

# Disentangling feature and lazy learning in deep neural networks: an empirical study

**Mario Geiger**

Institute of Physics

École Polytechnique Fédérale de Lausanne  
CH-1015 Lausanne, Switzerland

mario.geiger@epfl.ch

**Stefano Spigler**

Institute of Physics

École Polytechnique Fédérale de Lausanne  
CH-1015 Lausanne, Switzerland

stefano.spigler@epfl.ch

**Arthur Jacot**

Institute of Mathematics

École Polytechnique Fédérale de Lausanne  
CH-1015 Lausanne, Switzerland

arthur.jacot@epfl.ch

**Matthieu Wyart**

Institute of Physics

École Polytechnique Fédérale de Lausanne  
CH-1015 Lausanne, Switzerland

matthieu.wyart@epfl.ch

## Abstract

Two distinct limits for deep learning as the net width  $h \rightarrow \infty$  have been proposed, depending on how the weights of the last layer scale with  $h$ . In the “lazy-learning” regime, the dynamics becomes linear in the weights and is described by a Neural Tangent Kernel  $\Theta$  (NTK). By contrast, in the “feature-learning” regime, the dynamics can be expressed in terms of the density distribution of the weights. Understanding which regime describes accurately practical architectures and which one leads to better performance remains a challenge. We answer these questions and produce new characterizations of these regimes for the MNIST data set, by considering deep nets  $f$  whose last layer of weights scales as  $\frac{\alpha}{\sqrt{h}}$  at initialization, where  $\alpha$  is a parameter we vary. We performed systematic experiments on two setups (A) fully-connected *Softplus* momentum full batch and (B) convolutional *ReLU* momentum stochastic. We find that (i)  $\alpha^* = \frac{1}{\sqrt{h}}$  separates the two regimes. (ii) for (A) and (B) feature learning outperforms lazy learning, a difference in performance that decreases with  $h$  and becomes hardly detectable asymptotically for (A) but is very significant for (B). (iii) In both regimes, the fluctuations  $\delta f$  induced by initial conditions on the learned function follow  $\delta f \sim 1/\sqrt{h}$ , leading to a performance that increases with  $h$ . This improvement can be instead obtained at intermediate  $h$  values by ensemble averaging different networks. (iv) In the feature regime there exists a time scale  $t_1 \sim \alpha\sqrt{h}$ , such that for  $t \ll t_1$  the dynamics is linear. At  $t \sim t_1$ , the output has grown by a magnitude  $\sqrt{h}$  and the changes of the tangent kernel  $\|\Delta\Theta\|$  become significant. Ultimately, it follows  $\|\Delta\Theta\| \sim (\sqrt{h}\alpha)^{-a}$  for *ReLU* and *Softplus* activation functions, with  $a < 2$  and  $a \rightarrow 2$  when depth grows.

## 1 Introduction and related works

Deep neural networks are successful at a variety of task, yet understanding why they work remains a challenge. A surprising observation is that their performance on supervised tasks keeps increasing with their width  $h$  in the over-parametrized regime where they already fit all the training data [Neyshabur et al., 2017, Bansal et al., 2018, Advani and Saxe, 2017, Spigler et al., 2018]. This

fact underlines the importance of describing deep learning in the limit  $h \rightarrow \infty$ , which has received considerable interest recently. When all the weights  $w$  of the network are initialized with a value of order  $h^{-1/2}$  at initialization, the transmission of the signal through the network in the infinite-width limit is now well understood. The output function  $f(w, x)$  is a Gaussian random processes with some covariance that can be computed [Neal, 1996, Williams, 1997, Lee et al., 2018, de G. Matthews et al., 2018, Novak et al., 2019, Yang, 2019].

**Lazy learning** More recently, it has been shown that beyond initialization, the learning dynamics in this limit also simplifies [Jacot et al., 2018, Du et al., 2019, Allen-Zhu et al., 2018, Lee et al., 2019, Arora et al., 2019, Park et al., 2019] and is entirely described by the neural tangent kernel (NTK) [Jacot et al., 2018] defined as:

$$\Theta(w, x_1, x_2) = \nabla_w f(w, x_1) \cdot \nabla_w f(w, x_2), \quad (1)$$

where  $x_1, x_2$  are two inputs and  $\nabla_w$  is the gradient with respect to the parameters  $w$ . For  $h \rightarrow \infty$ ,  $\Theta(w, x_1, x_2) \rightarrow \Theta_\infty(x_1, x_2)$  does not vary at initialization (and thus it does not depend on the specific choice of  $w$ ) and does not evolve in time. The dynamics is guaranteed to converge on a time independent of  $h$  to a global minimum of the loss, and as for usual kernel learning the function only evolves in the space spanned by the functions  $\Theta_\infty(x_\mu, x)$ , where  $\{x_\mu, \mu = 1 \dots n\}$  is the training set.

**Feature learning** Another limit, feature learning also called mean field in the literature, has been studied in several works focusing mostly on one-hidden layer networks [Mei et al., 2018, Rotskoff and Vanden-Eijnden, 2018, Chizat and Bach, 2018, Sirignano and Spiliopoulos, 2018, Mei et al., 2019, Nguyen, 2019]. In this setting the output function of the network with  $h$  hidden neurons corresponding to an input  $x$  is:

$$f(w, x) = \frac{1}{h} \sum_{i=1}^h c_i \sigma(a_i \cdot x + b_i), \quad (2)$$

where  $\sigma(\cdot)$  is the non-linear activation function and  $w_i = (a_i, b_i, c_i)$  are the parameters associated with a hidden neuron. Note that the last layer of weights scales as  $h^{-1}$ , which differs from the  $h^{-1/2}$  scaling used in the NTK setting. The network can be described in terms of the empirical distribution of a neuron's parameters

$$\rho_h(w_i) = \frac{1}{h} \sum_{j=1}^h \delta(w_i - w_j). \quad (3)$$

As  $h \rightarrow \infty$ , this distribution tends to a probability distribution  $\rho$ , Eq. (2) can be written as an integral, and the training dynamics is then controlled by a differential equation for  $\rho$ .

The existence of two distinct limits raises both fundamental and practical questions:

First, which limit best characterizes the neural networks that are used in practice, and which one leads to a better performance? These questions are still debated. In [Chizat and Bach, 2019], based on teacher-student numerical experiments, it was argued that the lazy limit is unlikely to explain the success of neural networks. However, it was recently shown that the lazy limit is able to achieve good performance on real datasets [Arora et al., 2019] (significantly better than alternative kernel methods). Moreover, some predictions of the lazy limit agree with observations in networks of practically-used size [Lee et al., 2019].

Second, it was argued that in the lazy limit the surprising improvement of performance with  $h$  stems from the  $h^{-1/2}$  fluctuations of the kernel at initialization, that ultimately leads to similar fluctuations in the learned function and degrade performance of networks of small width [Geiger et al., 2019]. These fluctuations can be removed by ensemble-averaging output functions obtained with different initial conditions, leading to excellent performance already near the underparametrized to overparametrized (or jamming transition)  $h^*$  beyond which all the training data are correctly fitted [Geiger et al., 2018]. Does this line of thought hold true in the feature-learning limit?

Finally, unlike in the lazy limit where preactivations vary very weakly during training, in the feature limit feature learning occurs. It is equivalent to saying that the tangent kernel (which can always be defined) evolves in time [Rotskoff and Vanden-Eijnden, 2018, Mei et al., 2019, Chizat and Bach, 2019]. What are the characteristic time scales and magnitude of this evolution?

