Redundant representations help generalization in wide neural networks*,**,†

Diego Doimo1,2,**, Aldo Glielmo2,3, Sebastian Goldt2 and Alessandro Laio2,4

1 Area Science Park, Trieste, Italy
2 International School for Advanced Studies, Trieste, Italy
3 Bank of Italy, Rome, Italy
4 International Center for Theoretical Physics, Trieste, Italy
E-mail: ddoimo@sissa.it

Received 23 May 2023
Accepted for publication 10 July 2023
Published 15 November 2023

Online at stacks.iop.org/JSTAT/2023/114011
https://doi.org/10.1088/1742-5468/aceb4f

Abstract. Deep neural networks (DNNs) defy the classical bias-variance trade-off; adding parameters to a DNN that interpolates its training data will typically improve its generalization performance. Explaining the mechanism behind this ‘benign overfitting’ in deep networks remains an outstanding challenge. Here, we study the last hidden layer representations of various state-of-the-art convolutional neural networks and find that if the last hidden representation is wide enough, its neurons tend to split into groups that carry identical information and differ from each other only by statistically independent noise. The number of these groups increases linearly with the width of the layer, but only if the

*The views and opinions expressed in this paper are those of the authors and do not necessarily reflect the official policy or position of Bank of Italy.
†This article is an updated version of: Doimo D, Glielmo A, Goldt S and Laio A 2022 Redundant representations help generalization in wide neural networks Advances in Neural Information Processing Systems vol 35, ed S Koyejo, S Mohamed, A Agarwal, D Belgrave, K Cho and A Oh (Curran Associates, Inc.) pp 19659–72.
**Author to whom any correspondence should be addressed.

Original Content from this work may be used under the terms of the Creative Commons Attribution 4.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
width is above a critical value. We show that redundant neurons appear only when the training is regularized and the training error is zero.

**Keywords:** deep learning, machine learning

### Contents

1. **Introduction** ................................................. 3

2. **Methods** ................................................. 5
2.1. Neural network architectures ................................. 5
2.1.1. Fully-connected networks on MNIST ....................... 5
2.1.2. Wide-ResNet-28 and DenseNet40 on CIFAR10/100 .......... 5
2.1.3. ResNet50 on ImageNet .................................... 5
2.2. Analytical methods ......................................... 6
2.2.1. Reconstructing a wide representation from a smaller chunk 6
2.2.2. Reproducibility ......................................... 6

3. **Results** .................................................. 6
3.1. The test error of chunks of $w_k$ neurons of the final representation
asymptotically scales as $w_k^{-1/2}$ .................................. 6
3.2. Clones interpolate the training data .......................... 8
3.3. Clones reconstruct the full representation almost perfectly ................ 8
3.4. Clones differ from each other by uncorrelated random noise ............ 10
3.5. The dynamics of training ..................................... 10
3.6. Clones appear only in regularized networks .................. 11
3.7. Clones appear only if a network interpolates the training set: the case of
ImageNet ......................................................... 12

4. **Discussion** ............................................... 13
4.1. The impact of network architecture .......................... 13
4.2. The impact of training ....................................... 14
4.3. Neural scaling laws ......................................... 14
4.4. Relation to theoretical results in the mean-field regime ............ 14
4.5. Bias-variance trade-off and implicit ensembling .................. 15
4.6. Impact for applications ..................................... 15

**Appendix A. Hyperparameters used and training procedures** ........... 15
A.1. Fully-connected networks on MNIST .......................... 15
A.2. Wide-ResNet-28 and DenseNet40-BC on CIFAR10/100 ........ 15
A.3. ResNet50 on ImageNet ..................................... 16

**Appendix B. Additional experiments** ................................ 17

**References** .................................................. 20

https://doi.org/10.1088/1742-5468/aceb4f
1. Introduction

Deep neural networks (DNNs) have enough parameters to achieve zero training error, even with random labels [1, 2]. Contrary to the classical bias-variance trade-off, the performance of these interpolating classifiers improves as the number of parameters increases, well beyond the number of training samples [3–6]. Despite recent progress in describing the implicit bias of stochastic gradient descent towards ‘good’ minima [7–12], and the detailed analysis of solvable models of learning [13–21], the mechanisms underlying this ‘benign overfitting’ [22] in DNNs remain unclear, especially since their loss landscape contains ‘bad’ local minima and stochastic gradient descent (SGD) can reach them [23].

In this paper, we describe a phenomenon in wide DNNs that could be a possible mechanism for benign overfitting when the networks are trained with regularization. We illustrate this mechanism in figure 1 for a family of increasingly wide DenseNet-40s [24] trained on CIFAR10 [25] following common practice, in particular, using weight decay (see section 2.1). For simplicity, we refer to the width \( W \) of the last hidden representation as the width of the network. The blue line in figure 1(b) shows that the average classification error \( \text{error} \) approaches the performance of a large ensemble of networks \( \text{error}_\infty \) [21] as we increase the network width \( W \). In agreement with [26], the performance of these DenseNets improves continuously with width. For widths greater than 350, the networks are wide enough to reach zero training error (see appendix B, figure S2(c)) and, interestingly, their test error decays approximately as \( W^{-1/2} \). Our goal is to understand how the error of the network can keep decaying beyond the interpolation threshold, and why it decays as \( W^{-1/2} \).

We make our key observation by performing the following experiment. We randomly select a number \( w_c \) of neurons from the last hidden layer of the widest DenseNet-40 and remove all the other neurons from that layer as well as their connections (figure 1(a)). We then evaluate the performance of this chunk of \( w_c \) neurons, without retraining the network. The orange profile of figure 1(b) shows a test error for chunks of varying sizes. There are two regimes. For small chunks, the error decays faster than \( w_c^{-1/2} \), while beyond a critical chunk size \( w_c^* \) (shaded area), the error of a chunk of \( w_c \) neurons is roughly the same as that of the full network with \( w_c \) neurons. Furthermore, the error of the chunks decays with the same power-law \( w_c^{-1/2} \) beyond this critical chunk size.

