# Understanding Nonlinear Implicit Bias via Region Counts in Input Space

**Jingwei Li** LJW22@MAILS.TSINGHUA.EDU.CN
*Institute for Interdisciplinary Information Sciences, Tsinghua University*

**Jing Xu** XUJING21@MAILS.TSINGHUA.EDU.CN
*Institute for Interdisciplinary Information Sciences, Tsinghua University*

**Zifan Wang** WZF22@MAILS.TSINGHUA.EDU.CN
*Institute for Interdisciplinary Information Sciences, Tsinghua University*

**Huishuai Zhang** ZHANGHUISHUAI@PKU.EDU.CN
*Wangxuan Institute of Computer Technology, Peking University*

**Jingzhao Zhang** JINGZHAOZ@MAIL.TSINGHUA.EDU.CN
*Institute for Interdisciplinary Information Sciences, Tsinghua University*

## Abstract

One explanation for the strong generalization ability of neural networks is implicit bias. Yet, the definition and understanding of implicit bias in non-linear contexts remains mysterious. In this work, we propose to characterize implicit bias by the count of connected regions in the input space with the same predicted label. Compared with parameter-dependent metrics (e.g., norm or normalized margin), region count can be better adapted to nonlinear, overparameterized models, because it is determined by the function mapping and is invariant to reparametrization. Empirically, we found that small region counts align with geometrically simple decision boundaries and correlate well with good generalization performance. We also observe that good hyper-parameter choices such as larger learning rates and smaller batch sizes can induce small region counts. We further establish the theoretical connections between region count and the generalization bound, and explain how larger learning rate can induce small region counts in neural networks.

## 1. Introduction

One mystery in deep neural networks lies in their ability to generalize, despite having significantly more learnable parameters than the number of training examples [50]. The choice of network architectures, including factors such as nonlinearity, depth, and width, along with training procedures like initialization, optimization algorithms, and loss functions, can result in vastly diverse generalization performances [27, 38, 40, 46]. The varied generalization abilities exhibited by neural networks are often explained by many researchers through the theory of *implicit bias* (or *implicit regularization*) [8, 39]. Implicit bias refers to inherent tendencies of how the network learns and generalizes from the training data, even without explicit regularizations or constraints.

The implicit bias of linear models and linear neural networks has been extensively studied. One of the classical setting is linear classification with logistic loss. [2, 8, 39] show that the parameter converges to the direction that maximizes the $L_2$ margin. For regression problems, it is proved that gradient descent or stochastic gradient descent converges to a parameter that is closest to the

© J. Li, J. Xu, Z. Wang, H. Zhang & J. Zhang.

initialization in terms of $L_2$ norm [13]. The results from the linear regime can be extended to linear neural networks by generalizing the definition of min-norm and max-margin solutions [20, 42, 47].

Compared to linear models, defining implicit biases in non-linear networks presents notable challenges. One line of papers studies homogeneous networks and shows that gradient flow solutions converge to a KKT point of the max-margin problem [19, 23, 28, 44]. Further studies extend the analysis and show that gradient flow converges to a max-margin solution in various norms [9, 32].

We note that previous definitions of implicit bias in neural networks mostly focus on certain metrics of network parameters. Such approaches enable explicit analyses of training trajectories, but face new challenges when applied to nonlinear networks: reparametrization of the network may preserve the function mapping but gives completely different parameters, and consequently, different implicit biases. We will discuss this point in detail in Section B.

Motivated by the above studies, we instead focus on leveraging the decision boundaries in the input space to characterize implicit bias. Our research identifies a metric called region count, which is defined by the average number of regions in the predictor's decision boundary (See Figure 1). We find that this region count has a strong correlation with the generalization gap across various setups: models with fewer region counts tend to generalize better. Furthermore, we show that neural networks trained with large learning rate or small batch size, which are typically deemed as beneficial for generalization, are biased towards solutions that have small region counts. Therefore, region count empirically serves as an accurate generalization metric as well as an implicit bias indicator.

The main contributions of this paper are listed as follows:

1. We introduce a novel measure of implicit bias via the region count in the input space. Through extensive experiments, we verify a strong correlation between region count and the generalization gap. This correlation remains robust across different learning methods, datasets, training parameters, and counting methods.

2. We assess the factors that induce small region count, discovering that training with larger learning rates and smaller batch sizes typically results in fewer regions. This provides a possible explanation for the implicit bias in neural networks.

3. We conduct theoretical analyses on region counts, demonstrating that small region counts lead to good generalization. We further show that large learning rates result in small region counts.

## 2. Preliminary

Although region count is a natural measure for the complexity of a predictor, and that it depends only on the decision function rather than the model parameterization, its formal definition and computability remains unclear. In this section, we first provide the definition and low-dimensional approximation of region counts. We then empirically verify that region counts correlate with generalization gap.

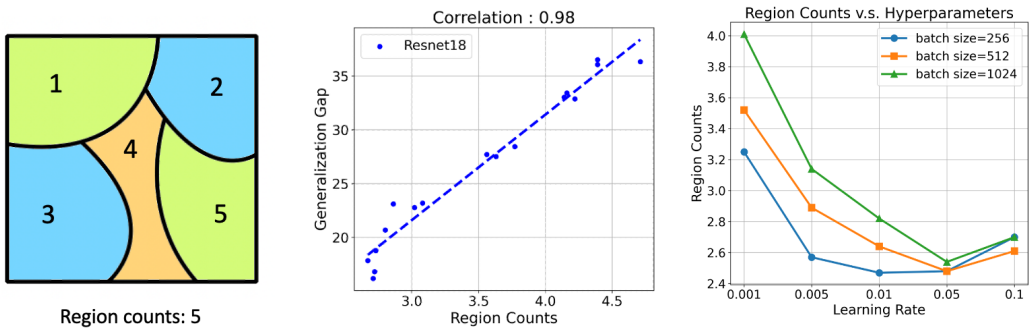


Figure 1: **A schematic illustration of main results in this paper.** Left: The region counts in input space. Each distinct region represents an area where the neural network makes the same prediction for all points within that region. Middle: A strong correlation between region counts and the generalization gap. Right: Larger learning rate or smaller batch size induces smaller region counts.

### 2.1. Definition of Region Counts

Let $d$ denote the training data dimension and $f : \mathbb{R}^d \to \{1, 2, \ldots, N\}$ denote a neural network for a classification task with $N$ classes. For a subset $U \subset \mathbb{R}^d$, we can define the connectedness of its element as follows:

**Definition 1 (Connectedness)** *We say the data points $x_1, x_2 \in U$ are (path) connected with respect to a neural network $f$ if they satisfy:*

- $f(x_1) = f(x_2) = c$,
- *There exist a continuous mapping $\gamma : [0, 1] \to U$, $\gamma(0) = x_1$, $\gamma(1) = x_2$, and for any $t \in (0, 1)$, $f(\gamma(t)) = c$.*

Then we define the connected region in this subset:

**Definition 2 (Maximally Connected Region)** *We say $V \subset U$ is a maximally connected region in $U \in \mathbb{R}^d$ if it satisfies the following property:*

- *For any $x, y \in V$, they are connected.*
- *For any $x \in V$, $y \in U \setminus V$, they are not connected.*

Finally, we formally define the region count as follows:

**Definition 3 (Region Count)** *For a subset $U \subseteq \mathbb{R}^d$, We define its region count $R_U$ as the number of maximally connected regions in $U$:*

$$R_U = card\{V \subset U | V \text{ is a maximally connected region}\},$$

*where card is the cardinality of a set.*

## 3. Region Counts Correlate with Generalization Gaps

In this section, we present our major empirical findings, which reveals a strong correlation between region counts and the generalization error of neural networks.

