Dropout Reduces Underfitting

Oscar Xu

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-70
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-70.html

May 8, 2023

Dropout Reduces Underfitting

by Zhiqiu Xu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Trevor Darrell
Research Advisor

5/1/23

(Date)

* * * * * * *

Professor Jiantao Jiao
Second Reader

5/1/23

(Date)

Dropout Reduces Underfitting

by

Zhiqiu Xu

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Trevor Darrell, Chair
Professor Jiantao Jiao

Spring 2023

Dropout Reduces Underfitting

Abstract

Dropout Reduces Underfitting

by

Zhiqiu Xu

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Trevor Darrell, Chair

Introduced by Hinton et al. in 2012, dropout has stood the test of time as a regularizer for preventing overfitting in neural networks. In this study, we demonstrate that dropout can also mitigate *underfitting* when used at the start of training. During the early phase, we find dropout reduces the directional variance of gradients across mini-batches and helps align the mini-batch gradients with the entire dataset's gradient. This helps counteract the stochasticity of SGD and limit the influence of individual batches on model training. Our findings lead us to a solution for improving performance in underfitting models - *early dropout*: dropout is applied only during the initial phases of training, and turned off afterwards. Models equipped with early dropout achieve *lower* final training loss compared to their counterparts without dropout. Additionally, we explore a symmetric technique for regularizing overfitting models - *late dropout*, where dropout is not used in the early iterations and is only activated later in training. Experiments on ImageNet and various vision tasks demonstrate that our methods consistently improve generalization accuracy. Our results encourage more research on understanding regularization in deep learning and our methods can be useful tools for future neural network training, especially in the era of large data. Code is available at https://github.com/facebookresearch/dropout.

To my parents, Jing Ye and Jie Xu

Contents

Acknowledgments

I would like to express my sincere appreciation to my supervisor, Prof. Trevor Darrell, for providing me with invaluable guidance and unwavering support throughout my research. I want to thank Prof. Trevor Darrell and Prof. Jiantao Jiao for being my master committee and providing valuable feedback for this project. I also want to thank Zhuang Liu, Joseph Jin, and Zhiqiang Shen for their contributions to this project. Their expertise and support have been crucial in advancing my research and achieving the goals.

My heartfelt thanks go to Dr. Zhuang Liu for his comprehensive mentorship, which began at the outset of my research journey in Fall 2021. I am also grateful to my mentor Ruiqi Zhong for inspiring me with his profound insights in this era of artificial intelligence. Their guidance was the primary reason for my venture into research.

I am extremely grateful for the wonderful opportunity to continue my academic journey as a Ph.D. student at the University of Pennsylvania, working under the supervision of Prof. Lingjie Liu and Prof. Kostas Daniilidis.

I would like to extend my special thanks to Haodi Zou and all my friends from UC Berkeley, Shanghai Foreign Language School, and other wonderful places. My life shines because of you.

Finally, I would like to thank my parents Jing Ye and Jie Xu, who bring me to this beautiful world. I am continually reminded of how fortunate I am to have such loving and supportive parents who have always encouraged me to pursue what I want. Their love is the cornerstone of my life, and I am forever grateful for everything they have done for me.

Chapter 1

Introduction

1.1 Dropout

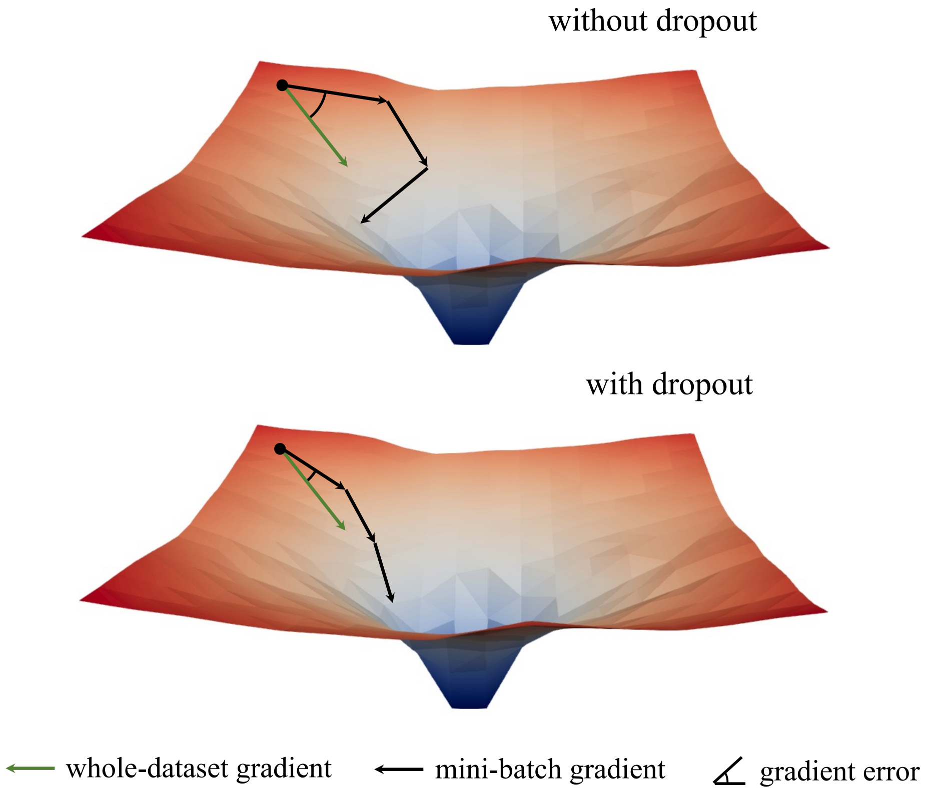
The year 2022 marks a full decade since AlexNet's pivotal "ImageNet moment" [34], which launched a new era in deep learning. It is no coincidence that dropout [29] also celebrates its tenth birthday in 2022: AlexNet employed dropout to substantially reduce its overfitting, which played a critical role in its victory at the ILSVRC 2012 competition. Without the invention of dropout, the advancements we currently see in deep learning might have been delayed by years.

Dropout has since become widely adopted as a regularizer to mitigate overfitting in neural networks. It randomly deactivates each neuron with probability p, preventing different features from co-adapting with each other [29, 55]. After applying dropout, training loss typically increases, while test error decreases, narrowing the model's generalization gap.

Deep learning evolves at an incredible speed. Novel techniques and architectures are continuously introduced, applications expand, benchmarks shift, and even convolution can be gone [14] – but dropout has stayed. It continues to function in the latest AI achievements, including AlphaFold's protein structure prediction [31], and DALL-E 2's image generation [51], demonstrating its versatility and effectiveness.

Despite the sustained popularity of dropout, its strength, represented by the drop rate p, has generally been decreasing over the years. In the original dropout work [29], a default drop rate of 0.5 was used. However, lower drop rates, such as 0.1, have been frequently adopted in recent years. Examples include training BERT [12] and Vision Transformers [14].

The primary driver for this trend is the exploding growth of available training data, making it increasingly difficult to overfit. In addition, advancements in data augmentation techniques [72, 9] and algorithms for learning with unlabeled or weakly-labeled data [4, 50, 24] have provided even more data to train on than the model can fit to. As a result, we may soon be confronting more problems with *underfitting* instead of overfitting.


without dropout

with dropout

Figure 1.1: **Dropout in early training** helps the model produce mini-batch gradient directions that are more consistent and aligned with the overall gradient of the entire dataset.

1.2 Early Dropout

Would dropout lose its relevance should such a situation arise? In this study, we demonstrate an alternative use of dropout for tackling underfitting. We begin our investigation into dropout training dynamics by making an intriguing observation on gradient norms, which then leads us to a key empirical finding: during the initial stages of training, dropout reduces gradient variance across mini-batches and allows the model to update in more consistent directions. These directions are also more aligned with the entire dataset's gradient direction (Figure 1.1). Consequently, the model can optimize the training loss more effectively with respect to the whole training set, rather than being swayed by individual mini-batches. In other words, dropout counteracts SGD and prevents excessive regularization due to randomness in sampling mini-batches during early training.

Based on this insight, we introduce *early dropout* – dropout is only used during early training – to help underfitting models fit better. Early dropout *lowers* the final training loss compared to no dropout and standard dropout. Conversely, for models that already use standard dropout, we propose to remove dropout during earlier training epochs to mitigate overfitting. We refer to this approach as *late dropout* and demonstrate that it improves generalization accuracy for large models. Figure 1.2 provides a comparison of standard dropout, early dropout, and late dropout.

We evaluate early and late dropout using different models on image classification and down-

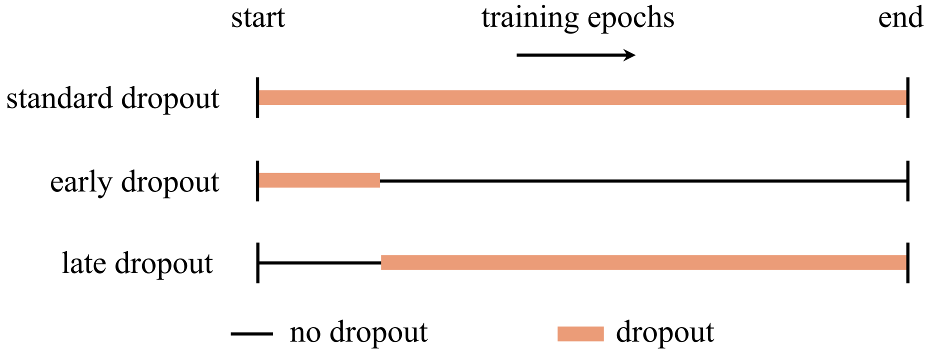


Figure 1.2: **Standard, early and late dropout.** We propose early and late dropout. Early dropout helps underfitting models fit the data better and achieve lower training loss. Late dropout helps improve the generalization performance of overfitting models.

stream tasks. Our methods consistently yield better results than both standard dropout and no dropout. We hope our findings can offer novel insights into dropout and overfitting, and motivate further research in developing neural network regularizers.

