When Does Re-initialization Work?

Sheheryar Zaidi[1*] **Tudor Berariu**[2*] **Hyunjik Kim**[3]
Jörg Bornschein[3] **Claudia Clopath**[2,3] **Yee Whye Teh**[1,3] **Razvan Pascanu**[3]
[1]University of Oxford, [2]Imperial College London, [3]DeepMind
[*]Equal contribution.
Correspondence to: `szaidi@stats.ox.ac.uk, t.berariu19@imperial.ac.uk`

Abstract

Re-initializing a neural network during training has been observed to improve generalization in recent works. Yet it is neither widely adopted in deep learning practice nor is it often used in state-of-the-art training protocols. This raises the question of when re-initialization works, and whether it should be used together with regularization techniques such as data augmentation, weight decay and learning rate schedules. In this work, we conduct an extensive empirical comparison of standard training with a selection of re-initialization methods to answer this question, training over 15,000 models on a variety of image classification benchmarks. We first establish that such methods are consistently beneficial for generalization in the absence of any other regularization. However, when deployed alongside other carefully tuned regularization techniques, re-initialization methods offer little to no added benefit for generalization, although optimal generalization performance becomes less sensitive to the choice of learning rate and weight decay hyperparameters. To investigate the impact of re-initialization methods on noisy data, we also consider learning under label noise. Surprisingly, in this case, re-initialization significantly improves upon standard training, even in the presence of other carefully tuned regularization techniques.

1 Introduction

Recent works [e.g. 1, 2] have proposed a set of techniques for training neural networks based on *re-initialization*. These methods, which we collectively refer to as *re-initialization methods*, involve re-initializing and transforming a part or all of the parameters of a neural network periodically throughout learning. Studies have shown that re-initialization can help in certain settings, such as small-data regimes [1, 27] and online learning [2]. However, despite having no overhead in terms of computation cost and small implementation overhead, such re-initialization techniques have not yet been adopted as common deep learning practice. Indeed, most state-of-the-art (SOTA) training protocols do not incorporate re-initialization techniques, relying instead on advances in *e.g.* optimization [10], architectures [22, 28, 5], data augmentation [8, 9] and pre-training [7].

However, prior work suggests that re-initialization can improve the generalization performance of neural networks. For example, in the context of online learning, Ash and Adams [2] studied a scenario in which training data arrives sequentially in "chunks" over time such that at any point in time the training dataset consists of the union of all chunks arrived so far. They compared training the network from scratch each time a new chunk arrives to *warm-starting*, where we continuously train (*i.e.* fine-tune) the model, finding that warm-starting significantly underperformed. In order to remedy this, they proposed Shrink & Perturb (defined in Section 2.2), a re-initialization technique applied each time a new chunk arrives. Interestingly, applying Shrink & Perturb not only closed the performance gap between warm-starting and training from scratch, but *improved upon the model trained from scratch*. Does this mean that the standard approach to training neural networks without re-initialization is sub-optimal?

I Can't Believe It's Not Better Workshop at NeurIPS 2022.

The motivation of our work is to empirically investigate whether re-initialization improves generalization in supervised learning and understand its benefits and limitations for training neural networks. To that end, we trained over 15,000 models to identify the settings under which re-initialization methods are helpful. We study the interaction between re-initialization and other widely used regularization and optimization techniques, including data augmentation, weight decay and learning rate schedules. We seek to answer a fundamental, practical question: are the benefits of re-initialization additive with those of common, existing techniques for improving generalization? Our experiments consider settings ranging from vanilla training to SOTA protocols which yield top performance. In addition to varying the training protocol, we also investigate the impact of re-initialization methods when learning on noisy data by studying what happens under the presence of label noise, leading to some surprising improvements.

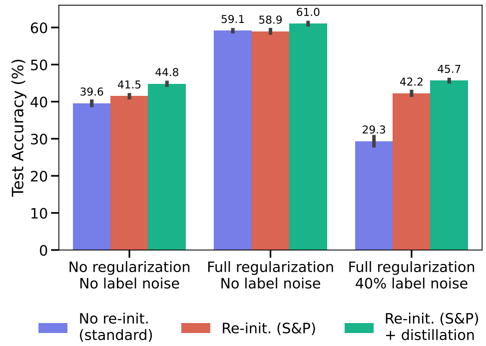

Figure 1: **Comparison of standard training (*i.e.* no re-initialization) with re-initialization using Shrink & Perturb in three scenarios on Tiny ImageNet with PreAct-ResNet-18**. Full regularization refers to the usage of carefully tuned data augmentation, weight decay and a learning rate schedule, whereas no regularization is without. Without label noise, Shrink & Perturb helps in the absence of other regularization but has limited benefit otherwise. With label noise, Shrink & Perturb helps significantly even alongside other regularization. Adding distillation further improves performance. All training protocols have approximately equal computational cost.

Our empirical study centers around two re-initialization methods. First, as our primary focus, we consider *Shrink & Perturb* [2] as described earlier. Second, we consider the recently proposed *Layer-wise Re-initialization* [1] that has been shown to outperform prior re-initialization schemes, particularly in small data regimes. Furthermore, we show that re-initialization methods can naturally be adapted to incorporate self-distillation [11] which typically leads to improvement, at negligible cost. Figure 1 summarizes our results on Tiny ImageNet. From here on, we will use *standard training* to refer to training *without* re-initialization.

Our contributions and findings are as follows:

- **Shrink & Perturb can benefit i.i.d. learning.** Although previously proposed for online learning, Shrink & Perturb can also benefit i.i.d supervised learning and outperforms techniques such as Layer-wise Re-initialization in certain settings.
- **Re-initialization improves generalization in the absence of other regularization (Section 3).** There is a consistent advantage in periodically re-initializing a neural network—even up to 25 times—during training, pointing to the inherent regularization benefit of re-initialization.
- **Re-initialization has little to no benefit relative to standard training in a SOTA setting (Section 4).** When data augmentation, weight decay and learning rate schedules are carefully tuned, re-initialization performs at par with standard training. However, the optimal performance becomes *more robust* to the choice of learning rate and weight decay, which is desirable in practice.
- **Under label noise, re-initialization methods lead to significant improvement in generalization (Section 5).** This improvement appears on top of other well-tuned regularization, revealing a setting where the effects of re-initialization and other regularization techniques do not overlap.