In this work we answer these questions empirically by learning a model of the form:

$$\alpha [f(w, x) - f(w_0, x)], \quad (4)$$

where  $f(w, x)$  is a deep net, considering both fully-connected (FC) and convolutional (CNN) architectures as well as *Softplus* or *ReLU* activation functions. This model is inspired by [Chizat and Bach, 2019]. For  $\alpha = \mathcal{O}(1)$  it falls into the framework of the NTK literature, whereas for  $\alpha = \mathcal{O}(h^{-1/2})$  it corresponds to the feature-learning framework. By focusing on MNIST, we are able to study systematically the role of both  $\alpha$  (varied on 8 orders of magnitude) and the width  $h$ , and for each setting learn ensembles of  $\approx 20$  networks so as to quantify precisely the magnitude of fluctuations induced by initialization, and the benefit of ensemble averaging. Our main result is that in the  $(\alpha, h)$  plane two distinct regimes can be identified, in which both performance and dynamics qualitatively differ, although some properties (such as the decay of the fluctuations  $\delta f$  with  $h$ ) are similar. Our work has both practical applications (in terms of which parameters and architectures lead to improved performance) and conceptual ones (in quantifying the dynamics of feature learning and providing informal explanations for these observations). More generally, it suggests that future empirical studies of deep learning would benefit from characterizing the regime in which they operate, since it will most likely impact their results.

The code used for this article is available online at [https://github.com/mariogeiger/feature\\_lazy](https://github.com/mariogeiger/feature_lazy).

## 2 Notations and set-up

We consider deep networks of  $L$  hidden layers in a binary classification task. The output is  $f(w, x)$ , where  $w$  is the set of parameters and  $x$  is an input pattern. The set of training data is  $\mathcal{T} = \{(x_\mu, y_\mu), \mu = 1, \dots, n\}$ , where  $y_\mu = \pm 1$  are the labels associated with the pattern  $x_\mu$  and  $n$  is the number of training data.  $\dot{a}$  is the time derivative of a variable  $a$  during training.

**Parameter  $\alpha$ :** We train  $\alpha(f(w, x) - f(w_0, x))$ , where  $w_0$  is the network's parameters at initialization. This functional form ensures that for  $\alpha \rightarrow \infty$ , we enter a lazy regime where changes of weights are small. Indeed in order to obtain a zero loss  $\alpha(f(w, x) - f(w_0, x))$  must be  $\mathcal{O}(1)$ , thus  $1 \sim \alpha(f(w) - f(w_0)) \sim \alpha \nabla_w f(w_0, x) \cdot dw$ , implying that  $\|dw\| \equiv \|w - w_0\| \sim \alpha^{-1}$  and that the dynamics become linear [Chizat and Bach, 2019].

We use as loss function:

$$\mathcal{L}(w) = \frac{1}{\alpha^2 n} \sum_{(x, y) \in \mathcal{T}} \ell(\alpha(f(w, x) - f(w_0, x)), y). \quad (5)$$

The prefactor  $\alpha^{-2}$  ensures that the convergence time does not depend on  $\alpha$  as  $\alpha \rightarrow \infty$ , since for that choice we have  $\alpha \dot{f}(w_0) = \mathcal{O}(1)$ . For  $\ell(f, y)$  we use the hinge loss  $\ell(f, y) = \max(0, 1 - fy)$ . It results in essentially identical performance to the commonly used cross-entropy in state-of-the-art architectures [Spigler et al., 2018], but leads to a dynamics that stops in finite time in the over-parametrized regime considered here, removing the need to introduce an arbitrary temporal cut-off.

**Dynamics** Since our goal is to build a connection between empirical and theoretical approaches, we focus on a discrete version of continuous dynamics obeying simple differential equation. The simplest is the *Vanilla gradient descent* which reads  $\dot{w} = -\nabla_w \mathcal{L}$  and does not depend on any hyper-parameters. Yet this approach is often too slow. So as to accelerate the dynamics we use *momentum*:

$$\begin{cases} \dot{w} = v \\ \dot{v} = -\frac{1}{\tau}(v + \nabla_w \mathcal{L}) \end{cases} \quad (6)$$

where  $v$  takes the role of a velocity and  $\tau$  takes the role of a relaxation time. In the  $\tau \rightarrow 0$  limit, this dynamics reduces to the vanilla gradient descent. In our experiments we either set  $\tau$  to a constant value or let it vary as a function of time  $\tau = t/\mu$ . We tested different values of  $\tau$  and chose the one that led to faster convergence in computation time. It is also possible to build stochastic variants of this dynamics (with a batch size smaller than the whole training set). We use such a dynamics for CNN, details are explained in Appendix A.

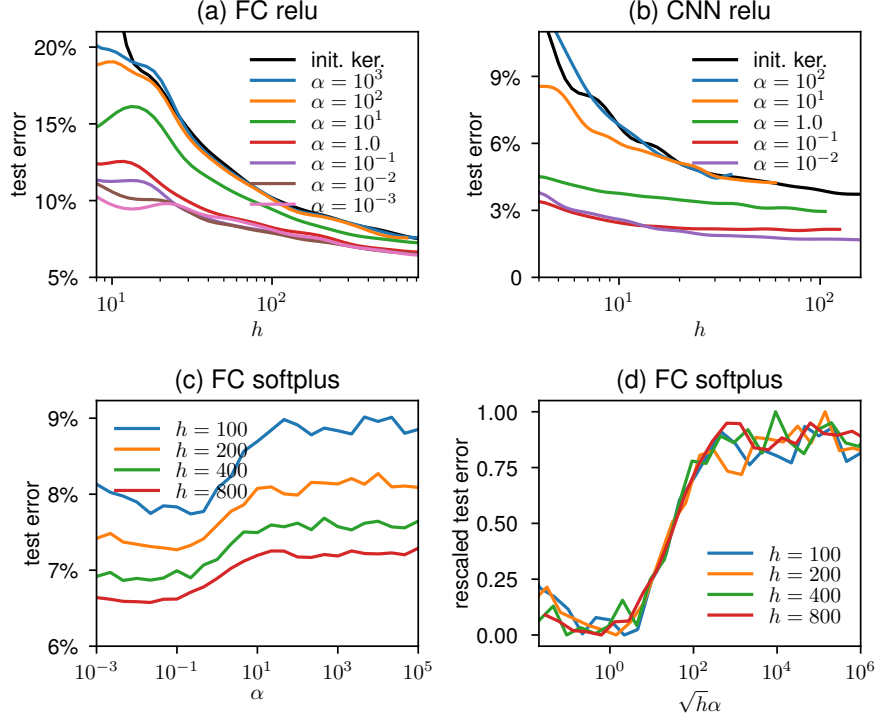


Figure 1: Running averaged test error of fully-connected (a) and convolutional (b) networks v.s. the network’s width  $h$  for different values of  $\alpha$  as indicated in legend. The black solid line is the test error of the frozen NTK at initialization, limit that is recovered as  $\alpha \rightarrow \infty$ . (c) Test error of fully-connected networks v.s.  $\alpha$ , for different widths  $h$ . (averaged over 10 initializations) (d) Same data as in (c): after rescaling the test error to be between 0 and 1, the curves collapse when plotted against  $\sqrt{h}\alpha$ .

**Activation function** We consider both non-smooth (*ReLU*) and smooth (*Softplus*) activation functions, the latter being defined as  $\sigma(x) = (a/\beta) \ln(1 + e^{\beta x})$ . We chose  $\beta = 5$ , and  $a \approx 1.404$  to have preactivation of variance unity at initialization. The non-smoothness of *ReLU* can introduce extra phenomena, for instance its impact on the evolution of the kernel is shown in Appendix H.

**Architectures** We consider two architectures:

- FC architecture ( $L = 5$ ) with *Softplus* or *ReLU* and full batch momentum gradient descent with  $\mu = 1000$ .
- CNN architecture ( $L = 4$ ) with *ReLU* and stochastic momentum gradient descent with  $\tau = 10$ .

We follow the initialization of [Jacot et al., 2018] and extend it to CNN. The FC architecture is described in Appendix B.

**Dataset** We train our network to classify the parity of handwritten digit numbers from the MNIST database [LeCun and Cortes, 2010]. For the FC architecture only, to avoid that most weights lie between the input and the first layer of hidden neurons, we reduce the dimension of the data using principal component analysis. Specifically, we subtract the averaged image and then compute the 10 first principal components. For  $(x, y) \in \mathcal{T}$  we have  $x \in \mathbb{R}^{10}$  and  $y = \pm 1$  with each component  $x_i$  having variance 1.

