Understanding The Effects of Incorporating Scientific Knowledge on Neural Network Outputs and Loss Landscapes

Mohannad Elhamod

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Computer Science and Applications

Anuj Karpatne, Chair

Jia-Bin Huang

Christopher North

Narendran Ramakrishnan

Chandan Reddy

April 26, 2023

Blacksburg, Virginia

Keywords: Knowledge-Guided Machine Learning, Machine Learning visualization, Loss landscape visualization

Understanding The Effects of Incorporating Scientific Knowledge on Neural Network Outputs and Loss Landscapes

Mohannad Elhamod

(ABSTRACT)

While machine learning (ML) methods have achieved considerable success on several mainstream problems in vision and language modeling, they are still challenged by their lack of interpretable decision-making that is consistent with scientific knowledge, limiting their applicability for scientific discovery applications. Recently, a new field of machine learning that infuses domain knowledge into data-driven ML approaches, termed Knowledge-Guided Machine Learning (*KGML*), has gained traction to address the challenges of traditional ML. Nonetheless, the inner workings of *KGML* models and algorithms are still not fully understood, and a better comprehension of its advantages and pitfalls over a suite of scientific applications is yet to be realized. In this thesis, I first tackle the task of understanding the role *KGML* plays at shaping the outputs of a neural network, including its latent space, and how such influence could be harnessed to achieve desirable properties, including robustness, generalizability beyond training data, and capturing knowledge priors that are of importance to experts. Second, I use and further develop loss landscape visualization tools to better understand ML model optimization at the network parameter level. Such an understanding has proven to be effective at evaluating and diagnosing different model architectures and loss functions in the field of *KGML*, with potential applications to a broad class of ML problems.

Understanding The Effects of Incorporating Scientific Knowledge on Neural Network Outputs and Loss Landscapes

Mohannad Elhamod

(GENERAL AUDIENCE ABSTRACT)

My research aims to address some of the major shortcomings of machine learning, namely its opaque decision-making process and the inadequate understanding of its inner workings when applied in scientific problems. In this thesis, I address some of these shortcomings by investigating the effect of supplementing the traditionally data-centric method with human knowledge. This includes developing visualization tools that make understanding such practice and further advancing it easier. Conducting this research is critical to achieving wider adoption of machine learning in scientific fields as it builds up the community's confidence not only in the accuracy of the framework's results, but also in its ability to provide satisfactory rationale.

Dedication

This dissertation is dedicated to my younger self, for believing in the pursuit of his dreams, for constantly reminding himself not to let his life slip through the cracks of others' choices, and for embracing the journey on the way to his goal.

Acknowledgments

I would like to express my deepest gratitude to my advisor, Prof. Anuj Karpatne, for leading me through a successful PhD journey with unwavering support, guidance, and mentorship that have been invaluable to my development as a scholar. I would also like to extend my thanks to my committee members, Prof. Jia-Bin Huang, Prof. Chris North, Prof. Naren Ramakrishnan, and Prof. Chandan Reddy, for their insightful feedback and invaluable expertise that helped shape this dissertation into its final form.

I am also grateful to my lab mates, especially Dr. Jie Bu, Arka Daw, and Md Abdullah Al Maruf, for all the interesting conversations, brilliant ideas, and fun times that helped me throughout my research and made the journey memorable and enjoyable.

I would like to acknowledge the constant support of The Department of Computer Science at Virginia Tech and Prof. Cliff A. Shaffer for their constant support of my endeavors and for giving me the opportunity to shine. I would also like to acknowledge the Graduate School at Virginia Tech, Dean Karen P. DePauw, and Dean Aimée M. Surprenant for the programs they have created and supported, particularly the Global Perspective Program (GPP) and Virginia Tech's Academy for Teaching Excellence (VTGrATE), and for bringing out and shaping my authentic pedagogical persona.

I would like to express my heartfelt gratitude to my family - my mother Rajaa Wahbeh, my father Ahmad Alhamod, my grandfather M. Said Wahbeh, and my late step-father Riad Alroumani - for believing in me and for supporting my academic journey since childhood, envisioning a future where I make them, and myself, proud. Finally, I want to thank my dear wife, Jonilda Bahja, for pushing me to be my best even at times when I thought less of myself.

Contents

List of Figures

List of Tables

List of Abbreviations

AE Auto-encoder

CoPhy-PGNN Competing Physics PGNN

CV Computer Vision

GAN Generative Adversarial Networks

HGNN Hierarchy-Guided Neural Networks

KGML Knowledge-Guided Machine Learning

ML Machine Learning

MLP Multi-Layer Perceptron

MSE Mean-Squared Error

PG Physics Guided

PGNN Physics-Guided Neural Networks

PhyloNN Phylogeny-Guided Neural Networks

PINN Phyiscs-Informed Neural Networks

VQ-GAN Vector-Quantized Generative Adversarial Networks

Chapter 1

Introduction

In recent years, Machine Learning (ML) has pervaded all aspects of life, and helped automate many laborious tasks in domains that traditionally required specialized human expertise. One prominent application is the use of deep learning algorithms to enhance medical diagnosis. Estava at al. [40] demonstrated that a convolutional neural network was able to identify skin cancer with a level of competence comparable to that of dermatologists, effectively revolutionizing the early detection of the disease. In the financial sector, machine learning has been employed to detect fraudulent transactions, thereby enhancing security measures. For instance, Carcillo et al. [16] proposed a hybrid model combining deep learning and traditional machine learning algorithms to detect credit card fraud with a high degree of accuracy. Additionally, machine learning has also found its way into the realm of environmental conservation. Collins et al. [25] showcased how machine learning algorithms can be utilized to monitor endangered species, such as the North Atlantic right whale, by analyzing acoustic data to identify their presence and protect their habitats. These real-world applications attest to the transformative potential of machine learning across various domains.

This success in deep learning methods has been primarily fueled by the availability of massive volumes (in the order of millions) of labeled training data (e.g., in Imagenet [30]). However, existing deep learning models do not generally learn features that map to human knowledge. Since most off-the-shelf deep learning models are *black-box* (i.e., it is hard to understand

their inner workings), they lack the ability to provide useful explanations to end-users in a manner consistent with existing scientific knowledge. In many applications, including those in scientific domains, understanding the decision-making process can be as important as the decision itself so that the method is trustworthy and reliable. Moreover, black-box models don't necessarily comply with existing domain understanding, and hence are susceptible to showing poor performance when presented with out-of-distribution data (i.e., lack of generalizability) [49] even when they produce accurate predictions on in-distribution data.

Also, since current standards of black-box ML solely rely on data-driven supervision, they are not fully equipped to discover novel knowledge in the form of interpretable features or attributes, despite being successful on benchmark problems. This is exacerbated by the fact that, unlike industrial and commercial applications, scientific domains still struggle with achieving good generalization performance of deep learning models due to the paucity of labeled data [31, 157]. For example, in the biological problems of species classification and trait detection, generating training labels (in the form of annotations) is labor-intensive and requires subjective human expertise. However, unlabeled data is easier to obtain (e.g., generating partial differential equations without solving them). This paucity of representative training data is one of the biggest impediments in the standardization of deep learning in science.

1.1 Knowledge-Guided Machine Learning (*KGML*)

To remedy the shortcomings of black-box ML mentioned above, namely its lack of explainability and generalizability, the emerging field of *Knowledge-Guided Machine Learning* (*KGML*) [71] attempts to bridge the gap between data-only and conventional science-only approaches, by informing the ML frameworks with scientific knowledge available in different

forms in various disciplines (e.g., principles of energy and mass conservation in physics, or tree-based structured knowledge in biology). For example, a model that predicts the location of two particles after collision could also benefit from optimizing its parameters to uphold the law of conservation of energy. Such constraints could be enforced by imposing additional, usually label-free, regularization that is informed by domain knowledge.

KGML seeks a radical departure from data-only and science-only methods by using scientific knowledge and data at an equal footing for learning generalizable models even in the paucity of representative training labels. For example, in addition to labeled data, Daw et. al. [28] enforce the depth-density relationship on unlabeled data to improve the performance of lake temperature modeling. Alternatively, instead of using unlabeled data, other researchers modify black-box architecture of neural networks such that the model is informed by specific domain knowledge. For example, Carranza-Rojas et. al. [17] improve image classification performance by adding an unsharp masking layer to vanilla ConvNets to inject prior image-enhancement knowledge. These are just examples of the growing interest in *KGML*, which is reflected in a series of planning reports and thought articles on this topic [44, 143]

1.2 Limitations and Challenges in *KGML*

While initial explorations in the nascent field of *KGML* have shown that "infusing" machine learning models with domain knowledge leads to improved performance, a complete understanding of this framework is still missing. More specifically, there has been little research concerning the effects of integrating domain knowledge into model architecture and optimization beyond simply measuring the improvement in predictive accuracy.

While a more detailed and mathematical understanding of machine learning modeling in general is still desired, the research community has applied several practical and emperical

techniques to study black-box machine learning models in more depth. For example, in terms of understanding model prediction and information extraction, researchers have used tools such as saliency maps [138] and Class Activation Maps (CAMs) [180] for model interpretability, as well as other tools such as t-SNE [65]. The use of these tools is wide-spread in the community, especially in computer vision. On the other hand, there has been little work in *KGML* that attempts to understand the advantages of knowledge guidance in the model output space. For example, an important question to answer is: *"Does using the* KGML *framework lead to more robust outputs that are resilient to noise and adversarial examples?"*. This and other questions are important to understand the full impact of using the framework.

In addition to studying the output space, very little work has been conducted to understand the influence of knowledge guidance on model optimization. For black-box models, in addition to theoretical machine learning research, visualization tools, such as loss landscape visualizations [91] and disconnectivity graphs [161] have gained attention in recent years, and have been used to investigate the effects of different hyper-parameters, such as batch size and model architecture, on model optimization. This line of research is mostly absent in the *KGML* framework research. As such, questions such as *"How does knowledge guidance influence the model's optimization through the lens of loss landscapes?"* have not been fully answered. Bridging this research gap is of great importance as it helps researchers and practitioners better understand the best practices for designing *KGML* solutions.

1.3 Thesis Goals

In this thesis, I aim to understand the impact of knowledge-guidance in machine learning through the following two lenses:

1. Achieving *a better understanding of the effect* KGML *plays in the model output space, including the latent space.* More specifically, I aim to investigate and understand the advantages of using *KGML* over black-box models through several perspectives. One such perspective is robustness to noise and adversarial occlusion, and generalizability under paucity of samples. Another perspective is achieving meaningfulness and consistency with scientific knowledge at the model outputs. This research goal is explored through an application in biology.

2. *Applying and developing the loss landscape visualization tool within* KGML such that the advantages and pitfalls of its models are better understood. The aim here is to demonstrate the usefulness of novel visualization tools and their feasibility for investigating model optimization when incorporating scientific knowledge as different loss functions. Ultimately, this research would promote a wider adoption of loss landscape visualization for investigating, diagnosing, and developing machine learning models. This research goal is explored through applications in physics.

A visual breakdown of the goals of this thesis can also be found in Fig. 1.1.

The following list outlines the organization of this thesis:

- Chapter 2 gives a comprehensive overview of the concepts discussed across this document.
- Chapter 3 gives an overall description of the major applications within which the experiments are conducted.
- Chapter 4 focuses on demonstrating and understanding the benefits of using *KGML* at improving the characteristics of model output, constraining it with desired properties such that it is grounded to established knowledge.

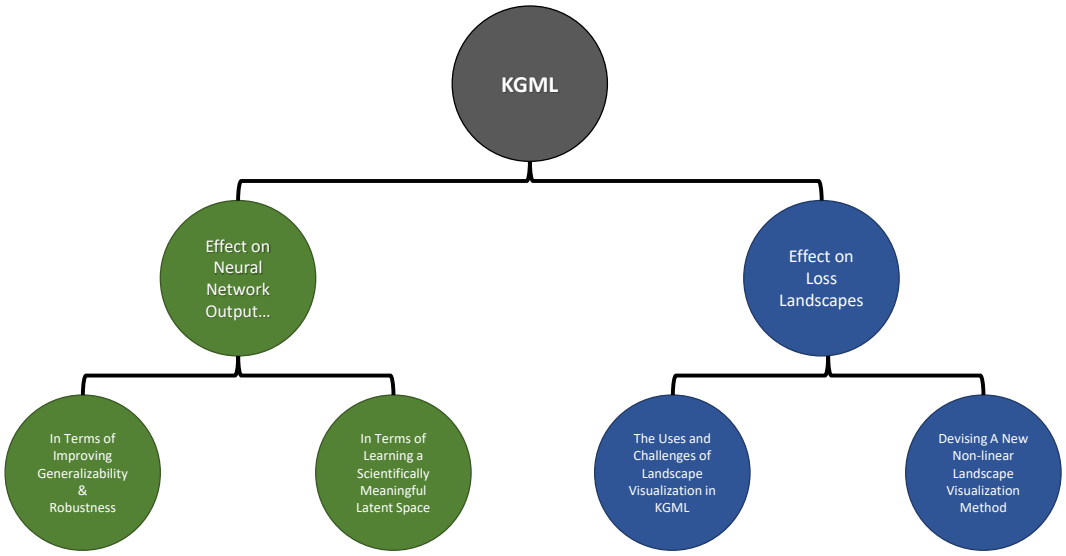


Figure 1.1: A visual breakdown of the goals of this thesis.

- Chapter 5 focuses on applying and developing novel loss landscape visualization tools to better understand *KGML* models and their different properties, including challenges in the optimization process.

- Chapter 6 gives a brief summary of the work in this thesis.

Chapter 2

Review of Literature

In this literature review, I survey some of the previous work done in areas relevant to my thesis. These areas are Knowledge-Guided Machine Learning (*KGML*), loss landscape visualizations, and some background on trait visualization in ML and biology

2.1 Knowledge-Guided Machine Learning (*KGML*)

In spite of the impressive progress achieved by ML in diverse scientific disciplines [4, 51], most research has only focused on purely labeled-data-driven approaches (i.e., black-box models). However, as was explained in Chapter 1, the ML community has recognized the advantage of imposing additional, often label-free, regularization techniques inspired from domain knowledge and its rich supervision to achieve better generalization and scientific consistency. This has been applied to and shown to be true in a range of application domains including chemistry, physics, biology, climate science, security, healthcare, transportation, and many other areas [71, 119, 169]. This emerging field comes by many names, including physics-guided machine learning (PGML), science-guided machine learning (SGML), and knowledge-guided machine learning *KGML*[28, 29, 71, 72, 115, 122, 163].

For example, 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* [71, 144]. An

early work in this category includes that by Stewart et al. [144] on the use of domain constraints (e.g., the kinematic equations of motion) as PG loss to supervise neural networks. 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. Similarly, physics-informed neural networks (PINNs) and its variants [120, 121, 122] have been recently developed to solve PDEs by solely minimizing PG loss functions for simple canonical problems such as Burger's equation. We refer to the class of neural networks that are trained using PG loss functions as physics-guided neural networks (*PGNN*s). *KGML* generally, and *PGNN*s specifically, have found successful applications in several disciplines including fluid dynamics [163, 170], climate science [29], and lake modeling [28, 68, 72].

Another form that *KGML* can take is architectural modifications. One drawback of adding PG losses is the need to carefully craft corrections in the objective function to avoid the learning of trivial solutions and ensure convergence to the desired solution. Architectural modifications, on the other hand, do away with such tricky loss function engineering and alter the internal components of the neural network such that the desired constraint is intrinsically imposed. For example, Ling et. al. [96] improve Navier-Stokes turbulence modeling by imposing rotational invariance as a physics constraint onto the last hidden layer of multi-layer perceptron. Similarly, Carranza-Rojas et. al. [17] infuse prior image-enhancement knowledge in the form of high frequencies through an additional layer in a ConvNet to improve image classification performance.

While the two aforementioned approaches are the most common, other forms of *KGML* exist, such as input pre-processing. Input pre-processing comes in many forms, including feature extraction, noise removal, and Fast Fourier Transforms (FFT) [119]. For a detailed survey of research directions in SGML, see Willard et. al. [169].

2.2 Model Understanding Through Landscape Visualization

Landscape visualization is the practice of plotting the value of an ANN's loss function w.r.t its model parameters (i.e., weights and biases). Commonly, such plot is a projection of the high-dimensional loss surface onto a low dimensional hyper-plane (usually 1D [48] or 2D [91]) to facilitate its visualization and understanding by humans.

The study of deep neural network's loss landscape has attracted much attention in recent years. For a while, researchers have tried to find guarantees for its convexity, even if under certain constraints. For example, Kawaguchi et al. [76] have shown that the local minima of deep linear neural networks with a squared loss function criterion are equivalent to a global minimum. In their literature review, Sun et al. [146] also list many beliefs about how the network's architecture and optimization relate to the geometry of the loss landscape, including over-parametrization and layer widths. Some of these beliefs have been corroborated by other studies [173]. However, such kind of work is limited in terms of the selection of architecture and loss function criterion.

In the absence of a generic theoretical understanding of deep neural networks' loss surface, a new trend has developed. Rather than expecting a rigorous proof, recent work has focused on qualitatively assessing high dimensional deep learning models by looking at their loss surface on a lower dimensional grid. Goodfellow et. al. [48] is one of the first papers to rely on this methodology. One of the seminal papers in this direction is by Li et. al. [91]. In their work, the authors call for using a projection on a random plane to visualize and assess the quality of the loss surface around a model. This allows to determine whether the surface is ill-regularized and filled with local minima, or it is mostly flat and easy to optimize.

While using random projections works well for visualizing the surface around a specific model (e.g., after model training converges), it does not give an informative picture if the training process in itself is to be examined. In that case, as suggested by the authors in [91], PCA could be used to extract the 2D projection that best fits the optimization trajectory. Another method that has been suggested is choosing the 2D projection that contains the two eigenvectors with the highest eigenvalues [19].

Though qualitative in nature, the analysis of the loss landscape has proven to be insightful [91] and is becoming a more common practice in the field [43, 102, 111], and an alternative to some quantitative methods such as the Fisher information matrix (FIM) [70] and Hessian analysis [53, 99]. For example, the relationship between a model's generalizability and the topology around it minima has been studied by several researchers [65, 175], and is a topic of debate. On one hand, by using dataset poisoning to find minima that overfit the training data, Huang et. al. [65] conclude that flat minima correlate with better generalizability. Sypherd et al [147], similarly, show that the relationship between generalizability and sharpness is inversely proportional, albeit for linear regression of a certain class of loss functions. On the other hand, Yang et. al. [175] argue that it is the global, not local, structure of the landscape that acts as an indicator of good generalization; they find that best generalization occurs at a locally flat and globally well-connected loss landscape. Prabhu at al. [116], Guiroy et al. [53], and Xu et al. [174] also cast doubt on the relationship between sharpness and generalization.

Other researchers have done more rigorous analysis of the geometry of the loss landscape to better understand its impact on model optimization. This has been more pronounced in the work that studies the hierarchical structure of the loss landscape, and how it is composed of minima that can be connected and grouped into separated basins or valleys [65, 99]. Furthermore, some work by Xing et al. [173] shows that stochastic gradient descent

(SGD) is able to jump over the barriers between these basins. Other investigations that involve studying the loss landscape include assessing the generalization of Model-Agnostic Multi-task Learning (MAML) [53], how noise smoothens sharp minima and lead to better generalization [168], and how skip connections help model optimization by removing bad valleys that contain sub-optimal minima [110].

Loss landscape visualization has also been used to investigate and answer a variety of practical questions regarding the training process. For example, Keskar et al. [77] study how larger batches lead to flatter minima and saddle points where generalization is worse. Another interesting study shows that gradient descent is quite good at avoiding bad minima [146]. Other deep neural network properties that have been studied using landscape visualizations include, but are not limited to, the different phases of the training process [175] and how the high-dimensionality of model parameters biases optimizers towards flat minima [65].

While the most commonly used method for loss landscape visualization is taking a linear 2D slice of the high dimensional space, others have used some non-linear methods such as UMAP and t-SNE [65], and other methods that are inspired by them, such as SHEAP [134] and PHATE [61]. However, these methods are more suitable for visualizing the relationship between models in a non-linear 2D manifold as opposed to visualizing the landscape around those points on the 2D grid.

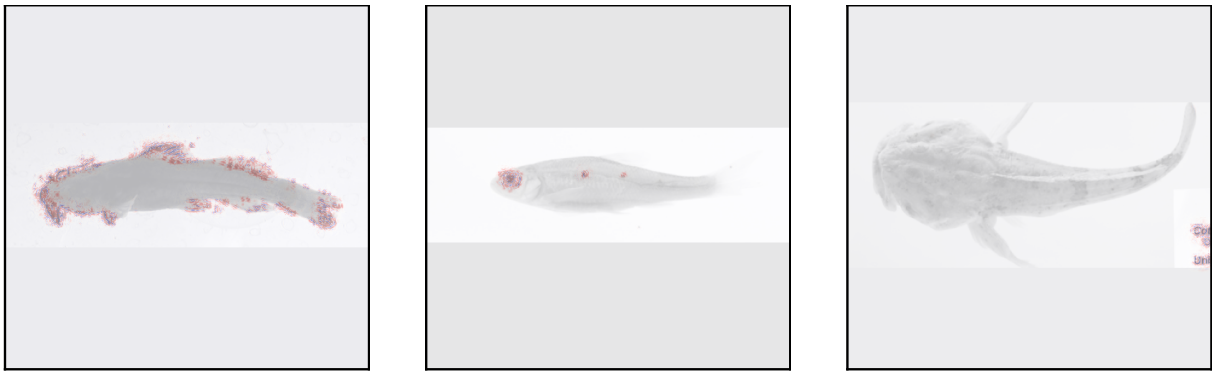
While other more formal and rigorous methods for describing the high-dimensional loss surface have been adopted, such as analysing the spectrum of the Hessian at a certain model [176], I focus in this thesis on improving the loss landscape visualization and making it more powerful and insightful.

2.3 Interpretability and Feature Visualization

In computer vision (CV), there is a plethora of visualization and explainability tools that have been developed, which are commonly used for understanding how deep vision models reason about input images. This section is an overview of such tools.

One visualization tool that is most commonly used is saliency maps (a.k.a. sensitivity maps) [138]. Saliency maps are heatmaps depicting the gradients of a neural network's output w.r.t its input. In other words, a saliency map shows how strongly do changes in pixel values affect the output class probability, thus highlighting the parts of the image that are most decisive for the classification problem. Fig. 2.1 shows some examples of saliency maps obtained for the species classification application (described later in Section 3.2). As can be seen, saliency maps can be an effective tool that detects the image attributes to which the model is most sensitive (e.g., barbels and fins in Fig. 2.1a and the eye in Fig. 2.1b). More importantly, saliency maps are also a good debugging tool that provides insight as to whether the model is "cheating" or looking at irrelevant features. An example of such a case is presented in Fig. 2.1c, where the model is incorrectly picking up on the label paper. As such, saliency maps can be used for "interpreting" CV models.

Another family of widely used visualization tools is Class Activation Maps (CAMs) [180] and its variants. CAMs generally attribute importance based on the weighted activations of the learned convolutional features. These weights can be parameter-based [180], gradient-based (e.g. Grad-CAM) [132], or perturbation-based (e.g., Score-CAM) [162]. Like saliency maps, CAMs highlight the salient parts of an image. However. while saliency maps tend to attribute importance at the pixel level, making them more detailed but also prone to noise, CAMs highlight broader regions due to the need for upsampling of low-resolution feature maps to match input image resolution. Thus, CAMs tend to be both less precise and less



(a) Barbels and fins (b) Eye (c) Label paper

Figure 2.1: Saliency maps of different fish images. Colored pixels denote image regions with high saliency scores, indicating their high importance for the fish classification task as perceived by the model. Figure adopted from [38].

noisy [162].

While most interpretability tools in CV suffer from approximating local linearities and are specific to a single input at a time, some researchers have focused on more global explainability methods. For example, Wu et. al. [171] approached the task through the lens of *concept importance* using feature occlusion. Such research direction is important for reaching a more holistic understanding and explainability of deep CV models.

In addition to interpreting model output using the aforementioned tools, researchers and practitioners are also interested in extracting and understanding the features the model internally learns and based on which it makes its prediction. For example in a ConvNet, the learned convolutional kernels (or filters) act as feature extractors. Hypothetically, the extracted features correspond to meaningful parts of the image that help explain the model's decision to humans. However, that is not always the case. In fact, it is quite possible to learn spurious features that lead to catastrophic failures, such as in the adversarial examples [49] case.

Some researchers argue that the correct approach to learning these features is to use models

that are intrinsically explainable [129]. An example of such models is Chen et. al.'s Prototypical Part Network (ProtoPNet) [20], where the model dissects the image and finds its prototypical parts that correspond to patches from training images, then uses those parts to perform the classification task. By using actual patches from real images, ProtoPNet lays itself as an intrinsically interpretable model. Similarly, Zhang et. al. [178] propose an interpretable ConvNet that pushes each filter to learn a specific object part during model training by masking the input image. Other researchers [145, 179] use attention-based methods. Hendricks et. al. [59] even go further and propose a model that learns to explain an image through language rather than visual traits, and apply their model to explain the classification of a bird by outputing an English sentence that describes it.

