DISCRETE INPUT LANDSCAPES OF NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Turning neural-network predictions into decisions often requires downstream optimization over the network's inputs. Local search can offer a particularly efficient approach to such optimization when the feasible set is discrete and one can move between feasible solutions using simple local changes. However, theoretical understanding of local-search performance over the input of trained neural-network landscapes remains limited. In this paper, we study how training shapes the discrete input landscapes of multilayer perceptrons over cardinality-constrained binary sets, equipped with the natural exchange neighborhood. In particular, we study the smoothness of such landscapes in the infinite-width neural tangent kernel (NTK) regime (Lee et al., 2020) through their autocorrelation (Weinberger, 1990), a notion particularly relevant to local-search performance. Under natural assumptions, we show that training produces smooth interpolations of the training labels, with expected autocorrelation approaching one as the input dimension grows. We also show that filtering out components of the trained network's output associated with smaller eigenvalues of the random-walk matrix on the exchange neighborhood can increase autocorrelation while approximately preserving the output values. Motivated by the latter result, we propose a Filtered Local Search (FLS) method and show that it improves average solution quality over ordinary local search both in controlled experiments close to our theoretical model and in applications with neural architectures and feasible sets beyond it.

1 INTRODUCTION

In many predict-then-optimize settings, a neural network is trained from data to learn a function $\widehat{f}(x)$ that predicts the cost of a decision x. The learned function is then used to select a new decision by solving a downstream optimization problem $\min_{x\in\mathcal{X}} \widehat{f}(x)$, where $\mathcal{X}$ is a feasible set. For example, Patel et al. (2022) learn neural approximations of second-stage costs in stochastic facility location and use them to choose which facilities to open. Similarly, Kaleem et al. (2026) replace the vehicle-routing cost term in a location-routing model with a trained neural network and solve the resulting model to choose depot locations and customer assignments. In combinatorial auctions, Weissteiner & Seuken (2020) train neural networks to predict bidders' values for item bundles and maximize total predicted value over feasible allocations to select the next bundles to query. However, the resulting downstream optimization problems often lead to computationally demanding mixed-integer or nonlinear formulations (Fischetti & Jo, 2018; Schweidtmann & Mitsos, 2019).

When downstream optimization is over a discrete feasible set $\mathcal{X}$ where one can move between feasible solutions using simple local changes, local search might offer a computationally efficient way to find good solutions through local improvements. Empirical studies have shown that local search can perform well on objectives constructed from trained neural networks. For example, Lombardi et al. (2017) obtained high-quality job assignments through local search on neural predictions of processor efficiency, while Gelman et al. (2021) obtained a protein variant with substantially stronger binding than the naturally occurring reference protein through hill climbing on an objective based on neural-network predictions.

There have also been attempts to establish theoretical guarantees for local search over neural-network objectives. Candogan & Foussoul (2026) design architectures with exchange properties generalizing those of M-convex functions, which yield global-optimality and running-time guar-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

antees for a local-descent algorithm. By contrast, Moon et al. (2019) and Liu et al. (2022) derive approximation guarantees for local search applied to standard neural networks that depend on how close the learned objective is to being submodular. While these results identify structural properties that yield local-search guarantees, they do not establish how training standard neural networks produces such properties or, more generally, affects local-search performance.

This paper takes a step toward bridging this gap. Given a discrete feasible set $\mathcal{X}$, a function f over $\mathcal{X}$, and a neighborhood structure $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ specifying the neighbors of each feasible solution, the triple $(f, \mathcal{X}, \mathcal{N})$ defines a landscape, which we refer to as the landscape of f over $\mathcal{X}$ equipped with $\mathcal{N}$. Studying properties of this landscape, such as how the values of f vary between neighboring solutions, can help understand local-search performance. We study the input landscapes of trained multilayer perceptrons over cardinality-constrained binary feasible sets $\mathcal{X} = \{x \in \{0,1\}^n \mid \sum_{i=1}^n x_i = k\}$ equipped with the natural exchange neighborhood structure $\mathcal{N}(x) = \{y \in \mathcal{X} \mid \|x - y\|_1 = 2\}$. These sets arise in several important optimization problems, including k-median (Arya et al., 2004), sensor placement (Krause et al., 2008), and influence maximization (Kempe et al., 2015). Other important feasible sets, including the binary hypercube, which arises for instance in MAX-CUT (Karp, 1972), MAX-SAT (Johnson, 1974), and binary quadratic optimization (Hammer & Rudeanu, 1968), and cardinality-budget sets, which arise for instance in feature selection (Das & Kempe, 2018), maximum coverage (Chekuri & Kumar, 2004), and assortment optimization (Rusmevichientong et al., 2010), can also be written as unions of polynomially many such sets and hence optimized over efficiently by optimizing over each component separately.

To this end, we study the smoothness of such landscapes through their autocorrelation. Autocorrelation is a classical landscape measure (Stadler, 1996), defined as the correlation between values at neighboring feasible solutions. Larger autocorrelation indicates smoother landscapes and hence less pronounced fluctuations between neighboring values, which can create shallow local optima that trap local search. Higher autocorrelation has been associated with better local-search performance in diverse applications. For example, Waeselynck et al. (2007) find that autocorrelation helps predict simulated-annealing performance in software test generation, and Muñoz-Herrera & Suchan (2022) report similar evidence for vehicle routing. Benedettini et al. (2013) associate higher autocorrelation with greater local-search success in Boolean-network design. In traveling-salesperson optimization, Angel & Zissimopoulos (2000) relate differences in autocorrelation between neighborhoods to their observed local-search performance. We also present experiments showing a strong positive association between autocorrelation and local-search performance on trained neural-network landscapes.

A standard approach widely used to understand different aspects of neural-network behavior is to consider the infinite-width limit (Jacot et al., 2018; Lee et al., 2020). Under suitable initialization and training, the network behaves as a kernel model with a kernel known as the neural tangent kernel (NTK), making the evolution of the network output analytically tractable. This approach has been used, for example, to study how prediction error varies with training-set size (Bordelon et al., 2020), to explain training difficulties in physics-informed neural networks (Wang et al., 2022), and to investigate the role of learned features in adversarial robustness (Tsilivis & Kempe, 2022). We use the NTK framework to study the input landscapes of trained neural networks and their autocorrelation. We now describe our contributions in more detail.

1.1 Contributions

Neural-Network Training Bias Toward Smooth Input Landscapes. We study the smoothness of the input landscapes of trained multilayer perceptrons over cardinality-constrained binary feasible sets $\mathcal{X}$ equipped with the natural exchange neighborhood structure $\mathcal{N}$, in the infinite-width limit (Lee et al., 2020). Under independent Gaussian initialization and gradient-flow training on the squared loss, the networks behave in this limit as kernel models with fixed neural tangent kernels (NTKs), allowing us to characterize their output f_∞ at convergence. We measure the smoothness of such landscapes through their autocorrelation $\rho_\infty = \frac{\langle \widetilde{f}_\infty, P\widetilde{f}_\infty \rangle}{\|\widetilde{f}_\infty\|^2}$, where $\widetilde{f}_\infty$ is f_∞ with its average over $\mathcal{X}$ subtracted and P is the transition matrix of the random walk that moves to a uniformly sampled neighbor. Higher autocorrelation indicates a smoother landscape and is empirically associated with better local-search performance. We establish the following two theoretical results.

(i) High autocorrelation bias. Our first result shows that f_∞ has high autocorrelation under natural assumptions. Training therefore tends to produce smooth landscapes favorable for local search.

Specifically, we assume that polynomially many training inputs are sampled uniformly without replacement and that k/n stays bounded away from zero and one and show that, for sufficiently large n, $\mathbb{E}[\rho_\infty] \geq 1 - \frac{C}{n}$, where the expectation is over the training inputs and initialization, and C is a constant independent of n and the labels. Thus, the expected autocorrelation approaches its maximum possible limit of one as the problem size grows.

(ii) Spectral bias. Our second result shows, roughly speaking, that we can further increase the autocorrelation of f_∞ using a suitable spectral filter, while approximately preserving its values. In particular, we relate the autocorrelation ρ_∞ to the spectral decomposition of f_∞ on the eigenspaces of P, showing that removing components associated with smaller eigenvalues can increase autocorrelation. For activations with bounded derivatives through order $q \geq 2$, the expected squared error from retaining only the first $R+1$ eigenspaces, ordered by decreasing eigenvalue, is bounded by a constant times $(R+1)^{-(q-1)}$. The constant depends on the network and a cost associated with fitting the labels. When this cost is controlled, the bound shows that removing higher-level components can give an approximation of f_∞ with higher autocorrelation and small expected mean-square error. This motivates our Filtered Local Search (FLS) method, which we discuss below.

Filtered Local Search. We propose Filtered Local Search (FLS), a two-phase procedure that applies best-improvement local search first to Pf_∞ and then to f_∞, starting the second phase from the solution found in the first. Multiplication by P reduces high-level components, providing a simple version of the filtering motivated above. This can help the search reach a promising basin before the second phase refines the solution on the original objective. As discussed below, we evaluate FLS both in controlled settings close to our theoretical model and in applications beyond it. The reported results show improvements in solution quality over ordinary best-improvement local search.

Controlled Experiments. We first conduct controlled experiments to assess our main claims empirically. In particular, we train 375 neural networks on datasets generated from a diverse set of target functions, including quadratic functions, Ising functions, and NK landscapes, across different input dimensions and training-set sizes. (i) We study the relationship between autocorrelation and local-search performance on the trained neural-network landscapes and find a strong positive association, with a median Spearman correlation of 0.80. (ii) We study the autocorrelation and spectral bias of the trained networks. We find a small average gap to the largest possible autocorrelation, consistent with our autocorrelation result. We also find that most output variance is concentrated in the first two spectral levels on average, consistent with our spectral bias result. (iii) We compare our filtered local-search procedure with ordinary local search on the trained neural-network landscapes, using identical starting solutions. Our procedure achieves higher average solution quality.

Applications. We further evaluate our filtered local-search procedure in applications with neural-network architectures, training procedures, and feasible sets beyond our theoretical setting. We consider the capacitated facility-location problem studied by Patel et al. (2022), where their neural networks predict service costs and we optimize facility-opening decisions. We also consider production scheduling, where the networks of Ghasemi et al. (2023) predict the makespan of production schedules and we optimize the order in which machines process operations. Using their released trained networks, we compare FLS with ordinary local search and, for facility location, the Gurobi MILP solver in terms of solution quality and runtime. In both applications, FLS improves average solution quality over ordinary local search. In facility location, it also obtains better solutions than Gurobi under matched optimization-time budgets.

2 Preliminaries

We first define the feasible set we consider and its neighborhood structure. We then describe the neural-network architecture we study and its training procedure and recall results describing its behavior in the infinite-width limit. Finally, we recall the notion of autocorrelation as a measure of the smoothness of the resulting input landscapes.

2.1 Feasible Set and Neighborhood Structure

We consider the cardinality-constrained binary feasible set

$$\mathcal{X} = \{x \in \{0,1\}^n : \textstyle\sum_{i=1}^n x_i = k\}\,, \tag{1}$$

where k is an integer satisfying $1 \leq k \leq n-1$.[1] This feasible set arises in a wide range of applications, including k-median (Arya et al., 2004), sensor placement (Krause et al., 2008), and influence maximization (Kempe et al., 2015). We define a neighborhood structure over $\mathcal{X}$, where for each $x \in \mathcal{X}$, its set of neighbors by

$$\mathcal{N}(x) = \{y \in \mathcal{X} \mid \|x - y\|_1 = 2\}. \tag{2}$$

Thus, each neighbor of x is obtained by exchanging one selected coordinate of x with one unselected coordinate.

We define the neighborhood graph associated with $\mathcal{X}$ and $\mathcal{N}$ as the undirected graph $G = (\mathcal{X}, E)$ with edge set $E = \{\{x, y\} : x \in \mathcal{X},\ y \in \mathcal{N}(x)\}$. Note that G is regular with degree $D = k(n-k)$.

2.2 Neural-Network Model and Infinite-Width Limit

We consider a fully connected feedforward network $f_\theta : \mathbb{R}^n \to \mathbb{R}$ with $L - 1 \geq 1$ hidden layers of widths $m_1, \ldots, m_{L-1}$, defined by

$$f_\theta(x) = \frac{W^{(L)}}{\sqrt{m_{L-1}}}\sigma\Big(\frac{W^{(L-1)}}{\sqrt{m_{L-2}}}\sigma\Big(\cdots\sigma\Big(\frac{W^{(1)}x}{\sqrt{n}} + \beta b^{(1)}\Big)\cdots\Big) + \beta b^{(L-1)}\Big) + \beta b^{(L)},$$

where $\theta = \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$, $\beta > 0$ is fixed, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonpolynomial, Lipschitz, and twice differentiable activation function with bounded second derivative, applied entrywise.[2]

At initialization θ_0, all parameter entries are drawn independently from $\mathcal{N}(0, 1)$. We train the network on $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N}$, with pairwise distinct inputs $x_i \in \mathcal{X}$ and targets $y_i \in \mathbb{R}$, by minimizing the empirical squared loss $\mathcal{L}(\theta) = \frac{1}{2N}\sum_{i=1}^{N}(f_\theta(x_i) - y_i)^2$ using gradient flow, $\frac{d}{dt}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t)$.[3]

We take the infinite-width limit by letting $m_1, \ldots, m_{L-1}$ tend sequentially to infinity while keeping L, n, $\mathcal{T}$, and β fixed. The initial network output f_{θ_0} converges in distribution to a centered Gaussian process $f_0 \sim \mathcal{GP}(0, \Sigma)$ (Jacot et al., 2018; Lee et al., 2018). At every fixed training time t, the network output f_{θ_t} converges in distribution to the output f_t of a kernel method with initial output f_0 and kernel Θ (called the neural tangent kernel, or NTK) (Jacot et al., 2018; Lee et al., 2020),

$$f_t(x) = f_0(x) + \mathbf{k}(x)^\top \mathbf{K}^{-1}\left(I - e^{-t\mathbf{K}/N}\right)(\mathbf{y} - \mathbf{f}_0), \qquad x \in \mathbb{R}^n, \tag{3}$$

where $\mathbf{y} = (y_i)_{i=1}^{N}$, $\mathbf{f}_0 = (f_0(x_i))_{i=1}^{N}$, $\mathbf{K}_{ij} = \Theta(x_i, x_j)$, and $\mathbf{k}(x) = (\Theta(x, x_i))_{i=1}^{N}$.[4] The covariance $\Sigma = \Sigma^{(L)}$ and the NTK $\Theta = \Theta^{(L)}$ are given, for $x, x' \in \mathbb{R}^n$, by the recursions

$$\begin{aligned} \Sigma^{(1)}(x, x') &= \tfrac{x^\top x'}{n} + \beta^2, & \Theta^{(1)}(x, x') &= \tfrac{x^\top x'}{n} + \beta^2, \\ \Sigma^{(\ell)}(x, x') &= \mathbb{E}[\sigma(U_\ell)\sigma(V_\ell)] + \beta^2, & \Theta^{(\ell)}(x, x') &= \Sigma^{(\ell)}(x, x') + \mathbb{E}[\sigma'(U_\ell)\sigma'(V_\ell)]\Theta^{(\ell-1)}(x, x'), \end{aligned} \tag{4}$$

where, for $\ell = 2, \ldots, L$, $(U_\ell, V_\ell) \sim \mathcal{N}\Big(0, \begin{pmatrix} \Sigma^{(\ell-1)}(x,x) & \Sigma^{(\ell-1)}(x,x') \\ \Sigma^{(\ell-1)}(x',x) & \Sigma^{(\ell-1)}(x',x') \end{pmatrix}\Big)$, and σ' denotes the derivative of σ. See Appendix A for further details and precise statements of these convergence results.

In the remainder of the paper, we consider this network in its infinite-width limit, with output f_t given by (3).

2.3 Autocorrelation and Landscape Smoothness

Let $f_\infty(x) = \lim_{t\to\infty} f_t(x)$ denote the output of the trained network, where f_t is given by (3). The function f_∞ induces a discrete input landscape over the cardinality-constrained binary feasible

[1] The spectral framework we develop in this paper extends seamlessly to intersections of disjoint blockwise cardinality constraints $\sum_{i\in I_b} x_i = k_b$, where $I_1, \ldots, I_B$ partition $[n]$. For simplicity of exposition, we focus here on a single cardinality constraint.

[2] This is known as the NTK parameterization (Lee et al., 2020) and ensures that the network output and its tangent kernel remain finite as the hidden-layer widths tend to infinity.

[3] This is the continuous-time analogue of full-batch gradient descent and is a standard theoretical model of wide-network training (Jacot et al., 2018; Lee et al., 2020).

[4] The nonpolynomiality of σ guarantees that $\mathbf{K}$ is invertible (Carvalho et al., 2025).

set $\mathcal{X}$ (1) equipped with the neighborhood structure $\mathcal{N}$ (2). Such a landscape is said to be smooth if neighboring solutions tend to have similar values. We study the smoothness of this landscape through its autocorrelation, given by

$$\rho_\infty = \mathrm{Corr}(f_\infty(X), f_\infty(Y)) \,. \tag{5}$$

Here, X is sampled uniformly from $\mathcal{X}$, Y is sampled uniformly from $\mathcal{N}(X)$, and Corr denotes Pearson correlation. Autocorrelation is widely used as a measure of landscape smoothness, with larger values empirically associated with better local-search performance (Weinberger, 1990; Benedettini et al., 2013). We will also present experiments showing a strong positive association between autocorrelation and local-search performance on trained neural-network landscapes.

3 Neural-Network Training Bias Toward Smooth Input Landscapes

In this section, we establish high-autocorrelation and spectral bias results for the trained network output f_∞ over $\mathcal{X}$ equipped with $\mathcal{N}$. To this end, in Section 3.1, we connect the autocorrelation ρ_∞ to the spectral content of f_∞ in the eigenspaces of the random-walk matrix on the neighborhood graph G. To characterize this spectral content, we need to understand how the NTK matrix acts on these eigenspaces, which we study in Section 3.2. Finally, in Section 3.3, we establish our results.

3.1 A Spectral Characterization of Autocorrelation

Let $P \in \mathbb{R}^{|\mathcal{X}|\times|\mathcal{X}|}$ denote the transition matrix of the simple random walk on G, defined by $P_{xy} = \frac{1}{D}\mathbf{1}\{y \in \mathcal{N}(x)\}$. The matrix P is real and symmetric and hence admits an orthogonal eigenspace decomposition. We first characterize its eigenvalues and eigenspaces explicitly.

The neighborhood graph G is isomorphic to the so-called Johnson graph $J(n,k)$, whose vertices are the k-subsets of $[n]$ and whose edges connect subsets that differ by one exchange. Let $s = \min(k, n-k)$. It is known that the adjacency matrix of the Johnson graph $J(n,k)$ admits the orthogonal eigenspace decomposition $\bigoplus_{r=0}^{s} \mathcal{E}_r$, with corresponding eigenvalues and dimensions $\alpha_r = D - r(n-r+1)$ and $d_r = \dim \mathcal{E}_r = \binom{n}{r} - \binom{n}{r-1}$, where $\binom{n}{-1} = 0$ (Brouwer & Haemers, 2012). Since P is the adjacency matrix of G divided by D, it admits the same orthogonal eigenspace decomposition, with corresponding eigenvalues $\lambda_r = 1 - \frac{r(n-r+1)}{D}$, $r = 0, \dots, s$.

Define the centered output $\widetilde{f}_\infty(x) = f_\infty(x) - |\mathcal{X}|^{-1}\sum_{z\in\mathcal{X}} f_\infty(z)$ and equip functions on $\mathcal{X}$ with the inner product $\langle u, v\rangle = |\mathcal{X}|^{-1}\sum_{x\in\mathcal{X}} u(x)v(x)$. Let Π_r denote the orthogonal projection onto $\mathcal{E}_r$. The autocorrelation ρ_∞ (5) can then be written as

$$\rho_\infty = \frac{\langle \widetilde{f}_\infty, P\widetilde{f}_\infty\rangle}{\|\widetilde{f}_\infty\|^2} = \frac{\sum_{r=1}^{s}\lambda_r\|\Pi_r\widetilde{f}_\infty\|^2}{\sum_{r=1}^{s}\|\Pi_r\widetilde{f}_\infty\|^2}. \tag{6}$$

Thus ρ_∞ is a weighted average of the random-walk eigenvalues λ_r, with weights proportional to the spectral content of the centered output in $\mathcal{E}_r$. Since $\lambda_0 > \lambda_1 > \cdots > \lambda_s$, having more relative spectral weight at smaller values of r favors higher autocorrelation and a smoother landscape.

