Expand-and-Cluster: Parameter Recovery of Neural Networks

Flavio Martinelli [1] **Berfin Şimşek** [1 2] **Wulfram Gerstner** [* 1] **Johanni Brea** [* 1]

Abstract

Can we identify the weights of a neural network by probing its input-output mapping? At first glance, this problem seems to have many solutions because of permutation, overparameterisation and activation function symmetries. Yet, we show that the incoming weight vector of each neuron is identifiable up to sign or scaling, depending on the activation function. Our novel method 'Expand-and-Cluster' can identify layer sizes and weights of a target network for all commonly used activation functions. Expand-and-Cluster consists of two phases: (i) to relax the non-convex optimisation problem, we train multiple overparameterised student networks to best imitate the target function; (ii) to reverse engineer the target network's weights, we employ an ad-hoc clustering procedure that reveals the learnt weight vectors shared between students – these correspond to the target weight vectors. We demonstrate successful weights and size recovery of trained shallow and deep networks with less than 10% overhead in the layer size and describe an 'ease-of-identifiability' axis by analysing 150 synthetic problems of variable difficulty.

1. Introduction

It is known since the 1980s that finding a solution to the XOR problem with gradient descent is easier with a larger hidden layer, even though a minimal network with two hidden neurons is theoretically sufficient to solve the problem (Rumelhart et al., 1986). Indeed, even very small networks have a non-convex loss function (Fukumizu & Amari, 2000; Mei et al., 2018; Frei et al., 2020; Yehudai & Ohad, 2020; Şimşek et al., 2024). Recent advances in the theory of artificial neural networks indicate that the loss function is rough – i.e. predominantly populated by saddle points – for networks of minimal size (Şimşek et al., 2021), but becomes effectively convex in the limit of infinitely large hidden layers (Jacot et al., 2018; Chizat & Bach, 2018; Du et al., 2019; Rotskoff & Vanden-Eijnden, 2022). In a teacher-student setup, where teacher and student share the same architecture, the complexity of the loss landscape is linked to the ratio between the amount of permutation-induced critical points (zero gradients, non-zero loss) and the number of global minima at zero population loss (Şimşek et al., 2021). Importantly, as the width of the student increases, this ratio undergoes a substantial change: from much larger than one to very close to zero; suggesting that already for mild overparameterisation the amount of interconnected global minima largely dominates the amount of critical points (Cooper, 2018; Şimşek et al., 2021). This is consistent with our empirical observations: without overparameterisation, students trained to imitate the target (teacher) network get stuck in local high-loss minima. Instead, if we expand the student width to four times the width of the target network we can reliably reach near-zero loss (Fig. 4). At zero loss, target and student networks are functionally identical; but they are networks of different sizes and parameters. Here we ask the following question: "Is it possible to recover the weights and width of the target network from (near-zero) loss, overparameterised students?"

To answer this question, we first characterise the equivalence class of zero-loss overparameterised students – i.e. all possible weight structures that can preserve functional equivalence of a given hidden layer. Three types of symmetries arise at the neuron level: *(i) Permutation symmetries:* student neurons copying the weights of teacher neurons can be found in arbitrary order within the hidden layer; *(ii) Overparameterisation symmetries:* redundant neuron groups can duplicate input weights to imitate a teacher neuron or cancel each others' contribution. *(iii) Activation function symmetries:* even, odd or scaling symmetries in the activation function give rise to combinations of neurons that degenerate into constant or linear outputs, weight vectors of opposite signs with respect to teacher neurons or scaling factors transferred to adjacent layers; and combinations thereof.

[*]Equal contribution [1]Department of Life Sciences and Computer Sciences, EPFL, Lausanne, Switzerland. [2]Center for Data Science, NYU, New York, United States. Correspondence to: Flavio Martinelli <flavio.martinelli@epfl.ch>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

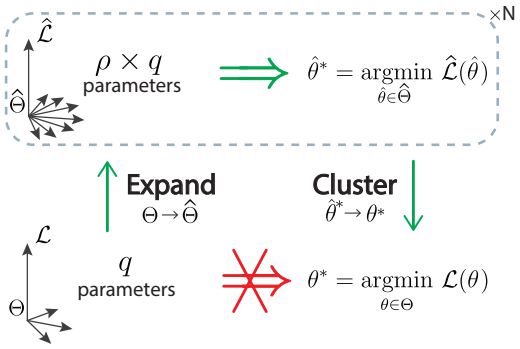


Figure 1. **Expand-and-Cluster**: we overcome the non-convex problem of recovering the q parameters of an unknown network by: **(i)** expanding the dimensionality of the parameter space Θ by a factor ρ to relax the optimisation problem, $\Theta \to \hat{\Theta}$; **(ii)** mapping the loss minimiser in expanded space $\hat{\theta}^*$ to the original parameter space via clustering of N overparameterised solutions, $\hat{\theta}^* \to \theta^*$.

As shown by Şimşek et al. (2021) in a restricted setting, at zero loss each teacher neuron is duplicated by the student at least once. Importantly, we show that each teacher neuron input weight vector is preserved in zero-loss students up to a sign and/or scaling factor. Even on smoother overparameterised landscapes, exact zero loss is not numerically achievable in practice. Therefore, we developed a clustering procedure to separate neurons that are consistently found across N different student networks from redundant units that do not persist across networks. The former are the ones copying the teacher's hidden neuron weights.

In summary, we tackle the highly non-convex parameter identification problem with a technique that resembles a popular strategy in the field of applied mathematics (Lovász & Schrijver, 1991): we **expand**, or lift, the optimisation space to higher dimensions, find a solution through standard optimisation techniques, and **cluster** to project the solution back to the original parameter space (Fig. 1). Our contributions can be summarised as follows:

- A small, constant overparameterisation factor is enough to solve the non-convex problem due to the combinatorial proliferation of global minima, in agreement with Şimşek et al. (2021);
- Building upon previous works (Petzka et al., 2020; Şimşek et al., 2021), we provide a complete formulation of overparameterisation and activation function symmetries. We discover a covert symmetry for activations functions composed of a linear and an even term (e.g. ReLU, GELU, softplus, SiLU);
- We introduce Expand-and-Cluster, the first parameter recovery method suitable for any activation function to solve the identification problem in shallow, deep feed-forward and convolutional layers of unknown widths.

2. Motivation

1. Understanding loss landscapes: In teacher-student setups, the ratio of global minima to other zero-gradient points is speculated to be an indicator of 'trainability' of a given student – i.e. to positively correlate to the probability of a training procedure to converge to a global minimum (Şimşek et al., 2021). However, this quantification gives only a *static* view of the trainability question, ignoring any effects that overparameterisation might induce to the *dynamics* of training. We empirically validate these speculations by showing that overparameterisation is a good indicator of student trainability for a rich family of differently shaped teachers (Fig. 4).

2. Neuroscience: Retrieving the connectivity of neural circuits is a daunting task. Neuroscientists either infer the connectivity with massive connectomic imaging endeavours (Shapson-Coe et al., 2024), or try to estimate it from partial recordings of unit activities (Haspel et al., 2023). Although the second approach has many advantages, it is unclear how accurately one can reverse engineer the connectivity from unit activation recordings. To start simple, we tackle this question in a deep-learning setting. In contrast to Rolnick & Kording (2020), we provide a learning-based solution adaptable to any activation function that relies on natural stimulation protocols, for example, the data the teacher is trained on. We see this work as a fundamental step towards reverse engineering brain circuits.

3. Model stealing attacks: Testing the extent to which it is possible to steal information from deployed models has important practical consequences (Carlini et al., 2024). For example, once all parameters of a deployed network are identified, stronger and malicious adversarial attacks (requiring gradient or architecture information) can be performed on a deployed service (Chakraborty et al., 2018), posing security threats and ethical issues. Given the small scale of reconstructed models in the field, this is not yet an issue of concern.

3. Related Work

1. Fundamental distinction with pruning and distillation: Despite the apparent similarities, our model identification problem differs substantially from the classic pruning setting (Hoefler et al., 2021). In pruning, one trades the number of parameters with a tolerated increase in test loss. In distillation settings, the focus is also on generalisation performance (Hinton et al., 2015). Moreover, students are often smaller and of a different architecture than the teacher (Beyer et al., 2022). Even in the case of self-distillation (Furlanello et al., 2018), the student is not expected to be functionally equivalent to the teacher (Stanton et al., 2021). In our setting, any increase in imitation loss or change in architecture is unacceptable, as it would result in reconstructing a poten-

tially good approximation of the target network but not its parameterisation. Loss-unaware pruning schemes such as magnitude pruning (LeCun et al., 1989; Han et al., 2015; Frankle & Carbin, 2019) or structural pruning based on weight properties (Srinivas & Babu, 2015; Mussay et al., 2021) have no means of preserving the target parameters within the student. While loss-aware structural pruning techniques (Hu et al., 2016; Chen et al., 2022) could have more control in trading pruning and loss increase, they come with no clear guarantee to remove all the non-trivial redundant structures described by the symmetries and, at the same time, preserve the target network parameters. Given the lack of alternative methods for non-relu activations, pruning constitutes one baseline for comparison in our experiments.

2. Non-convex optimisation and loss landscapes: Many non-convex optimisation problems are tackled with the following strategy: (i) expand, or *lift*, to a higher dimensional space to relax the problem and guarantee convergence to global minima; (ii) map, or *project*, the relaxed solution to the original space by exploiting the problem's intrinsic symmetries and geometry (Zhang et al., 2020). This approach is used in applied mathematics (Lovász & Schrijver, 1991; Lasserre, 2001), machine learning (Janzamin et al., 2015; Zhang et al., 2020) and many others (Sun, 2021). Despite the above-mentioned achievements, for neural networks, the picture is far from complete. Unlike infinitely-wide neural networks (Jacot et al., 2018), the loss functions of finite-width networks exhibit several non-convexities causing the gradient flow (from different initialisations) to converge to local minima non-identical to one another (Safran & Shamir, 2018; Arjevani & Field, 2021; Abbe et al., 2022). Yet, overparameterised solutions found by different initializations are similar in function space (Allen-Zhu & Li, 2020), they can be *approximately* mapped to each other by permutation of hidden neurons and exhibit the linear mode connectivity phenomenon (Singh & Jaggi, 2020; Wang et al., 2020; Entezari et al., 2022; Ainsworth et al., 2022; Jordan et al., 2022). Even though the mildly overparameterised regime is non-convex, we show that we can exploit its reduced complexity to find zero-loss solutions in our identification setup; and that any zero-loss student can be *exactly* mapped to every other zero-loss student.

3. Interpretability: Explaining in qualitative terms the behaviour of single neurons embedded in deep networks is a challenging task (Olah et al., 2018; Zhang et al., 2021). For example, in symbolic regression, small networks with vanishing training loss are desirable for interpretability (Udrescu & Tegmark, 2020; Liu et al., 2024). Complementary to the notion of 'superimposed-features' neurons found in underparameterised students (Elhage et al., 2022; Şimşek et al., 2024), we provide explanations about the role of each hidden neuron found in zero-loss overparameterised students relative to a teacher network of minimal size.

4. Functionally Equivalent Model Extraction: Our work focuses on functionally equivalent extractions, that is retrieving a model $\mathcal{M}$ such that $\forall x \in X, \mathcal{M}(x) = \mathcal{M}^*(x)$, where $\mathcal{M}^*(x)$ is the target model. This type of extraction is the hardest achievable goal in the field of Model Stealing Attacks (Oliynyk et al., 2022), using only input-output pairs (Jagielski et al., 2020). Out of all the functionally equivalent models we aim to extract the one of *minimal size*. Conditions for neural network identifiability and their symmetries have been studied theoretically for different activation functions (Sussmann, 1992; Fefferman, 1994; Zhong et al., 2017; Bui Thi Mai & Lampert, 2020; Petzka et al., 2020; Vlačić & Bölcskei, 2021; 2022; Bona-Pellissier et al., 2021; Stock & Gribonval, 2022), although overparameterised solutions were considered only in Tian et al. (2019) and Şimşek et al. (2021). Existing functionally equivalent extractions of trained networks rely on identifying boundaries between linear regions of shallow ReLU networks (Baum, 1991; Jagielski et al., 2020) and single output deep ReLU networks (Rolnick & Kording, 2020; Carlini et al., 2020). Janzamin et al. (2015) show a theoretical reconstruction based on third-order derivatives. Fornasier et al. (2022), building upon (Fornasier et al., 2019; Fu et al., 2020; Fornasier et al., 2021; Fiedler et al., 2023), propose an identification method for wide committee machines (shallow networks with unit norm second layer weights), where knowledge of the teacher layer size is necessary. Anachronistically, Tramèr et al. (2016) learn small, shallow teacher networks (20 hidden nodes) with overparameterised students but do not claim any parameter recovery. Shallow networks with *polynomial* activation functions can be globally optimised under some guarantees by lifting the optimisation problem to tensor decomposition (Janzamin et al., 2015; Mondelli & Montanari, 2019). We are the first to propose a recovery method for arbitrary activation functions on shallow and deep fully connected networks of unknown layer widths. Our method is learning-based and fundamentally different from the known approaches. However, none of these methods were shown to work in large-scale applications.