The decay rate of \(-1/2\) suggests that in this regime chunks of \( w_c \) neurons can be thought of as statistically independent estimators of the same features of the data, differing only by small, uncorrelated noise. In other words, beyond the critical width \( w_c^* \), the final hidden representation of an input in a trained, wide DNN becomes highly redundant. This motivates a possible mechanism for benign overfitting, schematically portrayed in figure 1(c). As the network becomes wider, additional neurons are first used to learn new features of the data. Beyond the critical width \( w_c^* \), additional neurons in the final layer do not fit new features in the data, and hence over-fit. Instead, they make a copy or a clone of a feature that is already part of the final representation. The last layer thus splits into more and more clones as the network grows wider, as illustrated at

https://doi.org/10.1088/1742-5468/aceb4f
Redundant representations help generalization in wide neural networks

**Figure 1.** Redundancy of representations in wide neural networks. (a) We analyze the final representations of DNNs, namely the activities of the last hidden layer of neurons (light blue). We focus on the performance and statistical properties of randomly chosen subsets of \( w_c \) neurons, which we call ‘chunks’. In the chunked network shown here, \( w_c = 5 \) out of 9 neurons are kept and used to predict the output. (b) As we increase the size of the chunk \( w_c \) that we keep in a state-of-the-art DNN, here a DenseNet40, the test error of the chunk (orange line) becomes similar to the test error of a full network of width \( W = w_c \) (blue line). In this regime, which is reached when \( w_c \) is larger than a threshold \( w_c^* \) (shaded area) the error approaches its asymptotic value \( \text{error}_\infty \) as a power-law \( w_c^{-1/2} \) (dashed line). \( \text{error}_\infty \) is the error of an ensemble average of 20 networks of the widest size. (c) Illustration of three final representations for networks of increasing width. In small networks, an additional neuron fits new features of the data (red neuron). As the network width goes beyond a critical width \( W^* \), additional neurons instead copy features already learned from the data, instead of over-fitting to features that are not relevant to the task. This mechanism is suggested by the \( w_c^{-1/2} \) decay of the chunk error, and by statistical analysis, we present in this paper.

the bottom of figure 1. The accuracy of these wide networks then improves with their width because the network implicitly averages over an increasing number of clones in its representations to make its prediction.

This paper provides a quantitative analysis of this phenomenon on various data sets and architectures. Our main findings can be summarized as follows:

1. A chunk of \( w_c \) random neurons in the last hidden representation of a wide neural network predicts the output with an error that decays as \( w_c^{-1/2} \) if the layer is wide enough and \( w_c \) is large enough. In this regime, we call the chunk a ‘clone’;
2. Clones fit the training set with zero error and can be linearly mapped to one another, or to the full representation, with an error that can be described as uncorrelated random noise.

3. Clones appear if the model is trained with weight decay and the training set is fitted with zero error. If training is stopped too early or if the training is performed without regularization, 1. and 2. do not take place, even if the last representation is very wide.

2. Methods

2.1. Neural network architectures

We report experimental results obtained with several architectures (fully connected networks, Wide-ResNet-28, DenseNet40, ResNet50) and data sets (CIFAR10/100 [25], ImageNet [27]). We train all the networks using SGD with momentum and, importantly, weight decay. The amount of weight decay is found with a small grid search, while the other relevant hyperparameters are set following standard practice. We give detailed information on our training setups in appendix A. All our experiments are run on Volta V100 GPUs. In the following paragraphs, we describe how we vary the width W of the models.

2.1.1. Fully-connected networks on MNIST. We train a fully-connected network to classify the parity of the MNIST digits [28] (pMNIST) following the protocol of Geiger et al [21]. MNIST digits are projected onto the first ten principal components, which are then used as inputs of a five-layer fully-connected network (FC5). The four hidden representations have the same width W and the output is a real number whose sign is the predictor of the parity of the input digit.

2.1.2. Wide-ResNet-28 and DenseNet40 on CIFAR10/100. We train CIFAR10 and CIFAR100 on a family of Wide-ResNet-28 [26] (WR28). The number W of the last hidden neurons in a WR28-n is 64 · n, obtained after average pooling of the last 64 · n channels of the network. In our experiments, we also analyze two narrow versions of the standard WR28-1, which are not typically used in the literature. We name them WR28-0.25 and WR28-0.5 since they have 1/4 and 1/2 of the number of channels of WR28-1. Our implementation of DenseNet40 follows the DenseNet40-BC variant [24]. We vary the number of input channels c in \{16, 32, 64, 128, 256\}, which is twice the growth rate k of the networks [24]. The number W of the last hidden features of this architecture is 5.5 · c.

2.1.3. ResNet50 on ImageNet. We modify the ResNet50 architecture [29] by multiplying by a constant factor c ∈ \{0.25, 0.5, 1, 2, 4\} the number of channels of all the layers after the input stem. When c = 2 our networks differ from the standard Wide-ResNet50-2 [26] since we double the channels of all the layers and not just those of the bottleneck of the ResNet blocks. As a consequence, in our implementation, the number of features after the last pooling layer is W = 2048 · c while in [26] W is fixed at 2048.

https://doi.org/10.1088/1742-5468/aceb4f
2.2. Analytical methods

2.2.1. Reconstructing a wide representation from a smaller chunk. To determine how well a subset of \( w \) neurons can reconstruct the full representation of size \( W \) we seek the \( W \times w \) linear map \( A \), which is able to minimize the squared difference \((x^{(W)} - \hat{x}^{(W)})^2\) between the \( W \) activations of the full layer representation, \( x^{(W)} \), and the activations predicted from the chunk of size \( w \), \( \hat{x}^{(W)} \):

\[
\hat{x}^{(W)} = Ax^{(w)}. \tag{1}
\]

This least-squares problem is solved with ridge regression [30] with regularization set to \( 10^{-8} \), and we use the \( R^2 \) coefficient of the fit to measure the predictive power of a given chunk size. The \( R^2 \) value is computed as an average of the single-activation \( R^2 \) values corresponding to the \( W \) output coordinates of the fit, weighted by the variance of each coordinate. We further compute the \( W \times W \) covariance matrix \( C_{ij} \) of the residuals of this fit, and from \( C_{ij} \) we obtain the correlation matrix as follows:

\[
\rho_{ij} = \frac{C_{ij}}{\sqrt{C_{ii}C_{jj}+10^{-8}}}, \tag{2}
\]

with a small regularization in the denominator to avoid instabilities when the standard deviation of the residuals falls below machine precision. To quantify how well the errors of the fit are correlated, we average the absolute values of the non-diagonal entries of the correlation matrix \( \rho_{ij} \). For short, we refer to this quantity as a ‘mean correlation’.

