frontiers | Frontiers in Artificial Intelligence

ORIGINAL RESEARCH
published: 11 May 2022
doi: 10.3389/frai.2022.889981

Shallow Univariate ReLU Networks as Splines: Initialization, Loss Surface, Hessian, and Gradient Flow Dynamics

Justin Sahs[1†], *Ryan Pyle*[1†], *Aneel Damaraju*[2], *Josue Ortega Caro*[1], *Onur Tavaslioglu*[3], *Andy Lu*[2], *Fabio Anselmi*[1] *and Ankit B. Patel*[1,2*]

[1] Department of Neuroscience, Baylor College of Medicine, Houston, TX, United States, [2] Department of Electrical Engineering, Rice University, Houston, TX, United States, [3] Department of Computational and Applied Mathematics, Rice University, Houston, TX, United States

OPEN ACCESS

Edited by:
Dongpo Xu,
Northeast Normal University, China

Reviewed by:
Qinwei Fan,
Xi'an Polytechnic University, China
Francesco Caravelli,
Los Alamos National Laboratory (DOE), United States

***Correspondence:**
Ankit B. Patel
ankit.patel@rice.edu

[†] *These authors share first authorship*

Specialty section:
This article was submitted to Machine Learning and Artificial Intelligence, a section of the journal Frontiers in Artificial Intelligence

Received: *04 March 2022*
Accepted: *04 April 2022*
Published: *11 May 2022*

Citation:
Sahs J, Pyle R, Damaraju A, Caro JO, Tavaslioglu O, Lu A, Anselmi F and Patel AB (2022) Shallow Univariate ReLU Networks as Splines: Initialization, Loss Surface, Hessian, and Gradient Flow Dynamics. Front. Artif. Intell. 5:889981. doi: 10.3389/frai.2022.889981

Understanding the learning dynamics and inductive bias of neural networks (NNs) is hindered by the opacity of the relationship between NN parameters and the function represented. Partially, this is due to symmetries inherent within the NN parameterization, allowing multiple different parameter settings to result in an identical output function, resulting in both an unclear relationship and redundant degrees of freedom. The NN parameterization is invariant under two symmetries: permutation of the neurons and a continuous family of transformations of the scale of weight and bias parameters. We propose taking a quotient with respect to the second symmetry group and reparametrizing ReLU NNs as continuous piecewise linear splines. Using this spline lens, we study learning dynamics in shallow univariate ReLU NNs, finding unexpected insights and explanations for several perplexing phenomena. We develop a surprisingly simple and transparent view of the structure of the loss surface, including its critical and fixed points, Hessian, and Hessian spectrum. We also show that standard weight initializations yield very flat initial functions, and that this flatness, together with overparametrization and the initial weight scale, is responsible for the strength and type of implicit regularization, consistent with previous work. Our implicit regularization results are complementary to recent work, showing that initialization scale critically controls implicit regularization via a kernel-based argument. Overall, removing the weight scale symmetry enables us to prove these results more simply and enables us to prove new results and gain new insights while offering a far more transparent and intuitive picture. Looking forward, our quotiented spline-based approach will extend naturally to the multivariate and deep settings, and alongside the kernel-based view, we believe it will play a foundational role in efforts to understand neural networks. Videos of learning dynamics using a spline-based visualization are available at http://shorturl.at/tFWZ2.

Keywords: neural networks, symmetry, implicit bias, splines, learning dynamics

1. INTRODUCTION

Deep learning has revolutionized the field of machine learning (ML), leading to state of the art performance in image segmentation, medical imaging, machine translation, chess playing, reinforcement learning, and more. Despite being intensely studied and widely used, theoretical understanding of some of its fundamental properties remains poor. One critical, yet mysterious property of deep learning is the root cause of the excellent generalization achieved by overparameterized networks (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 has even been observed for networks that are massively overparameterized, wherein, according to traditional ML theory, they should (over)fit the training data (Neyshabur et al., 2015). This overparameterization leads to a highly complicated loss surface, with increasingly many local minima. How does training networks with excess capacity lead to generalization? And how can generalization error decrease with overparameterization? How can gradient descent in a loss landscape riddled with local minima so frequently converge to near-global minima? One possible solution is a phenomenon known as implicit regularization (or implicit bias). Empirical studies (Zhang et al., 2016; Advani et al., 2020; Geiger et al., 2020) have shown that overparametrized NNs behave *as if* they possess strong regularization, even when trained from scratch with no explicit regularization—instead possessing *implicit* regularization or bias (IR or IB), since such regularization is *not* explicitly imposed by the designer.

We approach these issues by considering symmetries of the NN parameterization. Several recent works have brought attention to the importance of symmetries to deep learning (Badrinarayanan et al., 2015; Kunin et al., 2020; Tayal et al., 2020). One important advance is building symmetry aware architectures that automatically generalize, such as convolutional layers granting translation-invariant representations, or many other task-specific symmetries (Liu et al., 2016; Barbosa et al., 2021; Bertoni et al., 2021; Liu and Okatani, 2021). Understanding and exploiting symmetries may lead to similar improvements, resulting in better performance or models that require less training data due to implementation of expert-knowledge based symmetry groups.

We consider a new spline-based reparamaterization of a shallow ReLU NN, explicitly based on taking a quotient with respect to an existing weight symmetry. In particular, we focus on shallow fully connected univariate ReLU networks, whose parameters will always result in a Continuous Piecewise Linear (CPWL) output. We reparameterize the CPWL function implemented by the network in terms of the locations of the nonlinearities (*breakpoints*), the associated change in slope (*delta-slopes*), and a parameter that identifies which side of the breakpoint the associated neuron is active on (*orientation*). This parameterization is defined formally in section 2.1, and illustrated in **Figure 2**. We provide theoretical results for shallow networks, with experiments confirming these results.

1.1. Main Contributions

The main contribution of this work are as follows:

- *Initialization: Increasingly Flat with Width.* In the spline perspective, once the weight symmetry has been accounted for, neural network parameters determine the locations of breakpoints and their delta-slopes in the CPWL reparametrization. We prove that, for common initializations, these distributions are mean 0 with low standard deviation. Notably, the delta-slope distribution becomes increasingly concentrated near 0 as the width of the network increases, leading to flatter initial approximations.
- *A characterization of the Loss Surface, Critical Points and Hessian, revealing that Flat Minima are due to degeneracy of breakpoints.* We fully characterize the Hessian of the loss surface at critical points, revealing that the Hessian is positive semi-definite, with 0 eigenvalues occurring when elements of its Gram matrix set are linearly dependent. We show that many of these 0 eigenvalues occur due to symmetry artifacts of the standard NN parametrization. We characterize the ways this can occur, including whenever multiple breakpoints share the same active data—thus we expect that at any given critical point in an overparametrized network, the loss surface will be flat in many directions.
- *Implicit Regularization is due to Flat Initialization in the Overparametrized Regime.* We find that implicit regularization in overparametrized Fully Connected (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, made explicitly clear by our symmetry-quotiented reparametrization) and (iii) the role of gradient descent (GD) in preserving initialization and providing regularization of curvature. In particular, each neuron has an global, rather than local, impact as each contributes an affine ReLU. Thus, backpropogation distributes the "work" of fitting the training data over many units, leading to regularized delta-slope and breakpoints. All else equal, these non-local effects mean that more overparametrization leads to each unit mattering less, thus typically resulting in better generalization due to implicit regularization (Neyshabur et al., 2015, 2018).

2. THEORETICAL RESULTS

2.1. Spline Parametrization and Notation

Consider a fully connected ReLU neural net $\hat{f}(x;\theta)$ with a single hidden layer of width H, scalar input $x \in \mathbb{R}$ and scalar output $\hat{y} \in \mathbb{R}$, which we are attempting to match to a target function $f(x)$, from which we have N sample input/output data pairs (x_n, y_n); the full vectors of inputs and outputs are denoted x and y, respectively. $\hat{f}(\cdot;\theta)$ is a continuous piecewise linear (CPWL) function since the ReLU nonlinearity is CPWL.

$$\hat{f}(x;\theta_{\mathrm{NN}}) \triangleq b_0 + \sum_{i=1}^{H} v_i(w_i x + b_i)_+$$

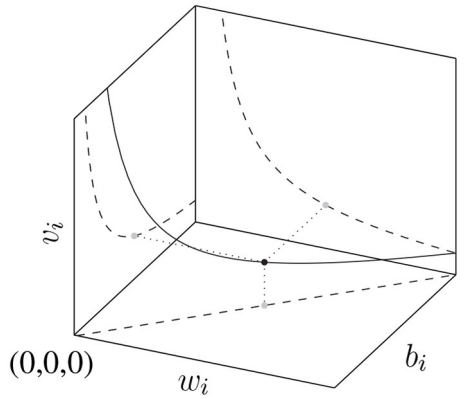

FIGURE 1 | The manifold generated by the scaling transformation of input weight, input bias, output weight: $(w_i, b_i, v_i) \mapsto (\alpha_i w_i, \alpha_i b_i, \alpha_i^{-1} v_i)$, which leaves $\hat{f}(x)$ invariant.

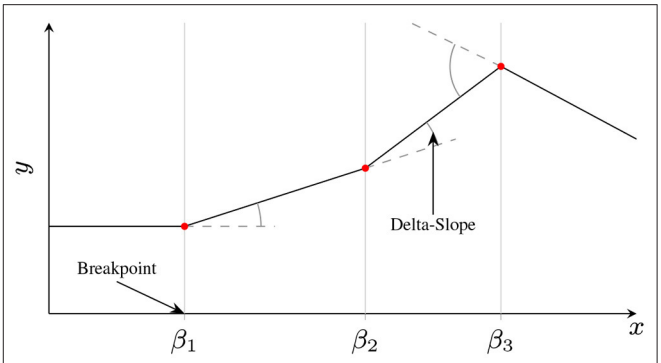

FIGURE 2 | Illustration of BDSO terminology of a CPWL function. Breakpoints are where adjacent linear pieces meet, and the delta-slope determines the difference in slope between the adjacent linear pieces. Orientations determine on which side of a breakpoint the delta-slope is applied. In a shallow NN, each neuron contributes exactly one breakpoint, with an associated facing s_i and delta-slope μ_i – in the above example, neurons 1 and 2 are right-facing with positive delta-slope ($s_1 = s_2 = 1, \mu_1, \mu_2 > 0$), and neuron 3 is left-facing with negative delta-slope ($s_3 = -1, \mu_3 < 0$).

Here the NN parameters $\theta_{\text{NN}} \triangleq \{b_0\} \cup \{(w_i, b_i, v_i)\}_{i=1}^H$ denote the global bias plus the input weight, bias, and output weight, respectively of neuron i, and $(\cdot)_+ \triangleq \max\{0, \cdot\}$ denotes the ReLU function.

An important observation is that the ReLU NN parametrization is redundant: for every function $\hat{f}$ represented by Equation (1) there exists infinitely many transformations of the parameters $\theta'_{\text{NN}} \equiv \mathcal{R}(\theta_{\text{NN}})$ s.t. the transformed function $\hat{f}(x; \theta'_{\text{NN}}) = \hat{f}(x; \theta_{\text{NN}})$. These invariant transformations $\mathcal{R}$ consist of (i) permutations of the hidden units and (ii) scalings of the weights and biases of the form $w_i \mapsto \alpha_i w_i, b_i \mapsto \alpha_i b_i, v_i \mapsto \alpha_i^{-1} v_i$ for $\alpha_i \in \mathbb{R}_{>0}$ (Rolnick and Kording, 2019). The manifold of constant loss generated by the α-scaling symmetry is shown in **Figure 1**. The set $\mathcal{G}$ of such function-invariant transformations together with function composition $\circ$ forms a group.