4. Symmetries of the identification problem

To characterise the symmetries of the identification problem we start by reviewing the results of Şimşek et al. (2021), which are for neurons with no biases and asymmetric activation function. We will later extend this formulation to the practical deep learning setting and describe a convert symmetry that arises when the activation function is a combination of a linear and an even function. For teacher-student setups, we call a student 'overparameterised' if it has more hidden neurons than the teacher in at least one layer. If an overparameterised student network replicates the teacher mapping with zero loss, the space of all possible solutions is fully described by the geometry of the global

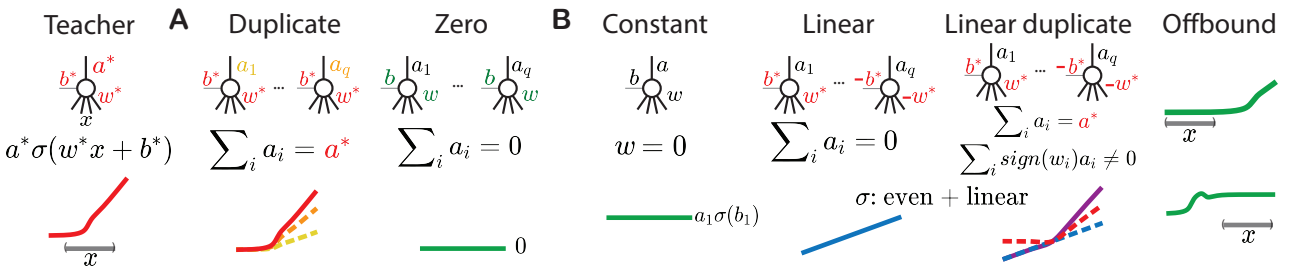
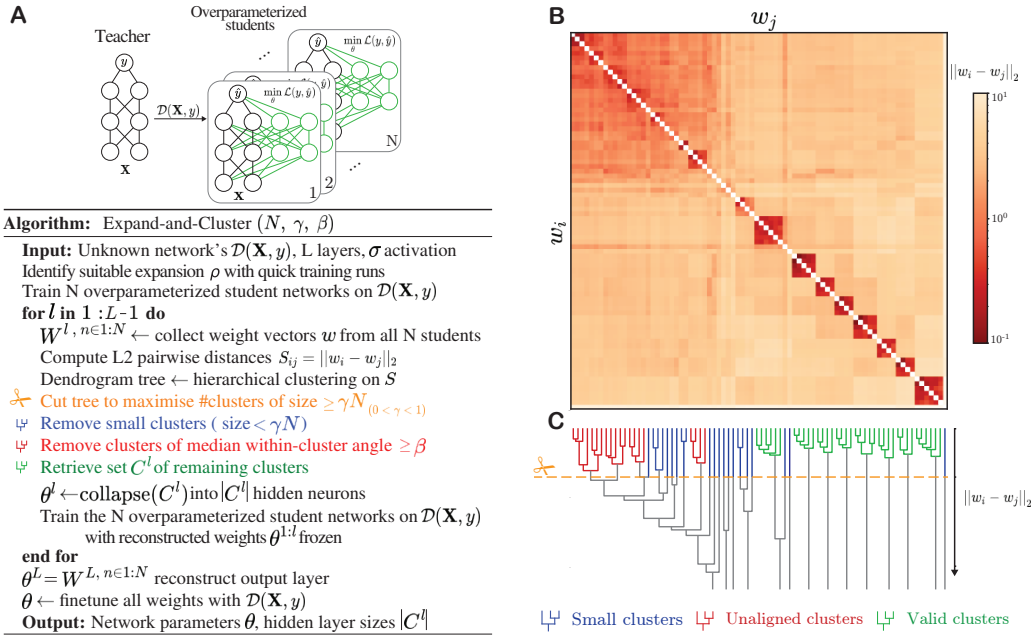


Figure 2. **Catalogue of student neuron types at zero loss:** zero loss overparameterised students can only contain a few neuron types. Left: a teacher neuron is defined along with a sketch of its output, the grey bar indicates the finite input support x to the neuron; the same colour-coded letters indicate equal quantities. **A) Neuron types from Şimşek et al. (2021) adapted with biases:** *duplicate-type* neurons combine to replicate a teacher neuron by copying its weight vector w^* and bias b^*, their activations a_i sum up to the teacher activation a^*. *Zero types* have aligned weight vectors and biases but cancel each other via output weights. **B) Novel neuron types:** *constant types* contribute a fixed amount to the next layer by learning a null vector. For even + linear activation functions, *linear type* groups combine to contribute a linear function. *Linear duplicate type* groups copy the teacher vector and its opposite, replicating the teacher neuron up to a linear mismatch. *Offbound types:* at non-exact zero loss, their input support is placed in the linear or zero region of σ.

minima manifold (Şimşek et al., 2021). The global minima manifold contains only two types of hidden units, namely duplicate and zero-type neurons (see theorem 4.2 of Şimşek et al. (2021) and Fig. 2A); under the following assumptions: one-hidden layer network $\sum_{i=1}^{m} a_i\sigma(w_ix)$, infinite input data support, population loss limit, no bias and analytical activation function σ with infinite non-zero even *and* odd derivatives evaluated at zero. The last assumption guarantees that the activation function has no symmetries around zero. The intuition for the result of Şimşek et al. (2021) is that, for zero-loss solutions, each teacher hidden neuron $a^*\sigma(w^*x)$ must be copied in the student by a *duplicate-type* group of one or more units, contributing $\sum_i a_i\sigma(w_ix)$. The duplicates' input weight vectors are all aligned with the teacher neuron, $w_i = w^*\ \forall i$, while their weighted contribution equals the teacher neuron's output weight $\sum_i a_i = a^*$. In the same student there can also exist *zero-type* neuron groups with a null contribution to the student input-output mapping, characterised by $w_1 = \cdots = w_q$, $\sum_i a_i = 0$. These neuron types are summarized in Figure 2A.

We extend the categorization of neuron types of Şimşek et al. (2021) to neurons with *bias*, *finite* input data support and activation functions that can be decomposed into *even* or *odd* functions plus a constant or a linear component. This generalization includes commonly used activation functions such as ReLU, GELU, SiLU, sigmoid, tanh, softplus, and others (see Appendix A.2 for details). The catalogue of new neuron types is sketched in Figure 2B. In the case of a teacher neuron with bias $a^*\sigma(w^*x + b^*)$, a group of *duplicate-type* $\sum_i a_i\sigma(w_ix + b_i)$ has input weight vectors and biases aligned to the teacher: $w_i = w^*$, $b_i = b^*\ \forall i$; $\sum_i a_i = a^*$. The new *zero-type* group has aligned, but arbitrary, weights and biases $w_1 = \cdots = w_q$, $b_1 = \cdots = b_q, \sum_i a_i = 0$. With biases, *constant-type* student neurons can also arise: they have vanishing input weights $w = 0$ and contribute a constant amount of $a\sigma(b)$ to the next layer; to keep an exact mapping of the teacher (zero loss), this constant contribution must be cancelled out in the next-layer biases or with another constant-type neuron.

Even + linear activation function: when the activation can be decomposed into an even and a linear function $\sigma(z) = \sigma_{lin}(z) + \sigma_{even}(z)$, for the sake of exposition we let $\sigma_{lin}(z) = z$, three phenomena arise: (i) Neurons can combine to contribute a linear + even function in non-trivial ways: $a_1\sigma(wx + b) + a_2\sigma(-wx - b) = (a_1-a_2)(wx+b)+(a_1+a_2)\sigma_{even}(wx+b)$. This allows two or more student neurons to combine in groups of positively aligned $(+wx+b)$ and negatively aligned $(-wx-b)$ neurons. If in such a group $\sum_i a_i = 0$, the neurons cancel the even component of the function and contribute a linear function: *linear-type* group. (ii) Linear-type groups can effectively 'flip' the sign of the input weight vector and bias of another neuron in the layer: $\sigma(wx+b)-2(wx+b) = \sigma(-wx-b)$. A linear-type group can therefore correct the contribution of a neuron that learns the opposite teacher vector, or, more generally, correct the linear component of a group of positively and negatively aligned duplicate neurons; the latter named *linear duplicate-type* group (Fig. 2B). (iii) These linear-type groups can ultimately be combined into an affine operation: $\Theta x + \beta = \sum_k^K [(a_k^* \times w_k^*)x + a_k^* b_k^*]$, where k is the index of duplicate neurons of opposite weight vector and bias, $\times$ is the outer vector product, $\Theta \in \mathbb{R}^{d_{out}\times d_{in}}$, $\beta \in \mathbb{R}^{d_{out}}$. Notably: $\mathrm{rank}(\Theta) = min(K, d_{in}, d_{out})$. See Appendix A.2.1 for a more detailed explanation. The even + linear symmetry is found in ReLU, LeakyReLU, GELU, Softplus, SiLU/Swish and other commonly used activation functions. A concrete numerical example of different neurons found by a softplus student is shown in Figure A.5. **Odd (+ constant) activation function**: when $\sigma(x) = c + \sigma_{odd}(x)$, student neurons can duplicate the teacher neuron with flipped signs:



Algorithm: Expand-and-Cluster $(N,\ \gamma,\ \beta)$

Input: Unknown network's $\mathcal{D}(\mathbf{X}, y)$, L layers, σ activation
Identify suitable expansion ρ with quick training runs
Train N overparameterized student networks on $\mathcal{D}(\mathbf{X}, y)$
for l **in** $1:L\text{-}1$ **do**
  $W^{l,\ n\in 1:N} \leftarrow$ collect weight vectors w from all N students
  Compute L2 pairwise distances $S_{ij} = ||w_i - w_j||_2$
  Dendrogram tree $\leftarrow$ hierarchical clustering on S
  Cut tree to maximise #clusters of size $\geq \gamma N$ $_{(0<\gamma<1)}$
  Remove small clusters (size $< \gamma N$)
  Remove clusters of median within-cluster angle $\geq \beta$
  Retrieve set C^l of remaining clusters
  $\theta^l \leftarrow$ collapse(C^l) into $|C^l|$ hidden neurons
  Train the N overparameterized student networks on $\mathcal{D}(\mathbf{X}, y)$ with reconstructed weights $\theta^{1:l}$ frozen
end for
$\theta^L = W^{L,\ n\in 1:N}$ reconstruct output layer
$\theta \leftarrow$ finetune all weights with $\mathcal{D}(\mathbf{X}, y)$
Output: Network parameters θ, hidden layer sizes $|C^l|$

Figure 3. **Parameter identification with Expand-and-Cluster. A) Training scheme:** once an overparameterisation factor yields near-zero training losses, train N overparameterised students on the teacher-generated dataset $\mathcal{D}(\mathbf{X}, y)$; **B) Similarity matrix**: L2-distance between hidden neurons' input weight vectors of layer l for all N students. Large-sized clusters are good candidate weight vectors. **C) Dendrogram obtained with hierarchical clustering:** the selected linkage threshold is shown in orange. Clusters are eliminated if too small (blue) or unaligned (red), the remaining clusters are shown in green. The code is available at https://github.com/flavio-martinelli/expand-and-cluster.

$a^*\sigma(w^*x+b) = -a^*\sigma(-w^*x-b)+c$. This is the case for tanh and sigmoid. **Positive scaling**: for piece-wise linear activation functions such as ReLU and LeakyReLU, a positive scalar can be transferred between input and output weight vectors: $a^*\sigma(w^*x+b^*) = ca^*\sigma(\frac{1}{c}(w^*x+b^*))$, where $c > 0$. Finally, in near zero-loss solutions, there can be *offbound-type* neurons: their hyperplane, spanned by $wx+b=0$, is placed outside of the input domain of the neuron. This results in the activation function being used in its asymptotic part (often constant or linear). The symmetries just described define the functional equivalence class between zero-loss overparameterised students. We show empirically, that the redundancy given by overparameterisation symmetries facilitates gradient descent in converging to a minimal loss. Given activation function symmetries, ReLU networks have the most amount (see A.2.3). Whether or not this fact plays a role in the ability to avoid the non-convexities of training is a matter of future research. Most importantly, the only way an overparameterised student can reach zero loss is by representing all the neurons at least once (Şimşek et al., 2021). By getting rid of all the redundant terms (zero, constant, linear and duplicate groups), teacher neurons can be identified up to permutation within the layer, a sign and/or a scaling factor. A numerical example of mapping a zero-loss overparameterised student into the teacher is shown in Figure A.6.

5. Expand-and-Cluster algorithm

If a student can be trained to exact zero loss, which is possible in small setups, it is almost trivial to identify the different neuron types, including the teacher neurons (see two exact numerical examples in figs. A.5, A.6). However, in larger setups, overparameterised networks are difficult to train to exact zero loss because of computational budgets and limited machine precision. Therefore we need ways to identify the teacher neurons from imperfectly trained students. These students present approximate duplicates of teacher neurons as well as neurons of other types that point in arbitrary directions. In a group of N imperfectly trained students, neurons that approximate the teacher can be found consistently across students, while redundant units have arbitrarily different parameterisations. Therefore we propose the following procedure (Fig. 3):

Step 1: Expansion phase. Find the correct overparameterisation by rapidly training a sequence of networks with increasing sizes of hidden layers to a fixed convergence criterion. To do so, use teacher-generated input-output pairs (teacher queries), $\mathcal{D} = \{\mathbf{X}, y\}$, and minimize the mean square error loss between un-normalized outputs (e.g. before the softmax operation) of teacher and student networks. This allows finding a network width m at which

convergence to nearly zero loss is possible (Fig. A.2B).

Step 2: Training phase. Train N students of L layers, width m, on teacher queries $\mathcal{D} = \{\mathbf{X}, y\}$ to minimize the loss to the lowest value achievable with classic optimiser techniques (Fig. 3A).

Step 3: Clustering phase. Collect the first unreconstructed hidden layer neurons of the N students, then cluster the input weight vectors with hierarchical clustering on the L2 distance. In the case of even or odd symmetries, we want to avoid finding different clusters for neurons of merely opposite signs. Therefore we arbitrarily align all input weight vectors to have a positive n^{th} element. This operation, which we call 'canonicalization', maintains the functional equivalence, since there are ways to compensate for the sign flip for both odd and even symmetries (see section above). In the case of positive scaling symmetry, the clustering is performed on the cosine distance. With a threshold selection criteria that maximizes the number of large clusters (size $\geq \gamma N$, $0 < \gamma \leq 1$), we obtain groups of aligned weight vectors; these clusters should include all duplicate teacher neurons. Proceed to filter out clusters whose elements are not aligned in angle (median alignment $\geq \beta$), removing eventual zero or constant type neuron clusters. Then, merge each remaining cluster of duplicate neurons into single hidden neurons by choosing the element in the cluster belonging to the lowest loss student. Due to the unidentifiability of weight vector signs for even+linear symmetries, we can recover the network up to an affine transformation $\Theta x + \beta$. Therefore, to preserve the functional equivalence of the network, we solve the linear regression problem to find the affine transformation that minimises the error between the overparameterised students and the network with the newly reconstructed layer. We noticed that higher layers align with the teacher weights only at prohibitively low losses (Tian, 2020). Therefore, if the student networks have more than one hidden layer left to reconstruct, we go back to Step 2 and train again N overparameterised students of $L \leftarrow L - 1$ layers, using as input $\mathbf{X}$ the output of the last reconstructed layer. Repeat this procedure until the last hidden layer is reconstructed (Fig. 3 Algorithm 1, more details in Appendix A.6.2).

