Deep Layer-wise Networks Have Closed-Form Weights

Chieh Wu*
Northeastern University

Aria Masoomi*
Northeastern University

Arthur Gretton
University College London

Jennifer Dy
Northeastern University

Abstract

There is currently a debate within the neuroscience community over the likelihood of the brain performing backpropagation (BP). To better mimic the brain, training a network *one layer at a time* with only a "single forward pass" has been proposed as an alternative to bypass BP; we refer to these networks as "layer-wise" networks. We continue the work on layer-wise networks by answering two outstanding questions. First, *do they have a closed-form solution?* Second, *how do we know when to stop adding more layers?* This work proves that the *Kernel Mean Embedding* is the closed-form weight that achieves the network global optimum while driving these networks to converge towards a highly desirable kernel for classification; we call it the *Neural Indicator Kernel.*

1 INTRODUCTION

Due to the brain-inspired architecture of Multi-layered Perceptrons (MLPs), the relationship between MLPs and our brains has been a topic of significant interest (Zador, 2019; Walker, 2019). This line of research triggered a debate (Whittington and Bogacz, 2019) around the neural plausibility of backpropagation (BP). While some contend that brains cannot simulate BP (Crick, 1989; Grossberg, 1987), others have proposed counterclaims with a new generation of models (Hinton, 2007; Lillicrap et al., 2007; Liao et al., 2016; Bengio et al., 2017; Guerguiev et al., 2017; Sacramento et al., 2018; Whittington and Bogacz, 2017). This debate has inspired the search for alternative optimization strategies beyond BP. To better mimic the brain, learning the network *one layer at a time* over a single forward pass (we call it layer-wise network) has been proposed as a more likely candidate to match existing understandings in neuroscience (Ma et al., 2019; Pogodin and Latham, 2020; van den Oord et al., 2018). Our theoretical work contributes to this debate by answering two open questions regarding layer-wise networks.

1. *Do they have a closed-form solution?*
2. *How do we know when to stop adding more layers?*

Question 1 asks if easily computable and *closed-form weights* can theoretically yield networks equally powerful as traditional MLPs, bypassing both BP and Stochastic Gradient Descent (SGD). This question is answered by characterizing the expressiveness of layer-wise networks using only *"trivially learned weights"*. Currently, the *Universal Approximation Theorem* states that a network can approximate any continuous function (Cybenko, 1989; Hornik, 1991; Zhou, 2020). However, it is not obvious that weights of "layer-wise networks" can be computed closed-form requiring only basic operations, a simplicity constraint inspired by biology. As our contribution, we prove that layer-wise networks can classify *any pattern* with trivially obtainable closed-form weights using only *addition.* Surprisingly, these weights turn out to be the *Kernel Mean Embedding*.

Identifying the network depth has been an open question for traditional networks. For layer-wise networks, this question reduces down to "*when should we stop adding layers?*. We posit that additional layers become unnecessary if layer-wise networks exhibit a *limiting behavior* where adding more layers ceases to meaningfully change the network. We proved that this is theoretically possible by using *Kernel Mean Embedding* as weights. In fact, we show that these networks could be modeled as a *mathematical sequence* of functions that converge by *intentional design.* Indeed, not only can these networks converge, they can be induced to converge towards a highly desirable kernel for classification; we call it the *Neural Indicator Kernel* (NIK).

Related Work. When earlier work investigated how MLPs might relate to the brain, Crick (1989) and

Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s). ** Signifies equal contribution as co-first authors.*

later Bengio et al. (2015) both claimed that it was "highly unlikely" that the brain is performing BP. According to their work, while BP requires the errors to flow backward to update the weights, the brain has no backward flow of information. Moreover, for BP to be feasible, each layer requires the precise gradient of a future layer. This implies that the brain would need to obtain the gradients of future layers while having access to them at an earlier layer.

To make MLPs biologically plausible, new research has proposed that weights can be obtained without future gradients by using only local information (Nøkland and Eidnes, 2019; Lindsey and Litwin-Kumar, 2020); this is called *local plasticity.* This idea has been successfully implemented by training the network *one layer at a time* (Ma et al., 2019; Pogodin and Latham, 2020; Belilovsky et al., 2019; Nøkland and Eidnes, 2019; Löwe et al., 2019), paving the groundwork on how layer-wise networks might bypass BP. While layer-wise training is an important advancement, they are still far from becoming a viable model for the brain, with residual theoretical questions left unanswered. Is the computation of gradients *theoretically necessary*, or is there a vastly simpler way to obtain the weights? How do we know the network depth? This theoretical work focuses on answering these questions.

Modeling MLPs with kernels has yielded highly impactful theoretical results. MLPs have been popularly modeled as a Gaussian process (GP) (Neal, 2012; Matthews et al., 2018; Lee et al., 2017), inspiring a significant amount of research (Hayou et al., 2019; Lee et al., 2019; Yang, 2019; Duvenaud et al., 2014). The Neural Tangent Kernel (NTK) (Jacot et al., 2018) has recently been proposed to describe the dynamics of the network during training (Jacot et al., 2018; Arora et al., 2019). Our work differs in that these kernel networks are not trained layer-wise.

Instead, our network performs layer-wise training as a composition of kernels and relates closer to work by (Fahlman and Lebiere, 1990; Ma et al., 2019; Pogodin and Latham, 2020; Belilovsky et al., 2019; Nøkland and Eidnes, 2019; Löwe et al., 2019; Kulkarni and Karande, 2017; Zhuang et al., 2011; Montavon et al., 2011; Mairal et al., 2014; Cho and Saul, 2009). While these implementations also employed composition of kernel networks, our work differs in two key aspects. First, their networks all require some form of optimization during training (commonly with SGD). In contrast, we focused on *identifying a closed-form solution* to entirely bypass optimization. Second, none of their work studied how closed-form weights relates to the network depth. While Duan et al. (2020) was able to relate depth to a complexity bound on composition of kernels under a different setting and objective, identifying a simple and closed-form solution was not their goal. Moreover, we further demonstrated that by using our closed-form solution, the network converges to the *Neural Indicator Kernel* (NIK).

2 MODELING LAYER-WISE NETWORKS

Layer Construction. Let $X \in \mathbb{R}^{n\times d}$ be a dataset of n samples with d features and let $Y \in \mathbb{R}^{n\times\tau}$ be its one-hot encoded labels with τ classes. The l^{th} layer consists of linear weights $W_l \in \mathbb{R}^{m\times q}$ followed by an activation function $\psi : \mathbb{R}^{n\times q} \to \mathbb{R}^{n\times m}$. We interpret each layer as a function ϕ_l parameterized by W_l with an input/output denoted as $R_{l-1} \in \mathbb{R}^{n\times m}$ and $R_l \in \mathbb{R}^{n\times m}$ where $R_l = \phi_l(R_{l-1}) = \psi(R_{l-1}W_l)$. The entire network ϕ is the composition of all layers where ϕ where $\phi = \phi_L \circ ... \circ \phi_1$.

Network Objective. Given x_i, y_i as the i^{th} sample and label of the dataset, the network output is used to minimize an empirical risk ($\mathcal{H}$) with a loss function (ℓ) with a general objective of

$$\min_{\phi} \quad \mathcal{H} \coloneqq \min_{\phi} \quad \frac{1}{n}\sum_{i=1}^{n} \ell(\phi(x_i), y_i). \tag{1}$$

As Eq. (1), we are structurally identical to conventional MLPs where each layer consists of linear weights and an activation function. Yet, we differ by introducing the composition of the first l layers as $\phi_{l^\circ} = \phi_l \circ ... \circ \phi_1$ where $l \leq L$. This notation (ϕ_{l°) connects the data directly to the lth layer output where $R_l = \phi_{l^\circ}(X)$ and leads to the *key novelty of our theoretical contribution.* Namely, we propose to optimize Eq. (1) layer-wise as a sequence of a growing networks by replacing ϕ in Eq. (1) incrementally with a sequence of functions $\{\phi_{l^\circ}\}_{l=1}^{L}$. This results in a sequence of empirical risks $\{\mathcal{H}_l\}_{l=1}^{L}$ which we incrementally solve. We refer to $\{\phi_{l^\circ}\}_{l=1}^{L}$ and $\{\mathcal{H}_l\}_{l=1}^{L}$ as the *Kernel Sequence* and the *$\mathcal{H}$-Sequence*.

Solving Eq. (1) as *$\mathcal{H}$-Sequence* is where we differ from tradition, this approach enables us to easily represent, analyze and optimize "layer-wise networks". In contrast to using BP with SGD, we now can use *"closed-form solutions"* for W_l to construct *Kernel Sequences* that drives the *$\mathcal{H}$-Sequence* to automatically minimize Eq. (1). To visualize the network structure and how they form the sequences, refer to Fig. 1.

Performing Classification. Classification tasks typically use objectives like Mean Squared Error (MSE) or Cross-Entropy (CE) to match the network output $\phi(X)$ to the label Y. While this approach achieves the desirable outcome, it also constrains the space of potential solutions where $\phi(X)$ must match

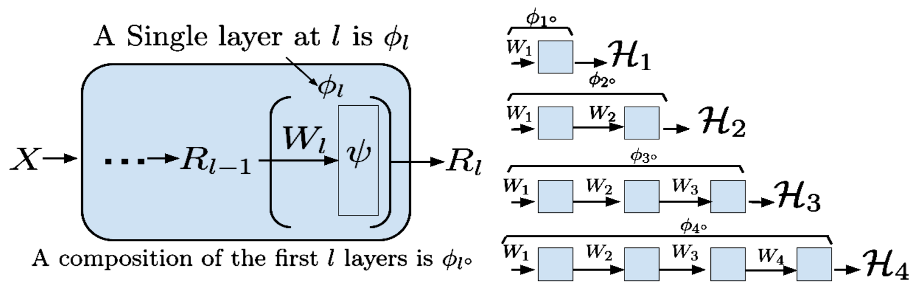

Figure 1: **Left - Distinguishing** ϕ_l **vs** $\phi_{l\circ}$**:** ϕ_l is a single layer while $\phi_{l\circ} = \phi_l \circ ... \circ \phi_{1\circ}$ is a composition of the first l layers. **Right - Visualize the *Kernel* and $\mathcal{H}$-*Sequences*:** Note that the *Kernel Sequence* is a converging sequence of "*functions*" $\{\phi_{l\circ}\}_{l=1} = \{\phi_{1\circ}, \phi_{2\circ}, ...\}$ and $\mathcal{H}$-*Sequence* is a converging sequence of scalar values $\{\mathcal{H}_l\}_{l=1} = \{\mathcal{H}_1, \mathcal{H}_2, ...\}$. To minimize Eq. (1) at $\mathcal{H}_l$, all weights before W_l are already identified and held fixed, only W_l is unknown. As $l \to \infty$, the *Kernel Sequence* converges to the *Neural Indicator Kernel*.

Y. Yet, if ϕ maps X to the labels $\{0, 1\}$ instead of the true label $\{-1, 1\}$, $\phi(X)$ may not match Y, but the solution is the same. Therefore, enforcing $\phi(X) = Y$ ignores an entire space of equivalently optimal classifiers. We posit that by relaxing this constraint and accepting a larger space of potential global optima, it will be easier during optimization to collide with this space. This intuition motivates us to depart from the tradition of label matching and instead seek alternative objectives that focus on solving the underlying prerequisite of classification, i.e., learning a mapping where samples from *similar and different* classes become easily distinguishable.

However, since there are many ways to define *similarity*, how do we choose the best one that leads to classification? We demonstrate that the Hilbert Schmidt Independence Criterion (HSIC (Gretton et al., 2005)) is a highly advantageous objective for this purpose. As we'll later show in corollaries 1 and 2, its maximization indirectly minimizes both Mean Square Error (MSE) and Cross-Entropy (CE) under different notions of "distance", enabling classification. Moreover, this is made possible because maximizing HSIC automatically learns the optimal notion of *similarity* as a kernel function. Using HSIC objective as $\mathcal{H}_l$ for each element of $\{\mathcal{H}_l\}_{l=1}^L$, the layer-wise formulation of Eq. (1) becomes

$$\begin{aligned} \max_{W_l} \quad & \operatorname{Tr}\left(\Gamma \left[\psi(R_{l-1}W_l)\psi^T(R_{l-1}W_l)\right]\right) \\ \text{s.t.} \quad & W_l^T W_l = I, \end{aligned} \tag{2}$$

where we let $\Gamma = HK_YH = HYY^TH$ with centering matrix H defined as $H = I_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^T$. I_n is an identity matrix of size $n \times n$ and $\mathbf{1}_n$ is a column vector of 1s also of length n.

Note that the HSIC objective is more familiarly written as $\operatorname{Tr}(HK_YHK_X)$ where $K_X = \psi(R_{l-1}W_l)\psi^T(R_{l-1}W_l)$. Yet, we purposely present it as Eq. (2) to highlight how the network structure leads to the objective. Namely, the layer input R_{l-1} first multiplies the weight W_l before passing through the activation function ψ. The layer output is then multiplied by itself and Γ to form the HSIC objective.

Lastly, since HSIC can be trivially maximized by setting each element of W as ∞, setting $W^TW = I$ is a constraint commonly used with HSIC while learning a projection (Wu et al., 2018, 2019; Niu et al., 2010).

Key Difference From Traditional MLPs. *We generalize the concept of the activation function to a kernel feature map.* Therefore, instead of using the traditional Sigmoid or ReLU activation functions, we use the feature map of a Gaussian kernel as the activation function ψ. Conveniently, HSIC leverages the kernel trick to spare us the direct computation of the inner product $\psi(R_{l-1}W_l)\psi^T(R_{l-1}W_l)$. Therefore, for each (r_i, r_j) pair we only need to compute

$$\begin{aligned} \mathcal{K}(W_l^T r_i, W_l^T r_j) &= \langle \psi(W_l^T r_i), \psi(W_l^T r_j) \rangle \\ &= \exp\{-\frac{||W_l^T r_i - W_l^T r_j||^2}{2\sigma_l^2}\}. \end{aligned} \tag{3}$$

How Does HSIC Learn the Kernel? Let $\mathcal{S}$ be a set of i, j sample pairs that belong to the same class. Its complement, $\mathcal{S}^c$ contains all sample pairs from different classes. By reinterpreting the kernel $\mathcal{K}(W_l^T r_i, W_l^T r_j)$ from Eq. (3) as a kernel function $\mathcal{K}_{W_l}(r_i, r_j)$ parameterized by W_l, Eq. (2) can be reformulated into the following objective to see how HSIC learns the kernel.

$$\begin{aligned} \max_{W_l} \quad & \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}\mathcal{K}_{W_l}(r_i, r_j) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_{W_l}(r_i, r_j) \\ \text{s.t.} \quad & W_l^T W_l = I. \end{aligned} \tag{4}$$

While Eq. (2) and Eq. (4) are equivalent objectives, Eq. (4) reveals how an optimal similarity measure is learned as a kernel function $\mathcal{K}_{W_l}(r_i, r_j)$ parameterized by W_l. First note that Γ came directly from the label with $\Gamma = HYY^T H$ such that the i, j_{th} element of Γ, denoted as $\Gamma_{i,j}$, is a positive value for samples pairs in $\mathcal{S}$ and negative for $\mathcal{S}^c$. The objective leverages the sign of $\Gamma_{i,j}$ as labels to guide the choice of W_l such that it increases $\mathcal{K}_{W_l}(r_i, r_j)$ when r_i, r_j belongs to the same class in $\mathcal{S}$ while decreasing $\mathcal{K}_{W_l}(r_i, r_j)$ otherwise. Therefore, by finding a W_l matrix that best parameterizes $\mathcal{K}_{W_l}$, HSIC identifies the optimal kernel function $\mathcal{K}_{W_l}(r_i, r_j)$ that separates samples into similar and dissimilar partitions to enable classification.

3 ANALYZING LAYER-WISE NETWORKS

Instead of maximizing Eq. (2) with traditional strategies like SGD, can we completely bypass the optimization step? Our analysis proves that this is possible. In fact, the weights W_l simply need to be set to the *Kernel Mean Embedding* (Muandet et al., 2016) at each layer. The stacking of layers with these known weights automatically drives *$\mathcal{H}$-Sequence* towards its theoretical global optimum. Specifically, if we let r_ι^j be the ι^{th} input sample in class j for layer l with ζ as a normalizer that can be ignored in practice, then the closed-form solution is

$$W_s = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & \dots & \sum_\iota r_\iota^{(\tau)} \end{bmatrix}. \quad (5)$$

We prove that as long as no two identical samples have conflicting labels, setting $W_l = W_s$ at each layer can generate a monotonically increasing *$\mathcal{H}$-Sequence* towards the global optimal, $\mathcal{H}^*$. Comparing to BP, this finding suggests that instead of using gradients or needing to propagate error backward, a deep classifier can be globally optimized with "only a single forward pass" by *simple addition.* We formally prove this discovery in App. B as the following theorem.

Theorem 1. *From any initial risk $\mathcal{H}_0$, there exists a set of bandwidths σ_l and a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^L$ parameterized by $W_l = W_s$ in Eq. (5) such that:*

I. $\mathcal{H}_L$ can approach arbitrarily close to $\mathcal{H}^*$ such that for any $L > 1$ and $\delta > 0$ we can achieve

$$\mathcal{H}^* - \mathcal{H}_L \leq \delta, \quad (6)$$

II. as $L \to \infty$, the $\mathcal{H}$-Sequence converges to the global optimum where

$$\lim_{L\to\infty} \mathcal{H}_L = \mathcal{H}^*, \quad (7)$$

III. the convergence is strictly monotonic where

$$\mathcal{H}_l > \mathcal{H}_{l-1} \quad \forall l \geq 1. \quad (8)$$

To summarize the proof, we identified a lower bound for HSIC that is guaranteed to increase at each layer by adjusting σ_l while using W_s. Since HSIC has a known global theoretical upper bound ($\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}$), we show that the lower bound can approach arbitrarily close to the upper bound by adding more layers.

Intuitively, the network incrementally discovers a better feature map to improve the *Kernel Mean Embedding*. Imagine classifying a set of dogs/cats, since *Kernel Mean Embedding* defines the *average* dog and cat as two points in a high dimensional space. As $L \to \infty$, *$\mathcal{H}$-Sequence* pushes these two points maximally apart from each other with representations that enable easy distinction. The continuity of the feature map ensures that dogs are closer to the average dog than the average cat, enabling classification.

