Neuro-Visualizer: A Novel Auto-Encoder-Based Loss Landscape Visualization Method With an Application in Knowledge-Guided Machine Learning

Mohannad Elhamod [1] **Anuj Karpatne** [1]

Abstract

In recent years, there has been a growing interest in visualizing the loss landscape of neural networks. Linear landscape visualization methods, such as principal component analysis, have become widely used as they intuitively help researchers study neural networks and their training process. However, these linear methods suffer from limitations and drawbacks due to their lack of flexibility and low fidelity at representing the high dimensional landscape. In this paper, we present a novel auto-encoder-based non-linear landscape visualization method called *Neuro-Visualizer* that addresses these shortcoming and provides useful insights about neural network loss landscapes. To demonstrate its potential, we run experiments on a variety of problems in two separate applications of knowledge-guided machine learning (KGML). Our findings show that *Neuro-Visualizer* outperforms other linear and non-linear baselines and helps corroborate, and sometime challenge, claims proposed by machine learning community. All code and data used in the experiments of this paper can be found at the link below [1].

1. Introduction

Understanding the loss landscape of deep neural network has attracted much attention in recent years, both from theoretical and visualization standpoints. In this work, we focus on the problem of *loss landscape visualization*, which is the practice of plotting a neural network's loss function w.r.t its high-dimensional model parameters (i.e., weights and biases) on a low-dimensional embedding space, usually 1-D (Goodfellow et al., 2014) or 2-D (Li et al., 2018), to qualitatively assess generalization performance and training convergence. A seminal work that established loss landscape visualization as an investigative tool in the field is that by (Li et al., 2018). This work introduced the approach of projecting model parameters to random planes in low dimensions (usually 2D), to visualize and assess the quality of the loss surface at the vicinity of a converged model, including whether the loss surface is ill-regularized and filled with local minima.

However, when it comes to visualizing multiple models (e.g., the set of models forming a training trajectory), the selection of the projection plane becomes more challenging as the chosen plane should capture "interesting" properties of the loss landscape for the entire set of models, not just a single one. Naturally, the most common way for finding this plane is by performing a Principal Component Analysis (PCA) on the set of models, selecting the two main principal components, and visualizing the landscape on that 2D plane such that it passes through the trajectory's final model (Li et al., 2018). Another approach is to use the two eigenvectors with the highest eigenvalues (Chatzimichailidis et al., 2019; Yao et al., 2020). Despite how helpful these projections can be, each of these practices have their own advantages and disadvantages, as detailed below.

First, since training trajectories do not necessarily fit on a 2-D plane, the use of linear projection methods such as PCA yield loss surfaces that are mostly accurate at the point of intersection, which is generally the trajectory's final model, but drop in accuracy as we move away from that point. Second, while some previous works on loss landscape visualization have used non-linear methods such as UMAP and t-SNE (Huang et al., 2020), SHEAP (Shires & Pickard, 2021), and PHATE (Horoi et al., 2022), these methods are more suitable for visualizing the relationship between models (e.g., model clustering) instead of visualizing landscapes of model trajectories. Third, other non-linear approaches such as Locally Linear Embeddings (LLE) (Roweis & Saul, 2000), and Laplacian Eigen-maps (Wang, 2012) suffer from the lack of an inverse transform, making them unsuitable for landscape visualization because constructing the loss

[1] Department of Computer Science, Virginia Tech, Blacksburg, VA, USA. Correspondence to: Mohannad Elhamod <elhamod@vt.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

[1] https://github.com/elhamod/NeuroVisualizer

landscape requires calculating the loss value at each grid point. Therefore, the chosen method must be capable of projecting points from the low-dimensional representation back into the original space of model parameters.A fourth issue with landscape visualization methods is the scale at which the loss surface is visualized. While the most common practice is to use 'filter normalization' (Li et al., 2018) to make the visualization scale invariant, this approach only works when visualizing the loss landscape at the vicinity of a single model, and is not applicable when visualizing training trajectories.

To mitigate the aforementioned shortcomings, this paper focuses on answering the question: can we devise a non-linear projection method for loss landscape visualization that: **(1)** ***faithfully* captures training trajectories and loss landscapes in their vicinity**, thus improving model optimization understanding, and **(2)** ***adaptively* scales the projection space** based on problem requirements?

In response to the question above, we propose a novel auto-encoder-based non-linear loss landscape visualization approach called *Neuro-Visualizer*. While our proposed approach can be applied to any general problem, we specifically demonstrate its potential in the context of two applications in the field of knowledge-guided machine learning (KGML) (Karpatne et al., 2017): solving the Schrodinger's equation in quantum mechanics using the framework of physics-guided neural networks with competing physics (*CoPhy*-PGNN) (Elhamod et al., 2022), and solving generic partial differential equations (PDEs) using the framework of physics-informed neural networks (PINNs) (Raissi et al., 2017a). While these applications in KGML have received considerable attention in recent years, what is missing in the field is a comprehensive understanding of the effects of adding physics-guided loss functions on the loss landscape of neural networks. By using *Neuro-Visualizer*, we are able to visualize and discover novel insights about the performance of competing KGML approaches proposed for the two applications, corroborating, and in some places even challenging, optimization claims proposed in previous KGML literature. Our work thus forges a novel *"collaborative bridge"* between two sub-fields of AI: neural loss landscape visualization and KGML. We anticipate our work to serve as a starting point for other researchers to develop novel visualization approaches for KGML in the future.

2. Related Works

Though qualitative in nature, the analysis of neural loss landscapes through visualization approaches (Li et al., 2018) is becoming a more common practice (Garipov et al., 2018; Mei et al., 2018; Nguyen et al., 2018) as an alternative to quantitative methods such as the Fisher information matrix (FIM) analysis (Karakida et al., 2019) and Hessian analysis (Ma et al., 2022; Guiroy et al., 2019). An example use-case of using visualization tools is to determine the impact of the loss landscape structure (e.g., flatness, valleys, and basins) on model generalization and overfitting (Huang et al., 2020; Sypherd et al., 2020; Yang et al., 2021; Prabhu et al., 2019; Xu et al., 2019). Other examples include understanding model optimization (Huang et al., 2020; Ma et al., 2022; Keskar et al., 2017; Sun et al., 2020; Yang et al., 2021), assessing the generalization of Model-Agnostic Multi-task Learning (MAML) (Guiroy et al., 2019), investigating the smoothing effect of noise over sharp minima (Wen et al., 2018), and studying the effectiveness of skip connections in removing bad valleys with sub-optimal minima during optimization (Nguyen, 2019).

3. Proposed Method: *Neuro-Visualizer*

As we have discussed the drawbacks of existing loss landscape visualization methods in Section 1, it is appropriate to echo (Shires & Pickard, 2021)'s thoughts on finding a non-linear manifold "such that the source data lie on, or close to, some low-dimensional manifold embedded within the original high-dimensional space". Hence, we propose using a neural auto-encoder, dubbed *Neuro-Visualizer*, to learn a non-linear manifold that embeds the points of interest (i.e., models) in the high-dimensional loss landscape.

3.1. Formal Definition

Let's assume that we have a trajectory $\mathcal{T}$ that consists of a set of models $\mathcal{M}_{\mathcal{T}} \subset \mathbb{R}^n$ where $\mathbb{R}^n$ is the n-dimensional model parameter space. We want to learn a 2-D manifold $\mathcal{L}$ such that $\mathcal{M}_{\mathcal{T}} \subset \mathcal{L}$. This manifold is to be scaled and visualized as a grid $\mathcal{G} \subset \mathbb{R}^2$. For convenience, and without the loss of generality, we standardize the grid to be strictly $\mathcal{G} = [-1, +1] \times [-1, +1]$. This task naturally poses itself to be solved by auto-encoders, which generally are known for *first* learning a manifold $\mathcal{L}$ that captures the data in $\mathbb{R}^n$ *and then* re-parametrizing that manifold in a lower-dimentional space R^d. In that sense, during its training, our auto-encoder is learning a manifold $\mathcal{L} \subset \mathbb{R}^n$ that contains trajectory points in the high-dimensional input space, and then reparametrizing it as a low-dimensional space $\mathbb{R}^2$ (Bengio et al., 2009). Mathematically, we propose learning a *Neuro-Visualizer* auto-encoder $\mathcal{N} : \mathbb{R}^n \rightarrow \mathbb{R}^n$ which consists of an encoder $E_{\mathcal{N}}$ and a decoder $D_{\mathcal{N}}$, such that $z = E_{\mathcal{N}}(m \in \mathcal{L}) \in \mathcal{G}$ and $m' = D_{\mathcal{N}}(z \in \mathcal{G})$. The visualized grid $\mathcal{G}$ results from sampling points (or coordinates) in the predefined area of interest $[-1, +1] \times [-1, +1] \subset R^2$. Of course, to calculate the loss at a grid point, the sampled coordinates are decoded back to the input space where the manifold lies.

To train the parameters of the auto-encoder $\theta_{\mathcal{N}}$, a reconstruction loss is minimized:

$$\mathrm{L}_{rec} = \mathrm{MSE}_{\mathcal{M}_{\mathcal{T}}} \left[m_i, \mathcal{N}(m_i) \right] \tag{1}$$

As $\mathcal{N}$ gets optimized, it learns a manifold that contains the training data points (e.g., the trajectory models $\mathcal{M}_{\mathcal{T}}$ sampled at equal intervals of epochs).

