A Functional Characterization of Randomly Initialized Gradient Descent in Deep ReLU Networks

Anonymous authors
Paper under double-blind review

Abstract

Despite their popularity and successes, deep neural networks are poorly understood theoretically and treated as ’black box’ systems. Using a functional view of these networks gives us a useful new lens with which to understand them. This allows us us to theoretically or experimentally probe properties of these networks, including the effect of standard initializations, the value of depth, the underlying loss surface, and the origins of generalization. One key result is that generalization results from smoothness of the functional approximation, combined with a flat initial approximation. This smoothness increases with number of units, explaining why massively overparamaterized networks continue to generalize well.

1 Introduction

Deep neural networks, trained via gradient descent, have revolutionized the field of machine learning. Despite their widespread adoption, theoretical understanding of fundamental properties of deep learning – the true value of depth, the root cause of implicit regularization, and the seemingly ‘unreasonable’ generalization achieved by overparameterized networks – remains mysterious.

Empirically, it is known that depth is critical to the success of deep learning. Theoretically, it has been proven that maximum expressivity grows exponentially with depth, with a smaller number of trainable parameters (Raghu et al., 2017; Poole et al., 2016). This theoretical capacity may not be used, as recently shown explicitly by (Hanin & Rolnick, 2019). Instead, the number of regions within a trained network is proportional to the total number of hidden units, regardless of depth. Clearly deep networks perform better, but what is the value of depth if not in increasing expressivity?

Another major factor leading to the success and widespread adoption of deep learning has been its surprisingly high generalization performance (Zhang et al., 2016). In contrast to other machine learning techniques, continuing to add parameters to a deep network (beyond zero training loss) tends to *improve* generalization performance. This is true despite the fact that networks are often massively overparameterized, wherein according to traditional ML theory they should (over)fit all the training data (Neyshabur et al., 2015). How does training deep networks with excess capacity lead to generalization? And how can it be that this generalization error decreases with overparameterization?

We believe that taking a functional view allows us a new, useful lens with which to explore and understand these issues. In particular, we focus on the case of deep fully connected univariate ReLU networks, whose parameters will always result in a Continuous Piecewise Linear (CPWL) approximation to the target function.

Our approach is related to previous work from (Savarese et al., 2019; Arora et al., 2019; Frankle & Carbin, 2018) in that we wish to characterize parameterization and generalization. We differ from these other works by doing using small widths, rather than massively overparamaterized or infinite, and by using a functional parameterization to measure properties such as roughness.

Main Contributions The main contribution of this work are as follows: *Functional Perspective of Initialization: Increasingly Flat with Depth.* In the functional perspective, neural network parameters determine the locations of breakpoints and their delta-slopes in the CPWL reparameterization.

We prove that, for common initializations, these distributions are mean 0 with low standard deviation. The delta-slope distribution becomes increasingly concentrated as the depth of the network increases, leading to flatter approximations. In contrast, the breakpoint distribution grows wider, allowing deeper network to better approximate over a broader range of inputs.

Value of Depth: Optimization, not Expressivity. Theoretically, depth adds an exponential amount of expressivity. Empirically, this is not true in trained deep networks. We find that expressivity scales with the number of total units, and weakly if at all with depth. However, we find that depth makes it easier for GD to optimize the resulting network, allowing for a greater flexibility in the movement of breakpoints, as well as the number of breakpoints induced during training.

Generalization is due to Flat Initialization in the Overparameterized Regime. We find that generalization in overparametrized FC ReLu nets is due to three factors: (i) the very flat initialization, (ii) the curvature-based parametrization of the approximating function (breakpoints and delta-slopes) and (iii) the role of gradient descent (GD) in preserving (i) and regularizing via (ii). In particular, the global, rather than local, impact of breakpoints and delta-slopes helps regularize the approximating function in the large gaps between training data, resulting in their smoothness. Due to these nonlocal effects, more overparameterization leads to smoother approximations (all else equal), and thus typically better generalization(Neyshabur et al., 2018; 2015).

2 Theoretical Results

2.1 ReLU Nets in Function Space: From Weights to Breakpoints & Slopes

Consider a fully connected ReLU neural net $\hat{f}_\theta(x)$ with a single hidden layer of width H, scalar input $x \in \mathbb{R}$ and scalar output $y \in \mathbb{R}$. $\hat{f}(\cdot;\theta)$ is continuous piecewise linear function (CPWL) since the ReLU nonlinearity is CPWL. We want to understand the *function* implemented by this neural net, and so we ask: How do the CPWL parameters relate to the NN parameters? We answer this by transforming from the NN parametrization (weights and biases) to two CPWL parametrizations:

$$\hat{f}(x;\theta_{NN}) \triangleq \sum_{i=1}^{H} v_i(w_i x + b_i)_+ \tag{1}$$

$$= \sum_{i=1}^{H} \mu_i(x-\beta_i) \begin{cases} [\![x > \beta_i]\!], & s_i = 1 \\ [\![x < \beta_i]\!], & s_i = -1 \end{cases} \triangleq \hat{f}(x;\theta_{BDSO}) \tag{2}$$

$$= \sum_{p=1}^{P} [\![\beta_p \leq x < \beta_{p+1}]\!] \,(m_p x + \gamma_p) \triangleq \hat{f}(x;\theta_{PWL}) \tag{3}$$

where the Iversen bracket $[\![b]\!]$ is 1 when the condition b is true, and 0 otherwise. Here the NN parameters $\theta_{NN} \triangleq \{(w_i, b_i, v_i)\}_{i=1}^{H}$ denote the input weight, bias, and output weight of neuron i, and $(\cdot)_+ \triangleq \max\{0,\cdot\}$ denotes the ReLU function. The first CPWL parametrization is $\theta_{BDSO} \triangleq \{(\beta_i, \mu_i, s_i)\}_{i=1}^{H}$, where $\beta_i \triangleq -\frac{b_i}{w_i}$ is (the x-coordinate of) the *breakpoint* (or *knot*) induced by neuron i, $\mu_i \triangleq w_i v_i$ is the *delta-slope* contribution of neuron i, and $s_i \triangleq \operatorname{sgn} w_i \in \{\pm 1\}$ is the *orientation* of β_i (left for $s_i = -1$, right for $s_i = +1$). Intuitively, in a good fit the breakpoints β_i will congregate in areas of high curvature in the ground truth function $|f''(x)| \geq 0$, while delta-slopes μ_i will actually implement the needed curvature by changing the slope by μ_i from one piece $p(i)$ to the next $p(i)+1$. As the number of pieces grows, the approximation will improve, and the delta-slopes (scaled by the piece lengths) approach the true curvature of f: $\lim_{P\to\infty} \mu_{p(i)}/(\beta_p - \beta_{p-1}) \to f''(x = \beta_i)$.

We note that the BDSO parametrization of a ReLU NN is closely related to but different than a traditional roughness-minimizing m-th order spline parametrization $\hat{f}_{\text{spline}}(x) \equiv \sum_{i=1}^{K} \mu_i(x-\beta_i) + \sum_{j=0}^{m} c_j x^j$: BDSO (i) lacks the base polynomial, and (ii) it has two possible breakpoint orientations $s_i \in \{\pm 1\}$ whereas the spline only has one. We note in passing that adding in the base polynomial (for linear case $m = 1$) into the BDSO ReLU parametrization yields a ReLU ResNet parametrization. We believe this is a novel viewpoint that may shed more light on the origin of the effectiveness of ResNets, but we leave it for future work.

The second parametrization is the canonical one for PWL functions: $\theta_{PWL} \triangleq \{(\beta_p, m_p, \gamma_p)\}_{p=1}^{P}$, where $\beta_0 < \beta_1 < \ldots < \beta_p \triangleq -\frac{b_{p(i)}}{w_{p(i)}} < \ldots < \beta_P$ is the sorted list of (the x-coordinates of) the $P \triangleq H + 1$ breakpoints (or knots), m_p, γ_p are the slope and y-intercept of piece p.

Computing the analogous reparametrization to function space for deep networks is more involved, so we present a basic overview here, and a more detailed treatment in the supplement. For $L \geq 2$ layers with widths $H^{(\ell)}$, the neural network's activations are defined as: $z_i^{(\ell)} = \sum_{j=1}^{H^{(\ell-1)}} w_{ij}^{(\ell)} x_j^{(\ell-1)} + b_i^{(\ell)}, x_i^{(\ell)} = (z_i^{(\ell)})_+, g_\theta(x) = z^{(L+1)}$ for all hidden layers $\ell \in \{1, 2, \ldots, L\}$ and for all neurons $i \in \{1, 2, \ldots, H^{(\ell)}\}$. Then $\beta_i^{(\ell)}$ is a *breakpoint induced by neuron i in layer ℓ* if it is a zero-crossing of the net input i.e. $z_i^{(\ell)}(\beta_i^{(\ell)}) = 0$. The definition of *active* breakpoints in deep nets is a bit more subtle; see Supplement for details.

Considering these parameterizations (especially the BDSO parameterization) provides a new, useful lens with which to analyze neural nets, enabling us to reason more easily and transparently about the initialization, loss surface, and training dynamics. The benefits of this approach derive from two main properties: (1) that we have 'modded out' the degeneracies in the NN parameterization and (2) the loss depends on the NN parameters θ_{NN} only through the BDSO parameters (the approximating function) θ_{BDSO} i.e. $\ell(\theta_{NN}) = \ell(\theta_{BDSO}(\theta_{NN}))$, analogous to the concept of a minimum sufficient statistic in exponential family models. Much recent related work has also veered in this direction, analyzing function space (Hanin & Rolnick, 2019; Balestriero et al., 2018) (see Related Work for more details).

2.2 Random Initialization in Function Space

We now study the random initializations commonly used in deep learning in function space. These include the independent Gaussian initialization, with $b_i \sim N(0, \sigma_b)$, $w_i \sim N(0, \sigma_w)$, $v_i \sim N(0, \sigma_v)$, and independent Uniform initialization, with $b_i \sim U[-a_b, a_b]$, $w_i \sim U[-a_w, a_w]$, $v_i \sim U[-a_v, a_v]$.

Theorem 1. *Consider a fully connected ReLU neural net with scalar input and output, and a single hidden layer of width H. Let the weights and biases be initialized randomly according to a zero-mean Gaussian or Uniform distribution. Then the induced distributions of the function space parameters (breakpoints β, delta-slopes μ) are as follows:*

(a) Under an independent Gaussian initialization,

$$p_{\beta,\mu}(\beta_i, \mu_i) = \frac{1}{2\pi\sigma_v\sqrt{\sigma_b^2 + \sigma_w^2\beta_i^2}} \exp\left[-\frac{|\mu_i|\sqrt{\sigma_b^2 + \sigma_w^2\beta_i^2}}{\sigma_b\sigma_v\sigma_w}\right]$$

(b) Under an independent Uniform initialization,

$$p_{\beta,\mu}(\beta_i, \mu_i) = \frac{[\![|\mu_i| \leq \min\{\frac{a_b a_v}{|\beta_i|}, a_w, a_v\}]\!]}{4a_b a_w a_v}\left(\min\{\frac{a_b}{|\beta_i|}, a_w\} - \frac{|\mu_i|}{a_v}\right)$$

Using this result, we can immediately derive marginal and conditional distributions for the breakpoints and curvatures (delta-slopes).

Corollary 1. *Consider the same setting as Theorem 1.*

(a) In the case of an independent Gaussian initialization,

$$p_\beta(\beta_i) = \textit{Cauchy}\left(\beta_i; 0, \frac{\sigma_b}{\sigma_w}\right) = \frac{\sigma_b\sigma_w}{\pi\left(\sigma_w^2\beta_i^2 + \sigma_b^2\right)}$$

$$p_\mu(\mu_i) = \frac{1}{2\pi\sigma_v\sigma_w} G_{0,2}^{2,0}\left(\frac{\mu_i^2}{4\sigma_v\sigma_w}\middle| 0, 0\right) = \frac{1}{\pi\sigma_v\sigma_w} K_0\left(\frac{|\mu_i|}{\sigma_v\sigma_w}\right)$$

$$p_{\mu|\beta}(\mu_i|\beta_i) = \textit{Laplace}\left(\mu_i; 0, \frac{\sigma_b\sigma_v\sigma_w}{\sqrt{\sigma_b^2 + \sigma_w^2\beta_i^2}}\right) = \frac{\sqrt{\sigma_b^2 + \sigma_w^2\beta_i^2}}{2\sigma_b\sigma_v\sigma_w} \exp\left[-\frac{|\mu_i|\sqrt{\sigma_b^2 + \sigma_w^2\beta_i^2}}{\sigma_b\sigma_v\sigma_w}\right],$$

where $G_{pq}^{nm}(\cdot|\cdot)$ is the Meijer G-function and $K_\nu(\cdot)$ is the modified Bessel function of the second kind.