Using the *Kernel Mean Embedding* has a secondary implication; the layer-wise network is performing *Kernel Density Estimation* (Davis et al., 2011). That is, W_s empowers these networks to predict the likelihood of an outcome without knowing the underlying distribution, relying solely on observations. For classification, multiplying the output of ϕ_{l-1° by its corresponding W_s is equivalent to approximating the probability of a sample being in each class. Moreover, by summarizing the data into the W_s, all past examples can be discarded while allowing new examples to update the network by adjusting only the embedding itself, enabling layer-wise networks to self-adapt given new information.

Theoretically, viewing a network as an *$\mathcal{H}$-Sequence* is the key contribution that enables these interpretations. Its usage raises new questions with tantalizing possibilities in the debate over the brain's likelihood of performing BP. Is the brain more likely computing derivatives with BP, or is it simply repeating *addition*? Can the brain use mechanisms similar to *Kernel Density Estimation* to approximate the likelihood of an event? This work only presents the theoretical feasibility and does not claim to have the answer.

What Does *Kernel Sequence* Converge To? As *$\mathcal{H}$-Sequence* converges toward $\mathcal{H}^*$, how does *Kernel Sequence* ultimately lead to classification? This section analyzes the limit behavior of the *Kernel Sequence* and explains how samples are geometrically transformed to induce classification as we add layers to the network.

To formalize this relationship, let us define the point before and after the activation function as *preactivation* $R_{l-1}W_l$ and *activation* $\psi(R_{l-1}W_l)$ (terminology utilized in NTK (Jacot et al., 2018)). This distinction is also crucial for us because as *Kernel Sequence* converges, each stage predictably converges to distinct geometric orientations that enable classification. Keeping this in mind, let us summarize the layer-wise output at *preactivation* using the within S_w^l and between S_b^l class scatter matrices as

$$S_w^l = \sum_{i,j \in \mathcal{S}} W_l^T (r_i - r_j)(r_i - r_j)^T W_l \tag{9}$$

$$S_b^l = \sum_{i,j \in \mathcal{S}^c} W_l^T (r_i - r_j)(r_i - r_j)^T W_l. \tag{10}$$

These matrices are historically important (Fisher, 1936; McLachlan, 2004) because their trace ratio $\mathcal{T} = \text{Tr}(S_w)/\text{Tr}(S_b)$ measures class separability, i.e., a small $\mathcal{T}$ signifies a clear separation of samples into distinguishable clusters, a crucial criterion for a classifier. Our analysis shows that by maximizing HSIC to achieve $\mathcal{H}_l \to \mathcal{H}^*$, it leads to $\mathcal{T} \to 0$ as a byproduct, implying a converging behavior where samples from the same class are incrementally clustering into a single point while pushing samples from other classes away. This orientation at convergence renders classification at *preactivation* trivial.

While sample pairs at *preactivation* are partitioned into clusters via the "Euclidean distance", they are similarly partitioned via the "angular distance" θ at *activation.* Indeed, our Thm. 2 indicates that as $\mathcal{H}_l \to \mathcal{H}^*$, sample pairs at *activation* within $\mathcal{S}$ achieves perfect alignment ($\theta = 0$) while pairs in $\mathcal{S}^c$ become orthogonal to each other, separated by a maximum angle of $\theta = \pi/2$. This implies that as the network layer increases, the inner product between sample pairs from $\mathcal{S}$ and $\mathcal{S}^c$ converges to 1 and 0, respectively. This inner product directly defines a kernel at the point of convergence which we call the *Neural Indicator Kernel.* Classification also becomes trivial once a network converges to NIK.

If the *Kernel Sequence* converges at layer L such that $\phi_{L^\circ} \approx \phi_{L+1^\circ} \approx \phi_{L+2^\circ}...$, additional layers stop changing the network as a function, rendering them "*unnecessary*". This finding yields a promising approach to identify network depth by stop adding layers once the objective is sufficiently close to the global optimum. Therefore, showing that *Kernel Sequence* converges in *preactivation* and *activation* is the key to classification and network depth. We highlight this visually observable convergent behavior in Fig. 3, and formally state our results as the following theorem with its detailed proof in App. C.

Theorem 2. *As $l \to \infty$ and $\mathcal{H}_l \to \mathcal{H}^*$,*

I. the scatter trace ratio $\mathcal{T}$ approaches 0 where

$$\lim_{l \to \infty} \frac{\text{Tr}(S_w^l)}{\text{Tr}(S_b^l)} = 0 \tag{11}$$

II. the Kernel Sequence converges to the following kernel (The Neural Indicator Kernel):

$$\lim_{l \to \infty} \mathcal{K}(x_i, x_j)^l = \mathcal{K}^*(x_i, x_j) = \begin{cases} 0 & \forall i, j \in \mathcal{S}^c \\ 1 & \forall i, j \in \mathcal{S} \end{cases}. \tag{12}$$

As corollaries to Thm. 2, the resulting partition of samples under Euclidean and angular distance implicitly satisfies different classification objectives. At *preactivation*, dataset of τ classes will be mapped into τ distinct points. While these τ points may not match the original labels, this difference is inconsequential for classification. In contrast, converged samples at *activation* will reside along τ orthogonal axes on a unit sphere. Realigning these results to the standard bases simulates the softmax output. Therefore, as $\mathcal{H}_l \to \mathcal{H}^*$, maximizing HSIC leads to a minimization of MSE and CE without matching the actual labels themselves: instead, *it matches the underlying geometry of how classes are distinguished.* We state these findings as two corollaries below with their proofs in App. E.

Corollary 1. *$\mathcal{H}_l \to \mathcal{H}^*$ leads to a minimization of MSE in preactivation via a translation of labels.*

Corollary 2. *$\mathcal{H}_l \to \mathcal{H}^*$ leads to a minimization of CE in activation via a change of bases.*

Optimal W^* at Each Layer. The analysis of *$\mathcal{H}$-Sequence* yields a simple W_s that provided us with an interesting neuroscience interpretation. However, *$\mathcal{H}$-Sequence* can also perform analysis from an optimization perspective. We performed this analysis and show in App. D that W_s is surprisingly not the optimal solution at each layer, or more accurately, we found that $\frac{\partial}{\partial W_l} \mathcal{H}_{l \neq L}(W_s) \neq 0$. This result suggests that satisfying the layer-wise optimality is not a prerequisite for *$\mathcal{H}$-Sequence* to eventually converge to the global optimum. Moreover, it also indicates the existence of a potentially superior W_l^* that may even outperform W_s. Using a similar derivation proposed by Wu et al. (2018, 2019), we set the derivative of Eq. (2) to 0 and found that the optimal W_l^* at each layer is the most dominant eigenvectors of

$$\mathcal{Q}_{l^i} = R_{l-1}^T (\hat{\Gamma} - \text{Diag}(\hat{\Gamma} 1_n)) R_{l-1}, \tag{13}$$

where $\hat{\Gamma}$ is a function of W_{l^i} computed with $\hat{\Gamma} = \Gamma \odot K_{R_{l-1} W_{l^i}}$ and the symbol $\odot$ indicates the element-wise product. Unlike W_s, this solution guarantees at layer l to achieve $\frac{\partial}{\partial W_l} \mathcal{H}_l(W_l^*) = 0$. While W_s is a closed-form solution inspired by the brain, *$\mathcal{H}$-Sequence* also gives us W^* as an optimal solution, highlighting the impressive versatility of *$\mathcal{H}$-Sequence* as a tool to analyze layer-wise networks.

4 GENERALIZATION

Besides being an optimum solution, W_l^* exhibits many advantages over W_s. For example, while W_s experimentally performs well, W^* converges with fewer layers and superior generalization. This raises a well-known question on generalization. It is known that overparameterized MLPs can generalize even without any explicit regularizer (Zhang et al., 2017). This observation contradicts classical learning theory and has been a longstanding puzzle (Cao and Gu, 2019; Brutzkus et al., 2017; Allen-Zhu et al., 2019). Therefore, by being overparameterized with an infinitely wide network, NIK's ability under HSIC to generalize raises similar questions. In both cases, W_s and W^*, the HSIC objective employs an infinitely wide network that should result in overfitting. We ask theoretically, under our framework, what makes HSIC and W^* special?

Recently, Poggio et al. (2020) have proposed that traditional MLPs generalize because gradient methods implicitly regularize the normalized weights given an exponential objective (like our HSIC). We discovered a similar impact the process of finding W^* has on HSIC, i.e., HSIC can be reformulated to isolate out n functions $[D_1(W_l), ..., D_n(W_l)]$ that act as a penalty term during optimization. Let $\mathcal{S}_i$ be the set of samples that belongs to the i_{th} class and let $\mathcal{S}_i^c$ be its complement, then each function $D_i(W_l)$ is defined as

$$\begin{aligned} D_i(W_l) = & \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}_i} \Gamma_{i,j} \mathcal{K}_{W_l}(r_i, r_j) - \\ & \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}_i^c} |\Gamma_{i,j}| \mathcal{K}_{W_l}(r_i, r_j). \end{aligned} \tag{14}$$

Notice that $D_i(W_l)$ is simply Eq. (4) for a single sample scaled by $\frac{1}{\sigma^2}$. Therefore, improving W_l also leads to an increase and decrease of $\mathcal{K}_{W_l}(r_i, r_j)$ associated with $\mathcal{S}_i$ and $\mathcal{S}_i^c$ in Eq. (14), thereby increasing the size of the penalty term $D_i(W_l)$. To appreciate how $D_i(W_l)$ penalizes $\mathcal{H}$, we propose an equivalent formulation in the theorem below with its derivation in App M.

Theorem 3. *Eq. (4) is equivalent to*

$$\begin{aligned} \max_{W_l} \sum_{i,j} & \frac{\Gamma_{i,j}}{\sigma^2} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i^T W_l W_l^T r_j) \\ & - \sum_i D_i(W_l) ||W_l^T r_i||_2. \end{aligned} \tag{15}$$

Based on Thm. 3, $D_i(W_l)$ adds a negative variable cost to the sample norm, $||W_l^T r_i||_2$, prescribing an implicit regularizer on HSIC. Therefore, as W_l improve HSIC, it also imposes an incrementally heavier penalty on Eq. (15), severely constraining W_l.

5 EXPERIMENTS

Experiment Settings. Our analysis from Thm. 1 shows that the Gaussian kernel's σ_l can be incrementally decreased to guarantee a monotonic increase. Our experiments learn the σ_l by incrementally decreasing its value until an improved HSIC is achieved. For results that involves W^*, the Gaussian kernel's σ_l is set to the value that maximizes HSIC, see App. G.

The activation function is approximated with RFF of width 300 (Rahimi and Recht, 2008). The convergence threshold for *$\mathcal{H}$-Sequence* is set at $\mathcal{H}_l > 0.99$. The network structures discovered by W^* for every dataset are recorded and provided in App. H. The MLPs that use MSE and CE have weights initialized via the method used in He et al. (2015). All datasets are centered to 0 and scaled to a standard deviation of 1. All sources are written in Python using Numpy, Sklearn and Pytorch (Jones et al., 2001–; Buitinck et al., 2013; Paszke et al., 2017), and ran on an Intel Xeon(R) CPU E5-2630 v3 @ 2.40GHz x 16 with 16 total cores.

Datasets. The experiments are divided into two sections. Since our contributions are completely theoretical in nature, the majority of the experiments will focus on supporting our thesis by verifying the various components of the theoretical claims. We use three synthetic datasets (Random, Adversarial, and Spiral) and five popular UCI benchmark datasets: wine, cancer, car, face, and divorce (Dheeru and Karra Taniskidou, 2017). They are included along with the source code in the supplementary, and their download link and statistics are in App. F. All theoretical claims are experimentally reproducible with its source code and datasets publicly available in the Supplementary and at `https://github.com/endsley`.

Competing Kernel Methods. We next compare W^* and W_s to several popular kernel networks that have publicly available code: NTK (Arora et al., 2020; Yu, 2019), GP (maka89, 2017; Wilson et al., 2016), Arc-cos (gionuno, 2017; Cho and Saul, 2009). For this comparison, we additionally included CIFAR10 (Krizhevsky, 2009). The images are preprocessed with a convolutional layer that outputs vectorized samples of $x_i \in \mathbb{R}^{10}$. The preprocessing code, the network output, and the network weights are included in the supplementary.

Evaluation Metrics. We report the HSIC objective as $\mathcal{H}$ at convergence along with the training/test accuracy for each dataset. Here, $\mathcal{H}$ is normalized to the range between 0 to 1 using the method proposed by Cortes et al. (2012). To corroborate Corollaries 1 and 2, we also record MSE and CE. To evaluate the sample geometry predicted by Eq. (11), we recorded the

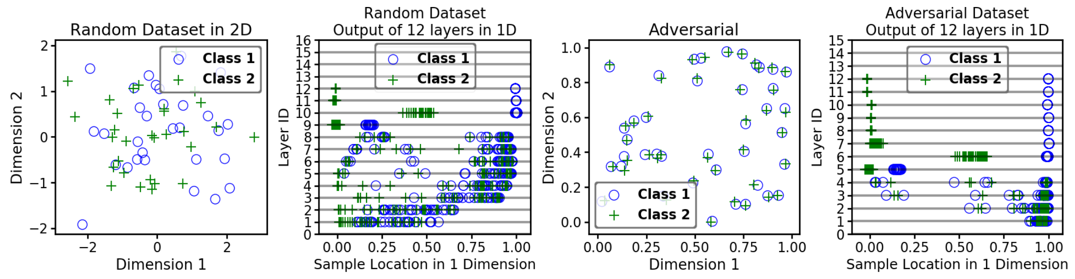

Figure 2: Thm. 1 is simulated on two highly complex datasets, Random and Adversarial. The 2D representation is shown, and next to it, the 1D output of each layer is displayed over each line. As we add more layers, we can see the samples from the two classes gradually separate from each other. Both datasets achieved the global optimum $\mathcal{H}^*$ at the 12_{th} layers. For additional results, see App. I

.

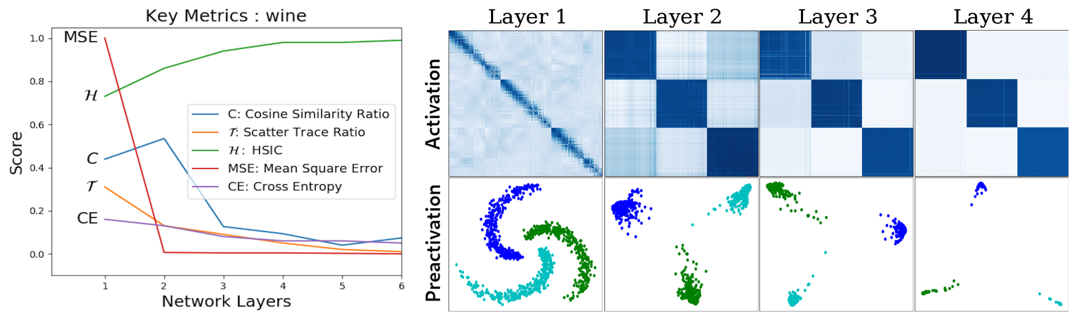

Figure 3: **Left - Key evaluation metrics at each layer:** Showing key statistics computed from the output of each layer. Notice $\mathcal{H}$ monotonically increases while $\mathcal{T}$ and C decrease. **Right - A visual confirmation of Thm. 2:** The top row plots out the kernel matrix from the output of the first 4 layers. Layer 1 shows the original kernel, while layer 4 represents the kernel when it has nearly converged to NIK. The perfect block diagonal structure at layer 4 confirms that the *Kernel Sequence* is indeed converging toward NIK. Below the kernels, we also plot out the convergent pattern in *preactivation*. Samples from the same class converge towards a single point while being pushed away from samples from different classes.

scatter trace ratio $\mathcal{T}$ to measure the compactness of samples within and between classes. The angular distance between samples in $\mathcal{S}$ and $\mathcal{S}^c$ as predicted by Eq. (12) is evaluated with the Cosine Similarity Ratio (C). The equations for normalized $\mathcal{H}$ and C are $\mathcal{H} = \frac{\mathcal{H}(\phi(X),Y)}{\sqrt{\mathcal{H}(\phi(X),\phi(X))\mathcal{H}(Y,Y)}}$ and $C = \frac{\sum_{i,j \in \mathcal{S}^c} \langle \phi(x_i), \phi(x_j) \rangle}{\sum_{i,j \in \mathcal{S}} \langle \phi(x_i), \phi(x_j) \rangle}$.

Confirming Theorem 1. Since Thm. 1 guarantees an optimal convergence for *any* dataset given W_s, we designed an Adversarial dataset of high complexity, i.e., the sample pairs in $\mathcal{S}^c$ are intentionally placed significantly closer than samples pairs in $\mathcal{S}$. We next designed a Random dataset with completely random labels. We then simulated Thm. 1 in Python and plotted the sample behavior in Fig. 2. The original 2-dimensional data is shown next to its 1-dimensional results: each line represents the 1D output at that layer. As predicted by the theorem, the $\mathcal{H}$-*Sequence* for both datasets converges to the global optimum at the 12th layer and perfectly separated the samples based on labels. *This experiment simultaneously confirms Thm. 1 and the idea that simple and repetitive patterns as weights can incrementally improve a network to classify any pattern during training. It also supports the argument that gradients might not be theoretically necessary to optimize deep networks.*

Does $\mathcal{H}$ Improve Monotonically? Thm. 1 also claims that $\mathcal{H}$-*Sequence* should be monotonically increasing. Fig. 3 (Left) plots out all key metrics during training *at each layer*. Here, the $\mathcal{H}$-*Sequence* is clearly monotonic and converging towards a $\mathcal{H}^* \approx 1$. Moreover, the trends for $\mathcal{T}$ and C indicate an incremental clustering of samples into separate partitions. Corresponding to low $\mathcal{T}$ and C values, the low MSE and CE errors at convergence further reinforces the claims of Corollaries 1 and 2. These patterns are highly repeatable, as shown across all datasets included in App. J.

