CoPhy-PGNN: Learning Physics-guided Neural Networks with Competing Loss Functions for Solving Eigenvalue Problems

Mohannad Elhamod*† Jie Bu*† Christopher Singh‡ Matthew Redell‡

Abantika Ghosh§ Viktor Podolskiy§ Wei-Cheng Lee‡ Anuj Karpatne†

Abstract

Physics-guided Neural Networks (PGNNs) represent an emerging class of neural networks that are trained using physics-guided (PG) loss functions (capturing violations in network outputs with known physics), along with the supervision contained in data. Existing work in PGNNs have demonstrated the efficacy of adding single PG loss functions in the neural network objectives, using constant trade-off parameters, to ensure better generalizability. However, in the presence of multiple physics loss functions with competing gradient directions, there is a need to *adaptively* tune the contribution of competing PG loss functions during the course of training to arrive at generalizable solutions. We demonstrate the presence of competing PG losses in the generic neural network problem of solving for the lowest (or highest) eigenvector of a physics-based eigenvalue equation, common to many scientific problems. We present a novel approach to handle competing PG losses and demonstrate its efficacy in learning generalizable solutions in two motivating applications of quantum mechanics and electromagnetic propagation. All the code and data used in this work is available at https://github.com/jayroxis/Cophy-PGNN.

Keywords: PGML, ML, Quantum physics, Ising model, Electromagnetic propagation

1 Introduction

With the increasing impact of deep learning methods in diverse scientific disciplines [1, 2], there is a growing realization in the scientific community to harness the power of artificial neural networks (ANNs) without ignoring the rich supervision available in the form of physics knowledge in several scientific problems [3, 4]. One of the promising lines of research in this direction is to modify the objective function of neural networks by adding loss functions that measure the violations of ANN outputs with physical equations, termed as *physics-guided (PG) loss functions* [5, 6]. By anchoring ANN models to be consistent with physics, PG loss functions have been shown to impart generalizability even in the paucity of training data across several scientific problems [7–10]. We refer to the class of neural networks that are trained using PG loss functions as physics-guided neural networks (PGNNs).

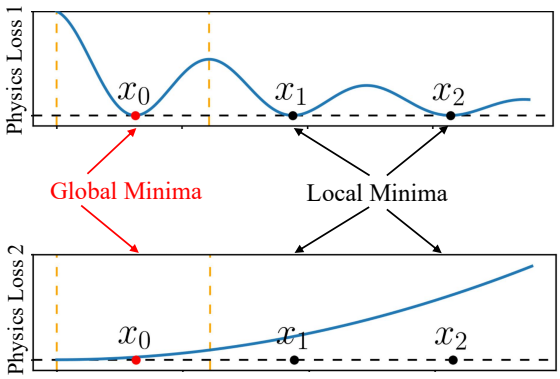

Figure 1: A toy example showing competing physics losses that can lead to local minima when minimized together.

While some existing work in PGNN have attempted to learn neural networks by solely minimizing PG loss (and thus being label-free) [6, 9], others have used both PG loss and data label loss using appropriate trade-off hyper-parameters [7, 8]. However, what is even more challenging is when there are multiple physics equations with *competing* PG loss functions that need to be min-

*equal contribution
†Department of Computer Science, Virginia Tech
‡Department of Physics, Binghamton University
§Department of Physics and Applied Physics, University of Massachusetts Lowell

imized together, where each PG loss may show multiple local minima. In such situations, simple addition of PG losses in the objective function with constant trade-off hyper-parameters may result in the learning of non-generalizable solutions. This may seem counter-intuitive since the addition of PG loss is generally assumed to offer generalizability in the PGNN literature [8,10,11].

Figure 1 shows a toy example with two competing physics losses to illustrate their effects on the generalizability of learned solutions. If we add the two loss functions with constant weights and optimize the weighted sum using gradient descent methods, it is easy to end up at local minima $\mathbf{x_1}$ and $\mathbf{x_2}$. However, if we pay importance to physics loss 2 in the first few epochs of gradient descent, and then optimize physics loss 1 at later epochs, we are more likely to arrive at the global minimum $\mathbf{x_0}$. This simple observation, although derived from an artificially constructed toy problem, motivates us to ask the question: *is it possible to adaptively balance the importance of competing PG loss functions at different stages of neural network learning to arrive at generalizable solutions?*

In this work, we introduce a novel framework of *CoPhy*-PGNN, which is an abbreviation for *Competing Physics* Physics-Guided Neural Networks, to handle competing PG loss functions in neural network training. We specifically consider the domain of scientific problems where physics knowledge are represented as eigenvalue equations and we are required to solve for the highest or lowest eigen-solution. This representation is common to many types of physics such as the Schrödinger equation in the domain of quantum mechanics and Maxwell's equations in the domain of electromagnetic propagation. In these applications, solving eigenvalue equations using exact numerical techniques (e.g., diagonalization methods) can be computationally expensive especially for large physical systems. On the other hand, PGNN models, once trained, can be applied on testing scenarios to predict their eigen-solutions in drastically smaller running times. We empirically demonstrate the efficacy of our *CoPhy*-PGNN solution on two diverse applications in quantum mechanics and electromagnetic propagation, highlighting the generalizability of our proposed approach to many physics problems.

The remainder of the paper is organized as follows. Section 2 provides background of our physics problems and describes related work. Section 3 presents our proposed approach. Section 4 describes evaluation setup while Sections 5 and 6 present our results and concluding remarks, respectively.

2 Background

2.1 Overview of Physics Problems: While our target applications of quantum mechanics and electromagnetic propagation are from quite distinct domains, they share a common property that the physics of the problem is avaialable in the form of an eigen-value equation of the form: $\hat{A}\boldsymbol{y} = b\boldsymbol{y}$, where, for a given input matrix $\hat{A}$, b is an eigenvalue and $\boldsymbol{y}$ is the corresponding eigenvector. We are interested in solving the lowest or highest eigen-solution of this equation in our target problems. We provide a brief overview of the two target applications in the following (detailed overviews are provided in the supplementary materials).

Quantum Mechanics: In this application, the goal is to predict the ground-state wave function of an Ising chain model with $n = 4$ particles. This problem can be described by the Schrödinger equation $\mathbf{H}\hat{\Psi} = \hat{E}\hat{\Psi}$, where $\hat{E}$, the energy level, is the eigenvalue; $\hat{\Psi}$, the wave function, is the eigenvector, and $\mathbf{H}$, the Hamiltonian, is the matrix. Since the ground-state weave function corresponds to the lowest energy level, we are interested in finding the lowest eigen-solution of this eigen-value equation.

Electromagnetic Propagation: To illustrate our model's scalability to large systems, we consider another application involving the propagation of the electromagnetic waves in periodically stratified layer stacks. The description of this propagation can be reduced to the eigenvalue problem $\hat{A}\vec{h}_m = k_z^{m^2} h_m$ where $k_z^{m^2}$, the propagation constant of the electromagnetic modes along the layers, is the eigenvalue; and $\vec{h}_m$, the coefficients of the Fourier transform of the spatial profile of the electromagnetic field, is the eigenvector. It is important to note that these quantities are complex valued, unlike in the target application of quantum mechanics. Also, unlike the quantum mechanics problem, we are interested in the largest eigenvalue rather than the smallest.

2.2 Related work in PGNN: PGNN has found successful applications in several disciplines including fluid dynamics [12–14], climate science [10], and lake modeling [7,8,15]. However, to the best of our knowledge, PGNN formulations have not been explored yet for our target applications of solving eigen-value equations in the field of quantum mechanics and electromagnetic propagation. Existing work in PGNN can be broadly divided into two categories. The first category involves label-free learning by only minimizing PG loss without using any labeled data. For example, Physics-informed neural networks (PINNs) and its variants [9,16,17] have been recently developed to solve PDEs by solely minimizing PG loss functions, for simple canonical problems such as Burger's equation. Since

these methods are label-free, they do not explore the interplay between PG loss and label loss. We consider an analogue of PINN for our target application as a baseline in our experiments.

