
Random Features for Normalization Layers

Anonymous Author(s)

Affiliation

Address

email

Abstract

Can we reduce the number of trainable parameters of neural networks by freezing a large portion of the initial weights? Training only BatchNorm parameters has shown great experimental promise, yet, the ability to express any potential target network would require a high amount of degrees of freedom. Even with sufficiently many parameters, contemporary optimization algorithms achieve only suboptimal performance. We systematically investigate both issues, expressiveness and trainability, and derive sparse random features which enjoy advantages in both aspects. In contrast to standard initialization approaches, they provably induce a well conditioned learning task and learning dynamics that are equivalent to the standard setting. They are also well aligned with target networks that can be approximated by random lottery tickets, which translates into a reduced bound on the number of required features. We obtain this bound by exploiting the layer-wise permutational invariance of target neurons, which applies to general feature distributions with good target alignment, and thus outline a path towards parameter efficient random features.

1 Introduction

Do we have to train all parameters of large neural network models or can we freeze a substantial portion at their random initial value? This question returns in different forms throughout the literature. Recent examples include [43, 17, 49]. Training batch normalization (BN) or layer normalization parameters has shown to be particularly effective during finetuning of large language models [3, 33, 28]. It is standard to include them in the set of trainable parameters during finetuning and they are seldomly pruned during neural network sparsification. They are considered important, even though they usually provide technically redundant parameters that do not contribute to the expressiveness of neural networks, since they can be integrated in the weight parameters. It has been conjectured that BN parameters do not only provide benefits for the optimization procedure by facilitating large learning rates, preconditioning [29], and promoting flatness [38], but also actively contribute to learning, as they find linear combinations of features [22, 8].

Affine transformations of features in general play a pivotal role in deep learning, as they are employed in most layer normalization techniques, pretrained model finetuning, or random feature models. Understanding the potential and limitations of linearly combining random features is central to the conjecture that we can reduce the number of trainable parameters of neural networks. [17] found that training BN layers only while keeping the remaining parameters frozen to their initial value showed great promise in this regard. The hope is that this could reduce the overall memory footprint of models, as random values can be reconstructed from a random seed and only a small number of parameters needs to be saved. Subsequently, [22, 8] have rigorously proven that training the parameters of normalization layers (including BN) is equally expressive as training the full model, provided that the number of trainable parameters induces a sufficient number of degrees of freedom.

The ability to represent an arbitrary target therefore demands a number of available random features that scales quadratically in the number of nodes, which inflicts substantial costs in practice. Furthermore, training BN parameters from scratch with frozen random weights could so far not achieve competitive performance in comparison to the classic parameterization that also trains weights. [17] posed therefore the question whether there could exist better suited random weight distributions that match to a smaller class of potential target networks.

Our answer is affirmative. By introducing maximally sparse, and optimally conditioned random features, we aim to address both open challenges, the trainability and high width requirement. By exploiting a permutational invariance of neural networks, we gain additional means to align a target network with given features, which is also a fundamental challenge in foundation model finetuning. The general question that we seek to answer is therefore this: How much target alignment is required and how much randomness or misalignment in the weights can we tolerate?

1.1 Contributions

By investigating the question whether some random features are more effective in supporting the expressiveness and trainability of affine linear transformations, we make the following contributions:

1. We identify two mechanisms that distinguish the suitability of random feature distributions: a) how well they align with a target network and b) the general conditioning of the learning task that they induce.
2. Standard weight initializations induce random features, which are ill conditioned for larger learning problems, as we verify in experiments. This can be partially remedied by overparameterization, but higher degrees are required for larger systems.
3. Based on these insights, we propose two non-standard random feature distributions that are maximally sparse and enjoy ideal conditioning, which is reflected in their advantageous training dynamics which are equivalent to the standard setting.
4. Exploiting permutational invariances of target neural networks, we reduce the number of random features that are required for their reconstruction.

Our results, however, have a major caveat. While the random features that we propose solve the two open problems of trainability and sparsity, which are required to gain something from freezing a part of the neural network parameters, the features that we propose induce a parameterization that is equivalent to the original weight parameters of the neural network. This implies that training (a random subset of) the weight space instead of the original BN space is, in fact, the best we can do. This insight highlights the relevance of additional knowledge about the task either in form of inductive bias guiding the random weight distribution, domain knowledge, or task related features of a foundation model. Also in case of task aligned features, our theoretical analysis quantifies the value of conditioning and the consideration of permutation invariance.

1.2 Related work

Benefits of Batch Normalization (BN) Batch Normalization (BN) is one of the first and most prominent normalization techniques of neurons in neural networks [26]. It introduces profound benefits for training speed and generalization, as it enables training with larger learning rates [4]. It makes the training success robust to different choices of parameter initializations [13, 27], and the loss landscape smoother [45]. While BN defines effectively an affine linear transformation of a neuron’s pre-activation, it also seems to simplify the learning task, as it tends to orthogonalize features (at least of linear neural networks) [12] and also acts as preconditioner [29]. Importantly, its conditioning operation applies to learning the weights of neural networks. In contrast, we study the conditioning of learning the parameters of the affine linear transformation on how different random feature distributions affect the learning task.

Alternatives to BN While highly effective, BN inflicts high computational costs and requires high amounts of memory, as it requires relatively high batch sizes to work well. Furthermore, it breaks the independence of minibatch samples and prevents adversarial training [48]. To alleviate some of these disadvantages, multiple alternatives have been proposed, including weight normalization [44, 25], weight standardization [41], instance normalization [47], instance enhancement batch normalization

[30], or switch normalization [34]. A combination of scaled weight standardization and gradient clipping could even outperform BN [5]. Our derivations for general affine transformations and different normalization mechanisms also cover these techniques.

Parameter initialization for trainability To reduce the need for BN, different initialization approaches that are tailored to specific architectures have been proposed [1, 9, 13, 50, 19]. A particularly successful approach have been orthogonal weight initializations [1, 9, 19] that generate dynamically isometric networks. Interestingly, we find that such orthogonal weight initializations induce suboptimal conditioning of the studied learning task, where affine linear transformations in two layers with random features approximate a standard target layer. In contrast, the random features that we derive enjoy ideal conditioning.

Training only BN If we commit to training only a small fraction of neural network parameters, experimental evidence suggests that BN parameters could be good candidates to be included in the set of training parameters. For instance, they have been found particularly effective in finetuning language models [3, 33]. Even if the neural network parameters are not pretrained but frozen to their initial values, training only the BN parameters was found to be more effective than training another random subset of neural network parameters while the remaining parameters were kept frozen [17]. This insights is supported by [43], which, however, has focused on relatively short training periods. Combining both lines of research, VeRA is a finetuning technique that learns a perturbation ΔW of a pretrained weight matrix by adding linear combinations of random weight matrices [28].

Expressiveness of affine linear transformations The experimental success of training only BN has been explained theoretically by [22] and [8] for fully-connected and convolutional architectures, respectively. They derive bounds on the number of random features that are required for an affine linear transformation to approximate a general target layer. While these bounds suggest that a high number of features are required to obtain strong function approximators of general targets, we show that good feature and target alignment can reduce the number of required random features. Furthermore, we address an overlooked conditioning issue that hampers the experimental applicability of the derived insights.

Random feature model Our setting can be interpreted as a deeper version of the random feature model [42], where features are generated by a single random neural network layer. Primarily, it has been employed to study fundamental effects of overparameterization and double descent [2].

Randomly masked neural networks To show that there exist random features that have lower width requirements than the theoretically predicted results that guaranty the reconstruction of general targets, we utilize the literature on lottery tickets [16, 15, 6, 7]. Randomly masked networks have shown surprisingly competitive performance on standard and large scale benchmark tasks [46, 36, 32, 18] up to sparsities of 90% and higher. This can be partially explained by width overparameterization [20], but also the existence of highly sparse representations of universal function approximators [10].

2 Expressiveness of affine transformations

Background and notation Freezing weights and learning only the parameters of normalization layers can be reduced to a layerwise student-teacher setup, as visualized in Fig. 1 (a) & (b). To keep the exposition simple, we focus here on linear, fully-connected layers. Our results generalize to convolutional and residual architectures, which can consist of more layers with different activation functions [8]. Each layer of a target network with weight parameters $\mathbf{W}^{(t)} \in \mathbb{R}^{c_2 \times c_0}$ is approximated by two student layers that keep their weights $\mathbf{W}^{(2)} \in \mathbb{R}^{c_2 \times c_1}$ and $\mathbf{W}^{(1)} \in \mathbb{R}^{c_1 \times c_0}$ frozen at their initial value and only adjust their normalization layer parameters $\gamma^{(2)}$ and $\gamma^{(1)}$ [22, 8]. (For notational clarity, we omit the layer index (l). Each layer is treated in the same manner.) The student thus has to minimize the loss

$$\mathcal{L}(\gamma^{(1)}, \gamma^{(2)}) = \sum_{ij} \left(\sum_k \gamma_i^{(2)} m_{(ij)k} \gamma_k^{(1)} - w_{ij}^{(t)} \right)^2, \quad (1)$$

where $\mathbf{M} = (m_{(ij)k})$ is a $(c_0 c_2) \times c_1$ -dimensional matrix with components $m_{(ij)k} = w_{ik}^{(2)} w_{kj}^{(1)}$. Note that details of different layer normalization approaches can be integrated into the parameters $\gamma^{(i)}$.