2.4 Image generation and Synthesis

There exists a large body of work in the field of deep learning that studies the use of neural networks for generating images. For example, auto-encoders generally have been used to learn a low-dimensional embedding space that is continuous and maps to the high dimensional image space. In the case of Variational Autoencoders (VAEs) [81] the learned embedding space is parametrized as a Gaussian distribution, making the the space more structured and disentangled, and better suited for image generation [34]. Another model used for image generation is Generative Adversarial Networks (GANs) [45]. GANs consist of two neural networks: a generator and a discriminator. The generator produces synthetic data intended to mimic real data, while the discriminator distinguishes between the real and generated data. The two networks are trained together and compete, ultimately making the generator produce realistic data to fool the discriminator. There has been many improvements over the original GAN proposal. Some of the latest GAN research can be found here

[47, 73, 74, 75, 125],

One of the more state-of-the-art methods that have gained great momentum is Transformer networks [36, 57]. The core idea behind the transformer architecture is the self-attention mechanism, which allows the model to attend to different parts of the input feature map while computing the output at each position, and multi-head attention, which allows the model to attend to different parts of the input feature map in parallel.

While conventional VAEs embed images in continuous feature spaces, a recent variant of the VAE termed Vector-Quantized VAE (VQVAE) [114] uses discrete feature spaces quantized using a learned codebook of feature vectors (or codes) and employs a PixelCNN [154] model for sampling in the discrete feature space. The rationale for converting images to discrete representations is to allow for easier and more complex manipulations than continuous features. This work was extended in [39] to produce VQGAN, which is different from VQVAE in two aspects. First, it adds a discriminator to its framework to improve the quality of the generated images. Second, it uses a Transformer model, namely the GPT architecture [118], to generate images from the quantized latent space instead of a PixelCNN. VQGAN is a state-of-the-art method that generates images of better quality efficiently at higher resolutions than other counterparts such as StyleGAN [75] and Vision Transformers [36, 57].

Chapter 3

Background on Scientific Problems

In this chapter, I introduce a comprehensive overview of the applications targeted in the remainder of this thesis. The concepts and experiments in the next chapters shall be understood and discussed within the context of these applications.

3.1 Eigen-decomposition in Physics

The general form of an eigen-decomposition is $\hat{A}\boldsymbol{y} = b\boldsymbol{y}$, where $\hat{A}$ is the input matrix, b is an eigenvalue of the input matrix and $\boldsymbol{y}$ is the corresponding eigen-vector. Here, the goal is to solve an eigen-equation for a specific eigen-solution. In my thesis, I address two different applications in physics. The first is finding the highest eigen-solution for electromagnetic propagation as governed by Maxwell's equations, and the second is finding the lowest eigen-solution for an Ising model as governed by the Schrödinger equation. Solving such 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. For more details on the physics of these two applications, please refer to appendices A and B.

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 (considered as expert ground-truth).

We consider the problem of learning an Artificial Neural Network (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 Taxonomy, Phylogeny, and Image Processing in Biology

Species image processing in biology is an important task that is the foundation of industrial, commercial, ecological, and scientific applications involving the study of species distributions, dynamics, and evolution.

Image classification in real-world biological problems such as species classification is fraught with several challenges that limit the usefulness of state-of-the-art deep learning methods trained on benchmark datasets. First, real-world images of specimens suffer from various data quality issues such as damaged specimens and occlusions of key morphological features [41], which can crucially impact classification performance. Second, real-world datasets for classification are limited in their scale in comparison to benchmark datasets, with limited representative power in terms of number of species [3, 27, 32, 88, 90, 113, 123, 124], or number of images per species [89, 127]. This is especially true for rare species [159]. Third, the hierarchy of features extracted by conventional deep learning frameworks, while useful for prediction, do not conform to known biological hierarchies and hence do not directly translate to advancing scientific knowledge, which is often a more important goal than improving predictive performance for a scientist [71].

In this thesis, within the framework of *KGML*, I investigate the usefulness of domain knowledge, in the form of taxonomy and phylogeny, to improve the performance of ML models in biology. Depicting the branching pattern of taxa, phylogeny represents a hypothesis of evolutionary relationships based on shared similarities derived from common ancestry [60]. It characterizes the evolutionary distances among species and their common ancestors represented as nodes of the tree. In this tree, the length of every edge represents the evolutionary distance between two nodes (measured in time intervals of thousands or millions of years).On average, longer edges accumulate higher levels of genetic or phenotypic trait change than shorter edges, and species that are recently diverged will be more similar both in their genetics and phenotypic traits. From conservation to zoology, phylogenetic relationships are critical for interpreting study results and implications in the biological sciences. However, this hierarchical information has yet to be fully incorporated in that of machine learning and image classification.

In Chapter 4, I focus on two problems in terms of model output space. First, I tackle the problem of classifying the species of a fish specimen given a 2D image. As a case-study, fishes are selected because they are a highly diverse, well-studied, and an ancient group of animals that comprise almost half of all vertebrate species [58] and they play critical roles in Earth ecosystems [107, 158]. Fishes are also the targets of recreation [5], aquaculture and fisheries [98], and conservation [6]. Also, phylogenetic relationships are well-studied [11, 66], and their taxonomic classification is generally aligned with phylogeny. Thus, they serve as an ideal benchmark, and I use them in an extensive number of experiments.

In addition to studying species classification, I also study the discovery of evolutionary traits that are heritable across species on the tree of life (also referred to as the phylogeny). Discovering these traits is of great interest to biologists to understand the evolution of different organisms. This task is particularly challenging to biologists as the measurement

of traits is often a subjective and labor-intensive process, making *trait discovery* a highly label-scarce problem. Also, these traits are influenced by genes and the environment. Thus, not all features extracted by deep learning models will show evolutionary signals, making it crucial to separate evolutionary information from unrelated factors. However, learning such biologically meaningful traits can help in a number of downstream tasks including species image generation and species-to-species image translation.

3.3 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 (PINNs) [42, 120, 121, 122] 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, $L_{residual}$, 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 [69]. Similarly, PINNs have also been used to solve the Schrödinger equation in quantum mechanics [92].

In Chapter 5, I target a certain type of PDEs called the "convection equation" to illustrate the usefulness of loss landscape visualizations in studying model properties. 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:

$$f = u_t - \beta u_x \tag{3.1}$$

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 [165, 166]. 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.

Chapter 4

Investigating The Effect of Knowledge Guidance from The Neural Network Output Lens

In the literature, the main focus has been to showcase how the *KGML* framework particularly shines in terms of accuracy improvement. In this chapter, I focus on understanding how the *KGML* framework influences the model's output space, including the latent space, from other important angles, and how it can yield models with desired properties that adhere to important constraints as set by domain experts. Examples of such properties are model resiliency to corrupted input and biologically-plausible latent space encoding.

In the next sections, I conduct my investigation in the context of biology, which lays itself to domain knowledge encoded as taxonomies and phylogenies. More specifically, I demonstrate my work within the applications of species classification and biological traits discovery. In these applications, I design *KGML* solutions that take advantage of the available domain knowledge and demonstrate the beneficial effects of this framework through careful empirical analysis as well as some visualization and interprtability tools.

For more information about the applications referenced in this chapter, please refer to Section 3.2.

4.1 Motivation: Failures of Black-box Models in the Output Space

Conventional approaches to image understanding in science and engineering use off-the-shelf machine learning (ML) methods such as existing Convolutional Neural Network (ConvNet) architectures. Deep neural networks have found immense success in image classification problems with state-of-the-art ConvNet models (e.g., GoogleNet [148], AlexNet [86], and VGGNet [137]) reaching unprecedented performance on large-scale benchmark datasets such as ImageNet [30] and CIFAR [85].

Deep CV models function by extracting a hierarchy of simpler to more complex forms of abstraction in hidden layers—simpler features at lower depths (e.g., edges and texture) are non-linearly composed to form complex features at higher depths (e.g., eyes and fins). This has motivated several recent architectural innovations in deep learning such as ResNet [56], ResNeXt [172], and DenseNet [64], that have enabled the learning of deep and complex hierarchy of hidden features.

However, the innate hierarchy extracted by neural networks from data is not necessarily tied to known hierarchical relationships in real-world applications. As a result, deep CV models can exhibit catastrophic failures when compared to human vision. One evidence of the discrepancy between the two systems is how easily deep models can be fooled by adversarial attacks [49]. By making subtle image changes that are imperceptible to humans, a deep CV model can be tricked into entirely changing its classification output. The existence of such adversarial examples implies that the image attributes captured by a CV model do not usually align with those of humans. Nguyen et. al. [109] even show that generating such examples can be easily automated using gradient ascent. Another evidence of this gap in feature similarity has been illustrated by Alcorn et. al. [1]. In their work, it is shown that

even state-of-the-art models can be easily tricked by applying simple pose transformations.

Such failures have motivated me to consider how *KGML* could help alleviate the problem by reconciling the relevant human knowledge, as encapsulated in the science, with that of the data-driven features captured by the deep model. In the following sections, I consider the application of species classification and trait extraction described in Section 3.2 as case-studies for incorporating domain knowledge, present in the form of the tree-structured taxonomies, into the model optimization process. Ultimately, I propose two novel deep learning models that successfully marry domain knowledge and machine learning, and that outperform black-box models as demonstrated in Sections 4.2 and 4.3.

4.2 The Effect of Knowledge Guidance on Model Generalizability and Robustness

To reconcile human knowledge with machine learning in the context of species classification (See Section 3.2), I have developed a deep model, named Hierarchy-Guided Neural Network (*HGNN*) [38]. Using taxonomic relationships, *HGNN* aims to incorporate hierarchy to improve the properties of model output. Specifically, *HGNN* incorporates known hierarchy among classes (available as a two-level taxonomy: genus and species) to guide the learning of features at the hidden layers of the neural network. This work builds on a history of multi-label and hierarchical classification techniques using pre-built taxonomies [136, 177]. While using taxonomic information for automated species classification is not novel [87], to my knowledge, the only body of work that has researched it before in the context of deep learning is by dos Santos et. al. [35]. However, *HGNN* is distinguished in two ways. First, while they have used the family and order information, I use the genus information. Based

on my experiments, I find incorporating the genus yields more information gain as it involves more discriminative features than the order and family. Second, their model only uses the taxonomic information in the last fully-connected layer, while *HGNN*'s philosophy is to use it at a convolutional level of the network as that allows for capturing localized visual features that are taxonomically plausible.

Fig. 4.1 shows the architecture of *HGNN*, which consists of two sub-modules (top and bottom rows) of ResNet models operating in parallel. A ResNet architecture is adopted as it is currently among the most widely-used and best-performing ConvNet models for benchmark computer vision problems, including fish identification [33, 67, 78, 158]. However, *HGNN* is generic and can work with any deep learning architecture. In Figure 4.1, the top row ResNet predicts the species class $\mathbf{s}$ of the input fish image $\mathbf{x}$, while the bottom row predicts the genus class $\mathbf{g}$. These ResNets learn a hierarchy of features (from simple to complex) at their hidden layers useful for the tasks of species and genus classification, respectively. While both these sub-modules can be viewed as learning separate features, we know that the genus features learned in the bottom ResNet ($\mathbf{y_g}$) represents features at a higher level of abstraction that are directly useful for the task of species classification. Building upon this knowledge in the proposed *HGNN* framework, the genus features learned at an intermediate depth H_g of the genus sub-module are harnessed and aggregate them with the species features $\mathbf{y_s}$ learned at the H_s depth of the species sub-modules. The combination of both species and genus features is then used for the task of species prediction.

This architecture is motivated by the assumption that hierarchical taxonomy of genus and species classes captures a notion of derived similarity in terms of the discriminatory anatomical features of every class. This is true, as illustrated in Figure 4.2, in the context of fish classification because species classes that belong to the same genus are more closely related phylogenetically than species classified in different genera. In the case of the species and

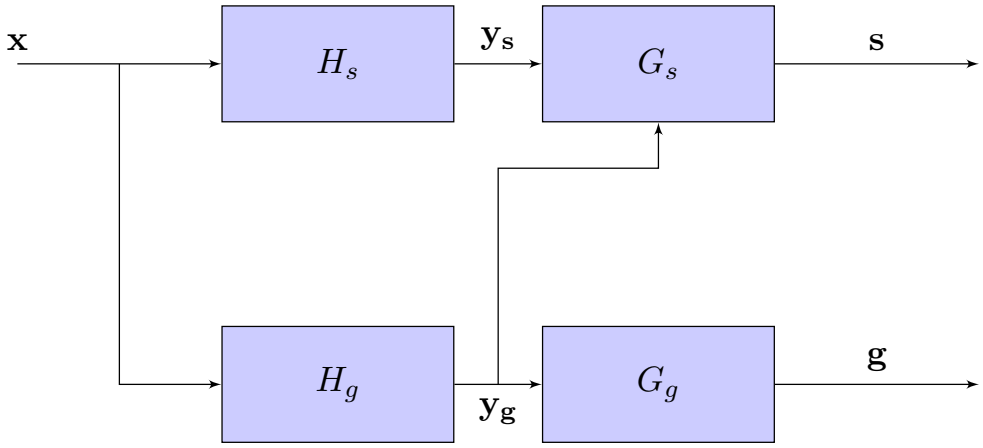


Figure 4.1: Schematic diagram of *HGNN*.

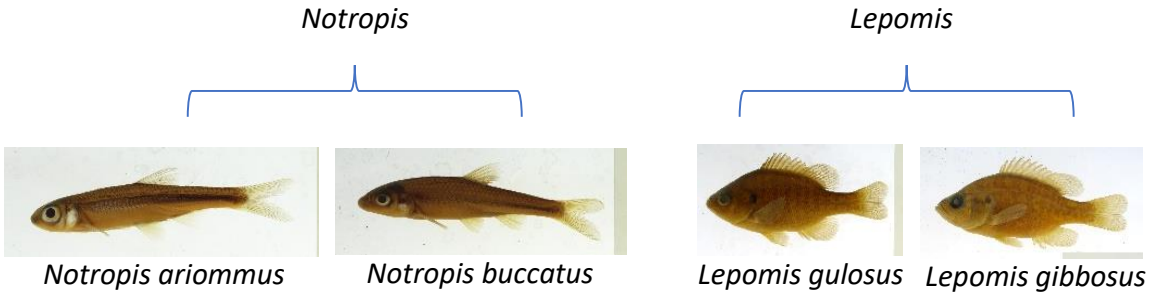


Figure 4.2: An example of the alignment between taxonomic and anatomic similarities in fishes. Figure is from [38]

genera analyzed here, with only a few exceptions, this is the case. Appendix E provides more information in that regard. As a result, species that map to the same genus $\mathbf{g}$ should generally share similar features at the internal representation of the neural network (e.g., filters learned at the convolutional layers). This observation seems to align with some earlier work [35, 100]. Second, while the mapping from $\mathbf{s}$ to $\mathbf{g}$ is one-to-one, the inverse mapping from $\mathbf{g}$ to $\mathbf{s}$ is not unique. Hence, along with the shared features learned for every $\mathbf{g}$, we also need to learn unique features for every $\mathbf{s}$ to differentiate between species belonging to the same genus.

Thus, as shown in the architecture proposed in Fig. 4.1, the functional block of layers H_g

extract hidden features $\mathbf{y_g}$ that are useful for predicting $\mathbf{g}$ (through the functional block G_g), as well as $\mathbf{s}$. The complete chain of function compositions from $\mathbf{x}$ to $\mathbf{g}$ can be represented as $F_g(\mathbf{x})$, where $F_g = G_g \circ H_g$ and $\circ$ represents the function composition operator. Similarly, the features from H_s and H_g are combined using matrix addition and fed to another functional block of layers, G_s that predicts the target species $\mathbf{s}$. The composition of functions mapping $\mathbf{x}$ to $\mathbf{s}$ can thus be given by $f(\mathbf{x})$, where $F = G_s \circ (H_g + H_s)$.

To train the functional blocks in the complete *HGNN* architecture, the following objective function is minimized:

$$\min_{H_s, H_g, G_s, G_g} \lambda_{\mathbf{s}} \; L_{\mathbf{s}}(\mathbf{s}, t_{\mathbf{s}}) + \lambda_{\mathbf{g}} \; L_{\mathbf{g}}(\mathbf{g}, t_{\mathbf{g}}) \tag{4.1}$$

where $L_{\mathbf{s}}$ and $L_{\mathbf{g}}$ are loss (or error) functions defined on the space of species labels and genus labels, respectively, on the training set. Specifically, these loss functions act as a measure of the difference between the correct classification ($t_{\mathbf{s}}$ and $t_{\mathbf{g}}$), and the prediction ($\mathbf{s}$ and $\mathbf{g}$) on the training samples, respectively. Cross-entropy function is used as the loss function. Further, $\lambda_{\mathbf{s}}$ and $\lambda_{\mathbf{g}}$ are trade-off hyper-parameters balancing the relative importance of $L_{\mathbf{s}}$ and $L_{\mathbf{g}}$, respectively; their values are automatically assigned using the adaptive smoothing algorithm proposed in [105]. Both the softmaxed outputs, $\mathbf{s}$ and $\mathbf{g}$, are probability vectors whose entries range from 0 to 1 proportional to the model's credence about each species and genus class, respectively.

After some experimentation, I have found that the best point to extract the intermediate genus features (i.e. the point between H_g and G_g) is right before the final max-pooling layer. The same point in the other ResNet is used to combine the genus and species features. Transfer learning [149] is employed using a ResNet pre-trained on the ImageNet benchmark. This ResNet is then fine-tuned by optimizing the loss function in equation (4.1) on the fish training dataset of interest.

The datasets used in the *HGNN* experiment comprise of images contributed by five museums that participated in the Great Lakes Invasives Network Project (GLIN). More information on the curation and preprocessing of these datasets, including data augmentation [135], can be found in Appendix F. Table 4.1 gives a statistical summary of these datasets.

Table 4.1: Statistics of the GLIN dataset and its subsets, which are used for training and evaluating *HGNN*.

Dataset	# of images	# of species	# of genera	# of images per species	Notes
GLIN (All)	63,758	575	187	1 to 7,935	The entire GLIN dataset.
Hard	4,882	102	26	30 to 50	A subset curated from all five museums. Poses a challenge due to the large number of labels (species), and the difference in the distribution of images among the museums.
Easy/50	1,900	38	11	50	A subset curated only from one museum, making the image distribution simpler and easier to learn than **Hard**.
Easy/100	3,762	38	11	63 to 100	Similar to **Easy**/50, but with more images per species, making the classification problem even easier.

4.2.1 Improving Model Prediction Under Noise

In this section, using (*HGNN*) [38], I demonstrate how having integrated the knowledge of taxonomic groupings (e.g., between species and genus classes) into its architecture and training scheme, robustness to adversarial occlusions, which mimic imperfections in real-world images, are significantly improved compared to a black-box model, which is a simple ResNet in this context.

As has already been discussed in Section 2.3, saliency maps can be a great tool that offers model interpretability and highlights regions of importance in an image. When used within the fish classification problem, saliency maps can offer an insight into which traits of the

fish contribute most to its classification. To that end, I use saliency maps to investigate the resiliency (or robustness) of neural networks to adversarial occlusions by occluding regions (or patches) with high saliency scores in the input image. Such occlusion impacts a model's reliability at making correct predictions. Consequently, the model's performance can be stress-tested by starving it off information from salient image regions.

To measure the robustness of a model to adversarial occlusions, the patches with the highest saliency score contributions are iteratively covered, thus confusing the model into making incorrect predictions. Each iteration, the probability of the correct class predicted by the model on an input image $\mathbf{x}$ is calculated and then averaged over all test images as $\mathbb{E}_{\mathbf{x}}(P_{t_s}(\mathbf{x}))$. The higher this metric, the less confused the model is about the input. Furthermore, by measuring drops in this metric as a consequence of a series of adversarial occlusions, the quality of the traits to which the model is sensitive can be evaluated.

Fig. 4.3 shows an example of this process on an illustrative fish specimen from the **Easy**/50 dataset. From left to right, the figure shows a progression from an image with no occlusion towards applying more patches of adversarial occlusions (shown as green square patches) on the same image. Below each image is the model's predicted probabilities over the 5 most probable species sorted in descending order, for both *HGNN* (top row) and *black-box* (bottom row). We make a number of observations here. First, saliency maps of both models highlight the features of importance for classifying a fish, namely the eye, nostrils, and the dorsal fin. However, notice that the saliency maps for *HGNN* are slightly different from those of *black-box*, demonstrating that the two models are not looking at the image in the exact same way (i.e., they have distinct saliency maps). Consequently, and for a fair comparison, the patches for each model will have to be applied at different positions. Second, even when there is no occlusion, while *black-box* makes the correct prediction, its probability of the correct species class is significantly lower than that of *HGNN*'s. This demonstrates

HGNN's ability to extract more useful and generalizable features. Third, after applying two patches of occlusions (in the middle column), we notice that even though both models get the species right, *black-box*'s second guess is not within the correct genus. Finally, and most importantly, after applying four patches of occlusions (in rightmost column), we notice that while both models start predicting the wrong class, *HGNN* is still within the correct genus, while *black-box* is not. It follows that the *black-box* model is not learning phylogenetic features that could be used in other tasks, such as trait segmentation.

To drive this point home, I automate this process for the entire **Easy**/50 dataset and compute the average predicted probability of the correct class across all images, as a function of the number of adversarial occlusions applied to the images. Table 4.2 reports the results for each number of patches ranging from 0 (no occlusion) to 4. We can see that *HGNN* shows higher average probability of the correct class across all number of patches in comparison with *black-box*. This demonstrates *HGNN*'s ability to generalize and handle image imperfections better, especially when the most informative (or salient) regions of the image are occluded.

Table 4.2: Average probability of the correct species class predicted by *black-box* and *HGNN* as a function of the number of adversarial occlusions applied to every image.

Model	Number of Occlusion Patches				
	0	1	2	3	4
black-box	0.473	0.355	0.291	0.232	0.187
HGNN	0.482	0.369	0.307	0.256	0.215

4.2.2 Improving Model Prediction Under Limited Data

When training a black-box model, it is common to observe higher generalization errors when the amount of training data is small. In the context of species classification, such scarcity of data hinders the successful application of CV in wildlife [156]. However, when *HGNN* is

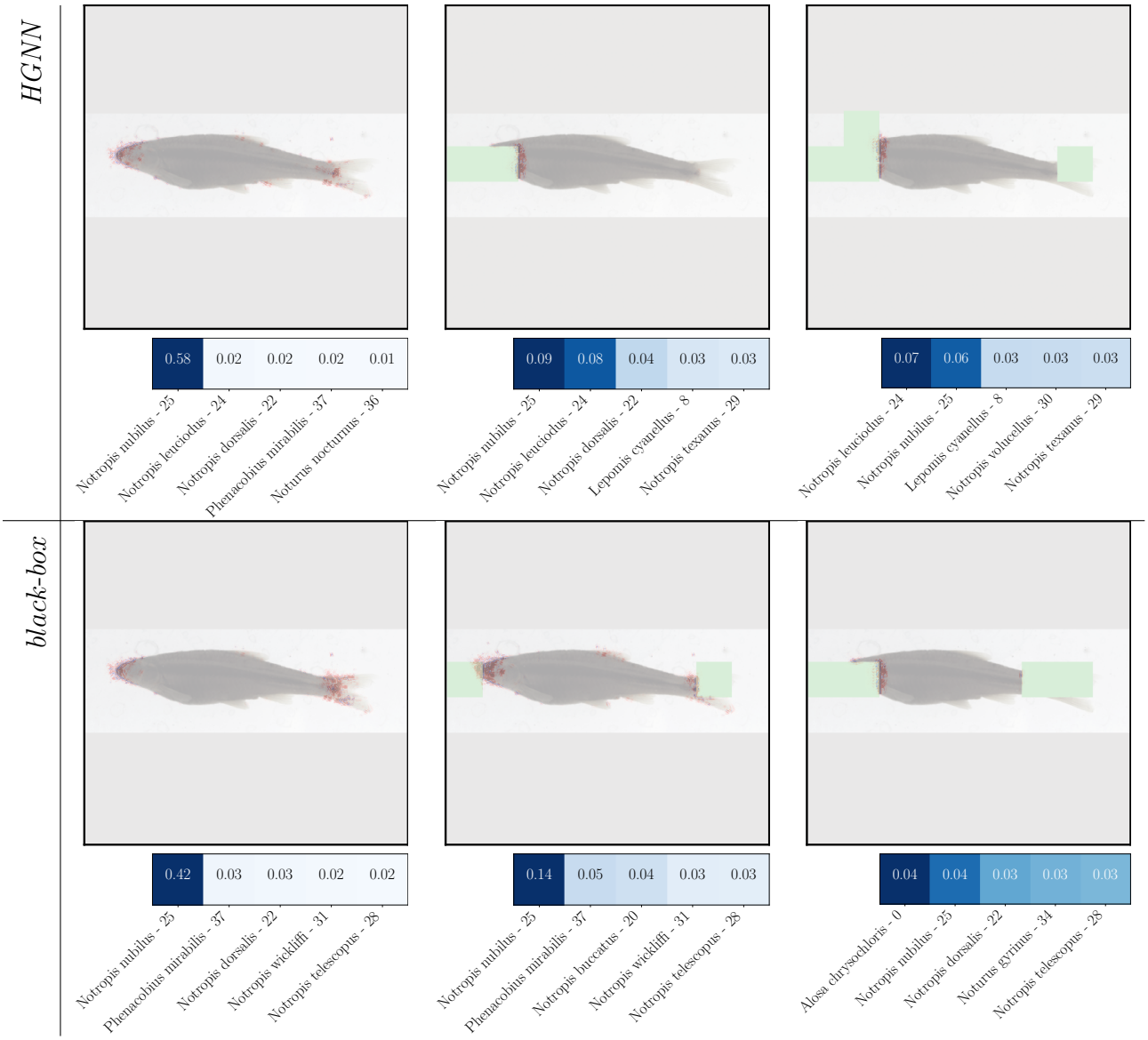


Figure 4.3: Saliency maps showing the effect of adversarial occlusions.

used, I show that by including a biological knowledge-guided loss term (see Equation 4.1) in the learning objective, a reasonably good generalization performance is especially achieved in situations where training data is scarce. This is in alignment with the observations made in previous work [68]. In Fig. 4.4, I compare the classification performance of *HGNN* to that of a baseline black-box neural network architecture comprising of a single ResNet on each of the three data subsets mentioned in Table 4.1. The f1-score [150] is used to measure classification performance. Box plots are used to show the model's performance over five random runs of model training. Since the boxes for the two models do not overlap, that means there is at least 95% confidence [101] that the median accuracy of one is higher than the other's, implying a statistically significant result.