1.3 Related Work

Neural network regularizers

Weight decay, or L_2 regularization, is one of the most commonly used regularization for training neural networks. Related to our findings, Krizhevsky et al. [34] observes that using weight decay decreases the training loss for AlexNet. L_1 regularization [60] can promote sparsity and select features [41]. Label smoothing [58] replaces one-hot targets output with soft probabilities. Data augmentation [72, 9] can also serve as a form of regularization. In particular, methods that randomly remove input parts, e.g., hide-and-seek [35], cutout [13] and random ereasing [73], can be seen as dropout applied at the input layer only.

Dropout methods

Dropout has many variants aimed at improving or adapting it. DropConnect [66] randomly deactivates network weights instead of neurons. Variational dropout [32] adaptively learns dropout rates for different parts of the network from a Bayesian perspective. Spatial dropout [62] drops entire feature maps in a ConvNet, and DropBlock [19] drops continuous regions in ConvNet feature maps. Other valuable contributions include analyzing dropout properties [2, 1, 68], applying dropout for compressing networks [44, 20] and representing uncertainty [16, 17]. We recommend the survey by [36] for a comprehensive overview.

Scheduled dropout

Neural networks generally tend to show overfitting behaviors more at later stages of training, which is why *early stopping* is often used to reduce overfitting. Curriculum dropout [45] proposes to increase the dropout rate as training progresses to more specifically address late-stage overfitting. NASNet [75] and EfficientNet-V2 [59] also increase the strength of dropout / drop-path [37] during neural architecture search. On the other hand, annealed dropout [53] gradually decreases dropout rates to near the end of training. Our approaches differ from previous research as we study dropout's effect in addressing *underfitting* rather than regularizing overfitting.

Chapter 2

Revisiting Overfitting vs. Underfitting

2.1 Overfitting

Overfitting occurs when a model is trained to fit the training data excessively well but generalizes poorly to unseen data. The model's capacity and the dataset scale are among the most critical factors in determining overfitting, along with other factors such as training length. Larger models and smaller datasets tend to lead to more overfitting.

We conduct several simple experiments to clearly illustrate this trend. First, when the model remains the same, but we use less data, the gap between training accuracy and test accuracy increases, leading to overfitting. Figure 2.1 (top) demonstrates this trend with ViT-Tiny/32 results trained on various amounts of ImageNet data. Second, when the model capacity increases while keeping the dataset size constant, the gap also widens. Figure 2.1 (bottom) illustrates this with ViT-Tiny (T), Small (S), and Base (B)/32 models trained on the same 100% ImageNet data. We train all models with a fixed 4,000 iterations without data augmentations.

2.2 Dropout

We briefly review the dropout method. At each training iteration, a dropout layer randomly sets each neuron to zero with a certain probability for its input tensor. During inference, all neurons are active but are scaled by a coefficient to maintain the same overall scale as in training. As each sample is trained by a different sub-network, dropout can be seen as an implicit ensemble of exponentially many models. It is a fundamental building block of deep learning and has been used to prevent overfitting in various of neural architectures and applications [65, 12, 51].

2.3 Stochastic depth

Various efforts have been made to design dropout variants [66, 25, 19]. In this work, we also consider a dropout variant called stochastic depth [30] (s.d. for short), which is designed for

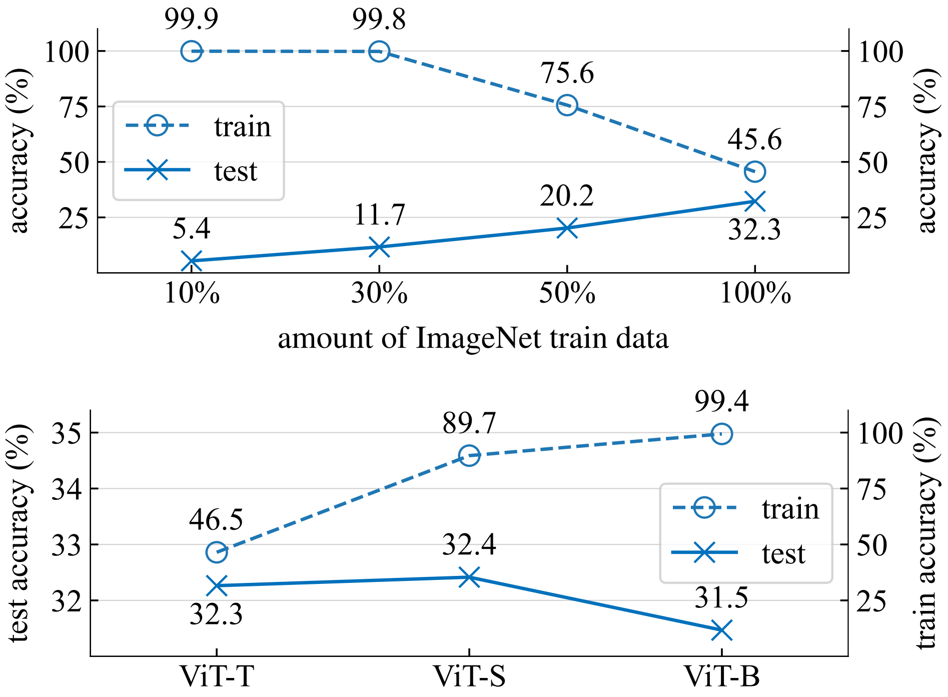


Figure 2.1: **Overfitting** can occur when either the amount of data decreases (top) or the capacity of the model increases (bottom).

regularizing residual networks [22]. For each sample or mini-batch, the network randomly selects a subset of residual blocks to skip, making the model shallower and thus earning its name "stochastic depth". It is commonly seen in modern vision networks, including DeiT [64], ConvNeXt [40] and MLP-Mixer [61]. Several recent models [56, 61] use s.d. together with dropout. Since s.d. can be viewed as specialized dropout at the residual block level, the term "dropout" that we use later could also encompass s.d., depending on the context.

2.4 Drop rate

The probability of setting a neuron to zero in dropout is referred to as the drop rate p, a hugely influential hyper-parameter. As an example, in Swin Transformers and ConvNeXts, the only training hyper-parameter that varies with the model size is the stochastic depth drop rate.

We apply dropout to regularize the ViT-B model and experiment with different drop rates. As shown in Figure 2.2, setting the drop rate too low does not effectively prevent overfitting, whereas setting it too high results in over-regularization and decreased test accuracy. In this case, the optimal drop rate for achieving the highest test accuracy is 0.15.

Different model architectures use different drop rates, and the selection of optimal drop rate p heavily depends on the network model size and the dataset size. In Figure 2.3, we plot the best

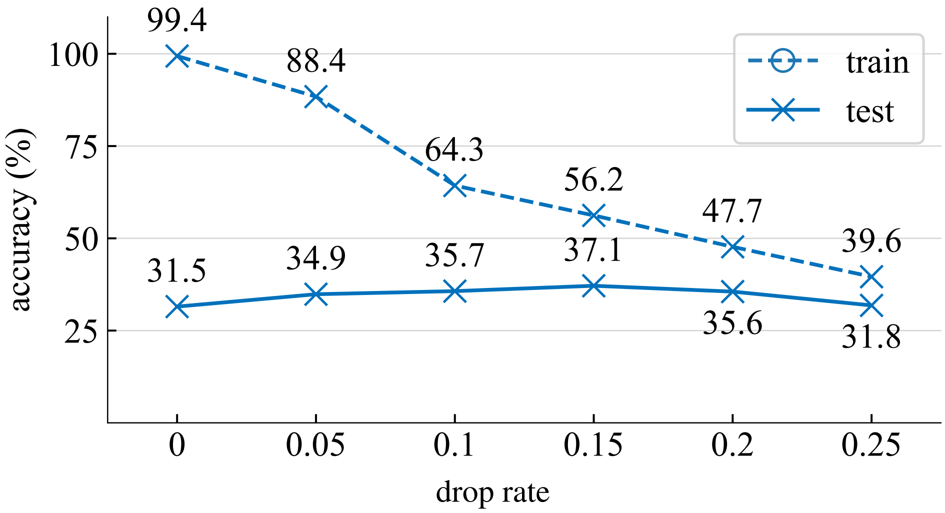


Figure 2.2: **Drop rate influence**. The training accuracy decreases as the drop rate increases, however, there is an optimal drop rate ($p = 0.15$ in this case) that maximizes the test accuracy.

dropout rate for model and data settings from Figure 2.1. We perform a hyper-parameter sweep for drop rate at intervals of 0.05 for each setting. From Figure 2.3, we observe that when the data is large enough, or when the model is small enough, the best drop rate p is 0, indicating that using dropout may not be necessary and could harm the model's generalization accuracy by underfitting the data.

2.5 Underfitting

In the literature, the drop rate used for dropout has generally decreased over the years. Earlier models such as VGG [54] and GoogleNet [57] use 0.5 or higher drop rates; ViTs [14] use a moderate rate of 0.1 on ImageNet and do not use dropout when pre-training on the much larger JFT-300M dataset; recent language-supervised or self-supervised vision models [50, 24] do not use dropout. This trend is likely due to the increasing size of datasets. The model does not overfit very easily to immense data.

With the rapidly growing amount of data being generated and distributed globally, it is possible that the scale of the available data may soon outpace the capacities of the models we train. While data is generated at a speed of quintillion bytes per day, models still need to be stored and run on finite physical devices such as servers, data centers, or mobile phones. Given such a contrast, future models may have more trouble fitting data properly rather than overfitting too severely. As our experiments above demonstrate, in such settings, standard dropout may not help generalization as a regularizer. Instead, we need tools to help models fit vast amounts of data better and reduce *underfitting*.

