Bias of Stochastic Gradient Descent or the Architecture: Disentangling the Effects of Overparameterization of Neural Networks

Amit Peleg [1 2] **Matthias Hein** [1 2]

Abstract

Neural networks typically generalize well when fitting the data perfectly, even though they are heavily overparameterized. Many factors have been pointed out as the reason for this phenomenon, including an implicit bias of stochastic gradient descent (SGD) and a possible simplicity bias arising from the neural network architecture. The goal of this paper is to disentangle the factors that influence generalization stemming from optimization and architectural choices by studying *random* and *SGD-optimized* networks that achieve zero training error. We experimentally show, in the low sample regime, that overparameterization in terms of increasing width is beneficial for generalization, and this benefit is due to the bias of SGD and not due to an architectural bias. In contrast, for increasing depth, overparameterization is detrimental for generalization, but random and SGD-optimized networks behave similarly, so this can be attributed to an architectural bias.

1. Introduction

The generalization of neural networks challenges common wisdom in classical statistical learning theory (Vapnik & Chervonenkis, 1971): the number of parameters in today's neural networks is much larger than necessary to fit the data (Zhang et al., 2017), and further increasing network size, i.e., more overparameterization, yields better generalization (Hestness et al., 2017; Allen-Zhu et al., 2019). The underlying mechanisms of generalization despite overparameterization remain, to a large extent, an open question.

While the contribution of certain aspects has been examined in the past (e.g., batch size (Keskar et al., 2017a) and learning rate (Li et al., 2019)), isolating the effect of gradient-based optimization is more challenging due to its integral role in today's well-performing networks. The implicit bias of Stochastic Gradient Descent (SGD) is thus often thought to be the main reason behind generalization (Arora et al., 2019; Shah et al., 2020). A recent thought-provoking study by Chiang et al. (2023) suggests the idea of the volume hypothesis for generalization: well-generalizing basins of the loss occupy a significantly larger volume in the weight space of neural networks than basins that do not generalize well. They argue that the generalization performance of neural networks is primarily a bias of the architecture and that the implicit bias of SGD is only a secondary effect. To this end, they randomly sample networks that achieve zero training error (which they term Guess and Check (G&C)) and argue that the generalization performance of these networks is qualitatively similar to networks found by SGD.

In this work, we revisit the approach of Chiang et al. (2023) and study it in detail to disentangle the effects of implicit bias of SGD from a potential bias of the choice of architecture. As we have to compare to randomly sampled neural networks, our analysis is restricted to binary classification tasks in the low sample regime. Specifically, we analyze the behavior of the LeNet (LeCun et al., 1998), Multi-Layer Perceptron (MLP), and ResNet (He et al., 2016) architectures.

In summary, we make the following contributions:

- We show that some findings of Chiang et al. (2023) are based on a sub-optimal initialization of SGD, which makes SGD artificially worse. Moreover, we argue that their normalized loss cannot be used to compare networks of different architectures. We suggest an alternative normalization and analysis scheme that allows us to study the effects of increasing overparameterization (in terms of width and depth).
- We show that increasing overparameterization in terms of width improves the generalization of SGD, while for randomly sampled networks, it remains mostly unaffected. This indicates a clear implicit bias of SGD.
- On the contrary, increasing overparameterization in terms of depth is detrimental to generalization both for SGD

[1]University of Tübingen, Germany [2]Tübingen AI Center, Germany. Correspondence to: Amit Peleg <amit.peleg@uni-tuebingen.de>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

and G&C. As they behave similarly, the negative impact of increased depth can be mainly attributed to a bias in the architecture.

Note that we do not claim that increasing depth is always detrimental to generalization, but rather that improvements regarding depth are due to architectural bias and not due to SGD. In contrast, increasing width, even for randomly sampled networks, does not harm generalization, and the implicit bias of SGD leads to networks with higher margins, significantly improving generalization.

2. Related Work

Understanding the generalization properties of deep neural networks is a long-standing research topic that has been tackled from different angles (Cybenko, 1989; Hornik, 1991; Wolpert, 1995; Bartlett & Mendelson, 2002; Hoffer et al., 2017; Jakubovitz et al., 2019).

Overparameterization and generalization. Traditional learning theory with complexity measures, e.g., the VC dimension (Vapnik & Chervonenkis, 1971), suggests that for generalization, models should not be able to fit any possible training data (Shalev-Shwartz & Ben-David, 2014). In particular, heavily over-parametrized models should overfit the data (Hastie et al., 2001). However, even though networks are large enough to fit the data perfectly (Haeffele & Vidal, 2017; Nguyen & Hein, 2018), we essentially observe the opposite (Belkin et al., 2018; Neal et al., 2019; Bartlett et al., 2020). Specifically, Zhang et al. (2017) showed that a CNN architecture can obtain perfect train accuracy both on random and non-random labels on the CIFAR10 dataset, proving that the models are expressive enough to memorize while still being able to generalize. Additionally, Arpit et al. (2017) illustrated that the networks do not just memorize the data but learn simple patterns first, and Neal et al. (2019) reported that both bias and variance can decrease as the number of parameters grows.

Implicit bias from the optimizer. The success of modern deep learning is often attributed to the implicit bias of the optimizers (Neyshabur et al., 2015): Soudry et al. (2018) showed that SGD on linearly separable data converges to the maximum-margin linear classifier, and Arora et al. (2019) hypothesized that the implicit regularization of gradient-based optimizers goes beyond what can be expressed by standard regularizers. Galanti & Poggio (2022) investigated the effects of small batch size on the network's rank. Liu et al. (2020) related the gradient signal and noise to generalization, Advani et al. (2020) highlighted that learning with gradient descent effectively takes place in a subspace of the weights, and Lee et al. (2020) observed simplified learning dynamics for wider networks. Andriushchenko et al. (2023b) showed that large step-size SGD leads to low-rank features.

Loss landscape perspectives. Several works connect generalization properties to the geometry of the loss landscape, often involving sharpness quantities (Dziugaite & Roy, 2017; Keskar et al., 2017b; Jiang* et al., 2020; Foret et al., 2021). The relevance of this relation, however, is unclear (Andriushchenko et al., 2023a). This connection between flatness and well-performing models is also present in the Bayesian literature (Izmailov et al., 2018; Wilson & Izmailov, 2020).

The volume hypothesis. Pérez et al. (2019) observed a bias towards simple functions in deep networks, irrespective of optimization. Huang et al. (2020) hypothesized that the volume of bad minima in weight space might be smaller than that of well-generalizing minima. Mingard et al. (2021) further argued that, in simplified terms, SGD behaves like a Bayesian sampler and that the inductive bias in deep learning does not primarily stem from the optimizer. However, their approximation requires infinite width and thus does not apply to practical networks. Geiping et al. (2022) questioned the relevance of stochasticity in SGD by illustrating that gradient descent with explicit regularization can perform comparably. Chiang et al. (2023) argued that the implicit bias of SGD is only a secondary effect, and the main contribution to generalization comes from the architecture.

3. Setup

Our main goal in this work is to disentangle the architectural bias – the bias inherent in the function class obtained by randomly sampling the network parameters – from the bias of SGD regarding generalization in the overparameterized setting. Our setup is similar to Chiang et al. (2023). However, we deviate in our evaluation, arguing below that their conclusions are potentially inaccurate due to issues in their application of SGD and the quantities they considered. To achieve our objective, we utilize a random sampling procedure, termed G&C in Chiang et al. (2023), as well as SGD to attain neural networks that achieve zero training error. However, due to the exponentially increasing computational cost of G&C, we limit our study to binary classification tasks from MNIST (LeCun et al., 1998) and CIFAR10 (Krizhevsky et al., 2009) in the low sample regime.

3.1. Neural Network Architecture

Following the setup of Chiang et al. (2023), we focus on versions of the LeNet architecture (LeCun et al., 1998) in all of our experiments. Additionally, we confirm our main results with an MLP and a small ResNet (He et al., 2016). Details on the architectures can be found in Appendix B.1. For the LeNet, the ReLU activation function, $a(z) = \max\{z, 0\}$, is applied after each layer (except for the pooling layers). The

resulting function $f : \mathbb{R}^d \to \mathbb{R}^K$ induced by this network can be written as,

$$f(W_1, \dots, W_L, x) = W_L \circ a \circ W_{L-1} \circ \dots \circ a \circ W_1 x,$$

with input dimension d and K classes. Note that the output vector of f contains the logits of the classes.

When using the ReLU activation function, the function f is one-homogeneous in the weights W_l of each layer and in the input x. That is, for each $\gamma > 0$,

$$f(W_1, ..., \gamma W_l, ..., W_L, x) = \gamma f(W_1, ..., W_l, ..., W_L, x).$$

For compactness, we use the notation $f(W, x)$, where $W = (W_1, \dots, W_L)$.

We use the $\{-1, 1\}$ label encoding for binary classification and denote the two components of f as f_1 and f_{-1}. Let $\mathcal{S}_{\text{train}} = (x_i, y_i)_{i=1}^n$ be the training set, where $x_i \in \mathbb{R}^d$, and $y_i \in \{-1, 1\}$. The one-homogeneity of the ReLU network implies that if a network realizes zero-training error, that is

$$y_i \left(f_1(W, x_i) - f_{-1}(W, x_i)\right) \geq 0, \quad i = 1, \dots, n,$$

and one uses cross-entropy loss (which reduces to logistic loss in the binary case),

$$l(y, f(W, x)) = \log \left(1 + e^{-y_i (f_1(W, x_i) - f_{-1}(W, x_i))}\right),$$

then simply upscaling the weights of any layer to infinity yields zero loss. Since upscaling the weights does not change the classifier, the training loss itself is not a suitable criterion for distinguishing two ReLU networks that achieve zero training error. Therefore, as the goal of this paper is to study the overparameterized regime, where the network is large enough to achieve zero training error, one needs a suitable normalization, which is discussed in the next paragraph.

3.2. Geometric Margin as a Criterion to Compare Networks Achieving Zero Training Error

As discussed above, the (cross-entropy) loss itself is not a suitable criterion to distinguish networks that achieve zero training error. A proper criterion should eliminate the degree of freedom for upscaling weights (Farhang et al., 2022). Let

$$g(W, x) := f_1(W, x) - f_{-1}(W, x),$$

and denote with $L(g(W))$ its Lipschitz constant for the ℓ_2-distance with respect to the input x. Then, the criterion

$$\rho(x) \triangleq \frac{g(W, x)}{L(g(W))}, \tag{1}$$

eliminates the scaling degree of freedom in the weights as the Lipschitz constant $L(g(W))$ has the same homogeneity as $g(W, x)$ with respect to the weights. Furthermore, as shown below, $|\rho(x)|$ provides a lower bound on the distance to the decision boundary. Let z be on the decision boundary, that is $g(W, z) = 0$. Then,

$$|g(W, x)| = |g(W, x) - g(W, z)| \leq L(g(W)) \left\|x - z\right\|_2,$$

thus $|\rho(x)|$ is a lower bound on the ℓ_2-distance of x to the decision boundary of g. However, it is difficult to determine the Lipschitz constant of a neural network.

As a ReLU-network is piecewise linear as a function of x, it holds

$$L(g(W)) = \max_{x \in \mathbb{R}^d} \left\|\nabla_x g(W, x)\right\|_2.$$

We determine a lower bound on $L(g(W))$ by taking the maximum over the union of train and test data, i.e.,

$$\tilde{\rho}(x) = \frac{g(W, x)}{\max_{z \in \mathcal{S}_{\text{train}} \cup \mathcal{S}_{\text{test}}} \left\|\nabla_x g(W, z)\right\|_2}. \tag{2}$$

We then plug $\tilde{\rho}(x)$ into the loss instead of $g(W, x)$ and term it **Lipschitz normalized loss**. We note that by replacing the true Lipschitz constant with a lower bound, we cannot argue anymore that $|\tilde{\rho}(x)|$ is a lower bound of the distance to the decision boundary. However, this data-based estimate is a tighter estimate of $L(g(W))$ than a direct upper bound that uses the 1-Lipschitzness of the ReLU activation,

$$L(g(W)) \leq \prod_{l=1}^{L} \left\|W_l\right\|_{2,2},$$

where $\left\|W_l\right\|_{2,2}$ denotes the spectral norm of the weight matrix. In fact, the normalization of Chiang et al. (2023) by the product of Frobenius norms,

$$\gamma(x) = \frac{g(W, x)}{\prod_{l=1}^{L} \left\|W_l\right\|_F}, \tag{3}$$

is even a coarser upper bound. To show that, let $\sigma_i(W)$ be the singular values of W, then $\left\|W\right\|_F = \sqrt{\sum_i \sigma_i^2(W)} \geq \max_i \sigma_i(W) = \left\|W\right\|_{2,2}$. Note that this gap grows with the rank of W. In their analysis, $\gamma(x)$ is plugged into the loss, which we term **weight normalized loss**. While their normalized loss is fine when comparing neural networks of the *same* architecture, it is problematic when comparing *different* architectures with different ranks of W, e.g., networks of different widths. In Section 4.2, we show that this ambiguity in the definition of the normalized loss can change the interpretation of the results of Chiang et al. (2023). While we believe the **Lipschitz normalized loss** is a more accurate estimate, we base our main observations on quantities independent of this normalization to avoid the resulting ambiguities. If not specified differently in the paper, "normalized loss" always denotes our Lipschitz normalized loss.

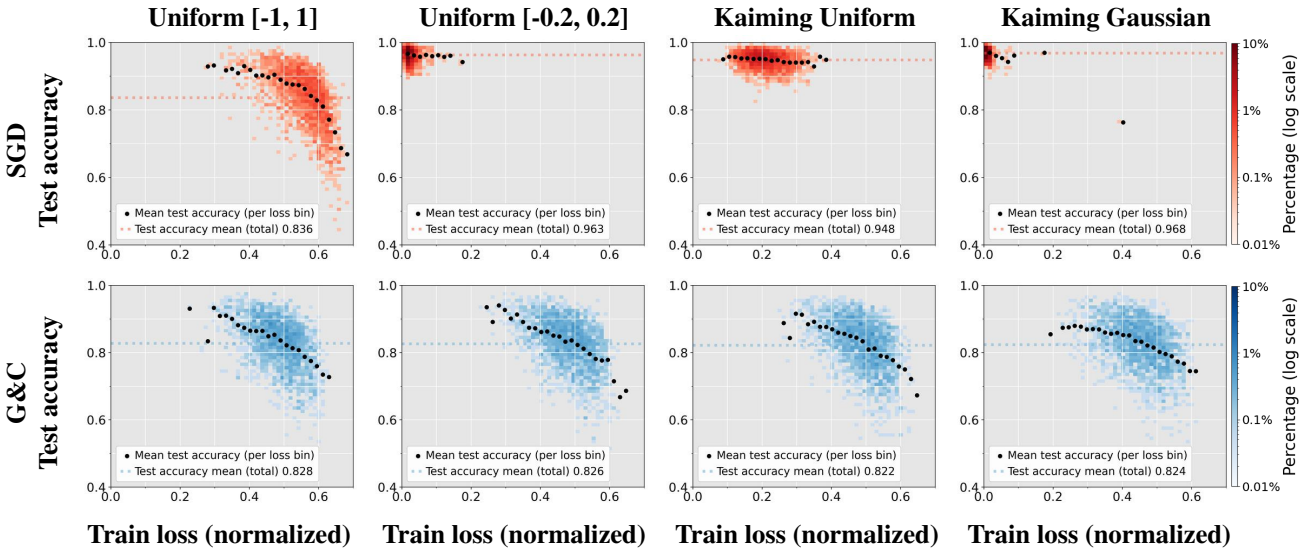


Figure 1. **Generalization of SGD (optimized) versus G&C (randomly sampled) in dependency of the prior on the weights $\mathrm{P}(W)$:** We "train" 2000 LeNet models to 100% train accuracy for 16 training samples from classes *0* and *7* of MNIST. Test accuracies for G&C are similar across initializations, and the normalized loss (see Section 3) is similar across the uniform distributions. **Column 1:** For $\mathcal{U}[-1,1]$ initialization, as used by Chiang et al. (2023), the normalized losses and the test accuracies of SGD and G&C are similar, except for the convergence of SGD towards more low-margin solutions. The claim in Chiang et al. (2023) that the average test accuracy of G&C resembles SGD, conditional on the normalized loss bin (black dots), is an artifact of the suboptimal convergence of SGD caused by this initialization. **Columns 2-4:** For other initializations, SGD (first row) improves considerably both in terms of loss and accuracy. In contrast, G&C remains unaffected, as it is independent of the scale of the weights in each layer. Results for different numbers of samples and other classes from MNIST and CIFAR10 are in Appendix C.