3.2. Additional Constraints in *Neuro-Visualizer*

While the reconstruction loss is sufficient to guarantee learning a manifold that embeds the trajectory models, it does not guarantee any other properties of this manifold beyond continuity. However, this turns out to be a feature, not a bug, as additional desired properties of the embedding space can be imposed by adding other constraints in the form of loss functions. This is in contrast to baseline linear methods (e.g., PCA), where once the projection method is selected, there is little control over the properties of the resultant plane or manifold. Here, we list some interesting and useful constraints that we later adopt in our experiments in Section 4. This list, however, is not exhaustive; *Neuro-Visualizer* is flexible and can be customized with many other possible constraints. These constraints can also be combined as a weighted sum in a multi-task learning formulation. Thus, the total loss would be $\mathrm{L}_{total} = \mathrm{c}_{rec}\mathrm{L}_{rec} + \sum_i c_i L_i$. In Table 2 in Appendix A, we provide the values of the weighing coefficients, along with the rest of the hyper-parameters, for each experiment. These hyper-parameter were tuned such that when minimizing the total loss, the individual losses are also to minimized appropriately.

3.2.1. Location anchoring constraints:

This type of constraints anchors a set of points (e.g., trajectory models) onto certain locations on the grid. This helps orient the training trajectory such that certain aspects of the optimization process are highlighted. The general form of a location anchoring constraint is:

$$\mathrm{L}_{anch} = \mathrm{MSE}_{\mathcal{M}_{\mathcal{T}}' \subseteq \mathcal{M}_{\mathcal{T}}} \left[E_{\mathcal{N}}(m_i), \mathcal{A}_i \right], \tag{2}$$

where $\mathcal{A} \subset \mathcal{G}$ is the set of desired anchoring points on the grid and $\mathcal{M}_{\mathcal{T}}'$ is the set of models that correspond to those anchoring points. In this work, we chose to demonstrate three examples of anchoring constraints as detailed below (see Appendix A for further details):

- *Polar pinning (L_{anch1})* : This constraint places the trajectory's first and last models at the bottom left and top right corners of the grid, respectively. This helps stretch the trajectory across the grid and utilize the entire space.
- *Center pinning (L_{anch2})* : This constraint positions the last model at the center of the grid—a perspective suitable for showing the final stages of optimization in detail.
- *Circle pinning (L_{anch3})* : This constraint positions the trajectory models at equal distant from each other on a circle with a specified radius.

3.2.2. Grid scaling constraints:

Another type of constraints can be devised to ensure the grid has a certain scale. Unlike PCA, *Neuro-Visualizer*'s grid does not generally have a uniform and linear scale. Rather, it is more flexible with a variable scaling factor across the grid, allowing it to show more details at certain areas while zooming out on the rest. To capitalize on this property, we construct a constraint to capture and control the zooming behavior as follows. We scale the grid such that vicinity of trajectory models, an area which of particular interest and importance, has a relatively higher density relative to the rest of the grid. To formalize this, we construct the following grid scaling loss:

$$\mathrm{L}_{grid} = \mathrm{MSE}_{m \in \mathcal{M}_{\mathcal{G}}} \left[\log\left(d_m\right) - l_m, \log\left(d^{max}\right) - l^{max} \right] \tag{3}$$

where d_m is the distance between a grid mesh-point m and the closest trajectory point to it in the parameter space, d^{max} is the distance between the first and last model on the trajectory, and l_m is the distance equivalent to d_m in the grid space. Finally, l^{max} is a hyper-parameter chosen based on the desired scaling factor. By minimizing L_{grid}, a constant logarithmic scale between grid space and parameter space distances is enforced. More formally, the function of L_{grid} is to maintain a proportional relationship between the distance l_m in the 2D-grid space, from a grid mesh-point to its nearest trajectory point, and the corresponding distance d_m in the original parameter space. This proportionality is maintained by setting the ratio $\frac{\log(d_m)}{l_m}$ equal to $\frac{\log(d^{max})}{l^{max}}$, where d^{max} represents the distance between the first and last points on the trajectory in the original parameter space, and l^{max} is a hyper-parameter chosen to control the scaling factor. The logarithmic form of this ratio is used to ensure computational feasibility. The larger the value of l^{max}, the greater the emphasis or 'zooming effect'.

4. Results and Applications

While assessing the "correctness" of loss landscape visualization methods is non-trivial (see Appendix B), we demonstrate the usefulness of *Neuro-Visualizer* in discovering novel insights and its advantages compared to existing loss landscape visualization methods in the context of two KGML applications. While we focus mainly on KGML applications in this paper, we also demonstrate *Neuro-Visualizer*'s usefulness beyond KGML applications in Appendix D

4.1. Applying *Neuro-Visualizer* on *CoPhy*-PGNN

To show the effectiveness of *Neuro-Visualizer*, we take *CoPhy*-PGNN (Elhamod et al., 2022) in quantum mechanics as a use-case application. Within the Knowledge-Guided Machine Learning (*KGML*) framework (Karpatne et al., 2017), *CoPhy*-PGNN is a neural network that is trained with adaptively balanced physics loss terms in order to solve eigen-value problems with better generalization than a purely data-driven neural network (see Appendix C for details).

4.1.1. Qualitative and Quantitative Assessments of *Neuro-Visualizer* Against Other Baselines.

Figure 1 compares *Neuro-Visualizer*'s and other baselines' overall physics loss landscapes for *CoPhy*-PGNN. Note that the contours in all sub-figures have been scaled equally for fair comparison. Zoomed-in perspectives of the same landscapes are shown in Figure 2. We make the following observations.

First, ***Neuro-Visualizer* shows richer details.** In Figure 1, both PCA and *Neuro-Visualizer* show that, due to optimizing multiple conflicting loss terms, the training took a detour before descending into a terminal minima. *Neuro-Visualizer*'s story in Figure 1b, however, is much richer. Here, we can see that the "terminal" minima is not truly a simple minima, but rather a basin of a complex surface and with many local minima, causing the model to bounce around during the final stages of its optimization. This observation becomes even clearer as we zoom into the convergence area (see Figure 2b). Being able to observe such complexity of the loss landscape is crucial for determining how well-designed the loss terms are.

Second, ***Neuro-Visualizer*'s manifold fits the trajectory models better.** Looking at the model colors, which corresponds to their loss values, in Figure 1 and Figure 2, we can see that the actual model loss values for PCA (Figure 2a) do not quite match with their corresponding locations on the learned manifold (i.e., point and contour colors do not generally match at their corresponding locations). The explanation for this discrepancy is that PCA is a linear method that only intersects with the high-dimensional trajectory at one point. This is unlike *Neuro-Visualizer* (Figures 1b and 2b) where the points match in color with their underlying contours because the learned 2-D manifold passes through all the trajectory models, with good approximation.

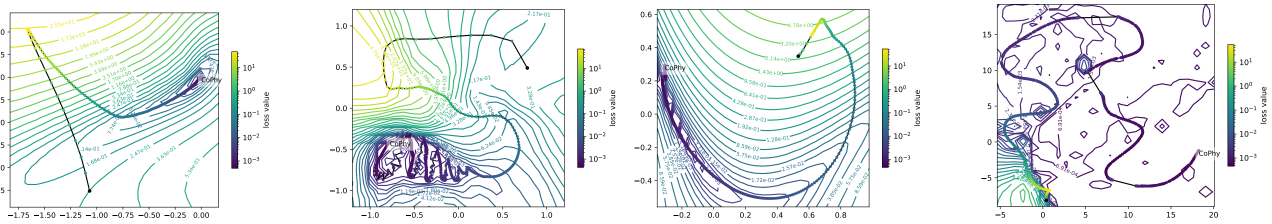
Metric	*Neuro-Visualizer*	PCA	Kernel-PCA	UMAP
$e_{\textbf{relative}}$	0.0095	1.6782	4.7250	0.4295
$e_{\textbf{proj}}$	0.0005	0.2832	0.0865	0.2307

Table 1. A quantitative comparison of *Neuro-Visualizer* against other baselines in terms of average relative physics loss error $e_{\textbf{relative}}$ and average projection error $e_{\textbf{proj}}$. *Neuro-Visualizer* outperforms all other baselines across the board.

To compare *Neuro-Visualizer* to other non-linear methods, we visualize the same trajectory using UMAP (McInnes et al., 2018) and Kernel-PCA (Schölkopf et al., 1997) with an RBF kernel in Figure 1. Compared to *Neuro-Visualizer*, we make two observations. First, UMAP (Figure 1d) uses almost the entire grid to show the local minima at which the model arrives, with little attention to the initial stage of optimization. Thus, UMAP fails to give the full picture of the training trajectory when compared to *Neuro-Visualizer*. Similarly, while Kernel-PCA is a non-linear method, its landscape visualization (Figure 1c) looks too simplistic with little insight to provide beyond that of PCA's, even when zoomed in (Figure 2c). Appendix E provides more visualizations on the projection and loss errors of *Neuro-Visualizer* compared to the other baselines. For a more quantitative assessment, however, we provide some comparative metrics in Table 1. By looking at both the average relative error in loss values, $e_{\text{relative}} = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{|L_{m_i} - L_{N(m_i)}|}{L_{m_i}}\right)$, and the average projection error in the parameter space, $e_{\text{proj}} = \frac{1}{n}\sum_{i=1}^{n}|m_i - N(m_i)|$, we can see that *Neuro-Visualizer* outperforms all the other methods by orders of magnitude. Here, m_i denotes the i^{th} sampled model on the learning trajectory, and $N(m_i)$ represents the reconstruction of m_i obtained using the auto-encoder in our method, or the inverse transform in other baseline methods such as PCA. L_{m_i} is the original loss of m_i, while $L_{N(m_i)}$ is the loss of the reconstructed model.