Related work is discussed in Appendix A.

2 Background on Re-initialization Methods

Let $f_{\boldsymbol{\theta}}$ be a neural network with parameters $\boldsymbol{\theta} \in \mathbb{R}^p$ trained to minimize a loss function $L(\boldsymbol{\theta})$. Moreover, let p_{init} be a probability distribution over $\mathbb{R}^p$ from which we sample the *initialization parameters* $\boldsymbol{\theta}_0 \sim p_{\text{init}}$. Then, we define a *re-initialization* of $f_{\boldsymbol{\theta}}$ to be the function $f_{\boldsymbol{\theta}_{\text{RI}}}$, where we have replaced the current parameters $\boldsymbol{\theta}$ with re-initialized parameters $\boldsymbol{\theta}_{\text{RI}} = R(\boldsymbol{\theta}_{\text{init}}, \boldsymbol{\theta})$. The function R computes the re-initialized parameters using a fresh initialization $\boldsymbol{\theta}_{\text{init}} \sim p_{\text{init}}$ and the current parameters $\boldsymbol{\theta}$. Different re-initialization methods will differ in terms of R.

Re-initialization methods will modify standard training as follows. Assuming a computational budget corresponding to N epochs over the training data, we train the model in T *stages*, each for $\lfloor N/T \rfloor$ epochs. Between any two stages, we apply re-initialization as described earlier. Explicitly, let $\boldsymbol{\theta}_{t-1}^{\text{end}}$ denote the parameters at the end of the previous stage $t-1$. Stage t then consists of training the model starting from initial parameters $\boldsymbol{\theta}_t^{\text{start}} = R(\boldsymbol{\theta}_{\text{init}}, \boldsymbol{\theta}_{t-1}^{\text{end}})$. The initial parameters for stage 1 are sampled directly from p_{init}. See Algorithm 1 for pseudo-code.

2.1 Incorporating Self-Distillation With Re-initialization

Our experiments will also consider the effect of incorporating *self-distillation*, which is a natural extension of re-initialization methods because training is split into multiple sequential stages. Specifically, when using distillation, we train the model in stage t by minimizing a combination of the training loss and a distillation loss between the current model (the student) and the model at the end of the previous stage $t-1$ (the teacher), that is, we minimize the loss

$$L(\boldsymbol{\theta}) + \beta_{\text{distill}}\, \mathrm{KL}(f_{\boldsymbol{\theta}_{t-1}^{\text{end}}} \parallel f_{\boldsymbol{\theta}}), \tag{1}$$

with respect to $\boldsymbol{\theta}$, where the second term measures the KL-divergence between the predicted class probabilities of the teacher and the student on the training data and β_{distill} is a hyperparameter. In stage $t = 1$, we only optimize the loss $L(\boldsymbol{\theta})$. Note that the setting in which re-initialization simply consists of replacing all the old model parameters with a new initialization, *i.e.* $\boldsymbol{\theta}_{\text{RI}} = \boldsymbol{\theta}_{\text{init}}$, combined with distillation is equivalent to Born-Again Networks (BANs) [11] supervised by the true and teacher labels.

2.2 Shrink & Perturb

Our primary focus in this work will be Shrink & Perturb, a method proposed by Ash and Adams [2] in the context of online learning and warm-starting neural network training. Shrink & Perturb consists of defining the re-initialized parameters to be:

$$\boldsymbol{\theta}_{\text{RI}} = \lambda \boldsymbol{\theta} + \gamma \boldsymbol{\theta}_{\text{init}},$$

where $\lambda, \gamma \in [0, 1]$ are *shrinkage* and *perturbation* hyperparameters respectively. Note that contrary to the setup of Ash and Adams [2] where Shrink & Perturb is applied for online learning, we will view Shrink & Perturb as a re-initialization method for training on a stationary dataset as described earlier.

2.3 Layer-wise Re-initialization

Alabdulmohsin et al. [1] recently proposed *Layer-wise Re-initialization*, which re-initializes the architecture block-by-block during training. Assuming that there are K "blocks" in the architecture, we have a total of $T = K$ stages during training. At the end of stage t, we re-initialize all layers after the t-th block, that is, we set:

$$\boldsymbol{\theta}_{\text{RI}} = \boldsymbol{\theta} \odot \boldsymbol{m}^{(t)} + \boldsymbol{\theta}_{\text{init}} \odot (1 - \boldsymbol{m}^{(t)}).$$

Here $\boldsymbol{m}^{(t)} \in \{0, 1\}^p$ is a vector such that $\boldsymbol{m}_i^{(t)} = 1$ if the i-th parameters belongs to the first t blocks and is otherwise zero. $\odot$ denotes an element-wise product. More generally, note that we can have $T = KM$ total stages, where at the end of each of the first M stages $t = 1, \ldots, M$, we re-initialize all layers after the first block. At the end of each of the second M stages $t = M+1, \ldots, 2M$, we re-initialize all layers after the second block and so forth. In addition to re-initializing all blocks after the t-th block, Layer-wise Re-initialization also (1) re-scales the norm of the first t blocks to their norm at initialization and (2) adds a normalization layer after block t.

3 The Regularizing Effect of Re-initialization

We first investigate the effect of re-initialization when no other regularization is deployed, akin to settings where prior work [2] suggests re-initialization helps. We show that re-intiliazation provides a consistent and considerable advantage over standard training, even beyond the small-data regime [1, 27], as is the case of Tiny ImageNet. See Appendices C and D for details of the experimental setup and hyperparameter choices.

Table 1: Test accuracy (%) of different methods in settings ranging from basic to SOTA protocols on CIFAR-100 with ResNet-18.

Setting Abbrev.	Data Aug.	Cosine Anneal.	Weight Decay	No Re-initialization (standard training)	Self-distillation (fixed-budget BAN)	SGDR	Layer-wise Re-initialization		Shrink & Perturb	
							w/o dist.	w/ dist.	w/o dist.	w/ dist.
∅	✗	✗	✗	55.5 ±0.6	56.4 ±0.5	N/A	61.0 ±0.6	62.5 ±0.2	63.1 ±0.6	**63.5** ±0.3
D	✓	✗	✗	70.8 ±0.1	70.5 ±0.5	N/A	72.1 ±0.3	**74.7** ±0.2	71.9 ±0.1	74.0 ±0.6
DC	✓	✓	✗	71.2 ±0.2	70.9 ±0.4	71.0 ±0.6	74.6 ±0.5	**75.4** ±0.2	**75.4** ±0.3	**75.4** ±0.4
DCW	✓	✓	✓	**77.9** ±0.2	77.2 ±0.1	77.5 ±0.2	77.5 ±0.1	77.3 ±0.3	77.5 ±0.2	77.0 ±0.3

Table 2: Test accuracy (%) of standard training and Shrink & Perturb on Tiny ImageNet with PreAct-ResNet-18.