The second category of methods incorporate PG loss as additional terms in the objective function along with label loss, using constant trade-off hyper-parameters. This includes work in basic Physics-guided Neural Networks (PGNNs) [7, 8] for the target application of lake temperature modeling. We use an analogue of this basic PGNN as a baseline in our experiments.

While some recent works have investigated the effects of PG loss on generalization performance [11] and the importance of normalizing the scale of hyper-parameters corresponding to PG loss terms [18], they do not study the effects of competing physics losses which is the focus of this paper. Our work is related to the field of multi-task learning (MTL) [19], as the minimization of physics losses and label loss can be viewed as multiple shared tasks. For example, alternating minimization techniques in MTL [20] in MTL can be used to alternate between minimizing different PG loss and label loss terms over different mini-batches. We consider this as a baseline approach in our experiments.

3 Methodology

3.1 Problem statement: From an ML perspective, we are given a collection of training pairs, $\mathcal{D}_{Tr} := \{\hat{A}_i, (\boldsymbol{y}_i, b_i)\}_{i=1}^{N}$, where $(\boldsymbol{y}_i, b_i)$ is generated by diagonalization solvers. We consider the problem of learning an ANN model, $(\hat{\boldsymbol{y}}, \hat{b}) = f_{NN}(\hat{A}, \theta)$, that can predict $(\boldsymbol{y}, b)$ for any input matrix, $\hat{A}$, where θ are the learnable parameters of ANN. We are also given a set of unlabeled examples, $\mathcal{D}_U := \{\hat{A}_i\}_{i=1}^{M}$, which will be used for testing. We consider a simple feed-forward architecture of f_{NN} in all our formulations.

3.2 Designing physics-guided loss functions: A naïve approach for learning f_{NN} is to minimize the mean sum of squared errors (MSE) of predictions on the training set, referred to as the Train-MSE(θ) := $1/N(\sum_i ||\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i||^2 + ||\hat{b}_i - b_i||^2)$. However, instead of solely relying on Train-MSE, we consider the following PG loss terms to guide the learning of f_{NN} to generalizable solutions:

Characteristic Loss: A fundamental equation we want to satisfy in our predictions, $(\hat{\boldsymbol{y}}, \hat{b})$, for any input $\hat{A}$ is the eigen-value equation, $\hat{A}\hat{\boldsymbol{y}} = \hat{b}\hat{\boldsymbol{y}}$. Hence, we consider minimizing the following equation:

$$C\text{-Loss}(\theta) := \sum_i \frac{||\hat{A}_i\hat{\boldsymbol{y}}_i - \hat{b}_i\hat{\boldsymbol{y}}_i||^2}{\hat{\boldsymbol{y}}^\top\hat{\boldsymbol{y}}}, \tag{3.1}$$

where the denominator term ensures that $\hat{\boldsymbol{y}}$ resides on a unit hyper-sphere with $||\hat{\boldsymbol{y}}|| = 1$, thus avoiding scaling issues. Note that by construction, C-Loss only depends on the predictions of f_{NN} and does not rely on true labels, $(\boldsymbol{y}, b)$. Hence, C-Loss can be evaluated even on the unlabeled test data, $\mathcal{D}_U$.

Spectrum Loss: Note that there are many non-interesting solutions of $\hat{A}\hat{\boldsymbol{y}} = \hat{b}\hat{\boldsymbol{y}}$ that can appear as "local minima" in the optimization landscape of C-Loss. For example, for every input $\hat{A}_i \in \mathcal{D}_U$, there are d possible eigen-solutions (where d is the length of $\hat{\boldsymbol{y}}$), each of which will result in a perfectly low value of C-Loss $= 0$, thus acting as a local minima. However, we are only interested in a specific eigenvalue -usually the smallest or the largest- for every $\hat{A}_i$. Therefore, we consider minimizing another PG loss term that ensures the predicted $\hat{b}$ at every sample is the desired one. In the case of the quantum mechanics application, we use the following loss to find the smallest eigen-solution:

$$S\text{-Loss}(\theta) := \sum_i \exp\Big(\hat{b}_i\Big) \tag{3.2}$$

The use of exp function ensures that E-Loss is always positive, even when predicted eigenvalues are negative. As for the electromagnetic propagation application, we simply direct the optimization towards the largest eigenvalue by replacing $\hat{b}_i$ with $-\mathrm{Re}(\hat{b}_i)$, where Re extracts the real part of the complex eigenvalue.

3.3 Adaptive tuning of PG loss weights: A simple strategy for incorporating PG loss terms in the learning objective of f_{NN} is to add them to Train-MSE using trade-off weight parameters, λ_C and λ_S, for C-Loss and S-Loss, respectively. Conventionally, such trade-off weights are kept constant to a certain value across all epochs of gradient descent. This inherently assumes that the importance of PG loss terms in guiding the learning of f_{NN} towards a generalizable solution is constant across all stages (or epochs) of gradient descent, and they are in agreement with each other. However, in practice, we empirically find that C-Loss, S-Loss, and Train-MSE compete with each other and have varying importance at different stages (or epochs) of ANN learning. Hence, we consider the following ways of adaptively tuning the trade-off weights of C-Loss and S-Loss, λ_C and λ_S as a function of the epoch number t.

Annealing λ_S: The first observation we make is that S-Loss plays a critical role in the initial stages of learning, where gradient descent has a tendency to move towards a local minima solution and then refine the solution until convergence. Having a large value of λ_S in the beginning few epochs is thus helpful to avoid the selection of local minima and instead converge towards

a generalizable solution. Hence, we consider performing a simulated annealing of λ_S that takes on a high value in the beginning epochs, that slowly decays to 0 after sufficiently many epochs. Specifically, we consider the following annealing procedure for λ_S:

$$\lambda_S(t) = \lambda_{S0} \times (1 - \alpha_S)^{\lfloor t/T \rceil}, \tag{3.3}$$

where, λ_{S0} is a hyper-parameter denoting the starting value of λ_S at epoch 0, $\alpha_S < 1$ is a hyper-parameter that controls the rate of annealing, and T is a scaling hyper-parameter.

Cold Starting λ_C: The second observation we make is on the effect of C-Loss on the convergence of gradient descent towards a generalizable solution. Note that C-Loss, while being critical in ensuring physical consistency of our predictions with the eigenvalue equation, suffers from a large number of local minima and hence is susceptible to favoring the learning of non-generalizable solutions due to its high non-convexity. Hence, in the beginning epochs, when we are taking large steps in the gradient descent algorithm to move towards a minimum, it is important to keep C-Loss turned off so that the learning process does not get stuck in one of the non-generalizable minima of C-Loss. Once we have crossed a sufficient number of epochs and have already zoomed into a region in the parameter space in close vicinity to a generalizable solution, we can safely turn on C-Loss so that it can help refine θ to converge to the generalizable solution. Based on this observation, we consider "cold starting" λ_C, where its value is kept to 0 in the beginning epochs after which it is raised to a constant value, as given by the following procedure:

$$\lambda_C(t) = \lambda_{C0} \times \mathit{sigmoid}(\alpha_C \times (t - T_a)), \tag{3.4}$$

where, λ_{C0} is a hyper-parameter denoting the constant value of λ_C after a sufficient number of epochs, α_C is a hyper-parameter that dictates the rate of growth of the sigmoid function, and T_a is a hyper-parameter that controls the cutoff number of epochs after which λ_C is activated from a cold start of 0.

Overall Learning Objective: Combining all of the innovations described above in designing and incorporating PG loss functions, we consider the following overall learning objective:

$$E(t) = \text{Train-Loss} + \lambda_C(t)\ C\text{-Loss} + \lambda_S(t)\ S\text{-Loss}$$

Note that Train-Loss is only computed over $\mathcal{D}_{Tr}$, whereas the PG loss terms, C-Loss and S-Loss, are computed over $\mathcal{D}_{Tr}$ as well as the set of unlabeled samples, $\mathcal{D}_U$. We refer to our proposed model trained using the above learning objective as *CoPhy*-PGNN, which is an abbreviation for *Competing Physics* PGNN.