3.3. Training of SGD and Guess and Check (G&C)

We deliberately avoid commonly used techniques for improving generalization, such as augmentation, weight decay, and sophisticated learning rate schedulers (Ruder, 2016). Instead, we train SGD using cross-entropy loss for 60 epochs and a batch size of 2, with a fixed learning rate. We run SGD from random initialization (see Section 4.1) and only keep networks that achieve zero training error. Due to the strong overparameterization and our small number of training samples, zero training error is attained for almost every initialization.

The main analysis tool of this work is randomly sampling networks from a prior on the weights and only accepting networks that achieve zero training error. This is called Guess and Check algorithm (G&C) in Chiang et al. (2023), and we keep this name for simplicity. We discuss the effects of different priors in Section 4.1.

The advantage of random sampling networks from a prior $P(w)$ is that we can directly estimate the probability of achieving zero training error for a given training set $\mathcal{S}_{\text{train}} = (x_i, y_i)_{i=1}^n$. Given that the prior distribution has compact support, e.g., the uniform distribution, this is proportional to the "volume" in the weight space of perfectly fitting solutions. Let $M_{\mathcal{S}_{\text{train}}}$ be the number of sampled networks needed to get N networks that have zero training error, then we can estimate this probability as

$$\mathrm{P}_{W\sim P(W)}(\text{Train Error}(W) = 0 \,|\, \mathcal{S}_{\text{train}}) \approx \frac{N}{M_{\mathcal{S}_{\text{train}}}}.$$

This probability indicates how difficult it is to fit the data perfectly. A random function $f : \mathbb{R}^d \to \{-1, 1\}$ yields probability $\mathrm{P}_f(\text{Train Error}(W) = 0 \,|\, \mathcal{S}_{\text{train}}) = \frac{1}{2^n}$ for n training samples. This baseline is indicated with a black dotted line when we plot the probability to find a perfectly fitting network vs the training set size in Figures 3 and 4. Note that random networks can be biased towards constant predictions and thus be worse than a random function for a small number of training samples.

4. Experiments

In the case of overparameterization, SGD typically converges to a solution with zero training error. It is also a common belief that SGD is biased towards networks that generalize better. Thus, we clearly expect, even in a low sample regime, that SGD finds, on average, better generalizing networks than G&C. It is, however, unclear to which extent the inductive bias of the function class, which is implemented by the neural network architecture, contributes to better generalization. By increasing the overparameterization of the networks and comparing the average test error

over networks that have achieved zero training error, we can disentangle the effects of the architecture and the optimization. Specifically, we overparametrized the networks by increasing width or depth. Then, when SGD and G&C behave similarly, the change can be purely attributed to the architecture. In contrast, when they behave differently, the optimization process has a clear additional bias.

As all networks found by G&C have zero training error, their average test error serves as an estimator for the expected generalization ability of random networks that achieve zero training error, which is

$$\mathbb{E}[\text{Test Error}(W) \mid \text{Train Error}(W) = 0].$$

We use average test accuracy as the main quantity to compare SGD and G&C, as it remains unaffected by potential issues related to weight normalization. In the remainder of this section, we explore in Section 4.1 the effect of initialization of the network (different priors for the weights) and the behavior when increasing the width and depth of the network (Section 4.2 and Section 4.3, respectively).

4.1. Dependency of SGD and G&C on the Prior $P(W)$

Networks randomly sampled from $P(W)$ correspond to the initialization used in SGD and are the ones checked for zero training error in G&C. We note that G&C is independent of the scale of the weights of each layer as the ReLU-network is one-homogeneous in every layer, and thus, it only changes the scale of the logits but not the classification. This is different for SGD, where it is well-known that the scale of each layer's initialization significantly affects the training dynamics. This is why, in practice, the Kaiming initialization, also known as He initialization (He et al., 2015), is often used, in which the weights of each layer are sampled from a uniform or Gaussian distribution and then normalized.

Chiang et al. (2023) find that conditioned on a specific bin of the weight normalized loss, G&C and SGD have similar test errors. From this, they conclude that the main effect of generalization in the overparameterized setting is not due to a bias of SGD but is inherent in the network architecture. However, they do not apply the standard Kaiming initialization but sample each weight uniformly at random in $[-1, 1]$. This leads to large logits and low loss once zero training error is achieved, resulting in small gradients. Consequently, SGD essentially stops the optimization early in the training process, as illustrated in Figure 7 in the appendix, leaving the margin to the decision boundary small. Thus, they effectively compare a suboptimal early-stopped SGD-optimized network with a randomly sampled network, which leads to their impression that G&C and SGD behave similarly.

In Figure 1, we plot the test accuracy vs (Lipschitz) normalized loss for SGD and G&C using three different uniform initializations (their Uniform $[-1, 1]$, Uniform $[-0.2, 0.2]$, Kaiming Uniform) and the Kaiming Gaussian initialization. For each algorithm, we "trained" 2000 models to achieve 100% train accuracy on a fixed subset of 16 samples from classes *0* and *7* from the MNIST dataset (similar results for different numbers of training samples and other classes from MNIST and CIFAR10 can be found in Appendix C).

As argued in Section 3.1, one observes that G&C is not affected by the scale of the uniform initialization, whereas the outcome of SGD depends heavily on it. For the Uniform $[-1, 1]$ used in Chiang et al. (2023), the average test accuracy is indeed almost the same for SGD and G&C (83.6% vs. 82.8%). However, with proper initializations, the results of SGD improve drastically to 96.3%. Additionally, the plot demonstrates that it is essentially impossible to compare SGD and G&C conditional on loss bins, as they achieve significantly different normalized losses. We stress that these findings are independent of which normalized loss is used. In Figure 11, we show the same comparison as Figure 1, but with the weight normalized loss of Chiang et al. (2023).

In addition to Figure 7, which illustrates the problems with suboptimal initialization, Figure 12 in the appendix shows the trajectory of SGD with good initialization (Kaiming uniform) as one varies the number of training epochs. The distribution of normalized losses of SGD after one epoch (i.e., not fully optimized networks) is comparable to that of G&C and SGD with suboptimal initialization. However, even in this case, conditional on the bin of the normalized loss, the average test accuracy of SGD is better than G&C.

Therefore, for the remainder of the paper, we use Kaiming initialization for SGD to get properly optimized networks. However, we sample each layer from a uniform distribution so that our results of G&C are directly comparable to the ones of Chiang et al. (2023), and since Kaiming Gaussian did not result in major differences both for SGD and G&C.

The consistent convergence of properly initialized SGD towards functions that generalize better than G&C within the pool of networks fitting the training data shows that such well-generalizing functions occupy a small volume in the set of networks with zero training error. We explicitly test this by initializing SGD with networks found by G&C, that is, randomly sampled networks using Kaiming uniform, which fit the data perfectly. The test accuracy of the converged SGD versus the test accuracy of the initialization of SGD found by G&C is shown in Figure 13a in the appendix. In almost all cases, SGD improves compared to the G&C-initialization, which again demonstrates the positive bias of SGD regarding generalization in the heavily overparameterized regime. In addition, the normalized loss and test accuracy values for these optimized networks, depicted in Figure 13b, exhibit no significant difference from those associated with SGD with the standard Kaiming initialization in Figure 1. This suggests that initializing a network with a

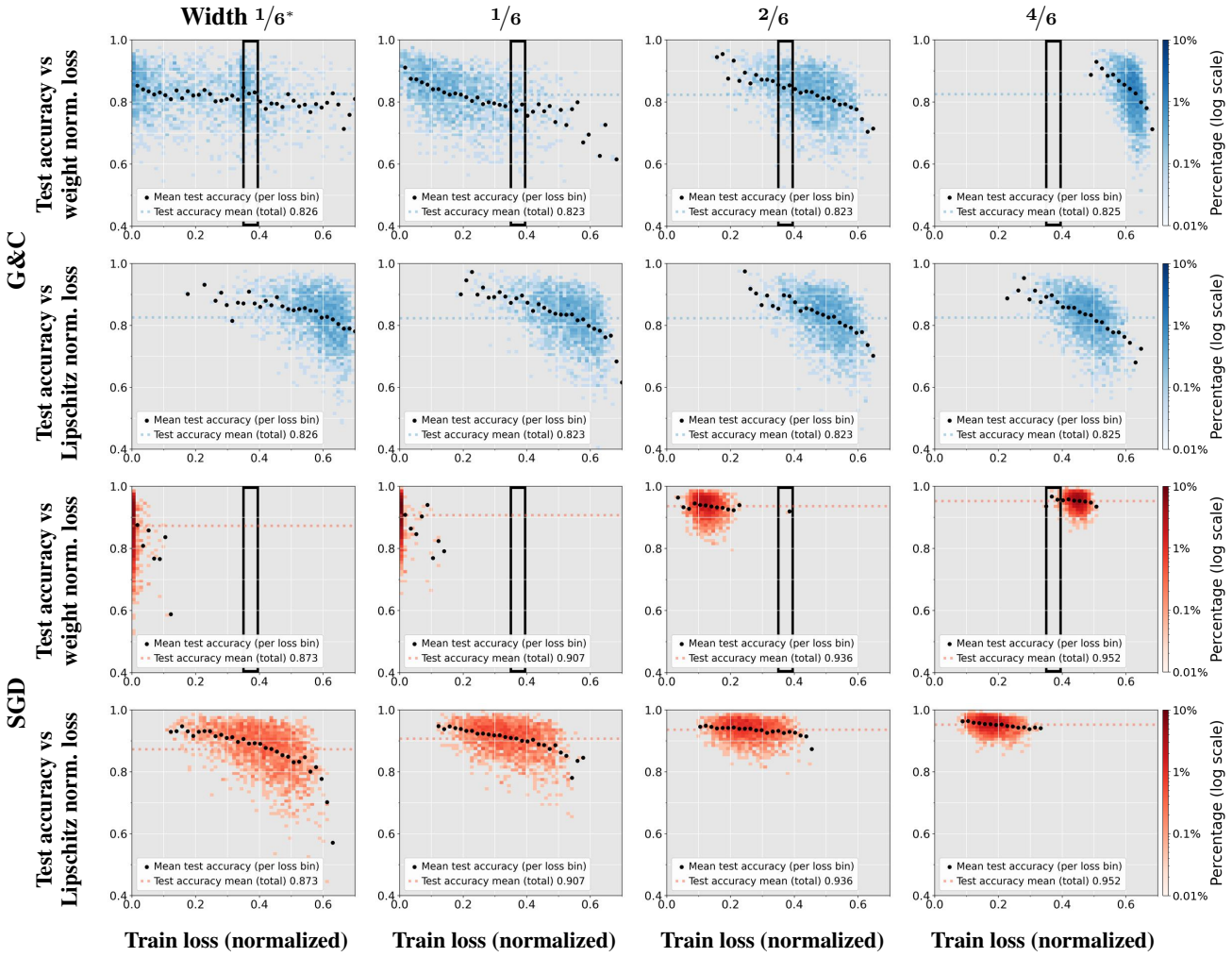


Figure 2. **Analysis of overparameterization when increasing the width.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (2) of **SGD** and **G&C** for classes *0* & *7* of MNIST and 16 training samples across 2000 LeNet models. **Row 4:** Widening the networks enhances geometric margin (lower normalized loss) and average test accuracy for SGD, while for G&C **(Row 2)**, the margin improves only slightly, and average test accuracy remains the same. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias (see Figure 3). **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by black boxes), which led them to conclude that G&C improves with increasing width. With our Lipschitz normalized loss, one would arrive at the opposite conclusion, which shows the problem of normalization. Results for different numbers of samples and other classes from MNIST and CIFAR10 are in Appendix D.

solution that already perfectly fits the data does not give any advantage to the convergence of SGD.

To summarize, in this section, we examined the influence of the prior on SGD and G&C. We found that the performance of SGD heavily depends on proper initialization, while G&C is unaffected by the scale of the weights. This contrasts Chiang et al. (2023), who reported similar behavior of SGD and G&C, using suboptimal initialization. Since proper initialization significantly improves SGD results while keeping G&C performance unaffected, we conclude that there exists a positive bias of SGD for generalization in overparameterized regimes.

4.2. Overparameterization in Terms of Increasing Width

Following Chiang et al. (2023), we examine the influence of increasing the width of the network. To this end, we shrink all layers proportionally to the original LeNet architecture, e.g., width $2/6$ keeps 2 out of the 6 convolutions in the first layer and $\text{floor}(2/6 \cdot 16) = 5$ out of 16 convolutions in the second layer. Width $1/6^*$ means a reduction of $1/6$ for the convolution layers and $1/24$ for the fully connected layers (details in Tables 1 and 2 in the appendix). We note that all the networks of smaller widths, even the smallest one, are overparameterized in the sense that the network can fit any

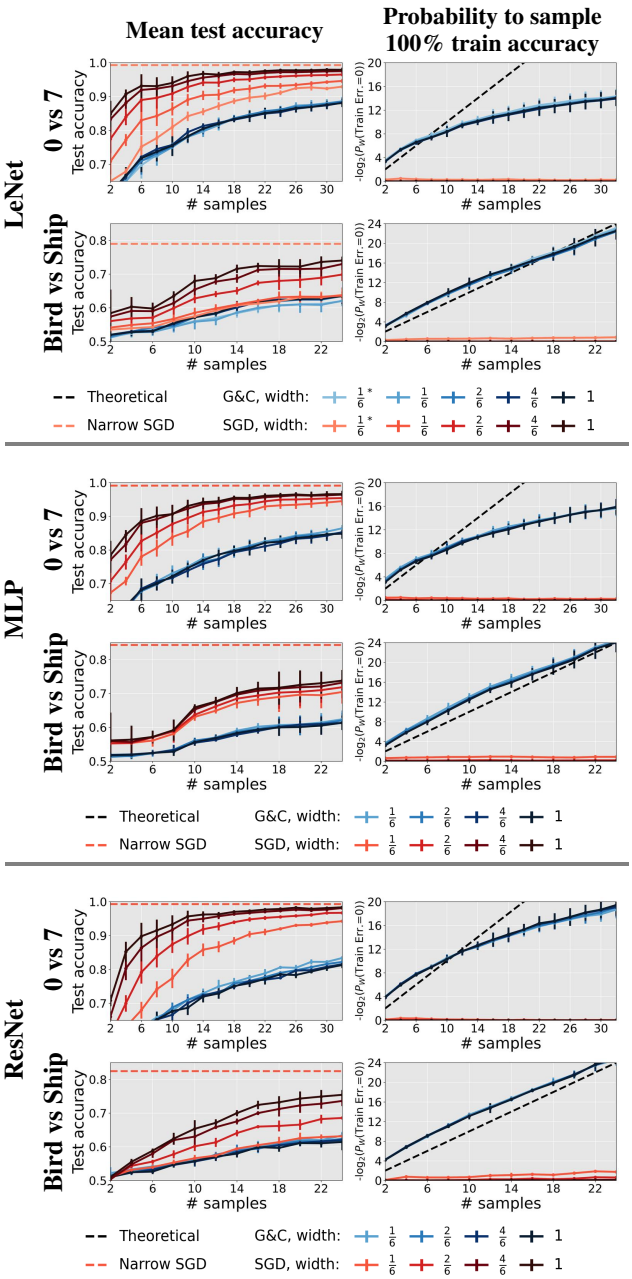


Figure 3. **Increasing width is a positive optimization bias.** From top to bottom, the architectures are LeNet, MLP, and ResNet. Within each architecture, the first row corresponds to MNIST and the second to CIFAR10. **Left:** Mean test accuracy vs number of training samples across network widths. SGD improves for wider networks, while G&C behaves similarly for all widths. Thus, for increasing width, SGD has a bias towards better generalizing networks independent of an architectural bias. However, there is also no overfitting for G&C and thus no sign that overparameterization hurts. **Right:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). This number remains the same for different widths, indicating that the pool of "fitting networks" does not change with increasing width. More class pairs of MNIST and CIFAR10 are provided in Appendix D.