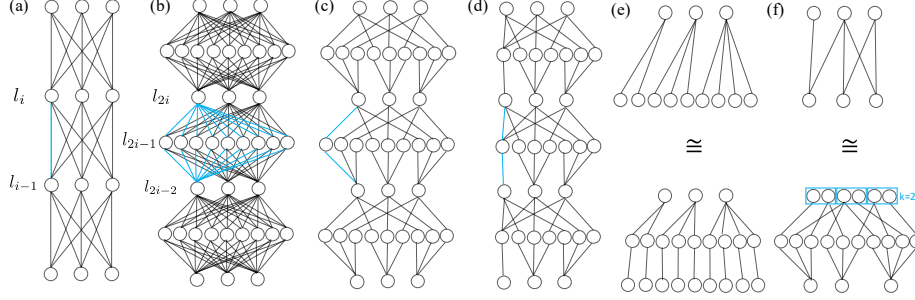


Figure 1: **Problem setup:** a) Target neural network. Each layer is approximated with two layers consisting of fixed (random) weights and trainable normalization layer parameters (b-d). The highlighted blue link is reconstructed utilizing the blue features in the other cases. (b) Most random features are dense. (c) Proposal of features (ID Cond) that induce a learning problem that is equivalent to (a). (d) A more parameter efficient variant of (c). (e) Target (top) and random features (bottom) for Thm. 4.2. (f) Target (top) and random features (bottom) for Thm. 4.3.

137 We have two options how we approach the reduction of this problem to linear regression. a)
 138 Eliminating $\gamma^{(2)}$ by replacing it as a function of $\gamma^{(1)}$ leads to the lowest established width requirement.
 139 b) Ignoring $\gamma^{(1)}$ (or setting all entries to 1), which leaves it free for utilization as normalization layer
 140 in practice, leaves the original structure of the matrix \mathbf{M} intact. It is easier to analyze and usually
 141 better conditioned than a). However, it leads to a worse bound on the number of required random
 142 features c_1 , since it does not utilize additionally available degrees of freedom. Our analysis generally
 143 applies to both scenarios and the conceptual differences are not significant.

144 **Width requirement for target reconstruction** In the following, we state the associated known
 145 results that provide a bound on c_1 to guarantee lossless reconstruction.

146 **Theorem 2.1** (Width requirement [22, 8]). *Problem (1) can be solved with $\mathcal{L}(\gamma^{(1)}, \gamma^{(2)}) = 0$
 147 if the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ have full rank and $c_1 \geq c_2 c_0$. The parameters $\gamma_i^{(2)} = 1$ and
 148 $\gamma^{(1)} = \mathbf{M}^+ \mathbf{v}$ define a solution, where \mathbf{M}^+ denotes the Moore-Penrose inverse and $v_{(ij)} = w_{(ij)}^{(t)}$
 149 is a flattened vector representation of the target. If we utilize $\gamma^{(2)}$ and $\gamma^{(1)}$ solves $\mathbf{M}\gamma^{(1)} = 0$ with
 150 $m_{(ij)k} = w_{i'k}^{(2)} w_{kj}^{(1)} - \frac{w_{ij}^{(t)}}{w_{ij_i}^{(t)}} w_{i'k}^{(2)} w_{kj_i}^{(1)}$, where i' corresponds to a pivotal element of the target row, then
 151 $c_1 \geq c_2(c_0 - 1) + 1$ features suffice.*

152 This theorem restates results by [22, 8]. The bound is far from encouraging, as the number of features
 153 c_1 scales quadratically with the number of target nodes. This is reasonable, since we need to attain the
 154 original number of degrees of freedom if we want to represent any possible target network. However,
 155 in practice, even this width requirement is not sufficient for standard random weight distributions, as
 156 the corresponding learning task tends to be badly conditioned, as we establish next.

157 2.1 Bad conditioning implies higher width requirement in practice

158 Training only BN parameters from scratch with frozen random weights could generally not recover the
 159 performance of a model that was trained with the original parameterization [17] despite the theoretical
 160 existence of a solution [8]. Training in a student-teacher setup using a target layer as a teacher could
 161 provide empirical evidence that it should be possible to train exclusively the normalization layers, but
 162 also this reconstruction performed well only with more features than the theoretical bound requests.
 163 The following section identifies the main source of the issue, namely, that the learning task is ill
 164 conditioned, as Fig. 2 (b) illustrates. Fig. 2 (c)&(d) furthermore explores the impact of different
 165 mitigation strategies that pre-condition the learning task. While they reduce the approximation error,
 166 their effect is limited and they do not scale to realistic neural network sizes. The spectrum of the
 167 random feature matrix \mathbf{M} for typical random weight distributions is the core of the problem.

168 **Standard random features** While the matrices $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ would be pre-trained in finetuning
 169 tasks, here, we aim to analyze how far we can get with random features that only encode little

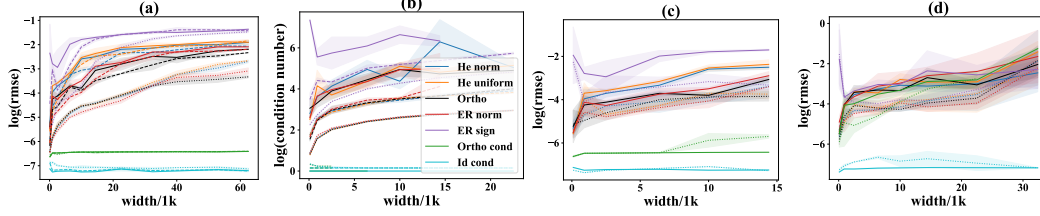


Figure 2: **Conditioning:** (a) Log_{10} of the root mean squared error (rmse) obtained by LBFSGS reconstructing random targets and random features following the same distribution. Solid lines: The exact number of features that are required for perfect reconstruction are provided. Dashed line: 10 additional features. Dotted line: 100 additional features. (b) Condition number of the respective random features. It could not be identified due to numerical instabilities if it is not shown. (c) As in (a) but a singular value decomposition of \mathbf{M} is first used to precondition M and construct a good initialization of LBFSGS. (d) As (c), but after conditioning we utilize the Pytorch linear algebra solver *torch.linalg.solve* if possible and then finetune the solution with LBFSGS.

information about the potential target. [17, 8] had primarily investigated freezing the weights to their initial value and therefore considered the popular He [23] and orthogonal [24] initializations so that $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are independent. In addition, we also study sparse variants that are randomly masked (with probability 1/2) and thus have random iid normally or binary $w_{ij} \in \{-1, 1\}$ distributed entries, which are labeled ER norm or ER sign, respectively.

Spectral properties The singular values S of the appropriately scaled feature matrix \mathbf{M} are a crucial factor in determining the hardness of the optimization problem that we try to solve, which is commonly measured by the condition number $\kappa(\mathbf{M}) = \bar{s}/\underline{s}$, i.e. the quotient of the maximum singular value \bar{s} and the minimum nonzero singular value \underline{s} . The singular values follow approximately the Wigner’s semi-circle law (see Fig. 3) with probability density

$$p(x) = \frac{1}{\lambda\pi x} \sqrt{(x^2 - \lambda_-^2)(\lambda_+^2 - x^2)}$$

with $\lambda = \max((c_2c_0)/c_1, c_1/(c_2c_0))$, $\lambda_- = 1 - \sqrt{\lambda}$, and $\lambda_+^2 = 1 + \sqrt{\lambda}$ and $x \in [\lambda_-, \lambda_+]$. This follows from the fact that \mathbf{M} is a Whishart matrix, i.e. its entries are iid distributed with mean 0, because all $w_{ik}^{(2)}$ and $w_{kj}^{(1)}$ are independent with mean 0. It is well known that the eigenvalue distribution of $\mathbf{M}\mathbf{M}^T/\sigma$ converges asymptotically to the Marchenko-Pastur distribution [37] (where σ denotes an appropriate scaling so that $w_{ik}^{(2)}w_{kj}^{(1)}/\sigma$ has variance $1/(c_0c_2)$). Interestingly, if $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are normally distributed, the resulting feature matrix \mathbf{M} tends to have smaller singular values and thus worse conditioning in comparison with uniformly distributed $\mathbf{W}^{(i)}$.

