Last-SAM

Daniel Zhang
Vanderbilt University
daniel.zhang.1@vanderbilt.edu

Abstract

Current deep learning models often rely on overparameterization, making the training loss an insufficient indicator of generalization performance. Sharpness-Aware Minimization (SAM) and Adaptive Sharpness-Aware Minimization (ASAM) have emerged as promising approaches to improve generalization by simultaneously minimizing loss value and loss sharpness. However, SAM's effectiveness comes at the cost of increased complexity and computational cost throughout the training process. In this work, we propose LastSAM, a variant of SAM that applies sharpness-aware minimization only after a model has been trained using standard optimization techniques. We hypothesize that this approach will allow us to find minima that perform well on the training data while being robust to parameter perturbations, leading to improved test-time performance. LastSAM aims to be more efficient than SAM by decoupling the sharpness optimization from the regular training process and potentially mitigating convergence instability associated with SAM. We evaluate LastSAM on benchmark datasets such as CIFAR-10, and compare its performance to SAM and other state-of-the-art methods. Additionally, we investigate the impact of LastSAM on reducing the need for extensive hyperparameter tuning. We find that LastSAM improves generalization performance similarly to ASAM/SAM and helps promote more stable convergence compared to SAM. This work explores the potential of post-training sharpness-aware minimization as a computationally efficient approach to enhancing model generalization. LastSAM Repo.

1 Introduction

In recent years, deep learning models have achieved remarkable success in various domains, often relying on overparameterization to improve performance. While this approach has led to state-of-the-art results, it has also raised questions about the relationship between training loss and generalization ability. Research has shown that models can converge stably to global optima on training data in overparameterized regimes (1), and that optimizer choices can bias models towards flat minima, which helps explain good test-time performance (2).

To further improve generalization performance, several techniques have been applied. Sharpness-Aware Minimization (SAM) (3) simultaneously minimizes loss value and loss sharpness to find flat minima, while Adaptive SAM (ASAM) (4) adapts the sharpness-aware optimization to the loss landscape geometry for scale-invariant learning. Data augmentation (5) increases the diversity of training data to improve generalization, and batch normalization (6) reduces internal covariate shift to accelerate training and improve generalization. Dropout (7) randomly drops out neurons during training to prevent overfitting and improve generalization, and label smoothing (8) softens the ground-truth labels to prevent overconfident predictions and improve generalization. Mixup (9) trains on convex combinations of input-label pairs to enhance robustness and generalization, while Cutout (10) randomly masks out square regions of input data to improve robustness and generalization. Cyclic learning rates (11) vary the learning rate cyclically to escape sharp minima and find flatter

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

minima, and Stochastic Weight Averaging (SWA) (12) averages model weights along the optimization trajectory to find flatter minima.

Sharpness-Aware Minimization (SAM) and its adaptive variant, Adaptive SAM (ASAM), have shown promising results in improving the generalization performance of deep learning models. These techniques are motivated by the observation that flatter minima in the loss landscape tend to yield better generalization compared to sharp minima. By simultaneously minimizing the loss value and the loss sharpness, SAM encourages the model to converge towards flatter regions of the loss landscape (2). ASAM further enhances this approach by adapting the sharpness-aware optimization to the geometry of the loss landscape, enabling scale-invariant learning (4). The effectiveness of these methods lies in their ability to find solutions that are robust to small perturbations in the input space, thereby reducing overfitting and improving the model's ability to generalize to unseen data. However, some drawbacks to using SAM/ASAM in practice are that they add complexity (e.g., tuning the hyperparameter ρ for stable convergence), and computational expense (i.e., two forward-backward passes per step) (2).

To address these limitations, we propose LastSAM, an approach that applies sharpness-aware minimization only after a model has been trained using standard optimization techniques. This will be much less computationally expensive, as we avoid having to do two forwards-backwards passes for most of training. By decoupling flatness optimization from regular training, LastSAM may promote more stable convergence by staying in already-found minima. We believe optimizing for flatness at the end of training should be powerful enough to improve generalization since the loss landscape of overparameterized networks contain manifolds of global minima (1), which should allow us to find flat areas without dramatically increasing training loss. Premature optimization for flatness, as employed by ASAM and SAM, may not necessarily lead to convergence towards flatter minima. Moreover, this early emphasis on flatness could potentially cause the model to become trapped in suboptimal minima, as the gradient may lack sufficient informativeness at this stage.

