A New Neural Kernel Regime: The Inductive Bias of Multi-Task Learning

Julia Nakhleh
Department of Computer Science
University of Wisconsin-Madison
Madison, WI
jnakhleh@wisc.edu

Joseph Shenouda
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI
jshenouda@wisc.edu

Robert D. Nowak
Department of Electrical and Computer Engineering
University of Wisconsin-Madison
Madison, WI
rdnowak@wisc.edu

Abstract

This paper studies the properties of solutions to multi-task shallow ReLU neural network learning problems, wherein the network is trained to fit a dataset with minimal sum of squared weights. Remarkably, the solutions learned for each individual task resemble those obtained by solving a kernel method, revealing a novel connection between neural networks and kernel methods. It is known that single-task neural network training problems are equivalent to minimum norm interpolation problem in a non-Hilbertian Banach space, and that the solutions of such problems are generally non-unique. In contrast, we prove that the solutions to univariate-input, multi-task neural network interpolation problems are almost always unique, and coincide with the solution to a minimum-norm interpolation problem in a first-order Sobolev (reproducing kernel) Hilbert Space. We also demonstrate a similar phenomenon in the multivariate-input case; specifically, we show that neural network training problems with a large number of diverse tasks are approximately equivalent to an ℓ^2 (Hilbert space) minimization problem over a fixed kernel determined by the optimal neurons.

1 Introduction

This paper investigates the nature of functions learned by training neural networks on multiple "tasks" (i.e., multiple sets of labels on the same set of data points) and contrasts these solutions with those obtained by training separate networks for each individual task. We show that the solutions of multi-task neural network learning problems can differ dramatically from the solutions to single-task problems, even if the tasks are unrelated to one another. Unlike standard intuitions Caruana (1997) and existing theory Ben-David and Schuller (2003); Maurer et al. (2016) regarding the effects and benefits of multi-task learning, our results do not rely on similarity between tasks.

We focus on shallow, vector-valued (multi-output) neural networks with Rectified Linear Unit (ReLU) activation functions, which are functions $f_{\boldsymbol{\theta}} : \mathbb{R}^d \to \mathbb{R}^T$ of the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) \;=\; \sum_{k=1}^{K} \boldsymbol{v}_k \big(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\big)_+ \;+\; \boldsymbol{A}\boldsymbol{x} + \boldsymbol{c} \tag{1}$$

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

where $(\cdot)_+ = \max\{0, \cdot\}$ is the ReLU activation function and $\boldsymbol{w}_k \in \mathbb{R}^d$, $\boldsymbol{v}_k \in \mathbb{R}^T$, and $b_k \in \mathbb{R}$ are the input and output weights and bias of the k^{th} neuron. K is the number of neurons and T denotes the number of tasks (outputs) of the neural network. The affine term $\boldsymbol{Ax} + \boldsymbol{c}$ is the residual connection (or skip connection), where $\boldsymbol{A} \in \mathbb{R}^{T \times d}$ and $\boldsymbol{c} \in \mathbb{R}^T$. The set of all parameters is denoted by $\boldsymbol{\theta} := \big(\{\boldsymbol{v}_k, \boldsymbol{w}_k, b_k\}_{k=1}^K, \boldsymbol{A}, \boldsymbol{c}\big)$.

Neural networks are trained to fit data using gradient descent methods and often include a form of regularization called *weight decay*, which penalizes the ℓ^2 norm of the network weights. We consider weight decay applied only to the input and output weights of the neurons—no regularization is applied to the biases or residual connection. This is a common setting studied frequently in past work Savarese et al. (2019); Ongie et al. (2019); Parhi and Nowak (2021). Intuitively, only the input and output weights—not the biases or residual connection—affect the "regularity" of the neural network function as measured by its second (distributional) derivative, which is why it makes sense to regularize only these parameters. Given a set of training data points $(\boldsymbol{x}_1, \boldsymbol{y}_1), \ldots, (\boldsymbol{x}_N, \boldsymbol{y}_N) \in \mathbb{R}^d \times \mathbb{R}^T$ and a fixed width[1] $K \geq N^2$, we consider the weight decay interpolation problem:

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2^2 + \|\boldsymbol{w}_k\|_2^2 \text{ , subject to } f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \boldsymbol{y}_i,\ i = 1, \ldots, N \text{ .} \tag{2}$$

By homogeneity of the ReLU activation function (meaning that $(\alpha x)_+ = \alpha(x)_+$ for any $\alpha \geq 0$), the input and output weights of any ReLU neural network can be rescaled as $\boldsymbol{w}_k \mapsto \boldsymbol{w}_k / \|\boldsymbol{w}_k\|_2$ and $\boldsymbol{v}_k \mapsto \boldsymbol{v}_k \|\boldsymbol{w}_k\|_2$ without changing the function that the network represents. Using this fact, several previous works Grandvalet (1998); Grandvalet and Canu (1998); Neyshabur et al. (2015); Parhi and Nowak (2023); Shenouda et al. (2024) note that problem (2) is equivalent to

$$\min_{\boldsymbol{\theta}} \sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2, \text{ subject to } \{\|\boldsymbol{w}_k\|_2 = 1\}_{k=1}^K,\ f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) = \boldsymbol{y}_i,\ i = 1, \ldots, N \tag{3}$$

in that the minimal objective values of both training problems are the same, and any network $f_{\boldsymbol{\theta}}$ which solves (2) also solves (3), while any $f_{\boldsymbol{\theta}}$ which solves (3) also solves (2) after rescaling of the input and output weights. The regularizer $\sum_{k=1}^K \|\boldsymbol{v}_k\|_2$ is reminiscent of the multi-task lasso Obozinski et al. (2006). It has recently been shown to promote *neuron sharing* in the network, meaning that only a few neurons contribute to all tasks Shenouda et al. (2024).

The optimizations in (2) and (3) are non-convex and in general, they may have multiple global minimizers. As an example, consider the single-task, univariate dataset in Figure 1. For this dataset, (3) has infinitely many global solutions Savarese et al. (2019); Ergen and Pilanci (2021); Debarre et al. (2022); Hanin (2022). Two of the global minimizers are shown in Figure 1. In some scenarios, the solution on the right may be preferable to the one on the left, since the interpolation function stays closer to the training data points. Moreover, recent theoretical work shows that this solution could have other favorable generalization and robustness properties Joshi et al. (2024). Current training methods, however, might produce any one of the infinite number of solutions, depending on the random initialization of neural network weights as well as other possible sources of randomness in the training process. It is impossible to control this using existing training algorithms, which might explain many problems associated with current neural networks such as their sensitivity to adversarial attacks. In contrast, as we show in this paper, if we train a neural network to interpolate the data in Figure 1 along with additional interpolation tasks with different labels, then the solution is almost always unique and given by the (potentially preferable) interpolation depicted on the right. This demonstrates that multi-task learning can have a profound effect on the nature of neural network solutions.

The main contributions of our paper are:

Uniqueness of Multi-task Solutions. In the univariate setting ($d = 1$) we prove that the solution to multi-task training with different tasks is almost always a unique function, and we give a precise condition for the exceptional cases where solutions are non-unique.

Multi-task Training $\equiv$ Kernel Method (almost always). In the case that the solution to the univariate weight decay problem is unique, it is given by the connect-the-dots interpolant of the training

[1] By an argument similar to the proof of Theorem 5 of Shenouda et al. (2024), the optimal objective value of (3) (hence of (2)) for a given dataset is the same for any $K \geq N^2$.

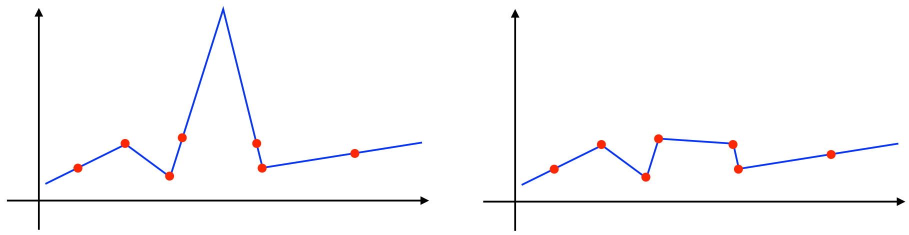


Figure 1: Two solutions to ReLU neural network interpolation (blue) of training data (red). The functions on the left and right both interpolate the data and both are global minimizers of (2) and (3), and minimize the second-order total variation of the interpolation function Parhi and Nowak (2021). In fact, all convex combinations of the two solutions above are also global solutions to both training problems.

data points: i.e., the optimal solution is a linear spline which performs straight-line interpolation between consecutive data points in all tasks. On the support of the data, this solution agrees with the solution to minimum-norm interpolation in a Sobolev reproducing kernel Hilbert space (RKHS) which contains all functions with first derivatives in L^2 De Boor and Lynch (1966). In contrast, solutions to the single-task training problem are non-unique in general and are given by minimum-norm interpolating functions in the non-Hilbertian Banach space BV^2 Parhi and Nowak (2021), which contains all functions with second distributional derivatives[2] in L^1. This shows that the solution to each individual task in a multi-task solution is equivalent to that of a kernel method, whereas single-task solutions generally are not.

Insights on Multivariate Multi-Task Problems. We provide empirical evidence and mathematical analysis which indicate that similar conclusions hold in multivariate settings. Specifically, with a large number of diverse tasks, the solutions for each individual task in multi-task learning are approximately minimum-norm solutions in a particular RKHS space determined by the optimal neurons. In contrast, learning each task in isolation results in solutions that are minimum-norm with respect to a non-Hilbertian Banach norm over the optimal neurons.

2 Related Works

Characterizations of ReLU neural network solutions: Hanin (2021); Stewart et al. (2023) characterized the neural network solutions to (2) in the univariate input/output setting. Boursier and Flammarion (2023) showed that in the univariate input/output case, when weight decay is modified to include the biases of each neuron the solution is unique. Moreover, under certain assumptions it is often the sparsest interpolator (in terms of the number of neurons). Our work differs from these as we study the multi-task setting, showing that the solutions are almost always unique and that they are the connect-the-dots solution, which is generally not the sparsest and corresponds to minimum norm interpolation over a particular RKHS space. While characterizing solutions to (2) in the multivariate setting is more challenging, there exist some results for very particular datasets Ergen and Pilanci (2021); Ardeshir et al. (2023); Zeno et al. (2024).

Function spaces associated with neural networks: For single-output ReLU neural networks, Savarese et al. (2019); Ongie et al. (2019) related weight decay regularization on the parameters of the model to regularizing a particular semi-norm on the neural network function. Ongie et al. (2019) showed that this semi-norm is not an RKHS semi-norm, highlighting a fundamental difference between neural networks and kernel methods. Parhi and Nowak (2021, 2022); Bartolucci et al. (2023); Unser (2021) studied the function spaces associated with this semi-norm, and developed representer theorems showing that optimal solutions to the minimum-norm data fitting problem over these spaces are realized by finite-width ReLU networks. Consequently, finite-width ReLU networks trained with weight decay are optimal solutions to the regularized data-fitting problem posed over these

[2]Technically, BV^2 contains all functions with second distributional derivatives in $\mathcal{M}$, the space of Radon measures with finite total variation. $\mathcal{M}$ can be viewed as a "generalization" of L^1 (see Ch. 7.3, p.223 in Folland (1999)).

spaces. Function spaces and representer theorems for multi-output and deep neural networks were later developed in Korolev (2022); Parhi and Nowak (2022); Shenouda et al. (2024).

