and J. D. Joannopoulos. *Quantum theory of materials.* Cambridge University Press, 2019. doi:10.1017/9781139030809.

[99] Samuel Kim, Peter Y Lu, Srijon Mukherjee, Michael Gilbert, Li Jing, Vladimir Čeperić, and Marin Soljačić. Integration of neural network-based symbolic regression in deep learning for scientific discovery. *IEEE Transactions on Neural Networks and Learning Systems*, 32(9):4166–4177, 2020.

[100] Samuel Kim, Peter Y Lu, Charlotte Loh, Jamie Smith, Jasper Snoek, and Marin Soljačić. Deep learning for Bayesian optimization of scientific problems with high-dimensional structure. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=tPMQ6Je2rB. https://openreview.net/forum?id=tPMQ6Je2rB, Accessed 8 Feb. 2023.

[101] Samuel Kim, Thomas Christensen, Steven G. Johnson, and Marin Soljačić. Automated discovery and optimization of 3d topological photonic crystals. *ACS Photonics*, 2023. doi:10.1021/acsphotonics.2c01866. URL https://doi.org/10.1021/acsphotonics.2c01866.

[102] Davis E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009.

[103] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[104] Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast bayesian optimization of machine learning hyperparameters on large datasets. In *Artificial Intelligence and Statistics*, pages 528–536. PMLR, 2017.

[105] Dmitrii Kochkov, Jamie A Smith, Ayya Alieva, Qing Wang, Michael P Brenner, and Stephan Hoyer. Machine learning–accelerated computational fluid dynamics. *Proceedings of the National Academy of Sciences*, 118(21), 2021.

[106] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations. In *International Conference on Artificial Intelligence and Statistics*, pages 3393–3403. PMLR, 2020.

[107] JohnR. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, jun 1994. ISSN 0960-3174. doi:10.1007/BF00175355. URL http://link.springer.com/10.1007/BF00175355.

[108] Jorrit Kruthoff, Jan de Boer, Jasper van Wezel, Charles L. Kane, and Robert-Jan Slager. Topological classification of crystalline insulators through band structure combinatorics. *Phys. Rev. X*, 7:041069, 2017. doi:10.1103/PhysRevX.7.041069.

[109] Volodymyr Kuleshov, Nathan Fenner, and Stefano Ermon. Accurate uncertainties for deep learning using calibrated regression. In *International conference on machine learning*, pages 2796–2804. PMLR, 2018.

[110] Matt J Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In *International conference on machine learning*, pages 1945–1954. PMLR, 2017.

[111] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in neural information processing systems*, pages 6402–6413, 2017.

[112] Alan Lapedes and Robert Farber. Nonlinear signal processing using neural networks: Prediction and system modelling. Technical report, 6 1987. URL https://www.osti.gov/biblio/5470451.

[113] Boyan S Lazarov, Fengwen Wang, and Ole Sigmund. Length scale and manufacturability in density-based topology optimization. *Arch. Appl. Mech.*, 86(1): 189–218, 2016. doi:10.1007/s00419-015-1106-4.

[114] Quhao Li, Wenjiong Chen, Shutian Liu, and Liyong Tong. Structural topology optimization considering connectivity constraint. *Struct. Multidiscip. Optim.*, 54(4):971–984, 2016. doi:10.1007/s00158-016-1459-5.

[115] Weibai Li, Fei Meng, Yafeng Chen, Yang fan Li, and Xiaodong Huang. Topology optimization of photonic and phononic crystals and metamaterials: A review. *Adv. Theory Simul.*, 2(7):1900017, 2019. doi:10.1002/adts.201900017.

[116] Zin Lin, Lysander Christakis, Yang Li, Eric Mazur, Alejandro W. Rodriguez, and Marko Lončar. Topology-optimized dual-polarization Dirac cones. *Phys. Rev. B*, 97:081408, Feb 2018. doi:10.1103/PhysRevB.97.081408. URL `https://link.aps.org/doi/10.1103/PhysRevB.97.081408`.

[117] Zin Lin, Victor Liu, Raphaël Pestourie, and Steven G Johnson. Topology optimization of freeform large-area metasurfaces. *Optics express*, 27(11):15765–15775, 2019.

[118] Boyuan Liu, J D Joannopoulos, Steven G Johnson, and Ling Lu. Generalized Gilat–Raubenheimer method for density-of-states calculation in photonic crystals. *J. Opt.*, 20:044005, 2018. doi:10.1088/2040-8986/aaae52.

[119] Gui-Geng Liu, Zhen Gao, Qiang Wang, Xiang Xi, Yuan-Hang Hu, Maoren Wang, Chengqi Liu, Xiao Lin, Longjiang Deng, Shengyuan A Yang, et al. Topological chern vectors in three-dimensional photonic crystals. *Nature*, 609(7929): 925–930, 2022. doi:10.1038/s41586-022-05077-2.

[120] Haitao Liu, Yew-Soon Ong, Xiaobo Shen, and Jianfei Cai. When gaussian process meets big data: A review of scalable gps. *IEEE transactions on neural networks and learning systems*, 31(11):4405–4423, 2020.

[121] Jinlong Liu, Guoqing Jiang, Yunzhi Bai, Ting Chen, and Huayan Wang. Understanding why neural networks generalize well through gsnr of parameters. *arXiv preprint arXiv:2001.07384*, 2020.

[122] Kexin Liu, Linfang Shen, and Sailing He. One-way edge mode in a gyromagnetic photonic crystal slab. *Opt. Lett.*, 37(19):4110, 2012. doi:10.1364/ol.37.004110.

[123] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022.

[124] Charlotte Loh, Thomas Christensen, Rumen Dangovski, Samuel Kim, and Marin Soljacic. Surrogate-and invariance-boosted contrastive learning for data-scarce applications in science. *arXiv preprint arXiv:2110.08406*, 2021.

[125] Zichao Long, Yiping Lu, Xianzhong Ma, and Bin Dong. PDE-Net: Learning PDEs from Data. In *Proceedings of Machine Learning Research*, pages 3208–3216, jul 2018. URL `http://proceedings.mlr.press/v80/long18a.html`.

[126] Zichao Long, Yiping Lu, and Bin Dong. Pde-net 2.0: Learning pdes from data with a numeric-symbolic hybrid deep network. *Journal of Computational Physics*, 399:108925, 2019. doi:10.1016/j.jcp.2019.108925.

[127] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[128] Christos Louizos and Max Welling. Multiplicative normalizing flows for variational bayesian neural networks. *arXiv preprint arXiv:1703.01961*, 2017.

[129] Christos Louizos, Max Welling, and Diederik P. Kingma. Learning Sparse Neural Networks through L_0 Regularization. *arXiv preprint arXiv:1712.01312*, dec 2017. URL `https://arxiv.org/abs/1712.01312`.

[130] Ling Lu, Liang Fu, John D Joannopoulos, and Marin Soljačić. Weyl points and line nodes in gyroid photonic crystals. *Nat. Photonics*, 7(4):294–299, 2013. doi:10.1038/nphoton.2013.42.

[131] Ling Lu, John D Joannopoulos, and Marin Soljačić. Topological photonics. *Nat. Photonics*, 8(11):821–829, 2014. doi:10.1038/nphoton.2014.248.

[132] Ling Lu, Zhiyu Wang, Dexin Ye, Lixin Ran, Liang Fu, John D Joannopoulos, and Marin Soljačić. Experimental observation of Weyl points. *Science*, 349 (6248):622–624, 2015. doi:10.1126/science.aaa9273.

[133] Ling Lu, Haozhe Gao, and Zhong Wang. Topological one-way fiber of second Chern number. *Nat. Commun.*, 9(1):1–7, 2018. doi:10.1038/s41467-018-07817-3.

[134] Peter Y Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Physical Review X*, 10(3):031056, 2020. doi:10.1103/PhysRevX.10.031056.

[135] Peter Y Lu, Joan Ariño Bernad, and Marin Soljačić. Discovering sparse interpretable dynamics from partial observations. *Communications Physics*, 5(1): 1–7, 2022. doi:10.1038/s42005-022-00987-z.

[136] Peter Y Lu, Rumen Dangovski, and Marin Soljačić. Discovering conservation laws using optimal transport and manifold learning. *arXiv preprint arXiv:2208.14995*, 2022.

[137] Yingtao Luo, Qiang Liu, Yuntian Chen, Wenbo Hu, and Jun Zhu. Ko-pde: Kernel optimized discovery of partial differential equations with varying coefficients. *arXiv preprint arXiv:2106.01078*, 2021.

[138] B. Q. Lv, H. M. Weng, B. B. Fu, X. P. Wang, H. Miao, J. Ma, P. Richard, X. C. Huang, L. X. Zhao, G. F. Chen, Z. Fang, X. Dai, T. Qian, and H. Ding. Experimental discovery of Weyl semimetal TaAs. *Phys. Rev. X*, 5:031013, 2015. doi:10.1103/PhysRevX.5.031013.

[139] Wesley J Maddox, Maximilian Balandat, Andrew G Wilson, and Eytan Bakshy. Bayesian optimization with high-dimensional outputs. *Advances in Neural Information Processing Systems*, 34, 2021.

[140] Wesley J Maddox, Samuel Stanton, and Andrew G Wilson. Conditioning sparse variational gaussian processes for online decision-making. *Advances in Neural Information Processing Systems*, 34, 2021.

[141] Martin Maldovan, Chaitanya K Ullal, W Craig Carter, and Edwin L Thomas. Exploring for 3d photonic bandgap structures in the 11 fcc space groups. *Nature materials*, 2(10):664–667, 2003. doi:10.1038/nmat979.

[142] Cédric Malherbe and Nicolas Vayatis. Global optimization of lipschitz functions. *arXiv preprint arXiv:1703.02628*, 2017.

[143] J. L. Mañes. Existence of bulk chiral fermions and crystal symmetry. *Phys. Rev. B*, 85:155118, 2012. doi:10.1103/physrevb.85.155118.

[144] Georg Martius and Christoph H. Lampert. Extrapolation and learning equations. *arXiv preprint arXiv:1610.02995*, oct 2016. URL `http://arxiv.org/abs/1610.02995`.

[145] Florian Maurin, Claus Claeys, Elke Deckers, and Wim Desmet. Probability that a band-gap extremum is located on the irreducible Brillouin-zone contour for the 17 different plane crystallographic lattices. *Int. J. Solids Struct.*, 135: 26–36, 2018. doi:10.1016/j.ijsolstr.2017.11.006.

[146] Han Men, Ngoc Cuong Nguyen, Robert M Freund, Pablo A Parrilo, and Jaume Peraire. Bandgap optimization of two-dimensional photonic crystals using semidefinite programming and subspace methods. *J. Comput. Phys.*, 229(10): 3706–3725, 2010. doi:10.1016/j.jcp.2010.01.023.

[147] Han Men, Robert M Freund, Ngoc C Nguyen, Joel Saa-Seoane, and Jaime Peraire. Fabrication-adaptive optimization with an application to photonic crystal design. *Oper. Res.*, 62(2):418–434, 2014. doi:10.1287/opre.2013.1252.

[148] Han Men, Karen YK Lee, Robert M Freund, Jaime Peraire, and Steven G Johnson. Robust topology optimization of three-dimensional photonic-crystal band-gap structures. *Opt. Express*, 22(19):22632–22648, 2014. doi:10.1364/OE.22.022632.

[149] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger,

Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017. ISSN 2376-5992. doi:10.7717/peerj-cs.103. URL `https://doi.org/10.7717/peerj-cs.103`.

[150] O. D. Miller, C. W. Hsu, M. T. H. Reid, W. Qiu, B. G. DeLacy, J. D. Joannopoulos, M. Soljačić, and S. G. Johnson. Fundamental limits to extinction by metallic nanoparticles. *Phys. Rev. Lett.*, 112:123903, 2014. doi:10.1103/PhysRevLett.112.123903.

[151] Brian Mills, Yannick Salamin, Gaojie Yang, Daniel Oran, Yi Sun, Shai Maayani, Steven E Kooi, Amel Amin Elfadil Elawad, Josue J Lopez, Corban Swain, et al. Implosion fabrication as a platform for three-dimensional nanophotonics. In *CLEO: Science and Innovations*, pages STh4J–1. Optical Society of America, 2021. doi:10.1364/CLEO_SI.2021.STh4J.1.

[152] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. Variational dropout sparsifies deep neural networks. In *34th International Conference on Machine Learning, ICML 2017*, volume 5, pages 3854–3863. International Machine Learning Society (IMLS), 2017. ISBN 9781510855144.

[153] Sean Molesky, Zin Lin, Alexander Y. Piggott, Weiliang Jin, Jelena Vucković, and Alejandro W. Rodriguez. Inverse design in nanophotonics. *Nat. Photonics*, 12(11):659–670, 2018. doi:10.1038/s41566-018-0246-9.

[154] Alejandro Molina, Patrick Schramowski, and Kristian Kersting. Pad\'e activation units: End-to-end learning of flexible activation functions in deep networks. *arXiv preprint arXiv:1907.06732*, 2019.

[155] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

[156] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021 (12):124003, 2021.

[157] Balas Kausik Natarajan. Sparse Approximate Solutions to Linear Systems. *SIAM Journal on Computing*, 24(2):227–234, apr 1995. ISSN 0097-5397. doi:10.1137/S0097539792240406. URL `http://epubs.siam.org/doi/10.1137/S0097539792240406`.

[158] Vu Nguyen and Michael A. Osborne. Knowing the what but not the where in Bayesian optimization. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of

Proceedings of Machine Learning Research, pages 7317–7326. PMLR, 13–18 Jul 2020. URL `https://proceedings.mlr.press/v119/nguyen20d.html`.

[159] Roman Novak, Lechao Xiao, Jiri Hron, Jaehoon Lee, Alexander A. Alemi, Jascha Sohl-Dickstein, and Samuel S. Schoenholz. Neural tangents: Fast and easy infinite neural networks in python. In *International Conference on Learning Representations*, 2020. URL `https://github.com/google/neural-tangents`.

[160] Eric Nussbaum, Nir Rotenberg, and Stephen Hughes. Optimizing the chiral Purcell factor for unidirectional single-photon emitters in topological photonic crystal waveguides using inverse design. *Phys. Rev. A*, 106:033514, 2022. doi:10.1103/PhysRevA.106.033514.

[161] Daniel Oran, Samuel G Rodriques, Ruixuan Gao, Shoh Asano, Mark A Skylar-Scott, Fei Chen, Paul W Tillberg, Adam H Marblestone, and Edward S Boyden. 3d nanofabrication by volumetric deposition and controlled shrinkage of patterned scaffolds. *Science*, 362(6420):1281–1285, 2018. doi:10.1126/science.aau5119.