4 Evaluation setup

Data in Quantum Physics: We considered $n = 4$ spin systems of Ising chain models for predicting their ground-state wave-function under varying influences of two controlling parameters: B_x and B_z, which represent the strength of external magnetic field along the X axis (parallel to the direction of Ising chain), and Z axis (perpendicular to the direction of the Ising chain), respectively. The Hamiltonian matrix $\mathbf{H}$ for these systems is then given as:

$$\mathbf{H} = -\sum_{i=0}^{n-1} \sigma_i^z \sigma_{i+1}^z - B_x \sum_{i=0}^{n-1} \sigma_i^x - B_z \sum_{i=0}^{n-1} \sigma_i^z, \tag{4.5}$$

where $\sigma^{x,y,z}$ are Pauli operators and ring boundary conditions are imposed. Note that the size of $\mathbf{H}$ is $d = 2^n = 16$. We set B_z to be equal to 0.01 to break the ground state degeneracy, while B_x was sampled from a uniform distribution from the interval [0, 2].

Note that when $B_x < 1$, the system is said to be in a ferromagnetic phase, since all the spins prefer to either point upward or downward collectively. However, when $B_x > 1$, the system transitions to paramagnetic phase, where both upward and downward spins are equally possible. Because the ground-state wave-function behaves differently in the two regions, the system actually exhibits different physical properties. Hence, in order to test for the generalizability of ANN models when training and test distributions are different, we generate training data only from the region deep inside the ferromagnetic phase for $B_x < 0.5$, while the test data is generated from a much wider range $0 < B_x < 2$, covering both ferromagnetic and paramagnetic phases. In particular, the training set comprises of $N = 100,000$ points with B_x uniformly sampled from 0 to 0.5, while the test set comprises of $M = 20,000$ points with B_x uniformly sampled from 0 to 2. Labels for the ground-state wave-function for all training and test points were obtained by direct diagonalization of the Ising Hamiltonian using Intel's implementation of LAPACK (MKL). We used uniform sub-sampling and varied N from 100 to $20,000$ to study the effect of training size on the generalization performance of comparative ANN models. For validation, we also used sub-sampling on the training set to obtain a validation set of 2000 samples. We performed 10 random runs of uniform sampling for every value of N, to show the mean and variance of the performance metrics of comparative ANN models, where at every run, a different random initializtion of the ANN models is also used. Unless otherwise stated, the results in any experiment are presented over training size $N = 2000$.

Data in Electromagnetic Propagation: We considered a periodically stratified layer stack of 10 layers of equal length per period. The refractive index n of each layer was randomly assigned an integer value between 1 and 4. Hence, the permittivity $\epsilon = n^2$ can take values from $\{1, 4, 9, 16\}$. We chose the period of the multilayer stack to be 5 times the free-space wavelength at the operating frequency ($\omega/c = 2\pi; \Lambda = 5$). The increase of the complexity of the electromagnetic structure leads to an increase in the matrix size $\hat{A}$, making the solution of the eigenvalue problem both computationally and memory-intensive. Note that the majority of eigenvalue solvers rely on iterative algorithms and are therefore not easily deployable in GPU environments. However, a neural net can, in principle, learn to predict the eigenvalues and eigenvectors of the matrix $\hat{A}$. To demonstrate the scalability of our approach we generate $N = 2000$ realizations of the layered structure. For each example, we also generate the associated $\hat{A}$ of size 401×401 complex values, making the scale of this problem about 2500 times larger than that of the quantum mecanics problem. The combination of the challenging scale of this eigen-decompostion and the scarcity of training data makes this problem interesting from scalability and generalizaility perspective. To demonstrate extrapolation ability, we take a training size $|\mathcal{D}_{Tr}| = 370$ realizations that has a refractive index of only 1 in its first layer. On the other hand, we take a test set of size $|\mathcal{D}_U| = 1630$ with the first layer's refractive index unconstrained (i.e. any value from the set {1,2,3,4}).

Baseline Methods: Since there does not exist any related work in PGNN that has been explored for our target applications, we construct analogue versions of PINN-*analogue* [9] and PGNN-*analogue* [8] adapted to our problem using their major features. We describe these baselines along with others in the following:

1. Black-box NN (or NN): This refers to the "black-box" ANN model trained just using Train-Loss without any PG loss terms.
2. PGNN-*analogue*: The analogue version of PGNN [8] for our problem where the hyper-parameters corresponding to S-Loss and C-Loss are set to a constant value.
3. PINN-*analogue*: The analogue version of PINN [9] for our problem that performs label-free learning only using PG loss terms with constant weights. Note that the PG loss terms are not defined as PDEs in our problem.
4. *MTL*-PGNN: Multi-task Learning (MTL) variant of PGNN where PG loss terms are optimized alternatively [20] by randomly selecting one from all the loss terms for each mini-batch in every epoch.

Models	MSE ($\times 10^2$)	Cosine Similarity
CoPhy-PGNN **(proposed)**	$\mathbf{0.35 \pm 0.12}$	$\mathbf{99.50 \pm 0.12}\%$
Black-box NN	1.06 ± 0.16	$95.32 \pm 0.58\%$
PINN-*analogue*	6.27 ± 6.94	$87.37 \pm 12.87\%$
PGNN-*analogue*	0.91 ± 1.90	$97.97 \pm 4.89\%$
MTL-PGNN	6.33 ± 2.69	$84.26 \pm 6.33\%$
CoPhy-PGNN (only-$\mathcal{D}_{Tr}$)	1.82 ± 0.36	$93.61 \pm 0.91\%$
CoPhy-PGNN (w/o S-Loss)	10.97 ± 0.71	$76.27 \pm 0.80\%$
CoPhy-PGNN (Label-free)	9.97 ± 4.42	$63.97 \pm 16.20\%$

Table 1: Test-MSE and Cosine Similarity of comparative ANN models on training size $N = 1000$ on the quantum physics application.

We also consider the following ablation models:

1. *CoPhy*-PGNN (only-$\mathcal{D}_{Tr}$): This is an ablation model where the PG loss terms are only trained over the training set, $\mathcal{D}_{Tr}$. Comparing our results with this model will help in evaluating the importance of using unlabeled samples $\mathcal{D}_U$ in the computation of PG loss.
2. *CoPhy*-PGNN (w/o S-Loss): This is another ablation model where we only consider C-Loss in the learning objective, while discarding S-Loss.
3. *CoPhy*-PGNN (Label-free): This ablation model drops Train-MSE from the learning objective and hence performs label-free (LF) learning only using PG loss terms.

Evaluation Metrics: We use two evaluation metrics: (a) Test MSE, and (b) Cosine Similarity between our predicted eigenvector, $\hat{\boldsymbol{y}}$, and the ground-truth, $\boldsymbol{y}$, averaged across all test samples. We particularly chose the cosine similarity for multiple reasons. First, Euclidean distances are not very meaningful in high-dimensional spaces of wave-functions, such as the ones we are considering in our analyses. Second, an ideal cosine similarity of 1 provides an intuitive baseline to evaluate goodness of results. But most importantly, in the electromagnetic propagation application, it is crucial to compare not just Fourier coefficients of the expansion (which is what the neural net produces) but rather the actual profile of the magnetic field in the real space. The accuracy of this prediction can be tested by calculating the overlap integral between the exact and the predicted profiles. That integral, due to orthogonality of Fourier expansion, reduces to the cosine similarity. This facilitates testing whether our predicted vectors re valid eigenvectors from a physical standpoint.