Multi-Task Learning: The advantages of multi-task learning has been extensively studied in the machine learning literature Obozinski et al. (2006, 2010); Argyriou et al. (2006, 2008). Theoretical properties of multi-task neural networks in particular, have also been studied in Lindsey and Lippl (2023); Collins et al. (2024); Shenouda et al. (2024). The underlying intuition in these past works has been that learning multiple related tasks simultaneously can help select or learn the most useful features for all tasks. Our work differs from this traditional paradigm as we consider multi-task neural networks trained on very general tasks which may be diverse and unrelated.

3 Univariate Multi-Task Neural Network Solutions

For any function f that can be represented by a neural network (1) with width K, we define its representational cost to be

$$R(f) := \inf_{\boldsymbol{\theta}} \sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2 \quad \text{s.t.} \quad \|\boldsymbol{w}_k\|_2 = 1 \; \forall k, \quad f = f_{\boldsymbol{\theta}} \tag{4}$$

where $\boldsymbol{\theta} = \left(\{\boldsymbol{v}_k, \boldsymbol{w}_k, b_k\}_{k=1}^{K}, \boldsymbol{A}, \boldsymbol{c}\right)$. Taking an inf over all neural networks that realize f is necessary as there are multiple neural networks which can represent the same function. Solutions to (3) minimize this representational cost subject to the data interpolation constraint. This section gives a precise characterization of the solutions to the multi-task neural network interpolation problem in the univariate setting ($d = 1$).

For the training data points $(x_1, \boldsymbol{y}_1), \ldots, (x_N, \boldsymbol{y}_N) \in \mathbb{R} \times \mathbb{R}^T$, let y_{it} denote the t^{th} coordinate of the label vector $\boldsymbol{y}_i$. For each $t = 1, \ldots, T$, let $\mathcal{D}_t$ denote the univariate dataset $(x_1, y_{1t}), \ldots, (x_N, y_{Nt}) \in \mathbb{R} \times \mathbb{R}$, and let

$$s_{it} = \frac{y_{i+1,t} - y_{it}}{x_{i+1} - x_i} \tag{5}$$

denote the slope of the straight line between (x_i, y_{it}) and $(x_{i+1}, y_{i+1,t})$. The connect-the-dots interpolant of the dataset $\mathcal{D}_t$ is the function $f_{\mathcal{D}_t}$ which connects the consecutive points in dataset $\mathcal{D}_t$ with straight lines (see Figure 2). Its slopes on $(-\infty, x_2]$ and $[x_{N-1}, \infty)$ are s_{1t} and s_{N-1t}, respectively.

In the following section, we state a simple necessary and sufficient condition under which the connect-the-dots interpolant $f_{\mathcal{D}} = (f_{\mathcal{D}_1}, \ldots, f_{\mathcal{D}_T})$ is the *unique* optimal interpolant of the datasets $\mathcal{D}_1, \ldots, \mathcal{D}_T$. We also demonstrate that the set of multi-task datasets which satisfy the necessary condition for non-uniqueness, viewed as a subset of $\mathbb{R}^N \times \mathbb{R}^{T \times N}$, has Lebesgue measure zero. This result raises an interesting new connection between data fitting with ReLU neural networks and traditional kernel-based learning methods, because connect-the-dots interpolation is also the minimum-norm solution to a data-interpolation problem in the first-order Sobolev space $H^1([x_1, x_N])$, itself an RKHS whose norm penalizes the L^2 norm of the derivative of the function. In particular, $f_{\mathcal{D}_t}$ agrees on $[x_1, x_N]$ with the function $f(x) = \sum_{j=1}^{N} \alpha_j k(x, x_j)$ whose coefficients α_j solve the kernel optimization problem

$$\min_{\alpha_1, \ldots, \alpha_N \in \mathbb{R}} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j \, k(x_i, x_j) \;\; \text{s.t.} \;\; \sum_{j=1}^{N} \alpha_j \, k(x_i, x_j) = y_{it}, \quad i = 1, \ldots, N,$$

with the kernel $k(x, x') = 1 - (x - x')_+ + (x - x_1)_+ + (x_1 - x')_+$ De Boor and Lynch (1966). Therefore, our result shows that the individual outputs of solutions to (3) for $T > 1$ tasks almost always coincide on $[x_1, x_N]$ with this kernel solution; for example, this occurs with probability one if the task labels are sampled from an absolutely continuous distribution. In contrast, optimal solutions to the (3) in the case $T = 1$ are generally non-unique and may not coincide with the connect-the-dots kernel solution Hanin (2022). We note that for $T = 1$, our result is consistent with the characterization of univariate solutions to (3) in Hanin (2022).

Previous works Shenouda et al. (2024) and Lindsey and Lippl (2023) showed that multi-task training encourages *neuron sharing*, where all task are encouraged to utilize the same set of neurons or

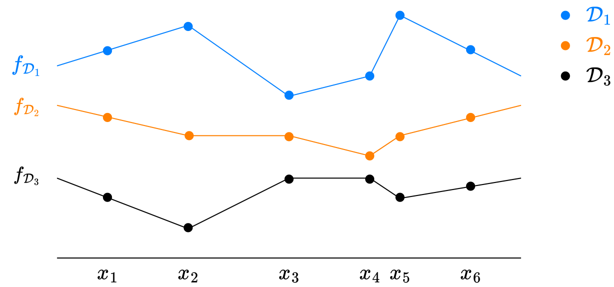


Figure 2: The connect-the-dots interpolant $f_{\mathcal{D}} = (f_{\mathcal{D}_1}, f_{\mathcal{D}_2}, f_{\mathcal{D}_3})$ of three datasets $\mathcal{D}_1, \mathcal{D}_2, \mathcal{D}_3$.

representations. Our result shows that univariate multi-task training is an extreme example of this phenomenon, since $f_{\mathcal{D}}$ can be represented using only $N-1$ neurons, all of which contribute to all of the network outputs. Therefore, in the scenario we study here, neuron sharing almost always occurs even if the tasks are unrelated.

3.1 Connect-the-dots Interpolation is Almost Always the Unique Solution to (3)

Our main result is stated in the following theorem:

Theorem 3.1. *The connect-the-dots function $f_{\mathcal{D}}$ is always a solution to* (3). *Moreover, the solution to problem* (3) *is non-unique if and only if the following condition is satisfied: for some $i = 2, \dots, N-2$, the two vectors*

$$\boldsymbol{s}_i - \boldsymbol{s}_{i-1} = \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i} - \frac{\boldsymbol{y}_i - \boldsymbol{y}_{i-1}}{x_i - x_{i-1}} \tag{6}$$

and

$$\boldsymbol{s}_{i+1} - \boldsymbol{s}_i = \frac{\boldsymbol{y}_{i+2} - \boldsymbol{y}_{i+1}}{x_{i+2} - x_{i+1}} - \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i} \tag{7}$$

are both nonzero and aligned[3] *If this condition not satisfied, then $f_{\mathcal{D}}$ is the unique solution to* (3). *Furthermore, as long as $T > 1$ and $N > 1$, the set of all possible data points $x_1, \dots, x_N \in \mathbb{R}$ and $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N \in \mathbb{R}^T$ which admit non-unique solutions has Lebesgue measure zero (as a subset of $\mathbb{R}^N \times \mathbb{R}^{T \times N}$).*

Remark 1. *The proof of Theorem 3.1, which relies mainly on Lemma 3.2 as we describe below, also characterizes solutions of the regularized loss problem*

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i) + \lambda \sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2 \quad \text{subject to} \quad |w_k| = 1, \ k = 1, \dots, K \tag{8}$$

for input dimension $d = 1$, any $\lambda > 0$, and any loss function $\mathcal{L}$ which is lower semicontinuous in its second argument. Specifically, any solution f of (8) *is linear between consecutive data points $[x_i, x_{i+1}]$ unless the vectors $\hat{\boldsymbol{s}}_i - \hat{\boldsymbol{s}}_{i-1}$ and $\hat{\boldsymbol{s}}_{i+1} - \hat{\boldsymbol{s}}_i$ are both nonzero and aligned, where $\hat{\boldsymbol{s}}_i := \frac{\hat{\boldsymbol{y}}_{i+1} - \hat{\boldsymbol{y}}_i}{x_{i+1} - x_i}$ and $\hat{\boldsymbol{y}}_i := f_{\boldsymbol{\theta}}(x_i)$.*

The full proof of Theorem 3.1 appears in Appendix 6.1. We outline the main ideas here. Our proof relies on the fact that ReLU neural networks are continuous piecewise linear (CPWL) functions, where the change in slope of the function at the k^{th} knot is equivalent to the magnitude of the k^{th} output weight vector. This fact allows us to prove the theorem using reasoning about the knots and slope changes of CPWL functions. The main ingredient of the proof is the following lemma:

Lemma 3.2. *Let $f : \mathbb{R} \to \mathbb{R}^T$ be a function whose t^{th} output f_t is a CPWL function which interpolates $\mathcal{D}_t$. Suppose that at some $\tilde{x} \in \mathbb{R}$ between consecutive data points, one or more of the outputs f_t has a knot. Let $\tilde{x}_1$ and $\tilde{x}_2$ be the x-coordinates of the closest knots before and after $\tilde{x}$, respectively.*

[3] Two vectors $\boldsymbol{u}_1$ and $\boldsymbol{u}_2$ are *aligned* if $\boldsymbol{u}_1^\top \boldsymbol{u}_2 = \|\boldsymbol{u}_1\| \|\boldsymbol{u}_2\|$.

Denote the slopes of f_t around this interval in terms of a_t, b_t, c_t, and δ_t as in Figure 3, and let $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \boldsymbol{\delta} \in \mathbb{R}^T$ be the vectors containing the respective values for each task.