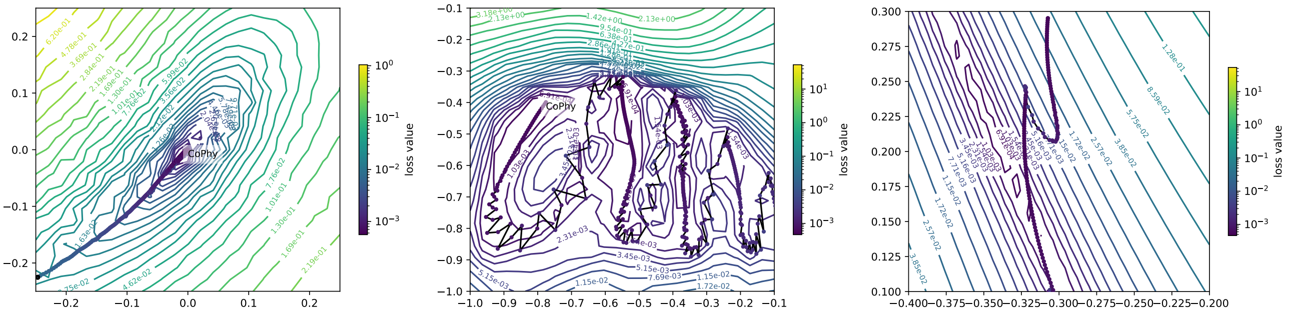
4.1.2. Using *Neuro-Visualizer* to Study the Advantages of the *CoPhy*-PGNN Approach.

We here show *Neuro-Visualizer*'s usefulness for comparing different models by plotting the trajectories of *CoPhy*-PGNN and its baseline Black-box Neural Network in Figure 3. Both trajectories start from the same model initialization (marked with a thick border). Figures 3a and 3c use PCA to visualize Test-MSE and the spectrum loss, S-Loss, respectively, where S-Loss is one of the two physics losses used to train the model. Figures 3b and 3d show the corresponding loss landscapes for *Neuro-Visualizer*. We notice that PCA utterly fails at capturing Black-box Neural Network's critical points (Figure 3c), PCA not only misses Black-box Neural Network's minima, but also shows inconsistency in its loss values (i.e., the model colors indicate that it is descending, while the contours indicate the opposite). This is in contrast to *Neuro-Visualizer*, where Black-box Neural Network's and *CoPhy*-PGNN's minima



(a) PCA (b) *Neuro-Visualizer* (c) Kernel-PCA (d) UMAP

Figure 1. A comparison of *Neuro-Visualizer* and other baselines in terms of the consistency of *CoPhy*-PGNN's overall physics loss values between trajectory models and their corresponding manifold projections. Clearly, *Neuro-Visualizer* shows richer details and a manifold that better fits the trajectory models.



(a) PCA (zoomed-in) (b) *Neuro-Visualizer* (zoomed-in) (c) Kernel-PCA (zoomed-in)

Figure 2. A zoomed-in perspective of the landscapes visualized in Figure 1

are distinctly visualized with high consistency in terms of loss values. Another inconsistency is found by looking at Black-box Neural Network's trajectory in Figure 3a. Here, PCA shows that it crosses the $2.28\mathrm{e}{-02}$ contour twice, implying that the trajectory descends and then ascends again. This, however, is false as can be verified by looking at the trajectory model colors and how they become darker as the final model is approached, suggesting that the loss monotonically decreases. This contradiction leads to confusion that is not present in *Neuro-Visualizer*'s case in Figure 3b, making the latter more useful.

4.2. Applying *Neuro-Visualizer* on PINNs

We here explore *Neuro-Visualizer*'s usefulness in studying Physics-Informed Neural Networks (PINNs), which have been widely and successfully used by many researchers in recent years to solve partial differential equations (PDEs), which appear in many real-world engineering and scientific applications. Moreover, PINNs present themselves as a convenient test-bed for our proposed landscape visualization method due to the significant effect of varying optimization hyper-parameters on PINN performance.

For training PINNs, the loss terms that play a role are the residual loss L_r, the initial condition loss L_{ic}, and the boundary condition loss L_{bc}. The total loss being optimized is:

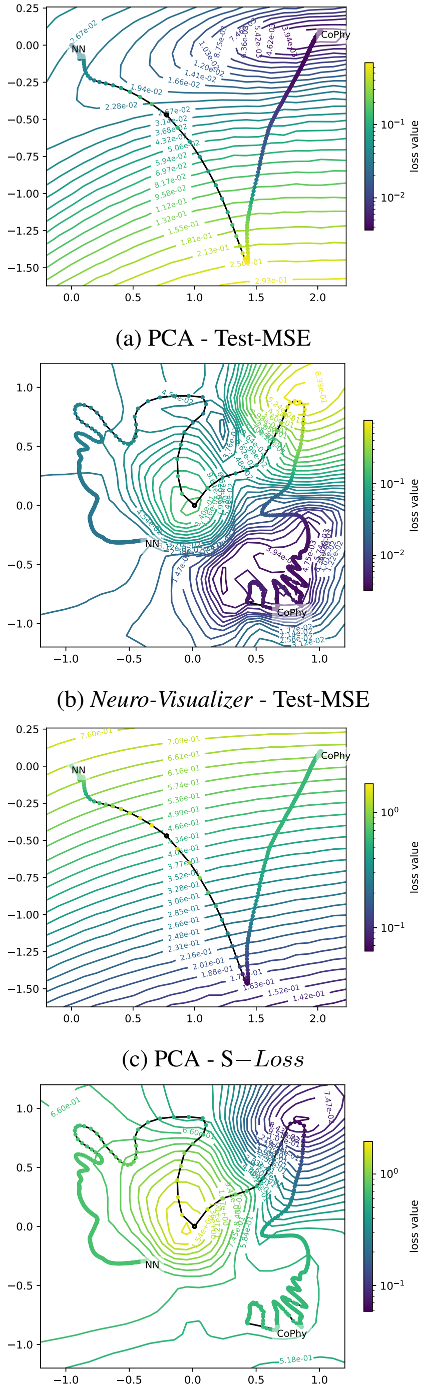
$$\mathrm{L}_{total} = \mathrm{c}_r \times \mathrm{L}_r + \mathrm{c}_{ic} \times \mathrm{L}_{ic} + \mathrm{c}_{bc} \times \mathrm{L}_{bc}. \qquad (4)$$

See Appendix F for a detailed literature review on PINNs.

4.2.1. Demonstrating *Neuro-Visualizer*'s Flexibility With Different Constraints.

As discussed in Section 3.2, one of *Neuro-Visualizer*'s advantages is its ability to warp the learned manifold to satisfy certain constraints. To demonstrate this, we use PINN for the Convection equation as a target application. We set $\beta = 30$, a high value, making the PDE harder to solve and the loss landscape more complex and interesting to visualize (see Appendix F for details).

Figure 4 shows a progression of *Neuro-Visualizer* models visualizing L_{test} (i.e., the prediction error at test domain points) of the same PINN model. However, these *Neuro-Visualizer* models are trained with different constraints. First, Figure 4a shows the loss landscape with no constraints. Subsequent sub-figures show the different manifolds obtained by varying the *Neuro-Visualizer*'s training constraints. Figure 4b uses L_{anch1}. As a result, the trajectory stretches almost perfectly across the grid between two opposite corners. Alternatively, to show the effect of L_{grid}, we use a large $l^{max} = 8$ to impose high grid den-



(a) PCA - Test-MSE

(b) *Neuro-Visualizer* - Test-MSE

(c) PCA - S$-Loss$

(d) *Neuro-Visualizer* - S$-Loss$

Figure 3. The loss landscapes of *CoPhy*-PGNN and Black-box Neural Network for different loss terms using PCA and *Neuro-Visualizer*. Comparing the two, it is clear that *Neuro-Visualizer* tells a more accurate and insightful story.

sity at the vicinity of the trajectory models. As expected, Figure 4c shows a grid that zooms almost entirely onto the vicinity of the trajectory, highlighting more details of that area compared to the previous sub-figures.

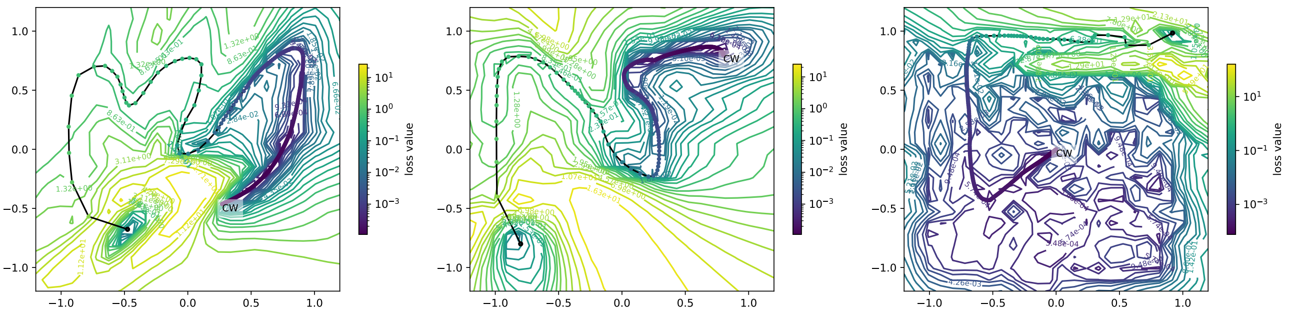
We further study the effect of L_{grid} by varying l^{max}, the hyper-parameter that correlates with the grid density near trajectory models. Figure 5 shows the *density* landscape (i.e., the color-coding indicates grid density and not loss values) for two different values of l^{max}. We use a CKA-similarity-based density that is defined as:

$$\rho_{m \in \mathcal{M}_\mathcal{G}} = \sum_{m' \in \mathcal{M}_\mathcal{G} \setminus \{m\}} \mathrm{CKA}(m', m), \tag{5}$$

where $\mathrm{CKA}(m', m)$ is the CKA similarity measure between two neural networks as defined in (Kornblith et al., 2019). As can be seen in Figure 5, the density of the grid especially near trajectory models increases with l^{max}. This shows that L_{grid} can be a vital tool for engineering the manifold visualization through the appropriate choice of l^{max}.