Setting Abbrev.	Data Aug.	Cosine Anneal.	Weight Decay	No Re-initialization (standard training)	Shrink & Perturb	
					w/o dist.	w/ dist.
∅	✗	✗	✗	39.55±0.69	41.47±0.48	**44.81**±0.46
DCW	✓	✓	✓	59.12±0.33	58.95±0.56	61.03±0.38

The different settings we consider are labeled ∅, D, DC and DCW, as described on the left of Table 1. Setting ∅ is the basic setting considered in this section, where we use a constant learning rate, no weight decay and no data augmentation. As shown in the first rows of Tables 1 and 3, both Shrink & Perturb and Layer-wise Re-initialization improve generalization performance over standard training for both CIFAR datasets, with Shrink & Perturb outperforming Layer-wise Re-initialization by a margin of 1-2 percentage points in both cases. Distillation is also beneficial, though more for CIFAR-10 than CIFAR-100. These findings also hold for the larger Tiny ImageNet dataset, where we focus on only Shrink & Perturb with and without distillation, as shown in the first row of Table 2. In summary, re-initialization significantly boosts generalization without additional computational cost in setting ∅.

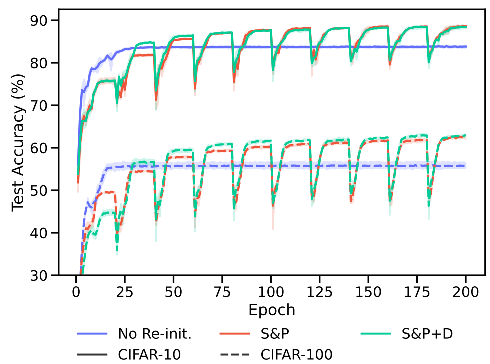

Figure 2: **Example test accuracy curves on the CIFAR datasets with ResNet-18 in setting ∅.** Shrink & Perturb involves 10 stages. For each method and dataset, the learning rate is tuned separately – the optimal learning rates are on average 10 times smaller for re-initialization than standard training here.

Examples of the test accuracy learning curves are shown in Figure 2 for Shrink & Perturb with 10 stages. Notice that each re-initialization causes a sudden drop in performance from which there is a quick recovery. Whereas the test accuracy for standard training stagnates early on, training with Shrink & Perturb keeps improving test accuracy with each successive re-initialization. All models shown in this plot achieved approximately 100% training accuracy and close to zero training negative log-likelihood. See Figures 9 and 10 for examples of training curves.

4 Re-initialization Alongside Other Regularization

Next, we consider how beneficial re-initialization is when deployed alongside other regularization techniques that are commonly used to achieve high performance in SOTA training protocols. Beginning with the simple setting explored in Section 3, we gradually add **d**ata augmentation (setting D), **c**osine annealing (setting DC) and **w**eight decay (setting DCW), achieving around SOTA performance in setting DCW. Our motivation is to explore whether the effect of these different techniques is *additive* with the benefit of re-initialization. As shown in Tables 1 and 3, all re-initialization methods except fixed-budget BAN noticeably improve upon standard training in settings ∅, D and DC. In most cases, the best performing method is Shrink & Perturb with distillation, improving accuracy over standard training by upto 5 percentage points on CIFAR-10 and 8 percentage points on CIFAR-100.

However, a key negative finding here is that **when all regularization techniques are used and carefully tuned (setting DCW), the best re-initialization methods do not improve generalization performance over standard training**. Note that setting DCW is arguably more important and common in practice than previous settings. We speculate that Shrink & Perturb affects the norm of

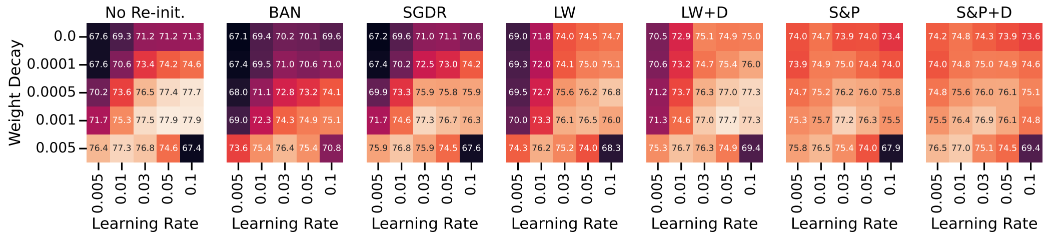

Figure 3: **Re-initialization can make performance less sensitive to the choice of learning rate and weight decay.** Performance of different methods over a grid search of learning rate and weight decay combinations on CIFAR-100 with ResNet-18 in setting DCW. All re-initialization methods use 5 stages. Shrink & Perturb stands out most, as its performance varies much less over the grid than other methods, especially standard training (no re-initialization).

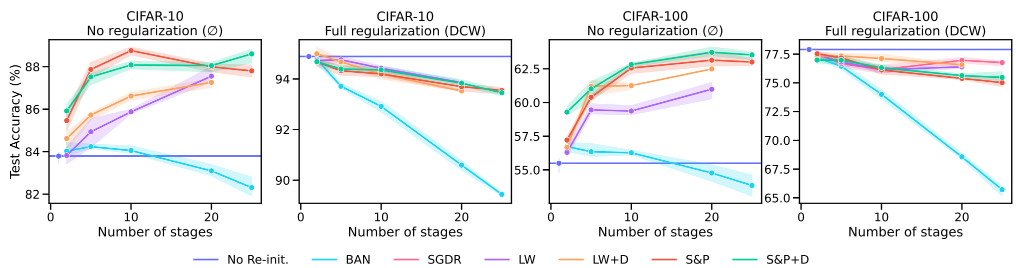

Figure 4: **Test accuracy as a function of the number of re-initialization stages in settings $\varnothing$ and DCW.** In setting $\varnothing$, *any* number of stages improves upon standard training for Shrink & Perturb and Layer-wise Re-initialization. In setting DCW, adding more stages monotonically worsens generalization. Recall that for Layer-wise Re-initialization, the number of stages is tied to the architecture itself, and $T \in \{2, 5, 10, 20\}$.

the weights during training which may overlap with the effect of weight decay, as discussed further in Appendix B.1. We also show the impact of re-initialization in setting DCW for Tiny ImageNet in the last row of Table 2. Similar to before, the improvement in performance from using Shrink & Perturb is much smaller here than in setting $\varnothing$. Shrink & Perturb has no impact on performance, whereas Shrink & Perturb with distillation slightly improves performance over standard training.