Then removing the knot at $\tilde{x}$ and instead connecting $\tilde{x}_1$ and $\tilde{x}_2$ by a straight line does not increase $R(f)$. Furthermore, if $\boldsymbol{a} - \boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{c}$ are not *aligned, then doing so will strictly decrease $R(f)$.*

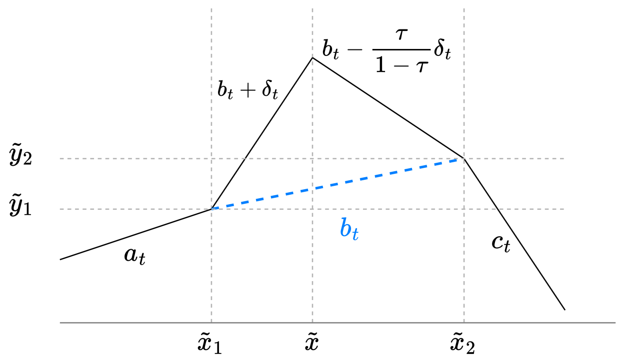


Figure 3: The function output f_t around the knot at $\tilde{x}$, where $\tau = \frac{\tilde{x}-\tilde{x}_1}{\tilde{x}_2-\tilde{x}_1}$. Each line segment in the figure is labeled with its slope. For any particular output t, it may be the case that f_t does not have a knot at $\tilde{x}$ (in which case $\delta_t = 0$); does not have a knot at $\tilde{x}_1$ (in which case $a_t = b_t + \delta_t$); and/or does not have a knot at $\tilde{x}_2$ (in which case $b_t - \frac{\tau}{1-\tau}\delta_t = c_t$).

Proof of Lemma 3.2. The representation cost (4) is separable across neurons. The contribution of these neurons to $R(f)$ is:

$$\begin{aligned}
& \|\boldsymbol{\delta} + \boldsymbol{b} - \boldsymbol{a}\|_2 \ + \ \frac{1}{1-\tau}\|\boldsymbol{\delta}\|_2 \ + \ \|\boldsymbol{c} - \boldsymbol{b} + \tau\boldsymbol{\delta}/(1-\tau)\|_2 \\
\geq \ & \|\boldsymbol{b} - \boldsymbol{a}\|_2 - \|\boldsymbol{\delta}\|_2 \ + \ \frac{1}{1-\tau}\|\boldsymbol{\delta}\|_2 \ + \ \|\boldsymbol{c} - \boldsymbol{b}\|_2 - \frac{\tau}{1-\tau}\|\boldsymbol{\delta}\|_2 \\
= \ & \|\boldsymbol{b} - \boldsymbol{a}\|_2 \ + \ \|\boldsymbol{c} - \boldsymbol{b}\|_2
\end{aligned} \tag{9}$$

where the inequality follows from the triangle inequality. This shows that taking $\delta_t = 0$ for all outputs, which corresponds to connecting $\tilde{x}_1$ and $\tilde{x}_2$ with a straight line in all outputs, will never increase the representational cost of f. The triangle inequality used in (9) holds with equality for some $\boldsymbol{\delta} \neq \boldsymbol{0}$ if and only if $\boldsymbol{a} - \boldsymbol{b}$, $\boldsymbol{b} - \boldsymbol{c}$, and $\boldsymbol{\delta}$ are aligned with $\|\boldsymbol{\delta}\|_2 \leq \min\{\|\boldsymbol{a} - \boldsymbol{b}\|_2, \frac{1-\tau}{\tau}\|\boldsymbol{b} - \boldsymbol{c}\|_2\}$. $\square$

Lemma 3.2 states that removing neurons which are located away from the data points and replacing them with a straight line will never increase the representational cost of the network, and it will strictly decrease the representational cost unless $\boldsymbol{a} - \boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{c}$ are aligned. This result implies that the connect-the-dots interpolant $f_{\mathcal{D}}$ is always a solution to (3), since we may take any solution f of (3) and remove all knots from it (resulting in the function $f_{\mathcal{D}}$) without increasing its representational cost. If $\boldsymbol{s}_i - \boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_{i+1} - \boldsymbol{s}_i$ are aligned for some $i = 2, \ldots, N-2$, we can view any interpolant on the interval $[x_i, x_{i+1}]$ as an instance of Figure 3 with $\boldsymbol{a} = \boldsymbol{s}_{i-1}$, $\boldsymbol{b} = \boldsymbol{s}_i$, and $\boldsymbol{c} = \boldsymbol{s}_{i+1}$. By Lemma 3.2, any neural network with a knot at some point $\tilde{x} \in (x_i, x_{i+1})$ can have the same representational cost as the connect-the-dots solution on this interval, only if $\boldsymbol{a} - \boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{c}$ are aligned.

We can also prove by contradiction that optimal solutions are unique on $[x_i, x_{i+1}]$ as long as $\boldsymbol{s}_i - \boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_{i+1} - \boldsymbol{s}_i$ are not aligned. Suppose that there is some other optimal interpolant f which is *not* the connect-the-dots solution $f_{\mathcal{D}}$ on an interval $[x_i, x_{i+1}]$ for which $\boldsymbol{s}_i - \boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_{i+1} - \boldsymbol{s}_i$ are not aligned. Then apply the lemma repeatedly to remove all knots from f which are not located at the data points, except for a single remaining knot at some $\tilde{x}$ between consecutive data points. If this knot occurs after x_2 (the second data point) or before x_{N-1} (second to last data point), the lemma implies automatically (again taking $\boldsymbol{a} = \boldsymbol{s}_{i-1}$, $\boldsymbol{b} = \boldsymbol{s}_i$, and $\boldsymbol{s}_{i+1}$) that removing this knot would strictly decrease the representational cost of the function, contradicting optimality of f. To conclude the proof, it remains only to show that any optimal interpolant of the dataset must agree with the connect-the-dots interpolant $f_{\mathcal{D}}$ before x_2 and after x_{N-1}; the details of this argument appear in

Appendix 6.1. To summarize, the connect-the-dots interpolation is the unique solution to (3) except in the special case where $\boldsymbol{s}_i - \boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_{i+1} - \boldsymbol{s}_i$ are aligned for one or more $i = 2, \dots, N-2$. As our theorem and corollary quantify, real-world regression datasets (which are typically real-valued and often assumed to incorporate some random noise from an absolutely continuous distribution, e.g. Gaussian) are extremely unlikely to satisfy this special alignment condition; hence, our claim that connect-the-dots interpolation is almost always the unique solution to (3). We provide numerical experiments in Appendix 7.1 to illustrate the variety of solutions obtained from single-task versus multi-task learning when training with gradient descent. The single-task problem can produce a variety of solutions, in contrast the multi-task learning problem almost always produces a unique solution.

4 Multivariate Multi-Task Neural Network Training

In Section 3, we proved that the univariate functions learned by neural networks trained on multiple tasks can be profoundly different from the functions learned by networks trained for each task separately. In this section, we demonstrate similar behavior in multivariate settings. Just like in the univariate case, it is known that multivariate single-task training problems are equivalent to optimization over a non-Hilbertian Banach spaces and that the solutions may be non-unique Parhi and Nowak (2021) . Indeed, in Figure 4 we show that multivariate training problems can be non-unique in terms of their spatial orientation. The following mathematical analysis and empirical evidence shows that, similar to our results in the univariate setting, multivariate multi-task learning can produce solutions that are strikingly different compared to the corresponding single-task training solutions. Moreover, as with univariate multi-task solutions, multivariate multi-task solutions can be related to functions belonging to an RKHS.

Here we analyze neural networks of the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \sum_{k=1}^{K} \boldsymbol{v}_k \left(\boldsymbol{w}_k^\top \boldsymbol{x} + b_k\right)_+ \tag{10}$$

where $\boldsymbol{w}_k \in \mathbb{S}^{d-1}$, $b_k \in \mathbb{R}$, $\boldsymbol{v}_k \in \mathbb{R}^T$, and $\boldsymbol{\theta} := \{\boldsymbol{v}_k, \boldsymbol{w}_k, b_k\}_{k=1}^K$. Since the analysis in this section is not dependent on the residual connection, we omit it for ease of exposition. We consider the multivariate-input, T-task neural network training problem

$$\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} \mathcal{L}\left(\boldsymbol{y}_i, f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)\right) + \lambda \sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2 \tag{11}$$

for some dataset $(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_N, \boldsymbol{y}_N) \in \mathbb{R}^d \times \mathbb{R}^T$, where $\mathcal{L}$ is any convex loss function which is lower semicontinuous in its second argument and separable across the T tasks.

To characterize the nature of the functions learned through multi-task training, consider the optimization from the perspective of a single task. The optimal output weights for task s are the solution to

$$J(v_{1s}, \dots, v_{Ks}) := \sum_{i=1}^{N} \mathcal{L}\left(y_{is}, \sum_{k=1}^{K} v_{ks} \boldsymbol{\Phi}_{ik}\right) + \lambda \sum_{k=1}^{K} \left\| \begin{bmatrix} v_{ks} \\ \boldsymbol{v}_{k\setminus s}^* \end{bmatrix} \right\|_2 \tag{12}$$

where $\boldsymbol{v}_{k\setminus s}^*$ denotes the vector $\boldsymbol{v}_k^*$ with its s^{th} element v_{ks}^* excluded and $\boldsymbol{\Phi} \in \mathbb{R}^{N\times K}$ is a matrix whose i, k^{th} entry is $\boldsymbol{\Phi}_{ik} = \left(\boldsymbol{x}_i^\top \boldsymbol{w}_k^* + b_k^*\right)_+$. J is the objective function of (11) with all parameters except for $v_{1s}, \dots, v_{Ks}$ held fixed at their optimal values. Note that the optimal values $v_{1s}^*, \dots, v_{Ks}^*$ for (11) also minimize J; otherwise they would not be optimal for (11).

We are interested in analyzing the behavior of solutions to (12) as the number of tasks T grows. Intuitively, if T is large, then the magnitude of the output weight v_{ks}^* for an individual neuron k and task s would be relatively small compared to $\|\boldsymbol{v}_{k\setminus s}^*\|_2 := \sqrt{\sum_{t\neq s}(v_{ks}^*)^2}$. Then by a Taylor series approximation, the k^{th} term of the regularizer in (11) would be approximately equal to

$$\sqrt{(v_{ks})^2 + \|\boldsymbol{v}_{k\setminus s}^*\|_2^2} \;\approx\; \|\boldsymbol{v}_{k\setminus s}^*\|_2 + \frac{(v_{ks})^2}{2\|\boldsymbol{v}_{k\setminus s}^*\|_2}\,. \tag{13}$$

Notice that the approximation is a quadratic function of v^*_{ks}, which suggests that the regularization term of (11) resembles a weighted ℓ^2 regularizer when v_{ks} is close to its optimal value v^*_{ks}.

The reasoning above can be made precise by assuming that that the vectors $\boldsymbol{y}_{:1}, \ldots, \boldsymbol{y}_{:T}$, where $\boldsymbol{y}_{:t} = [y_{1t}, \ldots, y_{Nt}]$ is the vector of labels corresponding to task t, are statistically independent conditioned on the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. Intuitively, independence of the task labels means that among many has little effect on the output weights associated with other tasks. This is observed experimentally, as shown in Figure 4. As an example of a random process generating conditionally independent task labels, consider the following: let μ be a probability measure over a family of conditional probability distributions $\mathcal{P}$. Suppose that $p_1, \ldots, p_T$ are independently and identically distributed according to μ and $\boldsymbol{y}_{:t}|\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \sim p_t$, where $\boldsymbol{y}_{:t} := [y_{1t}, \ldots, y_{Nt}]$ is the vector of labels corresponding to tasks $t = 1, \ldots, T$. Then $\boldsymbol{y}_{:1}, \ldots, \boldsymbol{y}_{:T}$ are independent, conditioned on the data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$.