(b) In the case of an independent Uniform initialization,

$$
\begin{aligned}
p_\beta(\beta_i) &= \frac{1}{4a_b a_w}\left(\min\left\{\frac{a_b}{|\beta_i|}, a_w\right\}\right)^2 \\
p_\mu(\mu_i) &= \frac{[\![-a_w a_v \le \mu_i \le a_w a_v]\!]}{2a_w a_v}\log\frac{a_w a_v}{|\mu_i|} \\
p_{\mu|\beta}(\mu_i|\beta_i) &= \mathit{Tri}(\mu_i; a_v \min\{a_b/|\beta_i|, a_w\}) = \frac{[\![|\mu_i| \le a_v \min\{a_b/|\beta_i|, a_w\}]\!]}{a_v \min\{a_b/|\beta_i|, a_w\}}\left(1 - \frac{|\mu_i|}{a_v \min\{a_b/|\beta_i|, a_w\}}\right),
\end{aligned}
$$

where Tri$(\cdot; a)$ is the symmetric triangular distribution with base $[-a, a]$ and mode 0.

Implications. Corollary 1 implies that the breakpoint density drops as quickly away from the origin. If f has significant curvature in the boundaries, then it will far more difficult to fit than if it were near the origin. We show that this is indeed the case by training a shallow ReLU NN on samples from $f(x) \equiv \sin(x)$ with gradient descent(GD) (see Table **??** for details). Another important implication is the need for data-dependent initializations. If one has knowledge of f, namely where its curvature lies, then an initialization that allocates more breakpoints to such areas will be faster to train and, potentially, require less breakpoints (and thus lower width) overall. We show that simple data-dependent initialization that change/adapt the breakpoint density to be closer to ground truth do indeed converge faster and achieve a better training loss(see Sec 3 and Table **??**).

Theorem 2. *The roughness $\rho_0 \equiv \sum_{i=1}^{H} \mu_i^2$ of the function induced by the random Gaussian initialization has mean $(\sigma_v\sigma_w)^2 H = 4H/(H+1)^2 = O(1/H)$ and variance $8(\sigma_v\sigma_w)^2 = 128/(H+1)^4$ where we have used $\sigma_v = \sigma_w = \sqrt{2/(H+1)}$, the default weight variance used in standard He and Glorot initializations. The tail probability for the initial roughness is* $\Pr(\rho_0 > 4/H + \lambda) \le 1/(1 + \lambda^2 H^3/128) = O(1)$.

As width H increases, the roughness of the initial function $\hat{f}$ decreases as $1/H$. This smoothness has implications for the implicit regularization/generalization phenomenon observed in recent work (Neyshabur et al., 2018)(see Sec 3,3,3 for generalization/smoothness analysis during training).

Related Work. Several recent works analyze the random initialization in deep networks. However, there are two main differences, First, they focus on the infinite width case (Savarese et al., 2019; Jacot et al., 2018; Lee et al., 2017) and can thus use the Central Limit Theorem (CLT), whereas we focus on finite width case and cannot use the CLT, thus requiring nontrivial mathematical machinery (see Supplement for detailed proofs). Second, they focus the activations as a function of input whereas we also compute the joint densities between the BDSO parameters e.g. breakpoints and curvatures (delta-slopes). The latter is particularly important for understanding the non-uniform density of breakpoints away from the origin as noted above.

2.3 Loss Surface in the Function Space

Consider the mean squared error (MSE) loss with respect to the NN parameters $\ell(\theta_{NN}) \triangleq \sum_{n=1}^{N} \frac{1}{2}(\hat{f}(x_n;\theta) - y_n)^2$. and the BDSO parameters $\tilde{\ell}(\theta_{BDSO})$. Now consider some $\theta_{BDSO} \in \Theta_{BDSO}$. Then $\hat{f}(\cdot;\theta_{BDSO})$ induces a partition $\Pi = (\pi_1, \ldots, \pi_{H+1})$ of the data $\{x_n\}_{n=1}^{N}$. Note that the restriction of $\hat{f}_{BDSO}$ to any piece of this partition, denoted $\hat{f}(\cdot;\theta_{BDSO})|_{\pi_p}$, is a linear function.

Theorem 3. *θ^*_{BDSO} is a critical point of $\tilde{\ell}(\theta_{BDSO})$ if for all pieces $p \in [P]$ we have that $\hat{f}(\cdot;\theta_{BDSO})|_{\pi_p}$ is an Ordinary Least Squares fit of the data in piece p, and we refer to the critical point as a (C)PWL-OLS solution. Furthermore every critical point θ^*_{BDSO} of $\tilde{\ell}(\theta_{BDSO})$ corresponds to an equivalence class of critical points $\theta^*_{NN} \in [\mathcal{G}\theta^*_{BDSO}]$ of $\ell(\theta_{NN})$ where $\mathcal{G}$ is the set of transformations on the NN parameters that leaves the function (BDSO parameters) invariant.*

Since each θ_{BDSO} induces a partition $\Pi_{N,H}$ of the input data, we can count the number of critical points by counting the number of possible partitions of N data points with H breakpoints into $P = H + 1$ pieces as simply $C(N + H, H) \triangleq (N + H)!/N!H!$. This is not quite right as we should only count the continuous PWL OLS solutions (since $\hat{f}$ is CPWL) which will in general be less than the total PWL OLS solutions $C(N+H, H)$. How much less? It is difficult to analytically characterize the CPWL OLS solutions so we resort to simulation and find a lower bound that suggests the number of critical points grows at least polynomially in N, H (Fig. 7). We believe this is the function space explanation for why GD cannot move the breakpoints very far, analogous to the weight space explanation provided by (Arora et al., 2019) wherein the Gram matrix $\mathbf{H}(t)$ remains very close to its initial value $\mathbf{H}(0)$. A key difference between our results is theirs relies on *massive* overparametrization ($H = \Omega(N^7)$) whereas our applies for all H, albeit with an unproven conjecture. However, in the overparametrized regime $H > N$ we can prove the following result:

Theorem 4. *Consider the partition $\Pi_{N,H}$ as defined above. A partition is lonely if each datapoint n is alone in its own piece p. (a) The PWL OLS solution for a lonely partition is (i) CPWL, (ii) a local minima and (iii) a global minima of $\tilde{\ell}$. (b) Furthermore, if we assume that H breakpoints are uniformly spaced and that N data points are uniformly distributed within the region of breakpoints, then in the overparametrized regime $H \geq \alpha N^2$ for some constant $\alpha > 1$, the induced partition $\Pi_{N,H}$ is lonely with high probabilility $1 - e^{-N^2/(H+1)} = 1 - e^{-1/\alpha}$. Furthermore, the total number of lonely partitions, and thus (a lower bound on) the total number of global minima of $\tilde{\ell}$ is $\binom{H+1}{N} = O(N^{\alpha N})$*

The proof is straightforward: each piece p has two degrees of freedom, one to perfectly fit the data (b,c) and the other to insure continuity with adjacent pieces to the (say) right (a). Note how simple and transparent the function space explanation is for why overparametrization makes optimization easy, as compared to the weight space explanation (Arora et al., 2019).

Things to note : every partition is equally likely (in the case of uniformly spaced breakpoints, random data uniformly in this range). If this is not true, the theorem still holds, but need to increase alpha to account for 'wasted' extra partitions where data is sparse.

2.4 Gradient Descent Dynamics in the Function Space

Theorem 5. *For a one hidden layer univariate ReLU network trained with gradient descent with respect to the neural network parameters $\theta_{NN} = \{(w_i, b_i, v_i)\}_{i=1}^H$, the gradient flow dynamics of the function space parameters $\theta_{BDSO} = \{(\beta_i, \mu_i)\}_{i=1}^H$ are governed by the following laws:*

$$\frac{\mathrm{d}\beta_i}{\mathrm{d}t} = -\frac{\partial \ell(\theta_{NN})}{\partial \beta_i} = \frac{v_i(t)}{w_i(t)} \Big[\underbrace{\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1}\rangle}_{\textit{net relevant residual}} + \beta_i(t) \underbrace{\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{x}\rangle}_{\textit{correlation}}\Big] \quad (4)$$

$$\frac{\mathrm{d}\mu_i(t)}{\mathrm{d}t} = -\frac{\partial \ell(\theta_{NN})}{\partial \mu_i} = -(v_i^2(t) + w_i^2(t)) \langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{x}\rangle - w_i(t) b_i(t) \langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1}\rangle \quad (5)$$

2.5 Generalization: Implicit Regularization via Delta-Slope Parametrization

Given the above dynamics, we ask the question: how can we make sense of the phenomena like implicit regularization in function space? In Sec 3 we confirm that we can reproduce these phenomena in our FC ReLu networks with target functions from various classes. We also find that the smoothness (roughness) of the initialization matters quite a bit. But smoothness alone is not enough; here we show that the delta-slope parametrization is critical in enabling implicit regularization.

Consider a dataset like that shown in Fig. 8 with a data gap between regions of two continuous functions f_L, f_R and consider a breakpoint i with orientation s_i in the gap. Starting with a flat initialization, the dynamics of the i-th delta-slope are $\dot{\mu}_i(t) = -\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{x}\rangle + \beta_i(t) \langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1}\rangle \triangleq r_{2,s_i}(t) + r_{3,s_i}(t)\beta_i(t)$ where $r_{2,s}(t), r_{3,s}(t)$ are the (negative) net correlation and residual on the active side of i, in this case including data from the function f_{s_i} but not f_{-s_i}. Note that the both terms of the gradient $\dot{\mu}_i$ have a weak dependence on i through the orientation s_i, and the second term additionally depends on i through $\beta_i(t)$. Thus the vector of delta-slopes with orientation s evolves according to $\dot{\boldsymbol{\mu}}_s = r_{2,s}(t)\mathbf{1} + r_{3,s}(t)\boldsymbol{\beta}_s$. Now consider the regime of overparametrization $H \gg N$. It will turn out to be identical to taking a continuum limit $H \to \infty$ yielding $\mu_i/(\beta_i - \beta_{i-1}) \to \mu(x,t) \equiv \hat{f}''(x,t)$, the curvature of the approximation (the discrete index i has become a continuous index x) and $\dot{\beta}_i(t) \to 0$ (following from Thm 5, multiplying $\dot{\beta}_i(t)$ by $v_i(t)/w_i(t)$ and factoring out $\mu_i(t) \to 0$). Integrating the dynamics $\dot{\mu}_s(x,t) = r_{2,s}(t) + r_{3,s}(t)x$

over all time yields $\mu(x, t = \infty) = \mu(x, t = 0) + R^*_{2,s} + R^*_{3,s}x$, where the curvature $\mu(x, t = 0) \approx 0$ (Sec. 3) and $R^*_{j,s} \triangleq \int_0^\infty dt' r_{j,s}(t') < \infty$ (convergence of residuals $\epsilon_n(t)$ and immobility of breakpoints $\dot{\beta}_i(t) = 0$ implies convergence of $r_{j,s}(t)$). Integrating over space twice yields a cubic spline $\hat{f}(x, t) = c_{0,s} + c_{1,s}x + c_{2,s}(x - \xi_s)^2_s/2! + c_{3,s}(x - \xi_s)^3_s/3!$, where $c_{0,s}, c_{1,s}$ are integration constants determined by the boundary conditions $\hat{f}'(x = \xi_s, t = \infty) = \sum_s f'_s(x = \xi_s)$ and $\hat{f}(x = \xi_s, t = \infty) = \sum_s f_s(x = \xi_s)$, thus matching the 0-th and 1st derivatives at the gap endpoints. The other two coefficients $c_{k,s} \triangleq R^*_{k,s}, k \in \{2, 3\}$ and serve to match the 2nd and 3rd derivatives at the gap endpoints. Clearly, matching the training data only requires the two parameters $c_{0,s}, c_{1,s}$; and yet, surprisingly, two unexpected parameters $c_{2,s}, c_{3,s}$ emerge that endow $\hat{f}$ with smoothness in the data gap, despite the loss function not possessing any explicit regularization term. Tracing back to find the origin of these smoothness-inducing terms, we see that they emerge as a consequence of (i) the smoothness of the initial function and (ii) the active half space structure, which in turn arises due to the discrete curvature-based (delta-slope) parameterization. Stepping back, the ReLU net parameterization is a discretization of this underlying continuous 2nd-order ordinary differential equation. In Sec 3 we conduct experiments to test this theory.