4.1 Re-initialization Makes Optimal Performance More Robust to Hyperparameters

Although re-initialization methods do not offer much benefit in terms of generalization for optimal hyperparameters in setting DCW, we found that they can make performance less sensitive to the choice of learning rate and weight decay hyperparameters. Figure 3 shows the test accuracy achieved by different methods over a grid of learning rate and weight decay choices for CIFAR-100 in setting DCW. We observe that the performance of both Layer-wise Re-initialization and Shrink & Perturb varies less over the grid compared to no re-initialization. This is especially prominent for Shrink & Perturb. These methods can hence be beneficial when thorough hyperparameter tuning is infeasible.

4.2 Impact of the Number of Re-initialization Stages

The number of stages T used for re-initialization is an additional hyperparameter that we tuned in our experiments. Note that $T = 1$ trivially corresponds to standard training (*i.e.* no re-initialization). We show how test accuracy varies as a function of T in Figure 4 for the different methods over each of the CIFAR datasets in settings $\varnothing$ and DCW. In setting $\varnothing$, typically a large number of stages, in the range 5-20, is most beneficial for the re-initialization methods. In fact, for Shrink & Perturb and Layer-wise Re-initialization, *any* number of stages in the range shown (2-25) improves upon standard training. However, in setting DCW, the profile of the curves changes drastically. Increasing the number of stages monotonically lowers the test accuracy, which is consistent with our finding that in this setting, re-initialization methods offer little advantage. See Figure 11 in Appendix B for further experiments with MobileNetV2 on the CIFAR datasets.

Additional Results. Further experiments can be found in Appendix B, including discussions on the roles of self-distillation (Appendix B.2), the learning rate schedule (Appendix B.3) and preliminary results on ImageNet (Appendix B.4). We also explore the implications of our findings for online

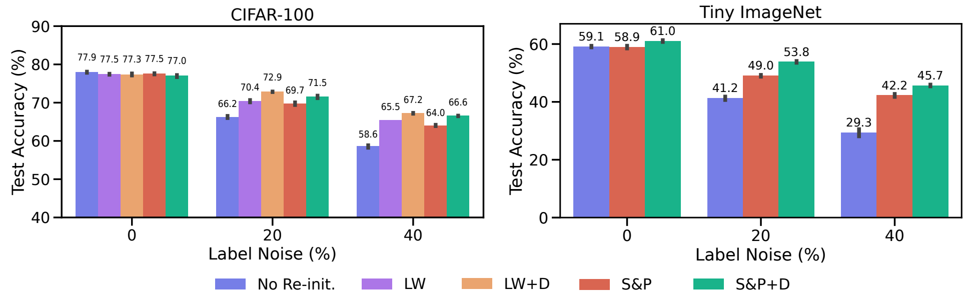

Figure 5: **Re-initialization is beneficial for learning under label noise even with full regularization.** Both re-initialization methods improve performance, with further improvements from distillation. The benefit of re-initialization compared to standard training increases with more label noise. See Figure 7 for corresponding results on CIFAR-10.

learning in Appendix B.5, where Shrink & Perturb was initially proposed, similarly finding a strong dependence on the presence of other regularization.

5 Re-initialization Under Label Noise

Having studied how different components of the training protocol interact with re-initialization, we next focus on learning under noisy data, where learning is more challenging, potentially requiring stricter regularization. In particular, we add label noise to the data by randomizing the labels of a randomly-chosen fraction q of the training data points, keeping the test set the same. We remain in setting DCW, where data augmentation, weight decay and learning rate schedules are used.

Figure 5 shows the performance of different methods as we add $q = 20\%$ and 40% label noise on the CIFAR and Tiny ImageNet datasets. Surprisingly, re-initialization methods consistently improve upon standard training here. For CIFAR datasets, both Shrink & Perturb and Layer-wise Re-initialization are beneficial each on their own, and performance further improves with the addition of distillation. The results are similar for Tiny ImageNet, where the difference between standard training and re-initialization is even starker: over 15 percentage points with 40% label noise.

We can therefore conclude that the effects of re-initialization do not completely overlap with those of data augmentation, weight decay and learning rate schedules. Indeed, even though re-initialization does not improve generalization in setting DCW without label noise, it shows a substantial improvement on generalization with label noise on top of standard regularization techniques. We suspect that one reason re-initialization helps in this scenario relates to the order in which neural networks learn training examples. "Difficult examples" (such as mislabelled inputs) tend to be learned *later* during training by *deeper* layers [3]. Re-initialization might be exploiting this property of learning by naturally selecting the clean data: since periodically re-initializing "resets" training to an extent, the network might be prevented from confidently fitting to the noise (which is learned later) to instead focus on learning the correctly labelled examples (which are learned earlier).

6 Conclusion, Limitations & Future Work

We have investigated when re-initialization methods, a set of techniques that are simple to implement with almost no computational overhead, improve generalization compared to standard training. Although they appear very promising in the absence of other regularization, they do not improve over standard training with sufficient regularization, unless the training data contains label noise. Moreover, since re-initialization methods reduce sensitivity to hyperparameter choices, unless the hyperparameters for standard training are carefully tuned, re-initialization may incorrectly appear to improve generalization. This highlights the importance in empirical research of ensuring (as much as computationally feasible) that sub-optimal hyperparameter choices do not lead to unfair comparisons and incorrect conclusions.

One limitation of our work is that, although we observe clear empirical trends in *when* re-initialization works, a deeper understanding of *why* it works or not is missing and constitutes important future work. Another limitation of our study is that we restrict ourselves to CIFAR-10/100 and Tiny ImageNet datasets and convolution-based network architectures for classification. Although this restriction allowed us to thoroughly explore the interaction between re-initialization and other techniques on these standard benchmarks and carefully tune hyperparameters given a limited compute budget, it would be interesting to extend the scope to other tasks and data modalities as future work.

