Protein Fitness Landscape: Spectral Graph Theory Perspective

Hao Zhu **Daniel M. Steinberg** **Piotr Koniusz**

Data61♥CSIRO
{allen.zhu, dan.steinberg, piotr.koniusz}@data61.csiro.au

Abstract

In this work, we present a novel theoretical framework for analyzing and modeling protein fitness landscapes using spectral graph theory. By representing the protein sequence space as a generalized Hamming graph and studying its spectral properties, we derive a set of powerful tools for quantifying the ruggedness, epistasis, and other key characteristics of the landscape. We prove strong approximation and sampling results, showing that the landscape can be efficiently learned and optimized from limited and noisy data. Building on this foundation, we introduce Propagational Convolutional Neural Networks (PCNNs), a new class of inductive surrogate oracle. We provide rigorous theoretical guarantees on the generalization and convergence properties of PCNNs, using techniques from the Neural Tangent Kernel framework. Extensive experiments on real-world protein engineering tasks demonstrate the superiority of PCNNs over state-of-the-art methods, achieving higher fitness and better generalization from limited data.

1 Introduction

Recent years have seen a surge in the application of machine learning techniques to protein engineering, with methods ranging from deep learning models [Alley et al., 2019, Wu et al., 2019, Biswas et al., 2021, Yang et al., 2019b] to Bayesian optimization strategies [Romero et al., 2013, Yang et al., 2019a]. These approaches have shown promise in predicting protein properties and guiding the search through sequence space [Fox et al., 2007]. However, one significant limitation persists: the disconnection between these data-driven methods and the underlying theoretical understanding of protein fitness landscapes.

Current theories of protein fitness landscapes, largely based on statistical physics and evolutionary biology, provide valuable insights into these landscapes' general properties. A protein fitness landscape maps each sequence to its corresponding fitness value (*e.g.*, enzyme activity, binding affinity, or stability), with topology characterized by ruggedness (prevalence of local optima) and epistasis (non-additive interactions between mutations). Understanding these properties is crucial for effective protein engineering, as they determine how easily the landscape can be navigated. However, these theories often fall short of directly informing the design and implementation of machine learning models. Many concepts, *e.g.*, NK models [Kauffman and Weinberger, 1989], do not readily translate into actionable strategies for model architecture or training procedures.

Conversely, machine learning approaches to protein engineering, while often empirically successful, operate as "black boxes", lacking a clear connection to the fundamental properties of fitness landscapes. This division limits our ability to interpret model predictions, understand their limitations, and systematically improve their performance across different protein engineering tasks. The gap between theory and practice is particularly evident in several key areas:

i. *Sample Efficiency* – lack of theoretical understanding of how the properties of fitness landscapes affect the sample complexity of learning algorithms.
ii. *Generalization* – existing frameworks fail to explain why certain machine learning models generalize well to unseen protein sequences, especially in the context of sparse and rugged fitness landscapes.
iii. *Robustness to noise* – theoretical models often assume idealized, noise-free landscapes, yet, real-world protein engineering must contend with sig-

Proceedings of the 28th International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258. Copyright 2025 by the author(s).

nificant experimental noise whose impact on model performance lacks a solid theoretical foundation.

To address these challenges, we present a novel theoretical framework that bridges the gap between the abstract models of fitness landscapes and the practical considerations of machine learning for protein engineering. Our framework leverages spectral graph theory to provide a unified approach to modeling and analyzing protein fitness landscapes. Building upon our framework, we introduce Propagational Convolutional Neural Networks (PCNNs), a novel architecture designed for learning of, and optimization on, protein fitness landscapes. Our PCNNs capture the complex structure of the landscape while maintaining sample efficiency and robustness to noise. The inductive nature of PCNNs helps them scale efficiently w.r.t. sequence length. We provide theoretical guarantees for the performance of PCNNs, establishing bounds on their sample complexity, generalization error, and convergence rate under mild assumptions on the properties of the fitness landscape. These guarantees leverage the spectral graph theory tools developed in our framework. We also conduct experiments on real-world protein engineering tasks, including the optimization of fluorescent proteins and the design of novel enzymes. Our results show that PCNNs consistently outperform state-of-the-art methods, achieving higher fitness and better generalization from limited and noisy data.

Our contributions are as follows:

1. We introduce a novel spectral graph theory framework for modeling and analyzing protein fitness landscapes, connecting the theoretical properties of the landscape to the practical performance of machine learning algorithms.
2. We propose Propagational Convolutional Neural Networks (PCNNs), a new architecture for learning and optimization on protein fitness landscapes, and provide theoretical performance guarantees.
3. We demonstrate the practical utility of our approach through experiments on real-world protein engineering tasks, showing that PCNNs outperform state-of-the-art methods.

2 Related Work

Protein Fitness Landscape Modeling. The concept of fitness landscapes is a fundamental tool for understanding and navigating the space of protein sequences and functions. Early works on protein fitness landscape modeling focused on simple statistical models, *e.g.*, the NK model [Kauffman and Weinberger, 1989], which capture the basic features of epistasis and ruggedness. More recent approaches have incorporated additional biophysical and evolutionary information, *e.g.*, the Gaussian process model of Romero et al. [2013]. However, such models often fail to capture the full complexity of real protein fitness landscapes, which are high-dimensional, non-linear, and characterized by long-range interactions between distant parts of the sequence. Our framework in Section 3 is an expressive approach to modeling these landscapes, which explicitly represents the structure of the sequence space and the dependencies between positions.

Spectral Graph Theory. Studies on the properties and applications of graph Laplacians and related operators [Von Luxburg, 2007] provide powerful tools for machine learning and data analysis. In particular, spectral methods have been used for clustering [Ng et al., 2001], dimensionality reduction [Belkin and Niyogi, 2003], and semi-supervised learning [Zhou et al., 2004] on graph-structured data. Our work leverages spectral graph theory to develop a novel representation and analysis of protein fitness landscapes. By modeling the sequence space as a generalized Hamming graph and studying its spectral properties, we derive a set of powerful tools for quantifying the ruggedness, epistasis, and other key characteristics of the landscape. To the best of our knowledge, this is the first application of spectral graph theory to the domain of protein fitness landscapes.

Machine Learning for Protein Fitness Prediction. The interest in applying machine learning techniques to predict protein fitness from sequence data is growing. These methods aim to learn a surrogate model of the fitness landscape to guide the search for improved variants. A major challenge in learning surrogate models for protein fitness is the scarcity of labeled data. Experimental fitness measurements are often time-consuming and expensive to obtain, limiting the size of the training datasets. Moreover, these measurements are typically noisy due to experimental variability and measurement errors, which can hinder learning of accurate models. Convolutional neural networks (CNNs) are a popular choice for this task due to their ability to capture local patterns and dependencies in protein sequences [Alley et al., 2019, Biswas et al., 2021, Wu et al., 2019]. However, existing CNN-based approaches often fail to model the complex non-linear interactions between distant positions in the sequence that are characteristic of real protein fitness landscapes. These methods often lack rigorous theoretical guarantees on their sample complexity, generalization error, and convergence rate. Recent works in Bayesian optimization by Deshwal et al. [2021] and Oh et al. [2019] have shown that Walsh functions can be used as a basis for diffusion kernels over combinatorial inputs, which shares mathematical similarities with our approach, though they focus on optimization rather than landscape characterization.

3 Theoretical Framework

Below we introduce our spectral graph theory framework for analyzing protein fitness landscapes. Firstly, we represent the protein sequence space as a generalized Hamming graph to capture the structure of the landscape and the relationships between sequences. We then study the spectral properties of this graph, deriving a set of powerful tools for quantifying the ruggedness, epistasis, and other key characteristics of the landscape. Building on these results, we investigate the problem of approximating the full fitness landscape from a small number of sampled sequences, and establish a formal relationship between our spectral framework and the well-known NK model.

3.1 Protein Sequence Space on Generalized Hamming Graph

We begin by formally defining the protein sequence space and its representation as a graph.

Definition 3.1 (Protein Sequence Space). *Let $\mathcal{V}$ be the set of all possible protein sequences of length N, where each position can be occupied by one of 20 standard amino acids. The size of this space is $|\mathcal{V}| = 20^N$.*

To capture the relationships between sequences, we introduce the concept of Hamming distance.

Definition 3.2 (Hamming Distance). *For two sequences $\boldsymbol{x} = (x_1, \ldots, x_N)$ and $\boldsymbol{y} = (y_1, \ldots, y_N)$, the Hamming distance $H(\boldsymbol{x}, \boldsymbol{y})$ is the number of positions at which the corresponding symbols differ,* i.e., $H(\boldsymbol{x}, \boldsymbol{y}) = |\{i : x_i \neq y_i\}|$.

Using the Hamming distance, we define the graph structure on the protein sequence space.

Definition 3.3 (Generalized Hamming Graph). *The protein sequence space is represented by a generalized Hamming graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (sequences) and $\mathcal{E}$ is the set of edges. Two sequences $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{V}$ are connected by an edge $(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{E}$ if and only if their Hamming distance satisfies $H(\boldsymbol{x}, \boldsymbol{y}) \leq k$, where k is a fixed constant.*

Properties of the generalized Hamming graph. The generalized Hamming graph $G(\mathcal{V}, \mathcal{E}, k)$ offers significant advantages for spectral analysis of protein fitness landscapes. Its key benefit lies in its invariant orthonormal basis of eigenfunctions, given by the Walsh functions, which depend only on the Hamming distance parameter k. This predetermined basis eliminates the need for computationally expensive eigendecomposition of the graph Laplacian, which would be infeasible for the large sequence space of proteins. The Walsh functions naturally represent different orders of interactions between sequence positions, directly capturing the concept of epistasis. This invariant basis enables efficient spectral decomposition of fitness functions, facilitating the analysis of landscape ruggedness and the characterization of epistatic interactions.

3.2 Spectral Analysis of the Fitness Landscape

To analyze the structure of the fitness landscape, we study the spectral properties of the graph Laplacian associated with the generalized Hamming graph.

Theorem 1 (Eigenvalues of the Generalized Hamming Graph). *For a distance-regular graph, the eigenvalues are related to the eigenvalues of its association scheme. In our case:*

$$\lambda_i = \sum_{j=0}^{k} P_i(j) \cdot K_j(N), \tag{1}$$

where $P_i(j)$ are orthogonal polynomials (specifically, q-Krawtchouk polynomials [Koekoek and Swarttouw, 1996]). $K_j(N)$ is the size of the j-th shell in the Hamming graph [Brouwer et al., 2012]. The multiplicity of λ_i is given by:

$$m_i = \frac{20^N - 1}{20^N \sum_{j=0}^{N} Q_i(j)^2 / K_j(N)}, \tag{2}$$

where Q_i are the dual polynomials to P_i.

This theorem establishes the eigenvalue structure of our generalized Hamming graph, connecting our approach to established results in spectral graph theory. The eigenvalues and their multiplicities characterize how protein fitness landscapes can be decomposed into spectral components with varying degrees of smoothness. This decomposition directly enables our subsequent analysis of landscape ruggedness and epistasis.

Theorem 2 (Eigenvectors of the Generalized Hamming Graph). *The eigenvectors of the graph Laplacian $\mathcal{L}$ are given by the generalized Walsh functions on the 20-letter amino acid alphabet. For a protein sequence $\mathbf{x} = (x_1, \ldots, x_N)$, where each $x_i \in \{0, 1, \ldots, 19\}$ represents one of the 20 standard amino acids, the generalized Walsh function corresponding to a subset $\mathcal{S} \subseteq \{1, \ldots, N\}$ (that can be interpreted as potential mutation sites or interacting residues in the protein) is defined as:*

$$\psi_{\mathcal{S}}(\mathbf{x}) = \prod_{i \in \mathcal{S}} \frac{1}{\sqrt{19}} \sum_{a=1}^{19} \omega^{a x_i}, \tag{3}$$

where $\omega = e^{2\pi i/20}$ is a primitive 20th root of unity. These generalized Walsh functions form a complete orthonormal basis:

$$\langle \psi_{\mathcal{S}}, \psi_{\mathcal{T}} \rangle = \frac{1}{20^N} \sum_{\mathbf{x} \in \mathcal{V}} \psi_{\mathcal{S}}(\mathbf{x}) \overline{\psi_{\mathcal{T}}(\mathbf{x})} = \delta_{\mathcal{S},\mathcal{T}}, \tag{4}$$

where $\delta_{\mathcal{S},\mathcal{T}}$ is the Kronecker delta, equal to 1 if $\mathcal{S} = \mathcal{T}$ and 0 otherwise.