### 3 Disentangling feature learning versus lazy learning from performance

To show the existence of two distinct limiting behaviors in deep neural networks we evaluate their performance in the  $(\alpha, h)$  plane. Fig. 1 (a-b) displays the test error of both fully-connected and convolutional networks respectively as a function of their width  $h$ , for different values of  $\alpha$ . We observe that as  $\alpha$  is increased the performance degrades, up to a point where it converges to a limiting curve (for  $\alpha \gtrsim 10^2$  in the figures). This limiting curve coincides with the test error found if the NTK is frozen at initialization (see Appendix C for a description of our dynamics in that case), represented by a black solid line. We refer to this limiting behavior as lazy learning.

Convergence to a distinct limit is also found when  $\alpha$  is decreased. To identify this regime, we show in Fig. 1 (c) the test error as a function of  $\alpha$  for several widths  $h$ , and the rescaled test error (in such a way that it takes values between 0 and 1) as a function of  $\alpha/\sqrt{h}$  in Fig. 1 (d). The curves collapse, supporting that in the  $(\alpha, h)$  plane the boundary between the two regimes lies at a scale  $\alpha^* = \mathcal{O}(h^{-1/2})$ . It is precisely the scaling used in feature learning, and we give that name to the asymptotic behavior observed for  $\alpha \ll \alpha^*$ .

Obviously, Fig. 1 makes another key point: feature learning outperforms lazy learning. Interestingly, the difference of performance diminishes with  $h$  and becomes quite small for the FC architecture. However, Fig. 1 (b) indicates that this difference remains very significant for the CNN for the largest  $h$  we probe.

### 4 Fluctuations of output function and effect of ensemble averaging

As recalled in Fig. 1, performance increases with  $h$ : adding more fitting parameters leads to better predictability. This surprising behavior was related to fluctuations  $\delta f$  of the output function induced by initial conditions, observed to decrease with  $h$  [Neal et al., 2019, Geiger et al., 2019]. To quantify them in both regimes, we train an ensemble of 20 identical functions  $f(w_i, x)$  with different initial conditions, and measure the ensemble average  $\bar{f}(x)$ . Then, fluctuations  $\delta f$  are captured by the variance around the mean:

$$\text{Var} f(w, x) = \left\langle [f(w_i, x_\mu) - \bar{f}(x_\mu)]^2 \right\rangle_{\substack{\mu \in \text{test} \\ i \in \text{ensemble}}}, \quad (7)$$

where the average is both over the 20 output functions and over all points  $x_\mu$  in a test set. In Fig. 2 (a) we compute these fluctuations either in the feature learning ( $\sqrt{h}\alpha = 10^{-2}$ ) or in the lazy learning ( $\sqrt{h}\alpha = 10^6$ ) regimes. In both cases we find:

$$\text{Var} \alpha (f(w, x) - \bar{f}(w_0, x)) \sim h^{-1}. \quad (8)$$

It was argued in [Geiger et al., 2019] that in the lazy learning regime this scaling simply stems from the fluctuations of the NTK at initialization, that go as  $\|\delta\Theta\| \sim 1/\sqrt{h}$  and lead to similar fluctuations in  $\delta f$ . These fluctuations were argued to lead to an asymptotic decrease of test error as  $1/h$ , consistent with observations. Interestingly, the same scaling for the fluctuations holds in the feature-learning regime, perhaps reflecting the approximations expected from the Central Limit Theorem (CLT) when Eq. (2) is replaced by an integral.

As a consequence of these fluctuations, ensemble averaging output functions lead to enhance performance in both regimes, as shown in Fig. 2 (b-c-d) both for FC and CNN. It is important to note that:

- (i) In each regime, the test error of the ensemble average is essentially independent of  $h$ . It implies that the variation of performance with  $h$  is only a matter of diminishing fluctuations in the over-parametrized case considered here (as shown below, we always fit all training data in these runs). It also supports that the plateau value of the ensemble-average performance we observe corresponds to the performance of single network in the  $h \rightarrow \infty$  limit.
- (ii) Once the ensemble average is made, feature and lazy performance are nearly identical for the FC case as shown in Fig. 2 (c), but very significant for the CNN as shown in Fig. 2 (d).
- (iii) In practical terms, our observations suggest that the best procedure consists in ensemble averaging networks in the feature-learning regime, with  $h$  that can be small yet sufficiently large to fit the data

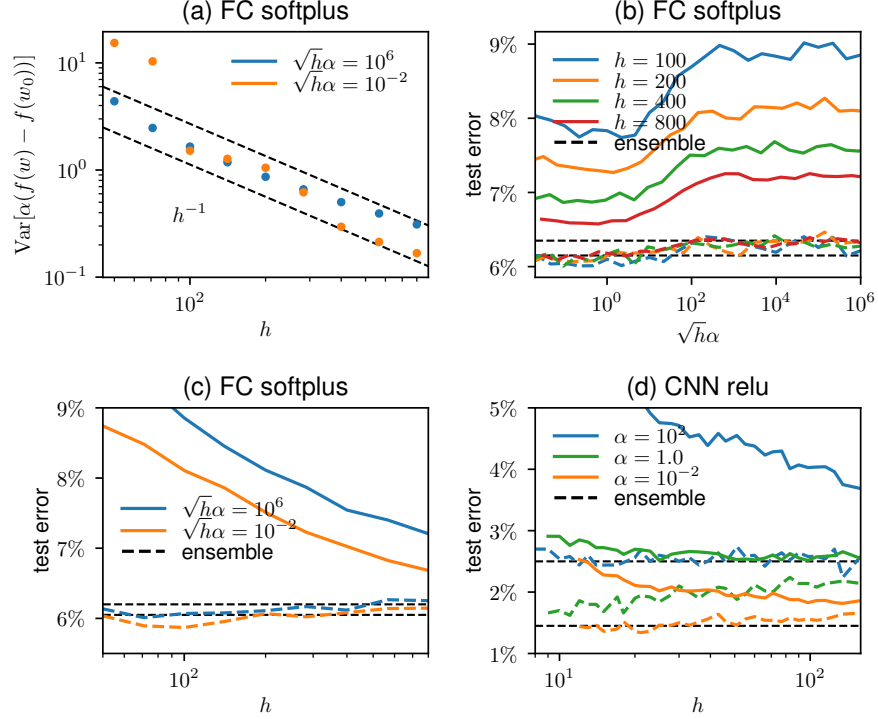


Figure 2: (a) Variance of the network’s output v.s. its width  $h$ , for the two regimes ( $\sqrt{h}\alpha$  is small in the feature-learning regime and large in the lazy one). In both regimes the variance scales as  $\text{Var} f \sim (\sqrt{h}\alpha)^{-2}$  (26 initializations per point). (b) Test error and ensemble-averaged test error v.s.  $\sqrt{h}\alpha$ , for fully-connected networks of several widths  $h$  (10 initializations per point). (c) Running averaged test error and ensemble-averaged test error v.s.  $h$  at fixed values of  $\sqrt{h}\alpha$  (26 initializations per point). Once ensemble averaged, feature learning performs slightly better than lazy learning:  $\epsilon_{\text{lazy}}/\epsilon_{\text{feature}} \approx 1.02$  where  $\epsilon_{\text{lazy}}$  and  $\epsilon_{\text{feature}}$  are the ensemble averaged test error in the two regimes. (d) Running averaged test error and the ensemble-averaged test error v.s.  $h$  at fixed values of  $\alpha$  (20 initializations per point). Once ensemble averaged, feature learning perform much better than lazy learning:  $\epsilon_{\text{lazy}}/\epsilon_{\text{feature}} \approx 1.7$ .