References

[1] Ibrahim Alabdulmohsin, Hartmut Maennel, and Daniel Keysers. The impact of reinitialization on generalization in convolutional neural networks. *arXiv preprint arXiv:2109.00267*, 2021.

[2] Jordan Ash and Ryan P Adams. On warm-starting neural network training. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 3884–3894. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/288cd2567953f06e460a33951f55daaf-Paper.pdf`.

[3] Robert John Nicholas Baldock, Hartmut Maennel, and Behnam Neyshabur. Deep learning through the lens of example difficulty. 2021.

[4] Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6241–6250, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[5] Andrew Brock, Soham De, Samuel L Smith, and Karen Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

[6] Lucas Caccia, Jing Xu, Myle Ott, Marc'Aurelio Ranzato, and Ludovic Denoyer. On anytime learning at macroscale. *arXiv preprint arXiv:2106.09563*, 2021.

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[9] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[10] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur. Sharpness-aware minimization for efficiently improving generalization. *arXiv preprint arXiv:2010.01412*, 2020.

[11] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1607–1616. PMLR, 10–15 Jul 2018. URL `https://proceedings.mlr.press/v80/furlanello18a.html`.

[12] Song Han, Jeff Pool, Sharan Narang, Huizi Mao, Enhao Gong, Shijian Tang, Erich Elsen, Peter Vajda, Manohar Paluri, John Tran, et al. Dsd: Dense-sparse-dense training for deep neural networks. *arXiv preprint arXiv:1607.04381*, 2016.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 630–645, Cham, 2016. Springer International Publishing. ISBN 978-3-319-46493-0.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[16] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=BJYwwY9ll`.

[17] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. Transient non-stationarity and generalisation in deep reinforcement learning. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=Qun8fv4qSby`.

[18] Pavel Izmailov, Dmitrii Podoprikhin, Timur Garipov, Dmitry P. Vetrov, and Andrew Gordon Wilson. Averaging weights leads to wider optima and better generalization. In Amir Globerson and Ricardo Silva, editors, *UAI*, pages 876–885. AUAI Press, 2018. URL `http://dblp.uni-trier.de/db/conf/uai/uai2018.html#IzmailovPGVW18`.

[19] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research), 2009. URL `http://www.cs.toronto.edu/~kriz/cifar.html`.

[20] Y. Le and X. Yang. Tiny imagenet visual recognition challenge, 2015.

[21] Xingjian Li, Haoyi Xiong, Haozhe An, Cheng-Zhong Xu, and Dejing Dou. Rifle: Backpropagation in depth for deep transfer learning through re-initializing the fully-connected layer. In *ICML*, 2020.

[22] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[24] Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A PAC-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=Skz_WfbCZ`.

[25] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and linear bottlenecks: Mobile networks for classification, detection and segmentation. *CoRR*, abs/1801.04381, 2018. URL `http://arxiv.org/abs/1801.04381`.

[26] Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017. doi: 10.1109/WACV.2017.58.

[27] Ahmed Taha, Abhinav Shrivastava, and Larry S Davis. Knowledge evolution in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12843–12852, 2021.

[28] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[29] Kaikai Zhao, Tetsu Matsukawa, and Einoshin Suzuki. Retraining: A simple way to improve the ensemble accuracy of deep neural networks for image classification. In *2018 24th International Conference on Pattern Recognition (ICPR)*, pages 860–867, 2018. doi: 10.1109/ICPR.2018.8545535.

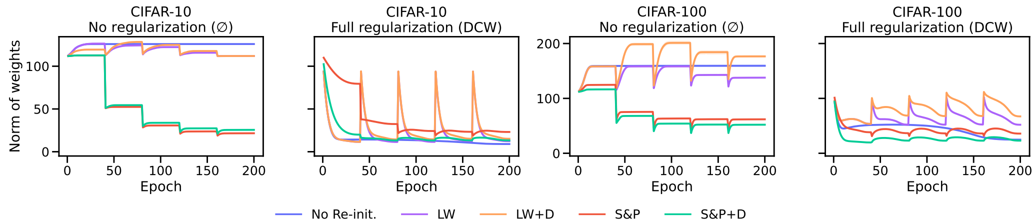

Figure 6: **Weight norms during training vary across methods more in setting ∅ than setting DCW.** Curves show how the norm of ResNet-18 weights varies through training for re-initialization methods (with 5 stages) and standard training. In setting ∅, Shrink & Perturb yields a model with smaller weight norm than standard training and Layer-wise Re-initialization, but in setting DCW, all methods yield models with similar weight norms.

A Related Work

Various re-initialization methods have been proposed in the literature, differing usually by motivation and choice of weights that are re-initialized. For example, motivated by sparsity, Han et al. [12] introduced dense-sparse-dense training, where after an initial stage of training, the smallest weights are pruned to induce sparsity, the network is trained and in the third and last stage, the pruned weights are re-initialized from zero. Taha et al. [27] also proposed "knowledge evolution" which partitions the weights of a network into two parts, one of which is continuously training and the other repeatedly re-initialized. For transfer learning, Li et al. [21] proposed to periodically re-initialize only the final layer, optionally with ensembling [29]. Recently, Alabdulmohsin et al. [1] compared a number of these methods with their proposed Layer-wise Re-initialization. They found it outperformed prior approaches, which is why we chose to study it in this work. Their empirical evaluation focused on the small-data regime, where one set of hyperparameters common across multiple datasets and architectures are used. They also did not study how re-initialization methods interact with other regularizers. Also, in the spirit of re-initialization, various approaches for "resetting" or "restarting" training can also be found in the literature [23, 16, 26, 18].

Re-initialization also recently appeared in the context of online learning and warm-starting neural networks, where data arrives sequentially [2, 6]. Ash and Adams [2] showed that *warm-starting* neural network training worsens generalization. They proposed Shrink & Perturb, a simple technique to improve performance in their online learning setup. The benefits of Shrink & Perturb as a generic re-initialization method for i.i.d. learning have not been studied before. Caccia et al. [6] studied techniques for expanding capacity by initializing *new* parameters as more data arrives. Igl et al. [17] observed a similar phenomenon in reinforcement learning and proposed distillation as a remedy. Generally, self-distillation [11, 15] itself closely relates to re-initialization (see Section 2.1).