When $\boldsymbol{y}_{:1}, \ldots, \boldsymbol{y}_{:T}$ are conditionally independent given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$, the optimal output weights for each task in (11) are identically distributed (however, due to the neuron sharing enforced by the $\sum_{k=1}^{K} \|\boldsymbol{v}_2\|_2$ regularization term, they are not independent). Two key facts follow from this observation. As the number of tasks T grows:

1. The magnitude of v^*_{ks} is dominated by $\|\boldsymbol{v}^*_{k\setminus s}\|_2$, which implies that the remainder in the quadratic Taylor series approximation tends to zero.
2. $\|\boldsymbol{v}^*_{k\setminus s}\|_2$ concentrates around the norm of the full vector of output weights $\|\boldsymbol{v}^*_k\|_2$, which means that the Taylor approximation tends to the same quadratic function for all tasks.

A more detailed analysis is given in the companion paper Nakhleh et al. (2024); here we informally state a theorem summarizing the main conclusion.

Theorem 4.1. *Nakhleh et al. (2024) Suppose that $\boldsymbol{y}_{:1}, \ldots, \boldsymbol{y}_{:T}$ are conditionally independent given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$. For an individual task s, consider the objective*

$$H(v_{1s}, \ldots, v_{Ks}) := \sum_{i=1}^{N} \mathcal{L}\left(y_{is}, \sum_{k} v_{ks} \boldsymbol{\Phi}_{ik}\right) + \lambda \sum_{k} \left(\|\boldsymbol{v}^*_{k\setminus s}\|_2 + \frac{v_{ks}^2}{2\|\boldsymbol{v}^*_{k\setminus s}\|_2}\right) \tag{14}$$

Then as T grows, the global minimizer $v'_{1s}, \ldots, v'_{Ks}$ of H satisfies

$$J(v'_{1s}, \ldots, v'_{Ks}) - J(v^*_{1s}, \ldots, v^*_{Ks}) \ \to\ 0\ . \tag{15}$$

with probability approaching one.

The theorem shows that the solution to the weighted ℓ^2 minimization problem

$$\min_{v_{1s}, \ldots, v_{Ks}} \sum_{i=1}^{N} \mathcal{L}\left(y_{is}, \sum_{k \in \mathcal{S}} v_{ks} \boldsymbol{\Phi}_{ik}\right) + \frac{\lambda}{2} \sum_{k \in \mathcal{S}} \gamma_{ks} v_{ks}^2 \tag{16}$$

where $\gamma_{ks} := 1/\|\boldsymbol{v}^*_{k/s}\|_2$ is approximately optimal for the original objective (12), with stronger approximation as T increases. In contrast, in the single-task setting ($T = 1$), the optimization

$$\min_{v_1, \ldots, v_K} \sum_{i=1}^{N} \mathcal{L}\left(y_i, \sum_{k=1}^{K} v_k \boldsymbol{\Psi}_{ik}\right) + \lambda \sum_{k=1}^{K} |v_k|. \tag{17}$$

yields output weights which are exactly optimal for (11). Note that the matrices $\boldsymbol{\Phi}$ in 16 and $\boldsymbol{\Psi}$ in 17 are not the same, since they are determined by the optimal input weights and biases for (11), which are themselves data- and task-dependent. Nonetheless, comparing (16) and (17) highlights the different nature of solutions learned for (11) in the single-task versus multi-task case. The multi-task learning problem with independent tasks favors linear combinations of the optimal neurons which have a minimal weighted ℓ^2 regularization penalty. In contrast, the single-task learning problem favors linear combinations of optimal neurons which have a minimal ℓ^1 penalty. Therefore, multi-task learning with a large number of diverse tasks promotes a fundamentally different linear combination of the optimal features learned in the hidden layer.

To gain further insight, note that because $\|\boldsymbol{v}^*_{k\setminus s}\|_2$ concentrates around the norm of the full vector of output weights $\|\boldsymbol{v}^*_k\|_2$ as T grows

$$\gamma_{ks} \;\approx\; \gamma_k \;:=\; \frac{1}{\|\boldsymbol{v}^*_k\|_2}\,. \tag{18}$$

This reveals a novel connection between the problem of minimizing (12) and a norm-regularized data fitting problem in an RKHS. Specifically, consider the finite-dimensional linear space

$$\mathcal{H} := \left\{ f_{\boldsymbol{v}} = \sum_{k=1}^{K} v_k \phi_k : \boldsymbol{v} \in \mathbb{R}^K \right\} \tag{19}$$

where $\phi_k(\boldsymbol{x}) = \left(\boldsymbol{x}^\top \boldsymbol{w}^*_k + b^*_k\right)_+$, equipped with the inner product

$$\langle f_{\boldsymbol{v}}, f_{\boldsymbol{u}} \rangle_{\mathcal{H}} = \boldsymbol{v}^T \boldsymbol{Q} \boldsymbol{u} \tag{20}$$

where $\boldsymbol{Q} = \operatorname{diag}\left(\frac{\gamma_1}{2}, \ldots, \frac{\gamma_K}{2}\right)$. As a finite-dimensional inner product space, $\mathcal{H}$ is necessarily a Hilbert space; furthermore, finite-dimensionality of $\mathcal{H}$ implies that all linear functionals (including the point evaluation functional) on $\mathcal{H}$ are continuous. Therefore, $\mathcal{H}$ is an RKHS, with reproducing kernel

$$\kappa(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{\phi}(\boldsymbol{x}') \boldsymbol{Q}^{-1}\, \boldsymbol{\phi}(\boldsymbol{x}), \tag{21}$$

where $\boldsymbol{\phi}(\boldsymbol{x})$ is a vector containing the outputs of all K neurons (i.e., $\phi_1(\boldsymbol{x}), \ldots, \phi_K(\boldsymbol{x})$). Note that κ indeed satisfies the reproducing property: that is for any $f(\cdot) = \boldsymbol{v}^\top \boldsymbol{\phi}(\cdot) \in \mathcal{H}$, we have

$$f(\boldsymbol{x}) \;=\; \langle \kappa(\cdot, \boldsymbol{x}), f(\cdot) \rangle_{\mathcal{H}} = \left\langle \boldsymbol{\phi}(\boldsymbol{x})^\top \boldsymbol{Q}^{-1} \boldsymbol{\phi}(\cdot), \boldsymbol{v}^\top \boldsymbol{\phi}(\cdot) \right\rangle_{\mathcal{H}} \tag{22}$$

This shows that (16) is well approximated by solving

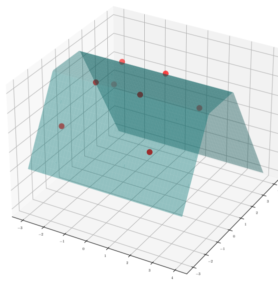
$$\arg\min_{f \in \mathcal{H}} \sum_{i=1}^{N} \mathcal{L}(y_{is}, f(\boldsymbol{x}_i)) + \lambda \|f\|^2_{\mathcal{H}}. \tag{23}$$

We provide empirical evidence for the claims presented in this section in Figure 4 on a simple multi-variate dataset. First, we demonstrate the variety of solutions that interpolate this dataset in a single task setting. In contrast, we show that the solutions obtained via multi-task learning with additional random tasks are very similar and often much smoother than those obtained by single-task learning supporting our claim that these solutions are well approximated by a kernel method. We also verify that solutions to (23) is a good approximation for solutions which minimize (12).

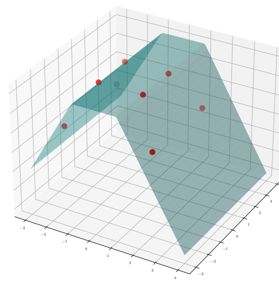
5 Conclusion and Discussion

We have shown that univariate, multi-task shallow ReLU neural networks which are trained to interpolate a dataset with minimal sum of squared weights almost always represent a unique function. This function performs straight-line interpolation between consecutive data points for each task. This solution is also the solution to a min-norm data-fitting problem in an RKHS. We provide mathematical analysis and numerical evidence suggesting that a similar conclusion may hold in the mulvariate-input case, as long as the tasks are sufficiently large in number and "diverse." These results indicate that multi-task training of neural networks can produce solutions that are strikingly different from those obtained by single-task training, and highlights a novel connection between these multi-task solutions and kernel methods.

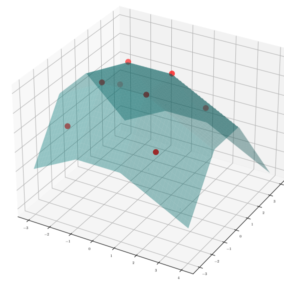
Here we study shallow ReLU networks due to their relative simplicity and amenability to mathematical analysis. Future work could aim to extend these results to other activation functions and deep network architectures. We also focus here on characterizing global solutions to the optimizations in (2) and (3). Whether or not networks trained with gradient descent-based algorithms will converge to global solutions remains an open question: our low-dimensional numerical experiments in Sections 3 and 4 indicate that they do, but a more rigorous analysis of the training dynamics would be an interesting separate line of research. Finally, while our analysis and experiments in Section 4 indicate that multivariate, multitask neural network solutions behave similarly to ℓ^2 regression over a fixed kernel, we have not precisely characterized what that kernel is in the multi-input case as we have in the single-input case: developing such a characterization is of interest for future work.



