

HOW DOES THE OPTIMIZER IMPLICITLY BIAS THE MODEL MERGING LOSS LANDSCAPE?

Chenxiang Zhang^{1*}, Alexander Theus², Damien Teney³
Antonio Orvieto⁴, Jun Pang¹ & Sjouke Mauw¹

ABSTRACT

Model merging methods combine models with different capabilities into a single one while maintaining the same inference cost. Two popular approaches are *linear interpolation*, which linearly interpolates between model weights, and *task arithmetic*, which combines task vectors obtained by the difference between fine-tuned and base models. While useful in practice, what properties make merging effective are poorly understood. This paper explores how the optimization process affects the loss landscape geometry and its impact on merging success. We show that a single quantity – the *effective noise scale* – unifies the impact of optimizer and data choices on model merging. Across architectures and datasets, the effectiveness of merging success is a non-monotonic function of effective noise, with a distinct optimum. Decomposing this quantity, we find that larger learning rates, stronger weight decay, smaller batch sizes, and data augmentation all independently modulate the effective noise scale, exhibiting the same qualitative trend. Unlike prior work that connects optimizer noise to the flatness or generalization of *individual* minima, we show that it also affects the *global* loss landscape, predicting when independently trained solutions can be merged. Our findings broaden the understanding of how optimization shapes the loss landscape geometry and its downstream consequences for model merging, suggesting the possibility of further manipulating the training dynamics to improve merging effectiveness.

1 INTRODUCTION

Model merging combines independently trained solutions to fuse their unique capabilities into a single solution. These methods have been applied to either further improve the performance of a single model (Wortsman et al., 2022) or to combine models with different, but similar, capabilities (Ilharco et al., 2023). Notably, this is achieved without additional computational cost at inference time. Given these practical advantages, merging methods have been applied to improve state-of-the-art architectures (Yadav et al., 2024). However, these approaches rely on extensive trial-and-error, requiring practitioners to train and evaluate multiple models to identify which candidates successfully merge. A fundamental unresolved question is *why some models are compatible for merging while others fail, despite achieving similar individual performance*.

The success of model merging depends on the *mode connectivity* phenomenon, the geometric property that independent solutions can be connected by paths of low-loss models (Draxler et al., 2018; Garipov et al., 2018). Frankle et al. (2020) demonstrated a stricter condition, linear mode connectivity, showing that solutions sharing a common initial optimization trajectory can be connected by a linear path with similar loss. This finding suggests that optimization dynamics, not just final convergence, fundamentally shape the geometry of the loss landscape between solutions. Yet, despite this insight, well-studied factors that affect the optimization dynamics, such as learning rate, weight decay, or batch size, are not well understood for the model merging loss landscape, thereby ignoring their potential role in determining merging compatibility.

In this work, we study the role of optimization dynamics on the outcome of model merging. First, we show that different optimizer components (learning rate, weight decay, batch size, and data augmen-

*Corresponding author: <chenxiang.zhang@uni.lu>

¹University of Luxembourg ²ETH Zürich, MPI for Learning Systems ³Idiap Research Institute ⁴ELLIS Institute Tübingen, MPI for Intelligent Systems, Tübingen AI Center

tation) control the same underlying factor, the *effective noise scale*, which captures the magnitude of stochastic fluctuations in the optimization trajectory. Our experiments demonstrate how the noise magnitude determines whether independently trained solutions occupy merging-compatible regions of the loss landscape. A noise regime that is too small or too large leads to almost no merging benefits, whereas a moderately large noise regime maximizes the benefits. After that, we decompose the noise into individual components, showing that each one exhibits the same qualitative trend. Starting from the learning rate component, we find that a larger (and stable) value can consistently identify solutions that merge more effectively than those trained with smaller learning rates, even when both converge to similar individual performance levels. Since decaying the learning rate (large or small) during the optimization can always lead to a converged solution, it is perhaps surprising that simply starting with a larger learning rate can change the merging outcomes. However, beyond classical research showing direct advantage of large learning rates on generalization (Keskar et al., 2016), recent works presented different implicit biases of training with a larger learning rate, such as a sparser activation (Andriushchenko et al., 2023b), a different sequence of pattern learning (Li et al., 2019), and a flatter solution (Andriushchenko et al., 2023a). Our results extend these benefits beyond single-task performance, showing that a larger learning rate unlocks more effective merging, quantified using performance gain (i.e. performance difference between the merged and the single model).

Similarly to the learning rate, we find that a large weight decay also enables more effective merging, beyond improvements in the single model performance. We explain this phenomenon using the notion of the *effective learning rate* (Van Laarhoven, 2017; Hoffer et al., 2018), which suggests that the main role of weight decay is to prevent the (effective) learning rate, and therefore the stochastic noise, from decaying to zero during training. Additional components, such as batch size and data augmentation, also inject noise into the optimization process. In particular, a smaller batch size creates noisier gradient estimates since each update is computed from fewer samples, leading to more variation in the optimization path (Keskar et al., 2016; Jastrzebski et al., 2017), while data augmentation introduces extra randomness into each minibatch (Hanin & Sun, 2021). We validate these findings across architectures (MLP, Resnet, Densenet, Transformer, and GPT), tasks (SVHN, CIFAR, TinyImagenet, WILDS, and TinyStories), and settings such as transfer learning.

Lastly, we study the effect of the learning rate in task arithmetic merging, which defines a different subspace of solutions than linear interpolation. We find that the geometry of the loss landscape in task arithmetic significantly changes depending on the initialization. Given a pretrained initialization (e.g. CLIP), a larger learning rate identifies solutions with greater merging compatibility (Figure 7). Moreover, the landscape is also flatter compared to a smaller learning rate. When merging solutions trained on two different downstream tasks using different learning rates, we observe that similar optimization configurations are more compatible to merge (Figure 8).

2 PRELIMINARIES

Linear interpolation merging (Frankle et al., 2020). Linear mode connectivity refers to a phenomenon where two minima with similar performance can be connected by a linear path in the parameter space without significant performance degradation along that path. Formally, given two neural networks with parameters θ_A and θ_B , we can define a linear interpolated model θ_{li} as:

$$\theta_{li} = (1 - \alpha)\theta_A + \alpha\theta_B \quad (1)$$

where $\alpha \in [0, 1]$ is the interpolation coefficient. A pair of models exhibits linear mode connectivity if the loss function $\mathcal{L}(\theta_\alpha)$ remains relatively low for all values of α along this linear path. Model merging relies on mode connectivity, but instead, it aims to find solutions with lower loss values.

Task arithmetic merging (Ilharco et al., 2023). Given a base model θ_{base} , a finetuned model θ_t on task t , the task vector is defined as $\tau_t = \theta_t - \theta_{base}$. This vector τ_t encodes all the properties of the task t . Interestingly, task arithmetic enables operations such as addition and scaling of different task vectors, creating a merged model θ_{ta} as:

$$\theta_{ta} = \theta_{base} + \sum_i \alpha_i \tau_{t_i} \quad (2)$$

where $\alpha_i \in \mathbb{R}$ is the coefficient that controls the influence of each task vector. For simplicity, α is usually the same for all the vectors. In order to succeed, both linear mode connectivity and task arithmetic assume that the loss landscape around the finetuned models θ_t is near-convex.

Effective noise scale. Stochastic optimization injects gradient noise ξ into the optimization dynamics. Writing the minibatch gradient as $g_t = \nabla \mathcal{L}(\theta_t) + \xi_t$ where $\mathbb{E}[\xi_t] = 0$ and $\text{Cov}[\xi_t] \approx \Sigma(\theta_t)/B$, the stochastic update (with momentum μ and decoupled weight decay λ) can be viewed as a discretized stochastic differential equation whose “temperature” scales with learning rate and inversely with batch size (Mandt et al., 2017; Smith et al., 2018; McCandlish et al., 2018). This can be summarized by the *effective noise scale*:

$$\mathcal{S}_{\text{eff}}(\eta, B, \mu; \mathcal{A}) \propto \frac{\eta}{B(1-\mu)^2} \text{tr} \Sigma_{\mathcal{A}}(\theta_t), \quad \tilde{\mathcal{S}} = \frac{\eta}{B(1-\mu)^2}, \quad (3)$$

where η is the learning rate, B the batch size, and $\Sigma_{\mathcal{A}}$ the gradient-noise covariance, which increases with stronger data augmentation $\mathcal{A}$ and data diversity. We report the results using the normalized proxy $\tilde{\mathcal{S}}$ and show how it controls the merging effectiveness.

3 THE OPTIMIZER’S IMPLICIT BIAS ON LINEAR INTERPOLATION

This section presents our key findings for linear interpolation merging. We begin by presenting the *effective noise scale* as the unifying implicit bias controlling the effectiveness of merging. Then, we demonstrate that each optimizer component affects merging effectiveness through this noise.