5 Results and analysis

5.1 Quantum Physics Application: Table 1 provides a summary of the comparison of *CoPhy*-PGNN with baseline methods on the quantum physics applica-

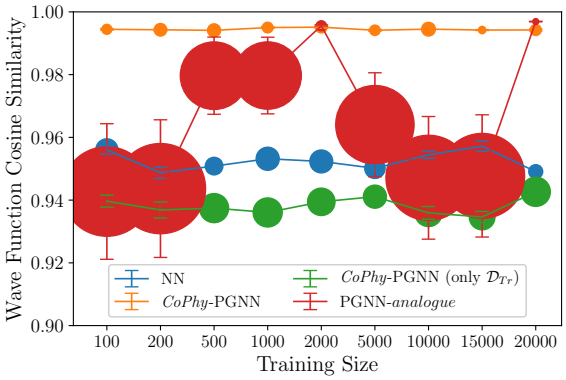

Figure 2: Cosine similarity across training sizes.

tion. We can see that our proposed model shows significantly better performance in terms of both Test-MSE and Cosine Similarity. In fact, the cosine similarity of our proposed model is almost 1, indicating almost perfect fit with test labels. (Note that even a small drop in cosine similarity can lead to cascading errors in the estimation of other physical properties derived from the ground-state wave-function.) An interesting observation from Table 1 is that *CoPhy*-PGNN (Label-free) actually performs even worse than black-box NN. This shows that solely relying on PG loss without considering Train-MSE is fraught with challenges in arriving at a generalizable solution. Indeed, using a small number of labeled examples to compute Train-MSE provides a significant nudge to ANN learning to arrive at more accurate solutions. Another interesting observation is that *CoPhy*-PGNN (only-$\mathcal{D}_{Tr}$) again performs even worse than Black-box NN. This demonstrates that it is important to use unlabeled samples in $\mathcal{D}_U$, which are representative of the test set, to compute the PG loss. Furthermore, notice that *CoPhy*-PGNN (w/o S-Loss) actually performs worst across all models, possibly due to the highly non-convex nature of C-Loss function that can easily lead to local minima when used without S-Loss. This sheds light on another important aspect of PGNN that is often over-looked, which is that it does not suffice to simply add a PG-Loss term in the objective function in order to achieve generalizable solutions. In fact, an improper use of PG Loss can result in worse performance than a black-box model.

5.1.1 Effect of varying training size: Fig. 2 shows the differences in performance of comparative algorithms as we vary the training size from 100 to 20000. We can see that PGNN-*analogue*, which does not perform adaptive tuning, shows a high variance in its results across training sizes. This is because without cold starting λ_C, C-Loss can be quite unstable in the beginning epochs and can guide the gradient descent into one of its many local minima, especially when the gradients of train-MSE are weak due to paucity of training data. On the other hand, *CoPhy*-PGNN performs consistently better than all other baseline methods, with smallest variance in its results across 10 random runs. In fact, our proposed model is able to perform well even over 100 training samples.

5.1.2 Studying convergence across epochs: Figure 3 shows the variations in Train-MSE, Test-MSE, and C-Loss terms for four comparative models at every epoch of gradient descent. We can see that all models are able to achieve a reasonably low value of Train-MSE at the final solution expect *CoPhy*-PGNN (Label-free), which is expected since it does not consider minimizing Train-MSE in the learning objective. Black-box NN actually shows the lowest value of Train-MSE than all other models. However, the quantity that we really care to minimize is not the Train-MSE but the Test-MSE, which is indicative of generalization performance. We can see that while our proposed model, *CoPhy*-PGNN shows slightly higher Train-MSE than Black-box NN, it shows drastically smaller Test-MSE at the converged solution, demonstrating the effectiveness of our proposed approach.

A contrasting feature of the convergence plots of *CoPhy*-PGNN relative to Black-box NN is the presence of an initial jump in the Test-MSE values during the first few epochs. This likely arises due to the competing nature of two different loss terms that we are trying to minimize in the beginning epochs: the Train-MSE, that tries to move towards local minima solutions favorable to training data, and S-Loss, that pushes the gradient descent towards generalizable solutions. Indeed, this initial jump in Test-MSE helps in moving out of local minima solutions, after which the Test-MSE plummets to significantly smaller values. Notice that *CoPhy*-PGNN (Label-free) shows a similar jump in Test-MSE in the beginning epochs, because it experiences a similar effect of S-Loss gradients during the initial stages of ANN learning. However, we can see that its Test-MSE is never able to drop beyond a certain value after the initial jump, as it does not receive the necessary gradients of Train-MSE that helps in converging towards generalizable solutions.

Another interesting observation is that *CoPhy*-PGNN (w/o S-Loss) does not show any jump in Test-MSE during the beginning epochs in contrast to our proposed model, since it is not affected by S-Loss. If we further look at C-Loss curves, we can see that *CoPhy*-PGNN (w/o S-Loss) achieves lowest values, since it only considers C-Loss as the PG loss term to be minimized in the learning objective. However, we know that C-

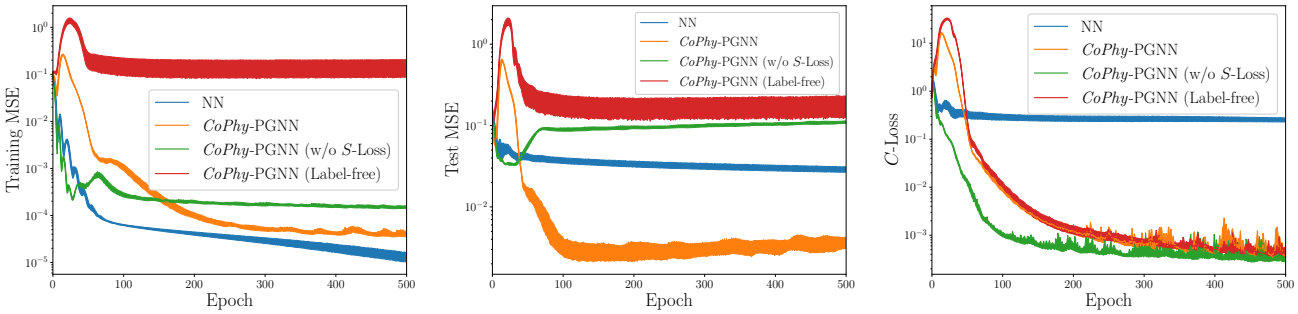

Figure 3: Convergence plots showing Train-MSE, Test-MSE, and C-Loss over epochs.

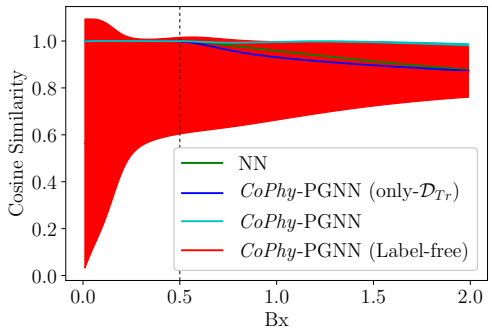

Figure 4: Cosine Similarity on test samples as a function of B_x. The dashed line represents the boundary between the interval used for training (left) and testing (right).

Loss is home to a large number of local minima, and for that reason, even though *CoPhy*-PGNN (w/o S-Loss) shows low C-Loss values, its test-MSE quickly grows to a large value, indicating its convergence on a local minima. These results demonstrate that a careful trade-off of PG loss terms along with Train-MSE is critical to ensure good generalization performance, such as that of our proposed model. To better understand the behavior of competing loss terms, we conducted a novel gradient analysis that can be found in the supplementary materials.

5.1.3 Evaluating generalization power: Instead of computing the average cosine similarity across all test samples, Figure 4 analyzes the trends in cosine similarity over test samples with different values of B_x, for four comparative models. Note that none of these models have observed any labeled data during training outside the interval of $B_x \in [0, 0.5]$. Hence, by testing for the cosine similarity over test samples with $B_x > 0.5$, we are directly testing for the ability of ANN models to generalize outside the data distributions it has been trained upon. Evidently, all label-aware models perform well on the interval of $B_x \in [0, 0.5]$. However, except for *CoPhy*-PGNN, all baseline models degrade significantly outside that interval, proving their lack of generalizability. Moreover, the label-free, *CoPhy*-PGNN (Label-free), model is highly erratic, and performs poorly across the board.