[162] Yasutomo Ota, Feng Liu, Ryota Katsumi, Katsuyuki Watanabe, Katsunori Wakabayashi, Yasuhiko Arakawa, and Satoshi Iwamoto. Photonic crystal nanocavity based on a topological corner state. *Optica*, 6(6):786–789, 2019. doi:10.1364/OPTICA.6.000786.

[163] Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, David Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *Advances in Neural Information Processing Systems*, pages 13991–14002, 2019.

[164] Tomoki Ozawa, Hannah M. Price, Alberto Amo, Nathan Goldman, Mohammad Hafezi, Ling Lu, Mikael C. Rechtsman, David Schuster, Jonathan Simon, Oded Zilberberg, and Iacopo Carusotto. Topological photonics. *Rev. Mod. Phys.*, 91 (1):015006, 2019. doi:10.1103/revmodphys.91.015006.

[165] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard Turner, and Mohammad Emtiyaz E Khan. Continual deep learning by functional regularisation of memorable past. *Advances in Neural Information Processing Systems*, 33:4453–4464, 2020.

[166] Giambattista Parascandolo, Heikki Huttunen, and Tuomas Virtanen. Taming the waves: sine as activation function in deep neural networks. 2016.

[167] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[168] Haedong Park, Stephan Wong, Xiao Zhang, and Sang Soon Oh. Non-Abelian charged nodal links in a dielectric photonic crystal. *ACS Photonics*, 8(9):2746–2754, 2021. doi:10.1021/acsphotonics.1c00876.

[169] Haedong Park, Wenlong Gao, Xiao Zhang, and Sang Soon Oh. Nodal lines in momentum space: topological invariants and recent realizations in photonic and other systems. *Nanophotonics*, 11(11):2779–2801, 2022. doi:10.1515/nanoph-2021-0692.

[170] Jaideep Pathak, Mustafa Mustafa, Karthik Kashinath, Emmanuel Motheau, Thorsten Kurth, and Marcus Day. Using machine learning to augment coarse-grid computational fluid dynamics simulations. *arXiv preprint arXiv:2010.00072*, 2020.

[171] Valerio Perrone, Rodolphe Jenatton, Matthias Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 6846–6856, 2018.

[172] John Peurifoy, Yichen Shen, Li Jing, Yi Yang, Fidel Cano-Renteria, Brendan G DeLacy, John D Joannopoulos, Max Tegmark, and Marin Soljačić. Nanophotonic particle simulation and inverse design using artificial neural networks. *Science advances*, 4(6):eaar4206, 2018.

[173] Alexander Y Piggott, Jesse Lu, Konstantinos G Lagoudakis, Jan Petykiewicz, Thomas M Babinec, and Jelena Vučković. Inverse design and demonstration of a compact and broadband on-chip wavelength demultiplexer. *Nature Photonics*, 9(6):374–377, 2015.

[174] Hoi Chun Po. Symmetry indicators of band topology. *J. Phys.: Condens. Matter*, 32(26):263001, 2020. doi:10.1088/1361-648X/ab7adb.

[175] Hoi Chun Po, Haruki Watanabe, Michael P Zaletel, and Ashvin Vishwanath. Filling-enforced quantum band insulators in spin-orbit coupled crystals. *Science Adv.*, 2(4):e1501782, 2016. doi:10.1126/sciadv.1501782.

[176] Hoi Chun Po, A. Vishwanath, and Haruki Watanabe. Symmetry-based indicators of band topology in the 230 space groups. *Nat. Commun.*, 8:50, 2017. doi:10.1038/s41467-017-00133-2.

[177] Yin Poo, Rui-xin Wu, Zhifang Lin, Yan Yang, and C. T. Chan. Experimental realization of self-guiding unidirectional electromagnetic edge states. *Phys. Rev. Lett.*, 106:093903, 2011. doi:10.1103/PhysRevLett.106.093903.

[178] Yin Poo, Rui-xin Wu, Zhifang Lin, Yan Yang, and C. T. Chan. Experimental realization of self-guiding unidirectional electromagnetic edge states. *Phys. Rev. Lett.*, 106:093903, 2011. doi:10.1103/PhysRevLett.106.093903.

[179] Wenjun Qiu, Brendan G DeLacy, Steven G Johnson, John D Joannopoulos, and Marin Soljačić. Optimization of broadband optical response of multilayer nanospheres. *Optics express*, 20(16):18494–18504, 2012.

[180] Srinivas Raghu and Frederick Duncan Michael Haldane. Analogs of quantum-Hall-effect edge states in photonic crystals. *Phys. Rev. A*, 78(3):033834, 2008. doi:10.1103/PhysRevA.78.033834.

[181] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[182] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017.

[183] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[184] Raghunathan Ramakrishnan, Pavlo O Dral, Matthias Rupp, and O Anatole Von Lilienfeld. Quantum chemistry structures and properties of 134 kilo molecules. *Scientific data*, 1(1):1–7, 2014.

[185] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[186] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017. URL `https://openreview.net/forum?id=rJY0-Kcll`.

[187] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. *arXiv preprint arXiv:1802.09127*, 2018.

[188] Thomas Harvey Rowan. *Functional stability analysis of numerical algorithms*. PhD thesis, The University of Texas at Austin, 1990.

[189] Binxin Ru, Xingchen Wan, Xiaowen Dong, and Michael Osborne. Interpretable neural architecture search via bayesian optimisation with weisfeiler-lehman kernels. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=j9Rv7qdXjd`.

[190] Lars Ruddigkeit, Ruud Van Deursen, Lorenz C Blum, and Jean-Louis Reymond. Enumeration of 166 billion organic small molecules in the chemical universe database gdb-17. *Journal of chemical information and modeling*, 52(11):2864–2875, 2012.

[191] Tim GJ Rudner, Zonghao Chen, and Yarin Gal. Rethinking function-space variational inference in bayesian neural networks. In *Third Symposium on Advances in Approximate Bayesian Inference*, 2020.

[192] Samuel Rudy, Alessandro Alla, Steven L. Brunton, and J. Nathan Kutz. Data-driven identification of parametric partial differential equations. *SIAM Journal on Applied Dynamical Systems*, 18(2):643–660, apr 2019. ISSN 15360040. doi:10.1137/18M1191944.

[193] Samuel H. Rudy, Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. Data-driven discovery of partial differential equations. *Science Advances*, 3(4): e1602614, apr 2017. ISSN 2375-2548. doi:10.1126/sciadv.1602614.

[194] Thomas Philip Runarsson and Xin Yao. Search biases in constrained evolutionary optimization. *IEEE Trans. Syst. Man Cybern.*, 35(2):233–243, 2005. doi:10.1109/TSMCC.2004.841906.

[195] Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pages 4442–4450. PMLR, 2018.

[196] S Saltsberger, I Steinberg, and Israel Gannot. Multilayer mie scattering model for investigation of intracellular structural changes in the nucleolus and cytoplasm. *International Journal of Optics*, 2012, 2012.

[197] Hayden Schaeffer. Learning partial differential equations via data discovery and sparse optimization. *Proceedings of the Royal Society A*, 473(2197), 2017. doi:10.6084/m9. URL `http://rspa.royalsocietypublishing.org/content/royprsa/473/2197/20160446.full.pdf`.

[198] Daniel Schlör, Markus Ring, and Andreas Hotho. inalu: Improved neural arithmetic logic unit. *Frontiers in Artificial Intelligence*, 3:71, 2020.

[199] Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science (New York, N.Y.)*, 324(5923):81–5, apr 2009. ISSN 1095-9203. doi:10.1126/science.1165893. URL `http://www.ncbi.nlm.nih.gov/pubmed/19342586`.

[200] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[201] Yichen Shen, Dexin Ye, Ivan Celanovic, Steven G Johnson, John D Joannopoulos, and Marin Soljačić. Optical broadband angular selectivity. *Science*, 343 (6178):1499–1501, 2014.

[202] Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(9), 2011.

[203] Xiaotian Shi and Jinkyu Yang. Spin-1 Weyl point and surface arc state in a chiral phononic crystal. *Phys. Rev. B*, 101(21):214309, 2020. doi:10.1103/physrevb.101.214309.

[204] Yanpeng Shi, Xiaodong Wang, Wen Liu, Tianshu Yang, Rui Xu, and Fuhua Yang. Multilayer silver nanoparticles for light trapping in thin film solar cells, 2013.

[205] Aditya Siddhant and Zachary C Lipton. Deep bayesian active learning for natural language processing: Results of a large-scale empirical study. *arXiv preprint arXiv:1808.05697*, 2018.

[206] Ole Sigmund. Morphology-based black and white filters for topology optimization. *Struct. Multidiscip. Optim.*, 33(4):401–424, 2007. doi:10.1007/s00158-006-0087-x.

[207] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3693–3702, 2017.

[208] Scott A. Skirlo, Ling Lu, and Marin Soljačić. Multimode one-way waveguides of large Chern numbers. *Phys. Rev. Lett.*, 113:113904, 2014. doi:10.1103/PhysRevLett.113.113904.

[209] Scott A Skirlo, Ling Lu, Yuichi Igarashi, Qinghui Yan, John Joannopoulos, and Marin Soljačić. Experimental observation of large Chern numbers in photonic crystals. *Phys. Rev. Lett.*, 115(25):253901, 2015. doi:10.1103/PhysRevLett.115.253901.

[210] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*, 25:2951–2959, 2012.

[211] Jasper Snoek, Oren Rippel, Kevin Swersky, Ryan Kiros, Nadathur Satish, Narayanan Sundaram, Mostofa Patwary, Mr Prabhat, and Ryan Adams. Scalable bayesian optimization using deep neural networks. In *International conference on machine learning*, pages 2171–2180, 2015.

[212] Zhida Song, Tiantian Zhang, and Chen Fang. Diagnosis for nonmagnetic topological semimetals in the absence of spin-orbital coupling. *Phys. Rev. X*, 8(3): 031069, 2018. doi:10.1103/PhysRevX.8.031069.

[213] Josep M Sopena, Enrique Romero, and Rene Alquezar. Neural networks with periodic and monotonic activation functions: a comparative study in classification problems. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 1, pages 323–328 vol.1. IET, 1999. doi:10.1049/cp:19991129.

[214] Jost Tobias Springenberg, Aaron Klein, Stefan Falkner, and Frank Hutter. Bayesian optimization with robust bayesian neural networks. *Advances in neural information processing systems*, 29:4134–4142, 2016.

[215] Suraj Srinivas, Akshayvarun Subramanya, and R. Venkatesh Babu. Training Sparse Neural Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 455–462. IEEE, jul 2017. ISBN 978-1-5386-0733-6. doi:10.1109/CVPRW.2017.61. URL `http://ieeexplore.ieee.org/document/8014795/`.

[216] Shengyang Sun, Guodong Zhang, Jiaxin Shi, and Roger Grosse. Functional variational bayesian neural networks. *arXiv preprint arXiv:1903.05779*, 2019.

[217] Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

[218] Guo-Jing Tang, Xin-Tao He, Fu-Long Shi, Jian-Wei Liu, Xiao-Dong Chen, and Jian-Wen Dong. Topological photonic crystals: Physics, designs, and applications. *Laser Photonics Rev.*, 16(4):2100300, 2022. doi:10.1002/lpor.202100300.

[219] Wenjie Tang and Graeme Henkelman. Charge redistribution in core-shell nanoparticles to promote oxygen reduction. *The Journal of chemical physics*, 130(19):194504, 2009.

[220] The GPyOpt authors. GPyOpt: A bayesian optimization framework in python. `http://github.com/SheffieldML/GPyOpt`, 2016.

[221] Sebastian Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[222] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[223] Andrew Trask, Felix Hill, Scott Reed, Jack Rae, Chris Dyer, and Phil Blunsom. Neural Arithmetic Logic Units. *Advances in neural information processing systems*, 31, 2018.

[224] Austin Tripp, Erik Daxberger, and José Miguel Hernández-Lobato. Sample-efficient optimization in the latent space of deep generative models via weighted retraining. *Advances in Neural Information Processing Systems*, 33, 2020.

[225] Ryan Turner, David Eriksson, Michael J. McCourt, Juha Kiili, Eero Laaksonen, Zhen Xu, and Isabelle Guyon. Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. In *NeurIPS*, 2020.

[226] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[227] Tsuyoshi Ueno, Trevor David Rhone, Zhufeng Hou, Teruyasu Mizoguchi, and Koji Tsuda. Combo: an efficient bayesian optimization library for materials science. *Materials discovery*, 4:18–21, 2016.

[228] Malika Uteuliyeva, Abylay Zhumekenov, Rustem Takhanov, Zhenisbek Assylbekov, Alejandro J Castro, and Olzhas Kabdolov. Fourier neural networks: A comparative study. *Intelligent Data Analysis*, 24(5):1107–1120, 2020.

[229] Mark Van der Wilk, Carl Edward Rasmussen, and James Hensman. Convolutional gaussian processes. *arXiv preprint arXiv:1709.01894*, 2017.

[230] D. Vanderbilt. *Berry Phases in Electronic Structure Theory.* Cambridge University Press, 2018. doi:10.1017/9781316662205.

[231] Ian Walker and Ben Glocker. Graph convolutional gaussian processes. In *International Conference on Machine Learning*, pages 6495–6504. PMLR, 2019.

[232] Dongyang Wang, Biao Yang, Qinghua Guo, Ruo-Yang Zhang, Lingbo Xia, Xiaoqiang Su, Wen-Jie Chen, Jiaguang Han, Shuang Zhang, and Che Ting Chan. Intrinsic in-plane nodal chain and generalized quaternion charge protected nodal link in photonics. *Light: Sci. Appl.*, 10(1):1–9, 2021. doi:10.1038/s41377-021-00523-8.

[233] Hai-Xiao Wang, Yige Chen, Zhi Hong Hang, Hae-Young Kee, and Jian-Hua Jiang. Type-II Dirac photons. *npj Quant. Mater.*, 2:54, 2017. doi:10.1038/s41535-017-0058-z.

[234] HaiXiao Wang, Lin Xu, HuanYang Chen, and Jian-Hua Jiang. Three-dimensional photonic Dirac points stabilized by point group symmetry. *Phys. Rev. B*, 93:235155, 2016. doi:10.1103/PhysRevB.93.235155.

[235] Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q Weinberger, and Andrew Gordon Wilson. Exact gaussian processes on a million data points. *Advances in Neural Information Processing Systems*, 32, 2019.

[236] Linnan Wang, Rodrigo Fonseca, and Yuandong Tian. Learning search space partition for black-box optimization using monte carlo tree search. *Advances in Neural Information Processing Systems*, 33:19511–19522, 2020.