We conduct image classification experiments on the CIFAR-10 dataset, using different architectures, including ResNet18 [16], EfficientNetB0 [41], and SeNet18 [17]. Results on other datasets and architectures are deferred to ablation studies. We vary the hyperparameters for training, such as learning rate, batch size and weight decay coefficient, whose numbers are reported in Table 1. The region count is calculated using randomly generated 1D hyperplanes, as described in Example 2. We ran each experiment 100 times and report the average region count.

We plot the region count and generalization gap of different setups in Figure 2, and calculate the correlation between them. For each network architecture, we observe a strong correlation as high as 0.98. The overall correlation for all the three networks still reaches 0.93. This reveals a remarkably high correlation between region counts and generalization gap.

We also explore whether such a strong correlation exists for traditional machine learning algorithms. We conduct experiments with decision trees and random forests on the same classification tasks, with hyperparameters specified in Table 1. We observe a similar linear trend between region count and generalization gap, with a correlation of 0.96 in decision trees and 0.98 in random forests. The overall correlation coefficient was 0.90. Therefore, region count serves as a good indicator for generalization performance across various setups.

Table 1: **The hyperparameters for experiments.** Left: We vary the learning rate, batch size, and weight decay for training a neural network, to modulate the model's generalization ability. Right: We adjust the training parameters for traditional machine learning models such as decision tree and random forest.

| Hyperparameters | Value |
|---|---|
| Learning rate | 0.1, 0.01, 0.001 |
| Batch size | 256, 512, 1024 |
| Weight decay | 1e−5, 1e−6, 1e−7 |

| Hyperparameters | Value |
|---|---|
| Depth | $3, 4 \cdots, 17$ |
| Criterions | gini, entropy |
| Splitter | best, random |

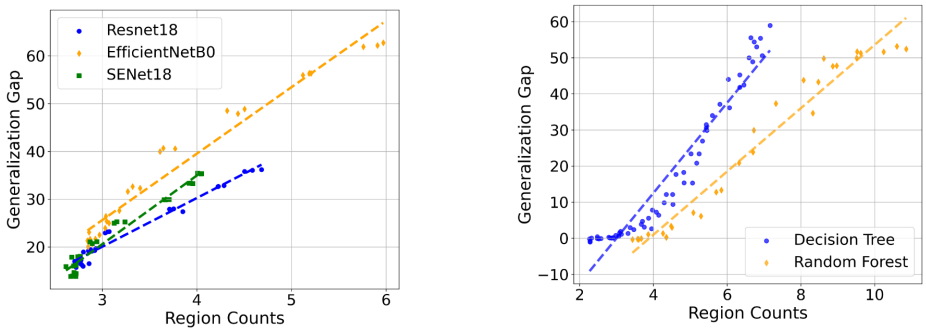


Figure 2: **Strong correlation between region counts and generalization gap.** Left: We conduct experiments using three neural networks on the CIFAR-10 dataset, with various hyperparameters. Right: We conduct experiments using Decision Tree and Random Forest on the CIFAR-10 dataset.

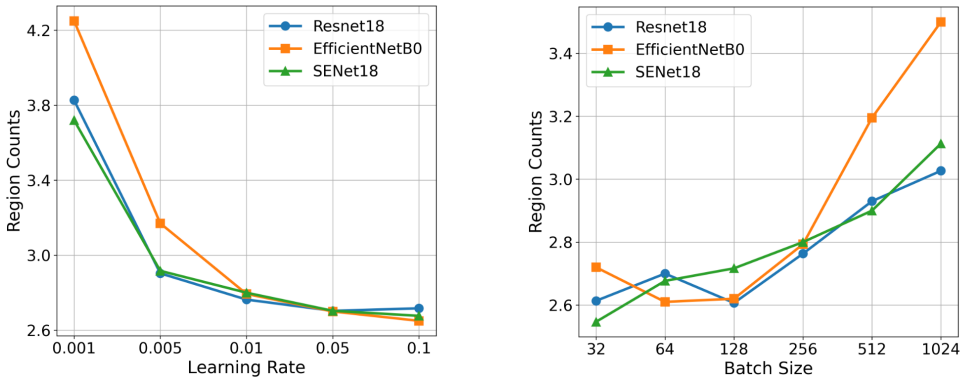


Figure 3: **Large learning rate and small batch size reduce region counts.** We train three networks on the CIFAR-10 dataset, varying the batch sizes and learning rates.

## 4. Region Counts Quantify Implicit Bias

In this section, we further investigate the implicit bias of neural networks via region counts. We show both empirically and theoretically that neural networks trained with appropriate hyperparameters tend to have smaller region counts, thus achieving better generalization performance.

Training neural networks requires careful selection of many hyperparameters, such as learning rates, batch sizes, optimizers, epochs and so on. Here, we primarily focus on learning rate and batch size, and study their impact on the region count.

**Learning Rates.** We provide the relationship between the learning rate and the number of regions in Figure 3. Our findings indicate that a larger learning rate tends to simplify the decision boundary and results in a smaller region count in the hyperplane. This accords well with real practices, where large learning rates of $0.1$ or $0.01$ are often favored for better generalization.

**Batch Sizes.** Similarly, the training batch size can impact the number of regions. As shown in Figure 3, smaller batch sizes lead to a model with fewer regions. This result reveals the advantage of small-batch training, which leads to better generalization accuracy.

Previous researches find that certain hyperparameters, such as a large learning rate and a small batch size, can improve the generalization of the neural network. Our observations provide a possible explanation: These good hyperparameter choices lead to a reduced region count. Such simplicity bias can decrease the generalization gap of neural networks.

We also provide some theortical analysis in Appendix G.

## 5. Conclusions and Future Directions

We study the region counts in the input space and identify its strong correlation with generalization gap in non-linear neural networks. Our analysis offers a new perspective to quantify and understand the generalization property and implicit bias of neural networks.

Our paper suggests several promising directions for future research. Firstly, our analyses of why large learning rate induces small region counts mainly focus on a simplified setup. The analyses for more general settings remain open. Secondly, extending the definition of region count to non-classification tasks, such as natural language generation, would be a worthwhile direction.

## References


[1] Kwangjun Ahn, Sébastien Bubeck, Sinho Chewi, Yin Tat Lee, Felipe Suarez, and Yi Zhang. Learning threshold neurons via edge of stability. *Advances in Neural Information Processing Systems*, 36, 2024.

[2] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[3] Sanjeev Arora, Zhiyuan Li, and Abhishek Panigrahi. Understanding gradient descent on the edge of stability in deep learning. In *International Conference on Machine Learning*, pages 948–1024. PMLR, 2022.

[4] David Bertoin, Jérôme Bolte, Sébastien Gerchinovitz, and Edouard Pauwels. Numerical influence of relu'(0) on backpropagation. *Advances in Neural Information Processing Systems*, 34:468–479, 2021.