The Walsh functions form a complete orthonormal basis for the space of functions on the Hamming graph, and they have a natural interpretation in terms of the interaction order of the fitness function (as we will see in the next subsection). Using the eigenvectors of the graph Laplacian, one can express any function on the generalized Hamming graph, including the fitness function, in terms of its spectral decomposition.

Theorem 3 (Spectral Decomposition of the Fitness Function)**.** *Let $f : \mathcal{V} \to \mathbb{R}$ be a fitness function defined on the protein sequence space. Then f can be uniquely expressed as a linear combination of the Walsh functions,*

$$f(\boldsymbol{x}) = \sum_{\mathcal{S} \subseteq \{1,\ldots,N\}} a_{\mathcal{S}} \psi_{\mathcal{S}}(\boldsymbol{x}),$$

where the coefficients $a_{\mathcal{S}}$ are given by the inner products $a_{\mathcal{S}} \equiv a_{\mathcal{S}}(f) = \langle f, \psi_{\mathcal{S}} \rangle$.

The spectral decomposition of the fitness function on the generalized Hamming graph lets us analyze the landscape in terms of its frequency components. The coefficients $a_{\mathcal{S}}$ associated with the Walsh functions $\psi_{\mathcal{S}}$ capture the contribution of different interaction orders to the overall fitness. This decomposition is analogous to the Fourier decomposition of the fitness function in the NK model, but here we use the Walsh basis, which is naturally suited to the hypercube structure of the sequence space. By studying the magnitude of the coefficients, we can quantify the degree of epistasis and the complexity of the fitness landscape.

Connection to ruggedness and epistasis. The spectral properties of the fitness function are closely related to the ruggedness and epistasis of the landscape. We introduce quantitative measures of these properties based on the spectral decomposition.

Definition 3.4 (Spectral Ruggedness Measure)**.** *The spectral ruggedness of a fitness function f is defined as*

$$R(f) = \frac{\sum_{\mathcal{S} \subseteq \{1,\ldots,N\}} |\mathcal{S}| \cdot a_{\mathcal{S}}^2}{\sum_{\mathcal{S} \subseteq \{1,\ldots,N\}} a_{\mathcal{S}}^2},$$

where $|\mathcal{S}|$ denotes the cardinality of subset $\mathcal{S}$, and $a_{\mathcal{S}}$ are the spectral coefficients of f.

This measure quantifies the relative contribution of higher-order interactions to the fitness landscape. A higher value of $R(f)$ indicates a more rugged landscape dominated by higher-order terms, while a lower value suggests a smoother landscape dominated by low-order terms. The measure is normalized by the total spectral energy, making it invariant to the overall scale of the fitness function. The range of $R(f)$ is $[0, N]$, with $R(f) \approx 0$ corresponding to a nearly additive landscape and $R(f) \approx N$ corresponding to a maximally rugged landscape with significant high-order interactions.

Definition 3.5 (Spectral Epistasis Measure)**.** *The spectral epistasis of a fitness function f is defined as*

$$E(f) = \frac{\sum_{\mathcal{S}:|\mathcal{S}|>1} a_{\mathcal{S}}^2}{\sum_{\mathcal{S} \subseteq \{1,\ldots,N\}} a_{\mathcal{S}}^2},$$

where the sum in the numerator is taken over all subsets $\mathcal{S}$ with cardinality greater than 1.

This measure quantifies the degree of epistatic interactions in the fitness landscape. Biologically, epistasis refers to the phenomenon where the effect of a mutation depends on the presence of other mutations. The spectral epistasis measure $E(f)$ ranges from 0 to 1, where: 1) $E(f) = 0$ occurs when there are no epistatic interactions (purely additive landscape); 2) $E(f) = 1$ occurs when the fitness is entirely determined by interactions between multiple positions; 3) Intermediate values indicate the relative importance of epistatic interactions.

3.3 Approximation with Sampling

In practical applications, one often has access to only a small number of fitness evaluations, and must approximate the full fitness landscape from this limited data. Our spectral framework provides natural tools to approach this problem.

Definition 3.6 (Sampled Graph)**.** *Given the full graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a sampling rate $p \in (0, 1]$, we define the sampled graph $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ where $\mathcal{V}_s \subseteq \mathcal{V}$ is a random subset with $|\mathcal{V}_s| \approx p|\mathcal{V}|$ and $\mathcal{E}_s = \{(\boldsymbol{u}, \boldsymbol{v}) \in \mathcal{E} : \boldsymbol{u}, \boldsymbol{v} \in \mathcal{V}_s\}$.*

Theorem 4 (Spectral Approximation for Generalized Hamming Graphs)**.** *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be our full generalized Hamming graph with Laplacian $\mathcal{L}$, and $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ be a sampled subgraph with Laplacian $\mathcal{L}_s$. For any $\epsilon > 0$, with probability at least $1 - \delta$:*

$$(1-\epsilon) \sum_{\mathcal{S}} a_{\mathcal{S}} \psi_{\mathcal{S}} \preceq \sum_{\mathcal{S}} a_{\mathcal{S}}^{(s)} \psi_{\mathcal{S}}^{(s)} \preceq (1+\epsilon) \sum_{\mathcal{S}} a_{\mathcal{S}} \psi_{\mathcal{S}}, \quad (5)$$

where $\preceq$ denotes element-wise inequality, $a_{\mathcal{S}}$ and $a_{\mathcal{S}}^{(s)}$ are the spectral coefficients for the full and sampled graphs respectively, and $\psi_{\mathcal{S}}$ and $\psi_{\mathcal{S}}^{(s)}$ are the corresponding Walsh functions. This approximation holds when the sampling rate p satisfies

$$p \geq C \frac{\log(d/\delta)}{\epsilon^2 d} \max_{\mathcal{S}} \frac{1}{|a_{\mathcal{S}}|} \quad (6)$$

for some constant C and the graph degree d.

This theorem establishes a strong connection between the spectral properties of the full graph and those of the sampled graph. It shows that with a sufficient sampling rate, the Walsh decomposition of the fitness function on the sampled graph closely approximates the true decomposition on the full graph, with high probability. For protein fitness landscapes, this condition is practically achievable due to the rapid decay of spectral coefficients $|a_{\mathcal{S}}|$ as$|\mathcal{S}|$ increases.

Corollary 5 (Coefficient-wise Approximation). *Under the same conditions as in Theorem 4, for each subset S,*

$$(1-\epsilon)\big|a_{\mathcal{S}}\big| \le \big|a_{\mathcal{S}}^{(s)}\big| \le (1+\epsilon)\big|a_{\mathcal{S}}\big|$$

with probability at least $1-\delta/2^N$*.*

This corollary provides a more precise guarantee, showing that each individual spectral coefficient is approximated well by its counterpart from the sampled graph.

Lemma 6 (Preservation of Interaction Order). *The sampled graph $\mathcal{G}_s$ preserves the maximum interaction order of the fitness landscape with high probability. Specifically, if $a_{\mathcal{S}} = 0$ for all $|\mathcal{S}| > K$ in the full graph, then with probability at least $1-\delta$, $a_{\mathcal{S}}^{(s)} = 0$ for all $|\mathcal{S}| > K$ in the sampled graph, provided that*

$$p \ge C\frac{K\log(N/\delta)}{N}$$

for some constant C. K is defined in Theorem 7.

This result ensures that our sampling approach maintains the structural sparsity of the fitness function in terms of interaction order, which is crucial for accurate characterization of protein fitness landscapes.

Proposition 1 (Ruggedness Preservation). *Let $R(f)$ and $R_s(f)$ be the spectral ruggedness measures for the full and sampled graphs respectively. Under the sampling conditions of Theorem 4:*

$$(1-\epsilon)R(f) \le R_s(f) \le (1+\epsilon)R(f)$$

with probability at least $1-\delta$*.*

This proposition guarantees that our sampling approach accurately preserves the ruggedness characteristics of the fitness landscape, which is essential for understanding the optimization challenges posed by different protein systems. Combined with Theorem 4, this result establishes that one can reliably assess landscape complexity from limited sampled data.

3.4 Connection to the NK Model

The NK model [Kauffman and Weinberger, 1989] is a widely used model of epistatic interactions in fitness landscapes. Our spectral framework provides a natural generalization of the NK model. Below, we cestablish a formal connection between the two models.

Theorem 7 (Connection to the NK Model). *Let f be a fitness function on the generalized Hamming graph $\mathcal{G}$. Let f_{NK} be the corresponding NK model with parameters N and K. For the standard NK model with overlapping neighborhoods, the spectral coefficients of f satisfy:*

$$a_{\mathcal{S}} = 0 \textit{ for all } \mathcal{S} \textit{ with } |\mathcal{S}| > 2K+1, \tag{7}$$

and the non-zero coefficients are related to the interaction coefficients of the NK model by:

$$a_{\mathcal{S}} = \frac{1}{N}\sum_{i=1}^{N} a_{\mathcal{S},i}, \tag{8}$$

where $a_{\mathcal{S},i}$ are the interaction coefficients of the i-th component of the NK model. For the special case of a modified NK model with non-overlapping neighborhoods, the stricter bound $a_{\mathcal{S}} = 0$ for all $\mathcal{S}$ with $|\mathcal{S}| > K+1$ holds.

This theorem demonstrates that the NK model with parameters N and K corresponds to a specific class within our spectral framework. For non-overlapping neighborhoods, interactions are limited to order $K+1$, while overlapping neighborhoods can generate interactions up to order $2K+1$. The spectral coefficients are weighted averages of local interaction terms, reflecting the NK model's composition as a sum of local fitness contributions. Our framework extends beyond NK models by allowing arbitrary interaction orders and structures. This flexibility is essential for protein fitness landscapes where interactions may involve distant residues due to tertiary structure contacts. The Walsh function basis efficiently captures these complex epistatic patterns without requiring predefined interaction neighborhoods.

4 Propagational Convolutional Neural Networks (PCNNs)

Propagational Convolutional Neural Networks (PCNNs) serve as a bridge between our theoretical framework of fitness landscapes and practical protein engineering challenges. The design of PCNNs is motivated by the spectral properties of the fitness landscape established in the previous section. By incorporating graph convolution operations on the generalized Hamming graph, PCNNs can effectively capture the complex interactions and dependencies between sequence positions, while maintaining the sample efficiency and noise robustness guaranteed by the spectral approximation results. PCNNs design is also related to inductive graph neural networks, particularly GraphSAGE

[Hamilton et al., 2017]. However, PCNNs incorporate key architectural modifications optimized for protein fitness landscapes. We decouple the feature extraction (by CNN) from graph propagation, leading to significant computational efficiency gains by avoiding repeated neighbor sampling during training. This design choice is crucial for protein engineering applications, where computational cost is a major constraint.

Definition of PCNNs. For a protein sequence $\boldsymbol{x}_i$, the PCNNs prediction $\tilde{f}(\boldsymbol{x}_i)$ is given by:

$$\tilde{f}(\boldsymbol{x}_i) = (1-\alpha) f_{CNN}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} f_{CNN}(\boldsymbol{x}_j), \tag{9}$$

where f_{CNN} is a CNN-based surrogate oracle, and $\alpha \in [0,1]$ is a propagation parameter. $\mathcal{N}(\boldsymbol{x}_i)$ is the neighborhood of $\boldsymbol{x}_i$ in the hypercube graph.

4.1 Basic Properties of PCNNs

Lemma 8 (Smoothness and Ruggedness)**.** *Let f_{CNN} be L-Lipschitz continuous on the Hamming graph, where the Hamming graph represents sequences differing by mutations at individual positions. Then $\tilde{f}$ is L'-Lipschitz continuous with $L' \leq L$. Moreover, the spectral ruggedness measure R of $\tilde{f}$ satisfies:*

$$R(\tilde{f}) \leq (1-\alpha) R(f_{CNN}) + \alpha R(f_{avg}), \tag{10}$$

where f_{avg} is the average of f_{CNN} over the neighborhood, and α is the propagation parameter.

