# Learning sparse features can lead to overfitting in neural networks

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

It is widely believed that the success of deep networks lies in their ability to learn a meaningful representation of the features of the data. Yet, understanding when and how this feature learning improves performance remains a challenge: for example, it is beneficial for modern architectures trained to classify images, whereas it is detrimental for fully-connected networks trained for the same task on the same data. Here we propose an explanation for this puzzle, by showing that feature learning can perform worse than lazy training (via random feature kernel or the NTK) as the former can lead to a sparser neural representation. Although sparsity is known to be essential for learning anisotropic data, it is detrimental when the target function is constant or smooth along certain directions of input space. We illustrate this phenomenon in two settings: *(i)* regression of Gaussian random functions on the $d$-dimensional unit sphere and *(ii)* classification of benchmark datasets of images. For *(i)*, we compute the scaling of the generalization error with number of training points, and show that methods that do not learn features generalize better, even when the dimension of the input space is large. For *(ii)*, we show empirically that learning features can indeed lead to sparse and thereby less smooth representations of the image predictors. This fact is plausibly responsible for deteriorating the performance, which is known to be correlated with smoothness along diffeomorphisms.

## 1 Introduction

Neural networks are responible for a technological revolution in a variety of machine learning tasks. Many such tasks require learning a high-dimensional function from a finite set of input data, thus should be generically hard due to the *curse of dimensionality* [1, 2]: understanding the success of neural networks is still an open question. A popular explanation for this success is that, during training, neurons adapt to features in the data that are relevant for the task [3], effectively reducing the input dimension and making the problem tractable [4–6]. However, understanding quantitatively if this intuition is true and how it depends on the structure of the task remains a challenge.

Recently much progress was made in characterizing the conditions which lead to features learning, in the overparameterized setting where networks generally perform best. When the initialization scale of the network parameters is large [7] one encounters the *lazy training regime*, where neural networks behave as kernel methods [8, 9] (coined Neural Tangent Kernel or NTK) and features are not learnt. By contrast, when the initialization scale is small, a *feature learning regime* is found [10–12] where the network parameters evolve significantly during training. This limit is much less understood apart from very simple architectures, where it can be shown to lead to sparse representations where a limited number of neurons are active after training [13]. Such sparse representations can also be obtained by regularizing the weights during training [2, 14].

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

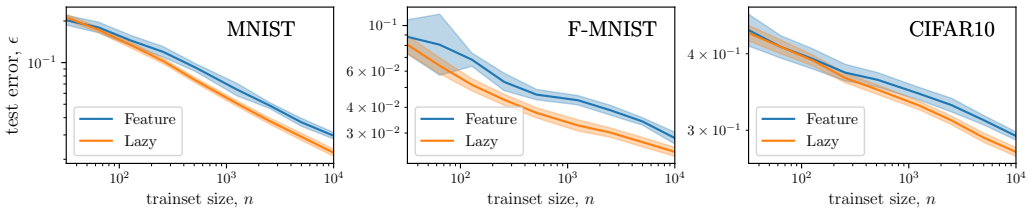



Figure 1: **Feature vs. Lazy in image classification.** Test error as a function of the training-set size $n$ for infinite-width FCNs trained in the feature (blue) and lazy regime (orange). In the latter case the limit is taken exactly by training an SVC algorithm with the analytical NTK [22]. In the former case, the infinite-width limit can be accurately approximated for these datasets by considering very wide nets ($H = 10^3$), and performing ensemble averaging on different initial conditions of the parameters as shown in [23, 24]. Panels correspond to different benchmark image datasets [25–27]. Results are averaged over 10 different initializations of the networks and datasets.

In terms of performance, most theoretical works have focused on fully-connected networks. For these architectures, feature learning was shown to significantly outperform lazy training [15–18, 10] for certain tasks, including approximating a function which depends only on a subset or a linear combination of the input variables. However, when such primitive networks are trained on image datasets, learning features is detrimental [19, 20], as illustrated in Fig. 1. A similar result was observed in simple models of data [21]. These facts are unexplained, yet central to understand the implicit bias of the feature learning regime.

### 1.1 Our contribution

Our central argument is that learning a sparse representation of data can be detrimental. This phenomenon occurs when the network requires a continuous density of neurons to represent an invariance to some transformation in input space. If the target function is smooth along this transformation, then the sparse representation results in a predictor which is less smooth and thus sub-optimal. We build our argument as follows.

- We consider the regression problem of a random target function of controlled smoothness on the $d$-dimensional unit sphere, and study the property of the minimizers of the empirical loss with $n$ observations, both in the lazy and the feature learning regimes. Our results on feature learning rely on solutions having an atomic support. This property can be justified for a one-hidden-layer neural networks with ReLU activations and weight decay. Yet, we also find such a sparsity empirically using gradient descent in the absence of regularization, if weights are initialized to be small enough.

- We find that lazy training leads to smoother predictors than feature learning. As a result, lazy training outperforms feature learning when the target function is also sufficiently smooth. Otherwise, the performances of the two methods is comparable, in the sense that they display the same asymptotic decay of generalization error with number of training examples. Our predictions are obtained from asymptotic arguments that we systematically back up with numerical studies.

- For image datasets, it is believed that diffeomorphisms of images are key transformations along which the predictor function should only mildly vary to obtain good performance [28]. From the results above, a natural explanation as to why lazy beats feature for fully connected networks is that it leads to predictors with smaller variations along diffeomorphisms. We confirm that this is indeed the case empirically on benchmark datasets.

Numerical experiments are performed in PyTorch [29], and the code for reproducing experiments is available online at anonymized-github-repository.

### 1.2 Related Work

The property that training ReLU networks in the feature regime leads to a sparse representation was observed empirically [30]. This property can be justified for one-hidden-layer networks by casting training as a L1 minimization problem [31, 2], then using a representer theorem [32, 14, 33].

There is currently no general framework to predict rigorously the learning curve exponent $\beta$ defined as $\epsilon(n) = \mathcal{O}(n^{-\beta})$ for kernels. Some of our asymptotic arguments for can be obtained by other approximations, such as assuming that data points lie on a lattice in $\mathbb{R}^d$ [34], or by using the non-rigorous replica method of statistical physics [35–37]. In the case $d = 2$, we provide a more explicit mathematical formulation of our results, which leads to analytical results for certain kernels. We systematically back up our predictions with numerical tests as $d$ varies.