Double Descent The biggest problem occurs exactly at the point of the minimum width requirement, where the linear system of equations becomes solvable, i.e., the interpolation point, where $\lambda = 1$ so that $\lambda_-^2 = 0$. The minimum singular value can get arbitrarily close to 0, which induces a high condition number. Learning components in corresponding singular directions becomes hard. On average, the corresponding condition number further increases for larger systems, which is associated with more erroneous target reconstruction (see Fig. 2). Yet, it goes through a double descent [40], which means that it peaks at the interpolation threshold.

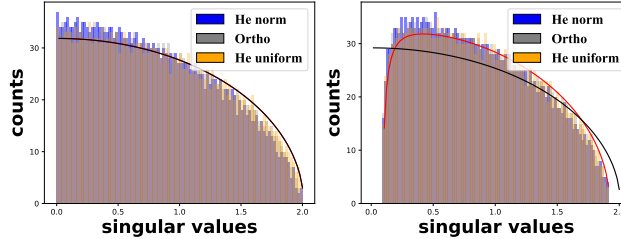


Figure 3: **Spectrum:** Histograms of the singular values of a random matrix M with dimensions $c_0 = c_2 = 50$ and $c_1 = 2500$ (left) or $c_1 = 3000$ (right). The black line marks the semi-circle law with scale $\lambda = 1$ and the red one with scale $\lambda = (c_0c_2)/c_1$, which adjusts for the excess features.

More or fewer features (s.t. $\lambda > 1$) make the problem algorithmically better solveable. However, to fight problematic conditioning, the number of excess or missing features must scale with the target dimension $c_0 c_2$, which inflicts potentially high computational and memory costs or hampers the reconstruction. Our declared goal is to reduce the number of required features by taking the properties of the target into account. Yet, leaving out too many (random) features still reduces the expressive power of a layer and thus limit its capacity. For that reason, we need additional ways to deal with bad conditioning. The random features, which we propose, enjoy perfect conditioning by design.

Orthogonal weights are not well conditioned. Note that orthogonal weights are considered to be an excellent choice for neural network initialization, because each matrix W separately has perfect condition number 1 (and thus contributes to dynamical isometry [1, 9, 19, 39]). Perhaps surprisingly, this does not transfer to \mathbf{M} and thus to task of learning normalization parameters. While the columns of \mathbf{M} are orthogonal, the rows are not, as $\sum_k w_{ik}^{(2)} w_{kj}^{(1)} w_{i'k}^{(2)} w_{kj'}^{(1)} \neq 0$, in general.

3 Advantageous random features

One of our main contributions is the derivation of random features that are specifically designed to support learning normalization layers, as they are perfectly conditioned and provably induce well behaved learning dynamics.

ID cond features The ID cond features are visualized in Fig. 1 (c) & (d). Let us first assume that \mathbf{M} has width $c_1 = c_0 c_2$. Every feature k can then be associated with a target dimension (i, j) so that

$$m_{(ij)k} := 1, \text{ if } k = k_{ij}, \text{ and } m_{(ij)k} := 0 \text{ otherwise.} \quad (2)$$

Without loss of generality, we assume that the indices k are arranged so that the feature matrix becomes the identity matrix $\mathbf{M} = \mathbf{I}$. To achieve the minimum width requirement, however, we have to remove $c_2 - 1$ features as illustrated in Fig. 1 (d) and utilize $\gamma^{(2)}$. For this case, we define:

$$m_{(ij)k} := 1, \text{ if } k = k_{ij}, \text{ and } m_{(ij)k} := 0 \text{ otherwise for } j > 1, \text{ } m_{(i1)k} := 1 \text{ for } k = k_1. \quad (3)$$

This matrix cannot be perfectly conditioned, but the solution of the target reconstruction is still straight forward: $w_{ij}^t = \gamma_{k_i}^{(2)} \gamma_{k_{ij}}^{(1)}$ for $j > 1$ and $w_{i1}^t = \gamma_{k_i}^{(2)} \gamma_{k_1}^{(1)}$. Note that $\gamma_{k_1}^{(1)}$ is a free parameter that results from the scale invariance of consecutive neural network layers.

Ortho cond features Following a similar principle, i.e., disentangling orthogonal dimensions, we can also define more general orthogonal features, where $m_{(ij)k} = u_{iq}^{(2)} u_{dj}^{(1)}$ with $k = (qd)$. $\mathbf{U}^{(2)}$ and $\mathbf{U}^{(1)}$ are random, quadratic orthogonal matrices. ID cond can be seen as special case where the orthogonal matrices are identity matrices. In our experiments, we draw random Ortho cond features by sampling $\mathbf{U}^{(1)} \sim O(c_0)$, $\mathbf{U}^{(2)} \sim O(c_2)$ independently from the respective orthogonal ensemble.

Ortho cond features are perfectly conditioned. Our proposed features are perfectly conditioned by design, since the corresponding feature matrix \mathbf{M} is orthogonal. This is easy to see, since the rows inherit the orthogonality from their components so that $\sum_{(ij),(qd)} m_{(i'j')(qd)} m_{(ij)(qd)} = \sum_q u_{iq}^{(2)} u_{i'q}^{(1)} \sum_d u_{dj}^{(2)} u_{dj'}^{(1)} = \delta_{ii'} \delta_{jj'}$, where δ denotes the Kronecker delta. This also holds for ID cond features, which are a special, maximally sparse case.

Ortho cond features in general are ideal for the learning task to reconstruct targets layerwise in a student-teacher setting. In practice though, we would want to train all layers simultaneously. As it turns out, ID cond features overcome also the challenge of ill-conditioned training from scratch, as we can achieve the same performance as training in the original weight space.

Theorem 3.1 (Equivalent Features). *Learning with ID cond features (2) is equivalent to learning a neural network in its original parameterization with $w_{ij}^{(t)} = \gamma_{k_{ij}}^{(1)}$. The corresponding feature matrix $\mathbf{M} = \mathbf{I}$ is maximally sparse and perfectly conditioned.*

The proof is given in the appendix. ID cond features induce not only advantages for optimization, they are also maximally sparse, which makes them attractive from a computational point of view. Yet, so far we have nothing gained compared to training just the original weights. Next, we investigate under which conditions we can reduce the theoretical width requirement.

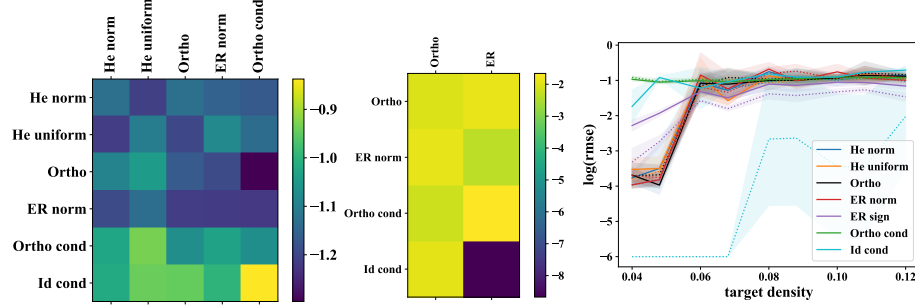


Figure 4: **Alignment after permutation with low rank targets:** The target matrix is drawn randomly as $\mathbf{W}_2^{(t)} \mathbf{W}_1^{(t)}$, where each \mathbf{W}_i follows the column distribution. The target dimensions are $(c_0, c_2) = (50, 50)$ with hidden dimension $c_1^{(t)}$. Similarly, the matrices comprising the feature matrix \mathbf{M} follow the row distribution, where the number of random features c_1 is lower than the width requirement. The average reconstruction error (over 10 independent samples) after Greedily permuting the target is color coded. Left: $c_1^{(t)} = 100$, $c_1 = 2440$. Middle: $c_1^{(t)} = 300$, $c_1 = 2400$. Right: The Bernoulli $Ber(p)$ iid mask, where p is the expected target density. $c_1 = 2400$. Solid lines: Reconstruction error without permutations. Dotted line: Permuting target neurons for better match with random features. Mean and standard errors are computed for 10 seeds.

245 4 Permutation invariance

246 As it turns out, the singular value space of the feature matrix \mathbf{M} determines not only the numerical
 247 difficulty of our optimization problem of interest, it also is key to answering the question whether we
 248 can further reduce the theoretical width requirement by exploiting the so far disregarded permutation
 249 invariance of a target. As implication, our target reconstruction problem has another degree of
 250 freedom that we can optimize, i.e. permutation matrices \mathbf{P} that permute the rows of the target matrix.
 251 Accordingly, we aim to minimize

$$\mathcal{L}(\gamma^{(1)}, \gamma^{(2)}, \mathbf{P}) = \left\| \mathbf{M}(\mathbf{P})\gamma^{(1)} - \text{vec}(\mathbf{P}\mathbf{W}^{(t)}) \right\|_2^2. \quad (4)$$

252 In the following, we study the opportunities resulting from optimizing \mathbf{P} in a target specific manner.
 253 Our guiding question will be whether we can reduce the width requirement and make learning only
 254 normalization layers practically feasible.