2.2.2. Reproducibility. We provide a code to reproduce our experiments and our analysis online at https://github.com/diegodoimo/redundant_representation.

3. Results

3.1. The test error of chunks of \( w_c \) neurons of the final representation asymptotically scales as \( w_c^{-1/2} \)

The mechanism that we propose is inspired by the following experiment. We compute the test accuracy of models obtained by selecting a random subset of \( w_c \) neurons from the final hidden representation of a wide neural network. We select \( w_c \) neurons at random and compute the test accuracy of a network in which we set to zero the activation of all the other \( w - w_c \) neurons of the final layer. Importantly, we do not fine-tune the weights after selecting the \( w_c \) neurons. All the remaining parameters of the previous layers are left unchanged and only the activations of the ‘killed’ neurons of the last hidden representation are not used to compute the logits. We take 500 random samples of neurons for each chunk width \( w_c \). We consider three different data sets: pMNIST trained on a fully connected network, CIFAR10 and CIFAR100 trained on convolutional
Figure 2. Scaling of the test error with width for various DNNs. Average test error of neural networks with various architectures approaches the test error of an ensemble of these networks as the network width increases. Network size shown here is the width of the final representation. For large width, we find a power-law behavior $\text{error} - \text{error}_\infty \propto W^{-1/2}$ across data sets and architectures. Full experimental details can be seen in section 2.1.

networks. The width $W$ of the network is 512 for pMNIST and CIFAR10, and $W = 1024$ for CIFAR100 (see section 2.1). In all these cases, $W$ is large enough to be firmly in the regime where the accuracy of the network scales (approximately) as $W^{-1/2}$ (see figure 2).

In figure 3, we plot the test error of the ‘chunked models’ as a function of $w_c$ (orange lines). This behavior is similar in all three networks. The test error decays as $w_c^{-1/2}$ for chunks that are larger than a critical value $w_c^*$, which depends on the data set and architecture used. This decay follows the same law observed for full networks of the same width (figure 2). This implies that a model obtained by selecting a random chunk of $w_c > w_c^*$ neurons from a wide final representation behaves in a similar way to a full network of width $W = w_c$. Furthermore, the decay with rate $-1/2$ suggests that the final representation of wide networks can be thought of as a collection of statistically independent estimates of a finite set of data features relevant for classification. Adding neurons to the chunk hence reduces their prediction error in the same way that an additional measurement reduces the measurement uncertainty, leading to the $-1/2$ decay.

At $w_c$ smaller than $w_c^*$ instead, the test error of the chunked models decays faster than $w_c^{-1/2}$ in all the cases we considered, including the DenseNet architecture trained on CIFAR10 shown in figure 1(b). In this regime, adding neurons to the final representation improves the quality of the model significantly quicker than it would in independently trained models of the same width (see figure 1(c) for a pictorial representation of this process). We call chunks of neurons of size $w_c \geq w_c^*$ clones. In the following, we characterize more precisely the properties of the clones.
Redundant representations help generalization in wide neural networks

Figure 3. Scaling of the test error of chunks of neurons extracted from the final hidden representation of wide neural networks. We plot how the test error of chunked networks approaches $\text{error}_\infty$, the error of an ensemble of 20 networks of the widest size (e.g. $W = 1024$ for CIFAR100), as the chunk size $w_c$ increases. Chunks are formed by selecting a number of $w_c$ neurons at random from the final hidden representation of the widest networks: a FC5 on pMNIST (width $W = 512$), and Wide-ResNet-28 for CIFAR10 ($W = 512$) and CIFAR100 ($W = 1024$). Shaded regions indicate regions where the error of the chunks with $w_c$ neurons decays as $w_c^{-1/2}$.

3.2. Clones interpolate the training data

A trained deep network often represents the salient features of the data set well enough to achieve (close to) zero classification error on the training data. In the top panels of figure 4, we show that wide networks can also interpolate their training set using just a subset of $w_c > w_c^*$ random neurons. The dark orange profiles show that when the size of a chunk is greater than $w_c^* \sim 50$ for pMNIST, 100 for CIFAR10 and 200 for CIFAR100, the predictive accuracy on the training set remains almost 100%. The minimal size of a clone $w_c^*$ can be identified with the minimal number of neurons required to interpolate the training set. Beyond $w_c^*$, the neurons in the final representation become redundant since the training error remains (close to) zero even after removing neurons from it. The number of distinct clones in a network of width $W$ is $n = W/w_c^*$. If distinct clones provide independent measures of the same salient features of the data, the test error decays approximately as $n^{-1/2}$ or equivalently $W^{-1/2}$. In the following, it can indeed be seen that distinct clones differ from each other by uncorrelated random noise.

3.3. Clones reconstruct the full representation almost perfectly

From a geometrical perspective, the important features of the final representation correspond to directions in which the data landscape shows large variations [31]. A clone is a chunk that is wide enough to encode almost exactly these directions (since its training error is almost zero), but uses far fewer neurons than the full final representation. We analyze this aspect by performing a linear reconstruction of the $W$ activations of the last hidden representation of the widest network starting from a random subset of $w_c$ activations using ridge regression with a small regularization penalty according to equation (1). The blue profiles in figures 4(d)–(f) show the $R^2$ coefficient of fit as

https://doi.org/10.1088/1742-5468/aceb4f
Redundant representations help generalization in wide neural networks

Figure 4. Three signatures of representation redundancy. (i) Training errors of the full networks (blue) and of the chunks taken from the widest network (orange) approach zero beyond a critical width/chunk size, resp. (panels (a)–(c)). (ii) Final representation of the widest network can be reconstructed from a chunk using linear regression (1) with an explained variance $R^2$ close to 1 (blue lines in panels (d)–(f)). (iii) Residuals of the linear map can be modeled as independent noise. We show this by plotting the mean correlation of these residuals (green line, panels (d)–(f)), averaged over 100 reconstructions starting from different chunks. Low correlation at high $R^2$ indicates that the chunk contains the information of the full representation with some statistically independent noise. Experimental setup: FC5 on pMNIST, Wide ResNet-28 on CIFAR10/100. Full details in methods section 2.1.

a function of the chunk size $w_c$ for pMNIST (left), CIFAR10 (center) and CIFAR100 (right). When $w_c$ is very small, say below 6 for pMNIST, 20 for CIFAR10 and 60 for CIFAR100, the $R^2$ coefficient grows almost linearly with $w_c$. In this regime, adding a randomly chosen activation from the full representation to the chunk substantially increases $R^2$. When $w_c$ becomes larger, $R^2$ almost reaches one. This transition happens when $w_c$ is still much smaller than $W$ and corresponds approximately to the regime in which the test error starts scaling with the inverse square root of $w_c$ (see figure 3). The almost perfect reconstruction of the original data landscape with a few neurons is a consequence of the low intrinsic dimension (ID) of the representation [32]. The ID of the widest representations gives a lower bound on the number of coordinates required to describe the data manifold, and hence on the neurons that a chunk needs in order to have the same classification accuracy as the whole representation. The ID of the last hidden representation is 2 in pMNIST, 12 in CIFAR10 and 14 in CIFAR100, numbers which are much lower than $w_c^*$, the width at which a chunk can be considered a clone.