Can Layer-wise Networks with W^*/W_s Compete Against MLPs with BP? We conduct 10-fold cross-validation spanning 8 datasets and report their mean and the standard deviation for all key metrics in Table 1. Once our model is trained and has learned its

	obj	Train Acc ↑	Test Acc ↑	Time(s) ↓	$\mathcal{H}$ ↑	MSE ↓	CE ↓	C ↓	$\mathcal{T}$ ↓
random	W^*	**1.00 ± 0.00**	0.38 ± 0.21	**0.40 ± 0.37**	**1.00 ± 0.01**	**0.00 ± 0.01**	0.05 ± 0.00	**0.00 ± 0.06**	0.02 ± 0.0
	W_s	0.99 ± 0.01	0.45 ± 0.20	0.52 ± 0.05	0.98 ± 0.01	2.37 ± 1.23	0.06 ± 0.13	0.05 ± 0.02	0.13 ± 0.01
	CE	**1.00 ± 0.00**	0.48 ± 0.17	25.07 ± 5.55	**1.00 ± 0.00**	10.61 ± 11.52	**0.0 ± 0.0**	**0.0 ± 0.0**	**0.0 ± 0.0**
	MSE	0.98 ± 0.04	**0.63 ± 0.21**	23.58 ± 8.38	0.93 ± 0.12	0.02 ± 0.04	0.74 ± 0.03	0.04 ± 0.04	0.08 ± 0.1
Advers	W^*	**1.00 ± 0.00**	0.38 ± 0.10	**0.52 ± 0.51**	**1.00 ± 0.00**	**0.00 ± 0.00**	**0.04 ± 0.00**	**0.01 ± 0.08**	**0.01 ± 0.0**
	W_s	0.99 ± 0.04	**0.42 ± 0.18**	2.82 ± 0.81	0.99 ± 0.19	15.02 ± 11.97	0.32 ± 0.15	0.30 ± 0.18	0.34 ± 0.19
	CE	0.59 ± 0.04	0.29 ± 0.15	69.54 ± 24.14	0.10 ± 0.07	0.65 ± 0.16	0.63 ± 0.04	0.98 ± 0.03	0.92 ± 0.0
	MSE	0.56 ± 0.02	0.32 ± 0.20	113.75 ± 21.71	0.02 ± 0.01	0.24 ± 0.01	0.70 ± 0.00	0.99 ± 0.02	0.95 ± 0.0
spiral	W^*	**1.00 ± 0.00**	**1.00 ± 0.00**	**0.87 ± 0.08**	0.98 ± 0.01	0.01 ± 0.00	0.02 ± 0.01	0.04 ± 0.03	0.02 ± 0.0
	W_s	**1.00 ± 0.00**	**1.00 ± 0.00**	13.54 ± 5.66	0.88 ± 0.03	38.60 ± 25.24	0.06 ± 0.02	0.08 ± 0.04	0.08 ± 0
	CE	**1.00 ± 0**	**1.00 ± 0**	11.59 ± 5.52	**1.00 ± 0**	57.08 ± 31.25	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	**1.00 ± 0**	0.99 ± 0.01	456.77 ± 78.83	**1.00 ± 0**	**0 ± 0**	1.11 ± 0.04	0.40 ± 0.01	**0 ± 0**
wine	W^*	0.99 ± 0	**0.99 ± 0.05**	**0.28 ± 0.04**	0.98 ± 0.01	0.01 ± 0	0.07 ± 0.01	0.04 ± 0.03	0.02 ± 0
	W_s	0.98 ± 0.01	0.94 ± 0.04	0.78 ± 0.09	0.93 ± 0.01	2.47 ± 0.26	0.06 ± 0.01	0.05 ± 0.01	0.08 ± 0.01
	CE	**1.00 ± 0.00**	0.94 ± 0.06	3.30 ± 1.24	**1.00 ± 0.00**	40.33 ± 35.5	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	**1.00 ± 0**	0.89 ± 0.17	77.45 ± 45.40	**1.00 ± 0**	**0 ± 0**	1.15 ± 0.07	0.49 ± 0.02	**0 ± 0**
cancer	W^*	0.99 ± 0	**0.98 ± 0.02**	**2.58 ± 1.07**	0.96 ± 0.01	0.02 ± 0.01	0.04 ± 0.01	0.02 ± 0.04	0.04 ± 0.0
	W_s	0.98 ± 0.01	0.96 ± 0.03	6.21 ± 0.36	0.88 ± 0.01	41.31 ± 56.17	0.09 ± 0.01	0.09 ± 0.02	0.16 ± 0.03
	CE	**1.00 ± 0**	0.97 ± 0.01	82.03 ± 35.15	**1.00 ± 0**	2330 ± 2915	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	**1.00 ± 0.00**	0.97 ± 0.03	151.81 ± 27.27	**1.00 ± 0**	**0 ± 0**	0.66 ± 0.06	**0 ± 0**	**0 ± 0**
car	W^*	**1.00 ± 0**	**1.00 ± 0.01**	**1.51 ± 0.35**	0.99 ± 0	**0 ± 0**	0.01 ± 0.00	0.04 ± 0.03	0.01 ± 0
	W_s	**1.00 ± 0**	**1.00 ± 0**	5.15 ± 1.07	0.93 ± 0.02	12.89 ± 2.05	0 ± 0	0.06 ± 0.02	0.08 ± 0.02
	CE	**1.00 ± 0**	**1.00 ± 0**	25.79 ± 18.86	**1.00 ± 0**	225.11 ± 253	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	**1.00 ± 0**	**1.00 ± 0**	504 ± 116.6	**1.00 ± 0**	**0 ± 0**	1.12 ± 0.07	0.40 ± 0	**0 ± 0**
face	W^*	**1.00 ± 0**	**0.99 ± 0.01**	**0.78 ± 0.08**	0.97 ± 0	**0 ± 0**	0.17 ± 0	0.01 ± 0	**0 ± 0**
	W_s	0.98 ± 0.01	0.94 ± 0.26	0.86 ± 0.04	3.15 ± 3.05	2.07 ± 1.04	0.28 ± 0.51	0.04 ± 0.01	0.01 ± 0
	CE	**1.00 ± 0**	0.79 ± 0.31	23.70 ± 8.85	**1.00 ± 0**	16099 ± 16330	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	0.92 ± 0.10	0.52 ± 0.26	745.2 ± 282	0.94 ± 0.07	0.11 ± 0.12	3.50 ± 0.28	0.72 ± 0.01	**0 ± 0**
divorce	W^*	0.99 ± 0.01	0.98 ± 0.02	**0.71 ± 0.41**	0.99 ± 0.01	0.01 ± 0.01	0.03 ± 0	**0 ± 0.05**	0.02 ± 0
	W_s	0.99 ± 0	0.95 ± 0.06	1.54 ± 0.13	0.91 ± 0.01	60.17 ± 70.64	0.04 ± 0.01	0.05 ± 0.01	0.08 ± 0
	CE	**1.00 ± 0**	**0.99 ± 0.02**	2.62 ± 1.21	**1.00 ± 0**	14.11 ± 12.32	**0 ± 0**	**0 ± 0**	**0 ± 0**
	MSE	**1.00 ± 0**	0.97 ± 0.03	47.89 ± 24.31	**1.00 ± 0**	**0 ± 0**	0.73 ± 0.07	**0 ± 0.01**	0.01 ± 0

Table 1: Each dataset contains 4 rows comparing the W^* and W_s against traditional MLPs trained using MSE and CE via SGD given the same network width and depth. The best results are in bold with ↑ / ↓ indicating larger/smaller values preferred.

structure, we use the same depth and width to train 2 additional MLPs via SGD, where instead of HSIC, MSE and CE are used as $\mathcal{H}$. *This allows us to compare NIK to traditional networks of identical sizes that are trained by BP/SGD.*

Observing Table 1, first note that $\mathcal{H} \approx 1$ for both W_s and W^*. This corroborates with Thm. 1 where $\mathcal{H}^*$ is guaranteed given enough layers. Next, notice that $\boldsymbol{\mathcal{T} \approx 0}$, $\boldsymbol{C \approx 0}$, **MSE** $\boldsymbol{\approx 0}$, and **CE** $\boldsymbol{\approx 0}$. These results are aligned with Thm. 2 and its corollaries since they indicate that samples of the same class are merging into a single point while minimizing MSE and CE for classification. Moreover, note that MSE/CE failed to shatter the adversarial dataset while W_s/W^* easily handled the data. This suggest that given the same network size, W_s/W^* is significantly more flexible.

Lastly, notice how W_s and W^* perform favorably against traditional MLPs on almost all benchmarks, confirming that layer-wise network weights can be obtained via basic operations to achieve comparable performance. These results affirmatively answer question 1 while validating layer-wise networks as a promising alternative to BP. *Our theoretical and experimental results strongly suggest that layer-wise networks with "closed-form weights" can match MLP performance.*

What is the Speed Difference? The execution time is also included for reference in Table 1. Since NIK can be obtained via a single forward pass while SGD requires many iterations of backpropagation, NIK *should be faster.* The Time column of Table 1 confirms this expectation by a wide margin. The biggest difference can be observed by comparing the face dataset: W^*/W_s finished in 0.78/0.86 seconds while MSE required 745 seconds, *which is almost a 1000 times difference.*

Does W_s and W^* Experimentally Generalize? With the exception of the two random datasets, the Test Accuracy of W_s consistently performed well against MLPs trained using BP/SGD. This suggests that W_s is a trivially obtainable network solution that generalizes. From an optimization perspective, W^* impressively generalized even better across all datasets. It further differentiates itself on a high dimension Face dataset where it was the only method that avoided overfitting. While the generalizability of W_s and W^* is still ongoing research, the experimental results are promising.

Is the Network Converging to the Neural Indicator Kernel? A visual pattern of *Kernel Sequence* converging toward NIK is shown on the right of Fig. 3. We rearrange the samples of the same class to be adjacent to each other. This allows us to evaluate the kernel quality via its block diagonal structure

	Training Accuracy					Test Accuracy				
	W*	Ws	GP	Arc-cos	NTK	W*	Ws	GP	Arc-cos	NTK
adversarial	**1.00 ± 0.00**	**1.00 ± 0.00**	0.56 ± 0.02	0.53 ± 0.02	0.52 ± 0.01	**0.53 ± 0.00**	0.50 ± 0.16	0.17 ± 0.15	0.25 ± 0.08	0.30 ± 0.06
random	**1.00 ± 0.00**	**1.00 ± 0.00**	0.95 ± 0.02	0.66 ± 0.02	0.63 ± 0.03	0.40 ± 0.02	0.32 ± 0.14	**0.55 ± 0.22**	0.53 ± 0.21	0.37 ± 0.21
spiral	**1.00 ± 0.00**	**1.00 ± 0.00**	0.99 ± 0.00	0.99 ± 0.00	0.99 ± 0.00	**0.99 ± 0.01**	0.97 ± 0.00	0.99 ± 0.01	**0.99 ± 0.01**	0.98 ± 0.02
wine	0.99 ± 0.00	1.00 ± 0.00	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	**0.99 ± 0.03**	0.94 ± 0.03	0.86 ± 0.11	0.94 ± 0.07	0.96 ± 0.04
cancer	**0.99 ± 0.00**	**0.99 ± 0.00**	**0.99 ± 0.00**	**0.99 ± 0.00**	0.98 ± 0.00	**0.98 ± 0.00**	0.98 ± 0.02	0.97 ± 0.02	0.97 ± 0.02	0.97 ± 0.02
car	**1 ± 0.0**	**1 ± 0.0**	**1 ± 0.0**	**1 ± 0.0**	**1 ± 0.0**	**1 ± 0.0**	0.99 ± 0.00	0.99 ± 0.01	**1.0 ± 0.0**	**1.0 ± 0.0**
face	**1.0 ± 0.0**	**1.0 ± 0.0**	0.55 ± 0.02	**1.0 ± 0.0**	**1.0 ± 0.0**	**1.0 ± 0.0**	0.99 ± 0.01	0.22 ± 0.05	0.79 ± 0.32	0.61 ± 0.39
divorce	0.99 ± 0.01	0.99 ± 0.01	**1.00 ± 0.0**	**1.00 ± 0.0**	**1.00 ± 0.0**	**0.99 ± 0.01**	0.97 ± 0.05	0.94 ± 0.06	0.95 ± 0.12	0.97 ± 0.07
cifar10	**0.99 ± 0.01**	0.81 ± 0.00	0.77 ± 0.01	0.94 ± 0.01	0.94 ± 0.01	**0.93 ± 0.01**	0.74 ± 0.01	0.72 ± 0.01	**0.93 ± 0.01**	**0.93 ± 0.01**

Table 2: Comparing Train and test accuracy between recent kernel networks against W_s/W^*. Notice that W^* consistently achieves the highest test Accuracy while W_s performs comparatively to GP. Also, note that W_s and W^* were the only models with a sufficiently large function class to shatter the Adversarial and random dataset. The 10-fold dataset is reshuffled to contrast against Table 1.

quickly. Since Gaussian kernels are restricted to values between 0 and 1, we let white and dark blue be 0 and 1 respectively, where the gradients reflect values in between. Our proof predicts that the *Kernel Sequence* converges to NIK, evolving from an uninformative kernel into a highly discriminating kernel of perfect *block diagonal structures*. Corresponding to the top row, the bottom row plots out the *preactivation* at each layer. As predicted by Thm. 2, the samples of the same class incrementally converge towards a single point. This pattern is consistently observed on all datasets, and the complete collection of the *kernel sequences* for each dataset can be found in App. K.

Comparing to Other Deep Kernel Frameworks. The development of $\mathcal{H}$-*Sequence* primarily focused on the biological motivation to model layer-wise networks; it was not designed to automatically outperforming all existing networks. Table 1 satisfies our primary objective by showing that it already performs comparably against traditional MLPs trained with BP. For the kernel community, here we supply additional experiments comparing W_s/W^* against several recently proposed kernel networks to demonstrate surprisingly competitive results in Table 2. First, note that W_s reliably achieves comparable test accuracy as GP while W^* consistently outperform all kernel networks. This suggests that $\mathcal{H}$-*Sequence* yields networks that not only generalizes competitively against traditional MLPs, it is also comparable to some recently proposed kernel networks. Next notice for the Training Accuracy, only W_s and W^* were sufficiently expressive to shatter both the adversarial and random dataset, confirming the expressiveness of layer-wise networks.

Conclusion. We have comprehensively tested each theoretical claim, demonstrating how a layer-wise network modeled as an $\mathcal{H}$-*Sequence* can yield closed-form solutions that perform comparably to MLPs. The convincing results from our experiments strongly align with the predictions made by our theorems, answering the two central questions of this exploratory work.

Indeed, a repetition of simple rules can incrementally construct powerful networks capable of classifying any pattern, bypassing both BP and SGD. By modeling MLPs as a *Kernel Sequence*, it allows us to design convergent behaviors for layer-wise networks to achieve classification while identifying the network depth.

Limitations. While $\mathcal{H}$-*Sequence* can be designed to bypass BP and "aspirationally" mimic the brain, it *cannot and does not claim* that the sequence itself models the brain; it only models layer-wise networks. Although the similarities are compelling, *any* connection relating $\mathcal{H}$-*Sequence* to the brain is currently premature. We also emphasize that $\mathcal{H}$-*Sequence* was not proposed to outperform existing commercial networks. It is a theoretical framework intended for *analysis* of layer-wise networks to *explore potential biological connections*. Therefore, theoretically proving the existence of a closed-form solution is the primary contribution. While $\mathcal{H}$-*Sequence* yields networks that perform comparably to traditional MLPs and recent kernel networks, carefully engineered networks will likely perform better, e.g., ResNet (He et al., 2016).

Climate Implication. This work is originally motivated by the desire to reduce computational requirements for deep networks, lowering their carbon footprint. For how this work hopes to help fight climate change, please refer to App. A.

Acknowledgements The work described was supported in part by Award Numbers U2COD023375, UH3OD023251, U01 HL089897, U01 HL089856, R01 HL124233, and R01 HL147326 from the NIH, the National Heart, Lung, and Blood Institute, the FDA Center for Tobacco Products (CTP), and NSF IIS 1546428. The authors would also like to thank Davin Hill, Chengzhi Shi, Zulqarnain Khan and Kathia Kirschner for helping to review the paper and provide fruitful discussions.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pages 6155–6166, 2019.

Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. *NeurIPS*, 2019.

S. Arora, S. Du, Zhiyuan Li, R. Salakhutdinov, Ruosong Wang, and Dingli Yu. Harnessing the power of infinitely wide deep nets on small-data tasks. *ArXiv*, abs/1910.01663, 2020.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, Ruslan Salakhutdinov, and Ruosong Wang. On exact computation with an infinitely wide neural net. *NeurIPS*, 2019.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

Yoshua Bengio, Dong-Hyun Lee, Jorg Bornschein, Thomas Mesnard, and Zhouhan Lin. Towards biologically plausible deep learning. *arXiv preprint arXiv:1502.04156*, 2015.

Yoshua Bengio, Thomas Mesnard, Asja Fischer, Saizheng Zhang, and Yuhuai Wu. Stdp-compatible approximation of backpropagation in an energy-based model. *Neural computation*, 29(3):555–577, 2017.

Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *ICLR*, 2017.

Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10835–10845, 2019.

Youngmin Cho and L. Saul. Kernel methods for deep learning. In *NIPS*, 2009.

Corinna Cortes, Mehryar Mohri, and Afshin Rostamizadeh. Algorithms for learning kernels based on centered alignment. *Journal of Machine Learning Research*, 13(Mar):795–828, 2012.

Francis Crick. The recent excitement about neural networks. *Nature*, 337(6203):129–132, 1989.

George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

Richard A Davis, Keh-Shin Lii, and Dimitris N Politis. Remarks on some nonparametric estimates of a density function. In *Selected Works of Murray Rosenblatt*, pages 95–100. Springer, 2011.

Dua Dheeru and Efi Karra Taniskidou. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Shiyu Duan, Shujian Yu, Yunmei Chen, and Jose C Principe. On kernel method–based connectionist models and supervised deep learning without backpropagation. *Neural computation*, 32(1):97–135, 2020.

David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani. Avoiding pathologies in very deep networks. In *Artificial Intelligence and Statistics*, pages 202–210, 2014.

Scott E Fahlman and Christian Lebiere. The cascade-correlation learning architecture. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 1990.

Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936.

gionuno. Arc cosine kernels. `https://github.com/gionuno/arc_cosine_kernels`, 2017.

Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

Stephen Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

Jordan Guerguiev, Timothy P Lillicrap, and Blake A Richards. Towards deep learning with segregated dendrites. *Elife*, 6:e22901, 2017.

Soufiane Hayou, Arnaud Doucet, and Judith Rousseau. On the impact of the activation function on deep neural networks training. *ICML*, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

Geoffrey Hinton. How to do backpropagation in a brain. In *Invited talk at the NIPS'2007 deep learning workshop*, volume 656, 2007.

Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257, 1991.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed ¡today¿].

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, MIT, 2009.

Mandar Kulkarni and Shirish Karande. Layer-wise training of deep networks using kernel similarity. *arXiv preprint arXiv:1703.07115*, 2017.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *ICLR*, 2017.

Jaehoon Lee, Lechao Xiao, Samuel S Schoenholz, Yasaman Bahri, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *NeurIPS*, 2019.