Finally, in the context of image classification, the connection between performance and 'stability' or smoothness toward small diffeomorphisms of the inputs has been conjectured by [28, 38]. Empirically, a strong correlation between these two quantities was shown to hold across various architectures for deep networks and real datasets [39]. In that reference, it was found that during training fully connected networks lose their stability. Here we show that this effect is much less pronounced in the lazy regime.

## 2 Problem and notation

**Task** We consider a supervised learning scenario with $n$ training points $\{\boldsymbol{x}_i\}_{i=1}^n$ uniformly drawn on the $d$-dimensional unit sphere $\mathbb{S}^{d-1}$. We assume that the target function $f^*$ is an isotropic Gaussian random process on $\mathbb{S}^{d-1}$ and control its statistics via the spectrum: by introducing the decomposition of $f^*$ into spherical harmonics (see App. A for definitions),

$$f^*(\boldsymbol{x}) = \sum_{k\geq 0}\sum_{\ell=1}^{\mathcal{N}_{k,d}} f^*_{k,\ell} Y_{k,\ell}(\boldsymbol{x}) \quad \text{with} \quad \mathbb{E}\left[f^*_{k,\ell}\right] = 0, \quad \mathbb{E}\left[f^*_{k,\ell} f^*_{k',\ell'}\right] = c_k \delta_{k,k'}\delta_{\ell,\ell'}. \tag{2.1}$$

We assume that all the $c_k$ with $k$ odd vanish apart from $c_1$: this is required to guarantee that $f^*$ can be approximated as well as desired with a one-hidden-layer ReLU network with no biases, as discussed in App. A. We also assume that the non-zero $c_k$ decay as a power of $k$ for $k \gg 1$, $c_k \sim k^{-2\nu_t-(d-1)}$. The exponent $\nu_t > 0$ controls the (weak) differentiability of $f^*$ on the sphere (see App. A) and also the statistics of $f^*$ in real space:

$$\mathbb{E}\left[|f^*(\boldsymbol{x}) - f^*(\boldsymbol{y})|^2\right] = O\left(|\boldsymbol{x}-\boldsymbol{y}|^{2\nu_t}\right) = O\left((1-\boldsymbol{x}\cdot\boldsymbol{y})^{\nu_t}\right) \quad \text{as} \quad \boldsymbol{x}\to\boldsymbol{y}. \tag{2.2}$$

**Neural network representations** We consider a *one-hidden-layer neural network* of width $H$,

$$f_H^{\xi}(\boldsymbol{x}) = \frac{1}{H^{1-\xi/2}}\sum_{h=1}^{H}\left(w_h\sigma(\boldsymbol{\theta}_h\cdot\boldsymbol{x}) - \xi w_h^0\sigma(\boldsymbol{\theta}_h^0\cdot\boldsymbol{x})\right), \tag{2.3}$$

where $\{\boldsymbol{\theta}_h\}_{h=1}^H$ (the features) and $\{w_h\}_{h=1}^H$ (the weights) are the network parameters to be optimized, $\{\boldsymbol{\theta}_h^0\}_{h=1}^H$ and $\{w_h^0\}_{h=1}^H$ are their values at initialization, $\xi$ is a parameters fixed to $\xi = 0$ for feature learning and $\xi = 1$ for lazy training, and $\sigma(x)$ denotes the ReLU function, $\sigma(x) = \max\{0, x\}$. We assume that $\{\boldsymbol{\theta}_h^0, w_h^0\}_{h=1}^H$ are drawn independently from a distribution $\mu_0$ with the properties that: all moments of $\mu_0$ exist; $\mu_0$ is absolutely continuous with respect to the Hausdorff measure on $\mathbb{S}^{d-1}\times\mathbb{R}$; and $\mu_0$ is centered, i.e. $\int_{\mathbb{S}^{d-1}\times\mathbb{R}}\boldsymbol{\theta}d\mu_0(\boldsymbol{\theta}, w) = \mathbf{0}$ and $\int_{\mathbb{S}^{d-1}\times\mathbb{R}} w d\mu_0(\boldsymbol{\theta}, w) = 0$.

**Feature Regime ($\xi = 0$)** If we assume that $\{\boldsymbol{\theta}_h, w_h\}_{h=1}^H$ are independently drawn from a probability measure $\mu$ on $\mathbb{S}^{d-1}\times\mathbb{R}$ such that the Radon measure $\gamma = \int_{\mathbb{R}} w\mu(\cdot, dw)$ exists, then as $H\to\infty$

$$\lim_{H\to\infty} f_H^{\xi=0}(\boldsymbol{x}) = \int_{\mathbb{S}^{d-1}}\sigma(\boldsymbol{\theta},\boldsymbol{x})d\gamma(\boldsymbol{\theta}) \qquad \text{a.e. on } \mathbb{S}^{d-1}. \tag{2.4}$$

This is the so-called mean-field limit [10, 11], and it is then natural to determine the optimal $\gamma$ via

$$\gamma^* = \arg\min_{\gamma}\int_{\mathbb{S}^{d-1}}|d\gamma(\boldsymbol{\theta})| \quad \text{subject to} \quad \int_{\mathbb{S}^{d-1}}\sigma(\boldsymbol{\theta}\cdot\boldsymbol{x}_i)d\gamma(\boldsymbol{\theta}) = f^*(\boldsymbol{x}_i) \quad \forall i = 1,\dots,n. \tag{2.5}$$

In practice, we can approximate this minimization problem by using a network with large but finite width, constraining the feature to be on the sphere $|\boldsymbol{\theta}_h| = 1$, and minimizing the following empirical loss with L1 regularization on the weights,

$$\min_{\substack{\{w_h,\theta_h\}_{h=1}^H \\ |\boldsymbol{\theta}_h|=1}} \frac{1}{2n}\sum_{i=1}^n \left(f^*(\boldsymbol{x}_i) - \frac{1}{H}\sum_{h=1}^H w_h\sigma(\boldsymbol{\theta}_h\cdot\boldsymbol{x}_i)\right)^2 + \frac{\lambda}{H}\sum_{h=1}^H |w_h|. \tag{2.6}$$

This minimization problem leads to (2.5) when $H \to \infty$ and $\lambda \to 0$. Note that, by homogeneity of ReLU, (2.6) can be shown to be equivalent to imposing a regularization on the L2 norm of all parameters [31], i.e. the usual weight decay.