---

3 The linear trend can not be clearly seen in figure 4 as we plot the x-axis with a logarithmic scale.
3.4. Clones differ from each other by uncorrelated random noise

When \( w_c > w^*_c \) the small residual difference between the chunked representation and the full representation, it can be approximately described as statistically independent random noise. The green profiles of the bottom panels of figure 4 show the mean correlation of the residuals of the linear fit (see section 2.2). Below \( w^*_c \), the residuals are not only large but also significantly correlated since they are related to relevant features of the data that are not covered by the neurons of the chunk. As the chunk width increases above \( w^*_c \), the correlation between residuals basically drops to zero. Therefore, in networks wider than \( w^*_c \) any two chunks of equal size \( w_c > w^*_c \) can be effectively considered as equivalent copies or clones of the same representation (that of the full layer), differing only by a small and non-correlated noise, consistent with the scaling law of the error shown in figure 3.

3.5. The dynamics of training

In the previous paragraphs, we set forth evidence in support of the hypothesis that large chunks of the final representation of wide DNNs behave approximately like an ensemble of independent measures of the full feature space. This allows us to interpret the decay of the test error of the full network with the network width observed empirically in figure 2. The three conditions that a chunked model satisfies in the regime in which its test error decays as \( w^{-1/2} \) are represented in figure 4: (i) the training error of the chunked model is close to zero; (ii) the chunked model can be used to reconstruct the full final representation with an \( R^2 \sim 1 \) and (iii) the residuals of this reconstruction can be modeled as independent random noise. These three conditions are all observed at the end of the training. We now analyze the training dynamics. It can be seen that for clones to arise, models not only need to be wide enough but they also, crucially, need to be trained to maximize their performance.

Clones are formed in two stages, which occur at different times during training. The first phase begins as soon as training starts. The network gradually adjusts the chunk representations in order to produce independent copies of the data manifold. This can be clearly observed in figure 5(a), which depicts the mean correlation between the residuals of the linear fit from the chunked to the full final representations of the network, the same quantity that we analyze in figures 4(d)–(f) (green profiles), but now as a function of the training epoch. Both figures 4 and 5 analyze the WR28-8 on CIFAR10. As training proceeds, the correlations between residuals gradually diminish until epoch 160 and become particularly low for chunks greater than 64. After epoch 160, further training does not bring any sizable reduction in their correlation. At epoch 160, the full network also achieves zero error on the training set, as shown in figure 5(b) (brown) and figure 5(c) (blue). This event marks the end of the first phase and the beginning of the second phase where the training error of the clones keeps decreasing while the full representation (blue) has already reached zero training error. For example, chunks of size 64 at epoch 150 have training errors comparable to the test error (dashed line of the middle panel). In the subsequent \( \sim 20 \) epochs the training error of clones of size 128 and 256 reaches exactly zero, and the training error of chunks of size 64 reaches a plateau.

https://doi.org/10.1088/1742-5468/aceb4f
Figure 5. Onset of clones during training. (a) As in figure 4, we show the mean correlation of the residuals of the linear reconstruction of the final representation from chunks, but this time as a function of training epochs. Small correlation indicates that the reconstruction error in going from chunks to final representation can be modeled as independent noise. Data obtained from the same WR28-8 trained on CIFAR10 as in figure 4. (b) Training error during training for chunks of different sizes. After the network has reached zero training error at ∼160 epochs, continuing to train improves the training accuracy of the chunks. (c) Test and training error during training for the full network. Between epochs 160 and 180, the clones of the full network progressively achieve zero training error. In the same epochs, one observes a small improvement in the test error.

Importantly, both phases improve the generalization properties of the network. This can be seen in figure 5(c), which reports the training and test error of the network, with the two phases highlighted. The figure shows that both phases lead to a reduction in test error, although the first phase leads by far to the greatest reduction, consistent with the fact that the greatest improvements in accuracy typically arise during the first epochs of training. The formation of clones is finished around epoch 180 when all the clones have reached almost zero error on the training set. After epoch 180, we also observe that the test error stops improving. In appendix B, we report the same analysis done on CIFAR100 (see figure S1) and CIFAR10 trained on a DenseNet40 (see figures S2(d)–(f)).

3.6. Clones appear only in regularized networks

So far in this work, we have shown only examples of regularized networks and data sets in which representations are redundant. However, if the network is not regularized, some of the signatures described above do not appear even if the width of the final representation is much larger than $W^*$ (the minimum interpolating width). Figure 6 shows the case of the Wide-ResNet28-8 analyzed in figure 5 trained on exactly the same data set (CIFAR10) but without weight decay. As shown in figure 6(a) in the network trained without regularization (blue line) the error does not scale as $w_c^{-1/2}$. This, as we have seen, indicates that the last hidden representation cannot be split into clones equivalent to the full layer. Indeed, the mean correlation between the residuals of the linear map of the chunks to the full representation remains approximately constant during training (figure 6(b)) and is always much higher than what we observe for the same architecture.
Redundant representations help generalization in wide neural networks

Figure 6. Network trained without weight decay on CIFAR10. (a) Test error of chunks of a Wide-ResNet28-8 trained without weight decay (blue) and with weight decay (orange, taken from figure 3(b)). (b) Mean correlation between residuals of the linear reconstruction of the full representation from chunks of different sizes for two networks: one trained without weight decay (thick lines), and one using weight decay (thin lines, same data as in figure 5(a)).

and data set when training is performed with weight decay. We performed a similar analysis on DenseNet40 (see figure S3), observing an analogous trend.