255 **Can we leverage properties of the target?** The next theorem derives a criterion that we can
 256 optimize to find permutations that minimize the mean squared error with respect to the target if we
 257 have only r features. Based on this result, we have defined a Greedy algorithm (see appendix) that
 258 seeks to maximize the alignment of features and targets in Fig. 4.

259 **Theorem 4.1** (Target Alignment). *Assume that \mathbf{M} has rank r and singular value decomposition*
 260 *$\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ with singular values sorted in decreasing order. Let \mathbf{P} be a permutation of the*
 261 *rows of the target $\mathbf{W}^{(t)}$ and $\mathbf{v}^{(t)}(\mathbf{P}) = \text{vec}(\mathbf{P}\mathbf{W}^{(t)})$ be the flattened vector representation of the*
 262 *permuted target. Then, the approximation error is minimized by $\min_{\mathbf{P}} \min_{\gamma} \left\| \mathbf{M}\hat{\gamma} - \mathbf{v}^{(t)}(\mathbf{P}) \right\|^2 =$*
 263 *$\min_{\mathbf{P}} \left\| \mathbf{P}_r \mathbf{U}^T \mathbf{v}^{(t)}(\mathbf{P}) \right\|^2$, where $\mathbf{P}_r = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \end{bmatrix}$ projects a vector to its last r components.*

264 The proof is given in the appendix. It seems intuitive that the permutation seeks to align the target
 265 with the leading singular vectors of the features \mathbf{M} and thus minimize its variation in the directions
 266 that cannot be covered by linear combination of the columns of \mathbf{M} . Accordingly, the alignment
 267 between the target and features \mathbf{M} in the singular value space defines the critical measure of the
 268 remaining approximation error $\left\| \mathbf{P}_r \mathbf{U}^T \mathbf{v}^{(t)} \right\|^2$.

269 **Target alignment** Note that the above statement applies to any feature matrix \mathbf{M} . It does not assume
 270 that \mathbf{M} is random but also applies to pre-trained features. Assuming such high alignment lies at the
 271 heart of many fine-tuning techniques. The expressiveness of the approach can be increased by also
 272 updating the singular vectors, which can aid generalization if the alignment of the pre-trained weights
 273 and the target weights is not sufficient [31].

Alignment with random targets How many degrees of freedom can we effectively gain by permuting output neurons for target alignment? If our target has a low rank or is sparse, matching should be easier and reduce our width requirement. In light of our insight that the reconstruction success hinges on target and feature alignment, we could even hypothesize that it could help if a random target that is drawn independently from the features still follows the same probability distribution and thus shares the inductive bias of the features.

Inductive bias from similar distributions Fig. 4 suggests that the permutations obtained by our Greedy matching algorithm hardly reduce our width requirement. Only highly sparse targets can be reconstructed with fewer sparse ID cond features. However, ID cond features seem less suitable than other random features to exploit low rankness of dense targets. Apart from sparsity, we do not observe strong positive associations between similar target and feature distributions. The standard dense features follow approximately Wigner’s semicircle law of Gaussian matrices though. Thus, they all align with similarly distributed targets. Visually, they are distinguishable from the two better conditioned feature distributions, which seem to match similarly distributed targets.

Sparse targets Sparse targets and features, however, present more promising candidates to realize width reductions. Fig. 4 (right) shows that permutations that improve the target alignment (see appendix for matching algorithm) can make reconstruction possible at a feature width c_1 that is smaller than our previous bound. However, the target density at which permutations help effectively is disappointingly low. Can we understand why this is the case?

Theorem 4.2 (Matching sparse targets and features). *Let the target matrix $\mathbf{W}^{(t)}$ be a sparse matrix with in-degrees $d_i = \sum_j \delta_{w_{ij}^{(t)} \neq 0}$ and the feature matrix \mathbf{M} correspond to a random $\text{Ber}(p)$ matrix (i.e. sparse ID cond features). Then, the probability that the (permuted) target can be accurately matched is upper bounded by $\mathbb{P}(\mathbf{W}^{(t)} \subset \mathbf{M}) \leq \Pi_i (1 - (1 - p^{d_i})^{c_2})$.*

Proof Outline. The proof is provided in the appendix. The main idea is to consider a special case that attains the lower bound. If all target output neurons are connected to mutually exclusive sets of input neurons (see Fig. 1 (e)), then all student outputs are potential matches for a target neuron, which maximizes the probability of a match.

This result explains, why permutations can only improve reconstruction performance at relatively low target density and high feature density (see Fig. 4). To see this, let us assume for simplicity that all target neurons have the same degree $d_i = d$. Then, Theorem 4.2 implies that we would need at least a density of order $1 - (\delta/c_2)^{1/c_2}$ to represent a target with dc_2 edges with success probability $1 - \delta$. For reference, a target with $c_2 = c_0 = 50$ and $d = 10$ (and thus density 0.2) would require more than $p = 0.8$ feature density for a significant chance ($1 - \delta > 0.8$) to reconstruct the target. Our target would therefore have to be very sparse (i.e. have small d) to allow for a significant reduction in the feature width (i.e. small c_1 or p). This conclusion is in line with Fig. 4 (right), where maximizing the target alignment by permuting output neurons affects the reconstruction loss positively only when the target has a low density.

Could we still succeed with fewer random features? Theorem 4.2 has highlighted a considerable limitation of learning based on sparse random features (i.e. small density p). Random directions in high-dimensional spaces are unlikely to align with unknown targets. And, yet, the unreasonable effectiveness of random pruning calls this into question [32, 20]. Related lottery ticket existence proofs that try to explain this fact postulate the existence of much sparser targets of lower width [15, 11, 6, 7] and smaller number of layers [7, 14]. Making c_2 wider than the target in our construction, would indeed provide many more permutation options, as we prove with the next theorem.

Theorem 4.3 (Lower dimensional target). *A random target with iid $\text{Ber}(q)$ entries and $c_2^{(t)}$ output neurons can be perfectly matched with probability $1 - \delta$ with random ID cond features with expected density p if $c_2 = kc_2^{(t)}$ with $k \geq \frac{\log(1 - (1 - \delta)^{1/c_2^{(t)}})}{\log(1 - (1 - q(1 - p))^{c_0})}$.*

The proof is given in the appendix and the setup is visualized in Fig. 1 (f). Note that the special target in Fig. 1 (e) would only need $k \approx (1 - \delta)^{1/c_2} / (c_2 p^d)$. In practice, [17] found that approximately 1/3 of activations are switched off when only BN parameters are learned, suggesting a $k \approx 1.5$. As we have established that sparse targets can be matched under certain conditions in support of empirical evidence showing that random masks work in practice [32, 18] and we have established that ID cond features are provably trainable from scratch, we conclude the following.

cond. num.	1	5413	10123	1541601
train acc	76.89	75.84	75.6	73.14
test acc	81.89	79.06	79.1	76.28

Table 1: ResNet50 trained on ImageNet with ID features scaled by a spectrum that was drawn from a semi-circle law with the reported average condition number assuming $c_0 c_2 = 10^6$. Note that many ResNet50 layers are larger are therefore worse conditioned.

Conclusion. If a task is solvable by training an iid randomly masked neural network with density p_l , then $c_1^l = p_l^l c_0^l c_2^l$ random ID cond features can achieve the same generalization performance.

In practice, $p_l^l \approx 0.2$ is often feasible without significant performance loss. This concludes our theory, as we have established that there exist random sparse features that support learning only normalization layers and these features are perfectly conditioned.

5 Experiments

Our experiments have identified bad conditioning as major obstacle in approximating given targets in a student teacher setting (see Fig. 2). Mitigating this issue by preconditioning does not scale to typical neural network sizes and is not as effective as using our well conditioned features ID cond and Ortho cond. Furthermore, we have permuted targets to align with our features to improve the reconstruction error (see Fig. 4) based on Thm. 4.2, which solidifies our theoretical insight that permutations make a relevant difference, yet, primarily for matching extremely sparse targets. In addition, we next explore the implications of conditioning for training from scratch.

CIFAR10 To resemble our theoretical setup, we replace the convolutional and fully connected layers of a Vgg18 with width 100 by two linear random He normally distributed weight layers for which we train linear combinations of the intermediary neurons on CIFAR10. If we train the network with intermediary width $c_1 = 9 * 100 * 100 + 100$ with 100 excesss features (without warmup) for 300 epochs, we obtain an accuracy of $(86.64 \pm 0.08)\%$ (averaged over 3 independent runs). Training longer for 500 epochs with warmup to combat problematic conditioning increases the accuracy $(90.08 \pm 0.03)\%$. Despite using excess features, it cannot compete with training the network in the original parameterization, which can achieve an accuracy in the order of 94%. These experiments are limited to small scale settings, as the required width of the middle random feature layer is substantial.