3 Experiments

Breaking Bad: Breakpoint densities that are mismatched to function curvature makes optimization difficult We first test our initialization theory against real networks. We initialize fully-connected ReLU networks of varying depths, according to the popular He initializations in which are weights are sampled from width-scaled Uniform and Gaussian distributions(He et al., 2015). Fig. 1 shows experimentally measured densities of breakpoints and delta-slopes. Our theory matches the experiments well. The main points to note are that: (i) breakpoints are indeed more highly concentrated around the origin, and that (ii) as depth increases, delta-slopes have lower variance and thus lead to even flatter initial functions. Guided by the theory, we ask whether the standard initializations will experience difficulty fitting functions that have significant variation in the boundary, a common situation in many important applications (e.g. learning the energy function of a protein molecule). We train ReLU networks to fit a periodic function ($\sin(x)$), which has high curvature both at and far from the origin. We find that the standard initializations do quite poorly in cases where there is significant curvature away from the origin, consistent with our theory that breakpoints are essential for modeling curvature. Probing further, we observe empirically that breakpoints cannot migrate very far from their initial location, even if there are plenty of breakpoints overall, leading to highly suboptimal fits. In order to prove that it is indeed the breakpoint density that is causally responsible, we attempt to rescue the poor fitting by using a simple data-dependent initialization that samples breakpoints uniformly over the training data range $[x_{min}, x_{max}]$, achieved by exploiting Eq. (1). We train shallow ReLU networks on training data sampled from a sine and a quadratic function, two extremes on the spectrum of curvature. The data shows that uniform breakpoint density rescues bad fits in cases with significant curvature far from the origin, with less effect on other cases, confirming the theory. We note that this could be a potentially useful data-dependent initialization strategy, one that can scale to high dimensions, but we leave this for future work.

Init	Sine	Quadratic
Standard	4.096 ± 2.25	.1032 ± 0404
Uniform	2.280 ± .457	.1118 ± .0248

Table 1: Test loss for standard vs uniform breakpoint initialization, on sine and quadratic $\frac{x^2}{2}$

Explaining and Quantifying the Suboptimality of Gradient Descent. The suboptimality seen above begs a larger question: under what conditions will GD be successful? Empirically, it has been observed that neural nets must typically be massively overparameterized (relative to the number of parameters needed to express the underlying function), in order to ensure good training performance. Our theory provides a possible explanation for this phenomenon: if GD cannot move breakpoints too far from their starting point, then one natural strategy is to sample as many breakpoints as possible everywhere, allowing us to fit an arbitrary f. The downside of this strategy is that many breakpoints will add little value, but the benefit is that it will increase the likelihood that areas of high curvature – wherever they are – will have access to some breakpoints nearby. In order to test this explanation and, more generally, understand the root causes of the GD's difficulty, we focus on the case of a fully connected 2-layer ReLU network. A

L	Sine	5 piece poly	Sawtooth	Arctan	Exponential	Quadratic
1	40 ± 0	40 ± 0	40 ± 0	40 ± 0	40 ± 0	40 ± 0
2	55.5 ± 2.9	52 ± 1.414	50 ± .7	49.25 ± 3.3	51.25 ± 6.1	49.25 ± 4.5
4	68 ± 3.1	57.25 ± 6.8	48.5 ± 2.5	42.5 ± 4.8	40.25 ± 3.9	40.25 ± 3.3
5	62.25 ± 15.1	49 ± 3.5	44.5 ± 5.1	38 ± 5.1	33.75 ± 1.1	31.5 ± 1.7

Table 2: Comparison of the number of pieces induced in a network of up to depth 5, with 40 units evenly distributed across layers, trained to fit varying target functions.

univariate input (i) enables us to use our theory, (ii) allows for visualization of the entire learning trajectory, and (iii) enables direct comparison with existing globally (near-)optimal algorithms for fitting PWL functions. The latter include the Dynamic Programming algorithm for 1D segmented regression (DP, (Bai & Perron, 1998)), and a very fast greedy approximation known as Greedy Merge (GM, (Acharya et al., 2016)). How do these algorithms compare to GD, across different target function classes, in terms of training loss, and the number of parameters/hidden units? Note that here we are referring to the BDSO (functional) parameters, as the GM and DP algorithms we are primarily concerned with the total number of available linear pieces in the CPWL approximation. We use this metric for the neural network as well, rather than the more typical number of trainable parameters.

Taking the functional approximation view allows us to directly compare neural network performance to these CPWL approximation algorithms. For a quadratic function (e.g. with high curvature, requiring many pieces), we find that the globally optimal DP algorithm can quickly reduce training error to near 0 with order 100 pieces. The GM algorithm, a relaxation of the DP algorithm, requires slightly higher pieces (e.g. 100 instead of 80), but requires significantly less computational power. On the other hand all variants of GD (vanilla, Adam, SGD w/ BatchNorm) all require far more pieces to reduce error below a target threshold. Even worse, they do not appear to use this large number of pieces efficiently, often showing little or even negative loss changes for order thousands of pieces. Interestingly, we observe is a strict ordering of optimization quality with Adam outperforming BatchNorm SGD outperforming Vanilla GD. These results (Fig. 1) show how inefficient GD is with respect to (functional) parameters, requiring order of magnitude more for similar performance to exact or approximate CPWL fitting algorithms.

Learned Expressivity is not Exponential in Depth. As shown previously, GD on ReLU NNs requires a large degree of overparameterization in order to optimize well. In the previous experiment, we counted the number of linear pieces in the CPWL approximation as the number of parameters, rather than the number of weights. Empirically, we know that the greatest successes have come from *deep* learning. This raises the question: how does the depth of a network affect its expressivity (as measured in the number of pieces)? Theoretically, it is well known that maximum expressivity increases exponentially with depth, which, in a deep ReLU neural network, means an exponential increase in the number of linear pieces in the CPWL approximation. Thus, theoretically the main power of depth is that it allows for more powerful function approximation relative to a fixed budget of parameters compared to a shallow network. However, recent work by Hanin and Rolnick (Hanin & Rolnick, 2019) has called this into question, finding that in realistic networks expressivity does not scale exponentially with depth. We perform a similar experiment here, asking how the number of pieces in the CPWL function approximation of a deep ReLU network varies with depth.

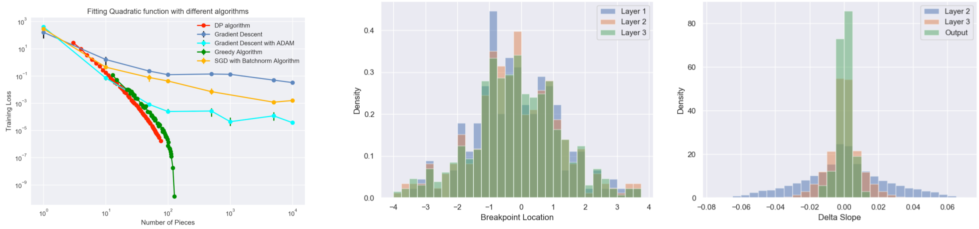

Figure 1: Left: Training loss vs number of pieces ($\propto$ number of parameters) for various algorithms fitting a CPWL function to a quadratic. Middle: Breakpoint distribution for a He initialization across a 3 layer network. Right: Delta-slope distribution for a He initialization across a 3 layer network.

The results in Table 2 clearly show that that our expressivity (the number of pieces) scales with order units, rather than with the theoretically expressivity that is exponential in depth. In fact, we find that depth only has a weak effect overall, although more study is needed to determine exactly what effect depth has on the number and variability of pieces. These results lend more support to the recent findings of Hanin and Rolnick(Hanin & Rolnick, 2019), and of taking a functional view of measuring parameterization. Intriguingly, variability in the number of pieces appears to increase with depth. From the functional approximation, we know that a deeper layer induces one or more breakpoints only if the ReLU function applied to the unit's CPWL approximation creates new breakpoints at zero crossings. In layer one, this happens exactly once per unit as the input to each ReLU is just a line over the input space. In deeper layers, the function approximation is learned, allowing for a varying number of new breakpoints. Given our previous results on the flatness of the standard initializations, this will generally only happen once per unit, implying that the number of pieces will strongly correlate with number of units at initialization.

Depth helps with Optimization by enabling the Creation, Annihilation and Mobility of Breakpoints. If depth does not strongly increase expressivity, then it is natural to ask whether its value lies with the optimization. In order to test this, we examine how the CPWL function approximation develops in each layer during learning, and how it depends on the target function. A good fit requires that breakpoints accumulate at areas of higher curvature in the training data, as these regions require more pieces. We argue that the deeper layers of a network help with this optimization procedure, allowing the breakpoints more mobility as well as the power to create and annihilate breakpoints.

As previously expressed, one key difference between the deeper layers of a network and the first layer is the ability for a single unit to induce multiple breakpoints, as the deeper layers learn CPWL functions more complex than just a line. As these functions change during learning, the number of breakpoints induced by deeper units in a network can vary, allowing for another degree of freedom for the network to optimize. Through the functional parameterization of the hidden layers, these ”births and deaths” of breakpoints can be tracked as changes in the number of breakpoints induced per layer. Another possible explanation for the value added of depth is breakpoint mobility, or that breakpoints in deeper layers can move more than those in shallow layers. We run experiments comparing how the velocity of breakpoints varies between layers of a deeper network. We also compare the number of times breakpoints in any layer undergo birth or death.

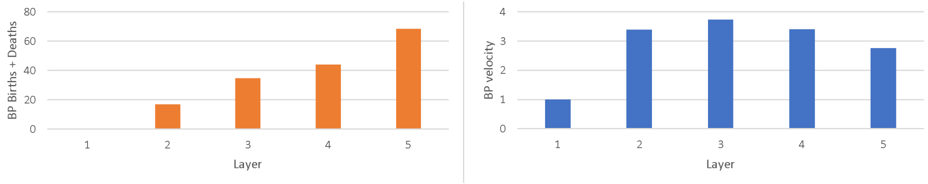

Figure 2: Total changes in number of breakpoints induced and average velocity of breakpoints relative to the first layer in each layer of a five layer ReLU network

Fig. 2 shows the results. The number of breakpoints in deeper layers changes more often than in the shallow layers, while the number of breakpoints in the first layer cannot change. The breakpoint velocity in deeper layers is also higher than the first layer, although not monotonically increasing. Both of these results provide support for the idea that later layers help significantly with optimization and breakpoint placement, even if they do not help as strongly with expressivity.

Note that breakpoints induced in by a layer of the network are present in the basis functions of all deeper layers. Their functional approximations thus become more complex with depth. However the roughness of the basis functions at initialization in the deeper layers is lower than that of the shallow layers (since the basis functions are very flat). But, as the network learns, for complex functions most of the roughness is in the later layers as seen in Fig. 3b).

Generalization: Implicit Regularization emerges from Flat Init and Curvature-based Parametrization. The experiments above show that the functional view can give us a new perspective on how depth and parameterization affect the training of neural networks. One of the most useful and perplexing properties of deep neural networks has been that, in contrast to other high

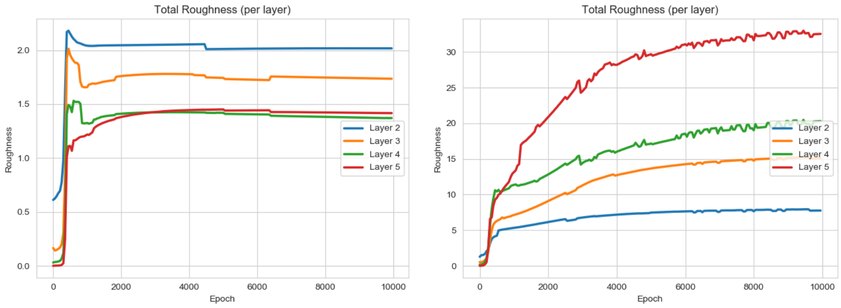

Figure 3: Roughness (summed by layer) during training for a 5 layer ReLU network with 8 units per hidden layer, learning the quadratic function $x^2/2$ (left) and the periodic function $\sin(x)$ (right)

Function	Shallow	Spiky Shallow	Deep	Spiky Deep
Sine	42.95 ± 6.406	157.5 ± 60.27	31.48 ± 7.078	122.0 ± 128.2
Arctan	.01252 ± .07650	2.499 ± 1.257	0.9795 ± 0.9355	32.57 ± 26.10
Sawtooth	156.9 ± 12.45	150.1 ± 61.48	148.1 ± 8.755	198.0 ± 170.9
Cubic	3.608 ± 1.683	136.7 ± 124.1	56.77 ± 98.91	191.6 ± 114.1
Quadratic	3.559 ± 4.553	150.6 ± 49.00	1.741 ± 1.296	46.02 ± 19.42
Exp	.6509 ± .5928	181.1 ± 75.36 ±	1.339 ± 1.292	54.50 ± 37.77

Table 3: Comparison of testing loss (generalization ability) of various network shallow and deep networks with a standard vs ’spiky’ initialization

capacity function approximators, overparameterizing a neural network does not tend to lead to excessive overfitting(Savarese et al., 2019). Where does this generalization power come from? Much recent work(Neyshabur et al., 2018; 2015) have argued that it comes from an implicit regularization inherent in the optimization algorithm itself (i.e. SGD). In contrast, for the case of shallow and deep univariate fully connected ReLU nets, we provide causal evidence that it is due to the specific, very flat CPWL initialization induced by common initialization methods. In order to test this in both shallow and deep ReLU networks, we compare training with the standard flat initialization to a ‘spiky’ initialization.