Two observations can be made here. First, as datasets become more complex (e.g., the **Hard** dataset) and/or subject to less training data (e.g., the **Easy**/50), the performance of the model deteriorates. Second, and more importantly, *HGNN*'s advantage is pronounced exactly when data is scarce and the dataset is complex. The non-overlap of the models' boxes for both **Easy**/50 and **Hard** highlights *HGNN*'s ability to compensate for the relative lack of data with respect to dataset complexity by incorporating biological knowledge.

4.3 The Effect of Knowledge Guidance on Learning A Scientifically Meaningful Latent Space

After showing in Section 4.2 how the *KGML* framework can yield a model where the output is resilient to noise and the pathologies of data scarcity, I turn my attention in this section to studying the advantages of using *KGML* from the perspective of the embedding space.

In this section, I explore the following quintessential question: *Can the* KGML *framework*

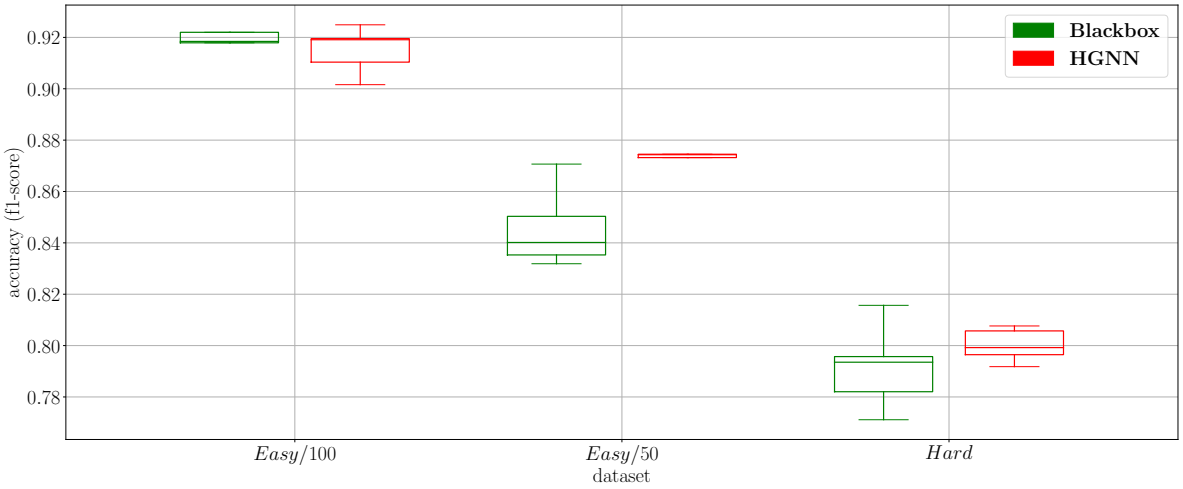


Figure 4.4: *HGNN*'s vs. *black-box*'s classification performance across different subsets of the GLIN dataset.

help devise a method that discovers biologically-valid and anatomically-relevant species traits and, as a by-product, deliver better generalization performance on downstream ML tasks?". Answering this question entails learning species traits as features in the latent space. The properties of this learned space and how it corresponds to domain knowledge (e.g., biology in this application) is the target of my study.

Taking the application of trait discovery, I devise a *KGML* model the latent space of which incorporates the complex biological knowledge available as time-scaled phylogenies. Such a **phylogeny-informed neural network** model (henceforth referred to as *Phylo-NN*) can extract useful feature representations of biological images that correspond to species traits. These features can help in improving the performance of downstream ML tasks such as species classification and trait segmentation. Also, this effort helps biologists define groups of organisms, their genetic and developmental underpinnings, and their interactions with environmental selection pressures [62]. Discovering these traits is also a starting point for linking traits to underlying genetic factors.

It is crucial, however, to separate evolutionary information from unrelated factors. To do

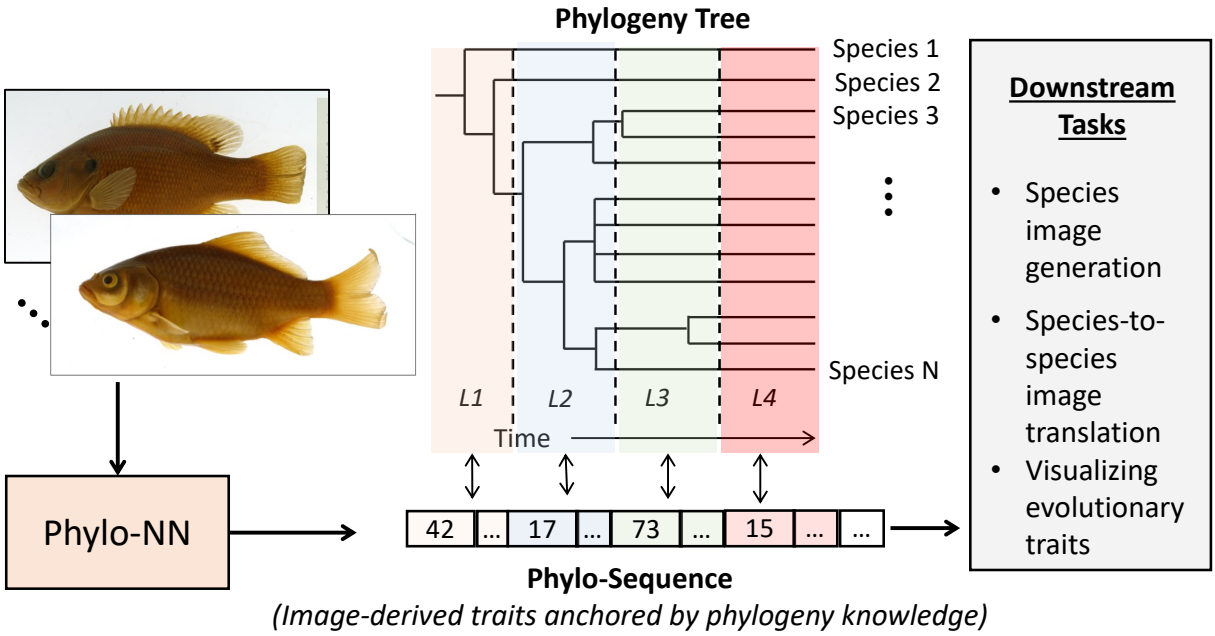


Figure 4.5: *Phylo-NN* converts images to discrete Phylo-sequences where different segments of the sequence (shown in distinct colors) capture evolutionary information at different levels of the phylogenetic tree (L1 to L4).

so, *Phylo-NN* encodes the image of an organism into a sequence of quantized feature vectors or "codes", where different segments of the sequence capture evolutionary signals at varying levels in the phylogeny tree (see Figure. 4.5), and other segments are reserved form non-evolutionary signals that are necessary to reconstruct the specimen. Analogous to gene sequences, these image-derived sequences (termed *Phylo-sequences*) enable discovering evolutionary traits that one species shares with other species as dictated by the phylogenetic tree.

Here, a decision is made to discretize the phylogeny tree with $n_l = 4$ levels, such that every species class (leaf node in the tree) has exactly $n_l - 1$ ancestors. More on the phylogeny preprocessing pipeline can be found in Appendix. G.

Figure 4.6 provides an overview of *Phylo-NN* and how it operates on the latent space of

any backbone encoder model E that takes in images as input and produces feature maps as output.

There are three computing blocks in *Phylo-NN* as shown in Figure. 4.6. The first block, Phylo-Encoder (*PE*), takes an input image and generates quantized feature sequences as output. These sequences comprise of two *disentangled* parts: $\mathbf{z}_{\mathrm{p}}^{Q}$, which captures evolutionary signals in the phylogeny (p) tree, and $\mathbf{z}_{\mathrm{np}}^{Q}$, which captures non-phylogeny (np) information that is still important for image reconstruction but is not influenced by evolutionary signals. The second block, Phylo-Decoder (*PD*), maps the sequences back to the original space of feature maps such that it exhibits good reconstruction. Then, the reconstructed feature map is fed into a backbone decoder model D that reconstructs the original image. Note that both the backbone models E and D are kept frozen while training, thus requiring low training time. Finally, to synthesize Phylo-sequences of a specimen belonging to a specific species, the third block of *Phylo-NN*, a transformer model T, takes in the species class as input, and generates a distribution of plausible Phylo-sequences corresponding to that class as output. These sequences can be fed to the *PD* model to generate a distribution of synthetic images.

Figure. 4.7 shows the operations performed inside the *PE* block. With a feature map of size $(H \times W \times C)$, the channels are split into two sets. The first C_{p} maps are fed into an MLP layer to learn a global set of feature vectors $\mathbf{z}_{\mathrm{p}}$ capturing phylogeny information. The size of $\mathbf{z}_{\mathrm{p}}$ is kept equal to $(n_{\mathrm{l}}\ n_{\mathrm{p}} \times d)$, where n_{l} is the number of phylogeny levels, n_{p} is the number of feature vectors we intend to learn at every phylogeny level, and d is the dimensionality of feature vectors. Similarly, the remaining maps are fed into an MLP layer to produce a set of feature vectors $\mathbf{z}_{\mathrm{np}}$ capturing non-phylogeny information of size $(n_{\mathrm{np}} \times d)$

Both $\mathbf{z}_{\mathrm{p}}$ and $\mathbf{z}_{\mathrm{np}}$ are converted to *quantized* sequences of feature vectors, $\mathbf{z}_{\mathrm{p}}^{Q}$ and $\mathbf{z}_{\mathrm{np}}^{Q}$, respectively, using the approach developed in *VQ-VAE* [114]. The basic idea of this quantization approach is to learn a set (or codebook) of n_{q} distinct feature vectors (or codes), such that

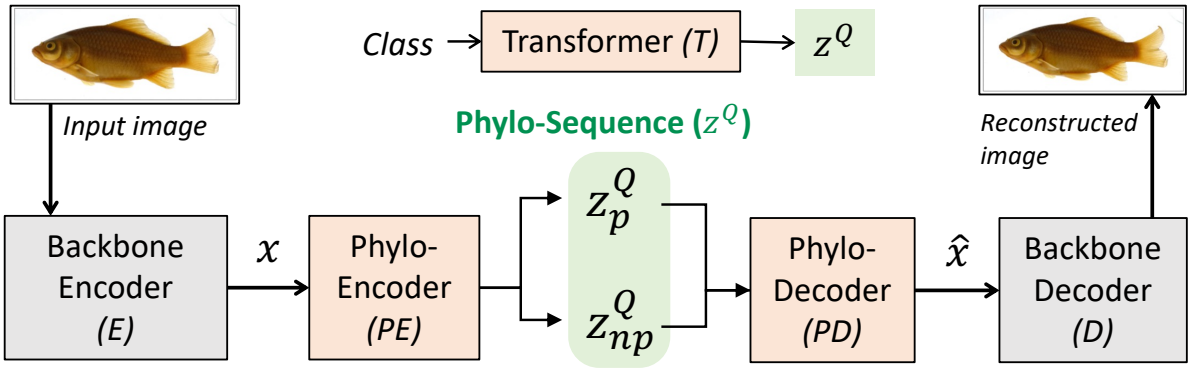


Figure 4.6: Overview of *Phylo-NN* model architecture.

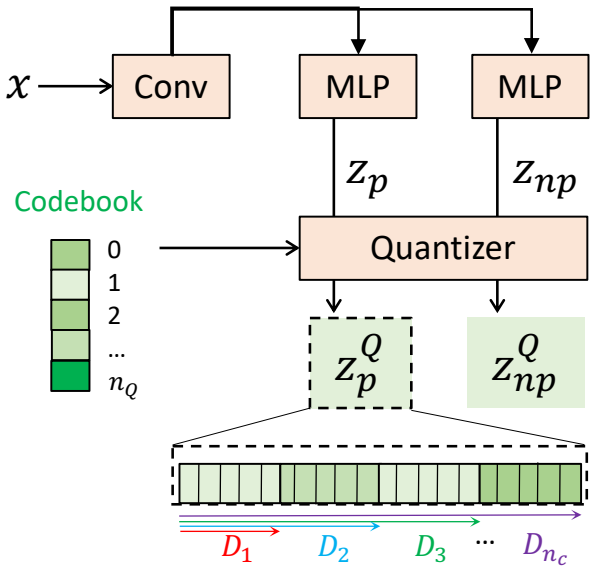


Figure 4.7: Detailed view of the Phylo-Encoder block in *Phylo-NN*.

every feature vector in $\mathbf{z}_{\mathrm{p}}$ and $\mathbf{z}_{\mathrm{np}}$ is replaced by its nearest counterpart in the codebook. This is achieved by minimizing the *quantization loss*, $L_{\mathrm{q}} = |\mathbf{z} - \mathbf{z}^{Q}|$. The advantage of working with quantized vectors is that every feature vector in $\mathbf{z}_{\mathrm{p}}^{Q}$ and $\mathbf{z}_{\mathrm{np}}^{Q}$ can be referenced just by its location (or index) in the codebook. This allows for faster feature manipulations in the space of discrete code positions than continuous feature vectors.

To guarantee that $\mathbf{z}_{\mathrm{p}}$ and $\mathbf{z}_{\mathrm{np}}$ exhibit properties that make them useful from a biological point of view, several constraints in the form of loss functions are imposed on *Phylo-NN*:

1. **Using phylogenetic knowledge:** To ensure that the quantized feature sequence $\mathbf{z}_{\mathrm{p}}^{Q}$ contains phylogeny information, I design a novel *phylogeny loss* for training $\mathbf{z}_{\mathrm{p}}^{Q}$. Note that $\mathbf{z}_{\mathrm{p}}^{Q}$ contains n_{l} sub-sequences of length n_{p}, where every sub-sequence corresponds to a different level of the phylogeny tree. While the first sub-sequence S_1 should capture information that is necessary for identifying ancestor nodes at level 1 of the phylogeny tree, S_2 should contain additional information that when combined with S_1 is sufficient to identify the correct ancestor node at level 2. In general, a Phylo-descriptor D_i is defined such that $D_i = \{S_1, S_2, \ldots, S_i\}$ that contains the necessary information for identifying nodes at level i (See Figure. 4.7). Each D_i is fed to an MLP layer that predicts the class probabilities of nodes at level i, which are then matched with the correct node class at that level, $c_i(\mathbf{x})$, by minimizing the following phylogeny loss, L_{p}:

$$L_{\mathrm{p}} = \sum_{i=0}^{n_{\mathrm{l}}} \beta_i \mathrm{CE}(\mathrm{MLP}_i(D_i(\mathbf{x})), c_i(\mathbf{x})) \tag{4.2}$$

   where CE is the cross-entropy loss and β_i is the weighting hyper-parameter for level i.

2. **Disentangling phylogenetic and non-phylogetic information:** The literature has a number of approaches that can be used to disentangle the features of a deep learning model and align them with target "concepts." This includes the approach of

"Concept whitening" [23], where the latent space of a classification model is whitened (i.e., normalized and decorrelated) such that the features along every axis of the latent space corresponds to a separate class. Another approach in this area is that of Latent Space Factorization [93], where the latent space of an autoencoder is linearly transformed using matrix subspace projections to partition it into features aligned with concept attributes and those that capture non-attribute information.

To ensure that $\mathbf{z}_{\text{np}}^{Q}$ focuses on complementary features and does not contain phylogeny information, an orthogonal convolution loss L_{o} (originally proposed in [164]) is applied to the convolutional layer of Phylo-Encoder, to constrain the learned convolutional kernels to be orthogonal to each other. To further ensure that $\mathbf{z}_{\text{np}}^{Q}$ has no phylogeny information, an *adversarial training* procedure is employed to incrementally remove phylogeny information from $\mathbf{z}_{\text{np}}^{Q}$. This is done by adding an MLP_{adv} layer that is used to construct the following *adversarial loss*:

$$L_{\text{adv}} = \sum_{i=0}^{n_{\text{l}}} \beta_i \text{CE}(\text{MLP}_i(MLP_{\text{adv}}(\mathbf{z}_{\text{np}}^{Q}(\mathbf{x}))), c_i(\mathbf{x})) \tag{4.3}$$

The parameters of MLP_{adv} are trained to detect any phylogeny information contained in $\mathbf{z}_{\text{np}}^{Q}$ by minimizng L_{adv}, while simultaneously maximizing it on the rest of *Phylo-NN*'s parameters, making $\mathbf{z}_{\text{np}}^{Q}$ irrelevant for the task of identifying nodes in the phylogeny tree.

4.3.1 Knowledge-Guided Information Disentanglement

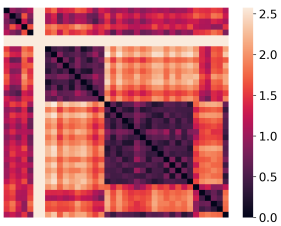
In order to evaluate the ability of *Phylo-NN* to extract evolutionary traits from images in an *unsupervised* manner (i.e., without using trait labels), I calculate the distances between species pairs in the embedding space of *Phylo-NN* and find their correlation with ground-

truth (GT) values, the phylogenetic ground truth, and the morphological ground truth. Phylogenetic ground truth corresponds to the *evolutionary distance* between species in the phylogenetic tree (i.e., the phylogenetic distance between two species is the sum total of edge lengths that is needed to traverse the path between their nodes in the phylogeny). The longer the path, the more distant the species are on the evolutionary scale.

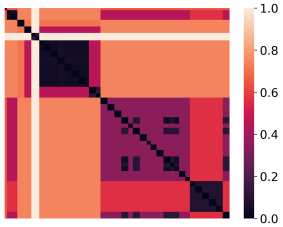
Morphological ground-truth characterizes the similarity in the traits of two species by using ground-truth measurements of linear morphological traits obtained from the FishShapes v1.0 dataset [117]. More on this dataset can be found in Appendix H.

To compute pair-wise distances in the embedding space of *Phylo-NN*, I first compute the probability distributions of quantized codes at every position of the Phylo-sequence (i.e., $\mathbf{z}_{\text{p}}^{Q}$ and $\mathbf{z}_{\text{np}}^{Q}$) in the test images for every species. I then compute the Jensen-Shannon (JS) divergence [103] between the probability distributions of codes for each pair of a species to measure the similarity of their learned *Phylo-NN* embeddings. A similar approach for computing the JS-divergence of species-pairs in the quantized feature space of vanilla VQGAN is used.

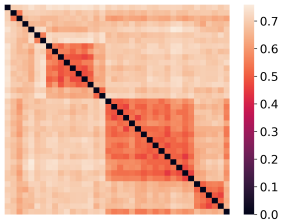
Figures 4.8(a) and 4.8(b) show the pair-wise species distance matrices for morphological and phylogenetic GTs, respectively. We can see that both ground-truths show a similar clustering structure of species, indicating groups of species that share evolutionary traits. However, there are differences too; while phylogenetic GT is solely based on phylogeny, the morphological GT uses both the phylogeny and information about phenotypic traits. Figures 4.8(c) and 4.8(d) show the JS-divergences among species computed separately for the two disentangled parts of PhyloNN's embeddings ($\mathbf{z}_{\text{p}}^{Q}$ and $\mathbf{z}_{\text{np}}^{Q}$). We can see that the embeddings containing phylogenetic information show a similar clustering structure of distances as the GT matrices, in contrast to the non-phylogenetic embeddings. This shows the ability of *Phylo-NN* to disentangle features related to phylogeny from other unrelated features. Figure



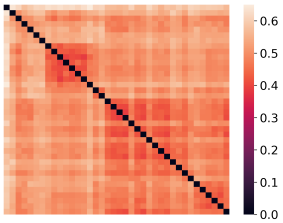
(a) Morphological ground truth



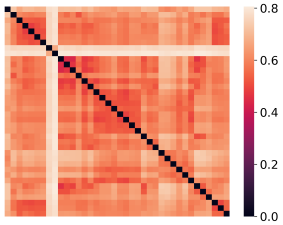
(b) Phylogenetic ground truth



(c) JS-divergence for *Phylo-NN*'s phylogenetic embedding



(d) JS-divergence for *Phylo-NN*'s non-phylogenetic embedding



(e) JS-divergence for Vanilla VQGAN's [39] embedding

Figure 4.8: Comparing embedding distance matrices of methods with morphological and phylogenetic ground-truths

Table 4.3: Correlations between GT distances and embedding distances

		Morphological	Phylogenetic
PhyloNN	level0	0.86	0.83
	level1	0.87	0.85
	level2	0.78	0.83
	species	0.70	0.78
vanilla VQGAN		0.31	0.24

4.8 also shows the embedding distance matrix of vanilla VQGAN, which does not show a good visual correspondence such as that of *Phylo-NN*'s in terms of matching with the GT matrices.

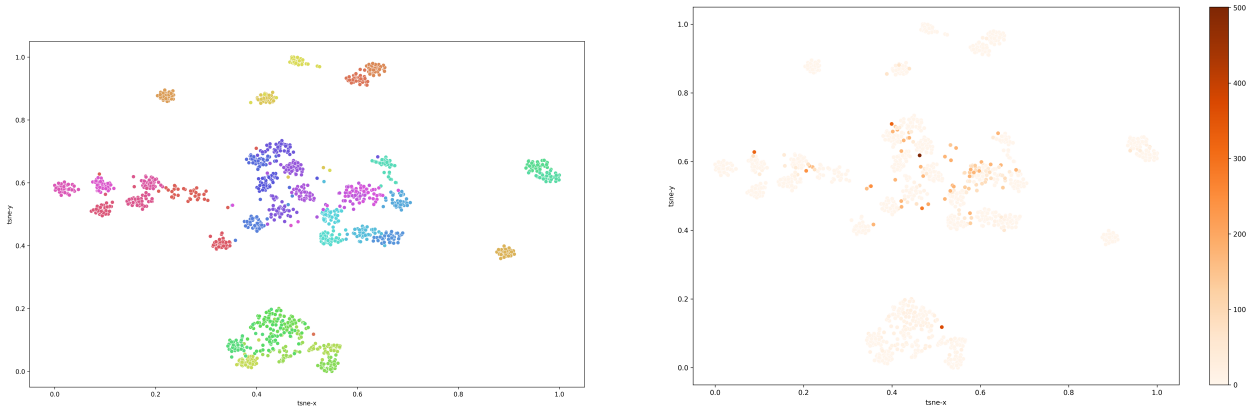
To quantitatively evaluate the ability of *Phylo-NN* to match with GT distances compared to a vanilla VQGAN, I compute the Spearman correlation between the GT distance matrices and embedding distance matrices for different methods as shown in Table 4.3. We can see that *Phylo-NN* shows higher correlations at the species level with both morphological and phylogenetic GTs. Furthermore, in contrast to a vanilla VQGAN, *Phylo-NN* allows us to compute the distance matrix at any ancestor level and compare it with GT matrices at the same ancestor level. We can see that *Phylo-NN* shows significantly higher correlations with GT matrices at higher ancestor levels than the species level.

An alternative approach I follow to assess *Phylo-NN*'s generated images is by visualizing their embedding space using t-SNE plots [155]. More precisely, I investigate how *Phylo-NN* clusters the embedding space compared to a vanilla VQGAN baseline by analyzing these models' t-SNE plots.

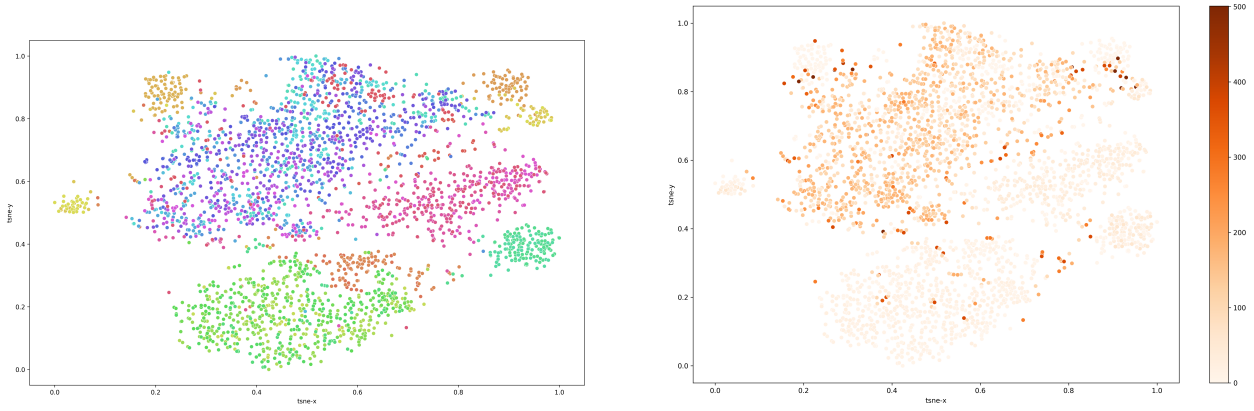
To construct the t-SNE plot for each model, I iterate through its Transformer T generated images, encode them, obtain the quantized embedding vector for each image ($\mathbf{z}_{\mathrm{p}}^{Q}$ and $\mathbf{z}^{Q}$ for *Phylo-NN* and vanilla VQGAN, respectively), and finally create the t-SNE plots.