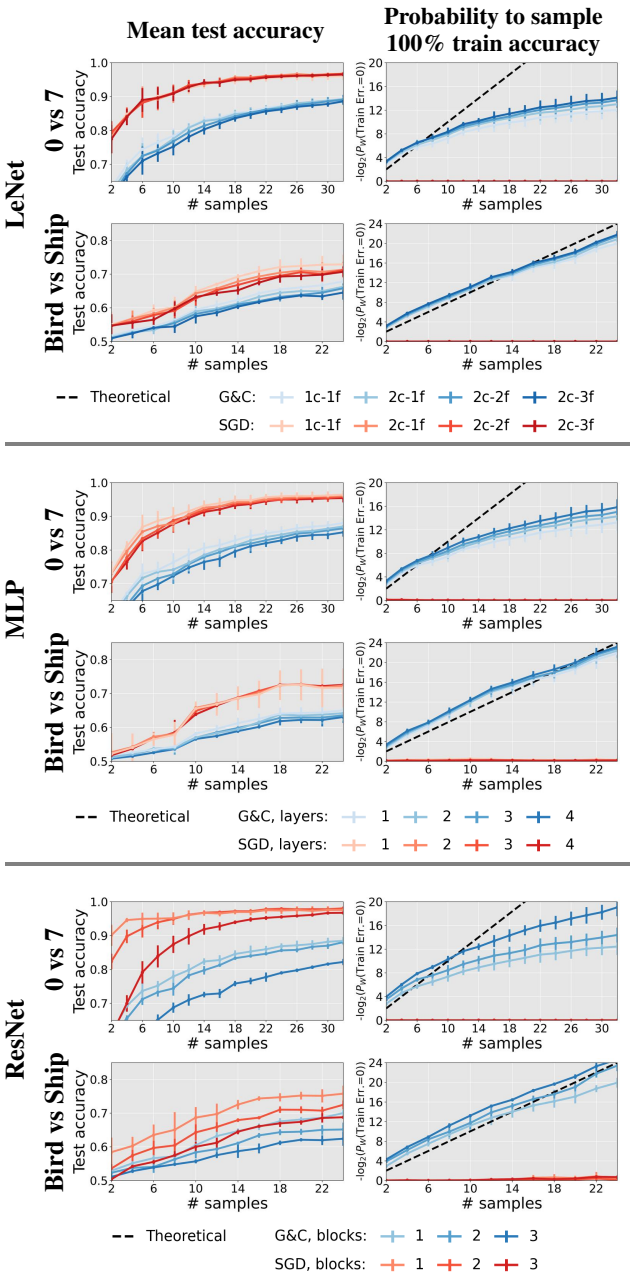


Figure 4. **Increasing depth is a negative architectural bias.** From top to bottom, the architectures are LeNet, MLP, and ResNet. Within each architecture, the first row corresponds to MNIST and the second to CIFAR10. Configuration "2c-1f" means two convolutional layers followed by a fully connected layer. **Left:** Mean test accuracy vs number of training samples across network depths. G&C always performs worse as depth increases, whereas SGD stagnates or gets worse. Thus, overparameterization in terms of depth results in overfitting instead of better generalization, unlike for the width. Since both G&C and SGD follow a similar trend, the decrease in performance with increased depth can be attributed to architectural bias. **Right:** Deeper networks have a lower probability for G&C to fit the training data, indicating that the network produces more complex functions. More class pairs of MNIST and CIFAR10 are provided in Appendix E.

set of training data (up to 32 samples for MNIST and 24 for CIFAR10). In fact, the result of Nguyen & Hein (2018) shows that if the number of neurons in the convolutional layer of the LeNet architecture exceeds the number of training points, then the set of parameters that do not produce linearly independent features at the convolutional layer has Lebesgue measure zero. This allows fitting 576 data points for MNIST and 784 data points for CIFAR10. Thus, we can see this experiment as starting with mild overparameterization for $1/6^*$ and increasing to heavy overparameterization for the original LeNet.

Comparison to the results of Chiang et al. (2023): Chiang et al. (2023) claim that widening the network widens the volume of good minima. They deduce this by observing that the average test error of G&C, conditional on a bin of their weight normalized loss, improves with increasing width, e.g., the boxes shown in the first row of Figure 2 for widths larger than $1/6$. However, as argued in Section 3.2, their weight normalized loss does not allow a direct comparison of different architectures conditional on loss bins, as it is only weakly related to the geometric properties of the function implemented by the network. Additionally, the comparison is complicated since some losses do not include samples, as evidenced by the empty boxes in Figure 2. To make it comparable across different architectures, we propose using our Lipschitz normalized loss.

We illustrate the discrepancy of both normalized losses in Figure 2, where we report the test accuracy vs the normalized loss for both the weight normalization of Chiang et al. (2023) and the Lipschitz normalization for four different widths $(1/6^*, 1/6, 2/6, 4/6)$. First, we observe that the plots of the weight normalized loss and Lipschitz normalized loss are completely different and also behave differently for varying widths. Indeed, we see an *improvement* of G&C in terms of average test accuracy for a fixed bin of the weight normalized loss, confirming the observation of Chiang et al. (2023). However, when looking at a fixed loss bin of the Lipschitz normalized loss across widths, the average test accuracy is *decreasing*, so we get the opposite statement of Chiang et al. (2023). It is noteworthy that the Lipschitz normalized loss decreases with width, indicating that the geometric margin to the decision boundary increases, strongly for SGD and only slightly for G&C. This shows that with increasing width, SGD has an implicit bias towards networks with larger geometric margins. The strong improvement in geometric margin is also linked to significant improvement in average test accuracy for SGD despite the *increasing* overparameterization.

Behavior of G&C for increasing width: Increasing the width of over-parameterized neural networks is known to improve the generalization of SGD (Hestness et al., 2017; Allen-Zhu et al., 2019). This is evident in the left column of Figure 3, where we observe a clear improvement of SGD across all scenarios.

Specifically, we report the average test accuracy of networks obtained by SGD and G&C as a function of increasing sample size for classes 0 vs 7 from MNIST and bird vs ship from CIFAR10, with the LeNet, MLP, and small ResNet architectures (See Appendix B.1 for details about the different architectures). For each training set size and each algorithm, we generate 100 fitting networks. The plotted curves are the average (and standard deviation) over four repetitions of this experiment, each with a different subset of training samples (See Appendix B.2 for more details).

In contrast to SGD, the test accuracy of G&C as a function of the width remains similar, except for a slight improvement in the LeNet architecture for CIFAR10 for widths up to $2/6$. This indicates that although G&C shows no clear sign of overfitting with increasing width, it also exhibits no further improvement with overparameterization. Moreover, even the probability $\mathrm{P}_W(\text{Train Error} = 0)$ of finding a perfectly fitting network (right column of Figure 3) remains constant. So, we can observe that even though the volume of random networks fitting the data does not seem to improve with the width, SGD manages to converge to better networks. **Thus, for increasing width, we have clear evidence of an implicit bias of SGD, which is helpful for improved generalization**. This bias results in convergence to networks of higher geometric margin (smaller Lipschitz normalized loss) with increasing width, as shown in Figure 2.

The orange dashed line in the plot of the test accuracy in Figure 3 shows the average test accuracy of the narrowest network for a significantly larger training set size of 400 samples (still perfectly fitting the training data). It shows that even for the narrowest network, there exists a set of weights that can lead to good generalization, but SGD does not converge to them. It is thus an interesting open question whether variants of SGD or novel optimizers could possess implicit biases that lead to better generalization even in narrower networks - or formulated differently: Is the current trend towards larger architecture only an artifact of the optimizer, or is it really necessary to achieve better generalization?

4.3. Overparameterization in Terms of Increasing Depth

Next, we explore the impact of overparameterization when adding layers to the network architecture. We work backward by starting with the $2/6$ architecture and then always discarding the largest layer. This choice is motivated by our observation in Section 4.2, where we noted a decline in G&C performance on CIFAR10 below this width for the LeNet architecture. The number of parameters in each configuration of the LeNet architecture can be found in Tables 3 and 4 for MNIST and CIFAR10, respectively. The results

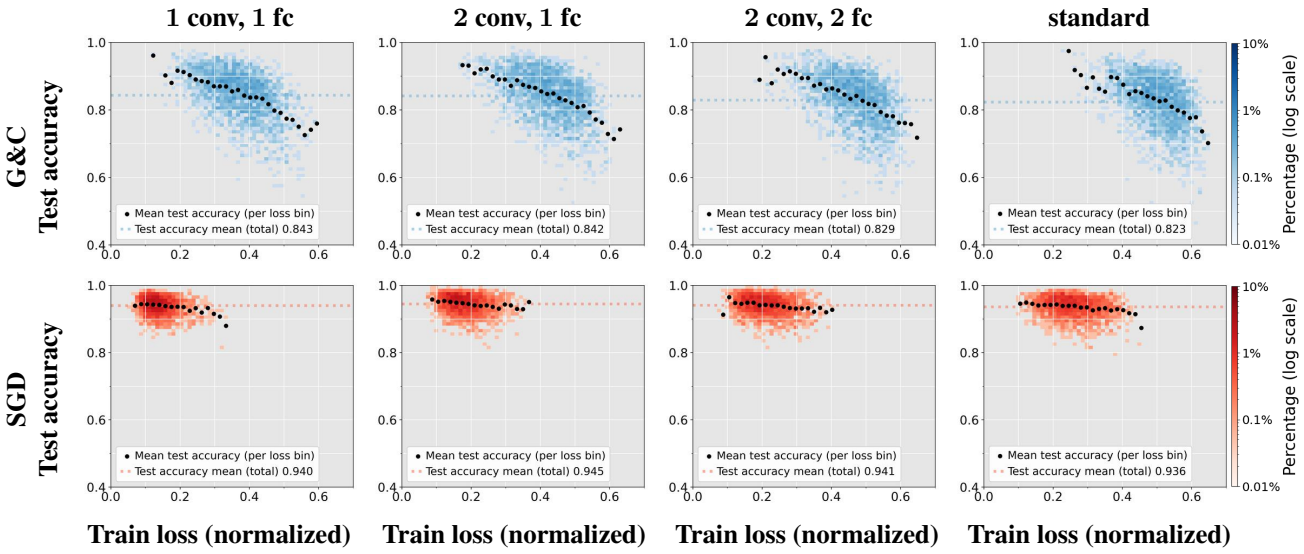


Figure 5. **Qualitative analysis of overparametrization in the depth.** In contrast to increasing width, increasing depth decreases the geometric margin (higher normalized loss). This decrease holds both for **G&C** (top) and **SGD** (bottom). We show 2000 LeNet models for each depth for classes 0 and 7 using a training set of size 16. Results are more pronounced for harder class pairs (see Figure 27).

are summarized in Figure 4 using a similar setup as the one described in the previous subsection for increasing width.

Interestingly, while overparameterizing the network in terms of width improved the result of SGD and at least did not hurt G&C performance, the effect of depth is the opposite. In the left column of Figure 4, we observe that G&C consistently performs worse with increasing depth, while the behavior of SGD either decreases or remains the same, depending on the architecture and the dataset. For example, in the LeNet architecture, for easy problems like 0 vs 7, SGD can compensate for the additional complexity due to depth with its bias towards more simple solutions. For more complex problems (CIFAR10, MNIST 3 vs 5, see also Figure 23 in the appendix), it suffers in the same way as G&C.

Thus, it seems that we see classical overfitting in overparameterized networks: more complex networks overfit on the small training set and, therefore, do not generalize as well. This is also reflected in $\mathrm{P}_W(\text{Train Error} = 0)$, as the probability of finding a shallow network fitting the data is much higher than for a deep one. Since G&C and SGD behave similarly in this context, one can state that there is a negative architectural bias with increasing depth, at least for the training set size considered here.

Note that we do not claim that increasing depth is detrimental in general. Instead, **we attribute the effect of depth to an architectural bias since it affects G&C and does not seem to influence the optimization process separately.** If sufficient data is available, it is well documented in the literature that deeper networks can yield strong generalization performance (He et al., 2016; Xie & Yuille, 2019). We speculate that this also holds for G&C, but it is out of reach to verify numerically due to the exponentially increasing cost of sampling.

Additionally, we present in Figure 5 the loss-accuracy plots for different depths, similar to the analysis in Figure 2. It can be seen that shallower networks with fewer parameters have lower Lipschitz normalized loss, which means larger geometric margins. This finding is in contrast with the results regarding overparameterization in terms of width, where wider networks with more parameters achieve larger geometric margins.

5. Conclusion

We performed an extensive comparison of SGD and G&C to disentangle the effects of the implicit bias of SGD versus the architectural bias of the network. Focusing on binary classification tasks in the low sample regime from the MNIST and CIFAR10 datasets, we demonstrated that the improvements of SGD with increasing overparameterization in terms of width can be attributed to its implicit bias, as G&C does not improve with width. Conversely, we showed that overparameterization in terms of depth is detrimental to generalization in the low sample regime and exhibits signs of classical overfitting. As SGD and G&C behave similarly in this case, this can be attributed to a negative architectural bias of the neural network. Disentangling the effects of optimization and architecture could provide deeper insights into why overparameterized networks perform so well in practice and ideally help to achieve similar generalization abilities with smaller networks.

Acknowledgements

The authors thank Václav Voráček for valuable discussions, Naama Pearl and Maximilian Müller for their assistance with writing the manuscript, and the International Max Planck Research School for Intelligent Systems (IMPRS-IS) for supporting AP. We acknowledge support from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy (EXC number 2064/1, project number 390727645), as well as in the priority program SPP 2298, project number 464101476.

Impact Statement

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

References

Advani, M. S., Saxe, A. M., and Sompolinsky, H. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 2020.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

Andriushchenko, M., Croce, F., Müller, M., Hein, M., and Flammarion, N. A modern look at the relationship between sharpness and generalization. In *ICML*, 2023a.

Andriushchenko, M., Varre, A. V., Pillaud-Vivien, L., and Flammarion, N. SGD with large step sizes learns sparse features. In *ICML*, 2023b.

Arora, S., Cohen, N., Hu, W., and Luo, Y. Implicit regularization in deep matrix factorization. In *NeurIPS*, 2019.

Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M. S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., and Lacoste-Julien, S. A closer look at memorization in deep networks. In *ICML*, 2017.

Bartlett, P., Long, P., Lugosi, G., and Tsigler, A. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 2020.

Bartlett, P. L. and Mendelson, S. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 2002.

Belkin, M., Hsu, D. J., Ma, S., and Mandal, S. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 2018.

Chiang, P.-y., Ni, R., Miller, D. Y., Bansal, A., Geiping, J., Goldblum, M., and Goldstein, T. Loss landscapes are all you need: Neural network generalization can be explained without the implicit bias of gradient descent. In *ICLR*, 2023.

Cybenko, G. V. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 1989.

Dziugaite, G. and Roy, D. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *UAI*, 2017.

Farhang, A. R., Bernstein, J., Tirumala, K., Liu, Y., and Yue, Y. Investigating generalization by controlling normalized margin. In *ICML*, 2022.

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *ICLR*, 2021.

Galanti, T. and Poggio, T. Sgd noise and implicit low-rank bias in deep neural networks. Technical report, Center for Brains, Minds and Machines (CBMM), 2022.

Geiping, J., Goldblum, M., Pope, P., Moeller, M., and Goldstein, T. Stochastic training is not necessary for generalization. In *ICLR*, 2022.

Haeffele, B. D. and Vidal, R. Global optimality in neural network training. In *CVPR*, 2017.

Hastie, T., Tibshirani, R., and Friedman, J. H. *The Elements of Statistical Learning*. Springer, 2001.

He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M. M. A., Yang, Y., and Zhou, Y. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Hoffer, E., Hubara, I., and Soudry, D. Train longer, generalize better: closing the generalization gap in large batch training of neural networks. In *NeurIPS*, 2017.

Hornik, K. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 1991.

Huang, W. R., Emam, Z., Goldblum, M., Fowl, L., Terry, J. K., Huang, F., and Goldstein, T. Understanding generalization through visualizations. In *NeurIPS Workshop "I Can't Believe It's Not Better!"*, 2020.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. Averaging weights leads to wider optima and better generalization. In *UAI*, 2018.

Jakubovitz, D., Giryes, R., and Rodrigues, M. R. Generalization error in deep learning. In *Compressed Sensing and Its Applications: Third International MATHEON Conference 2017*, pp. 153–193. Springer, 2019.

Jiang*, Y., Neyshabur*, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *ICLR*, 2020.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017a.

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *ICLR*, 2017b.

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

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Lee, J., Xiao, L., Schoenholz, S. S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent*. *Journal of Statistical Mechanics: Theory and Experiment*, 2020. URL https://dx.doi.org/10.1088/1742-5468/abc62b.

Li, Y., Wei, C., and Ma, T. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *NeurIPS*, 2019.

Liu, J., Jiang, G., Bai, Y., Chen, T., and Wang, H. Understanding why neural networks generalize well through gsnr of parameters. *ICLR*, 2020.

Mingard, C., Valle-Pérez, G., Skalse, J., and Louis, A. A. Is sgd a bayesian sampler? well, almost. *Journal of Machine Learning Research*, 2021.

Myrtle. How to train your resnet 4: Architecture - myrtle, 2019. URL https://github.com/davidcpage/cifar10-fast.

Neal, B., Mittal, S., Baratin, A., Tantia, V., Scicluna, M., Lacoste-Julien, S., and Mitliagkas, I. A modern take on the bias-variance tradeoff in neural networks, 2019. URL https://arxiv.org/pdf/1810.08591.pdf.