Qianli Liao, Joel Z Leibo, and Tomaso Poggio. How important is weight symmetry in backpropagation? In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

T.P. Lillicrap, A. Santoro, L. Marris, C. Akerman, and G. Hinton. Backpropagation and the brain. In *Nat Rev Neurosci (2020)*, volume 656, 2007.

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. *arXiv preprint arXiv:2006.09549*, 2020.

Sindy Löwe, Peter O'Connor, and Bastiaan S. Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *NeurIPS*, 2019.

Wan-Duo Kurt Ma, JP Lewis, and W Bastiaan Kleijn. The hsic bottleneck: Deep learning without backpropagation. *AAAI*, 2019.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. Convolutional kernel networks. In *Advances in neural information processing systems*, pages 2627–2635, 2014.

maka89. neural-tangent-kernel-uci. `https://github.com/maka89/Deep-Kernel-GP`, 2017.

Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *ICLR*, 2018.

Geoffrey J McLachlan. *Discriminant analysis and statistical pattern recognition*, volume 544. John Wiley & Sons, 2004.

Grasgoire Montavon, Mikio L Braun, and Klaus-Robert Matller. Kernel analysis of deep networks. *Journal of Machine Learning Research*, 12(Sep): 2563–2581, 2011.

Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint arXiv:1605.09522*, 2016.

Radford M Neal. *Bayesian learning for neural networks*, volume 118. Springer Science & Business Media, 2012.

Donglin Niu, Jennifer G Dy, and Michael I Jordan. Multiple non-redundant spectral clustering views. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 831–838, 2010.

Arild Nøkland and Lars Hiller Eidnes. Training neural networks with local error signals. In *International Conference on Machine Learning*, pages 4839–4850. PMLR, 2019.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. *PyTorch.org*, 2017.

Tomaso Poggio, Qianli Liao, and Andrzej Banburski. Complexity control by gradient descent in deep networks. *Nature Communications*, 11(1):1–5, 2020.

Roman Pogodin and P. Latham. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. *ArXiv*, abs/2006.07123, 2020.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

João Sacramento, Rui Ponte Costa, Yoshua Bengio, and Walter Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, pages 8721–8732, 2018.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp. *ACL*, 2019.

Aäron van den Oord, Y. Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, abs/1807.03748, 2018.

Edgar Y Walker. Deep neural networks uncover what the brain likes to see. *Nature Neuroscience*, 10(1): 1–7, 2019.

James CR Whittington and Rafal Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

James CR Whittington and Rafal Bogacz. Theories of error back-propagation in the brain. *Trends in cognitive sciences*, 2019.

A. Wilson, Zhiting Hu, R. Salakhutdinov, and E. Xing. Deep kernel learning. *ArXiv*, abs/1511.02222, 2016.

Chieh Wu, Stratis Ioannidis, Mario Sznaier, Xiangyu Li, David Kaeli, and Jennifer Dy. Iterative spectral method for alternative clustering. In *International Conference on Artificial Intelligence and Statistics*, pages 115–123, 2018.

Chieh T Wu, J. Miller, Y. Chang, M. Sznaier, and Jennifer G. Dy. Solving interpretable kernel dimensionality reduction. In *NeurIPS*, 2019.

Greg Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.

Leo Yu. neural-tangent-kernel-uci. `https://github.com/LeoYu/neural-tangent-kernel-UCI`, 2019.

Anthony M Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature communications*, 10(1):1–7, 2019.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *ICLR*, abs/1611.03530, 2017.

Ding-Xuan Zhou. Universality of deep convolutional neural networks. *Applied and computational harmonic analysis*, 48(2):787–794, 2020.

Jinfeng Zhuang, Ivor W Tsang, and Steven CH Hoi. Two-layer multiple kernel learning. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 909–917, 2011.

Supplementary Material: Deep Layer-wise Networks Have Closed-Form Weights

Appendix A How This Work Relates to Climate Change

Finding an alternative to BP also has significant climate implications. Strubell et al. (2019) have shown that some standard AI models can emit over 626,000 pounds of carbon dioxide; a carbon footprint five times greater than the lifetime usage of a car. This level of emission is simply not sustainable in light of our continual explosive growth. Therefore, the environmental impact of BP necessitates a cheaper alternative. Looking at nature, we can be inspired by the brain's learning capability using only a fraction of the energy. Perhaps artificial neurons can also train without the high energy cost to the environment. This is the moral and the foundational motivation for this work in identifying the existence of W_s. A closed-form solution holds the potential to significantly reduce the computational requirement and carbon footprint. Even if our work ultimately failed to mimic the brain, we hope to inspire the community to identify other closed-form solutions and go beyond BP.

This paper aims to promote the discussion of viewing backpropagation alternatives not only as an academic exercise but also as a climate imperative. Yet, this topic is largely ignored by the community. The authors believe the energy costs of training Neural Networks are having a detrimental climate impact and should be an added topic of interest. The earth also needs an advocate, why not us? Therefore, we as a community, must begin addressing how we can ameliorate our own carbon footprint. This exploratory work aims to share a potential path forward for further research that may address these concerns with the community. While W_s is still not ready for commercial usage, we sincerely hope that the community begins to build novel algorithms over our work on $\mathcal{H}$*-Sequence* and identify a simpler and cheaper path to train our networks.

Appendix B Proof for Theorem 1

Theorem 1: *For any $\mathcal{H}_0$, there exists a set of bandwidths σ_l and a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^{L}$ parameterized by $W_l = W_s$ in Eq. (5) such that:*

I. $\mathcal{H}_L$ can approach arbitrarily close to $\mathcal{H}^*$ such that for any $L > 1$ and $\delta > 0$ we can achieve

$$\mathcal{H}^* - \mathcal{H}_L \leq \delta, \tag{16}$$

II. as $L \to \infty$, the $\mathcal{H}$*-Sequence* converges to the global optimum where

$$\lim_{L\to\infty} \mathcal{H}_L = \mathcal{H}^*, \tag{17}$$

III. the convergence is strictly monotonic where

$$\mathcal{H}_l > \mathcal{H}_{l-1} \quad \forall l \geq 1. \tag{18}$$

Lemma 1. *Given σ_0 and σ_1 as the σ values from the last layer and the current layer, then there exists a lower bound for $\mathcal{H}_l$, denoted as $\mathscr{L}(\sigma_0, \sigma_1)$ such that*

$$\mathcal{H}_l \geq \mathscr{L}(\sigma_0, \sigma_1). \tag{19}$$

Basic Background, Assumptions, and Notations.

1. The simulation of this theorem for Adversarial and Random data is also publicly available on `https://github.com/anonymous`.

2. Here we show that this bound can be established given the last 2 layers.
3. σ_0 is the σ value of the previous layer
4. σ_1 is the σ value of the current layer
5. τ is the number of classes
6. n is total number of samples
7. n_i is number of samples in the i^{th} class
8. $\mathcal{S}$ is a set of all i, j sample pairs where r_i and r_j belong to the same class.
9. $\mathcal{S}^c$ is a set of all i, j sample pairs where r_i and r_j belong to different same classes.
10. $\mathcal{S}^\beta$ is a set of all i, j sample pairs that belongs to the same β^{th} classes.
11. $r_i^{(\alpha)}$ is the i^{th} sample in the α^{th} class among τ classes.
12. We assume no $r_i \neq r_j$ pair are equal $\forall i \neq j$.
13. Among all $r_i \neq r_j$ pairs, there exists an optimal r_i^*, r_j^* pair where $\langle r_i^*, r_j^* \rangle \geq \langle r_i, r_j \rangle \; \forall r_i \neq r_i^*$ and $r_j \neq r_j^*$. We denote this maximum inner product as
$$u_{\sigma_0} = \langle r_i^*, r_j^* \rangle. \tag{20}$$
14. Here, each r_i sample is assumed to be a sample in the RKHS of the Gaussian kernel, therefore all inner products are bounded such that
$$0 \leq \langle r_i, r_j \rangle \leq u_{\sigma_0}. \tag{21}$$
15. We let W be
$$W_s = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)} \end{bmatrix}. \tag{22}$$
Instead of using an optimal W^* defined as $W^* = \arg\max_W H_l(W)$, we use a suboptimal W_s where each dimension is simply the average direction of each class: $\frac{1}{\sqrt{\zeta}}$ is a unnecessary normalizing constant $\zeta = ||W_s||_2^2$. By using W_s, this implies that the $\mathcal{H}$ we obtain is already a lower bound compare $\mathcal{H}$ obtained by W^*. But, we will use this suboptimal W_s to identify an even lower bound. Note that based on the definition W^*, we have the property $\mathcal{H}(W^*) \geq \mathcal{H}(W) \, \forall W$.
16. We note that the objective $\mathcal{H}$ is
$$\mathcal{H} = \underbrace{\sum_{i,j \in \mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}}}_{\mathscr{W}} - \underbrace{\sum_{i,j \in \mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}}}_{\mathscr{B}} \tag{23}$$
where we let $\mathscr{W}$ be the summation of terms associated with the within cluster pairs, and let $\mathscr{B}$ be the summation of terms associated with the between cluster pairs.

Proof.

The equation is further divided into smaller parts organized into multiple sections.

For sample pairs in $\mathcal{S}$. The first portion of the function can be split into multiple classes where

$$\mathscr{W} = \underbrace{\sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{(r_i^{(1)} - r_j^{(1)})^T W W^T (r_i^{(1)} - r_j^{(1)})}{2\sigma_1^2}}}_{\mathscr{W}_1} + ... + \underbrace{\sum_{\mathcal{S}^\tau} \Gamma_{i,j} e^{-\frac{(r_i^{(\tau)} - r_j^{(\tau)})^T W W^T (r_i^{(\tau)} - r_j^{(\tau)})}{2\sigma_1^2}}}_{\mathscr{W}_\tau} \tag{24}$$

Realize that to find the lower bound, we need to determine the minimum possible value of each term which translates to **maximum** possible value of each exponent. Without of loss of generality we can find the lower bound for one term and generalize its results to other terms due to their similarity. Let us focus on the numerator of the exponent from $\mathscr{W}_1$. Given W_s as W, our goal is identify the **maximum** possible value for

$$\underbrace{(r_i^{(1)} - r_j^{(1)})^T W}_{\Pi_1} \underbrace{W^T (r_i^{(1)} - r_j^{(1)})}_{\Pi_2}. \tag{25}$$

Zoom in further by looking only at Π_1, we have the following relationships

$$\Pi_1 = \underbrace{{r_i^{(1)}}^T W}_{\xi_1} - \underbrace{{r_j^{(1)}}^T W}_{\xi_2} \tag{26}$$

$$\xi_1 = \frac{1}{\sqrt{\zeta}} {r_i^{(1)}}^T \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & \dots & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{27}$$

$$= \frac{1}{\sqrt{\zeta}} {r_i^{(1)}}^T \begin{bmatrix} (r_1^{(1)} + \dots + r_{n_1}^{(1)}) & \dots & (r_1^{(\tau)} + \dots + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{28}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} {r_j^{(1)}}^T \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & \dots & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{29}$$

$$= \frac{1}{\sqrt{\zeta}} {r_j^{(1)}}^T \begin{bmatrix} (r_1^{(1)} + \dots + r_{n_1}^{(1)}) & \dots & (r_1^{(\tau)} + \dots + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{30}$$

By knowing that the inner product is constrained between $[0, u_{\sigma_0}]$, we know the maximum possible value for ξ_1 and the minimum possible value for ξ_2 to be

$$\xi_1 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 + (n_1 - 1)u_{\sigma_0} & n_2 u_{\sigma_0} & n_3 u_{\sigma_0} & \dots & n_\tau u_{\sigma_0} \end{bmatrix} \tag{31}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \end{bmatrix}. \tag{32}$$

Which leads to

$$\Pi_1 = \frac{1}{\sqrt{\zeta}}(\xi_1 - \xi_2) = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} (n_1 - 1)u_{\sigma_0} & n_2 u_{\sigma_0} & n_3 u_{\sigma_0} & \dots & n_\tau u_{\sigma_0} \end{bmatrix} \tag{33}$$

Since $\Pi_2^T = \Pi_1$ we have

$$\Pi_1 \Pi_2 = \frac{1}{\zeta}[(n_1 - 1)^2 u_{\sigma_0}^2 + n_2^2 u_{\sigma_0}^2 + n_3^2 u_{\sigma_0}^2 + \dots + n_\tau^2 u_{\sigma_0}^2] \tag{34}$$

$$= \frac{1}{\zeta}[(n_1 - 1)^2 + n_2^2 + n_3^2 + \dots + n_\tau^2] u_{\sigma_0}^2 \tag{35}$$

The lower bound for just the $\mathscr{W}_1$ term emerges as

$$\mathscr{W}_1 \geq \sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{[(n_1-1)^2 + n_2^2 + n_3^2 + \dots + n_\tau^2] u_{\sigma_0}^2}{2\zeta\sigma_1^2}}. \tag{36}$$

To further condense the notation, we define the following constant

$$\mathscr{N}_g = \frac{1}{2\zeta}[n_1^2 + n_2^2 + \dots + (n_g - 1)^2 + \dots + n_\tau^2]. \tag{37}$$

Therefore, the lower bound for $\mathscr{W}_1$ can be simplified as

$$\mathscr{W}_1 \geq \sum_{\mathcal{S}^1} \Gamma_{i,j} e^{-\frac{\mathscr{N}_1 u_{\sigma_0}^2}{\sigma_1^2}} \tag{38}$$

and the general pattern for any $\mathscr{W}_g$ becomes

$$\mathscr{W}_g \geq \sum_{\mathcal{S}^i} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}. \tag{39}$$

The lower bound for the entire set of $\mathcal{S}$ then becomes

$$\sum_{i,j \in \mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}} = \mathscr{W}_1 + ... + \mathscr{W}_\tau \geq \underbrace{\sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}}_{\text{Lower bound}}. \tag{40}$$

For sample pairs in $\mathcal{S}^c$. To simplify the notation, we note that

$$-\mathscr{B}_{g_1,g_2} = -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{(r_i^{(g_1)} - r_j^{(g_2)})^T W W^T (r_i^{(g_1)} - r_j^{(g_2)})}{2\sigma_1^2}} \tag{41}$$

$$= -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T ((r_i^{(g_1)} - r_j^{(g_1)}))((r_i^{(g_1)} - r_j^{(g_2)}))^T W)}{2\sigma_1^2}} \tag{42}$$

$$= -\sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(g_1,g_2)} W)}{2\sigma_1^2}} \tag{43}$$

$$\tag{44}$$

We now derived the lower bound for the sample pairs in $\mathcal{S}^c$. We start by writing out the entire summation sequence for $\mathscr{B}$.

$$\begin{aligned}
\mathscr{B} = & -\underbrace{\sum_{i \in \mathcal{S}^1} \sum_{j \in \mathcal{S}^2} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(1,2)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{1,2}} - \underbrace{...}_{\mathscr{B}_{g_1 \neq g_2}} - \underbrace{\sum_{i \in \mathcal{S}^1} \sum_{j \in \mathcal{S}^\tau} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(1,\tau)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{1,\tau}} \\
& -\underbrace{\sum_{i \in \mathcal{S}^2} \sum_{j \in \mathcal{S}^1} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(2,1)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{2,1}} - \underbrace{...}_{\mathscr{B}_{g_1 \neq g_2}} - \underbrace{\sum_{i \in \mathcal{S}^2} \sum_{j \in \mathcal{S}^\tau} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(2,\tau)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{2,\tau}} \\
& ... \\
& -\underbrace{\sum_{i \in \mathcal{S}^\tau} \sum_{j \in \mathcal{S}^1} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(\tau,1)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{\tau,1}} - \underbrace{...}_{\mathscr{B}_{g_1 \neq g_2}} - \underbrace{\sum_{i \in \mathcal{S}^{\tau-1}} \sum_{j \in \mathcal{S}^\tau} |\Gamma_{i,j}| e^{-\frac{\text{Tr}(W^T A_{i,j}^{(\tau-1,\tau)} W)}{2\sigma_1^2}}}_{\mathscr{B}_{\tau-1,\tau}}
\end{aligned} \tag{45}$$

Using a similar approach with the terms from $\mathscr{W}$, note that $\mathscr{B}$ is a negative value, so we need to maximize this term to obtain a lower bound. Consequently, the key is to determine the **minimal** possible values for each exponent term. Since every one of them will behave very similarly, we can simply look at the numerator of the exponent from $\mathscr{B}_{1,2}$ and then arrive to a more general conclusion. Given W_s as W, our goal is to identify the **minimal** possible value for

$$\underbrace{(r_i^{(1)} - r_j^{(2)})^T W}_{\Pi_1} \underbrace{W^T (r_i^{(1)} - r_j^{(2)})}_{\Pi_2}. \tag{46}$$

Zoom in further by looking only at Π_1, we have the following relationships

$$\Pi_1 = \underbrace{r_i^{(1)^T} W}_{\xi_1} - \underbrace{r_j^{(2)^T} W}_{\xi_2} \tag{47}$$

$$\xi_1 = \frac{1}{\sqrt{\zeta}} r_i^{(1)^T} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{48}$$

$$= \frac{1}{\sqrt{\zeta}} r_i^{(1)^T} \begin{bmatrix} (r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{49}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} r_j^{(2)^T} \begin{bmatrix} \sum_\iota r_\iota^{(1)} & \sum_\iota r_\iota^{(2)} & ... & \sum_\iota r_\iota^{(\tau)} \end{bmatrix} \tag{50}$$

$$= \frac{1}{\sqrt{\zeta}} r_j^{(2)^T} \begin{bmatrix} (r_1^{(1)} + ... + r_{n_1}^{(1)}) & ... & (r_1^{(\tau)} + ... + r_{n_\tau}^{(\tau)}) \end{bmatrix} \tag{51}$$

By knowing that the inner product is constrained between $[0, u_{\sigma_0}]$, we know the **minimum** possible value for ξ_1 and the **maximum** possible value for ξ_2 to be

$$\xi_1 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 & 0 & 0 & ... & 0 \end{bmatrix} \tag{52}$$

$$\xi_2 = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} n_1 u_{\sigma_0} & 1 + (n_2 - 1) u_{\sigma_0} & n_3 u_{\sigma_0} & ... & n_\tau u_{\sigma_0} \end{bmatrix} \tag{53}$$

Which leads to

$$\Pi_1 = \frac{1}{\sqrt{\zeta}} (\xi_1 - \xi_2) = \frac{1}{\sqrt{\zeta}} \begin{bmatrix} 1 - n_1 u_{\sigma_0} & -(1 + (n_2 - 1) u_{\sigma_0}) & -n_3 u_{\sigma_0} & ... & -n_\tau u_{\sigma_0} \end{bmatrix} \tag{54}$$