[237] Zheng Wang, YD Chong, John D Joannopoulos, and Marin Soljačić. Reflection-free one-way edge modes in a gyromagnetic photonic crystal. *Phys. Rev. Lett.*, 100(1):013905, 2008. doi:10.1103/PhysRevLett.100.013905.

[238] Zheng Wang, Yidong Chong, John D Joannopoulos, and Marin Soljačić. Observation of unidirectional backscattering-immune topological electromagnetic states. *Nature*, 461(7265):772–775, 2009. doi:10.1038/nature08293.

[239] Zi Wang, Clement Gehring, Pushmeet Kohli, and Stefanie Jegelka. Batched large-scale bayesian optimization in high-dimensional spaces. In *International Conference on Artificial Intelligence and Statistics*, pages 745–754. PMLR, 2018.

[240] Haruki Watanabe and Ling Lu. Space group theory of photonic bands. *Phys. Rev. Lett.*, 121(26):263903, 2018. doi:10.1103/PhysRevLett.121.263903.

[241] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688, 2011.

[242] Yeming Wen, Dustin Tran, and Jimmy Ba. Batchensemble: an alternative approach to efficient ensemble and lifelong learning. *arXiv preprint arXiv:2002.06715*, 2020.

[243] Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin. How good is the bayes posterior in deep neural networks really? *arXiv preprint arXiv:2002.02405*, 2020.

[244] Sebastian J Wetzel, Roger G Melko, Joseph Scott, Maysum Panju, and Vijay Ganesh. Discovering symmetry invariants and conserved quantities by interpreting siamese neural networks. *Physical Review Research*, 2(3):033499, 2020.

[245] Andrew Gordon Wilson, Zhiting Hu, Ruslan Salakhutdinov, and Eric P Xing. Deep kernel learning. In *Artificial intelligence and statistics*, pages 370–378. PMLR, 2016.

[246] James Wilson, Frank Hutter, and Marc Deisenroth. Maximizing acquisition functions for bayesian optimization. *Advances in Neural Information Processing Systems*, 31:9884–9895, 2018.

[247] Wei Wu, Qinwei Fan, Jacek M. Zurada, Jian Wang, Dakun Yang, and Yan Liu. Batch gradient method with smoothing L1/2 regularization for training of feedforward neural networks. *Neural Networks*, 50:72–78, feb 2014. ISSN 0893-6080. doi:10.1016/J.NEUNET.2013.11.006. URL `https://www.sciencedirect.com/science/article/pii/S0893608013002700`.

[248] Lingbo Xia, Qinghua Guo, Biao Yang, Jiaguang Han, Chao-Xing Liu, Weili Zhang, and Shuang Zhang. Observation of hourglass nodal lines in photonics. *Phys. Rev. Lett.*, 122(10):103903, 2019. doi:10.1103/PhysRevLett.122.103903.

[249] Xin Xie, Jianchen Dang, Sai Yan, Weixuan Zhang, Huiming Hao, Shan Xiao, Shushu Shi, Zhanchun Zuo, Haiqiao Ni, Zhichuan Niu, et al. Optimization and robustness of the topological corner state in second-order topological photonic crystals. *Opt. Express*, 29(19):30735–30750, 2021. doi:10.1364/OE.438474.

[250] Hao Xu, Dongxiao Zhang, and Junsheng Zeng. Deep-learning of parametric partial differential equations from sparse and noisy data. *Physics of Fluids*, 33 (3):037132, 2021.

[251] Su-Yang Xu, Ilya Belopolski, Nasser Alidoust, Madhab Neupane, Guang Bian, Chenglong Zhang, Raman Sankar, Guoqing Chang, Zhujun Yuan, Chi-Cheng Lee, Shin-Ming Huang, Hao Zheng, Jie Ma, Daniel S. Sanchez, BaoKai Wang, Arun Bansil, Fangcheng Chou, Pavel P. Shibayev, Hsin Lin, Shuang Jia, and M. Zahid Hasan. Discovery of a Weyl fermion semimetal and topological Fermi arcs. *Science*, 349(6248):613–617, 2015. doi:10.1126/science.aaa9297.

[252] Zong-Ben Xu, Hai-Liang Guo, Yao Wang, and Hai Zhang. Representative of L1/2 Regularization among Lq ($0 < q \leq 1$) Regularizations: an Experimental Study Based on Phase Diagram. *Acta Automatica Sinica*, 38(7):1225–1228, jul 2012. ISSN 1874-1029. doi:10.1016/S1874-1029(11)60293-0. URL `https://www.sciencedirect.com/science/article/pii/S1874102911602930`.

[253] ZongBen Xu, Hai Zhang, Yao Wang, XiangYu Chang, and Yong Liang. L 1/2 regularization. *Science China Information Sciences*, 53(6):1159–1169, jun 2010. ISSN 1674-733X. doi:10.1007/s11432-010-0090-0. URL `http://link.springer.com/10.1007/s11432-010-0090-0`.

[254] Eli Yablonovitch. Inhibited spontaneous emission in solid-state physics and electronics. *Physical review letters*, 58(20):2059, 1987.

[255] Qinghui Yan, Rongjuan Liu, Zhongbo Yan, Boyuan Liu, Hongsheng Chen, Zhong Wang, and Ling Lu. Experimental discovery of nodal chains. *Nat. Phys.*, 14(5):461–464, 2018. doi:10.1038/s41567-017-0041-4.

[256] Yi Yan, Pai Liu, Xiaopeng Zhang, and Yangjun Luo. Photonic crystal topological design for polarized and polarization-independent band gaps by gradient-free topology optimization. *Optics Express*, 29(16):24861–24883, 2021.

[257] Zhenya Yan and VV Konotop. Exact solutions to three-dimensional generalized nonlinear schrödinger equations with varying potential and nonlinearities. *Physical Review E*, 80(3):036607, 2009.

[258] Erchan Yang, Biao Yang, Oubo You, Hsun-Chi Chan, Peng Mao, Qinghua Guo, Shaojie Ma, Lingbo Xia, Dianyuan Fan, Yuanjiang Xiang, et al. Observation of non-Abelian nodal links in photonics. *Phys. Rev. Lett.*, 125(3):033901, 2020. doi:10.1103/PhysRevLett.125.033901.

[259] Wanqian Yang, Lars Lorch, Moritz A Graule, Himabindu Lakkaraju, and Finale Doshi-Velez. Incorporating interpretable output constraints in bayesian neural networks. *arXiv preprint arXiv:2010.10969*, 2020.

[260] Yan Yang, Yin Poo, Rui xin Wu, Yan Gu, and Ping Chen. Experimental demonstration of one-way slow wave in waveguide involving gyromagnetic photonic crystals. *Appl. Phys. Lett.*, 102(23):231113, 2013. doi:10.1063/1.4809956.

[261] Zhi-Ming Yu, Zeying Zhang, Gui-Bin Liu, Weikang Wu, Xiao-Ping Li, Run-Wu Zhang, Shengyuan A. Yang, and Yugui Yao. Encyclopedia of emergent particles in three-dimensional crystals. *Sci. Bull.*, 67(4):375–380, 2022. doi:10.1016/j.scib.2021.10.023.

[262] Dongkun Zhang, Lu Lu, Ling Guo, and George Em Karniadakis. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. *Journal of Computational Physics*, 397:108850, 2019.

[263] Michael Zhang, Samuel Kim, Peter Y Lu, and Marin Soljačić. Deep learning and symbolic regression for discovering parametric equations. *arXiv preprint arXiv:2207.00529*, 2022.

[264] Ruqi Zhang, Chunyuan Li, Jianyi Zhang, Changyou Chen, and Andrew Gordon Wilson. Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*, 2019.

[265] Tiantian Zhang, Zhida Song, A. Alexandradinata, Hongming Weng, Chen Fang, Ling Lu, and Zhong Fang. Double-Weyl phonons in transition-metal monosilicides. *Phys. Rev. Lett.*, 120:016401, 2018. doi:10.1103/PhysRevLett.120.016401.

[266] Tiantian Zhang, Ling Lu, Shuichi Murakami, Zhong Fang, Hongming Weng, and Chen Fang. Diagnosis scheme for topological degeneracies crossing high-symmetry lines. *Phys. Rev. Research*, 2:022066, 2020. doi:10.1103/PhysRevResearch.2.022066.

[267] Tiantian Zhang, Ling Lu, Shuichi Murakami, Zhong Fang, Hongming Weng, and Chen Fang. Diagnosis scheme for topological degeneracies crossing high-symmetry lines. *Phys. Rev. Research*, 2:022066, 2020. doi:10.1103/PhysRevResearch.2.022066.

[268] Yunong Zhang, Danchi Jiang, and Jun Wang. A recurrent neural network for solving sylvester equation with time-varying coefficients. *IEEE Transactions on Neural Networks*, 13(5):1053–1063, 2002.

[269] David Zheng, Vinson Luo, Jiajun Wu, and Joshua B. Tenenbaum. Unsupervised learning of latent physical properties using perception-prediction networks. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, volume 1, pages 497–507. Association For Uncertainty in Artificial Intelligence (AUAI), 2018. ISBN 9781510871601.

[270] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[271] Liu Ziyin, Tilman Hartwig, and Masahito Ueda. Neural networks fail to learn periodic functions and how to fix it. *Advances in Neural Information Processing Systems*, 33:1583–1594, 2020.

Appendix A

EQL Network

A.1 EQL Network Details

The activation functions in each hidden layer consist of:

$$[1(\times 2), g(\times 4), g^2(\times 4), \sin(2\pi g)(\times 2), e^g(\times 2), \text{sigmoid}(20g)(\times 2), g_1 * g_2(\times 2)]$$

where the sigmoid function is defined as:

$$\text{sigmoid}(g) = \frac{1}{1+e^{-g}}$$

and the $(\times i)$ indicated the number of times each activation function is duplicated. The sin and sigmoid functions have multipliers inside so that the functions more accurately represent their respective shapes inside the input domain of $x \in [-1, 1]$. Unless otherwise stated, these are the activation functions used for the other experiments as well. The exact number of duplications is arbitrary and does not have a significant impact on the system's performance. Future work may include experimenting with a larger number of duplications.

We use two phases of training, where the first phase has a learning rate of 10^{-2} and regularization weight 5×10^{-3} for 2000 iterations. Small weights are frozen and set to 0 after the first phase. The second phase of training has a learning rate of 10^{-3}

for 10000 iterations.

To benchmark our symbolic regression system, we choose a range of trial functions that our architecture can feasibly construct, train the network through 20 trials, and count how many times it reaches the correct answer. Benchmarking results are shown in Table A.1. As mentioned in section 2.3.1, we only need the network to find the correct equation at least once since we can construct a system that automatically picks out the correct solution based on equation simplicity and test error.

Table A.1: Benchmark results for the EQL network.

	Success Rate	
Function	$L_{0.5}$	L_0
x	1	1
x^2	0.6	0.75
x^3	0.3	0.05
$\sin(2\pi x)$	0.45	0.85
xy	0.8	1
$\frac{1}{1+e^{-10x}}$	0.3	0.55
$\frac{xy}{2}+\frac{z}{2}$	0.05	0.95
$\exp(-x^2)$	0.05	0.15
$x^2+\sin(2\pi y)$	0.2	0.8
x^2+y-2z	0.6	0.9

A.2 Computational Efficiency

With respect to the task of symbolic regression, we should note that this algorithm does not offer an asymptotic speedup over conventional symbolic regression algorithms, as the problem of finding the correct expression requires a combinatorial search over the space of possible expressions and is NP-hard. Rather, the advantage here is that by solving symbolic regression problems through gradient descent, we can integrate symbolic regression with deep learning architectures.

Experiments are run on an Nvidia GTX 1080 Ti. Training the EQL network with 2 hidden layers ($L = 2$) for 20,000 epochs takes 37 seconds, and training the EQL network with 3 hidden layers ($L = 3$) takes 51 seconds.

In general, the computational complexity of the EQL network itself is the same as that of a conventional fully-connected neural network. The only difference are the activation functions which are applied by iterating over $\mathbf{g}$ and thus takes $\mathcal{O}(n)$ time where n is the number of nodes in each layer. However, the computational complexity of a neural network is dominated by the weight matrix multiplication which takes $\mathcal{O}(n^2)$ time for both the EQL network and the conventional fully-connected neural network.

A.3 Experiment Details

A.3.1 MNIST Arithmetic

The encoder network consists of a convolutional layer with 32 5×5 filters followed by a max pooling layer, another convolutional layer with 64 5×5 filters followed by a max pooling layer, and 2 fully-connected layers with 128 and 16 units each with ReLU activation units. The max pooling layers have pool size of 2 and stride length of 2. The fully-connected layers are followed by 1-unit layer with batch normalization. The output of the batch normalization layer is divided by 2 such that the standard deviation of the output is 0.5. This decreases the range of the inputs to the EQL network since the EQL network was constructed assuming an input domain of $x \in [-1, 1]$. Additionally, the output of the EQL network, $\hat{y}^*$, is scaled as $\hat{y} = 9\hat{y}^* + 9$ before being fed into the loss function so as the normalize the output against the range of expected y (this is equivalent to normalizing y to the range $[-1, 1]$).

The ReLU network that is trained in place of the EQL network for comparison consists of two hidden layers with 50 units each and ReLU activation.

We use two phases of training, where the first phases uses a learning rate of 10^{-2} and regularization weight $\lambda = 0.05$. The second phase uses a learning rate of 10^{-4} and no regularization. The small weights are frozen between the first and second phase with a threshold of $\alpha = 0.01$. Each phase is trained for 10000 iterations.

A.3.2 Kinematics

To generate the kinematics dataset, we sample 100 values for a and generate a time series $\{\mathbf{x}_t\}_{t=0}^{T_x-1}$ and $\{\mathbf{y}_t\}_{t=0}^{T_y}$ for each a. The input series is propagated for $T_x = 100$ time steps.

The dynamics encoder consists of 2 1D convolutional layers with 16 filters of length 5 in each layer. These are followed by a hidden layer with 16 nodes and ReLU activation function, an output layer with one unit, and a batch normalization layer with standard deviation 0.5. The ReLU network that is trained in place of the EQL network for comparison is the same as that of the MNIST task.

We use two phases of training, where the first phase uses a learning rate of 10^{-2} and a regularization weight of $\lambda = 10^{-3}$ for a total of 5000 iterations. The system is trained on $T_y = 1$ time step for the first 1000 iterations, and then $T_y = 5$ time steps for the remainder of the training. The small weights are frozen between the first and second phase with a threshold of $\alpha = 0.1$. The second phase uses a base learning rate of 10^{-3} and no regularization for 10000 iterations.