To proceed we will make the following assumption about the minimizer $\gamma^*$:

**Assumption 1.** *The minimizer $\gamma^*$ of (2.5) is unique and atomic, with $n_A \leq n$ atoms, i.e. there exists $\{w_i^*, \boldsymbol{\theta}_i^*\}_{i=1}^{n_A}$ such that*

$$\gamma^* = \sum_{i=1}^{n_A} w_i^* \delta_{\boldsymbol{\theta}_i^*} \tag{2.7}$$

The main component of the assumption is the uniqueness of $\gamma^*$; if it holds the sparsity of $\gamma^*$ follows from the representer theorem, see e.g. [32]. Both the uniqueness and sparsity of the minimizer can be justified as holding generically using asymptotic arguments involving recasting the $L1$ minimization problem 2.5 as a linear programming: these arguments are standard (see e.g. [40]) and are presented in App. B for the reader convenience. In our arguments below to deduce the scaling of the generalization error we will mainly use that $n_A = O(n)$—we shall confirm this fact numerically even in the absence of regularization, if the weights are initialized to be small enough.

**Lazy Regime ($\xi = 1$)** Assume that $\boldsymbol{\theta}_h = \boldsymbol{\theta}_h^0 + H^{-1/2}\tilde{\boldsymbol{\theta}}_h + o(H^{-1/2})$ and $w_h = w_h^0 + H^{-1/2}\tilde{w}_h + o(H^{-1/2})$ with $\{\boldsymbol{\theta}_h^0, w_h^0, \tilde{\boldsymbol{\theta}}_h, \tilde{w}_h\}_{h=1}^H$ drawn independently from a measure $d\mu(\boldsymbol{\theta}_0, w_0, \tilde{\boldsymbol{\theta}}, \tilde{w})$ such that $\int_{\mathbb{R}^d} \tilde{\boldsymbol{\theta}}\mu(d\tilde{\boldsymbol{\theta}}, d\tilde{w}, \cdot, \cdot) = \boldsymbol{g}_\theta(\boldsymbol{\theta}, w)d\mu_0$ and $\int_{\mathbb{R}^d} \tilde{w}\mu(d\tilde{\boldsymbol{\theta}}, d\tilde{w}, \cdot, \cdot) = g_w(\boldsymbol{\theta}, w)d\mu_0$ exist. Then

$$\lim_{H\to\infty} f_H^{\xi=1}(\boldsymbol{x}) = \int_{\mathbb{S}^{d-1}\times\mathbb{R}} \left(g_w(\boldsymbol{\theta}, w)\sigma(\boldsymbol{\theta}\cdot\boldsymbol{x}) + w\boldsymbol{x}\cdot\boldsymbol{g}_\theta(\boldsymbol{\theta}, w)\sigma'(\boldsymbol{\theta}\cdot\boldsymbol{x})\right) d\mu_0(\boldsymbol{\theta}, w) \quad \text{a.e. on } \mathbb{S}^{d-1} \tag{2.8}$$

and it is natural to specify $\boldsymbol{g}_\theta$ and $g_w$ as the minimizer of

$$\begin{aligned} &\min_{g_w, \boldsymbol{g}_\theta} \int_{\mathbb{S}^{d-1}\times\mathbb{R}} \left(g_w^2(w, \boldsymbol{\theta}) + |\boldsymbol{g}_\theta(w, \boldsymbol{\theta})|^2\right) d\mu_0(\boldsymbol{\theta}, w) \\ &\text{subject to} \int_{\mathbb{S}^{d-1}\times\mathbb{R}} \left(g_w(w, \boldsymbol{\theta})\sigma(\boldsymbol{\theta}\cdot\boldsymbol{x}_i) + w\boldsymbol{x}_i\cdot\boldsymbol{g}_\theta(w, \boldsymbol{\theta})\sigma'(\boldsymbol{\theta}\cdot\boldsymbol{x}_i)\right) d\mu_0(\boldsymbol{\theta}, w) = f^*(\boldsymbol{x}_i) \quad \forall i = 1, \ldots, n. \end{aligned} \tag{2.9}$$

The solution of Eq. 2.9 coincides with that of kernel ridge regression [41] with the *Neural Tangent Kernel*,

$$K^{\text{NTK}}(\boldsymbol{x}\cdot\boldsymbol{y}) = \int_{\mathbb{S}^{d-1}\times\mathbb{R}} \left(\sigma(\boldsymbol{\theta}\cdot\boldsymbol{x})\sigma(\boldsymbol{\theta}\cdot\boldsymbol{y}) + w^2\,\boldsymbol{x}\cdot\boldsymbol{y}\,\sigma'(\boldsymbol{\theta}\cdot\boldsymbol{x})\sigma'(\boldsymbol{\theta}\cdot\boldsymbol{y})\right) d\mu_0(\boldsymbol{\theta}, w), \tag{2.10}$$

Hence the function $f^{\text{NTK}}(\boldsymbol{x})$ obtained by evaluating the right hand side of (2.8) using the minimizer $\boldsymbol{g}_\theta^*$ and $g_w^*$ of (2.9) can be conveniently recast, via the kernel trick [41], as

$$f^{\text{NTK}}(\boldsymbol{x}) = \sum_{i=1}^n g_i K^{\text{NTK}}(\boldsymbol{x}_i\cdot\boldsymbol{x}) \quad \text{with } f^*(\boldsymbol{x}_j) = \sum_{i=1}^n g_i K^{\text{NTK}}(\boldsymbol{x}_i\cdot\boldsymbol{x}_j), \quad j = 1, \ldots, n. \tag{2.11}$$

Another lazy limit can be obtained equivalently by training only the weights while keeping the features to their initialization value. This is equivalent to forcing $\boldsymbol{g}_\theta(\boldsymbol{\theta}, w)$ to vanish in Eq. 2.9, resulting again in a kernel method. The kernel, in this case, is called *Random Feature Kernel* ($K^{\text{RFK}}$), and can be obtained from Eq. 2.10 by setting $d\mu_0(\boldsymbol{\theta}, w) = \delta_{w=0}dm_0(\boldsymbol{\theta})$. The minimizer can then be written as in Eq. 2.11 with $K^{\text{NTK}}$ replaced by $K^{\text{RFK}}$.