Figure 4.9 shows these constructed t-SNE plots with two different color-coding schemes. The first one (left column) color-codes the data-points based on the grouping of species at the second phylogenetic level (i.e., the direct ancestor of the specimen's species). Using this color-coding scheme allows us to inspect how different species cluster in the embedding space. The second color-coding (right column) is the average phylogenetic distance between the data-point and its k-nearest neighbors (KNN), where $k = 5$ in this setup. The higher the average distance (i.e., the darker the data-point's color), the more phylogenetically distant the specimen is from those k specimen's that are closest to it in the quantized embedding space. This color-coding helps us spot how well the different species are separated from each other in the embedding space, which generally characterizes the quality of the encoding and its propensity for downstream tasks, such as classification.

Looking at Figure 4.9, we can see that *Phylo-NN* (top row) clearly clusters the generated images better than vanilla VQGAN as evident from its hierarchical clustering where the specimens belonging to the same species clump into small clusters and these clusters in turn clump into larger clusters (representing ancestor nodes) that have a singular color. This demonstrates that *Phylo-NN* is able to learn a phylogenetically-meaningful encoding, whereas vanilla VQGAN's clustering is quite fuzzy and poorly characterizes any biological knowledge. Also, by looking at the right column, we can see that *Phylo-NN* commits very little clustering error in terms of its phylogenetic constraints because the average phylogenetic distance is low (almost zero) for the majority of the data points. This is in contrast to the vanilla VQGAN where there is quite a high clustering error as seen from the "heat" of its scatter plot.



(a) *Phylo-NN*



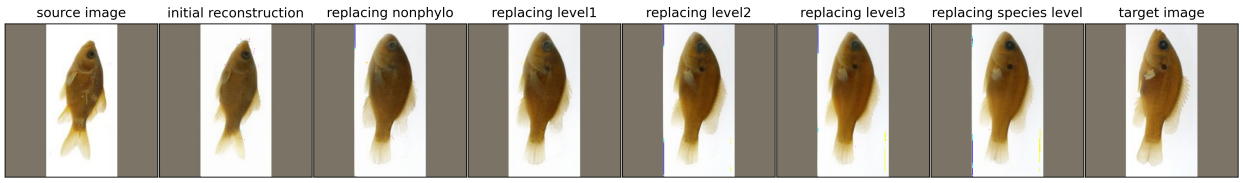
(b) Vanilla VQGAN

Figure 4.9: t-SNE plots of the images generated by *Phylo-NN* and vanilla VQGAN.

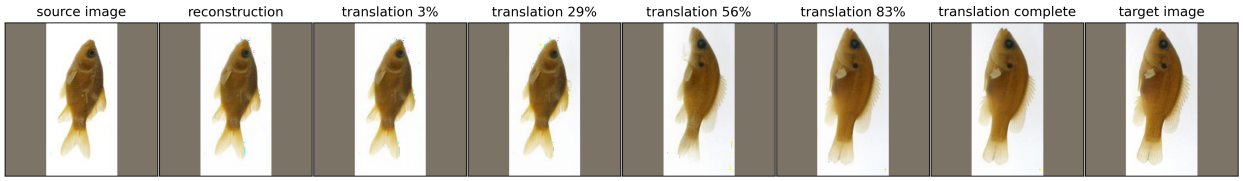
4.3.2 Knowledge-Guided Image Synthesis

To further assess how well *Phylo-NN*'s learned embedding captures phylogenetic traits, I investigate how altering the quantized embedding of an image specimen incrementally in a phylogenetically meaningful ordering affects the morphological traits when the altered embeddings are decoded back as an image. this is done by picking two specimen images from a pair of species. By encoding the two images using *Phylo-NN*, we obtain their corresponding encodings, $\mathbf{z}_1^Q$ and $\mathbf{z}_2^Q$. We then start to replace the codes in the Phylo-sequence $\mathbf{z}_1^Q$ with the corresponding codes in $\mathbf{z}_2^Q$ iteratively, until $\mathbf{z}_1^Q$ transforms completely into $\mathbf{z}_2^Q$. The order of this iterative replacement is by first replacing the codes representing the non-phylogenetic part of the embedding $\mathbf{z}_{\text{np}1}^Q$, then the part representing the information in the earliest ancestral level (level 0), to the next ancestor level (level 1), till we eventually reach the last level of the phylogenetic tree, which is the species level. At the final point, the entire Phylo-sequence $\mathbf{z}_1^Q$ has been replaced with $\mathbf{z}_2^Q$. This phylogeny-driven ordering of code replacements helps us capture key "snapshots" of the species-to-species translation process that are biologically meaningful. In particular, by observing the traits that appear or disappear at every ancestor level of code replacement, we can infer and generate novel hypotheses about the biological timing of trait changes as they may have happened in the process of evolution.

Figure 4.10 shows an example of such a translation process between a specimen of the species Carassius Auratus to a specimen of the species Lepomis Cyanellus. We can see that although the two specimens look similar on the surface, there are several subtle traits that are different in the two species that are biologically interesting. For example, the source species has a V-shaped tail fin (termed caudal fin), while the target species has a flatter tail fin. By looking at their place of occurrence in the translation process of *Phylo-NN*, we can generate novel biological hypotheses of whether they are driven by phylogeny or not, and whether they appeared earlier or later in the target species in the course of evolution.



(a) *Phylo-NN*



(b) Vanilla VQGAN

Figure 4.10: Comparing species-to-species image translations from a Carassius Auratus specimen to a Lepomis Cyanellus specimen

For example, we can see that the flat tail feature of the target species appears right after replacing the non-phylogenetic part of the embedding, indicating that this feature may not be capturing evolutionary signals and instead maybe be affected by the environment. On the other hand, if we observe another fin that appears in the middle of the body of the target species (termed pectoral fin), we can see that it seems to get more prominent only in the later levels (it is absent in level 0). This suggests the hypothesis that the presence of pectoral fin in the target species may have been added later in the course of evolution. Our work opens novel opportunities for generating such biological hypotheses, which can be further investigated by biologists to potentially accelerate scientific discoveries. The figure also shows the translations obtained by a vanilla VQGAN for the same pair of species specimens. We can see that the baselines are mostly performing a smooth interpolation between the source and target images. In particular, the transition points in the translation process do not correspond to biologically meaningful events, since they only rely on the information contained in data and do not use biological knowledge.

4.3.3 Generalizing on Unseen Classes

As the objective of *Phylo-NN* is to encode specimen images into their corresponding phylogenetic and non-phylogenetic quantized sequences of codes, I am interested in studying the distribution of codes used within different descriptors of phylogenetic concepts (e.g., species nodes or ancestor nodes). Naturally, we expect specimens that belong to the same species to largely share the same phylogenetic code in terms of the species descriptor D_{n_l}, while varying in terms of the non-phylogenetic codes. More generally, specimens belonging to species that share a common ancestor at a phylogenetic level i should largely share the codes with D_i while varying in terms of the rest of the phylogenetic and non-phylogenetic codes. This should also apply for specimens of *unseen* (or newly discovered) species that we have not yet observed in the training set. By looking at the similarity of codes generated for an unseen species during testing, we should be able to infer its ancestor lineage in terms of the species sampled during training.

To quantify this phenomenon, for a given phylogenetic concept of interest, I construct two sets of histograms, H_p and H_np of sizes $[n_\text{l} \times n_\text{p}]$ and $[n_\text{np}]$, respectively. Each value in the histograms, ${H_\text{p}}^{i,j}$ and ${H_\text{np}}^{k}$, describes the distribution of codes of that corresponding location in the sequence across all the specimen in the dataset of interest. Take a look at Appendix I for some examples.

Once these histograms are constructed, I evaluate the purity of *Phylo-NN* codes by calculating the entropy of each histogram in H_p and H_np. If the entropy is low for a certain sequence location, it means only a few possible codes occur at that location, alluding to the fact that those specific codes at that location are key at characterizing the phylogenetic concept in question. On the other hand, higher entropy means a variety of codes occur at that location, implying that such a location is not discriminative to the phylogenetic concept of interest.

Finally, to compare the code distributions for two species, I use the JS-divergence metric for calculating the difference between two histograms of a sequence location. Similar to what is done in Section 4.3.1, such a metric can be aggregated to quantify the coding differences between the species in question.

To assess the *Phylo-NN*'s ability to generalize to unseen species, I train it on a subset of the species in the original dataset and then evaluate the quality of the embedding space when the model is introduced to the left-out species (three species in my experiments). Once the model is trained, I look at the average JS-divergence distance between each of the three missing species and three other species in the tree. These three other species were selected such that each missing species has one seen species that is close to it phylogenetically (i.e., both species share the same ancestor at the immediate ancestor level) while the others are relatively far from it.

Table 4.4 shows the average distance of the phylogenetic codes among the six aforementioned species. Looking at the table, one can see that this distance is smallest for each unseen species and its counterpart that shares the same immediate ancestor (shown as the diagonal in the table). This confirms that even though the model has not seen the former species, it was able to characterize it using a coding sequence that is significantly closer to that of its seen counterpart than the other species'.

While Table 4.4 highlights the phylogenetic matching in the embedding space at the species descriptor level, D_{n_l}, Table 4.5 does the same but for the descriptor at a distant ancestor level (level 0), i.e., D_0. Based on the phylogeny tree we have used in this example, both the *Notropis* and *Noturus* species share the same distant ancestor at that descriptor level. On the other hand, *Lepomis* species does not share that ancestor. Hence, we find that the JS-divergences increase for the *Lepomis* unseen species with seen species that are not *Lepomis* as compared to Table 4.4. On the other hand, the JS-divergences decrease for the other two

Table 4.4: JS-diveregence of the phylogenetic codes at the species level between unseen and seen species

		Seen species		
		Notropis nubilus	Lepomis macrochirus	Noturus flavus
Unseen species	Notropis percobromus	0.47	0.71	0.62
	Lepomis megalotis	0.73	0.43	0.72
	Noturus miurus	0.62	0.71	0.48

Table 4.5: JS-diveregence of the phylogenetic codes at the earliest ancestral level between unseen and seen species

		Seen species		
		Notropis nubilus	Lepomis macrochirus	Noturus flavus
Unseen species	Notropis percobromus	0.26	0.81	0.50
	Lepomis megalotis	0.81	0.27	0.81
	Noturus miurus	0.52	0.80	0.31

unseen species w.r.t. seen species that are on the off-diagonals of the table. This confirms that D_0 specifically captures the phylogenetic information of that distant ancestor that is common across *Notropis* and *Noturus* seen and unseen species.

Finally, to confirm that this phylogenetic correlation is mainly constrained to the phylo-descriptors, I calculate the same distances but using the non-phylogenetic part of the sequences. The result is shown in Table 4.6. Here, we can see the distances are much closer to each other, implying that the non-phylogenetic embedding is not specialized at differentiating among the different species, and hence cannot be used to phylogenetically categorize the unseen species.

Table 4.6: JS-diveregence of the non-phylogenetic codes between unseen and seen species

		Seen species		
		Notropis nubilus	Lepomis macrochirus	Noturus flavus
Unseen species	Notropis percobromus	0.39	0.45	0.39
	Lepomis megalotis	0.46	0.36	0.48
	Noturus miurus	0.40	0.42	0.36

Chapter 5

The Effect of Knowledge Guidance from The Loss Landscape Lens

In Chapter 4, I discussed the benefits of using the *KGML* framework such that the model's output space exhibits certain desired and advantageous properties. In this chapter, I turn my attention to investigating *KGML* models from the loss landscape lens. Namely, I investigate how applying the *KGML* framework influences the model's optimization and how different loss terms interact under its regime. This investigation helps decipher *KGML*'s effectiveness, understand how to overcome its pitfalls, and achieve a better understanding of the mechanics of *KGML*. To achieve this goal, I apply the loss landscape visualization tool onto different problems, provide my observation on its usefulness and disadvantages, and further improve on it by addressing some of those disadvantages, thus making it more attractive to researchers in the field.

In the next sections, I conduct my investigation in the context of physics where domain knowledge is encoded in the form of equations. Namely, I start by studying a family of problems that are characterized by eigen-decomposition, and then move to other benchmark problems in physics, such as PDEs.

5.1 Challenges and Solutions to Optimizing Multiple Physics Objectives

The optimization of multiple objectives (i.e., loss terms) has been a common theme in multi-task learning [18] generally, and *KGML* specifically. Such loss terms are integrated into the learning objective of a neural network by adding them together using trade-off coefficients (or hyper-parameters). Traditionally, such trade-off coefficients are kept constant across all epochs of gradient descent [68, 72]. This inherently assumes that the importance of PG loss terms in guiding the learning process towards a generalizable solution is constant across all stages (or epochs) of gradient descent, and they are in agreement with each other. In reality, however, 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 [29, 72, 133].

To make matters worse, PG loss terms may also show *competing* directions of gradient descent because of differences in their properties (e.g., curvatures), making it difficult to balance them at different stages of ANN training. Finding the optimal setting of these hyper-parameters can be challenging because of the varying nature and gradient dynamics of PG loss and data-driven loss terms [165, 167]. While some recent works have investigated the effects of PG loss on generalization performance [133] and the importance of normalizing the scale of hyper-parameters corresponding to PG loss terms [167], they do not study the effects of competing PG losses.

This challenge is well known for many types of *KGML* models, including Physics-Informed Neural Networks (PINNs) [122]. PINNs have been widely used in the field of scientific

computing to learn governing equations from data. However, one of the main challenges in using PINNs is the difficulty of optimizing them. This is because they combine several physics-based constraints, which leads to optimization pathologies [167] and non-convex optimization landscapes [8]. Furthermore, the hyperparameters in PINNs require careful tuning to ensure good balancing of the loss terms. Balancing these loss terms is crucial for a PINN to converge to an optimal solution.

In the next section, I use the application of eigen-decomposition in physics, as described in Section 3.1, to explain how different loss terms have different importance at different stages of learning, and that it is important to weight them differently (rather than balancing their contributions) using adaptive weighting coefficients to ensure better generalizability.

5.1.1 Solving Competing Physics-Guided Losses for Eigen-equation Problems

In this chapter, the applications of eigen-decomposition in physics, as described in Section 3.1, are considered as a case-study for the challenge of incorporating domain knowledge in terms of equations into the model optimization process.

A naïve ML approach for learning a function f_{NN} that outputs the desired eigen-solution is to minimize the mean sum of squared errors (MSE) of predictions on the training set as follows:

$$\text{Train-Error}_{Ising} := 1/N(\sum_i ||\hat{\boldsymbol{y}}_i - \boldsymbol{y}_i||^2 + ||\hat{b}_i - b_i||^2 \tag{5.1}$$

This definition works reasonably well for the Ising model application. For the electromagnetic application, however, the presence of complex values (see Appendix A) warrants the use of

an overlap-integral-based loss instead. In particular, the error here is re-defined as:

$$\text{Train-Error}_{electromagnetic} := \frac{1}{N}\sum_i \left(-||\frac{\hat{\boldsymbol{y}}_i}{||\hat{\boldsymbol{y}}_i||} \cdot \frac{\boldsymbol{y}_i^*}{||\boldsymbol{y}_i^*||}||^2 + ||\hat{b}_i - b_i||^2 \right) \tag{5.2}$$

where $\boldsymbol{y}^*$ is the complex conjugate of $\boldsymbol{y}$. Similar to the cosine similarity of real-valued vectors, the 'overlap integral' of complex-valued vectors measures the alignment between the prediction $\hat{\boldsymbol{y}}$ and ground-truth $\boldsymbol{y}$ regardless of their scales. Minimizing $\text{Train-Error}_{electromagnetic}$ ensures that the eigen-values are correct and that the profiles of the predicted eigen-vector fit that of the ground-truth.

In the framework of *KGML*, instead of solely relying on Train-Error, 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 eigenvalue equation, $\hat{A}\hat{\boldsymbol{y}} = \hat{b}\hat{\boldsymbol{y}}$. Hence, we consider minimizing the following equation:

$$C\text{-Loss}(\theta) := \frac{1}{N}\sum_i \frac{||\hat{A}_i\hat{\boldsymbol{y}}_i - \hat{b}_i\hat{\boldsymbol{y}}_i||^2}{\hat{\boldsymbol{y}}_i^\top \hat{\boldsymbol{y}}_i} \tag{5.3}$$

where the denominator term ensures that $\hat{\boldsymbol{y}}_i$ does not tend toward the trivial solution of 0. By construction, C-Loss only depends on the predictions of f_{NN} and does not rely on true labels, $(\boldsymbol{y}, b)$. Hence, it can be evaluated even on the unlabeled test data, $\mathcal{D}_U$.

Spectrum Loss: The equation $\hat{A}\hat{\boldsymbol{y}} = \hat{b}\hat{\boldsymbol{y}}$ has 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 minimum. Since we are only interested in a specific eigen-solution for every $\hat{A}_i$, namely the eigen-solution that has the smallest or the largest eigenvalue, we consider minimizing another PG loss term that ensures the predicted $\hat{b}$ at every sample is optimal. In the case

of the quantum mechanics application, we use the following loss to find the smallest eigen-solution:

$$S\text{-Loss}(\theta) := \frac{1}{N}\sum_{i} \exp\left(\hat{b}_i\right) \tag{5.4}$$

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

These loss terms, C-Loss and S-Loss, are integrated into the learning objective of f_{NN} by adding them to Train-Error using trade-off coefficients, λ_C and λ_S, respectively. However, due to the issues previously raised in Section 5.1, setting these coefficients properly proves to be non-trivial. This is especially true since the proposed two PG losses have a widely different number of local minima and thus have different curvatures. On one hand, C-Loss suffers from a large number of local minima since every eigen-solution of $\hat{A}_i$ has C-Loss $= 0$. Among all these solutions, we are only interested in eigen-solutions with the smallest (or largest) eigenvalues across all $\hat{A}_i$. On the other hand, S-Loss is a monotonic function and thus has a single minimum in a bounded domain.

To this effect, the proposed framework *Competing Physics PGNN* (*CoPhy*-PGNN) [37], replaces the constant coefficients with adaptive ones, prefering PG loss terms that show fewer local minima (e.g., S-Loss) to dominate the gradient descent updates in the beginning epochs of ANN training, and then let more complex PG loss terms with larger number of local minima (e.g., C-Loss) dominate the gradient descent dynamics and refine the learned parameters to converge to a generalizable solution θ^*. This framework is inspired from continuation methods developed in the scientific optimization literature for non-convex problems [2], where the strategy is to first optimize a simpler (smoothed) objective function and gradually consider less smoothed ones. It is also related to curriculum learning [9, 142], where

"simpler" examples are presented to the neural network in an incremental manner before showing more "complex" ones.

Thus, *CoPhy*-PGNN imposes an annealing pattern for λ_S, and a cold starting pattern for λ_C. More regoriously, the two adaptive coefficients are defined as follows:

$$\lambda_S(t) = \lambda_{S0} \times (1 - \alpha_S)^{\lfloor t/T \rceil} \tag{5.5}$$

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 sets the frequency of the annealing update.

$$\lambda_C(t) = \lambda_{C0} \times sigmoid(\alpha_C \times (t - T_a)) \tag{5.6}$$

where, λ_{C0} is a hyper-parameter denoting the final 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 cut-off number of epochs after which λ_C is activated from a cold start of 0.

Finally, the overall learning objective of *CoPhy*-PGNN can be written as:

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

Note that Train-Error 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, $\mathcal{D}_U$.

More information about the data and hyper-parameter values is presented in Appendix C.

To illustrate how using PG losses properly as a form of *KGML* guidance can lead to better model optimization, *CoPhy*-PGNN is compared to the following baselines:

1. Black-box NN (or NN): This refers to the black-box ANN model trained just using Train-Error without any PG loss terms.
2. PGNN-*analogue*: The analogue version of PGNN [72] for our problem where the hyper-parameters corresponding to S-Loss and C-Loss are set to a constant value.
3. *CoPhy*-PGNN (only-$\mathcal{D}_{Tr}$): This is an ablation model where the PG loss terms are only trained over the training set, $\mathcal{D}_{Tr}$. Comparison with this model will help in evaluating the importance of using unlabeled samples $\mathcal{D}_U$ in the computation of PG loss.
4. *CoPhy*-PGNN (Label-free): This ablation model drops Train-Error from the learning objective and hence performs label-free (LF) learning only using PG loss terms.

To test *CoPhy*-PGNN's improvement in terms of extrapolation power, which reflects how well it is optimized over the set of different loss functions, I conducted the following experiment within the electromagnetic propagation application. In this experiment, a periodically stratified layer stack of 10 layers of equal length per period is considered. 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 the set $\{1, 4, 9, 16\}$. The period of the multilayer stack is chosen such that the size of $\hat{A}$ is 401×401 complex values. Finding the solution of such eigenv-equation is 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 network can, in principle, learn to predict the eigenvalues and eigenvectors of the matrix $\hat{A}$.

Next, a training size of $|\mathcal{D}_{Tr}| = 370$ realizations with $n \in \{1\}$ in its first layer is taken. On the other hand, the test set's $|\mathcal{D}_U| = 1630$ realizations are left with an unconstrained first layer's refractive index (i.e. $n \in \{1, 2, 3, 4\}$).

Two different types of evaluation metrics are used to assess the predictive accuracy of $\hat{\boldsymbol{y}}$

w.r.t. ground-truth $\boldsymbol{y}$. All metrics are averaged across all test samples.

1. Overlap Integral: Similar to the cosine similarity of real-valued vectors, the 'overlap integral' of complex-valued vectors measures the alignment between the prediction $\hat{\boldsymbol{y}}$ and ground-truth $\boldsymbol{y}$ regardless of their scales. In particular, the overlap integral between $\hat{\boldsymbol{y}}$ and $\boldsymbol{y}$ is defined as

$$\langle \frac{\hat{\boldsymbol{y}}}{||\hat{\boldsymbol{y}}||}, \frac{\boldsymbol{y}^*}{||\boldsymbol{y}^*||} \rangle \tag{5.7}$$

   where $\boldsymbol{y}^*$ is the complex conjugate of the ground-truth eigen-vector $\boldsymbol{y}$. The value of the overlap integral ranges from zero, which means no overlap (or orthogonal), to 1, which means an exact overlap. An overlap integral of 0.8 or above is commonly desired in this application domain. The overlap integral is also tied to the physical concept of measuring the accuracy of magnetic field profile predictions. Therefore, it facilitates testing whether the predicted vectors are valid eigen-vectors from a physical standpoint.

2. Relative Eigen-equation Error: While the overlap integral measures the closeness of the predicted eigen-vector's profile to the ground-truth, it is also important to understand whether a prediction that does not match the ground truth is still a valid eigen-solution or not. For this purpose, the relative eigen-equation error is used. This metric is defined as:

$$\frac{||\hat{A}\hat{\boldsymbol{y}} - \hat{b}\hat{\boldsymbol{y}}||}{||\hat{A}\hat{\boldsymbol{y}}||} \tag{5.8}$$

   This unitless quantity measures the error in the eigen-equation relative to its left-hand side. A value close to zero is desired, indicating that the prediction is a valid eigen-vector. Note that a value of 0 can be obtained even if the prediction is not the desired eigen-vector, in which case the overlap integral will also be equal to 0 because any two eigen-vector are orthogonality.

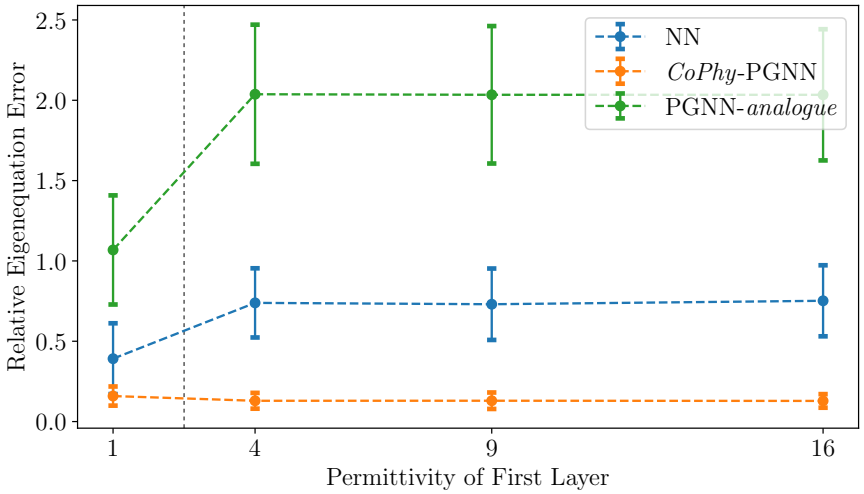


Figure 5.1: The relative eigen-equation error of *CoPhy*-PGNN compared to baseline models

To assess the extrapolation power of *CoPhy*-PGNN on out-of-distribution test samples, the relative eigen-equation error for comparative methods over the test samples is plotted in Figure 5.1. Note that the labeled set used for training only included samples with $\epsilon = 1$ (the leftmost point in the curves of Figure 5.1), while the test set included samples with $\epsilon \in \{1, 4, 9, 16\}$. One can see that both baselines, namely NN and PGNN-*analogue*, show low error at $\epsilon = 1$, but experience a significant jump in that error for test cases sampled with $\epsilon > 1$. This behavior is expected since both baselines were trained on labeled samples with ϵ only equal to 1. However, we can notice that in contrast to these baseline methods, *CoPhy*-PGNN shows the lowest relative error across all values of ϵ and does not show any apparent increase in relative error as we go beyond $\epsilon > 1$. This again demonstrates the ability of the proposed framework to generalize on unseen test distributions by making effective use of PG loss terms during ANN training.