A.3.3 SHO

To generate the SHO dataset, we sample 1000 datapoints values for ω^2 and generate time series time series $\{\mathbf{x}_t\}_{t=0}^{T_x-1}$ and $\{\mathbf{y}_t\}_{t=0}^{T_y}$ for each ω^2. The input series is propagated for $T_x = 500$ time steps with a time step of $\Delta t = 0.1$. The output series is propagated for $T_y = 25$ time steps with the same time step.

The dynamics encoder is the same architecture as used in the kinematics experiment. Due to the greater number of time steps that the system needs to propagate, the EQL network does not duplicate the activation functions for all functions. The functions in each hidden layer consist of:

$$[1(\times 2), g(\times 2), g^2, \sin(2\pi g), e^g, 10g_1 * g_2(\times 2)]$$

The ReLU network that is trained in place of the EQL network for comparison consists of four hidden layers with 50 units each and ReLU activation functions.

We use three phases of training, where the first phase uses a learning rate of 10^{-2} and a regularization weight of $\lambda = 4 \times 10^{-5}$ for a total of 2000 iterations. The system starts training on $T_y = 1$ time steps for the first 500 time steps and then add 2 more time steps every 500 iterations for a total of $T_y = 7$ time steps. In the second phase of training, we increase the number of time steps to $T_y = 25$, decrease the base learning rate to 2×10^{-3}, and increase the regularization weight to $\lambda = 2 \times 10^{-4}$. The small weights are frozen between the second and third phase with a threshold of $\alpha = 0.01$. The third and final phase of training uses a base learning rate of 10^{-3} and no regularization.

A.4 Additional MNIST Arithmetic Data

The results presented in Figure 2-6 and Table 2.1 are drawn from one of several trials, where each in each trial the network is trained from a different random initialization of the network weights. Due to the random initialization, the EQL does not reach the same equation every time. Here we present results from additional trials to demonstrate the variability in the system's behavior as well as the system's robustness to the random initializations.

The experimental details are described in Section 2.3.2 where digits $\chi_{1,2}$ are drawn from the entire MNIST training dataset. We refer to the results shown in Figure 2-6 and Table 2.1 as Trial 1.

The results for Trial 2 are shown in Figure A-1. Similar to Trial 1, Trial 2 produces a linear relationship between the true digit ϕ and the latent variable z, although there is a positive instead of negative correlation. As previously mentioned, there is no bias placed on the latent variable z so whether there is a positive or negative correlation is arbitrary and depends on the random initialization of the weights. The trained architecture produced the following expression from the EQL network:

$$\hat{y} = 1.565 z_1 + 1.558 z_2 + 9 \tag{A.1}$$

Note the positive coefficients in (A.1) which reflects the positive correlation shown in

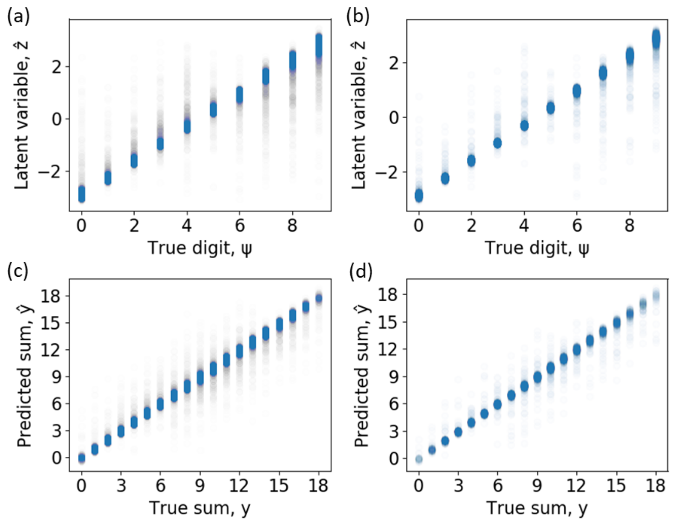


Figure A-1: The ability of the encoder to differentiate between digits as measured by the latent variable z versus the true digit ψ for digits χ drawn from the MNIST (a) training dataset and (b) test dataset. The ability of the entire architecture to fit the label y as measured by the predicted sum $\hat{y}$ versus the true sum y for digits χ drawn from the MNIST (c) training dataset and (d) MNIST test dataset.

Figure A-1(a-b). As shown in Figure A-1(c-d), the network is still able to accurately predict the sum y.

The results for Trial 3 are shown in Figure A-2. Note that in this case, the relationship between ϕ and z is no longer linear. However, the encoder still finds a one-to-one mapping between ϕ and z, and the EQL network is still able to extract the information from z such that it can predict the correct sum as shown in Figure A-2(c-d).

The equation found by the EQL network is:

$$\hat{y} = -4.64\sin(2.22z_1) - 4.63\sin(2.21z_2) + 9 \tag{A.2}$$

This is consistent with the insight that the curve in Figure A-2(a-b) represents an inverse sine function. Thus, (A.2) is first inverting the transformation from ϕ to z to produce a linear mapping and then adding the two digits together. So while the EQL network does not always give the exact equation we expect, we can still gain insight into the system from analyzing the latent variable and the resulting equation.

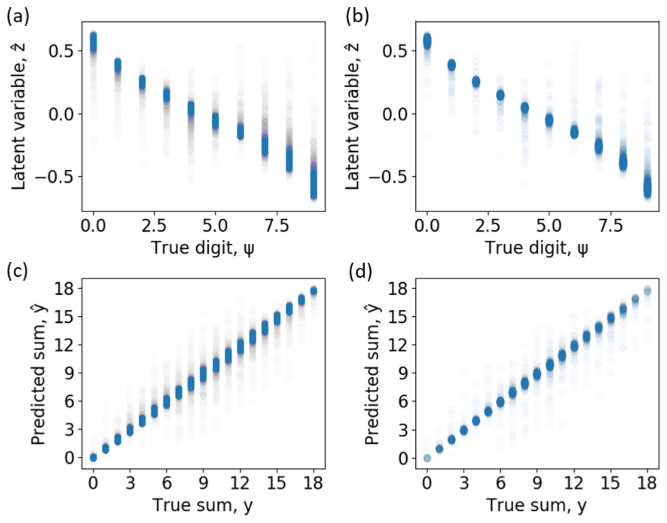


Figure A-2: The ability of the encoder to differentiate between digits as measured by the latent variable z versus the true digit ψ for digits χ drawn from the MNIST (a) training dataset and (b) test dataset. The ability of the entire architecture to fit the label y as measured by the predicted sum $\hat{y}$ versus the true sum y for digits χ drawn from the MNIST (c) training dataset and (d) MNIST test dataset.

Appendix B

Parametric EQL Network

B.1 Training Details

Each of the SEQL and HEQL consists of 2 hidden layers. The activation functions in each hidden layer consist of:

$$[1(\times 2), g(\times 4), g^2(\times 4), \sin(2\pi g)(\times 2), g_1 * g_2(\times 2)]$$

where the $(\times i)$ indicates the number of times each activation function is duplicated. The sin function has a multiplier inside so that the functions more accurately represent their respective shapes inside the input domain of $x \in [-1, 1]$. The exact number of duplications is arbitrary and does not have a significant impact on the system's performance.

For the HEQL, the MWU consists of a fully-connected neural network with 3 hidden layers of 64, 64, and 256 hidden units, respectively. The hidden layers in the MWU use the ReLU function as the activation.

All neural network architectures are implemented in Tensorflow [3]. The network is trained using the RMSProp optimizer, and the following loss function:

$$\mathcal{L} = \frac{1}{N} \sum (\hat{y}_i - y_i)^2 + \lambda L_r, \tag{B.1}$$

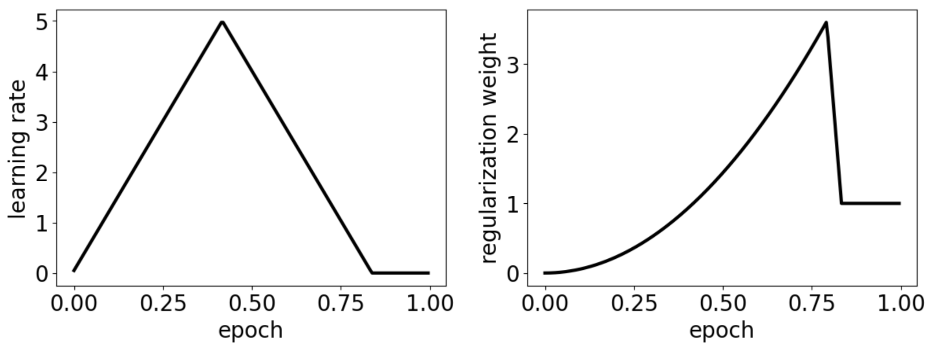


Figure B-1: (Left) Learning rate and (right) regularization weight schedules during training relative to `base_lr` and `base_rw`.

where N is the mini-batch size, λ is the regularization weight, and L_r is the total regularization. For the parameterized architecture $L_r = \mathcal{L}_R$ is simply the sparsity regularization, while the stacked architecture has an additional term $L_r = \mathcal{L}_R + \theta\mathcal{L}_S$ to incorporate the smooth weight regularization described in Section 2.6.1.

For both learning rate and regularization weight schedules, we use a one cycle policy, as shown in Figure B-1. We start off with a small learning rate and regularization to ensure the EQL network settles into a stable configuration containing many different terms such that the network weights do not explode. The learning rate is ramped up to allow the EQL network escape local minima in search of global minima, and the regularization is likewise increased to pare down the number of terms. Finally, we expect the EQL network to have learned the correct equation structure partway through training, and so we decrease learning rate and regularization to fine-tune the weights and optimize primarily for MSE.

To extract the learned equation from the trained EQL network, we can simply multiply the weights by the primitive functions using symbolic mathematics. We implement this using SymPy, which can automatically simplify the expression [149]. Additionally, we use a threshholding procedure in the final expression where we drop terms where the coefficient is smaller than a threshhold, which we set to 0.01.

Table B.1: Results for analytic expression benchmarks.

Benchmark	Mean (Standard Deviation) Test MSE over all trials	
	SEQL	HEQL
f_1	$\mathbf{1.84 \times 10^{-5}}$ $(\mathbf{1.97 \times 10^{-5}})$	2.24×10^{-5} (2.53×10^{-6})
f_2	1.32×10^{-4} (4.14×10^{-4})	$\mathbf{4.82 \times 10^{-8}}$ $(\mathbf{1.26 \times 10^{-8}})$
f_3	$\mathbf{5.72 \times 10^{-15}}$ $(\mathbf{3.83 \times 10^{-16}})$	6.60×10^{-6} $(1.45 \times 10^{-6}$
f_4	$\mathbf{8.32 \times 10^{-5}}$ $(\mathbf{4.74 \times 10^{-4}})$	1.04×10^{-3} (6.04×10^{-3})
f_5	2.53×10^{-5} (1.51×10^{-4})	$\mathbf{4.87 \times 10^{-8}}$ $(\mathbf{2.29 \times 10^{-8}})$

B.2 Additional Results

B.2.1 Analytic Expression

For all tests, 512 training data points with $x \in [-3, 3]$ are sampled for each of 128 fixed values of $t \in [-3, 3]$ for a total of $512 \cdot 128 = 65\,536$ training examples. To test generalization, the parametric EQL architectures are evaluated on test data points with $x \in [-5, 5]$.

Due to sensitivity of the parametric EQL architectures to the random initialization of network weights, 80 trials were run for each function. In practice, the networks only need to learn the correct equation once over a reasonable number of trials, since it is possible to construct a validation method that selects the best equation from a set of learned equations. For all the results in this paper, we simply select the trial with the lowest generalization error. Other considerations that can be integrated in the validation process are equation simplicity and prior beliefs about the equation form, for example.

Additional metrics, the mean and standard deviation of the test MSE over all the trials, are listed for the analytic benchmarks in Table B.1. Comparing these results with the MSE of the best trial listed in Table 2.8, we see taht the aggregate metrics over all the trials tend to be similar in magnitude to the metric of the best trial for many of the benchmarks, which signifies that the model has learned the correct equation is a large majority of the trials. When a model fails to learn the correct equation, the MSE on the test dataset tends to be several orders of magnitude larger than that of the best trial, which would skew the mean and standard deviation of the MSE. Interestingly, there is no clear trend on whether the SEQL or the HEQL performs better.

For simple benchmarks such as $f_3 = tx$, both the SEQL and HEQL architectures are able to find the correct equation structure nearly 100% of the time, even if the accuracy of the varying coefficients may vary slightly. However, in other cases such as f_4, the HEQL will sometimes learn the equation:

$$\hat{f}_{4,HEQL} = a(t)x^2 + b(t)x + c(t)\sin(d(t)x + e(t))$$

where $d(t)$ is small. This is likely because the architecture is using the approximately linear region of the low-frequency sinusoid, and adding it to the $b(t)x$ term. We also note that the SEQL is able to find the correct equation more often. Another interesting failure mode is in the case of the sinusoid functions (i.e., f_2 and f_5) where the HEQL will somtimes learn the equation:

$$\hat{f}_{5,HEQL} = a(t)\sin(b(t)x) + c(t)\sin(d(t)x)$$

where $b(t) \approx d(t)$ and $a(t) + c(t) \approx 1$. The symbolic manipulation is unable to combine the two terms, but one can see upon inspection that the HEQL has learned the correct form of the varying parameters.

B.2.2 Differential Equations

For the advection-diffusion equation, data was sampled from 256 different points in the x-domain and 512 different points in the t-domain, for a total of $256 \cdot 512 = 131\,072$ examples. The equation is solved numerically using a spectral method on the domain $x \in [-5, 5]$ and $t \in [0, 5]$ with $f(x) = -1.5 + \cos\left(\frac{2\pi x}{5}\right)$ and $\epsilon = 0.1$ using code from Ref. [192].