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 taking the quotient with respect to the above α-scaling symmetry, transforming from the NN parametrization of weights and biases to the following CPWL *spline* parametrization:

$$\hat{f}(x; \theta_{\text{BDSO}}) \triangleq b_0 + \sum_{i=1}^{H} \mu_i (x - \beta_i)_{s_i}, \tag{1}$$

where

$$(x - \beta_i)_{s_i} \triangleq (x - \beta_i) \cdot \begin{cases} [\![x > \beta_i]\!], \text{ if } s_i = 1 \\ [\![x < \beta_i]\!], \text{ if } s_i = -1 \end{cases}$$

and the Iversen bracket $[\![b]\!]$ is 1 when the condition b is true, and 0 otherwise. The CPWL spline parametrization is the *Breakpoint, Delta-Slope, Orientation (BDSO)* parametrization, $\theta_{\text{BDSO}} \triangleq \{b_0\} \cup \{(\beta_i, \mu_i, s_i)\}_{i=1}^H$, 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 $s_i = +1$). This terminology is illustrated in **Figure 2**.

Considering the BDSO parametrization 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 taking the quotient with respect to the α-scaling symmetry, leading to two useful properties: (1) the loss depends on the NN parameters θ_{NN} only through the BDSO parameters (the approximating function) θ_{BDSO} i.e., $\ell(\theta_{\text{NN}}) = \ell(\theta_{\text{BDSO}}(\theta_{\text{NN}}))$, analogous to the concept of a minimum sufficient statistic in exponential family models; and (2) the BDSO parameters are more intuitive, allowing us to reproduce known results (Williams et al., 2019) more succinctly, and expand out to new results more easily. Much recent related work has also moved in this direction, analyzing function space (Balestriero and Baraniu, 2018; Hanin and Rolnick, 2019).

The BDSO parametrization makes clear that, fundamentally, $\hat{f}(x; \theta_{\text{NN}})$ is represented in terms of its second derivative. Examining

$$\hat{f}''(x; \theta_{\text{BDSO}}) = \sum_{i=1}^{H} \mu_i s_i \delta(x - \beta_i) \tag{2}$$

further illustrates the meaning of the BDSO parameters, especially μ_i and β_i: the second derivative is zero everywhere except at $x = \beta_i$ for some i, where there is a Dirac delta of magnitude μ_i.

We note that the BDSO parametrization of a ReLU NN is closely related to but different than a traditional mth order spline parametrization $\hat{f}_{\text{spline}}(x) \triangleq \sum_{i=1}^K \mu_i (x - \beta_i)_+^m + \sum_{j=0}^m c_j x^j$ (Reinsch, 1967). The BDSO parametrization (i) lacks the base polynomial, and (ii) has two possible breakpoint

orientations $s_i \in \{\pm 1\}$ whereas the spline is canonically all right-facing. Additionally, adding in the base polynomial (for the linear case $m = 1$) into the BDSO parametrization yields a ReLU ResNet parametrization.

It is also useful to consider the set of sorted breakpoints, $\{\beta_p\}_{p=0}^{H+1}$, where $-\infty \triangleq \beta_0 < \beta_1 < \cdots < \beta_p < \cdots < \beta_{H+1} \triangleq \infty$, which induces two partitions: (i) a partition of the *domain* $\mathcal{X}$ into intervals $\mathcal{X}_p \triangleq [\beta_p, \beta_{p+1})$; and (ii) a partition of the *training data* $\mathcal{D}_x \triangleq \{x_n\}_{n=1}^N$ into pieces $\pi_p \triangleq \mathcal{X}_p \cap \mathcal{D}_x$, so that $\mathcal{D}_x = \cup_{p=0}^{H+1} \pi_p$. Note that sorting the breakpoints (and hence the neurons) removes the permutation symmetry present in the standard NN parametrization.

Finally, we introduce some notation relevant to the training of the network: let $\hat{\boldsymbol{\epsilon}} \in \mathbb{R}^{N\times 1}$ denote the vector of *residuals* $\mathrm{y} - \hat{\mathrm{y}}$, i.e., $\hat{\epsilon}_n = f(x_n) - \hat{f}(x_n)$, let $1_i \in \mathbb{R}^{N\times 1}$ denote the *activation pattern* of neuron i over all N inputs, $1_{i,n} = [\![w_i x_n + b_i > 0]\!]$, and let $\hat{\boldsymbol{\epsilon}}_i \triangleq \hat{\boldsymbol{\epsilon}} \odot 1_i \in \mathbb{R}^{N\times 1}$ and $\mathrm{x}_i \triangleq \mathrm{x} \odot 1_i \in \mathbb{R}^{N\times 1}$ denote the *relevant* residuals and inputs for neuron i, respectively. In words, relevant residuals and inputs for neuron i are those which are not zeroed out by the ReLU activation.

2.1.1. Basis Expansion, Infinite Width Limit

The expansion in Equation 2 can be further understood by viewing it as a basis expansion. Specifically, we can view the BDSO framework as a basis expansion of $\hat{f}''$ in a basis of Dirac delta functions, or, from Equation 1 as a basis expansion of $\hat{f}$ in a basis of functions $(x - \beta_i)_{s_i}$. The form of these basis functions is fixed as the ReLU activation—analogous to the mother basis function in wavelets (Rao, 2002)—but can be translated via changes in β_i, reflected by s_i, or re-weighted via changes in μ_i.

Also, note that if the breakpoints β_i are fixed, and only the delta-slope parameters μ_i are optimized, then we effectively have a (kernel) linear regression model. Thus, the power of neural network learning lies in the ability to translate/orient these basis functions in order to better model the data. Intuitively, in a good fit, the breakpoints β_i will tend to congregate near areas of high curvature in the ground truth function, i.e., $|f''(x)| \gg 0$, so that the sum-of-Diracs $\hat{f}''(x; \theta_{\mathrm{BDSO}})$ better approximates $f''(x)$.[1]

Consider the infinite width limit $H \to \infty$, rewriting Equation 2 as

$$\begin{aligned}\hat{f}''(x;\theta_{\mathrm{BDSO}}) &= \int c(\beta)\delta(x-\beta)\,\mathrm{d}\beta \\ &= \int p_\beta(\beta)\frac{c(\beta)}{p_\beta(\beta)}\delta(x-\beta)\,\mathrm{d}\beta \\ &= \mathbb{E}_\beta\left[\frac{c(\beta)}{p_\beta(\beta)}\delta(x-\beta)\right].\end{aligned} \tag{3}$$

Using the Law of Large Numbers, $\mathbb{E}_\beta[\frac{c(\beta)}{p_\beta(\beta)}\delta(x-\beta)] = \frac{1}{H}\sum_i \frac{c(\beta_i)}{p_\beta(\beta_i)}\delta(x-\beta_i) + O(1/H)$ and so we may write $\mu_i s_i \triangleq c(\beta_i)/Hp_\beta(\beta_i)$. Then, from Equation 3, we can expand this as $\mu_i s_i = \hat{f}''(\beta_i; \theta_{\mathrm{BDSO}})/Hp_\beta(\beta_i)$.

[1] Although a "breakpoints near curvature" fit will be a good fit, that does not mean that Gradient Descent from a random initialization can find such a fit; see section 2.3.

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 $b_i \sim \mathcal{N}(0, \sigma_b)$, $w_i \sim \mathcal{N}(0, \sigma_w)$, $v_i \sim \mathcal{N}(0, \sigma_v)$, and independent uniform initialization, with $b_i \sim \mathcal{U}[-a_b, a_b]$, $w_i \sim \mathcal{U}[-a_w, a_w]$, $v_i \sim \mathcal{U}[-a_v, a_v]$. We find that common initializations result in flat functions, becoming flatter with increasing width.

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,

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

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

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

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

$$\begin{aligned}p_\beta(\beta_i) &= \mathrm{Cauchy}\left(\beta_i; 0, \frac{\sigma_b}{\sigma_w}\right) \\ p_\mu(\mu_i) &= \frac{G_{0,2}^{2,0}\left(\frac{\mu_i^2}{4\sigma_v^2\sigma_w^2}\middle| 0,0\right)}{2\pi\sigma_v\sigma_w} = \frac{K_0\left(\frac{|\mu_i|}{\sigma_v\sigma_w}\right)}{\pi\sigma_v\sigma_w} \\ p_{\mu|\beta}(\mu_i|\beta_i) &= \mathrm{Laplace}\left(\mu_i; 0, \frac{\sigma_b\sigma_v\sigma_w}{\sqrt{\sigma_b^2+\sigma_w^2\beta_i^2}}\right)\end{aligned}$$

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,

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

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

2.2.1. Implications

Corollary 1 implies that the breakpoint density drops quickly away from the origin for common initializations. As breakpoints are necessary to fit curvature, if the ground truth function $f(x)$ has significant curvature far from the origin, then it may be far more difficult to fit. We show that this is indeed the case by training a shallow ReLU NN with an initialization that does not match the underlying curvature, with training becoming easier if the initial breakpoint distribution better matches the function curvature. This suggests that a data-dependent initialization, with more breakpoints near areas of high curvature, could potentially be faster and easier to train. Note that this simple curvature-based interpretation is made possible by our symmetry-quotiented reparametrization.

Next, we consider the typical Gaussian He (He et al., 2015) or Glorot (Glorot and Bengio, 2010) initializations. In the He initialization, we have $\sigma_w = \sqrt{2}$, $\sigma_v = \sqrt{2/H}$. In the Glorot initialization, we have $\sigma_w = \sigma_v = \sqrt{2/(H+1)}$. We wish to consider their effect on the smoothness of the initial function approximation. From here on, we measure the smoothness using the *roughness* of $\hat{f}$, $\rho \triangleq \int |\hat{f}''(x; \theta_{\mathrm{BDSO}})|^2 \, \mathrm{d}x = \sum_i \mu_i^2 = \|\boldsymbol{\mu}\|_2^2$, where lower roughness indicates a smoother approximation.

Theorem 2. *Consider the initial roughness* ρ_0 *under a Gaussian initialization. In the He initialization, we have that the tail probability is given by*

$$\mathbb{P}[\rho_0 - \mathbb{E}[\rho_0] \geq \lambda] \leq \frac{1}{1 + \frac{\lambda^2 H}{128}},$$

where $\mathbb{E}[\rho_0] = 4$. *In the Glorot initialization, we have that the tail probability is given by*

$$\mathbb{P}[\rho_0 - \mathbb{E}[\rho_0] \geq \lambda] \leq \frac{1}{1 + \frac{\lambda^2 (H+1)^4}{128 H}},$$

where $\mathbb{E}[\rho_0] = \frac{4H}{(H+1)^2} = O\left(\frac{1}{H}\right)$.

Thus, as the width H increases, the distribution of the roughness of the initial function $\hat{f}_0$ gets tighter around its mean. In the case of the He initialization, this mean is constant; in the Glorot initialization, it decreases with H. In either case, for reasonable widths, the initial roughness is small with high probability, corresponding to small initial delta-slopes. This smoothness has implications for the implicit regularization phenomenon observed in recent work (Neyshabur et al., 2018), and studied later in section 2.6, in particular, Theorem 7.

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 (Neal, 1994; Lee et al., 2017; Jacot et al., 2018; Savarese et al., 2019) 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 on the activations as a function of input whereas we also compute the joint densities of the BDSO parameters i.e., breakpoints and delta-slopes. The latter is particularly important for understanding the non-uniform density of breakpoints away from the origin as noted above. The most closely related work is Steinwart (2019), which considers only the breakpoints, and suggests a new initialization with uniform breakpoint distribution.

2.3. Loss Surface in the Spline Parametrization

We now consider the squared loss $\frac{1}{2}\sum_{n=1}^{N}(\hat{f}(x_n; \theta) - y_n)^2$ as a function of either the NN parameters $\ell(\theta_{\mathrm{NN}})$ or the BDSO parameters $\tilde{\ell}(\theta_{\mathrm{BDSO}})$. We begin with the symmetry-quotiented case:

Theorem 3. *The loss function* $\tilde{\ell}(\theta_{BDSO} = (\boldsymbol{\beta}, \boldsymbol{\mu}, \mathrm{s}))$ *is a continuous piecewise quadratic (CPWQ) spline. Furthermore, consider the evolution of the loss as we vary* β_i *along the x axis; this 1-dimensional slice* $\tilde{\ell}(\beta_i; \boldsymbol{\beta}_{-i}, \boldsymbol{\mu}, \mathrm{s})$ *is also a CPWQ spline in* β_i *with knots at datapoints* $\{x_n\}_{n=1}^{N}$. *Let* $p_1(\beta_i)$ *(resp.* $p_2(\beta_i)$*) be the quadratic function equal to* $\tilde{\ell}(\beta_i; \boldsymbol{\beta}_{-i}, \boldsymbol{\mu}, \mathrm{s})$ *for* $\beta_i \in [x_{n-1}, x_n]$ *(resp.* $[x_n, x_{n+1}]$*), which both have positive curvature, and let* $m_j \triangleq \arg\min p_j(\beta_i)$. *Then, with measure 1, the knots* x_n *fall into one of three types as shown in* **Figure 3***:*

- *(Type I, Passthrough)* $m_1, m_2 < x_n$, *or* $x_n < m_1, m_2$,
- *(Type II, Repeller)* $m_1 < x_n < m_2$
- *(Type III, Attractor)* $m_2 < x_n < m_1$.