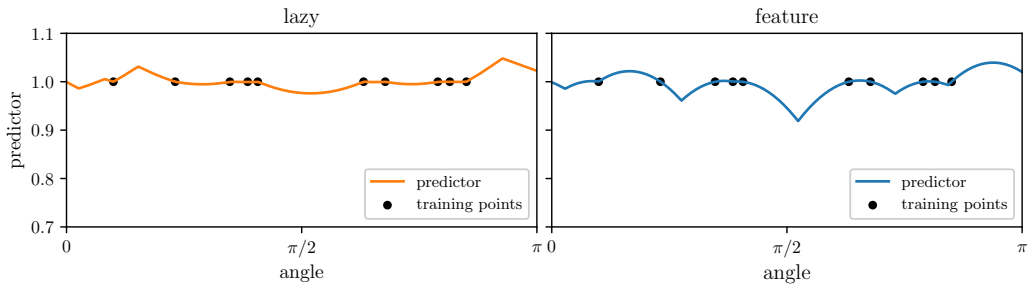



Figure 2: **Feature vs. Lazy Predictor.** Predictor of the lazy (left) and feature (right) regime when learning the constant function on the ring with 8 uniformly-sampled training points.

## 3 Asymptotic analysis of generalization

In this section we characterise the asymptotic decay of the generalization error $\bar{\epsilon}(n)$ averaged over several realizations of the target function $f^*$,

$$\bar{\epsilon}(n) = \mathbb{E}_{f^*}\left[\int d\tau^{d-1}(\boldsymbol{x})\ (f^n(\boldsymbol{x}) - f^*(\boldsymbol{x}))^2\right] = \mathcal{A}_d n^{-\beta} + o(n^{-\beta}), \tag{3.1}$$

for some constant $\mathcal{A}_d$ which might depend on $d$ but not on $n$. Both for the lazy (see Eq. 2.11) and feature regimes (see Eq. 2.7) the predictor can be written as a sum of $O(n)$ terms:

$$f^n(\boldsymbol{x}) = \sum_{j=1}^{O(n)} g_j \varphi(\boldsymbol{x}\cdot\boldsymbol{y}_j) := \int_{\mathbb{S}^{d-1}} g^n(\boldsymbol{y})\varphi(\boldsymbol{x}\cdot\boldsymbol{y})d\tau(\boldsymbol{y}), \tag{3.2}$$

where we have defined the density $g^n(\boldsymbol{x}) = \sum_j |\mathbb{S}^{d-1}| g_j \delta(\boldsymbol{x} - \boldsymbol{y}_j)$ so as to cast the predictor as a convolution on the sphere. Therefore, the projections of $f^n$ onto spherical harmonics $Y_{k,\ell}$ read $f^n_{k,\ell} = g^n_{k,\ell}\varphi_k$, where $g^n_{k,\ell}$ is the projection of $g^n(\boldsymbol{x})$ and $\varphi_k$ that of $\varphi(\boldsymbol{x}\cdot\boldsymbol{y})$. For ReLU neurons one has (as shown in App. E)

$$\varphi_k^{\text{LAZY}} \sim k^{-(d-1)-2\nu} \quad \text{with } \nu = 1/2 \text{ (NTK)}, 3/2 \text{ (RFK)}, \quad \varphi_k^{\text{FEATURE}} \sim k^{-\frac{d-1}{2}-3/2}. \tag{3.3}$$

**Main Result** Consider a target function $f^*$ with smoothness exponent $\nu_t$ as defined above, with data lying on $\mathbb{S}^{d-1}$. If $f^*$ is learnt with a one-hidden-layer network with ReLU neurons in the regimes specified above, then the generalization error follows $\bar{\epsilon}(n) \sim n^{-\beta}$ with:

$$\beta^{\text{LAZY}} = \frac{\min\{2(d-1)+4\nu, 2\nu_t\}}{d-1} \quad \text{with } \nu = \begin{cases} 1/2 \text{ for NTK}, \\ 3/2 \text{ for RFK}, \end{cases}, \tag{3.4a}$$

$$\beta^{\text{FEATURE}} = \frac{\min\{(d-1)+3, 2\nu_t\}}{d-1}. \tag{3.4b}$$

This is our central result. It implies that if the target function is a smooth isotropic Gaussian field (realized for large $\nu_t$), then lazy beats feature, in the sense that training the network in the lazy regime leads to a better scaling of the generalization performance with the number of training points.

**Strategy** There is no general framework for a rigorous derivation of the generalization error in the ridgeless limit $\lambda \to 0$: predictions such as that of Eq. 3.4 can be obtained by either assuming that training points (for Eq. 3.4a) and neurons (for Eq. 3.4b) lie on a periodic lattice [34], or (for Eq. 3.4a) using the replica method from physics [35] as shown in App. F. Here we follow a different route, by first characterizing the form of the predictor for $d = 2$ (proof in App. C). This property alone allows us to determine the asymptotic scaling of the generalization error. We use it to analytically obtain the test error in the NTK case with a slightly simplified function $\varphi$ (details in App. D). This calculation motivates a simple ansatz for the form of $g^n(\boldsymbol{x})$ entering Eq. 3.2 and its projections onto spherical harmonics, which extends naturally to arbitrary dimension. We confirm the predictions resulting from this ansatz systematically in numerical experiments.

**Properties of the predictor in $d = 2$** On the unit circle $\mathbb{S}^1$ all points are identified by a polar angle $x \in [0, 2\pi)$. Hence both target function and estimated predictor are functions of the angle, and all functions of a scalar product are in fact functions of the difference in angle. In particular, introducing $\hat{\varphi}(x) = \varphi(\cos(x))$,

$$f^n(x) = \sum_j g_j \hat{\varphi}(x - x_j) = \int \frac{dy}{2\pi} \left( \sum_j (2\pi g_j) \delta(y - x_j) \right) \hat{\varphi}(x - y) = \int \frac{dy}{2\pi} g^n(y) \hat{\varphi}(x - y), \tag{3.5}$$

Both for feature regime and NTK limit, the first derivative of $\hat{\varphi}(x)$ is continuous except for two values of $x$ (0 and $\pi$ for lazy, $-\pi/2$ and $\pi/2$ for feature), so that $\hat{\varphi}(x)''$ has a singular part consisting of two Dirac delta functions. As a result, the second derivative of the predictor $(f^n)''$ has a singular part consisting of many Dirac deltas: we denote with $(f^n)''_r$ the regular part, obtained by subtracting all the delta functions.