3.2 Spectrum of the Neural Tangent Kernel

Let $\mathbf{\Theta}$ be the restriction of the NTK Θ to $\mathcal{X}$, with entries $\mathbf{\Theta}_{xy} = \Theta(x,y)$ for $x, y \in \mathcal{X}$. To understand how training distributes the spectral weight of f_∞ across the random-walk eigenspaces $\mathcal{E}_r$, it is important to understand how $\mathbf{\Theta}$ acts on these spaces. For $0 \le \ell, r \le s$, define the Eberlein polynomial $E_\ell^{(n,k)}(r) = \sum_{h=0}^{\ell}(-1)^h\binom{r}{h}\binom{k-r}{\ell-h}\binom{n-k-r}{\ell-h}$, where a binomial coefficient is zero when its lower index lies outside the range from zero to its upper index. The following lemma shows that every $\mathcal{E}_r$ is an eigenspace of $\mathbf{\Theta}$ and computes the corresponding eigenvalue explicitly.

Lemma 1 *The NTK matrix $\mathbf{\Theta}$ admits the same orthogonal eigenspace decomposition $\bigoplus_{r=0}^{s}\mathcal{E}_r$ as the random-walk matrix P, with corresponding eigenvalues*

$$\mu_r = \textstyle\sum_{\ell=0}^{s}\vartheta(\ell)E_\ell^{(n,k)}(r) \ge 0, \qquad r = 0, \dots, s,$$

Here, $\vartheta = \vartheta^{(L)}$ is defined by the following recursion. For $d = 0, \dots, s$, set $\kappa^{(1)}(d) = \frac{k-d}{n} + \beta^2$ and $\vartheta^{(1)}(d) = \kappa^{(1)}(d)$. For $\ell = 2, \dots, L$, let $(U_{\ell,d}, V_{\ell,d})^\top \sim \mathcal{N}\Big(0, \begin{pmatrix} \kappa^{(\ell-1)}(0) & \kappa^{(\ell-1)}(d) \\ \kappa^{(\ell-1)}(d) & \kappa^{(\ell-1)}(0) \end{pmatrix}\Big)$ and define $\kappa^{(\ell)}(d) = \mathbb{E}[\sigma(U_{\ell,d})\sigma(V_{\ell,d})] + \beta^2$ and $\vartheta^{(\ell)}(d) = \kappa^{(\ell)}(d) + \mathbb{E}[\sigma'(U_{\ell,d})\sigma'(V_{\ell,d})]\,\vartheta^{(\ell-1)}(d)$.

Ignoring initialization in Equation (3), the spectral weight of f_∞ on $\mathcal{E}_r$ can be expressed using the action of $\mathbf{\Theta}$ on $\mathcal{E}_r$. For the component of f_∞ due to initialization, we bound its covariance matrix by $\mathbf{\Theta}$ to obtain a bound on its expected spectral weight on $\mathcal{E}_r$, expressed again using the same action. Lemma 1 characterizes this action exactly, which will allow us to characterize the spectral weight of f_∞ on each $\mathcal{E}_r$. The proof of Lemma 1 is given in Appendix B.2.

3.3 High Autocorrelation and Spectral Bias of the Trained Network Output

Taking $t \to \infty$ in Equation (3), write $f_\infty = h + g$, where $h(x) = \mathbf{k}(x)^\top \mathbf{K}^{-1}\mathbf{y}$ is the component learned from labels and $g(x) = f_0(x) - \mathbf{k}(x)^\top \mathbf{K}^{-1}\mathbf{f}_0$ is the component inherited from initialization that remains after training. The following lemma bounds the spectral weights of both components.

Lemma 2 *Suppose that σ is q times differentiable with bounded derivatives of orders $1, \dots, q$, for some integer $q \geq 2$. Then there exists a constant $C = C(q, \sigma, L, \beta) < \infty$ such that*

$$\textit{(i)} \quad \mathbb{E}_{\theta_0}\Big[\sum_{r=1}^{s} r^{q-1}\|\Pi_r g\|^2\Big] \leq C, \qquad \textit{(ii)} \quad \sum_{r=1}^{s} r^{q-1} d_r \|\Pi_r h\|^2 \leq C\|\mathbf{y}\|^2_{\mathbf{K}^{-1}},$$

where $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}} = \mathbf{y}^\top \mathbf{K}^{-1}\mathbf{y}$.

A complete proof is given in Appendix B.3.

Part (i) shows that the surviving initialization g is spectrally biased toward low levels. In particular, its expected spectral weight in $\mathcal{E}_r$ is at most $Cr^{-(q-1)}$. The constant C is independent of n and k, and hence of $s = \min(k, n-k)$. Therefore, for fixed q, σ, L, and β, the upper bound decays polynomially with r even as s grows and additional spectral levels become available.

Part (ii) implies that the squared weight of h in $\mathcal{E}_r$ decays at least as fast as $\frac{C}{d_r r^{q-1}}\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$. The dependence on $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$ accounts for the spectral weight needed to fit the labels. If fitting them requires substantial weight at high levels, this factor must be large enough for the bound to allow h to reproduce that weight. Conversely, when this factor is small, the bound forces the high-level weights of h to be small. For example, Lemma 6 in Appendix B.4 shows that, for the class of labels $y_i = u(x_i)$ generated by a centered, standardized linear function $u \in \mathcal{E}_1$ on $\mathcal{X}$, there is a constant C_{lin} independent of n, k, and the training subset such that $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}/d_r \leq C_{\text{lin}}$, and hence $\|\Pi_r h\|^2 \leq CC_{\text{lin}} r^{-(q-1)}$ for $r = 2, \dots, s$. In contrast, training on every point of $\mathcal{X}$ with the class of labels $y_i = u(x_i)$ generated by a centered, standardized target $u \in \mathcal{E}_s$ gives $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}/d_s \geq s^{q-1}/C$, so the label-dependent factor offsets the polynomial decay to allow for the large weight $\|\Pi_s h\|^2 = 1$ as $h = u$ on $\mathcal{X}$ in this case.

We now use Lemma 2 to establish our main results. The following theorem shows that, under natural assumptions, the expected autocorrelation of f_∞ is high.

Theorem 1 *Assume that $k/n \in [\alpha, 1-\alpha]$ for some $0 < \alpha < 1/2$, and that the training set $\mathcal{T}$ consists of N input-label pairs whose inputs are sampled uniformly without replacement from $\mathcal{X}$, independently of θ_0, where $1 \leq N \leq n^a$ for some $a > 0$. Then there exist positive constants $C, n_0 < \infty$, depending only on $a, \alpha, \sigma, L, \beta$, such that, for $n \geq n_0$,*

$$\mathbb{E}_{\mathcal{T},\theta_0}[\rho_\infty] \geq 1 - \frac{C}{n},$$

where the expectation is over the sampled training inputs and the random initialization.

Intuitively, polynomially many uniformly sampled training points allow some low-level variation from initialization to survive training, while Lemma 2 helps control high-level variation relative to the total variation, leading to high expected autocorrelation. A complete proof is given in Appendix B.5. Theorem 1 shows that, when the training inputs are sampled uniformly and their number grows at most polynomially with n, the expected autocorrelation of f_∞ converges to 1 as $n \to \infty$, provided k/n stays bounded away from 0 and 1. Thus, in terms of expected autocorrelation, the learned landscapes become asymptotically very smooth.

For $R = 0, \ldots, s$, define the filtered output $f_\infty^{(\leq R)} = \sum_{r=0}^{R} \Pi_r f_\infty$. The next theorem bounds the mean-square distance of f_∞ from $f_\infty^{(\leq R)}$, even under arbitrary training subsets and $1 \leq k < n$.

Theorem 2 *Suppose that σ is q times differentiable with bounded derivatives of orders $1, \ldots, q$, for some integer $q \geq 2$. Then there exists a constant $C = C(q, \sigma, L, \beta) < \infty$ such that, for $R = 0, \ldots, s$,*

$$\mathbb{E}_{\theta_0} \left\| f_\infty - f_\infty^{(\leq R)} \right\|^2 \leq \frac{C}{(R+1)^{q-1}} \left(1 + \|\mathbf{y}\|^2_{\mathbf{K}^{-1}}\right).$$

We obtain this result by combining the bounds in Lemma 2. A complete proof is given in Appendix B.6. Theorem 2 shows that, for a given value of the factor $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$, the expected squared distance between f_∞ and $f_\infty^{(\leq R)}$ has an upper bound that decreases polynomially with R. Thus, when this factor is controlled, f_∞ lies in the lower spectral levels up to a small mean-square error.[5]

The theorem has an important implication for local search. If we can filter out the higher levels, we can obtain a smoother function (the filtered output is either constant or has autocorrelation at least that of f_∞) that stays close to the overall shape of f_∞ and use it to guide the search. This motivates our Filtered Local Search approach, which we present next.

4 Filtered Local Search

The spectral analysis in Section 3 shows that filtering the high-level components of f_∞ can give a smoother approximation with small changes in output values. This motivates using local search on the filtered function to find a promising basin before refining the solution through local search on the original function. However, evaluating a spectral truncation is generally impractical because it may require evaluating f_∞ at every solution in $\mathcal{X}$. Instead, we propose the filtered function $\bar{f}_\infty = P f_\infty$. Multiplication by P scales each component $\Pi_r f_\infty$ by λ_r. The magnitudes $|\lambda_r|$ decrease with r while $\lambda_r \geq 0$. After the sign change, these factors are at most $1/\max(k, n-k) \leq 2/n$, so the highest-level components are strongly suppressed for large n. Since $\bar{f}_\infty(u) = D^{-1} \sum_{z \in \mathcal{N}(u)} f_\infty(z)$, evaluating $\bar{f}_\infty(u)$ requires evaluating f_∞ only at the neighbors of u, rather than throughout $\mathcal{X}$. Our Filtered Local Search (FLS) method applies best-improvement local search first to $\bar{f}_\infty$ and then to f_∞, starting from the solution found in the first phase. Pseudocode is given in Algorithm 1 in Appendix C.

We note that the idea of using neighborhood averages to guide local search has shown empirical benefits across several types of landscapes. For example, Borenstein & Philemotte (2008) improved randomized local search for the traveling-salesperson problem by selecting moves using average tour lengths in sampled neighborhoods rather than individual tour lengths. Whitley & Chen (2012) improved steepest-descent search on NK landscapes using averages over one-bit neighbors. For nearest-neighbor image classification, Nguyen et al. (2019) improved accuracy by choosing local-search moves using averages of neighboring images' distances to the target. In protein optimization, Kirjner et al. (2024) and Zhu et al. (2025) smoothed predictions from a trained neural network, then trained a second network on these smoothed predictions and used its output as the objective for local search over protein sequences. This improved performance on in-silico benchmarks. As opposed to these methods, our FLS method keeps the trained network fixed and applies local search first to its filtered predictions and then to its original predictions.

5 Controlled Experiments

To evaluate our claims empirically, we conduct three controlled experiments. We first summarize their shared experimental setup, then present the results.

Experimental setup. We generate training datasets from three families of ground-truth functions, namely quadratic functions, Ising functions, and NK landscapes. For each family, we consider five input dimensions $n \in \{15, 20, 25, 30, 35\}$ and generate five random instances for each dimension.

[5] Note that small filtering error does not guarantee high autocorrelation, hence the additional assumptions in Theorem 1. For example, consider $f = 1 + \varepsilon v$, where $v \in \mathcal{E}_s$, $\|v\| = 1$, and $\varepsilon > 0$. For any cutoff $R < s$, the squared filtering error is ε^2, while autocorrelation remains λ_s, however small ε is.

For each instance, we sample $1{,}024$ distinct inputs and compute their target values. We standardize these values to have zero mean and unit variance, then construct five nested training datasets using the first $32, 64, 384, 512, 1024$ observations, respectively. On each of the resulting $3\times5\times5\times5 = 375$ datasets, we train a fully connected neural network with two hidden layers of width 1024 and GELU activations. We initialize every unscaled weight and bias independently from $\mathcal{N}(0,1)$ and set the bias multiplier to $\beta = 0.1$. The five runs for each instance share the same random initialization. We minimize the mean squared error loss using full-batch gradient descent with learning rate 15 until either the mean squared training error is at most 10^{-4} or $10{,}000$ steps have been completed. For optimization, we consider the single-cardinality feasible set of the form (1) with the largest $k \le \lfloor n/2 \rfloor$ for which $\binom{n}{k} \le 100{,}000$. This gives $k = 7, 7, 5, 4, 4$, respectively, allowing us to evaluate each trained network at every feasible solution. Appendix D.1 gives more details about data generation, training, and the feasible sets.

Autocorrelation and local-search performance. For each trained network $\widehat{f}$, we compute its autocorrelation ρ and the coefficient $Q = \frac{\overline{f} - |\mathcal{X}|^{-1}\sum_{x\in\mathcal{X}} \widehat{f}(L(x))}{\overline{f} - \min_{x\in\mathcal{X}} \widehat{f}(x)}$. Here, $\overline{f}$ is the average of $\widehat{f}$ over $\mathcal{X}$ and $L(x)$ is the terminal solution reached from x by best-improvement local search. For a uniformly sampled starting solution, the numerator is the expected decrease in $\widehat{f}$ achieved by local search, while the denominator is the expected decrease obtained by always reaching a global minimum. Recall that, for each random target instance, we trained five networks on five nested training datasets. These networks are fitted to common observations from the smallest dataset and hence provide diverse input landscapes that fit the same observations and potentially have different autocorrelation values. We compute the Spearman correlation between ρ and Q across these five networks. The overall median Spearman correlation is 0.800. Appendix D.2 gives more detailed results.

High autocorrelation and spectral bias. For each trained network $\widehat{f}$, we compare its autocorrelation ρ with the largest autocorrelation possible on the corresponding graph, λ_1. On average, autocorrelation is 90.1% of this maximum. Across the nested datasets, autocorrelation is highest with fewer observations, consistent with the networks favoring smooth interpolations. We also compute the fraction of output variance in each random-walk eigenspace $\mathcal{E}_r$, given by $\|\Pi_r(\widehat{f} - \overline{f})\|^2 / \|\widehat{f} - \overline{f}\|^2$. The first two spectral levels contain on average 97.0% of the variance, showing strong concentration at low levels. Appendix D.3 provides more detailed results.

Filtered Local Search. We compare FLS (Algorithm 1) with best-improvement local search on the trained networks, running both methods from every feasible solution. We evaluate their terminal solutions on the original objective $\widehat{f}$ using Q. FLS increases mean Q from 0.9631 to 0.9785. It improves Q on 214 networks, leaves it unchanged on 149, and reduces it on 12. Appendix D.4 provides more detailed results.

6 Applications

The purpose of this section is to evaluate FLS in applications with training procedures, neural-network architectures, and feasible sets beyond those of the controlled experiments of Section 5, which closely follow our theoretical setting. We consider two applications in facility location and production scheduling.

6.1 Capacitated Facility Location

In the first application, we consider the two-stage stochastic capacitated facility location problem, in which one chooses which facilities to open before customer demands are observed.

Experimental setup. Patel et al. (2022) train three neural networks, one for each problem size $n \in \{10, 25, 50\}$, with n facilities and n customers. Each network takes a vector indicating which facilities are open and a customer-demand vector as inputs and predicts the cost of serving those customers. Using these predictions, they choose which facilities to open by solving

$$\min_{x\in\{0,1\}^n} \left\{ \sum_{i=1}^{n} a_i x_i + S^{-1} \sum_{j=1}^{S} \widehat{Q}(x, \xi^{(j)}) \right\},$$

where a_i is the opening cost of facility i, $\widehat{Q}$ is the network prediction, and $\xi^{(1)}, \ldots, \xi^{(S)}$ are sampled customer-demand vectors. They solve this problem using a mixed-integer linear formulation. We evaluate FLS on this problem using their three released networks. For each network, we consider four scenario counts, $S \in \{1000, 2000, 5000, 10000\}$. For each count, we generate ten random optimization instances by sampling S customer-demand vectors, giving $3 \times 4 \times 10 = 120$ instances in total. For each instance, we run ordinary best-improvement local search and FLS from the same 100 uniformly sampled binary starting solutions. A move opens or closes one facility. We also benchmark against Gurobi using the mixed-integer formulation, warm-started from the first of the shared random starting solutions and given the same optimization time as the complete 100-start FLS run on each instance. We compare the best solutions obtained by the three methods. See Appendix E.1.1 for more details.

Results. FLS obtains a lower final objective than ordinary local search on 8,219 of the $120 \times 100 = 12{,}000$ paired starts. Comparing the best of 100 starts, FLS wins in 70 instances, ties in 49, and loses in one. Against the warm-started Gurobi baseline, FLS obtains a lower objective in all 120 instances. Appendix E.1.2 gives more detailed results.

6.2 Production Scheduling

In the second application, we consider a production-scheduling problem, where one chooses the order in which machines process production operations. Each job consists of operations that must be performed in a prescribed order. The objective is to minimize the time at which all jobs are completed, called the makespan.

Experimental setup. Ghasemi et al. (2023) train four neural networks $\widehat{f}(x, c)$ to predict makespan, where x is a tuple of permutations, each ordering the operations assigned to one machine, and c specifies the production instance. The networks use recurrent layers to process sequences and either fully connected layers or attention to combine this information. Training uses schedules labeled by a production simulator. We evaluate FLS using their four released networks and ten production instances from their test set, each with eight machines, 11 or 12 jobs, and 98–119 operations. Let $\mathcal{O}_m(c)$ be the operations assigned to machine m and $\mathcal{D}(x, c)$ the directed graph joining consecutive operations in each fixed job order and each chosen machine order. We solve

$$\min_{x \in \mathcal{X}(c)} \widehat{f}(x, c),$$

where $\mathcal{X}(c) = \{x \in \prod_{m=1}^{8} \mathrm{Perm}(\mathcal{O}_m(c)) \mid \mathcal{D}(x, c) \text{ is acyclic}\}$ and $\mathrm{Perm}(\mathcal{O}_m(c))$ denotes all permutations of $\mathcal{O}_m(c)$. A move exchanges two operations on the same machine, provided that the resulting order remains feasible. For each of the $4 \times 10 = 40$ network–instance combinations, we run ordinary best-improvement local search and FLS from the same 100 random feasible starting solutions. See Appendix E.2.1 for more details.

Results. FLS obtains a lower final objective than ordinary local search on 3,137 of the $40 \times 100 = 4{,}000$ paired starts. Comparing the best solutions over 100 starts, FLS improves on ordinary local search in 35 of the 40 instances and returns a higher objective in 5. Appendix E.2.2 gives more detailed results.

7 Conclusion

We studied the discrete input landscapes of trained neural networks, establishing high-autocorrelation and spectral-bias results for infinite-width MLPs over cardinality-constrained binary sets. These results motivated Filtered Local Search, which improved average solution quality over ordinary local search in both controlled experiments and applications.

Our work suggests several directions for future research. One is to study other landscape properties, such as the structure of basins of attraction and their role in local-search performance. Another direction is to extend the analysis to finite-width networks, other architectures, and broader feasible sets. One could also design other filters for the network output beyond the neighborhood averaging Pf_∞, for example using insights from signal processing. Finally, one could reduce the computational cost of FLS, for example by using best-arm identification to select moves without computing every neighborhood average exactly.

AI USE STATEMENT

The authors received assistance through Codex from OpenAI models, including GPT-5.5, GPT-5.6, and GPT-6 Astra, as they became available during the project. This assistance covered the following aspects of the work.

Writing. The authors determined the manuscript's content and structure. AI tools assisted with polishing the language and improving readability.

Literature review. AI tools assisted with some of the literature review. The authors checked all cited sources and verified the accuracy of the references and the claims attributed to them.

Mathematical claims and proofs. The mathematical claims were formulated by the authors. The proofs were developed through iterative interaction with AI tools, with the authors directing their organization and exposition and checking the mathematical arguments.