(i.e. past the jamming transition  $h^*$ , below which performance is known to degrade [Spigler et al., 2018]). Interestingly, it is reported in [Geiger et al., 2019] that for a fixed  $\alpha$ , the smallest test error of the ensemble average is obtained at some finite  $h_{\min}$  beyond  $h^*$  implying that past  $h_{\min}$  performance is decreasing with growing  $h$ . It is consistent with the behavior of the orange dotted line in Fig. 2 (d), and can now be simply explained: at fixed  $\alpha$  one crosses-over from feature to lazy learning regime as  $h$  increases, since  $\sqrt{h}\alpha$  also increases and must eventually become much larger than one, leading to degraded performance.

## 5 Training dynamics differs in the two regimes

The network can learn features in the feature-learning regime, while it cannot in the lazy regime because the NTK is frozen. To quantify feature learning we measure the total variation of the kernel at the end of training:  $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\|$ , where the norm of a kernel is defined as  $\|\Theta(w)\|^2 = \sum_{\mu, \nu \in \text{test set}} \Theta(w, x_\mu, x_\nu)^2$ . This quantity is plotted in Fig. 3 (a, left) versus  $\alpha$  for several widths  $h$ , and versus  $\sqrt{h}\alpha$  in Fig. 3 (a, right) where the curves collapse, indicating that  $\alpha\sqrt{h}$  is the parameter controlling feature learning. In particular, the cross-over  $\alpha^* \sim h^{-1/2}$  precisely corresponds to the point where the change of the kernel is of the order of the initial kernel. Moreover,

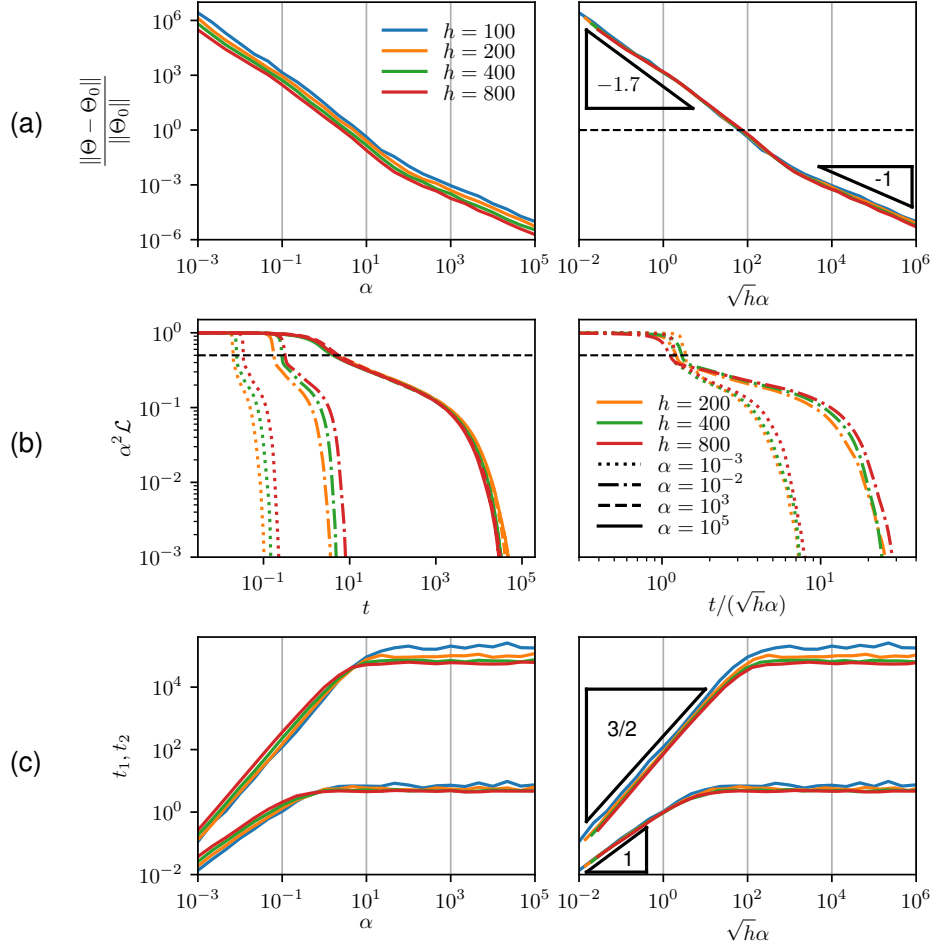


Figure 3: (a) Relative evolution of the kernel  $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\|$  v.s.  $\alpha$  (left) and  $\sqrt{h}\alpha$  (right). (b) Rescaled loss  $\alpha^2 \mathcal{L}$  v.s.  $t$  (left) and  $t/\sqrt{h}\alpha$  (right). Two characteristic times are defined as  $\alpha^2 \mathcal{L}(t_1) = 0.5$  and  $t_2$  is the smallest time for which  $\mathcal{L} = 0$  and the dynamic stops. (d)  $t_1, t_2$  v.s.  $\alpha$  (left) and  $\sqrt{h}\alpha$  (right). For (a) and (b) each point is averaged over 10 initializations.

in the feature-learning regime we find:

$$\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\| \sim (\sqrt{h}\alpha)^{-a}, \quad (9)$$

with an exponent  $a \approx 1.7$ . By contrast in the lazy regime  $\|\Theta(w) - \Theta(w_0)\|/\|\Theta(w_0)\| \sim 1/(\sqrt{h}\alpha)$  as expected [Authors, 2019, Geiger et al., 2019, Lee et al., 2019].

In Appendix C, we show in Fig. 5 an example of evolution of the Kernel (represented by its Gramm matrix). We also present an interesting finding: once the network has learned, performing kernel learning with the NTK obtained at the end of training essentially leads to identical performance.

We finally investigate the temporal evolution of learning, known to be characterized by several time scales [Baity-Jesi et al., 2018]. The rescaled loss  $\alpha^2 \mathcal{L}$  is shown in Fig. 3 (b). As expected [Jacot et al., 2018, Chizat and Bach, 2019], in the lazy regime  $\mathcal{L}(t)$  depends neither on  $\alpha$  nor  $h$ . This is not true in feature learning however. We define  $t_1$  as the time point where the loss decreases by 50% and  $t_2$  as the point where the loss reaches 0. Our key finding, shown in Fig. 3 (c), is that in the feature-learning regime:

$$t_1 \sim \alpha\sqrt{h}. \quad (10)$$

In Appendix E we show that  $t_1$  also marks the timescale below which the dynamics remains linear.



## 6 Informal arguments on kernel dynamics and regimes boundary

The arguments proposed in this section do not have the status of mathematical proofs. They provide heuristic explanations for the scaling  $t_1 \sim \alpha\sqrt{h}$  and  $\alpha^* \sim 1/\sqrt{h}$ , and support that the output function grows by a factor  $\sqrt{h}$  before the dynamics become highly non-linear. A simple assumption on the nature of the ensuing non-linear dynamics leads to the prediction Eq. (9) with  $a = 2/(1 + 1/L)$ , which we view as a good approximation (not necessarily exact) of our observations.