Step 4: Fine-tuning phase. Adjust all the reconstructed network parameters using the training data.

6. Results

6.1. Synthetic teacher experiments

To test the effects of overparameterisation on the traversability of landscapes, we devised a series of very challenging regression tasks inspired by the parity-bit problem (or multidimensional XOR), known to be a difficult problem for neural networks (Rumelhart et al., 1986). We construct synthetic one-hidden layer teacher networks of varying input dimension d_{in}, and hidden neuron number r. Our construction yields XOR-like and checkerboard-like functions where teacher neurons' hyperplanes are often parallel to each other and divide the input space into separate regions (Fig. 4A), see Appendix A.4 for more details. In contrast to our approach, constructing shallow networks with randomly drawn input weight vectors yields easy tasks since all weights tend to be orthogonal (Saad & Solla, 1995; Goldt et al., 2019; Raman et al., 2019). For this experiment we use the asymmetric activation function $\mathrm{g}(x) = \mathrm{softplus}(x) + \mathrm{sigmoid}(4x)$.

We trained overparameterised students on the family of teachers described above (Figs. 4 and A.17). For an overparameterisation factor ρ, the student hidden layer has $m = \rho r$ neurons. To not venture away from the theoretical setting of near-zero loss, we trained all the networks with the ODE solver MLPGradientFlow.jl (Brea et al., 2023). This allowed us to find global and local minima with machine precision accuracy for networks without overparameterisation (Fig. 4B, $\rho = 1$). However, even with optimised solvers and slightly larger networks, it becomes challenging to converge fully to global minima within a reasonable amount of time (Fig. 4C, $\rho \in \{2, 4, 8\}$). Hence, methods to deal with imperfectly trained students are needed. Since training is full-batch (30k data points), the only source of randomness is in the initialization; see Appendix A.6 for more details. Figure 4C shows a beneficial trend as overparameterisation increases, but also highlights a strong dependence on the dataset (or teacher) complexity r/d_{in}: as the number of hyperplanes per input dimension increases, it becomes harder for students to arrange into a global minimum configuration.

We find that direct training of 20 student networks without overparameterisation (teacher with $r = 4$ hidden neurons and input dimensionality $d_{in} = 4$) does not yield a single case of convergence to zero loss (Fig. 5A, $\rho = 1$, hidden layer size 4). For the same teacher, Expand-and-Cluster can reconstruct the network up to machine-error zero loss and correct hidden-layer size if an overparameterisation of $\rho \geq 2$ is used in Step 2 of the algorithm (Fig. 5A, star-shaped datapoints). This suggests that successful retrieval of all parameters of the teacher is possible (Fig. 5A). We test the quality of parameter identification with Expand-and-Cluster for each teacher network of Figure 4 and illustrate the final loss of the reconstructed networks in Figure 5B. For example, for $\rho = 4, 8$ and of the 30 teachers with input dimensionality $d_{in} = 8$, all except 2 networks were correctly identified as indicated by a zero-loss solution (RMSE $\leq 10^{-14}$, dark blue in Fig. 5B). Of 150 different teachers, 118 ($\sim 80\%$) were correctly identified with $\rho = 4$. In all but 7 out of 118 successful recoveries, the number of neurons found matches that of the teacher; the other cases have at most up to 4 neurons in excess (these can be easily categorised into zero-type and constant-type neurons, see e.g. Fig. A.5). One can trade reconstruction loss vs. excess neurons by tuning

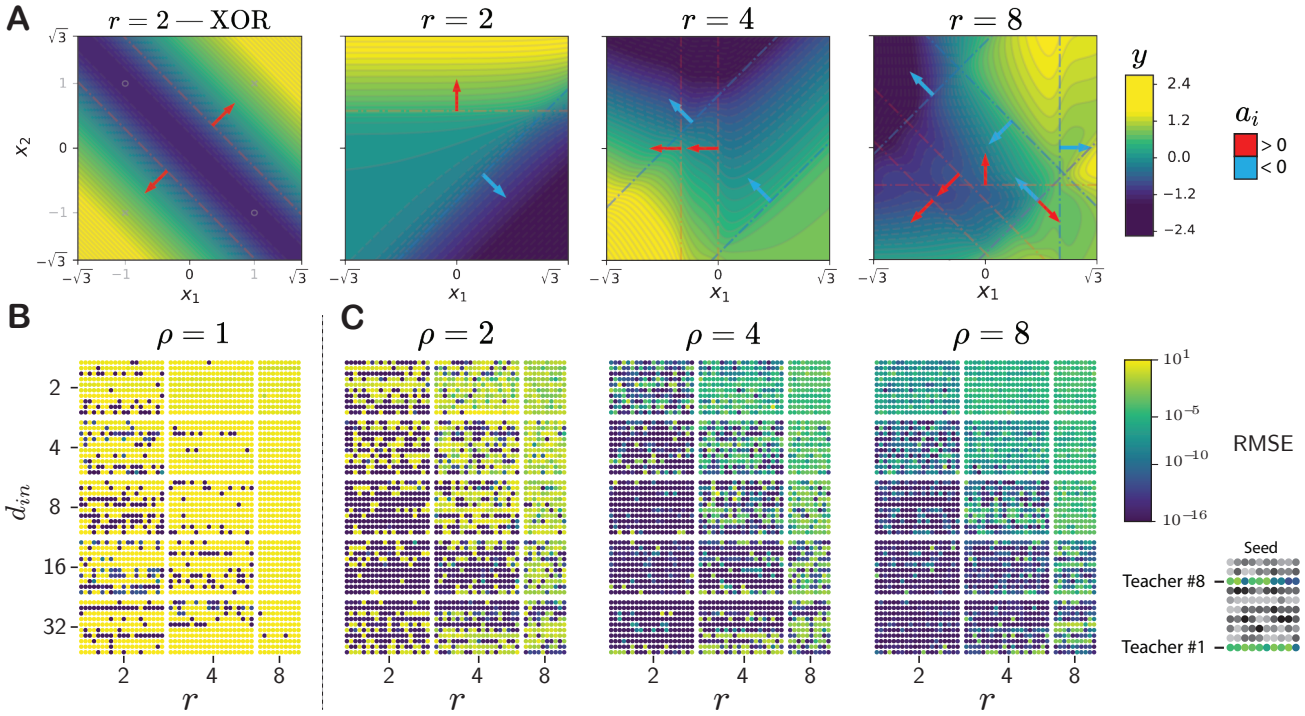


Figure 4. **Synthetic teachers define tasks of variable difficulty. A) For fixed $\mathbf{d_{in}}$, teacher complexity increases with number r of hidden neurons:** contourplot of the teacher network output. Each hidden neuron generates a hyperplane, ${w_i}^T x + b_i = 0$ (dashed lines); the direction of the weight vector w_i is indicated by an arrow starting from the hyperplane and the sign of the output weight a_i by its colour. Top left: generalization of the XOR or parity-bit problem to a regression setting. From left to right: As the number of teacher hidden neurons r increases the contour lines become more intricate. **B) Non-convexity prevents training to zero loss:** for each combination of $d_{in} = 2, 4, 8, 16, 32$ and $r = 2, 4, 8$ we generated 10 teachers; for each teacher, we trained 20 or 10 students (for $r = 8$) with different seeds. Each teacher corresponds to one row of dots while each dot corresponds to one seed (see inset bottom right). Dark blue dots indicate loss below 10^{-14}. Student networks of the same size as the teacher ($\rho = 1$) get often stuck in local minima. The effect is stronger for larger ratios r/d_{in}. **C) Effects of overparameterisation on convergence:** student networks with overparameterisation $\rho \geq 2$ are more likely to converge to near-zero loss than those without. We report the following general trends: (i) overparameterisation avoids high loss local minima, (ii) the dataset complexity, i.e. number of hidden neurons per input dimension r/d_{in}, determines the amount of overparameterisation needed for reliable convergence to near-zero loss. For difficult teachers, i.e. overcomplete ($r/d_{in} \geq 1$), training is very slow and convergence is not guaranteed in a reasonable amount of time (see Fig. A.3).

the β and γ parameters of Expand-and-Cluster (not shown). We conclude that, given a reasonably low loss, parameter identification is possible. Overparameterisation plays a key role in enabling gradient descent to find a global minimum, but this effect weakens as the ratio r/d_{in} increases. We note that given more training time, even the hardest teachers could be learnt with overparameterisation $\rho = 8$ (Fig. A.3).

6.2. Weight identification of trained networks

To show how the procedure scales to bigger applications, we recover parameters of networks trained on the MNIST (LeCun, 1998), FashionMNIST (Xiao et al., 2017) and CIFAR10 (Krizhevsky et al., 2009) datasets. We pre-trained different one-hidden layer teachers composed of r hidden neurons and different activation functions σ. For parameter recovery, we must have access to the classifier's probabilistic output (e.g. the values before or after the softmax) and not only the most probable class (e.g. values after an argmax). We then used input-output pairs generated by the last layer of the teacher to define a regression task for students of overparameterisation $\rho = 4$. After applying Expand-and-Cluster we obtain reconstructed networks of hidden layer size $\hat{m}$. We compare the reconstructed networks with teachers in Table 1, and show low RMSE loss (mismatch between teacher logits and reconstructed network's logits is on the order of 10^{-4}), close to perfect size recovery $\hat{m}/r \approx 1$ and low average cosine-distance $< d >$ between each teacher and reconstructed network neuron input-weight vectors. More metrics are shown in Table A.3. Note that the difficulty of the training dataset the teacher is trained on is not crucial for the identification process, as the student networks are trained on teacher-generated labels. The slight reduction in average cosine alignment $< d >$ for

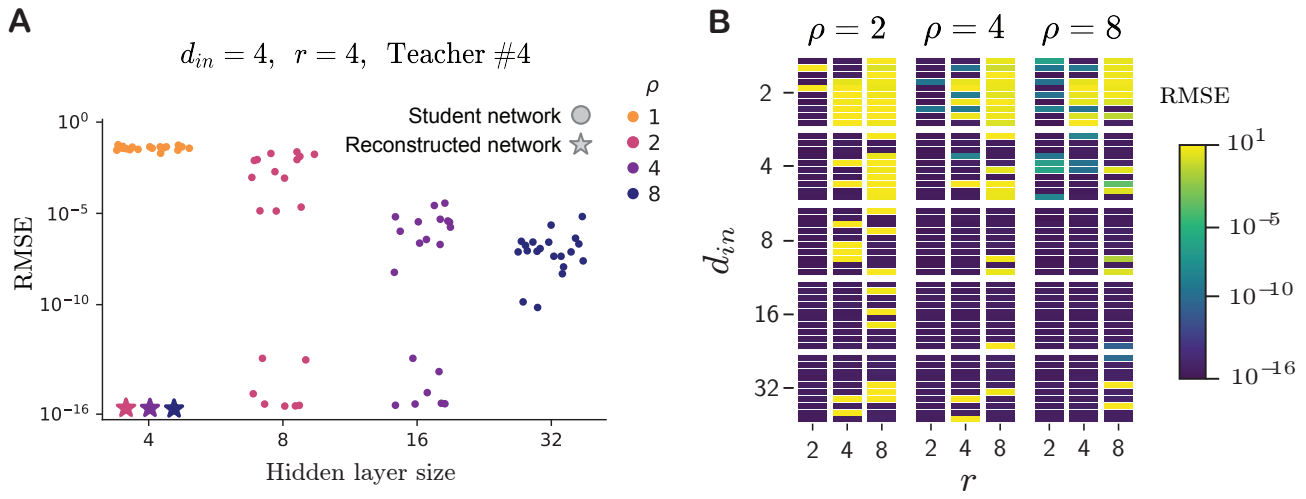


Figure 5. **A) Expand-and-Cluster applied to mildly overparameterised students reaches zero loss:** a total of 80 student networks with 4, 8, 16 or 32 hidden neurons have been trained using data generated by a teacher with $r = 4$ hidden neurons and $d_{in} = 4$ input dimensions. None of the 20 students with 4 hidden neurons reached zero loss (orange dots, $\rho = 1$), while all overparameterised student networks have zero loss with 4 hidden neurons after reconstruction (large coloured stars). **B) Loss after Expand-and-Cluster for all teacher networks and student sizes from Figure 4:** the colour of each small horizontal bar represents the final loss. Only a small fraction of teacher networks (i.e., those in yellow) were not identified correctly.

CIFAR10 teachers is likely due to the higher dimensionality of the input images (3 channels instead of 1). Experiments in Table 1 show successful identification when the input queries to the teacher are done with the same dataset the teacher was trained on. Depending on the application, one may not have access to the teacher's training data. We show in Table 2 that even if the teacher is trained on MNIST, the student can be trained on teacher queries from FashionMNIST and still achieve on par reconstruction performance. We speculate that not any type of input query will be suitable to collect informative data-points from the teacher. Further research is needed to understand the quantity and quality of queries needed for successful identification.

Expand-and-Cluster can identify deep fully connected networks trained on synthetic data (see Appendix A.5) and larger networks trained on MNIST, Figure 6A (3 hidden layers of 30 neurons each, $\rho = 3$, $\sigma = \mathrm{g}$). The reconstruction identifies layer sizes up to 4 neurons in excess for the last hidden layer and achieves a loss of 3 orders of magnitude lower than similar-sized networks trained from random initialization. Expand-and-Cluster can be applied "as is" also to convolutional layers. Without loss of generality, one can treat each convolution channel as a hidden neuron of an MLP and the same symmetries described in Section 4 apply. We show good preliminary reconstruction results for convolutional teacher layers in the appendix section A.7.

Given that our methodology for identification involves a compression step, we consider if currently available pruning methods are able to shrink overparameterised students back

Table 1. **Identification of shallow networks:** successful reconstruction to low RMSE $\mathcal{L}$, similar size $\hat{m}/r \approx 1$ and low average cosine distance d between identified and target neurons. All teachers are trained and queried with the same dataset, indicated in the leftmost column.