ImageNet To study the effects of bad conditioning in larger scale settings, where they should be even more detrimental, we study rescaled ID precond features with arbitrary singular values $s_{(ij)}$, which correspond to a parameterization $w_{ij} s_{ij}$ where s_{ij} is a fixed scaling and w_{ij} is trainable. Table 1 reports the results for spectra that resemble the classic random distributions (which approximately follow the semi-circle law). As expected, ill conditioning affects the training outcome negatively. The original parameterization corresponds to the ideally conditioned case.

6 Discussion

We have affirmatively addressed the open question posed by [17] regarding the existence of random weight distributions that are more conducive than others to training only normalization layers, while keeping all other parameters fixed. While standard random weight initializations often suffer from poor conditioning, artificially increasing the number of features—though impractical—can improve the optimization landscape. Conversely, reducing features also improves conditioning but at the cost of significantly reducing model expressiveness. To overcome these obstacles, we have introduced random weight distributions that are perfectly conditioned, maximally sparse, and provably trainable from scratch. They can also lower the theoretical width requirement. However, they are mathematically equivalent to the original parameterization, implying that training normalization layers alone with random weights [17, 8] does not yield performance gains over standard training. Nevertheless, our analysis suggests that fine-tuning normalization layers in pre-trained foundation models can enable efficient learning when the target task aligns well with the model’s singular value structure. These findings provide fundamental insights into how the structure of weight matrices influences the effectiveness of normalization-layer training, and could inform future approaches to jointly optimizing weights and normalization parameters more effectively.

References

- [1] David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pages 342–350, 2017.
- [2] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 2019.
- [3] Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. In *Annual Meeting of the Association for Computational Linguistics*, 2022.
- [4] Nils Bjorck, Carla P Gomes, Bart Selman, and Kilian Q Weinberger. Understanding batch normalization. In *Advances in Neural Information Processing Systems*, volume 31, 2018.
- [5] Andy Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. In *International Conference on Machine Learning*, pages 1059–1071, 2021.
- [6] Rebekka Burkholz. Convolutional and residual networks provably contain lottery tickets. In *International Conference on Machine Learning*, 2022.
- [7] Rebekka Burkholz. Most activation functions can win the lottery without excessive depth. In *Advances in Neural Information Processing Systems*, 2022.
- [8] Rebekka Burkholz. Batch normalization is sufficient for universal function approximation in CNNs. In *International Conference on Learning Representations*, 2024.
- [9] Rebekka Burkholz and Alina Dubatovka. Initialization of ReLUs for dynamical isometry. In *Advances in Neural Information Processing Systems*, volume 32, 2019.
- [10] Rebekka Burkholz, Nilanjana Laha, Rajarshi Mukherjee, and Alkis Gotovos. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022.
- [11] Arthur da Cunha, Emanuele Natale, and Laurent Viennot. Proving the lottery ticket hypothesis for convolutional neural networks. In *International Conference on Learning Representations*, 2022.
- [12] Hadi Daneshmand, Amir Joudaki, and Francis Bach. Batch normalization orthogonalizes representations in deep random networks. In *Advances in Neural Information Processing Systems*, volume 34, pages 4896–4906, 2021.
- [13] Soham De and Sam Smith. Batch normalization biases residual blocks towards the identity function in deep networks. *Advances in Neural Information Processing Systems*, 33:19964–19975, 2020.
- [14] Jonas Fischer and Rebekka Burkholz. Plant ‘n’ seek: Can you find the winning ticket?, 2021.
- [15] Jonas Fischer and Rebekka Burkholz. Towards strong pruning for lottery tickets with non-zero biases, 2021.
- [16] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*, 2019.
- [17] Jonathan Frankle, David J. Schwab, and Ari S. Morcos. Training BatchNorm and only Batch-Norm: On the expressive power of random features in CNNs. In *International Conference on Learning Representations*, 2021.
- [18] Advait Harshal Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. In *International Conference on Learning Representations*, 2024.

- [19] Advait Harshal Gadhikar and Rebekka Burkholz. Dynamical isometry for residual networks, 2022.
- [20] Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why random pruning is all we need to start sparse. In *International Conference on Machine Learning*, 2023.
- [21] Advait Harshal Gadhikar, Sohom Mukherjee, and Rebekka Burkholz. Why Random Pruning Is All We Need to Start Sparse. In *International Conference on Machine Learning*, 2023.
- [22] Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the normalization layers. In *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195, 2023.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [24] Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks. In *International Conference on Learning Representations*, 2020.
- [25] Lei Huang, Xianglong Liu, Yang Liu, Bo Lang, and Dacheng Tao. Centered weight normalization in accelerating training of deep neural networks. In *IEEE International Conference on Computer Vision (ICCV)*, 2017.
- [26] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456, 2015.
- [27] Amir Joudaki, Hadi Daneshmand, and Francis Bach. On bridging the gap between mean field and finite width deep random multilayer perceptron with batch normalization. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 15388–15400, 2023.
- [28] Dawid Jan Kopiczko, Tijmen Blankevoort, and Yuki M Asano. VeRA: Vector-based random matrix adaptation. In *International Conference on Learning Representations*, 2024.
- [29] Susanna Lange, Kyle Helfrich, and Qiang Ye. Batch normalization preconditioning for neural network training. *Journal of Machine Learning Research*, 23(72):1–41, 2022.
- [30] Senwei Liang, Zhongzhan Huang, Mingfu Liang, and Haizhao Yang. Instance enhancement batch normalization: An adaptive regulator of batch noise. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04), 2020.
- [31] Vijay Lingam, Atula Tejaswi Neerkaje, Aditya Vavre, Aneesh Shetty, Gautham Krishna Gudur, Joydeep Ghosh, Eunsol Choi, Alex Dimakis, Aleksandar Bojchevski, and sujay sanghavi. SVFT: Parameter-efficient fine-tuning with singular vectors. In *Neural Information Processing Systems*, 2024.
- [32] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Li Shen, Decebal Constantin Mocanu, Zhangyang Wang, and Mykola Pechenizkiy. The unreasonable effectiveness of random pruning: Return of the most naive baseline for sparse training. In *International Conference on Learning Representations*, 2021.
- [33] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Frozen pretrained transformers as universal computation engines. *AAAI Conference on Artificial Intelligence*, 36(7), 2022.
- [34] Ping Luo, Jiamin Ren, Zhanglin Peng, Ruimao Zhang, and Jingyu Li. Differentiable learning-to-normalize via switchable normalization. In *International Conference on Learning Representations*, 2019.
- [35] Jiancheng Lyu, Shuai Zhang, Yingyong Qi, and Jack Xin. Autosufflenet: Learning permutation matrices via an exact lipschitz continuous penalty in deep convolutional neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD ’20, page 608–616, 2020.

- [36] Xiaolong Ma, Geng Yuan, Xuan Shen, Tianlong Chen, Xuxi Chen, Xiaohan Chen, Ning Liu, Minghai Qin, Sijia Liu, Zhangyang Wang, and Yanzhi Wang. Sanity checks for lottery tickets: Does your winning ticket really win the jackpot? In *Advances in Neural Information Processing Systems*, 2021.
- [37] VA Marčenko and LA Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4), 1967.
- [38] Maximilian Mueller, Tiffany Joyce Vlaar, David Rolnick, and Matthias Hein. Normalization layers are all that sharpness-aware minimization needs. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.
- [39] Aleksandra Nowak, Łukasz Gniecki, Filip Szatkowski, and Jacek Tabor. Sparser, better, deeper, stronger: Improving static sparse training with exact orthogonal initialization. In *International Conference on Machine Learning*, volume 235, pages 38474–38494, 2024.
- [40] Tomaso Poggio, Gil Kur, and Andrzej Banburski. Double descent in the condition number, 2020.
- [41] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.
- [42] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.
- [43] A. Rosenfeld and J. K. Tsotsos. Intriguing properties of randomly weighted networks: Generalizing while learning next to nothing. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 9–16, 2019.
- [44] Tim Salimans and Durk P Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *Advances in neural information processing systems*, 29, 2016.
- [45] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.
- [46] Jingtong Su, Yihang Chen, Tianle Cai, Tianhao Wu, Ruiqi Gao, Liwei Wang, and Jason D Lee. Sanity-checking pruning methods: Random tickets can win the jackpot. In *Advances in Neural Information Processing Systems*, 2020.
- [47] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016.
- [48] Haotao Wang, Aston Zhang, Shuai Zheng, Xingjian Shi, Mu Li, and Zhangyang Wang. Removing batch normalization boosts adversarial training. In *International Conference on Machine Learning*, pages 23433–23445. PMLR, 2022.
- [49] Ezekiel Williams, Avery Hee-Woon Ryoo, Thomas Jiralerspong, Alexandre Payeur, Matthew G. Perich, Luca Mazzucato, and Guillaume Lajoie. Expressivity of neural networks with random weights and learned biases, 2024.
- [50] Hongyi Zhang, Yann N Dauphin, and Tengyu Ma. Fixup initialization: Residual learning without normalization. In *International Conference on Learning Representations*, 2018.