Neyshabur, B., Tomioka, R., and Srebro, N. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR*, 2015.

Nguyen, Q. and Hein, M. Optimization landscape and expressivity of deep CNNs. In *ICML*, 2018.

Pérez, G., Camargo, C., and Louis, A. Deep learning generalizes because the parameter-function map is biased towards simple functions. In *ICLR*, 2019.

Ruder, S. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

Shah, H., Tamuly, K., Raghunathan, A., Jain, P., and Netrapalli, P. The pitfalls of simplicity bias in neural networks. In *NeurIPS*, 2020.

Shalev-Shwartz, S. and Ben-David, S. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

Soudry, D., Hoffer, E., and Srebro, N. The implicit bias of gradient descent on separable data. In *ICLR*, 2018.

Vapnik, V. N. and Chervonenkis, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability & Its Applications*, 1971. URL https://doi.org/10.1137/1116025.

Wilson, A. G. and Izmailov, P. Bayesian deep learning and a probabilistic perspective of generalization. In *NeurIPS*, 2020.

Wolpert, D. H. Mathematics of generalization: Proceedings: Sfi-cnls workshop on formal approaches to supervised learning (1992: Santa fe, nm), 1995.

Xie, C. and Yuille, A. Intriguing properties of adversarial training at scale. In *ICLR*, 2019.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

A. Appendix

In the following sections of the appendix, we expand upon the experiments introduced in the main experiment section (Section 4). Initially, we provide details on the architectures and datasets in Appendix B. Appendix C further explores experiments related to the initialization analysis outlined in Section 4.1. Appendix D presents supplementary experiments for the width analysis discussed in Section 4.2. In Appendix E, we present additional experiments related to the depth analysis covered in Section 4.3.

B. Implementation Details

B.1. Network Architectures

B.1.1. LeNet

The LeNet architecture, as appears in LeCun et al. (1998), is depicted in Figure 6. The first two layers consist of convolutions with a kernel size of $5 \times 5 \times k$, where k represents the input channels' dimension of that layer. For MNIST, the first layer has 1 channel, and for CIFAR10, it has 3 channels. The remaining layers are fully connected.

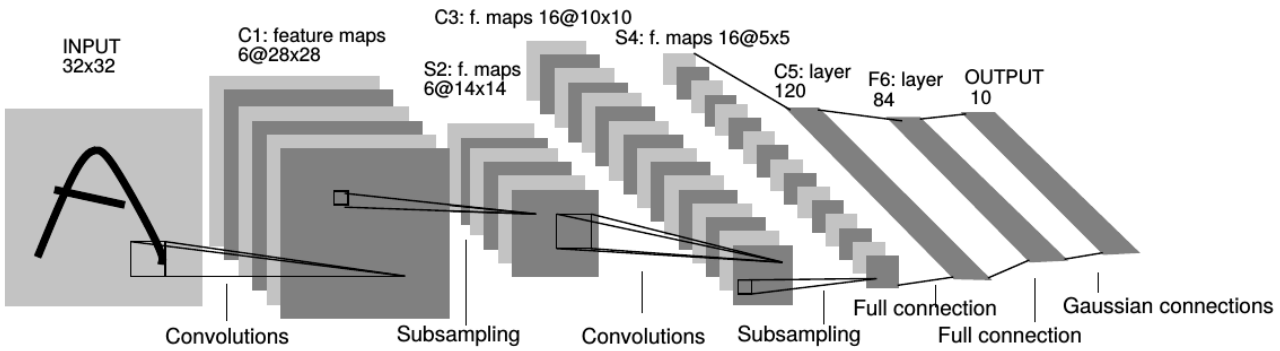


Figure 6. LeNet architecture. The figure is taken from the original paper (LeCun et al., 1998)

The original architecture's number of layers and parameter counts are summarized in the row indicated by a width factor of 1 in Table 1 (MNIST) and Table 2 (CIFAR10). Other rows illustrate the corresponding numbers for varying width factors (see Section 4.2). Additionally, we include the corresponding tables (Table 3 and Table 4) for varying numbers of layers (Section 4.3) for a width factor of $2/6$.

B.1.2. MLP

The MLP network has five layers with 120, 60, 30, 12, and 2 neurons, respectively. Since we want the network to be over-parametrized but remain within a feasible number of parameters, we resized the input images by 2 with a max pooling layer, yielding a network with 101,768 parameters for the CIFAR10 dataset and 33,128 for MNIST. Increasing the width is done in the same manner as for the LeNet. When changing the depth, we use a network with a width of 2/6 and gradually remove the largest layer, as was done for the LeNet architecture.

B.1.3. ResNet

For the ResNet-based architecture, we followed Myrtle (2019), which trains a custom ResNet architecture in 75 seconds with 94% accuracy on CIFAR10. The specific implementation contains three blocks, including a total of eight convolutional layers and one linear layer. We reduce the number of channels in each layer such that for a width factor of 1 and CIFAR10, the network contains 58,350 learnable parameters (similar to the LeNet). As with the other architectures, we used a constant learning rate and avoided augmentations of the dataset.

Table 1. **Parameter count in the LeNet architecture for the MNIST task with varying *width* factors**. A width factor of 1 indicates the standard LeNet architecture. The corresponding parameter numbers in the convolutional layers are indicated in parentheses. In the last fully connected layer, the parameter count solely depends on the input size and preceding layers. For improved readability, bias terms are excluded from the per-layer parameter count.

Width	Convolutions layers		Fully connected layers			Total
1/6*	1 (25)	2 (50)	5 (160)	3 (15)	2 (6)	269
1/6	1 (25)	2 (50)	20 (640)	14 (280)	2 (28)	1062
2/6	2 (50)	5 (250)	40 (3200)	28 (1120)	2 (56)	4753
3/6	3 (75)	8 (600)	60 (7680)	42 (2520)	2 (84)	11074
4/6	4 (100)	10 (1000)	80 (12800)	56 (4480)	2 (112)	18644
5/6	5 (125)	13 (1625)	100 (20800)	70 (7000)	2 (140)	29880
1	6 (150)	16 (2400)	120 (30720)	84 (10080)	2 (168)	43746

Table 2. **Parameter count in the LeNet architecture for the CIFAR10 task with varying *width* factors**.

Width	Convolutions layers		Fully connected layers			Total
1/6*	1 (75)	2 (50)	5 (250)	3 (15)	2 (6)	409
1/6	1 (75)	2 (50)	20 (1000)	14 (280)	2 (28)	1472
2/6	2 (150)	5 (250)	40 (5000)	28 (1120)	2 (56)	6653
3/6	3 (225)	8 (600)	80 (12000)	42 (2520)	2 (84)	15544
4/6	4 (300)	10 (1000)	80 (20000)	56 (4480)	2 (112)	26044
5/6	5 (375)	13 (1625)	100 (32500)	70 (7000)	2 (140)	41830
1	6 (450)	16 (2400)	120 (48000)	84 (10080)	2 (168)	61326

B.2. Dataset

For each binary classification task, we used a subset of the training images and the entire test set for the class pair. For instance, for MNIST 0 vs 7 with 16 training samples, a subset of 16 samples was selected from the original training set (eight images of 0 and eight images of 7), and the test set contains the entire original test set of 2008 images. For the loss vs test accuracy plots (e.g., Figures 1, 2, 5), the same subset was used across all experiments to ensure that randomnesses stem only from the initialization (G&C) and the additional randomness of the algorithm (SGD). For the samples vs test accuracy plots (e.g., Figures 3, 4), we average results over four different training subsets (same four subsets for each experiment). Additionally, when increasing the number of samples, we only add images to the existing sets, maintaining the previous training images (i.e., the training sets are increasing sets).

C. Dependency of SGD and G&C on the Prior $P(W)$

In Section 4.1, we analyze the effect of different initializations on the generalization capabilities of SGD and G&C. This section extends the experiments in Section 4.1 in the main paper.

First, we show in Figure 7 the norm of the gradient of the weights along the training process for different initializations. This figure illustrates that SGD essentially stops the optimization early in the training process with sub-optimal initialization. Next, we show the result of Figure 1 for another pair of classes from the MNIST dataset (Figure 8), two pairs of classes from the CIFAR10 dataset (Figure 9), and different numbers of samples (Figure 10). In Figure 11, we show the same result as in Figure 1, but with the *weight normalized loss* taken from (Chiang et al., 2023). Figure 12 shows the loss evolution along different epochs of SGD, which was initialized with Kaiming uniform distribution. Finally, we show in Figure 13 the complementary experiments of the discussion in Section 4.1 regarding the initialization of SGD with already "trained" networks of G&C.

Table 3. **Parameter count in the LeNet architecture for the MNIST task with varying *depth* factors**. The *width* of the network for all of these experiments is 2/6. The first row is the standard layers configuration of two convolution layers, followed by 3 fully connected layers. *c* and *f* represent a convolutional layer and a fully connected layer, respectively.

Layers	Convolutions layers		Fully connected layers			Total
2c-3f	2 (50)	5 (250)	40 (3200)	28 (1120)	2 (56)	4753
2c-2f	2 (150)	5 (250)	—	28 (2240)	2 (56)	2263
2c-1f	2 (150)	5 (250)	—	—	2 (160)	469
1c-1f	2 (50)	—	—	—	2 (144)	198

Table 4. **Parameter count in the LeNet architecture for the CIFAR10 task with varying *depth* factors**.

Layers	Convolutions layers		Fully connected layers			Total
2c-3f	2 (150)	5 (250)	40 (5000)	28 (1120)	2 (56)	6653
2c-2f	2 (150)	5 (250)	—	28 (3500)	2 (56)	3993
2c-1f	2 (150)	5 (250)	—	—	2 (250)	659
1c-1f	2 (150)	—	—	—	2 (196)	350

D. Overparameterization with Increasing Width

In the following section, we expand the experiments of the network *width* analysis from section 4.2.

The parameter counts for the LeNet architecture across various *width* values are detailed for the MNIST dataset in Table 1 and the CIFAR10 dataset in Table 2. Note that the number of parameters in the linear layers depends on the dimension of the input. The number of convolutions and neurons is determined by multiplying the number at width 1 by the width factor and rounding down the result. The notation $1/6^*$ indicates a width factor of $1/6$ for the convolution layers and a width factor of $1/24$ for the fully connected layers, given that there is already a single convolution in the first layer when using width factor $1/6$.

Figure 14 and Figure 15 extend the results shown in Figure 2 for different sample sizes. Figure 16 provides results for another pair of classes from MNIST. Similarly, in Figure 17 and Figure 18 we follow the analysis of Figure 2 for two additional classes of CIFAR10.

We further examine the performance of various network widths along different training set sizes. Figures 19 and 20 replicate the analysis from Figure 3 for the LeNet architecture but for different class pairs from the MNIST and CIFAR10 datasets, Figure 21 replicates the same for MLP. Moreover, Figure 22 presents results for classes 0 and 7 from MNIST with a training size of 128 to illustrate the consistency of our findings for larger training sets.

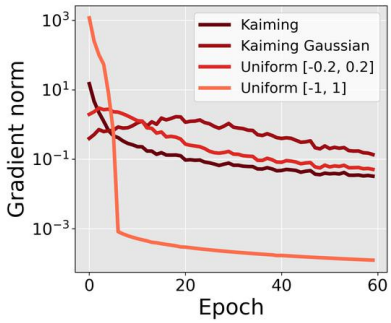


Figure 7. $\mathcal{U}[-1, 1]$ **initialization prevents effective gradient-based learning.** When using $\mathcal{U}[-1, 1]$, the initialized weights are large, leading to high logit values and, in turn, to low loss and small gradients as soon as one reaches zero training loss, effectively stopping SGD. For proper initialization, like the commonly used Kaiming or the downscaled $\mathcal{U}[-0.2, 0.2]$, logit values are lower, and SGD optimizes properly, see Section 4.1 for more details.

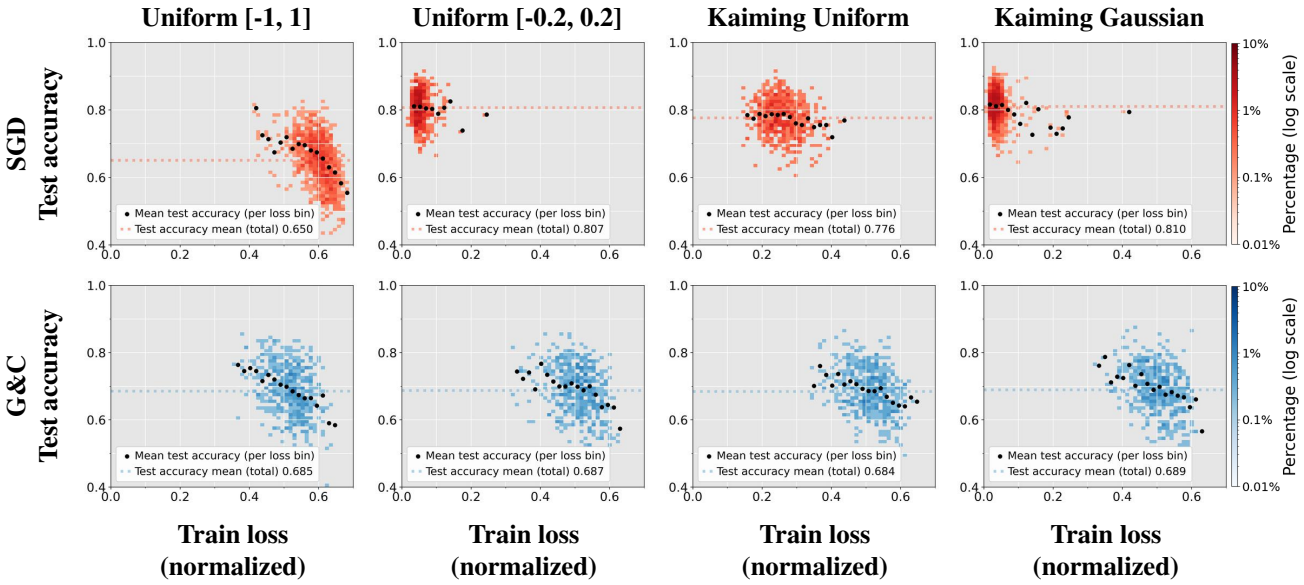


Figure 8. **Generalization of SGD (optimized) versus G&C (randomly sampled) in dependency of the prior on the weights $\mathrm{P}(W)$:** We "train" 500 models to 100% train accuracy for 16 training samples from classes *3* and *5* of the MNIST dataset. Test accuracies for G&C are similar across initializations, and the Lipschitz normalized loss (see Sec. 3) is similar across the uniform distributions. **Column 1:** For $\mathcal{U}[-1, 1]$ initialization (used in Chiang et al. (2023)), the Lipschitz normalized losses and the test accuracy of SGD and G&C are similar, except for the convergence of SGD towards more low margin solutions. The claim in Chiang et al. (2023) that G&C resembles SGD, conditional on average test accuracy in the normalized loss bin (black dots), is an artifact of the suboptimal convergence of SGD caused by this initialization. **Columns 2-4:** For other initializations, SGD (first row) improves considerably both in terms of loss and accuracy. In contrast, G&C remains unaffected, as it is independent of the weight scale in each layer.

E. Overparameterization with Increasing Depth

In the following section, we elaborate on the experiments of the network *depth* analysis described in section 4.3.

The different layer configurations for the LeNet, specifically the number of convolutional and fully connected layers, along with the corresponding parameter counts are depicted for the MNIST dataset in Table 3 and for the CIFAR10 dataset in Table 4. These experiments consider a *width* of $2/6$ for the network, as detailed in sections 4.2 and Appendix D. In our notation, c represents a convolutional layer, and f represents a fully connected layer.

In Figure 23, we show the performance of the LeNet with different numbers of layers along various sizes of training sets, as in Figure 4, but for other classes from the MNIST and CIFAR10 datasets. In Figure 24, we do the same for MLP. Additionally, in Figure 25, we present this for classes 0 and 7 for a training set of 128 to illustrate that the results hold for a larger number of samples. In Figures 26 and 27, we display the loss changes as in Figure 5, but for a different number of samples and different pairs of classes, respectively.