This lemma provides a theoretical guarantee that PCNNs maintain or improve the smoothness of the fitness landscape. Since $R(f_{avg}) \leq R(f_{CNN})$ due to the averaging effect, PCNNs effectively reduce the ruggedness of the learned fitness function, making optimization more tractable.

Lemma 9 (Noise Reduction and Epistasis Estimation)**.** *Let f_{true} be the true fitness function and $f_{CNN} = f_{true} + \varepsilon$, where ε is a zero-mean noise with variance σ^2. Let $\hat{E}(f)$ be the estimated spectral epistasis measure. Then:*

$$Var\big(\tilde{f}(\mathbf{x}_i)\big) = \Big((1-\alpha)^2 + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|}\Big)\sigma^2. \tag{11}$$

Moreover, the variance of the estimated spectral epistasis measure satisfies:

$$Var\big(\hat{E}(\tilde{f})\big) \leq \Big((1-\alpha)^2 + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|}\Big) Var\big(\hat{E}(f_{CNN})\big), \tag{12}$$

where $\alpha \in [0,1]$ is the propagation parameter and $|\mathcal{N}(\boldsymbol{x}_i)|$ is the number of neighbors in the generalized Hamming graph.

This lemma shows that PCNNs reduce the variance of both the fitness predictions and the estimated epistasis measure, which is crucial for handling the noisy experimental data typical in protein engineering. The noise reduction effect becomes stronger as the neighborhood size increases, allowing PCNNs to make more reliable predictions in rugged and noisy fitness landscapes.

Theorem 10 (PCNNs Approximation)**.** *The graph convolution operation in PCNNs on the sampled graph $\mathcal{G}_s$ approximates the corresponding operation on the full graph $\mathcal{G}$ in the following sense:*

$$\big\|\tilde{f}_s - \tilde{f}\big\|_2 \leq \epsilon \big\|\tilde{f}\big\|_2 + \mathcal{O}\left(\alpha \sqrt{\frac{\log(N/\delta)}{p|\mathcal{V}|}}\right), \tag{13}$$

where $\tilde{f}_s$ and $\tilde{f}$ are the PCNNs outputs on $\mathcal{G}_s$ and $\mathcal{G}$ respectively, α is the propagation parameter, and p is the sampling rate.

This theorem directly connects PCNNs to our spectral graph theory framework, showing that the approximation quality depends on both the propagation parameter α and the sampling rate p. For protein fitness landscapes with rapid spectral decay, moderate sampling rates suffice to achieve a good approximation.

4.2 The Neural Tangent Kernel Perspective

The Neural Tangent Kernel (NTK) framework [Jacot et al., 2018] provides powerful tools for studying the generalization properties of over-parameterized neural networks. One can use this framework to analyze the behavior of PCNNs and compare it to traditional CNNs and Graph Neural Networks (GNNs).

NTK Definition. Consider a $f_{\text{CNN}}(\boldsymbol{x};\boldsymbol{\theta}) : \mathcal{S} \to \mathbb{R}$ with initial parameters $\boldsymbol{\theta}_0$ and a fixed input sequence s. The linearized form of the neural network is:

$$f^{lin}_{\text{CNN}}(\boldsymbol{x};\boldsymbol{\theta}) = f_{\text{CNN}}(\boldsymbol{x};\boldsymbol{\theta}_0) + \nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x};\boldsymbol{\theta})^\top (\boldsymbol{\theta} - \boldsymbol{\theta}_0). \tag{14}$$

As the network width tends to infinity, the gradient $\nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x};\boldsymbol{\theta})$ becomes a constant feature map $\phi(\boldsymbol{x}) : \mathcal{S} \to \mathbb{R}^{|\boldsymbol{\theta}|}$, inducing the Neural Tangent Kernel:

$$\begin{aligned}\text{NTK}(\boldsymbol{x}_i, \boldsymbol{x}_j) &= \phi_{ntk}(\boldsymbol{x}_i)^\top \phi_{ntk}(\boldsymbol{x}_j) \\ &= \langle \nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x}_i;\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x}_j;\boldsymbol{\theta})\rangle.\end{aligned} \tag{15}$$

Proposition 2 (Equivalence of CNN and PCNNs Solutions)**.** *A two-layer CNN with a fully-connected layer followed by max-pooling, and its corresponding PCNNs have the same minimum Reproducing Kernel Hilbert Space (RKHS)-norm NTK kernel regression solution, but this solution differs from that of a standard GNN,* i.e., $\mathbf{w}^*_{cnn} = \mathbf{w}^*_{pcnn} \neq \mathbf{w}^*_{gnn}$.

The equivalence means PCNNs maintain the same inductive bias as the underlying CNN, and gain smoothing benefits of graph propagation. The feature map is transformed from $\phi_{cnn}(\mathbf{x})$ to $\phi_{pcnn}(\mathbf{x})$ in inference:

$$f_{cnn}(\mathbf{x}) = \mathbf{w}_{cnn}^{\top}\phi_{cnn}(\mathbf{x}), \quad f_{pcnn}(\mathbf{x}) = \mathbf{w}_{cnn}^{\top}\phi_{pcnn}(\mathbf{x}). \tag{16}$$

This transformation lets PCNNs leverage both the representation learning power of CNNs and the structural properties of the graph.

Thus, the superior generalization performance of PCNNs compared to CNN can be explained by the transformation of the feature map from $\phi_{\text{cnn}}(\boldsymbol{x})$ to $\phi_{\text{pcnn}}(\boldsymbol{x})$ during inference.

Lemma 11 (The GNTK Perspective of PCNNs)**.** *The Graph Neural Tangent Kernel (GNTK) [Du et al., 2019] for a CNN with neighbour aggregation, denoted as g_G, can be expressed by the following: $g_G(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{|\mathcal{N}(\mathbf{x}_i)||\mathcal{N}(\mathbf{x}_j)|}\sum_{\boldsymbol{u}\in\mathcal{N}(\mathbf{x}_i)}\sum_{\boldsymbol{v}\in\mathcal{N}(\mathbf{x}_j)} g_{CNN}(\boldsymbol{u}, \boldsymbol{v})$. $\mathcal{N}(\boldsymbol{x}_i)$ and $\mathcal{N}(\boldsymbol{x}_j)$ are the neighborhoods of sequences $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ in the graph, respectively.*

This GNTK formulation provides insights into how PCNNs incorporate the local structure of the fitness landscape, and promote smoothness.

4.3 Generalization Analysis and Extrapolation Behavior

Theorem 12 (Generalization Bound for PCNNs)**.** *For any $\delta > 0$, with probability at least $1-\delta$, the generalization error of the PCNNs model satisfies:*

$$\mathbb{E}_{\mathcal{D}}\left[\ell(f_{pcnn})\right] \leq \hat{\mathcal{R}}_{\mathcal{S}}(f_{pcnn}) + \mathcal{O}\left(\sqrt{\frac{\mathcal{C}(f_{pcnn}) + \log(1/\delta)}{|\mathcal{S}_{train}|}}\right), \tag{17}$$

where the complexity measure $\mathcal{C}(f_{pcnn})$ is defined as:

$$\mathcal{C}(f_{pcnn}) = (1-\alpha)^2\mathcal{C}(f_{CNN}) + \alpha^2\lambda_{\max}(\mathcal{L}). \tag{18}$$

$\lambda_{\max}(\mathcal{L})$ is the largest eigenvalue of the graph Laplacian $\mathcal{L}$. For our generalized Hamming graph with parameter k, $\lambda_{\max}(\mathcal{L}) \leq 2k$, which leads to a controlled complexity measure even for large graph structures.

This bound demonstrates how PCNNs balance the complexity of the underlying CNN and the spectral properties of the graph, resulting in improved generalization particularly when the training data is limited.

Theorem 13 (Extrapolation Behavior)**.** *Let $f_{pcnn}(\boldsymbol{x})$ be the PCNNs model applied to an infinitely-wide two-layer CNN with ReLU activation trained using the squared loss. For any sequence $\boldsymbol{x}_0 \in \mathcal{V}$ and a neighboring sequence $\boldsymbol{x}_1 \in \mathcal{N}(\boldsymbol{x}_0)$, the change in the PCNNs output satisfies:*

$$\left|f_{pcnn}(\boldsymbol{x}_1) - f_{pcnn}(\boldsymbol{x}_0)\right| \leq \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d}\cdot(d+1) + \mathcal{O}\left(\frac{1}{\sqrt{n}}\right), \tag{19}$$

where $c_{\boldsymbol{x}_0,\boldsymbol{x}_1}$ is a constant depending on sequences $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, $\mathcal{N}(\boldsymbol{x}_0)$ is the neighborhood of $\boldsymbol{x}_0$ in $\mathcal{G}$, d is the graph degree and n is the width of NTK.

This theorem shows how the output changes between neighboring sequences. The bound depends on the degree of the graph and scales with sequence length, indicating smooth transitions between neighboring sequences even in potentially rugged landscapes.

Theorem 14 (Convergence Rate)**.** *Under the same assumptions as in Theorem 13, the difference between the normalized change in the PCNNs output when going from $\boldsymbol{x}_0$ to $\boldsymbol{x}_1$ and its limiting value satisfies:*

$$\left|\frac{f_{pcnn}(\boldsymbol{x}_1) - f_{pcnn}(\boldsymbol{x}_0)}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}\cdot(d+1)/d} - 1\right| \leq \frac{C}{\sqrt{n}}\left(1 + \frac{\sqrt{1-\alpha_{\min}^2}}{\alpha_{\min}}\right), \tag{20}$$

where C is a constant independent of NTK width n, $\alpha_{\min}$ is the minimum cosine similarity between $\boldsymbol{x}_0$ and its neighbors in $\mathcal{G}$, d is the maximum degree in $\mathcal{G}$, and $c_{\boldsymbol{x}_0,\boldsymbol{x}_1} = \mathbf{w}_{CNN}^{\top}\big(\phi_{CNN}(\boldsymbol{x}_1) - \phi_{CNN}(\boldsymbol{x}_0)\big)$.

This convergence rate result suggests that PCNNs can efficiently adapt to local variations in the protein fitness landscape, with the accuracy improving as the sequence length increases. The dependence on the graph degree and local similarity ($\alpha_{\min}$) reflects the importance of the local structure in the generalized Hamming graph.

5 Experiments

Below we demonstrate the advantages of incorporating PCNNs in protein optimization tasks.

Baseline Methods. We compare our approach with representative prior works that have been evaluated on GFP and AAV protein design tasks. These include GFlowNets (GFN-AL) [Jain et al., 2022], model-based adaptive sampling (CbAS) [Brookes et al., 2019], greedy search (AdaLead) [Sinai et al., 2020], Bayesian optimization (BO-qei) [Wilson et al., 2017], conservative model-based optimization (CoMs) [Trabucco et al., 2021], proximal exploration (PEX) [Ren et al., 2019], and GGS [Kirjner et al., 2023].

Implementation. For our predictor model, we employ a 1D convolutional neural network (CNN) architecture. To ensure a fair comparison, we use the same model architecture for baseline methods where possible. Then we use our PCNNs to get smoothed predictions. Based on our hyperparameter tuning, we set $k = 3$ for the Levenshtein distance threshold and $\alpha = 0.001$ for the propagation parameter. For compatibility with other sequence exploration methods such as GWG, we distil the smoothed predictions with a new 1D CNN. Notice that our method focuses specifically on the task of sequence-to-function modeling, with the goal of predicting the fitness value of a given protein sequence. In our experiments, we use a GWG sampling process with a temperature of 0.1, 15 rounds, and 100 proposal sequences per round, as we observed convergence for these settings.

5.1 Benchmark Tasks

We evaluate our proposed method on the same benchmark tasks as described by Kirjner et al. [2023], based on two well-studied protein systems: Green Fluorescent Protein (GFP) and Adeno-Associated Virus (AAV) [Sarkisyan et al., 2016, Bryant et al., 2021]. These systems were chosen due to the availability of large amounts of measurement data and their ability to demonstrate how prior methods fail to extrapolate to unseen regions of the fitness landscape. The protein optimization task involves using a starting (training) set of sequences and fitness labels to propose a new set of sequences with higher fitness. Following Kirjner et al. [2023], we consider two measures of task difficulty: (1) the minimum number of mutations required to achieve the highest known fitness, which assesses a method's exploration capability, and (2) the fitness range of the starting set of sequences, which tests a method's ability to learn from barely functional proteins and exploit limited knowledge to find beneficial mutations. To ensure comprehensive evaluations, we use the same "medium" and "hard" difficulty levels for both GFP and AAV tasks as defined by Kirjner et al. [2023]. The fitness gap for the hard setting is 7 mutations, and has a fitness range below the 30th percentile of the whole dataset.