3.1 EFFECTIVE NOISE SCALE AS A UNIFYING FACTOR

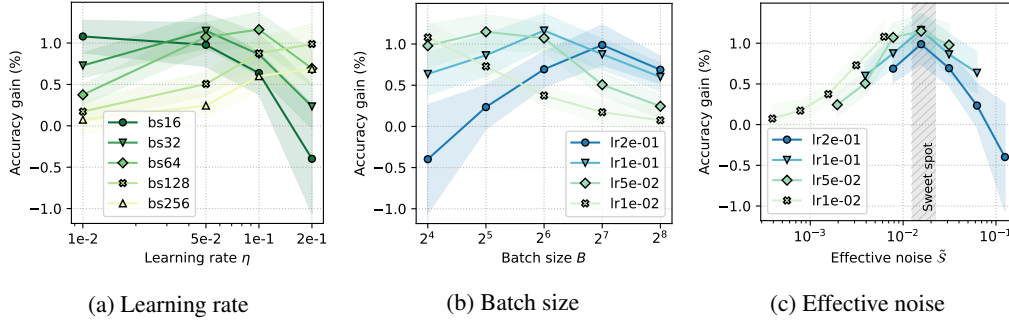


Figure 1: Effective noise scale controls the effectiveness of merging. The y-axis reports the test accuracy gain of merged models. On the x-axis, when plotting (a) batch sizes against learning rates or (b) vice versa, there is no clear trend. When reparameterized in terms of (c) effective noise scale, the curves are aligned, highlighting the interaction between different components for merging.

As introduced in Section 2, the effective noise scale $\tilde{\mathcal{S}}$ captures the joint interaction of learning rate, batch size, momentum, and augmentation of the stochasticity in SGD. Rather than treating these hyperparameters as independent, we study how $\tilde{\mathcal{S}}$ offers a unifying view on the compatibility of single models under linear interpolation merging. We use Resnet18 on CIFAR100 and sweep across different learning rates and batch sizes. The weight decay is fixed at $5e-4$, and the random flip and crop augmentation are used. To evaluate the effectiveness of merging, we define performance gain as $g(\theta_{\text{merge}}, \theta_{\text{single}}) = f(\theta_{\text{merge}}) - f(\theta_{\text{single}})$ given an evaluation function $f : \Theta \times \mathcal{D} \rightarrow \mathbb{R}$. Section A.1 describes the training and merging setup.

The results in Figure 1 illustrate how $\tilde{\mathcal{S}}$ affects the merging effectiveness. When we vary the learning rate or batch size independently, there is no clear trend across both dimensions at the same time. For example, when increasing the learning rate for a fixed batch size to $B = 16$, the accuracy gains monotonically decrease, whereas the opposite holds for a larger batch size of $B = 128$ or $B = 256$, which is commonly preferred in practice. However, once we represent the x-axis in terms of the normalized effective noise $\tilde{\mathcal{S}}$, capturing both learning rate and batch size together, the different curves become aligned. Importantly, this curve is *non-monotonic*: merging effectiveness improves as $\tilde{\mathcal{S}}$ increases from small values, reaches a “sweet spot”, and then degrades again once the noise grows too large.

In contrast to prior work that mainly links effective noise to properties of single solutions (Chaudhari et al., 2016; Mandt et al., 2017; Jastrzebski et al., 2017), our results show that it also governs

its surrounding solutions. Specifically, $\tilde{\mathcal{S}}$ not only biases SGD toward particular regions of the loss landscape, but also controls the compatibility of solutions found in different runs under linear interpolation merging. In the following sections, we ablate each optimizer component and analyze how it contributes to the merging effectiveness.

3.2 LARGE LEARNING RATE PRODUCES MORE COMPATIBLE SOLUTIONS

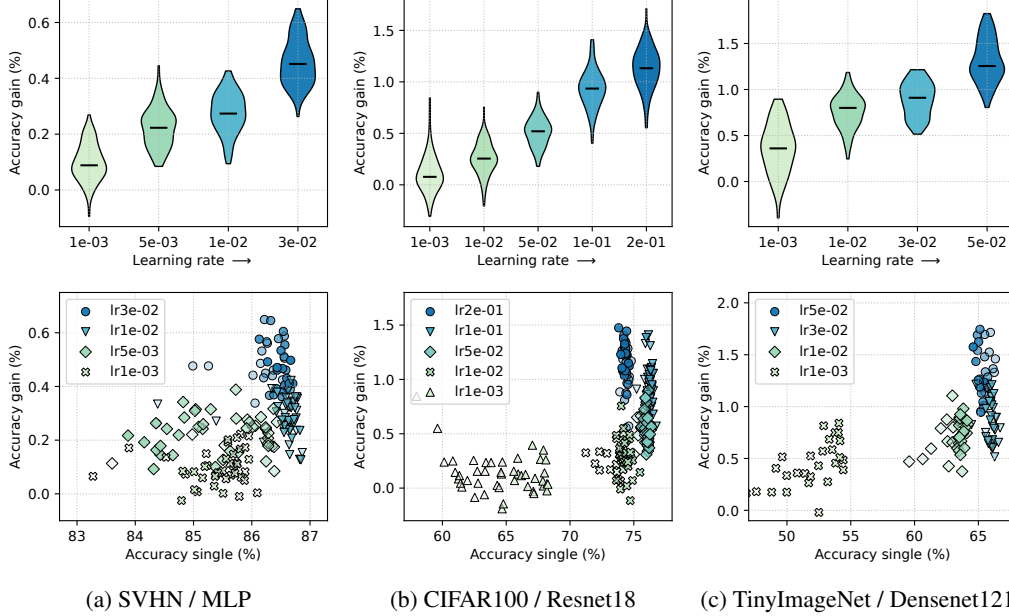


Figure 2: Larger learning rate leads to more effective merging. (top) The test accuracy gain of all the models. (bottom) Each point represents a single model accuracy on the x -axis and its accuracy gain after merging on the y -axis. The opacity indicates the number of training epochs. For each setup, we observe that a larger learning rates have a higher accuracy gain, even when there is a smaller learning rate with equivalent single model accuracy. Note, however, solutions found using a “too large” learning rate fail to merge (details in Section B.6).

We present empirical results for deep neural networks on vision tasks trained from scratch. The architectures used are a simple MLP, Resnet18, Densenet121 trained on SVHN, CIFAR10, CIFAR100, and TinyImageNet. Weight decay is fixed at $5e-4$, batch size of 128, and random flip and crop augmentation are used across the setups. Section A.1 describes the training and merging setup.

The results in Figure 2 show the key finding of our work. Using linear interpolation with a fixed $\alpha = 0.5$, we merge two solutions with the same learning rates at each checkpoint. We observe that the solutions identified with a larger learning rate are consistently more compatible to merge than those of a smaller learning rate. For example, in CIFAR100, the solutions found using an $\text{lr} = 0.2$ achieve $+1.2\%$ of the median gain compared to a $+0.2\%$ gain of $\text{lr} = 0.01$, despite having a similar performance for the single model of $\approx 75\%$ (x -axis). The same phenomenon is observed across different datasets (SVHN, CIFAR, and TinyImageNet) and architectures (MLP, Resnet, and Densenet). Furthermore, we also ensure that all the solutions are well-converged by asserting that the training loss is near-zero (details in Section B.5). As argued by Pascanu et al. (2025), it is important to understand how the implicit bias of the optimizer alters the final solutions and how to leverage this bias to find better solutions. Furthermore, recent works found different implicit biases of training with a larger learning rate, such as a sparser activation (Andriushchenko et al., 2023b; Sadrtadinov et al., 2024), a different order for feature learning order (Li et al., 2019), and a flatter solution (Andriushchenko et al., 2023a). Our results demonstrate an additional benefit: larger learning rate has an implicit bias on the loss landscape, identifying more compatible solutions for model merging. Note, however, Appendix Figure 15 shows that a too large learning rate leads to instabilities or failures in model merging.

3.3 WEIGHT DECAY AND EFFECTIVE LEARNING RATE

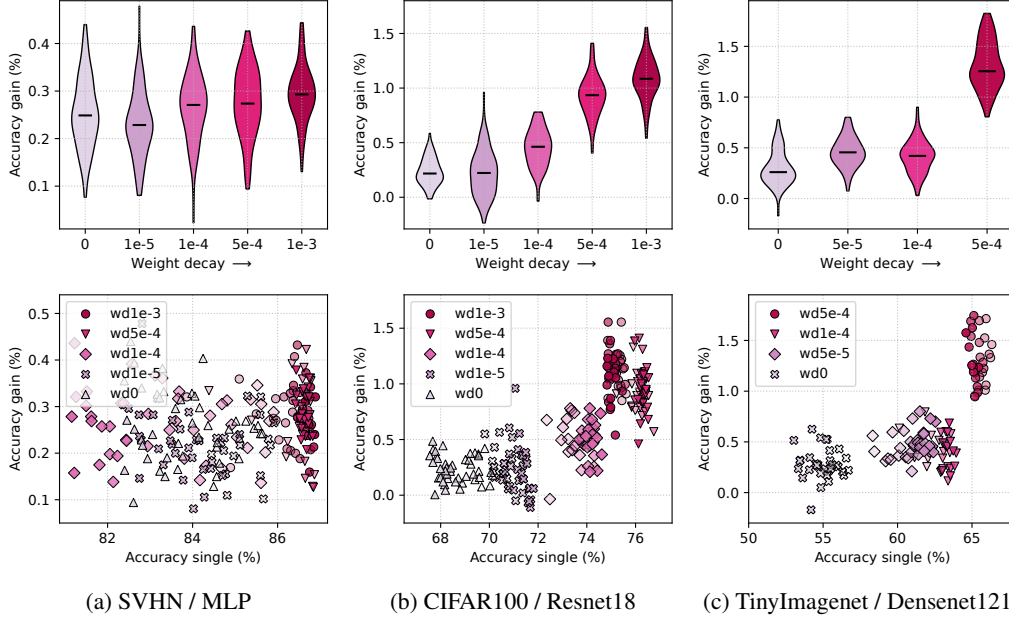


Figure 3: Weight decay has a similar effect as the learning rate. For CIFAR100 and TinyImagenet, we use scale-invariant networks (w/ normalization layers) and observe that a larger weight decay can not only improve the accuracy of the single model, but also the accuracy gain via the *effective learning rate* (Van Laarhoven, 2017). For MLP trained on SVHN, there is no trend as the architecture is not scale-invariant.