B Additional Results

B.1 Weight Norms in Setting DCW

We speculate that part of the reason why Shrink & Perturb does not outperform standard training in setting DCW as opposed to settings ∅, D and DC relates to its effect on the norm of the weights during training which may overlap with the effect of weight decay. As shown in Figure 6, in the absence of weight decay, the norm of the weights tends to increase monotonically without re-initialization, whereas Shrink & Perturb periodically reduces this weight norm. In setting ∅, this results in the learned models' weight norms varying across methods, while, in setting DCW, all methods learn models with similar final norms due to weight decay. Moreover, in setting ∅, Shrink & Perturb results in a model with norm similar to that of standard training in setting DCW. Therefore, both weight decay and Shrink & Perturb encourage the final learned model to have a small weight norm, a quantity that is known to be linked to generalization performance [24, 4]. Nevertheless, this does not fully explain why Shrink & Perturb works, as shown in Section 5.

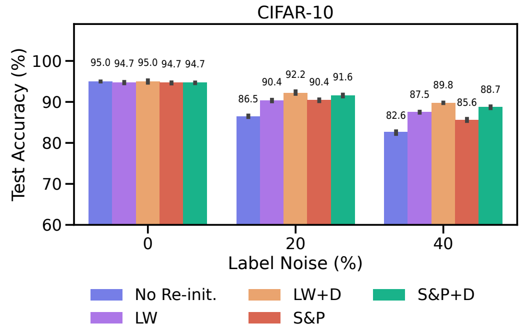

Figure 7: **Re-initialization is beneficial for learning under label noise even with full regularization.** Results for CIFAR-10. See Figure 5 and Section 5.

Table 3: Test accuracy (%) of different methods in settings ranging from basic to SOTA protocols on CIFAR-10 with ResNet-18.

Setting Abbrev.	Data Aug.	Cosine Anneal.	Weight Decay	No Re-initialization (standard training)	Self-distillation (fixed-budget BAN)	Layer-wise Re-initialization		Shrink & Perturb	
						w/o dist.	w/ dist.	w/o dist.	w/ dist.
∅	✗	✗	✗	83.8 ±0.3	84.0 ±0.4	87.6 ±0.4	87.3 ±0.2	**88.8** ±0.2	**88.6** ±0.4
D	✓	✗	✗	92.5 ±0.0	92.6 ±0.2	92.6 ±0.3	93.2 ±0.1	93.1 ±0.2	**94.1** ±0.0
DC	✓	✓	✗	92.8 ±0.2	92.6 ±0.1	93.4 ±0.1	93.4 ±0.2	**94.1** ±0.2	**94.2** ±0.1
DCW	✓	✓	✓	95.0 ±0.0	94.8 ±0.2	94.7 ±0.2	95.0 ±0.3	94.7 ±0.2	94.7 ±0.1

B.2 The Role of Self-Distillation

Another observation from Tables 1 and 3 is that fixed-budget BANs do not improve performance compared to standard training in most cases, sometimes leading to worse test accuracies, and are outperformed by Shrink & Perturb and Layer-wise Re-initialization in all settings except DCW. Recall from Section 2.1 that BANs can be viewed as a simple re-initialization method combined with distillation: we re-initialize the full network *from scratch* in each stage. Therefore, BANs constitute an important baseline for more sophisticated re-initialization methods. Our results indicate that while distillation can boost performance, it is computationally sub-optimal to *completely* re-initialize each network in the sequence. Under a fixed-budget setting as shown here, too many stages will then lead to too few gradient steps per stage, whereas too few stages do not reap the full benefits of distillation, both of which can negatively impact performance. On the other hand, Layer-wise Re-initialization and Shrink & Perturb partly "re-use" the model from the previous stage, circumventing the need for a large number of gradient steps per stage and improving learning efficiency. Therefore, re-initialization is itself crucial, and distillation does not suffice on its own. Figure 8 shows how complete re-initialization in BANs leads to large drops in performance from which it is more difficult to recover quickly.

B.3 The Role of Cosine Annealing

For re-initialization methods, in settings DC and DCW with a learning rate schedule, we applied cosine annealing *per stage*, yielding a cyclical learning rate [23]. Such a learning rate schedule can have its own benefits, therefore we compared re-initialization in settings DC and DCW with SGDR for CIFAR-100. As shown in Table 1, SGDR does not match the performance of the re-initialization methods in setting DC and performs similar in setting DCW. This indicates that when re-initialization is helpful, it is not an effect of the cyclical schedule.

B.4 Results on ImageNet

Our findings on the dependence of the benefits of Shrink & Perturb on other regularization were shown to hold for the CIFAR datasets and Tiny ImageNet. In this section, we preliminarily explore whether they also hold for ImageNet. As Table 4 shows, in setting CW, which lacks data augmentation, Shrink & Perturb clearly outperforms no re-initialization. However, in setting DCW with full regularization,

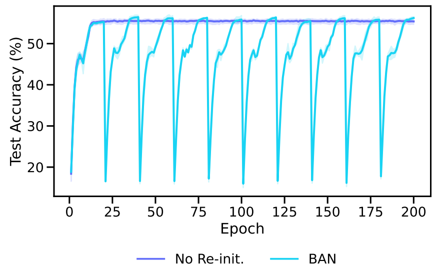

Figure 8: **Example test accuracy curves on the CIFAR-100 dataset with ResNet-18 in setting ∅.** BAN involves 10 stages. For each method and dataset, the learning rate is tuned separately. Notice that each re-initialization causes a large drop in performance in contrast with *e.g.* Shrink & Perturb as shown in Figure 2.

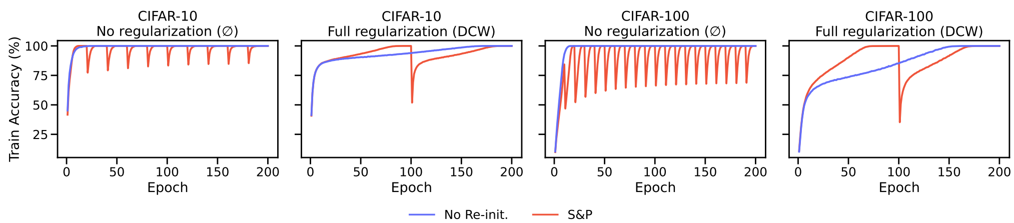

Figure 9: **Example train accuracy curves for standard training and Shrink & Perturb.** In each plot, Shrink & Perturb is shown with its optimal number of stages. Observe that in all cases the models achieve close to 100% accuracy.