5.2 The Uses and Challenges of Landscape Visualization in *KGML*

Model learning (a.k.a., model optimization or training) is the process by which model parameters are updated such that their final values, hopefully, generalize well. While most research compares ML methods by their final models in terms of accuracy on a hidden test set, such a comparison would not give the full picture of the advantages of one method over the other. For example, another important metric would be the speed of convergence.

In this section, I study how the *KGML* framework improves the optimization process using the tool of loss landscape visualization to understand the qualities of the model. Such an investigation not only allows me to 'explain' the advantages of using *KGML*, but also to make important observations and conclusions that help further refine the learning process. In this section, I discuss the shortcomings of the existing and commonly used loss landscape visualization methodology. This discussion is important to illustrate the need for an improved and more powerful visualization method that mitigates these shortcomings. Ultimately, I propose and discuss such a method, dubbed Neuro-Visualizer, in Section 5.3.1 and empirically demonstrate its advantages and effectiveness in Section 5.3.2.

5.2.1 Demonstrating *KGML*'s Improved Optimization Through Existing Loss Landscape Visualization Methods

Among the few studies that have employed landscape visualization within the *KGML* framework is Krishnapriyan et al. 's [84] where they have demonstrated the difficulty of training PINNs, and how changing a PDE's parameters plays an important role in the smoothness of the loss landscape, which in turn affects the ease at which the model can be optimized.

Similarly, Rohrhofer et al. [128] also used loss landscapes to emphasize the difficulty of training Physics-guided Neural Networks (PGNNs), and how different initial conditions of the PDE and their closeness to the PDE's fixed points affect the optimization process. They also show that using the physics loss alone leads to being stuck at bad minima.

In the following experiment, I visualize the landscape of different loss functions w.r.t. ANN model parameters to understand how the addition of PG losses improves the ANN's generalization performance for the Ising model problem within the application eigen-decomposition in physics (see Section 3.1 for more details). In Fig. 5.2, I plot a 2D view of the landscape of different loss functions, namely the mean-squared error on the training data (Train-MSE), the mean-squared error on the test data (Test-MSE), and PG-Loss (sum of C-Loss and S-Loss), in the neighborhood of a model solution. In each of the sub-figures of this plot, the model's parameters are treated with filter normalization as described in [91], 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 (1^{st}, 2^{nd} and 4^{th} columns) have found a minimum for Train-MSE. However, for Test-MSE, it is clear that only *CoPhy*-PGNN, unlike the other baselines, reaches a good minimum. 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. It is also clear 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.

5.2.2 Limitations of Existing Landscape Visualization Methods

As has been discussed in Section 2.2, the literature in recent years has slowly become more interested in using loss landscape visualization as a tool for investigating model optimization

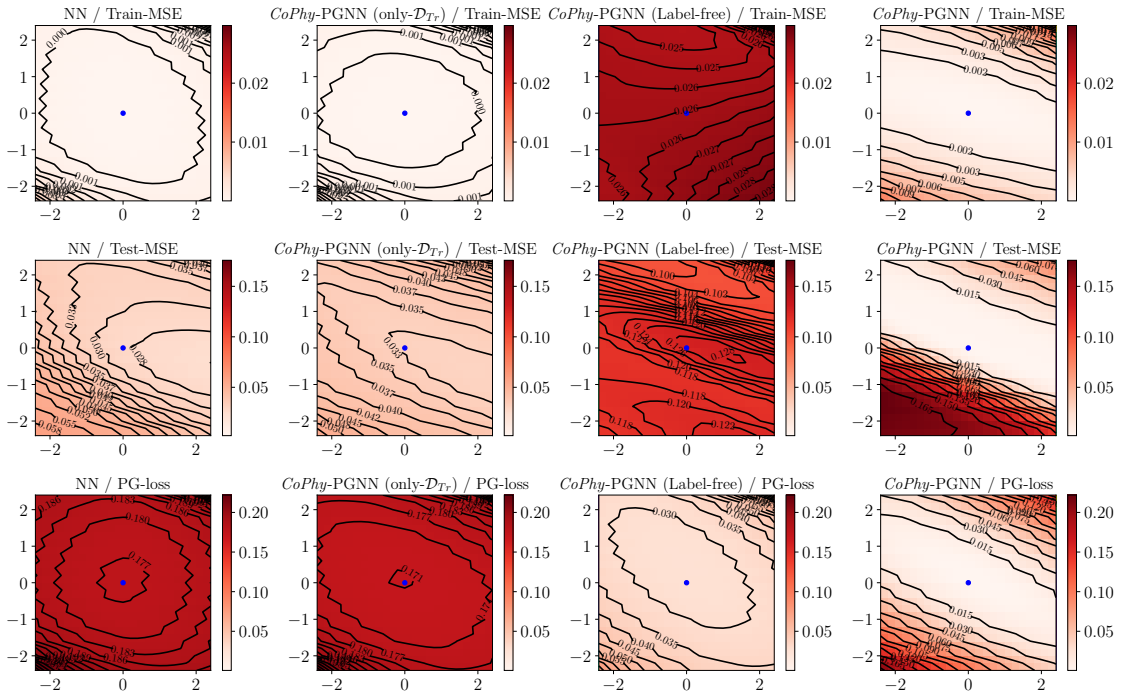


Figure 5.2: A comprehensive landscape comparison between *CoPhy*-PGNN and different baselines.

properties. The earliest methods and the ones most commonly used are linear in nature and involve heuristically choosing a planar 2D slice of the high dimensional parameter space to qualitatively study the loss surface and its properties. In the case of zooming onto a single model, which is generally the final model obtained through training, the most commonly used approach is to take a random plane defined by two orthogonal vectors in the high dimensional space [91]. This simple method was shown to be effective at qualitatively assessing the general properties of the loss landscape at the vicinity of the model, and that it gives a better perspective than, for example, taking the plane that passes through the two directions with maximum principal curvatures.

When it comes to visualizing multiple models (e.g., the set of models forming a training trajectory), however, the selection of the plane becomes more challenging as it is desired for that plane to show the "interesting" properties of the space for the entire set of points, 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 points, selecting the two main principal components, and visualizing the landscape on that plane such that it passes through the end of the trajectory [91]. Alternatively, other approaches include adopting the plane defined by the two eigenvectors with the highest eigenvalues [19, 176]. Each of these practices have their advantages and disadvantages.

Despite how helpful this linear projection approach can be, it has a number of limitations:

1. One problem with visualizing the model training trajectory using linear methods is that the loss surface 2D projection is mostly accurate at one point, whereas its fidelity drops drastically as we move away from that point. This is because the trajectory is highly unlikely to lie in a 2D plane (e.g., projecting the training trajectory on PCA's two major components). As such, a better method for visualizing the loss surface that contains such trajectory is worth investigating. Thus, researching the question of whether **a non-linear projection method that captures the trajectory and its surrounding landscape authentically, can be devised** is of utmost importance to mitigate this issue, leading to an improvement in model optimization understanding. And while there has been some body of work that uses certain non-linear projection methods, such as t-SNE and UMAP [65] (see Section 2.2 for more details), these methods are more suitable for plotting a "clustering" of the trajectory models and are unsuitable for grid plotting.

2. Another issue with landscape visualization is the scale at which the loss surface is visualized. While the most common practice is to use 'filter normalization' [91] to make the visualization scale invariant, this approach only works when visualizing the loss landscape near a single model, and is not applicable when visualizing training trajec-

tories. Thus, I address the question of whether it is possible to **devise a landscape visualization method that can flexibly adjust the scale across the projection space**. Answering this question is important because when the resolution is too low or the area is too narrow, there might be some critical points (e.g., minima) that are missed. Vice versa, when the number of calculated grid points is high, the process becomes computationally expensive.

5.3 Neuro-Visualizer: A New Non-linear Loss Landscape Visualization Method

As I consider the drawbacks of a linear loss landscape visualization method as described in Section 5.2.2, I can only echo Shire et al.'s [134] 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".

Instead of using PCA or non-linear clustering methods, such as t-SNE, I propose using a neural auto-encoder, dubbed Neuro-Visualizer, to learn a non-linear manifold that embeds the points of interest in the loss landscape.

5.3.1 Method Formulation

More formally, 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 visualize this trajectory on a 2D 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 $\mathcal{G} = [-1, +1] \times [-1, +1]$. Here, we are interested in finding a

manifold $\mathcal{L}$ that contains $\mathcal{M}_{\mathcal{T}}$ and mapping it onto the grid $\mathcal{G}$. This task naturally poses itself to be solved by auto-encodes, which generally take a high-dimensional input (e.g., $\mathbb{R}^n$), and map it onto a low dimensional one (e.g., $\mathbb{R}^2$). In the process of training an auto-encoder, the training data is fit into a learned low-dimensional manifold of the dimensionality of the latent space [10]. Mathematically, I propose learning a Neuro-Visualizer auto-encoder $\mathcal{N} : \mathcal{L} \rightarrow \mathcal{G}$ 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 $m' = D_{\mathcal{N}}(z \in \mathcal{G})$. To guarantee that $z \in [-1, +1] \times [-1, +1]$, a *tanh* activation function is appended to the output of the encoder $E_{\mathcal{N}}$. Also, $m \in \mathcal{L}$ is normalized for an easier training of $\mathcal{N}$.

To train the parameters of the model $\theta_{\mathcal{N}}$ such that the reconstructed model also lies on the same manifold, a reconstruction loss is minimized:

$$\mathrm{L}_{rec} = \mathrm{MSE}_{\mathcal{M}_{\mathcal{T}}} \left[m_i, \mathcal{N}(m_i) \right] \tag{5.9}$$

As $\mathcal{N}$ gets optimized, it learns to map a manifold that contains the training data points (e.g., the trajectory points $\mathcal{M}_{\mathcal{T}}$) onto grid $\mathcal{G}$.

While this loss is sufficient to guarantee having the trajectory models embedded into the learned manifold, it does not guarantee any other properties of the manifold outside those point other than continuity. However, this turns out to be a feature, not a bug. That is so because additional desired properties of the embedding space can be imposed using addition constraints in the form of other 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. Moreover, additional constraints need not be limited to the data-points lying on the optimization trajectory. Rather, more inventive and useful constraints can be crafted that are applied to other points in the input model parameter

space $\mathbb{R}^n$ or even a finite set of points in the latent grid space $\mathcal{M}_\mathcal{G} \subset \mathcal{G}$. Next, I list some interesting and useful constraints that I will demonstrate in Section 5.3.2. It is important, however, to note first that the following proposed constraints are just examples, and that Neuro-Visualizer is flexible and could be customized with many other possible constraints.

Location anchoring constraints The first and simplest form of constraints is anchoring (or pinning) some or all the points of the model trajectory onto certain locations on the grid. This serves as a means for orienting the trajectory to focus on certain aspects of the optimization process . One example would be to pin the last (i.e., converged) model onto the center of the grid to focus on visualizing the properties of the loss landscape of said converged model. The general form of a location anchoring constraint is:

$$\mathrm{L}_{anch} = \mathrm{MSE}_{\mathcal{M}_\mathcal{T}{}' \subseteq \mathcal{M}_\mathcal{T}} \left[E_\mathcal{N}(m_i), \mathcal{A}_i \right] \tag{5.10}$$

where $\mathcal{A} \subset \mathcal{G}$ is the set of desired anchoring points on the grid, and $\mathcal{M}_\mathcal{T}{}'$ is the subset of trajectory models that correspond to those anchoring points.

In this work, I choose to use two different anchoring constraints. The first is L_{anch_1} with $\mathcal{M}_\mathcal{T}{}' = \{m_0, m_{|\mathcal{M}_\mathcal{T}|}\}$ (i.e., the set that contains the first and last models in the trajectory). The corresponding set of anchoring points is $\mathcal{A} = \{(-r, -r), (r, r)\}$. The intention behind such selection of $\mathcal{A}$ is to place the first and last models at the bottom left and top right corners, respectively, with an $r = 0.8$, away from the center of the grid . This helps stretch the trajectory across the grid, maximizing utilized grid space. The second is L_{anch_2} with $\mathcal{M}_\mathcal{T}{}' = \{m_{|\mathcal{M}_\mathcal{T}|}\}$ and $\mathcal{A} = \{(0, 0)\}$. This constraint positions the last model at the center of the grid, with a perspective suitable for showing the final stage of the optimization process in more details.

Trajectory spacing constraints In addition to pinning certain trajectory models to certain locations on the grid using L_{anch}, we might be interested in spacing the trajectory models equally spaced such that all optimization steps have the same resolution. For that, I construct a trajectory spacing constraint, L_{traj}, such that:

$$\mathrm{L}_{traj} = \mathrm{MSE}_{\mathcal{M}_\mathcal{T}} \left[\left(E_\mathcal{N}(m_i) - E_\mathcal{N}(m_{i+1}) \right)^2, D^2 \right] \tag{5.11}$$

where D is a hyper-parameter that defines the appropriate grid distance between consecutive models. In this work, I define D adaptively such that $D = \frac{2\pi r}{|\mathcal{M}_\mathcal{T}|}$, where $r = 0.8$. This D is the ideal distance between consecutive models that could fit the trajectory on a circle of radius r. However, D can be set manually as desired for a shorter or a longer trajectory in the visualization.

Grid scaling constraints While the trajectory spacing loss, L_{traj}, is concerned with how the trajectory points spread, we may desire the grid itself to have a certain scale. However. unlike the case in PCA, Neuro-Visualizer's grid does not generally have a uniform and linear scale. This property allows Neuro-Visualizer to show more details at certain areas of the grid, while zooming out at other ones. Further more, a constraint can be constructed such that this zooming behavior is controlled.

In this work, I propose scaling the grid such that the vicinity of trajectory points has a relatively high grid density, and such that the grid zooms out for areas away from these trajectory points. To formalize this, I propose a grid scaling loss:

$$\mathrm{L}_{grid} = \mathrm{MSE}_{m \in \mathcal{M}_\mathcal{G}} \left[\frac{\log\left(d_m\right)}{l_m}, \frac{\log\left(d^{max}\right)}{L} \right] \tag{5.12}$$

To unpack this loss function, I first define the following terms: d_m is the distance between

grid 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. l_m is the distance equivalent to d_m in the grid space. Finally, L is a hyper-parameter chosen based on the desired zooming factor.

What minimizing L_{grid} does is enforcing a constant logarithmic scale $\frac{\log(d^{max})}{L}$ on the grid with respect to the trajectory models. This ratio defines how much parameter distance is covered for a specific grid distance. Thanks to the logarithmic scale, the smaller L is, the more quickly the scale changes as we move away from the trajectory. Thus, if zooming in is desired around trajectory points, an appropriately large L is chosen.

These different constraints can be combined in a multi-task learning fashion by weight-adding them. The weights in this work are constants and determined by hyper-parameter tuning. However, dynamic-weighting methods could potentially be used. In the next section, I apply combinations of these optional constraints to demonstrate their usefulness.

A word on the "correctness" of a loss landscape visualization method It is worth emphasizing that 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. Instead, one could approach the value of this tool through taking a more familiar example: Map projections. Earth map projections are a way of representing the three-dimensional surface of the Earth on a two-dimensional plane. Each projection method has its own advantages and limitations. For example, each method makes certain features or characteristics of the Earth more or less prominent. Thus, there are many valid projections, each useful for different purposes. One popular projection is the Mercator projection [141]. The Mercator projection is useful for navigation because it preserves angles and directions, but it distorts

the size and shape of objects near the poles. Another projection is the Robinson projection [140], which balances distortion of size and shape, making it a good all-purpose projection. However, it still distorts the sizes and shapes of landmasses, especially those near the poles. A third projection is the Winkel tripel projection as a compromise between the distortions of size and shape [141]. It accurately shows the relative sizes and shapes of landmasses, but it still distorts the shapes of some landmasses and oceans. Similarly, a non-linear loss landscape visualization will produce a non-linear manifold that gets distorted when visualized on a 2D planar grid. The user should be aware of these distortions and understand the visualization accordingly. Thus, despite these distortions, loss landscape visualization is still a valid and useful tool for understanding the properties of the loss surface. Another point to consider is that, for a set of points in the model parameter space, there are infinitely many manifolds that could contain these points. As such, it becomes all more important to select the manifold that exhibits the desired user-defined criteria, such as the scaling at different regions of the manifold.

5.3.2 Applications and Results

In this section, I use several applications in physics to demonstrate the power of Neuro-Visualizer and its advantages compared to the existing loss landscape visualization methodology.

Comparison of Neuro-Visualizer against PCA First, to show the drastic improvement in loss landscape quality when using Neuro-Visualizer, I take *CoPhy*-PGNN in quantum mechanics as a use-case application. Figures. 5.3 and 5.4 show and compare the PCA and Neuro-Visualizer Train-MSE loss landscapes for *CoPhy*-PGNN using different metrics. Note that the contours in all sub-figures have been scaled equally for fair comparison.

We make several observations in these figures:

1. **Neuro-Visualizer shows richer details:** In Fig. 5.3a, we notice that PCA shows quite the linear story where the model took a detour, due to optimizing multiple loss terms, and then descended into a terminal minima. The story of Neuro-Visualizer, however, is much richer. In Fig. 5.3b, while the detour is still clear, we can see that the "terminal" minima is not truly a simple minima, but rather a basin with a complex surface and many minima where the model bounces around during the final stages of its optimization. This observation becomes even clearer as we zoom into the convergence area using both methods (see bottom row). Being able to observe such complexity of the loss landscape is crucial for determining how well designed the model, the optimized loss terms, and the *KGML* method are.

2. **Neuro-Visualizer exhibits better consistency in loss values between the learned manifold and projected trajectories:** In Fig. 5.3, looking at the trajectory point color, which corresponds to the model loss, we can see that the model loss in the PCA plot does not always match that of the corresponding grid location (i.e., the points do not match in color with the contours at their respective locations). This is especially the case in the early stages of optimization. The reason underlying this phenomenon is that PCA is a linear method that only intersects with the trajectory at one point, which is the last model in the optimization trajectory by choice. This is unlike Neuro-Visualizer where most of the points match in color with their background because the learned 2D manifold approximately passes through all the trajectory points. This observation is further emphasized by looking at the first row of Fig. 5.4 where the absolute loss error (i.e., the error between model loss and the loss of its projection on the grid) is shown. In this figure, it is clear that the PCA trajectory has high error in terms of loss values. This is in contrast to the low error values in the

Method	Average relative loss error	Average projection error in the parameter space
Neuro-Visualizer	0.029	0.003
PCA	2.516	0.283

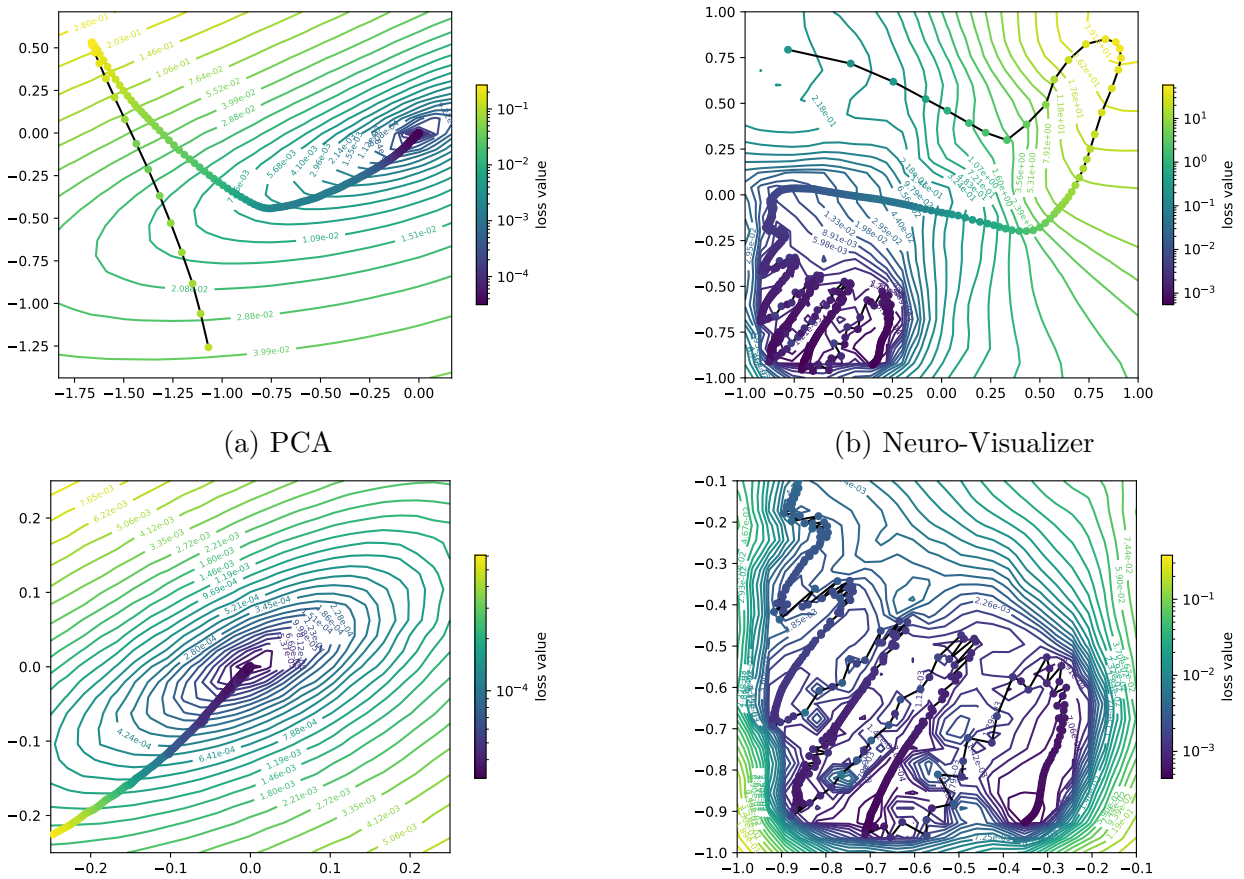
Table 5.1: A quantitative comparison of PCA and Neuro-Visualizer by calculating *CoPhy*-PGNN's Train-MSE errors between the trajectory and the grid in terms of loss values and distances in the parameter space.

Neuro-Visualizer's case.

3. **The manifold learned by Neuro-Visualizer fits trajectories better:** Looking at the second row in Fig. 5.4, where the points are color-coded as the distance between the manifold and the trajectory points in the parameter space, one can see how trajectory models in the PCA view have "hot" colors, which means that those models are significantly far from the grid's plane. This is in contrast to the cool colors in Neuro-Visualizer's view since its manifold fits the trajectory well.

To further quantitatively assess Neuro-Visualizer's advantages, In Tab. 5.1, I provide the average relative error in loss value between all models in the trajectory and their corresponding projections on the grid. Obviously, the smaller the error, the more consistent the plot is. Additionally, I provide the average distance in parameter space between trajectory models and their projections. Similarly, the smaller the distance, the more accurately the model lies on the grid. As can be seen, Neuro-Visualizer beats PCA by orders of magnitude, proving its supremacy.

To drive this point home on a larger scale, I plot the trajectories of *CoPhy*-PGNN and its baseline Black-box NN. Both trajectories start from the same initialization (marked with a thick border) and are displayed on the same landscape. Such a plot allows for an easy comparison between the two models in terms of optimization. Fig. 5.5 shows two columns, the left one is the PCA landscape visualizations of different losses, namely Test-MSE, S-Loss and C-Loss from top to bottom. The right column shows the corresponding landscapes for



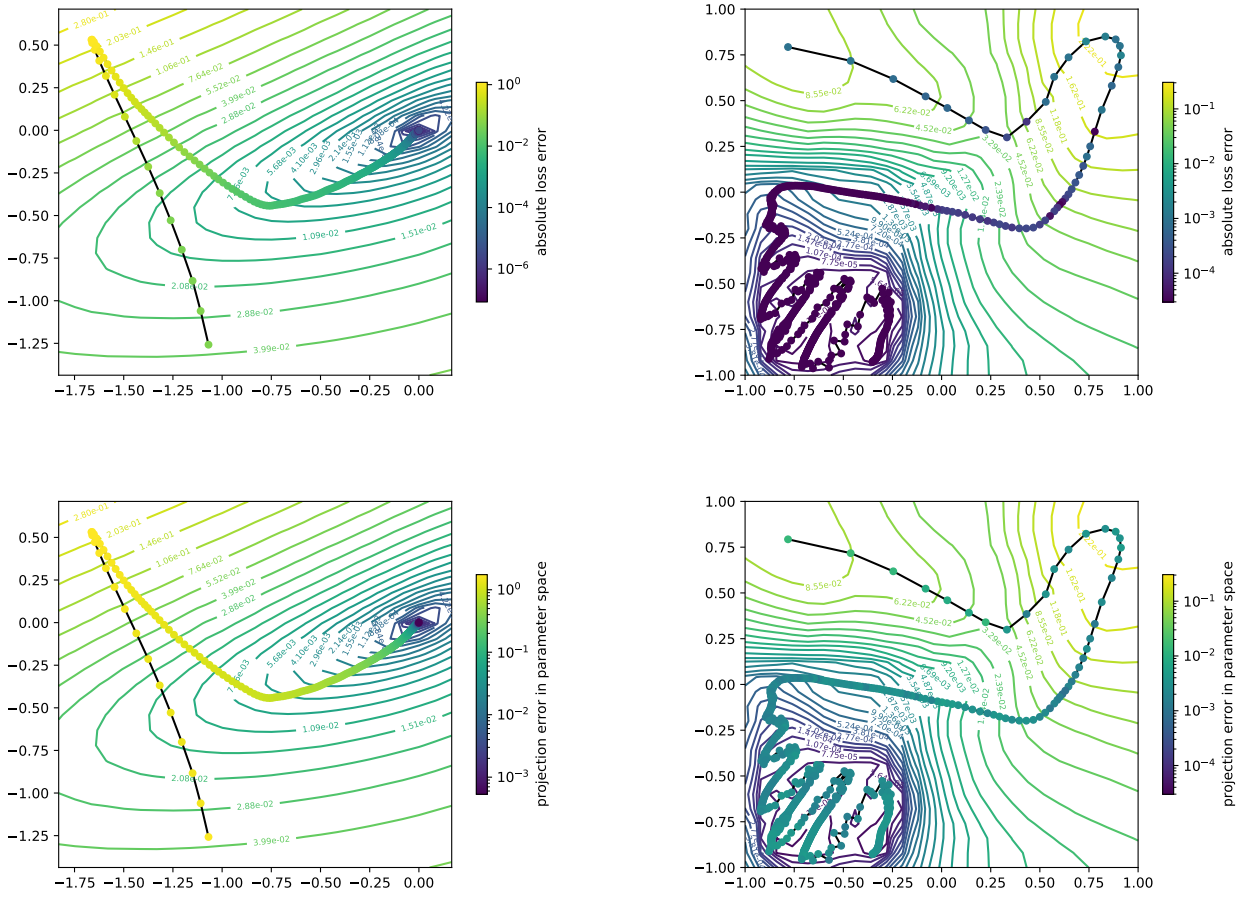
(a) PCA

(b) Neuro-Visualizer

(c) PCA

(d) Neuro-Visualizer

Figure 5.3: A comparison of PCA and Neuro-Visualizer in terms of consistency w.r.t *CoPhy*-PGNN's Train-MSE loss values between the trajectory and the grid. The bottom row is a zoomed-in view of the top row.