We call Type I knots Passthrough knots because gradient flow starting from the higher side will simply pass through the knot. Similarly, we call Type II knots Repellers and Type III knots Attractors because β_i will be repelled by or attracted to the knot during gradient flow.

We now return to the loss $\ell(\theta_{\mathrm{NN}})$ under the NN parametrization, and consider varying (w_i, v_i, b_i) such that the corresponding β_i changes but μ_i and s_i stay the same. Due to the α-scaling symmetry, this can be implemented in two distinct ways: (i) $(w_i, v_i, b_i) \mapsto (w_i, v_i, b_i + \delta)$ or (ii) $(w_i, v_i, b_i) \mapsto ((1+\delta)w_i, v_i/(1+\delta), b_i)$. Version (ii) can be implemented by applying version (i) followed by the transformation $(w_i, v_i, b_i) \mapsto (\alpha w_i, v_i/\alpha, \alpha b_i)$ for the appropriate α, which leaves the loss invariant. In fact, there is a continuum of transformations which could be achieved by version (i) followed by an arbitrary α-transformation. Thus, consider the 1-dimensional slice $\ell(b_i; \mathrm{w}, \mathrm{v}, \mathrm{b}_{-i})$, which is equal to $\tilde{\ell}(-b_i/w_i; \boldsymbol{\beta}_{-i}, \boldsymbol{\mu}, \mathrm{s})$, i.e., a horizontally reflected and scaled version of $\tilde{\ell}(\beta_i; \boldsymbol{\beta}_{-i}; \boldsymbol{\mu}, \mathrm{s})$. The 1-dimensional "slice" corresponding to version (ii) will similarly be a stretched version of $\tilde{\ell}(\beta_i; \boldsymbol{\beta}_{-i}; \boldsymbol{\mu}, \mathrm{s})$, with the caveat that this "slice" is along a hyperbola in the (v_i, w_i)-plane. Noting that any changes in (w_i, v_i, b_i) that implement the same change in β_i change the loss in the same way, let $\ell(\beta_i; \theta_{\mathrm{NN}} \setminus \beta_i)$ denote the equivalence class of 1-dimensional slices of $\ell(\theta_{\mathrm{NN}})$. This gives the following corollary:

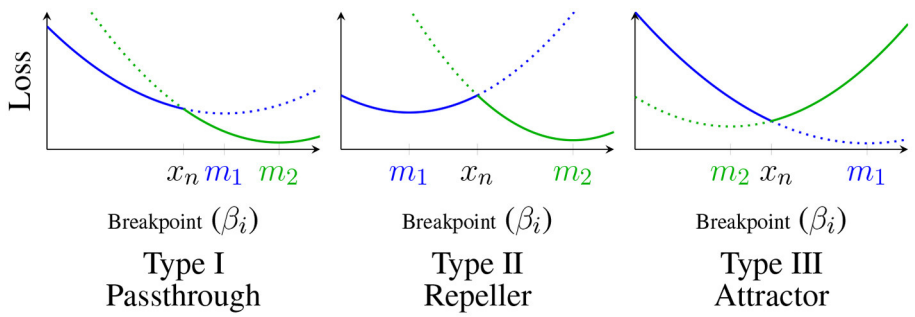

FIGURE 3 | Classification of critical points: $p_1(\beta_i)$ shown in blue, $p_2(\beta_i)$ shown in green; solid lines represent the 1 dimensional loss slice $\tilde{\ell}(\beta_i; \boldsymbol{\beta}_{-i}, \boldsymbol{\mu}, \mathbf{s})$, dotted lines represent the extension of $p_k(\beta_i)$ past x_n.

Corollary 2. *The 1-dimensional slices $\ell(\beta_i; \theta_{NN} \setminus \beta_i)$ of $\ell(\theta_{NN})$ that implement a change only in β_i are CPWQ with knots as in Theorem 3.*

2.3.1. Critical Points of the Loss Surface

In addition to the fixed points associated with attractor datapoints, there are also critical points that lie in the quadratic regions, i.e., local (and global) minima which correspond to the minima of some parabolic piece for each neuron. We characterize these critical points here.

Theorem 4. *Consider some arbitrary θ^*_{NN} such that there exists at least one neuron that is active on all data. Then, θ^*_{NN} is a critical point of $\tilde{\ell}(\cdot)$ if and only if for all domain partitions $\mathcal{X}_p$, the restriction $\hat{f}(\cdot; \theta^*_{NN})|_{\mathcal{X}_p}$ is an Ordinary Least Squares (OLS) fit of the training data contained in π_p.*

Corollary 3. *Suppose that $H \geq cN$ for some constant $c \geq 1$, the total number of lonely partitions, and thus a lower bound on the total number of* global *minima of $\tilde{\ell}$ is $\binom{H+1}{N} = O(N^{cN})$.*

Remark 1. *Suppose that the H breakpoints are uniformly spaced and that the N datapoints are uniformly distributed within the region of breakpoints. Then in the overparametrized regime $H \geq dN^2$ for some constant $d \geq 1$, the induced partition Π is lonely with high probability $1 - e^{-N^2/(H+1)} = 1 - e^{-1/d}$.*

Thus, with only $O(N^2)$ hidden units, we can almost guarantee a lonely partition at initialization. This makes optimization easier but is not sufficient to guarantee that learning will converge to a global minimum, as breakpoint dynamics could change the partition to crowded (see section 2.5). Note how simple and transparent the spline-based explanation is for why overparametrization makes optimization easier. For brevity, we include experiments testing our theory of loneliness in Figure 14 in **Appendix**.

$$\mathrm{H}_\ell(\theta) = \begin{pmatrix} \ddots & & & & \\ \langle v_j \mathrm{x}_j, v_i \mathrm{x}_i \rangle & \langle v_j \mathrm{x}_j, w_i \mathrm{x}_i + b_i 1_i \rangle & \langle v_j \mathrm{x}_j, v_i 1_i \rangle & \cdots & \langle v_j \mathrm{x}_j, 1 \rangle \\ \langle w_j \mathrm{x}_j + b_j 1_j, v_i \mathrm{x}_i \rangle & \langle w_j \mathrm{x}_j + b_j 1_j, w_i \mathrm{x}_i + b_i 1_i \rangle & \langle w_j \mathrm{x}_j + b_j 1_j, v_i 1_i \rangle & \cdots & \langle w_j \mathrm{x}_j + b_j 1_j, 1 \rangle \\ \langle v_j 1_j, v_i \mathrm{x}_i \rangle & \langle v_j 1_j, w_i \mathrm{x}_i + b_i 1_i \rangle & \langle v_j 1_j, v_i 1_i \rangle & \cdots & \langle v_j 1_j, 1 \rangle \\ \vdots & & & \ddots & \vdots \\ \langle 1, v_i \mathrm{x}_i \rangle & \langle 1, w_i \mathrm{x}_i + b_i 1_i \rangle & \langle 1, v_i 1_i \rangle & \cdots & \langle 1, 1 \rangle \end{pmatrix} \tag{4}$$

An open question is how many such critical points exist. A starting point is to consider that there are $C(N + H, H) \triangleq (N + H)!/N!H!$ possible partitions of the data. Not every such partition will admit a piecewise-OLS solution which is also continuous, and it is difficult to analytically characterize such solutions.

Using Theorem 4, we can characterize growth of global minima in the overparametrized case. Call a partition Π *lonely* if each piece $\pi_p \in \Pi$ contains at most one datapoint (see **Figure 4**). Then, we can prove the following results:

Lemma 1. *For any lonely partition Π, there are infinitely many parameter settings θ_{BDSO} that induce Π and are global minima with $\tilde{\ell}(\theta_{BDSO}) = 0$. Furthermore, in the overparametrized regime*

2.4. The Gradient and Hessian of the Loss $\ell(\theta_{\mathrm{NN}})$

Previous works has revealed many key insights into the Hessian of the loss surface of neural networks. It has long been empirically known that GD tends to work surprisingly well, and that most or all local minima are actually near-optimal (Nguyen and Hein, 2017), although this is known to depend on the overparametrization ratio and the data distribution (Pennington and Bahri, 2017; Granziol et al., 2019). Flatter local minima have been shown to be connected to better generalization (Sankar

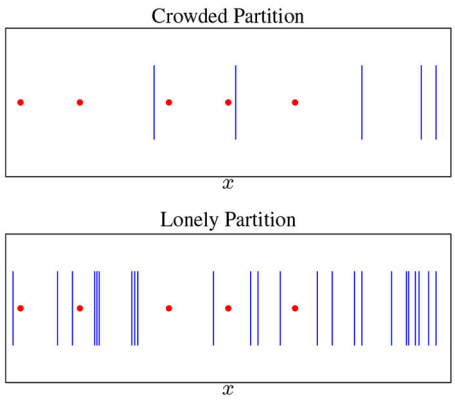

FIGURE 4 | Breakpoints (blue bars) vs. datapoints (red points). A lonely partition is when every datapoint is isolated—overparametrization makes this increasingly likely.

et al., 2020), but the areas around critical points may be unusually flat, with the bulk of Hessian eigenvalues near 0, and only a few larger outliers (Ghorbani et al., 2019), meaning that many local minima are actually connected within a single basin of attraction (Sagun et al., 2017). Our theory can shed new light on many of these phenomena:

The gradient of $\ell(\theta_{\text{NN}})$ can be expressed as

$$\begin{aligned} \frac{\partial \ell}{\partial b_0} &= -\langle \hat{\boldsymbol{\epsilon}}, 1\rangle & \frac{\partial \ell}{\partial w_i} &= -\langle \hat{\boldsymbol{\epsilon}}_i, v_i \mathrm{x}_i\rangle \\ \frac{\partial \ell}{\partial v_i} &= -\langle \hat{\boldsymbol{\epsilon}}_i, w_i \mathrm{x}_i + b_i 1_i\rangle & \frac{\partial \ell}{\partial b_i} &= -\langle \hat{\boldsymbol{\epsilon}}_i, v_i 1_i\rangle \end{aligned} \tag{5}$$

From this, we can derive expressions for the Hessian of the loss $\ell(\theta_{\text{NN}})$ at any parameter θ_{NN}.

Theorem 5. *Let θ^* be a critical point of $\ell(\cdot)$ such that $\beta_i^*(\theta^*) \neq x_n$ for all $i \in [H]$ and for all $n \in [N]$. Then the Hessian $\mathrm{H}_\ell(\theta^*)$ is the positive semi-definite Gram matrix of the set of $3H+1$ vectors*

$$\mathcal{B} \triangleq \{v_i \mathrm{x}_i, w_i \mathrm{x}_i + b_i 1_i, v_i 1_i\}_{i=1}^{H} \cup \{1\},$$

as shown in eq. (4). Thus, $\mathrm{H}_\ell(\theta^*)$ is positive definite iff the vectors of this set are linearly independent.