Since $\Pi_2^T = \Pi_1$ we have

$$\Pi_1 \Pi_2 = \frac{1}{\zeta} [(1 - n_1 u_{\sigma_0})^2 + (1 + (n_2 - 1) u_{\sigma_0})^2 + n_3^2 u_{\sigma_0}^2 + ... + n_\tau^2 u_{\sigma_0}^2]. \tag{55}$$

The lower bound for just the $\mathscr{B}_{1,2}$ term emerges as

$$-\mathscr{B}_{1,2} \geq -\sum_{\mathcal{S}^1} \sum_{\mathcal{S}^2} |\Gamma_{i,j}| e^{-\frac{(1 - n_1 u_{\sigma_0})^2 + (1 + (n_2 - 1) u_{\sigma_0})^2 + n_3^2 u_{\sigma_0}^2 + ... + n_\tau^2 u_{\sigma_0}^2}{2 \zeta \sigma_1^2}}. \tag{56}$$

To further condense the notation, we define the following function

$$\begin{aligned} \mathscr{N}_{g_1, g_2}(u_{\sigma_0}) = \frac{1}{2\zeta} [& n_1^2 u_{\sigma_0}^2 + n_2^2 u_{\sigma_0}^2 + ... \\ & + (1 - n_{g_1} u_{\sigma_0})^2 + ... + (1 + (n_{g_2} - 1) u_{\sigma_0})^2 \\ & + ... + n_\tau^2 u_{\sigma_0}^2]. \end{aligned} \tag{57}$$

Note that while for $\mathcal{S}$, the u_{σ_0} term can be separated out. But here, we cannot, and therefore $\mathscr{N}$ here must be a function of u_{σ_0}. Therefore, the lower bound for $\mathscr{B}_{1,2}$ can be simplified into

$$-\mathscr{B}_{1,2} \geq -\sum_{\mathcal{S}^1} \sum_{\mathcal{S}^2} |\Gamma_{i,j}| e^{-\frac{\mathscr{N}_{1,2}(u_{\sigma_0})}{\sigma_1^2}} \tag{58}$$

and the general pattern for any $\mathscr{B}_{g_1, g_2}$ becomes

$$-\mathscr{B}_{g_1, g_2} \geq -\sum_{\mathcal{S}^{g1}} \sum_{\mathcal{S}^{g2}} \Gamma_{i,j} e^{-\frac{\mathscr{N}_{g_1, g_2}(u_{\sigma_0})}{\sigma_1^2}}. \tag{59}$$

The lower bound for the entire set of $\mathcal{S}^c$ then becomes

$$-\sum_{i,j \in \mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma_1^2}} = -\mathscr{B}_{1,2} - \mathscr{B}_{1,3} - ... - \mathscr{B}_{\tau-1,\tau} \tag{60}$$

$$\geq \underbrace{-\sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathscr{N}_{g_1, g_2}(u_{\sigma_0})}{\sigma_1^2}}}_{\text{Lower bound}}. \tag{61}$$

Putting $\mathcal{S}$ and $\mathcal{S}^c$ Together.

$$\mathcal{H} = \mathscr{W} + \mathscr{B} \tag{62}$$

$$\geq \underbrace{\sum_{g=1}^{\tau}\sum_{\mathcal{S}^g}\Gamma_{i,j}e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}}}_{\text{Lower bound of } \mathscr{W}} - \underbrace{\sum_{g_1\neq g_2}^{\tau}\sum_{i\in\mathcal{S}^{g_1}}\sum_{j\in\mathcal{S}^{g_2}}|\Gamma_{i,j}|e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}}_{\text{Lower bound of } \mathscr{B}}. \tag{63}$$

Therefore, we have identified a lower bound that is a function of σ_0 and σ_1 where

$$\mathscr{L}(\sigma_0,\sigma_1) = \sum_{g=1}^{\tau}\sum_{\mathcal{S}^g}\Gamma_{i,j}e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}} - \sum_{g_1\neq g_2}^{\tau}\sum_{i\in\mathcal{S}^{g_1}}\sum_{j\in\mathcal{S}^{g_2}}|\Gamma_{i,j}|e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}. \tag{64}$$

From the lower bound, it is obvious why it is a function of σ_1. The lower bound is also a function of σ_0 because u_{σ_0} is actually a function of σ_0. To specifically clarify this point, we have the next lemma.

□

Lemma 2. *The u_{σ_0} used in Lemma 1 is a function of σ_0 where u_{σ_0} approaches to zero as σ_0 approaches to zero, i.e.*

$$\lim_{\sigma_0\to 0} u_{\sigma_0} = 0. \tag{65}$$

Assumptions and Notations.

1. We use Fig. 4 to help clarify the notations. We here only look at the last 2 layers.
2. We let $\mathcal{H}_0$ be the $\mathcal{H}$ of the last layer, and $\mathcal{H}_1$, the $\mathcal{H}$ of the current layer.
3. The input of the data is X with each sample as x_i, and the output of the previous layer are denoted as r_i. ψ_{σ_0} is the feature map of the previous layer using σ_0 and ψ_{σ_1} corresponds to the current layer.

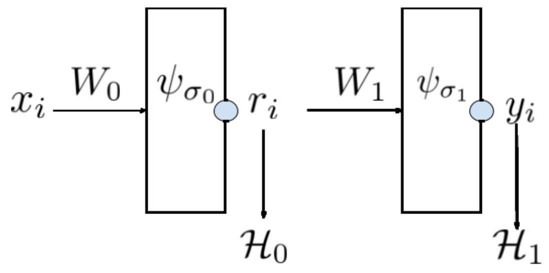

Figure 4: Figure of a 2 layer network.

4. As defined from Lemma 1, among all $r_i \neq r_j$ pairs, there exists an optimal r_i^*, r_j^* pair where $\langle r_i^*, r_j^*\rangle \geq \langle r_i, r_j\rangle$ $\forall r_i \neq r_i^*$ and $r_j \neq r_j^*$. We denote this maximum inner product as

$$u_{\sigma_0} = \langle r_i^*, r_j^*\rangle. \tag{66}$$

Proof.

Given Fig. 4, the equation for $\mathcal{H}_0$ is

$$\mathcal{H}_0 = \sum_{i,j\in\mathcal{S}}\Gamma_{i,j}e^{-\frac{(x_i-x_j)^T WW^T(x_i-x_j)}{2\sigma_0^2}} - \sum_{i,j\in\mathcal{S}^c}|\Gamma_{i,j}|e^{-\frac{(x_i-x_j)^T WW^T(x_i-x_j)}{2\sigma_0^2}} \tag{67}$$

$$= \sum_{i,j\in\mathcal{S}}\Gamma_{i,j}\langle\psi_{\sigma_0}(x_i),\psi_{\sigma_0}(x_j)\rangle - \sum_{i,j\in\mathcal{S}^c}|\Gamma_{i,j}|\langle\psi_{\sigma_0}(x_i),\psi_{\sigma_0}(x_j)\rangle \tag{68}$$

Notice that as $\sigma_0 \to 0$, we have

$$\lim_{\sigma_0 \to 0} \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j) \rangle = \begin{cases} 0 & \forall i \neq j \\ 1 & \forall i = j \end{cases} . \tag{69}$$

In other words, as $\sigma_0 \to 0$, the samples r_i in the RKHS of a Gaussian kernel approaches orthogonal to all other samples. Given this fact, it also implies that the σ_0 controls the inner product magnitude in RKHS space of the maximum sample pair r_i^*, r_j^*. We define this maximum inner product as

$$\langle \psi_{\sigma_0}(x_i^*), \psi_{\sigma_0}(x_j^*) \rangle \geq \langle \psi_{\sigma_0}(x_i), \psi_{\sigma_0}(x_j) \rangle \tag{70}$$

or equivalently

$$\langle r_i^*, r_j^* \rangle \geq \langle r_i, r_j \rangle \tag{71}$$

Therefore, given a σ_0, it controls the upper bound of the inner product. Notice that as $\sigma_0 \to 0$, every sample in RKHS becomes orthogonal. Therefore, the upper bound of $\langle r_i, r_j \rangle$ also approaches 0 when $r_i \neq r_j$. From this, we see the relationship

$$\lim_{\sigma_0 \to 0} u_{\sigma_0} = \lim_{\sigma_0 \to 0} \exp -(|.|/\sigma_0^2) = 0 \tag{72}$$

, where $|.|$ is bounded and has a minimum and maximum, because we have finite number of samples.

□

Lemma 3. *Given any fixed $\sigma_1 > 0$, the lower bound $\mathscr{L}(\sigma_0, \sigma_1)$ is a function with respect to σ_0 and as $\sigma_0 \to 0$, $\mathscr{L}(\sigma_0, \sigma_1)$ approaches the function*

$$\mathscr{L}(\sigma_1) = \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{1}{\zeta \sigma_1^2}}. \tag{73}$$

At this point, if we let $\sigma_1 \to 0$, we have

$$\lim_{\sigma_1 \to 0} \mathscr{L}(\sigma_1) = \sum_{i,j \in \mathcal{S}} \Gamma_{i,j} \tag{74}$$

$$= \mathcal{H}^*. \tag{75}$$

Proof.

Given Lemma 2, we know that

$$\lim_{\sigma_0 \to 0} u_{\sigma_0} = 0. \tag{76}$$

Therefore, having $\sigma_0 \to 0$ is equivalent to having $u_{\sigma_0} \to 0$. Since Lemma 1 provide the equation of a lower bound that is a function of u_{σ_0}, this lemma is proven by simply evaluating $\mathscr{L}(\sigma_0, \sigma_1)$ as $u_{\sigma_0} \to 0$. Following these steps, we have

$$\mathscr{L}(\sigma_1) = \lim_{u_{\sigma_0} \to 0} \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} e^{-\frac{\mathscr{N}_g u_{\sigma_0}^2}{\sigma_1^2}} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{\mathscr{N}_{g_1,g_2}(u_{\sigma_0})}{\sigma_1^2}}, \tag{77}$$

$$= \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} - \sum_{g_1 \neq g_2}^{\tau} \sum_{i \in \mathcal{S}^{g_1}} \sum_{j \in \mathcal{S}^{g_2}} |\Gamma_{i,j}| e^{-\frac{1}{\zeta \sigma_1^2}}. \tag{78}$$

At this point, as $\sigma_1 \to 0$, our lower bound reaches the global maximum

$$\lim_{\sigma_1 \to 0} \mathscr{L}(\sigma_1) = \sum_{g=1}^{\tau} \sum_{\mathcal{S}^g} \Gamma_{i,j} = \sum_{i,j \in \mathcal{S}} \Gamma_{i,j} \tag{79}$$

$$= \mathcal{H}^*. \tag{80}$$

□

Lemma 4. *Given any $\mathcal{H}_{l-2}$, $\delta > 0$, there exists a $\sigma_0 > 0$ and $\sigma_1 > 0$ such that*

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta. \tag{81}$$

Proof.

Observation 1.

Note that the objective of $\mathcal{H}_l$ is

$$\begin{aligned}\mathcal{H}_l = \max_W \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i^{(\mathcal{S})}-r_j^{(\mathcal{S})})^T W W^T (r_i^{(\mathcal{S})}-r_j^{(\mathcal{S})})}{2\sigma_1^2}} \\ - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i^{(\mathcal{S}^c)}-r_j^{(\mathcal{S}^c)})^T W W^T (r_i^{(\mathcal{S}^c)}-r_j^{(\mathcal{S}^c)})}{2\sigma_1^2}}.\end{aligned} \tag{82}$$

Since the Gaussian kernel is bounded between 0 and 1, the theoretical maximum of $\mathcal{H}^*$ is when the kernel is 1 for $\mathcal{S}$ and 0 for $\mathcal{S}^c$ with the theoretical maximum as $\mathcal{H}^* = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}$. Therefore Eq. (81) inequality is equivalent to

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j} - \mathcal{H}_l \leq \delta. \tag{83}$$

Observation 2.

If we choose a σ_0 such that

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \quad \text{and} \quad \mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2} \tag{84}$$

then we have identified the condition where $\sigma_0 > 0$ and $\sigma_1 > 0$ such that

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j} - \mathscr{L}(\sigma_0, \sigma_1) \leq \delta. \tag{85}$$

Note that the $\mathscr{L}^*(\sigma_1)$ is a continuous function of σ_1. Therefore, a σ_1 exists such that $\mathscr{L}^*(\sigma_1)$ can be set arbitraty close to $\mathcal{H}^*$. Hence, we choose an σ_1 that has the following property:

$$\mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2}. \tag{86}$$

We next fix σ_1, we also know $\mathscr{L}(\sigma_0, \sigma_1)$ is a continuous function of σ_0, and it has a limit $\mathscr{L}^*(\sigma_1)$ as σ_0 approaches to 0, hence there exits a σ_0, where

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \tag{87}$$

Then we have:

$$\mathscr{L}^*(\sigma_1) - \mathscr{L}(\sigma_0, \sigma_1) \leq \frac{\delta}{2} \quad \text{and} \quad \mathcal{H}^* - \mathscr{L}^*(\sigma_1) \leq \frac{\delta}{2}. \tag{88}$$

By adding the two $\frac{\delta}{2}$, we conclude the proof.

□

Lemma 5. *There exists a Kernel Sequence $\{\phi_{l^\circ}\}_{l=1}^L$ parameterized by a set of weights W_l and a set of bandwidths σ_l such that*

$$\lim_{l\to\infty} \mathcal{H}_l = \mathcal{H}^*, \quad \mathcal{H}_{l+1} > \mathcal{H}_l \quad \forall l \tag{89}$$

Before, the proof, we use the following figure, Fig. 5, to illustrate the relationship between *Kernel Sequence* $\{\phi_{l^\circ}\}_{l=1}^L$ that generates the *$\mathcal{H}$-Sequence* $\{\mathcal{H}_l\}_{l=1}^L$. By solving a network greedily, we separate the network into L separable problems. At each additional layer, we rely on the weights learned from the previous layer. At each network, we find σ_{l-1}, σ_l, and W_l for the next network. We also note that since we only need to prove the existence of a solution, this proof is done by ***Proof by Construction***, i.e, we only need to show an example of its existence. Therefore, this proof consists of us constructing a *$\mathcal{H}$-Sequence* which satisfies the lemma.

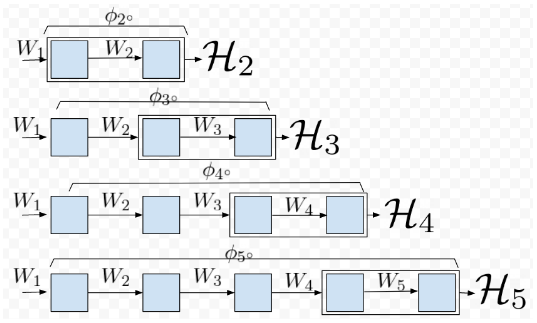

Figure 5: Relating *Kernel Sequence* to $\mathcal{H}$-*Sequence*.

Proof.

We first note that from Lemma 4, we have previously proven given any $\mathcal{H}_{l-2}$, $\delta > 0$, there exists a $\sigma_0 > 0$ and $\sigma_1 > 0$ such that

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{90}$$

This implies that based on Fig. 5, at any given layer, we could reach arbitrarily close to $\mathcal{H}^*$. Given this, we list the 2 steps to build the $\mathcal{H}$-*Sequence*.

Step 1: Define $\{\mathcal{E}_n\}_{n=1}^{\infty}$ as a sequence of numbers $\mathcal{H}^* - \frac{\mathcal{H}^*-\mathcal{H}_0}{n}$ on the real line. We have the following properties for this sequence:

$$\lim_{n\to\infty} \mathcal{E}_n = \mathcal{H}^*, \quad \mathcal{E}_1 = \mathcal{H}_0. \tag{91}$$

Using these two properties, for any $\mathcal{H}_{l-1} \in [\mathcal{H}_0, \mathcal{H}^*]$ there exist an unique n, where

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1}. \tag{92}$$

Step 2: For any given l, we choose δ_l to satisfies Eq. (90) by the following procedure, First find an n that satisfies

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1}, \tag{93}$$

and second define δ_l to be

$$\delta_l = \mathcal{H}^* - \mathcal{E}_{n+1}. \tag{94}$$

To satisfy Eq. (90), the following must be true.

$$\mathcal{H}^* - \mathcal{H}_{l-1} \leq \delta_{l-1}. \tag{95}$$

and further we found n such that

$$\mathcal{E}_n \leq \mathcal{H}_{l-1} < \mathcal{E}_{n+1} \implies \mathcal{H}^* - \mathcal{E}_n \geq \mathcal{H}^* - \mathcal{H}_{l-1} > \mathcal{H}^* - \mathcal{E}_{n+1}. \tag{96}$$

Thus combining Eq. (94), Eq. (95), and Eq. (96) we have

$$\delta_{l-1} > \delta_l. \tag{97}$$

Therefore, $\{\delta_l\}$ is a decreasing sequence.

Step 3: Note that $\{\mathcal{E}_n\}$ is a converging sequence where

$$\lim_{n\to\infty} \mathcal{H}^* - \frac{\mathcal{H}^* - \mathcal{H}_0}{n} = \mathcal{H}^*. \tag{98}$$

Therefore, $\{\Delta_n\} = \mathcal{H}^* - \{\mathcal{E}_n\}$ is also a converging sequence where

$$\lim_{n\to\infty} \mathcal{H}^* - \mathcal{H}^* + \frac{\mathcal{H}^* - \mathcal{H}_0}{n} = 0 \tag{99}$$

and $\{\delta_l\}$ is a subsequence of $\{\Delta_l\}$. Since any subsequence of a converging sequence also converges to the same limit, we know that

$$\lim_{l\to\infty} \delta_l = 0. \tag{100}$$

Following this construction, if we always choose $\mathcal{H}_l$ such that

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{101}$$

As $l \to \infty$, the inequality becomes

$$\mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l \leq \lim_{l\to\infty} \delta_l, \tag{102}$$

$$\leq 0. \tag{103}$$

Since we know that

$$\mathcal{H}^* - \mathcal{H}_l \geq 0 \, \forall l. \tag{104}$$

The condition of

$$0 \leq \mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l \leq 0 \tag{105}$$

is true only if

$$\mathcal{H}^* - \lim_{l\to\infty} \mathcal{H}_l = 0. \tag{106}$$

This allows us to conclude

$$\mathcal{H}^* = \lim_{l\to\infty} \mathcal{H}_l. \tag{107}$$

Proof of the Monotonic Improvement.