Our starting point is that for the NTK initialization, we have at  $t = 0$  that  $\tilde{z} = \mathcal{O}(1)$  and  $\partial f / \partial \tilde{z} = \mathcal{O}(1/\sqrt{h})$ , where  $\tilde{z}$  is the preactivation of a hidden neuron. The second point can be derived iteratively starting from the last hidden layer. Denoting by  $B$  a bias on a hidden neuron,  $W^L$  a weight connected to the output, and  $W^\ell$  a weight connecting two hidden neurons, it is then straightforward to show using the chain rule and  $\sigma'(\tilde{z}) = \mathcal{O}(1)$  that (see also [Arora et al., 2019]):

$$\frac{\partial f}{\partial B} = \mathcal{O}\left(\frac{1}{\sqrt{h}}\right); \quad \frac{\partial f}{\partial W^L} = \mathcal{O}\left(\frac{1}{\sqrt{h}}\right); \quad \frac{\partial f}{\partial W^\ell} = \mathcal{O}\left(\frac{1}{h}\right). \quad (11)$$

From which we deduce that:

$$\dot{B} = \mathcal{O}\left(\frac{1}{\alpha\sqrt{h}}\right); \quad \dot{W}^L = \mathcal{O}\left(\frac{1}{\alpha\sqrt{h}}\right); \quad \dot{W}^\ell = \mathcal{O}\left(\frac{1}{\alpha h}\right); \quad \dot{\tilde{z}} = \mathcal{O}\left(\frac{1}{\alpha\sqrt{h}}\right) \quad (12)$$

To obtain the last relation we note that the variation of preactivation stems both from the variation of the bias on that neuron, and from the variations of the activities of previous neurons. We have assumed that the second effect is not much larger than the first, and checked this assumption in Appendix F. From Eq. (12) we expect that:

$$\forall t \ll t_1 \equiv \alpha\sqrt{h}, \quad W^L(t) - W^L(0) = o(1); \quad \tilde{z}(t) - \tilde{z}(0) = o(1) \quad (13)$$

Thus for  $t \ll t_1$ , we are in the lazy regime where preactivation and weights did not have time to evolve, and we expect the kernel variations to be small (see [Mei et al., 2019] for a related discussion). Since the lazy regime finds a zero loss solution and stops in a time  $\mathcal{O}(1)$  (see Section 2), if  $t_1 = \alpha\sqrt{h} \gg 1$  the network remains in it throughout learning. Thus  $\alpha^* \sim 1/\sqrt{h}$ , as proposed for a single layer in [Chizat and Bach, 2019].

By contrast, if  $\alpha\sqrt{h} \ll 1$  the dynamics has not stopped at times  $t \sim t_1$ , for which we have  $W^L(t_1) - W^L(0) = \mathcal{O}(1)$  and  $\tilde{z}(t_1) - \tilde{z}(0) = \mathcal{O}(1)$ : both the preactivations and the weights of the last layer have changed significantly, leading to significant changes of  $\nabla_w f$  and  $\Theta$ . It is important to note that at  $t \sim t_1$ , the scale of the output function is expected to change. Indeed at initialization the output function, which is a sum made on the last layer of hidden neurons  $f(w, x) = \frac{1}{\sqrt{h}} \sum_{i=1}^h W_i^L \sigma(\tilde{z}_i^L)$ , is  $\mathcal{O}(1)$  as expected from the CLT applied to  $h$  uncorrelated terms [Neal, 1996]. However, for  $t \sim t_1$  this independence does not hold anymore, since the terms  $W_i^L \sigma(\tilde{z}_i^L)$  have evolved by  $\mathcal{O}(1)$  to change the function  $f(w, x)$  in a specific direction. We thus expect these correlations to build up linearly in time for  $t \in [0, t_1]$  and to ultimately increase the output by a factor  $\sqrt{h}$  at  $t \sim t_1$ , as confirmed in Appendix E. Note that this effect does not appear at intermediate layers in the network, because the weights evolve much more slowly there as follows from Eq. (12).

Still, such an increase in the output is insufficient to find solutions for feature learning, since  $\alpha[f(w(t_1)) - f(w(0))] \sim \alpha\sqrt{h} \ll 1$ . We propose that for activation functions that increase linearly at large arguments as those we use, the dynamics for  $t \gg t_1$  approximately corresponds to an inflation of the weights along the direction  $\dot{w}(t_1)$ . Specifically, we define an amplification factor  $\lambda(t) = \|w(t) - w(0)\| / \|w(t_1) - w(0)\|$ , and assume for simplicity that this amplification is identical in each of the  $L + 1$  layers of weights (we disregard in particular the fact that the last and first layer may behave differently, as discussed in [Arora et al., 2019]). By definition,  $\lambda(t_1) = 1$ . At the end  $t_2$  of training, for activation functions that increase linearly at large arguments, we expect to have  $\lambda(t_2) \sim [\alpha\sqrt{h}]^{-1/(L+1)}$  to ensure that  $\alpha[f(w(t_2)) - f(w(0))] = \mathcal{O}(1)$ . Gradient with respect to weights are increased by  $\lambda^L$ , leading to an overall inflation of the kernel:

$$\Theta(t_2) - \Theta(0) \sim \Theta(t_2) \sim \lambda^{2L} \sim [\alpha\sqrt{h}]^{-\frac{2}{1+1/L}} \quad (14)$$

leading to  $a = 1.66$  consistent with Eq. (9). We have checked this prediction with success for shallow networks with  $L = 2$ , as shown in Appendix G.



## 7 Conclusion

We have shown that as the width  $h$  and the output scale  $\alpha$  at initialization are varied, two regimes appear depending on the value of  $\alpha\sqrt{h}$ . In the feature-learning regime features are learned, in the lazy regime the dynamics is controlled by a kernel. Our key findings are that: (i) in our two experiments feature-learning outperformed lazy-learning. (ii) In both regimes, fluctuations induced by initialization decrease with  $h$ , explaining why performance increases with width. This observation supports that the best strategy is to ensemble average networks in the feature-learning regime at intermediate  $h$ . (iii) In feature learning, the learning dynamics is linear for  $t \ll t_1 \sim \sqrt{h}\alpha$  where the output of the model becomes of order  $\sqrt{h}\alpha$ . If  $\sqrt{h}\alpha \ll 1$ , in order to fit the data the dynamics enters a non-linear regime for  $t \sim t_1$  that affects the magnitude of the kernel.

On the empirical side, our work supports that studies of deep learning (e.g. on the role of regularization) should specify in which regime their networks operate, since it is very likely to affect their results. On the theoretical side, there is little quantitative understanding of how much performance should differ in two regimes. For example, there is no guarantee that feature learning always outperforms lazy learning. Answering these questions appears necessary to ultimately understand why deep learning works.

## Acknowledgements

We thank Levent Sagun, Clément Hongler, Franck Gabriel, Giulio Biroli, Stéphane d’Ascoli for helpful discussions. We thank Riccardo Rivasio and Jonas Paccolat for proofreading. This work was partially supported by the grant from the Simons Foundation (#454953 Matthieu Wyart). M.W. thanks the Swiss National Science Foundation for support under Grant No. 200021-165509.

## References

- M. S. Advani and A. M. Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.
- Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.
- S. Arora, S. S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. *arXiv preprint arXiv:1904.11955*, 2019.
- A. Authors. Anonymized for icml submission. *Somewhere*, 2019.
- M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G. B. Arous, C. Cammarota, Y. LeCun, M. Wyart, and G. Biroli. Comparing dynamics: Deep neural networks versus glassy systems. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323, Stockholm, Sweden, 10–15 Jul 2018. PMLR. URL <http://proceedings.mlr.press/v80/baity-jesi18a.html>.
- Y. Bansal, M. Advani, D. D. Cox, and A. M. Saxe. Minnorm training: an algorithm for training overcomplete deep neural networks. *arXiv preprint arXiv:1806.00730*, 2018.
- L. Chizat and F. Bach. On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport. In *Advances in Neural Information Processing Systems 31*, pages 3040–3050. Curran Associates, Inc., 2018.
- L. Chizat and F. Bach. A Note on Lazy Training in Supervised Differentiable Programming. working paper or preprint, Feb. 2019. URL <https://hal.inria.fr/hal-01945578>.
- A. G. de G. Matthews, J. Hron, M. Rowland, R. E. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018. URL <https://openreview.net/forum?id=H1-nGgWC->.