Experiments. AI tools assisted with planning the experiments and generated the code for synthetic-data generation and experimental implementation. The authors tested the code and led the analysis and interpretation of the results.

The authors take full responsibility for the final content of the paper.

REFERENCES

Eric Angel and Vassilis Zissimopoulos. On the classification of NP-complete problems in terms of their correlation coefficient. *Discrete Applied Mathematics*, 99(1–3):261–277, 2000. doi: 10.1016/S0166-218X(99)00138-9. URL https://www.sciencedirect.com/science/article/pii/S0166218X99001389.

Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristics for k-median and facility location problems. *SIAM Journal on Computing*, 33(3):544–562, 2004. doi: 10.1137/S0097539702416402.

Francisco Barahona, Martin Grötschel, Michael Jünger, and Gerhard Reinelt. An application of combinatorial optimization to statistical physics and circuit layout design. *Operations Research*, 36(3):493–513, 1988. doi: 10.1287/opre.36.3.493.

Stefano Benedettini, Marco Villani, Andrea Roli, Roberto Serra, Mattia Manfroni, Antonio Gagliardi, Carlo Pinciroli, and Mauro Birattari. Dynamical regimes and learning properties of evolved Boolean networks. *Neurocomputing*, 99:111–123, 2013. doi: 10.1016/j.neucom.2012.05.023. URL https://doi.org/10.1016/j.neucom.2012.05.023.

Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1024–1034. PMLR, 2020. URL https://proceedings.mlr.press/v119/bordelon20a.html.

Yossi Borenstein and Christophe Philemotte. Preprocessing fitness functions. Technical Report TR/IRIDIA/2008-003, IRIDIA, Université Libre de Bruxelles, January 2008. URL https://iridia.ulb.ac.be/IridiaTrSeries/link/IridiaTr2008-003.pdf.

Andries E. Brouwer and Willem H. Haemers. *Spectra of Graphs*. Universitext. Springer, New York, 2012. doi: 10.1007/978-1-4614-1939-6.

Ozan Candogan and Ayoub Foussoul. Deep flow networks. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://icml.cc/virtual/2026/poster/63203.

Luís Carvalho, João L. Costa, José Mourão, and Gonçalo Oliveira. The positivity of the neural tangent kernel. *SIAM Journal on Mathematics of Data Science*, 7(2):495–515, 2025. doi: 10.1137/24M1659534.

Chandra Chekuri and Amit Kumar. Maximum coverage problem with group budget constraints and applications. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, volume 3122 of *Lecture Notes in Computer Science*, pp. 72–83. Springer, 2004. doi: 10.1007/978-3-540-27821-4_7.

Abhimanyu Das and David Kempe. Approximate submodularity and its applications: Subset selection, sparse approximation and dictionary selection. *Journal of Machine Learning Research*, 19 (3):1–34, 2018. URL https://www.jmlr.org/papers/v19/16-534.html.

Philippe Delsarte. *An Algebraic Approach to the Association Schemes of Coding Theory*. Number 10 in Philips Research Reports Supplements. N.V. Philips' Gloeilampenfabrieken, Eindhoven, 1973.

Matteo Fischetti and Jason Jo. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3):296–309, 2018. doi: 10.1007/s10601-018-9285-6.

Sam Gelman, Sarah A. Fahlberg, Pete Heinzelman, Philip A. Romero, and Anthony Gitter. Neural networks to learn protein sequence–function relationships from deep mutational scanning data. *Proceedings of the National Academy of Sciences*, 118(48):e2104878118, 2021. doi: 10.1073/pnas.2104878118. URL https://www.pnas.org/doi/10.1073/pnas.2104878118.

Amir Ghasemi, Yavar Taheri Yeganeh, Andrea Matta, Kamil Erkan Kabak, and Cathal Heavey. Deep learning enabling digital twin applications in production scheduling: Case of flexible job shop manufacturing environment. In *Proceedings of the 2023 Winter Simulation Conference*, pp. 2148–2159. IEEE, 2023. doi: 10.1109/WSC60868.2023.10407811. URL https://doi.org/10.1109/WSC60868.2023.10407811.

Gurobi Optimization, LLC. Gurobi optimizer reference manual, 2026. URL https://docs.gurobi.com/projects/optimizer/en/current/.

Peter L. Hammer and Sergiu Rudeanu. *Boolean Methods in Operations Research and Related Areas*, volume 7 of *Econometrics and Operations Research*. Springer-Verlag, Berlin, 1968. doi: 10.1007/978-3-642-85823-9.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (GELUs). *arXiv preprint arXiv:1606.08415*, 2016. URL https://arxiv.org/abs/1606.08415.

Ernst Ising. Beitrag zur Theorie des Ferromagnetismus. *Zeitschrift für Physik*, 31:253–258, 1925. doi: 10.1007/BF02980577. URL https://link.springer.com/article/10.1007/BF02980577.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://proceedings.neurips.cc/paper/2018/file/5a4be1fa34e62bb8a6ec6b91d2462f5a-Paper.pdf.

David S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9(3):256–278, 1974. doi: 10.1016/S0022-0000(74)80044-9.

Waquar Kaleem, Doyoung Lee, Changhyun Kwon, and Anirudh Subramanyam. Neural embedded mixed-integer optimization for location-routing problems. *arXiv preprint arXiv:2412.05665*, 2026. URL https://arxiv.org/abs/2412.05665.

Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher (eds.), *Complexity of Computer Computations*, pp. 85–103. Plenum Press, New York, 1972. doi: 10.1007/978-1-4684-2001-2_9.

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. doi: 10.1016/S0022-5193(89)80019-0.

David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11(4):105–147, 2015. doi: 10.4086/toc.2015.v011a004. URL https://theoryofcomputing.org/articles/v011a004/.

Andrew Kirjner, Jason Yim, Raman Samusevich, Shahar Bracha, Tommi Jaakkola, Regina Barzilay, and Ila Fiete. Improving protein optimization with smoothed fitness landscapes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=rxlF2Zv8x0.

Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(8):235–284, 2008. URL https://www.jmlr.org/papers/v9/krause08a.html.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1EA-M-0Z.

Jaehoon Lee, Lechao Xiao, Samuel S. Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12): 124002, 2020. doi: 10.1088/1742-5468/abc62b. URL https://iopscience.iop.org/article/10.1088/1742-5468/abc62b.

Shengcai Liu, Ning Lu, Cheng Chen, and Ke Tang. Efficient combinatorial optimization for word-level adversarial textual attack. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:98–111, 2022. doi: 10.1109/TASLP.2021.3130970. URL https://ieeexplore.ieee.org/document/9627642/.

Michele Lombardi, Michela Milano, and Andrea Bartolini. Empirical decision model learning. *Artificial Intelligence*, 244:343–367, 2017. doi: 10.1016/j.artint.2016.01.005.

Seungyong Moon, Gaon An, and Hyun Oh Song. Parsimonious black-box adversarial attacks via efficient combinatorial optimization. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4636–4645, 2019. URL https://proceedings.mlr.press/v97/moon19a.html.

Sebastián Muñoz-Herrera and Karol Suchan. Constrained fitness landscape analysis of capacitated vehicle routing problems. *Entropy*, 24(1):53, 2022. doi: 10.3390/e24010053. URL https://www.mdpi.com/1099-4300/24/1/53.

Radford M. Neal. *Bayesian Learning for Neural Networks*, volume 118 of *Lecture Notes in Statistics*. Springer, 1996. doi: 10.1007/978-1-4612-0745-0.

Thanh Tan Nguyen, Ali Shameli, Yasin Abbasi-Yadkori, Anup Rao, and Branislav Kveton. Sample efficient graph-based optimization with noisy observations. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 3333–3341. PMLR, 2019. URL https://proceedings.mlr.press/v89/nguyen19b.html.

Rahul Mihir Patel, Justin Dumouchelle, Elias Khalil, and Merve Bodur. Neur2SP: Neural two-stage stochastic programming. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23992–24005. Curran Associates, Inc., 2022. doi: 10.52202/068431-1742. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/9793671e4be9858a69a32545204d59d1-Abstract-Conference.html.

Paat Rusmevichientong, Zuo-Jun Max Shen, and David B. Shmoys. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations Research*, 58(6): 1666–1680, 2010. doi: 10.1287/opre.1100.0866.

Artur M. Schweidtmann and Alexander Mitsos. Deterministic global optimization with artificial neural networks embedded. *Journal of Optimization Theory and Applications*, 180(3):925–948, 2019. doi: 10.1007/s10957-018-1396-0.

Peter F. Stadler. Landscapes and their correlation functions. *Journal of Mathematical Chemistry*, 20 (1):1–45, 1996. doi: 10.1007/BF01165154.

Nikolaos Tsilivis and Julia Kempe. What can the neural tangent kernel tell us about adversarial robustness? In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/72f9c316440c384a95c88022fd78f066-Abstract-Conference.html.

B. Waclaw and Z. Burda. Counting metastable states of Ising spin glasses on arbitrary graphs. *Physical Review E*, 77(4):041114, 2008. doi: 10.1103/PhysRevE.77.041114. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.77.041114.

Hélène Waeselynck, Pascale Thévenod-Fosse, and Olfa Abdellatif-Kaddour. Simulated annealing applied to test generation: Landscape characterization and stopping criteria. *Empirical Software Engineering*, 12(1):35–63, 2007. doi: 10.1007/s10664-006-7551-5. URL https://link.springer.com/article/10.1007/s10664-006-7551-5.

Sifan Wang, Xinling Yu, and Paris Perdikaris. When and why PINNs fail to train: A neural tangent kernel perspective. *Journal of Computational Physics*, 449:110768, 2022. doi: 10.1016/j.jcp.2021.110768. URL https://doi.org/10.1016/j.jcp.2021.110768.

Edward D. Weinberger. Correlated and uncorrelated fitness landscapes and how to tell the difference. *Biological Cybernetics*, 63(5):325–336, 1990. doi: 10.1007/BF00202749.

Jakob Weissteiner and Sven Seuken. Deep learning-powered iterative combinatorial auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 2284–2293, 2020. doi: 10.1609/aaai.v34i02.5606.

L. Darrell Whitley and Wenxiang Chen. Constant time steepest descent local search with lookahead for NK-landscapes and MAX-kSAT. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, pp. 1357–1364. ACM, 2012. doi: 10.1145/2330163.2330351.

Angelika Wiegele. Biq Mac Library - a collection of max-cut and quadratic 0-1 programming instances of medium size. Technical report, Alpen-Adria-Universität Klagenfurt, 2007. URL https://biqmac.aau.at/biqmaclib.pdf.

Christopher K. I. Williams. Computation with infinite neural networks. *Neural Computation*, 10 (5):1203–1216, 1998. doi: 10.1162/089976698300017412. URL https://doi.org/10.1162/089976698300017412.

Hao Zhu, Daniel M. Steinberg, and Piotr Koniusz. Protein fitness landscape: Spectral graph theory perspective. In *Proceedings of the 28th International Conference on Artificial Intelligence and Statistics*, volume 258 of *Proceedings of Machine Learning Research*, pp. 2827–2835. PMLR, 2025. URL https://proceedings.mlr.press/v258/zhu25c.html.

A Details of the Infinite-Width Limit

Recall from Section 2.2 that we consider a fully connected feedforward network $f_\theta : \mathbb{R}^n \to \mathbb{R}$ with $L-1 \geq 1$ hidden layers of widths $m_1, \ldots, m_{L-1}$, given by

$$f_\theta(x) = \frac{W^{(L)}}{\sqrt{m_{L-1}}}\sigma\left(\frac{W^{(L-1)}}{\sqrt{m_{L-2}}}\sigma\left(\cdots\sigma\left(\frac{W^{(1)}x}{\sqrt{n}} + \beta b^{(1)}\right)\cdots\right) + \beta b^{(L-1)}\right) + \beta b^{(L)},$$

where $\theta = \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$ denotes the network parameters, $\beta > 0$ is fixed, and $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonpolynomial, Lipschitz, and twice differentiable activation function with bounded second derivative, applied entrywise.

At initialization θ_0, all entries of the weight matrices and bias vectors are drawn independently from $\mathcal{N}(0,1)$. We train the network on $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{N}$, with pairwise distinct inputs $x_i \in \mathcal{X}$ and targets $y_i \in \mathbb{R}$, by minimizing the empirical squared loss

$$\mathcal{L}(\theta) = \frac{1}{2N}\sum_{i=1}^{N}\left(f_\theta(x_i) - y_i\right)^2$$

using gradient flow,

$$\frac{d}{dt}\theta_t = -\nabla_\theta \mathcal{L}(\theta_t), \qquad \theta_{t=0} = \theta_0.$$

Let f_{θ_t} denote the network function at time $t \geq 0$. A direct application of the chain rule gives, for every $x \in \mathbb{R}^n$,

$$\frac{d}{dt} f_{\theta_t}(x) = -\frac{1}{N} \sum_{i=1}^{N} \widehat{\Theta}_t(x, x_i) \left(f_{\theta_t}(x_i) - y_i \right),$$

where, for $x, x' \in \mathbb{R}^n$,

$$\widehat{\Theta}_t(x, x') = \langle \nabla_\theta f_{\theta_t}(x), \nabla_\theta f_{\theta_t}(x') \rangle .$$

Intuitively, the function $\widehat{\Theta}_t$, known as the empirical tangent kernel at time t (Lee et al., 2020), measures how similarly the network outputs at the two inputs respond to small changes in the parameters.

We next take the infinite-width limit by letting the hidden-layer widths $m_1, \ldots, m_{L-1}$ tend sequentially to infinity while keeping the depth L, input dimension n, training set $\mathcal{T}$, and β fixed. The following three properties hold.

(i) *Gaussian-process initialization.* The initial network output converges in distribution to a centered Gaussian process $f_0 \sim \mathcal{GP}(0, \Sigma)$. More precisely, given any finite set of inputs $\mathcal{S} = \{x^{(1)}, \ldots, x^{(q)}\} \subset \mathbb{R}^n$, it holds that

$$\big(f_{\theta_0}(x^{(1)}), \ldots, f_{\theta_0}(x^{(q)})\big) \xrightarrow{\mathcal{D}} \big(f_0(x^{(1)}), \ldots, f_0(x^{(q)})\big).$$

Intuitively, the first-layer preactivations are Gaussian by initialization. Conditional on the activations of one layer, the preactivations in the next layer are independent Gaussian processes because they are linear combinations of independent Gaussian weights. As the preceding layer becomes wide, the law of large numbers makes their conditional covariance converge to a deterministic limit. Repeating this argument layer by layer shows that the preactivations within every hidden layer converge to i.i.d. centered Gaussian processes across neurons and that the scalar network output converges to f_0. Early work established the Gaussian-process limit and its covariance structure for one-hidden-layer networks (Neal, 1996; Williams, 1998). For fully connected multilayer networks such as the architecture considered here, the Gaussian-process limit and its covariance structure were established by Jacot et al. (2018) and Lee et al. (2018). In particular, the initialization covariance Σ can be computed recursively as follows. For $x, x' \in \mathbb{R}^n$, let

$$\Sigma^{(1)}(x, x') = \frac{x^\top x'}{n} + \beta^2.$$

For every $\ell = 2, \ldots, L$, let (U_ℓ, V_ℓ) be a centered Gaussian vector with covariance matrix

$$\begin{pmatrix} \Sigma^{(\ell-1)}(x, x) & \Sigma^{(\ell-1)}(x, x') \\ \Sigma^{(\ell-1)}(x', x) & \Sigma^{(\ell-1)}(x', x') \end{pmatrix},$$

and set

$$\Sigma^{(\ell)}(x, x') = \mathbb{E}[\sigma(U_\ell)\sigma(V_\ell)] + \beta^2.$$

The initialization covariance is $\Sigma = \Sigma^{(L)}$.

(ii) *A deterministic limiting NTK.* At initialization, the empirical tangent kernel $\widehat{\Theta}_0$ converges in probability to a deterministic kernel Θ. More precisely, given any finite set of inputs $\mathcal{S} = \{x^{(1)}, \ldots, x^{(q)}\} \subset \mathbb{R}^n$, it holds that

$$\left[\widehat{\Theta}_0\Big(x^{(a)}, x^{(b)}\Big)\right]_{a,b=1}^{q} \xrightarrow{\mathrm{P}} \left[\Theta\Big(x^{(a)}, x^{(b)}\Big)\right]_{a,b=1}^{q}.$$

Intuitively, $\widehat{\Theta}_0$ is formed from averages over the randomly initialized neurons. As the widths tend to infinity, these averages converge to deterministic expectations. This result was established by Jacot et al. (2018), who showed that Θ can be computed recursively as follows. Let σ' denote the derivative of σ. Using the Gaussian pairs (U_ℓ, V_ℓ) defined above, set

$$\Theta^{(1)} = \Sigma^{(1)}, \qquad \Theta^{(\ell)} = \Sigma^{(\ell)} + \mathbb{E}[\sigma'(U_\ell)\sigma'(V_\ell)]\,\Theta^{(\ell-1)}, \quad \ell = 2, \ldots, L,$$

where multiplication is pointwise on $\mathbb{R}^n \times \mathbb{R}^n$. The limiting NTK is $\Theta = \Theta^{(L)}$.

(iii) *Kernel stability during training.* During every fixed finite training interval, the empirical tangent kernel converges uniformly in probability to Θ. More precisely, given any finite set of inputs $\mathcal{S} = \{x^{(1)}, \dots, x^{(q)}\} \subset \mathbb{R}^n$ and any $0 \le T < \infty$, it holds that

$$\sup_{0 \le t \le T} \left| \widehat{\Theta}_t\Big(x^{(a)}, x^{(b)}\Big) - \Theta\Big(x^{(a)}, x^{(b)}\Big) \right| \xrightarrow{\mathrm{P}} 0, \qquad a, b = 1, \dots, q.$$

Intuitively, under the NTK parameterization, the change in the network output during training is distributed across an increasing number of parameters. As the widths grow, each parameter's contribution to this change vanishes, while their combined contribution remains of order one. Consequently, the network output can change during training while the empirical NTK remains fixed in the limit. For a proof of this result, see Jacot et al. (2018).

Let $(f_t)_{t \ge 0}$ denote the random process initialized by $f_0 \sim \mathcal{GP}(0, \Sigma)$ and satisfying the following differential equation for every input $x \in \mathbb{R}^n$,

$$\frac{d}{dt} f_t(x) = -\frac{1}{N} \sum_{i=1}^{N} \Theta(x, x_i) \left(f_t(x_i) - y_i \right).$$

Together, the three properties above imply that the network outputs converge in distribution to f_t at every fixed training time. More precisely, given any finite set of inputs $\mathcal{S} = \{x^{(1)}, \dots, x^{(q)}\} \subset \mathbb{R}^n$ and any fixed $t \ge 0$, it holds that

$$\big(f_{\theta_t}(x^{(1)}), \dots, f_{\theta_t}(x^{(q)})\big) \xrightarrow{\mathcal{D}} \big(f_t(x^{(1)}), \dots, f_t(x^{(q)})\big).$$

The differential equation defining f_t can be solved explicitly (Jacot et al., 2018; Lee et al., 2020). In particular, let $\mathbf{y} = (y_1, \dots, y_N)^\top$, $\mathbf{f}_0 = (f_0(x_1), \dots, f_0(x_N))^\top$, $\mathbf{K}_{ij} = \Theta(x_i, x_j)$, and $\mathbf{k}(x) = (\Theta(x, x_i))_{i=1}^N$. The solution of the limiting kernel flow is given, for every $x \in \mathbb{R}^n$ and $t \ge 0$, by

$$f_t(x) = f_0(x) + \mathbf{k}(x)^\top \mathbf{K}^{-1} \left(I - e^{-t\mathbf{K}/N} \right) (\mathbf{y} - \mathbf{f}_0).$$

B Omitted Proofs from Neural-Network Training Bias Toward Smooth Input Landscapes

B.1 Auxiliary lemmas

We begin with the following result about the distance matrices of G, which follows from a classical lemma of Delsarte (1973) and which we use in the proofs of Lemmas 1, 2, and 6.