Given Eq. (92) and Eq. (94), at each step we have the following:

$$\mathcal{H}_{l-1} < \mathcal{E}_{n+1} \tag{108}$$

$$\leq \mathcal{H}^* - \delta_l. \tag{109}$$

Rearranging this inequality, we have

$$\delta_l < \mathcal{H}^* - \mathcal{H}_{l-1}. \tag{110}$$

By combining the inequalities from Eq. (110) and Eq. (101), we have the following relationships.

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l < \mathcal{H}^* - \mathcal{H}_{l-1} \tag{111}$$

$$\mathcal{H}^* - \mathcal{H}_l < \mathcal{H}^* - \mathcal{H}_{l-1} \tag{112}$$

$$-\mathcal{H}_l < -\mathcal{H}_{l-1} \tag{113}$$

$$\mathcal{H}_l > \mathcal{H}_{l-1}, \tag{114}$$

$$\tag{115}$$

which concludes the proof of theorem.

□

Appendix C Proof for Theorem 2

Theorem 2: *As $l \to \infty$ and $\mathcal{H}_l \to \mathcal{H}^*$, the following properties are satisfied:*

I the scatter ratio approaches 0 where

$$\lim_{l\to\infty} \frac{\mathrm{Tr}(S_w^l)}{\mathrm{Tr}(S_b^l)} = 0 \tag{116}$$

II the *Kernel Sequence* converges to the following kernel:

$$\lim_{l\to\infty} \mathcal{K}(x_i, x_j)^l = \mathcal{K}^* = \begin{cases} 0 & \forall i, j \in \mathcal{S}^c \\ 1 & \forall i, j \in \mathcal{S} \end{cases} . \tag{117}$$

Proof. We start by proving condition II starting from the $\mathcal{H}$ objective using a GK

$$\max_W \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}\mathcal{K}_W(r_i,r_j) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i,r_j) \tag{118}$$

$$\max_W \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}| e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} \tag{119}$$

Given that $\mathcal{H}_l \to \mathcal{H}^*$, and the fact that $0 \leq \mathcal{K}_W \leq 1$, this implies that the following condition must be true:

$$\mathcal{H}^* = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j} = \sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|(0). \tag{120}$$

Based on Eq. (90), our construction at each layer ensures to satisfy

$$\mathcal{H}^* - \mathcal{H}_l \leq \delta_l. \tag{121}$$

Substituting the definition of $\mathcal{H}^*$ and $\mathcal{H}_l$, we have

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1) - \left[\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}\mathcal{K}_W(r_i,r_j) - \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i,r_j)\right] \leq \delta_l \tag{122}$$

$$\sum_{i,j\in\mathcal{S}} \Gamma_{i,j}(1-\mathcal{K}_W(r_i,r_j)) + \sum_{i,j\in\mathcal{S}^c} |\Gamma_{i,j}|\mathcal{K}_W(r_i,r_j) \leq \delta_l. \tag{123}$$

Since every term within the summation in Eq. (123) is positive, this implies

$$1 - \mathcal{K}_W(r_i,r_j) \leq \delta_l \quad i,j\in\mathcal{S} \tag{124}$$

$$\mathcal{K}_W(r_i,r_j) \leq \delta_l \quad i,j\in\mathcal{S}^c. \tag{125}$$

So as $l \to \infty$ and $\delta_l \to 0$, every component getting closer to limit Kernel, i.e, taking the limit from both sides and using the fact that is proven is theorem 1 $\lim_{l\to\infty} \delta_l = 0$ leads to

$$\lim_{l\to\infty} 1 \leq \mathcal{K}_W(r_i,r_j) \quad i,j\in\mathcal{S} \tag{126}$$

$$\lim_{l\to\infty} \mathcal{K}_W(r_i,r_j) \leq 0 \quad i,j\in\mathcal{S}^c \tag{127}$$

both terms must instead be strictly equality. Therefore, we see that at the limit point $\mathcal{K}_W$ would have the form

$$\mathcal{K}^* = \begin{cases} 0 & \forall i,j\in\mathcal{S}^c \\ 1 & \forall i,j\in\mathcal{S} \end{cases}. \tag{128}$$

First Property:

Using Eq. (124) and Eq. (125) we have:

$$1 - \delta_l \leq e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} \quad i,j\in\mathcal{S} \tag{129}$$

$$e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} \leq \delta_l \quad i,j\in\mathcal{S}^c. \tag{130}$$

As $\lim_{l\to\infty} \delta_l = 0$, taking the limit from both side leads to:

$$\begin{cases} e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} = 1 & \forall i,j\in\mathcal{S} \\ e^{-\frac{(r_i-r_j)^T WW^T(r_i-r_j)}{2\sigma^2}} = 0 & \forall i,j\in\mathcal{S}^c \end{cases}. \tag{131}$$

If we take the log of the conditions, we get

$$\begin{cases} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = 0 & \forall i,j \in \mathcal{S} \\ \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \infty & \forall i,j \in \mathcal{S}^c \end{cases}. \tag{132}$$

This implies that as $l \to \infty$ we have

$$\lim_{l\to\infty} \sum_{i,j\in\mathcal{S}} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \lim_{l\to\infty} \mathrm{Tr}(S_w) = 0. \tag{133}$$

$$\lim_{l\to\infty} \sum_{i,j\in\mathcal{S}^c} \frac{1}{2\sigma^2}(r_i - r_j)^T W W^T (r_i - r_j) = \lim_{l\to\infty} \mathrm{Tr}(S_b) = \infty, \tag{134}$$

This yields the ratio

$$\lim_{\mathcal{H}_l\to\mathcal{H}^*} \frac{\mathrm{Tr}(S_w)}{\mathrm{Tr}(S_b)} = \frac{0}{\infty} = 0. \tag{135}$$

□

Appendix D Proof for W_sOptimality

Given $\mathcal{H}_l$ as the empirical risk at layer $l \neq L$, we have

$$\frac{\partial}{\partial W_l}\mathcal{H}_l(W_s) \neq 0 \tag{136}$$

Proof. Given $\frac{1}{\sqrt{\zeta}}$ as a normalizing constant for $W_s = \frac{1}{\sqrt{\zeta}}\sum_\alpha r_\alpha$ such that $W^TW = I$. We start with the Lagrangian

$$\mathcal{L} = -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i-r_j)^T W W^T (r_i-r_j)}{2\sigma^2}} - \mathrm{Tr}(\Lambda(W^TW - I)). \tag{137}$$

If we now take the derivative with respect to the Lagrange, we get

$$\nabla\mathcal{L} = \frac{1}{\sigma^2}\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i-r_j)^T W W^T (r_i-r_j)}{2\sigma^2}} (r_i - r_j)(r_i - r_j)^T W - 2W\Lambda. \tag{138}$$

By setting the gradient to 0, we have

$$\left[\frac{1}{2\sigma^2}\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i-r_j)^T W W^T (r_i-r_j)}{2\sigma^2}} (r_i - r_j)(r_i - r_j)^T\right] W = W\Lambda. \tag{139}$$

$$\mathcal{Q}_l W = W\Lambda. \tag{140}$$

From Eq. (140), we see that W is only the optimal solution when W is the eigenvector of Q_l. Therefore, by setting W to $W_s = \frac{1}{\sqrt{\zeta}}\sum_\alpha r_\alpha$, it is not guaranteed to yield an optimal for all σ_l. □

Appendix E Proof for Corollary 1 and 2

Corollary 1: *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in IDS solves MSE via a translation of labels.*

Proof.

As $\mathcal{H}_l \to \mathcal{H}^*$, Thm. 2 shows that sample of the same class are mapped into the same point. Assuming that ϕ has mapped the sample into c points $\alpha = [\alpha_1, ..., \alpha_c]$ that's different from the truth label $\xi = [\xi_1, ..., \xi_c]$. Then the MSE objective is minimized by translating the ϕ output by

$$\xi - \alpha. \tag{141}$$

□

Corollary 2: *Given $\mathcal{H}_l \to \mathcal{H}^*$, the network output in RKHS solves CE via a change of bases.*

Assumptions, and Notations.

1. n is the number of samples.
2. τ is the number of classes.
3. $y_i \in \mathbb{R}^\tau$ is the ground truth label for the i^{th} sample. It is one-hot encoded where only the j^{th} element is 1 if x_i belongs to the j^{th} class, all other elements would be 0.
4. We denote ϕ as the network, and $\hat{y}_i \in \mathbb{R}^\tau$ as the network output where $\hat{y}_i = \phi(x_i)$. We also assume that $\hat{y}_i$ is constrained on a probability simplex where $1 = \hat{y}_i^T \mathbf{1}_n$.
5. We denote the j^{th} element of y_i, and $\hat{y}_i$ as $y_{i,j}$ and $\hat{y}_{i,j}$ respectively.
6. We define

   Orthogonality Condition: A set of samples $\{\hat{y}_1, ..., \hat{y}_n\}$ satisfies the orthogonality condition if

$$\begin{cases} \langle \hat{y}_i, \hat{y}_j \rangle = 1 & \forall \quad i,j \text{ same class} \\ \langle \hat{y}_i, \hat{y}_j \rangle = 0 & \forall \quad i,j \text{ not in the same class} \end{cases}. \tag{142}$$

7. We define the Cross-Entropy objective as

$$\arg\min_{\phi} - \sum_{i=1}^{n} \sum_{j=1}^{\tau} y_{i,j} \log(\phi(x_i)_{i,j}). \tag{143}$$

Proof.

From Thm. 2, we know that the network ϕ output, $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$, satisfy the orthogonality condition at $\mathcal{H}^*$. Then there exists a set of orthogonal bases represented by $\Xi = [\xi_1, \xi_2, ..., \xi_c]$ that maps $\{\hat{y}_1, \hat{y}_2, ..., \hat{y}_n\}$ to simulate the output of a softmax layer. Let $\xi_i = \hat{y}_j, j \in \mathcal{S}^i$, i.e., for the i_{th} class we arbitrary choose one of the samples from this class and assigns ξ_i of that class to be equal to the sample's output. Realize in our problem we have $< \hat{y}_i, \hat{y}_i >= 1$, so if $< \hat{y}_i, \hat{y}_j >= 1$, then subtracting these two would lead to $< \hat{y}_i, \hat{y}_i - \hat{y}_j >= 0$, which is the same as $\hat{y}_i = \hat{y}_j$. So this representation is well-defined and its independent of choices of the sample from each group if they satisfy orthogonality condition. Now we define transformed labels, Y as:

$$Y = \hat{Y}\Xi. \tag{144}$$

Note that $Y = [y_1, y_2, ..., y_n]^T$ which each y_i is a one hot vector representing the class membership of i sample in c classes. Since given Ξ as the change of basis, we can match $\hat{Y}$ to Y exactly, CE is minimized.

□

Appendix F Dataset Details

No samples were excludes from any of the dataset.

Wine. This dataset has 13 features, 178 samples, and 3 classes. The features are continuous and heavily unbalanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/wine`.

Divorce. This dataset has 54 features, 170 samples, and 2 classes. The features are discrete and balanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set`.

Car. This dataset has 6 features, 1728 samples and 2 classes. The features are discrete and balanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Car+Evaluation`.

Cancer. This dataset has 9 features, 683 samples, and 2 classes. The features are discrete and unbalanced in magnitude. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)`.

Face. This dataset consists of images of 20 people in various poses. The 624 images are vectorized into 960 features. The dataset can be downloaded at `https://archive.ics.uci.edu/ml/datasets/CMU+Face+Images`.

Random. This dataset has 2 features, 80 samples and 2 classes. It is generate with a gaussian distribution where half of the samples are randomly labeled as 1 or 0.

Adversarial. This dataset has 2 features, 80 samples and 2 classes. It is generate with the following code:

```
#!/usr/bin/env python

n = 40
X1 = np.random.rand(n,2)
X2 = X1 + 0.01*np.random.randn(n,2)

X = np.vstack((X1,X2))
Y = np.vstack(( np.zeros((n,1)), np.ones((n,1)) ))
```

CFAR10 Test. The test set images from CIFAR10 are preprocessed with a convolutional layer that outputs vectorized samples of $x_i \in \mathbb{R}^{10}$. This dataset has 10 features and 10,000 samples. The preprocessing code to map the images to $\mathbb{R}^{10}$ data is included in the supplementary. The link to download the data is at `https://www.cs.toronto.edu/~kriz/cifar.html`.

Raman. The dataset consists of 4306 samples, 700 frequencies, and 35 different cell types. Since this is proprietary data, a download link is not included.

Appendix G Optimal Gaussian σ for Maximum Kernel Separation

Although the Gaussian kernel is the most common kernel choice for kernel methods, its σ value is a hyperparameter that must be tuned for each dataset. This work proposes to set the σ value based on the maximum kernel separation. The source code is made publicly available on `https://github.com/anonamous`.

Let $X \in \mathbb{R}^{n\times d}$ be a dataset of n samples with d features and let $Y \in \mathbb{R}^{n\times\tau}$ be the corresponding one-hot encoded labels where τ denotes the number of classes. Given $\kappa_X(\cdot,\cdot)$ and $\kappa_Y(\cdot,\cdot)$ as two kernel functions that applies respectively to X and Y to construct kernel matrices $K_X \in \mathbb{R}^{n\times n}$ and $K_Y \in \mathbb{R}^{n\times n}$. Given a set $\mathcal{S}$, we denote $|\mathcal{S}|$ as the number of elements within the set. Also let $\mathcal{S}$ and $\mathcal{S}^c$ be sets of all pairs of samples of (x_i, x_j) from a dataset X that belongs to the same and different classes respectively, then the average kernel value for all (x_i, x_j) pairs with the same class is

$$d_{\mathcal{S}} = \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}} e^{-\frac{||x_i-x_j||^2}{2\sigma^2}} \tag{145}$$

and the average kernel value for all (x_i, x_j) pairs between different classes is

$$d_{\mathcal{S}^c} = \frac{1}{|\mathcal{S}^c|}\sum_{i,j\in\mathcal{S}^c} e^{-\frac{||x_i-x_j||^2}{2\sigma^2}}. \tag{146}$$

We propose to find the σ that maximizes the difference between $d_{\mathcal{S}}$ and $d_{\mathcal{S}^c}$ or

$$\max_{\sigma} \quad \frac{1}{|\mathcal{S}|}\sum_{i,j\in\mathcal{S}} e^{-\frac{||x_i-x_j||^2}{2\sigma^2}} - \frac{1}{|\mathcal{S}^c|}\sum_{i,j\in\mathcal{S}^c} e^{-\frac{||x_i-x_j||^2}{2\sigma^2}}. \tag{147}$$

It turns out that is expression can be computed efficiently. Let $g = \frac{1}{|\mathcal{S}|}$ and $\bar{g} = \frac{1}{|\mathcal{S}^c|}$, and let $\mathbf{1}_{n\times n} \in \mathbb{R}^{n\times n}$ be a matrix of 1s, then we can define Q as

$$Q = -gK_Y + \bar{g}(\mathbf{1}_{n\times n} - K_Y). \tag{148}$$

Or Q can be written more compactly as

$$Q = \bar{g}\mathbf{1}_{n\times n} - (g+\bar{g})K_Y. \tag{149}$$

Given Q, Eq. (147) becomes

$$\min_{\sigma} \quad \mathrm{Tr}(K_X Q). \tag{150}$$

This objective can be efficiently solved with BFGS.

Below in Fig. 6, we plot out the average within cluster kernel and the between cluster kernel values as we vary σ. From the plot, we can see that the maximum separation is discovered via BFGS.

Relation to HSIC. From Eq. (150), we can see that the σ that causes maximum kernel separation is directly related to HSIC. Given that the HSIC objective is normally written as

$$\min_{\sigma} \quad \mathrm{Tr}(K_X H K_Y H), \tag{151}$$

by setting $Q = HK_YH$, we can see how the two formulations are related. While the maximum kernel separation places the weight of each sample pair equally, HSIC weights the pair differently. We also notice that the $Q_{i,j}$ element is positive/negative for (x_i, x_j) pairs that are with/between classes respectively. Therefore, the argument for the global optimum should be relatively close for both objectives. Below in Figure 7, we show a figure of HSIC values as we vary σ. Notice how the optimal σ is almost equivalent to the solution from maximum kernel separation. For the purpose of *$\mathcal{H}$-Sequence*, we use σ that maximizes the HSIC value.

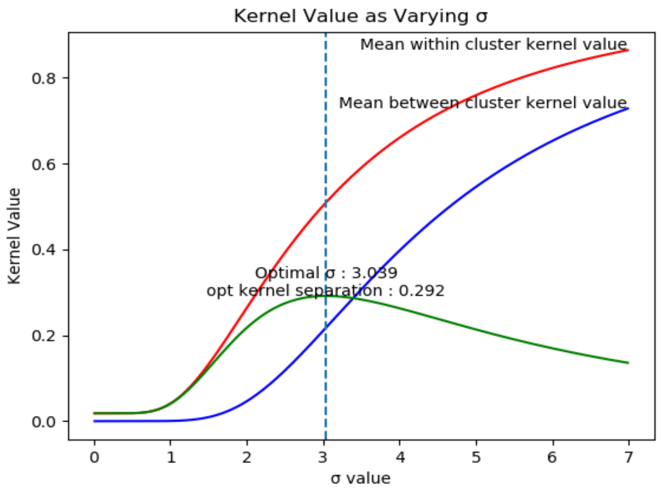

Figure 6: Maximum Kernel separation.

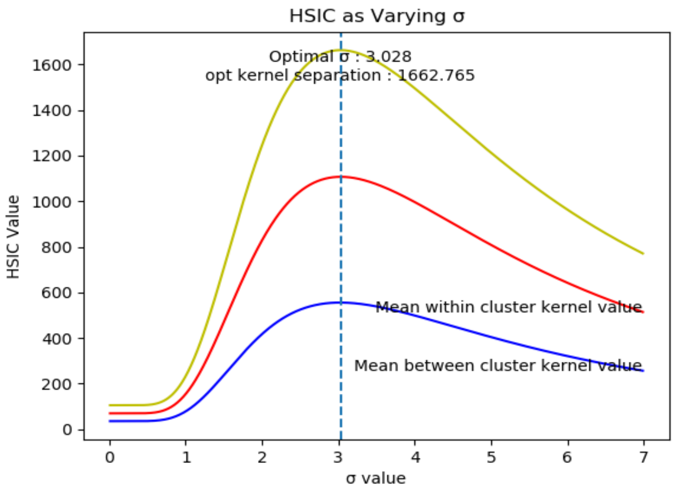

Figure 7: Maximal HSIC.