Remark 2. *The zero eigenvalues of $\mathrm{H}_\ell(\theta^*)$ correspond to*

1. ***α-scaling:** For each neuron i, the scaling transformation $(w_i, b_i, v_i) \mapsto (\alpha_i w_i, \alpha_i b_i, \alpha_i^{-1} v_i)$ leaves the approximating function $\hat{f}(x)$ (and thus the training loss) invariant, thus generating a 1-dimensional hyperbolic manifold of constant loss; thus, the tangent vector of this manifold will be in the null space of $\mathrm{H}_\ell(\theta)$ for all θ. This manifold is depicted in **Figure 1**.*
2. ***0 singularities:** For each neuron i, if either of w_i or v_i are 0:*
   - *if $v_i = 0$, then $\mu_i = 0$, so that changes to the value of β_i (via either b_i or w_i) do not change $\hat{f}(x)$, thus contributing two 0 eigenvalues*
   - *if $w_i = 0$, then $\mu_i = 0$ and $\beta_i = \pm\infty$, so any changes to b_i or v_i do not change $\hat{f}(x)$, thus contributing two 0 eigenvalues*
3. ***Functional changes:** Any functional change that does not change the value of $\hat{f}(x)$ at the training data will leave the loss unchanged. In the sufficiently overparametrized regime, there are many ways to make such changes. Some examples are shown in* **Figure 5**.

The first two types are artifacts of the NN parametrization, caused by invariance to the α-scaling symmetry.

Additionally, we note that these are all also zero eigenvalues for non-critical points, but that there are other zero eigenvalues for non-critical points. While $\mathrm{H}_\ell(\theta^*)$ does not depend on $\boldsymbol{\epsilon}$ at the above critical points, at other points, $\boldsymbol{\epsilon}$ terms can "cancel out" the corresponding term in eq. (4), yielding additional zero eigenvalues (as well as negative eigenvalues).

From this, we note that for any critical point θ^*, there is a large connected sub-manifold of parameter space of constant loss. Moving along this manifold consists of continuously deforming $\hat{f}(x; \theta^*_{\text{NN}})$ such that the output values at datapoints are left invariant or moving along the α-scaling manifolds which leave $\hat{f}(x; \theta^*_{\text{NN}})$ invariant. The 0 singularities lie at the intersection of every α-scaling manifold for neuron i, corresponding to the $\alpha = 0$ and $\alpha = \infty$ limits.

Consider the μ-only Hessian under the BDSO parametrization, $\frac{\mathrm{d}^2\ell}{\mathrm{d}\boldsymbol{\mu}^2}$, and rewrite our network as $\hat{\mathrm{y}} = \boldsymbol{\Phi}(\mathrm{x}; \boldsymbol{\beta})\boldsymbol{\mu}$, where $\boldsymbol{\Phi}$ is the $N \times H$ feature matrix of activations (or basis functions) $(x_n - \beta_i)_{s_i}$, which is constant with respect to $\boldsymbol{\mu}$. Then, we have $\frac{\mathrm{d}^2\ell}{\mathrm{d}\boldsymbol{\mu}^2} = \boldsymbol{\Phi}^\top\boldsymbol{\Phi}$. Applying an SVD decomposition $\boldsymbol{\Phi} = USV^\top$, then $\boldsymbol{\Phi}^\top\boldsymbol{\Phi} = VS^2V^\top$, such that the eigenvalues of the Hessian will be the singular values of $\boldsymbol{\Phi}$ squared. Thus, we can decompose the vector of delta-slopes $\boldsymbol{\mu} = \boldsymbol{\mu}_r + \boldsymbol{\mu}_n$ where $\boldsymbol{\mu}_r$ is in the range of $\boldsymbol{\Phi}$ while $\boldsymbol{\mu}_n$ is in the nullspace. Thus, $\hat{\mathrm{y}} = \boldsymbol{\Phi}(\mathrm{x}; \boldsymbol{\beta})\boldsymbol{\mu}_r + \boldsymbol{\Phi}(\mathrm{x}; \boldsymbol{\beta})\boldsymbol{\mu}_n = \boldsymbol{\Phi}(\mathrm{x}; \boldsymbol{\beta})\boldsymbol{\mu}_r$, and the gradient with respect to $\boldsymbol{\mu}$ is always constrained to the linear subspace $\boldsymbol{\mu}_r$.

Figure 6A shows a 2-dimensional representation of the loss surface, which is quadratic along $\boldsymbol{\mu}_r$, and constant along $\boldsymbol{\mu}_n$. **Figure 6B** illustrates the effect of varying $\boldsymbol{\beta}$, which changes $\boldsymbol{\Phi}(\mathrm{x}; \boldsymbol{\beta})$ and hence $\boldsymbol{\mu}_r$ and $\boldsymbol{\mu}_n$.

2.4.1. The Flatness of the Hessian

An important question is: How does overparametrization impact (the spectrum of) the Hessian? The symmetry-quotiented spline parametrization enables us to lower bound the number of zero eigenvalues of the Hessian [e.g., the "flatness" (Li et al., 2018)] as follows.

Corollary 3. *Let θ^* be a critical point of $\ell(\cdot)$ such that its data partition is lonely, and either at least one neuron is active on all data or there is at least one pair of oppositely-faced neurons in the same data gap, so that $\mathrm{x} \in \mathrm{span}(\mathcal{B})$. Then, $\ell(\theta^*) = 0$ and $\mathrm{H}_\ell(\theta^*)$ has exactly N non-zero eigenvalues, and thus $3H + 1 - N$ zero eigenvalues.*

Intuitively, as overparametrization H/N increases, the number of neurons with shared activation patterns increases, which in turn means many redundant breakpoints between each pair of datapoints, which increases the number of flat directions

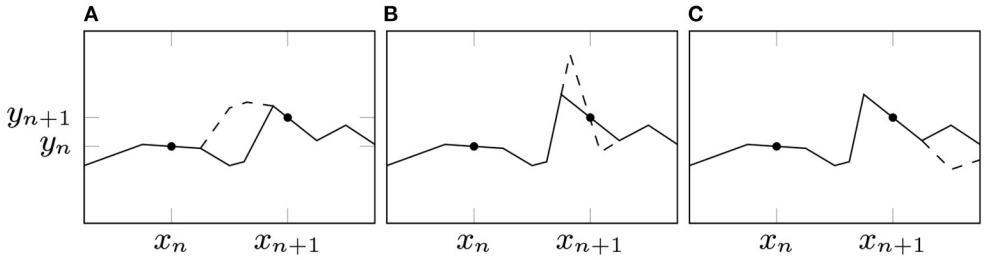

FIGURE 5 | Examples of functional changes that leave the loss invariant: **(A)** when there are multiple breakpoints with the same facing in the interval (x_n, x_{n+1}), they can change in ways that cancel out, leaving the function unchanged on the data; **(B)** if two breakpoints on either side of a datapoint (with no other data in between) have the same facing, they may "rotate" the portion of the function between them around x_n; **(C)** any neuron outside of the data range and facing away (which therefore has no active data) is completely unconstrained by the loss until β_i enters the data range.

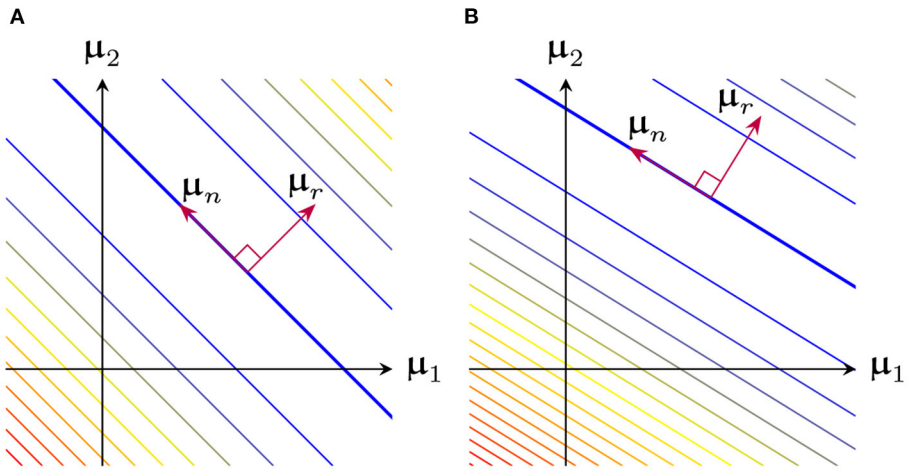

FIGURE 6 | 2-dimensional representation of the loss surface as a function of $\boldsymbol{\mu}$, broken down into a loss-sensitive range and loss-independent nullspace **(A)** Example for fixed $\boldsymbol{\beta}$ **(B)** Same system as before, except $\boldsymbol{\beta}$ was varied, resulting in modified $\boldsymbol{\mu}$ range and nullspace.

(zero eigenvalues) of the Hessian. Additionally, each neuron is guaranteed one degree of freedom in the form of the α-scaling symmetry, leading to H intrinsic zero eigenvalues of the Hessian.

2.5. Gradient Flow Dynamics for Spline Parameters

Theorem 6. *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:*

$$\dot{\beta}_i = \frac{v_i(t)}{w_i(t)} \Big[\underbrace{\langle \hat{\boldsymbol{\epsilon}}_i(t), 1 \rangle}_{\text{net relevant residual}} + \beta_i(t) \underbrace{\langle \hat{\boldsymbol{\epsilon}}_i(t), \mathrm{x} \rangle}_{\text{correlation}} \Big] \tag{6}$$

$$\dot{\mu}_i = w_i^2(t) \left[-\left(1 + \left(\frac{v_i(t)}{w_i(t)} \right)^2 \right) \langle \hat{\boldsymbol{\epsilon}}_i(t), \mathrm{x} \rangle + \beta_i(t) \langle \hat{\boldsymbol{\epsilon}}_i(t), 1 \rangle \right] \tag{7}$$

Here $i \in [H]$ indexes all hidden neurons and the initial conditions $\beta_i(0), \mu_i(0) \forall i \in [H]$ must be specified by the initialization used (see **Appendix B.8** for derivation).

2.5.1. Impact of Init Scale α

As mentioned previously, the standard NN parametrization has symmetries such that the function $\hat{f}$ and the loss is invariant to α-scaling transformations of the NN parameters (section 2.1). However, such scalings do have a large effect on the *initial* NN gradients; specifically, applying the α-scaling transformation in eqs. (6) and (7), the *initial* learning rate for β_i scales as $1/\alpha_i^2$, while that of μ_i scales approximately as α_i^2. Changing α at initialization has a large impact on the final learned solution. In Section 2.6 we will show how α determines the kind of (implicit) regularization seen in NN training (Woodworth et al., 2020).

2.5.2. Breakpoint Dynamics

Next, we wish to build some intuition of the conditions that lead to the different knot types and the effect the types have on

training dynamics and implicit regularization. First, we rewrite eq. (6) as

$$\dot{\beta}_i = \frac{v_i(t)}{w_i(t)} \left\langle \hat{\boldsymbol{\epsilon}}_i(t), 1_i + \beta_i(t) \mathrm{x}_i \right\rangle,$$

and note that the dot product is a cumulative sum of weighted residuals (summing from the furthest datapoint on the active side of neuron i and moving toward β_i). Next, let $\mathcal{A}_i = \{n | 1_{i,n} = 1\}$ denote the set of data indices which are active for neuron i, and $\mathcal{X}_i$ be the active half-space of neuron i [i.e., $(-\infty, \beta_i)$ or (β_i, ∞)]. If we view $\hat{\boldsymbol{\epsilon}}_i(t)$ as $\hat{\boldsymbol{\epsilon}}(\beta_i, t) \triangleq \sum_{n \in \mathcal{A}_i} \hat{\epsilon}_n \delta(\beta_i - x_n)$, we can rewrite the dot product as an integral and write $\dot{\beta}_i$ as a function $\varrho_i(\beta, t)$:

$$\begin{aligned} \varrho_i(\beta, t) &= \frac{v_i(t)}{w_i(t)} \int_{\mathcal{X}_i} \hat{\boldsymbol{\epsilon}}(x, t)(1 + \beta x)\, \mathrm{d}x \\ &\triangleq \frac{v_i(t)}{w_i(t)} \psi_{s_i}(\beta, t) \end{aligned}$$

Finally, note that $\psi_+(\beta, t) = \psi_-(\infty, t) - \psi_-(\beta, t)$, i.e., $\psi_+(\cdot, \cdot)$ is just $\psi_-(\cdot, \cdot)$ reflected across the (finite) total integral; accordingly, we drop the subscript and refer to $\psi(\cdot, \cdot)$ from now on.