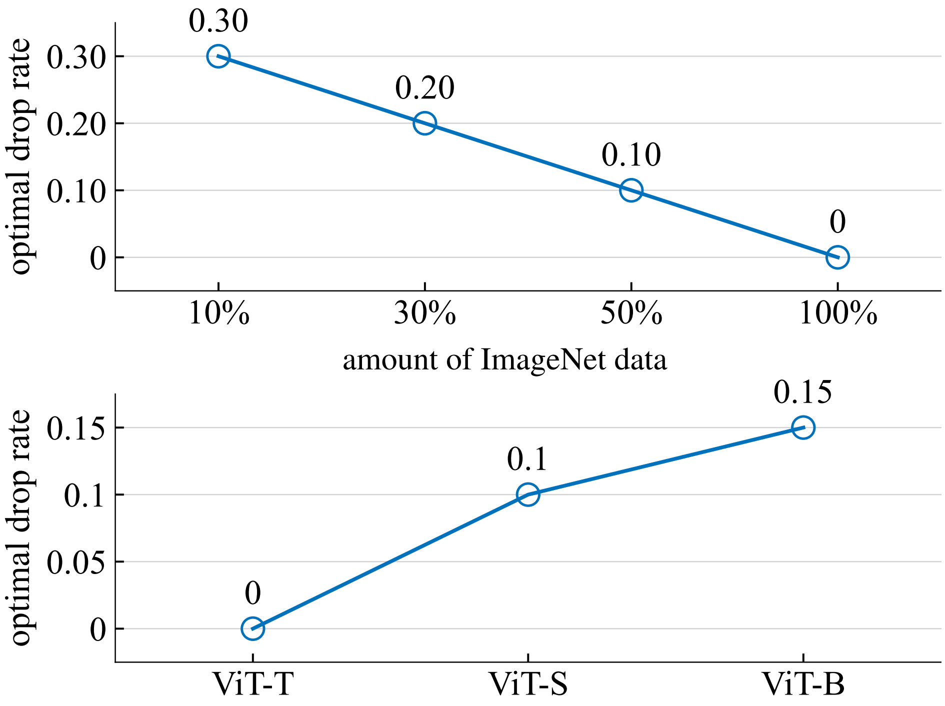


Figure 2.3: **Optimal drop rate**. Training with a larger dataset (top) or using a smaller model (bottom) both result in a lower optimal drop rate, which may even reach 0 in some cases.

Chapter 3

How Dropout Can Reduce Underfitting

In this study, we explore whether dropout can be used as a tool to reduce underfitting. To this end, we conduct a detailed analysis on the training dynamics of dropout using our proposed tools and metrics. We compare two ViT-T/16 training processes on ImageNet [11]: one without dropout as the baseline, and the other with a 0.1 dropout rate throughout training.

3.1 Gradient norm

We begin our analysis by investigating the impact of dropout on the strength of gradients g, measured by their L_2 norm $||g||_2$. For the dropout model, we measure the entire model's gradient, even though a subset of weights may have been deactivated due to dropout. As shown in Figure 3.1 (left), the dropout model produces gradients with smaller norms, indicating that it takes smaller steps at each gradient update.

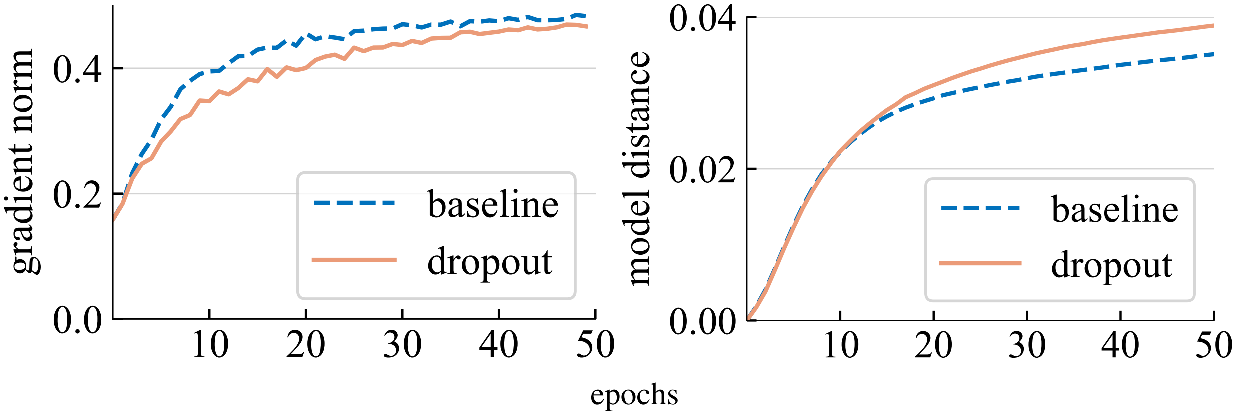


Figure 3.1: **Gradient norm** (left) and **model distance** (right). The model with dropout has smaller gradient magnitudes, but it moves a greater distance in the parameter space.

3.2 Model distance

Since the gradient steps are smaller, we expect the dropout model to travel a smaller distance from its initial point than the baseline model. To measure the distance between the two models, we use the L_2 norm, represented by $||W_1 - W_2||_2$, where W_i denotes the parameters of each model. In Figure 3.1 (right), we plot each model's distance from its random initialization. However, to our surprise, the dropout model actually moved by a *larger* distance than the baseline model, contrary to what we initially anticipated based on the gradient norms.

Let us imagine two people walking. One walks with large strides while the other walks with small strides. Despite this, the person with smaller strides covers a greater distance from their starting point over the same time period. Why? This may be because the person is walking in a more *consistent* direction, whereas the person with larger strides may be taking random, meandering steps and not making much progress in any one particular direction.

3.3 Gradient direction variance

We hypothesize the same for our two models: the dropout model is producing more consistent gradient directions across mini-batches. To test this, we collect a set of mini-batch gradients G by training a model checkpoint on randomly selected batches. We then measure the gradient direction variance (GDV) by computing the average pairwise cosine distance:

$$GDV = \frac{2}{|G| \cdot (|G| - 1)} \sum_{g_i, g_j \in G, i \neq j} \underbrace{\frac{1}{2}(1 - \frac{< g_i, g_j >}{||g_i||_2 \cdot ||g_j||_2})}_{\text{cosine distance}}$$

As seen in Figure 3.2, the comparison of variance supports our hypothesis. Up to a certain iteration (approximately 1000), the dropout model exhibits a lower gradient variance and moves in a more consistent direction.

3.4 Gradient direction error

However, the question remains – what should be the correct direction to take? To fit the training data, the underlying objective is to minimize the loss on the entire training set, not just on any single mini-batch. We compute the gradient for a given model on the whole training set, where dropout is set to inference mode to capture the full model's gradient. Then, we evaluate how far the actual mini-batch gradient g_{step} is from this whole-dataset "ground-truth" gradient $\hat{g}$. We define the average cosine distance from all $g_{step} \in G$ to $\hat{g}$ as the gradient direction "error" (GDE):

$$GDE = \frac{1}{|G|} \sum_{g_{step} \in G} \underbrace{\frac{1}{2}(1 - \frac{< g_{step}, \hat{g} >}{||g_{step}||_2 \cdot ||\hat{g}||_2})}_{\text{cosine distance}}$$

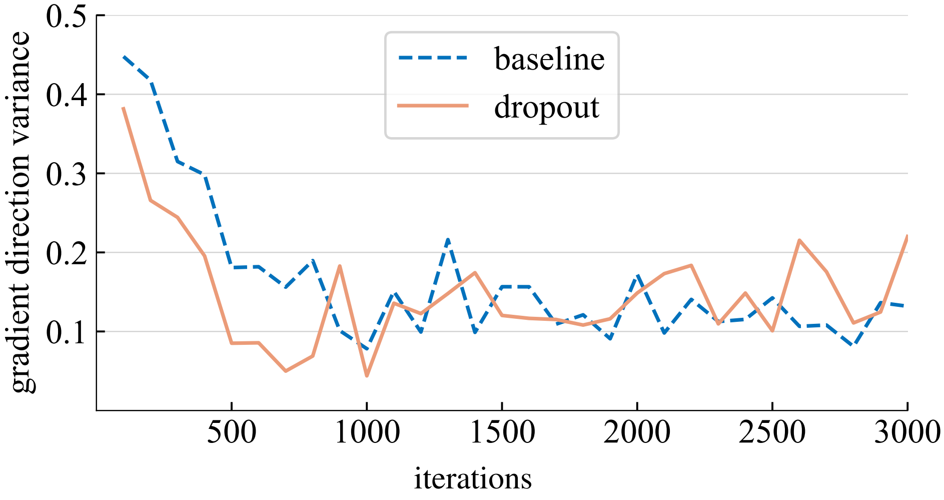


Figure 3.2: **Gradient direction variance**. The model with dropout produces more consistent mini-batch gradients during the initial phase of training, up to approximately 1000 iterations.

We calculate this error term and plot it in Figure 3.3. At the beginning of training, the dropout model's mini-batch gradients have smaller distances to the whole-dataset gradient, indicating that it is moving in a more desirable direction for optimizing the total training loss (as illustrated in Figure 1.1). After approximately 1000 iterations, however, the dropout model produces gradients that are farther away. This could be the turning point where dropout transitions from reducing underfitting to reducing overfitting.

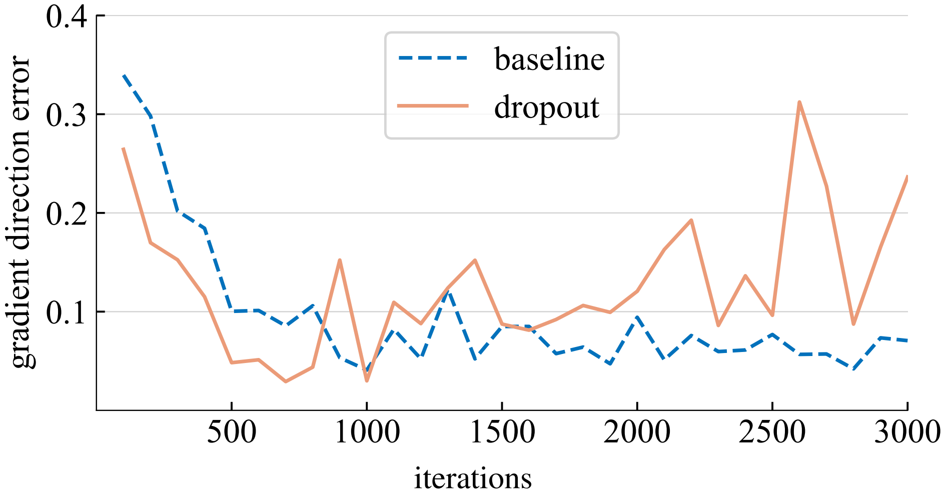


Figure 3.3: **Gradient direction error**. Dropout leads to mini-batch gradients that are more aligned with the gradient of the entire dataset at the beginning of training.