5.1.4 Analysis of loss landscapes: To truly understand the effect of adding PG loss to ANN's generalization performance, here we visualize the landscape of different loss functions w.r.t. ANN model parameters. In particular, we use the code in [21] to plot a 2D view of the landscape of different loss functions, namely Train-MSE, Test-MSE, and PG-Loss (sum of C-Loss and S-Loss), in the neighborhood of a model solution, as shown in Figure 5. In each of the sub-figures of this plot, the model's parameters are treated with filter normalization as described in [22], and hence, the coordinate values of the axes are unit-less. Also, the model solutions are represented by blue dots. As can be seen, all label-aware models have found a minimum in Train-MSE landscape. However, when the test-MSE loss surface is plotted, it is clear that while the *CoPhy*-PGNN model is still at a minimum, the other baseline models are not. This is a strong indication that using the PG loss with unlabeled data can lead to better extrapolation; it allows the model to generalize beyond in-distribution data. We can see that without using labels, *CoPhy*-PGNN (Label-free) fails to reach a good minimum of Test-MSE, even though it arrives at a minimum of PG Loss.

To understand the interactions among competing PG loss terms, we further computed the projection of the gradient of every loss term w.r.t. the optimal gradient direction (computed empirically) at every epoch and investigated the importance of PG loss terms in guiding towards the optimal gradient at different stages of neural network learning. See supplementary materials for more details.

5.2 Electromagnetic Propagation Application: For this application, the size of $\hat{A}$ is 401×401, making it a daunting task for an eigensolver in terms of compu-

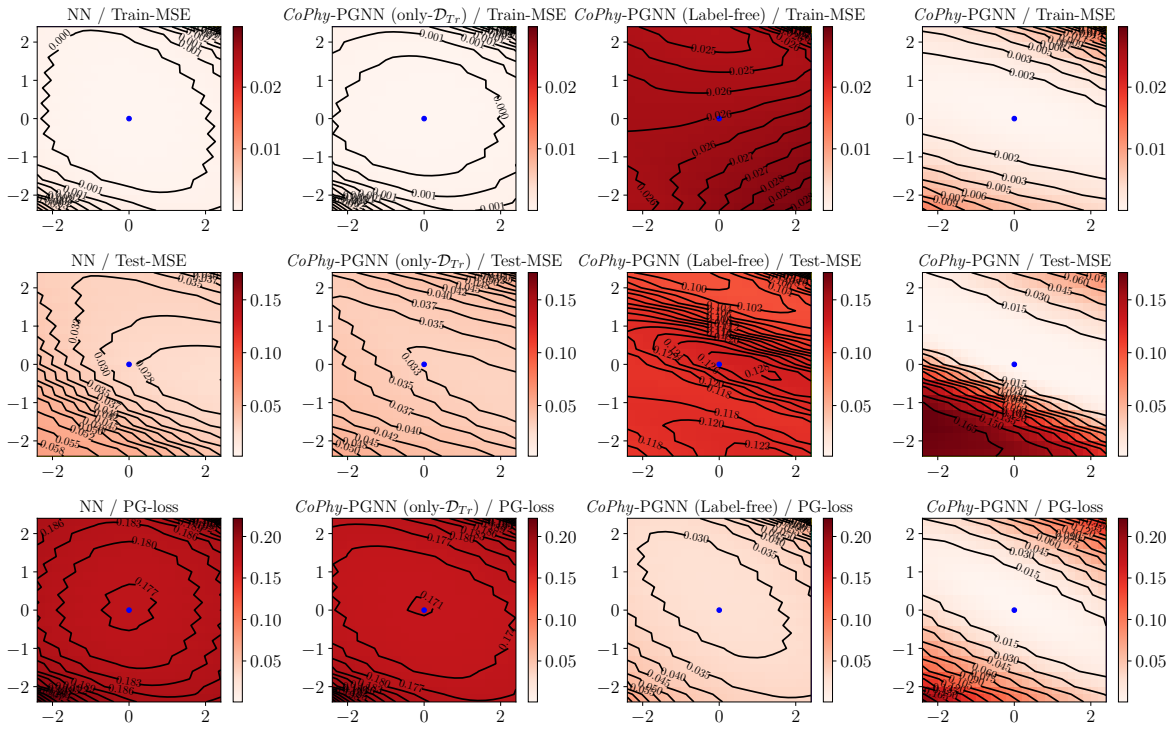

Figure 5: A comprehensive comparison between *CoPhy*-PGNN and different baselines. The 1st and 2nd columns show that without using unlabeled data, the model does not generalize well. On the other hand, the 3rd column shows that without labeled data, the model fails to reach a good minimum. Only the last column, our proposed model, shows a good fit across both labeled and unlabeled data. The best performing model is also the model that best optimizes the PG loss.

tation time. As a result, a grid search hyper-parameter tuning of ANN models is prohibitively expensive. This is due to the large number of epochs needed to optimize a model for a problem of this scale. Nonetheless, we were still able to optimize a model to do fairly well by manually adjusting the hyper-parameters and architecture of *CoPhy*-PGNN to yield acceptable results on the validation set (see supplementary material for more details). We emphasize, however, that a more exhaustive tuning could lead to better results that surpass the ones we obtained. Figure 6 shows that *CoPhy*-PGNN is still able to better extrapolate than a Black-box NN on testing scenarios with permittivity greater than 1. In fact, we have observed that as Black-box NN solely optimizes Train-MSE, its cosine similarity measure deteriorates on the test set. This is in contrast to *CoPhy*-PGNN's ability to maintain a cosine similarity close to 1 even though its validation loss is comparable to Black-box NN's.

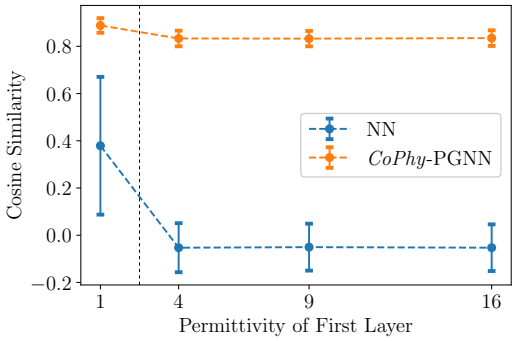

Figure 6: Cosine similarity of *CoPhy*-PGNN compared to Black-box NN for the electromagnetic propagation application. The dashed line represents the boundary between the interval used for training (left) and testing (right).

While training our model still takes a significant amount of time (about 12 hours), its effectiveness with respect to testing speed is demonstrated in Table 2. We can see that our approach is at least an order of magnitude faster during testing than any numerical eigensolver. This highlights the promise in using neural networks to solve physics-based eigenvalue problems, since, once trained, they can be used to produce eigen-solutions on test points much faster than numerical methods. Further, while *CoPhy*-PGNN shows higher error than numerical solvers, note that the cosine similarity of our model's predictions with ground-truth is close to 0.8, thus admitting physical usability.

Solver	average time (seconds)	average $\lvert\hat{A}\boldsymbol{y} - b\boldsymbol{y}\rvert$
CoPhy-PGNN	0.0430	1.878×10^{2}
`numpy.linalg.eig` [23]	93.743	7.714×10^{-6}
`Matlab` [24]	0.196	8.747×10^{-12}
`torch.eig` [25]	16.565	6.821×10^{-13}
`scipy.linalg.eig` [26]	106.223	7.538×10^{-4}
`scipy.sparse.linalg.eigs` [27]	8.893	4.418×10^{-3}

Table 2: Comparison of speed and accuracy between *CoPhy*-PGNN and other numerical eigensolvers.

6 Conclusions and future work

This work proposed novel strategies to address the problem of competing physics loss functions in PGNN. For the general problem of solving eigen-value equations, we designed a PGNN model *CoPhy*-PGNN and demonstrated its efficacy in two target applications in quantum mechanics and electromagnetic propagation. From our results, we found that: 1) PG loss helps to extrapolate and gives the model better generalizablity; and 2) Using labeled data along with PG loss results in more stable PGNN models. Moreover, we visualized the loss landscape to give a better understanding of how the combination of both labeled data loss and PG loss leads to better generalization performance. We have also demonstrated the generalizability of our *CoPhy*-PGNN to multiple application domains with varying types of physics loss functions, as well as its scalability to large systems. Future work can focus on reducing the training time of our model so as to perform extensive hyper-parameter tuning to reach a better global minima. Finally, while this work empirically demonstrated the value of *CoPhy*-PGNN in combating with competing PG loss terms, future work can focus on theoretical analyses of our approach.