The traditional understanding of the role of weight decay regularization is that it reduces overfitting by proportionally decaying the weights towards zero, favouring less “complex” models. In practice, this is achieved by adding a penalty term $\lambda \|\theta\|_2^2$ to the objective $\mathcal{L}(\theta)$. However, modern neural network architectures ubiquitously use normalization layers (Ioffe & Szegedy, 2015; Ba et al., 2016) and are therefore weight scale-invariant. The output is invariant to the scale of the weights as $f(x, \alpha\theta) = f(x, \theta)$. Then, what is the new role of weight decay regularization in scale-invariant networks? Van Laarhoven (2017); Hoffer et al. (2018) answer this question by demonstrating that weight decay controls the *effective learning rate*, and thus the optimization noise scale, during training. In practice, without any weight constraints, scale-invariant networks will decay the learning rate over time, hindering the learning process.

We hypothesize that weight decay has a similar effect to the learning rate in model merging. That is, solutions found with a larger weight decay are easier to merge than those found with a smaller or no weight decay. Note that this should hold only for scale-invariant networks. We use the same experimental setup as in Section 3.2, except that we now sweep across different weight decay values instead of learning rates.

The results in Figure 3 confirm our hypothesis about the implicit bias of weight decay. Larger weight decay increases the *effective learning rate* for scale-invariant networks during training, affecting also the model merging effectiveness. For example, in TinyImagenet, the solutions found using a $wd = 0.0005$ report +1.2% of the median gain compared to a +0.5% of other wd values. The same trend is observed in other settings. Furthermore, interestingly, for the architecture MLP trained on SVHN, larger weight decay does not differ from smaller ones in terms of merging effectiveness. This validates that weight decay affects model merging only in scale-invariant architectures. Our results extend how the weight decay affects the loss landscape of an individual minima (Van Laarhoven, 2017; D’Angelo et al., 2024) to its connection with other minima. Lastly, similarly to the learning rate, Appendix Figure 16 shows that a too large weight decay leads to failure in model performance and model merging.

3.4 BATCH SIZE AND DATA AUGMENTATION

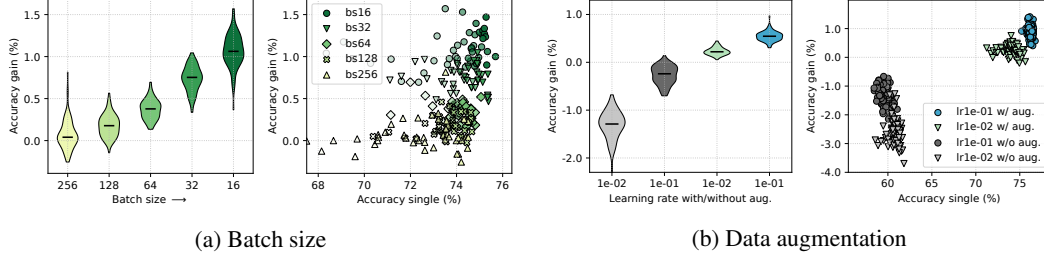


Figure 4: Batch size and data augmentation control the noise during the optimization dynamics. (left) A smaller batch size has more noise and improves the merging effectiveness. (right) Data augmentation improves performance and retains merging properties.

We now consider two additional components that affect the effective noise scale: batch size and data augmentation. For batch size, we use the same setup as in Section 3.2, except that we sweep across different batch sizes and fix the training steps to 200k steps instead of using epochs. For data augmentation, we simply turn off the augmentation during the training phase. The experimental setup uses CIFAR100 with Resnet18. Further results can be found in Section B.1 and Section B.2.

Batch size. The stochastic gradient of a minibatch $\hat{g}$ is unbiased but its variance scales as $Var(\hat{g}) \propto \sigma^2/B$, where σ^2 is the per-sample variance and B the batch size. Prior work has emphasized that this inverse scaling underlies the implicit regularization of SGD: smaller batch sizes inject more gradient noise, often leading to flatter solutions and better generalization (Jastrzebski et al., 2017; Smith & Le, 2018; Keskar et al., 2016). Our results in Figure 4 (left) extend this observation to model merging. We find that solutions obtained with smaller batch sizes are more compatible under linear interpolation: the smallest setup with $B = 16$ achieves a median accuracy gain of +1%, while a larger setup with $B = 256$ yields almost no benefit. Thus, in addition to generalization, batch-size-induced noise also improves the merging effectiveness of independently trained models.

Data augmentation. Data augmentation can likewise be viewed as injecting stochasticity into the optimization process: by applying random transformations to the data, the effective gradient covariance Σ_A changes, introducing additional variance beyond minibatch sampling. Previous work has argued that augmentation acts as a form of implicit regularization and invariance enforcement (Hernández-García & König, 2018; Yun et al., 2019), with recent perspectives interpreting augmentation as an additional source of optimization noise (Hanin & Sun, 2021). Our results in Figure 4 (right) show that augmentation not only improves single-model accuracy but also retains merging effectiveness. Interestingly, even without augmentation, a sufficiently large learning rate can yield positive merging gains, though this effect is not universal (Figure 10). Overall, augmentation-induced noise complements the minibatch noise and learning-rate noise, shaping solutions that are both stronger individually and compatible when merged.

3.5 WHAT ABOUT LANGUAGE MODELING?

Now we consider a language modeling task using the TinyStories dataset (Eldan & Li, 2023). We train a small GPT Transformer model with two layers using the AdamW optimizer with a constant learning rate for 200k steps and save a checkpoint every 2k steps, following the setup at Section A.1. Two endpoint models are trained for an additional 20k steps using a decayed learning rate scheduler. We use the loss performance gain to quantify the merging process. The lower the loss gain, the easier the merging process.

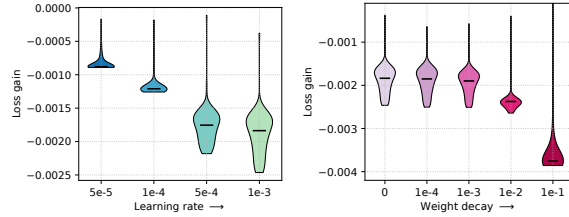


Figure 5: Larger learning rate and weight decay enable more effective merging in language modeling. (left) A larger learning rate has a better loss gain. (right) Adding a larger weight decay offers further merging gains. Section B.4 shows the scatter plots.

The results in Figure 5 extend our previous findings to the language domain. For the learning rate experimental setup, we fix $wd = 0$. Figure 13 (left) shows that a larger learning rate, such as $lr = 0.001$, requires fewer steps to reach a loss value of 2.20 compared to $lr = 0.0001$. Not only that, Figure 5 shows that a larger learning rate also has the implicit bias of simplifying the merging process, as measured by a lower loss gain. This behaviour is similar to the results from vision in Section 3.2. When weight decay regularization is added to the equation, there are further merging benefits. We fix the $lr = 0.001$ and sweep across weight decay. Figure 5 (right) shows that a larger weight decay leads to a better loss gain than smaller ones. Specifically, the largest weight decay, such as $wd = 0.1$, has the best loss gain, but also has a slower convergence (x -axis). The second largest weight decay of $wd = 0.01$ has a similar convergence speed as the smaller one, in addition to better loss gain. Lastly, smaller values have similar results to training without weight decay.

3.6 WHAT ABOUT TRANSFER LEARNING?

In the previous sections, we analyzed settings where models are trained from scratch. Now we consider setups involving transfer learning. We finetuned only the vision encoder of the pretrained model CLIP ViT-B/16 (Radford et al., 2021) on the WILDS-FMoW (Koh et al., 2021) dataset using the AdamW optimizer with cosine scheduler. Since varying the learning rate changes the speed of convergence, we carefully tune the number of training epochs for each setup and ensure proper convergence (details in Section A.2). We train three seeds for each setup and merge each different pair, obtaining three different merged models per learning rate.

The results on the left of the Figure 6 shows that a larger learning rate identifies solutions that are easier to merge. Specifically, the smallest values lie in a flatter loss landscape region where the performance gain is $4\times$ smaller than the largest learning rate when merged. The Pearson correlation coefficient is $r = 0.981$, indicating an almost perfect linear correlation between accuracy gain and learning rate. Note, however, that one should not blindly use the largest learning rate. Figure 6 on the right shows that the merged models with the best performance are the one with a moderate learning rate, as also observed by (Wortsman et al., 2022). The largest learning rate setup has the largest accuracy gain, but the worst-performing single model. Section B.3 presents similar results using different datasets and a pretrained model.