For a shallow ReLU network, we can test a ’spiky’ initialization by exactly solving for network parameters to generate a given arbitrary CPWL function. This network initialization is then compared against a standard initialization, and trained against a smooth function with a small number of training data points. Note that in a 1D input space we need a small number of training data points to create a situation similar to that of the sparsity caused by high dimensional input, and to allow for testing generalization between data points. The generalization/test set is then just a dense set of points sampled from the input space. We find that both networks fit the training data near perfectly, reaching a global minima of the training loss, but that the ’spiky’ initialization has much worse generalization error (Table 3). Visually, we find that the initial ’spiky’ features of the starting point CPWL representation are preserved in the approximation of the smooth target function(Figs. 4, 6). One final metric tested was roughness, which remained near constant throughout training in both cases, but from a significantly higher initial value in the ’spiky’ initialization case (not shown). For a deep ReLU network, it is more difficult to exactly solve for a ’spiky’ initialization. Instead, we train a network to approximate an arbitrary CPWL function, and call those network parameters the ’spiky’ initialization. Once again, the ’spiky’ initialization has near identical training performance, hitting all data points, but has noticeably worse generalization performance. One noticeably difference is that it is harder to see the exact ’spiky’ properties conserved from initialization to convergence, as breakpoint mobility means that these properties are more mutable.

It appears that generalization performance it not automatically guaranteed by GD, but instead due to the flat initializations which are then preserved by GD. ’Spiky’ initialization also have their (higher) curvature preserved by GD. This idea makes sense, as generalization depends on our target function smoothly varying, and a smooth approximation is promoted by a smooth initialization.

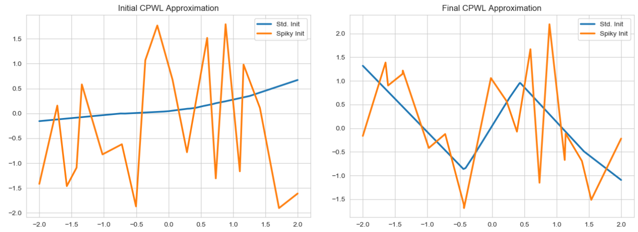

Figure 4: ’Spiky’ (orange) and standard initialization (blue), compared before (left) and after (right) training. Note both cases had similar, very low training set error.

Smoothness in Data Gaps increases with Hidden Units and Decreases with Initial Weight Variance. Our last experiment examines how smoothness (roughness) depends on the number of units, particularly in the case where there are large gaps in the training data. We use a continuous and discontinuous target function (shown in Supp. Fig. 8).We trained shallow ReLU networks with varying width H and initial weight variance σ_w on these training data until convergence, and measured the total roughness of resulting CPWL approximation in the data gaps.

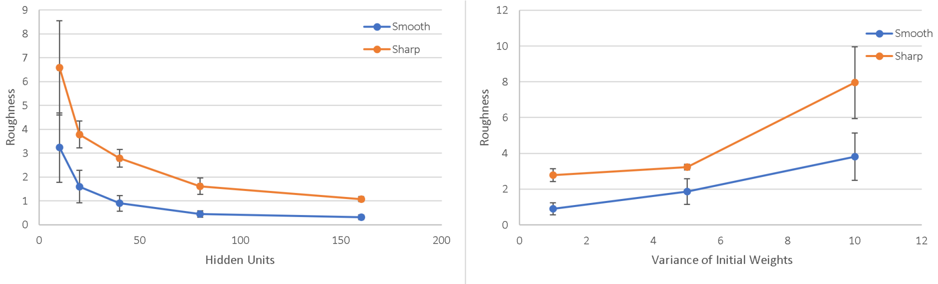

Figure 5: Roughness vs. Width (left) and the variance of the initialization (right) for both data gap cases shown in Fig. 8. Each data point is the result of averaging over 4 trials trained to convergence.

Fig. 5 shows that roughness in the data gaps decreases with width and increases with initial weight variance, confirming our theory. A spiky (and thus rougher) initialization leads to increased roughness at convergence as well, lending support to the idea that roughness in data gaps can be ‘remembered’ from initialization. On the other hand, higher number of pieces spreads out the curvature work over more units, leading to smaller overall roughness. Taken together, our experiments indicate that smooth, flat initialization is partly (if not wholly) responsible for the phenomenon of implicit regularization in univariate fully connected ReLU nets, and that increasing overparameterization leads to even better generalization.

Conclusions. We show in this paper that examining deep networks through the lens of function space can enabled new theoretical and practical insights. We have several interesting findings: the value of depth in deep nets seems to be less about expressivity and more about learnability, enabling GD to finding better quality solutions. The functional view also highlights the importance initialization: a smooth initial approximation seems to encourage a smoother final solution, improving generalization. Fortunately, existing initializations used in practice start with smooth initial approximations, with smoothness increasing with depth. Analyzing the loss surface for a ReLU net in function space gives us a surprisingly simple and transparent view of the phenomenon of overparameterization: it makes clear that increasing width relative to training data size leads w.h.p. to lonely partitions of the data which are global minima. Function space shows us that the mysterious phenomenon of implicit regularization may arise due to a hidden 2nd order differential equation that underlies the ReLU parameterization. In addition, this functional lens suggests new tools, architectures and algorithms. Can we develop tools to help understand how these CPWL functions change across layers or during training? Finally, our analysis shows that bad local minima are often due to breakpoints getting trapped in bad local minima: Can we design new learning algorithms that make *global* moves *in the BDSO parameterization* in order to avoid these local minima?

References

Jayadev Acharya, Ilias Diakonikolas, Jerry Li, and Ludwig Schmidt. Fast algorithms for segmented regression. *arXiv preprint arXiv:1607.03990*, 2016.

Sanjeev Arora, Simon S Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv preprint arXiv:1901.08584*, 2019.

Jushan Bai and Pierre Perron. Estimating and testing linear models with multiple structural changes. *Econometrica*, pp. 47–78, 1998.

Randall Balestriero et al. A spline theory of deep networks. In *International Conference on Machine Learning*, pp. 383–392, 2018.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Boris Hanin and David Rolnick. Deep relu networks have surprisingly few activation patterns. *arXiv preprint arXiv:1906.00904*, 2019.

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

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

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Behnam Neyshabur, Ruslan R Salakhutdinov, and Nati Srebro. Path-sgd: Path-normalized optimization in deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2422–2430, 2015.

Behnam Neyshabur, Zhiyuan Li, Srinadh Bhojanapalli, Yann LeCun, and Nathan Srebro. The role of over-parametrization in generalization of neural networks. 2018.

Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Maithra Raghu, Ben Poole, Jon Kleinberg, Surya Ganguli, and Jascha Sohl Dickstein. On the expressive power of deep neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2847–2854. JMLR. org, 2017.

Pedro Savarese, Itay Evron, Daniel Soudry, and Nathan Srebro. How do infinite width bounded norm networks look in function space? *arXiv preprint arXiv:1902.05040*, 2019.

Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

Appendix

3.1 Experimental Details

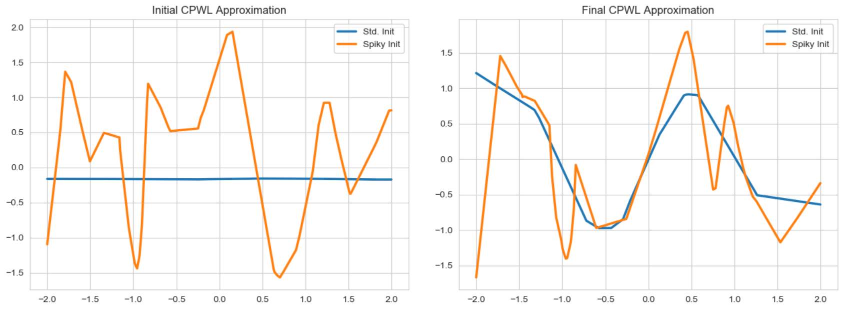

Figure 6: ’Spiky’ (orange) and standard initialization (blue), compared before training (left) and post-training (right) using a deep network

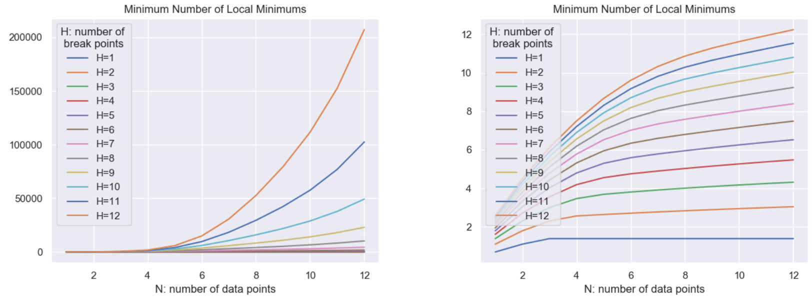

Figure 7: Growth in the (minimum) amount of local minima, as a function of the number of breakpoints and data points. Right plot is identical, but with log scaling

3.1.1 Uniform Initialization

Trained on a shallow, 21 unit FC ReLU network. Trained on function over the interval [-2,2]. Learning rate = 5e-5, trained via GD over 10000 epochs. Compared against pytorch default of He initialization. Training data sampled uniformly every .01 of the target interval. Each experiment was run 5 times, with results reported as mean ± standard deviation. Breakpoints y values were taken from the original standard initialization for the uniform initialization plus a small random noise term N(0,.01), making initial condition within the target interval nearly identical.

3.1.2 Roughness by Layer Plots

Trained on a deep, 5 layer network, with 4 hidden layers of width 8. Trained on function over the interval [-2,2]. Learning rate = 1e-4, trained via GD over 10000 epochs, with roughness measured every 50 epochs. Roughness per layer was summed over all units within that layer.

3.1.3 Spiky Initialization Plots

Shallow version trained on a 21 unit FC ReLU Network. Deep version trained on a deep, 5-layer network with 4 hidden layers of width 8. In both cases, the ’spiky’ initialization was a 20 - breakpoint CPWL function, with $y_n \sim \text{Uniform}([-2, 2])$. In the deep case, the spiky model was initialized with the same weights as the non-spiky model, and then pre-trained for 10,000 epochs to fit the CPWL. After that, gradient descent training proceeded on both models for 20,000 epochs, with all training having learning rate 1e-4. Training data was 20 random points in the range [-2,2], while the testing data (used to measure generalization) was spaced uniformly at every $\Delta x = .01$ of the target interval of the target function.

In the shallow case, there was no pre-training, as the 'spiky' model was directly set to be equal to the CPWL. In the shallow model, training occurred for 20,000 epochs. All experiment were run over 5 trials, and values in table are reported as mean ± standard deviation. Base shallow learning rate was 1e-4 using gradient descent method, with learning rate divided by 5 for the spiky case due to the initialization method generating larger weights. Despite differing learning rates, both models had similar training loss curves and similar final training loss values, e.g. for sine, final training loss was .94 for spiky and 1.02 for standard. Functions used were $\sin(x), \arctan(x)$, a sawtooth function from [-2,2] with minimum value of -1 at the endpoints, and 4 peaks of maximum value 1, cubic $\frac{x^3}{4} + \frac{x^2}{2} - \frac{x}{2}$, quadratic $\frac{x^2}{2}$, and $\exp(.5x)$ Note GD was chosen due to the strong theoretical focus of this paper - similar results were obtained using ADAM optimizer, in which case no differing learning rates were necessary.

3.2 Breakpoints Induced by Deep Networks

We used networks with a total of $H = 40$ hidden units, spread over $L \in \{1, 2, 3, 4, 5\}$ hidden layers. Training data consiste of uniform samples of function over the interval $x \in [-3, 3]$. Learning rate $= 5 \cdot 10^{-5}$, trained via GD over $25,000$ epochs. The target functions tested were $\sin(\pi x)$, a 5-piece polynomial with maximum value of 2 in the domain $[-3, 3]$, a sawtooth with period 3 and amplitude 1, $\arctan(x)$, $\exp(x)$, and $\frac{1}{9}x^2$. Each value in the table was the average of 5 trials.

3.3 Breakpoint Mobility in Deep Networks

We use a deep, 6-layer network, with 5 hidden layers of width 8. Training data consists of the 'smooth' and 'sharp' functions over the interval $x \in [-3, 3]$. Learning rate = 5e-5, trained via GD until convergence, where convergence was defined as when the loss between two epochs changed by less than 10^{-8}. Breakpoints were calculated every 50 epochs. The velocity of breakpoints was then calculated, and the values seen in the figure are normalized to the velocity of the first layer.