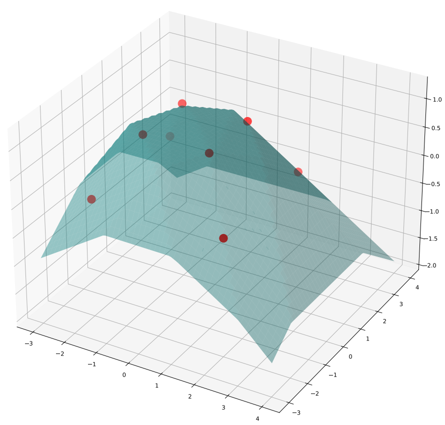
(a) Solution to (11) with $T = 1$ (single-task training)



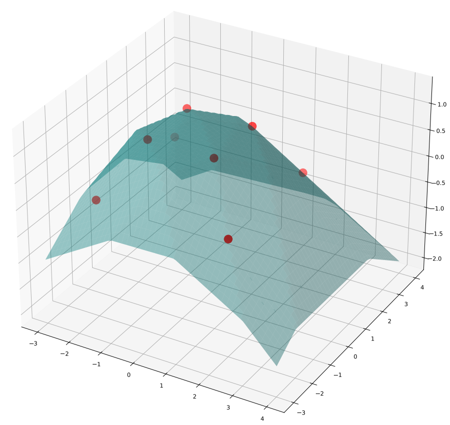
(b) Solution to (11) with $T = 1$ (single-task training)



(c) Solution to (11) with $T = 1$ (single-task training)



(d) Solution to (11) with $T = 101$ (multi-task training)



(e) Solution to (23) (RKHS approximation)

Figure 4: ReLU network interpolation in two-dimensions. The solutions shown were obtained with regularization parameter $\lambda \approx 0$. *Top Row – Solutions to single-task training*: Figures 4a, 4b and 4c show solutions to ReLU neural network interpolation (blue surface) of training data (red). The eight data points are located at the vertices of two squares, both centered at the origin. The outer square has side-length two and values of 0 at the vertices. The inner square has side-length one and values of 1 at the vertices. All three functions interpolate the data and are global minimizers of (2) and (3) when solving for just this task (i.e., $T = 1$). Due to the simplicity of this dataset the optimality of the solutions in the first row were confirmed by solving the equivalent convex optimization to (2) developed in Ergen and Pilanci (2021). *Bottom Row – Solutions to multi-task training:* Figure 4d shows the solution to the first output of a multi-task neural network with $T = 101$ tasks. The first output is the original task depicted in the first row while the labels for other 100 tasks are randomly generated i.i.d from a Bernoulli distribution with equal probability for one and zero. Here we show one representative example; more examples are depicted in Appendix 7 showing that this phenomenon holds across many runs. Figure 4e shows the solution to fitting the training data by solving (23) over a fixed set of features learned by the multi-task neural network with $T = 100$ random tasks. We observe that unlike the highly variable solutions of single-task optimization problem, the solutions obtained by solving the multi-task optimizations are nearly identical, as one would have for kernel methods. Moreover, the solution obtained by solving (23) is also similar to the solution of the full multi-task training problem with all $T = 101$ tasks.

References

N. Ardeshir, D. J. Hsu, and C. H. Sanford. Intrinsic dimensionality and generalization properties of the r-norm inductive bias. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 3264–3303. PMLR, 2023.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems*, 19, 2006.

A. Argyriou, T. Evgeniou, and M. Pontil. Convex multi-task feature learning. *Machine Learning*, 73: 243–272, 2008.

F. Bartolucci, E. De Vito, L. Rosasco, and S. Vigogna. Understanding neural networks with reproducing kernel banach spaces. *Applied and Computational Harmonic Analysis*, 62:194–236, 2023.

S. Ben-David and R. Schuller. Exploiting task relatedness for multiple task learning. In *Learning Theory and Kernel Machines: 16th Annual Conference on Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003. Proceedings*, pages 567–580. Springer, 2003.

E. Boursier and N. Flammarion. Penalising the biases in norm regularisation enforces sparsity. *Advances in Neural Information Processing Systems*, 36:57795–57824, 2023.

R. Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

L. Collins, H. Hassani, M. Soltanolkotabi, A. Mokhtari, and S. Shakkottai. Provable multi-task representation learning by two-layer relu neural networks. In *International Conference on Machine Learning*, 2024.

C. De Boor and R. E. Lynch. On splines and their minimum properties. *Journal of Mathematics and Mechanics*, 15(6):953–969, 1966.

T. Debarre, Q. Denoyelle, M. Unser, and J. Fageot. Sparsest piecewise-linear regression of one-dimensional data. *Journal of Computational and Applied Mathematics*, 406:114044, 2022.

S. Diamond and S. Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(1):2909–2913, 2016.

T. Ergen and M. Pilanci. Convex geometry and duality of over-parameterized neural networks. *Journal of Machine Learning Research*, 22(212):1–63, 2021.

G. B. Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

Y. Grandvalet. Least absolute shrinkage is equivalent to quadratic penalization. In *ICANN 98: Proceedings of the 8th International Conference on Artificial Neural Networks, Skövde, Sweden, 2–4 September 1998 8*, pages 201–206. Springer, 1998.

Y. Grandvalet and S. Canu. Outcomes of the equivalence of adaptive ridge with least absolute shrinkage. *Advances in Neural Information Processing Systems*, 11, 1998.

B. Hanin. Ridgeless interpolation with shallow relu networks in $1d$ is nearest neighbor curvature extrapolation and provably generalizes on lipschitz functions. *arXiv preprint arXiv:2109.12960*, 2021.

B. Hanin. On the implicit bias of weight decay in shallow univariate relu networks. 2022.

N. Joshi, G. Vardi, and N. Srebro. Noisy interpolation learning with shallow univariate relu networks. In *International Conference on Learning Representations*, 2024.

Y. Korolev. Two-layer neural networks with values in a Banach space. *SIAM Journal on Mathematical Analysis*, 54(6):6358–6389, 2022.

J. W. Lindsey and S. Lippl. Implicit regularization of multi-task learning and finetuning in overparameterized neural networks. *arXiv preprint arXiv:2310.02396*, 2023.

A. Maurer, M. Pontil, and B. Romera-Paredes. The benefit of multitask representation learning. *Journal of Machine Learning Research*, 17(81):1–32, 2016.

J. Nakhleh, J. Shenouda, and R. D. Nowak. The effects of multi-task learning on relu neural network functions. *arXiv preprint arXiv:2410.21696*, 2024.

B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *International Conference on Learning Representations (Workshop)*, 2015.

G. Obozinski, B. Taskar, and M. Jordan. Multi-task feature selection. *Statistics Department, UC Berkeley, Tech. Rep*, 2(2.2):2, 2006.

G. Obozinski, B. Taskar, and M. I. Jordan. Joint covariate selection and joint subspace selection for multiple classification problems. *Statistics and Computing*, 20:231–252, 2010.

G. Ongie, R. Willett, D. Soudry, and N. Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. In *International Conference on Learning Representations*, 2019.

R. Parhi and R. D. Nowak. Banach space representer theorems for neural networks and ridge splines. *Journal of Machine Learning Research*, 22(43):1–40, 2021.

R. Parhi and R. D. Nowak. What kinds of functions do deep neural networks learn? insights from variational spline theory. *SIAM Journal on Mathematics of Data Science*, 4(2):464–489, 2022.

R. Parhi and R. D. Nowak. Deep learning meets sparse regularization: A signal processing perspective. *IEEE Signal Processing Magazine*, 40(6):63–74, 2023.

P. Savarese, I. Evron, D. Soudry, and N. Srebro. How do infinite width bounded norm networks look in function space? In *Conference on Learning Theory*, pages 2667–2690. PMLR, 2019.

J. Shenouda, R. Parhi, K. Lee, and R. D. Nowak. Variation spaces for multi-output neural networks: Insights on multi-task learning and network compression. *Journal of Machine Learning Research*, 25(231):1–40, 2024.

L. Stewart, F. Bach, Q. Berthet, and J.-P. Vert. Regression as classification: Influence of task formulation on neural network features. In *International Conference on Artificial Intelligence and Statistics*, pages 11563–11582. PMLR, 2023.

M. Unser. A unifying representer theorem for inverse problems and machine learning. *Foundations of Computational Mathematics*, 21(4):941–960, 2021.

C. Zeno, G. Ongie, Y. Blumenfeld, N. Weinberger, and D. Soudry. How do minimum-norm shallow denoisers look in function space? *Advances in Neural Information Processing Systems*, 36, 2024.

6 Appendix

6.1 Proof of Theorem 3.1

Proof. We break the proof into the following sections.

Unregularized residual connection. First, we will briefly discuss the utility of the unregularized residual connection in our mathematical analysis. Consider a single input/output function f that is realizable by a ReLU neural network $f_{\boldsymbol{\theta}} : \mathbb{R} \to \mathbb{R}$ of the form (1) with unit norm input weights. Suppose that $f_{\boldsymbol{\theta}}$ contains two neurons

$$\eta_1(x) = v_1(w_1 x + b_1)_+, \quad \eta_2(x) = v_2(w_2 x + b_2)_+$$

which "activate" at the same location, that is, $b_1/w_1 = b_2/w_2$. If $\operatorname{sgn}(w_1) = \operatorname{sgn}(w_2)$, the neurons η_1 and η_2 activate in the same direction (i.e. both "turn on" or "turn off" at the point $b_1/w_1 = b_2/w_2$) and thus they can be combined into a single neuron η. The resulting network will represent the same function as $f_{\boldsymbol{\theta}}$ with a representational cost that is no greater than that of $f_{\boldsymbol{\theta}}$. Now suppose instead that the neurons activate at the same location, i.e. $b_1/w_1 = b_2/w_2$, but that (without loss of generality) $w_1 = 1$ and $w_2 = -1$. In this case, the sum of these two neurons η_1 and η_2 can we rewritten as

$$\begin{aligned} v_1(w_1 x + b_1)_+ + v_2(w_2 x + b_2)_+ &= v_1(x + b_1)_+ + v_2(-x - b_1)_+ \\ &= (v_1 + v_2)(x + b_1)_+ - (v_1 + v_2)(x + b_1) \end{aligned} \tag{24}$$

which follows from the fact that for any x we have $x = (x)_+ - (-x)_+$. Since the $(v_1 + v_2)(x + b_1)$ term can be incorporated into the unregularized residual connection, the resulting network again represents the same function with representational cost less than or equal to that of the original network.

The above reasoning shows that, when discussing optimal neural networks which solve (3), we may limit ourselves to considering neural networks for which no two neurons activate at the same location. Any such network $f_{\boldsymbol{\theta}}$ with K neurons represents T univariate continuous piecewise linear (CPWL) functions $f_1, \ldots, f_T$ with a combined total of K knots, where each knot k is located at x-coordinate b_k/w_k, and the change in slope of the function f_t at that point is given by $w_k v_{kt}$ (recall that $w_k = \pm 1$). Conversely, any set of T CPWL $\mathbb{R} \to \mathbb{R}$ functions with a combined total of K knots at locations $x_1, \ldots, x_K \in \mathbb{R}$, where the slope changes of the t^{th} function are denoted $\mu_{1t}, \ldots, \mu_{Kt}$ (some of which may be zero), is represented by a $\mathbb{R} \to \mathbb{R}^T$ network of width K whose parameters satisfy $b_k/w_k = x_k$ and $\mu_{kt} = w_k v_{kt}$ (hence $|\mu_{kt}| = |v_{kt}|$) for each $k = 1, \ldots, K$ and each $t = 1, \ldots, T$. This identification allows us to prove the theorem statement entirely using reasoning about CPWL functions: in the remainder of the proof, we use the terms "knot" and "neuron" interchangeably, and we use $|v_{kt}|$ to denote both the absolute slope change of output f_t at knot k and, equivalently, the t^{th} entry of the output weight vector $\boldsymbol{v}_k$.