	σ	r	$\mathcal{L}$	$\hat{m}/r$	$<d(w_i, w_i^*)>$
MNIST	g	256	$1.5 \cdot 10^{-3}$	1.008	$2.1 \cdot 10^{-4}$
	sigmoid	256	$8.4 \cdot 10^{-4}$	1.08	$3.8 \cdot 10^{-4}$
	tanh	128	$1.7 \cdot 10^{-3}$	1.04	$1.3 \cdot 10^{-4}$
	softplus	64	$2.3 \cdot 10^{-3}$	1.08	$1.4 \cdot 10^{-4}$
	relu	64	$8.2 \cdot 10^{-3}$	1.12	$4.6 \cdot 10^{-4}$
	gelu	64	$1.8 \cdot 10^{-3}$	1.08	$1.4 \cdot 10^{-4}$
	leakyrelu	64	$3.3 \cdot 10^{-3}$	1.01	$1.8 \cdot 10^{-4}$
Fashion	g	64	$4.7 \cdot 10^{-4}$	1.04	$3.3 \cdot 10^{-5}$
	sigmoid	64	$2.5 \cdot 10^{-3}$	1.17	$6.8 \cdot 10^{-3}$
	softplus	64	$3.7 \cdot 10^{-3}$	1.28	$1.5 \cdot 10^{-3}$
	tanh	64	$3.3 \cdot 10^{-4}$	1.11	$4.8 \cdot 10^{-5}$
CIFAR10	g	64	$3.7 \cdot 10^{-3}$	1.00	$5.0 \cdot 10^{-3}$
	sigmoid	64	$1.2 \cdot 10^{-3}$	1.06	$2.7 \cdot 10^{-3}$
	softplus	64	$7.1 \cdot 10^{-3}$	1.06	$1.4 \cdot 10^{-2}$
	relu	64	$2.3 \cdot 10^{-3}$	1.92	$2.4 \cdot 10^{-3}$

Table 2. **Identification with a dataset different from the teacher training dataset:** a teacher network trained on MNIST is reconstructed by training students from teacher queries performed with a different dataset: FashionMNIST.

σ	r	$\mathcal{L}$	$\hat{m}/r$	$<d(w_i, w_i^*)>$
g	64	$1.14 \cdot 10^{-3}$	1.11	$2.26 \cdot 10^{-4}$

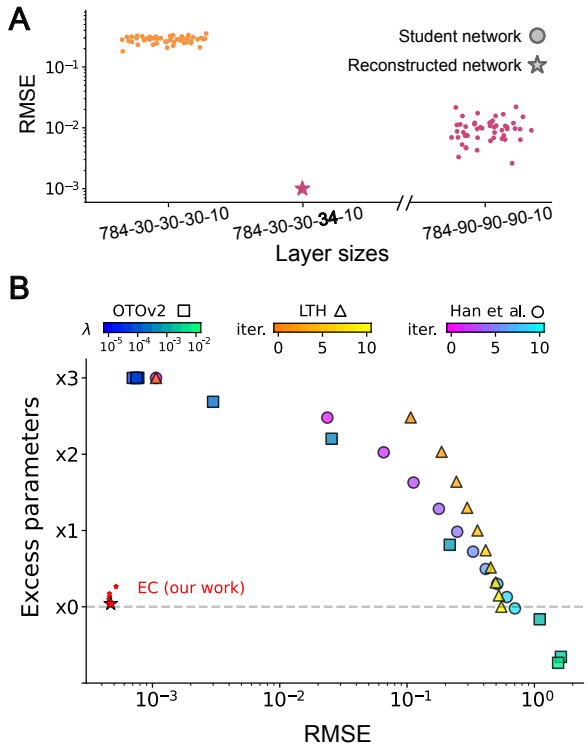


Figure 6. **A) Identification of multiple hidden layers**: A deep teacher of layer sizes 784-30-30-30-10 is reconstructed with Expand-and-Cluster($N = 50, \gamma = 0.5, \beta = \pi/5$) applied to students of factor $\rho = 3$ overparameterisation (magenta dots) with 4 excess neurons in the last hidden layer (magenta star). **B) Baseline comparison with pruning techniques:** recovery of a shallow network trained on MNIST. Weight pruning methods: Han et al. (2015), lottery ticket hypothesis (LTH) (Frankle & Carbin, 2019) and structural pruning OTOv2 (Chen et al., 2022) cannot achieve low imitation loss (RMSE) and equal teacher size (excess parameters = 0) simultaneously, while our algorithm Expand-and-Cluster (EC) is successful (star-points). Different colours indicate different hyperparameters of the methods (regularisation coefficient λ for OTOv2 and pruning iteration number for the other two).

to the teacher size. We compare Expand-and-Cluster to classic pruning techniques such as magnitude pruning (Han et al., 2015), lottery ticket hypothesis (Frankle & Carbin, 2019) and the state-of-the-art structural pruning method OTOv2 (Chen et al., 2022). OTOv2 employs mixed ℓ_1/ℓ_2 regularisation at the level of units, combined with projected gradients to push units to zero. To ensure a fair comparison in terms of computational budget, we compare one run of Expand-and-Cluster ($N = 10, r = 128, \sigma = \tanh$) with 10 runs of the other methods where we sweep their hyperparameters ($\rho = 4$ for all students), Figure 6B. Expand-and-Cluster (EC) requires virtually no hyperparameter tuning, as γ and β are robust to changes in their values (Fig. A.7). We also show that with $N \geq 5$ student networks Expand-and-Cluster identifies the teacher network with at most 10% additional neurons (Fig. A.8). The comparison shown in Figure 6B highlights a clear boundary between our method and current alternatives. We speculate that the iterative nature of these pruning algorithms is not suited to the identification problem, as gradually shrinking the parameter space can lead to rougher landscapes that exhibit high non-convexity, leading to convergence to local minima. While Expand-and-Cluster fully exploits the expressivity of the overparameterised parameter space. Notably, the weight pruning methods of Han et al. (2015) and Frankle & Carbin (2019) are not capable of pruning entire units (Fig. A.7).

7. Conclusions and future work

Given data generated by a teacher of known size and architecture, it is usually impossible to recover its parameters by fitting a student network of the same size as the teacher with standard gradient-based training procedures. The detour to network expansion and clustering is our proposed approach to reliably find the teacher parameters.

Considering the symmetries involved, each neuron weight vector can be fully identified in networks with asymmetric activation functions, identified up to a sign for activation functions with odd or even+linear symmetries and identified up to a constant scaling for activation functions with positive scaling symmetry. Convolutional layer symmetries are mappable to the ones listed in Section 4, by simply treating individual channel weight kernels as hidden neuron weight vectors. Pooling and normalisation layers are not expected to introduce new symmetries. Other symmetries induced by deep ReLU networks are discussed in Grigsby et al. (2023), but those are mostly unidentifiable structures that do not contribute to the input-output mapping of a network. To identify modern convolutional networks, an analysis of the symmetries of residual layers is needed.

At this stage, our results are limited to small-scale setups, primarily due to the high computational budget required to reach low losses in bigger networks. A straightforward extension of this work is to focus on scaling to deeper networks and layer types, as well as speeding up the training process. Another future research direction concerns the query dataset needed to reconstruct the teacher network. Theoretical developments are required to answer questions such as: what type of input queries are informative for the identification process? How can one generate more queries when access to teacher training data is limited or not granted? How many queries are sufficient? Towards reverse engineering biological neural circuits, we show that we can solve the weight identifiability problem in artificial feed-forward networks. However, further steps are needed to improve the reconstruction of connectivity from neural activity measurements, to cope, for example, with recurrent connectivity, unknown and noisy activations, partial observability or the integration of anatomical information.

Acknowledgements

This work was supported by Sinergia Project CRSII5_198612 and SNF Project 200020_207426. The authors thank João Sacramento, Angelika Steger, Benjamin Grewe and Alexander Mathis for early feedback on the research.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abbe, E., Cornacchia, E., Hazla, J., and Marquis, C. An initial alignment between neural network and target is needed for gradient descent to learn. In *International Conference on Machine Learning*, pp. 33–52. PMLR, 2022.

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Allen-Zhu, Z. and Li, Y. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020.

Arjevani, Y. and Field, M. Analytic study of families of spurious minima in two-layer relu neural networks: a tale of symmetry ii. *Advances in Neural Information Processing Systems*, 34:15162–15174, 2021.

Baum, E. B. Neural net algorithms that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19, 1991.

Beyer, L., Zhai, X., Royer, A., Markeeva, L., Anil, R., and Kolesnikov, A. Knowledge distillation: A good teacher is patient and consistent. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10925–10934, 2022.

Bona-Pellissier, J., Bachoc, F., and Malgouyres, F. Parameter identifiability of a deep feedforward relu neural network. *arXiv preprint arXiv:2112.12982*, 2021.

Brea, J., Martinelli, F., Şimşek, B., and Gerstner, W. Mlpgradientflow: going with the flow of multilayer perceptrons (and finding minima fast and accurately). *arXiv preprint arXiv:2301.10638*, 2023.

Bui Thi Mai, P. and Lampert, C. Functional vs. parametric equivalence of relu networks. In *8th International Conference on Learning Representations*, 2020.

Carlini, N., Jagielski, M., and Mironov, I. Cryptanalytic extraction of neural network models. In *Annual International Cryptology Conference*, pp. 189–218. Springer, 2020.

Carlini, N., Paleka, D., Dvijotham, K. D., Steinke, T., Hayase, J., Cooper, A. F., Lee, K., Jagielski, M., Nasr, M., Conmy, A., et al. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634*, 2024.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*, 2018.

Chen, T., Liang, L., Tianyu, D., Zhu, Z., and Zharkov, I. Otov2: Automatic, generic, user-friendly. In *The Eleventh International Conference on Learning Representations*, 2022.

Chizat, L. and Bach, F. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

Cooper, Y. The loss landscape of overparameterized neural networks. *arXiv preprint arXiv:1804.10200*, 2018.

Du, S. S., Zhai, X., Poczos, B., and Singh, A. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

Elhage, N., Hume, T., Olsson, C., Schiefer, N., Henighan, T., Kravec, S., Hatfield-Dodds, Z., Lasenby, R., Drain, D., Chen, C., Grosse, R., McCandlish, S., Kaplan, J., Amodei, D., Wattenberg, M., and Olah, C. Toy models of superposition. *Transformer Circuits Thread*, 2022. https://transformer-circuits.pub/2022/toy_model/index.html.

Entezari, R., Sedghi, H., Saukh, O., and Neyshabur, B. The role of permutation invariance in linear mode connectivity of neural networks. In *10th International Conference on Learning Representations: ICLR 2022*, 2022.

Fefferman, C. Reconstructing a neural net from its output. *Revista Matemática Iberoamericana*, 10(3):507–555, 1994.

Fiedler, C., Fornasier, M., Klock, T., and Rauchensteiner, M. Stable recovery of entangled weights: Towards robust identification of deep neural networks from minimal samples. *Applied and Computational Harmonic Analysis*, 62: 123–172, 2023.

Fornasier, M., Klock, T., and Rauchensteiner, M. Robust and resource-efficient identification of two hidden layer neural networks. *Constructive Approximation*, pp. 1–62, 2019.

Fornasier, M., Vybíral, J., and Daubechies, I. Robust and resource efficient identification of shallow neural networks by fewest samples. *Information and Inference: A Journal of the IMA*, 10(2):625–695, 2021.

Fornasier, M., Klock, T., Mondelli, M., and Rauchensteiner, M. Finite sample identification of wide shallow neural networks with biases. *arXiv preprint arXiv:2211.04589*, 2022.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.

Frei, S., Cao, Y., and Gu, Q. Agnostic learning of a single neuron with gradient descent. *Advances in Neural Information Processing Systems*, 33:5417–5428, 2020.

Fu, H., Chi, Y., and Liang, Y. Guaranteed recovery of one-hidden-layer neural networks via cross entropy. *IEEE transactions on signal processing*, 68:3225–3235, 2020.

Fukumizu, K. and Amari, S.-i. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural networks*, 13(3):317–327, 2000.

Furlanello, T., Lipton, Z., Tschannen, M., Itti, L., and Anandkumar, A. Born again neural networks. In *International conference on machine learning*, pp. 1607–1616. PMLR, 2018.

Glorot, X. and Bengio, Y. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Goldt, S., Advani, M., Saxe, A. M., Krzakala, F., and Zdeborová, L. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *Advances in neural information processing systems*, 32, 2019.

Grigsby, E., Lindsey, K., and Rolnick, D. Hidden symmetries of relu networks. In *International Conference on Machine Learning*, pp. 11734–11760. PMLR, 2023.

Han, S., Pool, J., Tran, J., and Dally, W. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

Haspel, G., Boyden, E. S., Brown, J., Church, G., Cohen, N., Fang-Yen, C., Flavell, S., Goodman, M. B., Hart, A. C., Hobert, O., et al. To reverse engineer an entire nervous system. *arXiv preprint arXiv:2308.06578*, 2023.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., and Peste, A. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *J. Mach. Learn. Res.*, 22(241):1–124, 2021.

Hu, H., Peng, R., Tai, Y.-W., and Tang, C.-K. Network trimming: A data-driven neuron pruning approach towards efficient deep architectures. *arXiv preprint arXiv:1607.03250*, 2016.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Jagielski, M., Carlini, N., Berthelot, D., Kurakin, A., and Papernot, N. High accuracy and high fidelity extraction of neural networks. In *29th USENIX security symposium (USENIX Security 20)*, pp. 1345–1362, 2020.

Janzamin, M., Sedghi, H., and Anandkumar, A. Beating the perils of non-convexity: Guaranteed training of neural networks using tensor methods. *arXiv preprint arXiv:1506.08473*, 2015.

Jordan, K., Sedghi, H., Saukh, O., Entezari, R., and Neyshabur, B. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. 2009.

Kursa, M. B. and Rudnicki, W. R. Feature selection with the boruta package. *Journal of statistical software*, 36: 1–13, 2010.

Lasserre, J. B. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001.

LeCun, Y. The mnist database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

LeCun, Y., Denker, J., and Solla, S. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

Liu, Z., Wang, Y., Vaidya, S., Ruehle, F., Halverson, J., Soljačić, M., Hou, T. Y., and Tegmark, M. Kan: Kolmogorov-arnold networks. *arXiv preprint arXiv:2404.19756*, 2024.

Lovász, L. and Schrijver, A. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991.