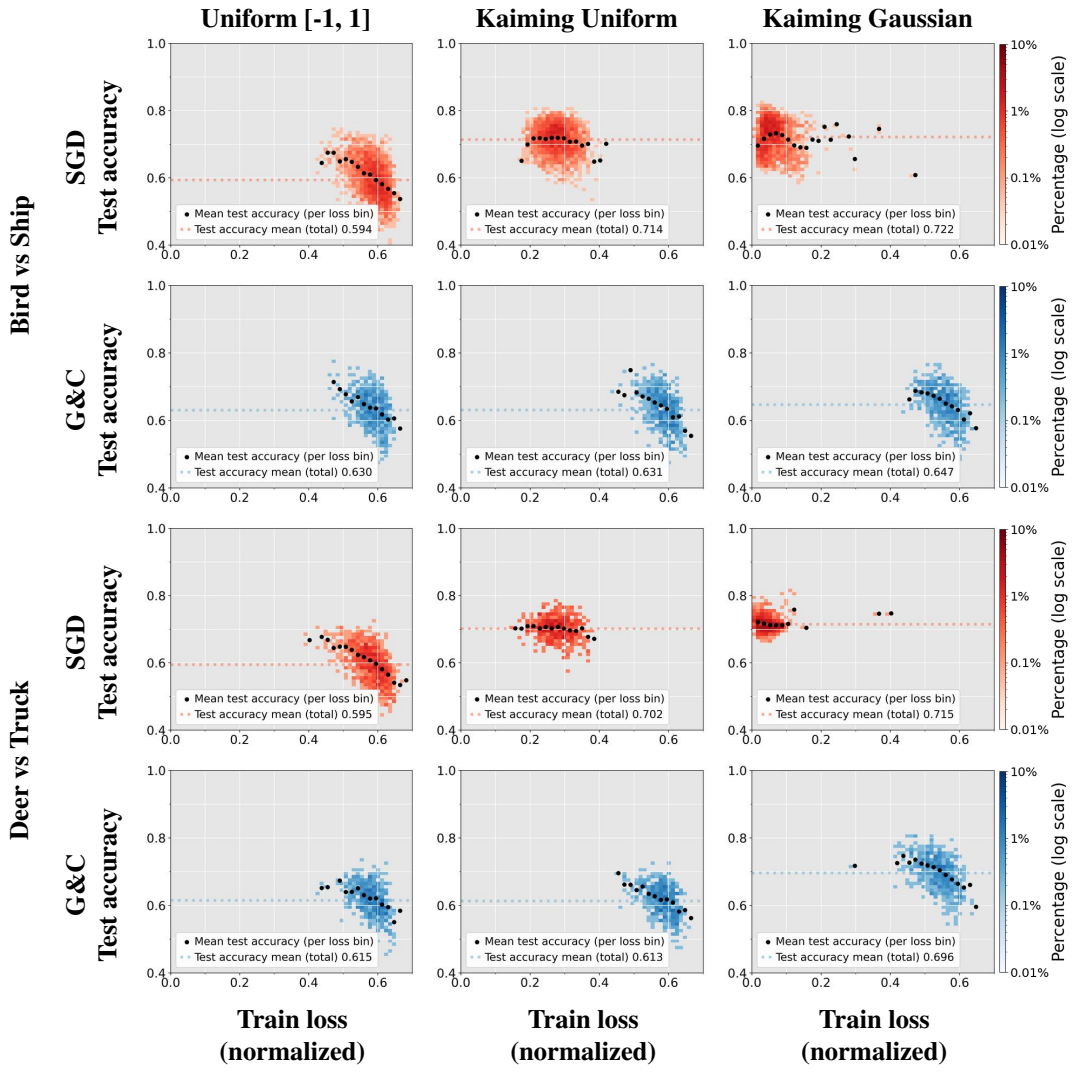


Figure 9. **Generalization of SGD (optimized) versus G&C (randomly sampled) in dependency of the prior on the weights $\mathrm{P}(W)$:** We "train" 500 models to 100% train accuracy for 16 training samples for different pairs of classes from the **CIFAR10** dataset. **Rows 1-2:** classes *bird* and *ship*. **Rows 3-4:** classes *Deer* and *Truck*. Test accuracies and the normalized Lipschitz losses (see Sec. 3) for G&C are similar across uniform initializations. **Column 1:** For $\mathcal{U}[-1,1]$ initialization (used in Chiang et al. (2023)), the Lipschitz normalized losses and the test accuracy of SGD and G&C are similar, except for the convergence of SGD towards more low margin solutions. The claim in Chiang et al. (2023) that G&C resembles SGD, conditional on average test accuracy in the normalized loss bin (black dots), is an artifact of the suboptimal convergence of SGD caused by this initialization. **Column 2:** For Kaiming uniform, SGD (first and third rows) improves considerably both in terms of loss and accuracy. In contrast, G&C (second and fourth rows) remains unaffected, as it is independent of the weight scale in each layer. **Column 3:** For Kaiming Gaussian, there is an improvement for both G&C and SGD, but they still arrive at different losses and accuracies.

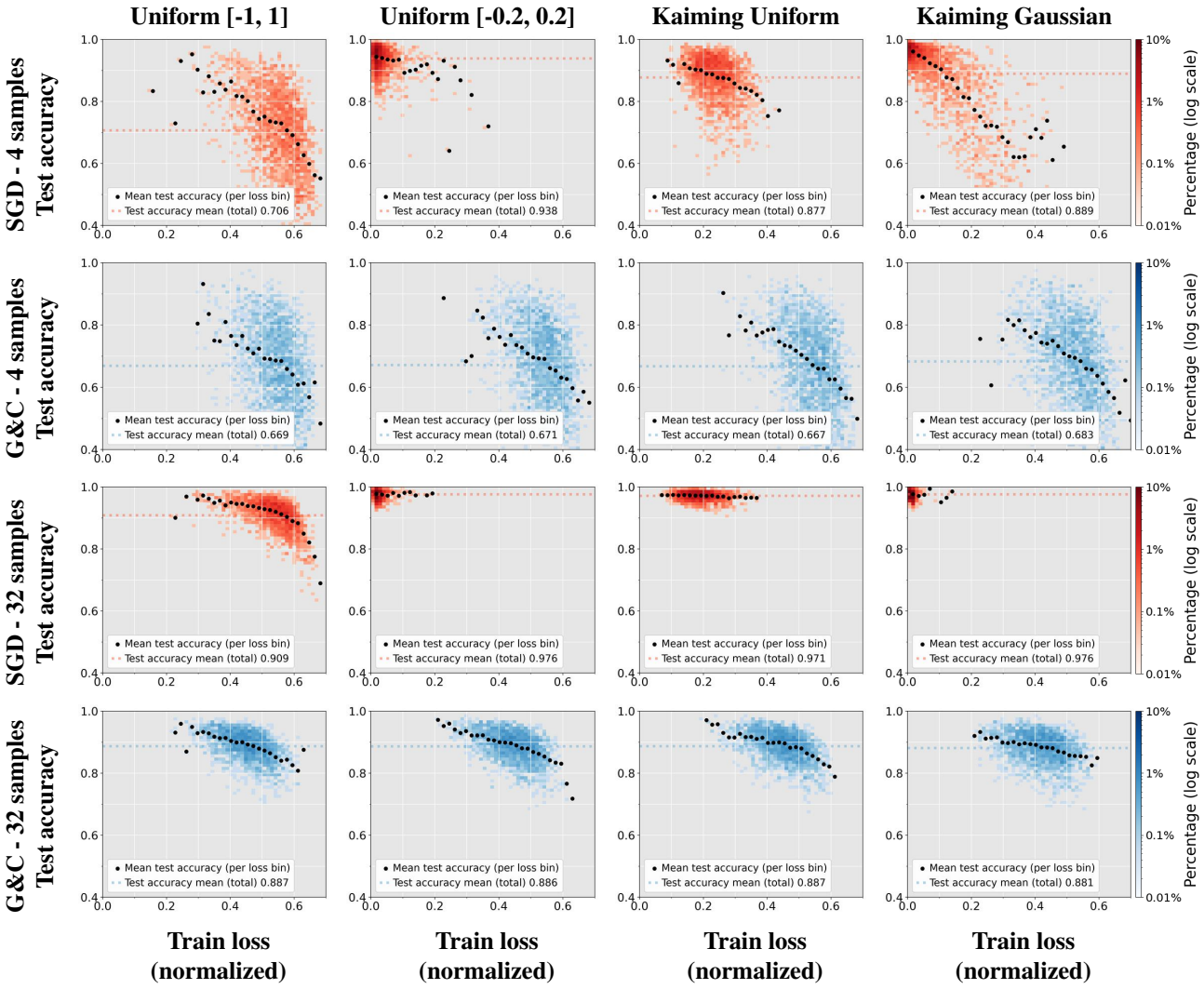


Figure 10. **Generalization of SGD (optimized) versus G&C (randomly sampled) in dependency of the prior on the weights $\mathrm{P}(W)$:** We "train" 2000 models to 100% train accuracy for 4 and 32 training samples from classes *0* and *7* of the MNIST dataset. Test accuracies for G&C are similar across initializations, and the normalized Lipschitz loss (see Sec. 3) is similar across the uniform distributions. **Column 1:** For $\mathcal{U}[-1,1]$ initialization (used in Chiang et al. (2023)), the normalized Lipschitz losses and the test accuracy of SGD and G&C are similar, except for the convergence of SGD towards more low margin solutions. The claim in Chiang et al. (2023) that G&C resembles SGD, conditional on average test accuracy in the normalized loss bin (black dots), is an artifact of the suboptimal convergence of SGD caused by this initialization. **Columns 2-4:** For other initializations, SGD (first row) improves considerably both in terms of loss and accuracy. In contrast, G&C remains unaffected, as it is independent of the weight scale in each layer.

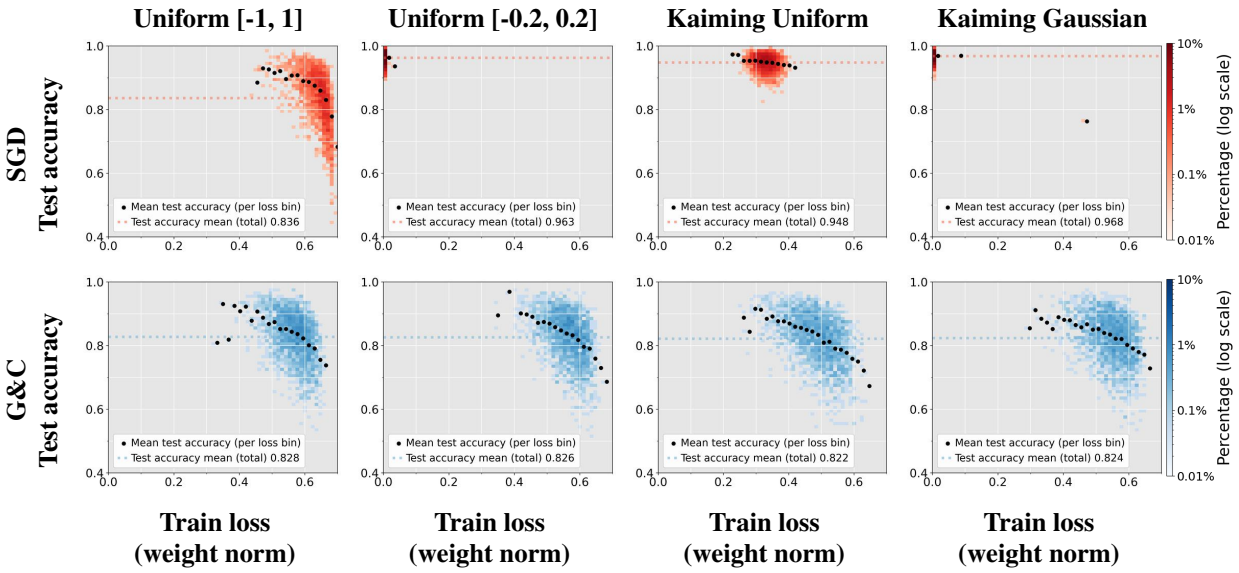
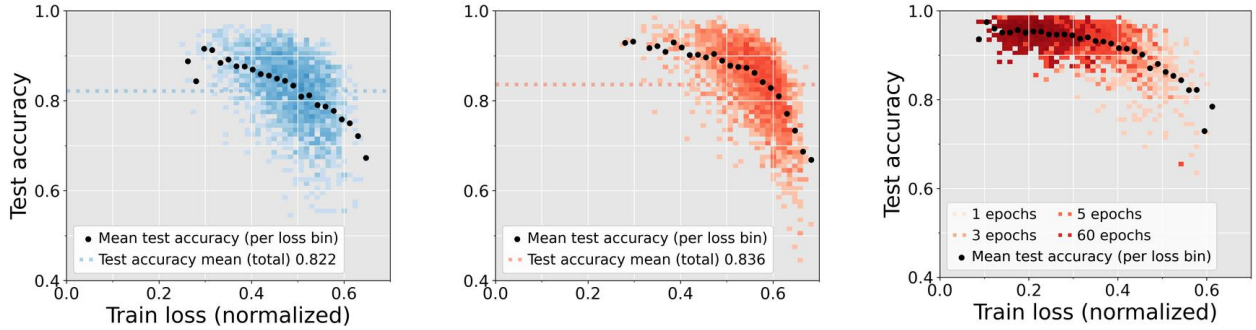


Figure 11. **Generalization of SGD (optimized) versus G&C (randomly sampled) in dependency of the prior on the weights $\mathrm{P}(W)$:** We "train" 2000 models to 100% train accuracy for 16 training samples from classes *0* and *7* of the MNIST dataset. Test accuracies for G&C are similar across initializations, and the weight normalized loss (see Sec. 3) is similar across the uniform distributions. **Column 1:** For $\mathcal{U}[-1, 1]$ initialization (used in Chiang et al. (2023)), the weight normalized losses and the test accuracy of SGD and G&C are similar, except for the convergence of SGD towards more low margin solutions. The claim in Chiang et al. (2023) that G&C resembles SGD, conditional on average test accuracy in the normalized loss bin (black dots), is an artifact of the suboptimal convergence of SGD caused by this initialization. **Columns 2-4:** For other initializations, SGD (first row) improves considerably both in terms of loss and accuracy. In contrast, G&C remains unaffected, as it is independent of the weight scale in each layer. **This graph uses the weight norm normalization (Chiang et al., 2023).**

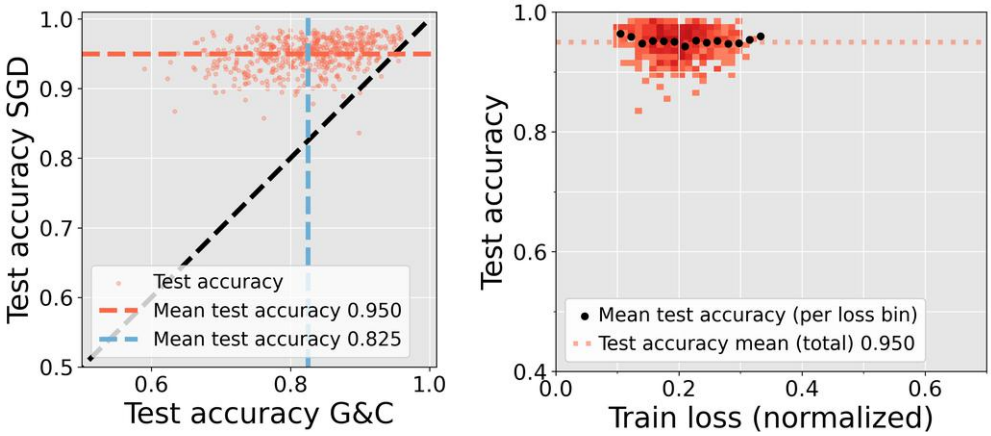


(a) Train loss vs. Test accuracy of G&C initialized with Kaiming uniform distribution.

(b) Train loss vs. Test accuracy of SGD initialized with $\mathcal{U}[-1, 1]$.

(c) Train loss vs. Test accuracy of SGD with Kaiming uniform initialization along different epochs.

Figure 12. Train loss vs. Test accuracy of SGD with Kaiming uniform initialization along different epochs next to G&C. This configuration of SGD resembles the one from $\mathcal{U}[-1, 1]$, as models from earlier epochs still have not converged. In spite of that, the results of SGD are better than the ones of G&C given a loss bin, and on average.



(a) (b)

Figure 13. **Initialization of SGD with G&C networks does not improve generalization.** We initialize SGD with 500 models obtained by G&C (randomly sampled networks from Kaiming uniform with 100% training accuracy) for MNIST with classes *0* and *7* and 16 training samples. **Left**: Most models improve through optimization (above the black line). Thus, SGD converges to a smaller volume of perfectly fitting networks that generalize better, resulting in a mean test accuracy of 95% for SGD instead of 82.5% for G&C. **Right**: distribution of test accuracy versus normalized loss is similar to SGD with Kaiming uniform initialization (Figure 1). Thus, initializing with G&C is not better than random initialization.