Then, to identify Repeller and Attractor knots, we are interested in the datapoints $\beta_i = x_n$ where $\dot{\beta}_i$ changes sign, i.e., we are interested in the zero-crossings of $\psi(\cdot, t)$. Very roughly, at earlier stages of learning, large regions of the input space have high residual, so that many neurons get large, similar updates, leading to broad, smooth changes to $\hat{f}(x; \theta_{\mathrm{NN}})$ and hence broad, smooth changes to $\hat{\boldsymbol{\epsilon}}(x, t)$ and $\psi(\beta, t)$. Applying such a change to $\psi(\beta, t)$ will have the effect of "centering" $\psi(\beta, t)$ such that it has an average value of 0 over large regions. Because the changes are broad at first, the steeper regions of $\psi(\beta, t)$ will retain most of their steepness. Together with the "centering," this leads to the conclusion that the local extrema of $\frac{\partial \psi(\beta, t)}{\partial \beta}$ will be near zero-crossings of $\psi(\beta, t)$.

$$\frac{\partial \psi(\beta, t)}{\partial \beta} = \hat{\boldsymbol{\epsilon}}(\beta, t)(1 + \beta^2) + \int_{-\infty}^{\beta} \hat{\boldsymbol{\epsilon}}(x, t) x \, \mathrm{d}x$$

Due to the broad centering effect, the second term ≈ 0,

$$\approx \hat{\boldsymbol{\epsilon}}(\beta, t)(1 + \beta^2)$$

Therefore, the local extrema of $\hat{\boldsymbol{\epsilon}}(\beta, t)$ will be local extrema of $\frac{\partial \psi(\beta, t)}{\partial \beta}$ and hence near zero-crossings of $\psi(\beta, t)$.

Note that the above analysis shows that $\dot{\beta}_i$ and hence the classification of each datapoint is determined by the same $\psi(\beta, t)$, plus a scaling factor $v_i(t)/w_i(t)$ and possibly a reflection based on $\mathrm{sgn}(w_i)$, i.e., groups of neurons with the same signs of v_i and w_i will yield the same datapoint classification. If datapoint persists in being an Attractor knot for a group of nearby neurons for a long enough time, and is surrounded by Passthrough knots, then those nearby neurons will converge on the datapoint and form a cluster. This leaves the question: what conditions lead to local extrema of $\hat{\boldsymbol{\epsilon}}(\beta, t)$ that persist long enough for this behavior?

Empirically, we find *concentrations* of breakpoints forming near regions of high curvature or discontinuities in the training data, leading to a final fit that was close to a linear spline interpolation of the data. In particular, breakpoints migrate toward nearby curvature which is being underfit and whose inflection is the same as that breakpoint's delta-slope. These clusters can remain fixed for some time, before suddenly shifting (all breakpoints in a cluster move together away from the shared attractor x_n) or splitting (two groups of β form two new sub-clusters that move away from x_n in each direction). These new movements can cause "smearing," when the cluster loses coherence. Sometimes, this "smearing" effect is caused by the need to fit smooth curvature near a cluster: once the residual at the cluster falls below the residual of nearby curvature being underfit by the linear fit provided between clusters, the tendency is for the cluster to spread out; see **Figure 7E**.

As an illustrative example to explore this question, consider fitting a data set such as that shown in **Figure 7**. At initialization (**Figure 7A**), $\hat{f}(x; \theta_{\mathrm{NN}})$ is much flatter than $f(x)$. After a short time of training (**Figure 7B**), $\hat{f}(x; \theta_{\mathrm{NN}})$ has matched the average value of $f(x)$, but is still much flatter, leading to relatively broad regions where $\hat{\boldsymbol{\epsilon}}(\beta, t)$ is large and of the same sign, with zero crossings of $\psi(\beta, t)$ near the first two inflection points. After a little more training (**Figure 7C**), we see that all three inflection points correspond to zero crossings in $\psi(\beta, t)$, and that these zero crossings have persisted long enough that clusters of breakpoints have started to form, although the leftmost inflection point is already fit well enough that $\psi(\beta, t)$ (and hence $\dot{\beta}_i$ is quite low in the region, so that those breakpoints are unlikely to make it "all the way" to a cluster before the data is well fit. Later (**Figure 7D**), the right two inflection points have nearby tight clusters. However, once $\hat{f}(x; \theta_{\mathrm{NN}})$ reaches $f(x)$ at these clusters, the local maxima of $\hat{\boldsymbol{\epsilon}}(\beta, t)$ move away from the clusters, and the clusters "spread out," as shown in the converged fit **Figure 7E**.

2.5.3. Videos of Gradient Descent/Flow Dynamics

We have developed a BDSO spline parametrization-based visualization. For many of the experiments in this paper, the corresponding videos showing the learning dynamics are available at http://shorturl.at/tFWZ2.

2.6. Implicit Regularization

One of the most useful and perplexing properties of deep neural networks has been that, in contrast to other high capacity function approximators, overparametrizing a neural network does not tend to lead to excessive overfitting (Savarese et al., 2019). Where does this generalization power come from? What is the mechanism? One promising idea is that implicit regularization (IR) plays a role. IR acts as if a extra regularization term had been applied to the optimizer, despite no such regularization term being explicitly added by a designer. Instead, IR is added implicitly, brought about by some combination of architecture, initialization scheme, or optimization. Much recent work (Neyshabur et al., 2015, 2018) has argued that this IR is due to the optimization algorithm itself (i.e., SGD). Our symmetry-quotiented spline perspective makes the role of α-scaling symmetry apparent: IR depends critically on breakpoint and delta-slope learning dynamics and initialization. These results follow from section 2.5 , showing that changes to the initialization scale result in dramatic changes to the

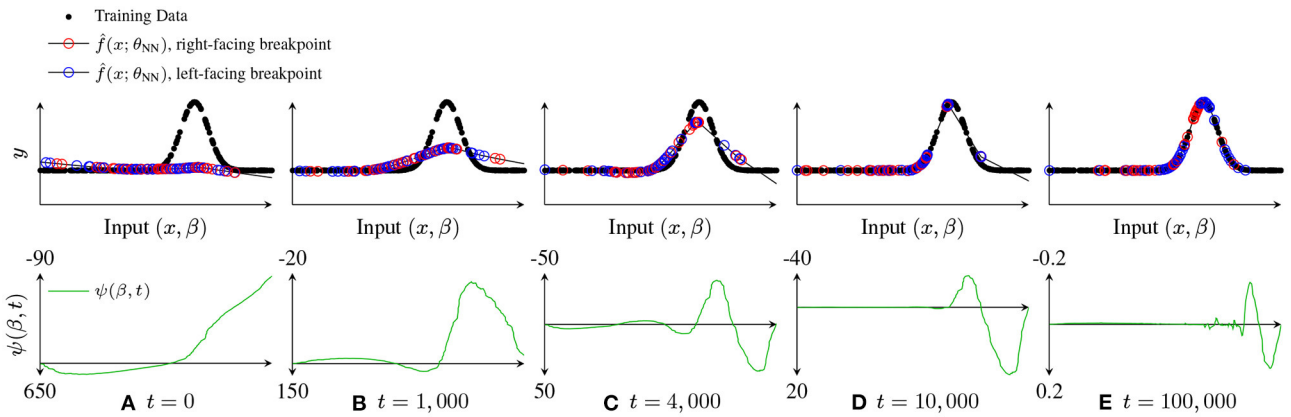

FIGURE 7 | Illustration of breakpoint dynamics, showing how $\hat{f}(x;\theta_{NN})$ and the breakpoint dynamics factor $\psi(\beta,t)$ evolve over time. For $t = 0, 1,000, 4,000, 10,000, 100,000$. **(A)** $t = 0$, **(B)** $t = 1{,}000$, **(C)** $t = 4{,}000$, **(D)** $t = 10{,}000$, and **(E)** $t = 100{,}000$.

relative speeds of learning dynamics for different parameters. In particular, in the kernel regime $\alpha \to \infty$, breakpoints move very little whereas delta-slopes move a lot, resulting in diffuse populations of breakpoints that distribute curvature. In stark contrast, in the adaptive regime $\alpha \to 0$ breakpoints move a lot whereas delta-slopes move very little, resulting in tight clusters of breakpoints that concentrate curvature.

2.6.1. Kernel Regime

We first analyze the so-called kernel regime, inspired by Chizat et al. (2019) where it is referred to as "lazy training." In this section, we omit the overall bias b_0.

Lemma 2. *Consider the dynamics of gradient flow on $\ell(\cdot)$ started from $\theta_{NN,\alpha} \triangleq (\alpha \mathrm{w}_0, \alpha \mathrm{b}_0, \mathrm{v}_0 = 0)$, where $w_i \neq 0 \forall i \in [H]$. In the limit $\alpha \to \infty$, $\boldsymbol{\beta}(t)$ does not change, i.e., each breakpoint location and orientation is fixed. In this case, the θ_{NN} model reduces to a (kernel) linear regression:*

$$\hat{\mathrm{y}} = \boldsymbol{\Phi}(\mathrm{x};\boldsymbol{\beta})\boldsymbol{\mu} \tag{8}$$

where $\boldsymbol{\mu} \in \mathbb{R}^H$ are the regression weights and $\boldsymbol{\Phi}(\mathbf{x};\boldsymbol{\beta}) \in \mathbb{R}^{N\times H}$ are the nonlinear features $\phi_{ni} \triangleq (x_n - \beta_i)_{s_i}$.

Using this, we can then adapt a well known result about linear regression (see e.g., Zhang et al., 2016):

Theorem 7. *Let $\boldsymbol{\mu}^*$ be the converged $\boldsymbol{\mu}$ parameter after gradient flow on the BDSO model eq. (8) starting from some $\boldsymbol{\mu}_0$, with $\boldsymbol{\beta}$ held constant. Furthermore, suppose that the model perfectly interpolates the training data $\tilde{\ell}(\theta_{BDSO}) = 0$. Then,*

$$\boldsymbol{\mu}^* = \arg\min_{\boldsymbol{\mu}} \|\boldsymbol{\mu} - \boldsymbol{\mu}_0\|_2^2 \text{ s.t. } \mathrm{y} = \boldsymbol{\Phi}(\mathrm{x};\boldsymbol{\beta})\boldsymbol{\mu}.$$

Thus, the case where breakpoint locations and orientations are fixed, and $\boldsymbol{\mu}_0 = 0$ reduces to ℓ_2-regularized linear regression on the delta-slope parameters $\boldsymbol{\mu}$. A geometric representation of this process is shown in **Figure 8**. Recalling that $\|\boldsymbol{\mu}\|_2^2$ is the roughness

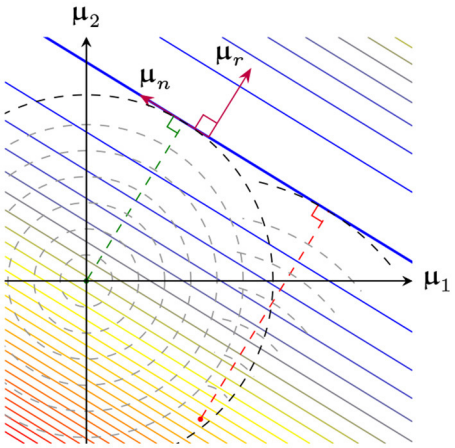

FIGURE 8 | View of the loss surface as in **Figure 6**. GD starting from the origin (green dashed line) monotonically increases $\|\boldsymbol{\mu}\|_2$ (shown as concentric circles). GD starting from some other initialization (red dashed line) minimizes $\|\boldsymbol{\mu} - \boldsymbol{\mu}_0\|_2$ instead (shown as concentric arcs).

of $\hat{f}(x;\theta_{\mathrm{BDSO}})$, this result demonstrates the importance of the initialization, and in particular the initial roughness (Theorem 2): with a high-roughness initialization minimizing $\|\boldsymbol{\mu} - \boldsymbol{\mu}_0\|_2^2$ will yield a high-roughness final fit.