Mei, S., Bai, Y., and Montanari, A. The landscape of empirical risk for nonconvex losses. *The Annals of Statistics*, 46(6A):2747–2774, 2018.

Misra, D. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

Mondelli, M. and Montanari, A. On the connection between learning two-layer neural networks and tensor decomposition. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1051–1060. PMLR, 2019.

Murtagh, F. and Contreras, P. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(1): 86–97, 2012.

Mussay, B., Feldman, D., Zhou, S., Braverman, V., and Osadchy, M. Data-independent structured pruning of neural networks via coresets. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

Olah, C., Satyanarayan, A., Johnson, I., Carter, S., Schubert, L., Ye, K., and Mordvintsev, A. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

Oliynyk, D., Mayer, R., and Rauber, A. I know what you trained last summer: A survey on stealing machine learning models and defences. *arXiv preprint arXiv:2206.08451*, 2022.

Petzka, H., Trimmel, M., and Sminchisescu, C. Notes on the symmetries of 2-layer relu-networks. In *Proceedings of the Northern Lights Deep Learning Workshop*, volume 1, pp. 6–6, 2020.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

Raman, D. V., Rotondo, A. P., and O'Leary, T. Fundamental bounds on learning performance in neural circuits. *Proceedings of the National Academy of Sciences*, 116(21): 10537–10546, 2019.

Rolnick, D. and Kording, K. Reverse-engineering deep relu networks. In *International Conference on Machine Learning*, pp. 8178–8187. PMLR, 2020.

Rotskoff, G. and Vanden-Eijnden, E. Trainability and accuracy of artificial neural networks: An interacting particle system approach. *Communications on Pure and Applied Mathematics*, 75(9):1889–1935, 2022.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. Learning internal representations by error propagation. In Rumelhart, D. E., McClelland, J. L., and Group, P. R. (eds.), *Parallel Distributed Processing*, volume 1, chapter 8, pp. 318–362. MIT press Cambridge, MA, 1986.

Saad, D. and Solla, S. A. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

Safran, I. and Shamir, O. Spurious local minima are common in two-layer relu neural networks. In *International conference on machine learning*, pp. 4433–4441. PMLR, 2018.

Shapson-Coe, A., Januszewski, M., Berger, D. R., Pope, A., Wu, Y., Blakely, T., Schalek, R. L., Li, P. H., Wang, S., Maitin-Shepard, J., et al. A petavoxel fragment of human cerebral cortex reconstructed at nanoscale resolution. *Science*, 384(6696):eadk4858, 2024.

Şimşek, B., Bendjeddou, A., Gerstner, W., and Brea, J. Should under-parameterized student networks copy or average teacher weights? *Advances in Neural Information Processing Systems*, 36, 2024.

Şimşek, B., Ged, F., Jacot, A., Spadaro, F., Hongler, C., Gerstner, W., and Brea, J. Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances. In *International Conference on Machine Learning*, pp. 9722–9732. PMLR, 2021.

Singh, S. P. and Jaggi, M. Model fusion via optimal transport. *Advances in Neural Information Processing Systems*, 33:22045–22055, 2020.

Srinivas, S. and Babu, R. V. Data-free parameter pruning for deep neural networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, pp. 31.1–31.12, September 2015.

Stanton, S., Izmailov, P., Kirichenko, P., Alemi, A. A., and Wilson, A. G. Does knowledge distillation really work? *Advances in Neural Information Processing Systems*, 34: 6906–6919, 2021.

Stock, P. and Gribonval, R. An embedding of relu networks and an analysis of their identifiability. *Constructive Approximation*, pp. 1–47, 2022.

Sun, J. Provable nonconvex methods/algorithms, 2021. URL https://sunju.org/research/nonconvex/.

Sussmann, H. J. Uniqueness of the weights for minimal feedforward nets with a given input-output map. *Neural networks*, 5(4):589–593, 1992.

Tian, Y. Student specialization in deep rectified networks with finite width and input dimension. In *International Conference on Machine Learning*, pp. 9470–9480. PMLR, 2020.

Tian, Y., Jiang, T., Gong, Q., and Morcos, A. Luck matters: Understanding training dynamics of deep relu networks. *arXiv preprint arXiv:1905.13405*, 2019.

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. Stealing machine learning models via prediction {APIs}. In *25th USENIX security symposium (USENIX Security 16)*, pp. 601–618, 2016.

Udrescu, S.-M. and Tegmark, M. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Vlačić, V. and Bölcskei, H. Affine symmetries and neural network identifiability. *Advances in Mathematics*, 376: 107485, 2021.

Vlačić, V. and Bölcskei, H. Neural network identifiability for a family of sigmoidal nonlinearities. *Constructive Approximation*, 55(1):173–224, 2022.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. Federated learning with matched averaging. In *International Conference on Learning Representations*, 2020.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Yehudai, G. and Ohad, S. Learning a single neuron with gradient methods. In *Conference on Learning Theory*, pp. 3756–3786. PMLR, 2020.

Zhang, Y., Qu, Q., and Wright, J. From symmetry to geometry: Tractable nonconvex problems. *arXiv preprint arXiv:2007.06753*, 2020.

Zhang, Y., Tiňo, P., Leonardis, A., and Tang, K. A survey on neural network interpretability. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.

Zhong, K., Song, Z., Jain, P., Bartlett, P. L., and Dhillon, I. S. Recovery guarantees for one-hidden-layer neural networks. In *International conference on machine learning*, pp. 4140–4149. PMLR, 2017.

A. Appendix

A.1. Code availability

The code is available at https://github.com/flavio-martinelli/expand-and-cluster.

A.2. Activation functions

A.2.1. Even plus linear activation functions

Considering a neuron with an activation function that can be decomposed into linear plus even terms:

$$a\sigma(wx+b) = ac_1 \cdot (wx+b) + a\sigma_{\text{even}}(wx+b)$$

where c_1 is the slope of the linear approximation around 0. The total contribution of aligned, $+(wx+b)$, and opposite, $-(wx+b)$, student neurons is:

$$\sum_{i\in N^+} a_i\sigma(wx+b) + \sum_{j\in N^-} a_j\sigma(-(wx+b)) = \underbrace{c_1\left(\sum_{i\in N^+} a_i - \sum_{j\in N^-} a_j\right)(wx+b)}_{\text{Linear}} + \underbrace{\sum_{k\in N^\pm} a_k\,\sigma_{\text{even}}(wx+b)}_{\text{Even}}$$

where $N^+, N^-, N^\pm$ contain aligned, opposite or all neuron indices, respectively.

If $\sum_{k\in N^\pm} a_k = 0$, we obtain a *linear-type* group that contributes a linear function to the next layer; to keep an exact mapping of the teacher, there must be another *linear-type* group in the same layer contributing the exact opposite linear term. If $\sum_{k\in N^\pm} a_k = a^*$, then the neuron group is a *linear duplicate-type*, replicating a teacher neuron up to a misaligned linear contribution; that can be accounted for by another linear group in the same layer. An example of different neurons reached by a softplus student is shown in supplementary Figure A.5.

A.2.2. Constant plus odd activation functions

Considering a neuron with an activation function that can be decomposed into a constant plus odd term:

$$a\sigma(wx+b) = ac_0 + a\sigma_{\text{odd}}(wx+b)$$

where $c_0 = \sigma(0)$. The total contribution of aligned and opposite student neurons is:

$$\sum_{i\in N^+} a_i\sigma(wx+b) + \sum_{j\in N^-} a_j\sigma(-(wx+b)) = \underbrace{c_0\sum_{k\in N^\pm} a_k}_{\text{Constant}} + \underbrace{\left(\sum_{i\in N^+} a_i - \sum_{j\in N^-} a_j\right)\sigma_{\text{odd}}(wx+b)}_{\text{Odd}}$$

where $N^+, N^-, N^\pm$ contain aligned, opposite or all neuron indices, respectively.

If $\sum_{i\in N^+} a_i - \sum_{j\in N^-} a_j = 0$, we obtain a *constant-type* group; to keep an exact mapping of the teacher, the sum of all *constant-type* groups in the same layer and the next layer bias must add up to contribute the original bias of the next layer neuron. If $\sum_{i\in N^+} a_i - \sum_{j\in N^-} a_j = a^*$, then the neuron group is a *duplicate-type*, replicating a teacher neuron up to a misaligned constant contribution; that can be accounted for by other constant groups in the same layer or biases in the next. For tanh, there is no constant contribution term ($c_0 = 0$) while it is the case for sigmoid.

A.2.3. Classification of activation functions based on symmetries

- **No symmetries** sigmoid + softplus, Mish (Misra, 2019), SELU (Klambauer et al., 2017),
- **Odd**: tanh

- ▷ **Odd + constant**: sigmoid

- **Even + linear**: GELU (Hendrycks & Gimpel, 2016), Swish/SiLU (Ramachandran et al., 2017)
  - ▷ **Even + linear + constant**: softplus
  - ▷ **Even + linear + positive scaling**: ReLU, LeakyReLU

A.3. Expand-and-Cluster detailed procedure

The input weight vectors of all neurons in a given layer of N mildly overparameterised students are clustered with hierarchical clustering (Murtagh & Contreras, 2012) using the average linkage function $\ell(\mathcal{A},\mathcal{B}) = \frac{1}{|\mathcal{A}|\cdot|\mathcal{B}|}\sum_{i\in\mathcal{A}}\sum_{j\in\mathcal{B}} S_{ij}$, where $S_{ij} = \|w_i - w_j\|_2$ and w_i and w_j are input weight vectors of neurons in the same layer but potentially from different students (Fig. 3B). To identify the clusters of teacher neurons we look for the appropriate height h to cut the dendrogram resulting from hierarchical clustering (Fig. 3C). With $K(h) = \{\kappa_1, \kappa_2, \ldots\}$ the set of clusters at threshold h, and $|\kappa_i|$ the size of cluster i, we define the set $C(h, \gamma N) = \{\kappa_i \in K(h) \mid |\kappa_i| \geq \gamma N\}$ of clusters larger than a fraction $\gamma \in (0, 1]$ of the number of students N. For $\gamma > 1/N$, the set $C(h, \gamma N)$ usually does not contain small clusters of zero or offbound type neurons, because they are not aligned between different students. We cut the dendrogram at the height that maximises the number of big clusters: $\hbar = \arg\max_h |C(h, \gamma N)|$. The set $C(\hbar, \gamma N)$ may still contain clusters of non-aligned neurons whose input weights are close in Euclidean distance (e.g. approximate constant-type neurons with $\|w_i\|$ near zero) but not aligned in angle. Therefore we remove clusters whose within-cluster median angle is higher than β. Each remaining cluster is then collapsed into a single hidden neuron with a winner-take-all policy: we choose the weight vector(s) that belong to the best-performing student in the cluster. Note that if multiple output weights $a_1, \ldots, a_q$ of the same network belong to the same cluster, they can be combined into a single output weight $a = \sum_i a_i$, following the definition of duplicate neurons (Fig. 2). If the best student duplicates a given teacher neuron more than once, we take their average. Alternatively, we have also tried to average over all neurons in each cluster and obtained a similar final performance. The hyperparameters of Expand-and-Cluster (N, γ, β) used for each experiment are summarised in Appendix A.6.4.

The biases could approximately be reconstructed by identifying all the constant-type neurons, but we found this procedure somewhat brittle. Instead, we fine-tune all bias parameters (keeping constant the weight vectors) with a few steps of gradient descent on the reconstructed network. After each layer reconstruction, we retrain N students with fixed non-trainable reconstructed layers but learnable biases, and the remaining layers are overparameterised and learnable. In this retraining phase, we monitor the learned bias values b_k^n of the last reconstructed layer, if the same neuron k learns different bias values across different students n ($\mathrm{std}_n(b_k^n) > 0.1$) we consider that neuron badly clustered and therefore remove it from the layer. We repeat this procedure until the last hidden layer, the final output layer can be reconstructed by simply retrieving the output weights of the last hidden layer neurons. Ultimately further fine-tuning can be performed on the whole network to minimise the loss as much as possible.

A.4. Synthetic data in teacher-student networks

All tasks with synthetic data have d-dimensional uniformly distributed input data in the range $x_i \in [-\sqrt{3}, \sqrt{3}]$. A specific task is defined by the parameters of a teacher network. Each hidden neuron i of the teacher is randomly sampled from a set of input weights $w_i \in \{-1, 0, 1\}^{d_{in}}$, output weights $a_i \in \{-1, 1\}$ and biases $b_i \in \{-\frac{2}{3}\sqrt{3}, -\frac{1}{3}\sqrt{3}, 0, \frac{1}{3}\sqrt{3}, \frac{2}{3}\sqrt{3}\}$. We repeat the sampling if two hidden neurons are identical up to output weights signs to avoid two hidden neurons cancelling each other. The resulting input weight vectors w are first normalised to unity and then both w and b are multiplied by a factor of 3. The above procedure yields hyperplanes in direction w located at a distance $|b|/||w||$ from the origin, and a steeply rising (or falling) activation on the positive side of the hyperplane. Finally, analogous to batch normalization, the output weights and biases are scaled such that the output has zero-mean and unit variance when averaged over the input distribution: $a \leftarrow a/\mathrm{std}(y)$ and $b_2 = -\langle y\rangle/\mathrm{std}(y)$, where y is the output vector of the network. We study teachers with input dimensionality $d_{in} \in \{2, 4, 8, 16, 32\}$ and hidden layer size $r \in \{2, 4, 8\}$. Figure 4A shows examples of different teachers with input dimension $d_{in} = 2$ and a single hidden layer. We use the asymmetric activation function $\sigma = \sigma_{sig}(4x) + \sigma_{soft}(x)$, where $\sigma_{sig} = \frac{1}{1+e^{-x}}$ and $\sigma_{soft} = log(1 + e^x)$ for all our simulations unless specified otherwise. The construction of deep teacher networks follows the same mechanism as described above, with the additional extra step of scaling the weights such that each unit of the successive layer receives standardised inputs, analogous to what batch-norm would do for a full batch. This procedure ensures that the input support to each layer does not drift to values that are off-bounds with respect to where each hidden neuron hyperplane is placed. Classically trained networks also follow this scheme to have useful activations, but it is crucial to avoid potential support drift in arbitrarily constructed deep networks.