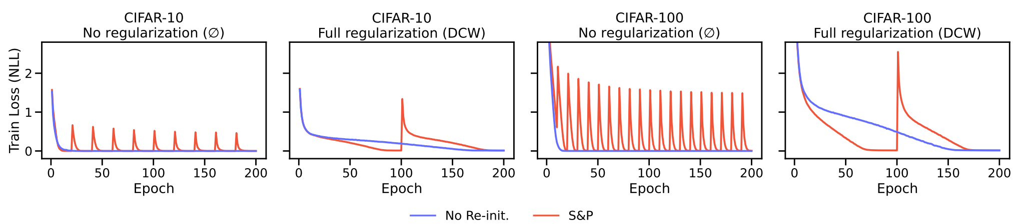

Figure 10: **Example train negative log-likelihood curves for standard training and Shrink & Perturb.** In each plot, Shrink & Perturb is shown with its optimal number of stages. Observe that in all cases the models achieve close to zero loss.

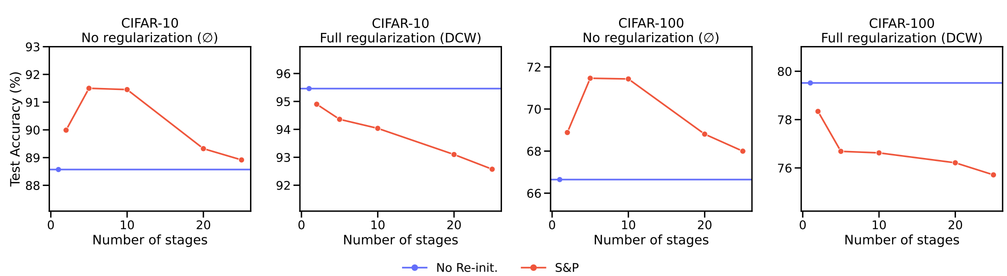

Figure 11: **Effect of Shrink & Perturb on the CIFAR datasets in settings ∅ and DCW with a MobileNetV2 architecture.** Our results in this figure indicate that the overall trend observed in our experiments with ResNet-18 also appears for MobileNetV2. In particular, in setting ∅, Shrink & Perturb improves performance for any number of stages shown, whereas in setting DCW, it provides no benefit over standard training (no re-initialization).

Shrink & Perturb even slightly underperforms compared to standard training. These conclusions also match our findings in Sections 3 and 4.

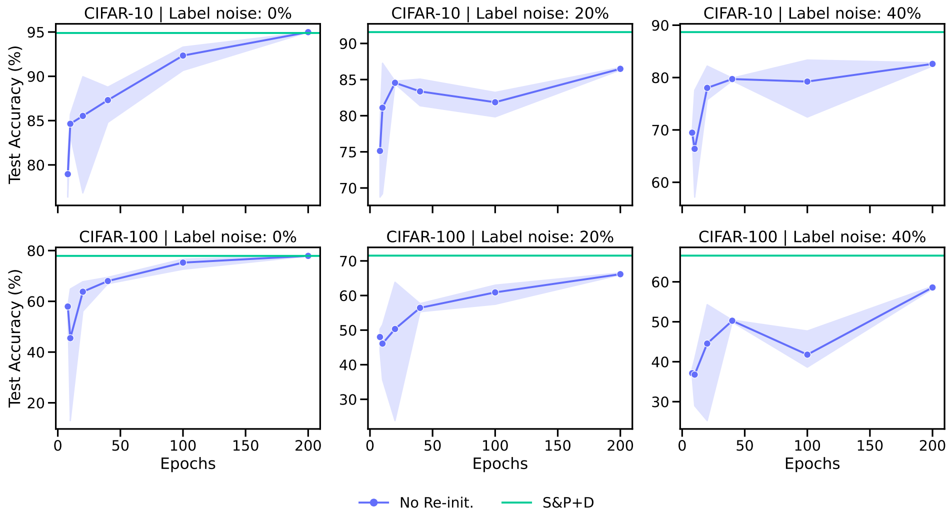

Figure 12: **Under label noise, tuning the epoch budget for standard training does not close the gap with re-initialization methods.** See the discussion in Section 5.

Table 4: Test accuracy (%) of standard training and Shrink & Perturb on ImageNet with ResNet-18.

Setting Abbrev.	Data Aug.	Cosine Anneal.	Weight Decay	No Re-initialization (standard training)	Shrink & Perturb
CW	✗	✓	✓	59.1	63.7
DCW	✓	✓	✓	70.8	69.7

B.5 Implications for Online Learning

Shrink & Perturb was originally proposed by Ash and Adams [2] in the context of online learning (*cf.* Section 2.2), so we explore whether our conclusions also hold there. In particular, how do different methods compare in the absence of other regularization (similar to the setting originally studied by Ash and Adams [2]) and with full regularization? In Figure 13, we assume that training data accumulates by arriving sequentially in five "chunks" leading to five stages of training, and we compare three methods: (1) initializing the network from scratch and re-training it every time a new chunk arrives, (2) warm-starting it (*i.e.* continuously fine-tuning) and (3) applying Shrink & Perturb every time a new chunk arrives. In line with the findings of Ash and Adams [2], without regularization, the warm-started network significantly underperforms the network initialized from scratch, whereas applying Shrink & Perturb outperforms both (left plot in Figure 13). However, under sufficient regularization, all three methods interestingly perform approximately the same (right plot in Figure 13). This also closely matches our findings from Sections 3 and 4.

We also find that the warm-starting generalization gap [2] (*i.e.* warm-started models underperforming models initialized from scratch in each stage) does not exist on ImageNet in a setting with full regularization as shown in Figure 14.