7 Acknowledgments

This work was supported by the National Science Foundation via grants 2026710 (for M. Elhamod, J. Bu, and A. Karpatne), 2026702 (for C. Singh, M. Redell, and W-C. Lee), and 2026703 (for A. Ghosh and V. Podolskiy).

References

[1] Tim Appenzeller. The scientists' apprentice. *Science*, 357(6346):16–17, 2017.

[2] D Graham-Rowe, D Goldston, C Doctorow, M Waldrop, C Lynch, F Frankel, R Reid, S Nelson, D Howe, and SY Rhee. Big data: science in the petabyte era. *Nature*, 455(7209):8–9, 2008.

[3] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

[4] Jared Willard, Xiaowei Jia, Shaoming Xu, Michael Steinbach, and Vipin Kumar. Integrating physics-based modeling with machine learning: A survey. *arXiv preprint arXiv:2003.04919*, 2020.

[5] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

[6] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, 2017.

[7] Xiaowei Jia, Jared Willard, Anuj Karpatne, Jordan Read, Jacob Zwart, Michael Steinbach, and Vipin Kumar. Physics guided rnns for modeling dynamical systems: A case study in simulating lake temperature profiles. In *Proceedings of the 2019 SIAM International Conference on Data Mining*, pages 558–566. SIAM, 2019.

[8] 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.

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

[10] Emmanuel de Bezenac, Arthur Pajot, and Patrick Gallinari. Deep learning for physical processes: Incorporating prior scientific knowledge. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124009, 2019.

[11] Yeonjong Shin, Jerome Darbon, and George Em Karniadakis. On the convergence and generalization of physics informed neural networks. *arXiv preprint arXiv:2004.01806*, 2020.

[12] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning approach for reconstructing reynolds stress modeling discrepancies based on dns data. *Physical Review Fluids*, 2(3):034603, 2017.

[13] Jian-Xun Wang, Jin-Long Wu, and Heng Xiao. Physics-informed machine learning for predictive turbulence modeling: Using data to improve rans modeled reynolds stresses. *arXiv preprint arXiv:1606.07987*, 2016.

[14] Jian-Xun Wang, Jinlong Wu, Julia Ling, Gianluca Iaccarino, and Heng Xiao. A comprehensive physics-informed machine learning framework for predictive turbulence modeling. *arXiv preprint arXiv:1701.07102*, 2017.

[15] Arka Daw, R Quinn Thomas, Cayelan C Carey, Jordan S Read, Alison P Appling, and Anuj Karpatne.

Physics-guided architecture (pga) of neural networks for quantifying uncertainty in lake temperature modeling. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 532–540. SIAM, 2020.

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

[17] Maziar Raissi, Paris Perdikaris, and George Em Karniadakis. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017.

[18] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient pathologies in physics-informed neural networks. *arXiv preprint arXiv:2001.04536*, 2020.

[19] Rich Caruana. Multitask learning: A knowledge-based source of inductive bias. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, page 41–48, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[20] Zhuoliang Kang, Kristen Grauman, and Fei Sha. Learning with whom to share in multi-task feature learning. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 521–528, Madison, WI, USA, 2011. Omnipress.

[21] Marcello De Bernardi. loss-landscapes, 2019.

[22] Hao Li, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. Visualizing the loss landscape of neural nets. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6389–6399. Curran Associates, Inc., 2018.

[23] `numpy.linalg.eig`. `https://numpy.org/doc/stable/reference/generated/numpy.linalg.eig.html`. Accessed: 2020-10-11.

[24] `Matlab eigs('largestreal')`. `https://www.mathworks.com/help/matlab/ref/eigs.html`. Accessed: 2020-10-11.

[25] `torch.eig`. `https://pytorch.org/docs/stable/generated/torch.eig.html`. Accessed: 2020-10-11.

[26] `scipy.linalg.eig`. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.linalg.eig.html`. Accessed: 2020-10-11.

[27] `scipy.sparse.linalg.eig`. `https://docs.scipy.org/doc/scipy/reference/generated/scipy.sparse.linalg.eigs.html#scipy-sparse-linalg-eigs`. Accessed: 2020-10.

[28] O. F. de Alcantara Bonfim, B. Boechat, and J. Florencio. Ground-state properties of the one-dimensional transverse ising model in a longitudinal magnetic field. *Phys. Rev. E*, 99:012122, Jan 2019.

[29] M. Brando, D. Belitz, F. M. Grosche, and T. R. Kirkpatrick. Metallic quantum ferromagnets. *Rev. Mod. Phys.*, 88:025006, May 2016.

[30] Yi Zhou, Kazushi Kanoda, and Tai-Kai Ng. Quantum spin liquid states. *Rev. Mod. Phys.*, 89:025003, Apr 2017.

[31] Barbara M. Terhal. Quantum error correction for quantum memories. *Rev. Mod. Phys.*, 87:307–346, Apr 2015.

[32] M. G. Moharam and T. K. Gaylord. Rigorous coupled-wave analysis of planar-grating diffraction. *J. Opt. Soc. Am.*, 71(7):811–818, Jul 1981.

[33] Thomas Schuster, Johannes Ruoff, Norbert Kerwien, Stephan Rafler, and Wolfgang Osten. Normal vector method for convergence improvement using the rcwa for crossed gratings. *J. Opt. Soc. Am. A*, 24(9):2880–2890, Sep 2007.

[34] Philippe Lalanne and Eric Silberstein. Fourier-modal methods applied to waveguide computational problems. *Opt. Lett.*, 25(15):1092–1094, Aug 2000.

[35] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

[36] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(1):281–305, February 2012.

[37] Anna Choromanska, Mikael Henaff, Michaël Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *CoRR*, abs/1412.0233, 2014.

Appendix

Mohannad Elhamod*† Jie Bu*† Christopher Singh‡ Matthew Redell‡

Abantika Ghosh§ Viktor Podolskiy§ Wei-Cheng Lee‡ Anuj Karpatne†

8 Relevant Physics Background

8.1 Ising Chain and Quantum Mechanics Quantum mechanics provides a theoretically rigorous framework to investigate physical properties of quantum materials by solving the Schrödinger's equation—the fundamental law in quantum mechanics. The Schrödinger's equation is essentially a PDE that can be easily transformed into an eigenvalue problem of the form: $\mathbf{H}\Psi = E\Psi$, where $\mathbf{H}$ is the Hamiltonian Matrix, Ψ is the wave-function, and E is the energy, a scalar quantity. (Note that many other PDEs in physical sciences, e.g., Maxwell's equations, yield to a similar transformation to an eigenvalue problem.) All information related to the dynamics of the quantum system is encoded in the eigen-vectors of $\hat{H}$, i.e., Ψ. Among these eigen-vectors, the ground state wave-function, Ψ_0, defined as the eigen-vector with lowest energy, E_0, is a fundamental quantity for understanding the properties of quantum systems. Exploring how Ψ_0 evolves with controlling parameters, e.g., magnetic field and bias voltage, is an important subject of study in material science.

A major computational bottleneck in solving for the ground-state wave-function Ψ_0 is the diagonalization of the Hamiltonian matrix, $\mathbf{H}$, whose dimension grows exponentially with the size of the system. In order to study the effects of controlling parameters on the physical properties of a quantum system, theorists routinely have to perform diagonalizations on an entire family of Hamiltonian matrices, with the same structure but slightly different parameters.