[5] Pascal Bianchi, Walid Hachem, and Sholom Schechtman. Convergence of constant step stochastic gradient descent for non-smooth non-convex functions. *Set-Valued and Variational Analysis*, 30(3):1117–1147, 2022.

[6] Jérôme Bolte and Edouard Pauwels. A mathematical model for automatic differentiation in machine learning. *Advances in Neural Information Processing Systems*, 33:10809–10819, 2020.

[7] Jérôme Bolte and Edouard Pauwels. Conservative set valued fields, automatic differentiation, stochastic gradient methods and deep learning. *Mathematical Programming*, 188:19–51, 2021.

[8] Alon Brutzkus, Amir Globerson, Eran Malach, and Shai Shalev-Shwartz. Sgd learns over-parameterized networks that provably generalize on linearly separable data. *arXiv preprint arXiv:1710.10174*, 2017.

[9] Lenaic Chizat and Francis Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020.

[10] Jeremy Cohen, Simran Kaur, Yuanzhi Li, J Zico Kolter, and Ameet Talwalkar. Gradient descent on neural networks typically occurs at the edge of stability. In *International Conference on Learning Representations*, 2020.

[11] Alex Damian, Eshaan Nichani, and Jason D Lee. Self-stabilization: The implicit bias of gradient descent at the edge of stability. *arXiv preprint arXiv:2209.15594*, 2022.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

[14] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. Implicit bias of gradient descent on linear convolutional networks. *Advances in neural information processing systems*, 31, 2018.

[15] Boris Hanin and David Rolnick. Complexity of linear regions in deep networks. In *International Conference on Machine Learning*, pages 2596–2604. PMLR, 2019.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

[18] Arthur Jacot. Implicit bias of large depth networks: a notion of rank for nonlinear functions. *arXiv preprint arXiv:2209.15055*, 2022.

[19] Arthur Jacot, Eugene Golikov, Clément Hongler, and Franck Gabriel. Feature learning in $l_2$-regularized dnns: Attraction/repulsion and sparsity. *Advances in Neural Information Processing Systems*, 35:6763–6774, 2022.

[20] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

[21] Ziwei Ji and Matus Telgarsky. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

[22] Ziwei Ji and Matus Telgarsky. The implicit bias of gradient descent on nonseparable data. In *Conference on Learning Theory*, pages 1772–1798. PMLR, 2019.

[23] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

[24] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. Fantastic generalization measures and where to find them. *arXiv preprint arXiv:1912.02178*, 2019.

[25] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *Computer Aided Verification: 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I 30*, pages 97–117. Springer, 2017.

[26] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[27] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[28] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *arXiv preprint arXiv:1906.05890*, 2019.

[29] Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Advances in Neural Information Processing Systems*, 34:12978–12991, 2021.

[30] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European conference on computer vision (ECCV)*, pages 116–131, 2018.

[31] Mor Shpigel Nacson, Nathan Srebro, and Daniel Soudry. Stochastic gradient descent on separable data: Exact convergence with a fixed learning rate. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 3051–3059. PMLR, 2019.

[32] Greg Ongie, Rebecca Willett, Daniel Soudry, and Nathan Srebro. A function space view of bounded norm infinite width relu nets: The multivariate case. *arXiv preprint arXiv:1910.01635*, 2019.

[33] Scott Pesme, Loucas Pillaud-Vivien, and Nicolas Flammarion. Implicit bias of sgd for diagonal linear networks: a provable benefit of stochasticity. *Advances in Neural Information Processing Systems*, 34:29218–29230, 2021.

[34] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020.

[35] Itay Safran, Gal Vardi, and Jason D Lee. On the effective number of linear regions in shallow univariate relu networks: Convergence guarantees and implicit bias. *Advances in Neural Information Processing Systems*, 35:32667–32679, 2022.

[36] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018.

[37] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[38] Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[39] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[40] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[41] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019.

[42] Tomas Vaskevicius, Varun Kanade, and Patrick Rebeschini. Implicit regularization for optimal sparse recovery. *Advances in Neural Information Processing Systems*, 32, 2019.

[43] Roman Vershynin. High-dimensional probability. *University of California, Irvine*, 2020.

[44] Bohan Wang, Qi Meng, Wei Chen, and Tie-Yan Liu. The implicit bias for adaptive optimization algorithms on homogeneous neural networks. In *International Conference on Machine Learning*, pages 10849–10858. PMLR, 2021.

[45] Lily Weng, Huan Zhang, Hongge Chen, Zhao Song, Cho-Jui Hsieh, Luca Daniel, Duane Boning, and Inderjit Dhillon. Towards fast computation of certified robustness for relu networks. In *International Conference on Machine Learning*, pages 5276–5285. PMLR, 2018.

[46] Ashia C Wilson, Rebecca Roelofs, Mitchell Stern, Nati Srebro, and Benjamin Recht. The marginal value of adaptive gradient methods in machine learning. *Advances in neural information processing systems*, 30, 2017.

[47] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

[48] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[49] Chulhee Yun, Shankar Krishnan, and Hossein Mobahi. A unifying view on implicit bias in training linear neural networks. *arXiv preprint arXiv:2010.02501*, 2020.

[50] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization, 2017.

[51] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

## Appendix A. Related Works

**Implicit Bias of Linear Neural Network** The implicit bias in linear neural networks are thoroughly investigated in recent works. For linear logistic regression on linearly separable data, full-batch gradient descent converges in the direction of the maximum margin solution [39]. This foundational work has various follow-ups, including extensions to non-linearly-separable data [21, 22], stochastic gradient descent [31], and other loss functions and optimizers [13].

These findings in linear logistic regression are generalized to deep linear networks. For fully-connected neural networks with linear separable data, [20] show that the direction of weight also converges to $L_2$ max-margin solution. For linear diagonal networks, the gradient flow maximizes the margin with respect to a specific quasi-norm that is related to the depth of network [14, 33, 47], leading to a bias towards sparse linear predictors as the depth goes to infinity. This sparsity bias also exists in linear convolutional networks [14, 49].

**Implicit Bias of Non-linear Neural Network** The non-linearity of modern non-linear neural network poses challenges to studying its implicit bias. Initial works in this area [23, 28] focus on homogeneous networks. These studies show that with exponentially-tailed classification losses, both gradient flow and gradient descent converge directionally to a KKT point in a maximum-margin problem. Further researches, for instance in [44], consider a more general setup that includes different optimizers and prove that both Adam and RMSProp are capable of maximizing the margin in neural networks while Adagrad is not. [9, 32] showcased a bias towards maximizing the margin in a variation norm for infinite-width two-layer homogeneous networks. [19, 29] identified margin maximization in two-layer Leaky-ReLU networks trained with linearly separable and symmetric data. More recent investigations into non-linear neural networks, such as [18], focus on the homogeneity of the non-linear layer, demonstrating an implicit bias characterized by a novel non-linear rank.

Our research is related to [35], which prove that for a two-layer ReLU network with width $r$, gradient flow will directionally converge to a network characterized by no more than $O(r)$ linear regions. The number of linear regions expressed by ReLU networks is also studied in [15]. Compared with their main focus to develop the upper bound for the number of regions, our work delves into the correlation between the region counts and generalization gap. Furthermore, our metric counts number of different decision regions rather than linear regions, and can be efficiently estimated.