4.2.2. Using *Neuro-Visualizer* to Study PINNs' Training Pathologies.

Next, we use *Neuro-Visualizer* to visually verify (Wang et al., 2022)'s findings on PINNs through the lens of Neural Tangent Kernel (NTK). In their work, the authors hypothesize that PINN training pathologies result from a discrepancy in the convergence rate of the different loss components. More precisely, they show that the PDE's residual loss (L_r) converges faster than the boundary condition loss (L_{bc}), leading to a sub-optimal model. To verify this claim, we train a *Neuro-Visualizer* to visualize the loss landscape for the two different optimization approaches considered in (Wang et al., 2022). Namely, an approach that trains with constant loss weighting, and another with NTK-based adaptive loss weighting. An L_{anch3} pinning constraint is imposed to place the models over the perimeter of a circle, making it easy to compare the two approaches. As can be seen in Figure 6, the authors' claim is easily verifiable using our proposed visualization method. First, in line with the authors' hypothesis, while the terminal L_r and L_{ic} are relatively worse for the NTK-based approach, its terminal L_{bc} is much better, indicating that an optimal L_{bc} is essential for obtaining a well-trained PINN. Another observation we make in accordance with their paper is that the NTK-based approach reaches flat minima for all losses at a somewhat similar cadence. This is in contrast to the baseline model where the different losses converge at variable rates, traversing loss landscapes that are less flat and of variable slopes.

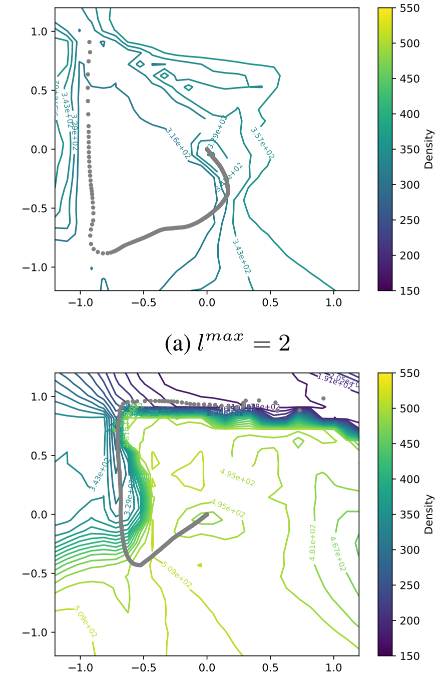


(a) vanilla

(b) with L_{anch1}

(c) with L_{anch2} and L_{grid}

Figure 4. A series of *Neuro-Visualizer* landscape visualizations of L_r with different constraints. Notice the versatility of the proposed method and its ability to learn the desired manifold by employing appropriate constaints.



(a) $l^{max} = 2$

(b) $l^{max} = 8$

Figure 5. A comparison of two *Neuro-Visualizer density* landscape visualizations as a result of using L_{grid} with two different l^{max} values. As can be seen, through this constraint, *Neuro-Visualizer* grants its user greater flexibility to decide the appropriate zooming factor.

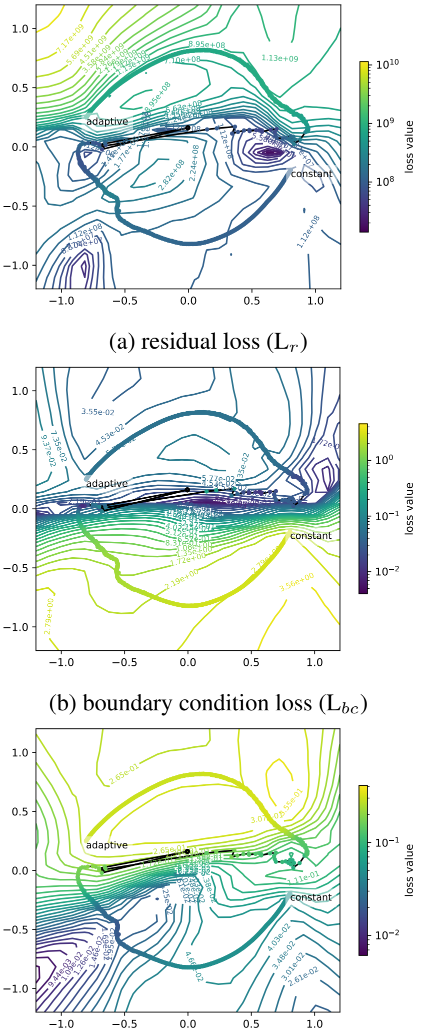
4.2.3. Using *Neuro-Visualizer* To Study PINNs' Failure Modes.

Inspired by (Krishnapriyan et al., 2021)'s work on understanding the effect of the PDE's complexity and PINN's regularization on the loss landscape and optimization outcome, we use *Neuro-Visualizer* to study PINN performance on the convection equation. Namely, we want to validate whether a higher β parameter in a convection PDE or an increase in PINN regularization (i.e., an increase in the value of c_r) leads to a more complex loss landscape that is harder to optimize. As such, we run an experiment where we train PINNs with varying β values, and then compare their loss landscapes using *Neuro-Visualizer*. A similar experiment on c_r can be found in Appendix G.

Looking at Figure 7, it is easy to verify that an increase in β renders the loss landscape more non-convex and harder to optimize. This agrees with the authors' findings. However, it is worth noting that the loss landscape visualizations presented in (Krishnapriyan et al., 2021) used linear methods. When Figure 7 is compared to its counterpart (i.e., Figure 3 in (Krishnapriyan et al., 2021)), it is evident that their loss landscape visualizations are less intuitive and hard to visually interpret without the authors' commentary, indicating that our proposed method is more effective.

4.2.4. Investigating Different Loss Balancing Techniques.

Balancing different loss terms in frameworks such as multitask learning (MTL) or *KGML* can be a daunting task (Wang et al., 2022; 2021; Elhamod et al., 2022). Thus, the ability to compare different loss balancing techniques is an important research effort. Generally, different loss balancing techniques are compared based on the final model's accuracy. However, this metric does not provide a fundamental understanding of the optimization process and whether a loss balancing algorithm is more suitable for one specific task than another.

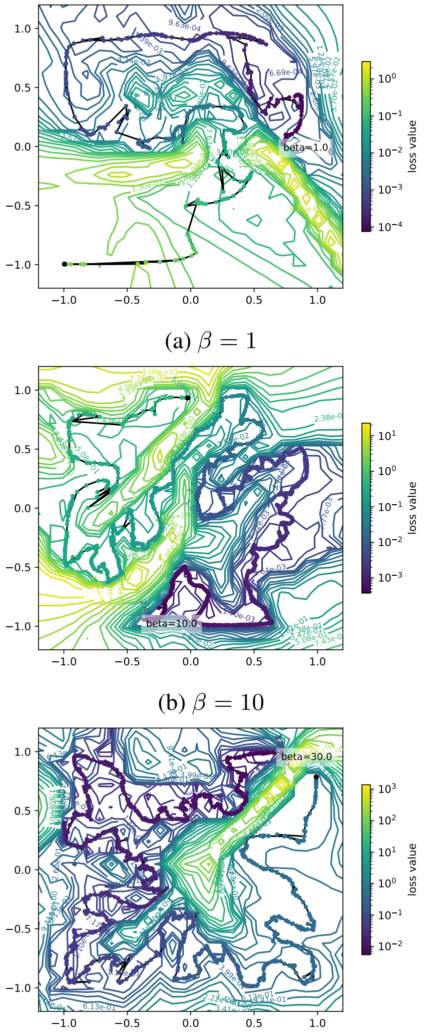


(a) residual loss (L$_r$)

(b) boundary condition loss (L$_{bc}$)

(c) initial condition loss (L$_{ic}$)

Figure 6. A comparison of the two optimization approaches studied by (Wang et al., 2022) using *Neuro-Visualizer*. This visualization verifies the authors' claims and shows that an NTK-based adaptive approach emphasizes on a better optimization of the boundary condition loss and reaches flatter minima across all losses.



(a) $\beta = 1$

(b) $\beta = 10$

(c) $\beta = 30$

Figure 7. A comparison of PINNs solving convection PDEs of varying βs. *Neuro-Visualizer* verifies the claim of (Krishnapriyan et al., 2021) that increasing β causes L_{total}'s landscape to become more non-convex and difficult to optimize.

Here, we consider six different loss balancing methods that are commonly used in the literature; namely Equal Weights (*EW*), Constant Weights (*CW*), Dynamic Weight Averaging (*DWA*), Learning Rate Annealing (Wang et al., 2021) (*LR*$_{annealing}$), Gradient Normalization (Chen et al., 2018) (*GradNorm*), and Random Loss Weighting (Lin et al., 2022) (*RLW*). Appendix I gives a detailed account of these algorithms.

To compare these algorithms, starting from the same model initialization, we train multiple PINNs with the listed algorithms to solve the Convection problem with $\beta = 10$.

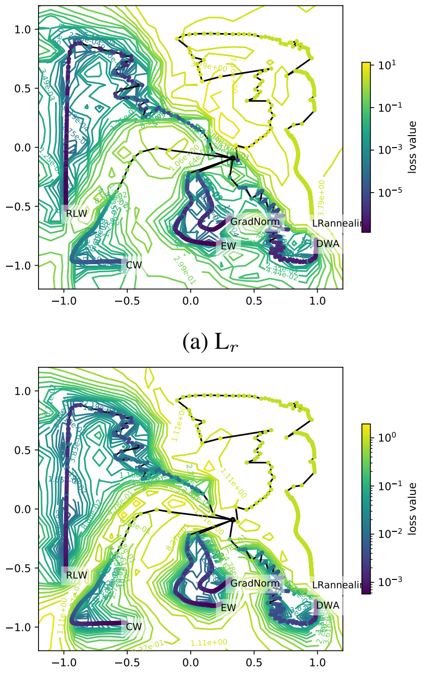
In Figure 8, we inspect the landscapes of two loss terms: L_r and L_{test}. Using *Neuro-Visualizer*, the different trajectories and minima can be easily found and compared. In particular, it is clear that while *GradNorm* and *EW* take slightly different trajectories, they converge to the same minima. On the other hand, *DWA* converges to the same basin as *GradNorm* and *EW* w.r.t L_r, but not the same minima. Additionally, looking at Figure 8b, it is clear that *LR*$_{annealing}$'s trajectory is nowhere near a minima w.r.t L_{test}. This is surprising since *LR*$_{annealing}$ has shown success at solving the Helmholtz equation and the Klein Gordon equation as demonstrated in (Wang et al., 2021), implying that not all PINN tasks benefit from this method. The valuable insight that was visually, easily, and intuitively inferred from *Neuro-Visualizer* in Figure 8 would not have been possible with a baseline method such as PCA due to its linear scale and planar manifold. Appendix H compares our results to those obtained using PCA, further verifying our claim.