(a) PCA (b) Neuro-Visualizer

Figure 5.4: A comparison of PCA and Neuro-Visualizer, showing *CoPhy*-PGNN's Train-MSE landscape visualization in terms of errors between the trajectory and the grid in loss values and distances in the parameter space.

Neuro-Visualizer. Here are some observations regarding the advantages of Neuro-Visualizer over PCA in a multi-trajectory setting:

1. **Neuro-Visualizer successfully captures all minima of interest:** Since its plane passes through the last model of *CoPhy*-PGNN's trajectory, PCA fails completely at capturing Black-box NN's minima for most of the losses. This is especially obvious in Fig. 5.5c where not only Black-box NN's minima is not present, but also the loss values of Black-box NN's trajectory and the corresponding grid locations are inconsistent. This is in contrast to Neuro-Visualizer, where both Black-box NN's and *CoPhy*-PGNN's minima are distinctly visualized and the consistency between the trajectory and the grid is not violated.

2. **Neuro-Visualizer successfully captures gradient descent:** PCA may not clearly show whether a model terminally reaches a minima. For example, Fig. 5.5a gives the false impression that Black-box NN ends up somewhere in a concavity but does not follow its gradient towards the minima at the final epochs of the optimization process, even though it is not influenced by any loss terms other than the mean-square-error. This contradicts the notion that optimizers such as Stochastic Gradient Descent (SGD) or ADAM roughly follow the direction of the gradient. On the other hand, the trajectory of Black-box NN in Fig. 5.5b using Neuro-Visualizer clearly shows that Black-box NN "descends" towards a plateau, where the gradient is weak. This visualization fits with the common conception about gradient-descent-based optimization.

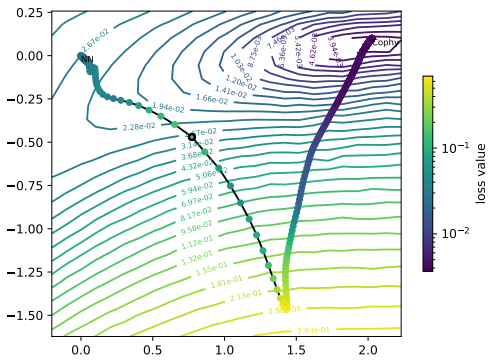
3. **Neuro-Visualizer avoids telling inconsistent and contradictory stories:** Looking at Black-box NN's trajectory in Fig. 5.5a, PCA shows that it crosses the contour 0.0228 twice, implying that the trajectory value descends and then ascends again. This, however, is false because looking at the trajectory loss values, they consistently

decrease as the final model is approached. Thus, there is an inconsistency in PCA's visualization that leads to confusion. This inconsistency, however, is not present in Neuro-Visualizer's visualization in Fig. 5.5b. Similarly, looking at Black-box NN's trajectory in Fig. 5.5c, PCA shows that while the trajectory loss values decrease over late epochs, the corresponding grid loss values increase. This inconsistency is confusing and undrmines the tool's usefulness. On the other hand, Neuro-Visualizer in Fig. 5.5d does not exhibit such an inconsistency, making it more effective and useful than PCA.

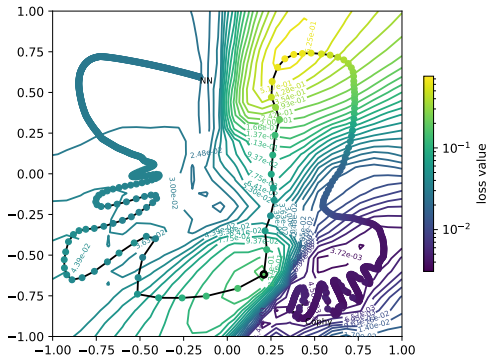
Investigating Different Loss Balancing Techniques in *KGML* Using Neuro-Visualizer

As has been clear throughout this chapter, balancing the different losses in the *KGML* framework can be a daunting task. Thus, the ability to compare different loss balancing techniques in terms of how well they harness the synergy of the different losses is considered an important research effort. Generally, different loss balancing techniques are compared in terms of the model's final accuracy. However, this criterion only shows one side of the story, and does not convey whether there is a fundamental difference among the different balancing methods. For example, it is desired to know whether a certain loss balancing algorithm struggles with one type of problem or architecture while being suitable for another. Also, while some insight could be gained by looking at the loss per epoch plots, such a plot only shows the evolution of a single model instance, without necessarily describing the loss landscape more generally.

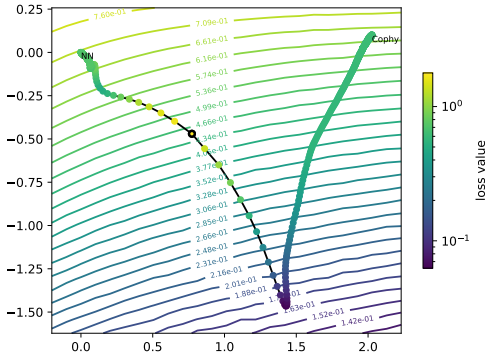
Expanding on the previous experiment to investigate the suitability of an *KGML* framework and comparing it to other baselines, I consider six different loss balancing methods that are commonly used in the literature. Three of these methods are selected from the multi-task learning (MTL) literature. In multi-task learning, an algorithm is devised such that the coefficient of each loss is adaptively updated each epoch based on a certain criterion, such



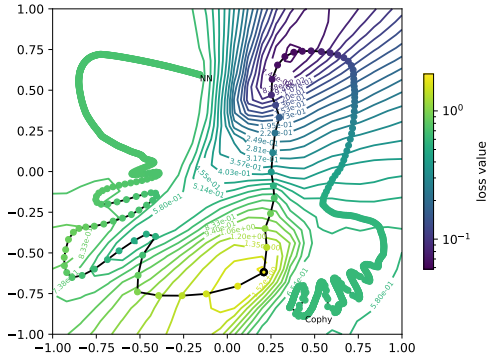
(a) PCA - Test-MSE



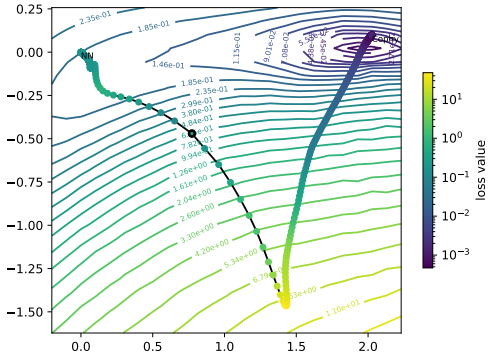
(b) Neuro-Visualizer - Test-MSE



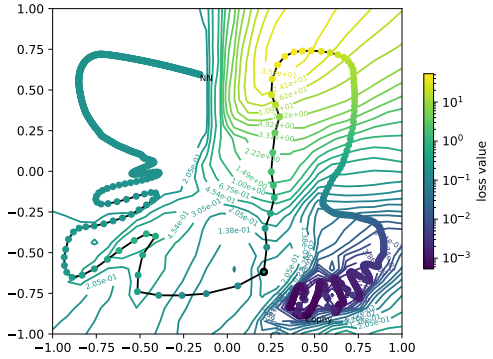
(c) PCA - S−*Loss*



(d) Neuro-Visualizer - S−*Loss*



(e) PCA - C−*Loss*



(f) Neuro-Visualizer - C−*Loss*

Figure 5.5: The loss landscapes of *CoPhy*-PGNN and Black-box NN for different loss terms using PCA (left) and Neuro-Visualizer (right)

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: $c_{residual} = 1.0, c_{ic} = c_{bc} = 100.0$.
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 [166]	LRannealing	Gradient statistics are utilized during model training to balance the interplay between the losses.
Gradient Normalization [22]	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 Weighing [95]	RLW	The weight of each task is randomly updated each epoch.

Table 5.2: A list of the loss balancing algorithms considered in this section. More details on each algorithm can be found in [94]

as the gradient of the losses. Additionally, I consider a loss balancing algorithm that is more specific to PINNs. Namely, the Learning Rate Annealing method [166]. Finally, I also include two other naive baseline methods that do not involve dynamic coefficient updates. Tab. 5.2 gives a detailed account of the algorithms used in this section.

To compare these algorithms, I train different PINNs to solve the Convection problem with $\beta = 10$ (see Section 3.3 for more details) using the different loss balancing algorithms listed in Tab. 5.2. The loss terms that interplay in this framework are the residual loss $\mathrm{L}_{residual}$, the initial condition loss L_{ic}, and the boundary condition loss L_{bc}. The total loss being optimize is:

$$\mathrm{L}_{total} = c_{ic} \times \mathrm{L}_{residual} + c_{ic} \times \mathrm{L}_{ic} + c_{bc} \times \mathrm{L}_{bc}. \tag{5.13}$$

Starting from the same initial model, I train six different instances of the model using the different loss balancing methods. In Fig. 5.6, I inspect the landscapes of two loss terms:

$L_{residual}$ and L_{test} (i.e., the prediction error at test domain points). For each loss term, two loss landscape visualizations are produced using Neuro-Visualizer and PCA.

Inspecting Fig. 5.6, we can make the following observations:

1. **Neuro-Visualizer detects model convergence more clearly:** Looking at Figs. 5.6d, it is easy to see, using Neuro-Visualizer, that all loss balancing methods, except for LRannealing, converge to some minima. This is in contrast to PCA's landscape in Fig. 5.6c where it is unclear whether several trajectories (e.g., RLW, GradNorm, and DWA) converge at all because the grid does not show any minima where they terminate. Thus, Neuro-Visualizer is evidently more intuitive and useful for a qualitative assessment of these different methods.

2. **Neuro-Visualizer is more capable of distinguishing among different minima:** Looking at the residual loss (top row), it is unclear in the PCA plot (left) whether some of the loss balancing methods converge to the same minima, several minima in the same basin, or different basins altogether. This is due to the limited view and the unfavorable scale PCA's linear projection provides. In contrast, the loss landscape plotted using Neuro-Visualizer shows a much more holistic view of the loss balancing methods and how their trajectories compare. In particular, it clearly shows that while GradNorm and EW take slightly different trajectories, they converge to the same minima. Additionally, Neuro-Visualizer shows that DWA converges to the same basin as GradNorm and EW, but not the same minima. This is in contrast to PCA's view where all three models are lumped together. This analysis would not be possible without the unique non-linear manifold Neuro-Visualizer learns, which allows for different scaling factors at different parts of the space.

3. **Neuro-Visualizer helps discover novel insights:** Looking at 5.6d, it is clear that

Method	Average absolute loss error	Average projection error in the parameter space
Neuro-Visualizer	0.147	1.583
PCA	0.923	12.801

Table 5.3: A quantitative comparison of PCA and Neuro-Visualizer by calculating L_{test} errors between the trajectory and the grid in terms of loss values and distances in the parameter space.

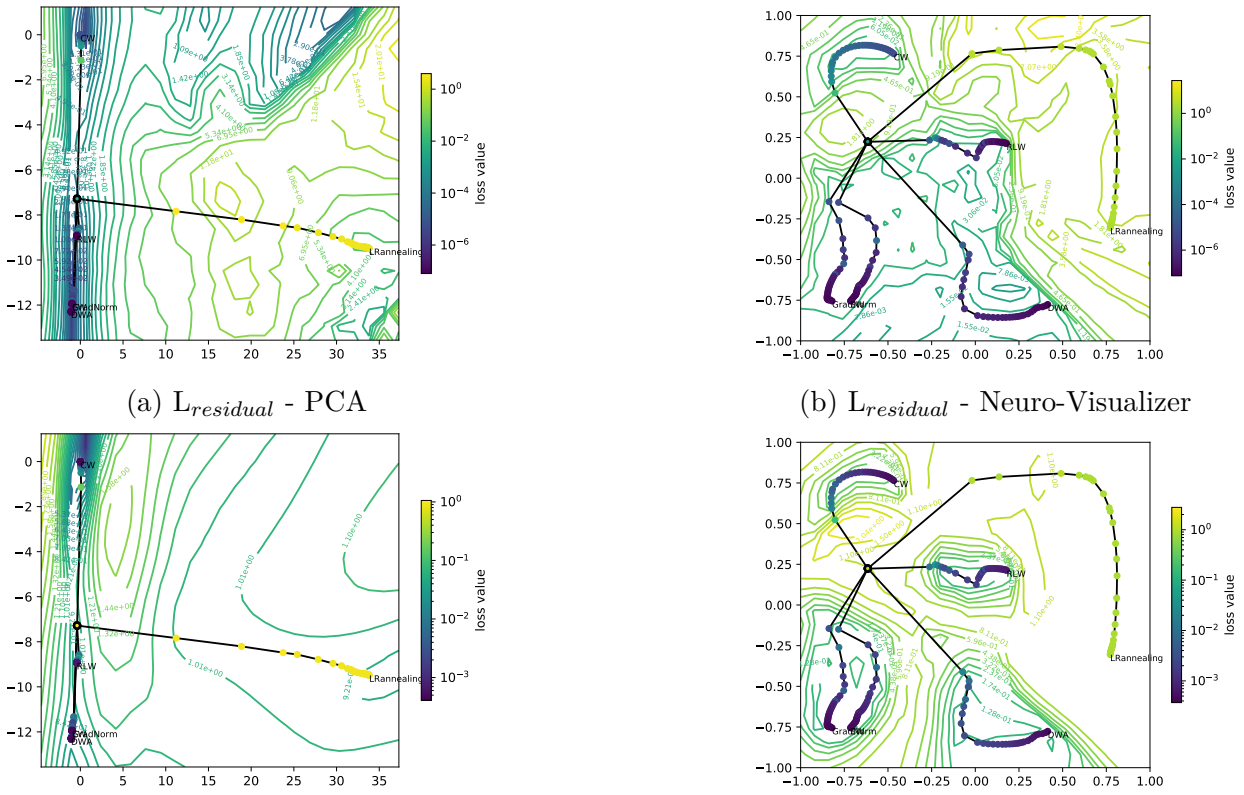
LRannealing's trajectory is nowhere near a minima w.r.t L_{test}. This goes against the initial expectation that such a loss balancing method specifically designed for PINNs should perform better than the other methods. Thus, while LRannealing shows great success for solving the Helmholtz equation and the Klein Gordon equation as demonstrated in [167], it seems to fail at achieving good optimization for the Convection problem. Thanks to Neuro-Visualizer, it is easy to make this conclusion visually.

It is also worth noting that while Neuro-Visualizer's manifold does exhibit some fitting error in Fig. 5.6, as evident from the difference in color between trajectory models and their corresponding grid background colors, it still fits the trajectories significantly better than that of PCAs. Tab. 5.3 demonstrates that quantitatively.

Investigating The Different Constraints within Neuro-Visualizer's Framework

Next, I take the PINN application further to study the effects of applying the different constraints discussed in Section 5.3. Here, I use the same PINN to solve the Convection problem but with $\beta = 30$, thus making the PDE harder to solve. This should lead to a more non-convex loss landscape that is harder to optimize, making it an interesting candidate for landscape visualizations.

Fig. 5.7 shows a progression of different Neuro-Visualizers visualizing L_{test}. The first sub-figure is the visualization with no constraints other than L_{rec}. Then, in each consecutive sub-figure, I add an additional constraint, or a combination of constraint and comment on



(a) $L_{residual}$ - PCA

(b) $L_{residual}$ - Neuro-Visualizer

(c) L_{test} - PCA

(d) L_{test} - Neuro-Visualizer

Figure 5.6: Plotting the loss landscapes of multiple Convection PDE solving models using PCA (left) and Neuro-Visualizer (right)

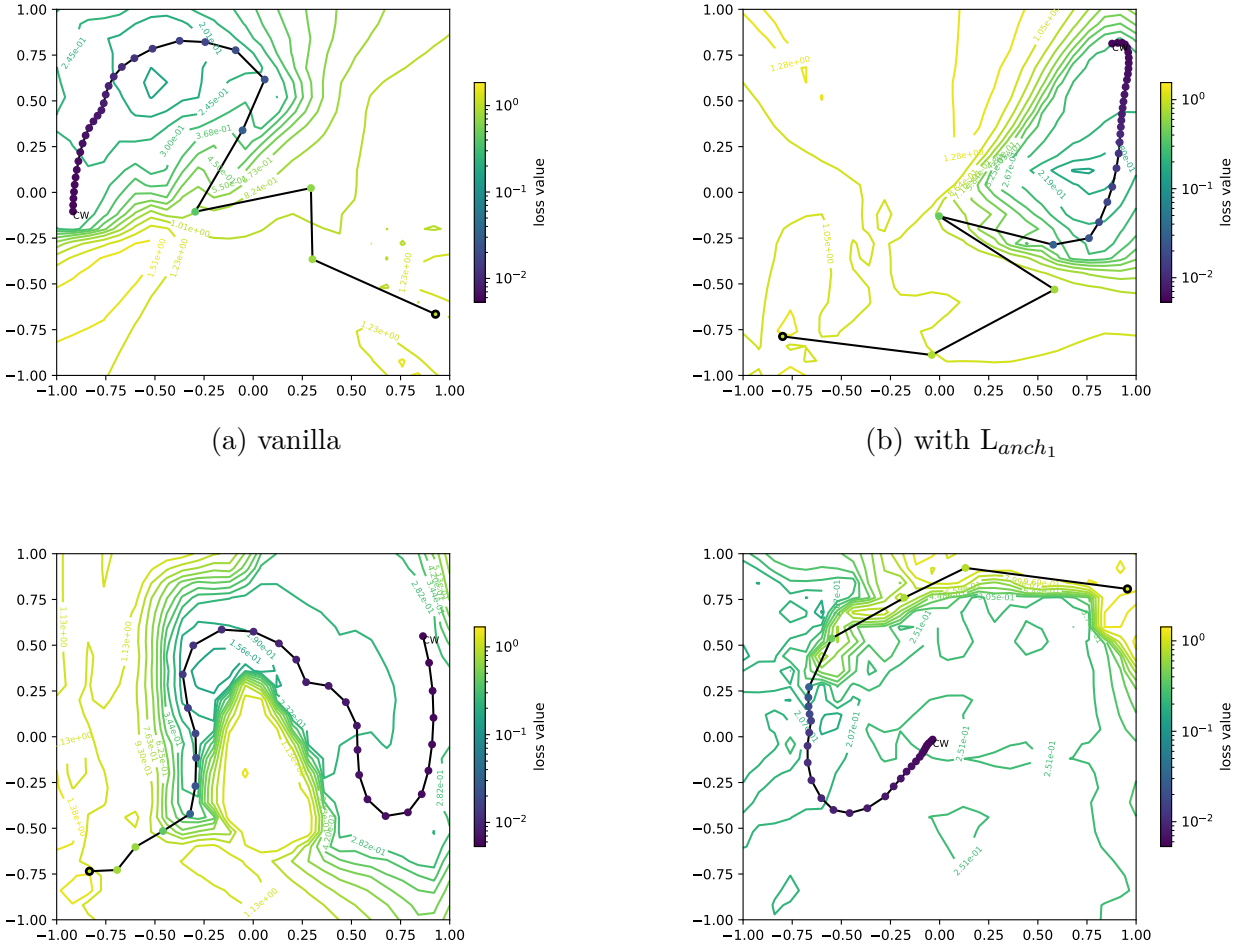
the resulting landscape visualization.

After the "vanilla" landscape shown in Fig. 5.7a, Fig. 5.7b shows the same trajectory constrained with L_{anch_1}. Notice that the constraint manifests almost perfectly and allows the trajectory to "stretch". However, notice that the steps on the trajectory are not equidistant in the grid space. Next, in Fig. 5.7c, another constraint is added, namely L_{traj}. Now, the trajectory both stretches between the two corners and has equidistant steps in the grid space. Finally, to show the effect of L_{grid}, I combine it with L_{anch_2}. A large $L = 8$ is chosen to impose high grid density around the trajectory's models. Fig. 5.7d shows how the grid space around the last stage of model optimization (i.e., near the grid center) is expanded with much more details than the previous visualizations.

It is worth studying the effect of L_{grid} in more details. In particular, Fig. 5.8 demonstrates the effect of changing L, the hyper-parameter that correlates with the grid density near trajectory models. Each sub-figure shows the *density* landscape at a different value of L where the color-coding indicates grid density. More precisely, at each grid point, a CKA-similarity-based density is calculated using the formula:

$$\rho_{m \in \mathcal{M}_{\mathcal{G}}} = \sum_{m' \in \mathcal{M}_{\mathcal{G}} \setminus \{m\}} \mathrm{CKA}(m', m) \tag{5.14}$$

where $\mathrm{CKA}(m', m)$ is the CKA similarity measure between two neural networks as defined in [83]. Thus, the more similar a grid point is to the other grid points in the parameter space, the higher the density is at that point in the grid space. As can be seen in Fig. 5.8, the density of the grid, especially near the trajectory points, increases as L increases. Also, while the density is generally highest near the trajectory models, it decreases for farther grid points. This demonstrates that L_{grid} can be a vital tool for engineering the manifold through the appropriate choice of L.



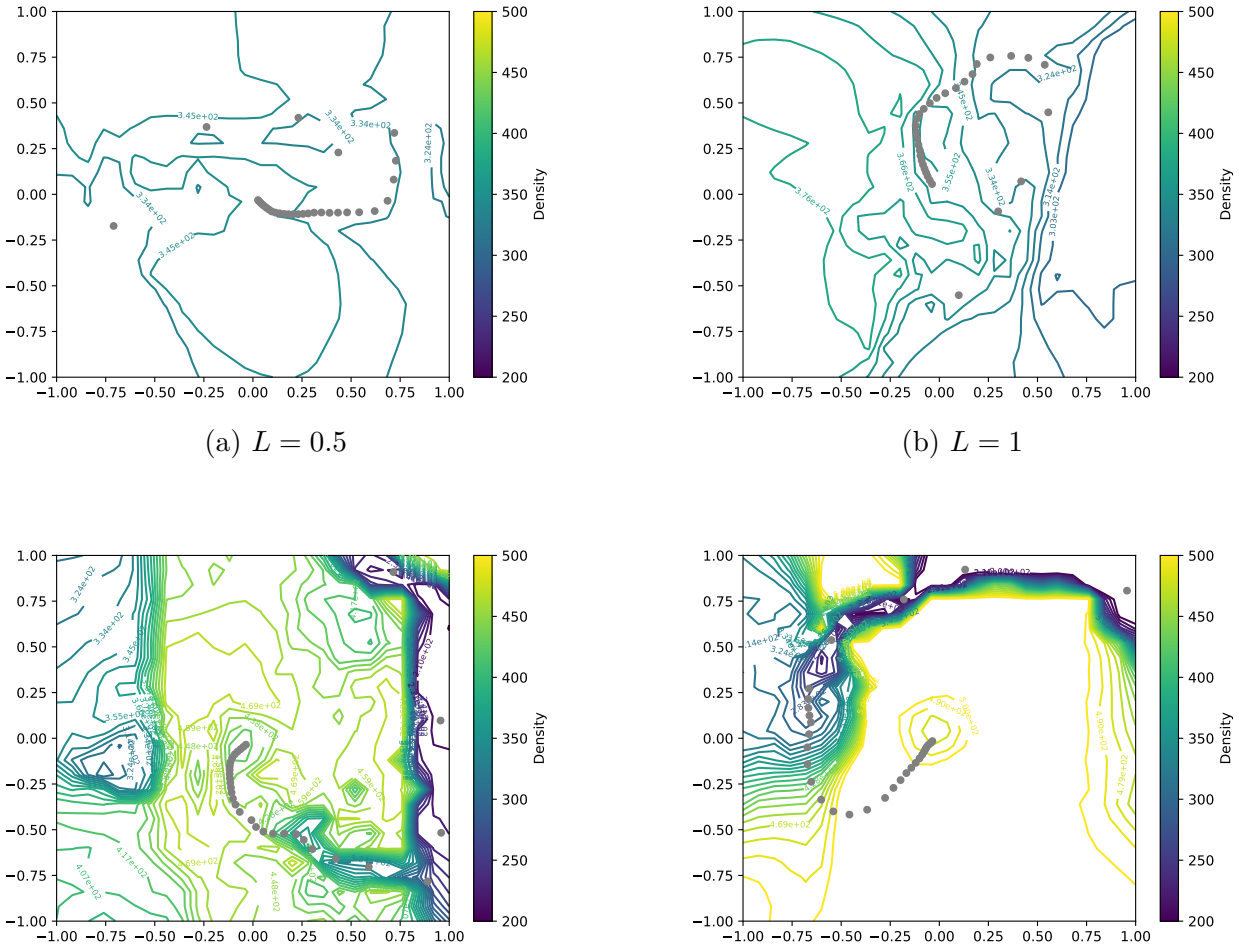
(a) vanilla

(b) with L_{anch_1}

(c) with L_{anch_1} and L_{traj}

(d) with L_{anch_2} and L_{grid}

Figure 5.7: A series of Neuro-Visualizer landscape visualizations of L_{test} with different constraints



(a) $L = 0.5$

(b) $L = 1$

(c) $L = 4$

(d) $L = 8$

Figure 5.8: A progression of Neuro-Visualizer *density* landscape visualizations with different values of the L hyper-parameter within the L_{grid} constraint. As can be seen, higher L values generally lead to higher grid densities.