**Proposition 2.** ***(informal)***

$$\lim_{k\to\infty} \lim_{n\to\infty} \frac{(f^n)''_r(k)}{f^*(k)} = c, \tag{3.6}$$

*where $f(k)$ denotes the Fourier transform of $f(x)$ and $c$ is a constant. Thus as $n \to \infty$, $(f^n)''(x)$ converges to a Gaussian field of smoothness $\nu_t$ identical to that of $f^*$.*

In simple terms, this result follows from the convergence of $g^n$ to the function satisfying $f^*(x) = \int \frac{dy}{2\pi} g(y) \hat{\varphi}_r(x - y)$, which is guaranteed under our assumptions on the covariance kernel used to generate target functions—a detailed proof is given in App. C. Proposition 2 implies that, in the large $n$ limit, the predictor has a well-defined curvature outside of singular points. Thus, as $n$ gets large and the singular points approach each other, the predictor can be approximated by a chain of parabolas, as highlighted in Fig. 2 and noticed in [37] for a Laplace kernel. This property alone allows to determine the asymptotic scaling of the error in $d = 2$.

**Decay of the error in $d = 2$ (sketch)** The full calculation is in App. D. Consider a slightly simplified problem where $\hat{\varphi}$ has a single discontinuity in its derivative, located at $x = 0$. In this case, $f^n(x)$ is singular if and only if $x$ is a data point. Consider then the interval $x \in [x_i, x_{i+1}]$ and set $\delta_i = x_{i+1} - x_i$, $x_{i+1/2} = (x_{i+1} + x_i)/2$. If the target function is smooth enough ($\nu_t > 2$), then a Taylor expansion implies $|f^*(x_{i+1/2}) - f^n(x_{i+1/2})| \sim \delta_i^2$. Since the distances $\delta_i$ between adjacent singular points are random variables with mean of order $1/n$ and finite moments, it is straightforward to obtain that $\bar{\epsilon}(n) \sim \sum_i (f^*(x_{i+1/2}) - f^n(x_{i+1/2}))^2 \sim \sum_i \delta_i^4 \sim n^{-4}$. By contrast if $f^*$ is not sufficiently smooth ($\nu_t \leq 2$), then $|f^*(x_{i+1/2}) - f^n(x_{i+1/2})| \sim \delta_i^{2\nu_t}$, leading to $\bar{\epsilon}(n) \sim n^{-2\nu_t}$. Note that for this asymptotic argument to apply to the feature learning regime, one must ensure that the distribution of the rescaled distance between adjacent singularities $n\delta_i$ has a finite fourth moment. This is obvious in the lazy regime, where the $\delta_i$'s are controlled by the position of the training points, but not in the feature regime, where the distribution of singular points is determined by that of the neuron's features. Nevertheless, we show that it must be the case in our setup in App. D.

**Interpretation in terms of spectral bias** From the discussion above it is evident that there is a length scale $\delta$ of order $1/n$ such that $f^n(x)$ is a good approximation of $f^*(x)$ over scales larger than $\delta$. In terms of Fourier modes, one has: *i)* $f^n(k)$ matches $f^*(k)$ at long wavelengths, i.e. for $k \ll k_c \sim 1/n$. *ii)* In addition, since the phases $\exp(ikx_j)$ become effectively random phases for $k \gg k_c$, $g^n(k) = \sum_j g_j \exp(ikx_j)$ becomes a Gaussian random variable with zero mean and fixed variance and thus *iii)* $f^n(k) = g^n(k)\hat{\varphi}(k)$ decorrelates from $f^*$ for $k \gg k_c$. Therefore

$$\bar{\epsilon}(n) \sim \sum_{|k|>k_c} \mathbb{E}_{f^*} \left[ (g^n(k)\hat{\varphi}(k) - f^*(k))^2 \right] \sim \sum_{|k|\geq k_c} \mathbb{E}_{f^*} \left[ (g^n(k))^2 \right] \hat{\varphi}(k)^2 + \mathbb{E}_{f^*} \left[ (f^*(k))^2 \right]. \tag{3.7}$$

For $\nu_t > 2$, one has $\sum_j g_j^2 \sim \frac{1}{n} \lim_{n\to\infty} \int g^n(x)^2 dx \sim \frac{1}{n}$. It follows (see App. E for details) that the sum is dominated by the first term, hence entirely controlled by the Fourier coefficients of $f^n(k)$ at large $k$. A smoother predictor corresponds to a faster decay of $f^n(k)$ with $k$, thus a faster decay of the error with $n$. Plugging the relevant decays yields $\bar{\epsilon} \sim n^{-4}$ for feature regime and lazy regime with the NTK, and $n^{-6}$ for lazy regime with the RFK (which is smoother than the NTK). For $\nu_t \leq 2$, the two terms have comparable magnitude, see App. E. It follows immediately that $\bar{\epsilon} \sim n^{-2\nu_t}$.

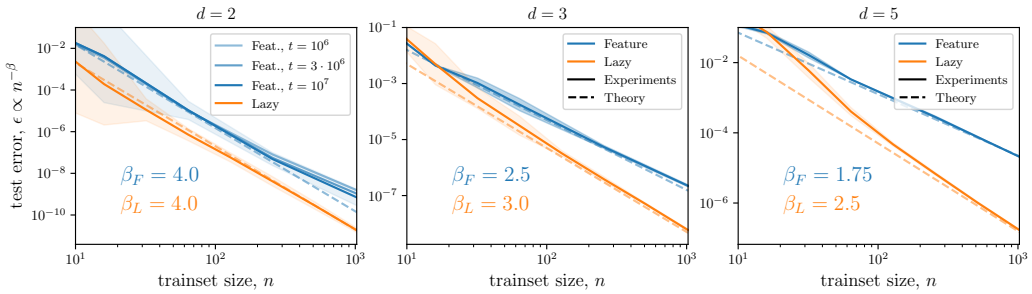



Figure 3: **Generalization error for a constant function $f^*(\boldsymbol{x}) = 1$.** Test error as a function of the training set size $n$ for a network trained in the feature regime with L1 regularization (blue) and kernel regression corresponding to the infinite-width lazy regime (orange). Numerical results (full lines) and the exponents predicted by the theory (dashed) are plotted. Panels correspond to different input-space dimensions ($d = 2, 3, 5$). Results are averaged over 10 different initializations of the networks and datasets. For $d = 2$ and large $n$, the gap between experiments and prediction for the feature regime is due to the finite training time $t$. Indeed our predictions become more accurate as $t$ increases, as illustrated in the left.