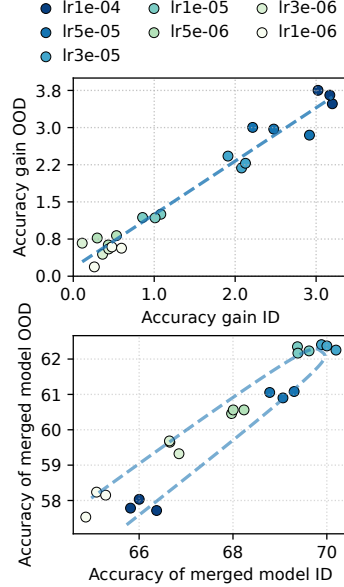


Figure 6: Merging effectiveness in transfer learning for ID and OOD data. (top) Accuracy gain linearly correlates with learning rate measured via Pearson correlation $r = 0.981$. (bottom) However, a larger learning rate leads to a suboptimal merged model, despite having the largest accuracy gain.

4 THE OPTIMIZER’S IMPLICIT BIAS ON TASK ARITHMETIC

In the previous section, we have seen how the optimizer implicitly biases the loss landscape of linear interpolation merging. We now consider task arithmetic interpolation, which defines a different subspace of solutions. This section studies how the principal optimizer component, the learning rate, affects the loss landscape of task arithmetic merging.

4.1 LOSS LANDSCAPE OF TASK ARITHMETIC

So far, we have only considered merging using linear interpolation (see Equation (1)). Task arithmetic interpolates two models along a different subspace compared to linear interpolation, identifying functionally different solutions. We apply task arithmetic interpolation to two settings:

- (a) Models w/ pretraining weight from Section 3.6 (i.e. pretraining dataset is different from the finetuning dataset). Task arithmetic is applied to a base model and two task vectors.

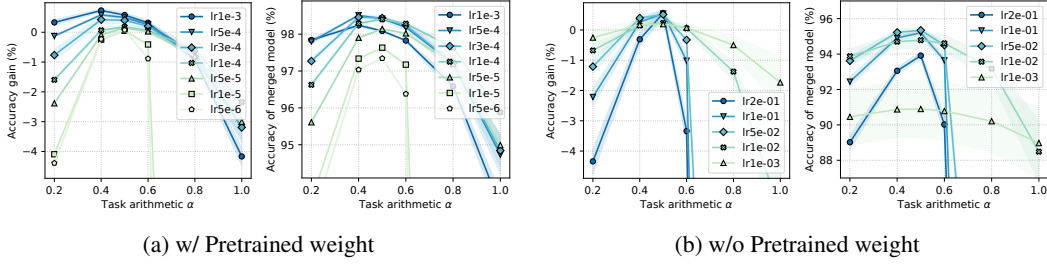


Figure 7: Task arithmetic loss landscape drastically changes depending on the initialization model. (a) With a pretrained initialization on ImageNet, larger learning rate solutions have higher gain and are more robust to task arithmetic interpolation. (b) Without a pretrained weight, a larger learning rate solution lies in a sharper minima (i.e. more sensitive to α changes).

The base model θ_{base} is the pretrained model CLIP ViT-B/16, and the task vectors are the finetuned models with different random seeds.

- (b) Models w/o pretraining weight from Section 3.2 (i.e. pretraining dataset is the same as the finetuning dataset). Task arithmetic is applied to a base model and two task vectors. For each checkpoint, the base model θ_{base} is the checkpoint itself, and the task vectors are obtained from the endpoint models $\tau_A = \theta_A - \theta_{base}$ and $\tau_B = \theta_B - \theta_{base}$.

For each learning rate setup, we traverse the subspace defined by the task arithmetic interpolation by changing the coefficient α . This measures the performance change as a function of α , which can also be seen as a measure of landscape flatness. For simplicity, we use the same α for the two task vectors when applying task arithmetic.

The results in Figure 7 show the robustness of each learning rate to task arithmetic interpolation for CIFAR10. There is a clear dichotomy between the two settings. In setting (a), a larger learning rate identifies merged solutions that are more robust to α -interpolation, corresponding to a flatter landscape (Andriushchenko et al., 2023a). However, in setting (b), the opposite is true. This highlights that a larger learning rate has to be used together with a suitable initialization to achieve a smoother and flatter landscape (Wortsman et al., 2022). Furthermore, as in linear interpolation merging, a “too large” learning rate becomes unstable. Section C.3 presents further results for different datasets.

4.2 LOSS LANDSCAPE OF MERGING DIFFERENT TASKS

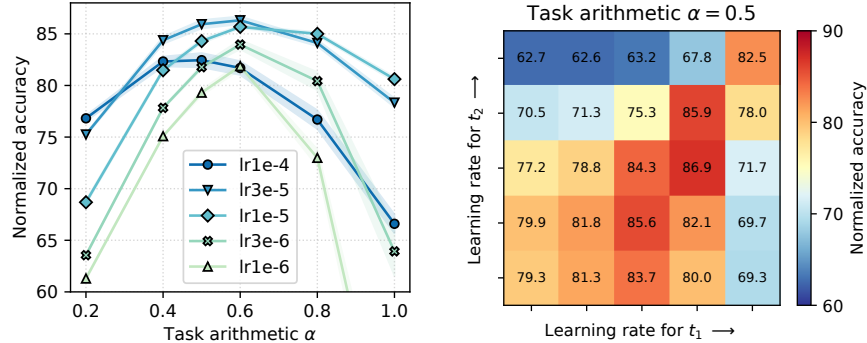


Figure 8: Task arithmetic merging of models trained on two different tasks. (left) The merged models are finetuned using the same learning rate. Larger learning rate solutions have better performance and are more robust to task arithmetic interpolation, unless it is too large. (right) The models are merged using different learning rates. Merging pairs of similar and larger learning rates has the best performance. Results are averaged over three seeds.

We now consider task arithmetic merging of two models sharing the same initialization θ_{base} finetuned on two different, but similar, tasks t . As in Section 3.6, we finetune one CLIP ViT-B/16 on

task t_1 WILDS-FMoW and another on task t_2 RESISC45 (Cheng et al., 2017). Then, task arithmetic merging is applied to merge the two models. To quantify the merging success, we use the averaged normalized accuracy, which measures the average ratio of the merged model performance over each single model performance (details in Section A.3).

The results in Figure 8 show how the learning rate affects the merging effectiveness of two models trained on two different tasks. In the Figure 8 on the left, as a proxy of the task arithmetic loss landscape, we merge and study the robustness of models finetuned using the same hyperparameters when interpolating α . We observe that the larger learning rate solutions perform better compared to the smaller ones (except for $\text{lr} = 0.0001$, which is the limit for stability). Moreover, larger values are also more robust to changes of α , representing flatter minima connecting the two different tasks. On the right of Figure 8, we merge models finetuned with different hyperparameters. The merged models with the *best performance are those merged with similar and moderately large learning rates* (near the antidiagonal). Merging models with a larger learning rate can result in better performance, but at the cost of losing flexibility for merging with other configurations. In particular, the largest learning rate $\text{lr} = 0.0001$ is the most unstable to merge with different learning rate models. Ilharco et al. (2023) also observed performance degradation when merging models trained with too large learning rates. Section C.4 reports additional results with further α values.

5 RELATED WORKS

Model merging. Early works on merging independently trained solutions on the same task can be found on mode connectivity (Garipov et al., 2018; Draxler et al., 2018). Linear mode connectivity has a stricter condition such that connecting paths are linear (Frankle et al., 2020; Neyshabur et al., 2020). When this is not possible, re-basin methods can be used to reparametrize the solution and restore the linear connectivity (Entezari et al., 2022; Ainsworth et al., 2023; Theus et al., 2025). Built upon these results, model merging methods have been developed to increase the performance on a single task (Wortsman et al., 2022) or to combine models trained on different tasks into one (Matena & Raffel, 2022; Ilharco et al., 2023). Yadav et al. (2025) provides a comprehensive survey of the latest merging methods.

Optimization dynamics. Standard optimization theory (Garrigos & Gower, 2023) shows that both batch sizes and learning rates drastically affect stability and convergence properties of SGD. In particular, through an analysis of SGD’s stationary distribution on simple quadratic potentials (Jastrzebski et al., 2017), it is possible to evince that, for single model training, the loss statistics at convergence only depend on the ratio between batch size and learning rates – as also validated empirically by Smith et al. (2020). In turn, either high learning rates or low batch sizes are known to favor flat minima (Keskar et al., 2016). While for more sophisticated optimizers, correlations between batch size, learning rates, and generalization might be more complex (Zhang et al., 2019; Malladi et al., 2022), other factors might more severely affect simple relations, such as non-Gaussianity (Simsekli et al., 2019) of gradient noise and non-convexity (Xie et al., 2021).

6 CONCLUSION

We study how optimizer choices implicitly shape the model-merging loss landscape and highlight the *effective noise scale* as a unifying factor. Learning rate, weight decay, batch size, and data augmentation all modulate this noise, which in turn determines whether independently trained solutions are compatible for merging. The relationship is non-monotonic – too little noise yields incompatible solutions, too much destabilizes training, but an intermediate “sweet spot” enables effective merging. These results extend prior works connecting noise to flatness and generalization of individual models, showing that noise also shapes the compatibility across independent solutions. Future work should explore how to further tune effective noise, architectural choices, and pretraining strategies to train models that are not only strong individually but also inherently mergeable. To summarize our contributions in one sentence: *tune the noise to tune merging effectiveness*.