5.2 In-silico Evaluation

Following prior works, we use a trained evaluator model as a proxy for real-world experimental validation. While transformer models such as TAPE have been popular, we notice superior performance from a simpler convolutional neural network (CNN) architecture, consistent with the findings of Dallago et al. [2021]. For each method, we generate 128 sample sequences and predict their fitness using the surrogate oracle. Additionally, we report the diversity and novelty metrics, as used in previous work. It is important to note that higher diversity and novelty are not necessarily equivalent to better performance but provide insights into the exploration and exploitation trade-offs of different methods.

Table 1: GFP optimization results. Bold indicates improvement with smoothing. In parentheses: std.

	Medium difficulty			Hard difficulty		
Method	Fitness	Diversity	Novelty	Fitness	Diversity	Novelty
GFN-AL	0.09 (0.1)	25.1 (0.5)	213 (2.2)	0.1 (0.2)	23.6 (1.0)	214 (4.2)
CbAS	0.14 (0.0)	9.7 (1.1)	7.2 (0.4)	0.18 (0.0)	9.6 (1.3)	7.8 (0.4)
AdaLead	0.56 (0.0)	3.5 (0.1)	2.0 (0.0)	0.18 (0.0)	5.6 (0.5)	2.8 (0.4)
BOqei	0.20 (0.0)	19.3 (0.0)	0.0 (0.0)	0.0 (0.5)	94.6 (71)	54.1 (81)
CoMS	0.0 (0.1)	133 (25)	192 (12)	0.0 (0.1)	144 (7.5)	201 (3.0)
PEX	0.47 (0.0)	3.0 (0.0)	1.4 (0.2)	0.0 (0.0)	3.0 (0.0	1.3 (0.3)
GGS	**0.76 (0.0)**	3.7 (0.2)	5.0 (0.0)	0.71 (0.0)	3.6 (0.1)	8.0 (0.0)
Ours	**0.87 (0.0)**	2.2 (0.0)	5.4 (0.0)	**0.76 (0.0)**	2.0 (0.0)	6.4 (0.0)

Table 2: AAV optimization results. Bold indicates improvement with smoothing. In parentheses: std.

	Medium difficulty			Hard difficulty		
Method	Fitness	Diversity	Novelty	Fitness	Diversity	Novelty
GFN-AL	0.2 (0.1)	9.6 (1.2)	19.4 (1.1)	0.1 (0.1)	11.6 (1.4)	19.6 (1.1)
CbAS	0.43 (0.0)	12.7 (0.7)	7.2 (0.4)	0.36 (0.0)	14.4 (0.7)	8.6 (0.5)
AdaLead	0.46 (0.0)	8.5 (0.8)	2.8 (0.4)	0.4 (0.0)	8.53 (0.1)	3.4 (0.5)
BOqei	0.38 (0.0)	15.22 (0.8)	0.0 (0.0)	0.32 (0.0)	17.9 (0.3)	0.0 (0.0)
CoMS	0.37 (0.1)	10.1 (5.9)	8.2 (3.5)	0.26 (0.0)	10.7 (3.5)	10.0 (2.8)
PEX	0.4 (0.0)	2.8 (0.0)	1.4 (0.2)	0.3 (0.0)	2.8 (0.0)	1.3 (0.3)
GGS	**0.51 (0.0)**	4.0 (0.2)	5.4 (0.5)	**0.44 (0.0)**	4.5 (0.5)	7.0 (0.0)
Ours	**0.53 (0.0)**	3.0 (0.2)	4.8 (0.5)	**0.47 (0.0)**	4.5 (0.5)	7.0 (0.0)

5.3 Results

Tables 1 and 2 summarize the results for the GFP and AAV tasks, respectively. We report the average metric across five random seeds, along with the standard deviation in parentheses. Our PCNNs method significantly outperforms all baselines, achieving the highest fitness scores on both tasks and across all difficulty levels. Notably, in the challenging hard difficulty setting for GFP, where most baselines struggle to surpass the fitness of the training set, our method consistently discovers sequences with improved fitness. We attribute the superior performance of PCNNs to their ability to efficiently leverage the local structure of the fitness landscape as predicted by our theoretical framework. By incorporating neighboring sequence information through the message passing step, PCNNs effectively smooth the learned fitness function in accordance with Lemma 8, reducing both ruggedness and noise with Lemma 9 to guide the exploration towards promising regions.

In addition to the high fitness scores, our method maintains competitive diversity and novelty metrics, indicating that it can explore the sequence space effectively without sacrificing fitness. The novelty of the sequences generated by PCNNs falls within the range of the mutational gap for each difficulty level, suggesting that our method strikes a balance between exploiting the learned fitness function and exploring novel sequences. These experimental results validate our theoretical framework, demonstrating that PCNNs can effectively navigate rugged fitness landscapes by leveraging their spectral properties, even with limited and noisy training data.

5.4 Validation of Spectral Measures

To validate our spectral measures for landscape characterization, we conduct simulation experiments using NK models with varying interaction parameters.

Simulation Setup. We generated artificial landscapes with controlled complexity using standard NK models with different K values ($K = 2, 4, 6$) and a modified Block-NK model featuring distinct functional domains. For each landscape type, we generated 1000 random sequences, measured fitness values according to NK rules, added realistic experimental noise, constructed neighborhood graphs (Hamming distance = 2), and calculated both traditional and our spectral measures. Each experiment was repeated five times to ensure statistical reliability.

Results. Table 3 shows comparative results between traditional ruggedness (R) and epistasis (E) measures versus our spectral counterparts.

Table 3: Comparison of traditional and spectral measures across different landscape types.

Model Type	Traditional R	Spectral R	Traditional E	Spectral E
Block-NK	1.166 ± 0.072	0.367 ± 0.013	0.500 ± 0.000	0.999 ± 0.000
NK-2	1.147 ± 0.116	0.357 ± 0.022	0.200 ± 0.000	0.998 ± 0.001
NK-4	1.210 ± 0.052	0.364 ± 0.036	0.400 ± 0.000	0.999 ± 0.000
NK-6	1.232 ± 0.022	0.367 ± 0.013	0.600 ± 0.000	0.999 ± 0.000

Our spectral measures demonstrate several key advantages over traditional approaches. The lower variance in spectral ruggedness (R) values across repeated experiments indicates greater robustness to noise, while the consistently high spectral epistasis (E) values reflect enhanced sensitivity in detecting complex interactions. Notably, our spectral ruggedness measure effectively filters spurious roughness, resulting in lower, more consistent R values across different landscape complexities. The spectral measures also show remarkable consistency across landscapes with different underlying mutation patterns and selection pressures, enabling more reliable comparison of fitness landscapes across diverse protein families.

5.5 Ablation Studies

To understand the contribution of each component in PCNNs, we conduct ablation studies to analyze the effects of key hyperparameters on model performance.

The Impact of Propagation Parameter. The propagation parameter α controls the balance between the CNN prediction and neighborhood averaging. Table 4 shows the impact of different α values on the performance of GFP (Medium difficulty). The results show that even a small amount of graph propagation ($\alpha = 0.001$) dramatically improves performance compared to a standard CNN. However, increasing α beyond this optimal value gradually reduces performance, suggesting that over-smoothing can diminish the model's ability to capture important local features.

Table 4: The impact of α (propagation parameter) on fitness.

α Value	Fitness
$\alpha = 0.0$	0.10
$\alpha = 0.001$	0.87
$\alpha = 0.01$	0.84
$\alpha = 0.1$	0.79

Table 5: The impact of Levenshtein distance threshold k on fitness.

Leven. Dist	Fitness
$k = 1$	0.65
$k = 2$	0.76
$k = 3$	0.87
$k = 4$	0.82

The Impact of Levenshtein Distance. The Levenshtein distance threshold k determines the neighborhood size for graph propagation. Since amino acid substitutions in protein space result in even Hamming distances only, we employ Levenshtein distance for more effective neighborhood definition. Table 5 shows how different k values affect performance of GFP (Medium difficulty). Performance peaks at $k = 3$, beyond which we observe a drop. This aligns with our theoretical framework, suggesting that most informative epistatic interactions in these protein systems occur within a relatively local sequence neighborhood, while more distant relationships inject more noise than signal.

6 Conclusions

We have presented a novel modeling for protein fitness landscapes. By bridging the gap between abstract models of fitness landscapes and practical machine learning approaches, our framework provides a rigorous foundation for understanding and improving computational protein engineering methods. The integration of spectral graph theory offers new insights into the challenges of model architecture design, sample efficiency, and generalization in the context of sparse and rugged fitness landscapes. Our analysis of Propagational Convolutional Neural Networks (PCNNs) demonstrates how this theoretical framework can inform the development and understanding of advanced machine learning models for protein engineering tasks. The generalization bounds and extrapolation behavior results we derived provide a quantitative basis for assessing and improving performance.

Acknowledgements

This work is funded by the CSIRO Science Digital and Advanced Engineering Biology Future Science Platforms, and was supported by resources and expertise provided by CSIRO IMT Scientific Computing.

References

Erik C Alley, Grigory Khimulya, Surojit Biswas, Mohammed AlQuraishi, and George M Church. Unified rational protein engineering with sequence-based deep representation learning. *Nature methods*, 16 (12):1315–1322, 2019.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Surojit Biswas, Grigory Khimulya, Ethan C Alley, Kevin M Esvelt, and George M Church. Low-n protein engineering with data-efficient deep learning. *Nature methods*, 18(4):389–396, 2021.

David Brookes, Hahnbeom Park, and Jennifer Listgarten. Conditioning by adaptive sampling for robust design. In *International conference on machine learning*, pages 773–782. PMLR, 2019.

Andries E Brouwer, Willem H Haemers, Andries E Brouwer, and Willem H Haemers. *Distance-regular graphs*. Springer, 2012.

Drew H Bryant, Ali Bashir, Sam Sinai, Nina K Jain, Pierce J Ogden, Patrick F Riley, George M Church, Lucy J Colwell, and Eric D Kelsic. Deep diversification of an aav capsid protein by machine learning. *Nature Biotechnology*, 39(6):691–696, 2021.

Christian Dallago, Jody Mou, Kadina E Johnston, Bruce J Wittmann, Nicholas Bhattacharya, Samuel Goldman, Ali Madani, and Kevin K Yang. Flip: Benchmark tasks in fitness landscape inference for proteins. *bioRxiv*, pages 2021–11, 2021.

Aryan Deshwal, Syrine Belakaria, and Janardhan Rao Doppa. Bayesian optimization over hybrid spaces. In *International Conference on Machine Learning*, pages 2632–2643. PMLR, 2021.

Simon S Du, Kangcheng Hou, Russ R Salakhutdinov, Barnabas Poczos, Ruosong Wang, and Keyulu Xu. Graph neural tangent kernel: Fusing graph neural networks with graph kernels. *Advances in neural information processing systems*, 32, 2019.

Richard J Fox, S Christopher Davis, Emily C Mundorff, Lisa M Newman, Vesna Gavrilovic, Steven K Ma, Loleta M Chung, Charlene Ching, Sarena Tam, Sheela Muley, et al. Improving catalytic function by prosar-driven enzyme evolution. *Nature biotechnology*, 25(3):338–344, 2007.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Moksh Jain, Emmanuel Bengio, Alex Hernandez-Garcia, Jarrid Rector-Brooks, Bonaventure F. P. Dossou, Chanakya Ajit Ekbote, Jie Fu, Tianyu Zhang, Michael Kilgour, Dinghuai Zhang, Lena Simine, Payel Das, and Yoshua Bengio. Biological sequence design with GFlowNets. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 9786–9801. PMLR, 17–23 Jul 2022.

Stuart A Kauffman and Edward D Weinberger. The nk model of rugged fitness landscapes and its application to maturation of the immune response. *Journal of theoretical biology*, 141(2):211–245, 1989.

Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi S Jaakkola, Regina Barzilay, and Ila R Fiete. Improving protein optimization with smoothed fitness landscapes. In *The Twelfth International Conference on Learning Representations*, 2023.

Roelof Koekoek and Rene F Swarttouw. The askey-scheme of hypergeometric orthogonal polynomials and its q-analogue. *arXiv preprint math/9602214*, 1996.

Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In T. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems*, volume 14. MIT Press, 2001.

Changyong Oh, Jakub Tomczak, Efstratios Gavves, and Max Welling. Combinatorial bayesian optimization using the graph cartesian product. *Advances in Neural Information Processing Systems*, 32, 2019.

Jie Ren, Peter J Liu, Emily Fertig, Jasper Snoek, Ryan Poplin, Mark DePristo, Joshua V Dillon, and Balaji Lakshminarayanan. Likelihood ratios for out-of-distribution detection. In *Advances in Neural Information Processing Systems*, pages 14707–14718, 2019.

Philip A Romero, Andreas Krause, and Frances H Arnold. Navigating the protein fitness landscape with gaussian processes. *Proceedings of the National Academy of Sciences*, 110(3):E193–E201, 2013.

Karen S Sarkisyan, Dmitry A Bolotin, Margarita V Meer, Dinara R Usmanova, Alexander S Mishin, George V Sharonov, Dmitry N Ivankov, Nina G Bozhanova, Mikhail S Baranov, Onuralp Soylemez, et al. Local fitness landscape of the green fluorescent protein. *Nature*, 533(7603):397–401, 2016.

Sam Sinai, Richard Wang, Alexander Whatley, Stewart Slocum, Elina Locane, and Eric D Kelsic. Adalead: A simple and robust adaptive greedy search algorithm for sequence design. *arXiv preprint arXiv:2010.02141*, 2020.

Daniel A Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 563–568, 2008.

Brandon Trabucco, Aviral Kumar, Xinyang Geng, and Sergey Levine. Conservative objective models for effective offline model-based optimization. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 10358–10368. PMLR, 18–24 Jul 2021.

Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17:395–416, 2007.

James T Wilson, Riccardo Moriconi, Frank Hutter, and Marc Peter Deisenroth. The reparameterization trick for acquisition functions. *arXiv preprint arXiv:1712.00424*, 2017.

Zachary Wu, Shawn Beng Jie Kan, Russell D Lewis, Blake J Wittmann, and Frances H Arnold. Machine learning-assisted directed protein evolution with combinatorial libraries. *Proceedings of the National Academy of Sciences*, 116(18):8852–8858, 2019.

Kevin K Yang, Yuxin Chen, Alycia Lee, and Yisong Yue. Batched stochastic bayesian optimization via combinatorial constraints design. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3410–3419. PMLR, 2019a.

Kevin K Yang, Zachary Wu, and Frances H Arnold. Machine learning in protein engineering. *arXiv preprint arXiv:1811.10775*, 2019b.

Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bernhard Sch"olkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16:321–328, 2004.

A Proofs and Derivations

In this section, we provide detailed proofs and derivations for the key theoretical results presented in the main paper. We follow the order of presentation in the main text, providing additional context and intermediate steps where necessary.

A.1 Proof of Theorem: Eigenvalues of the Generalized Hamming Graph

Proof. The proof proceeds in several steps:

1) First, recall that our generalized Hamming graph is distance-regular. This means that the number of vertices at distance j from a given vertex v depends only on j, not on the choice of v.

2) For distance-regular graphs, the adjacency matrix $\boldsymbol{A}$ can be expressed as a polynomial in the distance matrices $\boldsymbol{A}_j$:

$$\boldsymbol{A} = \sum_{j=0}^{k} c_j \boldsymbol{A}_j, \tag{21}$$

where $\boldsymbol{A}_j$ is the matrix with $(\boldsymbol{A}_j)_{uv} = 1$ if $d(u,v) = j$ and 0 otherwise, and c_j are constants.

3) The matrices $\boldsymbol{A}_j$ form an association scheme, which means they commute and can be simultaneously diagonalized. Their common eigenvectors are given by the columns of the matrix P of dual eigenvectors.

4) The eigenvalues of $\boldsymbol{A}_j$ are given by the j-th column of P. Denote these by $P_i(j)$.

5) Therefore, the eigenvalues of $\boldsymbol{A}$ are given by:

$$\lambda_i = \sum_{j=0}^{k} c_j P_i(j). \tag{22}$$

6) For the generalized Hamming graph, the coefficients c_j are precisely $K_j(N)$, the size of the j-th shell in the Hamming graph.

7) Finally, for the Hamming scheme, the dual eigenvectors $P_i(j)$ are given by the q-Krawtchouk polynomials.

Combining these facts leads to the stated theorem. □

Corollary 15. *The multiplicity of λ_i is given by:*

$$m_i = \frac{20^N - 1}{20^N \sum_{j=0}^{N} Q_i(j)^2 / K_j(N)}, \tag{23}$$

where Q_i are the dual polynomials to P_i.

Proof. This follows from the general theory of association schemes. The multiplicity formula is a consequence of the orthogonality relations between P and Q:

$$\sum_{i=0}^{N} P_i(j) Q_i(l) = 20^N \delta_{jl} \tag{24}$$

and

$$\sum_{j=0}^{N} K_j(N) P_i(j) P_l(j) = 20^N m_i \delta_{il}. \tag{25}$$

Combining these relations and solving for m_i yields the stated formula. □

A.2 Proof of Theorem: Eigenvectors of the Generalized Hamming Graph

Proof. 1) First, recall that for a regular graph, the eigenvectors of the graph Laplacian $\mathcal{L}$ are the same as the eigenvectors of the adjacency matrix $\boldsymbol{A}$.

2) We have established that the eigenvalues of $\boldsymbol{A}$ are given by the q-Krawtchouk polynomials. The corresponding eigenvectors are the columns of the character table of the Hamming scheme.

3) For the Hamming scheme on $\mathbb{Z}_{20}^N$, the characters are precisely the generalized Walsh functions as defined in the theorem.

4) To verify that these functions are eigenvectors, we compute:

$$(\boldsymbol{A}\psi_{\mathcal{S}})(\boldsymbol{x}) = \sum_{\boldsymbol{y}:H(\boldsymbol{x},\boldsymbol{y})\leq k} \psi_{\mathcal{S}}(\boldsymbol{y}) \tag{26}$$

$$= \sum_{\boldsymbol{y}:H(\boldsymbol{x},\boldsymbol{y})\leq k} \prod_{i\in\mathcal{S}} \frac{1}{\sqrt{19}} \sum_{a=1}^{20} \omega^{a y_i}. \tag{27}$$

For vertices $\boldsymbol{y}$ with Hamming distance j from $\boldsymbol{x}$, the character values differ exactly at j positions. Let $\mathcal{D}_j(\boldsymbol{x})$ be the set of vertices at distance j from $\boldsymbol{x}$. Then:

$$(\boldsymbol{A}\psi_{\mathcal{S}})(\boldsymbol{x}) = \sum_{j=0}^{k} \sum_{\boldsymbol{y}\in\mathcal{D}_j(\boldsymbol{x})} \psi_{\mathcal{S}}(\boldsymbol{y}) \tag{28}$$

$$= \sum_{j=0}^{k} \sum_{\boldsymbol{y}\in\mathcal{D}_j(\boldsymbol{x})} \prod_{i\in\mathcal{S}} \frac{1}{\sqrt{19}} \sum_{a=1}^{20} \omega^{a y_i}. \tag{29}$$

For each position $i \in \mathcal{S}$, if $y_i \neq x_i$, the term contributes a factor of $\frac{1}{\sqrt{19}} \sum_{a=1}^{20} \omega^a = 0$. If $y_i = x_i$, the term contributes $\frac{1}{\sqrt{19}} \sum_{a=1}^{20} \omega^{a x_i} = \frac{20}{\sqrt{19}}$.

Let $\mathcal{S}_j$ be the number of positions in $\mathcal{S}$ that differ between $\boldsymbol{x}$ and $\boldsymbol{y}$ when $H(\boldsymbol{x},\boldsymbol{y}) = j$. Then:

$$(\boldsymbol{A}\psi_{\mathcal{S}})(\boldsymbol{x}) = \sum_{j=0}^{k} K_j(N) \cdot P_{|\mathcal{S}|}(j) \cdot \psi_{\mathcal{S}}(\boldsymbol{x}) \tag{30}$$

$$= \left(\sum_{j=0}^{k} K_j(N) \cdot P_{|\mathcal{S}|}(j) \right) \cdot \psi_{\mathcal{S}}(\boldsymbol{x}) \tag{31}$$

$$= \lambda_{|\mathcal{S}|} \psi_{\mathcal{S}}(\boldsymbol{x}), \tag{32}$$

where $P_{|\mathcal{S}|}(j)$ is the $|\mathcal{S}|$-th q-Krawtchouk polynomial evaluated at j.

5) This confirms that the generalized Walsh functions are indeed eigenvectors of $\boldsymbol{A}$, and thus of $\mathcal{L}$, with eigenvalues $\lambda_{|\mathcal{S}|}$. □

A.3 Proof of Theorem: Spectral Decomposition of the Fitness Function

Proof. 1) We have established that the generalized Walsh functions $\{\psi_{\mathcal{S}}\}$ form an orthonormal basis for functions on our generalized Hamming graph. This means that any function f on this space can be expressed as a linear combination of these basis functions.

2) The coefficients $a_{\mathcal{S}}$ in this linear combination are given by the inner products of f with the basis functions:

$$a_{\mathcal{S}} = \langle f, \psi_{\mathcal{S}} \rangle = \frac{1}{20^N} \sum_{\boldsymbol{x} \in \mathcal{V}} f(\boldsymbol{x}) \overline{\psi_{\mathcal{S}}(\boldsymbol{x})}, \tag{33}$$

where the sum is over all sequences $\boldsymbol{x}$ in the protein sequence space $\mathcal{V}$, and the bar denotes complex conjugation.

3) To prove uniqueness, assume there are two different decompositions:

$$f(\boldsymbol{x}) = \sum_{\mathcal{S}} a_{\mathcal{S}} \psi_{\mathcal{S}}(\boldsymbol{x}) = \sum_{\mathcal{S}} b_{\mathcal{S}} \psi_{\mathcal{S}}(\boldsymbol{x}). \tag{34}$$

Taking the inner product with $\psi_{\mathcal{T}}$ on both sides:

$$\langle f, \psi_{\mathcal{T}} \rangle = \sum_{\mathcal{S}} a_{\mathcal{S}} \langle \psi_{\mathcal{S}}, \psi_{\mathcal{T}} \rangle = \sum_{\mathcal{S}} b_{\mathcal{S}} \langle \psi_{\mathcal{S}}, \psi_{\mathcal{T}} \rangle. \tag{35}$$

By orthonormality of the Walsh functions, $\langle \psi_{\mathcal{S}}, \psi_{\mathcal{T}} \rangle = \delta_{\mathcal{S},\mathcal{T}}$, so:

$$a_{\mathcal{T}} = b_{\mathcal{T}}. \tag{36}$$

This holds for all $\mathcal{T}$, proving uniqueness.

4) The completeness of the Walsh function basis ensures that any fitness function can be represented in this form.

□

A.4 Proof of Theorem: Spectral Approximation for Generalized Hamming Graphs

Proof. 1) We start by considering the sampled graph $\mathcal{G}_s$ as a sparsification of the full graph $\mathcal{G}$. The theory of graph sparsification [Spielman and Srivastava, 2008] tells us that for a suitable sampling procedure, the Laplacians of $\mathcal{G}$ and $\mathcal{G}_s$ are spectrally close.

2) Specifically, if edges are sampled independently with probability p, then with probability at least $1 - \delta$:

$$(1 - \epsilon)\mathcal{L} \preceq \mathcal{L}_s \preceq (1 + \epsilon)\mathcal{L}, \tag{37}$$

where $\preceq$ denotes the semidefinite ordering.