A Technical Appendices and Supplementary Material

A.1 Proofs of Theorems

A.1.1 Proof of Theorem 3.1

Statement (Equivalent Features, Thm. 3.1). Learning with ID cond features (2) is equivalent to learning a neural network in its original parameterization with $w_{ij}^{(t)} = \gamma_{k_{ij}}^{(1)}$. The corresponding feature matrix $\mathbf{M} = \mathbf{I}$ is maximally sparse and perfectly conditioned.

Proof. Our first objective is to show that ID cond features lead to an equivalent parameterization $w_{ij}^{(t)} = \gamma_{k_{ij}}^{(1)}$. This follows directly from the construction, where we have to track the indices from the operation that flattens matrices to vectors. For instance, concatenating the rows of the target matrix $w_{ij}^{(t)}$ yields an index $k_{ij} = (i-1)c_0 + j$ for the original index pair (ij) . Setting

$$m_{(ij)k} := 1, \text{ if } k = k_{ij}, \text{ and } m_{(ij)k} := 0 \text{ otherwise,} \quad (5)$$

results in a feature matrix that equals the identity matrix $\mathbf{M} = \mathbf{I}$, which is obviously perfectly conditioned. $\mathbf{M}\gamma = \text{vec}(\mathbf{W}^{(t)})$ implies then $w_{ij}^{(t)} = \gamma_{k_{ij}}^{(1)}$.

The open question is how this feature matrix can be generated by $\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$. Fig. 1 (c) visualizes the construction. Exactly one path leads from each input neuron j to each output neuron i so that the corresponding $\gamma_{k_{ij}}^{(1)}$ can represent the target weight $w_{ij}^{(t)}$. Accordingly, we define

$$w_{kj}^{(1)} = 1 \text{ and } w_{ki}^{(2)} = 1, \text{ if } k = k_{ij}, \text{ and } w_{kj}^{(1)} = w_{ki}^{(2)} = 0 \text{ otherwise.} \quad (6)$$

$\mathbf{W}^{(1)}$ and $\mathbf{W}^{(2)}$ are maximally sparse, as each feature index k is connected to exactly one input (in $\mathbf{W}^{(1)}$) or one output (in $\mathbf{W}^{(2)}$). Removing such a link would render the feature useless and thus correspond to an effective removal of the feature. For that reason, 1 is a lower bound for row sum and column sum of , respectively. This bound is attained by the above construction. Furthermore, exactly $c_1 = c_0 c_2$ features are provided, which corresponds to the degrees of freedom of a potential target matrix. This number of features is necessary to express any target matrix $\mathbf{W}^{(t)} \in \mathbb{R}^{c_2 \times c_0}$. In conclusion, ID cond features are maximally sparse.

Equivalent dynamics are introduced if the second layer with trainable $\gamma^{(2)}$ assume the typical role of a normalization layer (e.g. BN), while $\gamma^{(2)}$ encodes the linear target layer. Yet, the second layer is potentially equipped with a nonlinear activation function. This ensures that the loss function in the original parameterization $\mathcal{L}(\mathbf{W}^{(t)})$ is identical to the loss in the parameterization with random features so that $\mathcal{L}(\mathbf{W}^{(t)}) = \mathcal{L}(\gamma^{(1)})$ for each target layer. It follows that also $\nabla \mathcal{L}(\mathbf{W}^{(t)}) = \nabla \mathcal{L}(\gamma^{(1)})$. \square

ID cond features that attain the minimum width requirement by utilizing $\gamma^{(2)}$ in addition to $\gamma^{(1)}$ create only one representative intermediary neuron for one of the input neurons (see Fig. 1 (d)). The corresponding weight matrices are defined as:

$$w_{kj}^{(1)} = 1 \text{ and } w_{ki}^{(2)} = 1, \text{ if } k = k_{ij} \text{ for } i > 1. w_{k1}^{(1)} = 1 \text{ and } w_{ik}^{(2)} = 1 \text{ if } k = k_{11} \text{ for } j = 1 \text{ and all } i.$$

and $w_{kj}^{(1)} = w_{ki}^{(2)} = 0$ otherwise. Note that we skip the feature indices k_{i1} for $i > 1$. The corresponding training dynamics change due to the quadratic nature of the parameterization. Yet, solving the student-teacher setup for this parameterization is still straight-forward because of the high sparsity of the features and known solution. In comparison, Ortho cond features, which are generally not sparse, lead to suboptimal reconstruction error in comparison, even though they are also well conditioned.

A.1.2 Proof of Theorem 4.1

Statement (Target Alignment, Thm. 4.1). Assume that \mathbf{M} has rank r and singular value decomposition $\mathbf{M} = \mathbf{USV}^T$ with singular values sorted in decreasing order. Let \mathbf{P} be a permutation of the rows of the target $\mathbf{W}^{(t)}$ and $\mathbf{v}^{(t)}(\mathbf{P}) = \text{vec}(\mathbf{PW}^{(t)})$ be the flattened vector representation of the

545 permuted target. Then, the approximation error is minimized by $\min_{\mathbf{P}} \min_{\gamma} \|\mathbf{M}\gamma - \mathbf{v}^{(t)}(\mathbf{P})\|^2 =$
 546 $\min_{\mathbf{P}} \|\mathbf{P}_r \mathbf{U}^T \mathbf{v}^{(t)}(\mathbf{P})\|^2$, where $\mathbf{P}_r = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \end{bmatrix}$ projects a vector to its last r components.

547 *Proof.* In contrast to the previous theorem, we assume that we might not have sufficiently many
 548 degrees of freedom to attain perfect reconstruction. Hence, our objective is to minimize the recon-
 549 struction error using all our options, including learning the $\gamma^{(1)}$ parameters and permuting the target
 550 neurons to align the random features with the target:

$$\min_{\mathbf{P}} \min_{\gamma} \|\mathbf{M}\gamma - \mathbf{v}^{(t)}(\mathbf{P})\|^2. \quad (7)$$

551 To do so, let us first solve the inner optimization problem (i.e. fix the permutation \mathbf{P}) and find the
 552 optimal parameters given the flattened permuted target $\mathbf{v}^{(t)}(\mathbf{P})$. It is well known that the least square
 553 solution is given by $\hat{\gamma} = \mathbf{M}^+ \mathbf{v}^{(t)}(\mathbf{P})$, where \mathbf{M}^+ denotes the Moore-Penrose inverse of the feature
 554 matrix \mathbf{M} . Utilizing the singular value decomposition of $\mathbf{M} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, this leads to the solution

$$\hat{\gamma}(\mathbf{P}) = \mathbf{M}^+ \mathbf{v}^{(t)}(\mathbf{P}) = (\mathbf{M}^T \mathbf{M})^{-1} \mathbf{M}^T \mathbf{v}^{(t)}(\mathbf{P}) = \mathbf{V}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{U}^T \mathbf{v}^{(t)}(\mathbf{P}). \quad (8)$$

555 Plugging this solution into the objective, Eq. (7), turns the nested optimization problem into

$$\begin{aligned} \min_{\mathbf{P}} \min_{\gamma} \|\mathbf{M}\gamma - \mathbf{v}^{(t)}(\mathbf{P})\|^2 &= \min_{\mathbf{P}} \|\mathbf{M}\hat{\gamma} - \mathbf{v}^{(t)}(\mathbf{P})\|^2 \\ &= \min_{\mathbf{P}} \|\mathbf{U}\mathbf{S}\mathbf{V}^T \mathbf{V}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T \mathbf{U}^T - \mathbf{I}\| \mathbf{v}^{(t)}(\mathbf{P})\|^2 \\ &= \min_{\mathbf{P}} \|\mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T - \mathbf{I}\| \mathbf{U}^T \mathbf{v}^{(t)}(\mathbf{P})\|^2, \end{aligned}$$

556 where we have used that $\mathbf{U}\mathbf{U}^T = \mathbf{I}$ (and $\mathbf{V}^T \mathbf{V} = \mathbf{V}\mathbf{V}^T = \mathbf{I}$) and the fact that orthogonal
 557 transformations \mathbf{U} leave the l2-norm invariant.