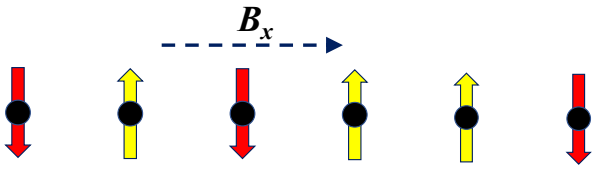

Figure 7: Schematic illustration of the Ising spin chain. Each site is occupied by a spin that can only take two values, either spin up (+1) or spin down (-1). The external magnetic field B_x is applied along the chain direction.

Here we study a quintessential model of the transverse field: Ising chain model [28], which is a unidimensional spin chain model under the influence of a transverse magnetic field B_x, as shown in Fig. 7. Spin is the intrinsic angular momentum possessed by elementary particles including electrons, protons, and neutrons. The Ising spin chain model describes a system in which multiple spins are located along a chain and they interact only with their neighbors. By adding an external magnetic field (B_x), the ground state wave-function could change dramatically. This model and its derivatives have been used to study a number of novel quantum materials([29, 30]) and can also be used for quantum computing [31], since the qubit, the basic unit of quantum computing, can also be represented as a spin. However, the challenge in finding the ground-state wave-function of this model is that the dimension d of the Hamiltonian $\mathbf{H}$ grows exponentially as $d = 2^n$, where n equals the number of spins. We aim to develop PGNN approaches that can learn the predictive mapping from the space of Hamiltonians, $\mathbf{H}$, to ground-state wave-functions, Ψ, using the physics of the Schrödinger's equation along with labels produced by diagonalization solvers on training set.

*equal contribution
†Department of Computer Science, Virginia Tech
‡Department of Physics, Binghamton University
§Department of Physics and Applied Physics, University of Massachusetts Lowell
*equal contribution
†Department of Computer Science, Virginia Tech
‡Department of Physics, Binghamton University
§Department of Physics and Applied Physics, University of Massachusetts Lowell

8.2 Electromagnetic Propagation and Maxwell Equations Mathematically, the propagation of the electromagnetic waves in periodically stratified layer stacks can be described by Maxwell equations:

$$\vec{\nabla} \times \vec{H} = \frac{1}{c}\epsilon\frac{\partial \vec{E}}{\partial t} \tag{8.6}$$

$$\vec{\nabla} \times \vec{E} = -\frac{1}{c}\frac{\partial \vec{H}}{\partial t}, \tag{8.7}$$

where $\vec{E}$, $\vec{H}$, ϵ, and c represent electric and magnetic fields, permittivity of layered material, and speed of light in vacuum (in Gaussian units) respectively, and t stands for time. When permittivity of layered array is periodic function of x: $\epsilon(x+\Lambda) = \epsilon(x)$, propagation of monochromatic ($\vec{E}, \vec{H} \propto e^{-i\omega t}$) transverse-magnetic (TM) polarized waves ($\vec{H} \| \hat{y}$) can be reduced to the eigenvalue problem:

$$\hat{A}\vec{h}_m = k_z^m h_m \tag{8.8}$$

where k_z represents the propagation constant of the electromagnetic modes along the layers ($\vec{E}, \vec{H} \propto e^{ik_z z}$), $\vec{h}_m$ represent coefficients of the Fourier transform of the spatial profile of $\vec{H}(x)$, and the elements of the matrix $\hat{A}$ is related to operating frequency ω, structure of the electromagnetic waves in the direction normal to the layers (parameterized by quasi-wavenumber k_x), and spatial profile of permittivity $\epsilon(x)$ [32]. This technique, known as Rigorous Coupled Wave Analysis, has been extended to electromagnetic structures with two dimensional periodicity [33] as well as to aperiodic structures [34].

9 Hyper-parameter Selection

9.1 Quantum physics application

9.1.1 Hyperparameter search To exploit the best potential of the models, we conducted hyperparameter search prior to many of our experiments[1] on training set of size $N = 20000$, by doing random sampling in a fixed range for every hyper-parameter value. We chose the average of the top-5 hyper-parameter settings that showed the lowest error on the validation set, which was 2000 instances sampled from the training set. For the proposed *CoPhy*-PGNN model, this resulted in the following set of hyper-parameters: $\{\lambda_{S0} = 2.3, \alpha_S = 0.14, \lambda_{C0} = 0.85, \alpha_C = 0.17, T_a = 51\}$, and we chose $T = 50$ fixed for all models. The same hyper-parameter values were used across all values of N in our experiments to show the robustness of these values.

We searched for the best model architecture using simple multi-layer neural networks that does not show significant overfitting or underfitting, then we fixed the architecture for all the models in our work. The models comprise of four fully-connected layers with *tanh* activation and an linear output layer. The widths of all the hidden states are 100. All the experiments used *Adamax* [35] optimizer and set maximum training epochs to 500 that most of the models will converge before that limit.

Since different models may use different loss terms, the numbers of hyperparameters to search are different, and some of them were not being searched. We use random search [36] and run around 300 to 500 runs per model to keep a balance between search quality and the time spent. The hyperparameters we searched include:

1. For S-Loss: $\lambda_{C0} \sim \mathcal{U}(0, 5)$, $\alpha_E \sim \mathcal{U}(0, 0.5)$.
2. For C-Loss: $\lambda_{C0} \sim \mathcal{U}(0, 2)$, $\alpha_C \sim \mathcal{N}(0, 0.5)$, $T_\alpha \sim \mathcal{U}(0, 200)$

9.1.2 Sigmoid Cold-start and Other Different Modes Additionally, to further prove our choice on *sigmoid* is indeed effective. We compared three other modes with *sigmoid*: *quick-drop* (Eq. 9.10), *quick-start* (Eq. 9.11), *inversed-sigmoid* (Eq. 9.9).

$$\lambda_C(t) = \lambda_{C0} \times [1 - sigmoid(\alpha_C(t - T_\alpha))] \tag{9.9}$$

$$\lambda_C(t) = \lambda_{C0} \times (1 + \alpha_C)^{\min(0, -t+T_\alpha)} \tag{9.10}$$

$$\lambda_C(t) = \lambda_{C0} \times [1 - (1 + \alpha_C)^{\min(0, -t+T_\alpha)}] \tag{9.11}$$

We replaced the *sigmoid* cold-start with the three modes in *CoPhy*-PGNN and ran 400 times for each mode to do hyperparameter search on λ_{C0}, α_C, T_α. Using the average hyperparameter values for the top-5 models that have the lowest validation error. The results are:

1. *quick-drop*: $\lambda_{C0} = 0.836881$, $\alpha_C = 0.062851$, $T_\alpha = 14.0$.
2. *quick-start*: $\lambda_{C0} = 0.936669$, $\alpha_C = 0.073074$, $T_\alpha = 61.2$.
3. *sigmoid*: $\lambda_{C0} = 0.846349$, $\alpha_C = 0.020170$, $T_\alpha = 51.0$.
4. *inversed-sigmoid*: $\lambda_{C0} = 0.939779$, $\alpha_C = 0.171778$, $T_\alpha = 59.2$.

[1] All the code and data used in this work is available at https://github.com/jayroxis/Cophy-PGNN. A complete set of code, data, pretrained models, and stored variables can be found at: https://osf.io/ps3wx/?view_only=9681ddd5c43e48ed91af0db019bf285a (cophy-pgnn.tar.gz).