5. Conclusions and Future Work

In this paper, we have shown that our proposed auto-encoder-based method, *Neuro-Visualizer*, is capable of learning non-linear manifolds in the input model parameter space and mapping such manifolds onto a 2-D grid for loss landscape visualization. Additionally, we have demonstrated that *Neuro-Visualizer* surpasses many other linear and non-linear landscape visualization approaches in terms of representation accuracy and malleability to learning manifolds with user-defined properties. Finally, we used two applications, *CoPhy*-PGNN (Elhamod et al., 2022) and PINNs (Raissi et al., 2017a), to derive insightful findings, including the importance of certain hyper-parameters for neural network training and the efficacy of different deep learning frameworks such as Multi-Task Learning (MTL) and Knowledge-Guided Machine Learning (*KGML*). Future work could explore the full potential of *Neuro-Visualizer* by using it to study other deep learning phenomena, such as the relationship between landscape sharpness and generalizability (Huang et al., 2020).

One of *Neuro-Visualizer*'s limitations is the lack of scalability for large input parameter spaces. For example, the PINNs studied in this paper have an input dimensionality of the order of $10,000$ parameters, which is relatively small. In contrast, computer vision applications generally use much larger models (e.g., a VQ-GAN (Esser et al., 2021) has approximately 88 million parameters). Such a large input space would be prohibitively difficult for *Neuro-Visualizer* to encode directly. A future research direction could investigate appropriate ways to encode and visualize such large models effectively.

Finally, despite the impressive results achieved with *Neuro-Visualizer* in comparison to current state-of-the-art methods, we find that the implicit assumption baked into auto-encoders have not been fully studied yet. Some literature (Qian et al., 2019; Berthelot* et al., 2019) shows that interpolation in the learned manifold of an auto-encoder is a complex research topic and depends on several factors, such as the data itself and the architecture of the auto-encoder. As such, in this work, we make no concrete assumptions in regards to interpolations in the learned manifold. We look forward to further investigating this area theoretically and empirically in future work.



(a) L_r

(b) L_{test}

Figure 8. Neuro-Visualizer's loss landscapes of PINN models with different loss balancing methods intuitively and visually provide valuable insights that would otherwise be hard to extract.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgements

We would like to thank the reviewers for their valuable feedback on this work. Access to computing facilities was provided by the Advanced Research Computing (ARC) Center at Virginia Tech.

References

Bengio, Y. et al. Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127, 2009.

Berthelot*, D., Raffel*, C., Roy, A., and Goodfellow, I. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1fQSiCcYm.

Chatzimichailidis, A., Keuper, J., Pfreundt, F.-J., and Gauger, N. R. Gradvis: Visualization and second order analysis of optimization surfaces during the training of deep neural networks. In *2019 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC)*, pp. 66–74, 2019. doi: 10.1109/MLHPC49564.2019.00012.

Chen, Z., Badrinarayanan, V., Lee, C.-Y., and Rabinovich, A. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/chen18a.html.

Elhamod, M., Bu, J., Singh, C., Redell, M., Ghosh, A., Podolskiy, V., Lee, W.-C., and Karpatne, A. Cophy-pgnn: Learning physics-guided neural networks with competing loss functions for solving eigenvalue problems. *ACM Transactions on Intelligent Systems and Technology*, 13 (6):1–23, 2022.

Esser, P., Rombach, R., and Ommer, B. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Gao, H., Sun, L., and Wang, J.-X. Phygeonet: Physics-informed geometry-adaptive convolutional neural networks for solving parameterized steady-state pdes on irregular domain. *Journal of Computational Physics*, 428:110079, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.110079. URL https://www.sciencedirect.com/science/article/pii/S0021999120308536.

Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 8789–8798. Curran Associates, Inc., 2018.

Goodfellow, I. J., Vinyals, O., and Saxe, A. M. Qualitatively characterizing neural network optimization problems, 2014.

Guiroy, S., Verma, V., and Pal, C. Towards understanding generalization in gradient-based meta-learning, 2019.

Horoi, S., Huang, J., Rieck, B., Lajoie, G., Wolf, G., and Krishnaswamy, S. Exploring the geometry and topology of neural network loss landscapes. In Bouadi, T., Fromont, E., and Hüllermeier, E. (eds.), *Advances in Intelligent Data Analysis XX*, pp. 171–184, Cham, 2022. Springer International Publishing. ISBN 978-3-031-01333-1.

Huang, W. R., Emam, Z., Goldblum, M., Fowl, L., Terry, J. K., Huang, F., and Goldstein, T. Understanding generalization through visualizations. In Zosa Forde, J., Ruiz, F., Pradier, M. F., and Schein, A. (eds.), *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pp. 87–97. PMLR, 12 Dec 2020. URL https://proceedings.mlr.press/v137/huang20a.html.

Jin, X., Cai, S., Li, H., and Karniadakis, G. E. Nsfnets (navier-stokes flow nets): Physics-informed neural networks for the incompressible navier-stokes equations. *Journal of Computational Physics*, 426:109951, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2020.109951. URL https://www.sciencedirect.com/science/article/pii/S0021999120307257.

Karakida, R., Akaho, S., and Amari, S.-i. The normalization method for alleviating pathological sharpness in wide neural networks. *Advances in neural information processing systems*, 32, 2019.

Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A., Shekhar, S., Samatova, N., and Kumar, V. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions* *on Knowledge and Data Engineering*, 29(10):2318–2331, 2017.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=H1oyRlYgg.

Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pp. 3519–3529. PMLR, 2019.

Krishnapriyan, A., Gholami, A., Zhe, S., Kirby, R., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31*, pp. 6389–6399. Curran Associates, Inc., 2018.

Li, J. and Li, B. Mix-training physics-informed neural networks for the rogue waves of nonlinear schrödinger equation. *Chaos, Solitons & Fractals*, 164:112712, 2022.

Liang, S. and Zhang, Y. A simple general approach to balance task difficulty in multi-task learning. *arXiv preprint arXiv:2002.04792*, 2020.

Lin, B., Feiyang, Y., Zhang, Y., and Tsang, I. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*, 2022.

Ma, C., Kunin, D., Wu, L., and Ying, L. Beyond the quadratic approximation: the multiscale structure of neural network loss landscapes. *arXiv preprint arXiv:2204.11326*, 2022.

McInnes, L., Healy, J., Saul, N., and Großberger, L. Umap: Uniform manifold approximation and projection. *Journal of Open Source Software*, 3(29):861, 2018.

Mei, S., Montanari, A., and Nguyen, P.-M. A mean field view of the landscape of two-layers neural networks, 2018.

Nguyen, N. Q. Optimization landscape of deep neural networks. 2019.

Nguyen, Q., Mukkamala, M. C., and Hein, M. On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749*, 2018.

Prabhu, V. U., Yap, D. A., Xu, J., and Whaley, J. Understanding adversarial robustness through loss landscape geometries. *arXiv preprint arXiv:1907.09061*, 2019.

Qian, S., Li, G., Cao, W.-M., Liu, C., Wu, S., and Wong, H.-S. Improving representation learning in autoencoders via multidimensional interpolation and dual regularizations. In *IJCAI*, pp. 3268–3274, 2019.

Raissi, M., Perdikaris, P., and Karniadakis, G. Physics informed deep learning (part i): Data-driven solutions of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10561*, 2017a.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. Physics informed deep learning (part ii): Data-driven discovery of nonlinear partial differential equations. *arXiv preprint arXiv:1711.10566*, 2017b.

Raissi, M., Perdikaris, P., and Karniadakis, G. E. 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.

Roweis, S. T. and Saul, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290 (5500):2323–2326, 2000. doi: 10.1126/science.290.5500.2323. URL https://www.science.org/doi/abs/10.1126/science.290.5500.2323.

Schölkopf, B., Smola, A., and Müller, K.-R. Kernel principal component analysis. In Gerstner, W., Germond, A., Hasler, M., and Nicoud, J.-D. (eds.), *Artificial Neural Networks — ICANN'97*, pp. 583–588, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. ISBN 978-3-540-69620-9.

Shires, B. W. B. and Pickard, C. J. Visualizing energy landscapes through manifold learning. *Phys. Rev. X*, 11:041026, Nov 2021. doi: 10.1103/PhysRevX.11.041026. URL https://link.aps.org/doi/10.1103/PhysRevX.11.041026.

Snyder, J. P. The robinson projection—a computation algorithm. *Cartography and Geographic Information Systems*, 17(4):301–305, 1990.

Snyder, J. P. *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

Sun, R., Li, D., Liang, S., Ding, T., and Srikant, R. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5):95–108, 2020. doi: 10.1109/MSP.2020.3004124.

Sypherd, T., Diaz, M., Sankar, L., and Dasarathy, G. On the alpha-loss landscape in the logistic model, 2020.

Wang, J. *Laplacian Eigenmaps*, pp. 235–247. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-27497-8. doi: 10.1007/978-3-642-27497-8_12. URL https://doi.org/10.1007/978-3-642-27497-8_12.

Wang, S., Teng, Y., and Perdikaris, P. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021. doi: 10.1137/20M1318043. URL https://doi.org/10.1137/20M1318043.

Wang, S., Yu, X., and Perdikaris, P. When and why pinns fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022.