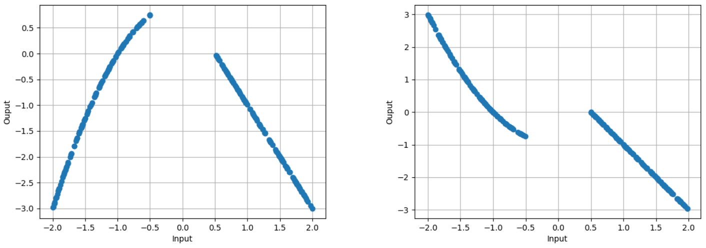

Figure 8: Training data sampled from two ground truth functions, one smoothly (left) and the other sharply (right) discontinuous, each with a data gap at $[-0.5, 0.5]$.

4 More details on Breakpoints for Deep ReLU Nets

Each neuron of the second hidden layer receives as input the result of a CPWL function $z_i^{(2)}(x)$ as defined above. The output of this function is then fed through a ReLU, which has two implications: first, every zero crossing of $z_i^{(2)}$ is a breakpoint of $x_i^{(2)}$; second, any breakpoints $\beta_j^{(1)}$ of $z_i^{(2)}$ such that $z_i^{(2)}(\beta_j^{(1)}) < 0$ will not be breakpoints of $x_i^{(2)}$. Importantly, the number of breakpoints in $g_\theta(x)$ is now a function of the parameters θ, rather than equal to fixed H as in the $L = 1$ case; in other words, breakpoints can be dynamically created and annihilated throughout training. This fact will have dramatic implications when we explore how gradient descent optimizes breakpoints in order to model curvature in the training data (see Sec 3). But first, due to complexities of depth, we must carefully formalize the notion of a breakpoint for a deep network.

Definition 1. $\beta_i^{(\ell)}$ *is a* breakpoint induced by neuron i in layer ℓ *if* $z_i^{(\ell)}(\beta_i^{(\ell)}) = 0$*. Since the function* $z_i^{(\ell)}(\cdot)$ *is nonlinear, neuron* i *may induce multiple breakpoints, which we denote* $\beta_{i,k}^{(\ell)}$*. A breakpoint* $\beta_{i,k}^{(\ell)}$ *is* active *if there exists some path* π *through neuron* i *such that for all other neurons* $j \neq i \in \pi$*,* $z_j^{(\ell(j))} > 0$*, i.e.* $\widehat{a}_j(x) = 1$*. If two neurons* i *and* j *in layers* ℓ *and* ℓ' *induce the same breakpoint(s),* $\beta_{i,k}^{(\ell)} = \beta_{j,k'}^{(\ell')}$*, then both are referred to as* degenerate *breakpoints.*

Let $\widehat{a}_\pi(x) = \prod_{i \in \pi} \widehat{a}_i$. Then, $\beta_i^{(\ell)}$ is active iff there exists some path π such that $\widehat{a}_\pi$ is discontinuous at $x = \beta_i^{(\ell)}$. Thus, $g_\theta(x)$ is non-differentiable at x if $x = \beta_i^{(\ell)}$ for some (ℓ, i). If no degenerate breakpoints exist, then the converse also holds. (If there do exist degenerate breakpoints $\beta_i^{(\ell)}$ and $\beta_j^{(\ell')}$, then it is possible that $\mu_i^{(\ell)} = -\mu_j^{(\ell')}$, i.e. the changes in slope cancel out and $g_\theta(x)$ remains linear and differentiable.)

5 Proofs of Theoretical Results

5.1 Reparametrization from ReLu Network to Piecewise Linear Function

Proof of Eqn (1)**-(3):**

$$
\begin{aligned}
\hat{f}_{\theta,H}(x) &= \sum_{i=1}^{H} v_i \phi(w_i x + b_i) \\
&= \sum_{i=1}^{H} v_i (w_i x + b_i) [\![w_i x + b_i > 0]\!] \\
&= \sum_{i=1}^{H} v_i w_i (x - \beta_i) \begin{cases} [\![x > \beta_i]\!] & w_i > 0 \\ [\![x < \beta_i]\!] & w_i < 0 \end{cases} \quad \text{where } \beta_i \triangleq -\frac{b_i}{w_i} \\
&= \sum_{i=1}^{H} \mu_i (x - \beta_i) \begin{cases} [\![x > \beta_i]\!] & w_i > 0 \\ [\![x < \beta_i]\!] & w_i < 0 \end{cases} \quad \text{where } \mu_i \triangleq v_i w_i \\
&= \sum_{p=0}^{H} \left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} (x - \beta_{(i)}) + \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} (x - \beta_{(i)}) \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!] \quad \text{where } \beta_{(0)} \triangleq -\infty,\ \beta_{(H+1)} \triangleq \infty \\
&= \sum_{p=0}^{H} \left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \left(\mu_{(i)} x - \mu_{(i)} \beta_{(i)} \right) + \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \left(\mu_{(i)} x - \mu_{(i)} \beta_{(i)} \right) \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!] \\
&= \sum_{p=0}^{H} \left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} x - \sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} \beta_{(i)} + \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} x - \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} \beta_{(i)} \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!] \\
&= \sum_{p=0}^{H} \left(\left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} \right) x - \sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} \beta_{(i)} + \left(\sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} \right) x - \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} \beta_{(i)} \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!] \\
&= \sum_{p=0}^{H} \left(\underbrace{\left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} + \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} \right)}_{\overline{\mu}_{(p)}} x - \underbrace{\left(\sum_{\substack{i=1 \\ w_{(i)}>0}}^{p} \mu_{(i)} \beta_{(i)} + \sum_{\substack{i=p+1 \\ w_{(i)}<0}}^{H} \mu_{(i)} \beta_{(i)} \right)}_{\overline{x}^*_{(p)}} \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!] \\
&= \sum_{p=0}^{H} \left(\overline{\mu}_{(p)} x - \overline{x}^*_{(p)} \right) [\![\beta_{(p)} \leq x < \beta_{(p+1)}]\!]
\end{aligned}
$$

5.2 Random Initialization in Function Space

Proof of Theorem 1(a-b). Suppose (b_i, w_i, v_i) are initialized independently from a distribution with density $f_{B,W,V}(b_i, w_i, v_i)$. Then, we can derive the density of (β_i, μ_i) by considering the invertable continuous transformation given by $(\beta_i, \mu_i, u) = g(b_i, w_i, v_i) = (b_i/w_i, v_i|w_i|, w_i)$, where $g^{-1}(\beta_i, \mu_i, u) = (\beta_i u, u, \mu_i/|u|)$. The density of (β_i, μ_i, u) is given by $f_{B,M,V}(\beta_i u, u, \mu_i/|u|)|J|$, where J is the Jacobian determinant of g^{-1}. Then, we have $J = -\operatorname{sgn} w_i$ and $|J| = 1$. The density of (β_i, μ_i) is then derived by integrating out the dummy variable u: $f_{\beta,\mu}(\beta_i, \mu_i) = \int_{-\infty}^{\infty} f_{B,W,V}(\beta_i u, u, \frac{\mu_i}{u})\,\mathrm{d}u$. If (b_i, w_i, v_i) are independent, this expands to $\int_{-\infty}^{\infty} f_B(\beta_i u) f_W(u) f_V(\frac{\mu_i}{u})\,\mathrm{d}u$. See below for next parts of proof for the Gaussian and Uniform cases separately.

5.3 Gaussian Initialization in Function

Proof of Theorem 1(a). Using the preliminary results above, we now specialize to the case of Gaussian initialization:

$$
\begin{aligned}
(\beta, \mu, U) = g(B, W, V) &= (B/W, VW, W) \\
g^{-1}(\beta, \mu, U) &= (\beta U, U, \mu/U) \\
J &= \begin{vmatrix} \frac{\partial B}{\partial \beta} & \frac{\partial B}{\partial \mu} & \frac{\partial B}{\partial U} \\ \frac{\partial W}{\partial \beta} & \frac{\partial W}{\partial \mu} & \frac{\partial W}{\partial U} \\ \frac{\partial V}{\partial \beta} & \frac{\partial V}{\partial \mu} & \frac{\partial V}{\partial U} \end{vmatrix} = \begin{vmatrix} U & 0 & \beta \\ 0 & 0 & 1 \\ 0 & \frac{1}{U} & -\frac{\mu}{U^2} \end{vmatrix} = 0 + 0 + 0 - 0 - 0 - 1 = -1 \\
f_{\beta,\mu,U}(\beta, \mu, u) &= f_{B,W,V}(\beta u, u, \mu/u) \\
f_{\beta,\mu}(\beta, \mu) &= \int f_{B,W,V}(\beta u, u, \mu/u) du \\
&= \int f_B(\beta u) f_W(u) f_V(\mu/u) du \\
&= \int \frac{1}{\sqrt{2\pi\sigma_b^2}} e^{-\frac{(\beta u)^2}{2\sigma_b^2}} \frac{1}{\sqrt{2\pi\sigma_w^2}} e^{-\frac{u^2}{2\sigma_w^2}} \frac{1}{\sqrt{2\pi\sigma_v^2}} e^{-\frac{(\mu/u)^2}{2\sigma_v^2}} du \\
\text{(sympy)} &= \begin{cases} \frac{\exp\left[-\frac{\mu\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\pi\sigma_v\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}} & \mu > 0 \\ \text{unknown} & \text{otherwise} \end{cases}
\end{aligned}
$$

but $f_{\beta,\mu}$ has symmetric marginals, so it should be symmetric in μ, so

$$
= \frac{\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\pi\sigma_v\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}
$$

Marginalizing out μ from this density in Sympy returns the appropriate $f_\beta(\beta)$ from above (Sympy cannot compute the other marginal).

$$
\begin{aligned}
\int_{-\infty}^{\infty} \frac{\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2\beta^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\pi\sigma_v\sqrt{\sigma_b^2+\sigma_w^2x^2}} dx &= \frac{1}{2\pi\sigma_v} \int_{-\infty}^{\infty} \frac{\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2\beta^2}}{\sigma_b\sigma_v\sigma_w}\right]}{\sqrt{\sigma_b^2+\sigma_w^2x^2}} dx \\
\left(\phi(\beta) = \frac{\beta}{\sigma_w}\right) \qquad &= \frac{1}{2\pi\sigma_v \underbrace{\sigma_w}_{\phi'(\beta)}} \int_{-\infty}^{\infty} \frac{\exp\left[\frac{-|\mu|\sqrt{\sigma_b^2+\beta^2}}{\sigma_b\sigma_v\sigma_w}\right]}{\sqrt{\sigma_b^2+\beta^2}} dx
\end{aligned}
$$

from [Table], p. 396, we have

$$K_0(ab) = \int_0^\infty \frac{\exp\left(-a\sqrt{\beta^2+b^2}\right)}{\sqrt{\beta^2+b^2}}\,\mathrm{d}\beta \qquad [\operatorname{Re} a > 0, \operatorname{Re} b > 0]$$

$$\text{(integrand is even in } \beta\text{)} = \frac{1}{2}\int_{-\infty}^\infty \frac{\exp\left(-a\sqrt{\beta^2+b^2}\right)}{\sqrt{\beta^2+b^2}}\,\mathrm{d}\beta \qquad [\operatorname{Re} a > 0, \operatorname{Re} b > 0]$$

applying this with $a = \frac{|\mu|}{\sigma_b\sigma_v\sigma_w}$ and $b = \sigma_b$,

$$\frac{1}{2\pi\sigma_v\sigma_w}\int_{-\infty}^\infty \frac{\exp\left[\frac{-|\mu|\sqrt{\sigma_b^2+\beta^2}}{\sigma_b\sigma_v\sigma_w}\right]}{\sqrt{\sigma_b^2+\beta^2}}\,\mathrm{d}\beta = \frac{1}{\pi\sigma_v\sigma_w}K_0\left(\frac{|\mu|}{\sigma_v\sigma_w}\right)$$

as desired.

$$f_\mu(\mu|\beta) = \frac{\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\sigma_b\sigma_v\sigma_w} = \text{Laplace}\left(\mu; 0, \frac{\sigma_b\sigma_v\sigma_w}{\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}\right)$$

$$f_\beta(\beta|\mu) = \frac{\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\pi\sigma_v\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}\frac{1}{\frac{1}{\pi\sigma_v\sigma_w}K_0\left(\frac{|\mu|}{\sigma_v\sigma_w}\right)}$$

$$= \frac{\exp\left[-\frac{|\mu|\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}{\sigma_b\sigma_v\sigma_w}\right]}{2\sqrt{\sigma_b^2+\sigma_w^2(\beta)^2}}\frac{\sigma_w}{K_0\left(\frac{|\mu|}{\sigma_v\sigma_w}\right)}. \ \square$$