For the Burgers' equation, data was sampled from 512 different points in the x-domain and 256 different points in the t-domain, for a total of $512 \cdot 256 = 131\,072$ examples. The equation was solved numerically using a spectral method on the

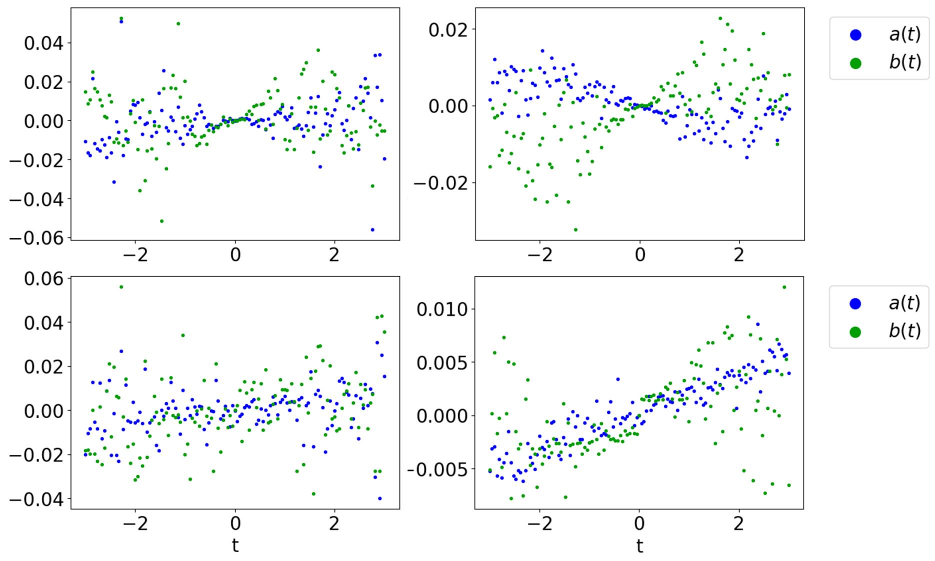


Figure B-2: Prediction errors of the parametric coefficients for the (left) HEQL and the (right) SEQL on the analytical expressions (top) f_4 and (bottom) f_1.

domain $x \in [-8, 8]$ and $t \in [0, 10]$ using code from Ref. [192]. Similar to the analytic expression experiments, 80 trials were run for each equation and the trial with the lowest generalization error was selected.

Figure B-3 displays results for SEQL on the advection-diffusion equation, and Figure B-4 displays results for HEQL on Burgers' equation. We see that for each dataset, the results are visually similar that of the contrasting architecture (i.e. SEQL versus HEQL).

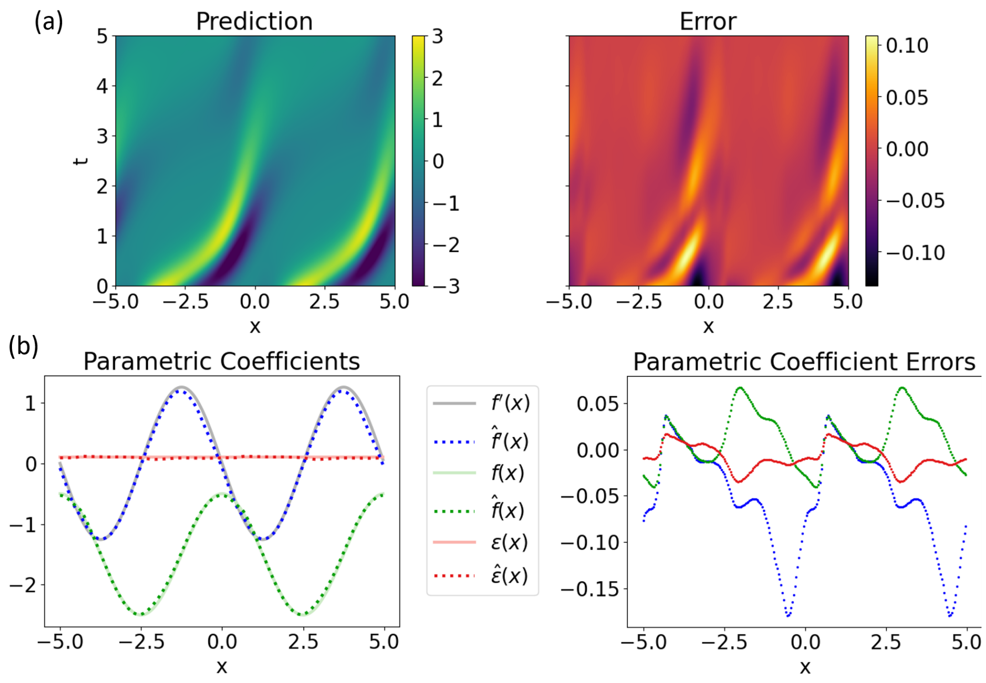


Figure B-3: Results for learning the advection-diffusion equation using the SEQL network. (a) Predicted vs. actual values of u_t. (b) Predicted coefficient functions and prediction errors.

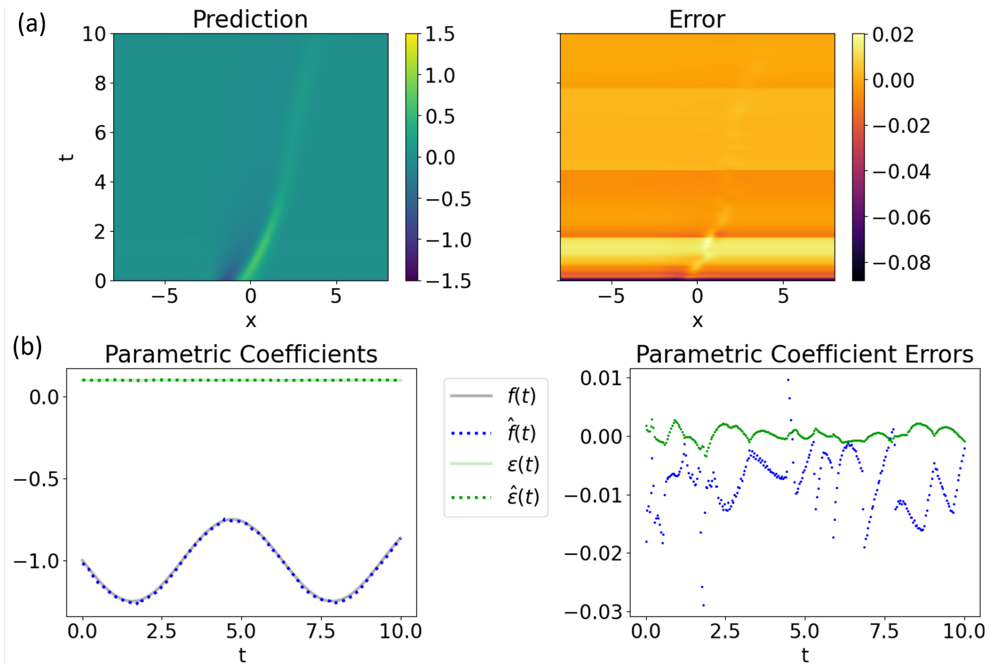


Figure B-4: Results for learning Burgers' equation using the HEQL network. (a) Predicted vs. actual values of u_t. (b) Predicted coefficient functions and prediction errors.

Appendix C

Bayesian Optimization

C.1 Datasets

The dimensionalities of the datasets are summarized in table C.1. The continuous input dimension for chemical molecules refers to the SOAP descriptor. While the space of chemical molecule graphs in general do not have a well-defined dimensionality as chemical molecules can be arbitrarily large and complex, we limit the size of molecules by only sampling from the QM9 dataset, and can define the dimensionality as the sum of the adjacency, node, and edge matrix dimensionalities.

The high dimensionalities of all of these problems make Bayesian neural networks well-suited as surrogate models to enable scaling. Note that the nanoparticle scattering problem can be adjusted to be less or more difficult by either changing the input dimensionality (i.e. the number of nanoparticle layers) or the auxiliary dimension

Table C.1: Summary of dataset dimensionalities. Note that alternate inputs for photonic crystal and organic molecule datasets are binary images and molecule graphs, respectively.

	Continuous input dimension	Alternate input dimension	Auxiliary dimension
Nanoparticle scattering	6	N/A	201
Photonic crystal DOS	51	$32 \times 32 = 1024$	500
Chemical molecule	480	$9 + 9 \times 9 + 9 \times 9 = 171$	9

(i.e. the resolution or range of wavelengths that are sampled).

C.1.1 Nanoparticle Scattering

The multilayer nanoparticle consists of a lossless silica core surrounded by alternating spherical layers of lossless TiO_2 and lossless silica. The relative permittivity of silica is $\varepsilon_{\text{silica}} = 2.04$. The relative permittivity of TiO_2 is dispersive and depends on the wavelength of light:

$$\varepsilon_{\text{TiO}_2} = 5.913 + \frac{0.2441}{10^{-6}\lambda^2 - 0.0803} \tag{C.1}$$

where λ is the wavelength given in units of nm. The entire particle is surrounded by water, which has a relative permittivity of $\varepsilon_{\text{water}} = 1.77$.

For a given set of thicknesses, we analytically solve for the scattering spectrum, i.e. the scattering cross-section $\sigma(\lambda)$ as a function of wavelength λ, using Mie scattering as described in Qiu et al. [179]. The code for computing σ was adapted from Peurifoy et al. [172].

The objective functions for the narrowband and highpass objectives are:

$$h_{\text{nb}}(\mathbf{z}) = \frac{\int_{\lambda\in\text{nb}} \sigma(\lambda)\, d\lambda}{\int_{\text{elsewhere}} \sigma(\lambda)\, d\lambda} \approx \frac{\sum_{i=126}^{145} z_i}{\sum_{i=1}^{125} z_i + \sum_{i=146}^{201} z_i} \tag{C.2}$$

$$h_{\text{hp}}(\mathbf{z}) = \frac{\int_{\lambda\in\text{hp}} \sigma(\lambda)\, d\lambda}{\int_{\text{elsewhere}} \sigma(\lambda)\, d\lambda} \approx \frac{\sum_{i=126}^{201} z_i}{\sum_{i=1}^{125} z_i} \tag{C.3}$$

where $\mathbf{z} \in \mathbb{R}^{201}$ is the discretized scattering cross-section $\sigma(\lambda)$ from $\lambda = 350\,\text{nm}$ to $750\,\text{nm}$.

C.1.2 Photonic Crystal

The photonic crystal (PC) consists of periodic unit cells with periodicity $a = 1\,\text{au}$, where each unit cell is depicted as a "two-tone" image, with the white regions representing silicon with permittivity $\varepsilon_1 = 11.4$ and black regions representing vacuum (or air) with permittivity $\varepsilon_0 = 1$.

The photonic crystal (PC) structure is defined by a spatially varying permittivity

$\varepsilon(x, y) \in \{\varepsilon_0, \varepsilon_1\}$ over a 2D periodic unit cell with spatial coordinates $x, y \in [0, a]$. To parameterize ε, we choose a level set of a Fourier sum function ϕ, defined as a linear combination of plane waves with frequencies evenly spaced in the reciprocal lattice space up to a maximum cutoff. Intuitively, the upper limit on the frequencies roughly corresponds to a lower limit on the feature size such that the photonic crystal remains within reasonable fabrication constraints. Here we set the cutoff such that there are 25 complex frequencies corresponding to 50 real coefficients $\mathbf{c} = (c_1, c_2, ..., c_{50})$.

Explicitly, we have

$$\phi[\mathbf{c}](x, y) = \mathrm{Re}\left\{ \sum_{k=1}^{25} (c_k + i c_{k+25})\, e^{2\pi i (n_x x + n_y y)/a} \right\}, \tag{C.4}$$

where each exponential term is composed from the 25 different pairs $\{n_x, n_y\}$ with $n_x, n_y \in \{-2, -1, 0, 1, 2\}$. We then choose a level-set offset Δ to determine the PC structure, where regions with $\phi > \Delta$ are assigned to be silicon and regions where $\phi \leq \Delta$ are vacuum. Thus, the photonic crystal unit cell topology is parameterized by a 51-dimensional vector, $[c_1, c_2, ..., c_{50}, \Delta] \in \mathbb{R}^{51}$. More specifically,

$$\varepsilon(x, y) = \varepsilon[\mathbf{c}, \Delta](x, y) = \begin{cases} \varepsilon_1 & \phi[\mathbf{c}](x, y) > \Delta \\ \varepsilon_0 & \phi[\mathbf{c}](x, y) \leq \Delta \end{cases}, \tag{C.5}$$

which is discretized to result in a 32×32 pixel image $\mathbf{v} \in \{\varepsilon_0, \varepsilon_1\}^{32\times 32}$. This formulation also has the advantage of enforcing periodic boundary conditions.

For each unit cell, we use the MIT Photonics Bands (MPB) software [92] to compute the band structure of the photonic crystal, $\omega(\mathbf{k})$, up to the lowest 10 bands, using a 32×32 spatial resolution (or equivalently, 32×32 k-points over the Brillouin zone $-\frac{\pi}{a} < k < \frac{\pi}{a}$). We also extract the group velocities at each k-point and compute the density-of-states (DOS) via an extrapolative technique, adapted from Liu et al. [118]. The DOS is computed at a resolution of 20,000 points, and a Gaussian filter of kernel size 100 is used to smooth the DOS spectrum. To normalize the frequency scale across the different unit cells, the frequency is rescaled via $\omega\sqrt{\varepsilon_{avg}} \to \omega_{norm}$, where $\varepsilon_{avg} = \frac{1}{a^2}\int_0^a \int_0^a \varepsilon(x,y)\,dx\,dy \approx \frac{1}{(32)^2}\sum_{i,j} v_{i,j}$ is the average permittivity over all pixels. Finally, the DOS spectrum is truncated at $\omega_{norm} = 1.2$ and interpolated using 500 points to give $\mathbf{z} \in \mathbb{R}^{500}$.

The objective function aims to minimize the DOS in a small frequency range and maximize it elsewhere. We use the following:

$$h_{\mathrm{DOS}}(\mathbf{z}) = \frac{\sum_{i=1}^{300} z_i + \sum_{i=351}^{500} z_i}{1 + \sum_{i=301}^{350} z_i}, \tag{C.6}$$

where the 1 is added in the denominator to avoid singular values.

C.1.3 Organic Molecule Quantum Chemistry

The Smooth Overlap of Atomic Positions (SOAP) descriptor [33] uses smoothed atomic densities to describe local environments for each atom in the molecule through a fixed-length feature vector, which can then be averaged over all the atoms in the molecule to produce a fixed-length feature vector for the molecule. This descriptor is invariant to translations, rotations, and permutations. We use the SOAP descriptor implemented by DScribe [72] using the parameters: local cutoff `rcut` = 5, number of radial basis functions `nmax` = 3, and maximum degree of spherical harmonics `lmax` = 3. We use `outer` averaging, which averages over the power spectrum of different sites.

The graph representation of each molecule is processed by the Spektral package [59]. Each graph is represented by a node feature matrix $\mathbf{X} \in \mathbb{R}^{s \times d_n}$, an adjacency matrix $\mathbf{A} \in \mathbb{R}^{s \times s}$, and an edge matrix $\mathbf{E} \in \mathbb{R}^{e \times d_e}$, where s is the number of atoms in the molecule, e is the number of bonds, and d_n, d_e are the number of features for nodes and edges, respectively.