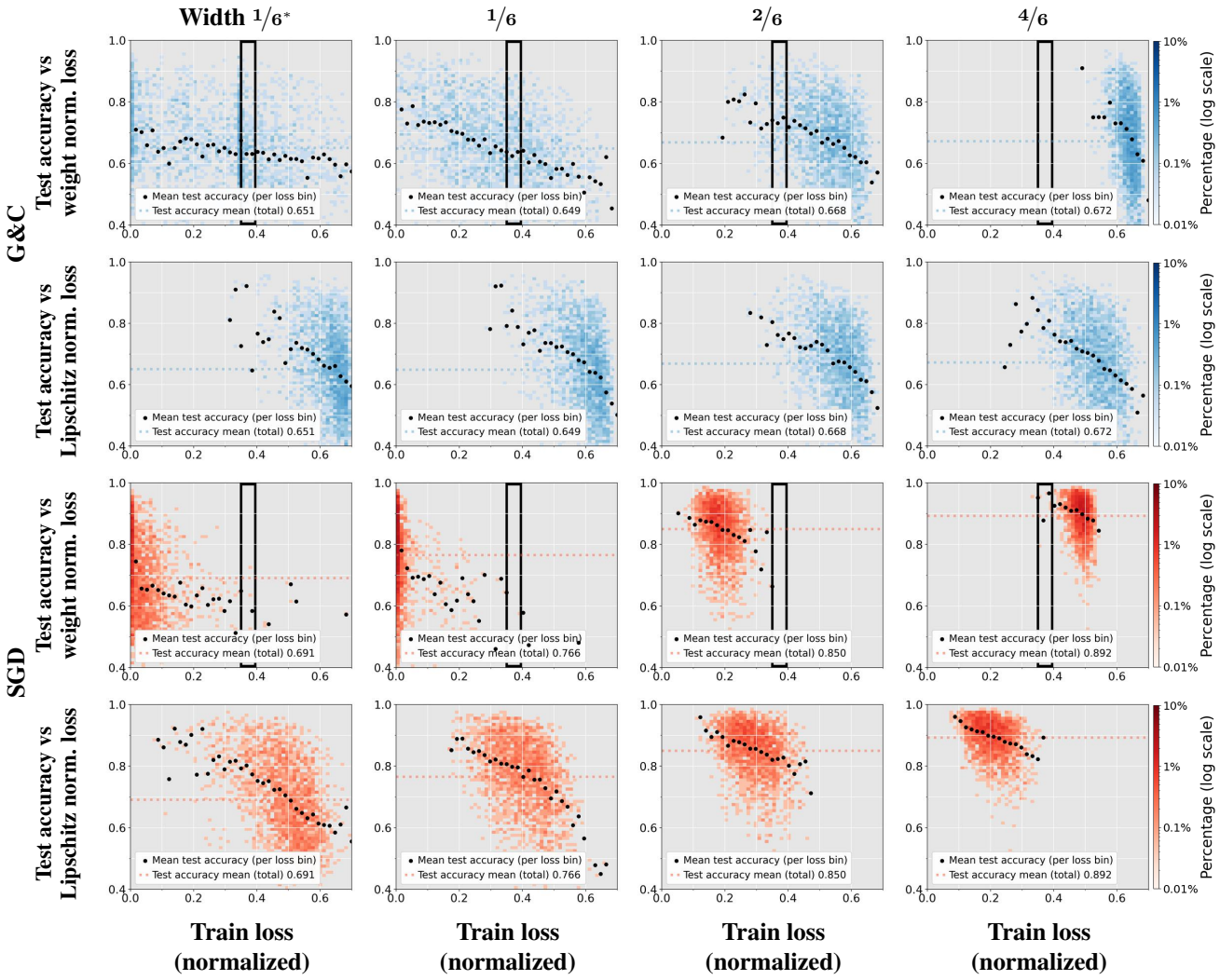


Figure 14. **Analysis of overparameterization when increasing the width.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (2) of SGD and G&C for classes *0* & *7* of MNIST and 4 training samples. **Row 4:** Widening the networks enhances geometric margin and average test accuracy for SGD, while for G&C **(Row 2)**, the margin and average test accuracy improve only slightly. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias (see Fig. 3). **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by black boxes), which led them to conclude that G&C improves with increasing width. With our Lischitz normalized loss, one would arrive at the opposite conclusion, which shows the problem of normalization.

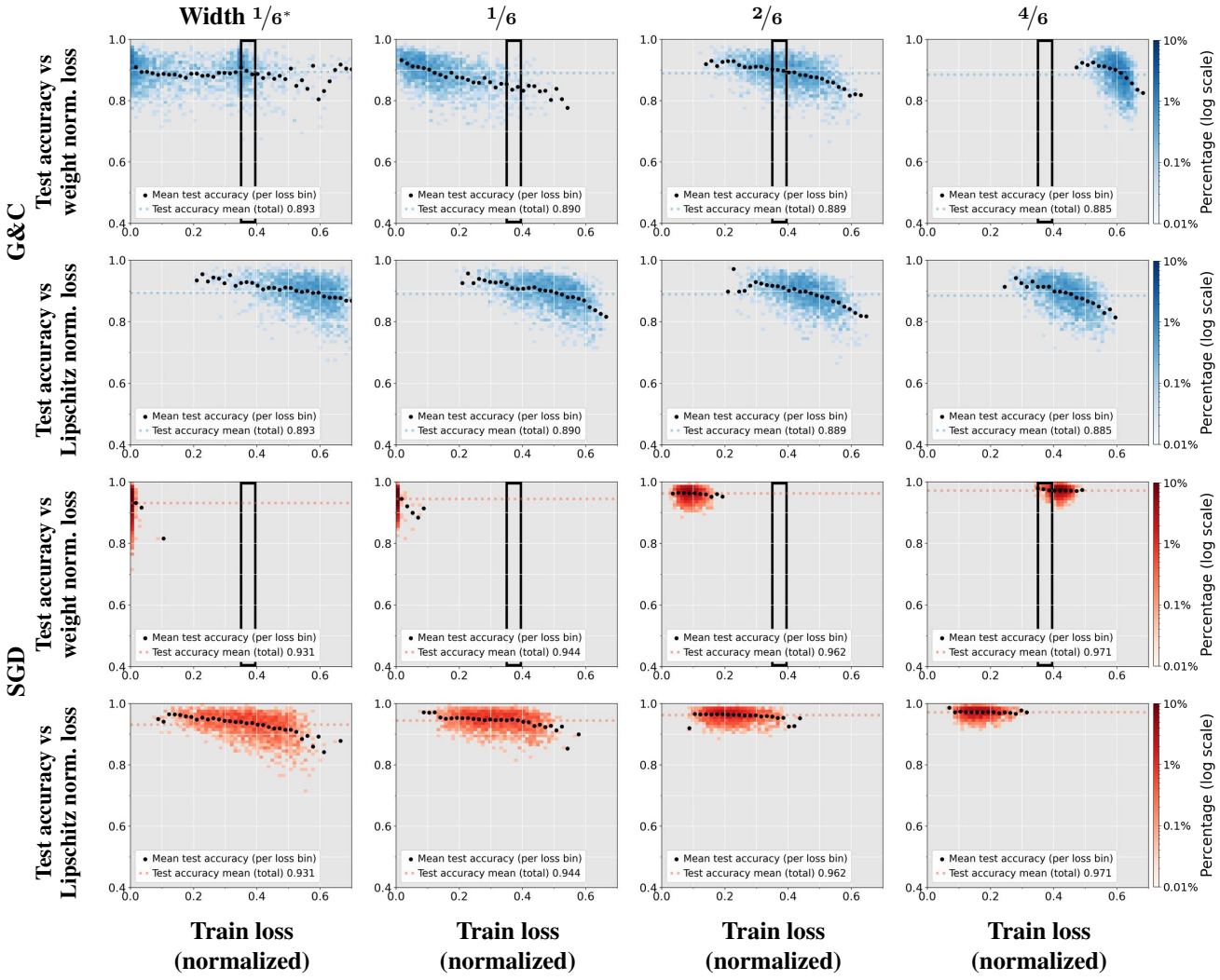


Figure 15. **Analysis of overparameterization when increasing the width.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (2) of SGD and G&C for classes *0* & *7* of MNIST and 32 training samples. **Row 4:** Widening the networks enhances geometric margin and average test accuracy for SGD, while for G&C **(Row 2)**, the margin improves only slightly, and average test accuracy remains the same. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias (see Fig. 3). **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by black boxes), which led them to conclude that G&C improves with increasing width. With our Lischitz normalized loss, one would arrive at the opposite conclusion, which shows the problem of normalization.

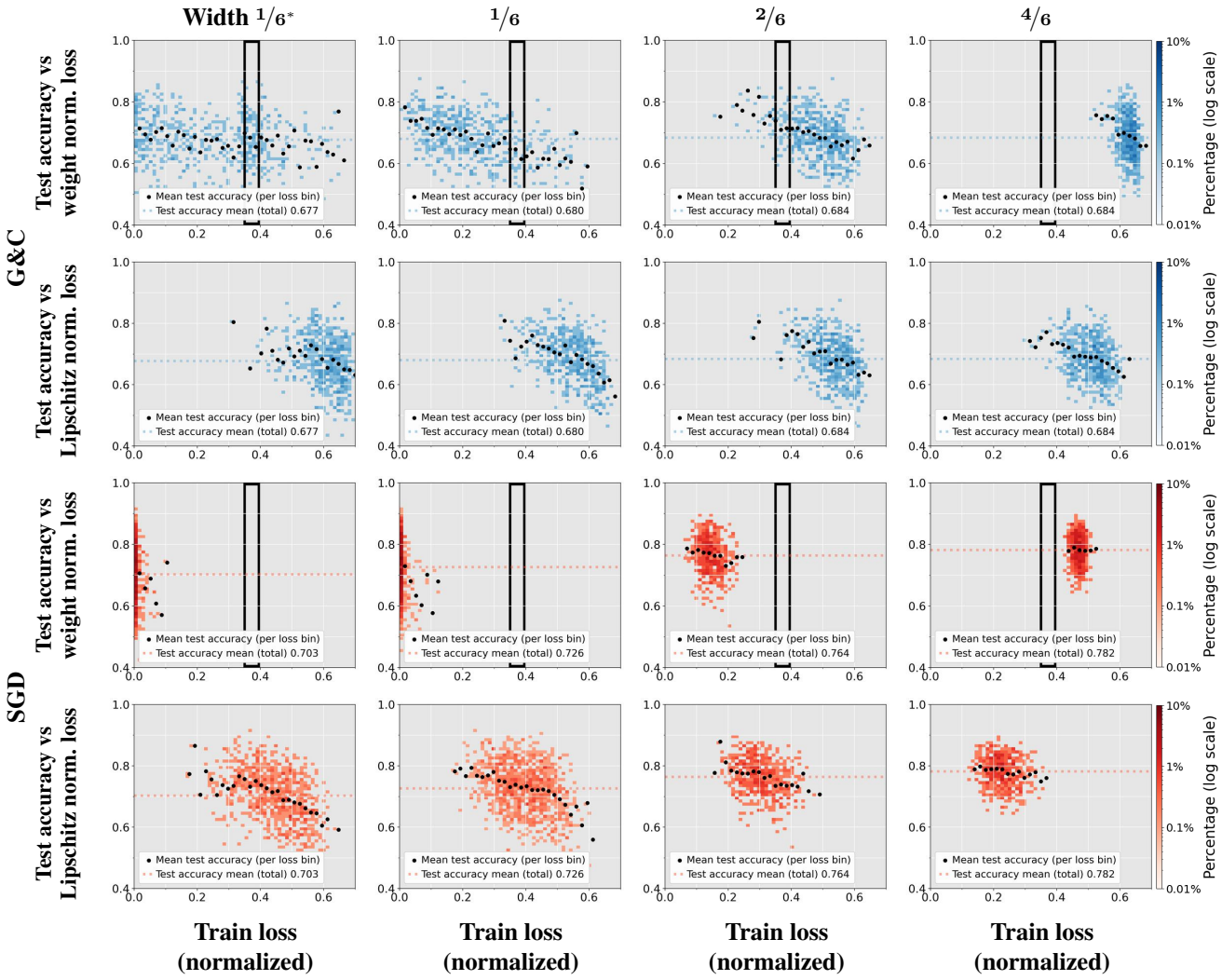


Figure 16. **Qualitative analysis of overparameterization when widening the networks.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (1) of **SGD** and **G&C** for classes *3* & *5* of MNIST and 16 training samples. **Bottom row:** Widening the networks enhances both geometric margin and average test accuracy for SGD, while for G&C **(second row)**, margin and average test accuracy improve only slightly. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias. **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by the black boxes), which led them to the conclusion of an increased volume of good minima for wider networks. However, with our Lischitz normalized loss, one would arrive at the opposite conclusion, which shows the problems of comparisons across network architectures.

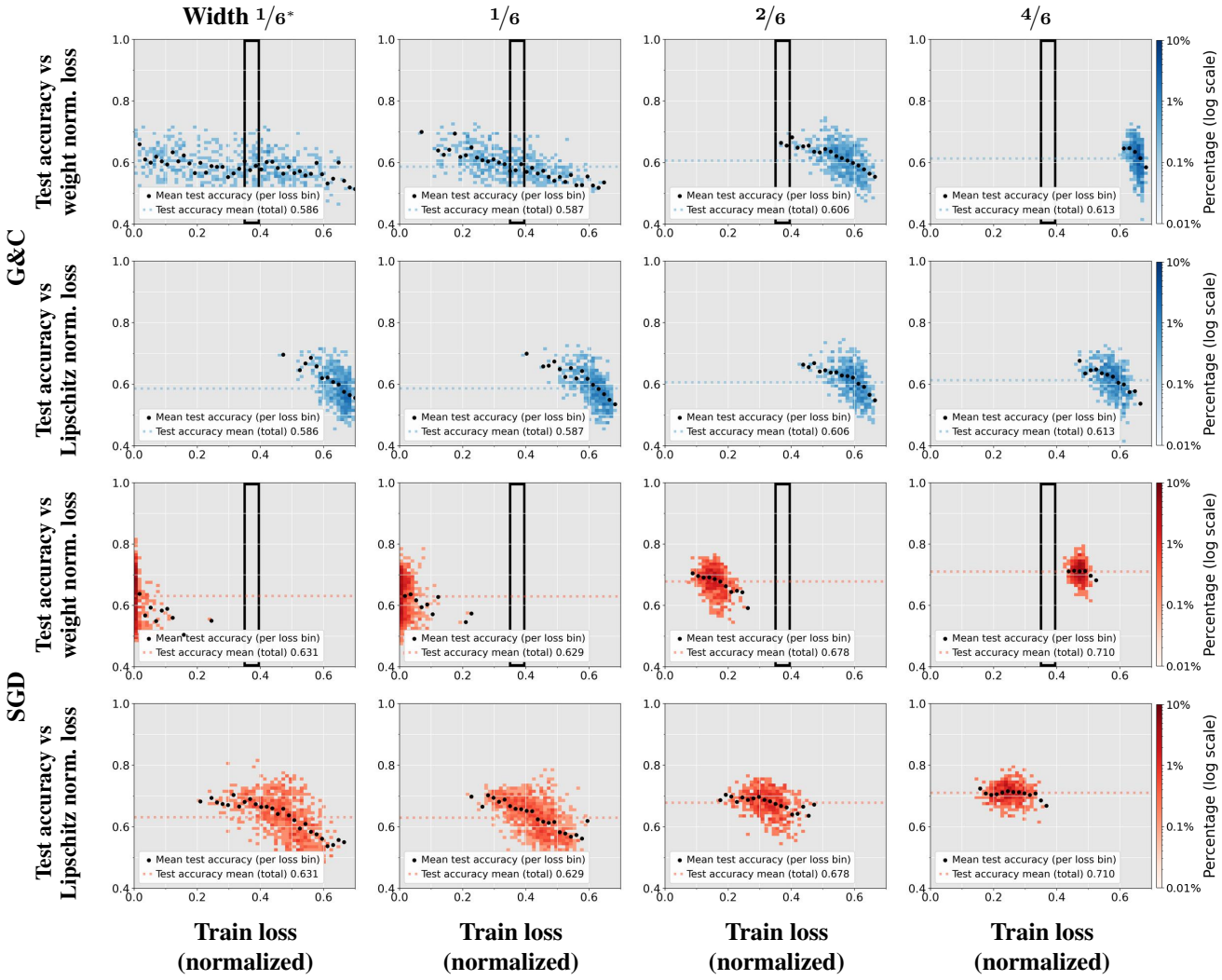


Figure 17. **Qualitative analysis of overparameterization when widening the networks.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (1) of **SGD** and **G&C** for classes *Deer* & *Truck* of CIFAR10 and 16 training samples. **Bottom row:** Widening the networks enhances both geometric margin and average test accuracy for SGD, while for G&C **(second row)**, margin and average test accuracy improve up to a width of $2/6$ and then only slightly. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias. **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by the black boxes), which led them to the conclusion of an increased volume of good minima for wider networks. However, with our Lischitz normalized loss, one would arrive at the opposite conclusion, which shows the problems of comparisons across network architectures.