558 It is easy to see that the matrix $[\mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T - \mathbf{I}]$ defines a projection $\mathbf{P}_r = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_r \end{bmatrix}$. Note that
 559 $\mathbf{S} = \begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix}$, where \mathbf{D} is a diagonal matrix containing the nonzero singular values of \mathbf{M} on the diagonal.

560 Accordingly, we have $\mathbf{S}(\mathbf{S}^T \mathbf{S})^{-1} \mathbf{S}^T = \begin{bmatrix} \mathbf{D} \\ \mathbf{0} \end{bmatrix} \mathbf{D}^{-2} [\mathbf{D} \quad \mathbf{0}] = \begin{bmatrix} \mathbf{D}\mathbf{D}^{-2}\mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}$. It thus
 561 follows that $\min_{\mathbf{P}} \min_{\gamma} \|\mathbf{M}\gamma - \mathbf{v}^{(t)}(\mathbf{P})\|^2 = \min_{\mathbf{P}} \|\mathbf{P}_r \mathbf{U}^T \mathbf{v}^{(t)}(\mathbf{P})\|^2 = \min_{\mathbf{P}} \|\mathbf{U}_2^T \mathbf{v}^{(t)}(\mathbf{P})\|^2$,
 562 where $\mathbf{U} = [\mathbf{U}_1 \quad \mathbf{U}_2]$. \square

563 This theorem gives us a criterion to evaluate the quality of permutations in projection feature
 564 dimensions away that do not align well with the target. We will approximately minimize this criterion
 565 in a Greedy algorithm that permutes two rows of the target in each iteration. As the feature matrix
 566 \mathbf{M} is generally extremely large, i.e., of the order of $c_0 c_2$ times $c_0 c_2$, computing the singular value
 567 decomposition of \mathbf{M} becomes computationally infeasible for realistic neural network architectures.
 568 This also limits our ability to precondition arbitrary random feature matrices. In contrast, our
 569 random feature proposals induce known singular value decompositions that do not require an explicit
 570 numerical estimation. Sparse solutions are particular helpful to construct solutions with low error.
 571 The following theorems explore how many features we can expect to require in this setting.

572 A.1.3 Proof of Theorem 4.2

573 *Statement* (Matching sparse targets and features, Thm. 4.2). Let the target matrix $\mathbf{W}^{(t)}$ be a sparse
 574 matrix with in-degrees $d_i = \sum_j \delta_{w_{ij}^{(t)} \neq 0}$ and the feature matrix \mathbf{M} correspond to a random $Ber(p)$
 575 matrix (i.e. sparse Id cond features). Then, the probability that the (permuted) target can be accurately
 576 matched is upper bounded by $\mathbb{P}(\mathbf{W}^{(t)} \subset \mathbf{M}) \leq \prod_i (1 - (1 - p^{d_i})^{c_2})$.

577 *Proof.* We consider here the general setting where we assume that a sparse target network is given and
 578 we can approximate this target with a random subset of the ID cond features. The latter is equivalent

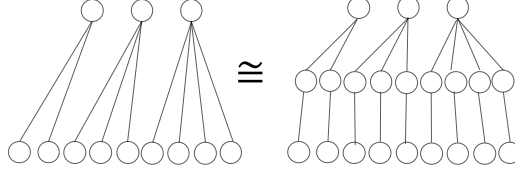


Figure 5: Visualization of construction to obtain the lower bound in Thm. 5. Left: Target network. Right: Random features used to approximate the target.

to a single layer with random Erdoes-Renyi mask with iid Bernoulli entries, where each edge is masked (i.e. is dropped/missing as a feature) with probability $1 - p$, while it exists with probability p . A target output neuron can be accurately represented by an output neuron of the feature matrix if all its d_i incoming target edges exist also in the feature matrix. This is the case with probability p^{d_i} . The question is now how many feature neurons can one choose from to match a specific target neuron. In principle, one could start with one target neuron and try to find the best match among all available c_2 feature output neurons. After a match, the next target neuron could be matched with one of the remaining $c_2 - 1$ feature neurons, etc. The last target neuron would only have one neuron available for a match. Accordingly, the probability of finding a match this way would be relatively low. Yet, this Greedy strategy would likely also not be optimal. Considering all $c_2!$ possible matches would lead to more overall matching options for each neuron. Yet, the number of potential matches would be lower than c_2 in most cases. We can exploit this fact to construct an upper bound on the matching probability.

In fact, this bound can also be attained in a special case. Consider Fig. 1 (e) and Fig. 5 for this special case. If all target output neurons are connected to mutually exclusive sets of input neurons, then all student outputs are potential matches for a target neuron, which maximizes the probability of a match. The reason is that the target neurons do not overlap. Thus, a match with one specific target neuron does not reduce the probability of another neuron to match with this target. Target neurons therefore do not compete for matches.

In this case, the probability that a given target neuron i can be matched with any of c_2 independent output neurons in the feature matrix is given by $1 - (1 - p^{d_i})^{c_2}$. All these matches are independent. Thus, all targets can be thus matched simultaneously with probability $\prod_{i=1}^{c_2} (1 - (1 - p^{d_i})^{c_2})$. \square

As discussed in the main paper, despite being an upper bound, this probability is still relatively small, as we have to match a set of c_2 output target neurons with c_2 output feature matrix neurons. To increase the probability of matches, we have to increase the number of candidates to match with, as proposed by the next theorem.

A.1.4 Proof of Theorem 4.3

Ideally, we would be able to match every target neuron with one of kc_2 available feature matrix output neurons. However, for general targets, the new matches would not be independent of previous target matches. By making the matches independent by partitioning the potential output match candidates, the next theorem gives a relatively conservative bound by not allowing to reuse refused matches. This makes it possible to consider relatively general sparse targets. The theorem provides an intuition how this target sparsity q interacts with the output width increase k and the number of available features p .

Statement (Lower dimensional target, Thm. 4.3). A random target with iid $Ber(q)$ entries and $c_2^{(t)}$ output neurons can be perfectly matched with probability $1 - \delta$ with random ID cond features with expected density p if $c_2 = kc_2^{(t)}$ with $k \geq \frac{\log(1 - (1 - \delta)^{1/c_2^{(t)}})}{\log(1 - (1 - q(1 - p))^{c_0})}$.

Proof. Fig. 6 visualizes the main idea to partition the number of output candidate matches into disjoint subsets, as this allows for independent matches of target neurons. Accordingly, every target neuron can be matched with one of k candidates. A match happens if all target edges exist also in the feature matrix candidate. A single edge is matched with probability $qp + (1 - q)$, since it does not exist in the target with probability $1 - q$ making a match obsolete or it exists with probability qp in both the

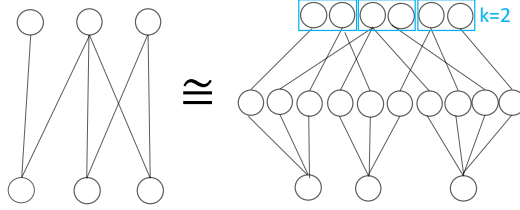


Figure 6: Visualization of construction to obtain the bound on width in Thm. 6. Left: Target network. Right: Random features used to approximate the target. Note that each target neuron can be matched with any output neuron in a set of size k .

target and the feature matrix. All c_0 incoming edges can be matched with probability $(qp + (1 - q))^{c_0}$ and a match with at least one of k available candidates is given by $1 - [1 - qp + (1 - q)]^{c_0 k}$. The theorem statement follows from requesting that all c_2 targets have matches with probability at least $1 - \delta$ such that $(1 - [1 - qp + (1 - q)]^{c_0 k})^{c_2} \geq 1 - \delta$. \square

Note that, in practice, more than k candidates would be available for matching. In the special case of Thm. 4.2, even kc_2 candidates would be available in every matching attempt. However, in general, those matches would not be independent and hence difficult to assess probabilistically.

The previous theorem has still expected a student-teacher setup for matching a single sparse random layer. Deep neural networks offer, however, many more options to represent or approximate different ways to solve a task, thus, providing more options to succeed with sparse networks. Empirical results for random lottery tickets [32, 21] suggest that random networks with medium sparsity can achieve competitive performance on standard benchmark tasks. This suggests that sparse ID cond features, which are equivalent to such random masks (see Thm. 3.1), can attain the same performance, leading to the following conclusion.

Conclusion. If a task is solvable by training an iid randomly masked neural network with density p_l , then $c_1^l = p_l^l c_0^l c_2^l$ random ID cond features can achieve the same generalization performance.

Proof. Explicit lottery ticket existence proofs for random source networks [21] require a substantial overparameterization of the source network in comparison to a target network. For this reason, the theory does not directly support lower width requirements relative to a target network unless the target network has much lower dimensions (and thus we have large k). However, we can follow a similar construction idea to obtain a better width requirement by using a 3-layer instead of a 2-layer normalization layer construction to approximate a single sparse random target network. \square

This insight concludes our theoretical insights that show that there exist sparse random features, which support lower than previously assumed width requirements for learning only normalization layers.