- S. S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=S1eK3i09YQ>.
- M. Geiger, S. Spigler, S. d’Ascoli, L. Sagun, M. Baity-Jesi, G. Biroli, and M. Wyart. The jamming transition as a paradigm to understand the loss landscape of deep neural networks. *arXiv preprint arXiv:1809.09349*, 2018.
- M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d’Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019.
- A. Jacot, F. Gabriel, and C. 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. URL <http://dl.acm.org/citation.cfm?id=3327757.3327948>.
- Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010. URL <http://yann.lecun.com/exdb/mnist/>.
- J. Lee, L. Xiao, S. S. Schoenholz, Y. Bahri, J. Sohl-Dickstein, and J. Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. *arXiv preprint arXiv:1902.06720*, 2019.
- J. H. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *ICLR*, 2018.
- S. Mei, A. Montanari, and P.-M. Nguyen. A mean field view of the landscape of two-layers neural networks. *arXiv preprint arXiv:1804.06561*, 2018.
- S. Mei, T. Misiakiewicz, and A. Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. *arXiv preprint arXiv:1902.06015*, 2019.
- B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. 2019. URL <https://openreview.net/forum?id=HkgmzhC5F7>.
- R. M. Neal. *Bayesian Learning for Neural Networks*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996. ISBN 0387947248.
- B. Neyshabur, R. Tomioka, R. Salakhutdinov, and N. Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.
- P.-M. Nguyen. Mean field limit of the learning dynamics of multilayer neural networks. *arXiv preprint arXiv:1902.02880*, 2019.
- R. Novak, L. Xiao, Y. Bahri, J. Lee, G. Yang, D. A. Abolafia, J. Pennington, and J. Sohl-dickstein. Bayesian deep convolutional networks with many channels are gaussian processes. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=Big30j0qF7>.
- D. S. Park, J. Sohl-Dickstein, Q. V. Le, and S. L. Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. *arXiv preprint arXiv:1905.03776*, 2019.
- G. M. Rotskoff and E. 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.
- U. Simsekli, L. Sagun, and M. Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5827–5837, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL <http://proceedings.mlr.press/v97/simsekli19a.html>.

- J. Sirignano and K. Spiliopoulos. Mean field analysis of neural networks. *arXiv preprint arXiv:1805.01053*, 2018.
- S. Spigler, M. Geiger, S. d’Ascoli, L. Sagun, G. Biroli, and M. Wyart. A jamming transition from under-to over-parametrization affects loss landscape and generalization. *arXiv preprint arXiv:1810.09665*, 2018.
- C. K. Williams. Computing with infinite networks. In *Advances in neural information processing systems*, pages 295–301, 1997.
- S. Yaïda. Fluctuation-dissipation relations for stochastic gradient descent. In *International Conference on Learning Representations*, 2019. URL <https://openreview.net/forum?id=SkNks0RctQ>.
- G. Yang. Scaling limits of wide neural networks with weight sharing: Gaussian process behavior, gradient independence, and neural tangent kernel derivation. *arXiv preprint arXiv:1902.04760*, 2019.
- S. Zagoruyko and N. Komodakis. Wide residual networks. In *BMVC*, 2016.

## A Continuous dynamics with SGD

In this section we briefly describe the algorithm that we use to perform training, that is a variant of stochastic gradient descent. Stochastic gradient descent introduces noise into the dynamics by computing the gradient on a (random) batch of data of limited size rather than computing the gradient of the total loss on the full batch (i.e. the full training set with  $n$  elements). At fixed batch size  $b$ , the amount of noise introduced by this dynamics is proportional to the discrete time step  $dt$ . To keep the noise constant, we introduce a different parameter  $T$ , playing a role of an effective temperature [Park et al., 2019, Yaida, 2019], and we require that it stays constant throughout learning.

Let assume that the gradient of the loss associated to one pattern  $(x_\mu, y_\mu)$  is equal to the gradient of the total loss plus some random contribution  $r_\mu$ :

$$\nabla_w \ell(f(w, x_\mu), y_\mu) = \nabla_w \mathcal{L}(\{(x_\nu, y_\nu)\}_{\nu=1}^n) + r_\mu. \quad (15)$$

From this we obtain a Fokker-Planck equation for the probability distribution  $p(w, t)$  of the network's weights  $w$  during training:

$$\dot{p}(w, t) = \nabla_w \cdot (\nabla_w \mathcal{L}(w) p(w, t)) + \frac{1}{2} \frac{dt}{b} \left(1 - \frac{b-1}{n-1}\right) \sum_{ij} \partial_i \partial_j (\Sigma_{ij}(w) p(w, t)) \quad (16)$$

where  $\Sigma$  is the covariance matrix that characterizes the random variable  $r$ . Further study of the structure of  $r$  can be found at [Simsekli et al., 2019]. This equation depends on the time step  $dt$  and on the batch size  $b$  only through the effective temperature  $T$ :

$$T = \frac{dt}{b} \left(1 - \frac{b-1}{n-1}\right). \quad (17)$$

The term in parenthesis appears by taking into account that  $\sum_{\mu=1}^n r_\mu = 0$  (since the sum of all the gradients in Eq. (15) equals to the total gradient). A null temperature corresponds to either the limit of continuous dynamics  $dt \rightarrow 0$  or to full-batch gradient descent  $b = n$ . In our implementation the time step  $dt$  is chosen at each step in such a way that the variation of the network's output is bounded. If  $w_i$  is the current set of parameters at step  $i$  and  $w_{i+1}$  is the parameters after a step of our dynamics, we always verify that

$$10^{-4} < \alpha \max_{\mu} |f(w_{i+1}, x_\mu) - f(w_i, x_\mu)| < 10^{-2}, \quad (18)$$

and if it is not the case we change the step  $dt$ . Then, since we fix the temperature  $T$ , the batch size is adapted accordingly as

$$b = \frac{n}{\frac{T}{dt}(n-1) + 1}. \quad (19)$$

## B Architectures and weight initialization

**Fully-connected network** We use a constant-width fully-connected architecture based on [Jacot et al., 2018]. Given an input pattern  $x$ , we denote iteratively the vector of preactivations  $\tilde{z}^\ell$  as

$$\tilde{z}^1 = d^{-1/2} W^0 x + B^0, \quad (20)$$

$$\tilde{z}^{\ell+1} = h^{-1/2} W^\ell \tilde{z}^\ell + B^\ell, \quad (21)$$

$$f(w, x) = h^{-1/2} W^L \tilde{z}^L + B^L, \quad (22)$$

where  $W^\ell$  and  $B^\ell$  are the parameters of the  $\ell$  layer,  $h$  is the width of the network,  $L$  is its depth ( $L = 5$  in our simulations), and  $d$  is the dimension of the input data.  $z^\ell$  is the vector of activations in the previous layer, and it is related to the corresponding preactivation by  $z^\ell = \sigma(\tilde{z}^\ell)$ . Fig. 4 illustrate our choice of notation. All the weights are Gaussian-distributed at initialization  $W_{ij}^\ell \sim \mathcal{N}(0, 1)$ , and the biases  $B^\ell$  are initialized at 0. With this initialization, all the preactivations are of order  $\mathcal{O}(1)$ .

**Convolutional neural network** We initialize the parameters of the convolutional networks as standard Gaussians in such a way that the preactivations are of order unity. Our architecture has 4 hidden layers, and we use a stride of  $/2$  before the first layer and in the middle of the network. After the convolutions we average the spatial dimensions, and the last layer is a simple perceptron. The code describing the architecture is available in the Supplementary Material.

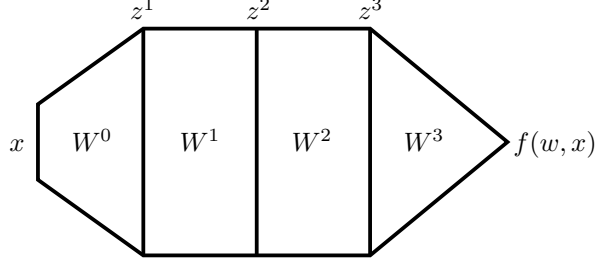


Figure 4: Fully-connected architecture with  $L = 3$  hidden layers.

**Wide resnet** Our resnet architecture (used in Appendix C) is based on [Zagoruyko and Komodakis, 2016]. We use no batch normalization and initialization is as in our fully-connected networks. The code describing the architecture is available in the supplementary material.