The experiments described above use ViT optimized with AdamW [42], but we observe similar trends when using other optimizers such as SGD and momentum SGD, or architectures such as

Swin Transformer.

3.5 Bias-variance tradeoff

This analysis at early training can be viewed through the lens of the bias-variance tradeoff. For no-dropout models, an SGD mini-batch provides an unbiased estimate of the whole-dataset gradient because the expectation of the mini-batch gradient is equal to the whole-dataset gradient. However, with dropout, the estimate becomes more or less biased, as the mini-batch gradients are generated by different sub-networks, whose expected gradient may not match the full network's gradient. Nevertheless, the gradient variance is significantly reduced, leading to a reduction in gradient error. Intuitively, this reduction in variance and error helps prevent the model from overfitting to specific batches, especially during the early stages of training when the model is undergoing significant changes.

Chapter 4

Approach

From the analysis above, we know that using dropout early can potentially improve the model's ability to fit the training data. Based on this observation, we present our approaches.

4.1 Underfitting and overfitting regimes

Whether it is desirable to fit the training data better depends on whether the model is in an underfitting or overfitting regime, which can be difficult to define precisely. In this work, we use the following criterion and find it is effective for our purpose: if a model generalizes better with standard dropout, we consider it to be in an overfitting regime; if the model performs better without dropout, we consider it to be in an underfitting regime. The regime a model is in depends not only on the model architecture but also on the dataset used and other training parameters.

4.2 Early dropout

In their default settings, models at underfitting regimes do not use dropout. To improve their ability to fit the training data, we propose *early dropout*: *using dropout before a certain iteration, and then disabling it for the rest of training*. Our experiments show that early dropout reduces final training loss and improves accuracy.

4.3 Late dropout

Overfitting models already have standard dropout included in their training settings. During the early stages of training, dropout may cause overfitting unintentionally, which is not desirable. To reduce overfitting, we propose *late dropout*: *not using dropout before a certain iteration, and then using it for the rest of training.* This is a symmetric approach to early dropout.

4.4 Hyper-parameters

Our methods are straightforward both in concept and implementation, illustrated in Figure 1.2. They require two hyper-parameters: 1) the number of epochs to wait before turning dropout on or off. Our results show that this choice can be robust enough to vary from 1% to 50% of the total epochs. 2) The drop rate p, which is similar to the standard dropout rate and is also moderately robust.

Chapter 5

Experiments

We conduct empirical evaluations on ImageNet-1K classification with 1,000 classes and 1.2M training images [11] and report top-1 validation accuracy.

5.1 Early Dropout

Settings

To evaluate early dropout, we choose small models at underfitting regimes on ImageNet-1K, including ViT-T/16 [64], Mixer-S/32 [61], ConvNeXt-Femto (F) [69], and a Swin-F [39] of similar size to ConvNeXt-F. These models have 5-20M parameters and are relatively small for ImageNet-1K. We conduct separate evaluations for dropout and stochastic depth (s.d.), i.e., only one is used in each experiment. We use the training recipe from ConvNeXt [40] as our basic recipe. The drop rates are selected from $0.1, 0.2, 0.3$ for dropout and $0.3, 0.5, 0.7$ for s.d. Each result is average with 3 seeds, and the average standard deviation is 0.142%. The usage of dropout does not affect training time noticeably. See Appendix A for more details on the experimental setup.

Results

Table 5.1 presents the results. Early dropout consistently improves the test accuracy, and also *decreases* the training loss, indicating dropout at an early stage helps the model fit the data better. The results are compared to standard dropout and s.d. using a drop rate of 0.1, which both have a negative impact on the models.

Additionally, we double the training epochs and reduce mixup [72] and cutmix [71] strength to arrive at an improved recipe for these small models. Table 5.2 (bottom) shows the results. The baselines now achieve much-improved accuracy, sometimes surpassing previous literature results by a large margin. Nevertheless, early dropout still provides a further boost in accuracy.

model	top-1 acc.	change	train loss	change
		results with basic recipe		
ViT-T	73.9	-	3.443	-
+ standard dropout	67.9	↓6.0	3.885	↑0.442
+ standard s.d.	72.6	↓1.3	3.681	↑0.238
+ early dropout	**74.3**	↑0.4	3.394	↓0.049
+ early s.d.	**74.4**	↑0.5	3.435	↓0.008
Mixer-S∗	68.7	-	-	-
Mixer-S	71.0	-	3.635	-
+ standard dropout	67.1	↓3.9	4.058	↑0.423
+ standard s.d.	70.5	↓0.5	3.813	↑0.178
+ early dropout	**71.3**	↑0.3	3.591	↓0.044
+ early s.d.	**71.7**	↑0.7	3.552	↓0.083
ConvNeXt-F	76.1	-	3.472	-
+ standard s.d.	75.5	↓0.6	3.647	↑0.175
+ early s.d.	**76.3**	↑0.2	3.443	↓0.029
Swin-F	74.3	-	3.411	-
+ standard dropout	71.6	↓2.7	3.717	↑0.306
+ standard s.d.	73.7	↓0.6	3.644	↑0.233
+ early dropout	**74.7**	↑0.4	3.378	↓0.033
+ early s.d.	**75.2**	↑0.9	3.353	↓0.058

Table 5.1: **Classification accuracy on ImageNet-1K. (Basic Recipe)** Early dropout or stochastic depth (s.d.) lowers training loss and improves test accuracy for underfitting models, while standard ones hurt both. Literature baselines: ∗ [61], †[64], ‡[69].

5.2 Analysis

We carry out ablation studies to understand the characteristics of early dropout. Our default setting is ViT-T and early dropout with the improved recipe.

Scheduling strategies

In previous studies, different strategies for scheduling dropout or related regularizers have been explored. These strategies typically involve either gradually increasing [45, 75, 59] or decreasing [53] the strength of dropout over the entire or nearly the entire training process. The purpose of these strategies, however, is to reduce overfitting rather than underfitting.

For comparison, we also evaluate linear decreasing / increasing strategies where the drop rate starts from p / 0 and ends at 0 / p, as well as previously proposed curriculum [45] and annealed [53] strategies. For all strategies, we conduct a hyper-parameter sweep for the rate p. The results are

model	top-1 acc.	change	train loss	change
ViT-T†	72.8	-	-	-
ViT-T‡	75.5	-	-	-
ViT-T	76.3	-	3.033	-
+ standard dropout	71.5	↓4.8	3.437	↑0.404
+ standard s.d.	75.6	↓0.7	3.243	↑0.210
+ early dropout	**76.7**	↑0.4	2.991	↓0.042
+ early s.d.	**76.7**	↑0.4	3.022	↓0.011
ConvNeXt-F‡	77.5	-	-	-
ConvNeXt-F	77.5	-	3.011	-
+ standard s.d.	77.4	↓0.1	3.177	↑0.166
+ early s.d.	**77.7**	↑0.2	2.990	↓0.021
Swin-F	76.1	-	2.989	-
+ standard dropout	73.5	↓2.6	3.305	↑0.316
+ standard s.d.	75.6	↓0.5	3.241	↑0.252
+ early dropout	**76.6**	↑0.5	2.966	↓0.023
+ early s.d.	**76.6**	↑0.5	2.958	↓0.031

Table 5.2: **Classification accuracy on ImageNet-1K. (Improved Recipe)** Early dropout or stochastic depth (s.d.) lowers training loss and improves test accuracy for underfitting models, while standard ones hurt both. Literature baselines: $*$ [61], †[64], ‡[69].

presented in Table 5.3a. All strategies produce either similar or much worse results than no-dropout. This suggests existing dropout scheduling strategies are not effective for underfitting.

Early dropout scheduling

There is still a question on how to schedule the drop rate in the early phase. Our experiments use a linear decreasing schedule from an initial value p to 0 by default. A simpler alternative is to use a constant value. It can also be useful to consider a cosine decreasing schedule commonly adopted for learning rate schedules. The optimal p value for each option may differ and we compare the best result for each option. Table 5.3b presents the results. All three options manifest similar results and can serve as valid choices. This indicates early dropout does not depend on one particular schedule to work. Additional results for constant early dropout can be found in Appendix C.

Model sizes

According to our analysis in Section 3, early dropout helps models fit better to the training data. This is particularly useful for underfitting models like ViT-T. We take ViTs of increasing sizes, ViT-T, ViT-S, and ViT-B, and examine the trend in Table 5.3c. The baseline column represents the

strategy	acc.	train loss
no dropout	76.3	3.033
constant	71.5	3.437
increasing	75.2	3.285
decreasing	74.7	3.113
annealed	76.3	3.004
curriculum	70.4	3.490
early	**76.7**	2.996

(a) **Scheduling strategies**. Early dropout outperforms alternative strategies.

schedule	acc.	train loss
linear	**76.7**	2.991
constant	76.6	3.025
cosine	76.6	2.988

(b) **Early dropout scheduling**. Early dropout is robust to various schedules.

model	baseline	early dropout
ViT-T	76.3	**76.7**
ViT-S	80.4	**80.8**
ViT-B	78.7	78.7

(c) **Model size**. Early dropout does not help models at overfitting regimes.

Table 5.3: **Early dropout ablation results** with ViT-T/16 on ImageNet-1K.

results obtained by the best standard dropout rates (0.0 / 0.0 / 0.1) for each of the three models. Our results show that early dropout is effective in improving the performance of the first two models, but was not effective in the case of the larger ViT-B.

Training curves

We plot the training loss and test accuracy curves for ViT-T with early dropout and compare it with a no-dropout baseline in Figure 5.1. The early dropout is set to 50 epochs and uses a constant dropout rate. During the early dropout phase, the train loss for the dropout model is higher and the test accuracy is lower. Intriguingly, once the early dropout phase ends, the train loss decreases dramatically and the test accuracy improves to surpass the baseline.