Connect-the-dots interpolation is always a solution to (3). Using Lemma 3.2, we proceed to prove Theorem 3.1. Let $S^*_{\boldsymbol{\theta}}$ denote the set of parameters of optimal neural networks which solve (3) for the given data points, and let

$$S^* := \{f : \mathbb{R} \to \mathbb{R}^T \mid f(x) = f_{\boldsymbol{\theta}}(x) \ \forall x \in \mathbb{R}, \ \boldsymbol{\theta} \in S^*_{\boldsymbol{\theta}}\} \tag{25}$$

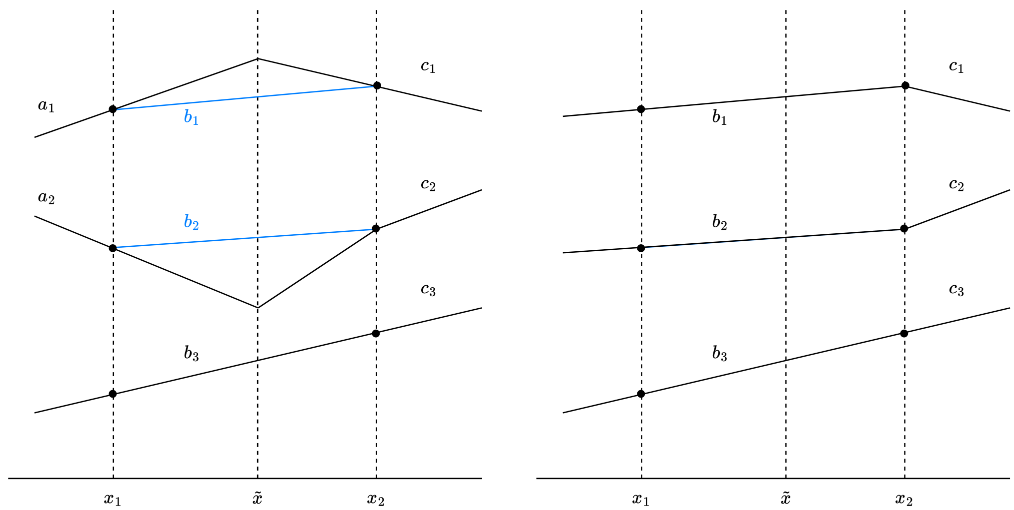
be the set of functions represented by neural network with optimal parameters in $S^*_{\boldsymbol{\theta}}$. First, note that the connect-the-dots interpolant $f_{\mathcal{D}}$ is in the solution set S^*. To see this, fix any $f \in S^*$, and apply Lemma 3.2 repeatedly to remove all "extraneous" knots (i.e., knots located away from the data points $x_1, \ldots, x_N$) from f. By Lemma 3.2, the resulting function—which is simply $f_{\mathcal{D}}$—has representational cost no greater than the original f, and since f had optimal representational cost, so does $f_{\mathcal{D}}$.

Conditions for non-unique solutions. We first prove the conditions for which (3) admits an infinite number of solutions. For some $i = 2, \ldots, N-2$, consider the two vectors

$$\boldsymbol{s}_i - \boldsymbol{s}_{i-1} = \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i} - \frac{\boldsymbol{y}_i - \boldsymbol{y}_{i-1}}{x_i - x_{i-1}} \tag{26}$$

and

$$\boldsymbol{s}_{i+1} - \boldsymbol{s}_i = \frac{\boldsymbol{y}_{i+2} - \boldsymbol{y}_{i+1}}{x_{i+2} - x_{i+1}} - \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i}. \tag{27}$$



(a) A function g with a knot at $\tilde{x} \in (x_1, x_2)$.

(b) The function $f_{\mathcal{D}}$.

Figure 5: Left: a function g which has a knot in one or more of its outputs at a point $\tilde{x} \in (x_1, x_2)$. Right: the connect-the-dots interpolant $f_{\mathcal{D}}$. The representational cost of g is strictly greater than that of $f_{\mathcal{D}}$.

and assume they are aligned. Then we may view the function $f_{\mathcal{D}}$ around the interval $[x_i, x_{i+1}]$ as an instance of Lemma 3.2 with $\boldsymbol{a} = \boldsymbol{s}_{i-1}$, $\boldsymbol{b} = \boldsymbol{s}_i$, and $\boldsymbol{c} = \boldsymbol{s}_{i+1}$. Fix some point $\tilde{x} \in (x_i, x_{i+1})$ and denote

$$\tau = \frac{\tilde{x} - x_i}{x_{i+1} - x_i}.$$

Let $\boldsymbol{\delta}$ be any vector which is aligned with $\boldsymbol{a} - \boldsymbol{b}$ and $\frac{1-\tau}{\tau}(\boldsymbol{b} - \boldsymbol{c})$ and has smaller norm than both, and let $f : \mathbb{R} \to \mathbb{R}^T$ be the function whose output slopes on $(x_i, \tilde{x})$ are given by $\boldsymbol{\delta}$ and whose slopes on $(\tilde{x}, x_{i+1})$ are given by $\boldsymbol{b} - \frac{\tau}{1-\tau}\boldsymbol{\delta}$. Then by Lemma 3.2, $R(f) = R(f_{\mathcal{D}})$ and thus $f \in S^*$. Since there are infinitely many $\boldsymbol{\delta}$'s which satisfy the condition above to choose the solution to (3) is non-unique in this case with infinitely many optimal solutions.

Necessary and sufficient condition under which $f_{\mathcal{D}}$ is the unique solution. For the other direction of the proof, suppose that for any $i = 1, \ldots, N-1$, the vectors $\boldsymbol{s}_i - \boldsymbol{s}_{i-1}$ and $\boldsymbol{s}_{i+1} - \boldsymbol{s}_i$ are *not* aligned, and assume by contradiction that there is some $f \in S^*$ which is *not* of the form $f_{\mathcal{D}}$. This f must not have any knots on $(-\infty, x_1]$ or $[x_N, \infty)$, since removing such a knot would strictly decrease $R(f)$ without affecting the ability of f to interpolate the data, contradicting optimality of f. So it must be the case that f has an extraneous knot at some $\tilde{x}$ which lies between consecutive data points x_i and x_{i+1}. Let g denote the function obtained by removing all extraneous knots from f *except* the one located at $\tilde{x}$. By Lemma 3.2, $R(g) \leq R(f)$.

Now, suppose the extraneous knot is between $[x_i, x_{i+1}]$ for $i = 2, \ldots, N-2$. Since g has no extraneous knots away from $\tilde{x}$, it must be the case that g agrees with $f_{\mathcal{D}}$ on $[x_{i-1}, x_i]$ and $[x_{i+1}, x_{i+2}]$. We may view the behavior of g around the interval $[x_i, x_{i+1}]$ as an instance of Lemma 3.2 with $\boldsymbol{a} = \boldsymbol{s}_{i-1}$, $\boldsymbol{b} = \boldsymbol{s}_i$, and $\boldsymbol{c} = \boldsymbol{s}_{i+1}$. By assumption, $\boldsymbol{a} - \boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{c}$ are *not* aligned, so by Lemma 3.2, removing the knot at $\tilde{x}$ would strictly reduce $R(g)$. This contradicts optimality of g, hence of f.

Finally, consider the case where the extraneous knot is on the interval $[x_i, x_{i+1}]$ where $i = 1$ (the case $i = N-1$ follows by an analogous argument). Let $\boldsymbol{a}$ denote the vector of incoming slopes of g at x_1. Define

$$\boldsymbol{b} = \frac{\boldsymbol{y}_2 - \boldsymbol{y}_1}{x_2 - x_1} \quad \boldsymbol{c} = \frac{\boldsymbol{y}_3 - \boldsymbol{y}_2}{x_3 - x_2}.$$

Since g has no extraneous knots except for $\tilde{x}$, the slopes of g coming out of x_2 are $\boldsymbol{c}$. By optimality of g, it must be the case that $\boldsymbol{a} - \boldsymbol{b}$ and $\boldsymbol{b} - \boldsymbol{c}$ are aligned (otherwise we could invoke Lemma 3.2 and strictly reduce the representational cost of f by removing the knot at $\tilde{x}$, a contradiction), which implies that $\operatorname{sgn}(a_t - b_t) = \operatorname{sgn}(b_t - c_t)$ for each $t = 1, \ldots, T$. For any outputs t which have a knot at $\tilde{x}$, this quantity is nonzero, in which case $|c_t - b_t| < |c_t - a_t|$ (see Figure 5a). Let $1, \dots, t_0$ be the indices of the outputs which have a knot at $\tilde{x}$, and let $t_0 + 1, \dots, T$ be the indices of the outputs which do *not* have a knot at $\tilde{x}$. We may again invoke Lemma 3.2 to remove the knots from g, resulting in a new function $\tilde{g}$ (satisfying $R(g) \geq R(\tilde{g})$) which has slopes $\boldsymbol{a}$ coming into x_1, $\boldsymbol{b}$ between x_1 and x_2, and $\boldsymbol{c}$ coming out of x_2. The contribution of these knots to $R(\tilde{g})$ is then given by:

$$\begin{aligned}\left\| \begin{bmatrix} b_1 - a_1 \\ \vdots \\ b_{t_0} - a_{t_0} \end{bmatrix} \right\|_2 + \left\| \begin{bmatrix} c_1 - b_1 \\ \vdots \\ c_{t_0} - b_{t_0} \end{bmatrix} \right\|_2 &\geq \left\| \begin{bmatrix} b_1 - a_1 + c_1 - b_1 \\ \vdots \\ b_{t_0} - a_{t_0} + c_{t_0} - b_{t_0} \end{bmatrix} \right\|_2 \\ &= \left\| \begin{bmatrix} c_1 - a_1 \\ \vdots \\ c_{t_0} - a_{t_0} \end{bmatrix} \right\|_2 \\ &> \left\| \begin{bmatrix} c_1 - b_1 \\ \vdots \\ c_{t_0} - b_{t_0} \end{bmatrix} \right\|_2\end{aligned}$$

but the last quantity is exactly the contribution of these knots to $R(f_{\mathcal{D}})$ (see Figure 5b). This contradicts optimality of $\tilde{g}$, hence of g and of f. The remainder of the proof is dedicating to showing that such datasets which admit non-unique solutions are rare when the data is randomly sampled from a continuous distribution.