## C Frozen NTK dynamics

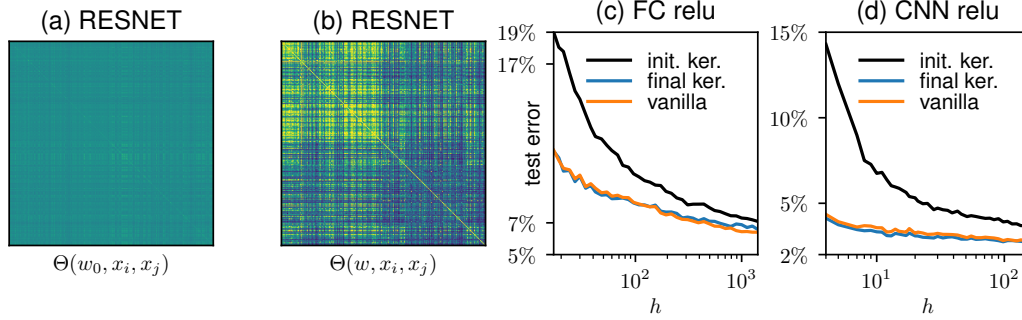


Figure 5: Gram matrix of the test set at initialization  $\Theta(w_0, x_\mu, x_\nu)$  (a) and at the end of training  $\Theta(w, x_\mu, x_\nu)$  (b), for a wide-resnet 28x10 architecture [Zagoruyko and Komodakis, 2016] ( $L = 25$  hidden layers) trained on a binary version of CIFAR10. The first half of the indices  $\mu = 1 \dots n/2$  has label  $y = 1$  and the other half has label  $y = -1$ . The kernel inflates during learning in a way that depends on the two classes. See Appendix B for a description of the architecture. (c-d) Test error v.s. the width  $h$  for the regular dynamics, the dynamics with the frozen kernel at initialization and the dynamics with the frozen kernel of the end of training. The training performance is captured by the kernel.

Let consider the first order approximation  $\tilde{f}_{w_1}(w, x)$  of a model  $f(w, x)$  around  $w = w_1$ ,

$$\tilde{f}_{w_1}(w, x) = \nabla_w f(w_1, x) \cdot w. \quad (23)$$

For instance,  $w_1$  can be the value at initialization or at the end of another dynamics. We can then train this linearized model keeping the gradients fixed:

$$\dot{\tilde{f}}_{w_1}(w) = \nabla_w f(w_1) \cdot \dot{w}, \quad (24)$$

where  $\dot{w}$  depends on the gradient descent procedure. Using gradient descent with momentum as in Eq. (6),

$$\begin{aligned} \dot{\tilde{f}}_{w_1}(w) &= \nabla_w f(w_1) \cdot v, \\ -\tau \dot{v} &= v + \frac{1}{n} \sum_{(x,y) \in \mathcal{T}} \ell'(\tilde{f}_{w_1}(w, x), y) \nabla_w f(w_1, x). \end{aligned} \quad (25)$$

If we define  $\tilde{v} = \nabla_w f(w_1) \cdot v$  we can rewrite the previous equation as

$$\begin{aligned} \dot{\tilde{f}}_{w_1}(w) &= \tilde{v} \\ -\tau \dot{\tilde{v}} &= \tilde{v} + \frac{1}{n} \sum_{(x,y) \in \mathcal{T}} \ell'(\tilde{f}_{w_1}(w, x), y) \Theta(w_1, x), \end{aligned} \quad (26)$$

where  $\Theta$  is the neural tangent kernel defined in Eq. (1). We call these equations the *frozen kernel dynamics*.

The results presented in Fig. 3 state that the network learns a kernel during the training dynamics, and that this learned kernel coincides with the frozen kernel in the lazy regime as  $\sqrt{h}\alpha \rightarrow \infty$ . Another way to see that the kernel changes during training is to plot the so-called Gram matrix of the frozen kernel, namely the matrix  $(\Theta(w, x_\mu, x_\nu))_{\mu, \nu \in \text{test set}}$ : in Fig. 5 (a-b) we show the Gram matrix of the neural tangent kernel evaluated before and after training, where it is clear that there is an emergent structure that depends on the dataset.

It is interesting to test if the performance of deep networks after learning is entirely encapsulated in the kernel it has learned. We argue that indeed this is the case, and to make our point, we proceed as follows. At any time  $t$  during training, we can compute the instantaneous neural tangent kernel  $\Theta(w(t))$  as in Eq. (1); then, we perform a frozen kernel dynamics using that instantaneous kernel, and we evaluate its performance on the test set. In Fig. 5 (c-d) we plot the test error of a network with  $\alpha = 1$  (referred to as “vanilla”) and compare it to the test error of the frozen kernel, both at initialization ( $t = 0$ ) and at the end of training. Quite remarkably  $\tilde{f}_{w(t_2)}$  achieves the full performance of  $f$ .

## D Dynamics of the Weights

In Fig. 6 we plot the rescaled evolution of the parameters  $\sqrt{h}\|w - w_0\|/\|w_0\|$  versus  $\alpha$  (left) and versus  $\sqrt{h}\alpha$  (right) showing that the curves collapse. We find:

$$\frac{\|w - w_0\|}{\|w_0\|} \sim \frac{1}{h\alpha} \quad (27)$$

in the lazy regime. It is expected from Eq. (12), and the fact that the dynamics lasts  $\mathcal{O}(1)$  in this regime, jointly implying that the  $\mathcal{O}(h^2)$  internal weights evolve by  $\mathcal{O}(1/(h\alpha))$ . Finally

$$\frac{\|w - w_0\|}{\|w_0\|} \sim \frac{(\sqrt{h}\alpha)^{-b}}{\sqrt{h}} \quad (28)$$

in the feature-learning regime, where  $b \approx 0.23$  is compatible with  $1/(L+1) \approx 0.17$  as proposed in Section 6. Note that the denominator  $\sqrt{h}$  is also expected from Section 6, where it corresponds to the term  $\|w(t_1) - w(0)\|$  entering in the definition of  $\lambda$ . It comes from the fact that  $\|w(t_1) - w(0)\| \sim \sqrt{h}$ , as follows from the  $\mathcal{O}(h^2)$  internal weights evolving by  $\mathcal{O}(1/\sqrt{h})$  on the time scale  $t_1$ , as can be deduced from Eq. (12).

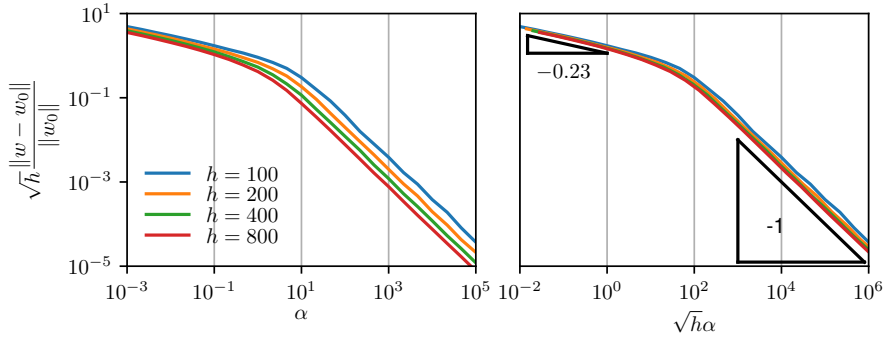


Figure 6: Relative evolution of the parameters  $\sqrt{h}\|w - w_0\|/\|w_0\|$  v.s.  $\alpha$  (left) and  $\sqrt{h}\alpha$  (right). Each measure is averaged over 10 initializations.



## E Dynamics of the Output function

To measure the amplitude of the output of a network  $f$  we define its norm as follow

$$\|f(w)\| = \sqrt{\langle f(w, x_\mu)^2 \rangle_{\mu \in \text{test}}} \quad (29)$$

In Section 6 we argued that the dynamics is linear for  $t \ll t_1 \sim \alpha\sqrt{h}$ . Fig. 7 (a,c,d) confirms that the dynamics, characterized by  $\|f(w_t) - f(w_0)\|$ , is indeed linear on a time scale of order  $t_1$ , independently of the value of  $\alpha$  as shown in Fig. 7 (a) or  $h$  as shown in Fig. 7 (c,d).