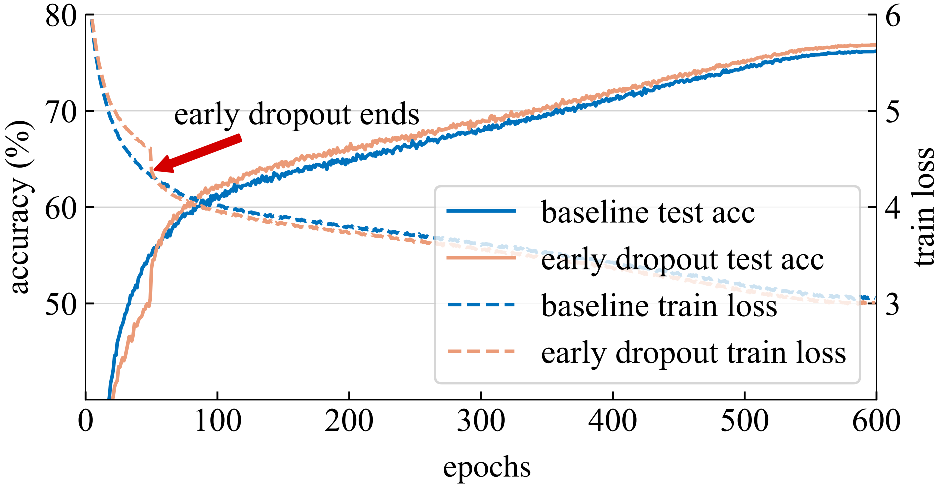


Figure 5.1: **Training Curves** When early dropout ends, the model experiences a significant decrease in training loss and a corresponding increase in test accuracy.

Dropout epochs

We investigate the impact of the number of epochs for early dropout. By default, we use 50 epochs. We vary the number of early dropout epochs and observe its effect on the final accuracy. The results, shown in Figure 5.2, are based on the average of 3 runs with different random seeds. The results indicate that the favorable range of epochs for both early dropout is quite broad, ranging from as few as 5 epochs to as many as 300, out of a total of 600 epochs. This robustness makes early dropout easy to adopt in practical settings.

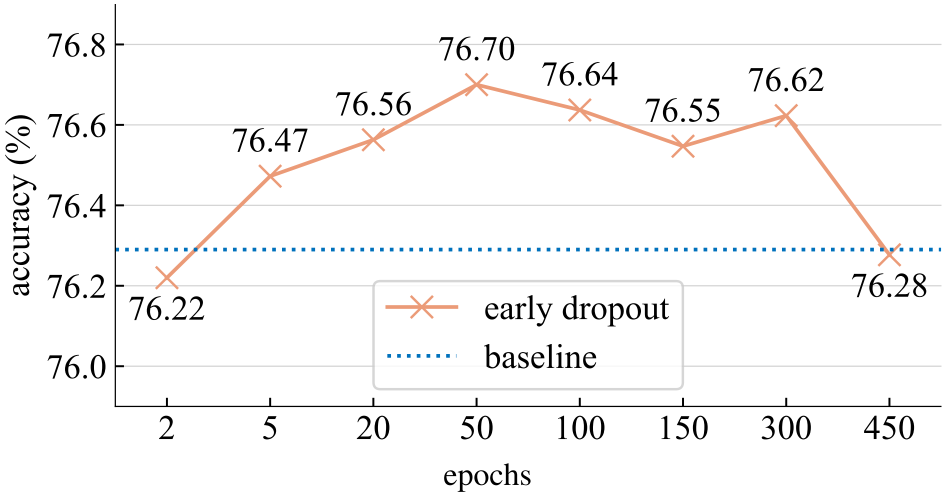


Figure 5.2: **Early dropout epochs**. Early dropout is effective with a wide range of dropout epochs.

Drop rates

The dropout rate is another hyper-parameter, similar to standard dropout. The impact of varying the rate for early dropout and early s.d. is shown in Figure 5.3. The results indicate that the performance of early s.d. is not that sensitive to the rate, but the performance of early dropout is highly dependent on it. This could be related to the fact that dropout layers are more densely inserted in ViTs than s.d. layers. In addition, the s.d. rate represents the maximum rate among layers [30], but the dropout rate represents the same rates for all layers, so the same increase in dropout rate results in a much stronger regularizing effect. Despite that, both early dropout and early s.d. are less sensitive to the rate than standard dropout, where a drop rate of 0.1 can significantly degrade accuracy (Table 5.1).

Learning rate warmup

Learning rate (lr) warmup [22, 21] is a technique that also specifically targets the early phase of training, where a smaller lr is used. We are curious in the effect lr warmup on early dropout. Our default recipe uses a 50-epoch linear lr warmup. We vary the lr warmup length from 0 to 100 and compare the accuracy with and without early dropout in Figure 5.4. Our results show that early dropout consistently improves the accuracy regardless of the use of lr warmup.

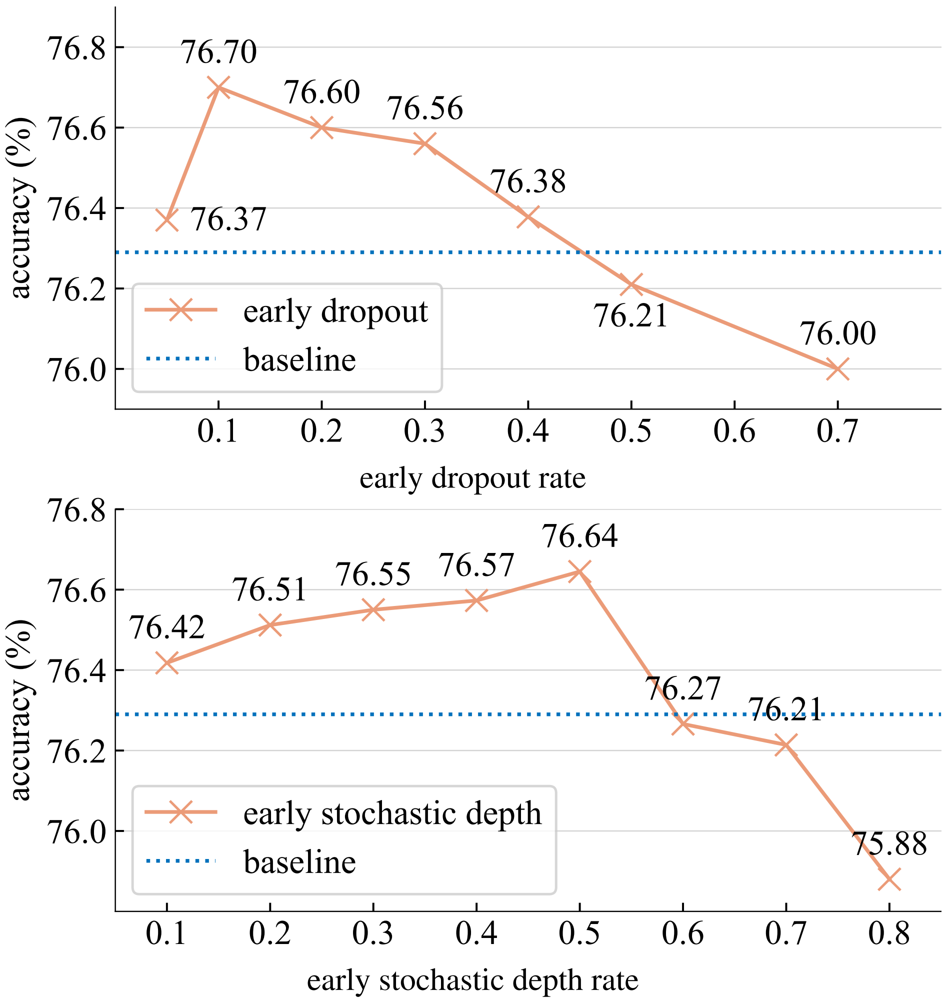


Figure 5.3: **Drop rates**. The performance of early dropout on ViT-T is affected by the dropout rate (top) but is more stable with the stochastic depth rate (bottom).

5.3 Late Dropout

Settings

To evaluate late dropout, we choose larger models, ViT-B and Mixer-B, with 59M and 86M parameters respectively, and use the basic training recipe. These models are considered to be in the overfitting regime as they already use standard s.d. We evaluate late s.d. because we find the baseline results using standard s.d. are much better than standard dropout for these models. For this experiment, we set the drop rate for late s.d. directly to their optimal drop rate for standard s.d. No s.d. is used for the first 50 epochs, and a constant s.d. rate is used for the rest of training.

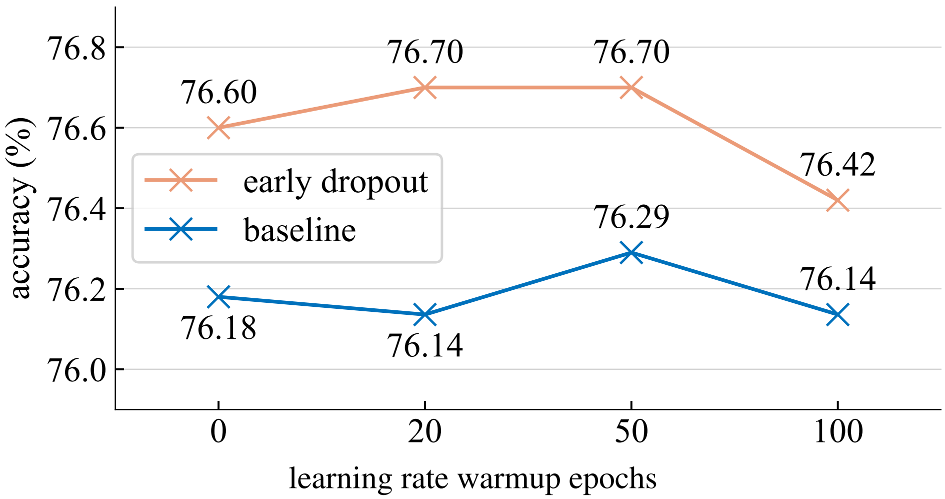


Figure 5.4: Early dropout leads to accuracy improvements even when the number of **learning rate warmup** epochs varies.

model	top-1 acc.	change	train loss	change
ViT-B (standard s.d.)*	81.8	-	-	-
ViT-B (standard s.d.)	81.6	-	2.817	-
+ no s.d.	77.0	↓4.8	2.255	↓0.562
+ linear-increasing s.d.	82.1	↑0.5	2.939	↑0.122
+ curriculum‡ s.d.	82.0	↑0.4	2.905	↑0.088
+ late s.d.	**82.3**	↑0.7	2.808	↓0.009
Mixer-B (standard s.d.)†	76.4	-	-	-
Mixer-B (standard s.d.)	78.0	-	2.810	-
+ no s.d.	76.0	↓2.0	2.468	↓0.342
+ late s.d.	**78.6**	↑0.6	2.865	↑0.055

Table 5.4: **Classification accuracy on ImageNet-1K for late s.d.** Late s.d. leads to improved test accuracy for overfitting models compared to their standard counterparts. Literature baselines: $*$[64], †[61].