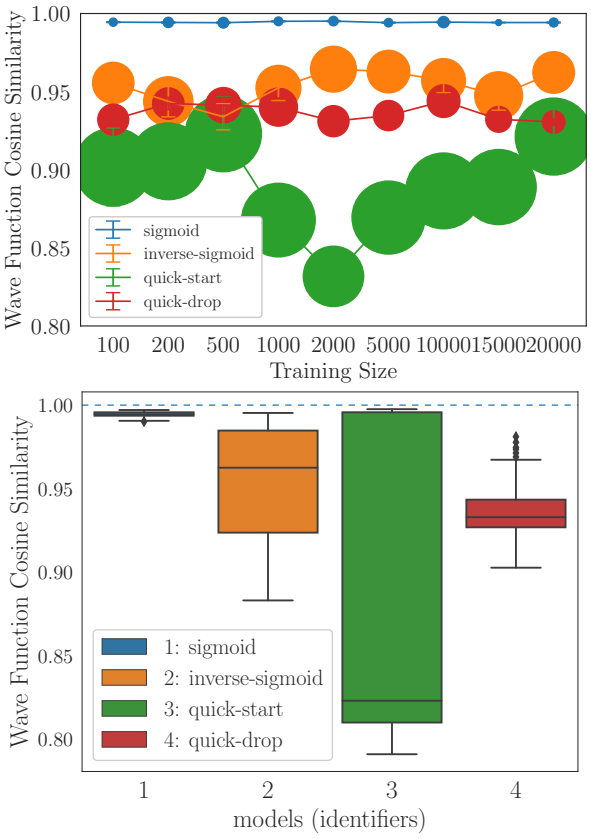

Figure 8: Wave function cosine similarity of different adaptive C-loss w.r.t. different training size. Left: cosine similarity on different training sizes. Right: Mean cosine similarity over all training sizes.

Using the hyperparameter values to set up models, and run 10 times per setting on different training sizes, the results are shown in Fig. 8. We can see that *sigmoid* consistently performed better than other modes in both stability and accuracy. Another important information it conveys is that, *quick-start*, like *sigmoid* though it increases the weight of C-Loss much faster, results in a much more unstable results. Actually our results show that $quick - start$ dominates the leaderboard of both top-10 best and worst performances (also showed in the barplot in Figure 8). It implies that a smooth and gradual switch of dominance between different loss terms is better in terms of stability.

9.2 Electromagnetic Propagation Application For this application, due to the prohibitively expensive training time, we have not conducted an exhaustive hyper-parameter search. Instead, we have manually tweaked the model until the optimization converged to an acceptable solution. The hyper-parameters we used are $\lambda_{C0} = 10^{-8}, \lambda_{S0} = 0.01, \alpha_S = 0.9, \alpha_C = 0.003, T = 50$, and $T_a = 1500$. In terms of architecture, we used a fully-connected ANN, with 2 hidden layers of width 100. Also, instead of using Train-MSE, we used an *L1-loss* based formula *Train-Loss(θ)*:

$$\sum_i \left(\sum_j \left(|\hat{\boldsymbol{y}}_{ij} - \boldsymbol{y}_{ij}| + |\hat{b}_{ij} - b_{ij}| \right) + (||\hat{\boldsymbol{y}}_i|| - ||\boldsymbol{y}_i||) \right) \tag{9.12}$$

Here, the sum of errors was used instead of the mean as we have empirically found it to help the model converge faster. We hypothesize that the mean-squared-error was too small to drive the optimization. We trained the model for 4000 epochs, and selected the best snapshot based on the lowest validation error.

10 Analysis of gradients

10.1 Contribution of Loss Terms The complicated interactions between competing loss terms motivate us to further investigate the different role each loss term plays in the aggregated loss function. The sharp bulge in both MSE and C-Loss in the first few epochs (Fig. 3 in the main document) shows that the optimization process is not quite smooth. Our speculation is that S-Loss and C-Loss are competing loss terms in the combined optimization problem. To monitor the contribution of every loss term in the learning process, we need to measure if the gradients of a loss term points towards the optimal direction of descent to a generalizable model. One way to achieve this is to compute the component of the gradient of a loss term in the optimal direction of descent (leading to a generalizable model). Suppose the desired (or optimal) direction is $\boldsymbol{d}^*$ and the gradient of a loss term L is ∇L. We can then compute the projection of ∇L along the direction of $\boldsymbol{d}^*$ at the k^{th} epoch as:

$$p_L^{(k)} = \frac{\langle \nabla L^{(k)}, \boldsymbol{d}^{*(k)} \rangle}{\|\boldsymbol{d}^{*(k)}\|}. \tag{10.13}$$

A higher projection value indicates a larger step toward the optimal direction at the k^{th} epoch, $\boldsymbol{d}^{*(k)}$, which is defined as:

$$\boldsymbol{d}^{*(k)} = \boldsymbol{\theta}^{(k)} - \boldsymbol{\theta}^*, \tag{10.14}$$

where $\boldsymbol{\theta}^{(k)}$ is the model parameters $\boldsymbol{\theta}$ (i.e., weights and biases of the neural network) at the k^{th} epoch and $\boldsymbol{\theta}^*$ is the optimal state of the model that is known to be generalizable. Note that finding an exact solution for $\boldsymbol{\theta}^*$ that is the global optima of the loss function is practically infeasible for deep neural networks [37]. Hence, in our experiments, we consider the final model

arrived on convergence of the training process as a reasonable approximation of $\boldsymbol{\theta}^*$. For methods such as PGNN-*analogue* and *CoPhy*-PGNN, the final models at convergence performed significantly well and showed a cosine similarity of $\geq 99.5\%$ with ground-truth. This is very close to a model trained directly on the test set that only reaches 99.8%. This gives some confidence that the final models at convergence are good approximations to $\boldsymbol{\theta}^*$. To compute the inner products between ∇L and $\boldsymbol{d}^*$, we used a flattened representation of the model parameters by concatenating the weights and biases across the layers.

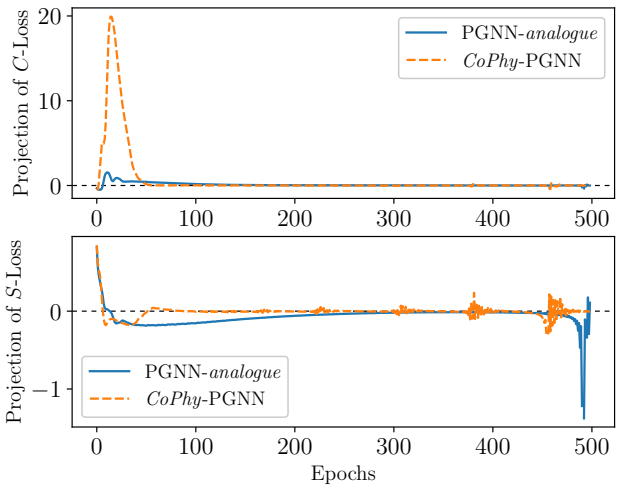

Figure 9: The projection of gradient of each term on the optimal direction. The optimal direction of a certain iteration points from current state to the optimal state.

10.2 Experiment Results We analyze the role of S-Loss and C-Loss in the training process of the two methods: *CoPhy*-PGNN and PGNN-*analogue*. Both these methods were run with the same initialization of model parameters. Training size is 2000 and the rest of settings are same as in Section 5 of the main document.

Figure 9 shows that in the early epochs, the S-Loss has positive projection values, which means that it is helping the method to move towards the optimal state. On the other hand, the projection of C-Loss starts with a negative value, indicating that the gradient of the C-Loss term is counterproductive at the beginning. Hence, S-Loss helps in moving out of the neighborhood of the local minima of C-Loss towards a generalizable solution. However, the projection of C-Loss does not remain negative (and thus counterproductive) across all epochs. In fact, C-Loss makes a significant contribution by having a large positive projection value after around 50 epochs. This shows that as long as we manage to escape from the initial trap caused by the local minima of the C-Loss, then it can turn to guide the model towards desired direction $\boldsymbol{d}^*$. By initially setting λ_C close to zero, it allows the S-Loss to dominate in the initial epochs and move out of the local minima. Later, we let C-Loss to recover to a reasonable value and it will start to play its role. These findings align quite well with the *cold-start* and *annealing* idea proposed in this work and show that it works best when the two loss terms are combined together using adaptive weights.

Note that for this analysis method to produce valid findings, we need to ensure that the loss terms are not pointing towards the direction of an equally good $\boldsymbol{\theta}^*$ that can be arrived at from the same initialization. To ensure this, we investigated how similar the trained models (optimal states) are when started from the same initialization for the two methods. The parameters of the PGNN-*analogue* and *CoPhy*-PGNN showed an average cosine similarity of 98.6%, and in many cases reached 99%. This gave us more confidence to believe that our approximations to the optimal model were sufficient.