Wen, W., Wang, Y., Yan, F., Xu, C., Wu, C., Chen, Y., and Li, H. Smoothout: Smoothing out sharp minima to improve generalization in deep learning, 2018.

Xu, J., Yap, D. A., and Prabhu, V. U. Understanding adversarial robustness through loss landscape geometries. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, pp. 18, 2019.

Yang, Y., Hodgkinson, L., Theisen, R., Zou, J., Gonzalez, J. E., Ramchandran, K., and Mahoney, M. W. Taxonomizing local versus global structure in neural network loss landscapes, 2021.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pp. 581–590, 2020. doi: 10.1109/BigData50022.2020.9378171.

A. Hyper-parameter Selection and Additional Implementation Details

In all our experiments, the *Neuro-Visualizer* is a vanilla multi-layer perceptron (MLP) architecture. This MLP maps the weights of neural network models to 2D-manifolds and subsequently decodes points from these manifolds back into neural network weights. For specifics on the MLP configurations used across different experiments, please refer to Table 2 below.

To guarantee that an input model $m \in \mathcal{L}$ is encoded into $z \in [-1, +1] \times [-1, +1]$, a *tanh* activation function is appended to the output of the encoder $E_{\mathcal{N}}$. Also, m is normalized for an easier training of $\mathcal{N}$.

In terms of trajectories, they are sampled at equal intervals of epochs. These intervals are adjusted based on the complexity of the model and the computational resources available. You can refer to Table 2 for details on the number of models sampled per trajectory in each experiment.

With regards to location anchoring constraints L$_{anch}$, we here define the three different examples mentioned in Section 3.2.1 more rigorously:

- *Polar pinning (L_{anch1})* : This can be formally defined by setting $\mathcal{M_T}' = \{m_0, m_{|\mathcal{M_T}|}\}$ (i.e., the set that contains the first and last models in the trajectory), and $\mathcal{A} = \{(-r, -r), (r, r)\}$, with an $r = 0.8$.
- *Center pinning (L_{anch2})* : It is formally defined with $\mathcal{M_T}' = \{m_{|\mathcal{M_T}|}\}$ and $\mathcal{A} = \{(0, 0)\}$.
- *Circle pinning (L_{anch3})* : It is formally defined with $\mathcal{M_T}' = \{m_{|\mathcal{M_T}|}\}$ and $\mathcal{A} = \{(r \cdot \sin(\frac{2\pi k}{n}), r \cdot \cos(\frac{2\pi k}{n}))$ where $k \in \{0, 1, ..., n-1\}\}$

It is worth mentioning that we use another constraint, L$_{traj}$, in some of our experiment to help space the trajectory models equally so that they are spread out across the grid. The loss term for this constraint penalizes for the differences in step sizes in the latent grid space between consecutive trajectory models.

Table 2 shows the hyper-parameters used to train *Neuro-Visualizer* for both *CoPhy*-PGNN and PINNs.

Finally, the experiments in this paper were run on Nvidia DGX A100 GPUs. Each experiment needed a single GPU and 8 CPU cores.

B. On the "Correctness" of Loss Landscape Visualization Methods

Loss landscape visualization is a subjective tool that provides a qualitative and holistic description of the landscape, rather than a quantitative one. As such, the notion of "correctness" is not the best vantage point from which this tool can be appreciated. One analogy that clarifies this point is map projections. Earth map projections represent the 3-D Earth surface on a 2-D plane. Different projection methods have different advantages and limitations; each method makes certain features or characteristics of the Earth surface more or less prominent. Thus, there are many valid projections, each useful for different purposes. One popular method is the Mercator projection (Snyder, 1997), which is useful for navigation because it preserves angles and directions. However, it distorts the size and shape of objects near the poles. Another method is the Robinson projection (Snyder, 1990), which balances distortions of size and shape, making it a good all-purpose projection. A third method is the Winkel tripel projection (Snyder, 1997) which accurately shows the relative sizes and shapes of landmasses, but still distorts the shapes of some landmasses and oceans.

Similarly, a non-linear loss landscape visualization method will produce a non-linear manifold that gets distorted when visualized on a 2-D planar grid. . Thus, while the user should be aware of these distortions and understand the visualization accordingly, the method is still valid and useful for understanding the properties of the loss surface. And while there are infinitely many manifolds that could contain a set of points in the parameter space (i.e., models), it is important to select the manifold(s) that exhibits the desired user-defined criteria, such as the scaling factors at different parts of the manifold.

C. A Brief Description of *CoPhy*-PGNN

CoPhy-PGNN (Elhamod et al., 2022), short for Competing Physics Physics-Guided Neural Networks, is a model uniquely tailored to solve eigenvalue problems, which are prevalent in scientific domains such as quantum mechanics and electromagnetic propagation. In addition to the data-drive Train-Loss, two physics-guided (PG) loss terms are used: C-Loss and S-Loss.C-Loss is designed to enforce the eigen-equation, and its minimization can lead to multiple solutions that satisfy the physical constraints of the problem. However, these multiple solutions may correspond to different energy levels. Generally,

	Encoder hidden-Layer sizes	Batch size	Epochs	LR	Models sampled per trajectory	Notes
CoPhy-PGNN. Section 4.1	$3141, 270, 23$	32	$40,000$	10^{-4}	500	
PINNs - ”Investigating Different Loss Balancing Techniques”. Section 4.2.4	$991, 125, 15$	32	$600,000$	5×10^{-4}	300	$c_{rec} = 10^4$
PINNs - ”Using *Neuro-Visualizer* To Study PINNs’ Training Pathologies”. Section 4.2.2	$1000, 500, 100$	32	$100,000$	10^{-5}	800	$c_{anch3} = 10^4$
PINNs - ”Using *Neuro-Visualizer* To Study PINNs’ Failure Modes”. Section 4.2.3	$991, 125, 15$	32	$600,000$	5×10^{-4}	Varies between 228 and 1678 (Early-stopping used)	$c_{traj} = 1$
PINNs - ”Demonstrating *Neuro-Visualizer*’s Flexibility With Different Constraints”. Section 4.2.1	$991, 125, 15$	32	$80,000$	10^{-4}	300	$c_{anch1} = 10^2$ $c_{anch2} = 10^2$, $c_{grid} = 1$

Table 2. A table of the hyper-parameter values used to train *Neuro-Visualizer* for each experiment.

only one solution is desired. This is where S-Loss comes into play. S-Loss guides the network towards the specific energy level of interest by minimizing the difference between the predicted and target eigenvalues

The overall learning objective of *CoPhy*-PGNN is:

$$E(t) = \text{Train-Loss} + \lambda_C(t)\ C\text{-Loss} + \lambda_S(t)\ S\text{-Loss} \tag{6}$$

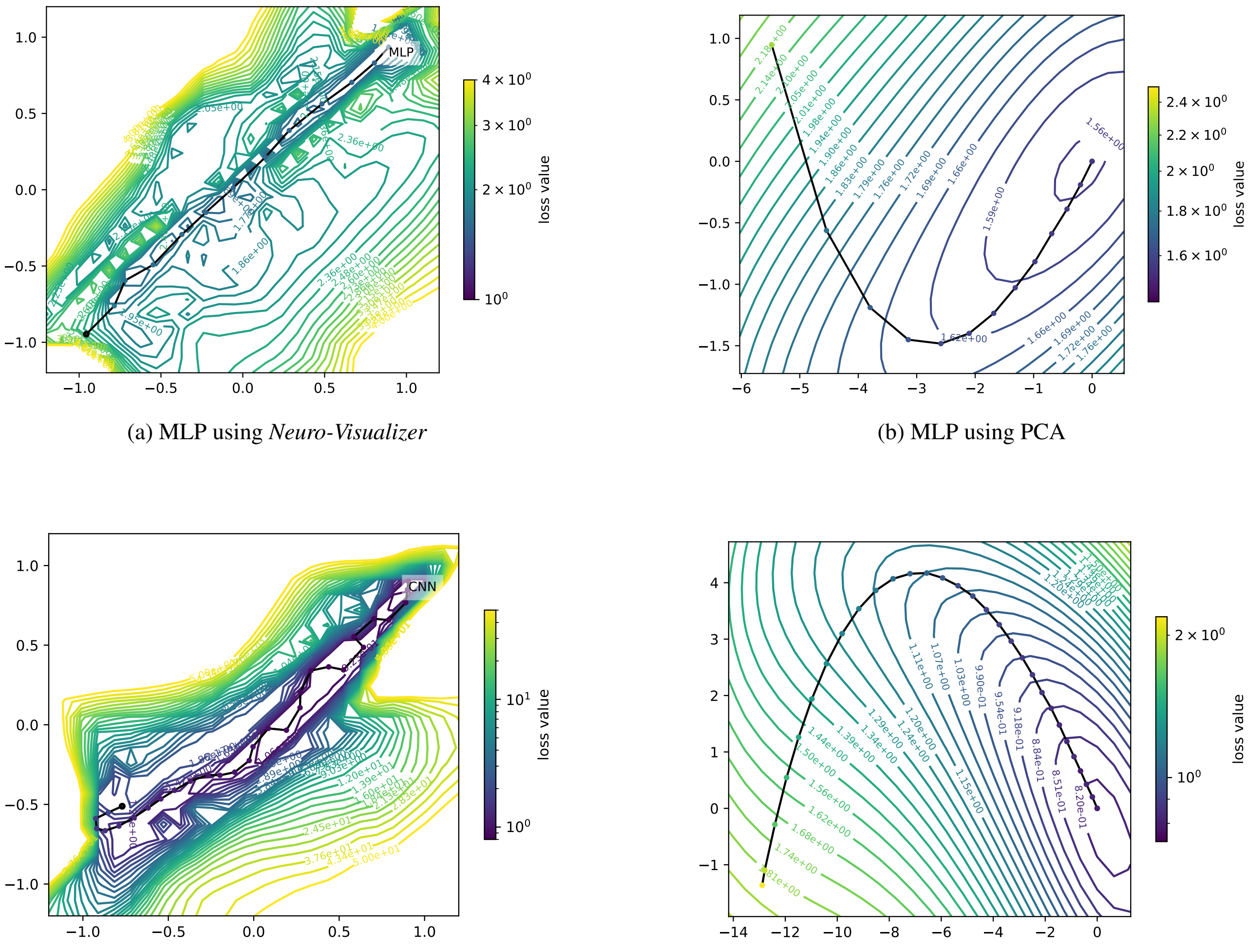
The methodology of *CoPhy*-PGNN involves adaptively tuning the coefficients of these loss functions by annealing λ_S so as to steer the model towards the correct energy level in the initial stages of training. Conversely, λ_C is cold started, allowing the physical constraints to be gradually enforced and ensuring that the solution adheres to the underlying physics.