Datasets which admit non-unique solutions have Lebesgue measure zero. If $N = 2$ or $N = 3$, then $f_{\mathcal{D}}$ is the only solution to (3), so we focus on the case where $N \geq 4$. Suppose that for some $i \in \{2, \dots, N-2\}$, the data points $x_{i-1}, x_i, x_{i+1}, x_{i+2} \in \mathbb{R}$ and labels $\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2} \in \mathbb{R}^T$ satisfy the requirement that

$$\frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i} - \frac{\boldsymbol{y}_i - \boldsymbol{y}_{i-1}}{x_i - x_{i-1}} = w \left(\frac{\boldsymbol{y}_{i+2} - \boldsymbol{y}_{i+1}}{x_{i+2} - x_{i+1}} - \frac{\boldsymbol{y}_{i+1} - \boldsymbol{y}_i}{x_{i+1} - x_i} \right) \tag{28}$$

for some $w > 0$, where both vectors are nonzero. After some computation, this is equivalent to the requirement that

$$(x_{i+1} - x_i)\boldsymbol{y}_{i-1} - w(x_{i+2} - x_{i+1})\boldsymbol{y}_i + \big((1-w)x_i - x_{i+1} + wx_{i+2}\big)\boldsymbol{y}_{i+1} - w(x_{i+1} - x_i)\boldsymbol{y}_{i+2} = \mathbf{0} \tag{29}$$

or equivalently

$$\underbrace{[\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2}]}_{\boldsymbol{Y}_i \in \mathbb{R}^{T \times 4}} \left(\underbrace{\begin{bmatrix} x_{i+1} - x_i \\ 0 \\ x_i - x_{i+1} \\ 0 \end{bmatrix}}_{\boldsymbol{a}_1 \in \mathbb{R}^4} - w \underbrace{\begin{bmatrix} 0 \\ x_{i+2} - x_{i+1} \\ -x_{i+2} \\ x_{i+1} - x_i \end{bmatrix}}_{\boldsymbol{a}_2 \in \mathbb{R}^4} \right) = \mathbf{0} \tag{30}$$

for some $w > 0$. In order for this requirement to be satisfied, it must be the case that $\boldsymbol{Y}\boldsymbol{a}_1 = w\boldsymbol{Y}\boldsymbol{a}_2$ for some $w > 0$, or equivalently, that the matrix $\boldsymbol{U} = \boldsymbol{Y}[\boldsymbol{a}_1, \boldsymbol{a}_2] \in \mathbb{R}^{T \times 2}$ has rank one. Since the rank of any matrix and its Gram matrix are equivalent, this is equivalent to requiring that $\boldsymbol{U}\boldsymbol{U}^\top \in \mathbb{R}^{T \times T}$ has rank one, or equivalently (because $T > 1$), that $\det(\boldsymbol{U}\boldsymbol{U}^\top) = 0$. Now observe that, based on the definition of the determinant and of the matrix $\boldsymbol{U}$, the function $\det(\boldsymbol{U}\boldsymbol{U}^\top)$ is a real-valued polynomial function of the variables $\boldsymbol{Y}_i = [\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2}] \in \mathbb{R}^{T \times 4}$ and $x_{i-1}, x_i, x_{i+1}, x_{i+2} \in \mathbb{R}^4$. Therefore, as the zero set of a polynomial function (i.e. an algebraic variety), the set of all $\boldsymbol{Y}_i = [\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{y}_{i+1}, \boldsymbol{y}_{i+2}] \in \mathbb{R}^{T \times 4}$ and $x_{i-1}, x_i, x_{i+1}, x_{i+2} \in \mathbb{R}^4$ for which $\det(\boldsymbol{U}\boldsymbol{U}^\top) = 0$ has Lebesgue measure zero. This is true for any $i = 1, \dots, N$, so taking the union over $i = 1, \dots, N$, we have that the set of all possible data points $x_1, \dots, x_N \in \mathbb{R}$ and label vectors $\boldsymbol{y}_1, \dots, \boldsymbol{y}_N \in \mathbb{R}^T$ which admit non-unique solutions has Lebesgue measure zero (as a subset of $\mathbb{R}^N \times \mathbb{R}^{T \times N}$). $\square$

7 Additional Experimental Results and Details

7.1 Numerical Illustration of Theorem 3.1

We provide numerical examples to illustrate the difference in solutions obtained from single task versus multi-task training and validate our theorem. The first row in Figure 6 shows three randomly initialized univariate neural networks trained to interpolate the five red points with minimum representational cost. While all three of the learned functions have the same representational cost (i.e., all minimize the second-order total variation subject to the interpolation constraint), they each learn different interpolants. This demonstrates that gradient descent does not induce a bias towards a particular solution. The second row shows the function learned for the first output of a multi-task neural network. This network was trained on two tasks. The first task consists of interpolating the five red points while the second consists of interpolating five randomly generated labels sampled from a standard Gaussian distribution. When trained to minimum representation cost we see that the connect-the-dots solution is the only solution learned regardless of initialization, verifying Theorem 3.1. This solution simultaneously minimizes the second-order total variation and the norm in the RKHS associated with the kernel $k(x, x') = 1 - (x - x')_+ + (x - x_1)_+ + (x_1 - x')_+$ De Boor and Lynch (1966), subject to the interpolation constraints.

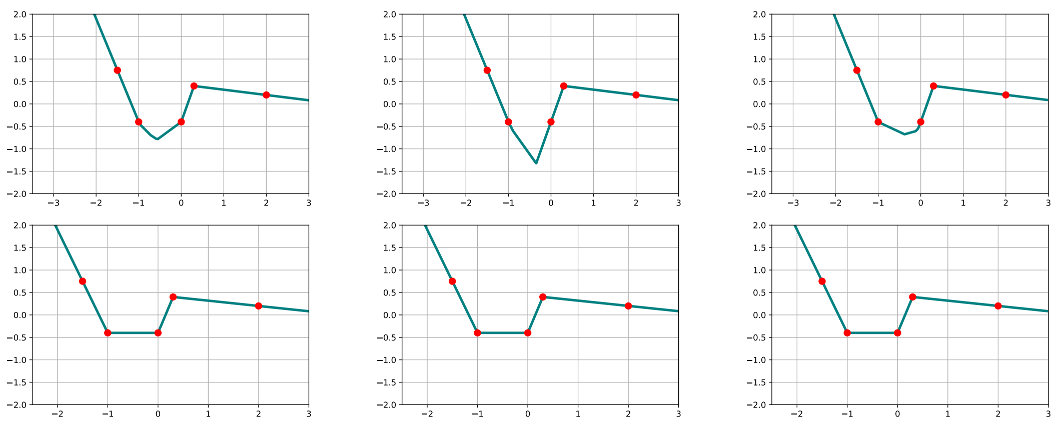


Figure 6: *Top Row*: Three randomly initialized neural networks trained to interpolate the five red points with minimum path-norm. *Bottom Row*: Three randomly initialized two-output neural networks trained to interpolate a multi-task dataset with minimum path-norm. The labels for the first task are the five red points shown while the labels for the second were randomly randomly sampled from a standard Gaussian distribution.

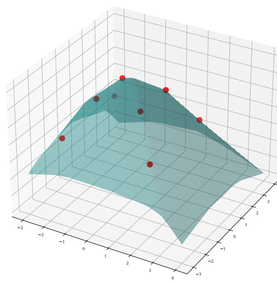
7.2 Additional Experiments from Section 4

All of our experiments were carried out in PyTorch and used the Adam optimizer. We trained the models with mean squared error loss and included the representational cost $\sum_{k=1}^{K} \|\boldsymbol{v}_k\|_2$ as a regularizer with $\lambda = 1e-5$ for the univariate experiments and $\lambda = 1e-3$ for the multi-variate experiments. All models were trained to convergence. The networks were initialized with 20 neurons for the univariate experiments and 800 neurons for the multi-variate experiments. For solving (23) we utilized CVXPy Diamond and Boyd (2016).

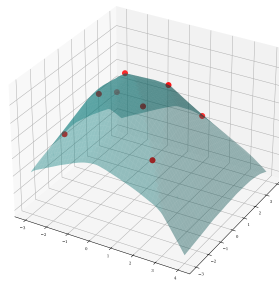
Figure 7 below provides additional experimental results to accompany the discussion in Section 4. The results demonstrate that our observations from Section 4 are true setting across multiple random initializations of the network.

7.3 High Dimensional Multivariate Experiments

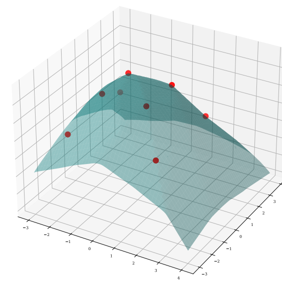
In this section we provide additional experiments in a higher dimensional setting to demonstrate how multi-task solutions can differ drastically from single-task. For these experiments we consider a student-teacher model. In particular, we generated 25 random ReLU neurons with unit norm



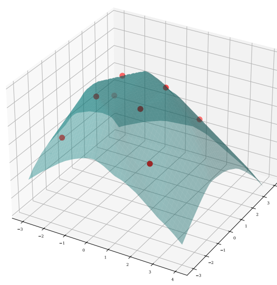
(a)



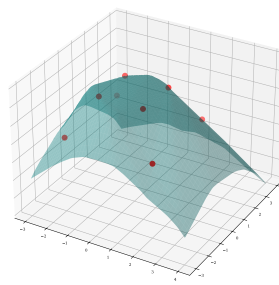
(b)



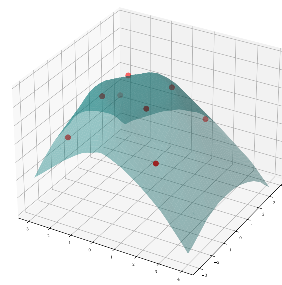
(c)



(d)



(e)



(f)

Figure 7: We present three more trials of the same experiment from Section 4. The top row corresponds to the solution of the fist output of a multi-task neural network with $T = 101$ tasks. The first task is the original (i.e. interpolating the red points), the other 100 are randomly sampled i.i.d from a Bernoulli distribution with equal probability for one and zero. The second row corresponds to the solutions obtained by solving (23). We see again that for the $T = 101$ multi-task neural network the learned function is consistent across multiple random initializations. Moreover, those solutions are also similar to the ones obtained by solving (23). These results suggest that with many diverse tasks the contributions of any one task on the optimal neurons are not significant.

input weights $\boldsymbol{w}_t \in \mathbb{R}^5$ for $t = 1, \ldots, 25$. These served as “teacher" neurons. We then generated a multi-task dataset $\{\boldsymbol{x}_i, \boldsymbol{y}_i\}_{i=1}^{20}$ where each $\boldsymbol{x}_i \in \mathbb{R}^5$ and sampled i.i.d from a standard multi-variate Gaussian distribution. The labels $\boldsymbol{y}_i \in \mathbb{R}^{25}$ were then generated according to the teacher ReLU neurons, that is,

$$y_{it} = (\boldsymbol{w}_t^T \boldsymbol{x}_i)_+.$$

Therefore, each task is generated by a single-index model. We then trained 25 student single-output ReLU neural networks on each tasks as well as a 25-output multi-task ReLU neural network on all the tasks. Both were trained to minimize MSE loss and were regularized a weight decay parameter of $\lambda = 1e - 4$. All networks nearly interpolated the data with MSE value less than $1e - 4$. Figure 10 plots the learned single task networks evaluated along a a unit norm vector $\boldsymbol{w} \in \mathbb{R}^5$. From the plots it is clear that the single task networks recover the ground truth function (i.e. a single ReLU neuron) as it looks like a ReLU ridge function in every direction. Moreover, we observed a an average sparsity of five active neurons across all the trained single-output networks.