A.5. Recovery of deep artificial teacher networks

We built two and three hidden layers networks by stacking the procedure to generate artificial teachers detailed in Appendix A.4. Expand-and-Cluster *without retraining step at each layer reconstruction* was applied to these networks: the reconstructed networks have one or two superfluous neurons for the two and three-layer teachers, respectively; (Fig. A.1). We emphasise that the final loss of the pruned student networks is at least 8 orders of magnitudes lower than the loss of the best out of 10 students trained with the *correct* number of neurons in each of the hidden layers. Thus, running 10 training runs with student networks overparameterised by a factor of $\rho = 8$ enables us to correctly identify or nearly identify the teacher network, while direct training was not successful with the same number of runs (Fig. A.1). Detailed explanations on training and reconstruction are explained in the following sections (Appendix A.6.2 and Appendix A.6.4).

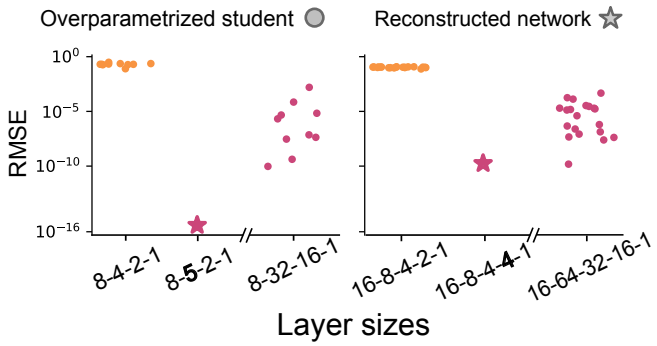


Figure A.1. **Deep artificial teachers**: a teacher with input dimension 8, a first hidden layer with 4 neurons, and a second hidden layer with 2 neurons, and a single output (denoted as 8-4-2-1, left) was constructed by stacking the procedure of Figure 4 and generated data to train students of overparameterisation $\rho = 8$ (denoted as 8-32-16-1). Similarly, a teacher with architecture 16-8-4-2 (right) generated data to train students with architecture 16-64-32-16. Expand-and-Cluster applied to students trained with an overparameterisation of $\rho = 8$ identified the teacher network with one additional neuron in the first hidden layer (left) or with two additional neurons in the third hidden layer (right). In both cases, the loss is below 10^{-10} whereas direct training with the 'correct' network architecture never reached a loss below 10^{-2}.

A.6. Training and reconstruction details

A.6.1. Synthetic teacher tasks

To explore overparameterised networks trained to exact zero-loss or up to machine precision (for Float64 machine precision is at 10^{16}), we integrated the gradient flow differential equation $\dot{\theta}(t) = -\nabla\mathcal{L}(\theta(t), \mathcal{D})$ with ODE solvers, where $\mathcal{L}$ is the mean square error loss. Specifically, we used the package MLPGradientFlow.jl (Brea et al., 2023) to follow the gradient with high accuracy and exact or approximate second-order methods to fine-tune convergence to a local or global minimum. All of the toy model networks are trained with Float64 precision on CPU machines (Intel Xeon Gold 6132 on Linux machines).

The networks of Figure 4 were trained on an input dataset $\mathbf{X}$ of 30k data points drawn from a uniform distribution between $-\sqrt{3}$ and $+\sqrt{3}$ and targets y computed by the teacher network on the same input $\mathbf{X}$. Students were initialised following the Glorot normal distribution, mean 0 and std = $\sqrt{\frac{2}{\text{fan\_in}+\text{fan\_out}}}$ (Glorot & Bengio, 2010). We allocated a fixed amount of iteration steps per student: 5000 steps of the ODE solver `KenCarp58` for all networks, plus an additional 5000 steps of exact second order method `NewtonTrustRegion` for non-overparameterised networks ($\rho = 1$) or 250 steps of `BFGS` for overparameterised networks ($\rho \geq 2$). The stopping criteria for the second training phase were: mean square error loss $\leq 10^{-31}$ or gradient norm $\nabla\mathcal{L}(\theta(t)) \leq 10^{-16}$. Each iteration step is full-batch, the only source of randomness in a student network is its initialization.

The networks shown in Figure 5, after reconstruction, were fine-tuned for a maximum of 15 minutes with `NewtonTrustRegion` if the number of parameters was below or equal to 32 or `LD_SLSQP` otherwise.

A.6.2. SYNTHETIC DEEP NETWORKS

Toy deep students were trained on a dataset generated in the same way as their shallow counterparts. We allocated 3 hours to solve the gradient flow ODE with `Runge-Kutta4` followed by 6 to 12 hours of approximate second-order optimisation with `BFGS`. After reconstruction, we gave 60 minutes of time budget to fine-tune with `LD_SLSQP` (Fig. 6A).

A.6.3. TRAINING OF TEACHERS AND STUDENTS

Several teachers with a single hidden layer of $10, 30$ or 60 neurons were trained with the activation function defined in Appendix A.2 to best classify the MNIST dataset by minimizing the cross-entropy loss with the Adam optimiser (Kingma & Ba, 2015) and early stopping criterion. For each hidden-layer size, the best-performing teacher was used to train the students, see Figure A.3A. The student networks were trained to replicate the teacher logits (activation values before being passed to the softmax function) by minimizing a mean square error loss. Students were initialised with the Glorot uniform distribution, ranging from $-a$ to a, where $a = \sqrt{\frac{6}{\text{fan\_in}+\text{fan\_out}}}$. The training was performed with the Adam optimiser on mini-batches of size 640 with an adaptive learning rate scheduler that reduces the learning rate after more than 100 epochs of non-decreasing training loss. A maximum of 25k epochs was allocated to train these students on GPU machines (NVIDIA Tesla V100 32G). For the layerwise reconstruction of deep teacher networks, every retraining was given a maximum of 10k epochs.

The reconstructed networks shown in Figure 6B are fine-tuned with the same optimising recipe for a maximum of 2000 epochs. For deep reconstructed networks, a final finetuning of 15k epochs was performed on all the parameters.

A.6.4. EXPAND-AND-CLUSTER HYPERPARAMETERS

The Expand-and-Cluster procedure has only 3 hyperparameters to tune: the number of student networks N, the cutoff threshold to eliminate small clusters γ and the maximally admitted alignment angle β.

- **Shallow synthetic teachers:** all the procedure was performed with $N = 10$ (for $r = 8$) or $N = 20$ (for $r = 2, 4$), $\gamma = 0.8$ and $\beta = \pi/24$.
- **Deep synthetic teachers:** we reconstructed the synthetic deep networks *without the retraining step at each layer reconstruction*. This required looser conditions, especially for the higher layers: for the synthetic deep networks of 2 hidden layers: $N = 10$, $\gamma = 0.4$ and $\beta = 1 - \arccos(0.01) \simeq 8°$; for the 3 hidden layer case: $N = 20$, $\gamma = 1/3$, $\beta \simeq 0.26°$ for the first layer and $\beta \simeq 30°$ for the other layers. The alignment gets quickly lost as depth increases, motivating the looser β angles. We found this procedure too brittle and decided to modify the algorithm into the version in the main paper (Fig. 3) by adding the retraining step at each layer reconstruction. The reconstruction results for deep MNIST teachers follow this updated algorithm.
- **Shallow MNIST teachers:** for MNIST student networks we notice that no alignment of weight vectors can be expected for the corner pixels of the images because these pixels have the same value for nearly all input images. This prevents weights connected to corner pixels from moving from the random initialization. Therefore we removed the uninformative pixels using the tree-based feature importance method Boruta (Kursa & Rudnicki, 2010) to identify with statistical significance the informative features; the resulting map is shown in Figure A.3B. The parameters used for Expand-and-Cluster on MNIST shallow networks of Figure 6B are $\gamma = 0.5$ and $\beta = \pi/6$ while N sweeps from 2 to 20.
- **Deep MNIST teacher:** all the procedure was performed with $N = 50$, $\gamma = 0.5$ and $\beta = \pi/5$ for every layer. Layerwise reconstruction alternated to the training of new students is a much more stable procedure than the one described for deep synthetic teachers as there is no need to tune the Expand-and-Cluster parameters for every layer.

A.7. Convolutional layers reconstruction

We trained a convolutional neural network on the CIFAR10 dataset. The network is composed of two convolutional layers of 16 channels, each followed by a maxpool operation. After the convolutions, a fully connected layer of 32 hidden neurons computes the outputs. Students were trained with overparameterisation factor $\rho = 3$, we adapted the clustering algorithm to consider channels in the convolutional layers as analogous to hidden neurons of an MLP. The alignment of the different channel weights between the teacher and the reconstructed student is shown in Fig. A.16.

A.8. Supplementary figures and table

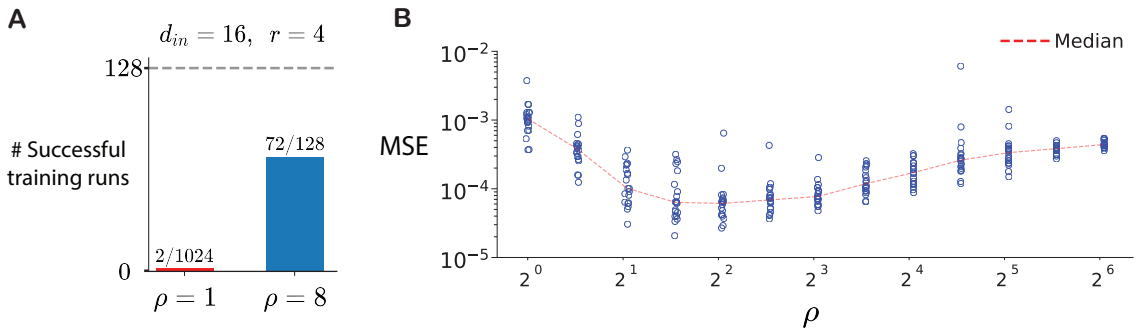


Figure A.2. **A) overparameterisation performance scales beyond simple linear parameter scaling:** we successfully retrieve an identical input-output mapping of the original network with a student of the same size ($\rho = 1$) only 2 times out of 1024 runs. If we instead train 8 times fewer networks ($n = 128$) that are 8 times bigger ($\rho = 8$) we achieve a significant improvement in success rate, 72 out of 128 runs. Note that the computational budget of the two experiments is equal. Example shown with teacher configuration $d_{in} = 16, r = 4$, teacher $\#7$. **B) Probing expansion factors:** It is impossible to know the overparameterisation factor of a student if the teacher is unknown. Nevertheless, one can probe different student sizes with quick training runs of stochastic gradient descent to identify a suitable student size that gives minimal losses. With a fixed time budget per training run, we notice an increase in MSE for large overparameterisation.

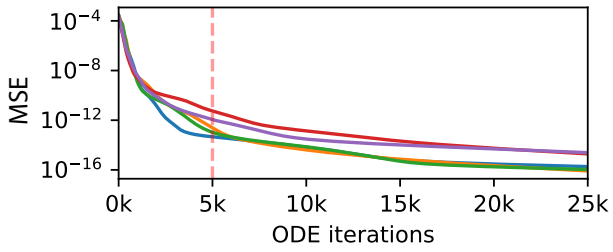


Figure A.3. **Training longer slowly decreases student loss for difficult teachers:** the results shown in Figure 4 for difficult teachers are not at convergence despite the large amount of ODE solver steps dedicated (5K for every simulation). If training is continued we can see a slow decrease in loss, hinting that a very complex landscape is generated by difficult teachers. This simulation took more than a day to compute. Each colour corresponds to a different student loss curve, the teacher used was of $d_{in} = 2, r = 8$.

Table A.3. **Detailed statistics of experiments of Table 6A**: $\mathcal{L}$ is the root mean square error loss of the student network, $\hat{m}/r$ is the number of neurons of the student divided by the teacher size, $< d(w_i, w_i^*) >$ and $\max_i d(w_i, w_i^*)$ are the average and maximum cosine distance between reconstructed and teacher input weight vectors (absolute value cosine distance in case of symmetries where the sign of the weight vector cannot be recovered), $< d(a_i, a_i^*) >$ and $\max_i d(a_i, a_i^*)$ are the same metrics for the output weights.

σ	r	N	γ	β	$\mathcal{L}$	$\hat{m}/r$	$< d(w_i, w_i^*) >$	$\max_i d(w_i, w_i^*)$	$< d(a_i, a_i^*) >$	$\max_i d(a_i, a_i^*)$
g	256	20	0.5	$\pi/3$	$1.53 \cdot 10^{-3}$	1.008	$2.13 \cdot 10^{-4}$	$1.92 \cdot 10^{-3}$	$3.48 \cdot 10^{-8}$	$2.41 \cdot 10^{-6}$
sigmoid	256	20	0.9	$\pi/3$	$8.37 \cdot 10^{-4}$	1.12	$3.77 \cdot 10^{-4}$	$6.00 \cdot 10^{-3}$	$5.49 \cdot 10^{-4}$	$2.84 \cdot 10^{-2}$
tanh	128	10	0.9	$\pi/3$	$1.73 \cdot 10^{-3}$	1.04	$1.29 \cdot 10^{-4}$	$9.26 \cdot 10^{-4}$	$3.06 \cdot 10^{-5}$	$3.91 \cdot 10^{-3}$
softplus	64	20	0.9	$\pi/3$	$2.97 \cdot 10^{-3}$	1.09	$1.08 \cdot 10^{-2}$	$6.82 \cdot 10^{-1}$	$5.66 \cdot 10^{-3}$	$3.62 \cdot 10^{-1}$
relu	64	20	0.5	$\pi/3$	$8.16 \cdot 10^{-3}$	1.12	$4.63 \cdot 10^{-4}$	$8.02 \cdot 10^{-3}$	$3.34 \cdot 10^{-5}$	$2.03 \cdot 10^{-3}$

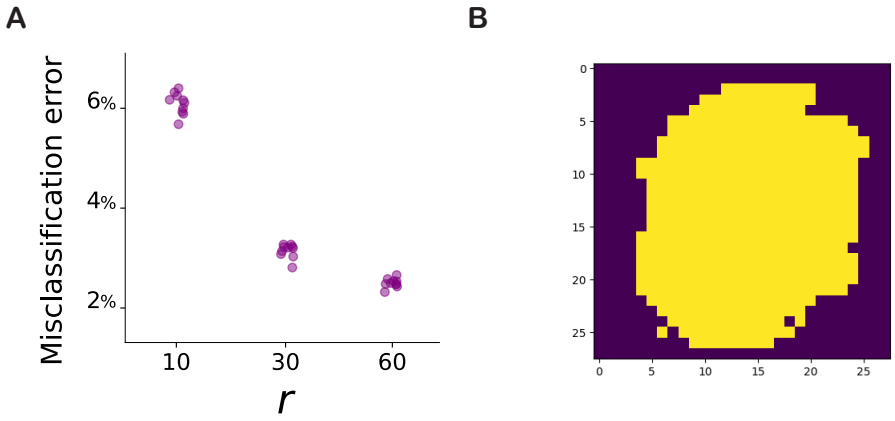


Figure A.4. **A) MNIST teachers misclassification error on test set**. **B) MNIST important pixels mask obtained with BORUTA (Kursa & Rudnicki, 2010):** only connections projecting from yellow pixels are considered for the Expand-and-Cluster procedure.