Using two applications – predicting the ground-state wave function of an Ising chain model in quantum mechanics, and modeling electromagnetic wave propagation in periodically stratified layer stacks – the authors demonstrate the effectiveness and extrapolative power of *CoPhy*-PGNN

D. An Application of *Neuro-Visualizer* in Computer Vision

While the aim of this paper is to specifically study the properties of different deep learning models within the KGML framework using *Neuro-Visualizer*, it is important to note that our proposed method is generally applicable to deep learning models of any type. To substantiate this claim, we visualize the test loss landscapes of two models trained on CIFAR-10 (Krizhevsky et al.), a computer vision task: **(1)** a CNN model of 3 convolutional layers, and **(2)** an MLP model with 1 hidden layer of 16 nodes.

Figure 9 provides a comparative analysis of the test loss landscapes for these models, utilizing both *Neuro-Visualizer* and PCA. With *Neuro-Visualizer*, as shown in Figure 9a, the MLP’s loss landscape appears fragmented, exhibiting many sub-optimal local minima that could potentially impede optimization. In contrast, Figure 9c demonstrates that the CNN's landscape features a single valley of minima with notably low test losses. More importantly, while *Neuro-Visualizer* effectively highlights the distinct loss landscapes of the CNN and MLP, PCA fails to capture these nuances, portraying a misleadingly smooth surface around the converged trajectory points for both models, as shown in Figures 9b and 9d.



(a) MLP using *Neuro-Visualizer*

(b) MLP using PCA

(c) CNN using *Neuro-Visualizer*

(d) CNN using PCA

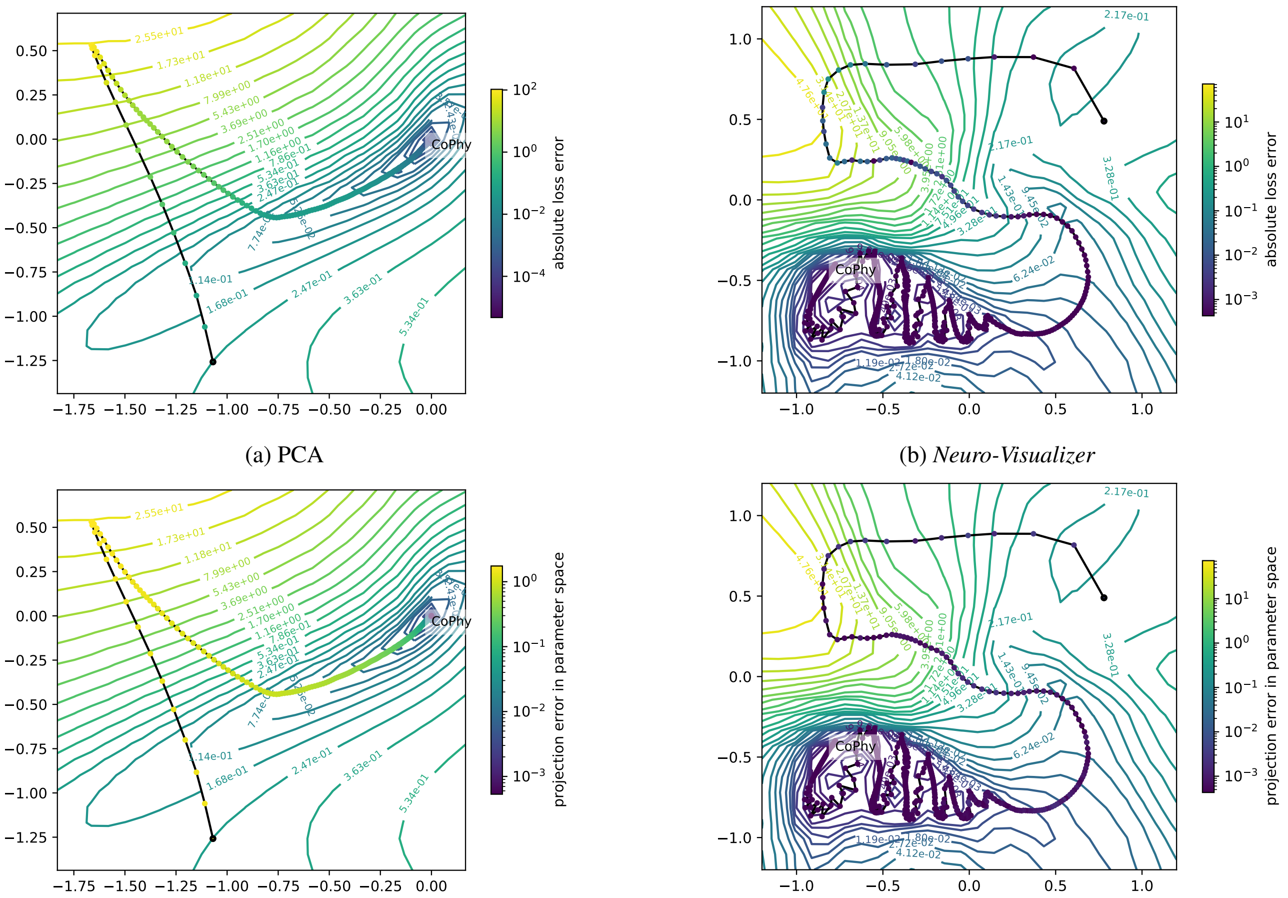
Figure 9. A comparison of *Neuro-Visualizer* and PCA in terms of visualizing the test loss landscapes of CNN and MLP models trained on CIFAR-10. Unlike PCA, *Neuro-Visualizer* is able to show the significant differences in optimization landscapes, highlighting the CNN's more favorable loss landscape compared to the MLP.

E. Loss Landscapes Error Plots For *CoPhy*-PGNN

Following up on Figure 1, and to visually illustrate the results of Table 1, Figures 10 and 11 show the absolute loss error (i.e., the error between model loss and the loss at its projection on the learned manifold) in the first row, and the distance between the models and the manifold in the parameter space in the second row. In both cases, the "hot" model colors of baseline methods, compared to *Neuro-Visualizer*, indicate that the resulting manifold or plane is not a good fit for the trajectory.

F. Solving Partial Differential Equations with Physics-Informed Neural Networks (PINNs)

Solving partial differential equations (PDEs) is a fundamental problem in many areas of science and engineering. Traditional numerical methods, such as finite element and finite difference methods, require a discretization of the domain and the PDEs, which can lead to high-dimensional and computationally expensive systems. Physics-informed neural networks