Limitations. No new theoretical guarantees are developed, and no truly large-scale experiments are conducted due to our limited computational resources. We studied the standard merging methods, that form the foundation of state-of-the-art approaches. Our goal was to use a set of *simple, diverse, but realistic* experimental setups to understand the role of optimization in model merging.

ACKNOWLEDGEMENTS

We thank Christopher Gadzinski and Razvan Pascanu for the helpful discussions. Chenxiang Zhang is supported by the University of Luxembourg using ULHPC¹ and MeluXina² computing clusters. Alexander Theus acknowledges the financial support from the Max Planck ETH Center for Learning Systems. Antonio Orvieto is supported by the Hector Foundation.

REFERENCES

- Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. Git re-basin: Merging models modulo permutation symmetries. In *International Conference on Learning Representations*, 2023.
- Maksym Andriushchenko, Francesco Croce, Maximilian Müller, Matthias Hein, and Nicolas Flammarion. A modern look at the relationship between sharpness and generalization. In *International Conference on Machine Learning*, 2023a.
- Maksym Andriushchenko, Aditya Vardhan Varre, Loucas Pillaud-Vivien, and Nicolas Flammarion. Sgd with large step sizes learns sparse features. In *International Conference on Machine Learning*, 2023b.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- Pratik Chaudhari, Anna Choromańska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Tour Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-sgd: biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2016.
- Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017.
- Francesco D’Angelo, Maksym Andriushchenko, Aditya Vardhan Varre, and Nicolas Flammarion. Why do we need weight decay in modern deep learning? In *Advances in Neural Information Processing Systems*, 2024.
- Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. Essentially no barriers in neural network energy landscape. In *International Conference on Machine Learning*, 2018.
- Ronen Eldan and Yuanzhi Li. Tinstories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*, 2023.
- Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. The role of permutation invariance in linear mode connectivity of neural networks. In *International Conference on Learning Representations*, 2022.
- Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, 2020.
- Timur Garipov, Pavel Izmailov, Dmitrii Podoprikin, Dmitry P Vetrov, and Andrew G Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, 2018.
- Guillaume Garrigos and Robert M Gower. Handbook of convergence theorems for (stochastic) gradient methods. *arXiv preprint arXiv:2301.11235*, 2023.
- Alexander Hägele, Elie Bakouch, Atli Kosson, Loubna Ben Allal, Leandro Von Werra, and Martin Jaggi. Scaling laws and compute-optimal training beyond fixed training durations. In *Advances in Neural Information Processing Systems*, 2024.

¹<https://hpc-docs.uni.lu/>

²<https://www.luxprovide.lu/meluxina/>

- Boris Hanin and Yi Sun. How data augmentation affects optimization for linear regression. In *Advances in Neural Information Processing Systems*, 2021.
- Alex Hernández-García and Peter König. Data augmentation instead of explicit regularization. *arXiv preprint arXiv:1806.03852*, 2018.
- Elad Hoffer, Ron Banner, Itay Golan, and Daniel Soudry. Norm matters: efficient and accurate normalization schemes in deep networks. In *Advances in Neural Information Processing Systems*, 2018.
- Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024.
- Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *International Conference on Learning Representations*, 2023.
- Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015.
- Stanislaw Jastrzebski, Zachary Kenton, Devansh Arpit, Nicolas Ballas, Asja Fischer, Yoshua Bengio, and Amos Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.
- Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.
- Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Bal-subramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, 2021.
- Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems*, 2019.
- Sadhika Malladi, Kaifeng Lyu, Abhishek Panigrahi, and Sanjeev Arora. On the sdes and scaling rules for adaptive gradient algorithms. *Advances in Neural Information Processing Systems*, 2022.
- Stephan Mandt, Matthew D Hoffman, and David M Blei. Stochastic gradient descent as approximate bayesian inference. *Journal of Machine Learning Research*, 2017.
- Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 2022.
- Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*, 2018.
- Behnam Neyshabur, Hanie Sedghi, and Chiyuan Zhang. What is being transferred in transfer learning? In *Advances in Neural Information Processing Systems*, 2020.
- Razvan Pascanu, Clare Lyle, Ionut-Vlad Modoranu, Naima Elosegui Borrás, Dan Alistarh, Petar Velickovic, Sarath Chandar, Soham De, and James Martens. Optimizers qualitatively alter solutions and we should leverage this. *arXiv preprint arXiv:2507.12224*, 2025.
- Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021.
- Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do cifar-10 classifiers generalize to cifar-10? *arXiv preprint arXiv:1806.00451*, 2018.

- Ildus Sadrtudinov, Maxim Kodryan, Eduard Pokonechny, Ekaterina Lobacheva, and Dmitry P Vetrov. Where do large learning rates lead us? In *Advances in Neural Information Processing Systems*, 2024.
- Umut Simsekli, Levent Sagun, and Mert Gurbuzbalaban. A tail-index analysis of stochastic gradient noise in deep neural networks. In *International Conference on Machine Learning*, 2019.
- Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *International Conference on Machine Learning*, 2020.
- Samuel L Smith and Quoc V Le. A bayesian perspective on generalization and stochastic gradient descent. In *International Conference on Learning Representations*, 2018.
- Samuel L Smith, Pieter-Jan Kindermans, Chris Ying, and Quoc V Le. Don’t decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018.
- Alexander Theus, Alessandro Cabodi, Sotiris Anagnostidis, Antonio Orvieto, Sidak Pal Singh, and Valentina Boeva. Generalized linear mode connectivity for transformers. In *Advances in Neural Information Processing Systems*, 2025.
- Twan Van Laarhoven. L2 regularization versus batch and weight normalization. *arXiv preprint arXiv:1706.05350*, 2017.
- Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, and Simon Kornblith. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022.
- Zeke Xie, Issei Sato, and Masashi Sugiyama. A diffusion theory for deep learning dynamics: Stochastic gradient descent exponentially favors flat minima. In *International Conference on Learning Representations*, 2021.
- Prateek Yadav, Tu Vu, Jonathan Lai, Alexandra Chronopoulou, Manaal Faruqui, Mohit Bansal, and Tsendsuren Munkhdalai. What matters for model merging at scale? *arXiv preprint arXiv:2410.03617*, 2024.
- Prateek Yadav, Colin Raffel, Mohammed Muqeeth, Lucas Caccia, Haokun Liu, Tianlong Chen, Mohit Bansal, Leshem Choshen, and Alessandro Sordoni. A survey on model moerging: Recycling and routing among specialized experts for collaborative learning. *Transactions on Machine Learning Research*, 2025.
- Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *International Conference on Computer Vision*, 2019.
- Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Computer Vision and Pattern Recognition*, 2022.
- Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George Dahl, Chris Shallue, and Roger B Grosse. Which algorithmic choices matter at which batch sizes? insights from a noisy quadratic model. *Advances in neural information processing systems*, 32, 2019.

A DETAILED EXPERIMENT SETTING

A.1 TRAINING AND MERGING SETUP

For Section 3.2, Section 3.3, Section 3.4, and Section 3.5, we use the following training setup.

We use the warmup-stable-decay (WSD) scheduler (Zhai et al., 2022; Hu et al., 2024). We use the square root decay as in Hägele et al. (2024). Given a single configuration (e.g. $\text{lr} = 0.1$), we use a constant learning rate to train a model for T_{stable} epochs, saving a checkpoint θ_i every i epochs. For each θ_i , we use a decay learning rate scheduler and continue the training for T_{decay} epochs, obtaining two final endpoint models $\theta_{i,A}$ and $\theta_{i,B}$. Finally, the merged model is a linear interpolation (Equation (1)) between $\theta_{i,A}$ and $\theta_{i,B}$ with $\alpha = 0.5$.

We provide an example. For the CIFAR100 task, we train a model using a constant learning rate for $T_{\text{stable}} = 2000$ epochs and save a checkpoint θ_i every $i = 20$ epochs. Then, for each checkpoint, we use a decay scheduler and create two endpoint models $\theta_{i,A}$ and $\theta_{i,B}$. This means that at the end, there will be $T_{\text{stable}}/i = 2000/20 = 100$ different merged models.

Note that, to account for the different magnitudes of settings (e.g. $\text{lr} = 0.1$ vs $\text{lr} = 0.01$), we use a T_{stable} of one order of magnitude larger than the standard setting to ensure convergence of single models. We use $T_{\text{stable}} = 2000$ for CIFAR10, CIFAR100, and SVHN and $T_{\text{stable}} = 1500$ for TinyImagenet. We use $T_{\text{decay}} = 30$ for CIFAR10 and CIFAR100, and $T_{\text{decay}} = 20$ for SVHN and TinyImagenet.

A.2 TRANSFER LEARNING EXPERIMENTAL SETUP

For Section 3.6 and Section B.3, we use the following training setup.

For CLIP ViT-B/16 finetuned on WILDS-FMoW, we discard the language model. We use the AdamW optimizer with a warmup-cosine learning rate scheduler. Since varying the learning rate changes the speed of convergence, we carefully tune the number of training epochs for each setup to ensure convergence (e.g. training loss = 0). The following hyperparams (epochs, lr) are used for each setup (20, 1e-4), (20, 5e-5), (20, 3e-5), (20, 1e-5), (30, 5e-6), (40, 3e-6), and (100, 1e-6).

For CLIP ViT-B/16 finetuned on RESISC45, we follow the above configuration. The following hyperparams are used (20, 1e-4), (20, 3e-5), (20, 1e-5), (20, 3e-6), and (20, 1e-6).