3.7. Clones appear only if a network interpolates the training set: the case of ImageNet

We see that a chunk of neurons can be considered a clone if it fully captures the relevant features of the data, achieving almost zero training error (see figure 4). This condition is not satisfied for most of the networks trained on ImageNet [16]. Therefore, we do not expect to see redundant representations in this important case. We verify this hypothesis by training a family of ResNet50s where we multiply all the channels of the layers after the input stem by a constant factor \(c \in \{0.25, 0.5, 1, 2, 4\}\). In this manner, the widest final representation we consider consists of 8192 neurons, which is four times wider than both the standard ResNet50 [29] and its wider version [26] (see section 2.1). We trained all the networks following standard protocols and achieved test errors comparable to or slightly lower than those reported in the literature (see appendix A). We find that, even in the case of the largest ResNet50, the top-1 error on the training set is \(~8\%\) (see figure 7(a)) and the network does not achieve interpolation, as also discussed in [16].

In this setting, none of the elements associated with the development of independent clones can be observed. The scaling of the test error of the chunks is steeper than \(w_c^{-1/2}\) (see figure 7(b)), suggesting that chunks remain significantly correlated to each other. Figure 7(c) shows that the mean correlation of the residuals does not decrease during training, as it happens for the networks we trained on CIFAR10 and CIFAR100. We conclude that in a ResNet50, a representation with 8192 neurons is too narrow to encode all the relevant features redundantly in ImageNet, and a chunk as large as 4096 activations is not able to reconstruct all the relevant variations of the data as it does in the cases analyzed in section 3.
4. Discussion

This work is an attempt to explain the paradoxical observation that over-parameterization boosts the performance of DNNs. This ‘paradox’ is not a peculiarity of DNNs. If one trains a prediction model with \( n \) parameters using the same training set but starting from independent initial weights and receiving samples in an independent way, one can obtain say, \( m \) models which, under suitable conditions, provide predictions of the same quantity with independent noise due to initialization and SGD scheduling, etc. If one estimates the target quantity by an ensemble average, the statistical error will (ideally) scale with \( m^{-1/2} \), and therefore with \( N^{-1/2} \), where \( N = nm \) is the total number of parameters of the combined model. This will happen even if \( N \) is much larger than the amount of data.

What is less trivial is that a DNN can accomplish this scaling within a single model, in which all the parameters are optimized collectively via the minimization of a single loss function. Our work describes a possible mechanism at the basis of this phenomenon in the special case of neural networks in which the last layer is very wide and the model is regularized. We observe that if the layer is wide enough, random subsets of its neurons can be viewed as approximately independent representations of the same data manifold (or clones). This implies a scaling of the error with the width of the layer as \( W^{-1/2} \), which is qualitatively consistent with our observations.

4.1. The impact of network architecture

The capability of a network to produce statistically independent clones is a genuine effect of the over-parameterization of the whole network, as we find that redundancies appear even if the last layer width is kept constant and the width of all intermediate layers is increased (see appendix B, figure S4(a)). At the same time, we also verified that if the network is too narrow to interpolate the training set, increasing the width
of only the final representation is not sufficient to make the last layer redundant. We give an example of this effect in figure S4(b), where we show that the test error of a WR28-1 on CIFAR10 does not decrease if only the width of the final representation is increased while the rest of the architecture is kept at a constant width.

4.2. The impact of training

The mechanism we describe is robust to different training objectives since we train the convolutional networks with cross-entropy loss and fully connected networks with hinge loss. However, even for wide enough architectures, clones appear only if the training is continued until the training error reaches zero. In our examples, by stopping the training too early, for example, when the training error is similar to the test error, the chunks of the last representation would not become entirely independent of one another, and therefore they could not be considered clones.

4.3. Neural scaling laws

Capturing the asymptotic performance of neural networks via scaling laws is an active research area. Hestness et al [33] gave an experimental analysis of the scaling laws w.r.t. the training data set size in a variety of domains. Rosenfeld et al [34] and Kaplan et al [35] experimentally explored the scaling of the generalization error of deep networks with the number of parameters/data points across architectures and application domains for supervised learning, while Henighan et al [36] identified empirical scaling laws in generative models. Bahri et al [37] showed the existence of four scaling regimes and described them theoretically in the NTK or lazy regime [38–40], where the network weights stay close to their initial values throughout training. None of these works proposes a mechanism that would explain these scalings with the properties of the representation. Geiger et al found that the generalization error can be related to the fluctuations of the output induced by initialization and showed that it scales as $W^{-1}$ in networks trained without weight decay both in the NTK [21] and in the mean field [41] regimes. Instead, we consider the feature learning regime and train our networks with weight decay, which is unavoidable to obtain models with state-of-the-art performance. This might explain the difference in the scaling law that we observe empirically. Previous theoretical work did not study the impact of weight decay on scaling laws, so we hope that our results can spark further studies on the role of this essential regularizer.

4.4. Relation to theoretical results in the mean-field regime

Our empirical results also agree with the recent theoretical results that were obtained for two-layer neural networks [42–47]. These works characterize the optimal solutions of two-layer networks trained on synthetic data sets with some controlled features. In the limit of infinite training data, these optimal solutions correspond to networks where neurons in the hidden layer duplicate key features of the data. These ‘denoising solutions’ or ‘distributional fixed points’ were found for networks with wide hidden layers [42–45] and wide input dimension [46, 47]. Another point of connection with

https://doi.org/10.1088/1742-5468/aceb4f
the theoretical literature is the concept of dropout stability. A network is said to be $\varepsilon$-dropout stable if its training loss changes by less than $\varepsilon$ when half the neurons are removed at random from each of its layers [48]. Dropout stability has been rigorously linked to several phenomena in neural networks, such as the connectedness of the minima of their training landscape [49, 50].

4.5. Bias-variance trade-off and implicit ensembling

The success of various deep learning architectures and techniques has been linked to some form of ensembling. The successful dropout regularization technique [51, 52] samples from an exponential number of ‘thinned’ networks during training to prevent co-adaptation of hidden units. While this can be seen as a form of (implicit) ensembling, here we observe that co-adaptation of hidden units in the form of clones occurs without dropout, and are crucial for their improved performance with width. Recent theoretical work on random features showed that ensembling and over-parameterization are two sides of the same coin and that both mitigate the increase in the variance of the network, which classically leads to worse performance with over-parameterization due to the bias-variance trade-off [18–20]. The plots of bias and variance in figure S5 for the architectures trained on the CIFAR10 and CIFAR100 data sets show that the clone size in these cases is slightly above the peak of the variance and almost coincides with the interpolation width of the full networks of the same size.