## Appendix B. Motivation

Norm-based and margin-based characterizations belong to the most popular measures of implicit bias. Various definitions for norm and margin exist. For simplicity, we consider the following two definitions.

**Example 1** *Let $W = \{W_1, \cdots W_l\}$ denotes the post-training weight parameters of an $l$-layer neural network $f_W(x) = W_l\sigma(W_{l-1}\cdots W_2\sigma(W_1 x))$, with $\sigma(\cdot)$ as the ReLU activation function. Denote the weight initialization as $W^0 = \{W_1^0, \cdots W_l^0\}$. Consider the Frobenious norm between network weights and initialization:*

$$d(W) = \sqrt{\sum_{i=1}^{l} \|W_i - W_i^0\|_F^2},$$

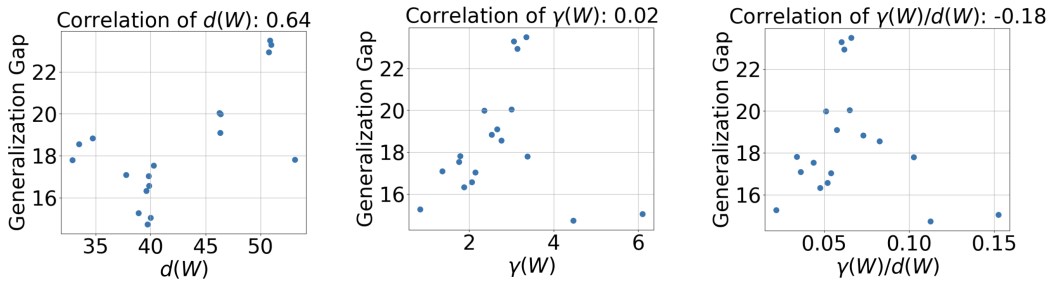


Figure 4: **Norm-based and margin-based measures may not be predictive of generalization gaps.** We train ResNet18 on the CIFAR-10 dataset using various hyperparameters. These implicit bias measures can be ineffective for general non-linear neural networks.

*and the output-space margin*

$$\gamma(W) = \mathbb{E}_{(x,y)}\left[f(x)_y - \max_{i \neq y} f(x)_i\right].$$

$d(\cdot)$ and $\gamma(\cdot)$ are commonly used indicator for implicit bias of linear models [21, 39]. However, both of them are not invariant to network reparameterization. We can construct a different set of network weight parameters by scaling the parameters as $\hat{W} = \{2W_1, \frac{1}{2}W_2, \cdots W_l\}$, such that $f_W = f_{\hat{W}}$ but $d(W) \neq d(\hat{W})$ in general. Similarly, we can scale the last layer weights and get $\tilde{W} = \{W_1, \cdots 2W_l\}$, such that $\gamma(\tilde{W}) \neq \gamma(W)$, but $\text{argmax}_i f_W(x) = \text{argmax}_i f_{\tilde{W}}(x)$. This reparameterization trick also works for more complicated norm-based and margin-based generalization metric in [24].

We numerically investigate whether they are effective measures, by training a ResNet18 on Cifar10 dataset, using different hyperparameters as in Table 1. The results are presented in Figure 4, which indicates that these measures have a low correlation with the generalization gap in the deep learning regime. One may choose other definitions of norm and get stronger correlations, yet such a choice often needs to be problem-specific and requires domain expertise.

The definition of margin may also be improved to the input-space margin, which is able to characterize the quality and robustness of the classifier. This metric is invariant to reparameterization and therefore more intrinsic to the underlying classifier. However, due to the highly nonconvex loss landscape, the input-space margin is NP-complete to compute and even hard to approximate [25, 45]. Therefore, quantitatively analyzing the decision boundary of a neural network and characterizing its implicit bias remains a challenge.

Our motivation can be summarized by a simple idea: although the margin in the input space is hard to compute, we can quantify the regions split by the decision boundary. This measure is invariant to model reparametrization and can also capture the complexity of the decision boundary. This motivates us to consider the region counts as an implicit bias metric.

## Appendix C. Experiment details

In this section, we provide the detailed experiment settings.

### C.1. Details on Architectures and Datasets

We conduct experiments on different neural network architectures, including ResNet18 and ResNet34 [16], EfficientNetB0 [41], SENet18 [17], VGG19 [37], MobileNetV2 [36], ShuffleNetV2 [30], RegNet200MF [34], SimpleDLA [48]. We conduct all experiments using NVIDIA RTX 6000 graphics card.

We use CIFAR-10/100 [26] and Imagenet-1k [12] as datasets. For CIFAR-10 and CIFAR-100 dataset, each network was trained for 200 epochs using the Stochastic Gradient Descent (SGD) algorithm with cosine learning rate schedule. We choose 27 combinations of hyperparameters in Table 1, and for each hyperparameter we use 3 random seeds and report the average metrics. For the Imagenet-1k dataset, each network was trained for 50 epochs with random data crop and random flip. We use the same optimizer and 27 combinations of hyperparameters as in CIFAR-10 and CIFAR-100 experiments. It is worth noting that we make minor adjustments on hyperparameters for certain networks to ensure stable training. For example, in the case of VGG19, the training is unable to converge when the learning rate is set to 0.1; therefore, we adjust it to 0.05.

### C.2. Correlation Plots

We show the correlation plot of average regions and test accuracy in Figure 5. The figure consists of results from training different networks on CIFAR-10 dataset with SGD for 200 epochs, using the hyperparameters specified in Table 1. The results show that different networks have different number of regions, ranging from 2 to 20. However, the correlation of test accuracy and average number of regions are consistently high in all the networks.

## Appendix D. How to Calculate the Number of Regions

In this section, we study different methods to calculate the region count, and discuss their impact on the results. Since it is impossible in practice to calculate the predictions of an infinite number of data points on the hyperplane, we select grid points from the hyperplane to calculate the region count.

### D.1. Estimating Region Counts

Calculating the region count in the original high-dimensional input space can be computationally intractable. Therefore, we propose a computational efficient surrogate, by calculating the region counts on low dimensional subspace spanned by training data points.