For ViT-S/16 pretrained on IN1k and finetuned on WILDS-FMoW, we use the AdamW optimizer with a warmup-cosine learning rate scheduler. The following hyperparams are used (20, 1e-3), (20, 3e-4), (20, 1e-4), (40, 3e-5), and (100, 1e-5).

For ConvNext-T pretrained on IN1k and finetuned on CIFAR10, we use the AdamW optimizer with a warmup-cosine learning rate scheduler. The following hyperparams are used (20, 1e-3), (20, 5e-4), (20, 3e-4), (40, 1e-4), (40, 5e-5), (80, 1e-5), and (80, 5e-6).

Note that, for each setup, we have grid searched and used the largest learning rate possible. This means that an even larger learning rate fails to converge.

A.3 DETAILS ON METRICS

Normalized accuracy compares the relative performance metric of the multi-task model to that of single finetuned models:

$$\text{accuracy}_{\text{norm}} = \frac{1}{T} \sum_{i=1}^T \frac{\text{accuracy}(\theta_M)}{\text{accuracy}(\theta_i)}$$

where T is the total number of tasks, θ_M represents the multi-task model and θ_i is the single finetuned model for the task t_i . This metric compares the baseline performance against each task.

B ADDITIONAL RESULTS FOR LINEAR INTERPOLATION MERGING

B.1 DATASET: CIFAR10

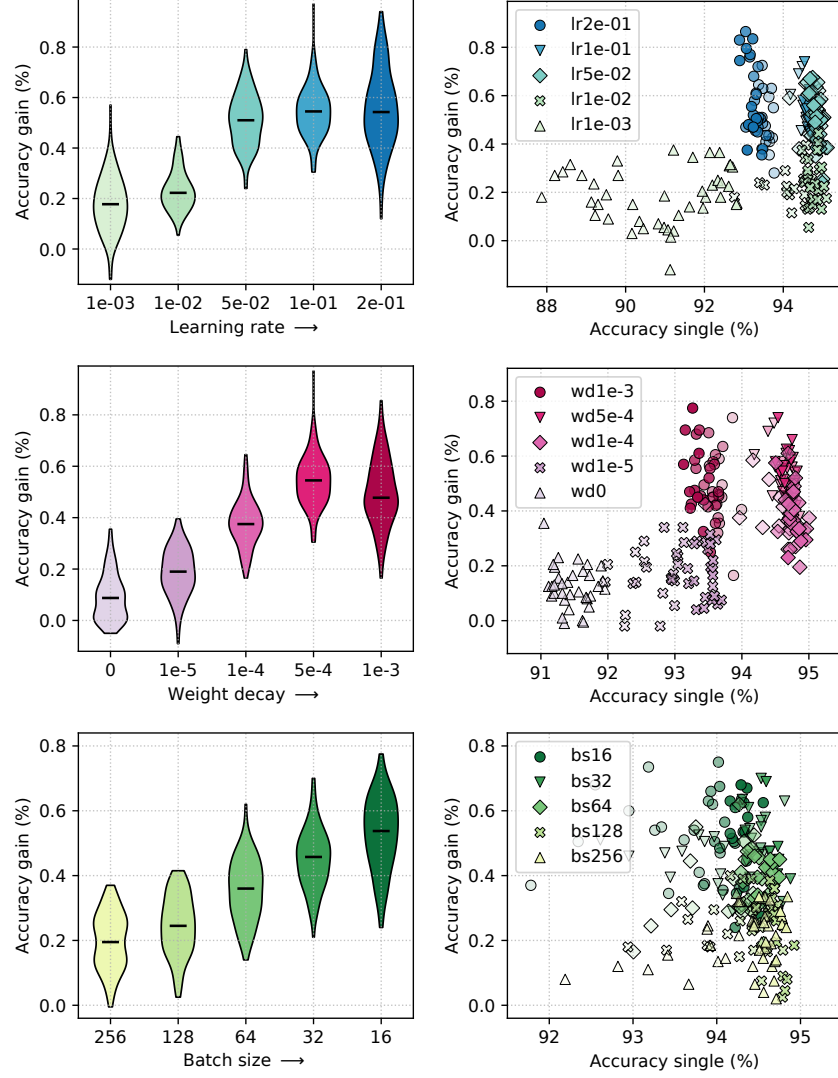


Figure 9: Larger learning rate / larger weight decay / smaller batch size all lead to a larger performance gain in CIFAR10 dataset.

B.2 DATA AUGMENTATION: SVHN, CIFAR10, TINYIMAGENET

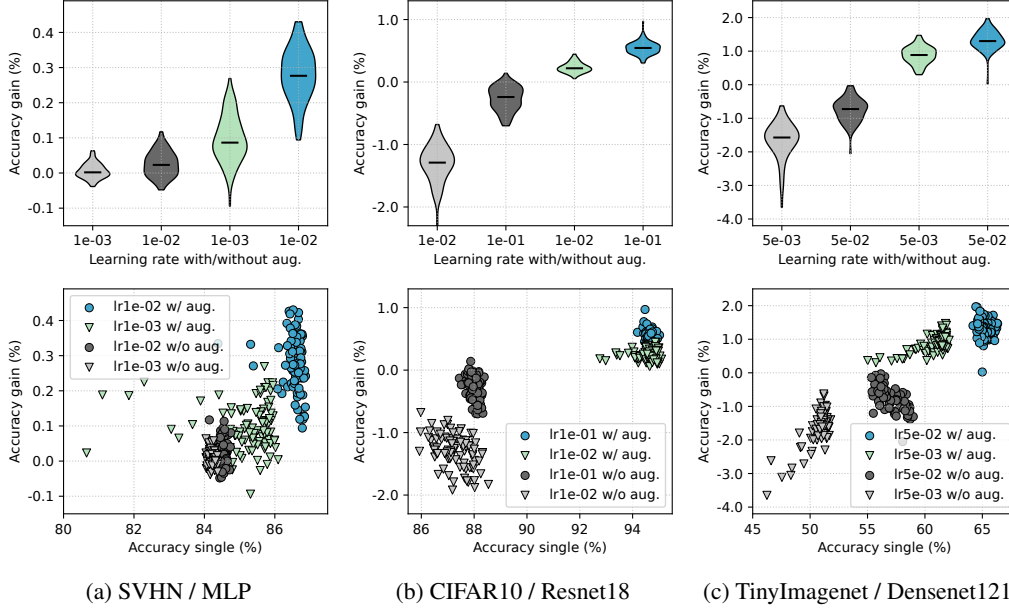


Figure 10: Accuracy gain and data augmentation. The merging fails w/o augmentation. However, a larger learning rate remains easier to merge than a smaller one.

B.3 TRANSFER LEARNING: ViT, CONVNEXT-T

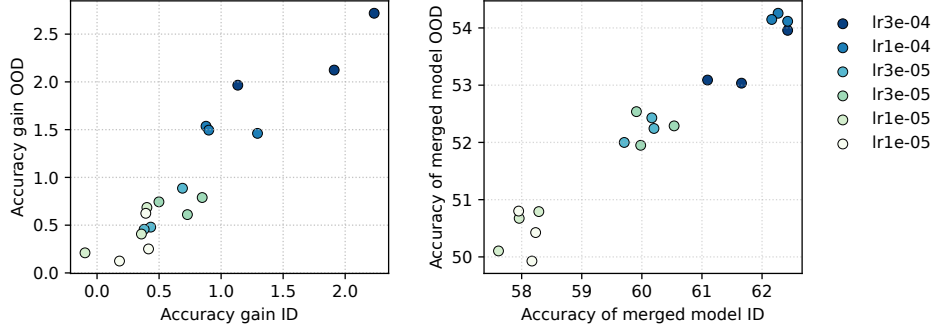


Figure 11: Larger learning rate enables easier merging under transfer learning for both ID and OOD datasets. The pretrained architecture is ViT trained on IN1k and finetuned on FMoW. The evaluation is done on the test set ID and OOD splits.

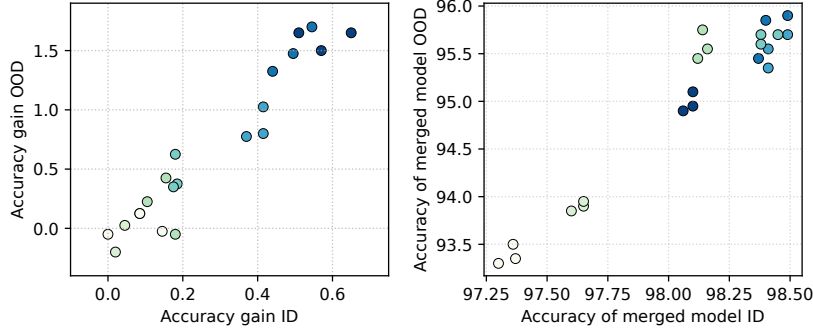


Figure 12: Larger learning rate enables easier merging under transfer learning for both ID and OOD datasets. The pretrained architecture is ConvNext-T trained on IN1k and finetuned on CIFAR10. The evaluation is done on the test set ID of CIFAR10 and test set OOD CIFAR10.1 (Recht et al., 2018).