In the overparametrized regime with an initialization that yields a lonely partition, the model will reach $\tilde{\ell}(\theta_{\mathrm{BDSO}}) = 0$. If we consider the infinite-width limit, we get the following result, which is a specialization of Theorem 5 of Williams et al. (2019):

Corollary 4. *Consider the setting of Theorem 7, with the additional assumptions that $\boldsymbol{\mu}_0 = 0$ and the breakpoints are uniformly spaced, and let $H \to \infty$. Then the learned function $\hat{f}_\infty(x;\boldsymbol{\mu}^*,\boldsymbol{\beta}^*)$ is the global minimizer of*

$$\inf_f \int_{-\infty}^{\infty} f''(x)^2\,\mathrm{d}x \text{ s.t. } y_n = f(x_n)\ \forall n \in [N], \tag{9}$$

As such, $\hat{f}_\infty(x; \boldsymbol{\mu}^*, \boldsymbol{\beta}^*)$ is a natural cubic smoothing spline with N degrees of freedom (Ahlberg et al., 1967).

Theorem 5 of Williams et al. (2019), is a generalization of Corollary 4 to the non-uniform case which replaces Equation (9) with

$$\inf_f \int_{-\infty}^{\infty} \frac{f''(x)^2}{p_\beta(x)}\,\mathrm{d}x \text{ s.t. } y_n = f(x_n)\ \forall n \in [N],$$

where $p_\beta(\beta)$ is the initial density of breakpoints induced by the specific initialization used.[2] Deriving this result using the BDSO framework allows us to see that the initialization has the impact of weighting the curvature of certain locations more than others.

2.6.2. Kernel Regime Dynamics as PDE

Assume a flat ($\mu_i(0) = 0\ \forall i \in [H]$) kernel regime initialization. Specializing our spline parameterization dynamics results (Theorem 6), we have

$$\begin{aligned}\dot{\mu}_i(t) &= -\langle \hat{\boldsymbol{\epsilon}}_i(t), \mathrm{x}\rangle + \langle \hat{\boldsymbol{\epsilon}}_i(t), 1\rangle \beta_i \\ &\triangleq r_{1,i}(t) + r_{x,i}(t)\beta_i.\end{aligned}$$

Note that the terms $r_{1,i}(t)$ and $r_{x,i}(t)$ only depend on the index i through the mask 1_i; in other words, they are the same for all breakpoints with the same activation pattern. This pattern is entirely determined by the orientation s_i and the data interval (x_n, x_{n+1}) that β_i falls into.

Thus, the possible activation patterns can be indexed by data point index n and orientation s. Letting $\varsigma_{n,s}$ denote the set of breakpoints with the activation pattern corresponding to (n, s), we have

$$\dot{\mu}_i(t) = r_{1,n,s}(t) + r_{x,n,s}(t)\beta_i \quad i \in \varsigma_{n,s}$$

Let $\mathrm{r}_{1,n,s}(t)$ and $\mathrm{r}_{x,n,s}(t)$ be vectors containing $|\varsigma_{n,s}|$ copies of $r_{1,n,s}(t)$ (resp. $r_{x,n,s}(t)$), and let $\mathrm{r}_1(t)$ and $\mathrm{r}_x(t)$ be the concatenation of these over all n and s. This allows us to write

$$\dot{\boldsymbol{\mu}}(t) = \mathrm{r}_1(t) + \mathrm{r}_x(t) \odot \boldsymbol{\beta},$$

where $\boldsymbol{\mu}$ and $\boldsymbol{\beta}$ are the vectors of μ_i (resp. β_i) values for all i. Then, $\boldsymbol{\mu}(t)$ can be viewed as a function $\boldsymbol{\mu} : [H] \times \{+, -\} \times \mathbb{R} \to \mathbb{R}$, which is isomorphic to the function $\mu : \mathbb{R} \times \{+, -\} \times \mathbb{R} \to \mathbb{R}$ given by $\mu(x, s, t) \triangleq \sum_{i=1}^{H} \mu_i(t)\delta(x - \beta_i)$, yielding the PDE

$$\dot{\mu}(x, s, t) = r_1(x, s, t) + r_x(x, s, t)x, \tag{10}$$

where $r_1(x, s, t)$ and $r_x(x, s, t)$ are piecewise constant functions of x with discontinuities at datapoints. Because $r_1(x, s, t)$ is piecewise constant in x for all t, $\int_0^t r_1(x, s, \tau)\,\mathrm{d}\tau$ will also be piecewise constant, and likewise for $r_x(x, s, t)$. Thus, $\mu(x, s, t)$ will be (discontinuous) piecewise linear in x for all t. Stepping back, we see that a finite-width ReLU network trained by gradient descent is a discretization (in both space and time) of this underlying continuous PDE.

[2] In the original formulation (Williams et al., 2019), the $p_\beta(x)$ term is defined much more opaquely as $v(x) \triangleq \int |w| p(b = ax|w)\,\mathrm{d}p(w)$.

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

Init	Sine	Quadratic
Standard	4.096 ± 2.25	0.1032 ± 0404
Uniform	2.280 ± 0.457	0.1118 ± 0.0248

Thus, $\hat{f}"_\infty(x; t) = \mu(x, +, t) + \mu(x, -, t)$, and as $\mu(x, s, t)$ is (discontinuous) piecewise linear, this implies that $\hat{f}_\infty(x; t)$ will be continuous piecewise cubic in x for all t, consistent with Corollary 4.

2.6.3. Adaptive Regime

Previous work (Arora et al., 2019a) has shown that the kernel regression regime is insufficient to explain the generalization achieved by modern DL. Instead, the non-convexity of the optimization (i.e., the adaptive regime, $\alpha \to 0$, called the "rich regime" in Woodworth et al., 2020) must be responsible. The spline parametrization clearly separates the two, with all adaptive regime effects due to breakpoint dynamics $\beta_i(t)$: as $\alpha \to 0$, breakpoints become more mobile, and the breakpoint dynamics described in section 2.5 becomes more and more dominant. As shown in section 2.3, high breakpoint mobility can lead to clusters of breakpoints forming at certain specific data points. For low enough α, this clustering of breakpoints yielding a fit that is more linear spline than cubic spline; intermediate values of α interpolates these two extremes, giving a fit that has higher curvature in some areas than others.

Intriguingly, this suggests an explanation for why the kernel regime is insufficient: the adaptive regime enables breakpoint mobility, allowing the NN to adjust the initial breakpoint distribution (and thus, basis of activation patterns). This suggests that data-dependent breakpoint initializations may be quite useful (see **Table 1** for a simple experiment in this vein).

Spline theory traditionally places a knot at each datapoint for smoothing splines (James et al., 2013). However, in circumstances where this is not feasible, either computationally or due to using a different spline, a variety of methods exist (MLE, at set quantiles of the regression variable, greedy algorithms, and more) (Park, 2001; Ruppert, 2002; Walker et al., 2005). More recent work has started to develop adaptive splines, where knots are placed to minimize roughness, length, or a similar metric, and knot locations are found via Monte Carlo methods or evolutionary algorithms (Miyata and Shen, 2005; Goepp et al., 2018). This adaptivity allows the spline to better model the function by moving knots to areas of higher curvature – remarkably similar to the behavior of a low-α NN.

In general, an individual breakpoint does not have a large impact on the overall function ($O(1/H)$ on average); it is only through the cooperation of many breakpoints that large changes to the function are made. Another way of formulating this distinction is that the function (and hence the loss) depend on the breakpoints only through the empirical joint density $\hat{p}_H(\beta, \mu, s)$. Similarly, training dynamics depend on the θ_{NN} joint density $\hat{p}_H(w, v, b)$. This formulation is explored in Mei et al. (2018),

where they derive a dynamics equation for the joint density (in θ_{NN}). Adapting this work to explore the dynamics of the θ_{BDSO} joint density is ongoing work.

2.6.4. Comparison With Concurrent Work

Independent of and concurrent with previous versions (Sahs et al., 2020a,b) of this work, Williams et al. (2019) has implicit regularization results in the kernel and adaptive regimes which parallel our results in this section rather closely. Despite the similarities, we take a significantly different approach. Comparing our results to those in Williams et al. (2019), the key differences are: (1) our BDSO parametrization has a clear geometric/functional interpretation whereas Williams *et al.*'s canonical parameters are opaque. (2) BDSO generalizes straightforwardly to high dimensions in a conceptually clean manner: oriented breakpoints become oriented breakplanes (see section 2.7); it is not clear what the multivariate analog of the canonical parameters would be. (3) Lemma 2 of Williams et al. (2019) and the surrounding text discuss the existence of what we call Attractor/Repulsor knot datapoints, but only gives an algebraic expression for their formation in terms of residuals and are unable to distinguish between the types, whereas we relate the persistence of such Attractors to the curvature of the ground truth function, particularly when breakpoints cross datapoints. (4) Our proof techniques are quite different from theirs, with different intuitions and intermediate results, and are also conceptually simpler. (5) Theorem 5 of Williams et al. (2019) is slightly more general form of our Corollary 4, extending to the case of non-uniform breakpoints. (6) Finally, we have many novel results regarding initialization, loss surface properties, Hessian and dynamics (everything outside of section 2.6). All in all, we believe our results are quite complementary to those of Williams et al., particularly as we extend our results to novel areas using our Hessian and loss surface analysis.

2.7. Extending to Multivariate Inputs

Throughout this work we chose to focus on the univariate case in order to build intuition and enable simpler theoretical results. However, in practice most networks have multivariate inputs; fortunately, our BDSO parametrization can be easily extended to multivariate inputs. For D-dimensional inputs, write

$$\hat{f}(\mathrm{x};\theta_{\text{NN}}) = \sum_{i=1}^{H} v_i \left(\langle \mathrm{w}_i, \mathrm{x}\rangle + b_i\right)_+ + b_0,$$

where the input weights are represented as D-dimensional vectors w_i. Using the reparametrization $\eta_i \triangleq v_i\|\mathrm{w}_i\|_2, \boldsymbol{\xi}_i \triangleq \frac{\mathrm{w}_i}{\|\mathrm{w}_i\|_2}$, $\gamma_i \triangleq \frac{-b_i}{\|\mathrm{w}_i\|_2}$, the representation becomes

$$\hat{f}(\mathrm{x};\theta_{\text{BDSO}}) = \sum_{i=1}^{H} \eta_i \left(\langle \boldsymbol{\xi}_i, \mathrm{x}\rangle - \gamma_i\right)_+ + b_0,$$

where η_i is a "delta-slope" parameter,[3] and $(\boldsymbol{\xi}_i, \gamma_i)$ parametrize the orientation and signed distance from the origin of a $D-1$-dimensional "oriented breakplane" (generalizing the 0-dimensional left-or-right oriented breakpoint represented by (s_i, β_i) in the 1-dimensional case). Generalizing our results to this parametrization is ongoing work.

[3]For $D = 1$, we have $\eta_i = v_i|w_i|$, differing from the delta-slope $\mu_i = v_i w_i$ used in this paper by the sign of w_i.

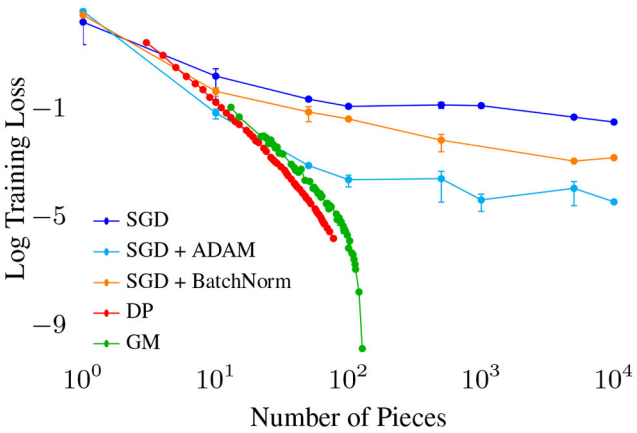

FIGURE 9 | Training loss vs. number of pieces ($\propto$ number of parameters) for various algorithms fitting a piece-wise linear (PWL) function to a quadratic. We observe a strict ordering of optimization quality with for the stochastic gradient descent based algorithms, with Adam (an optimizer with momentum) outperforming BatchNorm SGD outperforming Vanilla SGD.