Results

In the results shown in Table 5.4, late s.d. improves the test accuracy compared to standard s.d.. This improvement is achieved while either maintaining (ViT-B) or increasing (Mixer-B) the training loss, demonstrating that late s.d. effectively reduces overfitting. Previous works [45, 59, 75] have used dropout with gradually increasing strength to combat overfitting. In the case of ViT-B, we also compare our results with a linear increase and a curriculum schedule [45] with their best p over a hyperparameter sweep and find that late s.d. brings a larger improvement. Appendix B presents more detailed analysis for late s.d.

Chapter 6

Downstream Tasks

We evaluate the pre-trained ImageNet-1K models by fine-tuning them on downstream tasks. Our aim is to evaluate the learned representations without using early or late dropout during fine-tuning. Additionally, we conduct a direct evaluation of robustness benchmarks in Appendix D.

6.1 Object detection and segmentation on COCO

We fine-tune pre-trained Swin-F and ConvNeXt-F backbones with Mask-RCNN [23] on the COCO dataset. We use the 1× fine-tuning setting in MMDetection [5]. We follow the 1× fine-tuning setting in MMDetection [5]. The results are shown in Table 6.1. Models pre-trained with early dropout or s.d. consistently maintain their superiority when fine-tuned on COCO.

backbone	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
			Mask-RCNN 1× schedule			
Swin-F	36.4	58.8	38.8	34.2	55.6	36.0
+ early dropout	**37.1**	59.1	39.6	**34.6**	56.0	36.5
+ early s.d.	**36.9**	59.3	39.4	**34.5**	56.1	36.4
ConvNeXt-F	46.0	68.1	50.3	41.6	65.1	44.9
+ early s.d.	**46.2**	67.9	50.8	**41.7**	65.0	44.9

Table 6.1: **COCO object detection and segmentation results**.

6.2 Semantic segmentation on ADE20K

We fine-tune pre-trained models on the ADE-20K semantic segmentation task [74] with UperNet [70] for 80k iterations, following MMSegmentation [43]. As Table 6.2 shows, models pre-trained with our methods outperform baseline models.

method	ViT-T	ViT-B
baseline	39.2	44.3
+ early dropout	**40.0**	-
+ early s.d.	**39.8**	-
+ late s.d.	-	**45.7**

Table 6.2: **ADE20K semantic segmentation results (mIoU).**

6.3 Classification tasks

We also evaluate on several downstream classification datasets: CIFAR-100 [33], Flowers [47], Pets [48], STL-10 [8] and Food-101 [3]. Our fine-tuning procedures are based on the hyper-parameter settings from MoCo v3 [7] and SLIP [46]. Table 6.3 presents the results. Our methods show improved generalization performance on most classification tasks.

Model	C-100	Flowers	Pets	STL-10	F-101
ViT-T	87.4	96.2	92.2	97.6	89.7
+ early dropout	**87.9**	**96.4**	**93.1**	**97.8**	**89.9**
Swin-F	86.5	96.2	92.2	97.7	89.4
+ early dropout	**86.9**	**96.7**	**92.3**	**97.8**	**89.5**
ViT-B*	87.1	89.5	93.8	-	-
ViT-B†	90.5	97.7	93.2	-	-
ViT-B	90.5	97.5	**95.4**	98.5	90.6
+ late s.d.	**90.7**	**97.9**	95.3	**98.7**	**91.4**

Table 6.3: **Downstream classification accuracy** on five datasets. Literature baselines: $*$[15], †[7].

Chapter 7

Conclusion

Dropout has shined for 10 years for its excellence in tackling overfitting. In this work, we unveil its potential in aiding stochastic optimization and reducing underfitting. Our key insight is dropout counters the data randomness brought by SGD and reduces gradient variance at early training. This also results in stochastic mini-batch gradients that are more aligned with the underlying whole-dataset gradient. Motivated by this, we propose early dropout to help underfitting models fit better, and late dropout, to improve the generalization of overfitting models. We hope our discovery stimulates more research in understanding dropout and designing regularizers for gradient-based learning, and our approaches help model training with increasingly large datasets.

Bibliography

[1] Jimmy Ba and Brendan Frey. “Adaptive dropout for training deep neural networks.” In: *NeurIPS*. 2013.

[2] Pierre Baldi and Peter J Sadowski. “Understanding dropout.” In: *NeurIPS*. 2013.

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. “Food-101–mining discriminative components with random forests.” In: *EECV*. 2014.

[4] Tom Brown et al. “Language Models are Few-Shot Learners.” In: *NeurIPS*. 2020.

[5] Kai Chen et al. “MMDetection: Open MMLab Detection Toolbox and Benchmark.” In: *arXiv:1906.07155* (2019).

[6] Xiangning Chen, Cho-Jui Hsieh, and Boqing Gong. “When vision transformers outperform resnets without pre-training or strong data augmentations.” In: *ICLR*. 2022.

[7] Xinlei Chen, Saining Xie, and Kaiming He. “An empirical study of training self-supervised Vision Transformers.” In: *ICCV*. 2021.

[8] Adam Coates, Andrew Ng, and Honglak Lee. “An analysis of single-layer networks in unsupervised feature learning.” In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223.

[9] Ekin D Cubuk et al. “Randaugment: Practical automated data augmentation with a reduced search space.” In: *CVPR Workshops*. 2020.

[10] Terrance De Vries et al. “Does object recognition work for everyone?” In: *CVPR Workshops*. 2019.

[11] Jia Deng et al. “ImageNet: A large-scale hierarchical image database.” In: *CVPR*. 2009.

[12] Jacob Devlin et al. “Bert: Pre-training of deep bidirectional transformers for language understanding.” In: *arXiv preprint arXiv:1810.04805* (2018).

[13] Terrance DeVries and Graham W Taylor. “Improved regularization of convolutional neural networks with cutout.” In: *arXiv preprint arXiv:1708.04552* (2017).

[14] Alexey Dosovitskiy et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.” In: *ICLR*. 2021.

[15] Alexey Dosovitskiy et al. “Discriminative unsupervised feature learning with convolutional neural networks.” In: *NeurIPS*. 2014.

[16] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning.” In: *ICML*. 2016.

[17] Yarin Gal, Jiri Hron, and Alex Kendall. “Concrete dropout.” In: *NeurIPS* 30 (2017).

[18] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness.” In: *arXiv preprint arXiv:1811.12231* (2018).

[19] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. “Dropblock: A regularization method for convolutional networks.” In: *NeurIPS* (2018).

[20] Aidan N Gomez et al. “Learning sparse networks using targeted dropout.” In: *arXiv preprint arXiv:1905.13678* (2019).

[21] Priya Goyal et al. “Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour.” In: *arXiv:1706.02677* (2017).

[22] Kaiming He et al. “Deep Residual Learning for Image Recognition.” In: *CVPR*. 2016.

[23] Kaiming He et al. “Mask R-CNN.” In: *ICCV*. 2017.

[24] Kaiming He et al. “Masked autoencoders are scalable vision learners.” In: *arXiv:2111.06377* (2021).

[25] Kaiming He et al. “Spatial pyramid pooling in deep convolutional networks for visual recognition.” In: *ECCV*. 2014.

[26] Dan Hendrycks and Thomas Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations.” In: *ICLR*. 2018.

[27] Dan Hendrycks et al. “Natural adversarial examples.” In: *CVPR*. 2021.

[28] Dan Hendrycks et al. “The many faces of robustness: A critical analysis of out-of-distribution generalization.” In: *ICCV*. 2021.

[29] Geoffrey E Hinton et al. “Improving neural networks by preventing co-adaptation of feature detectors.” In: *arXiv:1207.0580* (2012).

[30] Gao Huang et al. “Deep networks with stochastic depth.” In: *ECCV*. 2016.

[31] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold.” In: *Nature* 596.7873 (2021), pp. 583–589.

[32] Durk P Kingma, Tim Salimans, and Max Welling. “Variational dropout and the local reparameterization trick.” In: *NeurIPS*. 2015.

[33] Alex Krizhevsky. “Learning multiple layers of features from tiny images.” In: *Tech Report* (2009).

[34] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. “Imagenet classification with deep convolutional neural networks.” In: *NeurIPS*. 2012.

[35] Krishna Kumar Singh and Yong Jae Lee. "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization." In: *ICCV*. 2017.

[36] Alex Labach, Hojjat Salehinejad, and Shahrokh Valaee. "Survey of dropout methods for deep neural networks." In: *arXiv preprint arXiv:1904.13310* (2019).

[37] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Fractalnet: Ultra-deep neural networks without residuals." In: *arXiv preprint arXiv:1605.07648* (2016).

[38] Hao Li et al. "Visualizing the loss landscape of neural nets." In: *NeurIPS*. 2018.

[39] Ze Liu et al. "Swin transformer: Hierarchical vision transformer using shifted windows." In: *ICCV*. 2021.

[40] Zhuang Liu et al. "A convnet for the 2020s." In: *CVPR*. 2022.

[41] Zhuang Liu et al. "Learning efficient convolutional networks through network slimming." In: *ICCV*. 2017.