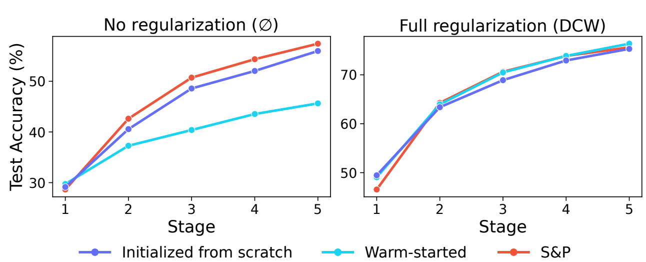

Figure 13: **The benefits of Shrink & Perturb and the warm-starting generalization gap in the online learning setup of Ash and Adams [2] also depend on whether other regularization is used.** Results shown are for a ResNet-18 trained on CIFAR-100. See Appendix B.5 for discussion.

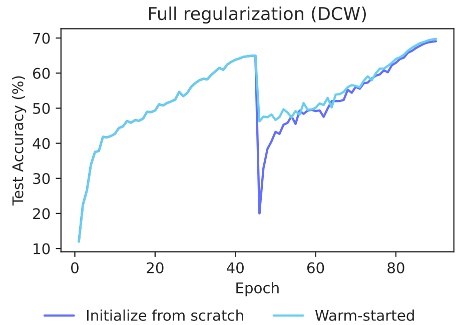

Figure 14: **There is no warm-starting generalization gap for ResNet-18 on ImageNet in the online learning setup of Ash and Adams [2] under full regularization.** Learning curves show the test accuracy on ImageNet where in the first stage we train on half the training data, followed by the full training data in the second stage. See Appendix B.5 for discussion.

B.6 Additional Figures

To ensure that standard training does not underperform re-initialization methods due to overfitting resulting from too many gradient steps (particularly when there is label noise), we explored whether the performance gap between re-initialization methods and standard training can be closed by tuning the epoch budget. As shown in Figure 12, this is *not* sufficient to close to gap.

Figures 9 to 11 are discussed in Sections 3 and 4.

C Experimental Setup

For each architecture (ResNet-18, PreAct-ResNet-18, MobileNetV2) [14, 13, 25] and dataset (CIFAR-10, CIFAR-100, Tiny ImageNet), we used SGD with momentum 0.9 for training with early stopping by validation accuracy. For the CIFAR datasets, we separately tuned the learning rate and weight decay hyperparameters for each method by cross validation. Note that this is different to the setup of Alabdulmohsin et al. [1] where the learning rate and weight decay hyperparameters were fixed for all combinations of datasets and architectures in their image classification experiments. For the larger and more computationally costly Tiny ImageNet dataset, we tuned the learning rate and weight decay hyperparameters for standard training (that is, no re-initialization) and kept them fixed for the re-initialization methods. This simulates a practical scenario: given tuned hyperparameters for standard training on an expensive and large dataset, does it help to use re-initialization without re-tuning the hyperparameters?

For re-initialization methods, we note that the number of stages T is an additional hyperparameter which we tune (*cf.* Section 4.2). Moreover, all models are optimized for the same number of gradient steps for all methods. All other hyperparameters such as batch size and number of epochs are the same for all methods and specified in Tables 5 and 6. Our code is provided for reproducibility and will be open-sourced.

D Hyperparameters & Pseudo-code of Re-initialization Methods

Hyperparameters for our experiments are described in Tables 5 and 6, and the pseudo-code for training with re-initialization is shown in Algorithm 1.

General	
Total training epochs	200
Stages $\times$ epochs per stage	$\{2 \times 100, 5 \times 40, 10 \times 20, 20 \times 10, 25 \times 8\}$
Batch size	125
Momentum	0.9
Learning rate grid	$\{0.005, 0.01, 0.03, 0.05, 0.1\}$
Weight decay grid	$\{0, 0.0001, 0.0005, 0.001, 0.005\}$
Distillation strength β_{distill} (if used)	1
Data augmentation (if used)	Random horizontal flips and size-preserving crops after padding 4 pixels.
Data normalization	$\mu = (0.485, 0.456, 0.406)$ $\sigma = (0.229, 0.224, 0.225)$
Shrink & Perturb	
Shrink λ (default unless explicitly defined otherwise)	0.4
Perturb γ (default unless explicitly defined otherwise)	0.1
Layer-wise Re-initialization	
(K, M)	$\{(2,1), (5,1), (5,2)\}$

Table 5: Hyperparameters common to all models trained on CIFAR-10 and CIFAR-100 (32×32 images) [19].

General	
Total training epochs	500
Stages $\times$ epochs per stage	$\{2 \times 100, 5 \times 40, 10 \times 20, 20 \times 10, 25 \times 8\}$
Batch size	100
Momentum	0.9
Learning rate grid	$\{0.01, 0.05, 0.1\}$
Weight decay grid	$\{0, 0.0001, 0.0005, 0.005\}$
Distillation strength β_{distill} (if used)	2
Data augmentation (if used)	Random horizontal flips and size-preserving crops after padding 4 pixels.
Data normalization	$\mu = (0.485, 0.456, 0.406)$ $\sigma = (0.229, 0.224, 0.225)$
Shrink & Perturb	
Shrink λ (default unless explicitly defined otherwise)	0.4
Perturb γ (default unless explicitly defined otherwise)	0.1

Table 6: Hyperparameters common to all models trained on Tiny ImageNet (64×64 images) [20].

Algorithm 1 Training with re-initialization function R.

Input: Training data $\mathcal{D}$, number of re-initialization stages T, initialization distribution p_{init}, total epochs N.

$P \leftarrow \lfloor N/T \rfloor$ *# epochs per stage*

for $t = 1$ **to** T **do**

  if $t == 1$ **then**

    $\boldsymbol{\theta}^{\text{start}} \sim p_{\text{init}}$ *# initialize training*

  else

    $\boldsymbol{\theta}^{\text{start}} \leftarrow R(\boldsymbol{\theta}_{\text{init}}, \boldsymbol{\theta}^{\text{end}})$, where $\boldsymbol{\theta}_{\text{init}} \sim p_{\text{init}}$ is an i.i.d. initialization. *# re-initialize network*

  end if

  $\boldsymbol{\theta}^{\text{end}} \leftarrow$ parameters after training for P epochs on $\mathcal{D}$, starting from $\boldsymbol{\theta}^{\text{start}}$, optionally with distillation from teacher network $f_{\boldsymbol{\theta}^{\text{end}}}$ if $t > 1$. *# optimization objective is defined in Equation* (1)

end for

return $\boldsymbol{\theta}^{\text{end}}$