Recall from Section 3.1 that G is isomorphic to $J(n, k)$ through the identification $x \mapsto \{i \in [n] : x_i = 1\}$. For each $\ell = 0, \dots, s$, let A_ℓ denote the distance-ℓ matrix of G, whose (u, v) entry equals one if u and v are at graph distance ℓ, and zero otherwise.

Lemma 3 (Delsarte (1973)) *The distance matrices $A_0, \dots, A_s$ all admit the same orthogonal eigenspace decomposition $\bigoplus_{r=0}^{s} \mathcal{E}_r$. The corresponding eigenvalue of A_ℓ on $\mathcal{E}_r$ is $E_\ell^{(n,k)}(r)$, the Eberlein polynomial evaluated at r.*

We next prove the following result about Gaussian expectations.

Lemma 4 *Let σ be nonpolynomial, Lipschitz, and twice differentiable, with bounded second derivative. For $q > 0$ and $0 \le v \le q$, define*

$$F_q(v) = \mathbb{E}[\sigma(U)\sigma(V)],$$

where (U, V) is a centered Gaussian pair with common variance q and covariance v. Then F_q is continuous on $[0, q]$ and infinitely differentiable on $(0, q)$, with

$$F_q'(v) = \mathbb{E}[\sigma'(U)\sigma'(V)], \qquad 0 < v < q.$$

Fix $0 < a < b < \infty$. For every integer $m \geq 0$, there is a constant $c_m > 0$ such that

$$F_q^{(m)}(v) \geq c_m, \qquad a \leq v < q \leq b.$$

If $m = 0$, or if σ has bounded derivatives through order m, there is also a constant $C_m > 0$ such that

$$F_q^{(m)}(v) \leq C_m, \qquad a \leq v < q \leq b.$$

Both constants depend only on m, σ, a, b. For $m = 0$, the bounds also hold when $v = q$.

Proof. We prove the claims in turn.

Well-definedness and continuity. We first show that F_q is well defined and continuous on $[0, q]$. Fix $q > 0$, and let Z, Z_1, Z_2 be independent standard Gaussian variables. For $v \in [0, q]$, set

$$U_v = \sqrt{v}\, Z + \sqrt{q - v}\, Z_1, \qquad V_v = \sqrt{v}\, Z + \sqrt{q - v}\, Z_2.$$

This pair is centered, with common variance q and covariance v, so $F_q(v) = \mathbb{E}[\sigma(U_v)\sigma(V_v)]$.

Since σ is Lipschitz, $|\sigma(z)| \leq |\sigma(0)| + K|z|$ for some constant K. Consequently,

$$|\sigma(U_v)\sigma(V_v)| \leq C_q(1 + Z^2 + Z_1^2 + Z_2^2),$$

where C_q depends only on q and σ. The right-hand side has finite expectation and does not depend on v. Thus $F_q(v)$ is finite. Moreover, $\sigma(U_v)\sigma(V_v)$ is continuous in v, so dominated convergence shows that F_q is continuous on $[0, q]$, including its endpoints.

Infinite differentiability. We now prove that F_q is infinitely differentiable on $(0, q)$. We do this by writing the expectation as an integral and differentiating the Gaussian density. Let $p_{q,v}(x, y)$ denote the density of (U_v, V_v), which exists for $0 < v < q$ because the determinant $q^2 - v^2$ is positive. Then

$$F_q(v) = \int_{\mathbb{R}^2} \sigma(x)\sigma(y)\, p_{q,v}(x, y)\, \mathrm{d}x\, \mathrm{d}y.$$

The Gaussian density formula shows that $p_{q,v}(x, y)$ is infinitely differentiable in v on $(0, q)$.

To justify differentiation under the integral, fix a closed interval $I \subset (0, q)$. Each differentiation of the Gaussian exponential introduces a quadratic factor in x, y. Repeated use of the product rule therefore gives a polynomial of degree at most $2m$ times the density for the mth derivative. Smoothness and nondegeneracy on I bound these polynomial coefficients and give

$$\left| \frac{\partial^m p_{q,v}}{\partial v^m}(x, y) \right| \leq C(1 + x^2 + y^2)^m e^{-c(x^2+y^2)}, \qquad v \in I,$$

for suitable constants $C, c > 0$ depending only on q, I, m. Since σ grows at most linearly, this bound remains integrable after multiplication by $|\sigma(x)\sigma(y)|$. We may therefore differentiate the expectation under the integral arbitrarily many times. The same bound gives continuity of each derivative by dominated convergence. Thus F_q is infinitely differentiable on $(0, q)$.

First-derivative formula. We now prove that $F_q'(v) = \mathbb{E}[\sigma'(U_v)\sigma'(V_v)]$ for $0 < v < q$. Direct differentiation of the Gaussian density gives

$$\frac{\partial p_{q,v}}{\partial v}(x, y) = \frac{\partial^2 p_{q,v}}{\partial x\, \partial y}(x, y).$$

Using differentiation under the integral, justified above, we obtain

$$F_q'(v) = \int_{\mathbb{R}^2} \sigma(x)\sigma(y) \frac{\partial^2 p_{q,v}}{\partial x\, \partial y}(x, y)\, \mathrm{d}x\, \mathrm{d}y.$$

Integrating by parts first in x and then in y transfers one derivative to each activation factor. The boundary terms vanish because the Gaussian density and its derivatives decay faster than any polynomial, while σ grows at most linearly and σ' is bounded. Therefore,

$$F_q'(v) = \int_{\mathbb{R}^2} \sigma'(x)\sigma'(y) p_{q,v}(x, y)\, \mathrm{d}x\, \mathrm{d}y = \mathbb{E}[\sigma'(U_v)\sigma'(V_v)].$$

Uniform lower bounds. We prove that for every $m \geq 0$, there is a constant $c_m > 0$, depending only on m, σ, a, b, such that

$$F_q^{(m)}(v) \geq c_m, \qquad a \leq v < q \leq b.$$

It suffices to prove that every derivative is strictly positive on $(0, q)$. Indeed, positivity of $F_q^{(m+1)}$ makes $F_q^{(m)}$ nondecreasing, giving

$$F_q^{(m)}(v) \geq F_q^{(m)}(a/2) \geq \min_{q' \in [a,b]} F_{q'}^{(m)}(a/2).$$

For $q \in [a, b]$, the Gaussian pair with covariance $a/2$ stays nondegenerate. The same density bounds used above show that $q \mapsto F_q^{(m)}(a/2)$ is continuous. Once strict positivity is proved, its minimum on $[a, b]$ is thus a positive constant c_m, proving the required bound.

To establish this positivity, fix $0 < v < q$. Repeated application of the density identity $\partial_v p_{q,v} = \partial_x \partial_y p_{q,v}$ gives

$$F_q^{(m)}(v) = \int_{\mathbb{R}^2} \sigma(x)\sigma(y) \frac{\partial^{2m} p_{q,v}}{\partial x^m \partial y^m}(x, y)\, \mathrm{d}x\, \mathrm{d}y.$$

Let φ_w denote the density of $\mathcal{N}(0, w)$. Conditional on $\sqrt{v}\, Z = z$, U_v and V_v are independent with distribution $\mathcal{N}(z, q - v)$. Therefore,

$$p_{q,v}(x, y) = \int_{\mathbb{R}} \varphi_v(z)\varphi_{q-v}(x - z)\varphi_{q-v}(y - z)\, \mathrm{d}z.$$

Substituting this expression and interchanging the integrals gives

$$F_q^{(m)}(v) = \int_{\mathbb{R}} \varphi_v(z) \left(\int_{\mathbb{R}} \sigma(x) \frac{\partial^m}{\partial x^m} \varphi_{q-v}(x - z)\, \mathrm{d}x \right)^2 \mathrm{d}z \geq 0.$$

Gaussian decay and the linear growth of σ justify these interchanges.

We now exclude equality. Define

$$A(z) = \int_{\mathbb{R}} \sigma(x)\varphi_{q-v}(x - z)\, \mathrm{d}x.$$

The inner integral in the squared expression is $(-1)^m A^{(m)}(z)$. Differentiating the Gaussian density in this integral shows that A is infinitely differentiable, using the same integrability argument as above.

If the squared integral were zero, continuity would imply $A^{(m)} = 0$ everywhere. Thus A would be a polynomial, or identically zero when $m = 0$. Since averaging preserves the Lipschitz property, A would necessarily be affine.

We show that this would force σ to be affine as well. Set $w = \sigma' - A'(0)$, which is bounded and continuous. Since A' is constant, differentiating the defining integral of A gives

$$\mathbb{E}[w(U_v) \mid Z] = A'(\sqrt{v}\, Z) - A'(0) = 0.$$

For $0 < u < q$, set

$$U_u = \sqrt{u}\, Z + \sqrt{q - u}\, Z_1, \qquad V_u = \sqrt{u}\, Z + \sqrt{q - u}\, Z_2,$$

and put $R(u) = \mathbb{E}[w(U_u)w(V_u)]$. Their conditional independence given Z gives

$$R(u) = \mathbb{E}\Big[\big(\mathbb{E}[w(U_u) \mid Z]\big)^2 \Big].$$

In particular, $R(v) = 0$. Whenever $R(u) = 0$, the conditional expectation in this square vanishes. The pair $(\sqrt{q}\, Z, U_u)$ has common variance q and covariance $\sqrt{qu}$, so

$$R(\sqrt{qu}) = \mathbb{E}[w(\sqrt{q}\, Z)\mathbb{E}[w(U_u) \mid Z]] = 0.$$

Starting from $u_0 = v$ and setting $u_{j+1} = \sqrt{qu_j}$, we therefore have $R(u_j) = 0$ and $u_j \to q$. Both U_{u_j} and V_{u_j} converge to $\sqrt{q}\, Z$. Bounded continuity of w consequently gives

$$0 = \lim_{j \to \infty} R(u_j) = \mathbb{E}[w(\sqrt{q}\, Z)^2].$$

Continuity implies $w = 0$ everywhere. Hence σ' is constant and σ is affine, contradicting nonpolynomiality. This proves strict positivity of every $F_q^{(m)}(v)$, and therefore the required uniform lower bound.

For $m = 0$, monotonicity and continuity also give $F_q(q) \geq F_q(a/2) \geq c_0$, covering the endpoint $v = q$.

Uniform upper bounds. We now prove that $F_q^{(m)}(v) \leq C_m$ for $a \leq v < q \leq b$, under the stated derivative assumptions.

For $m = 0$, Cauchy–Schwarz and the equal marginal distributions give

$$F_q(v) \leq \mathbb{E}[\sigma(\sqrt{q}\, Z)^2].$$

Since $|\sigma(z)| \leq |\sigma(0)| + K|z|$,

$$F_q(v) \leq 2\sigma(0)^2 + 2K^2 q \leq 2\sigma(0)^2 + 2K^2 b.$$

Thus we may take $C_0 = 2\sigma(0)^2 + 2K^2 b$. This argument also applies at $v = q$.

Now let $m \geq 1$ and assume that σ has bounded derivatives through order m. Starting from the density expression for $F_q^{(m)}(v)$, we integrate by parts m times in each variable. The boundary terms vanish by Gaussian decay, giving

$$F_q^{(m)}(v) = \mathbb{E}[\sigma^{(m)}(U_v)\sigma^{(m)}(V_v)] \leq \sup_{z \in \mathbb{R}} |\sigma^{(m)}(z)|^2.$$

The supremum is finite by assumption and provides the required constant C_m. □

Next, we prove the following lemma about the functions $\kappa^{(L)}$ and $\vartheta^{(L)}$ appearing in the formula for the NTK eigenvalues (Lemma 1). We use this result in the proofs of Lemmas 2 and 6 and Theorem 1.

Lemma 5 *The recursion in Lemma 1 is well defined for real arguments $t \in [0, s]$. The resulting functions $\kappa^{(L)}$ and $\vartheta^{(L)}$ are continuous on $[0, s]$ and infinitely differentiable on $(0, s)$.*

For each integer $m \geq 0$,

$$(-1)^m \frac{\mathrm{d}^m \kappa^{(L)}}{\mathrm{d}t^m}(t) \leq (-1)^m \frac{\mathrm{d}^m \vartheta^{(L)}}{\mathrm{d}t^m}(t), \qquad 0 < t < s.$$

There is a constant $c_m > 0$ such that

$$\frac{c_m}{n^m} \leq (-1)^m \frac{\mathrm{d}^m \kappa^{(L)}}{\mathrm{d}t^m}(t), \qquad 0 < t < s.$$

If $m = 0$, or if σ has bounded derivatives through order $m + 1$, there is also a constant $C_m > 0$ such that

$$(-1)^m \frac{\mathrm{d}^m \vartheta^{(L)}}{\mathrm{d}t^m}(t) \leq \frac{C_m}{n^m}, \qquad 0 < t < s.$$

Both constants depend only on m, σ, L, β. For $m = 0$, the bounds also hold at $t = 0$ and $t = s$.

Proof. We prove the lemma by induction on $L \geq 2$.

Base case $L = 2$. Set $q = k/n + \beta^2$ and $v = (k - t)/n + \beta^2$. The first-layer functions satisfy $\kappa^{(1)}(t) = \vartheta^{(1)}(t) = v$. The Gaussian pair $(U_{2,t}, V_{2,t})$ has common variance q and covariance v. Since $\beta^2 \leq v \leq q$, this pair is well defined for $t \in [0, s]$, and is nondegenerate for $0 < t < s$.

By definition of F_q, the first recursion gives

$$\kappa^{(2)}(t) = F_q(v) + \beta^2.$$

In the second recursion, Lemma 4 gives $\mathbb{E}[\sigma'(U_{2,t})\sigma'(V_{2,t})] = F_q'(v)$ for $0 < t < s$. Since $\vartheta^{(1)}(t) = v$, we obtain

$$\vartheta^{(2)}(t) = F_q(v) + \beta^2 + vF_q'(v).$$

Lemma 4 also states that F_q is infinitely differentiable on $(0, q)$, so both functions are infinitely differentiable on $(0, s)$.

We next verify continuity on $[0, s]$. Continuity of $\kappa^{(2)}$ follows from continuity of F_q in Lemma 4. For $\vartheta^{(2)}$, we must also show that $\mathbb{E}[\sigma'(U_{2,t})\sigma'(V_{2,t})]$ is continuous in t. To compute this expectation, we may choose independent standard Gaussian variables Z, Z' and set

$$U_{2,t} = \sqrt{q}\, Z, \qquad V_{2,t} = \frac{v}{\sqrt{q}} Z + \sqrt{q - \frac{v^2}{q}}\, Z'.$$

This pair has the required variances and covariance, and both variables depend continuously on $t \in [0, s]$. Since σ' is bounded and continuous, bounded convergence proves continuity of the expectation. The recursion

$$\vartheta^{(2)}(t) = \kappa^{(2)}(t) + v\, \mathbb{E}[\sigma'(U_{2,t})\sigma'(V_{2,t})]$$

therefore gives continuity of $\vartheta^{(2)}$ on $[0, s]$.

We next prove the following two inequalities for $m \geq 1$,

$$\frac{c_m}{n^m} \leq (-1)^m \frac{\mathrm{d}^m \kappa^{(2)}}{\mathrm{d}t^m}(t) \leq (-1)^m \frac{\mathrm{d}^m \vartheta^{(2)}}{\mathrm{d}t^m}(t).$$

Since $v' = -1/n$, differentiating gives

$$(-1)^m \frac{\mathrm{d}^m \kappa^{(2)}}{\mathrm{d}t^m}(t) = \frac{F_q^{(m)}(v)}{n^m},$$

$$(-1)^m \frac{\mathrm{d}^m \vartheta^{(2)}}{\mathrm{d}t^m}(t) = \frac{(m+1)F_q^{(m)}(v) + vF_q^{(m+1)}(v)}{n^m}.$$

Because $\beta^2 \leq v < q \leq 1 + \beta^2$, the lower bound in Lemma 4 gives $F_q^{(m)}(v) \geq c_m > 0$. The first formula therefore gives the required lower bound c_m/n^m. The second expression is at least the first, because $F_q^{(m)}(v) \geq 0$, $F_q^{(m+1)}(v) \geq 0$, and $v > 0$.

For the upper bound, suppose σ has bounded derivatives through order $m + 1$. Lemma 4, with $a = \beta^2$ and $b = 1 + \beta^2$, gives a constant B, depending only on m, σ, β, such that

$$F_q^{(m)}(v) \leq B, \qquad F_q^{(m+1)}(v) \leq B.$$

Substituting these bounds and $v \leq 1 + \beta^2$ into the formula for the derivative of $\vartheta^{(2)}$ gives

$$(-1)^m \frac{\mathrm{d}^m \vartheta^{(2)}}{\mathrm{d}t^m}(t) \leq \frac{(m+1)B + (1+\beta^2)B}{n^m} = \frac{(m+2+\beta^2)B}{n^m}.$$

Taking $C_m = (m + 2 + \beta^2)B$ proves the required upper bound.

For $m = 0$, we use the formulas for $\kappa^{(2)}$ and $\vartheta^{(2)}$. Positivity of $F_q(v)$ and $F_q'(v)$ gives

$$\kappa^{(2)}(t) = F_q(v) + \beta^2 \geq \beta^2,$$

$$\vartheta^{(2)}(t) - \kappa^{(2)}(t) = vF_q'(v) \geq 0.$$

Lemma 4 gives $F_q(v) \leq B_0$ for a constant B_0 depending only on σ, β. Its first-derivative formula also gives

$$0 \leq F_q'(v) \leq \sup_{z \in \mathbb{R}} |\sigma'(z)|^2 < \infty.$$

Consequently,

$$\vartheta^{(2)}(t) \leq B_0 + \beta^2 + (1 + \beta^2) \sup_{z \in \mathbb{R}} |\sigma'(z)|^2.$$

The right-hand side provides the constant C_0. Continuity extends the bounds for $m = 0$ to $t = 0, s$.

Induction step. Let $L \geq 3$ and assume the lemma holds for $L - 1$.

We first check that the Gaussian pair defining layer L is well defined on $[0, s]$ and nondegenerate on $(0, s)$. Set $q = \kappa^{(L-1)}(0)$. The bounds for $m = 0$ give positive constants a, b such that $a \leq$

$\kappa^{(L-1)}(t) \le \vartheta^{(L-1)}(t) \le b$ on $[0, s]$. The lower bound for $m = 1$ makes $\kappa^{(L-1)}$ strictly decreasing. Together with continuity, this gives

$$a \le \kappa^{(L-1)}(t) < q \le b, \qquad 0 < t \le s.$$

Thus the Gaussian pair $(U_{L,t}, V_{L,t})$ is well defined on $[0, s]$ and nondegenerate on $(0, s)$.

We next prove continuity and infinite differentiability at layer L. By definition of F_q, the first recursion is

$$\kappa^{(L)}(t) = F_q(\kappa^{(L-1)}(t)) + \beta^2.$$

In the second recursion, Lemma 4 identifies $\mathbb{E}[\sigma'(U_{L,t})\sigma'(V_{L,t})]$ with $F_q'(\kappa^{(L-1)}(t))$. Hence, for $0 < t < s$,

$$\vartheta^{(L)}(t) - \kappa^{(L)}(t) = F_q'(\kappa^{(L-1)}(t))\vartheta^{(L-1)}(t).$$

Lemma 4 gives infinite differentiability of F_q and F_q' on $(0, q)$. Together with the induction hypothesis, the two formulas therefore give infinite differentiability of $\kappa^{(L)}$ and $\vartheta^{(L)}$ on $(0, s)$. Continuity on $[0, s]$ follows by the Gaussian representation and bounded-convergence argument used in the base case, now with $v = \kappa^{(L-1)}(t)$.

We now prove the value bounds $c_0 \le \kappa^{(L)}(t) \le \vartheta^{(L)}(t) \le C_0$. Lemma 4 gives $0 \le F_q(v) \le C$ and $0 \le F_q'(v) \le C$ for $a \le v < q \le b$. The upper bound for F_q' uses only boundedness of σ'. Since $\vartheta^{(L-1)}(t) \le b$, the recursion gives

$$\beta^2 \le \kappa^{(L)}(t) \le \vartheta^{(L)}(t) \le C + \beta^2 + Cb.$$

These are the required bounds for $m = 0$, and continuity extends them to the endpoints.