**Generalization to higher dimensions** The argument above can be generalised for any $d$ by replacing Fourier modes with projections onto spherical harmonics. The characteristic distance between training points scales as $n^{-1/(d-1)}$, thus $k_c \sim n^{-1/(d-1)}$. Our ansatz is that, as in $d=2$: *i)* for $k \ll k_c$, the predictor modes coincide with those of the target function, $f^n_{k,l} \approx f^*_{k,l}$ (this corresponds to the spectral bias result of kernel methods, stating that the predictor reproduces the first $O(n)$ projections of the target in the kernel eigenbasis [35]); *ii)* For $k \gg k_c$, $g^n_{k,l}$ is a sum of uncorrelated terms, thus a Gaussian variable with zero mean and fixed variance; *iii)* $f^n_{k,\ell} = g^n_{k,\ell}\hat{\varphi}(k)$ decorrelates from $f^*_{k,\ell}$ for $k \gg k_c$. *i)*, *ii)* and *iii)* imply that:

$$\epsilon(n) \sim \sum_{l=1}^{\mathcal{N}_{k,d}} \left(f^n_{k,l} - f^*_{k,l}\right)^2 \sim \sum_{k \geq k_c} \sum_{l=1}^{\mathcal{N}_{k,d}} \left(\langle {g^n_{k,l}}^2 \rangle \varphi_k^2 + k^{-2\nu_t-(d-1)}\right). \tag{3.8}$$

As shown in App. E, from this expression is is straightforward to obtain Eq. 3.4. Notice again that when the target is sufficiently smooth, so that the predictor-dependent term dominates, the error is determined by the smoothness of the predictor. In particular, as $d > 2$, the predictor of feature learning is less smooth than both the NTK and RFK ones, due to the slower decay of the corresponding $\varphi_k$.

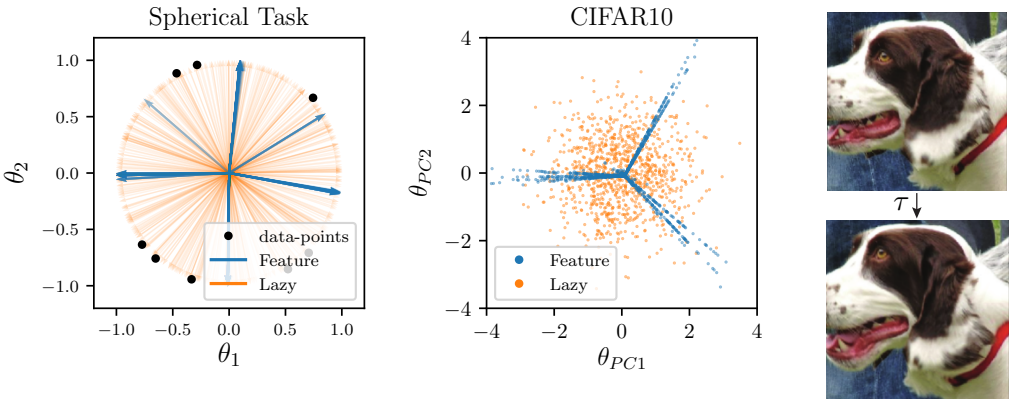
## 4 Numerical tests of the theory

We test successfully our predictions by computing the learning curves of both lazy and feature regimes when *(i)* the target function is constant on the sphere for varying $d$, see Fig. 3, and *(ii)* the target is a Gaussian random field with varying smoothness $\nu_t$, as shown in Fig. G.1 of App. G. For the lazy regime, we perform kernel regression using the analytical expression of the NTK [42] (see also Eq. A.19). For the feature regime, we find that our predictions hold when having a small regularization, although it takes unreachable times for gradient descent to exactly recover the minimal-norm solution—a more in-depth discussion can be found in App. G. An example of the atomic distribution of neurons found after training, that contrasts with the initial distribution, is displayed in Fig. 4a, left panel.

Another way to obtain sparse features is to initialize the network with very small weights [13], as proposed in [7]. As in the presence of an infinitesimal weights decay, this scheme also leads to sparse solutions with $n_A = \mathcal{O}(n)$ – an asymptotic dependence confirmed in Fig. G.3 of App. G. This observation implies that our predictions must apply in that case too, as we confirm in Fig. G.3.

## 5 Evidence for overfitting along diffeomorphisms in image datasets

For fully-connected networks, the feature regime is well-adapted to learn anisotropic tasks [15]: if the target function does not depend on a certain linear subspace of input space, e.g. the pixels at




(a) **Features sparsification.** 1st Panel: Distribution of neuron's feature for the task of learning a constant function on the sphere in 2D. Arrows represent a subset of the network features $\{\boldsymbol{\theta}_h\}_{h=1}^H$ after training in the lazy and feature regimes. Training is performed on $n = 8$ data-points (black dots). 2nd Panel: FCN trained on CIFAR10. On the axes the first two principal components of the features $\{\boldsymbol{\theta}_h\}_{h=1}^H$ after training on $n = 32$ points in the feature (blue) and lazy (orange) regimes. Similarly to what observed when learning a constant function, the $\boldsymbol{\theta}_h$ angular distribution becomes sparse with training in the feature regime.

(b) **Example of diffeomorphism.** Sample of a max-entropy deformation $\tau$ [39] when applied to a natural image, illustrating that it does not change the image class for the human brain.

Figure 4: **Features sparsificaton and example of a diffeomorphism.**

the corner of an image, then neurons align perpendicularly to these directions [18]. By contrast, our results highlight a drawback of this regime when the target function is constant or smooth along directions in input space that require a continuous distribution of neurons to be represented. In such a case, the adaptation of the weights to the training points leads to a predictor with a sparse representation. Such predictor would be less smooth than in the lazy regime and thus underperform.

Does this view hold for images, and explain why learning their features is detrimental for fully-connected networks? The first positive empirical evidence is that the neurons' distribution of networks trained on image data becomes indeed sparse in the feature regime, as illustrated in Fig. 4a, right, for CIFAR10 [27]. This observation raises the question of which are the directions in input space *i)* along which the target should vary smoothly, and *ii)* that are not easily represented by a discrete set of neurons. An example of such directions are global translations, which conserve the norm of the input and do not change the image class. Yet, this is a space of transformations of dimension 2, which is small in comparison with the full dimensionality $d$ of the data and thus may play a negligible role.