3. EXPERIMENTS

3.1. Suboptimality of Gradient Descent

Focusing on univariate networks allows us to directly compare against existing (near-) optimal algorithms for fitting Piecewise Linear (PWL) functions, including the Dynamic Programming algorithm (DP, Bai and Perron, 1998), and a very fast greedy approximation known as Greedy Merge (GM, Acharya et al., 2016) (**Figure 9**). Given a fixed budget of pieces ($\propto$ to number of parameters e.g., network width), how well do these algorithms compare to SGD variants in fitting a quadratic (high curvature) function? DP and GM both quickly reduce training error to near 0 with order 100 pieces, with GM requiring slightly more pieces for similar performance. All the GD variants require far more pieces to reduce error below any target threshold, and may not even monotonically decrease with number of pieces. These results show how inefficient GD is w.r.t parameters, requiring orders of magnitude more for similar performance compared with PWL fitting algorithms.

3.2. Effect of Initial Breakpoint Distribution

We first ask whether the standard initializations will experience difficulty fitting functions that have significant curvature away from the origin (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 away from the origin (**Table 1**, first row), consistent with our theory that breakpoints are essential for modeling curvature. Probing further, we observe empirically that breakpoints cannot

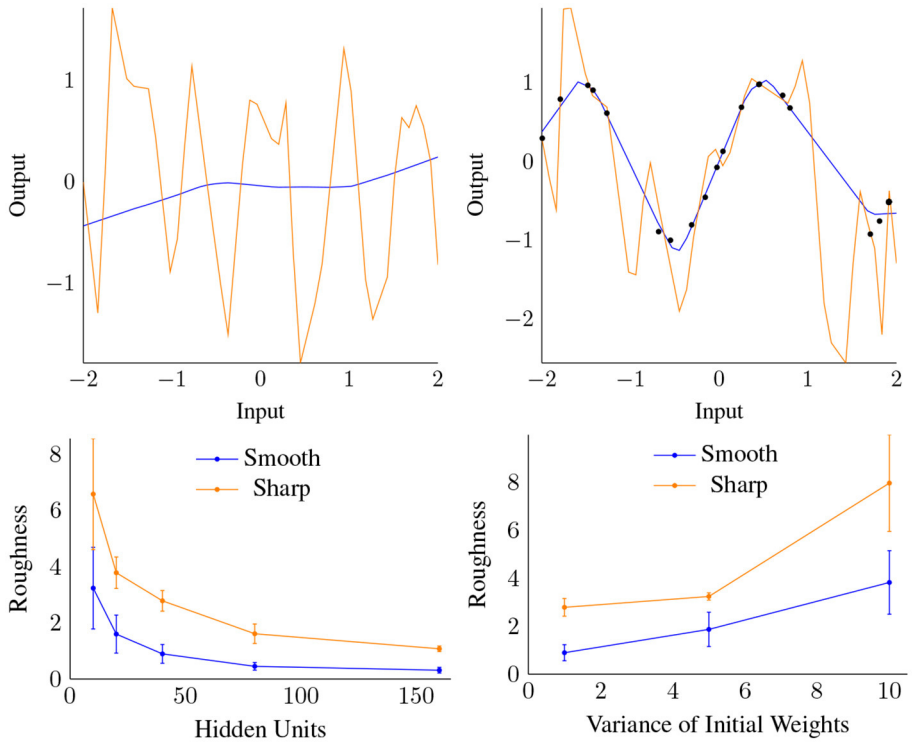

FIGURE 10 | Top: "Spiky" (orange) and standard initialization (blue), compared before (left) and after (right) training. Note both cases reached similar, very low training set error. Bottom: Roughness vs. Width (left) and the variance of the initialization (right) for both data gap cases shown in **Figure 11**. Each datapoint is the result of averaging over 4 trials trained to convergence.

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 (**Table 1**, second row) 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.

3.3. Generalization: Implicit Regularization Emerges From Flat Init and Curvature-Based Parametrization

Our theory predicts that IR depends critically upon flatness of the initialization (Theorem 7 and corollary 4). Here, we test this theory for the case of shallow and deep univariate ReLU nets. We compare training with the standard flat initialization to a "spiky" initialization, and find that both fit the training data near perfectly, but that the "spiky" initialization has much worse generalization error (**Figure 10** Top and Table 3 in **Appendix**).

It appears that generalization performance is not automatically guaranteed by GD, but is instead due in part to the flat initializations which are then *preserved* by GD. Our theoretical explanation is simple: integrating the dynamics in Equation (7) yields $\mu_i(t) = \mu_i(0) + \cdots$ and so the initialization's impact remains.

3.4. Impact of Width and Init Variance

Next, we examine how smoothness (roughness) depends on the width H, focusing on settings with large gaps in the training data. We use two discontinuous target functions (shown in **Figure 11**), leading to a gap in the data, and test how increasing H (with α unchanged) affects the smoothness of $\hat{f}$. We test this on a "smooth" data gap that is easily fit, as well as a "sharp" gap, where the fit will require a sharper turn. We trained shallow ReLU networks with varying width H and initial weight variance σ_w until convergence, and measured the total roughness of resulting CPWL approximation in the data gaps.

Figure 10 Bottom shows that roughness in the data gaps decreases with width and increases with initial weight variance, confirming our theory. A higher weight variance, and thus rougher initialization, acts in a similar fashion to the "spiky" initialization, and leads to increased roughness at convergence. In contrast, higher width distributes the curvature "work" over more units, leading to lower overall roughness.

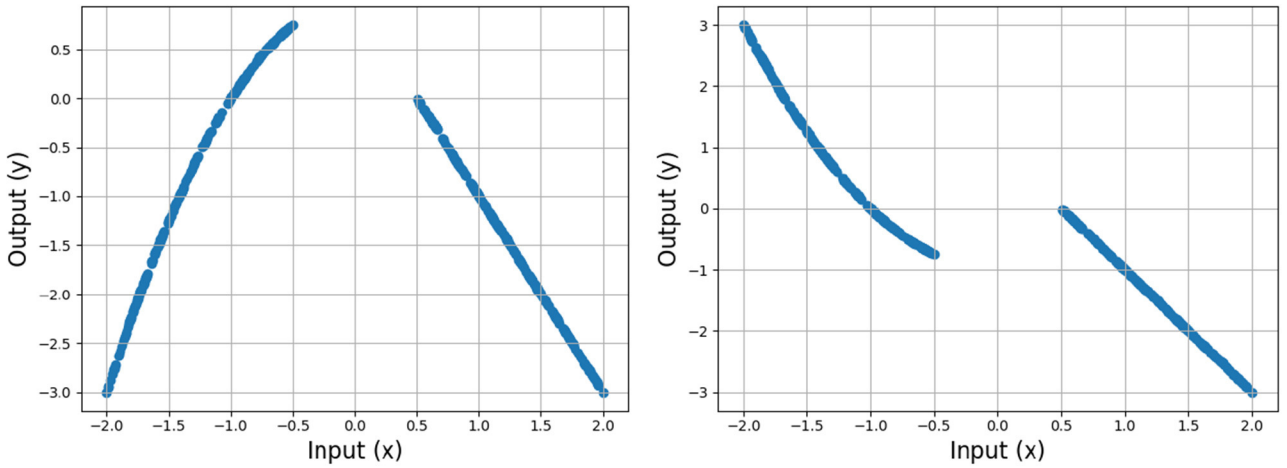

FIGURE 11 | Training data sampled from two ground truth functions, one smoothly (left) and the other sharply (right) discontinuous, each with a data gap at $[-0.5, 0.5]$.

3.5. Impact of Init Scale α

Finally, we explore how changing α impacts IR. Empirically, as α increases from 0 to ∞ we see three qualitative regimes: (i) an *under*fitting linear spline, (ii) an interpolating linear spline, and (iii) a roughness-minimizing natural cubic spline. This is quantified in **Table 2**, where we compare the NN fit to a linear interpolation and a natural cubic spline fit, for varying α. We first test in the special case that the initial function approximation is perfectly flat; we find excellent agreement with the linear interpolation and cubic spline fits for $\alpha = 3,100$ (Uniform initialization) and $\alpha = 10,100$ (He initialization). The impact of α on IR can be more easily visualized in our Supplementary Videos. In order to gauge the impact of the initialization breakpoint distribution, we also test with a standard He initialization (Cauchy distributed breakpoints, Corollary 1). In this case, we find that generalization error is uniformly higher for all α. More strikingly, the α regime (ii) above disappears, as a result of breakpoints being more concentrated around the origin and the initialization roughness being significantly nonzero. This supports the idea that the initial parameter settings, in particular the breakpoint distribution, has a critical impact on the final fit and its IR.

Taken together, our experiments strongly support that a smooth, flat initialization and overparametrization are both responsible for the phenomenon and strength of IR, while the initialization weight scale α critically determines the type of IR.

TABLE 2 | Comparison of neural network trained to near 0 training loss on random data against linear interpolation and natural cubic splines for varying α, with uniform initialization (top) and standard He (bottom).

α	MAE vs. Linear	RMSE vs. Linear	MAE vs. Cubic	RMSE vs. Cubic
0.1	0.251 ± 0.077	0.370 ± 0.11	0.326 ± 0.12	0.442 ± 0.16
1	0.137 ± 0.060	0.228 ± 0.074	0.199 ± 0.084	0.282 ± 0.12
3	0.0296 ± 0.0083	0.0749 ± 0.018	0.117 ± 0.034	0.158 ± 0.048
10	0.122 ± 0.027	0.157 ± 0.029	0.0341 ± 0.012	0.0481 ± 0.019
100	0.159 ± 0.042	0.210 ± 0.055	0.0299 ± 0.011	0.0501 ± 0.024
0.1	0.202 ± 0.079	0.320 ± 0.13	0.293 ± 0.11	0.418 ± 0.16
1	0.134 ± 0.064	0.233 ± 0.11	0.211 ± 0.10	0.308 ± 0.15
3	0.132 ± 0.065	0.239 ± 0.12	0.209 ± 0.089	0.329 ± 0.15
10	0.115 ± 0.046	0.163 ± 0.061	0.0884 ± 0.052	0.149 ± 0.11
100	0.161 ± 0.048	0.212 ± 0.055	0.0556 ± 0.015	0.0828 ± 0.021

Mean ± s.d. over 5 random seeds.

4. DISCUSSION

We show that removing the α-scaling symmetry and examining neural networks in spline space enabled us to glean new theoretical and practical insights. The spline view highlights the importance of initialization: a smooth initial approximation is required for a smooth final solution. Fortunately, existing initializations used in deep learning practice approximate this property. In spline space, we also achieve a surprisingly simple and transparent view of the loss surface, its critical points, its Hessian, and the phenomenon of overparametrization. It clarifies how increasing width relative to data size leads with high probability to lonely data partitions, which in turn are more likely to reach global minima. The spline view also allows us to explain the phenomenon of implicit regularization, and how it arises due to overparametrization and the initialization scale α.

4.1. Related Work

Our approach is related to previous work (Frankle and Carbin, 2018; Arora et al., 2019b; Savarese et al., 2019) in that we wish to characterize parameterization and generalization. We differ from these other works by focusing on small width networks, rather than massively overparametrized or infinite width networks, and by using a spline parameterization to study properties such as smoothness of the approximated function. Previous work (Advani and Saxe, 2017) has hinted at the importance of low norm initializations, but the

spline perspective allows us to prove implicit regularization properties in shallow networks. Finally, Williams et al. (2019) is closely related and is discussed in detail at the end of section 2.6.

4.2. Explanation for Correlation Between Flatness of Minima and Generalization Error

A key unexplained empirical observation has been that flatter local minima tend to generalize better (Li et al., 2018; Wei and Schwab, 2019). Our results above provide an explanation: as overparametrization $\mathcal{O} = H/N$ increases, the flatness of the local minima (as measured by the number of zero eigenvalues) increases (Corollary 3) and the smoothness of the implicitly regularized function (as measured by inverse roughness $\rho(\hat{f}_H) \geq \rho(\hat{f}_\infty)$) also increases. As previously shown (Wu et al., 2017), flatter and simpler local minima imply better generalization. Our work provides a parsimonious explanation for this: as we increase overparametrization, partitions become increasingly lonely, allowing for an increased redundancy in number of ways to exactly fit the training data (thus increasing the number of zero eigenvalues), while the inductive bias of gradient descent spreads the "work" (e.g., curvature changes due to delta-slopes) among many units, ensuring that each unit has a lesser effect and making the loss surface increasingly flat.