For $m \ge 1$, we first prove the lower bound

$$(-1)^m \frac{\mathrm{d}^m \kappa^{(L)}}{\mathrm{d}t^m}(t) \ge \frac{c_m}{n^m}.$$

We will expand this derivative into nonnegative terms and bound one of them from below. Repeated application of the chain rule expresses $(-1)^m \mathrm{d}^m \kappa^{(L)}/\mathrm{d}t^m$ as a sum of positive multiples of

$$F_q^{(p)}(\kappa^{(L-1)}(t)) \prod_{i=1}^{p} \left[(-1)^{j_i} \frac{\mathrm{d}^{j_i} \kappa^{(L-1)}}{\mathrm{d}t^{j_i}}(t) \right], \qquad j_i \ge 1, \quad \sum_{i=1}^{p} j_i = m.$$

Every factor is nonnegative by Lemma 4 and the induction hypothesis. Keeping only the term with $p = m$ and $j_1 = \cdots = j_m = 1$ gives

$$(-1)^m \frac{\mathrm{d}^m \kappa^{(L)}}{\mathrm{d}t^m}(t) \ge F_q^{(m)}(\kappa^{(L-1)}(t)) \left(-\frac{\mathrm{d}\kappa^{(L-1)}}{\mathrm{d}t}(t) \right)^m.$$

Lemma 4 bounds the first factor below by a positive constant, while the inductive first-derivative bound gives $-\mathrm{d}\kappa^{(L-1)}/\mathrm{d}t \ge c_1/n$. Their product is therefore at least c_m/n^m.

We next prove that the corresponding derivative of $\vartheta^{(L)}$ is at least as large. Equivalently, we must show

$$(-1)^m \frac{\mathrm{d}^m}{\mathrm{d}t^m}(\vartheta^{(L)} - \kappa^{(L)})(t) \ge 0.$$

The second recursion expresses this difference as the product $F_q'(\kappa^{(L-1)}(t))\vartheta^{(L-1)}(t)$. Applying the same chain-rule expansion to $F_q'(\kappa^{(L-1)}(t))$ shows that its derivatives, multiplied by $(-1)^r$ at order r, are nonnegative. The induction hypothesis gives the same property for $\vartheta^{(L-1)}$. Every term in the product-rule expansion therefore has the required sign, proving the inequality.

Finally, suppose σ has bounded derivatives through order $m + 1$. We must prove the upper bound

$$(-1)^m \frac{\mathrm{d}^m \vartheta^{(L)}}{\mathrm{d}t^m}(t) \le \frac{C_m}{n^m}.$$

For this, we bound every term in the same chain-rule and product-rule expansions. Lemma 4 bounds every derivative of F_q appearing in these expansions. In each chain-rule term, the inductive upper bounds contribute a factor

$$n^{-j_1} \cdots n^{-j_p} = n^{-m}.$$

In the product-rule terms for $\vartheta^{(L)} - \kappa^{(L)}$, the orders r and $m-r$ likewise give $n^{-r}n^{-(m-r)} = n^{-m}$. Each term is thus bounded by Cn^{-m}. Summing the finitely many terms gives the required upper bound.

This completes the induction. □

We now give the proofs of the lemmas and theorems in Section 3.

B.2 Proof of Lemma 1

We show that $\mathbf{\Theta}$ acts on each $\mathcal{E}_r$ by multiplication by the stated eigenvalue μ_r. To do so, we first express $\mathbf{\Theta}$ as a linear combination of the distance matrices $A_0, \dots, A_s$ of G, whose action on these spaces is given by Lemma 3.

For $x, y \in \mathcal{X}$, their distance in G is $d_G(x, y) = k - x^\top y$. We first show that the recursions in Equation (4) and Lemma 1 give

$$\Sigma^{(\ell)}(x, y) = \kappa^{(\ell)}(d_G(x, y)), \qquad \Theta^{(\ell)}(x, y) = \vartheta^{(\ell)}(d_G(x, y)), \qquad \ell = 1, \dots, L.$$

At the first layer, both identities follow from $x^\top y/n + \beta^2 = (k - d_G(x, y))/n + \beta^2$. Suppose they hold at layer $\ell - 1$ for every pair of inputs in $\mathcal{X}$. For $d = d_G(x, y)$, the centered Gaussian pair in the covariance recursion then has common variance $\kappa^{(\ell-1)}(0)$ and covariance $\kappa^{(\ell-1)}(d)$. It therefore has the same distribution as $(U_{\ell,d}, V_{\ell,d})$ in Lemma 1, so the updates defining $\Sigma^{(\ell)}(x, y)$ and $\kappa^{(\ell)}(d)$ coincide. The NTK update uses the same Gaussian pair and $\Theta^{(\ell-1)}(x, y) = \vartheta^{(\ell-1)}(d)$, giving the second identity at layer ℓ. This completes the induction.

Thus, $\mathbf{\Theta}_{xy} = \vartheta(d_G(x, y))$, where $\vartheta = \vartheta^{(L)}$. By the definition of A_ℓ, we obtain

$$\mathbf{\Theta} = \sum_{\ell=0}^{s} \vartheta(\ell) A_\ell.$$

By Lemma 3, A_ℓ acts on $\mathcal{E}_r$ by multiplication by $E_\ell^{(n,k)}(r)$. Therefore, $\mathbf{\Theta}$ acts on $\mathcal{E}_r$ by multiplication by

$$\mu_r = \sum_{\ell=0}^{s} \vartheta(\ell) E_\ell^{(n,k)}(r).$$

Finally, $\mathbf{\Theta}$ is the limit in probability of the empirical tangent kernel matrices, which are positive semidefinite (Appendix A). It is therefore positive semidefinite, giving $\mu_r \geq 0$. □

B.3 Proof of Lemma 2

Both parts follow from a bound on the NTK spectrum. We first prove this bound, then use it to control the surviving initialization g and the learned component h separately. Throughout the proof, C denotes a positive constant depending only on q, σ, L, β, whose value may increase between inequalities.

A bound on a weighted sum of the NTK eigenvalues. Our first goal is to show that

$$\frac{1}{|\mathcal{X}|} \sum_{r=1}^{s} r^{q-1} d_r \mu_r \leq C. \tag{7}$$

Let $1 \leq m \leq s$, with $m \leq q - 1$, and write $(a)_m = a(a-1)\cdots(a-m+1)$. We first bound the sum with weight r^m, then take $m = q - 1$.

We upper-bound the weight r^m by

$$r^m \leq C \left(1 + \frac{(r)_m (n - r + 1)_m}{n^m}\right).$$

For $r < m$, r^m is bounded by a constant depending only on m. For $r \geq m$, we have $(r)_m \geq (r/m)^m$ and $(n - r + 1)_m \geq (n/(2m))^m$, because $r \leq n/2$.

Since the eigenvalues μ_r are nonnegative, this gives

$$\frac{1}{|\mathcal{X}|}\sum_{r=1}^{s} r^m d_r \mu_r \le \frac{C}{|\mathcal{X}|}\sum_{r=0}^{s} d_r \mu_r + \frac{C}{n^m|\mathcal{X}|}\sum_{r=m}^{s} (r)_m (n-r+1)_m d_r \mu_r.$$

We now substitute the expression from Lemma 1,

$$\mu_r = \sum_{\ell=0}^{s} \vartheta(\ell) E_\ell^{(n,k)}(r),$$

and interchange the sums. The resulting upper bound is

$$\frac{C}{|\mathcal{X}|}\sum_{\ell=0}^{s}\vartheta(\ell)\sum_{r=0}^{s} d_r E_\ell^{(n,k)}(r) + \frac{C}{n^m|\mathcal{X}|}\sum_{\ell=0}^{s}\vartheta(\ell)\sum_{r=m}^{s}(r)_m(n-r+1)_m d_r E_\ell^{(n,k)}(r).$$

We evaluate the two inner sums separately.

By Lemma 3, the first inner sum is the trace of A_ℓ. Since $A_0 = I$ and A_ℓ has zero diagonal for $\ell > 0$,

$$\sum_{r=0}^{s} d_r E_\ell^{(n,k)}(r) = \begin{cases} |\mathcal{X}|, & \ell = 0, \\ 0, & \ell > 0. \end{cases}$$

Thus the first term is simply $C\vartheta(0)$.

For the second inner sum, set $r = j + m$. Substituting the formula for d_r and cancelling factorials gives

$$(j+m)_m(n-j-m+1)_m d_{j+m} = (n)_{2m}\left[\binom{n-2m}{j} - \binom{n-2m}{j-1}\right].$$

The Eberlein formula also gives

$$E_\ell^{(n,k)}(j+m) = \sum_{a=0}^{\min(m,\ell)} (-1)^a \binom{m}{a} E_{\ell-a}^{(n-2m,k-m)}(j).$$

This follows by expanding $\binom{j+m}{h}$ as $\sum_a \binom{m}{a}\binom{j}{h-a}$ in its defining formula.

Substituting these two identities and interchanging the sums over j and a, we obtain an inner sum of the form

$$\sum_{j=0}^{s-m}\left[\binom{n-2m}{j} - \binom{n-2m}{j-1}\right] E_{\ell-a}^{(n-2m,k-m)}(j).$$

This is exactly the first inner sum with parameters $n-2m, k-m$ and distance $\ell - a$. It therefore equals

$$\begin{cases} \binom{n-2m}{k-m}, & \ell = a, \\ 0, & \ell > a. \end{cases}$$

Only the term $a = \ell$ survives, giving

$$\sum_{r=m}^{s}(r)_m(n-r+1)_m d_r E_\ell^{(n,k)}(r) = |\mathcal{X}|(k)_m(n-k)_m(-1)^\ell\binom{m}{\ell},$$

where $\binom{m}{\ell} = 0$ for $\ell > m$. Here we used

$$(n)_{2m}\binom{n-2m}{k-m} = |\mathcal{X}|(k)_m(n-k)_m.$$

Our bound therefore reduces to

$$\frac{1}{|\mathcal{X}|}\sum_{r=1}^{s} r^m d_r \mu_r \le C\vartheta(0) + \frac{C(k)_m(n-k)_m}{n^m}\left|\sum_{\ell=0}^{m}(-1)^\ell\binom{m}{\ell}\vartheta(\ell)\right|. \tag{8}$$

We now bound the two quantities on the right-hand side of (8). Lemma 5 gives $\vartheta(0) \le C$. For the finite sum, we use the mean-value theorem to express it as a derivative, which Lemma 5 also bounds. Specifically, for $1 \le m \le \min(s, q-1)$, repeated application of the mean-value theorem gives

$$\left|\sum_{\ell=0}^{m}(-1)^{\ell}\binom{m}{\ell}\vartheta(\ell)\right| = \left|\frac{\mathrm{d}^m\vartheta}{\mathrm{d}t^m}(\xi)\right| \le Cn^{-m}$$

for some $\xi \in (0, m)$. The last inequality follows from Lemma 5, since σ has bounded derivatives through order $q \ge m+1$.

Substituting both bounds into (8) and using $(k)_m(n-k)_m \le n^{2m}$ gives

$$\frac{1}{|\mathcal{X}|}\sum_{r=1}^{s} r^m d_r\mu_r \le C.$$

Taking $m = q-1$ then proves Equation (7) when $q-1 \le s$.

When $q-1 > s$, we instead use $r^{q-1} \le (q-1)^{q-1}$ and $\mu_r \ge 0$ to obtain

$$\frac{1}{|\mathcal{X}|}\sum_{r=1}^{s} r^{q-1} d_r\mu_r \le \frac{(q-1)^{q-1}}{|\mathcal{X}|}\sum_{r=0}^{s} d_r\mu_r = (q-1)^{q-1}\vartheta(0) \le C.$$

Part (i). The surviving initialization. We will prove $\mathbb{E}_{\theta_0}\|\Pi_r g\|^2 \le d_r\mu_r/|\mathcal{X}|$. Summing these inequalities with weights r^{q-1} will give part (i) by Equation (7).

We first prove $\mathbf{\Sigma} \preceq \mathbf{\Theta}$, meaning that $\mathbf{\Theta} - \mathbf{\Sigma}$ is positive semidefinite.

Let $\mathbf{\Sigma}^{(\ell)}$ and $\mathbf{\Theta}^{(\ell)}$ denote the covariance and NTK matrices at layer ℓ. At the first layer, $\mathbf{\Theta}^{(1)} = \mathbf{\Sigma}^{(1)} \succeq 0$.

Suppose $\mathbf{\Theta}^{(\ell-1)} \succeq 0$. We show that the additional term in the NTK recursion is positive semidefinite. Let $\mathbf{Z} \sim \mathcal{N}(0, \mathbf{\Sigma}^{(\ell-1)})$, and let $\mathbf{D_Z}$ be diagonal with entries $\sigma'(Z_x)$. The recursion (4) gives

$$\mathbf{\Theta}^{(\ell)} - \mathbf{\Sigma}^{(\ell)} = \mathbb{E}\Big[\mathbf{D_Z}\mathbf{\Theta}^{(\ell-1)}\mathbf{D_Z}\Big].$$

Indeed, multiplying by these diagonal matrices multiplies each (x, y) entry by $\sigma'(Z_x)\sigma'(Z_y)$, exactly as in the recursion.

For every vector v,

$$v^\top\Big(\mathbf{\Theta}^{(\ell)} - \mathbf{\Sigma}^{(\ell)}\Big)v = \mathbb{E}\Big[(\mathbf{D_Z}v)^\top\mathbf{\Theta}^{(\ell-1)}(\mathbf{D_Z}v)\Big] \ge 0.$$

Thus $\mathbf{\Theta}^{(\ell)} \succeq \mathbf{\Sigma}^{(\ell)} \succeq 0$, where the second inequality holds because $\mathbf{\Sigma}^{(\ell)}$ is a covariance matrix. This completes the induction. Taking $\ell = L$ gives $\mathbf{\Sigma} \preceq \mathbf{\Theta}$.

Next, we show that the covariance of the surviving initialization g is also bounded by $\mathbf{\Theta}$.

Let $\mathbf{F}_0 = (f_0(x))_{x\in\mathcal{X}}$, and let $\mathbf{S}$ extract the training entries from this vector. Then $\mathbf{K} = \mathbf{S}\mathbf{\Theta}\mathbf{S}^\top$, and the definition of g gives

$$\mathbf{g} = \mathbf{A}\mathbf{F}_0, \qquad \mathbf{A} = I - \mathbf{\Theta}\mathbf{S}^\top\mathbf{K}^{-1}\mathbf{S}.$$

Because $\mathbf{F}_0$ is centered with covariance $\mathbf{\Sigma}$, the covariance of g is $\mathbf{A}\mathbf{\Sigma}\mathbf{A}^\top$. Using the comparison we just proved,

$$\mathrm{Cov}(\mathbf{g}) = \mathbf{A}\mathbf{\Sigma}\mathbf{A}^\top \preceq \mathbf{A}\mathbf{\Theta}\mathbf{A}^\top.$$

We now simplify the right-hand side. Expanding the product and using $\mathbf{S}\mathbf{\Theta}\mathbf{S}^\top = \mathbf{K}$ gives

$$\mathbf{A}\mathbf{\Theta}\mathbf{A}^\top = \mathbf{\Theta} - \mathbf{\Theta}\mathbf{S}^\top\mathbf{K}^{-1}\mathbf{S}\mathbf{\Theta}.$$

The subtracted matrix is positive semidefinite because $\mathbf{K}^{-1}$ is positive definite. Therefore,

$$\mathrm{Cov}(\mathbf{g}) \preceq \mathbf{\Theta}.$$

Now we use $\mathrm{Cov}(\mathbf{g}) \preceq \mathbf{\Theta}$ to bound the expected squared weight of g at each level.

Projecting onto $\mathcal{E}_r$ gives

$$\mathrm{Cov}(\Pi_r \mathbf{g}) \preceq \Pi_r \boldsymbol{\Theta} \Pi_r = \mu_r \Pi_r.$$

The last equality holds because $\boldsymbol{\Theta}$ acts as multiplication by μ_r on $\mathcal{E}_r$.

For a centered random vector, the expected squared Euclidean norm is the sum of its coordinate variances, namely the trace of its covariance. Our function norm also divides by $|\mathcal{X}|$. Therefore,

$$\mathbb{E}_{\theta_0} \|\Pi_r g\|^2 = \frac{\mathrm{tr}(\mathrm{Cov}(\Pi_r \mathbf{g}))}{|\mathcal{X}|} \leq \frac{\mu_r \,\mathrm{tr}(\Pi_r)}{|\mathcal{X}|} = \frac{d_r \mu_r}{|\mathcal{X}|},$$

where $\mathrm{tr}(\Pi_r) = d_r$ because Π_r projects onto a space of dimension d_r.

Finally, multiply by r^{q-1}, sum over the levels, and apply (7),

$$\mathbb{E}_{\theta_0} \left[\sum_{r=1}^{s} r^{q-1} \|\Pi_r g\|^2 \right] \leq \frac{1}{|\mathcal{X}|} \sum_{r=1}^{s} r^{q-1} d_r \mu_r \leq C.$$

This proves part (i).

Part (ii). The learned component. We want to prove

$$\sum_{r=1}^{s} r^{q-1} d_r \|\Pi_r h\|^2 \leq C \|\mathbf{y}\|^2_{\mathbf{K}^{-1}}.$$

Set $\mathbf{a} = \mathbf{K}^{-1}\mathbf{y}$. Using the same matrix $\mathbf{S}$ as above, the definition of h gives

$$\mathbf{h} = (h(x))_{x \in \mathcal{X}} = \boldsymbol{\Theta} \mathbf{S}^\top \mathbf{a}.$$

Since $\boldsymbol{\Theta}$ acts as multiplication by μ_r on $\mathcal{E}_r$,

$$\|\Pi_r h\|^2 = \frac{\mu_r^2}{|\mathcal{X}|} \|\Pi_r \mathbf{S}^\top \mathbf{a}\|_2^2,$$

where $\|\cdot\|_2$ is the Euclidean vector norm.

We now use (7) to bound one factor of μ_r. Every term in that sum is nonnegative, so $r^{q-1} d_r \mu_r / |\mathcal{X}| \leq C$. Multiplying the preceding identity by $r^{q-1} d_r$ and summing therefore gives

$$\sum_{r=1}^{s} r^{q-1} d_r \|\Pi_r h\|^2 \leq C \sum_{r=1}^{s} \mu_r \|\Pi_r \mathbf{S}^\top \mathbf{a}\|_2^2.$$

To finish, we bound the last sum. Adding the nonnegative $r = 0$ term lets us use the full eigendecomposition of $\boldsymbol{\Theta}$,

$$\sum_{r=1}^{s} \mu_r \|\Pi_r \mathbf{S}^\top \mathbf{a}\|_2^2 \leq \mathbf{a}^\top \mathbf{S} \boldsymbol{\Theta} \mathbf{S}^\top \mathbf{a} = \mathbf{a}^\top \mathbf{K} \mathbf{a}.$$

Finally, $\mathbf{a} = \mathbf{K}^{-1}\mathbf{y}$ implies $\mathbf{a}^\top \mathbf{K} \mathbf{a} = \|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$. This proves part (ii). □

B.4 Examples for the Label-Dependent Factor

Lemma 6 *Assume the conditions of Lemma 2 and $s \geq 2$. Let u be a target on $\mathcal{X}$ with mean zero and $\|u\|^2 = 1$, and set $y_i = u(x_i)$.*

For a linear target $u \in \mathcal{E}_1$, there is a constant $C_{\mathrm{lin}} = C_{\mathrm{lin}}(\sigma, L, \beta)$ such that every training subset satisfies

$$\frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{d_r} \leq C_{\mathrm{lin}}, \qquad \|\Pi_r h\|^2 \leq \frac{C C_{\mathrm{lin}}}{r^{q-1}}, \qquad r = 2, \ldots, s.$$

For linear targets, the bound therefore decays polynomially beyond the first level.

In contrast, if $u \in \mathcal{E}_s$ and we train on all of $\mathcal{X}$, then $h = u$ and

$$\|\Pi_s h\|^2 = 1, \qquad \frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{d_s} = \frac{|\mathcal{X}|}{d_s \mu_s} \geq \frac{s^{q-1}}{C}.$$

Here the factor grows enough to allow h to reproduce the target's highest-level component.

Proof. For linear targets, we first lower-bound μ_1, then use this to bound $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}/d_r$.