The properties that we use from the QM9 dataset are listed in Table C.2. We separate these properties into two categories: (1) the *ground state quantities* which are calculated from a single DFT calculation of the molecule and include geometric, energetic, and electronic quantities, and (2) the *thermodynamic quantities* which are typically calculated from a molecular dynamics simulation.

Table C.2: List of properties from the QM9 dataset used as labels

Property	Unit	Description
Ground State Quantities		
A	GHz	Rotational constant
B	GHz	Rotational constant
C	GHz	Rotational constant
μ	D	Dipole moment
α	a_0^3	Isotropic polarizability
ϵ_{HOMO}	Ha	Energy of HOMO
ϵ_{LUMO}	Ha	Energy of LUMO
ϵ_{GAP}	Ha	Gap ($\epsilon_{\text{LUMO}} - \epsilon_{\text{HOMO}}$)
$\langle R^2 \rangle$	a_0^2	Electronic spatial extent
Thermodynamic Quantities at 298.15 K		
U	Ha	Internal energy
H	Ha	Enthalpy
G	Ha	Free energy
C_V	$\frac{\text{cal}}{\text{mol K}}$	Heat capacity

The auxiliary information for this task consist of the properties listed in Table C.2 that are in the same category as the objective property, as these properties would be calculated together. The objective function then simply picks out the corresponding feature from the auxiliary information. More precisely, for the ground state objectives, the auxiliary information is

$$\mathbf{z} = \left[A, B, C, \mu, \alpha, \epsilon_{\text{HOMO}}, \epsilon_{\text{LUMO}}, \epsilon_{\text{gap}}, \langle R^2 \rangle\right] \in \mathbb{R}^9,$$

and the objective functions are

$$\begin{aligned} h_\alpha(\mathbf{z}) &= z_5 \\ h_{\alpha-\epsilon_{\text{gap}}}(\mathbf{z}) &= \frac{z_5 - 6}{191} - \frac{z_8 - 0.02}{0.6} \end{aligned}$$

where the quantities for the latter objective are normalized so that they have the same magnitude.

Algorithm 1 Bayesian optimization with auxiliary information

```
1: Input: Labelled dataset $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_n, \mathbf{z}_n, y_n)\}_{n=1}^{N_{\text{start}}=5}$
2: for $N = 5$ to $1000$ do
3:     Train $\mathcal{M}\colon \mathcal{X} \to \mathcal{Z}$ on $\mathcal{D}_{\text{train}}$
4:     Form an unlabelled dataset, $\mathcal{X}_{\text{pool}}$
5:     Find $\mathbf{x}_{N+1} = \arg\max_{\mathbf{x}\in\mathcal{X}_{\text{pool}}} \alpha\left(\mathbf{x}; \mathcal{M}, \mathcal{D}_{\text{train}}\right)$
6:     Label the data $\mathbf{z}_{N+1} = g(\mathbf{x}_{N+1})$, $y_{N+1} = h(\mathbf{z}_{N+1})$
7:     $\mathcal{D}_{\text{train}} = \mathcal{D}_{\text{train}} \cup \left(\mathbf{x}_{N+1}, \mathbf{z}_{N+1}, y_{N+1}\right)$
8: end for
```

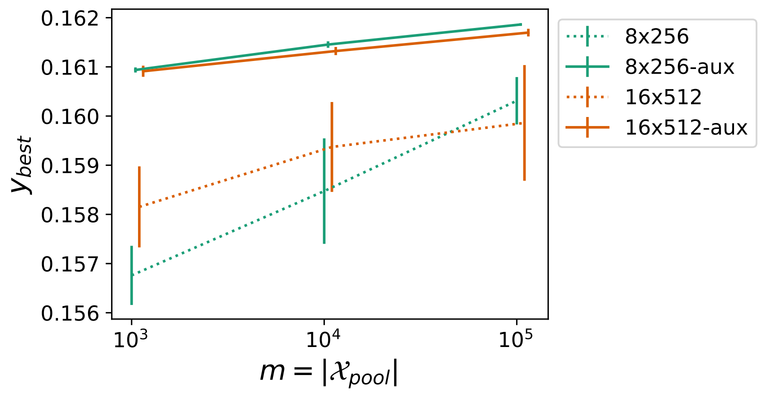


Figure C-1: Effect of $m = |\mathcal{X}_{\text{pool}}|$ used in the inner optimization loop to maximize the acquisition function on overall BO performance. y_{best} is taken from the narrowband objective function using the ensemble architecture. The "aux" in the legend denotes using auxiliary information and the numbers represent the architecture (i.e. 8 layers of 256 units or 16 layers of 512 units).

C.2 Bayesian Optimization and Acquisition Function

Our algorithm for Bayesian optimization using auxiliary information $\mathbf{z}$ is shown in Algorithm 1. This algorithm reduces to the basic BO algorithm in the case where h is the identity function and $\mathcal{Z} = \mathcal{Y}$ such that we can ignore mention of $\mathbf{z}$ in Algorithm 1.

As mentioned in the main text, the inner optimization loop in line 5 of Algorithm 1 is performed by finding the maximum value of α over a pool of $|\mathcal{X}_{\text{pool}}|$ randomly sampled points. We can see in Figure C-1 that increasing $|\mathcal{X}_{\text{pool}}|$ in the acquisition step

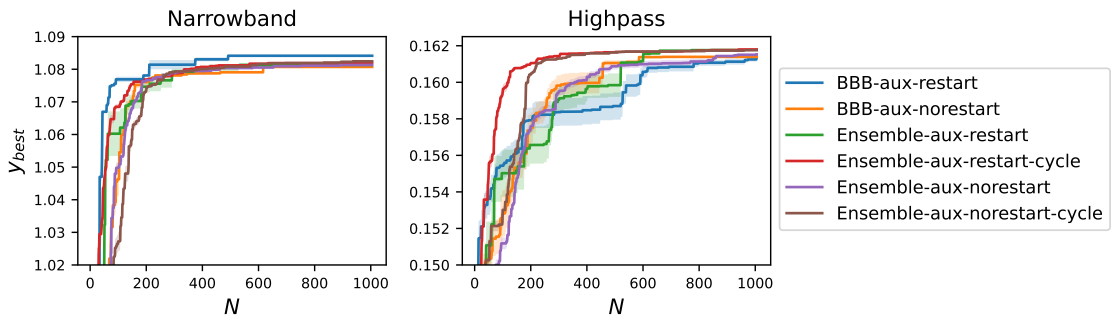


Figure C-2: Effect of restarting the BNN training from scratch in each BO iteration.

tends to improve BO performance. Thus, there is likely further room for improvement of the inner optimization loop using more sophisticated algorithms, possibly using the gradient information provided by BNNs. Unless otherwise stated, we optimize the inner loop of Bayesian optimization to choose the next data point to label by maximizing EI on a pool of $|\mathcal{X}_{\text{pool}}| = 10^5$ randomly sampled points.

C.3 Continued Training

As mentioned in Section 3.3.3 of the main text, the BNN is ideally trained from scratch until convergence in each iteration loop, although this comes at a great computational cost. An alternative is the warm restart method of continuing the training from the previous iteration which enables the model's training loss to converge in only a few epochs. However, as shown in Figure C-2, we have found that naive continued training can result in poor BO performance. This is likely because (a) training does not converge for the new data point $\mathcal{D}_{\text{new}} = (\mathbf{x}_{N+1}, y_{N+1})$ relative to the rest of the data under a limited computational budget, resulting in the acquisition function possibly labeling similar points in consecutive iterations, and (b) the BNN gets trapped in a local minima in the loss landscape that is not ideal for learning future data points. To mitigate this, we use the cosine annealing learning rate proposed in Loshchilov and Hutter [127]. The large learning rate at the start of training allows the model to more easily escape local minima and explore a multimodal posterior [78], while the small learning rate towards the end of the annealing cycle allows the model to converge more easily. Note that the idea of warm restart is similar to “continual learning,” which is an open and active sub-problem in machine learning research [221, 167]. In particular, we re-train the BNN using 10 epochs.

C.4 Models and Hyperparameters

C.4.1 Additional Surrogate Models

Variational BNNs model a prior and posterior distribution over the neural network weights, but use some approximation on the distributions to make the BNN tractable. In particular, we use Bayes by Backprop (BBB) (also referred to as the “mean field” approximation), which approximates the posterior over the neural network weights with independent normal distributions [14]. We also compare Multiplicative Normalizing Flows (MNF), which uses normalizing flows on top of each layer output for more expressive posterior distributions [128].

BOHAMIANN proposed to use BNNs in BO by using stochastic gradient Hamiltonian Monte Carlo (SGHMC) to approximately sample the BNN posterior, combined with scale adaptation to adapt it for an iterative setting [214].

NeuralLinear trains a conventional neural network on the data, but then replaces the last layer with Bayesian linear regression such that the neural network serves as an adaptive basis for the linear regression [211].

TuRBO (trust region Bayesian Optimization) is a method that maintains M trust regions and performs Bayesian optimization within each trust region, maintaining M local surrogate models, to scale BO to high-dimensional problems that require thousands of observations [41]. We use $M = 1$ and $M = 5$, labelled as “TuRBO-1” and “TuRBO-5”, respectively.

TPE (Tree Parzen Estimator) is a method that instead of modeling $p(y|x)$, models $p(x|y)$ and $p(y)$ for the surrogate model and fits into the BO framework [13]. The tree-structure of the surrogate model allows it to define leaf variables only when node variables take particular values, which makes it well-suited for hyper-parameter search (e.g. the learning rate momentum is only defined for momentum-based gradient descent methods).

LIPO is a parameter-free algorithm that assumes the underlying function is a Lipschitz function and estimates the bounds of the function [142]. We use the implementation provided by the dlib library [102].

DIRECT-L (DIviding RECTangles-Local) systematically divides the search domain into smaller and smaller hyperrectangles to efficiently search the space [48]. We use the implementation provided by the NLopt library [90].

CMA-ES (covariance matrix adaptation evolution strategy) is an evolutionary algorithm that samples new data based on a multivariate normal distribution and refines the parameters of this distribution until reaching convergence. We us the implementation provided by the pycma library [67].

C.4.2 Implementation Details

Unless otherwise stated, we set $N_{\text{MC}} = 30$. All BNNs other than the infinitely-wide networks are implemented in TensorFlow v1. Models are trained using the Adam optimizer using the cosine annealing learning rate with a base learning rate of 10^{-3} [127]. All hidden layers use ReLU as the activation function, and no activation function is applied to the output layer.

Infinite-width neural networks are implemented using the Neural Tangents library [159]. We use two different types of infinite networks: (1) "GP-" refers to a closed form expression for Gaussian process inference using the infinite-width neural network as a kernel, and (2) "INF-" refers to an infinite ensemble of infinite-width networks that have been "trained" with continuous gradient descent for an infinite time. We compare NNGP and NTK kernels as well as the parameterization of the layers. By default, we use the NTK parameterization, but we also use the standard parameterization, denoted by "-STD".

We implement BO using GPs with a Matérn kernel using the GPyOpt library [220]. The library optimizes over the acquisition function in the inner loop using the L-BFGS algorithm.

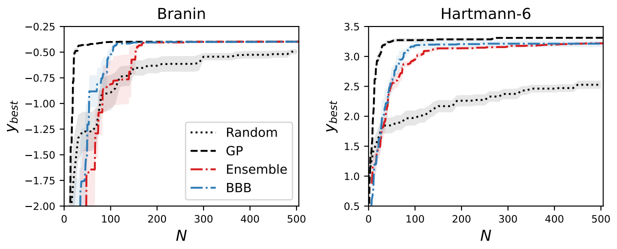


Figure C-3: BO results for the Branin and Hartmann-6 functions.

LIPO [142] is implemented in the dlib library [102]. DIRECT-L [48] is implemented in the NLopt library [90]. CMA-ES is implemented in the pycma library [67].

C.5 Additional Results

C.5.1 Test Functions

We test BO on several common synthetic functions used for optimization, namely the Branin and 6-dimensional Hartmann functions. We use BNNs with 4 hidden layers and 256 units in each hidden layer, where each hidden layer is followed by a ReLU activation function. Plots of the best value y_{best} at each BO iteration are shown in Figure C-3. As expected, GPs perform the best. Ensembles and BBB also perform competitively and much better than random sampling, showing that deep BO is viable even for low-dimensional black-box functions.

C.5.2 Nanoparticle Scattering

Detailed BO results for the nanoparticle scattering problem are shown in Table C.3.

Table C.3: BO results for the nanoparticle scattering task. * denotes that y_{best} is measured at $N = 100$ due to computational constraints

Model	Narrowband				Highpass			
	y_{best} at $N = 250$		y_{best} at $N = 1000$		y_{best} at $N = 250$		y_{best} at $N = 100$	
	Mean	SE	Mean	SE	Mean	SE	Mean	SE
GP	0.1606	0.0005	0.1621	0.0001	1.0839	0.0017	1.0851	0.0008
GP-aux	*0.1541	0.0019	-	-	*1.0110	0.0234	-	-
Ensemble	0.1558	0.0011	0.1607	0.0003	1.0729	0.0025	1.077	0.0021
Ensemble-aux	0.1578	0.0014	0.1593	0.0013	1.0783	0.0003	1.0822	0.001
BBB	0.1596	0.0006	0.1596	0.0006	1.0753	0.0005	1.0753	0.0005
BBB-aux	0.1601	0.001	0.1601	0.001	1.076	0.0028	1.076	0.0028
BBB-Anneal	0.1598	0.001	0.1611	0.0001	1.0813	0.0003	1.0821	0.0005
BBB-aux-Anneal	0.1613	0.0003	0.1619	0	1.0826	0.0008	1.0834	0.0005
MNF	0.15	0.0005	0.1547	0.0004	1.027	0.005	1.0312	0.0036
MNF-aux	0.1549	0.0014	0.1569	0.0006	0.9957	0.0168	1.028	0.0157
Neural Linear	0.1543	0.002	0.1579	0.0015	1.0798	0.0007	1.0836	0.0007
BOHAMIANN	0.1616	0.0001	-	-	1.0717	0.0031	-	-
Inf-NNGP	0.1541	0.0011	0.157	0.0009	1.055	0.0036	1.0653	0.0022
Inf-NTK	0.1536	0.0008	0.1571	0.001	1.041	0.004	1.0612	0.0011
Inf-NNGP-std	0.1551	0.0006	0.1598	0.0006	1.0615	0.0043	1.069	0.0018
Inf-NTK-std	0.1564	0.0006	0.1607	0.0001	1.0607	0.0039	1.0761	0.0014
GP-NNGP	0.1582	0.0007	0.1609	0.0001	1.0621	0.0027	1.0694	0.0019
GP-NTK	0.1573	0.001	0.1611	0.0001	1.0667	0.0032	1.0732	0.0012
GP-NNGP-std	0.1562	0.0008	0.1595	0.001	1.0615	0.0058	1.0718	0.0024
GP-NTK-std	0.1592	0.0011	0.1608	0.0002	1.0641	0.0033	1.0704	0.0017
TuRBO-1	0.1572	0.0017	0.1619	0.0011	1.0831	0.022	1.0871	0.0005
TuRBO-1	0.1605	0.0011	0.1619	0.0001	1.0867	0.0016	1.0890	0.0003
TPE	0.1561	0.0007	0.1615	0.0001	1.0517	0.0035	1.0794	0.0010
Random	0.1527	0.0008	0.1555	0.0006	1.0053	0.0063	1.0362	0.0047
LIPO	0.1604	0.0016	0.1619	0.0006	1.0792	0.0066	1.087	0.0034
DIRECT-L	0.1544	0	0.156	0	1.0777	0	1.0801	0
CMA	0.1424	0.0046	0.143	0.0048	1.059	0.0117	1.076	0.0127