A much larger class of transformations believed to have little effect on the target are small diffeomorphisms [28]. A diffeomorphism $\tau$ acting on an image is illustrated in Fig. 4b, which highlights that our brain still perceives the content of the transformed image as in the original one. Near-invariance of the task to these transformations is believed to play a key role in the success of deep learning, and in explaining how neural networks beat the curse of dimensionality [38]. Indeed, if modern architectures can become insensitive to these transformations, then the dimensionality of the problem is considerably reduced. In fact, it was found that the architectures displaying the best performance are precisely those which learn to vary smoothly along such transformations [39].

Small diffeomorphisms are likely the directions we are looking for. To test this hypothesis, following [39], we characterise the smoothness of a function along such diffeomorphisms, relative to that of random directions in input space. Specifically we use the *relative stability*:

$$R_{f^n} = \frac{\mathbb{E}_{x,\tau}\|f^n(\tau x) - f^n(x)\|^2}{\mathbb{E}_{x,\eta}\|f^n(x+\eta) - f^n(x)\|^2}. \tag{5.1}$$

In the numerator, the average is made over the test set and over an ensemble of maximum-entropy diffeomorphisms, reviewed in App. H. The magnitude of the diffeomorphisms is chosen so that each pixel is shifted by one on average. In the denominator, the average runs over the test set and the vectors $\eta$ sampled uniformly on the sphere of radius $\|\eta\| = \mathbb{E}_{x,\tau}\|\tau x - x\|$, and this fixes the transformations magnitude.

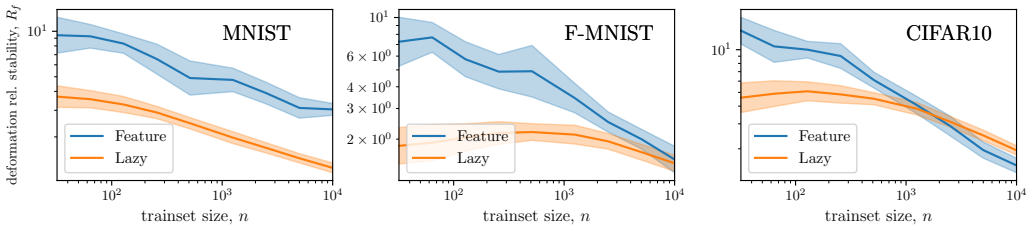



Figure 5: **Stability to diffeomorphisms vs number of training points.** Relative stability of the predictor to small diffeomorphisms of the input images, in the two regimes, for varying number of training points $n$ and different image datasets. Smaller values correspond to a smoother predictor, on average. Results are computed using the same predictors as in Fig. 1.

We measure $R_{f^n}$ as a function of $n$ for three benchmark datasets of images, as shown in Fig. 5. We indeed find that $R_{f^n}$ is consistently smaller in the lazy training where features are not learnt. Overall, this observation supports the view that learning sparse features is detrimental when data present (near) invariance to transformations that cannot be represented sparsely by the architecture considered. Fig. 1 supports the idea that—for benchmark image datasets—this negative effect overcomes well-known positive effects of learning features – e.g. becoming insensitive to pixels on the edge of images.

## 6 Conclusion

Our central result is that learning sparse features can be detrimental if the task presents invariance or smooth variations along transformations that are not adequately captured by the neural network architecture. For fully-connected networks, these transformations can be rotations of the input, but also continuous translations and diffeomorphisms.

Our analysis relies on the sparsity of the features learnt by a shallow fully-connected architecture: even in the infinite width limit, when trained in the feature learning regime such networks behave as $\mathcal{O}(n)$ neurons. The asymptotic analysis we perform for random Gaussian fields on the sphere leads to predictions for the learning curves exponents $\beta$ in different training regimes, which we verify. Such kind of results are scarce in the literature.

Note that our analysis focuses on ReLU neurons because *(i)* these are very often used in practice and *(ii)* in that case, $\beta$ will depend on the training regime, allowing for stringent numerical tests. If smooth activations (e.g. softplus) are considered, we expect that learning features will still be detrimental in the present settings. Yet, the difference will not appear in the exponent $\beta$, but in other aspects of the learning curves (including numerical coefficients and pre-asymptotic effects) that are harder to predict.

Most fundamentally, our results underline that the great success of learning features for modern architectures, e.g. on image data sets, still lacks a sufficient explanation. Indeed, most of the theoretical studies that previously emphasized the benefits of learning features have been considering fully-connected networks, for which learning features can be in practice a drawback. It is tempting to argue that in modern architectures, learning features is not at a disadvantage because smoothness along diffeomorphisms can be enforced from the start—due to the locally connected, convolutional and pooling layers [43, 28]. Yet the best architectures often do not perform pooling and are not stable toward diffeomorphisms at initialization. *During training*, learning features leads to more stable and smoother solutions along diffeomorphisms [44, 39]. Understanding why building sparse features enhances stability in these architectures may ultimately explain the magical feat of deep CNNs: learning tasks in high dimensions.

## References

[1] Ulrike von Luxburg and Olivier Bousquet. Distance-based classification with lipschitz functions. *Journal of Machine Learning Research*, 5(Jun):669–695, 2004.

[2] Francis Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[3] Quoc V Le. Building high-level features using large scale unsupervised learning. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 8595–8598. IEEE, 2013.

[4] Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

[5] Alessio Ansuini, Alessandro Laio, Jakob H Macke, and Davide Zoccolan. Intrinsic dimension of data representations in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 6111–6122, 2019.

[6] Stefano Recanatesi, Matthew Farrell, Madhu Advani, Timothy Moore, Guillaume Lajoie, and Eric Shea-Brown. Dimensionality compression and expansion in deep neural networks. *arXiv preprint arXiv:1906.00443*, 2019.

[7] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[8] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 8580–8589, USA, 2018. Curran Associates Inc.

[9] Simon S. Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[10] Grant M Rotskoff and Eric Vanden-Eijnden. Neural networks as interacting particle systems: Asymptotic convexity of the loss landscape and universal scaling of the approximation error. *arXiv preprint arXiv:1805.00915*, 2018.

[11] Song Mei, Andrea Montanari, and Phan-Minh Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, August 2018. Publisher: National Academy of Sciences Section: PNAS Plus.