A.2 Permutation algorithms

How can we find a permutation of the rows of the target network or, equivalently, of the feature matrix output neurons and thus the rows of $\mathbf{W}^{(2)}$? It is computationally infeasible to go through all $c_2!$ possible permutations even if we have to evaluate only $\|\mathbf{P}_r \mathbf{U}^T \mathbf{v}^{(t)}\|^2$ instead of solving each associated linear regression problem utilizing Thm. 4.1. Initially, we have implemented a continuous relaxation of the permutation [35]. Yet, this approach was outperformed by just sampling random permutations and picking the best one. This approach would hardly scale to larger output dimensions c_2 and also not leverage target information effectively in comparison what we can achieve in Fig. 4.

To obtain Fig. 4, we have implemented a Greedy algorithm that leverages the insight of Thm. 4.1 and matches each output index i_u that corresponds to a set of indices in the singular value space $u_{(i_u j)_k}$ with an output index i_t of the target $w_{i_t j}^{(t)}$. Concretely, we aim to minimize the objective $\min_{\pi} \sum_{k=c_2 c_0 - r + 1}^{c_2 c_0} \left(\sum_{ij} u_{(\pi(i) j)_k} w_{i_t, ij} \right)^2$ searching for a permutation π of the row indices that

657 facilitate the matching. The problem arises from the fact that the sum over i is inside the square.
658 Therefore, sum of the terms $u_{(\pi(i)j)k}w_{t,ij}$ could also have a positive or negative contribution to
659 the sum so that we cannot minimize the terms independently via a pairwise matching. Therefore,
660 we evaluate the change of the score (where the score is $\sum_{k=c_2c_0-r+1}^{c_2c_0} \left(\sum_{ij} u_{(\pi(i)j)k}w_{t,ij} \right)^2$) in
661 response to a match between a row of \mathbf{U} and $\mathbf{W}^{(t)}$. Starting with the indices i_u of \mathbf{U} with the largest
662 norm, we match them Greedily to the best free row index i_t of the target that minimizes the score.
663 Our specific Python implementation follows and is also shared as part of the supplementary code.

```
def svd_permute_fc(w01,w02,wt):
    n2, n1, m = w02.size(0), w01.size(1), w01.size(0)
    M = torch.einsum('ik,kj->ijk', w02, w01)
    M = M.reshape((n2*n1,m))
    try:
        U, S, Vh = torch.linalg.svd(M)
        ss = S.size(0)
        Umiss = U[:,ss:]
        Umiss = Umiss.reshape((n2,n1,-1))
        #Greedy
        order_match = torch.argsort(torch.sum(Umiss**2,dim=(1,2)),descending=True)
        perm_Greedy = torch.arange(n2)
        remain = torch.arange(n2)
        for i in order_match:
            i = i.item()
            back = torch.einsum('ijk,ij->k',Umiss,wt[perm_Greedy])
            ind_current = perm_Greedy[i].item()
            score = back-torch.einsum('jk,j->k',Umiss[i,:,:],wt[ind_current,:])
            sc = torch.sum(back**2).item() #torch.zeros(remain.size(0))
            pc = ind_current
            ic = i
            for p in remain:
                p = p.item()
                ip = torch.where(perm_Greedy==p)[0].item()
                x = score+torch.einsum('jk,j->k',Umiss[i,:,:],wt[p,:])..
                ..+torch.einsum('jk,j->k',Umiss[ip,:,:],wt[ind_current,:])
                x = x-torch.einsum('jk,j->k',Umiss[ip,:,:],wt[p,:])
                x = torch.sum(x**2)
                if x.item() < sc:
                    sc = x.item()
                    pc = p
                    ic = ip
            perm_Greedy[i] = pc
            perm_Greedy[ic] = ind_current
            remain = remain[remain!=pc]
        col_ind = torch.argsort(perm_Greedy)
        w02 = w02[col_ind]
        gamma1 = torch.transpose(Vh,0,1)[:,:ss]/S ..
        ..@ torch.transpose(U,0,1)[:ss,:] @ wt[perm_Greedy].reshape((-1,))
    return w02, gamma1
```

664 If \mathbf{M} is sparse, a pairwise distance minimization becomes feasible. We can directly maximize the
665 overlap of sparse features with the largest target links with the help of minimum weight matching.
666 In most cases, this sparse matching provides a strong signal for finding a good permutation. We
667 hypothesize that a better matching can often be obtained because the zeros do not incur additional
668 error that propagates through the network as in case of standard random weight matrices that induce
669 dense feature matrices. In addition to the Greedy approach above, we thus also consider a mask
670 overlap maximization, as implemented below.

```
def id_permute(w01,w02,wt):
    n2, n1, m = w02.size(0), w01.size(1), w01.size(0)
```

```

mask = torch.ones((n2,n2))-torch.einsum('ik,kj->ij', w02, w01)
deg = torch.sum(wt**2,dim=1)
#Greedy
order_match = torch.argsort(deg,descending=True)
perm_Greedy = torch.arange(n2)
remain = torch.arange(n2)
for i in order_match:
    i = i.item()
    sc = torch.sum((mask[perm_Greedy,:]*wt)**2).item()
    ind_current = perm_Greedy[i].item()
    pc = ind_current
    ic = i
    for p in remain:
        p = p.item()
        ip = torch.where(perm_Greedy==p)[0].item()
        x = sc+torch.sum((mask[p,:]*wt[i,:])**2)..
        ..+torch.sum((mask[ind_current,:]*wt[ip,:])**2)
        x = x-torch.sum((mask[ind_current,:]*wt[i,:])**2)..
        ..-torch.sum((mask[p,:]*wt[ip,:])**2)
        if x.item() < sc:
            sc = x.item()
            pc = p
            ic = ip
    perm_Greedy[i] = pc
    perm_Greedy[ic] = ind_current
    remain = remain[remain!=pc]
col_ind = perm_Greedy
w02=w02[col_ind]
err1 = torch.sqrt(torch.mean((mask[col_ind]*wt)**2)).item()
#second approach to identify permutation:
mask = torch.einsum('ik,kj->ij', w02, w01)
x, _ = torch.sort(torch.abs(wt).flatten(),descending=True)
dd=n1*n2
p=(m+n2-1)/dd
thr = min(int(dd*2*p/(1+p)),dd-1)
thr = x[thr]
maskt = torch.where(torch.abs(wt)>thr,1.0,0.0)
dist_mat_rows = torch.cdist(maskt,mask,p=2)
row_ind, col_ind = linear_sum_assignment(dist_mat_rows.cpu().detach().numpy())
mask = mask[col_ind]
err = torch.sqrt(torch.mean((mask*wt-wt)**2)).item()
if err < err1:
    w02=w02[col_ind]
    err=err1
return err, w02, col_ind

```

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We propose random features that enable learning only normalization layers. They are well conditioned, maximally sparse, and lower the theoretical width requirement in analyzed situations.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: The conclusion of our analysis is that we cannot attain fundamental efficiency gains by learning only normalization layers. This is a relevant insight for the community, as the underlying idea has been frequently proposed.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Proofs are provided in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The code is provided as supplement. Experimental details are stated in the main paper and the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Only public datasets were used. The code has been shared as supplement.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Only standard training settings were used and the hyperparameters are stated.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: 0.95 standard confidence intervals are reported.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Experiments were conducted on a machine with Intel(R) Core(TM) i9-10850K CPU @ 3.60GHz processor and GPU NVIDIA GeForce RTX 3080 Ti as stated in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The nature of the analysis is primarily theoretical and only public datasets were used in experiments.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: We have discussed the insights and limitations of the analysis, which is quite methodological in nature without a specific application with societal impact in mind.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our contributions are specific insights into deep learning.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: Only standard Python/Pytorch packages were used and cited.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. New assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No assets are introduced. Code is shared.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. Crowdsourcing and research with human subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. Institutional review board (IRB) approvals or equivalent for research with human subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- 981 • We recognize that the procedures for this may vary significantly between institutions
982 and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the
983 guidelines for their institution.
984 • For initial submissions, do not include any information that would break anonymity (if
985 applicable), such as the institution conducting the review.

986 **16. Declaration of LLM usage**

987 Question: Does the paper describe the usage of LLMs if it is an important, original, or
988 non-standard component of the core methods in this research? Note that if the LLM is used
989 only for writing, editing, or formatting purposes and does not impact the core methodology,
990 scientific rigorousness, or originality of the research, declaration is not required.

991 Answer: [NA]

992 Justification: The core method development in this research does not involve LLMs as any
993 important, original, or non-standard components.

994 Guidelines:

- 995 • The answer NA means that the core method development in this research does not
996 involve LLMs as any important, original, or non-standard components.
997 • Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>)
998 for what should or should not be described.