Despite its potential benefits, LastSAM also presents several risks. It is possible that the optima found after normal training are not large enough to find good "flat spots," resulting in no improvement. Additionally, there is a risk that applying SAM/ASAM could cause the model to leave the already found minima and simply retrain entirely, which is more computationally expensive as other methods. Lastly, LastSAM may be very sensitive to the choice of ρ, similar to other SAM-based methods.

To evaluate the effectiveness of LastSAM, we will investigate several key questions. First, we will examine whether LastSAM leads to flatter minima compared to SGD, SAM, and ASAM, and if it improves generalization performance. Second, we will assess the computational cost of LastSAM in comparison to other methods. Finally, we will study the convergence stability of LastSAM and its sensitivity to hyperparameters such as ρ and the number of epochs, comparing it to other methods. In the following sections, we will provide a detailed description of the LastSAM methodology, present our experimental setup and results, and discuss the implications of our findings for improving generalization performance in deep learning models.

2 LastSAM

Given a training dataset $\mathcal{D} = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, where each example (x_i, y_i) is drawn independently and identically distributed (i.i.d.) from some distribution $\mathcal{D}$, our objective is to learn a model f_θ parameterized by θ that generalizes well to the underlying population distribution $\mathcal{D}$, i.e., minimizes the expected risk:

$$\min_{\theta} \mathbb{E}(x, y) \sim \mathcal{D}[\mathcal{L}(f\theta(x), y)], \tag{1}$$

where $\mathcal{L}$ is a suitable loss function measuring the discrepancy between the model's predictions $f_\theta(x)$ and the true labels y. SAM (Sharpness-Aware Minimization) and ASAM (Adaptive Sharpness-Aware Minimization) are optimization techniques that aim to find flat minima of the loss landscape, as flatter minima are believed to correspond to better generalization performance. The motivation behind these methods is that flatter minima provide a bound on the population risk, making the model less sensitive to perturbations in the input space. SAM's approach to finding flat minima involves minimizing the sharpness of the loss function around the current weights θ while simultaneously minimizing the loss value. This is achieved by adding a regularization term to the loss function that measures the

maximum loss within a neighborhood of θ. The SAM objective can be written as:

$$\min_{\theta} \mathcal{L}(\theta) + \lambda \max_{|\epsilon| \leq \rho} \mathcal{L}(\theta + \epsilon), \tag{2}$$

where $\mathcal{L}$ is the loss function, λ is a hyperparameter controlling the trade-off between the loss value and the sharpness term, ρ is the neighborhood size, and ϵ is a small perturbation to the weights (2). ASAM extends SAM by adapting the neighborhood size ρ to the local geometry of the loss landscape (3).

By incorporating these sharpness-aware regularization terms into the optimization process, SAM and ASAM aim to find flatter minima that generalize better to the underlying population distribution, thereby improving the model's performance on unseen data.

In our approach to improving generalization performance, we focus on optimizing the model normally first and then optimizing for flatness in the loss landscape. This strategy allows us to find a good minimum in terms of training loss before searching for flatter regions around that minimum, which may lead to better generalization. We propose and evaluate two methods that build upon this general approach:

1. Post-training SAM/ASAM: After optimizing the model using standard techniques (e.g., SGD or Adam), we apply SAM or ASAM to further optimize the model towards flatter minima. This approach leverages the existing SAM and ASAM algorithms but applies them only after the model has converged to a good minimum in terms of training loss.

2. Orthogonal SAM/ASAM: To prevent the model from leaving the optima found during normal training, we modify the SAM and ASAM algorithms to make the sharpness-aware gradient updates orthogonal to the normal gradient. This is achieved by projecting the sharpness-aware gradient onto the orthogonal complement of the normal gradient. By doing so, we ensure that the updates for flatness do not interfere with the updates for minimizing the training loss, allowing the model to stay in the vicinity of the found optima while searching for flatter regions.

We found that the first approach, post-training with vanilla ASAM, is the most effective in improving generalization performance. In the following sections, we will present our experimental results for all the proposed approaches and discuss the potential reasons behind the superiority of the post-training SAM/ASAM method.

By comparing these approaches to vanilla SAM, ASAM, and standard optimization techniques, we aim to investigate the effectiveness of decoupling the optimization of training loss and flatness. We hypothesize that by first finding a good minimum and then searching for flatter regions around it, we can improve the model's generalization performance while maintaining computational efficiency and convergence stability.