B.4 LANGUAGE MODELING

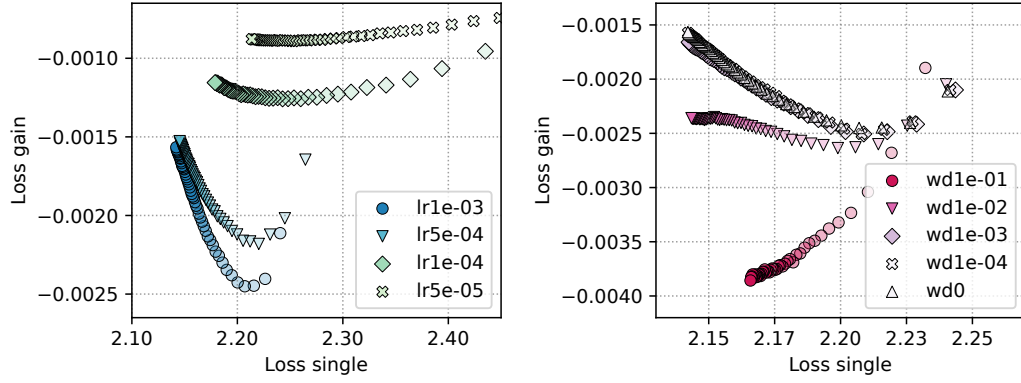


Figure 13: Larger learning rate and weight decay enable more effective merging in language modeling. (left) Different setups at loss single of ≈ 2.20 clearly differ in loss gain. (right) Similar phenomenon when tuning weight decay.

B.5 TRAINING LOSS OF DECAYED MODELS

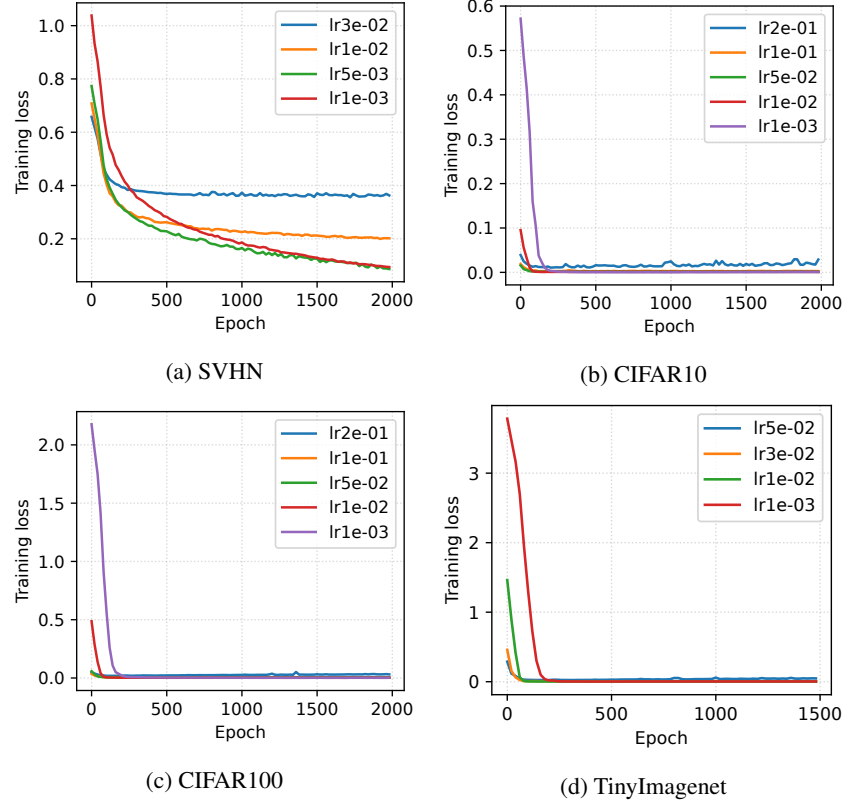


Figure 14: Training loss of decayed models from Section 3.2. For deep networks trained on CIFAR and TinyImageNet, we ensure that different setups reach near 0 training loss. For the simple MLP trained on SVHN, convergence to 0 training loss is slow. However, the largest learning rate $lr = 0.03$ has the highest accuracy model despite a larger loss.

B.6 MERGING FAILS DUE TO HIGH EFFECTIVE NOISE

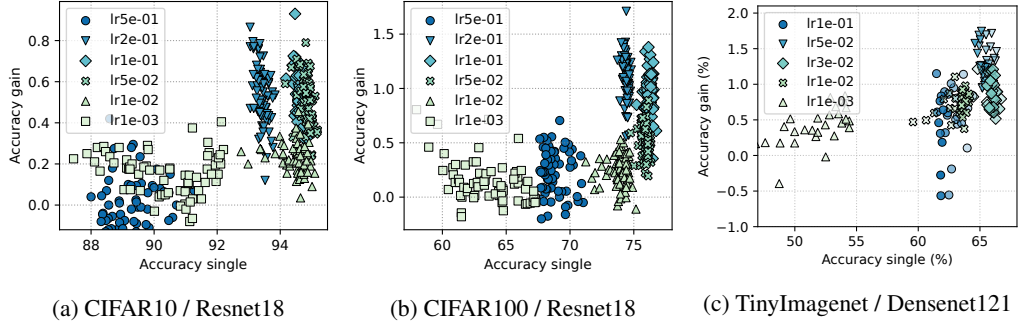


Figure 15: Too large learning rate causes instability/failure in merging.

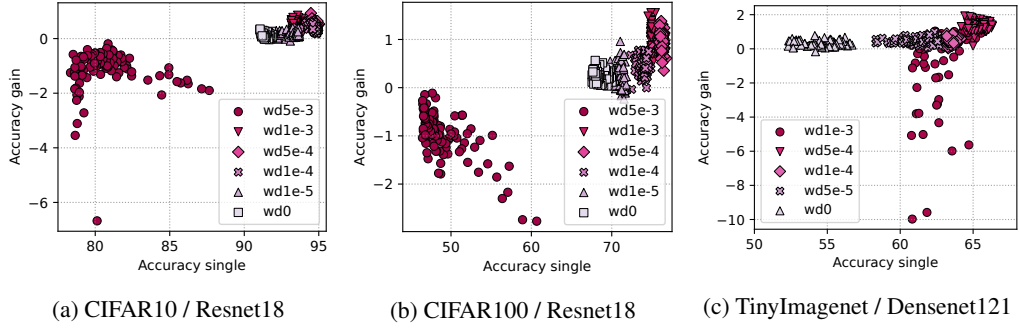


Figure 16: Too large weight decay causes instability/failure in merging.

C ADDITIONAL RESULTS FOR TASK ARITHMETIC

C.1 LEARNING RATE, WEIGHT DECAY

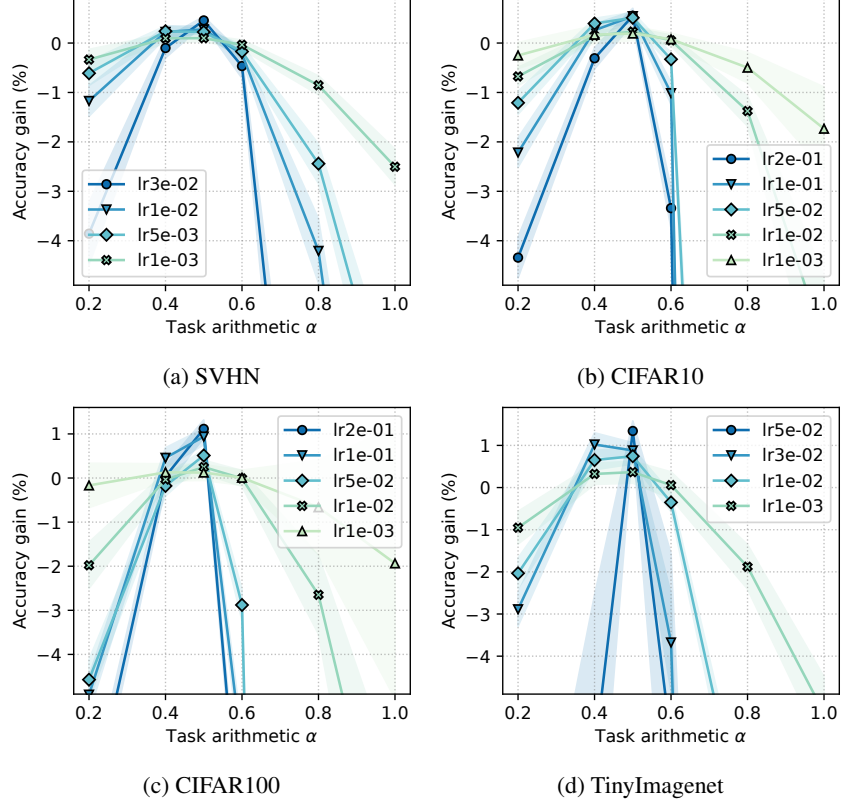


Figure 17: Task arithmetic interpolation robustness of models w/o Pretrained weight from the Section 3.2. In the absence of a pretrained weight, the largest learning rate is the least robust to task arithmetic interpolation.