**Definition 4 (Region Count in $d$-Dimensional Subspace)** *We randomly sample $d+1$ datapoints in the training set to generate a convex region in $\mathbb{R}^d$ subspace. The $d$-dimensional region count $R_d$ is defined as the expectation of number of maximally connected regions:*

$$R_d = E_{x_1, x_2, \ldots, x_{d+1} \in D_{train}} [N_{\mathit{Conv}\{x_1, x_2, \ldots, x_{d+1}\}}] \,,$$

*where $x_1, x_2, \ldots, x_{d+1}$ are sampled from the training dataset, and $\mathit{Conv}\{x_1, x_2, \ldots, x_{d+1}\}$ is the convex hull formed by these $d+1$ points.*

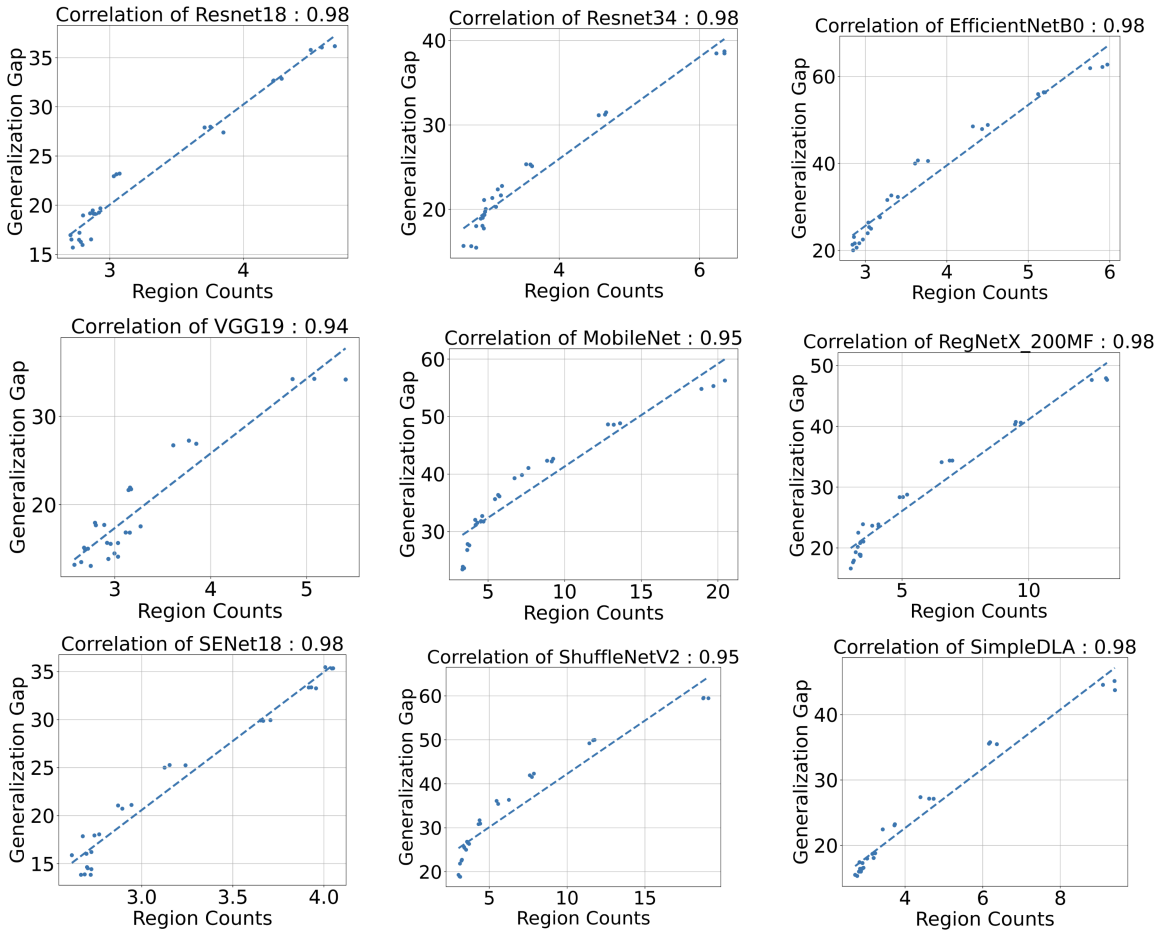


Figure 5: The correlation plot of all networks between average regions and test accuracy for CIFAR-10 dataset with optimizer SGD. From the graph we know that the correlations are all very high for different non-linear networks. Different structures of neural networks incur different scope of the average regions.

This paper primarily focuses on low dimension spaces, which is illustrated as below. In practice, we randomly sample training data points for multiple times and take the average region counts. In Section E, we show that the choice of subspace dimension $d$ does not significantly affect the results. The details on how to count the regions and generate the hyperplanes are provided in Appendix D.

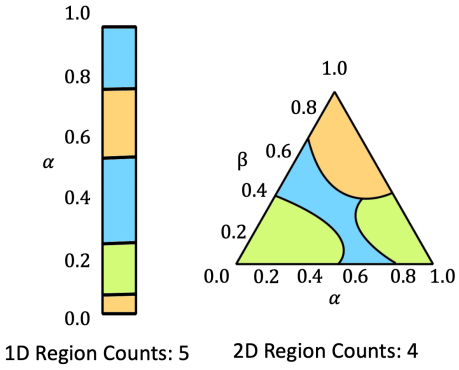


Figure 6: **Illustrations of region counts in 1D and 2D subspace.** We use different colors to represent different outputs of the neural network.

**Example 2 (Region counts in 1D and 2D subspace)** *For region count in 1-dimensional subspace, we randomly sample two data points, denoted as $x_1$ and $x_2$, from the training set, and calculate the re-*

*gion count on the line segment connecting them:*

$$\{\alpha x_1 + (1-\alpha)x_2\,, 0 \le \alpha \le 1\}.$$

*For the 2-dimensional case, we randomly sample three data points, $x_1$, $x_2$, and $x_3$, from the training set, and calculate the region count in the convex hull spanned by them:*

$$\{\alpha x_1 + \beta x_2 + (1-\alpha-\beta)x_3\,, \alpha \ge 0, \beta \ge 0, \alpha+\beta \le 1\}\,.$$

*We provide an illustration in Figure 6.*

Assume we have divided a region of the hyperplane into several equidistant small squares. We can use an algorithm similar to breadth-first search to calculate the number of connected components within these small squares, thereby determining the number of regions. Here, we use a 2-dimensional hyperplane as an example (the 1-dimensional case can be considered a degenerate version of this algorithm).

Therefore, it is necessary to determine the granularity of splits for the plane. We experimented with different setups of splitting parameters, and the results averaged by 100 independent trials are presented in Table 2 and Table 2. From the results, using 200 grid points in the 1D case and 30x30 grid points in the 2D case is an optimal choice. Splitting the plane into fewer points results in an inadequate approximation of regions, while increasing the number of points does not significantly enhance accuracy but incurs greater computational costs. Therefore, in our experiments, we split the plane into 200 grid points for the 1D case and 30x30 grid points for the 2D case.

Table 2: The mean value of region counts with different splitting numbers in 1d (left) and 2d (right) planes.

| Splitting Numbers | Region Counts |
|---|---|
| 50 | 2.74 |
| 100 | 2.76 |
| 200 | 2.78 |
| 300 | 2.78 |
| 500 | 2.78 |

| Splitting Numbers | Region Counts |
|---|---|
| 10×10 | 2.76 |
| 20×20 | 2.76 |
| 30×30 | 2.78 |
| 40×40 | 2.78 |
| 50×50 | 2.78 |

Subsequently, we study the number of random samples in calculating the average number of regions. We experiment with different numbers of hyperplanes, and the results are presented in Table 3. From the results, we know that using 100 samples to calculate the average provides a reliable answer with relatively low computational costs. Therefore, in the experiments we randomly generate 100 lines or planes and calculate the average number of regions.

In our paper we use the convex hull of two points $\{\alpha x_1 + (1-\alpha x_2)\}$ to calculate the region counts in 1D case with $\alpha \in [0,1]$. We also conduct ablation studies with varied coordinate ranges $\alpha$. We train ResNet18 on CIFAR10 using hyperparameter in Table 1 in our manuscript, where we vary the range of $\alpha$ and analyze the correlation. The results are shown in Table 4. These studies confirm that expanding the range does not influence the strong correlation between region counts and test accuracy.