[42] Ilya Loshchilov and Frank Hutter. "Decoupled weight decay regularization." In: *ICLR*. 2019.

[43] MMSegmentation-contributors. *MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark*. https://github.com/open-mmlab/mmsegmentation. 2020.

[44] Dmitry Molchanov, Arsenii Ashukha, and Dmitry Vetrov. "Variational dropout sparsifies deep neural networks." In: *International Conference on Machine Learning*. 2017.

[45] Pietro Morerio et al. "Curriculum dropout." In: *ICCV*. 2017.

[46] Norman Mu et al. "Slip: Self-supervision meets language-image pre-training." In: *ECCV*. 2022.

[47] Maria-Elena Nilsback and Andrew Zisserman. "Automated flower classification over a large number of classes." In: *Indian Conference on Computer Vision, Graphics & Image Processing*. 2008.

[48] Omkar M Parkhi et al. "Cats and dogs." In: *CVPR*. 2012.

[49] Boris T Polyak and Anatoli B Juditsky. "Acceleration of stochastic approximation by averaging." In: *SIAM Journal on Control and Optimization* (1992).

[50] Alec Radford et al. "Learning transferable visual models from natural language supervision." In: (2021).

[51] Aditya Ramesh et al. "Hierarchical text-conditional image generation with clip latents." In: *arXiv preprint arXiv:2204.06125* (2022).

[52] Benjamin Recht et al. "Do imagenet classifiers generalize to imagenet?" In: *ICML*. 2019.

[53] Steven J Rennie, Vaibhava Goel, and Samuel Thomas. "Annealed dropout training of deep networks." In: *2014 IEEE Spoken Language Technology Workshop (SLT)*. IEEE. 2014, pp. 159–164.

[54] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *ICLR*. 2015.

[55] Nitish Srivastava et al. "Dropout: A simple way to prevent neural networks from overfitting." In: *The Journal of Machine Learning Research* (2014), pp. 1929–1958.

[56] Andreas Steiner et al. "How to train your vit? data, augmentation, and regularization in vision transformers." In: *arXiv preprint arXiv:2106.10270* (2021).

[57] Christian Szegedy et al. "Going deeper with convolutions." In: *CVPR*. 2015.

[58] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision." In: *CVPR*. 2016.

[59] Mingxing Tan and Quoc Le. "Efficientnetv2: Smaller models and faster training." In: *ICML*. 2021.

[60] Robert Tibshirani. "Regression shrinkage and selection via the lasso." In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288.

[61] Ilya O Tolstikhin et al. "Mlp-mixer: An all-mlp architecture for vision." In: *NeurIPS*. 2021.

[62] Jonathan Tompson et al. "Efficient object localization using convolutional networks." In: *CVPR*. 2015.

[63] Hugo Touvron et al. "Going deeper with Image Transformers." In: *ICCV* (2021).

[64] Hugo Touvron et al. "Training data-efficient image transformers & distillation through attention." In: *arXiv:2012.12877* (2020).

[65] Ashish Vaswani et al. "Attention Is All You Need." In: *NeurIPS*. 2017.

[66] Li Wan et al. "Regularization of neural networks using dropconnect." In: *ICML*. 2013.

[67] Haohan Wang et al. "Learning robust global representations by penalizing local predictive power." In: *NeurIPS*. 2019.

[68] Sida Wang and Christopher Manning. "Fast dropout training." In: *ICML*. 2013.

[69] Ross Wightman. *GitHub repository: PyTorch Image Models*. `https://github.com/rwightman/pytorch-image-models`. 2019.

[70] Tete Xiao et al. "Unified perceptual parsing for scene understanding." In: *ECCV*. 2018.

[71] Sangdoo Yun et al. "Cutmix: Regularization strategy to train strong classifiers with localizable features." In: *ICCV*. 2019.

[72] Hongyi Zhang et al. "mixup: Beyond empirical risk minimization." In: *ICLR*. 2018.

[73] Zhun Zhong et al. "Random erasing data augmentation." In: *AAAI*. 2020.

[74] Bolei Zhou et al. "Semantic understanding of scenes through the ADE20K dataset." In: *IJCV* (2019).

[75] Barret Zoph et al. "Learning transferable architectures for scalable image recognition." In: *CVPR*. 2018.

Appendix A

Experimental Settings

A.1 Training recipe

We provide our basic training recipe with specific details in Table A.3. This recipe is based on the setting in ConvNeXt [40]. For the improved recipe, we increase the number of epochs to 600, and reduce mixup and cutmix to 0.3. All other configurations remain unchanged.

A.2 Drop rates

The drop rates for early dropout and early s.d. are listed in Table A.1. By default, the early dropout epochs are set to 50, with a linear decreasing schedule. A light search of early dropout rates was conducted from the values {0.1, 0.2, 0.3}. For Swin-F, we find including an additional range {0.5, 0.7} is useful. For early s.d. rate, we search from {0.3, 0.5, 0.7} for all models. The baselines do not use any dropout or s.d. The compared standard dropout / s.d. experiments all use a low drop rate of 0.1.

The late s.d. drop rates are listed in Table A.2. The basic training recipe is adopted. The baselines use standard s.d., whose rates are obtained with hyper-parameter sweeps. We find using the same rates for late s.d. proves to be effective.

model	early dropout rate	early s.d. rate
	with basic recipe	
ViT-T	0.1	0.5
Mixer-S	0.1	0.7
ConvNeXt-F	-	0.5
Swin-F	0.5	0.5
	with improved recipe	
ViT-T	0.1	0.7
ConvNeXt-F	-	0.5
Swin-F	0.7	0.5

Table A.1: Early dropout and early s.d. rates used in experiments.

model	standard s.d. rate	early s.d. rate
	with basic recipe	
ViT-B	0.4	0.4
Mixer-B	0.2	0.2

Table A.2: Late s.d. rates and standard s.d. rates used in experiments.

Training Setting	Configuration
weight init	trunc. normal (0.2)
optimizer	AdamW
base learning rate	4e-3
weight decay	0.05
optimizer momentum	β_1, β_2=$0.9, 0.999$
batch size	4096
training epochs	300
learning rate schedule	cosine decay
warmup epochs	50
warmup schedule	linear
stochastic depth rate [30]	0.0
dropout rate [29]	0.0
randaugment [9]	(9, 0.5)
mixup [72]	0.8
cutmix [71]	1.0
random erasing [73]	0.25
label smoothing [58]	0.1
layer scale [63]	1e-6
head init scale [63]	None
gradient clip	None
exp. mov. avg. (EMA) [49]	0.9999

Table A.3: Our basic training recipe, adapted from ConvNeXt [40].

Appendix B

Analaysis for Late Dropout

B.1 Training curves

We present the training curves for late s.d. in Figure B.1, comparing it with the baseline (standard s.d. with the best drop rate). When late s.d. begins, the training loss immediately increases. However, the final test accuracy of the late s.d. model is higher than the baseline and so is the training loss, demonstrating the effectiveness of late s.d. in reducing overfitting and closing the generalization gap.

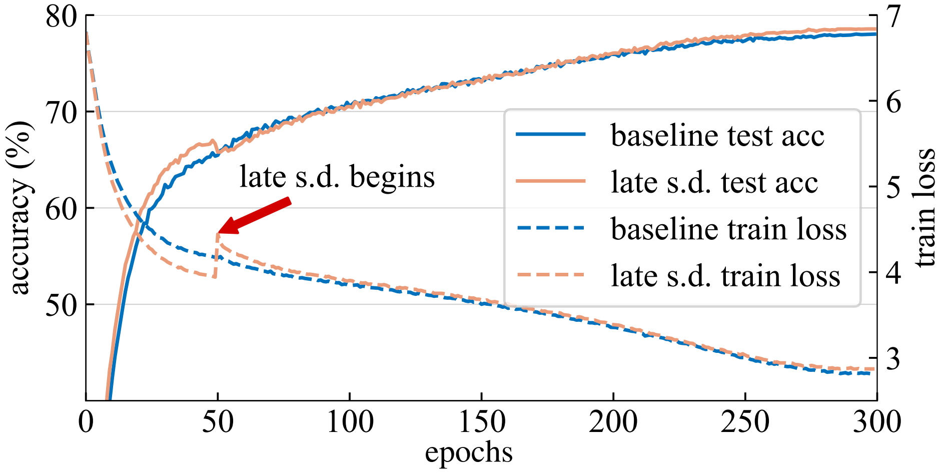


Figure B.1: **Training Curves.** When late s.d. begins, the model experiences a jump in training loss and a decrease in test accuracy.

B.2 Drop rates

We examine the impact of the drop rate for late s.d. As the models are in an overfitting regime, we also plot the results using different standard s.d. rates as baselines. In Figure B.2, we observe that late s.d. is less sensitive to changes in the drop rate and, overall, leads to improved generalization results.

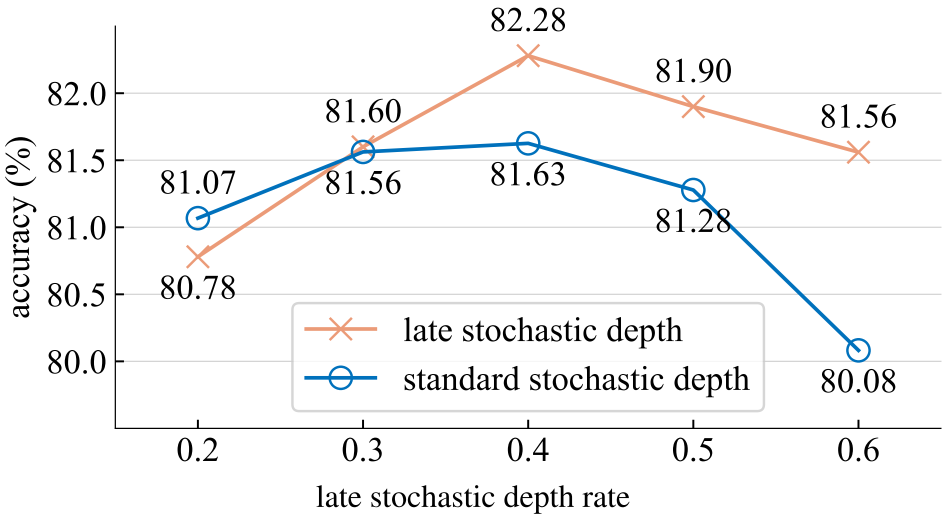


Figure B.2: **Late s.d. drop rates**. Late s.d. improves over standard s.d. for a broad range of drop rates.