3) This spectral approximation implies that for any vector $\boldsymbol{x}$:

$$(1 - \epsilon)\boldsymbol{x}^T \mathcal{L} \boldsymbol{x} \leq \boldsymbol{x}^T \mathcal{L}_s \boldsymbol{x} \leq (1 + \epsilon)\boldsymbol{x}^T \mathcal{L} \boldsymbol{x}. \tag{38}$$

4) Now, consider the Walsh functions $\psi_{\mathcal{S}}$ and $\psi_{\mathcal{S}}^{(s)}$ on the full and sampled graphs respectively. These are eigenvectors of $\mathcal{L}$ and $\mathcal{L}_s$:

$$\mathcal{L}\psi_{\mathcal{S}} = \lambda_{\mathcal{S}}\psi_{\mathcal{S}}, \quad \mathcal{L}_s \psi_{\mathcal{S}}^{(s)} = \lambda_{\mathcal{S}}^{(s)} \psi_{\mathcal{S}}^{(s)}. \tag{39}$$

5) Applying the spectral approximation to $\psi_{\mathcal{S}}$ yields:

$$(1 - \epsilon)\lambda_{\mathcal{S}} \leq \lambda_{\mathcal{S}}^{(s)} \leq (1 + \epsilon)\lambda_{\mathcal{S}}. \tag{40}$$

6) Now, consider the spectral decomposition of a function f on both graphs:

$$f = \sum_{\mathcal{S}} a_{\mathcal{S}} \psi_{\mathcal{S}}, \quad f^{(s)} = \sum_{\mathcal{S}} a_{\mathcal{S}}^{(s)} \psi_{\mathcal{S}}^{(s)}. \tag{41}$$

7) The coefficients $a_{\mathcal{S}}$ and $a_{\mathcal{S}}^{(s)}$ are given by inner products:

$$a_{\mathcal{S}} = \langle f, \psi_{\mathcal{S}} \rangle, \quad a_{\mathcal{S}}^{(s)} = \langle f^{(s)}, \psi_{\mathcal{S}}^{(s)} \rangle. \tag{42}$$

8) Using the spectral approximation and the Cauchy-Schwarz inequality, we can show:

$$(1-\epsilon)|a_{\mathcal{S}}| \leq |a_{\mathcal{S}}^{(s)}| \leq (1+\epsilon)|a_{\mathcal{S}}|. \tag{43}$$

9) The required sampling rate follows from the analysis of Spielman and Srivastava [2008], taking into account the structure of our generalized Hamming graph.

This completes the proof of the spectral approximation theorem. □

A.5 Proof of Theorem: Connection to the NK Model

Proof. 1) Recall that in the NK model, the fitness function is defined as:

$$f_{NK}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N} f_i(x_i, x_{i+1}, ..., x_{i+K}), \tag{44}$$

where indices are taken modulo N.

2) Each component f_i can be expressed in terms of Walsh functions on its $K+1$ inputs:

$$f_i(x_i, ..., x_{i+K}) = \sum_{\mathcal{T}\subseteq\{0,...,K\}} a_{\mathcal{T},i}\psi_{\mathcal{T}}(x_i, ..., x_{i+K}). \tag{45}$$

3) Substituting this into the NK model equation yields:

$$f_{NK}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\mathcal{T}\subseteq\{0,...,K\}} a_{\mathcal{T},i}\psi_{\mathcal{T}}(x_i, ..., x_{i+K}). \tag{46}$$

4) Now, we can rewrite this in terms of Walsh functions on the full sequence:

$$f_{NK}(\boldsymbol{x}) = \frac{1}{N}\sum_{i=1}^{N}\sum_{\mathcal{T}\subseteq\{0,...,K\}} a_{\mathcal{T},i}\psi_{\mathcal{S}(\mathcal{T},i)}(\boldsymbol{x}), \tag{47}$$

where $\mathcal{S}(\mathcal{T}, i) = \{i + j \mod N : j \in \mathcal{T}\}$.

5) Rearranging the sums yields:

$$f_{NK}(\boldsymbol{x}) = \sum_{\mathcal{S}\subseteq\{1,...,N\}} \left(\frac{1}{N}\sum_{i=1}^{N} a_{\mathcal{T}(\mathcal{S},i),i}\right)\psi_{\mathcal{S}}(\boldsymbol{x}), \tag{48}$$

where $\mathcal{T}(\mathcal{S}, i) = \{j - i \mod N : j \in \mathcal{S}\} \cap \{0, ..., K\}$.

6) This is exactly the spectral decomposition of f_{NK} on the generalized Hamming graph, with coefficients:

$$a_{\mathcal{S}} = \frac{1}{N}\sum_{i=1}^{N} a_{\mathcal{T}(\mathcal{S},i),i}. \tag{49}$$

7) By construction, $\mathcal{T}(\mathcal{S}, i)$ is empty if $|\mathcal{S}| > K + 1$, which proves the first part of the theorem.

8) The second part follows directly from the equation in step 6.

□

A.6 Proof of Theorem: PCNNs Approximation

Proof. 1) Recall that the PCNNs output is defined as:

$$\tilde{f}(\boldsymbol{x}_i) = (1-\alpha) f_{\text{CNN}}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} f_{\text{CNN}}(\boldsymbol{x}_j). \tag{50}$$

2) Denote the neighborhood average term as:

$$g(\boldsymbol{x}_i) = \frac{1}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} f_{\text{CNN}}(\boldsymbol{x}_j). \tag{51}$$

3) Then we can write:

$$\tilde{f} = (1-\alpha) f_{\text{CNN}} + \alpha g. \tag{52}$$

4) On the sampled graph $\mathcal{G}_s$, we have:

$$\tilde{f}_s = (1-\alpha) f_{\text{CNN}} + \alpha g_s, \tag{53}$$

where g_s is computed over the sampled neighborhood.
5) The difference between $\tilde{f}$ and $\tilde{f}_s$ is thus:

$$\tilde{f} - \tilde{f}_s = \alpha(g - g_s). \tag{54}$$

6) We can bound this difference using concentration inequalities. For each vertex, the sampled neighborhood average $g_s(\boldsymbol{x}_i)$ is an unbiased estimator of $g(\boldsymbol{x}_i)$, and by Hoeffding's inequality:

$$P(|g_s(\boldsymbol{x}_i) - g(\boldsymbol{x}_i)| > t) \leq 2\exp(-2p|\mathcal{N}(\boldsymbol{x}_i)|t^2). \tag{55}$$

7) Taking a union bound over all vertices and setting $t = \sqrt{\frac{\log(n/\delta)}{2p|\mathcal{N}(\boldsymbol{x}_i)|}}$, we get with probability at least $1-\delta$:

$$\|g - g_s\|_\infty \leq \sqrt{\frac{\log(n/\delta)}{2p|\mathcal{N}(\boldsymbol{x}_i)|}}. \tag{56}$$

8) Since $\|\cdot\|_2 \leq \sqrt{n}\|\cdot\|_\infty$ for n-dimensional vectors, we have:

$$\|\tilde{f} - \tilde{f}_s\|_2 \leq \alpha\sqrt{n}\sqrt{\frac{\log(n/\delta)}{2p|\mathcal{N}(\boldsymbol{x}_i)|}}. \tag{57}$$

9) Note that $|\mathcal{N}(\boldsymbol{x}_i)| = \Theta(|\mathcal{V}|)$ for our generalized Hamming graph, which yields the $\mathcal{O}\left(\alpha\sqrt{\frac{\log(n/\delta)}{p|\mathcal{V}|}}\right)$ term.

10) The $\epsilon\|\tilde{f}\|_2$ term comes from the approximation of f_{CNN} on the sampled graph, which we can bound using the spectral approximation result from Theorem 4.

Combining these bounds completes the proof. □

A.7 Proof of Proposition: Equivalence of CNN and PCNNs Solutions

Proof. 1) Start by defining the Neural Tangent Kernel (NTK) for a standard CNN:

$$K_{\mathrm{CNN}}(\boldsymbol{x}, \boldsymbol{x}') = \langle \phi_{\mathrm{CNN}}(\boldsymbol{x}), \phi_{\mathrm{CNN}}(\boldsymbol{x}') \rangle, \tag{58}$$

where ϕ_{CNN} is the feature map of the CNN.

2) For a PCNNs, the kernel is defined as:

$$K_{\mathrm{PCNN}}(\boldsymbol{x}, \boldsymbol{x}') = (1-\alpha)^2 K_{\mathrm{CNN}}(\boldsymbol{x}, \boldsymbol{x}') + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x})||\mathcal{N}(\boldsymbol{x}')|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}')} K_{\mathrm{CNN}}(\boldsymbol{u}, \boldsymbol{v}), \tag{59}$$

where $\mathcal{N}(\boldsymbol{x})$ is the neighborhood of $\boldsymbol{x}$ in the graph.

3) The minimum RKHS-norm solution for kernel regression with kernel $\boldsymbol{K}$ is given by:

$$f^*(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i), \tag{60}$$

where $\boldsymbol{\alpha} = (\boldsymbol{K} + \lambda \boldsymbol{I})^{-1}\boldsymbol{y}$, with $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$, $\boldsymbol{y}$ is the vector of target values, and λ is a regularization parameter.

4) For the CNN, this solution can be written as:

$$f^*_{\mathrm{CNN}}(\boldsymbol{x}) = \boldsymbol{w}^{*\top}_{\mathrm{CNN}} \phi_{\mathrm{CNN}}(\boldsymbol{x}), \tag{61}$$

where $\boldsymbol{w}^*_{\mathrm{CNN}} = \sum_{i=1}^{n} \alpha_i \phi_{\mathrm{CNN}}(\boldsymbol{x}_i)$.

5) For the PCNNs, we can write:

$$\begin{aligned} f^*_{\mathrm{PCNN}}(\boldsymbol{x}) &= \sum_{i=1}^{n} \alpha_i K_{\mathrm{PCNN}}(\boldsymbol{x}, \boldsymbol{x}_i) && (62) \\ &= (1-\alpha)^2 \sum_{i=1}^{n} \alpha_i K_{\mathrm{CNN}}(\boldsymbol{x}, \boldsymbol{x}_i) + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x})|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} \sum_{i=1}^{n} \alpha_i \frac{1}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_i)} K_{\mathrm{CNN}}(\boldsymbol{u}, \boldsymbol{v}). && (63) \end{aligned}$$

6) The first term is exactly $(1-\alpha)^2 f^*_{\mathrm{CNN}}(\boldsymbol{x})$. The second term can be rewritten as:

$$\frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x})|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} f^*_{\mathrm{CNN}}(\boldsymbol{u}) \tag{64}$$

7) Therefore, we can express f^*_{PCNN} in terms of f^*_{CNN}:

$$f^*_{\mathrm{PCNN}}(\boldsymbol{x}) = (1-\alpha)^2 f^*_{\mathrm{CNN}}(\boldsymbol{x}) + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x})|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} f^*_{\mathrm{CNN}}(\boldsymbol{u}). \tag{65}$$

8) This is exactly the form of the PCNNs prediction, with f^*_{CNN} in place of f_{CNN}. Thus, $\boldsymbol{w}^*_{\mathrm{PCNN}} = \boldsymbol{w}^*_{\mathrm{CNN}}$.

9) For a GNN, the kernel would involve message passing operations that are not present in the CNN or PCNNs kernels. Therefore, in general, $\boldsymbol{w}^*_{\mathrm{GNN}} \neq \boldsymbol{w}^*_{\mathrm{CNN}} = \boldsymbol{w}^*_{\mathrm{PCNN}}$.