Table 3: The mean value of region counts with different number of random samples.

| Number of Samples | Region Counts |
|---|---|
| 10 | 2.24 |
| 50 | 2.56 |
| 100 | 2.78 |
| 300 | 2.80 |
| 500 | 2.79 |

Table 4: The impact of interpolation range on region counts.

| The range of $\alpha$ | Region Counts | Correlation |
|---|---|---|
| $[0, 1]$ | 3.56 | 0.98 |
| $[-1, 2]$ | 4.47 | 0.96 |
| $[-2, 3]$ | 5.86 | 0.92 |
| $[-3, 4]$ | 6.39 | 0.93 |

## Appendix E. Ablation Studies

This section presents an ablation study to validate the robustness and consistency of our findings. We systematically vary key aspects of our experimental setup, including the network architecture, dataset, optimizer, and the method of computing the plane, and evaluate their impact on our main results of the correlation between region count and the generalization gap.

**More Architectures, Datasets and Hyperplane Dimensions.** We first examine the influence of neural network architectures and datasets on our results. We provide additional results on various neural network architectures such as ResNet34 [16], VGG19 [37], MobileNetV2 [36], ShuffleNetV2 [30], RegNet200MF [34], and SimpleDLA [48]. We also use various datasets such as CIFAR-100 [26] and ImageNet [12]. Region counts and generalization gaps are evaluated across various learning rates, batch sizes, and weight decay parameters as listed in Table 1.

We also explore the effects of different methods for generating the hyperplane in the input space. In our previous experiments, we generate the 1-dimensional plane using random pairs of samples from the training set and calculate the region count on them. Here we explore region counts in higher dimensional planes, that are spanned by 2 to 5 data randomly-selected points from the training set, using the CIFAR-10 dataset.

The experiment results are presented in Table 5. We also provide correlation plots for each network in Appendix C.2. We observe that the strong correlation between region count and the generalization gap remains consistent in various setups. The consistency indicates that our findings reveal a fundamental characteristic of non-linear neural networks.

We also provide evaluations by varying the optimizer, hyperplane generation algorithms and data generation methods. The results are deferred to Appendix F.

**Evolution of Region Counts during Training.** Next we study how the region count, generalization gap and their correlation evolve during training. Following the setup in Section **??**, we train a ResNet18 model on CIFAR-10 dataset, and report the region count and generalization gap during the training process. The statistics are averaged over different hyperparameter choices as in Table 1.

Table 5: **Experimental consistency across networks, datasets, and counting methods.** We conduct experiments on various types of networks across multiple datasets. We also alter the method of calculating the region counts. The results indicate that our findings are consistent across different setups.

| Network | Dataset | | | Counting Dimension | | | |
|---|---|---|---|---|---|---|---|
| | *CIFAR-10* | *CIFAR-100* | *ImageNet* | 2 | 3 | 4 | 5 |
| ResNet18 | 0.98 | 0.96 | 0.91 | 0.96 | 0.97 | 0.97 | 0.96 |
| ResNet34 | 0.98 | 0.98 | 0.82 | 0.98 | 0.98 | 0.98 | 0.99 |
| VGG19 | 0.94 | 0.85 | 0.78 | 0.88 | 0.86 | 0.84 | 0.86 |
| MobileNet | 0.95 | 0.95 | 0.92 | 0.99 | 0.99 | 0.99 | 0.99 |
| SENet18 | 0.98 | 0.85 | 0.80 | 0.97 | 0.97 | 0.97 | 0.93 |
| ShuffleNetV2 | 0.95 | 0.92 | 0.92 | 0.94 | 0.95 | 0.95 | 0.93 |
| EfficientNetB0 | 0.98 | 0.84 | 0.93 | 0.99 | 0.99 | 0.99 | 0.98 |
| RegNetX_200MF | 0.98 | 0.87 | 0.97 | 0.98 | 0.99 | 0.99 | 0.98 |
| SimpleDLA | 0.98 | 0.94 | 0.84 | 0.99 | 0.99 | 0.98 | 0.99 |

The results are provided in Table 6. We recorded the average values of region count and generalization gap for these data points in the second and third columns to show their changes during the training process. We observe that the correlation is very low at initialization, but steadily increases during training. This suggests that the metric of region count is not a property of the neural network initialization, but rather inherently involved with the neural network training algorithm.

Table 6: **The evolution of region count, generalization gap and their correlation.** The correlation is very low at initialization, but steadily increases during training.

| Training Epoch | Region Counts | Generalization Gap | Correlation |
|---|---|---|---|
| 0 | 1.13 | N/A | N/A |
| 20 | 3.14 | 0.11 | -0.53 |
| 40 | 3.02 | 0.07 | -0.29 |
| 60 | 3.10 | 0.18 | 0.35 |
| 80 | 3.22 | 0.36 | 0.77 |
| 100 | 3.25 | 0.48 | 0.98 |
| 130 | 3.28 | 0.51 | 0.97 |
| 160 | 3.27 | 0.51 | 0.98 |
| 200 | 3.26 | 0.49 | 0.98 |

## Appendix F. More ablation studies

**Gradient Optimizers.** We calculate the region count of models trained by different optimizers, including SGD, Adam, and Adagrad. The correlation between region count and the generalization gap is consistent for them, as detailed in Table 7.

**Hyperplane Generation Methods.** We explore the effects of different methods for generating the hyperplane in the input space. In the main experiments, we generate a 1-dimensional plane using

Table 7: The impact of optimizers on the correlation between region counts and generalization gap.

| Network \ Optimizer | *SGD* | *Adam* | *Adagrad* |
|---|---|---|---|
| ResNet18 | 0.98 | 0.92 | 0.96 |
| ResNet34 | 0.98 | 0.92 | 0.91 |
| VGG19 | 0.94 | 0.92 | 0.87 |
| MobileNet | 0.95 | 0.95 | 0.99 |
| SENet18 | 0.98 | 0.78 | 0.91 |
| ShuffleNetV2 | 0.95 | 0.83 | 0.99 |
| EfficientNetB0 | 0.98 | 0.97 | 0.99 |
| RegNetX_200MF | 0.98 | 0.95 | 0.99 |
| SimpleDLA | 0.98 | 0.95 | 0.88 |

random pairs of samples from the training set and calculated the number of distinct regions between them. In this section, we apply various techniques for plane generation: selecting two data points from the test set, choosing one data point from the training set and extending it in a random direction by a fixed length. We calculate the number of regions for each of these setups. The results in Table 8 are consistent across different hyperplane computational approaches.