B.3 Dropout epochs

Similarly, we analyze the effect of different late s.d. epochs in Figure B.3. The epoch refers to the point where s.d. begins. Overall, the improvement from late s.d. remains consistent when the start epoch varies from 5 to 100, with a peak observed at 50. The optimal epoch for late s.d. may vary based on the chosen drop rate.

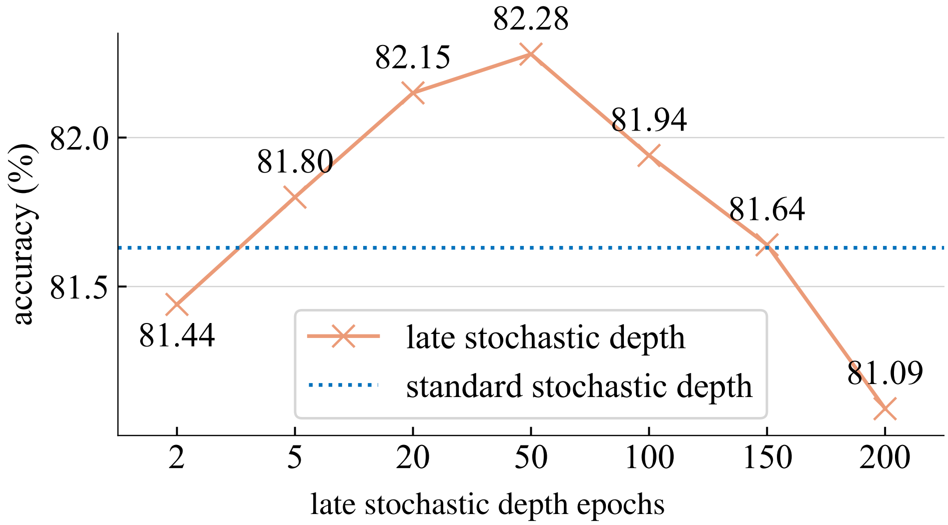


Figure B.3: **Late s.d. epochs**. The optimal epoch for late s.d. in this experiment is 50.

B.4 Other architectures

We attempted to use late s.d. on ConvNeXt-B and Swin-B, but were unable to find a set of hyper-parameters that resulted in a significant improvement over standard s.d.The differing results compared to those obtained with ViT-B and Mixer-B could be attributed to the inductive biases present in these architectures. Further investigation is needed to determine why late s.d. may not be suitable for certain architectures.

Appendix C

Constant Early Dropout

The majority of experiments described in paper use a linear decreasing schedule for early dropout. We now switch to a constant schedule, where the early dropout phase uses a constant drop rate, and then turned off to 0 when it ends. This is also discussed in Table 5.3b's experiments.

We find it beneficial to shorten the dropout epochs from 50 to 20. This is perhaps because the "accumulated" drop rate (calculated as the area under the curve on a drop rate vs. epoch plot) plays an important role, and constant schedule accumulates twice as much as the linear schedule if they both start at the same rate p and end at the same epoch.

We present the results in Table C.1. Constant early dropout consistently improves both training loss and test accuracy upon the baseline. This further demonstrates that early dropout is not limited to a linearly decreasing schedule to effectively reduce underfitting.

model	top-1 acc.	change	train loss	change
results with basic recipe				
ViT-T	73.9	-	3.443	-
+ early dropout	**74.4**	↑0.5	3.408	↓0.035
+ early s.d.	**74.0**	↑0.1	3.428	↓0.015
Mixer-S∗	68.7	-	-	-
Mixer-S	71.0	-	3.635	-
+ early dropout	**71.4**	↑0.4	3.572	↓0.063
+ early s.d.	**71.6**	↑0.6	3.553	↓0.082
ConvNeXt-F	76.1	-	3.472	-
+ early s.d.	**76.5**	↑0.4	3.449	↓0.023
Swin-F	74.3	-	3.411	-
+ early dropout	**74.6**	↑0.3	3.382	↓0.029
+ early s.d.	**75.1**	↑0.8	3.355	↓0.056
results with improved recipe				
ViT-T†	72.8	-	-	-
ViT-T‡	75.5	-	-	-
ViT-T	76.3	-	3.033	-
+ early dropout	**76.7**	↑0.4	2.994	↓0.043
+ early s.d.	**76.7**	↑0.4	3.008	↓0.025
ConvNeXt-F‡	77.5	-	-	-
ConvNeXt-F	77.5	-	3.011	-
+ early s.d.	**77.6**	↑0.1	2.989	↓0.022
Swin-F	76.1	-	2.989	-
+ early dropout	**76.4**	↑0.3	2.972	↓0.017
+ early s.d.	**76.8**	↑0.7	2.974	↓0.015

Table C.1: **Classification accuracy on ImageNet-1K with early dropout using a constant schedule.** We obtain consistent improvement with results similar to those obtained using a linear schedule. Literature baselines: ∗tolstikhin2021mlp, †[64], ‡[69].

Appendix D

Robustness Evaluation

We evaluate the models on common robustness benchmarks, which test their accuracy when the input images experience a change in distribution, such as corruption or style change. We report top-1 accuracy on ImageNet-A [27], ImageNet-R [28], ImageNet-Sketch [67], ImageNet-V2 [52], Stylized ImageNet [18], and mean Corruption Error (mCE) on ImageNet-C [26]. Table D.1 shows that the improvement is transferable across different conditions.

Model	Clean	A	R	SK	V2	Style	C (↓)
ViT-T	76.3	10.2	36.3	24.2	63.7	12.3	65.4
+ early dropout	**76.7**	**11.6**	**37.3**	**24.7**	**65.0**	**13.0**	**64.2**
+ early s.d.	**76.7**	**10.0**	**36.8**	**24.8**	**64.2**	**12.8**	**63.6**
Mixer-S	71.0	4.1	35.4	23.0	56.8	13.0	67.7
+ early dropout	**71.3**	**4.2**	**35.9**	**23.5**	**58.2**	**13.5**	**66.3**
+ early s.d.	**71.7**	**4.5**	**37.1**	**24.8**	**57.8**	**14.2**	**65.6**
ViT-B	81.6	25.9	47.0	33.3	70.2	19.8	49.1
+ late s.d.	**82.3**	**27.3**	**48.3**	**35.0**	**71.2**	**21.1**	**47.4**

Table D.1: **Robustness evaluation.** The accuracy gain achieved with our methods is consistent across various distributional shifts.

Appendix E

Loss Landscape

We visualize the loss landscape [38] of ViT-T models trained with and without early dropout in Figure E.1. From the figure, we do not observe any significant difference in flatness around the solution area. To quantitatively measure the curvature, we calculate δ, the average difference in loss values between neighboring points:

$$\delta = \frac{1}{|N|} \sum_{(p_i,p_j)\in N} |L(p_i) - L(p_j)|$$

where N is the set of all neighboring pairs of points on the loss landscape, and $L(\cdot)$ denotes the loss value at a given point. Smaller δ indicates a flatter landscape. We notice a very slight difference in δ, with 0.250 for early dropout and 0.258 for baseline. This suggests that early dropout may not improve generalization by finding flatter regions, unlike other methods such as [38, 6].

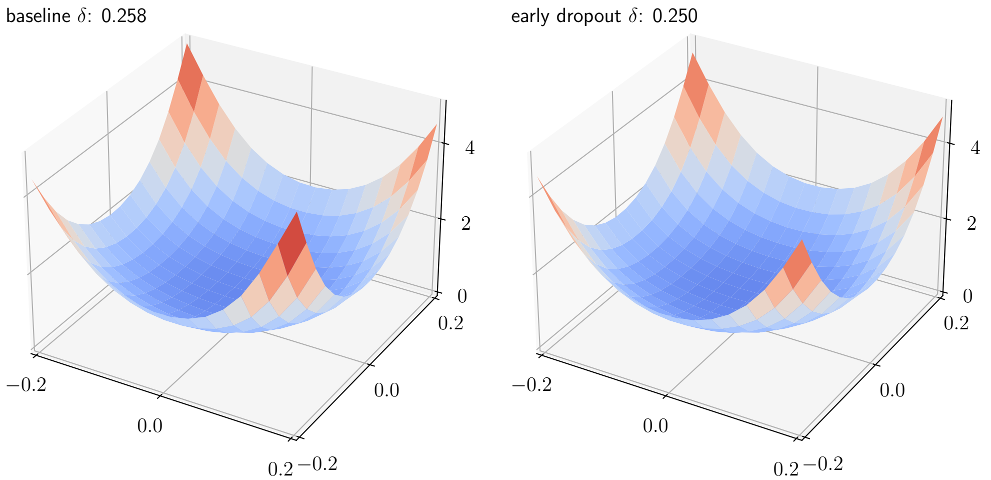


Figure E.1: **Loss Landscape Visualization** [38] comparing the baseline (left) and early dropout (right) models. Both models show similar levels of flatness both visually and when measured with the curvature metric δ.

Appendix F

Limitations

We show that early and late dropout can benefit the training of small and large networks in a range of supervised visual recognition tasks. However, the application of deep learning extends far beyond this, and further research is needed to determine the impact of early and late dropout on other areas, such as self-supervised pre-training or natural language processing. It would also be valuable to explore the interplay between early / late dropout and other factors such as training duration or optimizer choice.

Appendix G

Broader Impacts

The training and inference of deep neural networks can take an excessive amount of energy, especially in the large model and large data era. Our discovery on early dropout could spark more interest in developing training techniques for small models, which have far lower total energy usage and carbon emission than large models.

It is also important to note that the benchmark datasets used in this study were primarily designed for research purposes, and may not accurately reflect the real-world distributions and biases present in data [10]. Further research is needed to address these biases and develop training techniques that are robust to real-world data variability.