Another important result of Section 6 is that at the end of the linear regime, the output has increased by a relative amount  $\sim \sqrt{h}$ . This result is confirmed in Fig. 7(b) showing that  $\sqrt{h}\|f(w_t)\|$  is independent of  $h$  for  $t \sim t_1$ .

These two facts taken together imply:

$$\|f(w_t) - f(w_0)\| \sim \frac{t}{t_1} \sqrt{h}, \quad t \ll t_1 \text{ in feature learning.} \quad (30)$$

This prediction is confirmed in Fig. 7 (d) showing  $\alpha\|f(w_t) - f(w_0)\| \sim t/t_1(\sqrt{h}\alpha)$  which must behave as  $t/t_1$  if  $(\sqrt{h}\alpha)$  is hold fixed, as is the case in this figure.

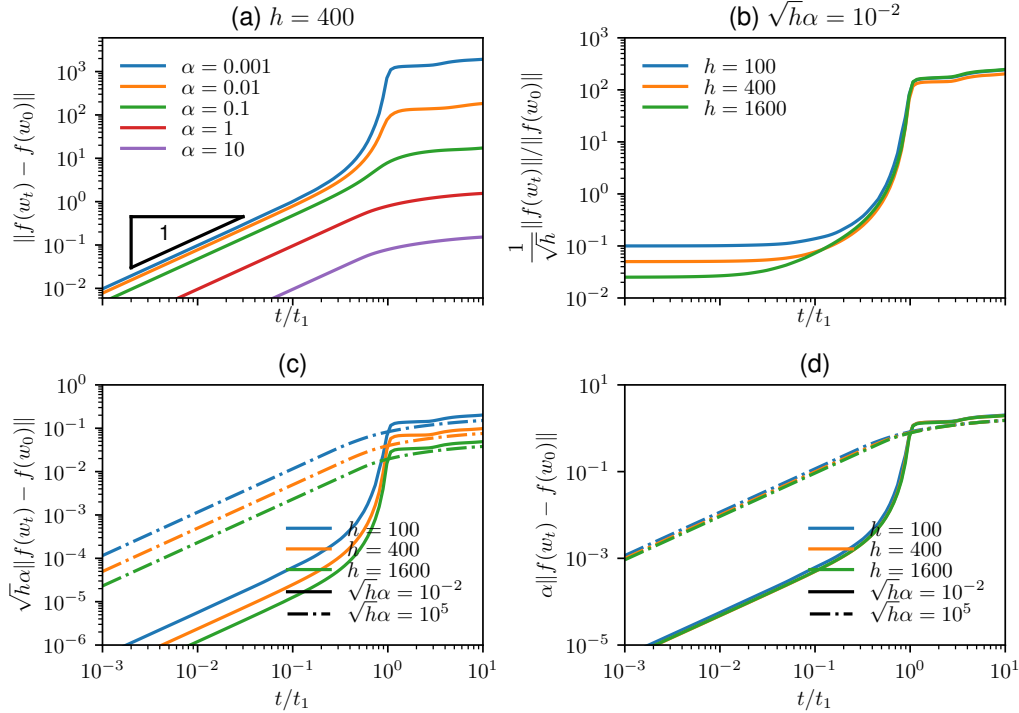


Figure 7: Different measures of the network norm v.s.  $t/t_1$  for (a) a fixed width  $h$  and various  $\alpha$ , (b,c,d) a fixed value of  $\sqrt{h}\alpha$  and various  $h$ . Here  $t_1$  is the time at which the loss reduced by half. The network used here has  $L = 2$  hidden layers and uses a *Softplus* activation function. Each curve is averaged along the y axis for 10 realizations.

## F Preactivation evolution

Fig. 8 shows the amplitude of  $\dot{z}$  at initialization. We measured it using finite difference method by applying a single gradient descent step. The observation is consistent with the assumption done in Section 6.

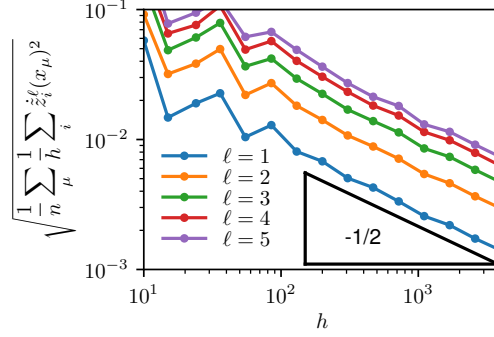


Figure 8:  $\dot{z}$  at  $t = 0$  v.s.  $h$  for different layers. Each measure is averaged over 5 networks. The network used here has  $L = 5$  hidden layers and uses a *Softplus* activation function.

## G Shallow network

In order to verify the depth dependence of our heuristic predictions about the powerlaw in  $\alpha$  of the quantities  $\|\Theta - \Theta_0\|$  and  $\|w - w_0\|$ , we reran the experiment with 2 hidden layers ( $L = 2$ ) with the fully-connected network and *Softplus*. In Table 1 we summarize the exponent found numerically, they are compatible the our predictions.

Observable	$L = 5$	$L = 2$	Prediction
$\ \Theta - \Theta_0\ $	1.7 (1.66)	1.25 (1.33)	$\frac{2}{1+1/L}$
$\ w - w_0\ $	0.23 (0.166)	0.35 (0.333)	$\frac{1}{1+L}$

Table 1: Powerlaw dependence in  $\alpha$ , measure and prediction (in parenthesis) of the exponent  $a$  where  $O \sim \alpha^{-a}$  for  $\alpha \ll 1$

## H ReLU activation function

Fig. 9 shows the evolution of the kernel as a function of  $\sqrt{h}\alpha$  for a network with *ReLU* activation function. Differently from the *Softplus* case (see Fig. 3 (a)), here we observe the existence of three regimes, each characterized by a different power law. The intermediate regime with slope  $-1/2$  is not present for *Softplus*, and it is compatible with the following explanation. The *ReLU* function  $x \mapsto \max(0, x)$  is non differentiable in  $x = 0$ . It implies that  $w \mapsto f(w)$  is not differentiable. For a finite dataset,  $w \mapsto \{f(w, x_{\mu})\}_{\mu}$  is differentiable only on small patches. As we can see in Fig. 9 for large enough  $\alpha$ ,  $w$  evolution is so small that  $f$  remains in a differentiable patch and we get the predicted result of slope  $-1$ . But for intermediate values of  $\alpha$ , the network change patches a number of times proportional to  $\alpha^{-1}$  and assuming that each changes in the kernel induced by these changes of patch are not correlated, their sum scales with  $\alpha^{-1/2}$ .

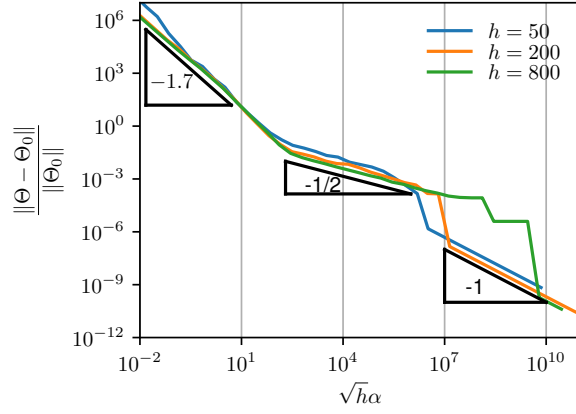


Figure 9:  $\frac{\|\Theta - \Theta_0\|}{\|\Theta_0\|}$  v.s.  $\sqrt{h\alpha}$  for different heights. Each measure is averaged over 3 networks. The network used here has  $L = 5$  hidden layers and uses a  $ReLU$  activation function.