Setting $r = 1$ in the Eberlein formula gives

$$E_d^{(n,k)}(1) = \binom{k-1}{d}\binom{n-k-1}{d} - \binom{k-1}{d-1}\binom{n-k-1}{d-1}.$$

Substituting into the expression for μ_1 and collecting consecutive terms yields

$$\mu_1 = \sum_{d=0}^{s-1} \binom{k-1}{d}\binom{n-k-1}{d}\big(\vartheta(d) - \vartheta(d+1)\big).$$

All weights are nonnegative. We will show that each difference is at least c/n, for some $c > 0$ depending only on σ, L, β.

Lemma 5, with $m = 1$, gives

$$-\frac{\mathrm{d}\vartheta}{\mathrm{d}t}(t) \geq \frac{c}{n}, \qquad 0 < t < s.$$

For each $d = 0, \ldots, s-1$, the mean-value theorem therefore gives

$$\vartheta(d) - \vartheta(d+1) = -\frac{\mathrm{d}\vartheta}{\mathrm{d}t}(\xi) \geq \frac{c}{n}$$

for some $\xi \in (d, d+1)$. Substituting into the formula for μ_1, and using

$$\sum_{d=0}^{s-1} \binom{k-1}{d}\binom{n-k-1}{d} = \binom{n-2}{k-1},$$

gives

$$\frac{\mu_1}{|\mathcal{X}|} \geq c\,\frac{k(n-k)}{n^2(n-1)} \geq \frac{c}{n^2}.$$

The last inequality uses $k(n-k) \geq n-1$.

We now use this eigenvalue bound to control the label norm. Let $\mathbf{u} = (u(x))_{x\in\mathcal{X}}$. Our normalization $\|u\|^2 = 1$ means

$$\|\mathbf{u}\|_2^2 = \sum_{x\in\mathcal{X}} u(x)^2 = |\mathcal{X}|.$$

With $\mathbf{S}$ selecting the training inputs and $\mathbf{a} = \mathbf{K}^{-1}\mathbf{y}$, we have $\mathbf{y} = \mathbf{S}\mathbf{u}$ and $\mathbf{K} = \mathbf{S}\mathbf{\Theta}\mathbf{S}^\top$. Since $\mathbf{u} \in \mathcal{E}_1$, we have $\mathbf{a}^\top\mathbf{y} = \mathbf{u}^\top\Pi_1\mathbf{S}^\top\mathbf{a}$. Cauchy–Schwarz and $\mathbf{\Theta} \succeq \mu_1\Pi_1$ therefore give

$$(\mathbf{a}^\top\mathbf{y})^2 \leq |\mathcal{X}|\,\|\Pi_1\mathbf{S}^\top\mathbf{a}\|_2^2 \leq \frac{|\mathcal{X}|}{\mu_1}\mathbf{a}^\top\mathbf{K}\mathbf{a}.$$

Both $\mathbf{a}^\top\mathbf{y}$ and $\mathbf{a}^\top\mathbf{K}\mathbf{a}$ equal $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$. Dividing by this quantity when it is positive gives

$$\|\mathbf{y}\|^2_{\mathbf{K}^{-1}} \leq \frac{|\mathcal{X}|}{\mu_1} \leq \frac{n^2}{c}.$$

The same inequality holds immediately when $\mathbf{y} = 0$.

Finally, $d_r \geq n^2/8$ for $2 \leq r \leq s$. For $r = 2$, this follows from $d_2 = n(n-3)/2$ and $n \geq 4$. For $3 \leq r \leq s \leq n/2$,

$$d_r = \binom{n}{r}\frac{n-2r+1}{n-r+1} \geq \frac{\binom{n}{3}}{n-2} = \frac{n(n-1)}{6} \geq \frac{n^2}{8}.$$

Hence, with $C_{\mathrm{lin}} = 8/c$,

$$\frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{d_r} \leq C_{\mathrm{lin}}, \qquad \|\Pi_r h\|^2 \leq \frac{CC_{\mathrm{lin}}}{r^{q-1}},$$

where the second inequality is Lemma 2(ii).

For the highest-level target, we only need to compute the label norm under full training. Here $\mathbf{K} = \mathbf{\Theta}$ and $h = u$, so $\|\Pi_s h\|^2 = \|u\|^2 = 1$. Since $\mathbf{y} \in \mathcal{E}_s$, the matrix $\mathbf{K}^{-1}$ multiplies $\mathbf{y}$ by $1/\mu_s$. Therefore,

$$\frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{d_s} = \frac{\|\mathbf{y}\|_2^2}{d_s\mu_s} = \frac{|\mathcal{X}|}{d_s\mu_s} \geq \frac{s^{q-1}}{C},$$

where the last inequality is the $r = s$ term of the weighted spectral bound in Equation (7). This proves the second case. □

B.5 Proof of Theorem 1

Proof. We prove that $\mathbb{E}_{\mathcal{T},\theta_0}[1-\rho_\infty] \le C/n$. Write $V = \mathrm{Var}_{\mathcal{X}}(f_\infty)$. Throughout the proof, c, C are positive constants depending only on $a, \alpha, \sigma, L, \beta$; their values may change between inequalities.

For $V > 0$, the autocorrelation identity (6) and $k(n-k) \ge \alpha(1-\alpha)n^2$ give

$$1-\rho_\infty = \frac{\sum_{r=1}^{s}(1-\lambda_r)\|\Pi_r f_\infty\|^2}{V} \le \frac{1}{\alpha(1-\alpha)n}\frac{\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2}{V}.$$

It therefore suffices to prove

$$\mathbb{E}_{\mathcal{T},\theta_0}\left[\frac{\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2}{\mathrm{Var}_{\mathcal{X}}(f_\infty)}\right] \le C.$$

We will also show that $V > 0$ almost surely, so this ratio is well defined.

We first separate the low levels, whose contribution is already bounded. Fix a positive integer R, to be chosen below independently of n. Since $V = \sum_{r=1}^{s}\|\Pi_r f_\infty\|^2$, we have

$$\sum_{r\le R} r\|\Pi_r f_\infty\|^2 \le RV.$$

For the remaining levels, we use $f_\infty = h + g$ and

$$\|\Pi_r f_\infty\|^2 \le 2\|\Pi_r h\|^2 + 2\|\Pi_r g\|^2.$$

Combining these bounds gives

$$\frac{\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2}{V} \le R + 2\frac{\sum_{r>R} r\|\Pi_r h\|^2}{V} + 2\frac{\sum_{r>R} r\|\Pi_r g\|^2}{V}. \tag{9}$$

It therefore remains to bound the expectation of each of the last two fractions by a constant.

For the learned component, we first bound the numerator in terms of the label-dependent factor. Lemma 2(ii), with $q = 2$, gives

$$\sum_{r=1}^{s} r\, d_r\|\Pi_r h\|^2 \le C\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}.$$

To remove the factors d_r, we bound them from below for $r > R$. For sufficiently large n and $R < r \le s \le n/2$, the dimension formula gives

$$d_r = \binom{n}{r}\frac{n-2r+1}{n-r+1} \ge \frac{1}{n+1}\binom{n}{R+1} \ge cn^R.$$

Here we used that $\binom{n}{r}$ is nondecreasing for $r \le n/2$, and that R is fixed. Consequently,

$$\frac{\sum_{r>R} r\|\Pi_r h\|^2}{V} \le Cn^{-R}\frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{V}. \tag{10}$$

We will first compare $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$ with the variance of h. We will then show that $\mathbb{E}_{\theta_0}[\mathrm{Var}_{\mathcal{X}}(h)/V] \le C$, using the surviving initialization to lower-bound V. Together, these bounds will control the left-hand side of (10).

We next compare $\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}$ with $\mathrm{Var}_{\mathcal{X}}(h)$ by bounding both using the same coefficients $\mathbf{a} = \mathbf{K}^{-1}\mathbf{y}$.

First, we bound the label-dependent factor from above. Lemma 5, with $m = 0$, gives $\mathbf{K}_{ii} = \vartheta(0) \le C$, so $\mathrm{tr}(\mathbf{K}) \le CN$. Since $\mathbf{K}$ is positive semidefinite, its largest eigenvalue is at most its trace. Therefore,

$$\|\mathbf{y}\|^2_{\mathbf{K}^{-1}} = \mathbf{a}^\top\mathbf{K}\mathbf{a} \le \mathrm{tr}(\mathbf{K})\|\mathbf{a}\|_2^2 \le CN\|\mathbf{a}\|_2^2.$$

We now need a lower bound on $\mathrm{Var}_{\mathcal{X}}(h)$ in terms of the same $\|\mathbf{a}\|_2^2$.

To lower-bound $\mathrm{Var}_{\mathcal{X}}(h)$, we retain its component in one eigenspace $\mathcal{E}_j$. Fix an integer $j \ge 1$, whose value will be chosen below. For sufficiently large n, we have $j \le s$ because $s \ge \alpha n$. Since variance is the sum of the squared weights at the nonconstant levels,

$$\mathrm{Var}_{\mathcal{X}}(h) \ge \|\Pi_j h\|^2.$$

We now express this lower bound using the coefficients $\mathbf{a}$. Let $\mathbf{S}$ select the training entries from vectors indexed by $\mathcal{X}$, and set $\mathbf{G} = \mathbf{S}\Pi_j\mathbf{S}^\top$. The vector of values of h is

$$\mathbf{h} = \mathbf{\Theta}\mathbf{S}^\top\mathbf{a}.$$

Since $\Pi_j\mathbf{\Theta} = \mu_j\Pi_j$ and $\Pi_j^2 = \Pi_j$, we obtain

$$\mathrm{Var}_{\mathcal{X}}(h) \geq \|\Pi_j h\|^2 = \frac{\mu_j^2}{|\mathcal{X}|}\mathbf{a}^\top\mathbf{G}\mathbf{a}.$$

We will lower-bound μ_j using Lemma 5, then bound $\mathbf{a}^\top\mathbf{G}\mathbf{a}$ from below in terms of $\|\mathbf{a}\|_2^2$ using uniform sampling.

We first show that $\mu_j/|\mathcal{X}| \geq cn^{-j}$. Substituting the formula for the Eberlein polynomials into the expression for μ_j, and setting $t = \ell - h$, gives

$$\mu_j = \sum_{t=0}^{s-j}\binom{k-j}{t}\binom{n-k-j}{t}\sum_{h=0}^{j}(-1)^h\binom{j}{h}\vartheta(t+h).$$

We bound each inner sum using Lemma 5. Repeated application of the mean-value theorem gives

$$\sum_{h=0}^{j}(-1)^h\binom{j}{h}\vartheta(t+h) = (-1)^j\frac{\mathrm{d}^j\vartheta}{\mathrm{d}t^j}(\xi_t) \geq \frac{c_j}{n^j}, \qquad \xi_t \in (t, t+j).$$

Since the outer weights are nonnegative, we obtain

$$\mu_j \geq \frac{c_j}{n^j}\sum_{t=0}^{s-j}\binom{k-j}{t}\binom{n-k-j}{t} = \frac{c_j}{n^j}\binom{n-2j}{k-j}.$$

To compare this with $|\mathcal{X}| = \binom{n}{k}$, observe that

$$\frac{\binom{n-2j}{k-j}}{\binom{n}{k}} = \frac{\prod_{i=0}^{j-1}(k-i)(n-k-i)}{\prod_{i=0}^{2j-1}(n-i)} \geq \left(\frac{\alpha}{2}\right)^{2j}$$

for sufficiently large n, because j is fixed and $\min(k, n-k) \geq \alpha n$. Consequently,

$$\frac{\mu_j}{|\mathcal{X}|} \geq \frac{c}{n^j},$$

where $c > 0$ depends only on $j, \alpha, \sigma, L, \beta$.

We next use uniform sampling to show that $\mathbf{a}^\top\mathbf{G}\mathbf{a}$ is unlikely to be small.

Coordinate permutations preserve $\mathcal{E}_j$ and carry any point of $\mathcal{X}$ to any other. The diagonal entries of Π_j are therefore equal and, since $\mathrm{tr}(\Pi_j) = d_j$, each equals $d_j/|\mathcal{X}|$. Moreover, $\Pi_j^2 = \Pi_j$ gives

$$\sum_{x\neq y}(\Pi_j)_{xy}^2 = d_j - \frac{d_j^2}{|\mathcal{X}|}.$$

Each ordered pair of distinct training inputs is uniformly distributed over the $|\mathcal{X}|(|\mathcal{X}|-1)$ distinct pairs. Consequently,

$$\mathbb{E}_{\mathcal{T}}\left\|\frac{|\mathcal{X}|}{d_j}\mathbf{G} - \mathbf{I}_N\right\|_{\mathrm{F}}^2 = \frac{|\mathcal{X}|^2}{d_j^2}\mathbb{E}_{\mathcal{T}}\sum_{i\neq\ell}(\Pi_j)_{x_i x_\ell}^2 = N(N-1)\frac{|\mathcal{X}| - d_j}{d_j(|\mathcal{X}|-1)} \leq \frac{N^2}{d_j},$$

where $\|\cdot\|_{\mathrm{F}}^2$ is the sum of the squared matrix entries.

Let $\mathcal{B}$ be the event that $\left\|(|\mathcal{X}|/d_j)\mathbf{G} - \mathbf{I}_N\right\|_{\mathrm{F}} \leq 1/2$. Markov's inequality gives

$$\mathbb{P}_{\mathcal{T}}(\mathcal{B}^c) \leq \frac{4N^2}{d_j}. \tag{11}$$

On $\mathcal{B}$, Cauchy–Schwarz gives

$$\left|\frac{|\mathcal{X}|}{d_j}\mathbf{a}^\top\mathbf{G}\mathbf{a} - \|\mathbf{a}\|_2^2\right| \le \left\|\frac{|\mathcal{X}|}{d_j}\mathbf{G} - \mathbf{I}_N\right\|_{\mathrm{F}} \|\mathbf{a}\|_2^2 \le \frac{1}{2}\|\mathbf{a}\|_2^2.$$

Thus

$$\mathbf{a}^\top\mathbf{G}\mathbf{a} \ge \frac{d_j}{2|\mathcal{X}|}\|\mathbf{a}\|_2^2.$$

For fixed j, the formula $d_j = \binom{n}{j} - \binom{n}{j-1}$ gives $cn^j \le d_j \le Cn^j$ for sufficiently large n. Combining the preceding bounds therefore yields, on $\mathcal{B}$,

$$\mathrm{Var}_{\mathcal{X}}(h) \ge \frac{d_j\mu_j^2}{2|\mathcal{X}|^2}\|\mathbf{a}\|_2^2 \ge cn^{-j}\|\mathbf{a}\|_2^2.$$

Together with $\|\mathbf{y}\|_{\mathbf{K}^{-1}}^2 \le CN\|\mathbf{a}\|_2^2$, this proves the comparison we needed,

$$\|\mathbf{y}\|_{\mathbf{K}^{-1}}^2 \le CNn^j\,\mathrm{Var}_{\mathcal{X}}(h) \qquad \text{on } \mathcal{B}. \tag{12}$$

We now need to control $\mathrm{Var}_{\mathcal{X}}(h)/V$. We first lower-bound V using a component of the initialization that training leaves unchanged and that h cannot cancel.

Fix the training inputs and consider the subspace

$$\mathcal{F} = \{v \in \mathcal{E}_j : \mathbf{S}v = 0\}.$$

These are the functions in $\mathcal{E}_j$ that vanish at every training input. Since these are at most N linear constraints, $\dim\mathcal{F} \ge d_j - N$. All expectations over initialization below are conditional on these inputs.

To see why training does not change the component in $\mathcal{F}$, take $v \in \mathcal{F}$ and $\mathbf{b} \in \mathbb{R}^N$. Because $\boldsymbol{\Theta}v = \mu_j v$, we have

$$v^\top\boldsymbol{\Theta}\mathbf{S}^\top\mathbf{b} = \mu_j(\mathbf{S}v)^\top\mathbf{b} = 0.$$

Both h and the correction subtracted from f_0 in the definition of g have the form $\boldsymbol{\Theta}\mathbf{S}^\top\mathbf{b}$. They are therefore orthogonal to $\mathcal{F}$.

Let $Z = \Pi_{\mathcal{F}} f_0$ be the orthogonal projection of f_0 onto $\mathcal{F}$. The preceding orthogonality shows that

$$g = Z + \zeta, \qquad \zeta \perp \mathcal{F}, \qquad h \perp \mathcal{F}.$$

Thus Z is orthogonal to $h + \zeta$. It also has zero average over $\mathcal{X}$, because $\mathcal{F} \subseteq \mathcal{E}_j$ and $j \ge 1$. Consequently,

$$V = \mathrm{Var}_{\mathcal{X}}(Z + h + \zeta) = \|Z\|^2 + \mathrm{Var}_{\mathcal{X}}(h + \zeta) \ge \|Z\|^2.$$

We next show that $\mathbb{E}_{\theta_0}[1/\|Z\|^2] \le C$. This will allow us to use the lower bound $V \ge \|Z\|^2$ inside expectations.

Let $\boldsymbol{\Sigma} = (\Sigma(x,y))_{x,y\in\mathcal{X}}$ be the covariance matrix of the values of f_0 on $\mathcal{X}$. To determine the covariance of Z, we first prove that

$$\Sigma^{(\ell)}(x,y) = \kappa^{(\ell)}(d_G(x,y)), \qquad x,y \in \mathcal{X}, \quad \ell = 1,\dots,L.$$

For $x, y \in \mathcal{X}$, the first-layer formula gives

$$\Sigma^{(1)}(x,y) = \frac{x^\top y}{n} + \beta^2 = \frac{k - d_G(x,y)}{n} + \beta^2 = \kappa^{(1)}(d_G(x,y)).$$

Suppose that $\Sigma^{(\ell-1)}(x,y) = \kappa^{(\ell-1)}(d_G(x,y))$ for every pair $x, y \in \mathcal{X}$. Fix $x, y \in \mathcal{X}$ and set $d = d_G(x,y)$.

Equation (4) and Lemma 1 define the next-layer values by

$$\Sigma^{(\ell)}(x,y) = \mathbb{E}[\sigma(U_\ell)\sigma(V_\ell)] + \beta^2,$$
$$\kappa^{(\ell)}(d) = \mathbb{E}[\sigma(U_{\ell,d})\sigma(V_{\ell,d})] + \beta^2.$$

By the induction hypothesis, the pair (U_ℓ, V_ℓ) has common variance $\kappa^{(\ell-1)}(0)$ and covariance $\kappa^{(\ell-1)}(d)$. These are also the variances and covariance of $(U_{\ell,d}, V_{\ell,d})$, by its definition in Lemma 1. Both pairs are centered Gaussian, so they have the same distribution. Therefore,

$$\mathbb{E}[\sigma(U_\ell)\sigma(V_\ell)] = \mathbb{E}[\sigma(U_{\ell,d})\sigma(V_{\ell,d})].$$

Adding β^2 to both sides gives

$$\Sigma^{(\ell)}(x,y) = \kappa^{(\ell)}(d_G(x,y)).$$

This completes the induction. In particular,

$$\mathbf{\Sigma}_{xy} = \Sigma(x,y) = \kappa^{(L)}(d_G(x,y)).$$

Lemma 3 therefore gives its eigenvalue on $\mathcal{E}_j$ as

$$\gamma_j = \sum_{\ell=0}^{s} \kappa^{(L)}(\ell) E_\ell^{(n,k)}(j).$$

Replacing ϑ by $\kappa^{(L)}$ in the preceding argument for μ_j, and using the derivative lower bound in Lemma 5, gives

$$\frac{\gamma_j}{|\mathcal{X}|} \geq cn^{-j}.$$

Choose the fixed integer $j > a$. Since $N \leq n^a$ and $d_j \geq cn^j$, the dimension $m = \dim \mathcal{F}$ satisfies

$$m \geq d_j - N \geq \frac{d_j}{2} > 2$$

for sufficiently large n.