Discussion of other potential applications While previous applications in *CoPhy*-PGNN and PINNs have shown how powerful Neuro-Visualizer can be, one of its limitations is its lack of scalability for large model parameter space. For example, the PINNs trained in this section have a number of model parameters of the order $10,000$, which is relatively small. In contrast, computer vision applications generally use much larger models. For example, a ResNet-18 [55] has 11.2 million parameters, and a VQ-GAN [39] can have as much as 88 million parameters. Such a large input space would be prohibitively difficult for Neuro-Visualizer to encode directly. As such, I propose to use network pruning strategies to reduce the number of parameters that need to be visualized, which can be further pursued in future work.

Chapter 6

Summary

In this thesis, I have explored the emerging field of Knowledge-Guided Machine Learning (*KGML*) and its advantages and limitations through both the lenses of output space and loss landscape visualization.

While investigating the effect of *KGML* on the output space, including the latent space, I have demonstrated, by means of an application in biology, how machine learning models could benefit from applying the *KGML* framework, especially when dealing with noisy and limited data. I have also shown that *KGML* not only helps improve model generalization, but can also engineer feature spaces with properties and constraints desired by domain experts.

Investigating the effect of *KGML* on the loss landscape, I have emphasized on the importance of developing visualization tools that aid machine learning researchers and practitioners at evaluating and diagnosing their models. By proposing a novel landscape visualization technique, I was able to demonstrate the usefulness and effectiveness of such a tool at comparing different model designs and optimization methods.

By targeting both goals, I hope I have provided novel insight and exciting prospects of the applicability and viability of the *KGML* framework, along with the tools to develop it further in future work.

Bibliography

[1] Michael A. Alcorn, Qi Li, Zhitao Gong, Chengfei Wang, Long Mai, Wei-Shinn Ku, and Anh Nguyen. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[2] Eugene L. Allgower and Kurt Georg. *Introduction to Numerical Continuation Methods*. Society for Industrial and Applied Mathematics, 2003.

[3] Vaneeda Allken, Nils Olav Handegard, Shale Rosen, Tiffanie Schreyeck, Thomas Mahiout, and Ketil Malde. Fish species identification using a convolutional neural network trained on synthetic data. *ICES Journal of Marine Science*, 76(1):342–349, jan 2019. ISSN 1054-3139. doi: 10.1093/icesjms/fsy147.

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

[5] Robert Arlinghaus and Steven J. Cooke. Recreational Fisheries: Socioeconomic Importance, Conservation Issues and Management Challenges. *Recreational Hunting, Conservation and Rural Livelihoods: Science and Practice*, pages 39–58, 2009. doi: 10.1002/9781444303179.ch3.

[6] Angela H. Arthington, Nicholas K. Dulvy, William Gladstone, and Ian J. Winfield. Fish conservation in freshwater and marine realms: status, threats and management. *Aquatic Conservation: Marine and Freshwater Ecosystems*, 26(5):838–857, 2016. ISSN 10990755. doi: 10.1002/aqc.2712.

[7] Brian B Avants. *ANTsR: ANTs in R: Quantification Tools for Biomedical Images*, 2019. R package version 0.5.4.2.

[8] Shamsulhaq Basir and Inanc Senocak. Critical investigation of failure modes in physics-informed neural networks. In *AIAA SCITECH 2022 Forum*. American Institute of Aeronautics and Astronautics, jan 2022. doi: 10.2514/6.2022-2353. URL https://doi.org/10.2514%2F6.2022-2353.

[9] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009.

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

[11] Ricardo Betancur-R, Edward O Wiley, Gloria Arratia, Arturo Acero, Nicolas Bailly, Masaki Miya, Guillaume Lecointre, and Guillermo Orti. Phylogenetic classification of bony fishes. *BMC evolutionary biology*, 17(1):162, 2017.

[12] 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.

[13] B. R. Bowen, B. R. Kreiser, P. F. Mickle, J. F. Schaefer, and S. B. Adams. Phylogenetic relationships among north american alosa species (clupeidae). *Journal of Fish Biology*, 72(5):1188–1201, 2008. doi: https://doi.org/10.1111/j.1095-8649.2007.01785.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1095-8649.2007.01785.x.

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

[15] Matthew A Campbell, Thaddaeus J Buser, Michael E Alfaro, and J Andrés López. Addressing incomplete lineage sorting and paralogy in the inference of uncertain salmonid phylogenetic relationships. *PeerJ*, 8:e9389, 2020.

[16] Fabrizio Carcillo, Yann-Aël Le Borgne, Olivier Caelen, Yacine Kessaci, Frédéric Oblé, and Gianluca Bontempi. Combining unsupervised and supervised learning in credit card fraud detection. *Information sciences*, 557:317–331, 2021.

[17] Jose Carranza-Rojas, Saul Calderon-Ramirez, Adán Mora-Fallas, Michael Granados-Menani, Jordina Torrents-Barrena, and Jose Carranza-Rojas. Unsharp masking layer: Injecting prior knowledge in convolutional networks for image classification. In *2019 International Conference on Artificial Neural Networks*, pages 3–16. Springer, Cham, September 2019.

[18] 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. ISBN 1558603077.

[19] Avraam Chatzimichailidis, Janis Keuper, Franz-Josef Pfreundt, and Nicolas R. Gauger. 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)*, pages 66–74, 2019. doi: 10.1109/MLHPC49564.2019.00012.

[20] Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: Deep learning for interpretable image recognition. In

H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/adf7ee2dcf142b0e11888e72b43fcb75-Paper.pdf.

[21] Lin Chen, Chuanju Dong, Shengnan Kong, Jiangfan Zhang, Xuejun Li, and Peng Xu. Genome wide identification, phylogeny, and expression of bone morphogenetic protein genes in tetraploidized common carp (cyprinus carpio). *Gene*, 627:157 – 163, 2017. ISSN 0378-1119. doi: https://doi.org/10.1016/j.gene.2017.06.020. URL http://www.sciencedirect.com/science/article/pii/S0378111917304705.

[22] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 794–803. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/chen18a.html.

[23] Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

[24] Julien Clavel, Gilles Escarguel, and Gildas Merceron. mvmorph: an r package for fitting multivariate evolutionary models to morphometric data. *Methods in Ecology and Evolution*, 6(11):1311–1319, 2015.

[25] SM Collins, S Yuan, PN Tan, SK Oliver, JF Lapierre, KS Cheruvelil, CE Fergus, NK Skaff, J Stachelek, Tyler Wagner, et al. Winter precipitation and summer temperature predict lake water quality at macroscales. *Water Resources Research*, 55(4): 2708–2721, 2019.

[26] Michael L. Collyer and Dean C. Adams. Phylogenetically aligned component analysis. *Methods in Ecology and Evolution*, 12(2):359–372, 2021. doi: https://doi.org/10.1111/2041-210X.13515. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13515.

[27] C. Costa, F. Antonucci, C. Boglione, P. Menesatti, M. Vandeputte, and B. Chatain. Automated sorting for size, sex and skeletal anomalies of cultured seabass using external shape analysis. *Aquacultural Engineering*, 52:58–64, jan 2013. ISSN 01448609. doi: 10.1016/j.aquaeng.2012.09.001.

[28] 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.

[29] 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.

[30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[31] Mona Diab. Data paucity and low resource scenarios: Challenges and opportunities. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 3612, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3409565. URL https://doi.org/10.1145/3394486.3409565.

[32] G. Ding, Y. Song, J. Guo, C. Feng, G. Li, B. He, and T. Yan. Fish recognition using convolutional neural network. In *OCEANS 2017 - Anchorage*, pages 1–4, 2017.

[33] Ellen Ditria, Michael Sievers, Sebastian Lopez-Marcano, Eric L. Jinks, and Rod M. Connolly. Deep learning for automated analysis of fish abundance: the benefits of training across multiple habitats. *bioRxiv*, 2020. doi: 10.1101/2020.05.19.105056. URL https://www.biorxiv.org/content/early/2020/05/22/2020.05.19.105056.

[34] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[35] Anderson Aparecido dos Santos and Wesley Nunes Gonçalves. Improving pantanal fish species recognition through taxonomic ranks in convolutional neural networks. *Ecological Informatics*, 53:100977, 2019. ISSN 1574-9541. doi: https://doi.org/10.1016/j.ecoinf.2019.100977. URL https://www.sciencedirect.com/science/article/pii/S1574954119301001.

[36] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[37] Mohannad Elhamod, Jie Bu, Christopher Singh, Matthew Redell, Abantika Ghosh, Viktor Podolskiy, Wei-Cheng Lee, and Anuj Karpatne. Cophy-pgnn: Learning physics-guided neural networks with competing loss functions for solving eigenvalue problems, 2020.

[38] Mohannad Elhamod, Kelly M. Diamond, A. Murat Maga, Yasin Bakis, Henry L. Bart Jr., Paula Mabee, Wasila Dahdul, Jeremy Leipzig, Jane Greenberg, Brian Avants, and Anuj Karpatne. Hierarchy-guided neural network for species classification. *Methods in Ecology and Evolution*, n/a(n/a), 2021. doi: https://doi.org/

10.1111/2041-210X.13768. URL https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13768.

[39] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[40] Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *nature*, 542(7639):115–118, 2017.

[41] David Glynne Fox and Thomas PV Hartman. Photographing fluid-preserved specimens. In *Biobanking*, pages 149–153. Springer, 2019.

[42] Han Gao, Luning Sun, and Jian-Xun Wang. 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.

[43] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8789–8798. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8095-loss-surfaces-mode-connectivity-and-fast-ensembling-of-dnns.pdf.

[44] Yolanda Gil and Bart Selman. A 20-Year Community Roadmap for Artificial Intelligence Research in the US. https://cra.org/ccc/wp-content/uploads/sites/2/2019/08/Community-Roadmap-for-AI-Research.pdf, 2019.

[45] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[46] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[48] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. Qualitatively characterizing neural network optimization problems, 2014.

[49] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2015.

[50] James M Grady. Phylogenetic relationships, modes of speciation, and historical biogeography of the madtom catfishes genus noturus rafinesque (siluriformes: Ictaluridae). *Systematics, historical ecology, and North American freshwater fishes*, 1993.

[51] 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.

[52] Terry Grande, Howard Laten, and J. Andrés López. Phylogenetic Relationships of Extant Esocid Species (Teleostei: Salmoniformes) Based on Morphological and Molecular Characters. *Copeia*, 2004(4):743 – 757, 2004. doi: 10.1643/CG-04-007R1. URL https://doi.org/10.1643/CG-04-007R1.

[53] Simon Guiroy, Vikas Verma, and Christopher Pal. Towards understanding generalization in gradient-based meta-learning, 2019.

[54] Phillip M. Harris, Kevin J. Roe, and Richard L. Mayden. A Mitochondrial DNA Perspective on the Molecular Systematics of the Sunfish Genus Lepomis (Actinopterygii: Centrarchidae). *Copeia*, 2005(2):340 – 346, 2005. doi: 10.1643/CG-04-035R1. URL https://doi.org/10.1643/CG-04-035R1.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[56] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[57] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022.

[58] Gene Helfman, Bruce B Collette, Douglas E Facey, and Brian W Bowen. *The diversity of fishes: biology, evolution, and ecology*. John Wiley & Sons, 2009.

[59] Lisa Anne Hendricks, Zeynep Akata, Marcus Rohrbach, Jeff Donahue, Bernt Schiele, and Trevor Darrell. Generating visual explanations. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 3–19, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[60] Willi Hennig. *Phylogenetic systematics*. 1966.

[61] Stefan Horoi, Jessie Huang, Bastian Rieck, Guillaume Lajoie, Guy Wolf, and Smita Krishnaswamy. Exploring the geometry and topology of neural network loss land-

scapes. In Tassadit Bouadi, Elisa Fromont, and Eyke Hüllermeier, editors, *Advances in Intelligent Data Analysis XX*, pages 171–184, Cham, 2022. Springer International Publishing. ISBN 978-3-031-01333-1.

[62] David Houle and Daniela M Rossoni. Complexity, evolvability, and the process of adaptation. *Annual Review of Ecology, Evolution, and Systematics*, 53, 2022.

[63] Tomas Hrbek, Jens Seckinger, and Axel Meyer. A phylogenetic and biogeographic perspective on the evolution of poeciliid fishes. *Molecular Phylogenetics and Evolution*, 43(3):986 – 998, 2007. ISSN 1055-7903. doi: https://doi.org/10.1016/j.ympev.2006.06.009. URL http://www.sciencedirect.com/science/article/pii/S1055790306002399.

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

[65] W. Ronny Huang, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. Understanding generalization through visualizations. In Jessica Zosa Forde, Francisco Ruiz, Melanie F. Pradier, and Aaron Schein, editors, *Proceedings on "I Can't Believe It's Not Better!" at NeurIPS Workshops*, volume 137 of *Proceedings of Machine Learning Research*, pages 87–97. PMLR, 12 Dec 2020. URL https://proceedings.mlr.press/v137/huang20a.html.

[66] Lily C Hughes, Guillermo Ortí, Yu Huang, Ying Sun, Carole C Baldwin, Andrew W Thompson, Dahiana Arcila, Ricardo Betancur-R, Chenhong Li, Leandro Becker, et al. Comprehensive phylogeny of ray-finned fishes (actinopterygii) based on transcriptomic and genomic data. *Proceedings of the National Academy of Sciences*, 115(24):6249–6254, 2018.

[67] Ahsan Jalal, Ahmad Salman, Ajmal Mian, Mark Shortis, and Faisal Shafait. Fish detection and species classification in underwater environments using deep learning with temporal information. *Ecological Informatics*, 57:101088, 2020. ISSN 1574-9541. doi: https://doi.org/10.1016/j.ecoinf.2020.101088. URL https://www.sciencedirect.com/science/article/pii/S1574954120300388.

[68] 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.

[69] Xiaowei Jin, Shengze Cai, Hui Li, and George Em Karniadakis. 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.

[70] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. *Advances in neural information processing systems*, 32, 2019.

[71] 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.

[72] 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.

[73] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019.

[74] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020.

[75] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. *Advances in Neural Information Processing Systems*, 34:852–863, 2021.

[76] Kenji Kawaguchi. Deep learning without poor local minima, 2016.

[77] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. 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.

[78] Asifullah Khan, Anabia Sohail, Umme Zahoora, and Aqsa Saeed Qureshi. A survey of the recent architectures of deep convolutional neural networks. *Artificial Intelligence Review*, 53(8):5455–5516, Apr 2020. ISSN 1573-7462. doi: 10.1007/s10462-020-09825-6. URL http://dx.doi.org/10.1007/s10462-020-09825-6.

[79] Ron Kikinis, Steve D. Pieper, and Kirby G. Vosburgh. 3D Slicer: A Platform for Subject-Specific Image Analysis, Visualization, and Clinical Support. In *Intraoperative Imaging and Image-Guided Therapy*, pages 277–289. Springer, New York, NY, 2014. ISBN 978-1-4614-7656-6 978-1-4614-7657-3. doi: 10.1007/978-1-4614-7657-3_19. URL https://link.springer.com/chapter/10.1007/978-1-4614-7657-3_19.

[80] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

[81] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[82] Andrew P. Kinziger, Robert M. Wood, and David A. Neely. Molecular Systematics of the Genus Cottus (Scorpaeniformes: Cottidae). *Copeia*, 2005(2):303 – 311, 2005. doi: 10.1643/CI-03-290R1. URL https://doi.org/10.1643/CI-03-290R1.

[83] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[84] Aditi Krishnapriyan, Amir Gholami, Shandian Zhe, Robert Kirby, and Michael W Mahoney. Characterizing possible failure modes in physics-informed neural networks. *Advances in Neural Information Processing Systems*, 34, 2021.

[85] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[86] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[87] Yakup Kutlu, Bilal Iscimen, and Cemal Turan. Multi-stage fish classification system using morphometry. *Fresenius Environmental Bulletin*, 26:1910–1916, 03 2017.

[88] Rasmus Larsen, Hildur Olafsdottir, and Bjarne Kjær Ersbøll. Shape and texture based classification of fish species. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 5575 LNCS, pages 745–749. Springer, Berlin, Heidelberg, 2009. ISBN 3642022294. doi: 10.1007/978-3-642-02230-2_76.

[89] D. J. Lee, Sharon Redd, Robert Schoenberger, Xiaoqian Xu, and Pengcheng Zhan. An Automated Fish Species Classification and Migration Monitoring System. In *IECON Proceedings (Industrial Electronics Conference)*, volume 2, pages 1080–1085, 2003. doi: 10.1109/IECON.2003.1280195.

[90] Dah Jye Lee, James K. Archibald, Robert B. Schoenberger, Aaron W. Dennis, and Dennis K. Shiozawa. Contour matching for fish species recognition and migration monitoring. *Studies in Computational Intelligence*, 122:183–207, 2008. ISSN 1860949X. doi: 10.1007/978-3-540-78534-7_8.

[91] 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.

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

[93] Xiao Li, Chenghua Lin, Ruizhe Li, Chaozheng Wang, and Frank Guerin. Latent space factorisation and manipulation via matrix subspace projection. In *International Conference on Machine Learning*, pages 5916–5926. PMLR, 2020.

[94] Sicong Liang and Yu Zhang. A simple general approach to balance task difficulty in multi-task learning. *arXiv preprint arXiv:2002.04792*, 2020.

[95] Baijiong Lin, YE Feiyang, Yu Zhang, and Ivor Tsang. Reasonable effectiveness of random weighting: A litmus test for multi-task learning. *Transactions on Machine Learning Research*.

[96] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016. doi: 10.1017/jfm.2016.615.

[97] John Graham Lundberg. *The evolutionary history of North American catfishes, family Ictaluridae*. PhD thesis, University of Michigan, 1970.

[98] Abigail J. Lynch, Steven J. Cooke, Andrew M. Deines, Shannon D. Bower, David B. Bunnell, Ian G. Cowx, Vivian M. Nguyen, Joel Nohner, Kaviphone Phouthavong, Betsy Riley, Mark W. Rogers, William W. Taylor, Whitney Woelmer, So Jung Youn, and T. Douglas Beard. The social, economic, and environmental importance of inland fish and fisheries. *Environmental Reviews*, 24(2):115–121, 2016. ISSN 11818700. doi: 10.1139/er-2015-0064.

[99] Chao Ma, Daniel Kunin, Lei Wu, and Lexing Ying. Beyond the quadratic approximation: the multiscale structure of neural network loss landscapes. *arXiv preprint arXiv:2204.11326*, 2022.

[100] Ernst Mayr. Biological classification: Toward a synthesis of opposing methodologies. *Science*, 214(4520):510–516, 1981. ISSN 0036-8075. doi: 10.1126/science.214.4520.510. URL https://science.sciencemag.org/content/214/4520/510.

[101] Robert McGill, John W Tukey, and Wayne A Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.

[102] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layers neural networks, 2018.

[103] ML Menéndez, JA Pardo, L Pardo, and MC Pardo. The jensen-shannon divergence. *Journal of the Franklin Institute*, 334(2):307–318, 1997.

[104] 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.

[105] Keerthiram Murugesan, Hanxiao Liu, Jaime Carbonell, and Yiming Yang. Adaptive smoothed online multi-task learning. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 4296–4304. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/a869ccbcbd9568808b8497e28275c7c8-Paper.pdf.

[106] Thomas J. Near, Daniel I. Bolnick, and Peter C. Wainwright. Investigating phylogenetic relationships of sunfishes and black basses (actinopterygii: Centrarchidae) using dna sequences from mitochondrial and nuclear genes. *Molecular Phylogenetics and Evolution*, 32(1):344 – 357, 2004. ISSN 1055-7903. doi: https://doi.org/10.1016/j.ympev.2003.12.010. URL http://www.sciencedirect.com/science/article/pii/S1055790304000284.

[107] Thomas J. Near, Ron I. Eytan, Alex Dornburg, Kristen L. Kuhn, Jon A. Moore, Matthew P. Davis, Peter C. Wainwright, Matt Friedman, and W. Leo Smith. Resolution of ray-finned fish phylogeny and timing of diversification. *Proceedings of the National Academy of Sciences of the United States of America*, 109(34):13698–13703, 2012. ISSN 00278424. doi: 10.1073/pnas.1206625109.

[108] Matthew E. Neilson and Carol A. Stepien. Escape from the ponto-caspian: Evolution and biogeography of an endemic goby species flock (benthophilinae: Gob-

iidae: Teleostei). *Molecular Phylogenetics and Evolution*, 52(1):84 – 102, 2009. ISSN 1055-7903. doi: https://doi.org/10.1016/j.ympev.2008.12.023. URL http://www.sciencedirect.com/science/article/pii/S105579030800612X.

[109] Anh Mai Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, pages 427–436, 2015. URL https://doi.org/10.1109/CVPR.2015.7298640.

[110] Ngoc Quynh Nguyen. Optimization landscape of deep neural networks. 2019.

[111] Quynh Nguyen, Mahesh Chandra Mukkamala, and Matthias Hein. On the loss landscape of a class of deep neural networks with no bad local valleys. *arXiv preprint arXiv:1809.10749*, 2018.

[112] B B Normark, A R McCune, and R G Harrison. Phylogenetic relationships of neopterygian fishes, inferred from mitochondrial DNA sequences. *Molecular Biology and Evolution*, 8(6):819–834, 11 1991. ISSN 0737-4038. doi: 10.1093/oxfordjournals.molbev.a040685. URL https://doi.org/10.1093/oxfordjournals.molbev.a040685.

[113] S O Ogunlana, O Olabode, S A A Oluwadare, and G B Iwasokun. Fish Classification Using Support Vector Machine. Technical Report 2, 2015. URL www.ajocict.net.

[114] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2017. URL https://arxiv.org/abs/1711.00937.

[115] Ghanshyam Pilania, Kenneth James McClellan, Christopher Richard Stanek, and Blas P Uberuaga. Physics-informed machine learning for inorganic scintillator discovery. *The Journal of chemical physics*, 148(24):241729, 2018.

[116] Vinay Uday Prabhu, Dian Ang Yap, Joyce Xu, and John Whaley. Under-

standing adversarial robustness through loss landscape geometries. *arXiv preprint arXiv:1907.09061*, 2019.

[117] Samantha A Price, Sarah T Friedman, Katherine A Corn, Olivier Larouche, Kasey Brockelsby, Anna J Lee, Maya Nagaraj, Nick G Bertrand, Mailee Danao, Megan C Coyne, et al. Fishshapes v1: Functionally relevant measurements of teleost shape and size on three dimensions, 2022.

[118] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[119] Rahul Rai and Chandan K. Sahu. Driven by data or derived through physics? a review of hybrid physics guided machine learning techniques with cyber-physical system (cps) focus. *IEEE Access*, 8:71050–71073, 2020. doi: 10.1109/ACCESS.2020.2987324.

[120] 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.

[121] 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.

[122] 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.

[123] Dhruv Rathi, Sushant Jain, and S. Indu. Underwater Fish Species Classification using Convolutional Neural Network and Deep Learning. In *2017 9th International Con-*

ference on Advances in Pattern Recognition, ICAPR 2017, pages 344–349. Institute of Electrical and Electronics Engineers Inc., dec 2018. ISBN 9781538622414. doi: 10.1109/ICAPR.2017.8593044.

[124] Hafiz Tayyab Rauf, Muhammad Ikram Lali, Saliha Zahoor, Syed Zakir Shah, Abd Rehman, and Syed Ahmad Chan Bukhari. Visual features based automated identification of fish species using deep convolutional neural networks. *Computers and Electronics in Agriculture*, 11 2019. doi: 10.1016/j.compag.2019.105075.

[125] Elad Richardson, Yuval Alaluf, Or Patashnik, Yotam Nitzan, Yaniv Azar, Stav Shapiro, and Daniel Cohen-Or. Encoding in style: a stylegan encoder for image-to-image translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2287–2296, 2021.

[126] Rachel Rodgers. *Phylogenomic Analysis of Fundulidae Using RNA-Sequencing Data*. PhD thesis, Southern Illinois University at Edwardsville, 2017.

[127] Marco T.A. Rodrigues, Flávio L.C. Pádua, Rogério M. Gomes, and Gabriela E. Soares. Automatic fish species classification based on robust feature extraction techniques and artificial immune systems. In *Proceedings 2010 IEEE 5th International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010*, pages 1518–1525, 2010. ISBN 9781424464388. doi: 10.1109/BICTA.2010.5645273.