This completes the proof of the proposition. □

A.8 Proof of Lemma: GNTK perspective of PCNNs

Proof. 1) Recall that the Neural Tangent Kernel (NTK) for a neural network $f(\boldsymbol{x};\boldsymbol{\theta})$ is defined as:

$$g(\boldsymbol{x},\boldsymbol{x}') = \langle \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x};\boldsymbol{\theta}), \nabla_{\boldsymbol{\theta}} f(\boldsymbol{x}';\boldsymbol{\theta})\rangle. \tag{66}$$

2) For a PCNNs, we have:

$$f_{\text{PCNN}}(\boldsymbol{x}_i) = (1-\alpha) f_{\text{CNN}}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_i)} f_{\text{CNN}}(\boldsymbol{x}_u). \tag{67}$$

3) Taking the gradient with respect to the parameters $\boldsymbol{\theta}$:

$$\nabla_{\boldsymbol{\theta}} f_{\text{PCNN}}(\boldsymbol{x}_i) = (1-\alpha)\nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_i)} \nabla_{\boldsymbol{\theta}} f_{\text{CNN}}(\boldsymbol{x}_u). \tag{68}$$

4) Now, compute the GNTK for PCNNs:

$$\begin{aligned}
g_G(\boldsymbol{x}_i,\boldsymbol{x}_j) &= \langle \nabla_{\boldsymbol{\theta}} f_{\text{PCNN}}(\boldsymbol{x}_i), \nabla_{\boldsymbol{\theta}} f_{\text{PCNN}}(\boldsymbol{x}_j)\rangle && (69)\\
&= (1-\alpha)^2 g_{\text{CNN}}(\boldsymbol{x}_i,\boldsymbol{x}_j) && (70)\\
&+ \frac{\alpha(1-\alpha)}{|\mathcal{N}(\boldsymbol{x}_j)|} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_j)} g_{\text{CNN}}(\boldsymbol{x}_i,\boldsymbol{x}_v) && (71)\\
&+ \frac{\alpha(1-\alpha)}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_i)} g_{\text{CNN}}(\boldsymbol{x}_u,\boldsymbol{x}_j) && (72)\\
&+ \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)||\mathcal{N}(\boldsymbol{x}_j)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_i)} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_j)} g_{\text{CNN}}(\boldsymbol{x}_u,\boldsymbol{x}_v). && (73)
\end{aligned}$$

5) As $\alpha \to 1$, the first three terms vanish, and one is left with:

$$g_G(\boldsymbol{x}_i,\boldsymbol{x}_j) = \frac{1}{|\mathcal{N}(\boldsymbol{x}_i)||\mathcal{N}(\boldsymbol{x}_j)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_i)} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_j)} g_{\text{CNN}}(\boldsymbol{x}_u,\boldsymbol{x}_v). \tag{74}$$

This completes the proof of the lemma. □

A.9 Proof of Theorem: Generalization Bound for PCNNs

Proof. 1) We start with the standard generalization bound based on Rademacher complexity:

$$\mathbb{E}_{\mathcal{D}}[\ell(f)] \le \hat{\mathcal{R}}_{\mathcal{S}}(f) + \mathcal{R}_n(\mathcal{H}) + \sqrt{\frac{\log(1/\delta)}{2n}}, \tag{75}$$

where $\hat{\mathcal{R}}_{\mathcal{S}}(f)$ is the empirical risk, and $\mathcal{R}_n(\mathcal{H})$ is the Rademacher complexity of the hypothesis class.
2) For PCNNs, we have:

$$f_{\text{pcnn}}(\boldsymbol{x}_i) = (1-\alpha) f_{\text{CNN}}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|} \sum_{\boldsymbol{x}_j\in\mathcal{N}(\boldsymbol{x}_i)} f_{\text{CNN}}(\boldsymbol{x}_j), \tag{76}$$

where α is the propagation parameter and $\mathcal{N}(\boldsymbol{x}_i)$ is the neighborhood of $\boldsymbol{x}_i$ in the graph.

3) The Rademacher complexity of this class of functions can be bounded as follows:

$$\mathcal{R}_n(\mathcal{H}) = \mathbb{E}_\sigma\left[\sup_{f\in\mathcal{H}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i f(\boldsymbol{x}_i)\right] \tag{77}$$

$$\leq (1-\alpha)\mathcal{R}_n(\mathcal{H}_{CNN}) + \alpha\mathcal{R}_n(\mathcal{H}_{GC}), \tag{78}$$

where $\mathcal{H}_{CNN}$ is the hypothesis class of the CNN and $\mathcal{H}_{GC}$ is the hypothesis class of the graph convolution operation.

4) The Rademacher complexity of the CNN part, $\mathcal{R}_n(\mathcal{H}_{CNN})$, can be bounded using standard techniques for neural networks. For the graph convolution part, we can use spectral graph theory:

$$\mathcal{R}_n(\mathcal{H}_{GC}) \leq \frac{1}{\sqrt{n}}\sqrt{\lambda_{\max}(L)}, \tag{79}$$

where L is the graph Laplacian and $\lambda_{\max}(L)$ is its largest eigenvalue.

5) Combining these bounds, we can define our complexity measure $\mathcal{C}(f_{\text{pcnn}})$ as:

$$\mathcal{C}(f_{\text{pcnn}}) = (1-\alpha)^2\mathcal{C}(f_{\text{CNN}}) + \alpha^2\lambda_{\max}(L), \tag{80}$$

where $\mathcal{C}(f_{\text{CNN}})$ is a suitable complexity measure for the CNN part.

6) This gives us the following bound on the Rademacher complexity:

$$\mathcal{R}_n(\mathcal{H}) \leq \frac{1}{\sqrt{n}}\sqrt{\mathcal{C}(f_{\text{pcnn}})}. \tag{81}$$

7) Substituting this back into our generalization bound yields:

$$\mathbb{E}_{\mathcal{D}}[\ell(f_{\text{pcnn}})] \leq \hat{\mathcal{R}}_{\mathcal{S}}(f_{\text{pcnn}}) + \frac{1}{\sqrt{n}}\sqrt{\mathcal{C}(f_{\text{pcnn}})} + \sqrt{\frac{\log(1/\delta)}{2n}}. \tag{82}$$

8) Using the inequality $\sqrt{a+b} \leq \sqrt{a} + \sqrt{b}$, we can simplify this to:

$$\mathbb{E}_{\mathcal{D}}[\ell(f_{\text{pcnn}})] \leq \hat{\mathcal{R}}_{\mathcal{S}}(f_{\text{pcnn}}) + \mathcal{O}\left(\sqrt{\frac{\mathcal{C}(f_{\text{pcnn}}) + \log(1/\delta)}{n}}\right). \tag{83}$$

This completes the proof of the generalization bound for PCNNs. □

A.10 Proof of Theorem: Extrapolation Behavior

Proof. 1) Recall that for PCNNs, we have:

$$f_{\text{pcnn}}(\boldsymbol{x}) = (1-\alpha)f_{\text{cnn}}(\boldsymbol{x}) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x})|}\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} f_{\text{cnn}}(\boldsymbol{u}). \tag{84}$$

2) For an infinitely-wide two-layer CNN with ReLU activation, we can use the Neural Tangent Kernel (NTK) theory. In the infinite-width limit, the NTK becomes deterministic and fixed during training:

$$f_{\text{cnn}}(\boldsymbol{x}) = \sum_{i=1}^{n} \alpha_i K(\boldsymbol{x}, \boldsymbol{x}_i) \tag{85}$$

, where $K(\boldsymbol{x}, \boldsymbol{x}')$ is the NTK, and α_i are coefficients determined by training.

3) The NTK for a two-layer ReLU network is:

$$K(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{x}^\top \boldsymbol{x}' + \frac{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}{2\pi}(\sin\theta + (\pi - \theta)\cos\theta), \tag{86}$$

where $\theta = \arccos\left(\frac{\boldsymbol{x}^\top \boldsymbol{x}'}{\|\boldsymbol{x}\|\|\boldsymbol{x}'\|}\right)$.

4) For neighboring sequences $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, we can approximate:

$$|K(\boldsymbol{x}_1, \boldsymbol{x}_i) - K(\boldsymbol{x}_0, \boldsymbol{x}_i)| \leq c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot \frac{1}{d}, \tag{87}$$

where $c_{\boldsymbol{x}_0,\boldsymbol{x}_1}$ is a constant that depends on $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, and d is the dimension of the sequence space.

5) Now, consider the difference in PCNNs outputs:

$$|f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)| \leq (1-\alpha)|f_{\text{cnn}}(\boldsymbol{x}_1) - f_{\text{cnn}}(\boldsymbol{x}_0)| \tag{88}$$

$$+ \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_1)|}\left|\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_1)} f_{\text{cnn}}(\boldsymbol{u}) - \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_0)} f_{\text{cnn}}(\boldsymbol{v})\right|. \tag{89}$$

6) For the first term:

$$|f_{\text{cnn}}(\boldsymbol{x}_1) - f_{\text{cnn}}(\boldsymbol{x}_0)| \leq \sum_{i=1}^{n} |\alpha_i||K(\boldsymbol{x}_1, \boldsymbol{x}_i) - K(\boldsymbol{x}_0, \boldsymbol{x}_i)| \leq \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d}\sum_{i=1}^{n}|\alpha_i|. \tag{90}$$

7) For the second term, note that $\mathcal{N}(\boldsymbol{x}_1)$ and $\mathcal{N}(\boldsymbol{x}_0)$ differ by at most one element. Therefore:

$$\left|\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_1)} f_{\text{cnn}}(\boldsymbol{u}) - \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_0)} f_{\text{cnn}}(\boldsymbol{v})\right| \leq \max_{u}|f_{\text{cnn}}(\boldsymbol{u})| \leq \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d}\sum_{i=1}^{n}|\alpha_i|. \tag{91}$$

8) Combining these results yields:

$$|f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)| \leq \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d}\sum_{i=1}^{n}|\alpha_i| \cdot (1-\alpha+\alpha) = \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d}\sum_{i=1}^{n}|\alpha_i|. \tag{92}$$

9) From NTK theory, we know that $\sum_{i=1}^{n}|\alpha_i| = \mathcal{O}(d)$ with high probability. Additionally, there is a $\mathcal{O}(1/\sqrt{n})$ term due to finite-width effects.

This gives us the final bound

$$|f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)| \leq \frac{c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d} \cdot (d+1) + O\left(\frac{1}{\sqrt{n}}\right) \tag{93}$$

completing the proof. □

A.11 Proof of Theorem: Convergence Rate

Proof. 1) We start with the PCNNs output difference from the previous theorem:

$$f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0) = (1-\alpha)(f_{\text{cnn}}(\boldsymbol{x}_1) - f_{\text{cnn}}(\boldsymbol{x}_0)) + \alpha(g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0)), \tag{94}$$

where $g(\boldsymbol{x}) = \frac{1}{|\mathcal{N}(\boldsymbol{x})|}\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} f_{\text{cnn}}(\boldsymbol{u})$.

2) In the infinite-width limit, we can express f_{cnn} using the NTK:

$$f_{\text{cnn}}(\boldsymbol{x}) = \mathbf{w}_{\text{CNN}}^\top \phi_{\text{CNN}}(\boldsymbol{x}), \tag{95}$$

where $\phi_{\text{CNN}}(\boldsymbol{x})$ is the feature map corresponding to the NTK.

3) The difference in CNN outputs can be written as:

$$f_{\text{cnn}}(\boldsymbol{x}_1) - f_{\text{cnn}}(\boldsymbol{x}_0) = \mathbf{w}_{\text{CNN}}^\top (\phi_{\text{CNN}}(\boldsymbol{x}_1) - \phi_{\text{CNN}}(\boldsymbol{x}_0)) = c_{\boldsymbol{x}_0,\boldsymbol{x}_1}. \tag{96}$$

4) For the neighborhood average term is:

$$g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0) = \frac{1}{|\mathcal{N}(\boldsymbol{x}_1)|} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_1)} f_{\text{cnn}}(\boldsymbol{u}) - \frac{1}{|\mathcal{N}(\boldsymbol{x}_0)|} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_0)} f_{\text{cnn}}(\boldsymbol{v}) \tag{97}$$

$$= \frac{1}{d} \sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x}_1)} \mathbf{w}_{\text{CNN}}^\top \phi_{\text{CNN}}(\boldsymbol{u}) - \frac{1}{d} \sum_{\boldsymbol{v}\in\mathcal{N}(\boldsymbol{x}_0)} \mathbf{w}_{\text{CNN}}^\top \phi_{\text{CNN}}(\boldsymbol{v}). \tag{98}$$

5) The difference between these sums can be bounded as follows:

$$|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0)| \leq \frac{2}{d} \|\mathbf{w}_{\text{CNN}}\|_2 \max_{u,v} \|\phi_{\text{CNN}}(\boldsymbol{u}) - \phi_{\text{CNN}}(\boldsymbol{v})\|_2. \tag{99}$$

6) For neighboring sequences, we can approximate:

$$\|\phi_{\text{CNN}}(\boldsymbol{u}) - \phi_{\text{CNN}}(\boldsymbol{v})\|_2 \approx c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot \frac{1}{d}. \tag{100}$$

7) Substituting this back yields:

$$|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0)| \leq \frac{2c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{d^2} \|\mathbf{w}_{\text{CNN}}\|_2. \tag{101}$$

8) Now, consider the normalized difference:

$$\left| \frac{f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1 \right| \tag{102}$$

$$= \left| \frac{(1-\alpha)c_{\boldsymbol{x}_0,\boldsymbol{x}_1} + \alpha(g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0))}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1 \right| \tag{103}$$

$$\leq \left| \frac{(1-\alpha)c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1 \right| + \left| \frac{\alpha(g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0))}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} \right|. \tag{104}$$

9) The first term can be bounded:

$$\left| \frac{(1-\alpha)c_{\boldsymbol{x}_0,\boldsymbol{x}_1}}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1 \right| = \left| \frac{(1-\alpha)d}{d+1} - 1 \right| \leq \frac{1}{d+1}. \tag{105}$$

10) For the second term, using our bound on $|g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0)|$ yields:

$$\left| \frac{\alpha(g(\boldsymbol{x}_1) - g(\boldsymbol{x}_0))}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} \right| \leq \frac{2\alpha\|\mathbf{w}_{\text{CNN}}\|_2}{d(d+1)}. \tag{106}$$

11) From NTK theory, we know that $\|\mathbf{w}_{\text{CNN}}\|_2 = \mathcal{O}(\sqrt{d})$ with high probability. Additionally, there is a $\mathcal{O}(1/\sqrt{n})$ term due to finite-width effects.