Appendix H W_l Dimensions for each 10 Fold of each Dataset

We report the input and output dimensions of each W_l for every layer of each dataset in the form of (α, β); the corresponding dimension becomes $W_l \in \mathbb{R}^{\alpha \times \beta}$. Since each dataset consists of 10-folds, the network structure for each fold is reported. We note that the input of the 1st layer is the dimension of the original data. However, after the first layer, the width of the RFF becomes the output of each layer; here we use 300.

The β value is chosen during the ISM algorithm. By keeping only the most dominant eigenvector of the Φ matrix, the output dimension of each layer corresponds with the rank of Φ. It can be seen from each dataset that the first layer significantly expands the rank. The expansion is generally followed by a compression of fewer and fewer eigenvalues. These results conform with the observations made by Montavon et al. (2011) and Ansuini et al. (2019).

Data	Layer 1	Layer 2	Layer 3	Layer 4
adversarial 1	(2, 2)	(300, 61)	(300, 35)	
adversarial 2	(2, 2)	(300, 61)	(300, 35)	
adversarial 3	(2, 2)	(300, 61)	(300, 8)	(300, 4)
adversarial 4	(2, 2)	(300, 61)	(300, 29)	
adversarial 5	(2, 2)	(300, 61)	(300, 29)	
adversarial 6	(2, 2)	(300, 61)	(300, 7)	(300, 4)
adversarial 7	(2, 2)	(300, 61)	(300, 34)	
adversarial 8	(2, 2)	(300, 12)	(300, 61)	(300, 30)
adversarial 9	(2, 2)	(300, 61)	(300, 33)	
adversarial 10	(2, 2)	(300, 61)	(300, 33)	

Data	Layer 1	Layer 2	Layer 3
Random 1	(3, 3)	(300, 47)	(300, 25)
Random 2	(3, 3)	(300, 46)	(300, 25)
Random 3	(3, 3)	(300, 46)	(300, 25)
Random 4	(3, 3)	(300, 47)	(300, 4)
Random 5	(3, 3)	(300, 47)	(300, 25)
Random 6	(3, 3)	(300, 45)	(300, 23)
Random 7	(3, 3)	(300, 45)	(300, 25)
Random 8	(3, 3)	(300, 45)	(300, 21)
Random 9	(3, 3)	(300, 45)	(300, 26)
Random 10	(3, 3)	(300, 47)	(300, 25)

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
spiral 1	(2, 2)	(300, 15)	(300, 6)	(300, 7)	(300, 6)	
spiral 2	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 3	(2, 2)	(300, 12)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 4	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 5	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	
spiral 6	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	
spiral 7	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	
spiral 8	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
spiral 9	(2, 2)	(300, 13)	(300, 6)	(300, 7)	(300, 6)	
spiral 10	(2, 2)	(300, 14)	(300, 6)	(300, 7)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
wine 1	(13, 11)	(300, 76)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 2	(13, 11)	(300, 76)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 3	(13, 11)	(300, 75)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 4	(13, 11)	(300, 76)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 5	(13, 11)	(300, 74)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 6	(13, 11)	(300, 74)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 7	(13, 11)	(300, 74)	(300, 6)	(300, 6)	(300, 6)	(300, 6)
wine 8	(13, 11)	(300, 75)	(300, 6)	(300, 7)	(300, 6)	(300, 6)
wine 9	(13, 11)	(300, 75)	(300, 6)	(300, 8)	(300, 6)	(300, 6)
wine 10	(13, 11)	(300, 76)	(300, 6)	(300, 7)	(300, 6)	(300, 6)

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6
car 1	(6, 6)	(300, 96)	(300, 6)	(300, 8)	(300, 6)	
car 2	(6, 6)	(300, 96)	(300, 6)	(300, 8)	(300, 6)	
car 3	(6, 6)	(300, 91)	(300, 6)	(300, 8)	(300, 6)	
car 4	(6, 6)	(300, 88)	(300, 6)	(300, 8)	(300, 6)	(300, 6)
car 5	(6, 6)	(300, 94)	(300, 6)	(300, 8)	(300, 6)	
car 6	(6, 6)	(300, 93)	(300, 6)	(300, 7)		
car 7	(6, 6)	(300, 92)	(300, 6)	(300, 8)	(300, 6)	
car 8	(6, 6)	(300, 95)	(300, 6)	(300, 7)	(300, 6)	
car 9	(6, 6)	(300, 96)	(300, 6)	(300, 9)	(300, 6)	
car 10	(6, 6)	(300, 99)	(300, 6)	(300, 8)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5
divorce 1	(54, 35)	(300, 44)	(300, 5)	(300, 5)	
divorce 2	(54, 35)	(300, 45)	(300, 4)	(300, 4)	
divorce 3	(54, 36)	(300, 49)	(300, 6)	(300, 6)	
divorce 4	(54, 36)	(300, 47)	(300, 7)	(300, 6)	
divorce 5	(54, 35)	(300, 45)	(300, 6)	(300, 6)	
divorce 6	(54, 36)	(300, 47)	(300, 6)	(300, 6)	
divorce 7	(54, 35)	(300, 45)	(300, 6)	(300, 6)	(300, 4)
divorce 8	(54, 36)	(300, 47)	(300, 6)	(300, 7)	(300, 4)
divorce 9	(54, 36)	(300, 47)	(300, 5)	(300, 5)	
divorce 10	(54, 36)	(300, 47)	(300, 6)	(300, 6)	

Data	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7	Layer 8	Layer 9	Layer 10
cancer 1	(9, 8)	(300, 90)	(300, 5)	(300, 6)	(300, 6)	(300, 5)	(300, 4)	(300, 5)	(300, 6)	(300, 6)
cancer 2	(9, 8)	(300, 90)	(300, 6)	(300, 7)	(300, 8)	(300, 11)	(300, 8)	(300, 4)		
cancer 3	(9, 8)	(300, 88)	(300, 5)	(300, 6)	(300, 7)	(300, 7)	(300, 6)	(300, 4)		
cancer 4	(9, 8)	(300, 93)	(300, 6)	(300, 7)	(300, 9)	(300, 11)	(300, 8)			
cancer 5	(9, 8)	(300, 93)	(300, 9)	(300, 10)	(300, 10)	(300, 11)	(300, 9)	(300, 7)		
cancer 6	(9, 8)	(300, 92)	(300, 7)	(300, 8)	(300, 8)	(300, 7)	(300, 7)			
cancer 7	(9, 8)	(300, 90)	(300, 4)	(300, 4)	(300, 5)	(300, 6)	(300, 6)	(300, 6)	(300, 6)	
cancer 8	(9, 8)	(300, 88)	(300, 5)	(300, 6)	(300, 7)	(300, 8)	(300, 7)	(300, 6)		
cancer 9	(9, 8)	(300, 88)	(300, 5)	(300, 7)	(300, 7)	(300, 7)	(300, 7)			
cancer 10	(9, 8)	(300, 97)	(300, 9)	(300, 11)	(300, 12)	(300, 13)	(300, 6)			

Data	Layer 1	Layer 2	Layer 3	Layer 4
face 1	(960, 233)	(300, 74)	(300, 73)	(300, 46)
face 2	(960, 231)	(300, 75)	(300, 73)	(300, 43)
face 3	(960, 231)	(300, 76)	(300, 73)	(300, 44)
face 4	(960, 232)	(300, 76)	(300, 74)	(300, 44)
face 5	(960, 231)	(300, 77)	(300, 73)	(300, 43)
face 6	(960, 232)	(300, 74)	(300, 72)	(300, 47)
face 7	(960, 232)	(300, 76)	(300, 73)	(300, 45)
face 8	(960, 230)	(300, 74)	(300, 74)	(300, 44)
face 9	(960, 233)	(300, 76)	(300, 76)	(300, 45)
face 10	(960, 231)	(300, 76)	(300, 70)	(300, 43)

Appendix I Sigma Values used for Random and Adversarial Simulation

The simulation of Thm. 1 as shown in Fig. 2 spread the improvement across multiple layers. The σ_l and $\mathcal{H}_l$ values are recorded here. We note that σ_l are reasonably large and not approaching 0 and the improvement of $\mathcal{H}_l$ is monotonic.

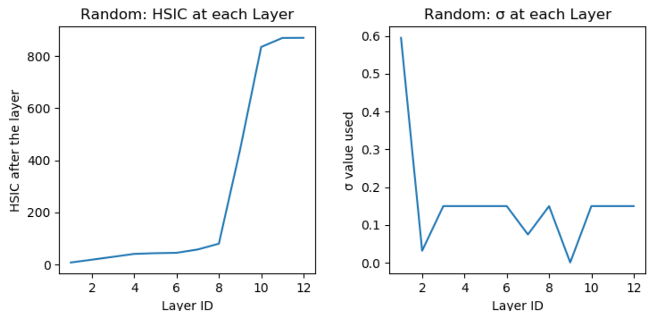

Figure 8

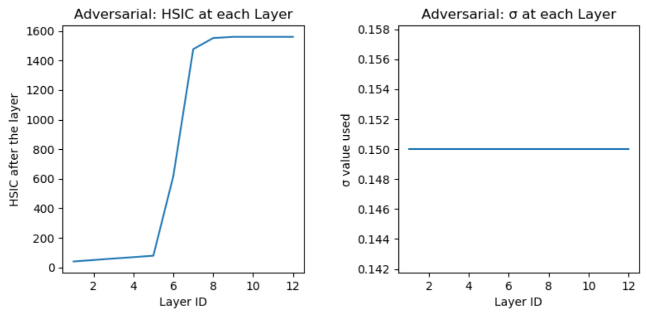

Figure 9

Given a sufficiently small σ_0 and σ_1, Thm. 1 claims that it can come arbitrarily close to the global optimal using a minimum of 2 layers. We here simulate 2 layers using a relatively small σ values ($\sigma_0 = 10^{-5}$) on the Random (left) and Adversarial (right) data and display the results of the 2 layers below. Notice that given 2 layer, it generated a clearly separable clusters that are pushed far apart.

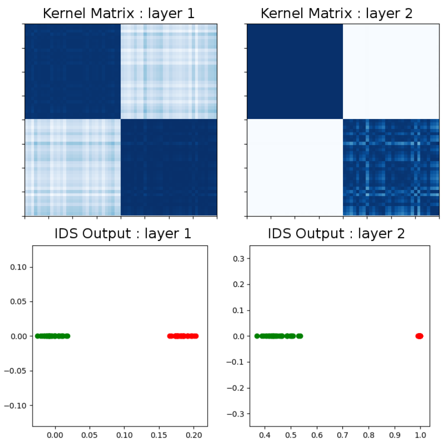

Figure 10: Random Dataset with 2 layers and $\sigma = 10^{-5}$

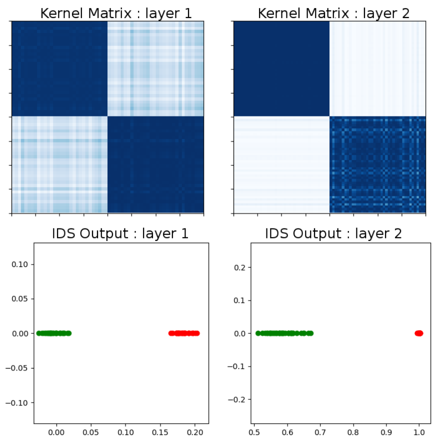

Figure 11: Adversarial Dataset with 2 layers and $\sigma = 10^{-5}$

Appendix J Evaluation Metrics Graphs

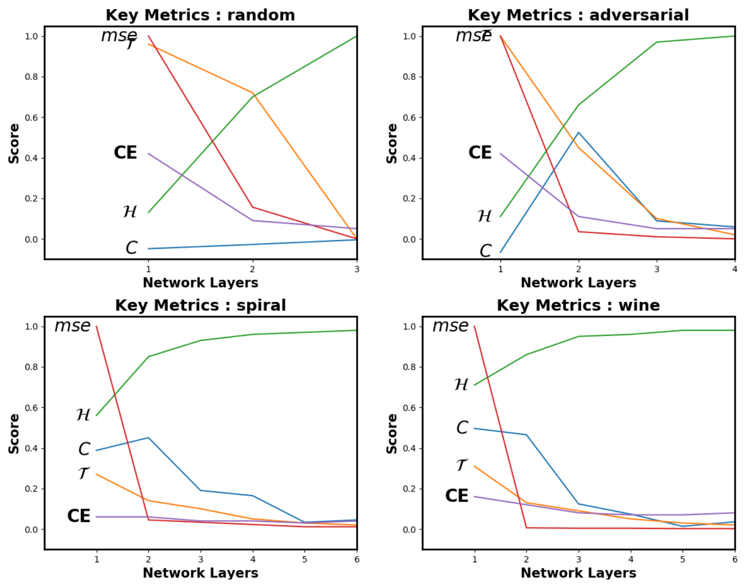

Figure 12

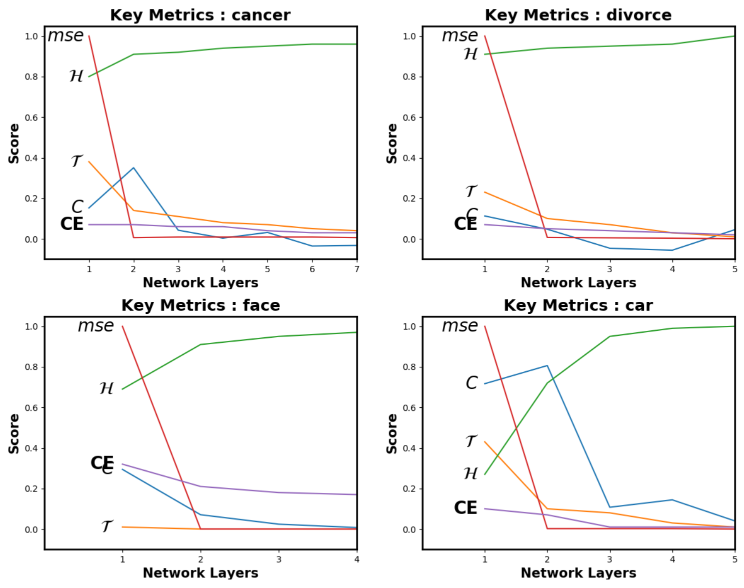

Figure 13: Figures of key metrics for all datasets as samples progress through the network. It is important to notice the uniformly and monotonically increasing $\mathcal{H}$-*Sequence* for each plot since this guarantees a converging kernel/risk sequence. As the $\mathcal{T}$ approach 0, samples of the same/difference classes in IDS are being pulled into a single point or pushed maximally apart respectively. As C approach 0, samples of the same/difference classes in RKHS are being pulled into 0 or $\frac{\pi}{2}$ cosine similarity respectively.

Appendix K Graphs of Kernel Sequences

A representation of the *Kernel Sequence* are displayed in the figures below for each dataset. The samples of the kernel matrix are previously organized to form a block structure by placing samples of the same class adjacent to each other. Since the Gaussian kernel is restricted to values between 0 and 1, we let white and dark blue be 0 and 1 respectively where the gradients reflect values in between. Our theorems predict that the *Kernel Sequence* will evolve from an uninformative kernel into a highly discriminating kernel of perfect block structures.

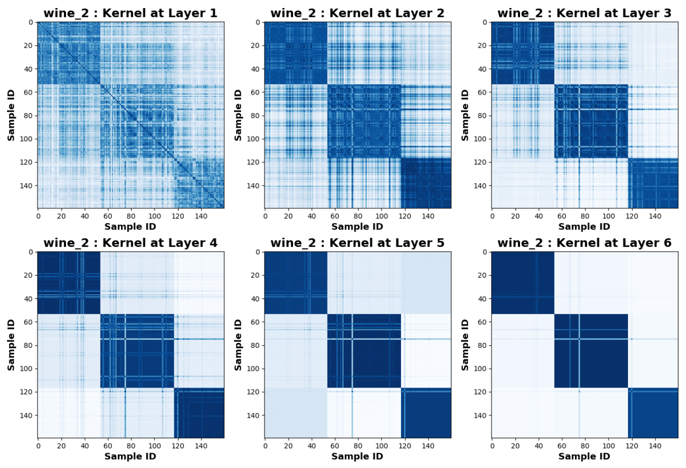

Figure 14: The kernel sequence for the wine dataset.

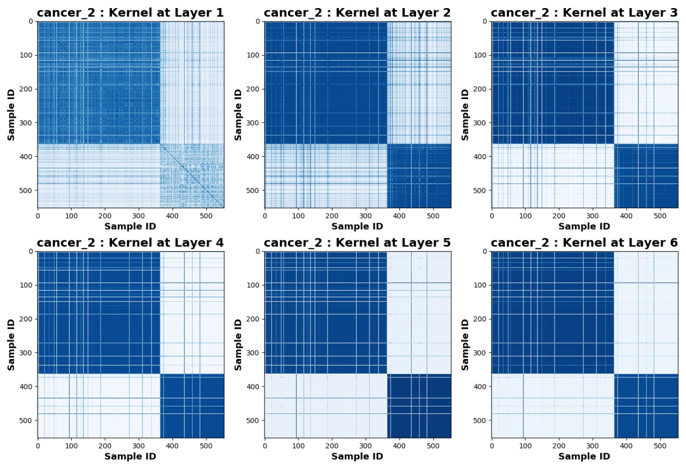

Figure 15: The kernel sequence for the cancer dataset.

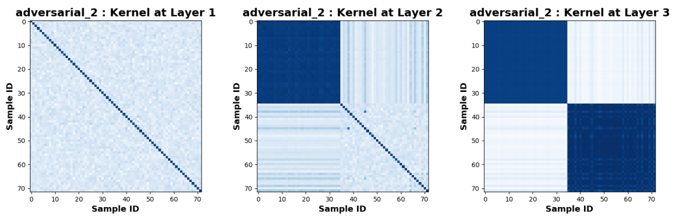

Figure 16: The kernel sequence for the Adversarial dataset.

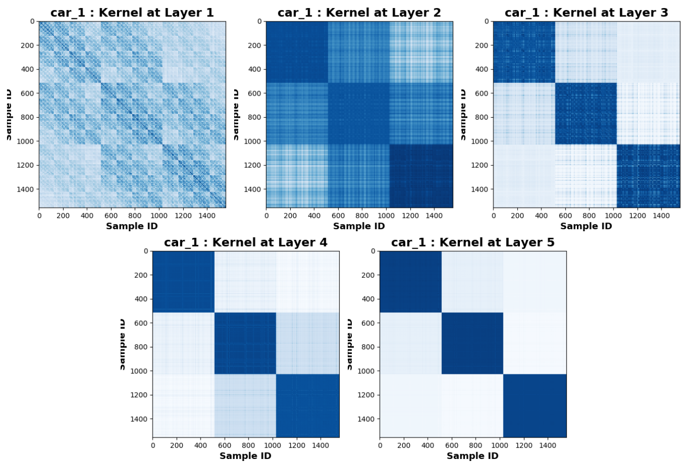

Figure 17: The kernel sequence for the car dataset.