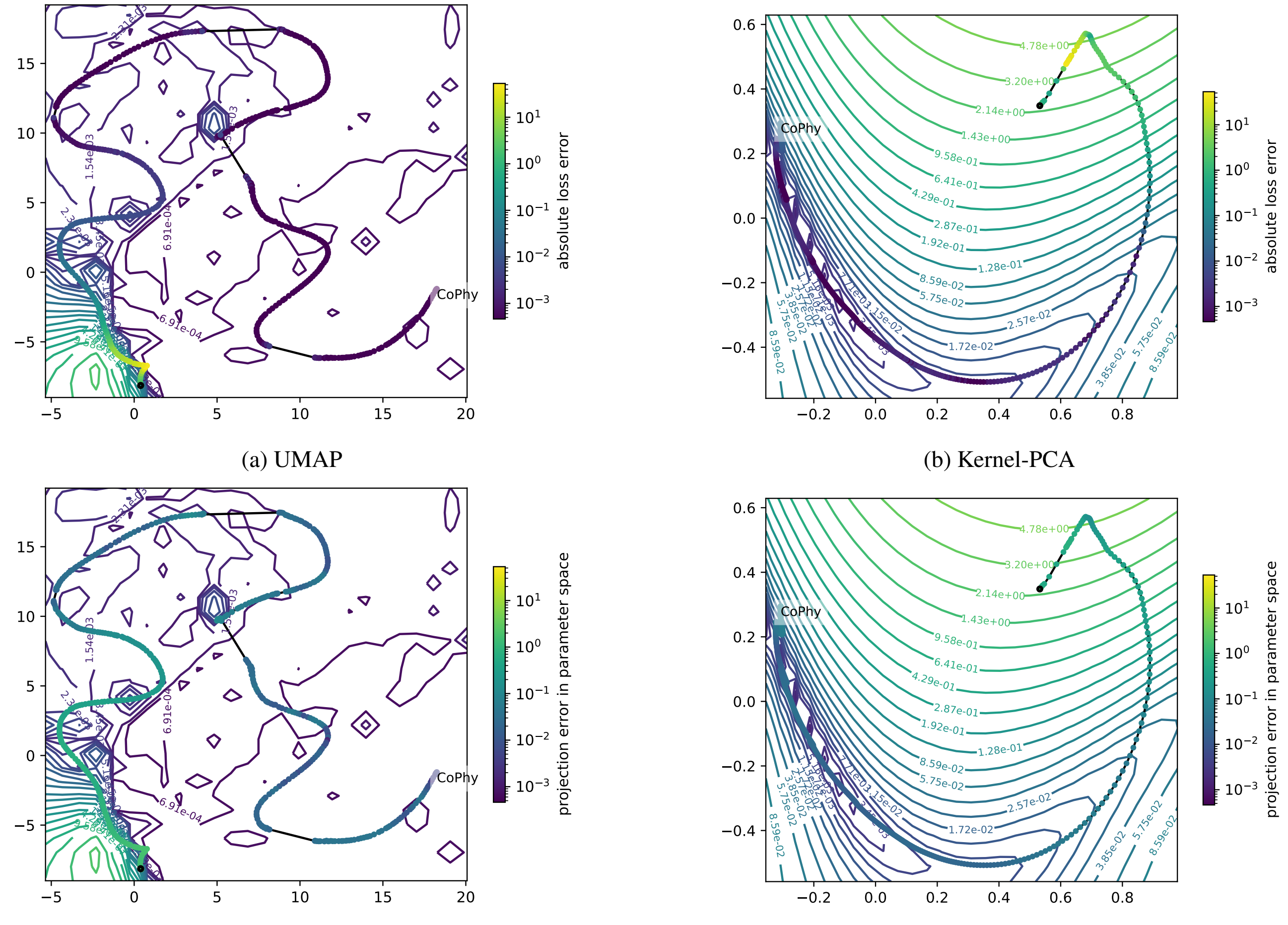
(a) PCA

(b) *Neuro-Visualizer*

(c) PCA

(d) *Neuro-Visualizer*

Figure 10. A comparison of PCA and *Neuro-Visualizer* in terms of the error in *CoPhy*-PGNN's total physics loss values (top two sub-figures) as well as projection errors (bottom two sub-figures). Note that the contour colors still refer to the loss values, similar to Figure 1. *Neuro-Visualizer* has lower error levels than PCA.



(a) UMAP

(b) Kernel-PCA

(c) UMAP

(d) Kernel-PCA

Figure 11. A comparison of UMAP and Kernel-PCA in terms of the error in *CoPhy*-PGNN's total physics loss errors (top row) and projection errors (bottom row). Compared to Figure 10, neither UMAP nor Kernel-PCA look favorable against *Neuro-Visualizer*. Note that the contour colors still refer to the loss values, similar to Figure 1.

(PINNs) (Raissi et al., 2017b; 2019; 2017a; Gao et al., 2021) have emerged as an alternative approach to solving PDEs, leveraging the representational power of neural networks and the structure of the underlying physics to learn a solution without discretization.

PINNs have shown great promise in solving various types of PDEs, including elliptic, parabolic, and hyperbolic problems. The main idea behind PINNs is to parameterize the solution of a PDE with a neural network and enforce the governing equations as constraints in the training process. This is achieved by incorporating the PDEs as a loss function that is minimized during training, along with a data-driven loss term that incorporates observed data.

PINNs' optimization objective comes in a number of different forms. Generally, however, there are three main losses that are present. The residual loss, L_r, is the most important term and ensures that the neural network satisfies the PDE at every point in the domain. It is calculated as the mean squared difference between the residual of the PDE and the output of the neural network at each point. L_{ic} ensures that the neural network satisfies the PDE at the initial condition. It is calculated as the mean squared difference between the output of the neural network at the initial time and the true initial condition. L_{bc} ensures that the neural network satisfies the PDE at the boundary conditions. It is calculated as the mean squared difference between the output of the neural network at each boundary point and the true boundary condition. Together, these three loss terms provide a comprehensive approach to training PINNs to solve PDEs. By balancing these terms, the neural network can learn the underlying physics of the problem and provide accurate solutions.

PINNs have been shown to be effective in solving a wide range of partial differential equations (PDEs). For example, in fluid mechanics, PINNs have been used to solve problems related to incompressible Navier-Stokes equations (Jin et al., 2021). Similarly, PINNs have also been used to solve the Schrödinger equation in quantum mechanics (Li & Li, 2022).

This paper has targeted a certain type of PDEs called the "convection equation" to illustrate the usefulness and superiority of the proposed model. The convection problem is a type of partial differential equation that arises in fluid mechanics and heat transfer. It models the transport of a quantity (such as mass, energy, or momentum) by a moving fluid, which can induce a net flow in the direction of the transport. The convection equation is:

$$f = u_t - \beta u_x \tag{7}$$

where the parameter β is the convection coefficient which represents the tendency of the substance to move with the fluid flow.

While PINNs have been useful at solving PDEs, the application of PINNs is not without challenges (Wang et al., 2022; 2021). In terms of the convection problem, one major challenge is the presence of the convection term β in the governing equations, which is highly nonlinear and can result in optimization difficulties, especially for higher values. Another challenge is the presence of multiple scales in convection problems, which can lead to numerical instability and slow convergence.

G. The Pathological Effect of Increasing Regularization in PINNs

Similar to Figure 7 where the pathological effect of increasing β is displayed, Figure 12 shows the effect of increasing c_r. Clearly, increasing the regularization factor also leads to an increase in loss landscape complexity.

H. A PCA Loss Landscape Analysis of Loss Balancing Methods

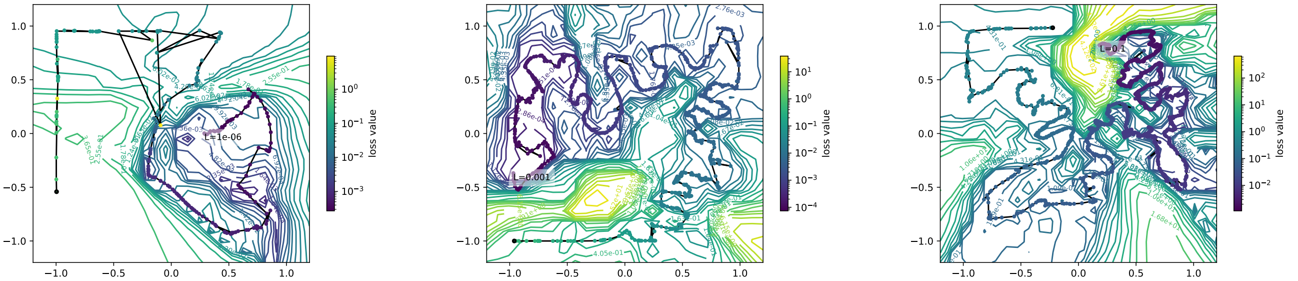
Figure 13 shows the PCA loss landscape for the same models in the loss balancing experiment outlined in Section 4.2.4 and visualized in Figure 8. As can be seen, PCA fails at delineating whether most models converge at all, converge to the same minima, converge to several minima in the same basin, or different basins altogether.

I. Description of Different Loss Balancing Algorithms

Table 3 expands on the different loss balancing algorithms that were used in Section 4.2.4 and shown in Figure 8.

Method	**Abbreviation**	**Brief description**
Equal Weights	*EW*	All loss terms have a coefficient of 1.0.
Constant Weights	*CW*	Different constants are used to balance the loss terms. In the context of this section, the following values where used based on some hyper-parameter tuning: $c_r = 1.0, c_{ic} = c_{bc} = 100.0$.
Dynamic Weight Averaging	*DWA*	The weight of each task's loss is adjusted based on the relative improvement of that task's performance compared to the performance of the other tasks.
Learning Rate Annealing (Wang et al., 2021)	*LR_{annealing}*	Gradient statistics are utilized during model training to balance the interplay between the losses.
Gradient Normalization (Chen et al., 2018)	*GradNorm*	Normalizes the gradients across tasks so that they have similar scales, encouraging the model to focus on the tasks with the most informative gradients.
Random Loss Weighting (Lin et al., 2022)	*RLW*	The weight of each task is randomly updated each epoch.

Table 3. A list of the loss balancing algorithms considered in Section 4.2.4. More details on each algorithm can be found in (Liang & Zhang, 2020).



(a) $L = 10^{-6}$ (b) $L = 10^{-3}$ (c) $L = 10^{-1}$

Figure 12. A comparison of convection-PDE-solving PINNs with different degrees of soft regularization. *Neuro-Visualizer* verifies the authors' claim (Krishnapriyan et al., 2021) that increasing c_r leads to an L_{total}'s landscape that is more non-convex and difficult to optimize.

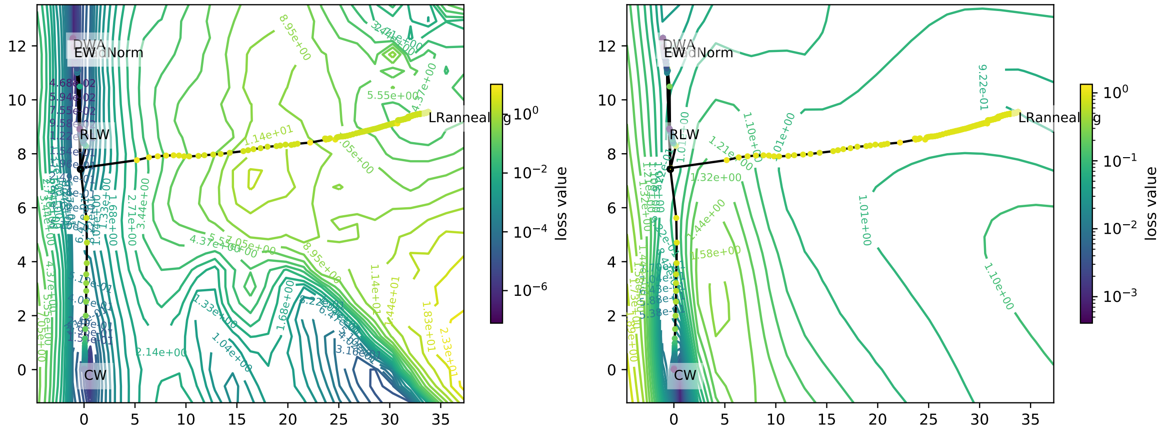


Figure 13. PCA's loss landscapes of multiple PINN models with different loss balancing methods. Clearly, it is hard to make useful or accurate inferences from this plot compared to the insights found through Figure 8.