Table 8: The impact of calculation methods on the correlation between region counts and generalization gap.

| Network \ Counting | *Test* | *Train* | *Random* |
|---|---|---|---|
| ResNet18 | 0.98 | 0.98 | 0.98 |
| ResNet34 | 0.98 | 0.96 | 0.94 |
| VGG19 | 0.94 | 0.89 | 0.78 |
| MobileNet | 0.95 | 0.94 | 0.88 |
| SENet18 | 0.98 | 0.96 | 0.99 |
| ShuffleNetV2 | 0.95 | 0.95 | 0.92 |
| EfficientNetB0 | 0.98 | 0.98 | 0.92 |
| RegNetX_200MF | 0.98 | 0.97 | 0.92 |
| SimpleDLA | 0.98 | 0.97 | 0.96 |

**Data Augmentations.** Mixup [51] is a data augmentation technique that creates training samples by linearly interpolating pairs of input data and their corresponding labels. We train a ResNet-18 model using mixup, with other hyperparameters in Table 1. The plot in Figure 7 illustrates that Mixup induces smoother decision boundaries with smaller region count and has a better generalization performance.

Random crop and random horizontal flip is another way to enhance the diversity of the training dataset. We apply random crop of size $32\times32$ with padding 4 and random horizontal flip with a probability of 0.5 as data augmentations. As depicted in Figure 8, we observe that compared with mixup, random crop and random flip result in a more evident vertical shift in the performance curve.

Both Figure 7 and Figure 8 show that employing these techniques does not alter the correlation between region counts and generalization gaps.

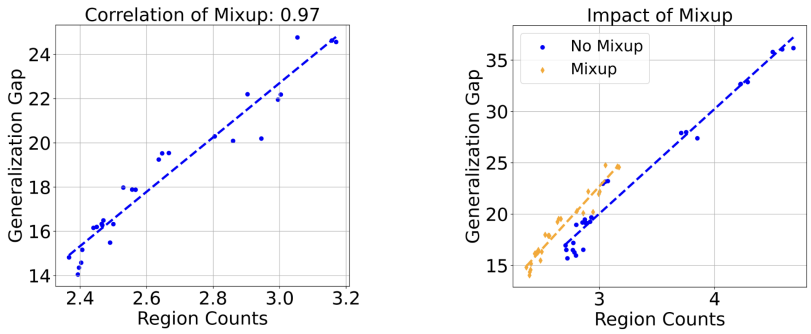


Figure 7: **The impact of mixup.** This figure shows that mixup improves the model's generalization ability and reduces the number of regions in the hyperplane.

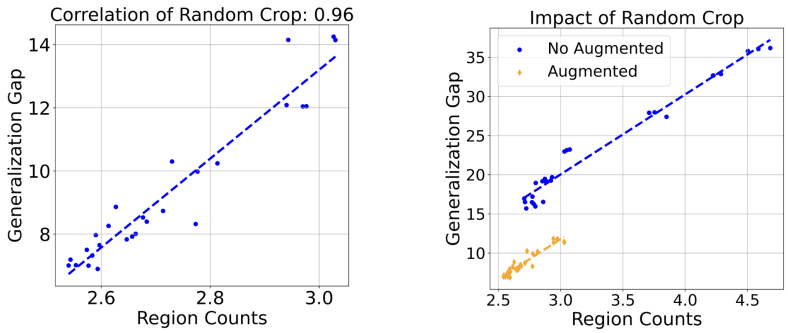


Figure 8: **The impact of random crop and random flip.** Unlike mixup, data augmentation results in a vertical shift in the performance curve, accompanied by a decrease in the number of regions and a more significant enhancement in test accuracy.

## Appendix G. Theortical Analysis

This section contains the proof of the theorems in this paper. First, we utilize the theory of VC (Vapnik-Chervonenkis) dimension to establish a relationship between the number of regions and the generalization gap.

**Theorem 5** *Let $f : \mathbb{R}^d \to \{0, 1\}$ be a model trained for a binary classification task, and $R(f)$ denote the maximum region count for $f$ in a 1D line. Define the function class $\mathcal{F} := \{f : \mathbb{R}^d \to \{0, 1\} | R(f) \leq k\}$, then we have an upper bound for the VC dimension $vc(\mathcal{F})$:*

$$vc(\mathcal{F}) \leq R(\mathcal{F}).$$

*Furthermore, we can bound the generalization gap. Let $X, X_1, X_2, \ldots, X_n$ be independent random data points in $\mathbb{R}^d$, then we get the following bound:*

$$\mathbb{E} \sup_{f \in \mathcal{F}} \left| \frac{1}{n} \sum_{i=1}^{n} f(X_i) - \mathbb{E} f(X) \right| = O\left( \sqrt{\frac{R(\mathcal{F})}{n}} \right).$$

This theorem shows that the region counts is an upper bound of the VC dimension. Therefore, if an algorithm only outputs neural networks with small region counts, then the algorithm can generalize well even though the hypothesis space is large.

The key idea is that classifying $k$ points with alternate labels in a line. For a function class with VC dimension equal to $k$, we can find a classifier in it that correctly fits these points, and this classifier has a region count of at least $k$.

**Proof** We prove this theorem by contradiction. Assume that $k$ is even number (the case of odd number is similar). Suppose the VC dimension of function class $\mathcal{F}$ is not smaller than $k+1$, then we consider the data point $x_1, x_2, \ldots, x_{k+1}$ in a line with alternate labels $y_1 = y_3 = \ldots = y_k = 0$, $y_2 = y_4 = \ldots = y_{k+1} = 1$. By the definition of VC dimension, we can find a function $f$ in $\mathcal{F}$ where every individual data point is assigned to the correct class. So we know that their must exist $k$ decision boundaries between $y_1$ and $y_2$, $y_2$ and $y_3$, ..., $y_k$ and $y_{k+1}$. Therefore, the $k$ decision boundaries split the line into $k+1$ regions while the neighbourhood region have different labels. By the definition of the region count, we know that $N(f) \geq k+1$, which contradicts with the condition $N(f) \leq k$. Therefore, we have proved that the VC dimension of function class $\mathcal{F}$ is no more than $k$.

The proof of the generalization bound is provided in Chapter 8.3.5 of the book [43], by replacing the VC dimension with region count. ■

Next, we present a theoretical analysis to explain why some hyperparameter choices, such as large learning rate, can lead to small region counts.

Consider a two layer ReLU neural network $f_W(x) = \sum_{i=1}^{p} a_i \sigma(w_i^\top x)$. The second layer weights $a_i$ are initialized uniformly from $\{1, -1\}$ and fixed throughout training. Let $\mathcal{D} = \{(x_i, y_i)\}_{1 \leq i \leq N}$ denote the training set. Consider training $f_W$ on $\mathcal{D}$ using gradient descent (GD) with learning rate $\eta$. We choose the quadratic loss $l(W, x, y) = \frac{1}{2}\|y - f_W(x)\|^2$ and denote $L(W) = \frac{1}{N}\sum_{i=1}^{N} l(W, x_i, y_i)$. Denote the GD trajectory as $\{W_i\}_{i \geq 0}$. For two input data $x_a, x_b$, Let $R(x_a, x_b, W)$ denote the region count on the line segment connecting them, and $N(x_a, W)$ denote the number of activated neurons with input $x_a$, i.e., the number of $j$ such that $w_j^\top x_a > 0$.