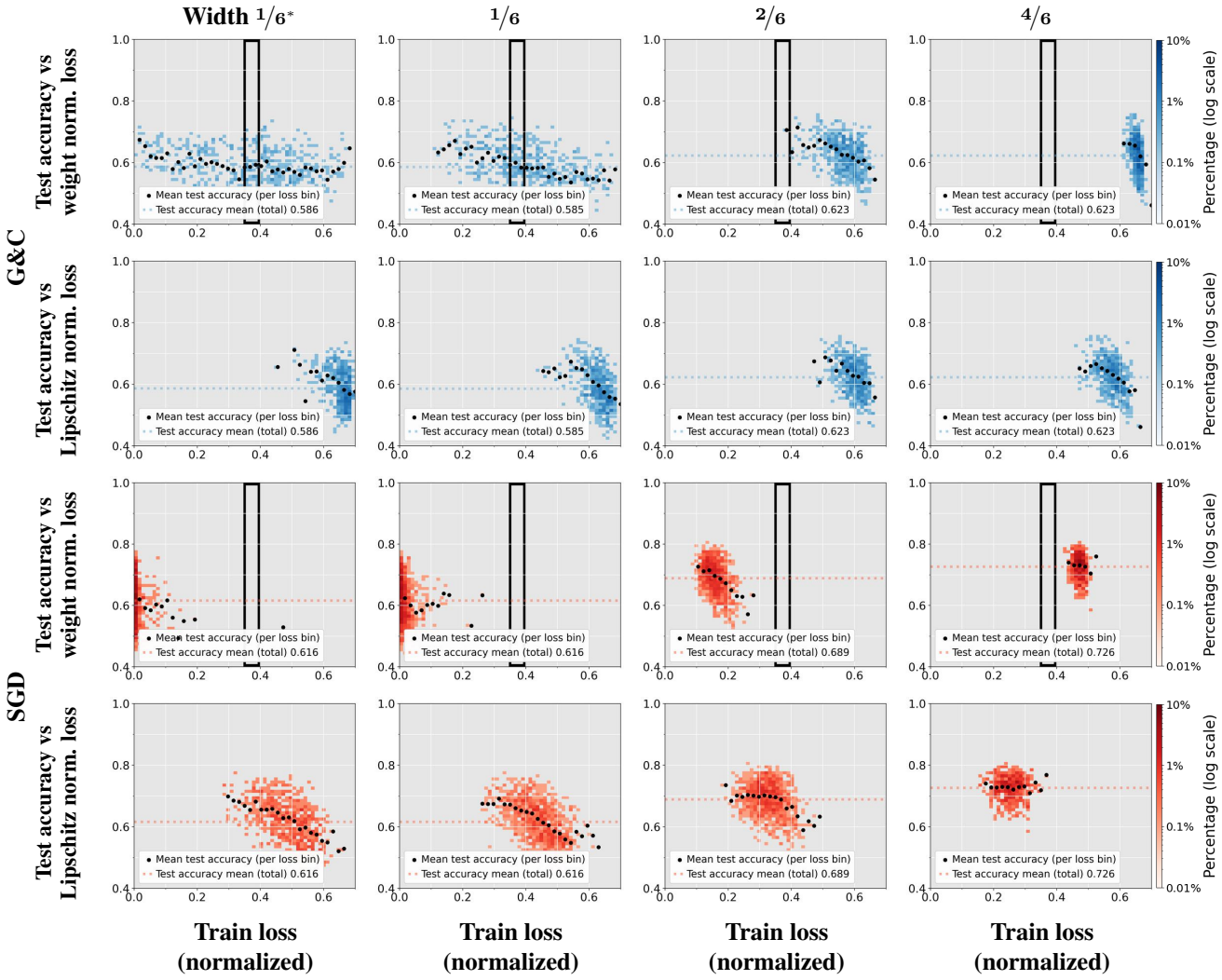


Figure 18. **Qualitative analysis of overparameterization when widening the networks.** Test accuracy vs weight normalized loss (3) of Chiang et al. (2023) and our Lipschitz normalized loss (1) of **SGD** and **G&C** for classes *Bird* & *Ship* of CIFAR10 and 16 training samples. **Bottom row:** Widening the networks enhances both geometric margin and average test accuracy for SGD, while for G&C **(second row)**, margin and average test accuracy improve up to a width of $2/6$ and then only slightly. This suggests that the improvement is mainly due to the bias of SGD and not due to an architectural bias. **Rows 1 and 3:** Chiang et al. (2023) compare networks conditional on the (weight) normalized loss bin (illustrated by the black boxes), which led them to the conclusion of an increased volume of good minima for wider networks. However, with our Lischitz normalized loss, one would arrive at the opposite conclusion, which shows the problems of comparisons across network architectures.

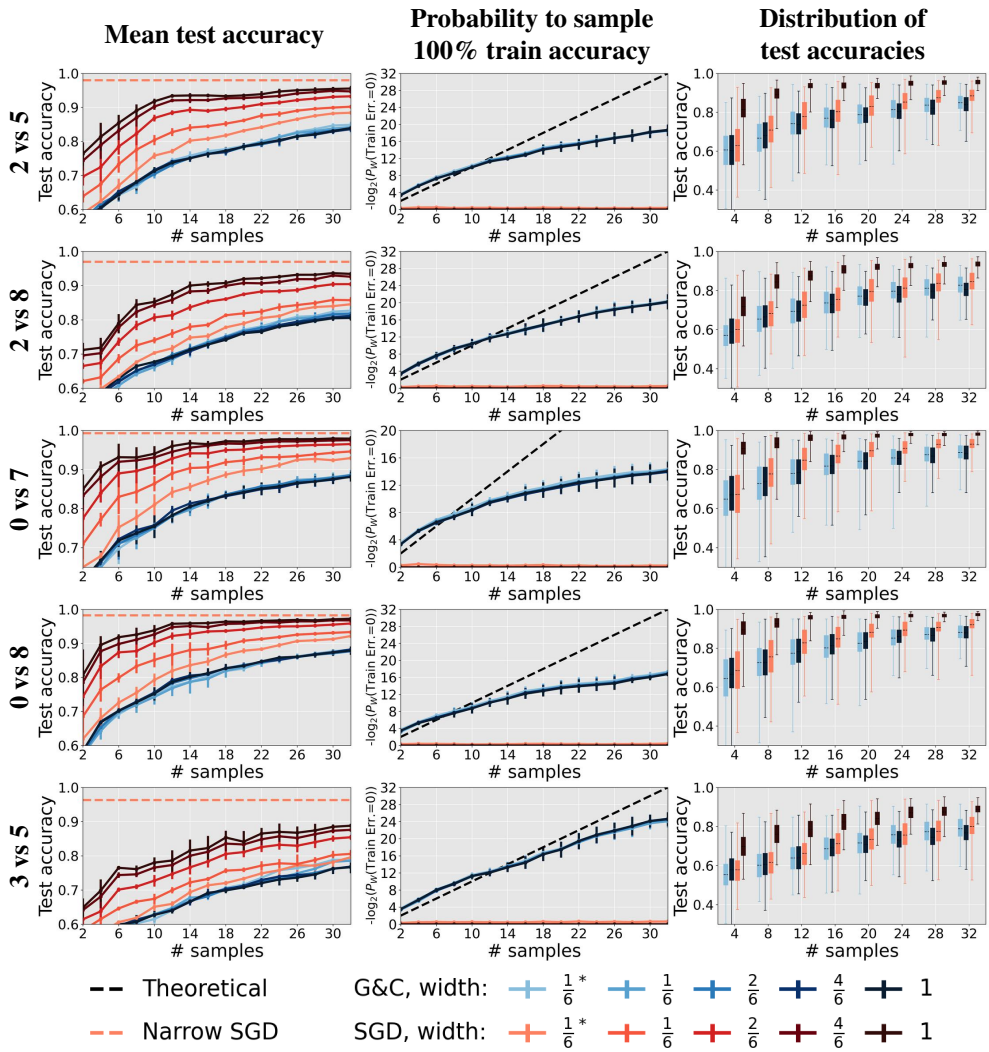


Figure 19. **Increasing width is a positive optimization bias.** The rows represent different pairs of classes from the MNIST dataset using the LeNet architecture. **Column 1:** Mean test accuracy vs number of training samples across network widths. Wider networks result in improvement for SGD, whereas G&C shows no change. Thus, for increasing width, there is a bias of SGD towards better generalizing networks independent of an architectural bias. However, overfitting does not occur for G&C, implying that overparameterization does not hurt generalization. **Column 2:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Note that this does not change with the width, which indicates that the pool of "fitting networks" does not change when the width increases. **Column 3:** We report the combined distribution of the four subsets for the extreme width factors. While for G&C, the entire distribution of models aligns, for SGD, the wider networks are considerably better than the narrow networks (and the G&C networks). For more details, please refer to Section 4.2.

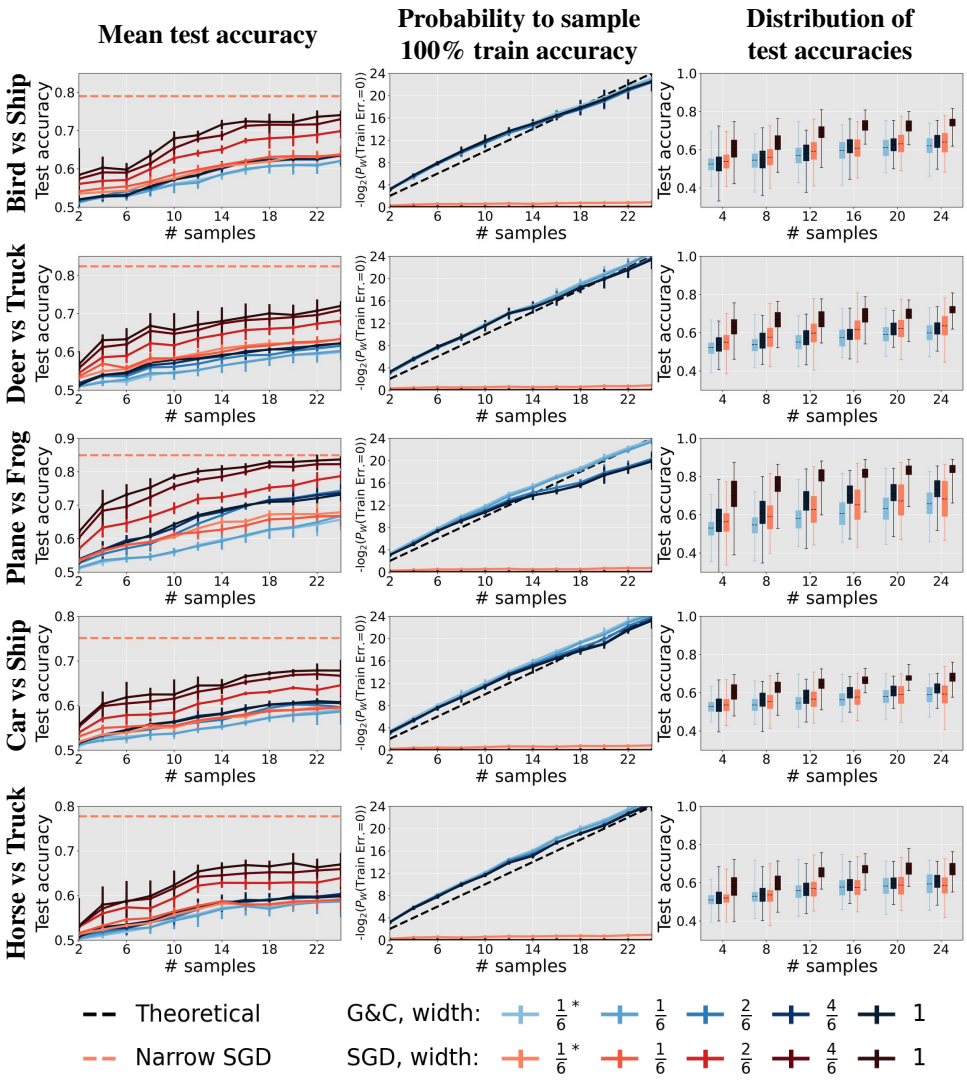


Figure 20. **Increasing width is a positive optimization bias.** The rows represent different pairs of classes from the CIFAR10 dataset using the LeNet architecture. **Column 1:** Mean test accuracy vs number of training samples across network widths. Wider networks result in improvement for SGD, whereas G&C improves for width up to 2/6 and then stagnates. Thus, for increasing width, SGD has a bias towards better generalizing networks independent of an architectural bias. However, overfitting does not occur for G&C, implying that overparameterization does not hurt generalization. **Column 2:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Note that this is constant or improves slightly with the width, which indicates that the pool of "fitting networks" does not change when the width increases. **Column 3:** We report the combined distribution of the four subsets for the extreme width factors. While for G&C, the entire distribution of models aligns, for SGD, the wider networks are considerably better than the narrow networks (and the G&C networks). For more details, please refer to Section 4.2.

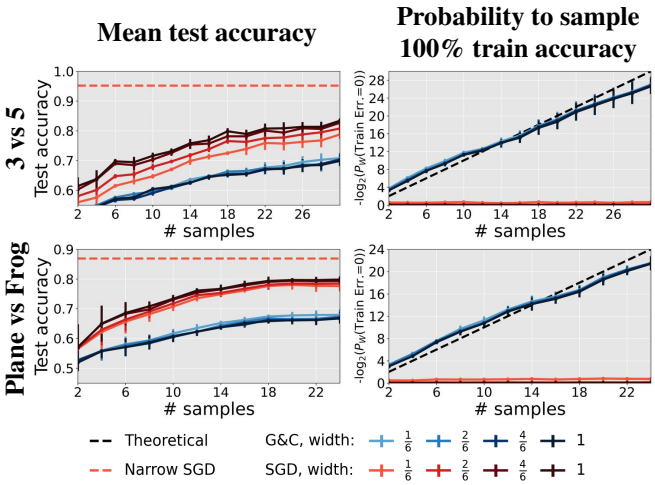


Figure 21. **Increasing width is a positive optimization bias.** The rows represent classes 3 vs 5 from MNIST and Plane vs Frog from CIFAR10 using the MLP architecture. **Left:** Mean test accuracy vs number of training samples across network widths. Wider networks result in improvement for SGD, whereas G&C stagnates. Thus, for increasing width, SGD has a bias towards better generalizing networks independent of an architectural bias. However, overfitting does not occur for G&C, implying that overparameterization does not hurt generalization. **Right:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Note that this does not change with the width, which indicates that the pool of "fitting networks" does not change when the width increases. For more details, please refer to Section 4.2.

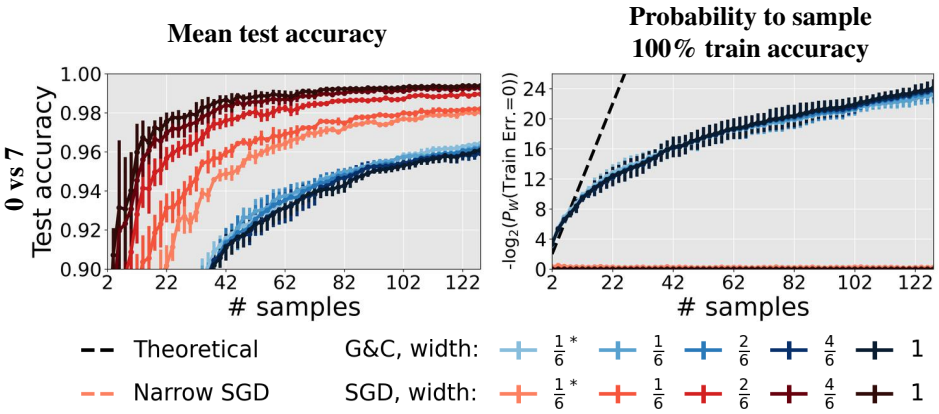


Figure 22. **Trend of SGD vs G&C when increasing width remains consistent with larger training set. Left:** Mean test accuracy vs number of training samples across network widths. Wider networks result in improvement for SGD, whereas G&C stagnates. Thus, for increasing width, SGD has a bias towards better generalizing networks independent of an architectural bias. However, overfitting does not occur for G&C, implying that overparameterization does not hurt generalization. **Right:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). This number remains the same for different widths, indicating that the pool of "fitting networks" does not change with increasing width. The fact that we see the same trend for 128 samples as we saw for 32 samples (MNIST) and 24 samples (CIFAR10) suggests that the results stated in the previous parts are not confined only to the low sample regime, at least for simpler problems like 0 vs 7.