[128] Franz M Rohrhofer, Stefan Posch, Clemens Gößnitzer, and Bernhard C Geiger. Understanding the difficulty of training physics-informed neural networks on dynamical systems. *arXiv preprint arXiv:2203.13648*, 2022.

[129] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215, 2019.

[130] Kateřina Rylková, Lukáš Kalous, Vendula Šlechtová, and Jörg Bohlen. Many branches, one root: First evidence for a monophyly of the morphologically highly diverse goldfish (carassius auratus). *Aquaculture*, 302(1):36 – 41, 2010. ISSN 0044-8486. doi: https://doi.org/10.1016/j.aquaculture.2010.02.003. URL http://www.sciencedirect.com/science/article/pii/S0044848610000918.

[131] Susana Schönhuth, Jasna Vukić, Radek Šanda, Lei Yang, and Richard L. Mayden. Phylogenetic relationships and classification of the holarctic family leuciscidae (cypriniformes: Cyprinoidei). *Molecular Phylogenetics and Evolution*, 127:781 – 799, 2018. ISSN 1055-7903. doi: https://doi.org/10.1016/j.ympev.2018.06.026. URL http://www.sciencedirect.com/science/article/pii/S1055790317308618.

[132] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. doi: 10.1109/ICCV.2017.74.

[133] Yeonjong Shin. On the convergence of physics informed neural networks for linear second-order elliptic and parabolic type pdes. *Communications in Computational Physics*, 28(5):2042–2074, Jun 2020. ISSN 1991-7120.

[134] Benjamin W. B. Shires and Chris J. Pickard. 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.

[135] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[136] Carlos N Silla and Alex A Freitas. A survey of hierarchical classification across different application domains. *Data Mining and Knowledge Discovery*, 22(1-2):31–72, 2011.

[137] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[138] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *CoRR*, abs/1312.6034, 2014.

[139] Brian L Sloss, Neil Billington, and Brooks M Burr. A molecular phylogeny of the percidae (teleostei, perciformes) based on mitochondrial dna sequence. *Molecular Phylogenetics and Evolution*, 32(2):545 – 562, 2004. ISSN 1055-7903. doi: https://doi.org/10.1016/j.ympev.2004.01.011. URL http://www.sciencedirect.com/science/article/pii/S105579030400048X.

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

[141] John P Snyder. *Flattening the earth: two thousand years of map projections*. University of Chicago Press, 1997.

[142] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *arXiv preprint arXiv:2101.10382*, 2021.

[143] Rick Stevens, Valerie Taylor, Jeff Nichols, Arthur Barney Maccabe, Katherine Yelick, and David Brown. Ai for science. Technical report, Argonne National Lab.(ANL), Argonne, IL (United States), 2020.

[144] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge, 2016.

[145] Ming Sun, Yuchen Yuan, Feng Zhou, and Errui Ding. Multi-attention multi-class con-

straint for fine-grained image recognition. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[146] Ruoyu Sun, Dawei Li, Shiyu Liang, Tian Ding, and Rayadurgam Srikant. The global landscape of neural networks: An overview. *IEEE Signal Processing Magazine*, 37(5): 95–108, 2020. doi: 10.1109/MSP.2020.3004124.

[147] Tyler Sypherd, Mario Diaz, Lalitha Sankar, and Gautam Dasarathy. On the alpha-loss landscape in the logistic model, 2020.

[148] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[149] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[150] P.N. Tan, M. Steinbach, A. Karpatne, and V. Kumar. *Introduction to data mining (2nd edition)*. Pearson Addison Wesley Boston, 2018.

[151] E. B. Taylor and J. J. Dodson. A molecular analysis of relationships and biogeography within a species complex of holarctic fish (genus osmerus). *Molecular Ecology*, 3(3):235–248, 1994. doi: https://doi.org/10.1111/j.1365-294X.1994.tb00057.x. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1365-294X.1994.tb00057.x.

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

[153] Nicholas J. Tustison, Zixuan Lin, Xue Feng, Nicholas Cullen, Jaime F. Mata, Lucia Flors, James C. Gee, Talissa A. Altes, John P. Mugler III, and Kun Qing. Convolutional

neural networks with template-based data augmentation for functional lung image quantification. *Academic Radiology*, 2018. URL https://www.ncbi.nlm.nih.gov/pubmed/30195415.

[154] Aaron Van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

[155] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

[156] W. Vanwinsen. Evaluating the performance of simulated imu data for animal activity recognition, August 2021. URL http://essay.utwente.nl/88458/.

[157] Umang Varma. *A Paucity of Data in Machine Learning: Applications in Single Cell RNA Sequencing and Ranking*. PhD thesis, 2019.

[158] Sebastien Villon, David Mouillot, Marc Chaumont, Gérard Subsol, Thomas Claverie, and Sébastien Villéger. A new method to control error rates in automated species identification with deep learning algorithms. *Scientific Reports*, 10:10972, 07 2020. doi: 10.1038/s41598-020-67573-7.

[159] Sébastien Villon, Corina Iovan, Morgan Mangeas, Thomas Claverie, David Mouillot, Sébastien Villéger, and Laurent Vigliola. Automatic underwater fish species classification with limited data using few-shot learning. *Ecological Informatics*, 63:101320, 2021. ISSN 1574-9541. doi: https://doi.org/10.1016/j.ecoinf.2021.101320. URL https://www.sciencedirect.com/science/article/pii/S1574954121001114.

[160] JR Waldman. Systematics of morone (pisces: Moronidae), with notes on the lower percoids (distribution, cladistics, striped bass, siniperca, centropomidae). 1987.

[161] David J Wales, Mark A Miller, and Tiffany R Walsh. Archetypal energy landscapes. *Nature*, 394(6695):758–760, 1998.

[162] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[163] 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.

[164] Jiayun Wang, Yubei Chen, Rudrasis Chakraborty, and Stella X. Yu. Orthogonal convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[165] Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why pinns fail to train: A neural tangent kernel perspective. *arXiv preprint arXiv:2007.14527*, 2020.

[166] Sifan Wang, Yujun Teng, and Paris Perdikaris. 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.

[167] Sifan Wang, Yujun Teng, and Paris Perdikaris. Understanding and mitigating gradient flow pathologies in physics-informed neural networks. *SIAM Journal on Scientific Computing*, 43(5):A3055–A3081, 2021.

[168] Wei Wen, Yandan Wang, Feng Yan, Cong Xu, Chunpeng Wu, Yiran Chen, and Hai Li. Smoothout: Smoothing out sharp minima to improve generalization in deep learning, 2018.

[169] 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.

[170] Jin-Long Wu, Heng Xiao, and Eric Paterson. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7), Jul 2018. ISSN 2469-990X.

[171] Weibin Wu, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R. Lyu, and Yu-Wing Tai. Towards global explanations of convolutional neural networks with concept attribution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[172] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[173] Chen Xing, Devansh Arpit, Christos Tsirigotis, and Yoshua Bengio. A walk with sgd, 2018.

[174] Joyce Xu, Dian Ang Yap, and Vinay Uday Prabhu. Understanding adversarial robustness through loss landscape geometries. In *Proc. of the International Conference on Machine Learning (ICML) Workshops*, page 18, 2019.

[175] Yaoqing Yang, Liam Hodgkinson, Ryan Theisen, Joe Zou, Joseph E. Gonzalez, Kannan

Ramchandran, and Michael W. Mahoney. Taxonomizing local versus global structure in neural network loss landscapes, 2021.

[176] Zhewei Yao, Amir Gholami, Kurt Keutzer, and Michael W. Mahoney. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 581–590, 2020. doi: 10.1109/BigData50022.2020.9378171.

[177] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *IEEE transactions on knowledge and data engineering*, 26(8):1819–1837, 2013.

[178] Quanshi Zhang, Ying Nian Wu, and Song-Chun Zhu. Interpretable convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[179] Heliang Zheng, Jianlong Fu, Zheng-Jun Zha, Jiebo Luo, and Tao Mei. Learning rich part hierarchies with progressive attention networks for fine-grained image recognition. *IEEE Transactions on Image Processing*, 29:476–488, 2020. doi: 10.1109/TIP.2019.2921876.

[180] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization, 2015.

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

Appendices

Appendix A

Electromagnetic Propagation and Maxwell Equation

In this application, electromagnetic waves propagate in periodically stratified layer stacks. This propagation can be described by Maxwell equations:

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

$$\vec{\nabla} \times \vec{E} = -\frac{1}{c}\frac{\partial \vec{H}}{\partial t}, \tag{A.2}$$

where $\vec{E}$, $\vec{H}$, ϵ, c, and t represent electric and magnetic fields, permittivity of layered material, and speed of light in vacuum (in Gaussian units), and time, respectively. When the permittivity of layered array is a 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 complex-valued eigen-value problem:

$$\hat{A}\vec{h}_m = k_z^m \vec{h}_m \tag{A.3}$$

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 $\epsilon(x)$ [104].

Appendix B

Quantum Mechanics and The Ising Chain Model

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.

All information related to the dynamics of the quantum system is encoded in the eigenvectors of $\hat{H}$, i.e., Ψ. Among these eigenvectors, the ground state wave-function, Ψ_0, defined as the eigenvector 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.

In this thesis, a quintessential model of the transverse field: Ising chain model [12], is targeted. The Ising chain model is a uni-dimensional spin chain model under the influence of a transverse magnetic field B_xSpin 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 [14, 181], and can also be used for quantum computing[152], 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.

Appendix C

CoPhy-PGNN Hyper-parameters

For both applications, a non-linearity of tanh and an optimizer of Adamax [80] were used.

	N	λ_{S0}	α_S	λ_{C0}	α_C	T_a	T	Architecture	input size	output size	number of epochs
Ising Model	100,000	2.3	0.14	0.85	0.17	51	50	FCC(4×100)	$2^4 \times 2^4$	$2^4 + 1$	100
Electromagnetic Propagation	2,000	1	0.95	0	0.01	2000	50	FCC(2×100)	$(400 \times 2) \times (400 \times 2)$	$(400 + 1) \times 2$	3000

Appendix D

Phylo-NN Hyper-parameters

In terms of hyper-parameter tuning, the following settings are selected for each of the trained models:

Vanilla VQGAN: A VQGAN is trained with a codebook of 1024 possible codes and an embedding sequence of 256 codes. The model is trained for 836 epochs with a learning rate of 4.5×10^{-6}. A batch size of 32 was used.

Phylo-NN Taking the base VQGAN model described above, a *Phylo-NN* that has $\mathbf{z}_{\mathrm{p}}^{Q}$ of dimensions $(n_{\mathrm{l}} = 4, n_{\mathrm{p}} = 8)$ is trained. $\mathbf{z}_{\mathrm{np}}^{Q}$ also has the same dimensionality. The dimensionality of the embedding itself is $d = 16$, and the size of the codebook $n_{\mathrm{q}} = 64$. A batch size of 32 was used.

Appendix E

The Taxonomies Used in *HGNN*

Table E.1 lists the 24 genera from which species were examined in the study of *HGNN*. Most of the genera (21) are monophyletic, and thus their taxonomy mirrors their evolutionary relationships. Three of the 24 genera, however, are not monophyletic (i.e., some of the species in the genus are more closely related to species from other genera, than they are to other species in the genus). Here the taxonomic hierarchy does not capture similarity due to common ancestry.

Table E.1: The monophyly of the genera used in studing *HGNN*.

Genus	Reference supporting monophyly	Monophyly
Alosa	[13]	Monophyletic
Carassius	[130]	Monophyletic
Chrosomus	[131]	Monophyletic
Cottus	[82]	Included species are members of monophyletic *Uranidea Clade*
Cyprinella	[131]	Monophyletic
Cyprinus	[21]	Monophyletic, tetraploidy
Enneacanthus	[106]	Monophyletic
Esox	[52]	Monophyletic
Fundulus	[126]	Not Monophyletic
Gambusia	[63]	Monophyletic
Hybopsis	[131]	Monophyletic
Lepisosteus	[112]	Monophyletic
Lepomis	[54]	Monophyletic
Luxilus (a.k.a *Luxilis*)	[131]	Not monophyletic
Lythrurus	[131]	Monophyletic
Morone	[160]	Monophyletic
Neogobius	[108]	Monophyletic
Notropis	[131]	Not monophyletic
Noturus	[97] and [50]	Monophyletic
Oncorhynchus	[15]	Monophyletic
Opsopoeodus	[131]	Monophyletic with *Pimephales*
Osmerus	[151]	Monophyletic
Perca	[139]	Monophyletic
Phenacobius	[131]	Monophyletic
Salmo	[15]	Monophyletic

Appendix F

Curating and Pre-processing of GLIN Dataset

For this work, and in an effort to create a diverse and statistically substantial dataset, 60,000 images of fish specimens from five ichthyological research collections were aggregated. These collections, which participated in the Great Lakes Invasives Network Project (GLIN) [1] are Field Museum of Natural History http://www.tubri.org/HDR/FMNH/, Illinois Natural History Survey http://www.tubri.org/HDR/INHS/, J. F. Bell Museum of Natural History (http://www.tubri.org/HDR/JFBM/), Ohio State University Museum of Biological Diversity (http://www.tubri.org/HDR/OSUM/), and the University of Wisconsin-Madison Zoological Museum (http://www.tubri.org/HDR/UWZM/). The GLIN project is digitizing 1.73 million historical biological specimens representing 2,550 species, including fishes, clams, snails, mussels, algae and plants that are potentially invasive to the Great Lakes Region of the U.S.

The GLIN dataset, as is typical for biological species images, is highly imbalanced; some species have only a few images, while others have thousands. To alleviate this problem, and for computational feasibility, a number of semi-balanced subsets owere created for the purpose of training and evaluation. Specifically, there are two subsets that differ in terms of classification complexity (or difficulty). The first subset is called **Easy** and comes from

[1] http://greatlakesinvasives.org/

a single museum (Illinois Natural History Survey). Therefore, its images are homogeneous in terms of lighting and camera conditions. The second is called **Hard** and its images are aggregated from across all museums, making it a larger, more diverse, and more complex dataset. Comparing results from these two datasets helps illustrate the effects of dataset complexity on classification performance. Furthermore, two subsets of the **Easy** dataset were created by capping the number of images per species in the **Easy** dataset to 50, 100, or 200. These different dataset sizes help illustrate how training data paucity impacts the model's classification performance. Henceforth, the suffix of the datset will refer to the number of images per species. For example, **Easy**/100 has 100 images per species. More information and statistics on the included species can be found in the Supporting Material published at [38]. Unless explicitly mentioned otherwise, for each of these subsets, 64% of the data is used for training, 16% for validation, and the remainder for testing.

The acquired fish images typically contained a ruler, specimen label(s), and species tags along with the fish specimen. To retain only the fish region in the images, a 2D Unet model [46] was trained using a small portion of our data in the ANTsRNet software [153]. The image was manually segmented into background, fish, scale bar, and field notes using 3D Slicer [79]. The model's weights were used to automatically mask and crop the fish specimen portion of the remainder images. With the exception of rare cases where the fish overlapped the scale bar and/or the field notes, which were discarded, this pipeline resulted in successful generation of RGB fish-only images at the original resolution. The pipeline was implemented in R using ANTsR [7] and ANTsRNet.

Additionally, I performed data augmentation by randomly applying standard image transformations used in deep learning for computer vision, including translations of up to 0.25 of the image dimension, flips with a probability of 30%, rotations of up to 60°, and Gaussian random intensity variations using PCA with $\sigma = 0.1$ of the color channel value [86].

Data augmentation is critical when using ConvNets for image processing and is a common practice for fish classification [158, 159], especially when the available data is limited [135]. By training the model on variations of the same image, the model is deterred from learning nuanced patterns in the images that can lead to spurious performance, such as the intensity of the background, and encouraged to be robust under variable input conditions.

Finally, for the purposes of trait discovery and specimen image reconstruction, I further created a smaller subset of **Easy**/200 dataset that contains 200 images per species. We take 160 images for training and 40 for testing.

Appendix G

Phylogeny Preprocessing

The phylogeny tree corresponding to the dataset used for the *Phylo-NN* work was obtained using *opentree* (https://opentree.readthedocs.io/en/latest/) python package. Phylogeny processing and manipulation were done using *ete3* (http://etetoolkit.org/) python package.

As mentioned in Section 4.3, I quantize the 38-species tree into $n_l = 4$ distinct phylogenetic levels. Each level groups the 38 species based on their common ancestry within that level. The thresholds at which the phylogeny tree was discretized were selected such that the number of nodes in each level are equal. Tables G.1 and G.2 outline each level with its corresponding species groupings.

Table G.1: Phylogenetic groupings of the species included in the *Phylo-NN* work

Level	Species groupings
0	Alosa chrysochloris, Carassius auratus, Cyprinus carpio, Notropis atherinoides, Notropis blennius, Notropis boops, Notropis buccatus, Notropis buchanani, Notropis dorsalis, Notropis hudsonius, Notropis leuciodus, Notropis nubilus, Notropis percobromus, Notropis stramineus, Notropis telescopus, Notropis texanus, Notropis volucellus, Notropis wickliffi, Noturus exilis, Noturus flavus, Noturus gyrinus, Noturus miurus, Noturus nocturnus, Phenacobius mirabilis Esox americanus, Gambusia affinis, Lepomis auritus, Lepomis cyanellus, Lepomis gibbosus, Lepomis gulosus, Lepomis humilis, Lepomis macrochirus, Lepomis megalotis, Lepomis microlophus, Morone chrysops, Morone mississippiensis Lepisosteus osseus, Lepisosteus platostomus
1	Alosa chrysochloris Carassius auratus, Cyprinus carpio, Notropis atherinoides, Notropis blennius, Notropis boops, Notropis buccatus, Notropis buchanani, Notropis dorsalis, Notropis hudsonius, Notropis leuciodus, Notropis nubilus, Notropis percobromus, Notropis stramineus, Notropis telescopus, Notropis texanus, Notropis volucellus, Notropis wickliffi, Phenacobius mirabilis Esox americanus Gambusia affinis, Lepomis auritus, Lepomis cyanellus, Lepomis gibbosus, Lepomis gulosus, Lepomis humilis, Lepomis macrochirus, Lepomis megalotis, Lepomis microlophus, Morone chrysops, Morone mississippiensi Lepisosteus osseus, Lepisosteus platostomus Noturus exilis, Noturus flavus, Noturus gyrinus, Noturus miurus, Noturus nocturnus

Table G.2: Phylogenetic groupings of the species included in this study at different ancestral levels (continued)

Level	Species groupings
2	Alosa chrysochloris
	Carassius auratus, Cyprinus carpi
	Esox americanus
	Gambusia affinis
	Lepisosteus osseus, Lepisosteus platostomus
	Lepomis auritus, Lepomis cyanellus, Lepomis gibbosus, Lepomis gulosus, Lepomis humilis, Lepomis macrochirus, Lepomis megalotis, Lepomis microlophus
	Morone chrysops, Morone mississippiensis
	Notropis atherinoides, Notropis blennius, Notropis boops, Notropis buccatus, Notropis buchanani, Notropis dorsalis, Notropis hudsonius, Notropis leuciodus, Notropis nubilus', Notropis percobromus, Notropis stramineus, Notropis telescopus, Notropis texanus, Notropis volucellus, Notropis wickliffi, Phenacobius mirabilis
	Noturus exilis, Noturus flavus, Noturus gyrinus, Noturus miurus, Noturus nocturnus

Appendix H

FishShapes Dataset

FishShapes v1.0 dataset [117] contains expert-measured traits that are known to carry evolutionary signals, defined and collected using traditional methods that are subjective and labor-intensive. In this thesis, 8 functionally relevant traits from the FishShapes dataset for every fish species are used. Some species were not available in the FishShapes dataset, so when possible, the closest relative was substituted. (*Notropis percobromus* was replaced with *Notropis rubellus*, and *Carassius auratus* was replaced with *Carassius carassius*). Also, two species of *Lepisosteus* had no close relatives and were thus removed

The 8 functionally relevant traits that were used from the FishShapes dataset include: standard length, maximum body depth, maximum fish width, lower jaw length, mouth width, head depth, minimum caudal peduncle depth, and minimum caudal peduncle.

To correct for overall size and allometry, each measurement was log transformed and regressed against Standard Length (SL) using a phylogenetic regression in the R package *phylolm*, with the residuals from the regression being the inputs into phylogenetically-aligned components analysis (PACA, [26]. Distances in the principal components of PACA were measured as the Mahalonobis distance between the multivariate means using a covariance matrix proportional to the evolutionary rate matrix from the multivariate Brownian Motion fit in the R package mvMORPH [24].

Appendix I

Examples of Phylo Histograms

In Section 4.3.3, by means of calculating the average JS-divergence of sequence histograms, I investigated how well the Phylo-sequences match for species that share a common ancestor, as opposed to those that don't. In this appendix, I show some of these histogram plots to illustrate their value and the insight they provide.

In Figures I.1, I.2, and I.3, each histogram represents a code location in the phylogenetic sequence. Each column represents one of the $n_{\mathrm{l}} = 4$ phylogenetic levels into which the phylogeny tree was quantized, starting with the species level from right and climbing the phylogenetic tree all the way to the earliest ancestral level on the left. Each column has $n_{\mathrm{p}} = 8$ codes. Each histogram shows the relative frequency of each code of the learned $n_{\mathrm{q}} = 64$ codes for its corresponding sequence location. The lower a histogram's entropy (i.e., when there is only one or a couple of codes that dominate the histogram's frequency spectrum), the more important that code location hypothetically is for characterizing the species at its corresponding phylogenetic level.

As can be seen, both *Notropis* species share many codes at many sequence locations up to, but not including, the species level. This is because these species share an immediate ancestor. In contrast, We can see that the *Lepomis* species has a distinct histogram signature and does share almost no codes with the *Notropis* species, except for the earlier ancestral level (i.e., the left column).

Figure I.1: *Notropis nubilus*

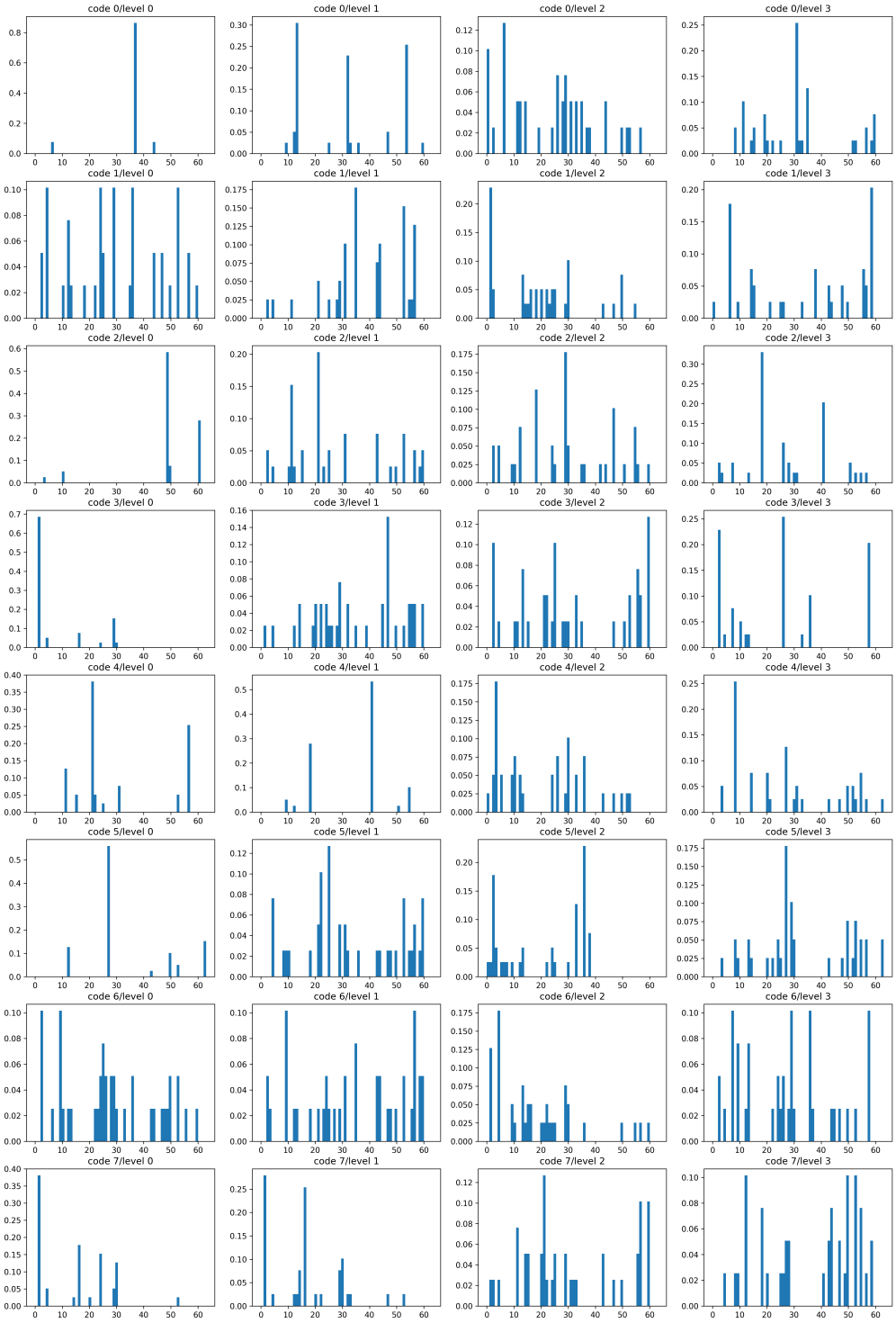


Figure I.2: *Notropis percobromus*

Figure I.3: *Lepomis macrochirus*