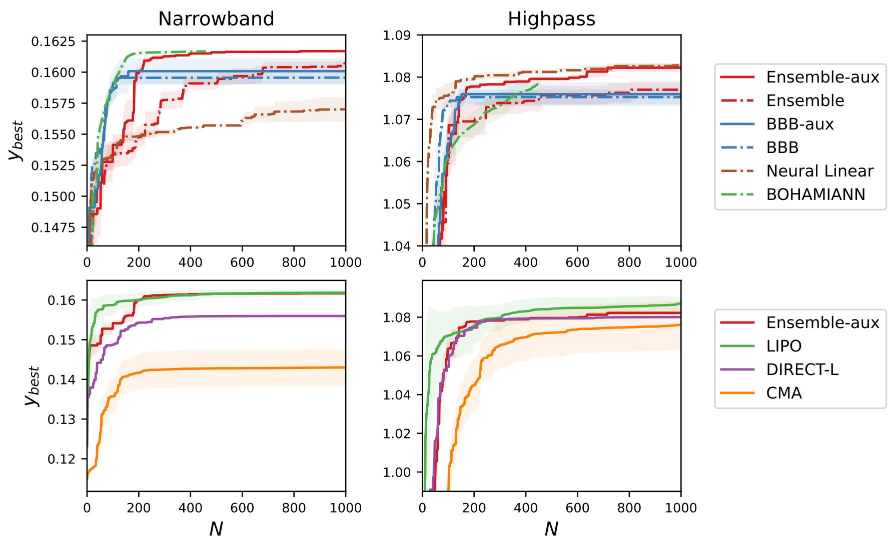


Figure C-4: Additional optimization result curves for the nanoparticle scattering task. (Top) Various BNNs. Note that results using auxiliary information are denoted by a solid line, while those that do not are denoted by a dashed line. Also note that the y-axis is zoomed in to differentiate the curves. (Bottom) Various non-BO algorithms. ENSEMBLE-AUX is replicated here for ease of comparison.

All the BNNs used for the nanoparticle scattering problem use an architecture consisting of 8 hidden layers with 256 units each, with the exception of BOHAMIANN where we used the original architecture consisting of 2 hidden layers with 50 units each. The infinite-width neural networks for the nanoparticle task consist of 8 hidden layers of infinite width, each of which are followed by ReLU activation functions.

We also experiment with KL annealing in BBB, a proposed method to improve the performance of variational methods for BNNs in which the weight of the KL term in the loss function is slowly increased throughout training [243]. For these experiments, we exponentially anneal the KL term with weight $\sigma_{KL}(i) = 10^{i/500-5}$ as a function of epoch i when training from scratch; during the continued training, the weight is held constant at $\sigma_{KL} = 10^{-3}$.

KL annealing in the BBB architecture significantly improves performance for the narrowband objective, although results are mixed for the highpass objective. Additionally, KL annealing has the downside of introducing more parameters that must be carefully tuned for optimal performance. MNF performs poorly, especially on the highpass objective where it is comparable to random sampling, and we have found

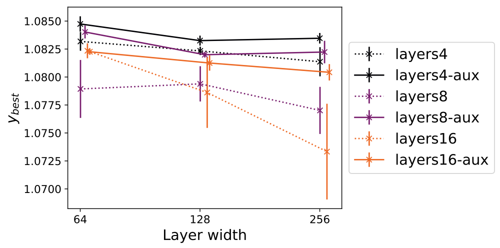


Figure C-5: Comparison of y_{best} at $N = 1000$ for the nanoparticle narrowband objective function for a variety of neural network sizes. All results are ensembles, and "aux" denotes using auxiliary information.

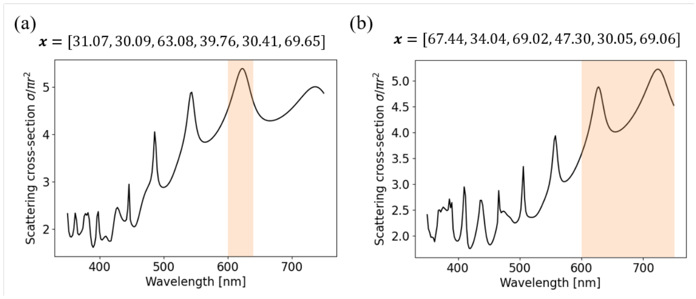


Figure C-6: Examples of optimized nanoparticles and their scattering spectrum using the "Ensemble-aux" architecture for the (a) narrowband and (c) highpass objectives. Orange shaded regions mark the range over which we wish to maximize the scattering.

that MNF is quite sensitive to the choice of hyperparameters for uncertainty estimates even on simple regression problems.

The different variants infinite-width neural networks do not perform as well as the BNNs on both objective functions, despite the hyper-parameter search.

LIPO seems to perform as well as GPs on both objective functions, which is impressive given the computational speed of the LIPO algorithm. Interestingly DIRECT-L does not perform as well as LIPO or GPs on the narrowband objective, and actually performs comparably to random sampling on the highpass objective. Additionally, CMA performs poorly on both objectives, likely due to the highly multimodal nature of the objective function landscape.

We also look at the effect of model size in terms of number of layers and units in Figure C-5 for ensembles. While including auxiliary information clearly improves performance across all architectures, there is not a clear trend of performance with

Table C.4: Various architectures for BNNs and BCNNs used in the PC problem. Numbers represent the number of channels and units for the convolutional and fully-connected layers, respectively. All convolutional layers use 3×3-sized filters with stride $(1, 1)$ and periodic boundaries. "MP" denotes max-pooling layers of size 2×2 with stride $(2, 2)$, and "AP" denotes average-pooling layers of size 2×2 with stride $(1, 1)$. "Conv" denotes BCNNs whereas "FC" denotes BNNs (containing only fully-connected layers) that act on the level-set parameterization $\mathbf{x}$ rather than on the image $\mathbf{v}$. "TI" denotes translation invariant architectures, whereas "TD" denotes translation dependent architectures (i.e. not translation invariant).

Architecture	Convolutional Layers	Fully-connected Layers
Conv-TI	16-MP-32-MP-64-MP-128-MP-256	256-256-256-256
Conv-TD	8-AP-8-MP-16-AP-32-MP-32-AP	256-256-256-256
FC	*n/a*	256-256-256-256-256

respect to the model size. Thus, the performance of BO seems to be somewhat robust to the exact architecture as long as the model is large enough to accurately and efficiently train on the data.

Examples of the optimized structures by the "Ensemble-aux" architecture are shown in Figure C-6. We can see that the scattering spectra peak in the shaded region of interest, as desired by the respective objective functions.

C.5.3 Photonic Crystal

The BNN and BCNN architectures that we use for the PC task are listed in Table C.4. The size of the "FC" architectures are chosen to have a similar number of parameters as their convolutional counterparts. Unless otherwise stated, all results in the main text and here use the "Conv-TI" and "FC" architectures for BCNNs and BNNs, respectively.

Table C.5: Select BO results for the PC problem. * denotes that y_{best} is measured at $N = 130$ due to computational constraints. † denotes that y_{best} is measured at $N = 750$ due to computational constraints.

Model	PC-A				PC-B			
	y_{best} at $N = 250$		y_{best} at $N = 1000$		y_{best} at $N = 250$		y_{best} at $N = 100$	
	Mean	SE	Mean	SE	Mean	SE	Mean	SE
GP	548	450	2109	448	781	394	3502	49
GP-aux	*16	4	-	-	*9	1	-	-
Ensemble	30	2	841	448	216	145	1318	465
Ensemble-aux	305	217	1310	509	2909	408	3633	130
ConvEnsemble	1140	471	2375	371	390	263	2070	505
ConvEnsemble-aux	2623	558	3468	120	3752	106	4002	92
BBB	75	31	350	207	704	502	780	485
BBB-aux	39	7	413	313	554	371	1605	544
ConvBBB	712	416	1486	490	928	600	930	599
ConvBBB-aux	2109	583	3124	43	1761	724	1928	711
NeuralLinear	1009	549	1235	481	685	488	2853	291
ConvNeuralLinear	1160	540	2524	479	1643	596	2722	647
Conv-Inf-NNGP	29	8	322	181	21	7	157	42
Conv-Inf-NTK	49	32	425	322	28	7	907	711
Conv-GP-NNGP	15	2	221	118	37	5	830	533
Conv-GP-NTK	20	10	194	139	34	12	85	45
Conv-Inf-NNGP-std	17	3	732	432	66	15	889	442
Conv-Inf-NTK-std	52	31	99	64	8	0	27	12
Conv-GP-NNGP-std	20	7	†101	59	100	55	†124	49
Conv-GP-NTK-std	13	5	†132	77	7	0	†686	575
Random	141	61	402	184	471	398	485	395
TuRBO-1	1150	638	4451	20	3865	289	4476	16
TuRBO-5	3738	92	4456	37	4128	49	4466	26
TPE	1001	648	3901	140	3045	571	4119	156
LIPO	940	1073	1280	1073	1837	1792	2266	1626
DIRECT-L	20	0	4351	1	8	0	2525	38
CMA	9	1	4078	126	10	3	1777	969

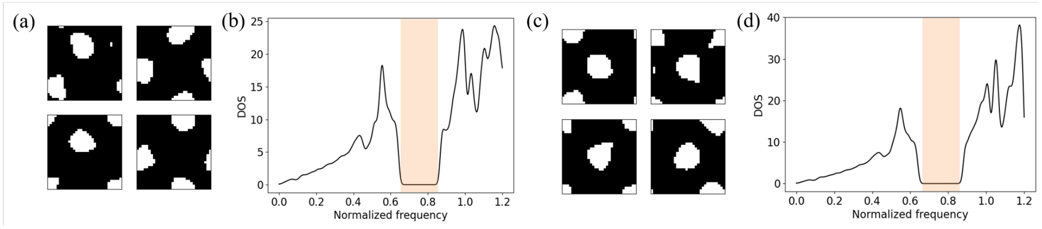


Figure C-7: Examples of optimized photonic crystal unit cells over multiple trials for (a) PC-A distribution and (c) PC-B distribution. (b,d) Examples of the optimized DOS. Note that the DOS has been minimized to nearly zero in a thin frequency range. Orange shaded regions mark the frequency range in which we wish to minimize the DOS. All results were optimized by the "Ensemble-aux" architecture.

The infinite-width convolutional neural networks (which act as convolutional kernels for GPs) in the PC task consist of 5 convolutional layers followed by 4 fully-connected layers of infinite width. Because the pooling layers in the Neural Tangents library are currently too slow for use in application, we increased the size of the filters to 5×5 to increase the receptive field of each filter.

Detailed BO results for the PC problem are shown in Table C.5. For algorithms that optimize over the level set parameterization $\mathbb{R}^{51}$, we see that GPs perform consistently well, although BNNs using auxiliary information (e.g. Ensemble-Aux) can outperform GPs. DIRECT-L and CMA perform extremely well on the PC-A distribution but performs worse than GP on the PC-B distribution.

Adding convolutional layers and auxiliary information improves performance such that BCNNs significantly outperform GPs. Interestingly, the infinite-width networks perform extremely poorly, although this may be due to a lack of pooling layers in their architecture which limits the receptive field of the convolutions.

Examples of the optimized structures by the "Ensemble-aux" architecture are shown in Figure C-7. The photonic crystal unit cells generally converged to the same shape: a square lattice of silicon posts with periodicity $\sqrt{2}a$.

Validation Metrics

To explore more deeply why certain surrogate models perform well while others do not, we track various metrics of the model during BO on a validation dataset with 1000 randomly sampled data points. In particular, we look at the mean squared error

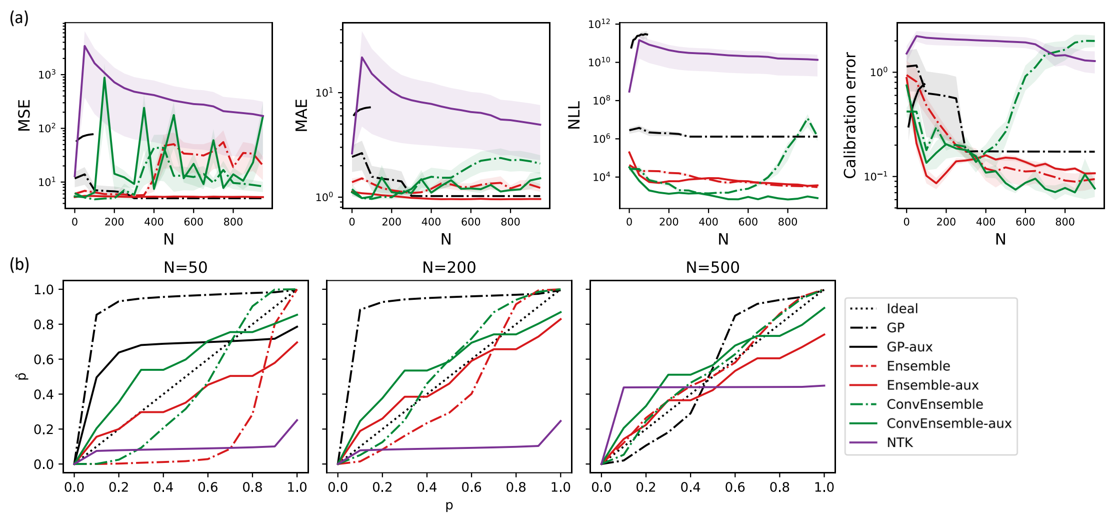


Figure C-8: (a) Various metrics tracked during BO of the PC-A dataset distribution on a validation dataset of 1000 datapoints. (b) Uncertainty calibration curves measured at various points during BO. Note that the calibration curve for GP-AUX is only shown for $N = 50$, as it becomes computationally intractable for larger N.