5.4 Uniform Initialization in Function Space

Proof of Theorem 1(b). Using the preliminary results above, we now specialize to the case of Uniform initialization:

$$\begin{aligned}
w_i &\sim U[-a_w, a_w] \\
b_i &\sim U[-a_b, a_b] \\
v_i &\sim U[-a_v, a_v] \\
f_{\beta,\mu}(\beta,\mu) &= \int_{-a_w}^{a_w} f_B(\beta u) f_W(u) f_V(\mu/u)\,\mathrm{d}u \\
&= \int_{-a_w}^{a_w} \frac{1}{2a_b}\llbracket -a_b \le \beta u \le a_b \rrbracket \frac{1}{2a_w}\llbracket -a_w \le u \le a_w \rrbracket \frac{1}{2a_v}\llbracket -a_v \le \mu/u \le a_v \rrbracket\,\mathrm{d}u \\
&= \int_{-a_w}^{a_w} \frac{1}{2a_b}\llbracket -a_b/|\beta| \le u \le a_b/|\beta| \rrbracket \frac{1}{2a_w}\llbracket -a_w \le u \le a_w \rrbracket \frac{1}{2a_v}\llbracket u \le -|\mu|/a_v \vee u \ge |\mu|/a_v \rrbracket\,\mathrm{d}u \\
&= \int_{-a_w}^{a_w} \frac{1}{8a_ba_wa_v}\llbracket -\min\{a_b/|\beta|, a_w\} \le u \le -|\mu|/a_v \vee |\mu|/a_v \le u \le \min\{a_b/|\beta|, a_w\} \rrbracket \llbracket |\mu| \le a_b a \\
&= \frac{\llbracket |\mu| \le a_ba_v/|\beta| \rrbracket}{8a_ba_wa_v}\int_{-a_w}^{a_w} \llbracket -\min\{a_b/|\beta|, a_w\} \le u \le -|\mu|/a_v \vee |\mu|/a_v \le u \le \min\{a_b/|\beta|, a_w\} \rrbracket\,\mathrm{d}u \\
&= \frac{\llbracket |\mu| \le a_ba_v/|\beta| \rrbracket}{4a_ba_wa_v}\int_0^{a_w} \llbracket |\mu|/a_v \le u \le \min\{a_b/|\beta|, a_w\} \rrbracket\,\mathrm{d}u \\
&= \frac{\llbracket |\mu| \le a_ba_v/|\beta| \rrbracket}{4a_ba_wa_v}\left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v\right)\llbracket -a_wa_v \le \mu \le a_wa_v \rrbracket \\
f_\mu(\mu) &= \int f_{\beta,\mu}(\beta,\mu)\,\mathrm{d}\beta \\
&= \int_{-\infty}^\infty \frac{\llbracket |\mu| \le a_ba_v/|\beta| \rrbracket}{4a_ba_wa_v}\left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v\right)\llbracket -a_wa_v \le \mu \le a_wa_v \rrbracket\,\mathrm{d}\beta
\end{aligned}$$

$$
\begin{aligned}
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} \int_{-\infty}^{\infty} [\![|\mu| \leq a_b a_v/|\beta|]\!] \left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v\right) \mathrm{d}\beta \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \int_{0}^{\infty} [\![|\mu| \leq a_b a_v/\beta]\!] \left(\min\{a_b/\beta, a_w\} - |\mu|/a_v\right) \mathrm{d}\beta \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \int_{0}^{\infty} [\![\beta \leq a_b a_v/|\mu|]\!] \left(\min\{a_b/\beta, a_w\} - |\mu|/a_v\right) \mathrm{d}\beta \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \int_{0}^{a_b a_v/|\mu|} \min\{a_b/\beta, a_w\} - |\mu|/a_v \,\mathrm{d}\beta \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \left(\int_{0}^{\min\{a_b a_v/|\mu|, a_b/a_w\}} a_w - |\mu|/a_v \,\mathrm{d}\beta + \int_{\min\{a_b a_v/|\mu|, a_b/a_w\}}^{a_b a_v/|\mu|} a_b/\beta - |\mu \right. \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \left(\int_{0}^{a_b/a_w} a_w - |\mu|/a_v \,\mathrm{d}\beta + \int_{a_b/a_w}^{a_b a_v/|\mu|} a_b/\beta - |\mu|/a_v \,\mathrm{d}\beta \right) \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \left((a_b/a_w)(a_w - |\mu|/a_v) + \int_{a_b/a_w}^{a_b a_v/|\mu|} a_b/\beta \,\mathrm{d}\beta - \int_{a_b/a_w}^{a_b a_v/|\mu|} |\mu|/a_v \,\mathrm{d}\beta \right) \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \left((a_b/a_w)(a_w - |\mu|/a_v) + \int_{a_b/a_w}^{a_b a_v/|\mu|} a_b/\beta \,\mathrm{d}\beta - (|\mu|/a_v)(a_b a_v/|\mu| - a_b/ \right. \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2 \left((a_b/a_w)(a_w - |\mu|/a_v) + a_b \log \frac{a_b a_v/|\mu|}{a_b/a_w} - (|\mu|/a_v)(a_b a_v/|\mu| - a_b/a_w \right. \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{4a_b a_w a_v} 2a_b \log \frac{a_b a_v/|\mu|}{a_b/a_w} \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{2a_w a_v} \log \frac{a_b a_v/|\mu|}{a_b/a_w} \\
&= \frac{[\![-a_w a_v \leq \mu \leq a_w a_v]\!]}{2a_w a_v} \log \frac{a_w a_v}{|\mu|}
\end{aligned}
$$

Sanity check:

$$
\begin{aligned}
&\int_{-a_w a_v}^{a_w a_v} \frac{1}{2a_w a_v} \log \frac{a_w a_v}{|\mu|} \,\mathrm{d}\mu \\
&= \frac{1}{a_w a_v} \int_{0}^{a_w a_v} \log \frac{a_w a_v}{\mu} \,\mathrm{d}\mu \\
&= 1 \\
\text{Var}[\mu] = \mathbb{E}[\mu^2] &= \int_{-a_w a_v}^{a_w a_v} \mu^2 \frac{1}{2a_w a_v} \log \frac{a_w a_v}{|\mu|} \,\mathrm{d}\mu \\
&= \frac{1}{a_w a_v} \int_{0}^{a_w a_v} \mu^2 \log \frac{a_w a_v}{\mu} \,\mathrm{d}\mu \\
&= \frac{a_w^3 a_v^3}{9} \\
\text{Kurtosis}(\mu) = \frac{\mathbb{E}[\mu^4]}{\text{Var}[\mu]^2} &= \frac{\frac{a_w^5 a_v^5}{25}}{\frac{a_w^6 a_v^6}{81}} \\
&= \frac{81}{25 a_w a_v} \\
f_\beta(\beta) &= \int f_{\beta,\mu}(\beta, \mu) \,\mathrm{d}\mu \\
&= \int_{-\infty}^{\infty} \frac{[\![|\mu| \leq a_b a_v/|\beta|]\!]}{4a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v\right) [\![-a_w a_v \leq \mu \leq a_w a_v]\!] \,\mathrm{d}\mu \\
&= \int_{-a_w a_v}^{a_w a_v} \frac{[\![|\mu| \leq a_b a_v/|\beta|]\!]}{4a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v\right) \mathrm{d}\mu
\end{aligned}
$$

$$
\begin{aligned}
&= \frac{1}{2a_b a_w a_v} \int_0^{a_w a_v} \llbracket \mu \leq a_b a_v / |\beta| \rrbracket \left(\min\{a_b/|\beta|, a_w\} - \mu / a_v \right) \mathrm{d}\mu \\
&= \frac{1}{2a_b a_w a_v} \left(\int_0^{a_w a_v} \llbracket \mu \leq a_b a_v / |\beta| \rrbracket \min\{a_b/|\beta|, a_w\} \,\mathrm{d}\mu - \int_0^{a_w a_v} \llbracket \mu \leq a_b a_v / |\beta| \rrbracket \mu / a_v \,\mathrm{d}\mu \right) \\
&= \frac{1}{2a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} \int_0^{a_w a_v} \llbracket \mu \leq a_b a_v / |\beta| \rrbracket \,\mathrm{d}\mu - \frac{1}{a_v} \int_0^{a_w a_v} \llbracket \mu \leq a_b a_v / |\beta| \rrbracket \mu \,\mathrm{d}\mu \right) \\
&= \frac{1}{2a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} \min\{a_w a_v, a_b a_v / |\beta|\} - \frac{1}{a_v} \int_0^{\min\{a_w a_v, a_b a_v / |\beta|\}} \mu \,\mathrm{d}\mu \right) \\
&= \frac{1}{2a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} \min\{a_w a_v, a_b a_v / |\beta|\} - \frac{1}{2a_v} \left(\min\{a_w a_v, a_b a_v / |\beta|\} \right)^2 \right) \\
&= \frac{1}{2a_b a_w a_v} \left(a_v \left(\min\{a_b/|\beta|, a_w\} \right)^2 - \frac{1}{2a_v} \left(a_v \min\{a_w, a_b/|\beta|\} \right)^2 \right) \\
&= \frac{1}{2a_b a_w a_v} \left(a_v \left(\min\{a_b/|\beta|, a_w\} \right)^2 - \frac{a_v}{2} \left(\min\{a_w, a_b/|\beta|\} \right)^2 \right) \\
&= \frac{1}{4a_b a_w} \left(\min\{a_b/|\beta|, a_w\} \right)^2
\end{aligned}
$$

Sanity check:

$$
\begin{aligned}
&\int_{-\infty}^{\infty} \frac{1}{4a_b a_w} \left(\min\{a_b/|\beta|, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{4a_b a_w} \int_{-\infty}^{\infty} \left(\min\{a_b/|\beta|, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{2a_b a_w} \int_0^{\infty} \left(\min\{a_b/\beta, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{2a_b a_w} \left(\int_0^{a_b/a_w} a_w^2 \,\mathrm{d}\beta + a_b^2 \int_{a_b/a_w}^{\infty} 1/\beta^2 \,\mathrm{d}\beta \right) \\
&= \frac{1}{2a_b a_w} \left(a_b a_w + a_b a_w \right) \\
&= 1 \\
\mathbb{E}[\beta^2] &= \int_{-\infty}^{\infty} \frac{\beta^2}{4a_b a_w} \left(\min\{a_b/|\beta|, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{4a_b a_w} \int_{-\infty}^{\infty} \beta^2 \left(\min\{a_b/|\beta|, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{2a_b a_w} \int_0^{\infty} \beta^2 \left(\min\{a_b/\beta, a_w\} \right)^2 \mathrm{d}\beta \\
&= \frac{1}{2a_b a_w} \left(\int_0^{a_b/a_w} \beta^2 a_w^2 \,\mathrm{d}\beta + a_b^2 \int_{a_b/a_w}^{\infty} 1 \,\mathrm{d}\beta \right) \\
&= \infty \\
f_\mu(\mu|\beta) &= \frac{\frac{\llbracket |\mu| \leq a_b a_v / |\beta| \rrbracket}{4a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v \right) \llbracket -a_w a_v \leq \mu \leq a_w a_v \rrbracket}{\frac{1}{4a_b a_w} \left(\min\{a_b/|\beta|, a_w\} \right)^2} \\
&= \frac{\frac{\llbracket |\mu| \leq a_b a_v / |\beta| \rrbracket \llbracket -a_w a_v \leq \mu \leq a_w a_v \rrbracket}{a_v} \left(\min\{a_b/|\beta *|, a_w\} - |\mu|/a_v \right)}{\left(\min\{a_b/|\beta|, a_w\} \right)^2} \\
&= \frac{\llbracket |\mu| \leq a_v \min\{a_b/|\beta|, a_w\} \rrbracket}{a_v \min\{a_b/|\beta|, a_w\}} \left(1 - \frac{|\mu|}{a_v \min\{a_b/|\beta|, a_w\}} \right) \\
f_X(\beta|\mu) &= \frac{\llbracket |\mu| \leq a_b a_v / |\beta| \rrbracket}{4a_b a_w a_v} \left(\min\{a_b/|\beta|, a_w\} - |\mu|/a_v \right) \llbracket -a_w a_v \leq \mu \leq a_w a_v \rrbracket \Bigg(\frac{\llbracket -a_w a_v \leq \mu \leq a_w a_v \rrbracket}{2a_w a_v} \log
\end{aligned}
$$

This completes the proof. □

Remarks. Note that the marginal distribution on μ_i is the distribution of a product of two independent random variables, and the marginal distribution on β_i is the distribution of the ratio of two random variables. For the Gaussian case, the marginal distribution on μ_i is a symmetric distribution with variance $\sigma_v^2\sigma_w^2$ and excess Kurtosis of 6. For the Uniform case, the marginal distribution of β_i is a symmetric distribution with no finite higher moments. The marginal distribution of μ_i is a symmetric distribution with bounded support and variance $\frac{2a_w^3 a_v^3}{9}$ and excess Kurtosis of $\frac{81}{50a_w a_v} - 3$. The conditional distribution of μ_i given β_i is a symmetric distribution with bounded support and variance $\frac{(a_v \min\{a_b/|\beta_i|, a_w\})^2}{6}$ and excess Kurtosis of $-\frac{3}{5}$.