4.6. Impact for applications

The framework introduced in this work allows verifying if a neural network is sufficiently expressive to encode multiple statistically independent representations of the same ground truth, which we believe is a fair proxy for model quality and robustness. In particular, we find that reaching interpolation on the training set is not necessarily detrimental to generalization and is instead a necessary condition for developing redundancies which, in turn, reduces the test error.

Appendix A. Hyperparameters used and training procedures

A.1. Fully-connected networks on MNIST

We train the fully-connected networks for 5000 epochs with stochastic gradient descent using the following hyperparameters: batch size = 256, momentum = 0.9, learning rate = $10^{-3}$, weight decay = $10^{-2}$. We optimize our networks using Adam.

A.2. Wide-ResNet-28 and DenseNet40-BC on CIFAR10/100

All the models are trained for 200 epochs with stochastic gradient descent with a batch size = 128, momentum = 0.9, and cosine annealing scheduler starting with a learning rate of 0.1. The training set is augmented with horizontal flips with 50% probability and random cropped images padded with four pixels on each side. On CIFAR10 trained
on WR28 we select a weight decay equal to $5 \times 10^{-4}$ and label smoothing magnitudes equal to 0.1 for WR28-$\{0.25, 0.5, 1, 2\}$ and equal to 0 for WR28-$\{4, 8\}$. On CIFAR10 trained on DenseNet40-BC we set a weight decay equal to $5 \times 10^{-4}$ and label smoothing magnitudes equal to 0.05 for all the networks. On CIFAR100 trained on WR28 we set weight decays equal to $\{10, 7, 5, 5, 5\} \times 10^{-4}$ and label smoothing magnitudes equal to $\{0.1, 0.07, 0.05, 0, 0\}$ for WR28-$\{1, 2, 4, 8, 16\}$, respectively. All the hyperparameters were selected with a small grid search.

**A.3. ResNet50 on ImageNet**

We train all the ResNet50 with mixed precision [53] for 120 epochs with a weight decay of $4 \times 10^{-5}$ and label smoothing rate of 0.1 [54]. The input size is $224 \times 224$ and the training set is augmented with random crops and horizontal flips with 50% probability. The per-GPU batch size is set to 128 and is halved for the widest networks to fit in the GPU memory. The networks are trained on 8 or 16 Volta V100 GPUs in order to keep the batch size $B$ equal to 1024. The learning rate is increased linearly from 0 to $0.1 \cdot B/256$ [55] for the first five epochs and then annealed to zero with a cosine schedule.

---

**Table 1.** Test accuracy (average over four runs).

| Network        | CIFAR10 | CIFAR100 | ImageNet (top1) |
|----------------|---------|----------|-----------------|
| Wide-RN28-0.25 | 84.1    | 70.4     | 67.0            |
| Wide-RN28-0.5  | 90.3    | 75.7     | 74.1            |
| Wide-RN28-1    | 93.4    | 79.6     | 77.6            |
| Wide-RN28-2    | 95.2    | 80.8     | 79.1            |
| Wide-RN28-4    | 95.9    | 81.9     | 79.5            |
| Wide-RN28-8    | 96.1    |          |                 |
| DenseNet40-BC ($k = 8$) | 91.6 |          |                 |
| DenseNet40-BC ($k = 16$) | 93.9 |          |                 |
| DenseNet40-BC ($k = 32$) | 95.1 |          |                 |
| DenseNet40-BC ($k = 64$) | 95.7 |          |                 |
| DenseNet40-BC ($k = 128$) | 96.0 |          |                 |
| RN50-0.25      | 67.0    |          |                 |
| RN50-0.5       | 74.1    |          |                 |
| RN50-1         | 77.6    |          |                 |
| RN50-2         | 79.1    |          |                 |
| RN50-4         | 79.5    |          |                 |
Appendix B. Additional experiments

Figure S1. Training dynamics on CIFAR100. (a) As in figure 4, we show the mean correlation of the residuals of the linear reconstruction of the final representation of a Wide-ResNet28-8 from chunks, but this time as a function of training epochs. Small correlation indicates that the reconstruction error in going from chunks to the final representation can be modeled as independent noise. (b) Training error of chunks of a Wide-ResNet28-8 and its full layer representation. From epoch 150 to epoch 185 the training error of the chunks with size 128/256 decreases below 0.5%, while for smaller chunk sizes it remains above 5%. Random chunks with sizes larger than 128/256 can fit the training set, thus having the same representational power as the whole network on the training data. For $W > 128/256$ the test accuracy is decaying approximately with the same law as that of independent networks with the same width (see figure 3). This picture suggests that for CIFAR100 the size of a clone is 128/256, slightly larger than the size of the clones in CIFAR10. (c) Training and test error dynamics for the same Wide-ResNet28-8. After epoch 150, the training error of the full network remains consistently smaller than 0.1% (orange profile) while the test error continues to decrease until epoch 185 from 0.194 to 0.1765 (blue profile). In the same range of epochs (150–185) the training error of smaller chunks decreases sensibly (see panel (b)).
Redundant representations help generalization in wide neural networks

Figure S2. DenseNet40 architecture. (a) Decay of the test error of independent networks (blue) and chunks of the widest network (orange) to the error of an ensemble average of ten of the widest networks (DenseNet40-BC, \(k = 128\)). (b) Blue profile: \(R^2\) coefficient of the ridge regression of a chunk of \(w_c\) neurons (x-axis) to the full layer representation. Green profile: mean correlation of the residuals of the mapping as described in section 2.2. (c) Training error of various DenseNet40 of increasing width (blue) and of chunks of the widest architecture (orange). (d) Mean correlation of the residuals from the linear reconstruction of the final representation from chunks of a given size for a DenseNet40-BC (\(k = 128\)) during training. (e) Training error dynamics of chunks of a DenseNet40-BC (\(k = 128\)). (f) Training and test error dynamics for a DenseNet40-BC (\(k = 128\)).