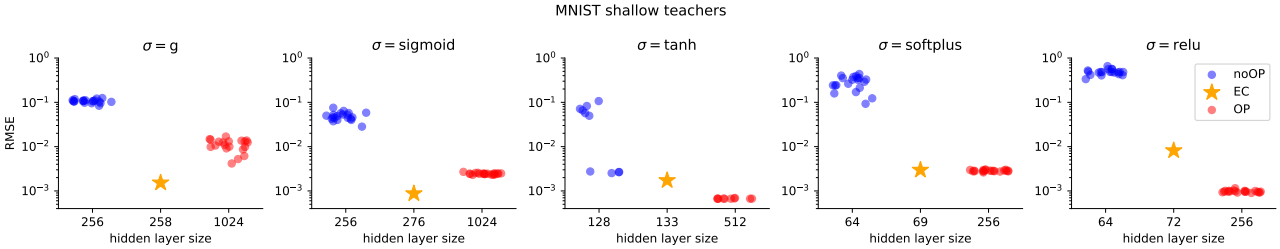


Figure A.9. **Loss and size comparisons for MNIST experiments of Table 1 of the main paper**: without overparameterisation (noOP, blue dots) training gets stuck at high loss values due to the extreme non-convexity of the landscape. These networks show poor imitation of the teachers and no alignment in weights. Only $\sigma = tanh$ seems to be an exception to this phenomenon. Overparameterisation is **necessary** to reliably reach lower losses due to the proliferation of global minima in the landscape (OP, red dots). Expand-and-Cluster can successfully reconstruct the teacher network from the overparameterised students (EC, yellow stars), with few excess neurons.

TEACHER

Neuron#	w_1	w_2	w_3	w_4	b	a
1	0.0798	-0.1591	-1.1932	0.1544	-0.4069	-0.6425
2	0.0885	0.0562	1.4687	-0.0843	1.8938	-3.2114
3	0.5641	-1.2116,	-0.9090	0.5318	0.0863	1.3291
4	-1.0788	-1.1397	-0.0288	0.7331	0.7795	1.3658

Final layer bias: 1.4088

STUDENT

Neuron#	w_1	w_2	w_3	w_4	b	a
1	0.0000	-0.0000	-0.0002	0.0000	0.0745	0.5669
2	-1.0788	-1.1397	-0.0288	0.7331	0.7795	1.3658
3	0.3933	-0.4649	-0.1984	0.1733	0.0283	-0.0000
4	-0.0381	0.0759	0.5690	-0.0736	0.1359	0.5343
5	-0.0798	0.1591	1.1932	-0.1544	0.4052	-0.2547
6	0.0675	-0.0581	-0.1721	0.1226	-0.0294	-0.0000
7	0.0000	0.0000	0.0000	0.0000	0.0518	0.2907
8	0.0049	0.0391	0.6151	-0.0676	-0.0452	-0.0000
9	0.0798	-0.1591	-1.1932	0.1544	-0.4079	-0.3878
10	0.5641	-1.2116	-0.9090	0.5318	0.0863	1.3291
11	-0.0001	0.0001	0.0002	-0.0001	0.1579	0.2025
12	0.0381	-0.0759	-0.5690	0.0736	-0.1359	-0.5343
13	0.0885	0.0562	1.4687	-0.0843	-2.0982	-3.2114
14	0.0000	-0.0000	-0.0003	0.0001	0.0881	0.5097
15	-0.2952	0.0844	0.3304	0.0897	0.0357	-0.0000
16	0.6287	0.6561	0.1238	-0.3991	-0.4162	-0.0000

Final layer bias: 0.2826.

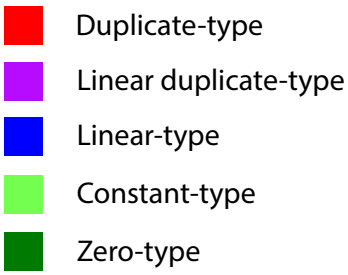


Figure A.5. **Example parameters of a softplus teacher and student:** the table above shows the parameters of a $d_{in} = 4, r = 4$ softplus teacher. The table below shows the parameters of a student of $\rho = 4$ trained on the defined teacher to RMSE $\sim 10^{-11}$. At a near-zero loss, we can classify all the different neurons (colour-coded) composing the student, following the classification scheme of Figure 2. Note that the presence of a linear duplicate type implies the presence of a linear type, this group of 4 neurons (blue and purple coded) collaborates to replicate teacher neuron #1.

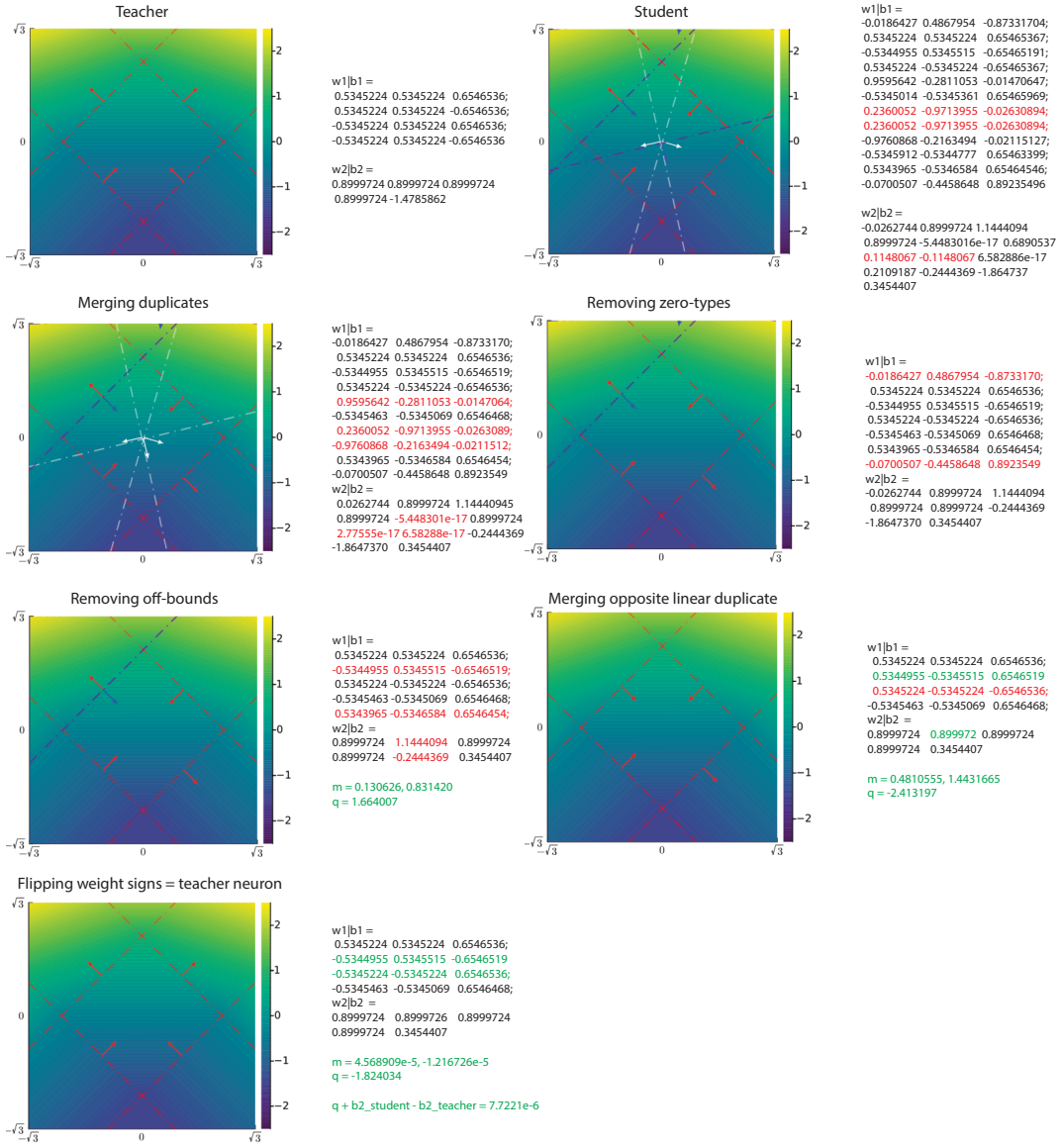


Figure A.6. **Mapping an exact zero loss overparameterised relu student to the teacher using symmetries:** A numerical example of a $\sigma = \mathrm{relu}, r = 4, d_{in} = 2, d_{out} = 1$ teacher. We follow the same network representation as in figure 4: we plot each hidden neuron hyperplane ($wx + b = 0$) and the input weight vector as an arrow, the colour of the arrow indicates if the scalar output weight is > 0 (red), < 0 (blue) or ≈ 0 (white). **Top left**: representation of the teacher, each input and output weight vector (concatenated with bias) are shown numerically on the right side of each figure. **Top right**: $\sigma = \mathrm{relu}, m = 12, d_{in} = 2, d_{out} = 1, \mathcal{L} = 10^{-16}$ overparameterised student. **Following plots from left to right, top to bottom:** each plot shows a step towards mapping the student network to the teacher. Each step maintains functional equivalence by making use of the symmetries of Section 4. Red parameters indicate the parameters modified in the next step, green parameters are the parameters that have been modified with the current step (if both green and red should be present, we leave it in red). In order: *Merging duplicates:* a pair of duplicate neurons is merged by summation of their output weights. *Removing zero-types:* neurons with near-zero output weights (white arrows) are removed. *Removing off-bounds:* off-bound neurons (one is barely visible at the top of the contour plot) are removed by taking care of their contribution. In this case, the off-bounds relu neurons have their hyperplane outside of the input domain and their weight vector pointing towards the input domain, hence they contribute a linear function to the output. Therefore a linear transformation $mx + q$ is added to maintain functional equivalence. *Merging opposite linear duplicates:* Neurons that share a hyperplane but features pointing in opposite directions can be merged into a unique neuron + linear component. *Flipping weight signs:* finally, the weight vectors opposite to the teacher are flipped by adding a linear component. We verify that the final linear component sums up to zero to confirm the functional equivalence of the mapping.

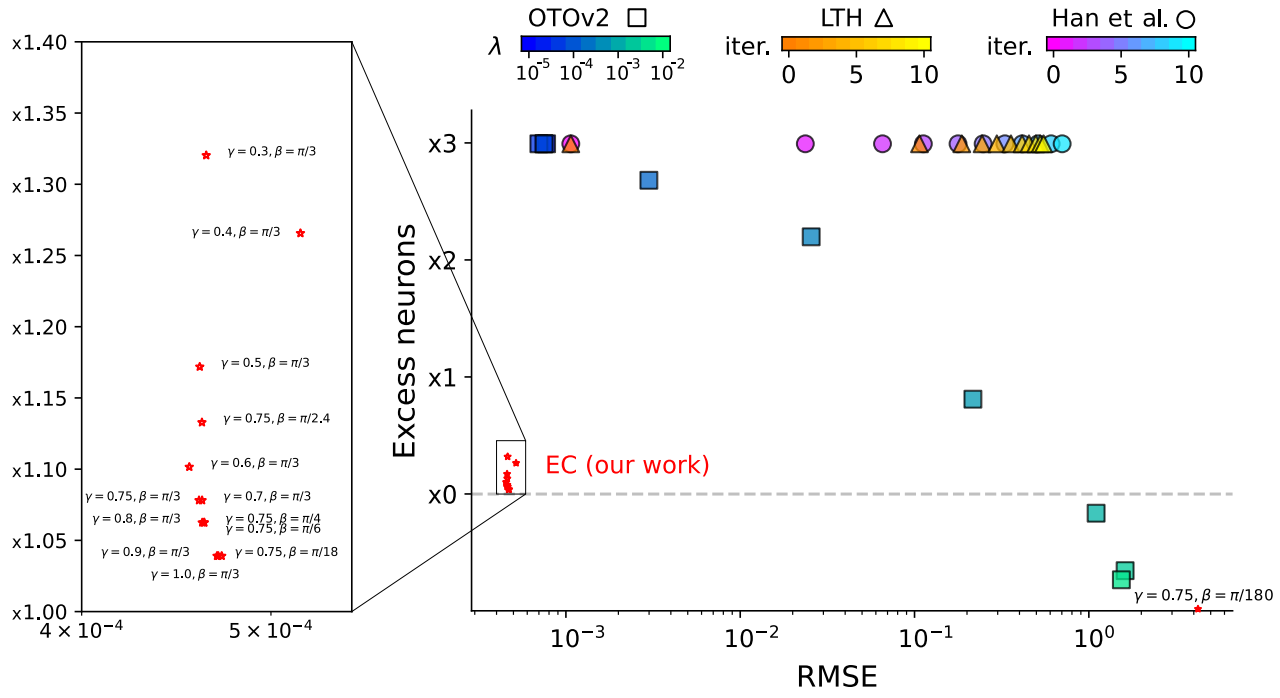


Figure A.7. **Neurons in excess with respect to other pruning methods:** same experiment as Figure 6A but looking at the number of neurons in excess. The weight pruning methods lottery ticket hypothesis (LTH) (Frankle & Carbin, 2019) and classic magnitude pruning (Han et al., 2015) never effectively push all the weights of any unit to zero (as they were not designed to do so). The structural pruning method OTOv2 (Chen et al., 2022) instead prunes only units. Training $N = 10$ overparameterised students for Expand-and-Cluster (EC) is computationally equivalent to training 10 runs of parameter search for the other methods. While EC and OTOv2 runs can be parallelised, for LTH and Han et al. they need to be run sequentially. Once $N = 10$ overparameterised students are trained, testing different hyperparameter sets for EC is not as computationally expensive. Different runs of EC for different parameters γ and β show how EC hyperparameter search is not as crucial as a search for a regularisation coefficient λ. **Inset – Expand-and-Cluster robustness to hyperparameter sweep:** The star-shaped points highlight the stable robustness in the performance of EC for different hyperparameter settings. We sweep $\gamma \in [0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0]$ while keeping $\beta = \pi/3$ and $\beta \in [\pi/2.4, \pi/3, \pi/4, \pi/6, \pi/18, \pi/180]$ while keeping $\gamma = 0.75$. The only set of parameters that fails the reconstruction is $\gamma = 0.75, \beta = \pi/180$. Looking at the role of these parameters in detail: β filters out unaligned clusters which tend to have elements orthogonal to each other. Any value of $\beta \geq \pi/6$ is sufficient to filter out unaligned clusters without filtering potentially important duplicates. While γ is involved in filtering out small clusters that we expect to be made of zero-type neurons, therefore any value of γ that is too low ($\gamma \leq 0.4$) would risk filtering out important clusters, while a value of γ too high risks to miss some clusters of duplicates.