5.5 Roughness of Random Initialization

Proof of Theorem 2. Using the moments of the delta-slope distribution computed in Theorem 1 and above, we can compute:

$$\begin{aligned}
\mathbb{E}[\rho_0] &= \sum_{i=1}^{H} \mathbb{E}[\mu_{i0}^2] = \sum_{i=1}^{H} \text{Var}[\mu_{i0}] + \mathbb{E}[\mu_{i0}]^2 = H(\sigma_v\sigma_w)^2 = 4H/(H+1)^2 \\
\mathbb{E}[\mu_{i0}^4] &= 9(\sigma_v\sigma_w)^4 \\
\text{Var}[\mu_{i0}] &= \mathbb{E}[\mu_{i0}^2] = (\sigma_v\sigma_w)^2 \\
\text{Var}[\mu_{i0}^2] &= \mathbb{E}[\mu_{i0}^4] - \mathbb{E}[\mu_{i0}^2]^2 = 9(\sigma_v\sigma_w)^4 - (\sigma_v\sigma_w)^4 = 8(\sigma_v\sigma_w)^4 = 128/(H+1)^4 \\
\Pr[\rho_0 - 4/H \geq \lambda] &\leq \frac{1}{1 + \frac{\lambda^2 H^3}{128}} \quad \text{[Cantelli's Theorem].} \ \square
\end{aligned}$$

5.6 Loss Surface in Function Space

Proof of Theorem 3:

Proof. If, for all i, $\hat{f}(\cdot;\theta_{BDSO})|_{\Pi_i}$ is an OLS fit of the data Π_i, then we must have $\langle \hat{\boldsymbol{\epsilon}}_{\Pi_i}, \Pi_i \rangle = 0$, where $\hat{\boldsymbol{\epsilon}}_{\Pi_i}$ is the residual for Π_i. Similarly, we must have that the net residual $\langle \hat{\boldsymbol{\epsilon}}_{\Pi_i}, \mathbf{1} \rangle = 0$.

Next, consider, for any neuron j, the vector $\hat{\mathbf{a}}_j$. If j is right-facing, $\hat{\mathbf{a}}_j = (0,\ldots,0,1,\ldots,1)$, where the transition from 0s to 1s corresponds to the data index n where $x_n > \beta_j$; if j is left-facing, a 1-to-0 transition occurs at n. Thus, $\hat{\mathbf{a}}_j$ is constant for $n \in \Pi_i$, as the boundaries of Π_i correspond to breakpoints β_i and β_{i+1}. Noting that these inner products are just sums of products, we have that, for any neuron j, $\langle \hat{\boldsymbol{\epsilon}} \odot \hat{\mathbf{a}}_j, \mathbf{x} \rangle$ can be decomposed into a sum $\sum_{\Pi_i} \langle \hat{\boldsymbol{\epsilon}}_{\Pi_i}, \Pi_i \rangle = 0$, where the sum is over the pieces on the active side of j. Similarly, $\langle \hat{\boldsymbol{\epsilon}} \odot \hat{\mathbf{a}}_j, \mathbf{1} \rangle = \sum_{\Pi_i} \langle \hat{\boldsymbol{\epsilon}}_{\Pi_i}, \mathbf{1} \rangle = 0$.

Applying Theorem 5, we see that $\frac{\mathrm{d}\beta_j}{\mathrm{d}t} = \frac{\mathrm{d}\mu_j}{\mathrm{d}t} = 0$ for all j, and so θ_{BDSO} is a critical point of $\tilde{\ell}(\theta_{BDSO})$. □

5.7 Dynamics in Function Space (Breakpoints and Delta-Slopes)

Proof of Theorem 5: Computing the time derivatives of the BDSO parameters and using the loss gradients of the loss with respect to the NN parameters gives us:

$$\begin{aligned}
\frac{\partial \ell(\theta_{NN})}{\partial w_i} &= v_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{x} \rangle \\
\frac{\partial \ell(\theta_{NN})}{\partial v_i} &= \langle \hat{\boldsymbol{\epsilon}}, \sigma(w_i \mathbf{x} + b_i \mathbf{1}) \rangle = \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, w_i \mathbf{x} + b_i \mathbf{1} \rangle = w_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{x} \rangle + b_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{1} \rangle \\
\frac{\partial \ell(\theta_{NN})}{\partial b_i} &= v_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{1} \rangle \\
\frac{\mathrm{d}\beta_i(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\left(-\frac{b_i(t)}{w_i(t)}\right) \\
&= -\frac{w_i(t)\frac{\mathrm{d}b_i(t)}{\mathrm{d}t} - b_i(t)\frac{\mathrm{d}w_i(t)}{\mathrm{d}t}}{w_i(t)^2}
\end{aligned}$$

$$
\begin{aligned}
&= -\frac{w_i(t)(-\frac{\partial \ell(\theta_{NN})}{\partial b_i(t)}) - b_i(t)(-\frac{\partial \ell(\theta_{NN})}{\partial w_i(t)})}{w_i(t)^2} \\
&= \frac{w_i(t)\frac{\partial \ell(\theta_{NN})}{\partial b_i(t)} - b_i(t)\frac{\partial \ell(\theta_{NN})}{\partial w_i(t)}}{w_i(t)^2} \\
&= \frac{w_i(t)v_i(t)\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1}\rangle - b_i(t)v_i(t)\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{x}\rangle}{w_i(t)^2} \\
&= \frac{v_i(t)\,\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), w_i(t)\mathbf{1} - b_i(t)\mathbf{x}\rangle}{w_i(t)^2} \\
&= \frac{v_i(t)}{w_i(t)}\left\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1} - \frac{b_i(t)}{w_i(t)}\mathbf{x} \right\rangle \\
&= \frac{v_i(t)}{w_i(t)}\left\langle \underbrace{\hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t),}_{\text{relevant residuals}} \mathbf{1} + \beta_i(t)\mathbf{x} \right\rangle \\
&= \frac{v_i(t)}{w_i(t)}\,[\underbrace{\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{1}\rangle}_{\text{net relevant residual}} + \beta_i(t)\,\underbrace{\langle \hat{\boldsymbol{\epsilon}}(t) \odot \mathbf{a}_i(t), \mathbf{x}\rangle}_{\text{correlation}}] \\
\frac{\mathrm{d}\mu_i(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t} w_i v_i \\
&= \frac{\mathrm{d}w_i}{\mathrm{d}t} v_i + w_i \frac{\mathrm{d}v_i}{\mathrm{d}t} \\
&= -\frac{\partial \ell(\theta_{NN})}{\partial w_i} v_i - w_i \frac{\partial \ell(\theta_{NN})}{\partial v_i} \\
&= -v_i^2 \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{x}\rangle - w_i^2 \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{x}\rangle - w_i b_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{1}\rangle \\
&= -(v_i^2 + w_i^2)\langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{x}\rangle - w_i b_i \langle \hat{\boldsymbol{\epsilon}} \odot \mathbf{a}_i, \mathbf{1}\rangle
\end{aligned}
$$

(From Du et al. 1)

$$
\begin{aligned}
\frac{\mathrm{d}u_i}{\mathrm{d}t} &= \frac{\mathrm{d}f(\mathbf{W}(t), \mathbf{a}(t), \mathbf{x}_i)}{\mathrm{d}t} \\
\text{(chain rule)} &= \sum_{r=1}^{\mu} \langle \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\mathrm{d}\mathbf{w}_r(t)}{\mathrm{d}t} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial a_r(t)} \frac{\mathrm{d}a_r(t)}{\mathrm{d}t} \\
&= \sum_{r=1}^{\mu} \langle \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\partial \ell(\theta)}{\partial \mathbf{w}_r(t)} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial a_r(t)} \frac{\partial \ell(\theta)}{\partial a_r(t)} \\
&= \sum_{r=1}^{\mu} \langle \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \sum_{j=1}^{n} (y_j - u_j) \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_j)}{\partial \mathbf{w}_r(t)} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial a_r(t)} \sum_{j=1}^{n} (y_j - u_j) \partial \\
&= \sum_{j=1}^{n} (y_j - u_j) \left(\sum_{r=1}^{\mu} \langle \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_j)}{\partial \mathbf{w}_r(t)} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial a_r(t)} \frac{\partial f(\mathbf{W}(t), \mathbf{a},}{\partial a_r(t)} \right. \\
&\triangleq \sum_{j=1}^{n} (y_j - u_j)(\mathbf{H}_{ij}(t) + \mathbf{G}_{ij}(t)) \\
\mathbf{H}_{ij}(t) &\triangleq \sum_{r=1}^{\mu} \langle \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_j)}{\partial \mathbf{w}_r(t)} \rangle \\
&= \sum_{r=1}^{\mu} \langle \frac{1}{\sqrt{\mu}} a_r(t) \mathbf{x}_i [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle \geq 0]\!], \frac{1}{\sqrt{\mu}} a_r(t) \mathbf{x}_j [\![\langle \mathbf{w}_r(t), \mathbf{x}_j \rangle \geq 0]\!] \rangle \\
&= \frac{1}{\mu} \langle \mathbf{x}_i, \mathbf{x}_j \rangle \sum_{r=1}^{\mu} a_r(t)^2 [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle \geq 0, \langle \mathbf{w}_r(t), \mathbf{x}_j \rangle \geq 0]\!]
\end{aligned}
$$

$$\begin{aligned}
\mathbf{G}_{ij}(t) &\triangleq \sum_{r=1}^{\mu} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_i)}{\partial a_r(t)} \frac{\partial f(\mathbf{W}(t), \mathbf{a}, \mathbf{x}_j)}{\partial a_r(t)} \\
\text{(this differs from the paper)} &= \sum_{r=1}^{\mu} \frac{1}{\sqrt{\mu}} \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle) \frac{1}{\sqrt{\mu}} \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_j \rangle) \\
&= \frac{1}{\mu} \sum_{r=1}^{\mu} \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle) \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_j \rangle)
\end{aligned}$$

(With biases $\mathbf{b}(t)$:)

$$\begin{aligned}
\frac{\mathrm{d}u_i}{\mathrm{d}t} &= \frac{\mathrm{d}f(\theta(t), \mathbf{x}_i)}{\mathrm{d}t} \\
\text{(chain rule)} &= \sum_{r=1}^{\mu} \langle \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\mathrm{d}\mathbf{w}_r(t)}{\mathrm{d}t} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial a_r(t)} \frac{\mathrm{d}a_r(t)}{\mathrm{d}t} + \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial b_r(t)} \frac{\mathrm{d}b_r(t)}{\mathrm{d}t} \\
&= \sum_{r=1}^{\mu} \langle \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\partial \ell(\theta)}{\partial \mathbf{w}_r(t)} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial a_r(t)} \frac{\partial \ell(\theta)}{\partial a_r(t)} + \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial b_r(t)} \frac{\partial \ell(\theta)}{\partial a_r(t)} \\
&= \sum_{j=1}^{n} (y_j - u_j(t)) \Bigg(\sum_{r=1}^{\mu} \langle \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial \mathbf{w}_r(t)}, \frac{\partial f(\theta, \mathbf{x}_j)}{\partial \mathbf{w}_r(t)} \rangle + \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial a_r(t)} \frac{\partial f(\theta(t), \mathbf{x}_j)}{\partial a_r(t)} + \sum_{r=1}^{\mu} \frac{\partial f(\theta(}{\partial b_r} \\
&= \sum_{j=1}^{n} (y_j - u_j(t)) \left(\mathbf{H}_{ij}(t) + \mathbf{G}_{ij}(t) + \mathbf{F}_{ij}(t) \right) \\
\mathbf{H}_{ij}(t) &\triangleq \frac{1}{\mu} \langle \mathbf{x}_i, \mathbf{x}_j \rangle \sum_{r=1}^{\mu} a_r(t)^2 [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle + b_r \geq 0, \langle \mathbf{w}_r(t), \mathbf{x}_j \rangle + b_r \geq 0]\!] \\
\mathbf{G}_{ij}(t) &\triangleq \frac{1}{\mu} \sum_{r=1}^{\mu} \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_i + b_r \rangle) \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_j + b_r \rangle) \\
\mathbf{F}_{ij}(t) &\triangleq \sum_{r=1}^{\mu} \frac{\partial f(\theta(t), \mathbf{x}_i)}{\partial b_r(t)} \frac{\partial f(\theta(t), \mathbf{x}_j)}{\partial b_r(t)} \\
&= \sum_{r=1}^{\mu} \frac{1}{\sqrt{\mu}} a_r(t) [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle + b_r(t) > 0]\!] \frac{1}{\sqrt{\mu}} a_r(t) [\![\langle \mathbf{w}_r(t), \mathbf{x}_j \rangle + b_r(t) > 0]\!] \\
&= \frac{1}{\mu} \sum_{r=1}^{\mu} a_r(t)^2 [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle + b_r(t) > 0, \langle \mathbf{w}_r(t), \mathbf{x}_j \rangle + b_r(t) > 0]\!]
\end{aligned}$$