Let $e_1, \ldots, e_m$ be a Euclidean orthonormal basis of $\mathcal{F}$. Because $\mathcal{F} \subseteq \mathcal{E}_j$, the coordinates of Z in this basis are independent $\mathcal{N}(0, \gamma_j)$ variables. We can therefore write

$$Z = \sqrt{\gamma_j} \sum_{i=1}^{m} \xi_i e_i, \qquad \xi_1, \ldots, \xi_m \text{ independent } \mathcal{N}(0,1).$$

Orthonormality gives $\sum_{x\in\mathcal{X}} Z(x)^2 = \gamma_j \sum_{i=1}^m \xi_i^2$. Our function norm divides this sum by $|\mathcal{X}|$, so

$$\|Z\|^2 = \frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} Z(x)^2 = \frac{\gamma_j}{|\mathcal{X}|} \sum_{i=1}^{m} \xi_i^2.$$

Write $\chi_m^2 = \sum_{i=1}^m \xi_i^2$, which has the chi-squared distribution with m degrees of freedom. Integrating its density gives $\mathbb{E}[1/\chi_m^2] = 1/(m-2)$. Hence

$$\mathbb{E}_{\theta_0}\left[\frac{1}{\|Z\|^2}\right] = \frac{|\mathcal{X}|}{\gamma_j(m-2)} \leq C. \tag{13}$$

This bound holds for every training set. It also follows that $V \geq \|Z\|^2 > 0$ almost surely.

For each fixed training set, we now prove

$$\mathbb{E}_{\theta_0}\left[\frac{\mathrm{Var}_{\mathcal{X}}(h)}{V}\right] \leq C.$$

To use the bound $V \geq \|Z\|^2$, we first show that Z and $\zeta = g - Z$ are independent. Since $\mathbf{\Sigma}$ acts on $\mathcal{F}$ by multiplication by γ_j, we have

$$\mathrm{Cov}(Z, f_0 - Z) = \Pi_{\mathcal{F}} \mathbf{\Sigma} (I - \Pi_{\mathcal{F}}) = \gamma_j \Pi_{\mathcal{F}} (I - \Pi_{\mathcal{F}}) = 0.$$

The vectors Z and $f_0 - Z$ are jointly Gaussian, so they are independent. Moreover, Z vanishes at the training inputs. Thus the correction subtracted from f_0 to obtain g depends only on $f_0 - Z$. Consequently, $\zeta = g - Z$ also depends only on $f_0 - Z$ and is independent of Z.

We also need an upper bound on the expected variance of ζ. The orthogonal decomposition $g = Z + \zeta$ and Lemma 2(i), with $q = 2$, give

$$\mathbb{E}_{\theta_0} \mathrm{Var}_{\mathcal{X}}(\zeta) \leq \mathbb{E}_{\theta_0} \mathrm{Var}_{\mathcal{X}}(g) \leq \mathbb{E}_{\theta_0} \sum_{r=1}^{s} r \|\Pi_r g\|^2 \leq C.$$

Finally, $h = f_\infty - g$ implies $\mathrm{Var}_{\mathcal{X}}(h) \le 2V + 2\,\mathrm{Var}_{\mathcal{X}}(g)$. Using $V \ge \|Z\|^2$ and $\mathrm{Var}_{\mathcal{X}}(g) = \|Z\|^2 + \mathrm{Var}_{\mathcal{X}}(\zeta)$, we obtain

$$\frac{\mathrm{Var}_{\mathcal{X}}(h)}{V} \le 2 + 2\frac{\mathrm{Var}_{\mathcal{X}}(g)}{V} \le 4 + 2\frac{\mathrm{Var}_{\mathcal{X}}(\zeta)}{\|Z\|^2}.$$

Taking expectations and using independence therefore gives

$$\mathbb{E}_{\theta_0}\left[\frac{\mathrm{Var}_{\mathcal{X}}(h)}{V}\right] \le 4 + 2\,\mathbb{E}_{\theta_0}\,\mathrm{Var}_{\mathcal{X}}(\zeta)\,\mathbb{E}_{\theta_0}\left[\frac{1}{\|Z\|^2}\right] \le C.$$

We can now bound both remaining fractions and complete the proof. Choose fixed integers $j > 2a + 2$ and $R > a + j + 1$, depending only on a.

For the learned component, the bounds proved on $\mathcal{B}$ give

$$\frac{\sum_{r>R} r\|\Pi_r h\|^2}{V} \le Cn^{-R}\frac{\|\mathbf{y}\|^2_{\mathbf{K}^{-1}}}{V} \le Cn^{a+j-R}\frac{\mathrm{Var}_{\mathcal{X}}(h)}{V}.$$

Taking expectations over initialization and using the preceding step, we obtain

$$\mathbb{E}_{\theta_0}\left[\frac{\sum_{r>R} r\|\Pi_r h\|^2}{V}\right] \le Cn^{a+j-R} \le C \qquad \text{on } \mathcal{B}.$$

For the surviving initialization, $Z \in \mathcal{E}_j$ and $R > j$ imply $\Pi_r g = \Pi_r \zeta$ for $r > R$. Thus $\sum_{r>R} r\|\Pi_r g\|^2$ is independent of Z. Using $V \ge \|Z\|^2$, Lemma 2(i), and the bound on $\mathbb{E}_{\theta_0}[1/\|Z\|^2]$, we get

$$\mathbb{E}_{\theta_0}\left[\frac{\sum_{r>R} r\|\Pi_r g\|^2}{V}\right] \le \mathbb{E}_{\theta_0}\left[\sum_{r>R} r\|\Pi_r g\|^2\right]\mathbb{E}_{\theta_0}\left[\frac{1}{\|Z\|^2}\right] \le C.$$

Together with the bound R for the low levels, these estimates give

$$\mathbb{E}_{\theta_0}\left[\frac{\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2}{V}\right] \le C \qquad \text{on } \mathcal{B}.$$

It remains to account for training sets outside $\mathcal{B}$. For every training set, the fraction is at most $s \le n$, since

$$\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2 \le s\sum_{r=1}^{s}\|\Pi_r f_\infty\|^2 = sV.$$

Our sampling bound gives $\mathbb{P}_{\mathcal{T}}(\mathcal{B}^c) \le 4N^2/d_j \le Cn^{2a-j}$. Consequently,

$$\mathbb{E}_{\mathcal{T},\theta_0}\left[\frac{\sum_{r=1}^{s} r\|\Pi_r f_\infty\|^2}{V}\right] \le C + n\,\mathbb{P}_{\mathcal{T}}(\mathcal{B}^c) \le C + Cn^{2a+1-j} \le C.$$

Substituting this into the initial bound on $1 - \rho_\infty$ proves

$$\mathbb{E}_{\mathcal{T},\theta_0}[\rho_\infty] \ge 1 - \frac{C}{n}.$$

Taking n_0 large enough for the preceding estimates gives the claimed threshold, depending only on $a, \alpha, \sigma, L, \beta$. □

B.6 Proof of Theorem 2

The function h is deterministic, while g is a deterministic linear transformation of the centered initial output f_0, so $\mathbb{E}_{\theta_0} g = 0$. For every $r \ge 1$, we therefore have

$$\mathbb{E}_{\theta_0}\|\Pi_r f_\infty\|^2 = \mathbb{E}_{\theta_0}\|\Pi_r(h+g)\|^2 = \|\Pi_r h\|^2 + \mathbb{E}_{\theta_0}\|\Pi_r g\|^2.$$

Since $d_r \ge 1$, Lemma 2 gives

$$\mathbb{E}_{\theta_0}\left[\sum_{r=1}^{s} r^{q-1}\|\Pi_r f_\infty\|^2\right] \le \sum_{r=1}^{s} r^{q-1} d_r\|\Pi_r h\|^2 + \mathbb{E}_{\theta_0}\left[\sum_{r=1}^{s} r^{q-1}\|\Pi_r g\|^2\right] \le C\|\mathbf{y}\|^2_{\mathbf{K}^{-1}} + C.$$

Orthogonality and $r^{q-1} \geq (R+1)^{q-1}$ for $r > R$ now yield

$$\begin{aligned}\mathbb{E}_{\theta_0}\left\|f_\infty - f_\infty^{(\leq R)}\right\|^2 &= \sum_{r=R+1}^{s} \mathbb{E}_{\theta_0}\|\Pi_r f_\infty\|^2 \\ &\leq \frac{1}{(R+1)^{q-1}}\mathbb{E}_{\theta_0}\left[\sum_{r=1}^{s} r^{q-1}\|\Pi_r f_\infty\|^2\right] \\ &\leq \frac{C}{(R+1)^{q-1}}\left(1 + \|\mathbf{y}\|^2_{\mathbf{K}^{-1}}\right).\end{aligned}$$

□

C Filtered Local Search Pseudocode

Algorithm 1 Filtered Local Search (FLS)

Require: Trained network f_∞, feasible set $\mathcal{X}$ equipped with neighborhood structure $\mathcal{N}$, and initial solution $x_0 \in \mathcal{X}$

$x \leftarrow x_0$
Define $\bar{f}_\infty(u) = |\mathcal{N}(u)|^{-1} \sum_{z \in \mathcal{N}(u)} f_\infty(z)$
Filtered phase
loop
  $a^\star \leftarrow \arg\min_{a \in \{x\} \cup \mathcal{N}(x)} \bar{f}_\infty(a)$, breaking ties randomly
  if $\bar{f}_\infty(a^\star) \geq \bar{f}_\infty(x)$ **then**
    break
  end if
  $x \leftarrow a^\star$
end loop
Original-objective phase
loop
  $a^\star \leftarrow \arg\min_{a \in \{x\} \cup \mathcal{N}(x)} f_\infty(a)$, breaking ties randomly
  if $f_\infty(a^\star) \geq f_\infty(x)$ **then**
    return x
  end if
  $x \leftarrow a^\star$
end loop

D Details of the Controlled Experiments

This appendix provides further details on the experimental setup and results of the controlled experiments presented in Section 5.

D.1 Details of the Experimental Setup

D.1.1 Datasets

We generate datasets from quadratic functions, Ising functions, and NK landscapes. For each family, we use five input dimensions, $n \in \{15, 20, 25, 30, 35\}$, and five random target instances per dimension, giving $3 \times 5 \times 5 = 75$ target instances. For each target instance $F : \{0,1\}^n \to \mathbb{R}$, we sample 1,024 distinct inputs $x^{(1)}, \ldots, x^{(1024)}$ uniformly without replacement from $\{0,1\}^n \setminus \{\mathbf{0}\}$, independently of the target parameters. We standardize the corresponding target values using their mean $\overline{F} = 1024^{-1} \sum_{i=1}^{1024} F(x^{(i)})$ and standard deviation $s_F = \left(1024^{-1} \sum_{i=1}^{1024} (F(x^{(i)}) - \overline{F})^2\right)^{1/2}$ to obtain $\widetilde{F}(x^{(i)}) = (F(x^{(i)}) - \overline{F})/s_F$. We then construct five nested training datasets $\mathcal{T}_j = \left((x^{(i)}, \widetilde{F}(x^{(i)}))\right)_{i=1}^{N_j}$, where $(N_1, \ldots, N_5) = (32, 64, 384, 512, 1024)$. Across the 75 target instances, this gives $3 \times 5 \times 5 \times 5 = 375$ training datasets.

The target functions and their sampling procedures are defined as follows.

Quadratic functions. We consider quadratic targets with linear coefficients $a_i \in \mathbb{R}$ and pairwise coefficients $b_{ij} \in \mathbb{R}$, given by

$$F(x) = \sum_{i=1}^{n} a_i x_i + \sum_{1 \le i < j \le n} b_{ij} x_i x_j.$$

We adapt the Pardalos–Rodgers generator described by Wiegele (2007). For each instance, we draw $d \sim \mathrm{Unif}[0.1, 1]$ and A, B uniformly from $\{1, \dots, 100\}$. Conditional on these parameters, we draw each a_i uniformly from $\{-A, \dots, A\}$. Independently, each b_{ij} is set to zero with probability $1 - d$ or drawn uniformly from $\{-B, \dots, B\}$ with probability d.

Ising functions. Ising functions were originally introduced to describe the energy of a system of interacting magnetic spins, each pointing in one of two opposite directions (Ising, 1925), and have since found applications beyond physics, including in combinatorial optimization (Barahona et al., 1988). Random Ising functions are known for rugged energy landscapes with many local minima (Waclaw & Burda, 2008). We consider random Ising functions of the form

$$F(x) = -\sum_{\{i,j\} \in E} J_{ij} s_i s_j - \sum_{i=1}^{n} b_i s_i,$$

where $s_i = 2x_i - 1 \in \{-1, 1\}$ encodes the binary choice x_i, E is a set of interacting coordinate pairs, $J_{ij} \in \mathbb{R}$ are pairwise coefficients, and $b_i \in \mathbb{R}$ are individual preference coefficients. For each instance, we include each pair $\{i, j\}$ with $i < j$ in E independently with probability $4/(n - 1)$. We draw J_{ij} for every included pair and b_i for every coordinate independently from $\mathcal{N}(0, 1)$, independently of E.

NK landscapes. NK landscapes were introduced to study adaptation in biological systems (Kauffman & Weinberger, 1989). Their ruggedness is controlled by the number of interacting variables. An NK function $F : \{0, 1\}^n \to [0, 1]$ is given by

$$F(x) = \frac{1}{n} \sum_{i=1}^{n} \phi_i(x_{S_i}).$$

Here, S_i contains coordinate i and ν other coordinates. Each ϕ_i is a table assigning a value in $[0, 1]$ to each of the $2^{\nu+1}$ binary configurations on S_i.

To generate each instance, we choose the ν other coordinates in each S_i uniformly without replacement from $\{1, \dots, n\} \setminus \{i\}$ and draw all table entries independently from $\mathrm{Unif}[0, 1]$. For each input dimension, we use $\nu = 3, 4, 6, 8, 9$ for the five instances, respectively.

D.1.2 Network and Training

We use a fully connected network in the NTK parameterization (Jacot et al., 2018), given by

$$f_\theta(x) = \frac{W^{(3)}}{\sqrt{m}}\, \sigma\!\left(\frac{W^{(2)}}{\sqrt{m}}\, \sigma\!\left(\frac{W^{(1)} x}{\sqrt{n}} + \beta b^{(1)}\right) + \beta b^{(2)}\right) + \beta b^{(3)},$$

where $m = 1024$, σ is a GELU activation (Hendrycks & Gimpel, 2016), and $\beta = 0.1$. We initialize every unscaled weight and bias independently from $\mathcal{N}(0, 1)$. For each target instance, all five runs on its nested datasets start from the same initialized parameters.

We train one network on each dataset by full-batch gradient descent with learning rate 15, minimizing the mean squared error loss. Training stops when the mean squared training error is at most 10^{-4} or after 10,000 updates.

Among the 375 trained networks, 318 reached the MSE tolerance and 57 reached the 10,000-step limit. The largest final MSE was 1.48×10^{-3}.

D.1.3 Feasible Sets

For each input dimension n, we use the cardinality-constrained feasible set $\mathcal{X}$ in (1) with the exchange neighborhood (2). We choose the largest $k \le \lfloor n/2 \rfloor$ such that $\binom{n}{k} \le 100{,}000$, allowing

Table 1: Feasible sets used for all controlled experiments.

n	k	$\lvert\mathcal{X}\rvert$	D	λ_1
15	7	6,435	56	0.7321
20	7	77,520	91	0.7802
25	5	53,130	100	0.7500
30	4	27,405	104	0.7115
35	4	52,360	124	0.7177

every trained network to be evaluated at every feasible solution. Table 1 lists the resulting values of k, the number of solutions $|\mathcal{X}|$, the neighborhood size $D = k(n-k)$, and the largest nonconstant random-walk eigenvalue λ_1. This eigenvalue is the maximum autocorrelation over nonconstant functions on the graph.

D.1.4 Hardware and Software

The experiments use Python 3.13.5, NumPy 2.3.4, SciPy 1.16.2, and PyTorch 2.8.0. They run on an Apple M4 Max machine with 14 CPU cores, a 32-core GPU, and 36 GB of memory.

D.2 Detailed Results for Autocorrelation and Local-Search Performance

The median Spearman correlation over all target instances is 0.800.[6] Table 2 gives a breakdown by target family and input dimension. Figure 1 plots ρ and Q for each trained network.

Table 2: Median Spearman correlations by target family and input dimension.

Family	$n = 15$	$n = 20$	$n = 25$	$n = 30$	$n = 35$
Quadratic	0.700	0.700	0.872	0.935	0.872
Ising	0.900	0.700	0.600	0.600	0.667
NK	0.900	0.700	0.872	0.872	0.872

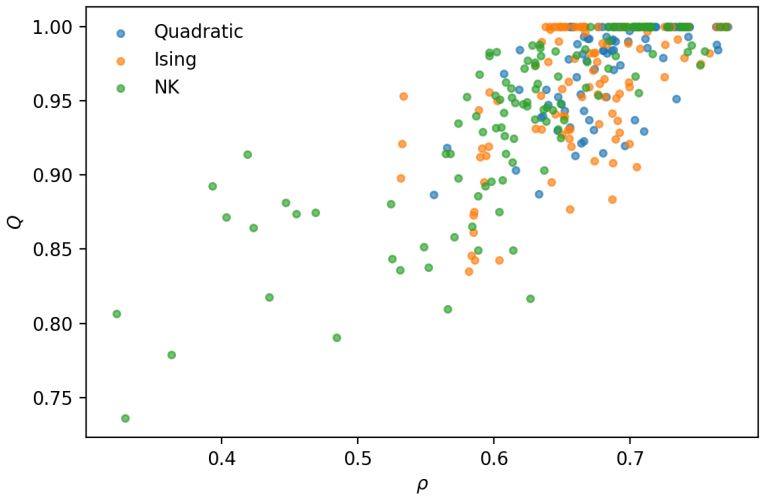

Figure 1: Autocorrelation and local-search performance.

[6] For six target instances, Q is constant across the five networks, so their Spearman correlations are undefined and omitted.

D.3 Detailed Results for High Autocorrelation and Spectral Bias

On average, autocorrelation is 90.1% of the graph maximum, as measured by the mean of ρ/λ_1. It generally decreases as the training-set size grows, consistent with the networks favoring smoother interpolations when fewer observations are imposed.

The first two spectral levels contain 97.0% of the output variance on average. This concentration is weaker for NK targets, consistent with their interactions among more than two variables. Even then, only 0.17% of the variance lies above level 4 on average. These observations support the spectral bias in Theorem 2, suggesting that trained networks have an inductive bias towards low spectral levels, with additional high-level weight reflecting the complexity of the labels being fitted.

We compute the spectral components by applying

$$\Pi_r = \prod_{\substack{0 \le j \le s \\ j \ne r}} \frac{P - \lambda_j I}{\lambda_r - \lambda_j}$$

to the centered network output, using sparse matrix-vector products.

Tables 3–5 give the breakdowns by input dimension n and training-set size N, averaging the five target instances. Table 6 gives the breakdown at every level, averaging over the five instances and five training sizes. A dash indicates an absent eigenspace.