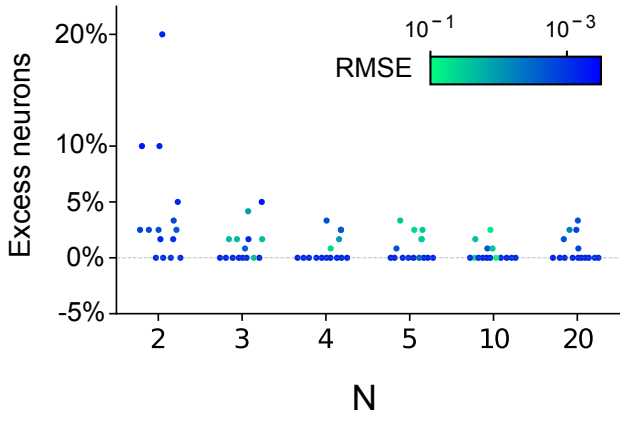


Figure A.8. **A) MNIST shallow teachers**: fraction of excess neurons with respect to the teacher size) clustered as a function of the N students used for Expand-and-Cluster($N, \gamma = 0.5, \beta = \pi/6$). Combined statistics across three shallow teachers of sizes $r \in \{10, 30, 60\}$ pre-trained on MNIST data.

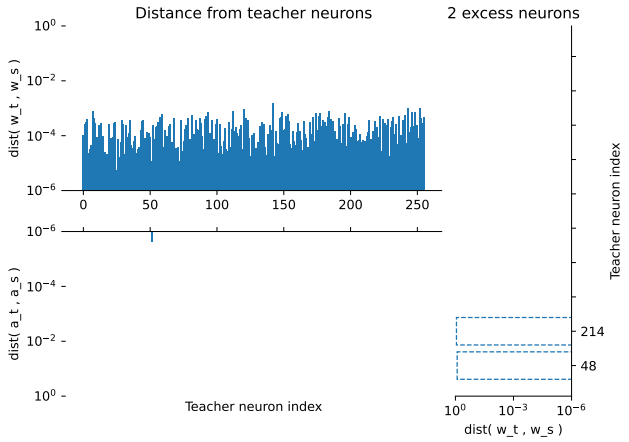


Figure A.10. Reconstruction of $\sigma = \mathrm{g}$ network.

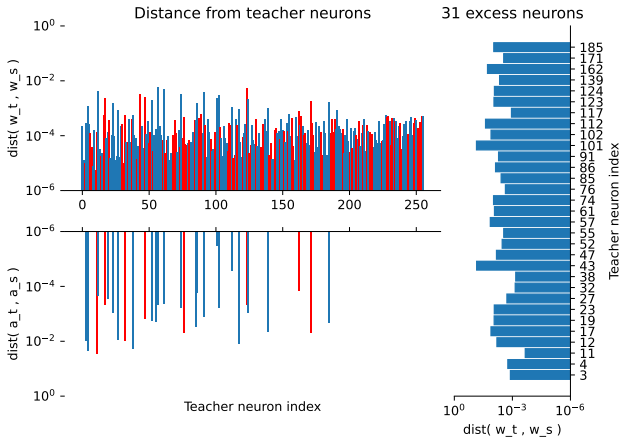


Figure A.11. Reconstruction of $\sigma = \mathrm{sigmoid}$ network.

Figure A.12. **Comparing reconstructed and target network of MNIST experiments in Table 6A:** The left plots show the distance of each teacher neuron input (top) or output (bottom) weight vector with respect to the closest neuron of the reconstructed network. To indicate the unidentifiability of the sign of the weight vectors, we indicate in red the bars of neurons identified with the opposite sign. The right vertical plot shows how many excess neurons were found and to what teacher neuron they align the closest (in input weight vector); empty dashed bars indicate excess neurons with output weight vector norms below 0.1 (putatively zero neurons). We can still find duplicate neurons in reconstructed networks, they can be seen in these plots by having an excess neuron closely aligned to a teacher neuron and, consequently, a worse precision in alignment of the output weights (because of imprecisions in the sum of the duplicates' output weight vectors). The teacher neurons are ordered by descending output weight vector norm, as the lowest norm output weight neurons tend to be learnt at later stages of the learning process (i.e. at lower losses (Tian, 2020)).

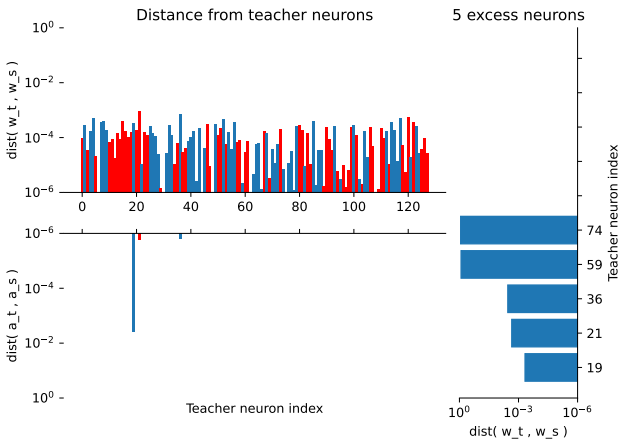


Figure A.13. Reconstruction of $\sigma = \tanh$ network.

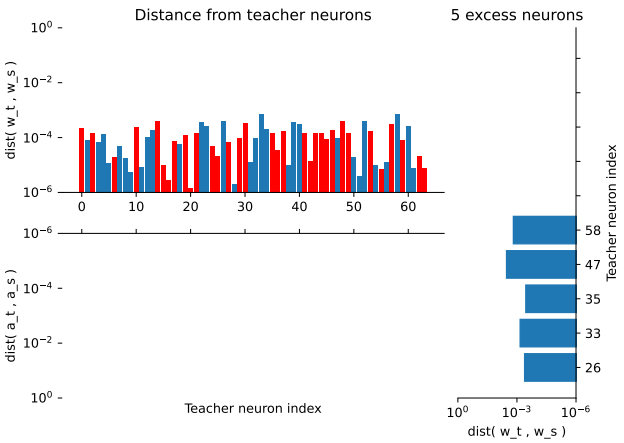


Figure A.14. Reconstruction of $\sigma = \mathrm{softplus}$ network.

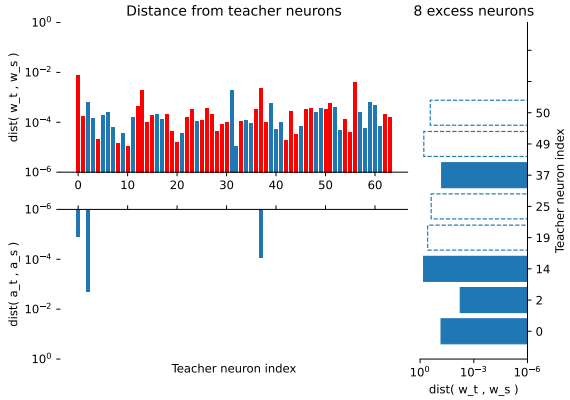
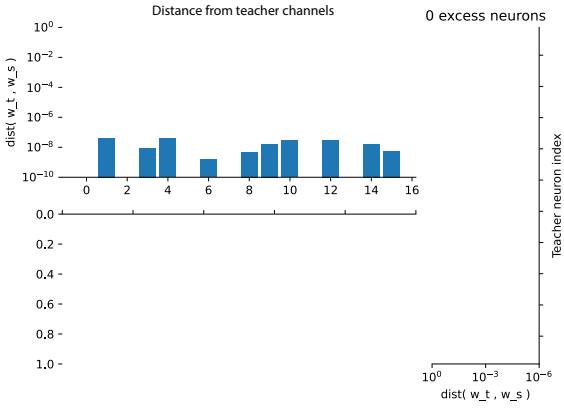


Figure A.15. Reconstruction of $\sigma = \mathrm{relu}$ network.



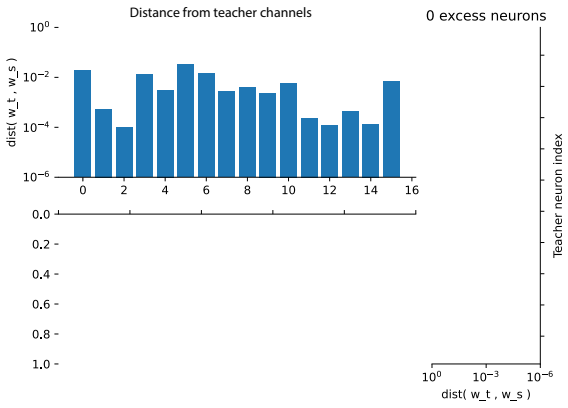


Figure A.16. **Alignment of channel weights between teacher and reconstructed student**: The figure shows the alignment of the different channel weights between the teacher and the reconstructed student for the convolutional network trained on the CIFAR10 dataset. The left plot is conv-layer 1, and the right plot is conv-layer 2.

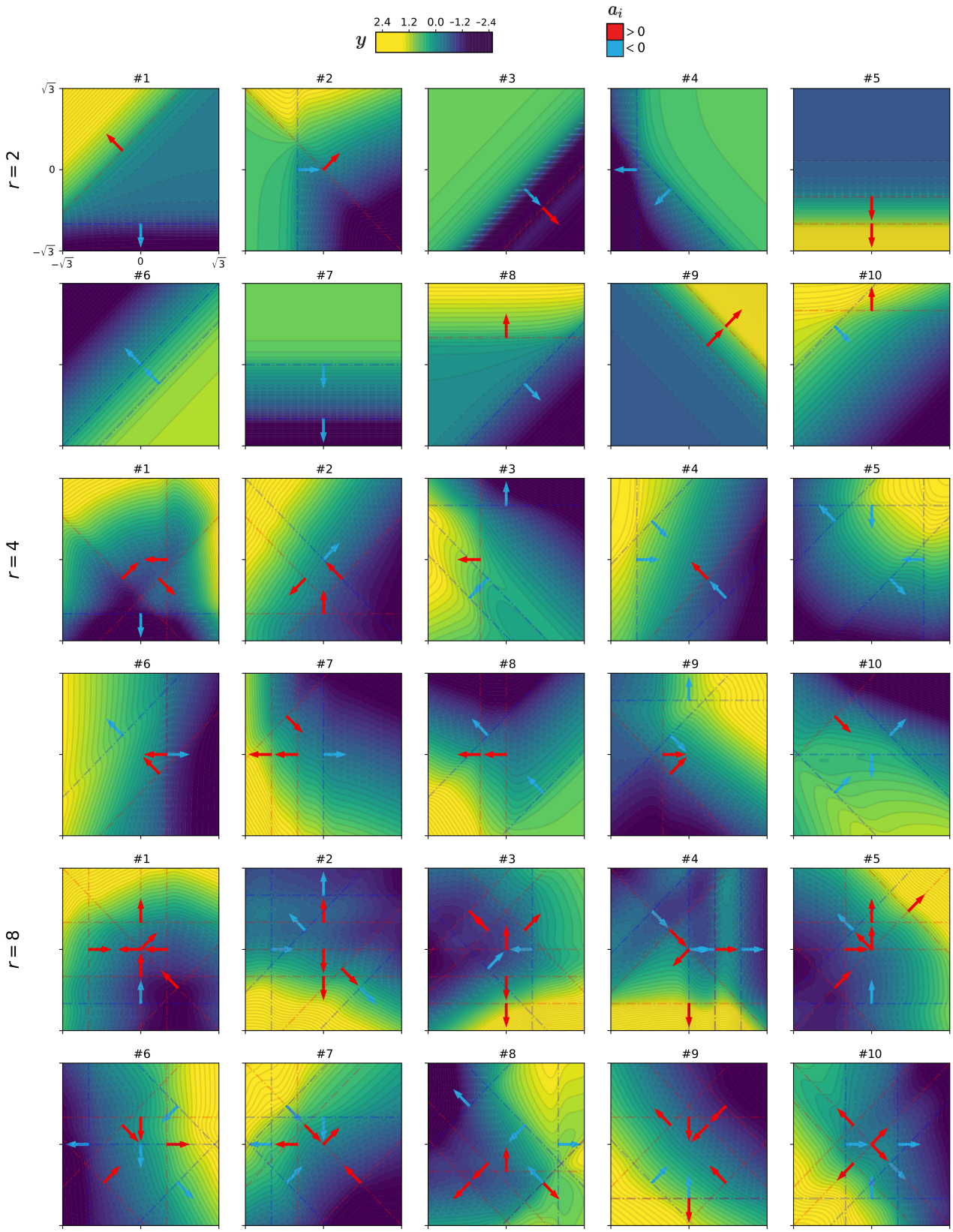


Figure A.17. **Visualization of all $\mathrm{d_{in}} = 2$ teachers used for results in Figure 4 and 5**