Figure S3. DenseNet40 not regularized. DenseNet40-BC (\(k = 128\)) trained on CIFAR10 without weight decay. This experiment reproduces on a DenseNet the analysis shown on a Wide-ResNet28 in section 3. It shows that (a) also in a DenseNet architecture not well regularized error -error\(_\infty\) decays faster than \(w_c^{-1/2}\) and (b) the mean correlation of the residuals do not decrease during training. Thin profiles of panel (b) are the same as those shown in figure S2(d).
Redundant representations help generalization in wide neural networks

Figure S4. Impact of the width of the intermediate layers. We study how the scaling of the test error is affected (figure (a)) by increasing the width of the intermediate representations while keeping the width of the last layer constant or (figure (b)) by increasing the last layer width while keeping the width of the network constant. In (a) we trained DenseNet40 on CIFAR10 with an additional $1 \times 1$ convolution to keep the number of output channels fixed at 128. Figure (a) shows that increasing the width of intermediate layers makes the test accuracy of the full network decay approximately as $w^{-1/2}$, even when the width of the final representation is fixed. A bottleneck of 128 channels makes the clones much smaller. Orange profile shows that a strong deviation from the $w^{-1/2}$ can be seen for chunk sizes smaller than 16 (versus 350 figure 1b, main paper). We also verified that 16 random neurons are sufficient to interpolate the training set (error $< 5 \times 10^{-3}$) and that the $R^2$ coefficient of fit to the full layer is 0.912 (0.98 for chunk sizes = 32). The phenomenology described in the manuscript also applies when a bottleneck of 128 channels is added at the end of the network. In a second experiment, we trained a ResNet28-1 increasing only the number of channels in the last layer. We modified the number of output channels of the last block of conv4 and analyzed the representation after average pooling, as we did in the other experiments. The network was trained for 200 epochs using the same hyperparameters and protocol described in section 2. Figure (b) shows that the test error of the modified ResNet28-1 is approximately constant (blue profile). In contrast, when we increase the width of the whole network the test error decays to the asymptotic test error with an approximate scaling of $1/\sqrt{w}$ (orange profile).
Bias-variance profiles in CIFRA10 and CIFAR100. We compute the bias and the variance profiles for the convolutional architectures analyzed in the paper: Wide-ResNets and DenseNets trained on CIFAR10, and Wide-ResNets trained on CIFAR100. Since we trained the models using the cross-entropy loss, the standard bias-variance decomposition, which assumes the square loss, does not apply. Instead, we used the method recently proposed by Yang et al [56] to estimate the bias and the variance on networks trained with cross-entropy loss. Average over the data distribution is approximated by splitting the CIFAR training sets into five disjoint subsets containing 10,000 images each and training the networks from scratch on each of them. We use the same regularization for all the networks, namely that of the largest architectures, with weight decay equal to $5 \times 10^{-4}$ and label smoothing equal to 0. We repeat the procedure four times, for a total of 20 training runs for each network width, as described in [56]. We show the test loss curves as well as the squared bias and variance. As expected, the bias of the models decreases as we add parameters and make the model more flexible. The variance of the models initially grows with width to reach its peak at $W_{\text{peak}} = 32$ and 64 for CIFAR10 and CIFAR100 trained on Wide-ResNet28 (a), (b) and $W_{\text{peak}} = 88$ on CIFAR10 trained on DenseNet40 (c). As we increase the width, the variance decreases, allowing the model to generalize better and better, defying the classical bias-variance trade-off. The clone sizes $w_c^*$ for these architectures are slightly above the widths at which the variance peaks and are $w_c^* = 64$, and 128 for CIFAR10 and CIFAR100 trained on Wide-ResNet28 (compare figures 3 and 4) and $w_c^* = 170/250$ (figure S2). In all cases, the onset of the clones occurs at a width that is approximately twice as large as $W_{\text{peak}}$, similar to the width at which an architecture of size $w_c^*$ interpolates the training set.

References

[1] Zhang C, Bengio S, Hardt M, Recht B and Vinyals O 2017 Understanding deep learning requires rethinking generalization Int. Conf. on Learning Representations
[2] Arpit D, Jastrzebski S, Kanwal M S, Maharaj T, Fischer A, Courville A and Bengio Y 2017 A closer look at memorization in deep networks Proc. 34th Int. Conf. on Machine Learning
[3] Geman S, Bienenstock E and Doursat R 1992 Neural networks and the bias/variance dilemma Neural Comput. 4 1–58
[4] Neyshabur B, Tomioka R and Srebro N 2015 In search of the real inductive bias: on the role of implicit regularization in deep learning Int. Conf. on Learning Representations
[5] Spigler S, Geiger M, d’Ascoli S, Saquet L, Biroli G and Wyart M 2019 A jamming transition from under-to over-parametrization affects generalization in deep learning J. Phys. A: Math. Theor. 52 474001
[6] Nakkiran P, Kaplun G, Bansal Y, Yang T, Barak B and Sutskever I 2020 Deep double descent: where bigger models and more data hurt Int. Conf. on Learning Representations

https://doi.org/10.1088/1742-5468/aceb4f
Redundant representations help generalization in wide neural networks

[7] Gunasekar S, Lee J, Soudry D and Srebro N 2018 Characterizing implicit bias in terms of optimization geometry Int. Conf. on Machine Learning (PMLR) pp 1832–41

[8] Gunasekar S, Lee J D, Soudry D and Srebro N 2018 Implicit bias of gradient descent on linear convolutional networks Advances in Neural Information Processing Systems vol 31, ed S Bengio, H Wallach, H Larochelle, K Grauman, N Cesa-Bianchi and R Garnett (Curran Associates, Inc.)

[9] Soudry D, Hoffer E and Srebro N 2018 The implicit bias of gradient descent on separable data Int. Conf. on Learning Representations

[10] Ji Z and Telgarsky M 2019 The implicit bias of gradient descent on nonseparable data Conf. on Learning Theory (PMLR) pp 1772–98

[11] Arora S, Cohen N, Hu W and Luo Y 2019 Implicit regularization in deep matrix factorization Advances in Neural Information Processing Systems vol 32, ed H Wallach, H Larochelle, A Beygelzimer, F d’Alché-Buc, E Fox and R Garnett (Curran Associates, Inc.)