In the following sections,we will then present our experimental setup, results, and analysis, comparing the performance of these methods to SAM and ASAM on image classification tasks.

3 Experiments and Evaluation: Image Classification, CIFAR-10

In this section, we present the empirical evaluation of our proposed methods for improving generalization performance by optimizing for flatness after normal training. We assess the effectiveness of our approaches on image classification tasks and compare their performance to vanilla SAM, ASAM, and standard optimization techniques.

3.1 Experimental Setup

We evaluate the performance of different models and optimization techniques on the CIFAR-10. We normalize the input images using dataset-specific mean and standard deviation values. For CIFAR-10, we use a mean of (0.4914, 0.4822, 0.4465) and a standard deviation of (0.2023, 0.1994, 0.2010). These normalization values are commonly used for these datasets and help to center and scale the pixel values. We set the batch size to 128, which provides a good balance between computational efficiency and convergence stability (3).

To enhance the model's performance and generalization ability, we employ several techniques during training. We apply data augmentation to the training set, including random cropping with a padding

of 4 pixels and random horizontal flipping. These augmentations help to increase the diversity of the training samples and improve the model's robustness to variations in the input. We also use label smoothing with a smoothing factor of 0.1, which helps to prevent the model from making overconfident predictions and encourages it to learn more robust features. We perform optimization using Stochastic Gradient Descent (SGD) with a momentum of 0.9 and a weight decay of 5e-4. The momentum term helps to accelerate convergence and dampen oscillations, while the weight decay regularizes the model parameters to prevent overfitting. We set the initial learning rate to 0.1 and adjust it using a cosine annealing schedule over the course of training (3).

With the setup described, we report the test accuracies obtained by SAM, ASAM, and LastSAM for the following networks:

1. Wide ResNet-28-10 (wrn28_10)
2. ResNet-20 (resnet20)

For each network architecture, we train the models using the standard training procedure until convergence. We then apply SAM, ASAM, and our proposed LastSAM method to further optimize the models for flatness in the loss landscape. We compare the performance of these methods in terms of the test accuracy achieved.

To ensure a fair comparison, we follow the hyperparameter settings reported in the original SAM and ASAM papers: we adopt the default learning rate and batch size as provided in the ASAM paper. We also conduct a hyperparameter search for the best values of ρ for both SAM and ASAM from $\{0.1, 0.2, ..., 0.5, 1, 2, 5\}$ and use the value with the highest validation accuracy. The models are trained for 200 epochs, and we evaluate the test accuracy after each epoch, reporting the best performance achieved during the entire training process. Later, we separately train models with different values of ρ for SAM, ASAM, and LastSAM, then evaluate them on the test set to evaluate robustness to choice of ρ.

To determine the optimal hyperparameters for our proposed LastSAM method, we conducted a grid search over two key factors: the number of epochs allocated to SAM/ASAM and SGD, and the value of ρ for the SAM/ASAM steps.

First, we fixed ρ at 0.5 for ASAM and 0.05 for SAM and explored various combinations of epochs for SGD and SAM/ASAM. After evaluating the validation accuracy of each combination, we found that running SGD for 100 epochs followed by 50 epochs of SAM/ASAM yielded the best performance.

Next, we performed a grid search over a range of values for ρ, specifically $\{0.1, 0.2, ..., 0.5, 1, 2, 5\}$. For each value of ρ, we trained the model using the previously determined epoch split (100 epochs of SGD and 50 epochs of SAM/ASAM) and evaluated its validation accuracy. The value of ρ that achieved the highest validation accuracy was selected as the optimal choice for our LastSAM method.

Furthermore, we present a plot of the test accuracy as a function of the key hyperparameters for each method, showcasing the robustness of our proposed LastSAM approach to variations in these settings.

Finally, we discuss the computational efficiency of our method compared to SAM and ASAM, highlighting the trade-offs between performance improvement and the additional computational cost incurred by optimizing for flatness.

By providing a comprehensive evaluation across multiple network architectures, analyzing the flatness of the discovered minima, and considering the sensitivity to hyperparameters, we aim to demonstrate the effectiveness and practicality of our proposed LastSAM method for improving generalization performance in deep learning models.