4.3. Future Work

Looking forward, there are still many questions to answer from the spline perspective: How does depth affect the expressive power, learnability, and IR? What kinds of regularization are induced in the adaptive regime and how do modern deep nets take advantage of them? How can data-dependent initializations of the breakpoints help rescue/improve the performance of GD? Can we design new global learning algorithms inspired based on breakpoint (re)allocation? We believe the BDSO perspective can help answer these questions.

DATA AVAILABILITY STATEMENT

The original contributions presented in the study are included in the article/**Supplementary Material**, further inquiries can be directed to the corresponding author/s.

AUTHOR CONTRIBUTIONS

JS led effort on theory section, with help from RP and AP. RP led effort on experiments, with help from all other authors. JS and RP were responsible for manuscript. All authors contributed to the article and approved the submitted version.

FUNDING

RP and AP were supported by Intelligence Advanced Research Projects Activity (IARPA) *via* Department of Interior/Interior Business Center (DoI/IBC) contract number D16PC00003. JS and AP were supported by NIH grant no. 1UF1NS111692-01. RP and AP were supported by funding from NSF NeuroNex grant no. 1707400.

SUPPLEMENTARY MATERIAL

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/frai.2022.889981/full#supplementary-material

REFERENCES

Acharya, J., Diakonikolas, I., Li, J., and Schmidt, L. (2016). Fast algorithms for segmented regression. *arXiv [Preprint]. arXiv*:1607.03990. doi: 10.48550/arXiv.1607.03990

Advani, M. S., and Saxe, A. M. (2017). High-dimensional dynamics of generalization error in neural networks. *arXiv [preprint]. arXiv*:1710.03667. doi: 10.48550/arXiv.1710.03667

Advani, M. S., Saxe, A. M., and Sompolinsky, H. (2020). High-dimensional dynamics of generalization error in neural networks. *Neural Netw.* 132, 428–446.

Ahlberg, J. H., Nilson, E. N., and Walsh, J. L. (1967). The theory of splines and their applications. *Math. Sci. Eng*. 38, 1–276.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R., and Wang, R. (2019a). On exact computation with an infinitely wide neural net. *arXiv [preprint]. arXiv*:1904.11955. doi: 10.48550/arXiv.1904.11955

Arora, S., Du, S. S., Hu, W., Li, Z., and Wang, R. (2019b). Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. *arXiv [preprint]. arXiv*:1901.08584. doi: 10.48550/arXiv.1901.08584

Badrinarayanan, V., Mishra, B., and Cipolla, R. (2015). Symmetry-invariant optimization in deep networks. *arXiv [preprint]. arXiv*:1511.01754. doi: 10.48550/arXiv.1511.01754

Bai, J., and Perron, P. (1998). Estimating and testing linear models with multiple structural changes. *Econometrica* 66, 47–78. doi: 10.2307/2998540

Balestriero, R., and Baraniu, R. G. (2018). "A spline theory of deep networks," in *International Conference on Machine Learning*, (Stockholm, Sweden) 383–392.

Barbosa, W. A., Griffith, A., Rowlands, G. E., Govia, L. C., Ribeill, G. J., Nguyen, M.-H., et al. (2021). Symmetry-aware reservoir computing. *Phys. Rev. E* 104, 045307. doi: 10.1103/PhysRevE.104.045307

Bertoni, F., Montobbio, N., Sarti, A., and Citti, G. (2021). Emergence of lie symmetries in functional architectures learned by cnns. *Front. Comput. Neurosci.* 15, 694505 doi: 10.3389/fncom.2021.694505

Chizat, L., Oyallon, E., and Bach, F. (2019). "On lazy training in differentiable programming," in *Advances in Neural Information Processing Systems* (Vancouver, Canada), 2933–2943.

Frankle, J., and Carbin, M. (2018). The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv [preprint]. arXiv*:1803.03635. doi: 10.48550/arXiv.1803.03635

Geiger, M., Jacot, A., Spigler, S., Gabriel, F., Sagun, L., d'Ascoli, S., et al. (2020). Scaling description of generalization with number of parameters in deep learning. *J. Stat. Mech.* 2020, 023401.

Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). An investigation into neural net optimization via hessian eigenvalue density. *arXiv [preprint]. arXiv*:1901.10159. doi: 10.48550/arXiv.1901.10159

Glorot, X., and Bengio, Y. (2010). "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (Sardinia, Italy), 249–256.

Goepp, V., Bouaziz, O., and Nuel, G. (2018). Spline regression with automatic knot selection. *arXiv [preprint]. arXiv:*1808.01770. doi: 10.48550/arXiv.1808.01770

Granziol, D., Garipov, T., Vetrov, D., Zohren, S., Roberts, S., and Wilson, A. G. (2019). *Towards Understanding the True Loss Surface of Deep Neural Networks Using Random Matrix Theory and Iterative Spectral Methods*. Available online at: https://openreview.net/forum?id=H1gza2NtwH.

Hanin, B., and Rolnick, D. (2019). Deep relu networks have surprisingly few activation patterns. *arXiv [preprint]. arXiv:*1906.00904. doi: 10.48550/arXiv.1906.00904

He, K., Zhang, X., Ren, S., and Sun, J. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision* (Santiago: IEEE), 1026–1034.

Jacot, A., Gabriel, F., and Hongler, C. (2018). "Neural tangent kernel: Convergence and generalization in neural networks," in *Advances in Neural Information Processing Systems* (Montreal, Canada), 8571–8580.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *An Introduction to Statistical Learning, Vol. 112*. Springer.

Kunin, D., Sagastuy-Brena, J., Ganguli, S., Yamins, D. L., and Tanaka, H. (2020). Neural mechanics: Symmetry and broken conservation laws in deep learning dynamics. *arXiv [preprint]. arXiv:*2012.04728. doi: 10.48550/arXiv.2012.04728

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. (2017). Deep neural networks as gaussian processes. *arXiv [preprint]. arXiv:*1711.00165. doi: 10.48550/arXiv.1711.00165

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. (2018). "Visualizing the loss landscape of neural nets," in *Advances in Neural Information Processing Systems* (Montreal, Canada), 6389–6399.

Liu, G., Yang, C., Li, Z., Ceylan, D., and Huang, Q. (2016). Symmetry-aware depth estimation using deep neural networks. *arXiv [preprint]. arXiv:*1604.06079. doi: 10.48550/arXiv.1604.06079

Liu, S., and Okatani, T. (2021). Symmetry-aware neural architecture for embodied visual navigation. *arXiv [preprint]. arXiv:*2112.09515. doi: 10.48550/arXiv.2112.09515

Mei, S., Montanari, A., and Nguyen, P.-M. (2018). A mean field view of the landscape of two-layer neural networks. *Proc. Natl. Acad. Sci. U.S.A.* 115, E7665–7671. doi: 10.1073/pnas.1806579115

Miyata, S., and Shen, X. (2005). Free-knot splines and adaptive knot selection. *J. Jpn. Stat. Soc.* 35, 303–324. doi: 10.14490/jjss.35.303

Neal, R. M. (1994). *Priors for Infinite Networks*. Technical report.

Neyshabur, B., Li, Z., Bhojanapalli, S., LeCun, Y., and Srebro, N. (2018). "The role of over-parametrization in generalization of neural networks," in *International Conference on Learning Representations* (New Orleans, USA).

Neyshabur, B., Salakhutdinov, R. R., and Srebro, N. (2015). "Path-sgd: Path-normalized optimization in deep neural networks," in *Advances in Neural Information Processing Systems* (Montreal, Canada), 2422–2430.

Nguyen, Q., and Hein, M. (2017). The loss surface of deep and wide neural networks. *arXiv [preprint]. arXiv:*1704.08045. doi: 10.48550/arXiv.1704.08045

Park, H. (2001). Choosing nodes and knots in closed b-spline curve interpolation to point data. *Comput. Aided Design* 33, 967–974. doi: 10.1016/S0010-4485(00)00133-0

Pennington, J., and Bahri, Y. (2017). "Geometry of neural network loss surfaces via random matrix theory," in *International Conference on Machine Learning* (Sydney, Australia), 2798–2806.

Rao, R. (2002). "Wavelet transforms and multirate filtering," in *Multirate Systems: Design and Applications (IGI Global)*, 86–104. doi: 10.4018/978-1-930708-30-3.ch003

Reinsch, C. H. (1967). Smoothing by spline functions. *Numer. Math.* 10, 177–183. doi: 10.1007/BF02162161

Rolnick, D., and Kording, K. P. (2019). Identifying weights and architectures of unknown relu networks. *arXiv[Preprint]. arXiv:*1910.00744.

Ruppert, D. (2002). Selecting the number of knots for penalized splines. *J. Comput. Graph. Stat.* 11, 735–757. doi: 10.1198/106186002853

Sagun, L., Evci, U., Guney, V. U., Dauphin, Y., and Bottou, L. (2017). Empirical analysis of the hessian of over-parametrized neural networks. *arXiv [preprint]. arXiv:*1706.04454. doi: 10.48550/arXiv.1706.04454

Sahs, J., Damaraju, A., Pyle, R., Tavaslioglu, O., Caro, J. O., Lu, H. Y., et al. (2020a). A functional characterization of randomly initialized gradient descent in deep relu networks. *ICLR 2020*. Available online at: https://openreview.net/pdf?id=BJl9PRVKDS

Sahs, J., Pyle, R., Damaraju, A., Caro, J. O., Tavaslioglu, O., Lu, A., et al. (2020b). Shallow univariate relu networks as splines: Initialization, loss surface, hessian, and gradient flow dynamics. *arXiv [preprint]. arXiv:*2008.01772. doi: 10.48550/arXiv.2008.01772

Sankar, A. R., Khasbage, Y., Vigneswaran, R., and Balasubramanian, V. N. (2020). A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *arXiv [preprint]. arXiv:*2012.03801. doi: 10.48550/arXiv.2012.03801

Savarese, P., Evron, I., Soudry, D., and Srebro, N. (2019). How do infinite width bounded norm networks look in function space? *arXiv [preprint]. arXiv:*1902.05040. doi: 10.48550/arXiv.1902.05040

Steinwart, I. (2019). A sober look at neural network initializations. *arXiv [preprint]. arXiv:*1903.11482. doi: 10.48550/arXiv.1903.11482

Tayal, K., Lai, C.-H., Kumar, V., and Sun, J. (2020). Inverse problems, deep learning, and symmetry breaking. *arXiv [preprint]. arXiv:*2003.09077. doi: 10.48550/arXiv.2003.09077

Walker, C., O'Sullivan, M., and Gordon, M. (2005). "Determining knot location for regression splines using optimisation," in *40th Annual Conference* (Citeseer) (Wellington, New Zealand), 225.

Wei, M., and Schwab, D. J. (2019). How noise affects the hessian spectrum in overparameterized neural networks. *arXiv [preprint]. arXiv:*1910.00195. doi: 10.48550/arXiv.1910.00195

Williams, F., Trager, M., Panozzo, D., Silva, C., Zorin, D., and Bruna, J. (2019). "Gradient dynamics of shallow univariate relu networks," in *Advances in Neural Information Processing Systems* (Vancouver, Canada), 8376–8385.

Woodworth, B., Gunasekar, S., Lee, J. D., Moroshko, E., Savarese, P., Golan, I., et al. (2020). "Kernel and rich regimes in overparametrized models," in *Conference on Learning Theory (PMLR)*, 3635–3673.

Wu, L., Zhu, Z., and Weinan, E. (2017). Towards understanding generalization of deep learning: perspective of loss landscapes. *arXiv [preprint]. arXiv:*1706.10239. doi: 10.48550/arXiv.1706.10239

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. (2016). Understanding deep learning requires rethinking generalization. *arXiv [preprint]. arXiv:*1611.03530. doi: 10.48550/arXiv.1611.03530

Conflict of Interest: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.