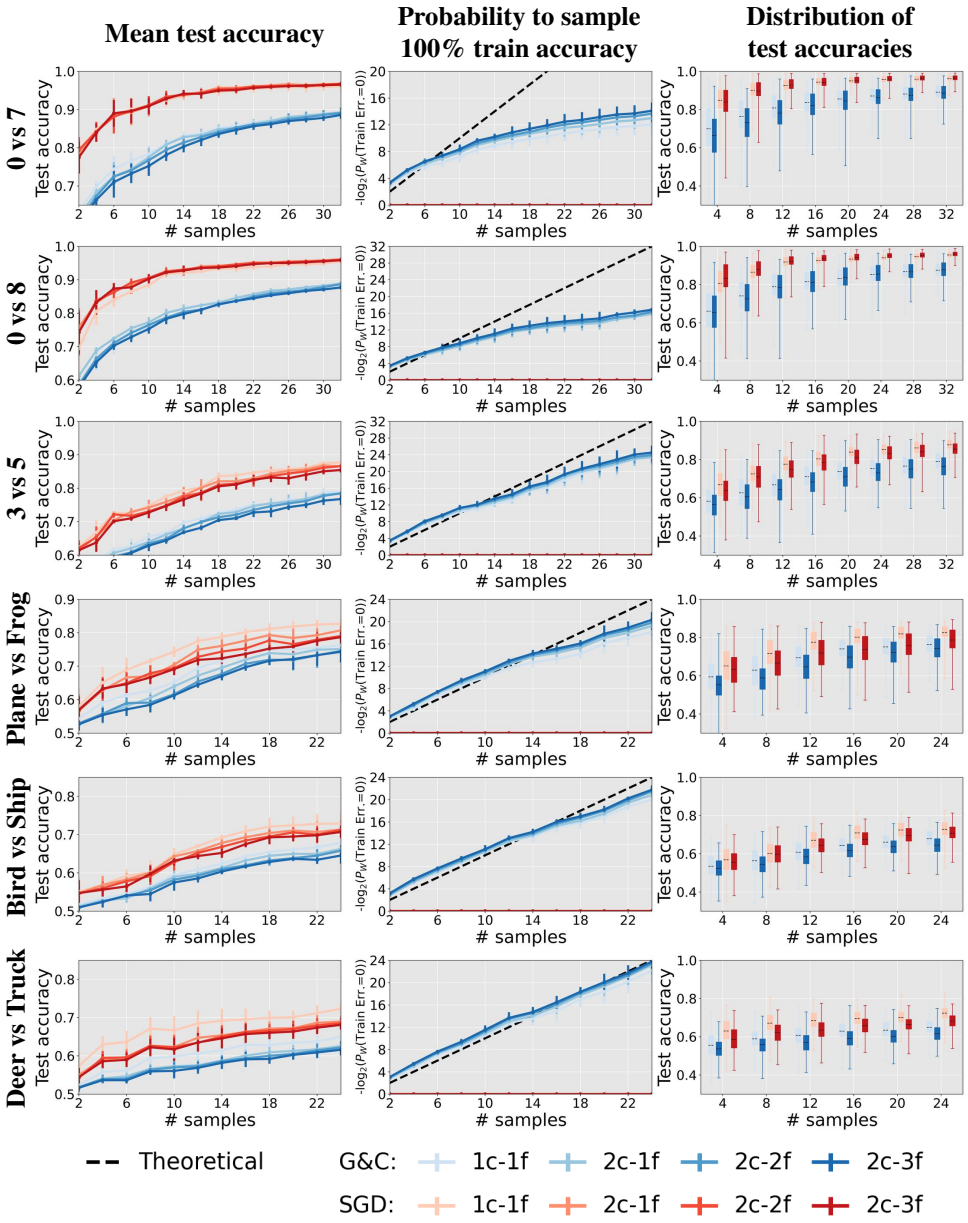


Figure 23. **Increasing depth is a negative architectural bias.** The rows correspond to different pairs of classes from the MNIST and CIFAR10 datasets using the LeNet architecture. Configuration "2c-1f" means two convolutional layers followed by a fully connected layer. **Column 1:** Mean test accuracy vs number of training samples across network depths. As the network becomes deeper, G&C always gets worse, whereas SGD stagnates for simple problems (top) and gets worse for harder problems (bottom). Thus, overparameterization in terms of depth results in overfitting instead of better generalization, unlike for the width. Since both G&C and SGD follow a similar trend, the decrease in performance with increased depth can be attributed to architectural bias. **Column 2:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Deeper networks have a lower probability for G&C to fit the training data, indicating that the network produces more complex functions. **Column 3:** We report the combined distribution of the four subsets for the extreme number of layers ("1c-1f", "2c-3f"). For more details, please refer to Section 4.3.

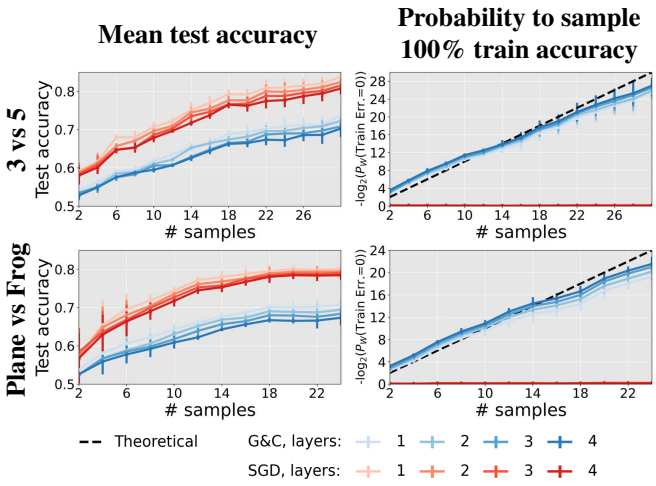


Figure 24. **Increasing depth is a negative architectural bias.** The rows represent classes 3 vs 5 from MNIST and Plane vs Frog from CIFAR10 using the MLP architecture. **Left:** Mean test accuracy vs number of training samples across network depths. As the network becomes deeper, G&C and SGD get worse. Thus, overparameterization in terms of depth results in overfitting instead of better generalization, unlike for the width. Since both G&C and SGD follow a similar trend, the decrease in performance with increased depth can be attributed to architectural bias. **Right:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Deeper networks have a lower probability for G&C to fit the training data, indicating that the network produces more complex functions. For more details, please refer to Section 4.3.

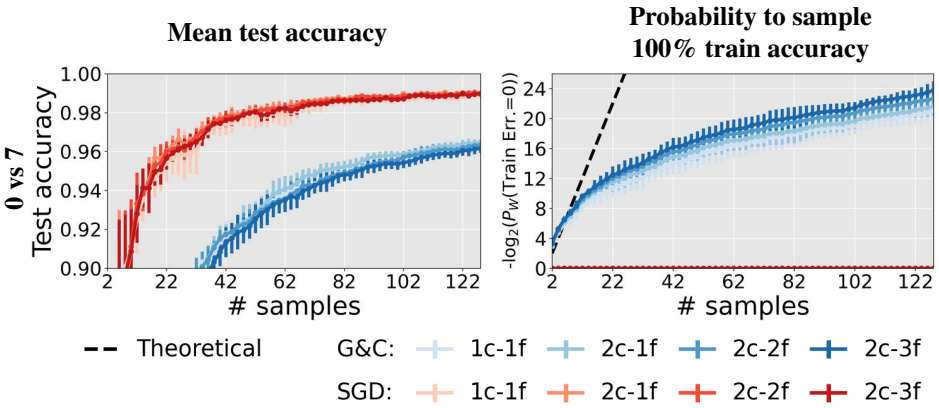


Figure 25. **Trend of SGD vs G&C when increasing depth remains consistent with larger training set.** Configuration "2c-1f" means two convolutional layers followed by a fully connected layer. **Left:** Mean test accuracy vs number of training samples across network depths. As the network becomes deeper, G&C gets worse, whereas SGD stagnates. Thus, overparameterization in terms of depth results in overfitting instead of better generalization, unlike for the width. **Right:** We report the negative log probability of G&C to find a network fitting the training data ($\mathrm{P}_W(\text{Train Error} = 0)$). Deeper networks have a lower probability for G&C to fit the training data, indicating that the network produces more complex functions. The fact that we see the same trend for 128 samples as we saw for 32 samples for classes 0 vs 7, suggests that the results stated in the previous parts are not confined only to the low sample regime, at least for simpler problems like 0 vs 7.

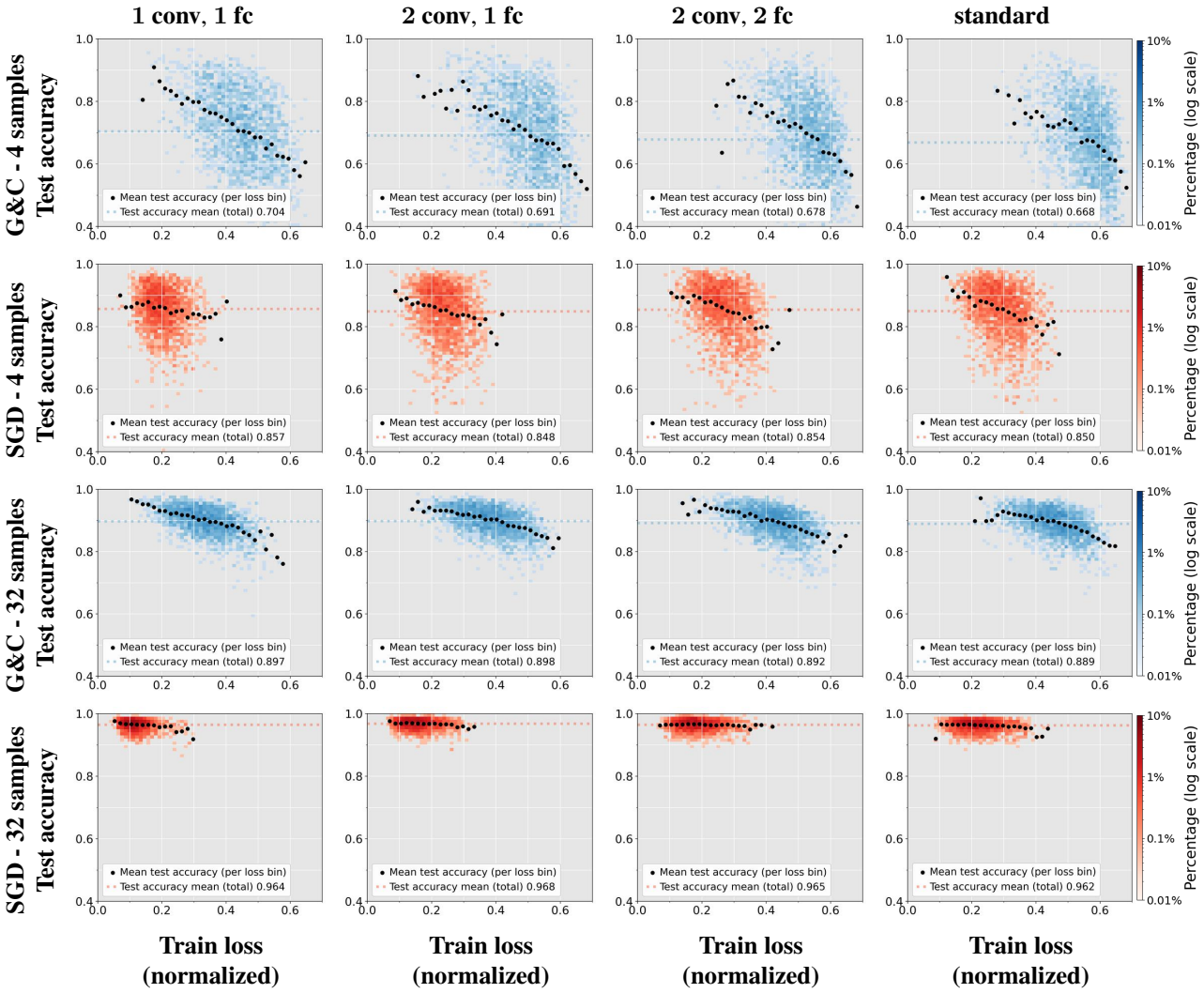


Figure 26. **Qualitative analysis of overparametrization in the depth.** In contrast to increasing width, increasing the depth decreases the geometric margin (higher Lipschitz normalized loss). This decrease is true both for **G&C** (Rows 1 and 3) and **SGD** (Rows 2 and 4). We show 500 networks for each depth for MNIST with classes *0* and *7* using a training set of size 4 and 32.

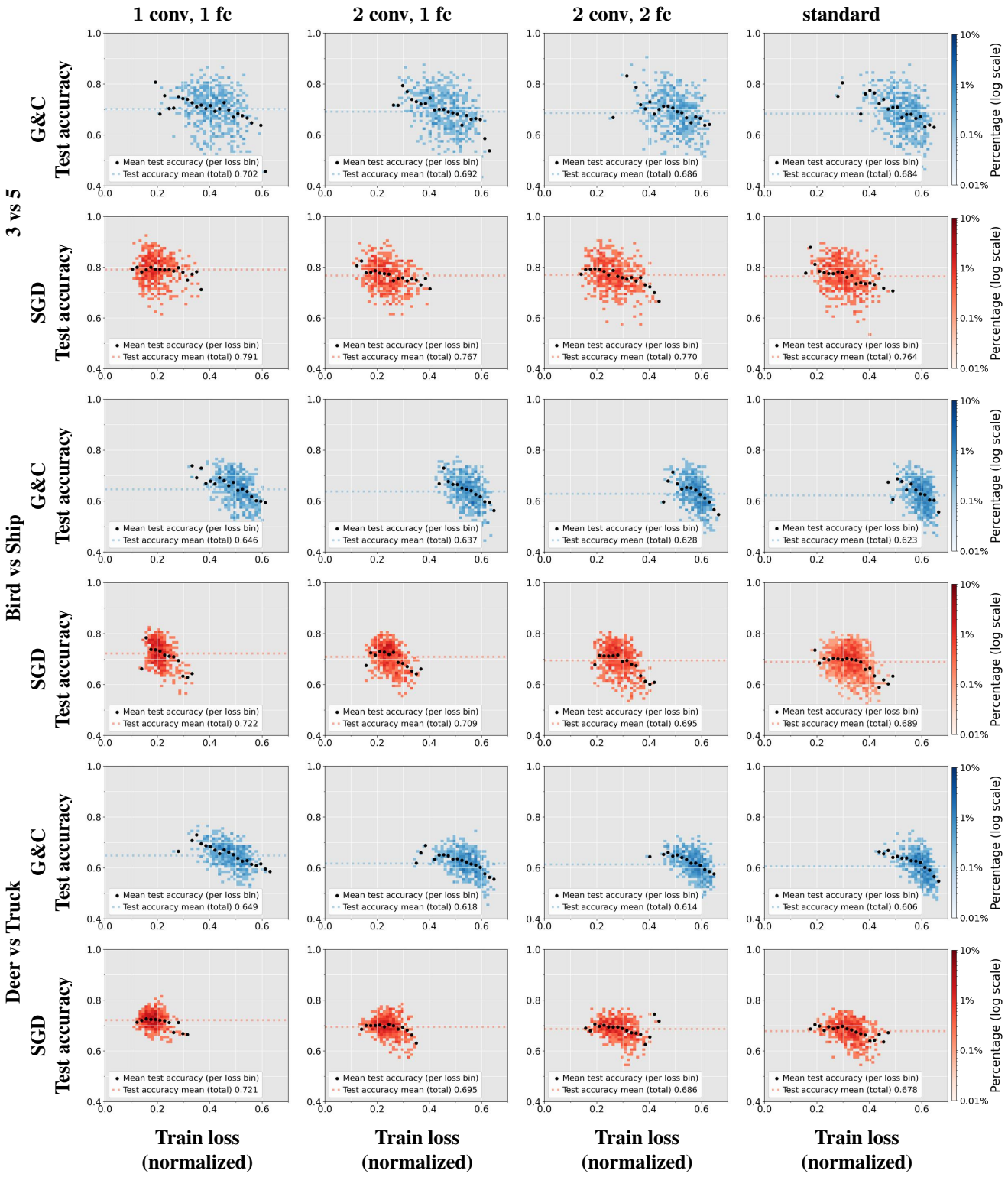


Figure 27. **Qualitative analysis of overparametrization in the depth.** In contrast to increasing width, increasing the depth decreases the geometric margin (higher Lipschitz normalized loss) and decreases the test accuracy. This decrease is true both for **G&C** (top) and **SGD** (bottom). We show 500 networks for each depth for MNIST with classes *3* and *5*, and CIFAR10 with classes Bird vs Ship and Deer vs Truck, using a training set of size 16.