3.2 Results

CIFAR-10: We evaluate our methods on the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. We use 50,000 images for training and 10,000 for testing. We train the models from scratch.

The results presented in the two tables demonstrate the effectiveness of the proposed LastSAM method in improving the generalization performance of deep learning models on the CIFAR-10 dataset. The key observation is that LastSAM, using either SAM or ASAM, achieves slightly better

Table 1: Comparison of test accuracies on Cifar10 dataset using ResNet20 architecture.

Method	Test Accuracy (%)
Default SAM	93.18
ASAM	93.69
lastSAM (using ASAM)	93.79
lastSAM (using SAM)	93.34

Table 2: Comparison of test accuracies on Cifar10 dataset using Resnet20 architecture.

Method	Test Accuracy (%)
Default SAM	96.41
ASAM	96.54
lastSAM (using ASAM)	96.98
lastSAM (using SAM)	96.13

or comparable test accuracies compared to the default SAM and ASAM methods, while requiring fewer computational resources during training.

For the ResNet20 architecture (Table 1), LastSAM using ASAM achieves the highest test accuracy of 93.79%, slightly outperforming both default SAM (93.18%) and ASAM (93.69%). Similarly, for the WRN-28-10 architecture (Table 2), LastSAM using ASAM obtains the best test accuracy of 96.98%, surpassing default SAM (96.41%) and ASAM (96.54%). LastSAM either improves or achieves comparable performance for both SAM and ASAM.

The key advantage of LastSAM lies in its computational efficiency. While default SAM and ASAM require 200 epochs of training, each involving two backward-forward passes to compute the SAM/ASAM updates, LastSAM achieves its superior performance in just 150 total epochs. Importantly, out of these 150 epochs, 100 are standard SGD epochs, which only require one backward-forward pass. This means that LastSAM effectively optimizes for flatness in the loss landscape using SAM/ASAM updates for only the last 50 epochs of training.

The significance of this finding is twofold. First, it demonstrates that the benefits of SAM and ASAM in improving generalization can be largely realized by applying these techniques only in the later stages of training. This suggests that the model's parameters are already in a reasonably good region of the loss landscape after the initial SGD training, and the SAM/ASAM updates help to fine-tune the model towards flatter minima.

Second, and more importantly, LastSAM's approach of using standard SGD for the majority of training and applying SAM/ASAM only in the last few epochs offers a more computationally efficient way to improve generalization. By reducing the number of SAM/ASAM updates required, LastSAM can significantly lower the computational burden associated with these techniques, making them more practical for larger-scale datasets and models.

The results on CIFAR-10 using ResNet20 and WRN-28-10 architectures provide strong evidence that LastSAM is an effective and efficient method for improving generalization in deep learning models. By leveraging the power of SAM and ASAM in the later stages of training, LastSAM achieves state-of-the-art performance while minimizing the computational overhead. This finding opens up new possibilities for applying SAM and ASAM to a wider range of datasets and models, as the computational cost is no longer a limiting factor.

3.2.1 Hyperparameter Sensitivity and Convergence Stability

To investigate the robustness of LastSAM to hyperparameter choices, particularly the neighborhood size ρ, we conduct a series of experiments on the CIFAR-10 dataset using the Resnet20 architecture. We compare the performance of LastSAM with both ASAM and SAM optimizers, as well as the baseline ASAM and SAM methods.

For LastSAM with ASAM, we first train the model using SGD for 100 epochs and then apply ASAM for the remaining epochs. We vary the value of ρ in the range [0.1, 0.2, 0.3, 0.4, 0.5, 1, 2, 5] and

Table 3: Comparison of test accuracies on Cifar10 dataset using WRN-28-10 architecture with different ρ values.

Method	Test Accuracy (%) for different ρ values							
	0.1	0.2	0.3	0.4	0.5	1	2	5
LastSAM (using ASAM)	93.09	**93.34**	**93.37**	**93.71**	**93.79**	93.38	**93.01**	**89.41**
LastSAM (using SAM)	**93.34**	93.20	92.69	92.19	91.68	88.55	83.24	57.15
ASAM	92.18	92.25	92.98	92.46	93.69	**93.52**	92.99	86.37
SAM	92.49	93.18	92.57	91.49	90.41	84.01	10.16	10.02

**For the three red-colored SAM failure runs, we took 5 runs and averaged the accuracies.*

report the test accuracy achieved. Similarly, for LastSAM with SAM, we train the model using SGD for 100 epochs and then apply SAM with different values of ρ in the same range.