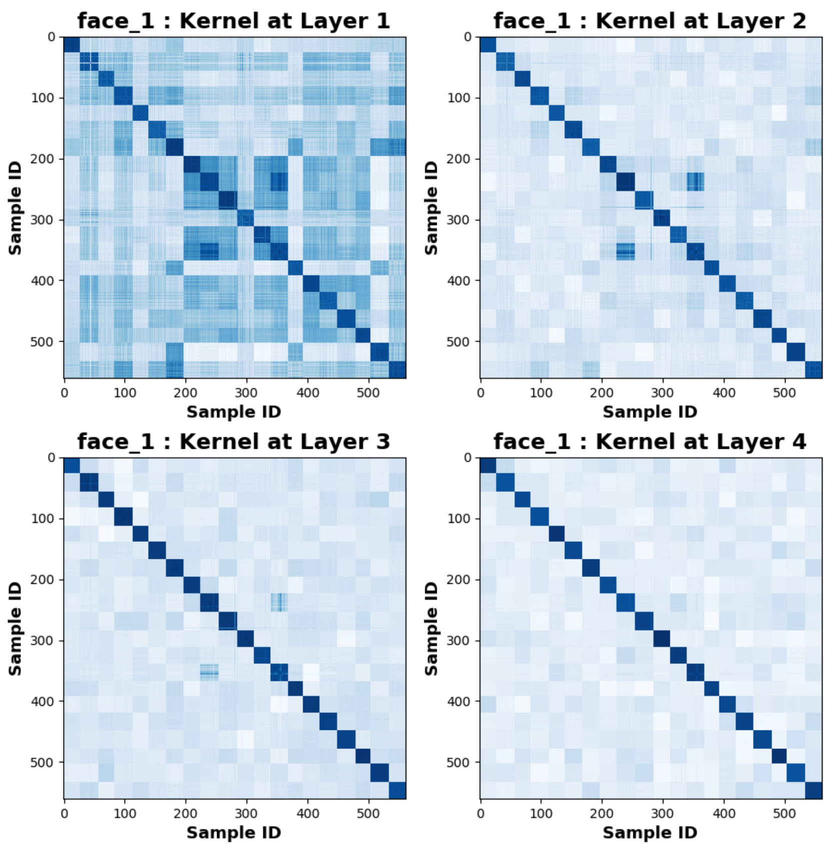

Figure 18: The kernel sequence for the face dataset.

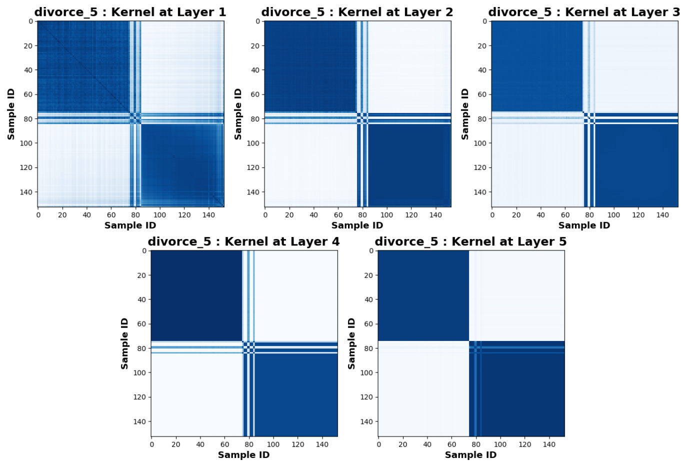

Figure 19: The kernel sequence for the divorce dataset.

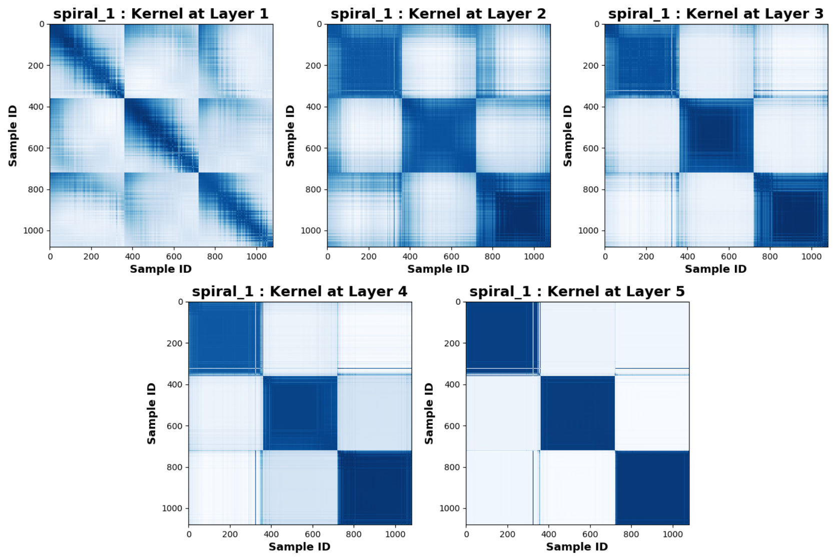

Figure 20: The kernel sequence for the spiral dataset.

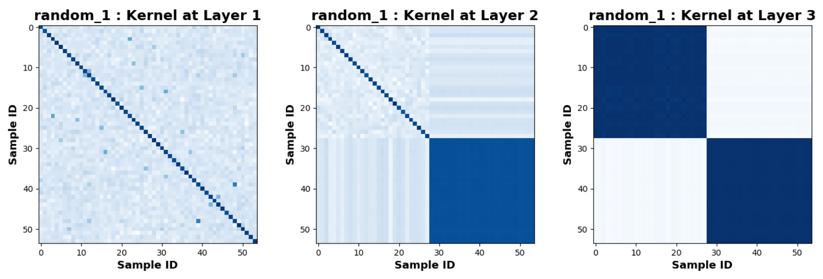

Figure 21: The kernel sequence for the Random dataset.

Appendix L On Generalization.

Besides being an optimum solution, W_l^* exhibits many advantages over W_s. For example, while W_s experimentally performs well, W^* converges with fewer layers and superior generalization. This raises a well-known question on generalization. It is known that overparameterized MLPs can generalize even without any explicit regularizer (Zhang et al., 2017). This observation contradicts classical learning theory and has been a longstanding puzzle (Cao and Gu, 2019; Brutzkus et al., 2017; Allen-Zhu et al., 2019). Therefore, by being overparameterized with an infinitely wide network, NIK's ability under HSIC to generalize raises similar questions. In both cases, W_s and W^*, the HSIC objective employs an infinitely wide network that should result in overfitting. We ask theoretically, under our framework, what makes HSIC and W^* special?

Recently, Poggio et al. (2020) have proposed that traditional MLPs generalize because gradient methods implicitly regularize the normalized weights given an exponential objective (like our HSIC). We discovered a similar impact the process of finding W^* has on HSIC, i.e., HSIC can be reformulated to isolate out n functions $[D_1(W_l), ..., D_n(W_l)]$ that act as a penalty term during optimization. Let $\mathcal{S}_i$ be the set of samples that belongs to the i_{th} class and let $\mathcal{S}_i^c$ be its complement, then each function $D_i(W_l)$ is defined as

$$D_i(W_l) = \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}_i} \Gamma_{i,j} \mathcal{K}_{W_l}(r_i, r_j) - \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}_i^c} |\Gamma_{i,j}| \mathcal{K}_{W_l}(r_i, r_j). \tag{152}$$

Notice that $D_i(W_l)$ is simply Eq. (4) for a single sample scaled by $\frac{1}{\sigma^2}$. Therefore, improving W_l also leads to an increase and decrease of $\mathcal{K}_{W_l}(r_i, r_j)$ associated with $\mathcal{S}_i$ and $\mathcal{S}_i^c$ in Eq. (14), thereby increasing the size of the penalty term $D_i(W_l)$. To appreciate how $D_i(W_l)$ penalizes $\mathcal{H}$, we propose an equivalent formulation in the theorem below with its derivation in App M.

Theorem 4. *Eq. (4) is equivalent to*

$$\max_{W_l} \sum_{i,j} \frac{\Gamma_{i,j}}{\sigma^2} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i^T W_l W_l^T r_j) - \sum_i D_i(W_l) ||W_l^T r_i||_2. \tag{153}$$

Based on Thm. 3, $D_i(W_l)$ adds a negative variable cost to the sample norm, $||W_l^T r_i||_2$, prescribing an implicit regularizer on HSIC. As W_l improve HSIC, it also imposes a heavier penalty on Eq. (15), severely constraining W_l.

Appendix M Proof for Theorem 3

Theorem 3: *Eq. (4) objective is equivalent to*

$$\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} (r_i^T W W^T r_j) - \sum_i D_i(W) ||W^T r_i||_2. \tag{154}$$

Proof. Let $A_{i,j} = (r_i - r_j)(r_i - r_j)^T$. Given the Lagranian of the HSIC objective as

$$\mathcal{L} = -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} - \text{Tr}[\Lambda(W^T W - I)]. \tag{155}$$

Our layer wise HSIC objective becomes

$$\min_W -\sum_{i,j} \Gamma_{i,j} e^{-\frac{(r_i - r_j)^T W W^T (r_i - r_j)}{2\sigma^2}} - \text{Tr}[\Lambda(W^T W - I)]. \tag{156}$$

We take the derivative of the Lagrangian, the expression becomes

$$\nabla_W \mathcal{L}(W, \Lambda) = \sum_{i,j} \frac{\Gamma_{i,j}}{\sigma^2} e^{-\frac{\text{Tr}(W^T A_{i,j} W)}{2\sigma^2}} A_{i,j} W - 2W\Lambda. \tag{157}$$

Setting the gradient to 0, and consolidate some scalar values into $\hat{\Gamma}_{i,j}$, we get the expression

$$\left[\sum_{i,j} \frac{\Gamma_{i,j}}{2\sigma^2} e^{-\frac{\text{Tr}(W^T A_{i,j} W)}{2\sigma^2}} A_{i,j}\right] W = W\Lambda \tag{158}$$

$$\left[\frac{1}{2}\sum_{i,j} \hat{\Gamma}_{i,j} A_{i,j}\right] W = W\Lambda \tag{159}$$

$$\mathcal{Q}W = W\Lambda. \tag{160}$$

From here, we see that the optimal solution is an eigenvector of $\mathcal{Q}$. Based on ISM, it further proved that the optimal solution is not just any eigenvector, but the eigenvectors associated with the smallest values of $\mathcal{Q}$. From this logic, ISM solves objective (156) with a surrogate objective

$$\min_W \quad \text{Tr}\left(W^T \left[\frac{1}{2}\sum_{i,j} \hat{\Gamma}_{i,j} A_{i,j}\right] W\right) \quad \text{s.t.}\, W^T W = I. \tag{161}$$

Given $D_{\hat{\Gamma}}$ as the degree matrix of $\hat{\Gamma}$ and $R = [r_1, r_2, ...]^T$, ISM further shows that Eq. (161) can be written into

$$\min_W \quad \text{Tr}\left(W^T R^T \left[D_{\hat{\Gamma}} - \hat{\Gamma}\right] RW\right) \quad \text{s.t.}\, W^T W = I \tag{162}$$

$$\max_W \quad \text{Tr}\left(W^T R^T \left[\hat{\Gamma} - D_{\hat{\Gamma}}\right] RW\right) \quad \text{s.t.}\, W^T W = I \tag{163}$$

$$\max_W \quad \text{Tr}\left(W^T R^T \hat{\Gamma} RW\right) - \text{Tr}\left(W^T R^T D_{\hat{\Gamma}} RW\right) \quad \text{s.t.}\, W^T W = I \tag{164}$$

$$\max_W \quad \text{Tr}\left(\hat{\Gamma} RWW^T R^T\right) - \text{Tr}\left(D_{\hat{\Gamma}} RWW^T R^T\right) \quad \text{s.t.}\, W^T W = I \tag{165}$$

$$\max_W \quad \sum_{i,j} \hat{\Gamma}_{i,j} [RWW^T R^T]_{i,j} - \sum_{i,j} D_{\hat{\Gamma}_{i,j}} [RWW^T R^T]_{i,j} \quad \text{s.t.}\, W^T W = I. \tag{166}$$

Since the jump from Eq. (161) can be intimidating for those not familiar with the literature, we included a more detailed derivation in App. N.

Note that the degree matrix $D_{\hat{\Gamma}}$ only have non-zero diagonal elements, all of its off diagonal are 0. Given $[RWW^T R^T]_{i,j} = (r_i^T WW^T r_j)$, the objective becomes

$$\max_W \quad \sum_{i,j} \hat{\Gamma}_{i,j} (r_i^T WW^T r_j) - \sum_i D_i(W) ||W^T r_i||_2 \quad \text{s.t.}\, W^T W = I. \tag{167}$$

Here, we treat D_i as a penalty weight on the norm of the $W^T r_i$ for every sample. □

To better understand the behavior of $D_i(W)$, note that $\hat{\Gamma}$ matrix looks like

$$\hat{\Gamma} = \frac{1}{\sigma^2}\begin{bmatrix} \left[\Gamma_{\mathcal{S}} e^{-\frac{(r_i - r_j)^T WW^T (r_i - r_j)}{2\sigma^2}}\right] & \left[-|\Gamma_{\mathcal{S}^c}| e^{-\frac{(r_i - r_j)^T WW^T (r_i - r_j)}{2\sigma^2}}\right] & ... \\ \left[-|\Gamma_{\mathcal{S}^c}| e^{-\frac{(r_i - r_j)^T WW^T (r_i - r_j)}{2\sigma^2}}\right] & \left[\Gamma_{\mathcal{S}} e^{-\frac{(r_i - r_j)^T WW^T (r_i - r_j)}{2\sigma^2}}\right] & ... \\ ... & ... & ... \end{bmatrix}. \tag{168}$$

The diagonal block matrix all $\Gamma_{i,j}$ elements that belong to $\mathcal{S}$ and the off diagonal are elements that belongs to $\mathcal{S}^c$. Each penalty term is the summation of its corresponding row. Hence, we can write out the penalty term as

$$D_i(W_l) = \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}|i} \Gamma_{i,j} \mathcal{K}_{W_l}(r_i, r_j) - \frac{1}{\sigma^2} \sum_{j \in \mathcal{S}^c|i} |\Gamma_{i,j}| \mathcal{K}_{W_l}(r_i, r_j). \tag{169}$$

From this, it shows that as W improve the objective, the penalty term is also increased. In fact, at its extreme as $\mathcal{H}_l \to \mathcal{H}^*$, all the negative terms are gone and all of its positive terms are maximized and this matrix approaches

$$\hat{\Gamma}^* = \frac{1}{\sigma^2}\begin{bmatrix} [\Gamma_{\mathcal{S}}] & [0] & ... \\ [0] & [\Gamma_{\mathcal{S}}] & ... \\ ... & ... & ... \end{bmatrix}. \tag{170}$$

From the matrix $\hat{\Gamma}^*$ and the definition of $D_i(W_l)$, we see that as $\mathcal{K}_W$ from $\mathcal{S}$ increase,
Since $D_i(W)$ is the degree matrix of $\hat{\Gamma}$, we see that as $\mathcal{H}_l \to \mathcal{H}^*$, we have

$$D_i^*(W) > D_i(W). \tag{171}$$

Appendix N Derivation for $\sum_{i,j} \Psi_{i,j}(x_i - x_j)(x_i - x_j)^T = 2X^T(D_\Psi - \Psi)X$

Since Ψ is a symmetric matrix, and $A_{i,j} = (x_i - x_j)(x_i - x_j)^T$, we can rewrite the expression into

$$\begin{array}{rcl} \sum_{i,j} \Psi_{i,j} A_{i,j} & = & \sum_{i,j} \Psi_{i,j}(x_i - x_j)(x_i - x_j)^T \\ & = & \sum_{i,j} \Psi_{i,j}(x_i x_i^T - x_j x_i^T - x_i x_j^T + x_j x_j^T) \\ & = & 2\sum_{i,j} \Psi_{i,j}(x_i x_i^T - x_j x_i^T) \\ & = & \left[2\sum_{i,j} \Psi_{i,j}(x_i x_i^T)\right] - \left[2\sum_{i,j} \Psi_{i,j}(x_i x_j^T)\right]. \end{array}$$

If we expand the 1st term, we get

$$2\sum_i^n \sum_j^n \Psi_{i,j}(x_i x_i^T) = 2\sum_i \Psi_{i,1}(x_i x_i^T) + \ldots + \Psi_{i,n}(x_i x_i^T) \tag{172}$$

$$= 2\sum_i^n [\Psi_{1,1} + \Psi_{1,2} + \ldots] x_i x_i^T \tag{173}$$

$$= 2\sum_i^n d_i x_i x_i^T \tag{174}$$

$$= 2X^T D_\Psi X \tag{175}$$

Given Ψ_i as the ith row, next we look at the 2nd term

$$2\sum_i \sum_j \Psi_{i,j} x_i x_j^T = 2\sum_i \Psi_{i,1} x_i x_1^T + \Psi_{i,2} x_i x_2^T + \Psi_{i,3} x_i x_3^T + \ldots \tag{176}$$

$$= 2\sum_i x_i(\Psi_{i,1} x_1^T) + x_i(\Psi_{i,2} x_2^T) + x_i(\Psi_{i,3} x_3^T) + \ldots \tag{177}$$

$$= 2\sum_i x_i \left[(\Psi_{i,1} x_1^T) + (\Psi_{i,2} x_2^T) + (\Psi_{i,3} x_3^T) + \ldots\right] \tag{178}$$

$$= 2\sum_i x_i \left[X^T \Psi_i^T\right]^T \tag{179}$$

$$= 2\sum_i x_i \left[\Psi_i X\right] \tag{180}$$

$$= 2\left[x_1 \Psi_1 X + x_2 \Psi_2 X + x_3 \Psi_3 X + \ldots\right] \tag{181}$$

$$= 2\left[x_1 \Psi_1 + x_2 \Psi_2 + x_3 \Psi_3 + \ldots\right] X \tag{182}$$

$$= 2X^T \Psi X \tag{183}$$

$$\tag{184}$$

Putting both terms together, we get

$$\sum_{i,j} \Psi_{i,j} A_{i,j} = 2X^T D_\Psi X - 2X^T \Psi X a \tag{185}$$

$$= 2X^T [D_\Psi - \Psi] X \tag{186}$$

$$\tag{187}$$