[12] Chizat L and Bach F 2020 Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss Conf. on Learning Theory (PMLR) pp 1305–38

[13] Advani M S, Saxe A M and Sompolinsky H 2020 High-dimensional dynamics of generalization error in neural networks Neural Netw. 132 428–46

[14] Neal B, Mittal S, Baratin A, Tantia V, Scieriuna M, Lacoste-Julien S and Mitliagkas I 2018 A modern take on the bias-variance tradeoff in neural networks (arXiv:1810.08591)

[15] Mei S and Montanari A 2022 The generalization error of random features regression: precise asymptotics and the double descent curve Commun. Pure Appl. Math. 75 667–706

[16] Belkin M, Hsu D, Ma S and Mandal S 2019 Reconciling modern machine-learning practice and the classical bias–variance trade-off Proc. Natl Acad. Sci. 116 15849–54

[17] Hastie T, Montanari A, Rosset S and Tibshirani R J 2022 Surprises in high-dimensional ridgeless least squares interpolation Ann. Stat. 50 949–86

[18] d’Ascoli S, Refinetti M, Birolì G and Krzakala F 2020 Double trouble in double descent: bias and variance(s) in the lazy regime Int. Conf. on Machine Learning

[19] Adlam B and Pennington J 2020 Understanding double descent requires a fine-grained bias-variance decomposition Advances in Neural Information Processing Systems vol 33, ed H Larochelle, M Ranzato, R Hadsell, M F Balcan and H Lin (Curran Associates, Inc.) pp 11022–32

[20] Lin L and Dobriban E 2021 What causes the test error? Going beyond bias-variance via ANOVA J. Mach. Learn. Res. 22 155–1

[21] Geiger M, Jacot A, Spigler S, Gabriel F, Saguin L, d’Ascoli S, Birolì G, Hongler C and Wyatt M 2020 Scaling description of generalization with number of parameters in deep learning J. Stat. Mech. 2023 114011

[22] Bartlett P L, Long P M, Lugosi G and Tsigler A 2020 Benign overfitting in linear regression Proc. Natl Acad. Sci. 117 30063–70

[23] Liu S, Papailiopoulos D and Achilleots D 2020 Bad global minima exist and SGD can reach them Advances in Neural Information Processing Systems vol 33, ed H Larochelle, M Ranzato, R Hadsell, M F Balcan and H Lin (Curran Associates, Inc.) pp 8543–52

[24] Huang G, Liu Z, Van Der Maaten L and Weinberger K Q 2017 Densely connected convolutional networks Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 4700–8

[25] Krizhevsky A et al 2009 Learning multiple layers of features from tiny images (available at: www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf)

[26] Zagoruyko S and Komodakis N 2016 Wide residual networks (arXiv:1605.07146)

[27] Deng J, Dong W, Socher R, Li L-J, Li K and Fei-Fei L 2009 ImageNet: a large-scale hierarchical image database 2009 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR09)

[28] Deng L 2012 The mnist database of handwritten digit images for machine learning research IEEE Signal Process. Mag. 29 141–2

[29] He K, Zhang X, Ren S and Sun J 2016 Deep residual learning for image recognition Proc. IEEE Conf. on Computer Vision and Pattern Recognition pp 770–8

[30] Hastie T, Tibshirani R and Friedman J 2001 The Elements of Statistical Learning (Springer Series in Statistics) (Springer)

[31] Bengio Y, Courville A and Vincent P 2013 Representation learning: a review and new perspectives IEEE Trans. Pattern Anal. Mach. Intell. 35 1798–828

[32] Ansuini A, Laio A, Macke J H and Zoccolan D 2019 Intrinsic dimension of data representations in deep neural networks Advances in Neural Information Processing Systems pp 6109–19

[33] Hestness J, Narang S, Ardalani N, Dianos G, Jun H, Kianinejad H, Patwary M, Ali M, Yang Y and Zhou Y 2017 Deep learning scaling is predictable, empirically (arXiv:1712.00409)
Redundant representations help generalization in wide neural networks

[34] Rosenfeld J S, Rosenfeld A, Belinkov Y and Shavit N 2020 A constructive prediction of the generalization error across scales Int. Conf. on Learning Representations

[35] Kaplan J, McCandlish S, Henighan T, Brown T B, Chess B, Child R, Gray S, Radford A, Wu J and Amodei D 2020 Scaling laws for neural language models (arXiv:2001.08361)

[36] Henighan T et al 2020 Scaling laws for autoregressive generative modeling (arXiv:2010.14701)

[37] Bahri Y, Dyer E, Kaplan J, Lee J and Sharma U 2021 Explaining neural scaling laws (arXiv:2102.06701)

[38] Jacot A, Gabriel F and Hongler C 2018 Neural tangent kernel: convergence and generalization in neural networks Advances in Neural Information Processing Systems vol 32 pp 8571–80

[39] Du S, Lee J, Tian Y, Singh A and Poczos B 2018 Gradient descent learns one-hidden-layer CNN: don’t be afraid of spurious local minima Proc. 35th Int. Conf. on Machine Learning vol 80 pp 1339–48

[40] Chizat L, Oyallon E and Bach F 2019 On lazy training in differentiable programming Advances in Neural Information Processing Systems pp 2937–47

[41] Geiger M, Spigler S, Jacot A and Wyart M 2020 Disentangling feature and lazy training in deep neural networks J. Stat. Mech. 113301

[42] Mei S, Montanari A and Nguyen P 2018 A mean field view of the landscape of two-layer neural networks Proc. Natl Acad. Sci. 115 E7665–71

[43] Rotskoff G M and Vanden-Eijnden E 2018 Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks Advances in Neural Information Processing Systems vol 31 pp 7146–55

[44] Sirignano J and Spiliopoulos K 2020 Mean field analysis of neural networks: a central limit theorem Stochastic Process. Appl. 130 1820–52

[45] Goyal P, Dollár P, Girshick R, Noordhuis P, Wesolowski L, Kyrola A, Tulloch A, Jia Y and He K 2017 Accurate, large minibatch SGD: training imagenet in 1 hour (arXiv:1706.02677)

[46] Yang Z, Yu Y, You C, Steinhardt J and Ma Y 2020 Rethinking bias-variance trade-off for generalization of neural networks Proc. 37th Int. Conf. on Machine Learning (Proc. Machine Learning Research vol 119) ed H Daumé III and A Singh (PMLR) pp 10767–77