Our results show that LastSAM with ASAM maintains high performance across a wide range of ρ values, with the test accuracy of 93.79% obtained at $\rho = 0.5$. Even for larger values of ρ such as 1 and 2, the test accuracy remains above 93%. This indicates that LastSAM with ASAM is relatively robust to the choice of ρ and maintains good convergence stability. On the other hand, LastSAM with SAM exhibits a higher sensitivity to the value of ρ. The test accuracy of 93.34% is achieved at $\rho = 0.1$, but the performance degrades as ρ increases. For $\rho \geq 0.3$, the test accuracy drops below 93%, and for larger values of ρ such as 1, 2, and 5, the performance significantly deteriorates. To better understand these observations, we compare the results of LastSAM with the baseline ASAM and SAM methods. For ASAM, the test accuracy of 93.69% is obtained at $\rho = 0.5$, which is consistent with the original ASAM paper. The performance of ASAM remains relatively stable across different values of ρ, with test accuracies above 92% for most settings. This suggests that ASAM is inherently robust to the choice of ρ.

In contrast, SAM exhibits a higher sensitivity to ρ, similar to the behavior observed in LastSAM with SAM. The best test accuracy of 93.18% is achieved at $\rho = 0.2$, but the performance drops significantly for larger values of ρ. When $\rho \geq 1$, the test accuracy falls below 85%, indicating a severe degradation in performance. There are failure runs where SAM fails to converge to a good optimum for the two largest values of ρ.

These results suggest that LastSAM with ASAM improves convergence stability compared to SAM and maintains the robustness of ASAM to hyperparameter choices. This can be attributed to the fact that SGD already guides the model towards good optima, and the subsequent application of ASAM helps to refine the solution while maintaining stability.

On the other hand, the sensitivity of LastSAM with SAM to ρ indicates that the choice of neighborhood size is crucial for the success of the SAM optimizer. When ρ is too large, SAM may lead the model towards flat regions of the loss landscape where the gradient information is insufficient for effective optimization, resulting in poor convergence and suboptimal solutions. However, doing SGD beforehand lowers the likelihood that we end up inthese bad minima.

In summary, our experiments demonstrate that LastSAM with ASAM offers improved convergence stability and robustness to hyperparameter choices compared to SAM. The combination of SGD and ASAM in LastSAM allows for stable convergence to good optima, while the choice of ρ in SAM is more critical for achieving optimal performance. These findings highlight the potential benefits of using LastSAM with ASAM for improving generalization performance without much added hyperparameter tuning complexity in deep learning models.

3.3 Orthogonalization Experiments

Motivated by the idea that SGD has already guided the model to a good minimum, we explore the concept of orthogonalizing the SAM/ASAM gradients with respect to the normal SGD gradients. The rationale behind this approach is to prevent the model from leaving the good minimum found by SGD and instead focus on searching for flatter regions within the local neighborhood of the minimum.

To achieve this, we modify the SAM/ASAM optimization process as follows. During the first backward pass of the two forward-backward passes in SAM/ASAM, we save the gradients computed

with respect to the model parameters. Let g_{SGD} denote these gradients. In the second backward pass, after computing the SAM/ASAM gradients $g_{SAM/ASAM}$, we orthogonalize them with respect to g_{SGD} by subtracting the projection of $g_{SAM/ASAM}$ onto g_{SGD}:

$$g_{ortho} = g_{SAM/ASAM} - \frac{g_{SAM/ASAM} \cdot g_{SGD}}{\|g_{SGD}\|^2} g_{SGD} \tag{3}$$

where $\cdot$ denotes the dot product and $\|\cdot\|$ represents the Euclidean norm. The resulting orthogonalized gradients g_{ortho} are then used to update the model parameters in the descent step.

Despite the theoretical appeal of this approach, our experiments yield disappointing results. We observe a significant degradation in performance compared to the standard SAM/ASAM optimization, and we are unable to find a configuration that achieves satisfactory performance.

There are several potential reasons for the poor performance of the orthogonalization approach. One possibility is that, even though we aim to stay within the local minimum found by SGD, the orthogonalized gradients may still lead to a large enough step that causes the model to escape the minimum. In such cases, the gradient information may not be sufficiently informative to guide the model towards a better solution.