**Assumption 1** *The training dataset $\mathcal{D} = \{(x_i, y_i)\}_{1 \leq i \leq N}$ satisfies the following two properties:*

1. *$\|x_i\| \geq r$ for all $1 \leq i \leq N$,*
2. *With probability one, any $W \in \{W_i\}_{i \geq 0}$ satisfies $w_i^\top x_j \neq 0$ for all $1 \leq i \leq q, 1 \leq j \leq N$ where the randomness comes from weight initialization.*

The validity of Assumption 1 comes from the fact that the bifurcation zone [4] of ReLU neural networks, which contains its non-differentiable points, has Lebesgue measure zero [5–7]. Therefore, if the distribution of weights are absolutely continuous with respect to the Lebesgue measure, the bifurcation zone can be avoided with probability one. We conjecture that it can be proved rigorously, but leave it as an assumption since the proof diverges from the main content in this paper.

The next assumption characterizes the sharpness along the training trajectory. This is actually from the well-known edge of stability phenomenon [1, 3, 10, 11], which states that the sharpness of neural networks, characterized by the Hessian $\ell_2$ norm, hovers around $\frac{2}{\eta}$.

**Assumption 2 (Edge of Stability)** *There exist a $T \in \mathbb{N}$, such that for $t \geq T$, with we have*

$$\lambda_{\max}(\nabla_W^2 L(W_t)) = \Theta\left(\frac{1}{\eta}\right),$$

*where $\lambda_{\max}$ denotes the maximum eigenvalue of a matrix.*

We are now ready to present the main theorem, which establishes a relationship between region count and learning rate.

**Theorem 6** *Under Assumption 1 and 2, we have that for neural net weights $W_t$ at training step $t \geq T$ with probability one, the average region count $R(X, X', W_t)$ for random training data point $X, X'$ can be bounded as:*

$$_{X,X'}[R(X, X', W_t)] = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} R(x_1, x_2, W_t) \leq O\left(\frac{N}{r^2 \eta}\right).$$

The theorem demonstrates that with a larger learning rate, gradient descent has the implicit bias to yield solutions with smaller region counts. This aligns well with the previous observations.

We then prove two lemmas. We first bound the region count using the activation pattern of ReLU neurons, as stated in the following lemma.

**Lemma 7** *The region count along a pair of data can lower bound the number of active neurons. For two inputs $x_a, x_b$, we have $R(x_a, x_b, W) \leq N(x_a, W) + N(x_b, W) + 2$.*

**Proof** If $R(x_a, x_b, W) \leq 2$, then the equation naturally holds. Next we consider $R(x_a, x_b, W) > 2$. From the definition of region count, one can find $R := R(x_a, x_b, W)$ points on the line segment between $x_a$ and $x_b$, such that the neural network gives different predictions. Denote these points as $\tilde{x}_1, \cdots, \tilde{x}_R$. We have

$$f_W(\tilde{x}_i) f_W(\tilde{x}_{i+1}) < 0, 0 \leq i \leq R-1.$$

Consider $\tilde{x}_i, \tilde{x}_{i+1}, \tilde{x}_{i+2}$. Since the neural network gives alternating predictions on these three points, it is nonlinear and has activation sign changes on the line segment connecting them. Therefore, we can find a $1 \leq n(i) \leq p$, such that $(w_{n(i)}^\top \tilde{x}_i)(w_{n(i)}^\top \tilde{x}_{i+2}) < 0$. Since $w_{n(i)}^\top x$ is linear in $x$, this implies that

$$(w_{n(i)}^\top x_a)(w_{n(i)}^\top x_b) < 0.$$

We prove it by contradiction. If they have the same sign then the convex combination of them have the same sign so $(w_{n(i)}^\top \tilde{x}_i)(w_{n(i)}^\top \tilde{x}_{i+2}) \geq 0$.

We have the following two observations about $n(i)$. Firstly, we can choose an $n(i)$ such that $a_{n(i)} w_{n(i)}^\top \tilde{x}_{i+2}$ and $f_W(\tilde{x}_{i+2})$ have the same sign, since there exists at least one such neuron that contributes to the sign change of $f_W$. This implies that $n(i) \neq n(i+1)$, since $f_W(\tilde{x}_i)$ have alternating signs. Secondly, since $w_{n(i)}^\top x$ is a linear function in $x$, it can only changes sign for at most one time. This implies that $n(i) \neq n(j)$ if $j - i \geq 2$. Putting them together, we know that $n(i) \neq n(j)$ for $i \neq j$.

Recall that for each $1 \leq i \leq R-2$, we have $(w_{n(i)}^\top x_a)(w_{n(i)}^\top x_b) < 0$. Therefore, there exists $R-2$ neurons that are activated for either $x_a$ or $x_b$. This gives $N(x_a, W) + N(x_b, W) \geq R-2$, which completes the proof.

■

Then we prove that the activation pattern gives a bound on the smoothness of the training loss.

**Lemma 8** *The sharpness of a neural network can be upper bounded by the number of active neurons: $\lambda_{\max}\left(\nabla_W^2 L(W)\right) \geq \frac{r^2}{N^2} \sum_{i=1}^{N} N(x_i, W)$.*

**Proof** The Hessian of $l(W, x, y)$ can be expressed as

$$\nabla_W^2 l(W, x, y) = \begin{bmatrix} v_1 v_1^\top & \cdots & v_1 v_p^\top \\ \vdots & & \vdots \\ v_p v_1^\top & \cdots & v_p v_p^\top \end{bmatrix},$$

where $v_i = a_i \sigma'(w_i^\top x) x$. Suppose $V = [v_1^\top, \cdots, v_p^\top]$. As the nonzero eigenvalue of $V^\top V$ and $VV^\top$ is the same, this implies that

$$\lambda_{\max}(\nabla_W^2 l(W, x, y)) = \lambda_{\max}(VV^\top) = \sum_{i=1}^{p} \|v_i\|_2^2 = \sum_{i=1}^{p} \sigma'(w_i^\top x)\|x\|^2 \geq \sum_{i=1}^{p} \sigma'(w_i^\top x) r^2.$$

From the definition of $\nabla_W^2 L(W)$ and the positive definiteness of Hessian matrices, we know that

$$\lambda_{\max}\left(\nabla_W^2 L(W)\right) = \frac{1}{N}\lambda_{\max}\left(\sum_{i=1}^{N} \nabla_W^2 l(W, x_i, y_i)\right) \geq \frac{1}{N^2}\sum_{i=1}^{N} \lambda_{\max}\left(\nabla_W^2 l(W, x_i, y_i)\right).$$

Plug in the previous calculation and use the definition of $N(x)$, we have

$$\lambda_{\max}\left(\nabla_W^2 L(W)\right) \geq \frac{r^2}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{p} \sigma'(w_i^\top x_j) = \frac{r^2}{N^2}\sum_{i=1}^{N} N(x_i, W).$$

■

**Proof** [Proof of Theorem 6] Theorem 6 is a direct consequence of Assumption 2 and the following two lemmas.

$$\begin{aligned}
{}_{X,X'}[R(X, X', W_t)] &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} R(x_1, x_2, W_t) \\
&\leq \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} (N(x_i, W_t) + N(x_j, W_t) + 2) \\
&= \frac{2}{N}\sum_{i=1}^{N} (N(x_i, W_t) + 1) \\
&\leq \frac{2N}{r^2}\lambda_{\max}(\nabla_W^2 L(W_t)) + 2 \\
&= O\left(\frac{N}{r^2 \eta}\right)
\end{aligned}$$

■