[12] Justin Sirignano and Konstantinos Spiliopoulos. Mean Field Analysis of Neural Networks: A Law of Large Numbers. *SIAM Journal on Applied Mathematics*, 80(2):725–752, January 2020. Publisher: Society for Industrial and Applied Mathematics.

[13] Blake Woodworth, Suriya Gunasekar, Jason D. Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and Rich Regimes in Overparametrized Models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, July 2020. ISSN: 2640-3498.

[14] Jaume de Dios and Joan Bruna. On sparsity in overparametrised shallow relu networks. *arXiv preprint arXiv:2006.10225*, 2020.

[15] Lénaïc Chizat and Francis Bach. Implicit Bias of Gradient Descent for Wide Two-layer Neural Networks Trained with the Logistic Loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, July 2020. ISSN: 2640-3498.

[16] Behrooz Ghorbani, Song Mei, Theodor Misiakiewicz, and Andrea Montanari. When Do Neural Networks Outperform Kernel Methods? *Advances in Neural Information Processing Systems*, 33, 2020.

[17] Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. *arXiv preprint arXiv:2102.11742*, 2021.

[18] Jonas Paccolat, Leonardo Petrini, Mario Geiger, Kevin Tyloo, and Matthieu Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001, April 2021. Publisher: IOP Publishing.

[19] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, November 2020. Publisher: IOP Publishing.

[20] Jaehoon Lee, Samuel S Schoenholz, Jeffrey Pennington, Ben Adlam, Lechao Xiao, Roman Novak, and Jascha Sohl-Dickstein. Finite versus infinite neural networks: an empirical study. *arXiv preprint arXiv:2007.15801*, 2020.

[21] Guillermo Ortiz-Jiménez, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. What can linearized neural networks actually say about generalization? *Advances in Neural Information Processing Systems*, 34, 2021.

[22] Yilan Chen, Wei Huang, Lam M. Nguyen, and Tsui-Wei Weng. On the Equivalence between Neural Network and Support Vector Machine. *arXiv:2111.06063 [cs, math, stat]*, November 2021. arXiv: 2111.06063.

[23] Mario Geiger, Arthur Jacot, Stefano Spigler, Franck Gabriel, Levent Sagun, Stéphane d'Ascoli, Giulio Biroli, Clément Hongler, and Matthieu Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, February 2020. Publisher: IOP Publishing.

[24] Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18, August 2021.

[25] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[26] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv:1708.07747 [cs, stat]*, September 2017. arXiv: 1708.07747.

[27] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

[28] Joan Bruna and Stéphane Mallat. Invariant scattering convolution networks. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1872–1886, 2013.

[29] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[30] Hartmut Maennel, Olivier Bousquet, and Sylvain Gelly. Gradient descent quantizes relu network features. *arXiv preprint arXiv:1803.08367*, 2018.

[31] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401. PMLR, 2015.

[32] Claire Boyer, Antonin Chambolle, Yohann De Castro, Vincent Duval, Frédéric De Gournay, and Pierre Weiss. On Representer Theorems and Convex Regularization. *SIAM Journal on Optimization*, 29(2):1260–1281, May 2019.

[33] Lenaic Chizat. Sparse optimization on measures with over-parameterized gradient descent. *Mathematical Programming*, pages 1–46, 2021.

[34] Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001, December 2020. Publisher: IOP Publishing.

[35] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum Dependent Learning Curves in Kernel Regression and Wide Neural Networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, November 2020. ISSN: 2640-3498.

[36] Hugo Cui, Bruno Loureiro, Florent Krzakala, and Lenka Zdeborová. Generalization error rates in kernel regression: The crossover from the noiseless to noisy regime. *Advances in Neural Information Processing Systems*, 34, 2021.

[37] Umberto M Tomasini, Antonio Sclocchi, and Matthieu Wyart. Failure and success of the spectral bias prediction for kernel ridge regression: the case of low-dimensional data. *arXiv preprint arXiv:2202.03348*, 2022.

[38] Stéphane Mallat. Understanding deep convolutional networks. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150203, 2016.

[39] Leonardo Petrini, Alessandro Favero, Mario Geiger, and Matthieu Wyart. Relative stability toward diffeomorphisms indicates performance in deep nets. In *Advances in Neural Information Processing Systems*, volume 34, pages 8727–8739. Curran Associates, Inc., 2021.

[40] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, 1998.

[41] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[42] Youngmin Cho and Lawrence K. Saul. Kernel methods for deep learning. In *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[43] Alberto Bietti and Julien Mairal. Group invariance, stability to deformations, and complexity of deep convolutional representations. *The Journal of Machine Learning Research*, 20(1):876–924, 2019.

[44] Avraham Ruderman, Neil C. Rabinowitz, Ari S. Morcos, and Daniel Zoran. Pooling is neither necessary nor sufficient for appropriate deformation stability in CNNs. *arXiv:1804.04438 [cs, stat]*, May 2018. arXiv: 1804.04438.

[45] Alex Smola, Zoltán Ovári, and Robert C. Williamson. Regularization with dot-product kernels. *Advances in neural information processing systems*, 13, 2000.

[46] Kendall Atkinson and Weimin Han. *Spherical harmonics and approximations on the unit sphere: an introduction*, volume 2044. Springer Science & Business Media, 2012.

[47] Costas Efthimiou and Christopher Frye. *Spherical harmonics in p dimensions*. World Scientific, 2014.

[48] Alberto Bietti and Francis Bach. Deep Equals Shallow for ReLU Networks in Kernel Regimes. *arXiv:2009.14397 [cs, stat]*, August 2021. arXiv: 2009.14397.

[49] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Journal on Scientific Computing*, 20(1):33–61, January 1998. Publisher: Society for Industrial and Applied Mathematics.

## Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] Yes, in the conclusion we state that this paper provides a better understanding of the feature regime for FCNs while a better understanding of its advantages in other architectures like CNNs is still lacking.
   (c) Did you discuss any potential negative societal impacts of your work? [No] We don't foresee possible negative societal consequences of our work.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] Yes, proofs are either provided in the main text or in appendix, depending on space constraints.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] We provide links to the (anonymized) github repository, code and instructions on how to run it are found there.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Training details are either in the text or in the experiments.md file, as indicated in App. G.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Statistical errors are reported as shaded area around the curves.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] Details are provided in App. G.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use the PyTorch framework and cite the authors in the text.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] The code written by the authors for running experiments is available as a URL.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]