(MSE), the mean absolute error (MAE), the negative log-likelihood (NLL), and the calibration error on the PC-A data distribution. Results are shown in Figure C-8(a).

The calibration error is a quantitative measure of the uncertainty of the model, which is important for the performance of BO as the acquisition function uses the uncertainty to balance exploration and exploitation. Intuitively, we expect that a 50% confidence interval contains the correct answer 50% of the time. In particular, we use the forecast calibration as proposed by [109], which is an extension of the calibration error proposed by [61] to regression tasks:

$$\mathrm{cal}(F_1, y_1, ..., F_T, y_T) = \sum_{j=1}^{m} (p_j - \hat{p}_j)^2$$

where F_j is the CDF of the predictive distribution, p_j is the confidence level, and $\hat{p}_j$ is the empirical frequency. We choose to measure the error along the confidence levels $p_j = (j-1)/10$ for $j = 1, 2, ..., 11$. The CDF $F_j(y_j)$ an be analytically calculated for models that have an analytical predictive distribution. For models that do not have an analytical predictive distribution, we use the empirical CDF:

$$F(y) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}_{\mu^{(i)} \leq y}$$

where $\mathbb{1}$ is the indicator function. We also plot the calibration, $\{(p_j, \hat{p}_j)\}_{j=1}^{M}$, in Figure C-8(b). Perfectly calibrated predictions correspond to a straight line.

Figure C-8 shows that the infinite neural network kernel (NTK) has the highest prediction error, which is likely a contributing factor to its poor BO performance. Interestingly, vanilla GPs have the lowest MSE, so the prediction error is not the only indicator for BO performance. Looking at the calibration, the infinite neural network kernel has the highest calibration error, and we see from Figure C-8(b) that it tends to be overconfident in its predictions. GPs have a higher calibration error than the ensemble neural network methods, and tends to be significantly underconfident in its predictions. GP-aux has higher validation loss, calibration error, and NLL than most, if not all, of the other methods, which explain its poor performance.

The ensemble NN methods tend to be reasonably well-calibrated. Within the ensemble NNs, the "-aux" methods have lower MSE and calibration error than their respective counterparts, and ConvEnsemble-aux has the lowest NLL calibration error out of all the methods, although interestingly Ensemble-aux seems to have the lowest MSE and MAE out of the ensemble NNs.

These results together show that calibration of Bayesian models is extremely important for use as surrogate models in BO.

Table C.6: BO results for the four different quantum chemistry objective functions. * denotes that y_{best} is measured at $N = 100$ due to computational constraints.

Model	y_{best} at $N = 500$							
	ϵ_{GAP}		$-\epsilon_{\text{GAP}}$		α		$\alpha - \epsilon_{\text{GAP}}$	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
GP	0.41	0.04	−0.10	0.02	101.08	1.05	0.29	0.07
GraphGP	*0.62	0.00	* − 0.10	0.02	*131.99	14.59	*0.24	0.03
Ensemble	0.62	0.00	−0.08	0.00	86.56	0.31	0.28	0.00
Ensemble-aux	0.62	0.00	−0.10	0.02	83.86	4.45	0.13	0.05
GraphEnsemble	0.62	0.00	−0.10	0.00	143.53	0.00	0.49	0.00
GraphEnsemble-aux	0.62	0.00	−0.10	0.00	143.53	0.00	0.49	0.00
GraphBBB	0.38	0.01	−0.11	0.01	94.46	1.16	0.25	0.01
GraphBBB-FC	0.62	0.00	−0.10	0.00	135.64	13.67	0.39	0.14
GraphNeuralLinear	0.62	0.00	−0.09	0.01	143.53	0.00	0.46	0.09
VAE-GP	0.62	0.06	-	-	123.33	13.02	0.61	0.34
VAE-GP-2	-	-	-	-	110.84	16.68	0.56	0.35
VAE-GP-latent128	-	-	-	-	154.66	35.96	0.40	0.10
VAE-GP-latent128-beta0.001	-	-	-	-	133.66	13.25	0.42	0.13
VAE-GP-latent32	-	-	-	-	114.83	14.64	0.53	0.38
Random	0.38	0.02	−0.11	0.03	105.19	7.87	0.29	0.07

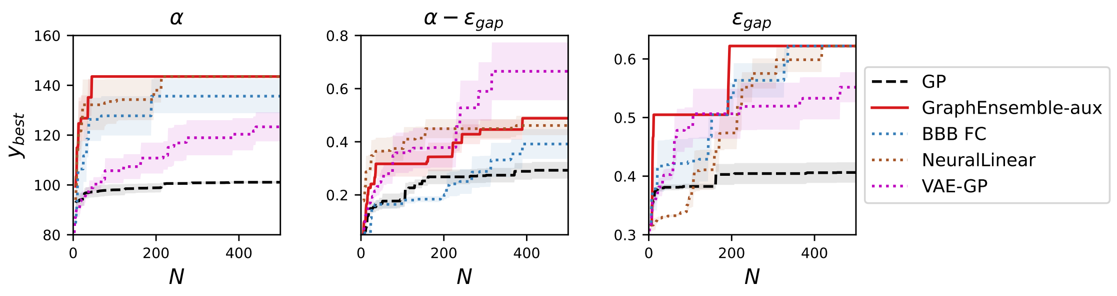


Figure C-9: Additional BO results for several different objective functions on the chemistry dataset. GP and GraphEnsemble-aux curves are replicated from the main text for convenience.

C.5.4 Organic Molecule Quantum Chemistry

The Bayesian graph neural networks (BGNNs) used for the chemical property optimization task consist of 4 edge-conditioned graph convolutional layers with 32 channels each, followed by a global average pooling operation, followed by 4 fully-connected hidden layers of 64 units each. The edge-conditioned graph convolutional layers [207] are implemented by Spektral [59].

More detailed results for the quantum chemistry dataset are shown in Table C.6 and Figure C-9. The architecture with the Bayes by Backprop variational approximation applied to every layer including the graph convolutional layers ("BBB"), performs extremely poorly, even worse than random sampling in some cases. However, only making the fully-connected layers Bayesian ("BBB-FC") performs surprisingly well.

Ensembles trained with auxiliary information ("Ensemble-aux") and neural linear ("NeuralLinear") perform the best on all objective functions. Adding auxiliary information to ensembles helps for the α objective function, and neither helps nor hurts for the other objective functions. Additionally, BNNs perform at least as well or significantly better than GPs in all cases. GPs perform comparably or worse than random sampling in several cases.

As noted in the main text, the performance of VAE-GP depends on the quality of the pre-trained VAE, as shown in Figure C-10. The VAE-GP benchmark uses the same pre-trained VAE, and "VAE-GP-2" refers to the same method using a different random seed for the VAE. Even with the exact same method, VAE-GP-2 performs

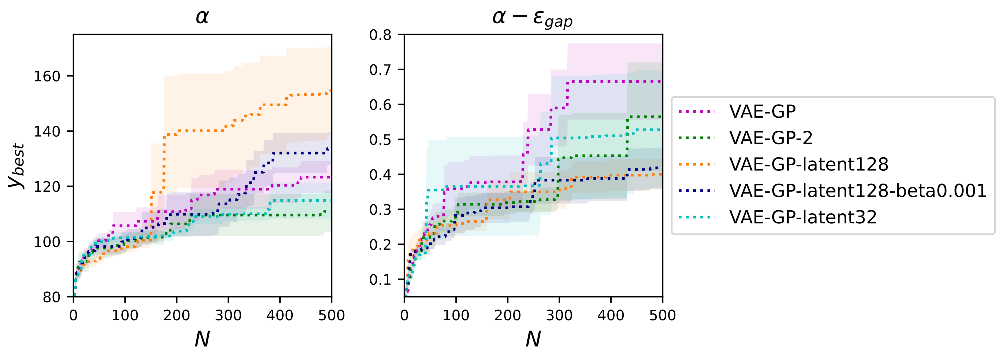


Figure C-10: Additional BO results for VAE-GP using different pre-trained VAEs.

significantly worse on both objective functions. We also increase the latent space dimensionality from 52 to 128 in the "VAE-GP-LATENT128" benchmark, which performs even worse on the $\alpha - \epsilon_{\text{gap}}$ benchmark although it performs significantly better on the α benchmark. We also adjust the learning rate momentum to $\beta = 0.001$ in "VAE-GP-LATENT128-BETA0.001", and the latent space dimensionality to 32 in "VAE-GP-LATENT32". There is no clear trend with the different hyper-parameters, which may point to the random seed of the VAE pre-training being a greater factor in BO performance than the hyper-parameters.

Validation Metrics

As in Appendix C.5.3, we track the MSE, NLL, and calibration error during optimization on the chemistry task. Results are shown in Figure C-11. The various metrics correlate with the respective methods' peformances during BO. For example, VAE-GP has an extremely high MSE and calibration error on the α objective, where it performs poorly, but has an MSE and calibration error more comparable with that of other methods as well as an extremely low NLL on the $\alpha - \epsilon_{\text{gap}}$ objective, where it performs extremely well. Likewise, the metrics for GRAPHGP are very high on the $\alpha - \epsilon_{\text{gap}}$ objective, where it performs poorly. GraphEnsemble tends to be among the better methods in terms of these metrics, which translates into good BO performance.

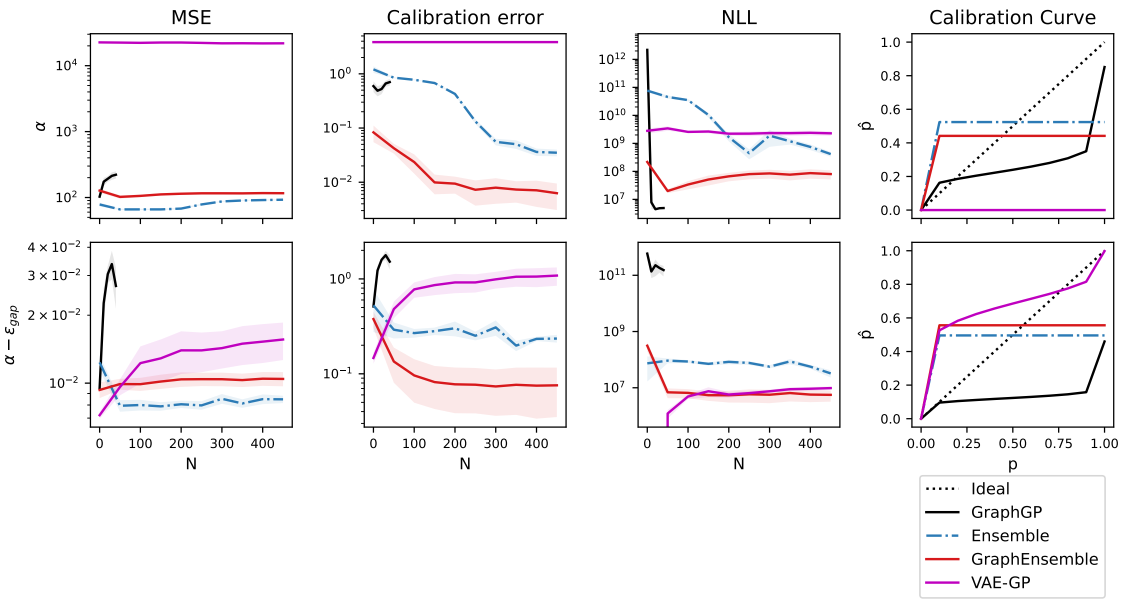


Figure C-11: (a) Various metrics tracked during BO of the PC-A dataset distribution on a validation dataset of 1000 datapoints. (b) Uncertainty calibration curves measured at various points during BO.

C.5.5 Additional Discussion

BBB performs reasonably well and is competitive with or even better than ensembles on some tasks, but it requires significant hyperparameter tuning. The tendency of variational methods such as BBB to underestimate uncertainty is likely detrimental to their performance in BO. Additionally, Sun et al. [216] shows that BBB has trouble scaling to larger network sizes, which may make them unsuitable for more complex tasks such as those in our work. BOHAMIANN performs very well on the nanoparticle narrowband objective and comparable to other BNNs without auxiliary information on the nanoparticle highpass objective. This is likely due to its effectiveness in exploring a multi-modal posterior. However, the need for SGHMC to sample the posterior makes this method computationally expensive, and so we were only able to run it for a limited number of iterations using a small neural network architecture.

Infinitely wide neural networks are another interesting research direction, as the ability to derive infinitely wide versions of various neural network architectures such as convolutions, and more recently graph convolutional layers [77] could potentially bring the power of GPs and BO to complex problems in low-data regimes. However, we find they perform relatively poorly in BO, are quite sensitive to hyperparameters (e.g. kernel and parameterization), and current implementations of certain operations such as pooling are too slow for practical use in an iterative setting. In particular, BO using an infinite ensemble of infinite-width networks performs poorly compared to normal ensembles, suggesting that the infinite-width formulations do not fully capture the dynamics of their finite-width counterparts.

Non-Bayesian global optimization methods such as LIPO and DIRECT-L are quite powerful in spite of their small computational overhead and can even outperform BO on some simpler tasks. However, they are not as consistent as BO, performing more comparably to random sampling on other tasks. CMA-ES performs poorly on all the tasks here. Also, like GPs, these non-Bayesian algorithms assume a continuous input space and cannot be effectively applied to structured, high-dimensional problems.

C.6 Compute

All experiments were carried out on systems with NVIDIA Volta V100 GPUs and Intel Xeon Gold 6248 CPUs. All training and inference using neural network-based models, graph kernels, and infinite-width neural network approximations are carried out on the GPUs. All other models are carried out on the CPUs.

Appendix D

Topological Photonic Crystal Optimization

D.1 Abbreviations and notation

For convenience, commonly used acronyms and mathematical notation in Chapter 4 are summarized below.

Abbreviation	**Word**
PhC	photonic crystal
SDP	semi-definite programming
BZ	Brillouin zone
SG	space group
HS	high-symmetry
DOS	density of states

Notation	**Quantity**
ε	permittivity
$\mathbf{r}$	spatial coordinate
$\mathbf{k}$	reciprocal lattice coordinate
ϕ	level-set function
$\mathbf{G}$	reciprocal lattice vectors
$\mathbf{R}$	lattice vectors
a, b, c	lattice vector lengths
α, β, γ	lattice angles
$\omega_{n\mathbf{k}}$	band frequency
$L(\omega_{n\mathbf{k}})$	objective function
$\boldsymbol{\nu}$	symmetry indicator with $v_j \in \mathbb{Z}_{\lambda_j}$
C, $\mathbf{C}$	Chern number, Chern vector