Another issue could be related to the fact that we only orthogonalize the SAM/ASAM gradients with respect to the SGD gradients for each individual batch. This batch-wise orthogonalization may not effectively capture the overall direction of the SGD gradients across the entire dataset, leading to suboptimal updates.

Furthermore, it is possible that the orthogonalization process itself is problematic and may send the model parameters in a completely random direction. By forcibly removing the component of the SAM/ASAM gradients that aligns with the SGD gradients, we may be discarding valuable information and introducing noise into the optimization process.

In summary, our attempts to improve the performance of LastSAM by orthogonalizing the SAM/ASAM gradients with respect to the SGD gradients have been unsuccessful. While the motivation behind this approach is intuitive, the practical implementation reveals significant challenges and limitations. Further research is needed to understand the dynamics of this orthogonalization process and to develop more effective strategies for leveraging the information from both SGD and SAM/ASAM optimization.

4 Conclusion

In this work, we proposed LastSAM, an approach to improving the generalization performance of deep learning models by applying sharpness-aware minimization techniques only after standard training. Our experimental results on the CIFAR-10 dataset using ResNet20 and WRN-28-10 architectures demonstrate that LastSAM achieves slightly better or comparable test accuracies compared to the default SAM and ASAM methods while requiring fewer computational resources during training.

The key advantage of LastSAM lies in its computational efficiency, as it effectively optimizes for flatness in the loss landscape using SAM/ASAM updates for only the last few epochs of training, while using standard SGD for the majority of the training process. This approach significantly reduces the computational burden associated with SAM and ASAM, making them more practical for larger-scale datasets and models. Moreover, our hyperparameter sensitivity analysis reveals that LastSAM, particularly when using ASAM, exhibits improved convergence stability compared to SAM. LastSAM with ASAM maintains high performance across a wide range of neighborhood size (ρ) values, indicating its robustness to hyperparameter choices. This stability can be attributed to the fact that SGD already guides the model towards good optima, and the subsequent application of ASAM helps to refine the solution while maintaining stability.

In summary, LastSAM offers a computationally efficient and effective approach to improving generalization performance in deep learning models. By leveraging the power of SAM and ASAM in the later stages of training, LastSAM achieves state-of-the-art performance while minimizing the computational overhead and enhancing convergence stability. This finding opens up new possibilities for applying SAM and ASAM to a wider range of datasets and models, as the computational cost and hyperparameter tuning complexity are no longer limiting factors. Further research could explore the

application of LastSAM to other datasets, architectures, and tasks, as well as investigate the potential of combining LastSAM with other regularization techniques to further improve generalization performance.

5 Citations

References

[1] Liu, C., Zhu, L., & Belkin, M. (2021) Loss landscapes and optimization in over-parameterized non-linear systems and neural networks. arXiv preprint arXiv:2003.00307.

[2] Jastrzębski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., & Storkey, A. (2018) Three Factors Influencing Minima in SGD. arXiv preprint arXiv:1711.04623.

[3] Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2021) Sharpness-Aware Minimization for Efficiently Improving Generalization. In International Conference on Learning Representations.

[4] Kwon, J., Kim, J., Park, H., & Choi, I.K. (2021) Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In International Conference on Machine Learning (pp. 5905–5914). PMLR.

[5] Zhang, C., Bengio, S., Hardt, M., Recht, B., & Vinyals, O. (2017) Understanding deep learning requires rethinking generalization. In International Conference on Learning Representations.

[6] Ioffe, S. & Szegedy, C. (2015) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In Proceedings of the 32nd International Conference on Machine Learning (pp. 448–456). PMLR.

[7] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014) Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929–1958.

[8] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016) Rethinking the Inception Architecture for Computer Vision. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 2818–2826).

[9] Zhang, H., Cisse, M., Dauphin, Y.N., & Lopez-Paz, D. (2018) mixup: Beyond Empirical Risk Minimization. In International Conference on Learning Representations.

[10] DeVries, T., & Taylor, G.W. (2017) Improved Regularization of Convolutional Neural Networks with Cutout. arXiv preprint arXiv:1708.04552.

[11] Smith, L.N. (2017, March) Cyclical learning rates for training neural networks. In 2017 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 464–472). IEEE.

[12] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A.G. (2018) Averaging Weights Leads to Wider Optima and Better Generalization. In Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence (pp. 876–885). AUAI Press.