Table 3: Autocorrelation and spectral weights for quadratic functions.

n	N	ρ	ρ/λ_1 (%)	Level 1 (%)	Level 2 (%)	Above 2 (%)
15	32	0.7210	98.48	95.28	4.64	0.08
15	64	0.7170	97.94	93.59	6.30	0.11
15	384	0.6949	94.91	84.02	15.90	0.08
15	512	0.6940	94.80	83.65	16.30	0.06
15	1024	0.6932	94.68	83.27	16.70	0.03
20	32	0.7688	98.53	94.32	5.58	0.10
20	64	0.7537	96.60	86.73	13.10	0.17
20	384	0.6965	89.28	58.03	41.65	0.32
20	512	0.6917	88.65	55.52	44.20	0.28
20	1024	0.6849	87.78	51.98	47.86	0.16
25	32	0.7419	98.92	96.54	3.39	0.06
25	64	0.7367	98.23	94.29	5.65	0.06
25	384	0.6866	91.55	72.63	27.16	0.21
25	512	0.6722	89.62	66.38	33.39	0.22
25	1024	0.6439	85.85	54.05	45.75	0.20
30	32	0.7067	99.33	98.24	1.73	0.02
30	64	0.7047	99.04	97.48	2.49	0.02
30	384	0.6743	94.77	86.25	13.68	0.07
30	512	0.6642	93.35	82.51	17.41	0.08
30	1024	0.6248	87.81	67.89	32.00	0.11
35	32	0.7136	99.42	98.46	1.52	0.02
35	64	0.7123	99.24	97.97	2.02	0.02
35	384	0.6891	96.02	89.30	10.65	0.05
35	512	0.6778	94.44	85.06	14.87	0.07
35	1024	0.6300	87.78	67.16	32.71	0.13

Table 4: Autocorrelation and spectral weights for Ising functions.

n	N	ρ	ρ/λ_1 (%)	Level 1 (%)	Level 2 (%)	Above 2 (%)
15	32	0.6844	93.48	79.67	20.06	0.27
15	64	0.6493	88.68	64.75	34.77	0.48
15	384	0.5784	79.00	34.08	65.62	0.30
15	512	0.5772	78.83	33.44	66.36	0.20
15	1024	0.5758	78.65	32.76	67.17	0.07
20	32	0.7658	98.15	92.84	7.04	0.13
20	64	0.7328	93.92	76.29	23.44	0.27
20	384	0.6831	87.55	51.28	48.31	0.41
20	512	0.6819	87.40	50.63	49.06	0.31
20	1024	0.6812	87.31	50.13	49.71	0.16
25	32	0.7399	98.66	95.68	4.26	0.06
25	64	0.7249	96.65	89.19	10.72	0.10
25	384	0.6755	90.06	67.85	31.90	0.25
25	512	0.6684	89.12	64.75	35.01	0.24
25	1024	0.6664	88.86	63.81	36.05	0.15
30	32	0.7044	99.00	97.39	2.58	0.03
30	64	0.6941	97.55	93.56	6.40	0.04
30	384	0.6448	90.62	75.31	24.57	0.12
30	512	0.6407	90.05	73.82	26.06	0.12
30	1024	0.6401	89.97	73.57	26.34	0.09
35	32	0.7129	99.32	98.20	1.78	0.02
35	64	0.7072	98.54	96.08	3.90	0.03
35	384	0.6566	91.48	77.12	22.78	0.09
35	512	0.6500	90.56	74.65	25.25	0.10
35	1024	0.6419	89.44	71.61	28.30	0.10

Table 5: Autocorrelation and spectral weights for NK landscapes.

n	N	ρ	ρ/λ_1 (%)	Level 1 (%)	Level 2 (%)	Above 2 (%)
15	32	0.6805	92.94	78.00	21.71	0.29
15	64	0.6273	85.68	55.67	43.42	0.91
15	384	0.4756	64.96	19.20	47.41	33.40
15	512	0.4507	61.55	16.45	42.27	41.29
15	1024	0.4021	54.92	15.04	28.73	56.24
20	32	0.7556	96.85	87.73	12.09	0.18
20	64	0.7211	92.42	70.51	29.05	0.44
20	384	0.6202	79.49	29.53	59.22	11.26
20	512	0.6014	77.08	26.51	55.22	18.27
20	1024	0.5641	72.30	23.99	40.72	35.30
25	32	0.7369	98.25	94.36	5.57	0.08
25	64	0.7151	95.35	84.96	14.90	0.14
25	384	0.6281	83.75	48.64	49.62	1.74
25	512	0.6337	84.49	51.81	45.66	2.54
25	1024	0.6153	82.04	48.86	43.26	7.88
30	32	0.7019	98.65	96.46	3.51	0.03
30	64	0.6839	96.12	89.80	10.14	0.06
30	384	0.5944	83.53	56.97	42.51	0.52
30	512	0.5887	82.73	55.19	43.95	0.86
30	1024	0.6004	84.38	60.77	37.07	2.16
35	32	0.7128	99.31	98.17	1.81	0.02
35	64	0.7036	98.03	94.71	5.26	0.03
35	384	0.6358	88.59	69.41	30.38	0.20
35	512	0.6303	87.82	67.43	32.28	0.30
35	1024	0.6122	85.29	61.34	37.61	1.05

Table 6: Mean output variance by spectral level (%).

Family	Level	$n = 15$	$n = 20$	$n = 25$	$n = 30$	$n = 35$
Quadratic	1	87.96	69.32	76.78	86.47	87.59
Quadratic	2	11.97	30.48	23.07	13.46	12.35
Quadratic	3	0.06	0.17	0.14	0.06	0.06
Quadratic	4	0.01	0.03	0.01	2.57×10^{-3}	2.21×10^{-3}
Quadratic	5	9.01×10^{-4}	2.44×10^{-3}	3.84×10^{-4}	–	–
Quadratic	6	1.58×10^{-4}	4.69×10^{-4}	–	–	–
Quadratic	7	1.06×10^{-5}	3.27×10^{-5}	–	–	–
Ising	1	48.94	64.24	76.25	82.73	83.53
Ising	2	50.80	35.51	23.59	17.19	16.40
Ising	3	0.22	0.22	0.15	0.08	0.06
Ising	4	0.04	0.03	0.01	3.16×10^{-3}	2.91×10^{-3}
Ising	5	4.34×10^{-3}	3.28×10^{-3}	4.52×10^{-4}	–	–
Ising	6	5.49×10^{-4}	4.42×10^{-4}	–	–	–
Ising	7	6.05×10^{-5}	3.86×10^{-5}	–	–	–
NK	1	36.87	47.65	65.72	71.84	78.21
NK	2	36.71	39.26	31.80	27.44	21.47
NK	3	23.06	12.47	2.39	0.71	0.31
NK	4	2.68	0.46	0.08	0.02	8.78×10^{-3}
NK	5	0.50	0.13	8.20×10^{-3}	–	–
NK	6	0.14	0.02	–	–	–
NK	7	0.04	6.68×10^{-3}	–	–	–

D.4 Detailed Results for Filtered Local Search

The second phase of FLS uses the same random tie choices as ordinary local search.

FLS increases mean Q from 0.9631 to 0.9785. It improves Q on 214 networks, leaves it unchanged on 149, and reduces it on 12. Mean gains are positive for every target family and input dimension, and generally larger with larger training sets. The larger gains are consistent with filtering becoming more useful as networks learn more of the targets' high-level variation. Table 7 gives the breakdowns of mean Q by target family, input dimension, and training-set size. Each entry averages 25 networks.

Table 7: Local-search performance.

Family	Method	$n = 15$	$n = 20$	$n = 25$	$n = 30$	$n = 35$
Quadratic	Local search	0.9955	0.9726	0.9735	0.9821	0.9841
	FLS	0.9982	0.9807	0.9877	0.9925	0.9894
Ising	Local search	0.9279	0.9513	0.9744	0.9851	0.9815
	FLS	0.9541	0.9684	0.9838	0.9924	0.9972
NK	Local search	0.8994	0.9284	0.9575	0.9559	0.9777
	FLS	0.9405	0.9500	0.9772	0.9755	0.9897
Family	Method	$N = 32$	$N = 64$	$N = 384$	$N = 512$	$N = 1024$
Quadratic	Local search	1.0000	0.9958	0.9807	0.9739	0.9574
	FLS	1.0000	0.9958	0.9898	0.9854	0.9776
Ising	Local search	0.9951	0.9869	0.9478	0.9517	0.9387
	FLS	0.9964	0.9953	0.9689	0.9669	0.9684
NK	Local search	0.9946	0.9820	0.9182	0.9240	0.9002
	FLS	0.9993	0.9897	0.9546	0.9481	0.9412

E Details of the Applications

This appendix gives further details on the application experiments in Section 6.

E.1 Capacitated Facility Location

E.1.1 Details of the Experimental Setup

Released networks. We use the three NN-P networks released by Patel et al. (2022), one for each $n \in \{10, 25, 50\}$, from commit `69683ce` of their Neur2SP repository. Each takes the concatenation of a binary facility-opening vector and a customer-demand vector, both of length n, and predicts the service cost. The architecture has one hidden layer of 64 ReLU units and a linear scalar output.

Training data and procedure. The released data-generation procedure samples 1,000 demand vectors, with independent entries uniformly distributed over $\{5, \dots, 35\}$. For each demand vector, it samples ten facility-opening vectors. Each opening vector is generated by drawing p uniformly from $\{0.1, 0.2, \dots, 0.9\}$ and independently closing each facility with probability p. The label is the service cost obtained by solving the corresponding customer-allocation problem, which charges a penalty for each unserved customer. These solves have a 60-second time limit and relative-gap tolerances of 1% for $n = 10$ and 5% for $n \in \{25, 50\}$, so the labels need not be exact optimal costs. The resulting 10,000 observations are randomly split into 8,000 training and 2,000 validation observations.

Training minimizes mean squared error with L1 and squared L2 penalties on the weights, excluding biases. Validation mean absolute error is evaluated every ten epochs, and the network with the smallest value is saved during 1,000 training epochs. Table 8 gives the settings recorded in the release; the L1 and L2 entries are the penalty coefficients.[7]

Table 8: Network training.

Setting	$n = 10$	$n = 25$	$n = 50$
Hidden width	64	64	64
Optimizer	RMSprop	Adam	Adam
Learning rate	0.00768	0.01522	0.04842
Batch size	32	32	128
Dropout probability	0.07346	0.12911	0.09570
L1 coefficient	0	0.00411	0.06220
L2 coefficient	0	0.01378	0.00984
Selected epoch	590	940	760

Optimization instances. For each network, the opening costs, capacities, and service costs are those generated by the released facility-location procedure. We consider $S \in \{1000, 2000, 5000, 10000\}$ demand scenarios and generate ten instances for each scenario count, giving $3 \times 4 \times 10 = 120$ optimization instances. For each instance, we sample an $S \times n$ array of independent demands uniformly from $\{5, \dots, 35\}$.

We minimize

$$F(x) = \sum_{i=1}^{n} a_i x_i + \frac{1}{S} \sum_{j=1}^{S} \widehat{Q}\left(x, \xi^{(j)}\right), \qquad x \in \{0,1\}^n,$$

where a_i is the opening cost of facility i, $\widehat{Q}$ is the released network, and $\xi^{(j)}$ is the jth demand vector. Every binary opening vector is feasible, since the customer-allocation problem allows unmet demand at a penalty. A move opens or closes one facility, giving n neighbors per solution.

Local search. For each instance, we generate 100 shared starting solutions by setting each opening decision independently to one with probability $1/2$. Ordinary best-improvement local search minimizes F. FLS first minimizes $\bar{F}(x) = n^{-1} \sum_{z \in \mathcal{N}(x)} F(z)$ and then minimizes F, using the stopping and random tie-breaking rules of Algorithm 1.

All scenario and neighborhood averages are computed exactly. For each hidden neuron, the output under scenario s is $\max\{0, a(x) + b_s\}$, where $a(x)$ depends on the opening decisions and b_s depends

[7] These recorded settings differ from those reported by Patel et al. (2022). We use the released checkpoints directly.

only on the scenario. We sort the b_s values once and store their cumulative sums. For a new solution x, binary search identifies the values above $-a(x)$. Their stored sum plus $a(x)$ times their number gives the total ReLU output, avoiding a separate network evaluation for every scenario. Objective values and neighborhood averages are cached during each search.

Gurobi benchmark. We minimize F using the NN-P mixed-integer linear formulation of Patel et al. (2022). We use the released formulation's big-M constant of 10^5 and check, before each solve, that it bounds the magnitude of every ReLU input over the feasible set. We use default solver tolerances, including a relative MIP-gap tolerance of 10^{-4}.

For each instance, we supply the first shared random starting solution as a warm start to Gurobi. Gurobi receives the same optimization time as the complete 100-start FLS run on that instance. This budget excludes model construction and warm-start preparation, but includes presolve and processing the starting solution. We report the better objective value of the supplied starting solution and any feasible solution returned within this budget.

Hardware and software. The experiments use Python 3.13.5, NumPy 2.3.4, PyTorch 2.8.0, and Gurobi 12.0.3 (Gurobi Optimization, LLC, 2026). All methods run separately on an Apple M4 Max machine with access to 14 CPU cores and 36 GB of memory, processing one instance at a time.

E.1.2 Detailed Results

Across the $120 \times 100 = 12{,}000$ paired starts, FLS returns a lower objective than ordinary local search in 8,219 cases, the same objective in 3,609, and a higher objective in 172. Table 9 gives the breakdown by input dimension and scenario count, reporting the mean objective over 1,000 runs per method, from ten instances with 100 shared starts each. The mean running time per start is 0.025 seconds for LS and 0.082 seconds for FLS.

Table 9: Mean objective values.

n	S	LS	FLS
10	1000	7,119.28	**6,988.62**
10	2000	7,120.61	**6,986.13**
10	5000	7,125.62	**6,996.72**
10	10000	7,124.43	**6,992.87**
25	1000	13,206.13	**12,583.28**
25	2000	13,212.08	**12,581.36**
25	5000	13,228.10	**12,590.12**
25	10000	13,211.92	**12,570.51**
50	1000	26,628.83	**26,097.77**
50	2000	26,611.22	**26,097.66**
50	5000	26,650.31	**26,126.72**
50	10000	26,627.25	**26,107.51**

Comparing the best solutions found over 100 starts, FLS improves on ordinary local search in 70 of the 120 instances, ties in 49, and returns a higher objective in one. It also returns a lower objective than the warm-started Gurobi baseline in all 120 instances. Table 10 reports the mean best objective for the three methods, averaging ten instances in each row. The complete 100-start runs take 0.196 seconds for LS and 0.629 seconds for FLS on average. Gurobi receives the same time budget as FLS on each instance. All 120 Gurobi runs reach this time limit with the supplied starting solution as their best solution and without certifying optimality.

Table 10: Mean best objective values.

n	S	LS	FLS	Gurobi
10	1000	**6,982.23**	**6,982.23**	11,804.03
10	2000	**6,978.09**	**6,978.09**	11,788.78
10	5000	**6,988.27**	**6,988.27**	11,794.87
10	10000	**6,987.12**	**6,987.12**	11,789.36
25	1000	12,218.95	**11,980.47**	17,176.71
25	2000	12,250.96	**11,995.15**	17,164.93
25	5000	12,263.34	**12,006.50**	17,209.87
25	10000	12,222.47	**11,993.76**	17,179.48
50	1000	25,498.33	**25,463.16**	37,684.47
50	2000	25,478.52	**25,441.69**	37,675.89
50	5000	25,533.99	**25,481.00**	37,749.95
50	10000	25,497.47	**25,468.74**	37,693.03

E.2 Production Scheduling

E.2.1 Details of the Experimental Setup

Released networks. We use all four networks released by Ghasemi et al. (2023), from commit `9f15c24` of their DeepLearning_FJS repository. The four architectures are GRU-MLP, LSTM-MLP, GRU-MHA, and LSTM-MHA. They differ in their recurrent encoders, which use gated recurrent units (GRUs) or long short-term memory units (LSTMs), and in how they combine the encodings, using a multilayer perceptron (MLP) or multi-head attention (MHA) followed by an MLP. Their widths and parameter counts are given in Table 11.

Table 11: Released scheduling networks.

Setting	GRU-MLP	GRU-MHA	LSTM-MLP	LSTM-MHA
Job encoder width	50	99	10	99
Machine encoder width	50	91	10	91
Setup encoding width	50	99	40	99
Final MLP hidden widths	$200, 100, 50, 20$	$100, 20$	$100, 50, 25, 10$	$100, 50$
Heads per attention layer	N/A	5	N/A	4
Parameters	302,141	351,844	51,896	397,076

Training data and procedure. The authors generate 10,000 scheduling observations using a production simulator with eight machines and twelve job slots. The total number of operations is sampled uniformly from 96 to 120. Processing times follow a gamma distribution with shape 26.98 and scale 2.448; setup times are five times a draw from this distribution. Machine availability times are generated by multiplying another draw by a uniformly sampled integer from zero to five. Each observation specifies the job sequences, machine availability times, setup pairs, and a schedule. Its label is the average makespan over 100 simulation replications.

The observations are randomly split into 8,000 training, 1,000 validation, and 1,000 test observations using seed zero. The released training scripts minimize mean squared error using Adam with learning rate 5×10^{-4} and batch size 128. Training lasts at most 200 epochs, stops after 20 consecutive epochs with validation error above the best previous value, and retains the checkpoint with the smallest validation error.

Optimization instances. Each production instance c specifies the set $\mathcal{O}_m(c)$ of operations assigned to each machine $m = 1, \ldots, 8$, the prescribed operation sequence $J_j(c)$ of each job $j = 1, \ldots, 12$, the machine availability times $b_m(c)$, and a multiset $\mathcal{S}(c)$ of ordered operation pairs requiring extra machine preparation when processed consecutively. We choose $x = (x_1, \ldots, x_8)$, where x_m is a permutation of $\mathcal{O}_m(c)$. Let $\mathcal{D}(x, c)$ be the directed graph on the operations, with an arc from each operation to the next one in its job sequence and to the next one in its machine order.

The optimization problem is

$$\min_{x \in \mathcal{X}(c)} \widehat{f}(x, c), \qquad \mathcal{X}(c) = \left\{ x \in \prod_{m=1}^{8} \mathrm{Perm}(\mathcal{O}_m(c)) \;\middle|\; \mathcal{D}(x, c) \text{ is acyclic} \right\},$$

where $\widehat{f}(x, c)$ is the network's predicted makespan and $\mathrm{Perm}(\mathcal{O}_m(c))$ denotes all permutations of $\mathcal{O}_m(c)$. Thus the machine orders are chosen, while the job orders and machine assignments are fixed.

We use the first ten observations in the released test split for every network, giving $4 \times 10 = 40$ optimization instances. They contain between 98 and 119 operations and either 11 or 12 nonempty jobs. The job sequences, machine availability times, setup pairs, and assignment of operations to machines remain fixed within each instance and form its context c. A neighbor exchanges any two positions on the same machine and retains the resulting order if it belongs to $\mathcal{X}(c)$.

Local search. For each production instance, we generate 100 shared starting solutions. We first draw a uniformly random permutation of all operations, reorder each job's operations within their occupied positions to respect its prescribed order, and then read off the operations assigned to each machine. This produces a feasible starting solution, shared by both methods and all four networks.[8] Ordinary best-improvement local search minimizes $\widehat{f}(\cdot, c)$. FLS first minimizes $\bar{f}(x, c) = |\mathcal{N}(x)|^{-1} \sum_{z \in \mathcal{N}(x)} \widehat{f}(z, c)$ and then minimizes $\widehat{f}(\cdot, c)$, using Algorithm 1. Each phase stops when no neighbor has a strictly smaller objective, and exact ties between improving moves are broken randomly. Every feasible neighbor is evaluated. We cache predictions and sequence encodings to avoid repeating network computations.

Hardware and software. The experiments use Python 3.13.5, NumPy 2.3.4, and PyTorch 2.8.0 on an Apple M4 Max machine with 14 CPU cores and 36 GB of memory. Independent paired runs are distributed across 14 workers, with one computational thread per worker.

E.2.2 Detailed Results

Across the $40 \times 100 = 4{,}000$ paired starts, FLS returns a lower objective than ordinary local search in 3,137 cases, the same objective in 146, and a higher objective in 717. Table 12 gives the breakdown by network, reporting the mean predicted makespan over 1,000 runs per method, from ten production instances with 100 shared starts each. The mean running time per start is 0.92 seconds for LS and 40.64 seconds for FLS, including the final neighborhood checks.

Comparing the best solutions found over 100 starts, FLS improves on ordinary local search in 35 of the 40 instances, ties in none, and returns a higher objective in 5. Table 13 reports the mean best objective for each method, averaging the ten production instances for each network.

Table 12: Mean objective values.

Network	LS	FLS
GRU-MLP	2,641.77	**2,634.59**
GRU-MHA	2,656.29	**2,651.42**
LSTM-MLP	2,576.06	**2,570.83**
LSTM-MHA	2,657.84	**2,656.25**

Table 13: Mean best objective values.

Network	LS	FLS
GRU-MLP	2,586.22	**2,583.00**
GRU-MHA	2,635.82	**2,632.49**
LSTM-MLP	2,515.28	**2,514.37**
LSTM-MHA	2,644.15	**2,643.37**

[8] Unlike the facility-location networks, which admit mixed-integer linear formulations, the networks here have recurrent and attention-based architectures that lead to complex nonconvex mixed-integer nonlinear formulations. Our tests revealed numerical failures in Gurobi 12.0.3, preventing a reliable comparison. We therefore compare FLS only with ordinary local search in this experiment.