Alternatively,

$$\begin{aligned}
\mathbf{H}'_{ij}(t) &\triangleq \frac{1}{\mu} \left(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle\right) \sum_{r=1}^{\mu} a_r(t)^2 [\![\langle \mathbf{w}_r(t), \mathbf{x}_i \rangle + b_r \geq 0, \langle \mathbf{w}_r(t), \mathbf{x}_j \rangle + b_r \geq 0]\!] \\
\frac{\mathrm{d}u_i}{\mathrm{d}t} &= \sum_{j=1}^{n} (y_j - u_j(t)) \left(\mathbf{H}'_{ij}(t) + \mathbf{G}_{ij}(t) \right)
\end{aligned}$$

(This is consistent with/equivalent to the practice of augmenting each $\mathbf{x}_i$ with an extra 1)

$$\begin{aligned}
\frac{\partial u(\mathbf{x}, t)}{\partial t} &= \sum_{j=1}^{n} (y_j - u_j(t)) (\mathbf{H}'_j(\mathbf{x}, t) + \mathbf{G}_j(\mathbf{x}, t)) \\
\mathbf{H}'_j(\mathbf{x}, t) &\triangleq \frac{1}{\mu} (1 + \langle \mathbf{x}, \mathbf{x}_j \rangle) \sum_{r=1}^{\mu} a_r(t)^2 [\![\langle \mathbf{w}_r(t), \mathbf{x} \rangle + b_r \geq 0, \langle \mathbf{w}_r(t), \mathbf{x}_j \rangle + b_r \geq 0]\!]
\end{aligned}$$

$$\mathbf{G}_j(\mathbf{x}, t) \triangleq \frac{1}{\mu} \sum_{r=1}^{\mu} \sigma(\langle \mathbf{w}_r(t), \mathbf{x} + b_r \rangle) \sigma(\langle \mathbf{w}_r(t), \mathbf{x}_j + b_r \rangle)$$

Converting to our notation,

$$\begin{aligned}
\frac{\mathrm{d}\hat{y}_i(t)}{\mathrm{d}t} &= \sum_{j=1}^{n} \hat{\epsilon}_j(t)(\mathbf{H}'_{ij}(t) + \mathbf{G}_{ij}(t)) \\
&= \sum_{j=1}^{n} \text{similarity of } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ according to the network, weighted by the residual at } \mathbf{x}_j \\
&= \sum_{j=1}^{n} \text{residual at } \mathbf{x}_j\text{, weighted by similarity to } \mathbf{x}_i \\
&= \langle \boldsymbol{\epsilon}, \mathbf{H}'_i(t) + \mathbf{G}_i(t) \rangle \\
\frac{\mathrm{d}\hat{\mathbf{y}}(t)}{\mathrm{d}t} &= (\mathbf{H}'(t) + \mathbf{G}(t))\hat{\boldsymbol{\epsilon}} \\
\mathbf{H}'_{ij}(t) &\triangleq \frac{1}{\mu}(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle) \sum_{r=1}^{\mu} v_r(t)^2 \hat{a}_{r,i}(t) \hat{a}_{r,j}(t) \\
&= \frac{1}{\mu}(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle) \langle \mathbf{v}(t) \odot \mathbf{v}(t) \odot \mathbf{a}_i(t), \mathbf{a}_j(t) \rangle \\
&= \frac{1}{\mu}(1 + \langle \mathbf{x}_i, \mathbf{x}_j \rangle) \langle \mathbf{v}(t) \odot \mathbf{v}(t), \mathbf{a}_i(t) \odot \mathbf{a}_j(t) \rangle \\
&= \frac{1}{\mu} \langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j \rangle \langle \mathbf{v}(t) \odot \mathbf{v}(t), \mathbf{a}_i(t) \odot \mathbf{a}_j(t) \rangle \\
&= \frac{1}{\mu} \langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j \rangle \; \langle \mathbf{a}_i(t), \mathbf{a}_j(t) \rangle_{\mathbf{v}(t) \odot \mathbf{v}(t)} \\
&= \frac{1}{\mu} \langle \widetilde{\mathbf{x}}_i, \widetilde{\mathbf{x}}_j \rangle \; \langle \mathbf{v}(t) \odot \mathbf{a}_i(t), \mathbf{v}(t) \odot \mathbf{a}_j(t) \rangle \\
\mathbf{G}_{ij}(t) &\triangleq \frac{1}{\mu} \sum_{r=1}^{\mu} \phi(\langle \mathbf{w}_r(t), \mathbf{x}_i + b_r \rangle) \phi(\langle \mathbf{w}_r(t), \mathbf{x}_j + b_r \rangle) \\
&\triangleq \frac{1}{\mu} \langle \boldsymbol{\phi}_i(t), \boldsymbol{\phi}_j(t) \rangle.
\end{aligned}$$

This completes the proof. □

5.8 Pre- and Post-Activations in Function Space

We now develop expressions for the pre-activation (net input) and activation of a given neuron:

$$\begin{aligned}
z_i^{(1)} &= w_i^{(1)} x + b_i^{(1)} \\
x_i^{(1)} &= \phi(z_i^{(1)}) \\
z_i^{(\ell)} &= \sum_{j=1}^{\mu^{(\ell-1)}} w_{ij}^{(\ell)} x_j^{(\ell-1)} + b_i^{(\ell)} \\
x_i^{(\ell)} &= \phi(z_i^{(\ell)}) \\
g_\theta(x) &= \sum_{i=1}^{\mu^{(L)}} w_i^{(L+1)} x_i^{(L)} + b^{(L+1)} \\
z_i^{(2)} &= \sum_{j \in \mathcal{A}^{(1)}(x)} w_{ij}^{(2)} w_j^{(1)} (x - \gamma_j^{(1)}) + b_i^{(2)}
\end{aligned}$$

$$
\begin{aligned}
z_i^{(3)} &= \sum_{j=1}^{\mu^{(2)}} w_{ij}^{(3)} \phi\left(\sum_{k\in\mathcal{A}^{(1)}(x)} w_{jk}^{(2)} w_k^{(1)} (x - \gamma_k^{(1)}) + b_j^{(2)}\right) + b_i^{(3)} \\
&= \sum_{j=1}^{\mu^{(2)}} w_{ij}^{(3)} \left(\sum_{k\in\mathcal{A}^{(1)}(x)} w_{jk}^{(2)} w_k^{(1)} (x - \gamma_k^{(1)}) + b_j^{(2)}\right) \\
&\qquad \times \left\llbracket \sum_{k\in\mathcal{A}^{(1)}(x)} w_{jk}^{(2)} w_k^{(1)} (x - \gamma_k^{(1)}) + b_j^{(2)} > 0 \right\rrbracket + b_i^{(3)} \\
(\gamma_{jk}^{(2)} &\triangleq \gamma_k^{(1)} - \frac{b_j^{(2)}}{w_{jk}^{(2)} w_k^{(1)}}) \\
&= \sum_{j\in\mathcal{A}^{(2)}(x)} w_{ij}^{(3)} \left(\sum_{k\in\mathcal{A}^{(1)}(x)} w_{jk}^{(2)} w_k^{(1)} (x - \gamma_{jk}^{(2)})\right) + b_i^{(3)} \\
&= \sum_{j\in\mathcal{A}^{(2)}(x)} \left(\sum_{k\in\mathcal{A}^{(1)}(x)} w_{ij}^{(3)} w_{jk}^{(2)} w_k^{(1)} (x - \gamma_{jk}^{(2)})\right) + b_i^{(3)} \\
&= \sum_{\mathfrak{p}\in\mathcal{A}_i^{(\leq 3)}(x)} \left(\prod_{w\in\mathfrak{p}} w\right) (x - \gamma_{\mathfrak{p}}^{(2)}) + b_i^{(3)} \\
z_i^{(4)} &= \sum_{j=1}^{\mu^{(3)}} w_{ij}^{(4)} \phi\left(\sum_{\mathfrak{p}\in\mathcal{A}_j^{(\leq 3)}(x)} \left(\prod_{w\in\mathfrak{p}} w\right) (x - \gamma_{\mathfrak{p}}^{(2)}) + b_j^{(3)}\right) + b_i^{(4)} \\
&= \sum_{j\in\mathcal{A}^{(3)}(x)} \left(\sum_{\mathfrak{p}\in\mathcal{A}_j^{(\leq 3)}(x)} w_{ij}^{(4)} \left(\prod_{w\in\mathfrak{p}} w\right) (x - \gamma_{\mathfrak{p}}^{(2)}) + b_j^{(3)}\right) + b_i^{(4)} \\
&= \sum_{\mathfrak{p}\in\mathcal{A}_i^{(\leq 4)}(x)} \left(\prod_{w\in\mathfrak{p}} w\right) (x - \gamma_{\mathfrak{p}}^{(3)}) + b_i^{(4)}
\end{aligned}
$$

Notation: Subscripts of $\mathfrak{p}$ needed; in the denominator, $\mathfrak{p}[:v]$ includes the vth element, ambiguously-consistent with the Python notation because it's not 0-indexed

Based on our derivations above, we can now write the general case as:

$$
\begin{aligned}
\gamma_{\mathfrak{p}}^{(\ell)} &\triangleq \gamma_{\mathfrak{p}[:-1]}^{(\ell-1)} - \frac{b_{\mathfrak{p}[-1]}^{(\ell)}}{\prod_{w\in\mathfrak{p}} w} \\
&= -\sum_{v=1}^{\ell} \frac{b_{\mathfrak{p}[v]}^{(v)}}{\prod_{w\in\mathfrak{p}[:v]} w} \\
z_i^{(\ell)} &= \sum_{\mathfrak{p}\in\mathcal{A}_i^{(\leq \ell)}(x)} \left(\prod_{w\in\mathfrak{p}} w\right) (x - \gamma_{\mathfrak{p}}^{(\ell-1)}) + b_i^{(\ell)} \\
&\triangleq \sum_{\mathfrak{p}\in\mathcal{A}_i^{(\leq \ell)}(x)} \mu_{\mathfrak{p}}^{(\ell)} (x - \gamma_{\mathfrak{p}}^{(\ell-1)}) + b_i^{(\ell)}
\end{aligned}
$$

$$\triangleq \overline{\mu}_{\mathcal{A}_i^{(\leq \ell)}(x)} x - \overbrace{\overline{\gamma}_{\mathcal{A}_i^{(\leq \ell)}(x)} \underbrace{+b_i^{(\ell)}}_{\text{could be absorbed into } \overline{\gamma}\text{...}}}^{(y\text{-intercept})}$$

$$= \overline{\mu}_{\mathcal{A}_i^{(\leq \ell)}(x)} \left(\underbrace{x - \frac{\overline{\gamma}_{\mathcal{A}_i^{(\leq \ell)}(x)} - b_i^{(\ell)}}{\overline{\mu}_{\mathcal{A}_i^{(\leq \ell)}(x)}}}_{x\text{-intercept of line segment containing } x \text{ (candidate breakpoint)}} \right)$$

Notation: does a path $\mathfrak{p} \in \mathcal{A}_i^{(\leq \ell)}(x)$ contain $b_i^{(\ell)}$ as its last bias? Above we say no, below, yes

$$\overline{\gamma}^{\star}_{\mathcal{A}_i^{(\leq \ell)}(x)} = -\frac{\sum_{\mathfrak{p} \in \mathcal{A}_i^{(\leq \ell)}(x)} \sum_{v=1}^{\ell} b^{(v)}_{\mathfrak{p}[v]} \prod_{w \in \mathfrak{p}[v:]} w}{\sum_{\mathfrak{p} \in \mathcal{A}_i^{(\leq \ell)}(x)} \prod_{w \in \mathfrak{p}} w}$$

Consider the layer-$(\ell - 1)$ breakpoints containing x, i.e. $x \in [\beta^{(\ell-1)}_{(k)}, \beta^{(\ell-1)}_{(k+1)}]$. Then, if $\overline{\gamma}^{\star}_{\mathcal{A}_i^{(\leq \ell)}(x)} \in [\beta^{(\ell-1)}_{(k)}, \beta^{(\ell-1)}_{(k+1)}]$, $\overline{\gamma}^{\star}_{\mathcal{A}_i^{(\leq \ell)}(x)}$ is an *active* breakpoint $\beta^{(\ell)}_{k}$ for some k.