12) Combining these results yields:

$$\left|\frac{f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1\right| \leq \frac{1}{d+1} + \frac{2\alpha\sqrt{d}}{d(d+1)} + O\left(\frac{1}{\sqrt{n}}\right). \tag{107}$$

13) The term $\frac{2\alpha\sqrt{d}}{d(d+1)}$ can be further bounded using the minimum cosine similarity $\alpha_{\min}$ between $\boldsymbol{x}_0$ and its neighbors:

$$\frac{2\alpha\sqrt{d}}{d(d+1)} \leq \frac{2\sqrt{1-\alpha_{\min}^2}}{\alpha_{\min}(d+1)}. \tag{108}$$

14) Combining all terms and simplifying them yields:

$$\left|\frac{f_{\text{pcnn}}(\boldsymbol{x}_1) - f_{\text{pcnn}}(\boldsymbol{x}_0)}{c_{\boldsymbol{x}_0,\boldsymbol{x}_1} \cdot (d+1)/d} - 1\right| \leq \frac{C}{\sqrt{n}}\left(1 + \frac{\sqrt{1-\alpha_{\min}^2}}{\alpha_{\min}}\right), \tag{109}$$

where C is a constant independent of n.

This completes the proof of the convergence rate theorem. □

A.12 Proof of Lemma: Smoothness and Ruggedness

Proof. 1) Lipschitz continuity: Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be two vertices in the Hamming graph. We need to show that: $|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{y})| \leq L'd(\boldsymbol{x}, \boldsymbol{y})$, where $d(\boldsymbol{x}, \boldsymbol{y})$ is the Hamming distance.

$$\begin{aligned}|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{y})| &= |(1-\alpha)(f_{\text{CNN}}(\boldsymbol{x}) - f_{\text{CNN}}(\boldsymbol{y})) + \alpha(f_{avg}(\boldsymbol{x}) - f_{avg}(\boldsymbol{y}))| \\ &\leq (1-\alpha)|f_{\text{CNN}}(\boldsymbol{x}) - f_{\text{CNN}}(\boldsymbol{y})| + \alpha|f_{avg}(\boldsymbol{x}) - f_{avg}(\boldsymbol{y})|.\end{aligned}$$

For f_{CNN}, we know $|f_{\text{CNN}}(\boldsymbol{x}) - f_{\text{CNN}}(\boldsymbol{y})| \leq Ld(\boldsymbol{x}, \boldsymbol{y})$ due to L-Lipschitz continuity.

For f_{avg}, we have:

$$\begin{aligned}|f_{avg}(\boldsymbol{x}) - f_{avg}(\boldsymbol{y})| &= \left|\frac{1}{|\mathcal{N}(\boldsymbol{x})|}\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} f_{\text{CNN}}(\boldsymbol{u}) - \frac{1}{|\mathcal{N}(\boldsymbol{y})|}\sum_{v\in\mathcal{N}(\boldsymbol{y})} f_{\text{CNN}}(\boldsymbol{v})\right| \\ &\leq \frac{1}{|\mathcal{N}(\boldsymbol{x})|}\sum_{\boldsymbol{u}\in\mathcal{N}(\boldsymbol{x})} |f_{\text{CNN}}(\boldsymbol{u}) - f_{\text{CNN}}(v(\boldsymbol{u}))|,\end{aligned}$$

where $v(\boldsymbol{u})$ is the corresponding neighbor of y.

Since $d(u, v(\boldsymbol{u})) \leq d(\boldsymbol{x}, \boldsymbol{y}) + 1$, we have: $|f_{avg}(\boldsymbol{x}) - f_{avg}(\boldsymbol{y})| \leq L(d(\boldsymbol{x}, \boldsymbol{y}) + 1)$.

Combining these yields:

$$\begin{aligned}|\tilde{f}(\boldsymbol{x}) - \tilde{f}(\boldsymbol{y})| &\leq (1-\alpha)Ld(\boldsymbol{x}, \boldsymbol{y}) + \alpha L(d(\boldsymbol{x}, \boldsymbol{y}) + 1) \\ &= Ld(\boldsymbol{x}, \boldsymbol{y}) + \alpha L.\end{aligned}$$

Therefore, $\tilde{f}$ is L'-Lipschitz continuous with $L' = L + \alpha L/d_{max}$, where d_{max} is the maximum Hamming distance. Since $d_{max} \geq 1$, we have $L' \leq L$.

2) Spectral Ruggedness: The spectral ruggedness measure R is defined as:

$$R(f) = \frac{\sum_{k=1}^{n} k^2|a_k|^2}{\sum_{k=0}^{n}|a_k|^2},$$

where a_k are the coefficients in the spectral decomposition of f.

For $\tilde{f}$, we have: $\tilde{f} = (1-\alpha)f_{\text{CNN}} + \alpha f_{avg}$.

Let a_k, b_k, and c_k be the spectral coefficients of $\tilde{f}$, f_{CNN}, and f_{avg} respectively. Then $a_k = (1-\alpha)b_k + \alpha c_k$
$|a_k|^2 \le ((1-\alpha)|b_k| + \alpha|c_k|)^2 \le (1-\alpha)|b_k|^2 + \alpha|c_k|^2$ (by convexity).

Therefore:

$$\begin{aligned} R(\tilde{f}) &= \frac{\sum_{k=1}^{n} k^2 |a_k|^2}{\sum_{k=0}^{n} |a_k|^2} \\ &\le \frac{(1-\alpha)\sum_{k=1}^{n} k^2|b_k|^2 + \alpha \sum_{k=1}^{n} k^2|c_k|^2}{(1-\alpha)\sum_{k=0}^{n}|b_k|^2 + \alpha\sum_{k=0}^{n}|c_k|^2} \\ &\le (1-\alpha)R(f_{\text{CNN}}) + \alpha R(f_{avg}). \end{aligned}$$

This completes the proof. □

A.13 Proof of Lemma: Noise Reduction and Epistasis Estimation

Proof. 1) For PCNNs, we have:

$$\tilde{f}(\boldsymbol{x}_i) = (1-\alpha)f_{\text{CNN}}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} f_{\text{CNN}}(\boldsymbol{x}_j). \tag{110}$$

2) Substituting $f_{\text{CNN}} = f_{true} + \varepsilon$ yields:

$$\begin{aligned} \tilde{f}(\boldsymbol{x}_i) &= (1-\alpha)(f_{true}(\boldsymbol{x}_i) + \varepsilon_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} (f_{true}(\boldsymbol{x}_j) + \varepsilon_j) \quad (111) \\ &= (1-\alpha)f_{true}(\boldsymbol{x}_i) + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} f_{true}(\boldsymbol{x}_j) + (1-\alpha)\varepsilon_i + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} \varepsilon_j. \quad (112) \end{aligned}$$

3) The variance affects only the noise terms:

$$\text{Var}(\tilde{f}(\boldsymbol{x}_i)) = \text{Var}\left((1-\alpha)\varepsilon_i + \frac{\alpha}{|\mathcal{N}(\boldsymbol{x}_i)|}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} \varepsilon_j\right). \tag{113}$$

4) Since the noise is independent with $\text{Cov}(\varepsilon_j, \varepsilon_k) = 0$ for $j \neq k$ we have:

$$\begin{aligned} \text{Var}(\tilde{f}(\boldsymbol{x}_i)) &= (1-\alpha)^2\text{Var}(\varepsilon_i) + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|^2}\sum_{\boldsymbol{x}_j \in \mathcal{N}(\boldsymbol{x}_i)} \text{Var}(\varepsilon_j) \quad (114) \\ &= (1-\alpha)^2\sigma^2 + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|^2} \cdot |\mathcal{N}(\boldsymbol{x}_i)| \cdot \sigma^2 \quad (115) \\ &= \left((1-\alpha)^2 + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|}\right)\sigma^2. \quad (116) \end{aligned}$$

5) Now, consider the estimated spectral epistasis measure $\hat{E}(f)$, which quantifies the contributions of higher-order interactions:

$$\hat{E}(f) = \frac{\sum_{|\mathcal{S}|\ge 2} |\hat{a}_{\mathcal{S}}|^2}{\sum_{\mathcal{S}} |\hat{a}_{\mathcal{S}}|^2}, \tag{117}$$

where $\hat{a}_{\mathcal{S}}$ are the estimated spectral coefficients.

6) The variance of this estimate depends on the variance of the function values. Using the delta method:

$$\text{Var}(\hat{E}(f)) \approx \nabla\hat{E}(f)^T \cdot \text{Var}(f) \cdot \nabla\hat{E}(f). \tag{118}$$

7) Given the linear relationship between $\tilde{f}$ and f_{CNN}, and the form of their variances we have:

$$\text{Var}(\hat{E}(\tilde{f})) \leq \left(\frac{\text{Var}(\tilde{f})}{\text{Var}(f_{\text{CNN}})}\right) \cdot \text{Var}(\hat{E}(f_{\text{CNN}})) \tag{119}$$

$$= \left((1-\alpha)^2 + \frac{\alpha^2}{|\mathcal{N}(\boldsymbol{x}_i)|}\right) \cdot \text{Var}(\hat{E}(f_{\text{CNN}})). \tag{120}$$

8) This result shows that PCNNs reduce noise in the spectral epistasis estimate by a factor that depends on both the propagation parameter α and the neighborhood size $|\mathcal{N}(\boldsymbol{x}_i)|$. The larger the neighborhood, the greater the noise reduction. □

Checklist

The checklist follows the references. For each question, choose your answer from the three possible options: Yes, No, Not Applicable. You are encouraged to include a justification to your answer, either by referencing the appropriate section of your paper or providing a brief inline description (1-2 sentences). Please do not modify the questions. Note that the Checklist section does not count towards the page limit. Not including the checklist in the first submission won't result in desk rejection, although in such case we will ask you to upload it during the author response period and include it in camera ready (if accepted).

In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all models and algorithms presented, check if you include:
   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]
   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]
   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]
2. For any theoretical claim, check if you include:
   (a) Statements of the full set of assumptions of all theoretical results. [Yes]
   (b) Complete proofs of all theoretical results. [Yes]
   (c) Clear explanations of any assumptions. [Yes]
3. For all figures and tables that present empirical results, check if you include:
   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [No]
   (b) All the training details (*e.g.*, data splits, hyperparameters, how they were chosen). [Not]
   (c) A clear definition of the specific measure or statistics and error bars (*e.g.*, with respect to the random seed after running experiments multiple times). [No]
   (d) A description of the computing infrastructure used. (*e.g.*, type of GPUs, internal cluster, or cloud provider). [No]
4. If you are using existing assets (*e.g.*, code, data, models) or curating/releasing new assets, check if you include:
   (a) Citations of the creator If your work uses existing assets. [Yes]
   (b) The license information of the assets, if applicable. [No]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Yes]
   (e) Discussion of sensible content if applicable, *e.g.*, personally identifiable information or offensive content. [Not Applicable]
5. If you used crowdsourcing or conducted research with human subjects, check if you include:

(a) The full text of instructions given to participants and screenshots. [Not Applicable]

(b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

(c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]