In the Figure 9 we also plot the output of the t^{th} function from the leanred multi-task network evaluated at the same $\boldsymbol{w}$. In this case, the functions look very different from a single-index model and do not recover the ground truth data-generating function for the respective task. Figure 8 shows the sparsity pattern of the weights for each neuron with roughly 150 neurons contributing to all the outputs.

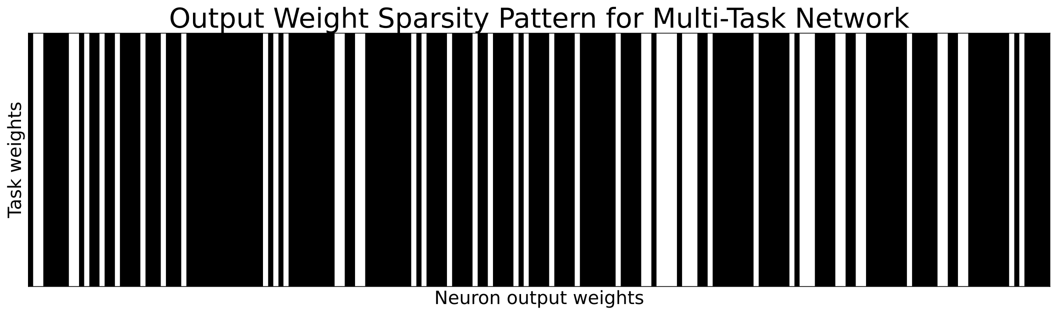


Figure 8: Sparsity pattern for output weight matrix of the multi-task student network. The k^{th} column in the matrix corresponds to the output weight of the k^{th} neuron. We observe that each neuron either contributes to all the tasks or none.

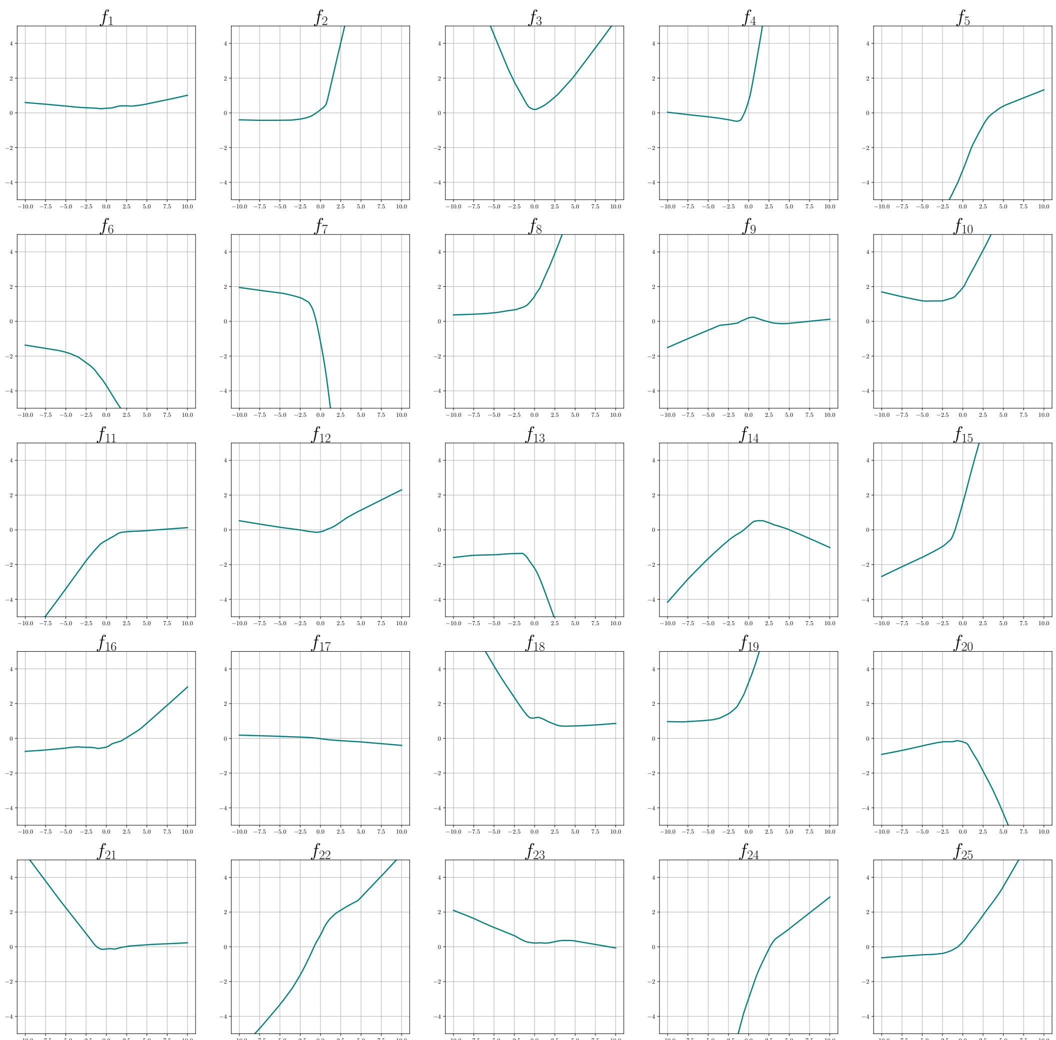


Figure 9: Multi-task solutions along the same direction $\boldsymbol{w}$. Here f_t denotes the t^{th} output of the multi-task network. We observe that unlike Figure 10 the functions do not look like ReLU ridge functions and have variation in multiple directions.

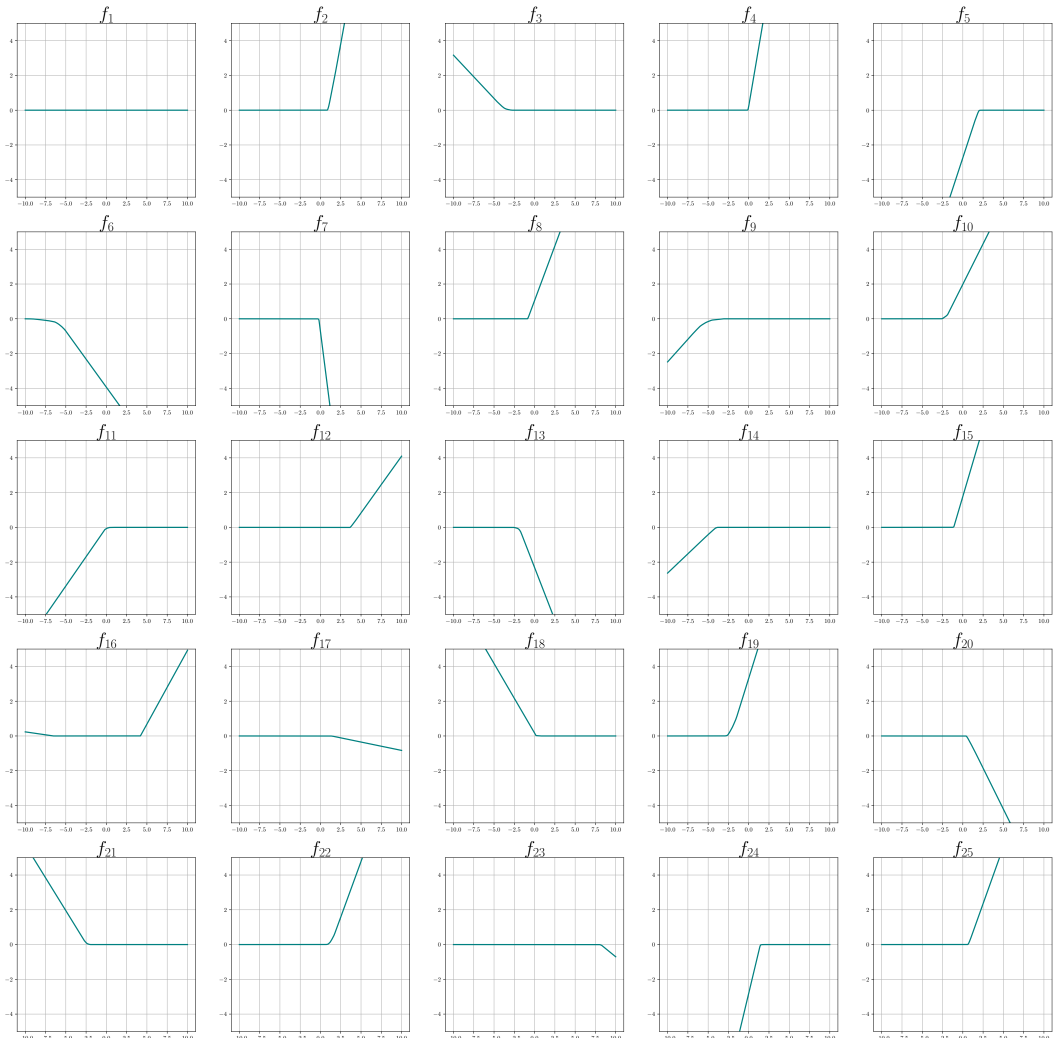


Figure 10: The 25 single-task networks evaluated along the same direction $\boldsymbol{w}$ as in Figure 9. Here f_t denotes the t^{th} single-task network trained on task t according to the data generating function above. Here as we expect the single-task nets are ReLU ridge functions. We note that these observations hold across different choices of the one-dimensional subspace $\boldsymbol{w}$.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our claims are supported by the theory/analysis and numerical experiments in Sections 3 and 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations and future research directions which could address these limitations in Section 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: The assumptions of all theorems and analysis are stated, and detailed proofs are provided either in the main body of the paper or in the appendix (in which case the relevant appendix section is clearly referenced within the paper).

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The relevant details needed to reproduce our (simple) numerical experiments are provided in Appendix 7.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Our experiments use only synthetically-generated data, and the procedure by which the data is generated is described in each case, so there is no real-world data to share. The numerical experiments are also very simple to run and we judge that providing code for them is not necessary: using the details provided in Appendix 7, they should be easy to manually reproduce.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All relevant training details to reproduce the experiments are included in either the figures where the results are presented or in Appendix 7.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Our experiments are included to visually illustrate the types of functions learned by multi-task neural networks with weight decay regularization, and as such, we do not report on error or other metrics since this is not what these experiments are intended to highlight.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Our experiments are computationally very simple to execute on any computer architecture and do not require GPUs, so we judge that these details are not important to include.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our paper involves no human subjects, no real-world data (hence no privacy concerns), and no other negative societal consequences that we are aware of.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: Our investigation here is primarily theoretical and, while we hope our work will contribute toward developing a stronger mathematical theory of deep learning (which may contribute toward the safety and robustness guarantees of machine learning at large), we are not aware of any direct application of our results here in a way that would have a substantial societal impact.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: Our paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The computational packages used (PyTorch and CVXPy Diamond and Boyd (2016)) are both open-source, and are explicitly mentioned and credited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: Our paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: Our paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: Our paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.