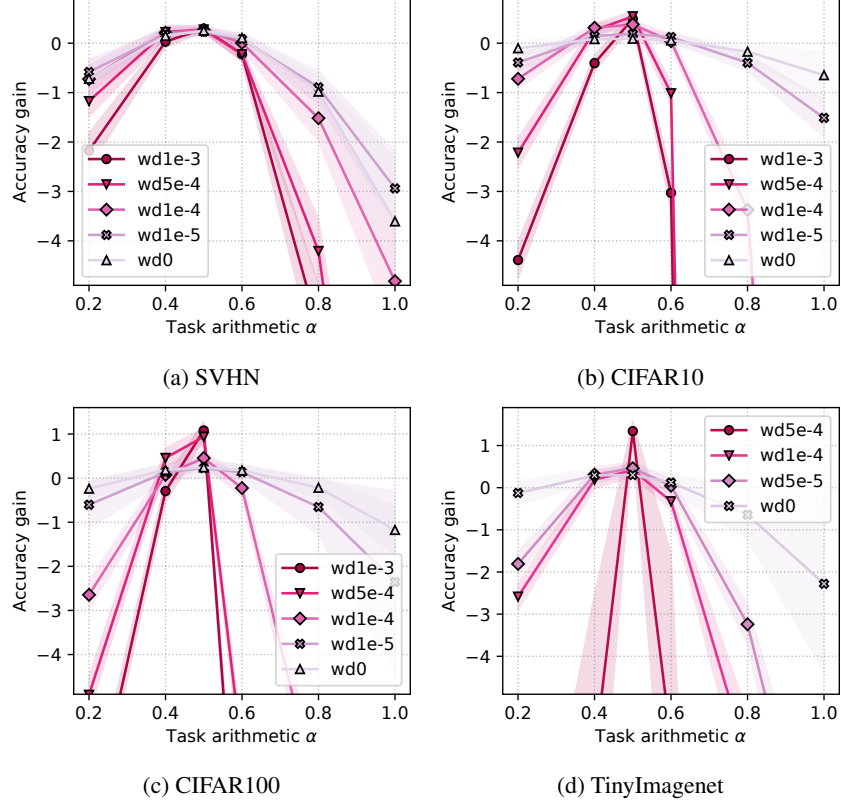


Figure 18: Task arithmetic interpolation robustness of models w/o Pretrained weight from the Section 3.3. In the absence of a pretrained weight, the largest weight decay is the least robust to task arithmetic interpolation.

C.2 LANGUAGE MODELING

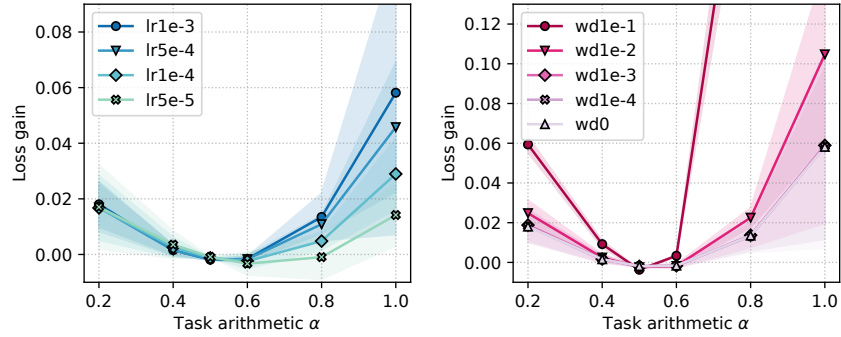


Figure 19: Task arithmetic loss gain in language modeling for a small GPT on the TinyStories dataset trained for 200k steps. In the absence of a pretrained weight, the largest learning rate/weight decay is the least robust to task arithmetic interpolation.

C.3 TRANSFER LEARNING: FMoW, RESISC45

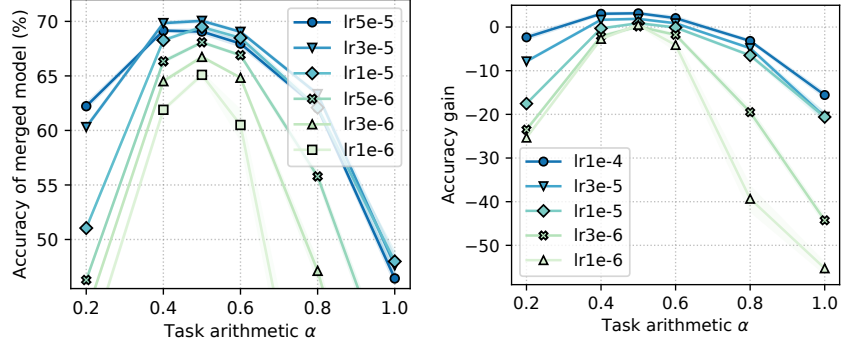


Figure 20: Task arithmetic robustness and gain for CLIP ViT-B/16 finetuned on FMoW.

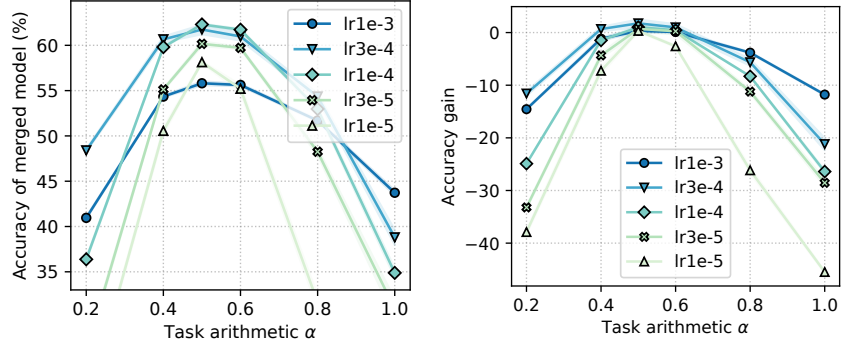


Figure 21: Task arithmetic robustness and gain for ViT-S/16 pretrained on IN1k finetuned on FMoW.

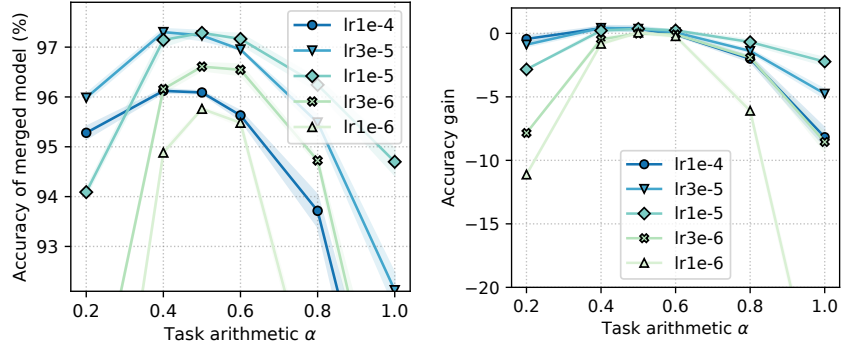


Figure 22: Task arithmetic robustness and gain for CLIP ViT-B/16 finetuned on RESISC45.

C.4 MERGING DIFFERENT TASKS

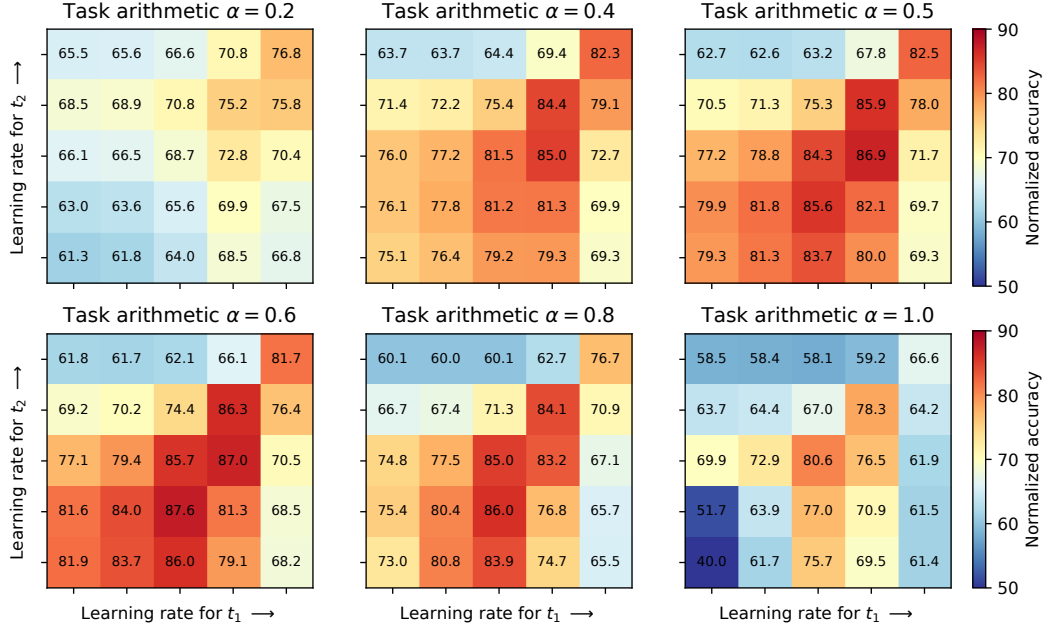


Figure 23: Task arithmetic merging of two different tasks across α values. Similar setups (antidiagonal) consistently have better merged models. Note that for $\alpha = 0.2$ smallest learning rate models do not merge well. Same for $\alpha = 1.0$, indicating a sharper minima defined by task arithmetic subspace, similar as Section 4.1.

D ADDITIONAL RESULTS ON LOSS LANDSCAPE

D.1 TRANSITION PHASE: HILLS, FLATLAND, AND VALLEYS

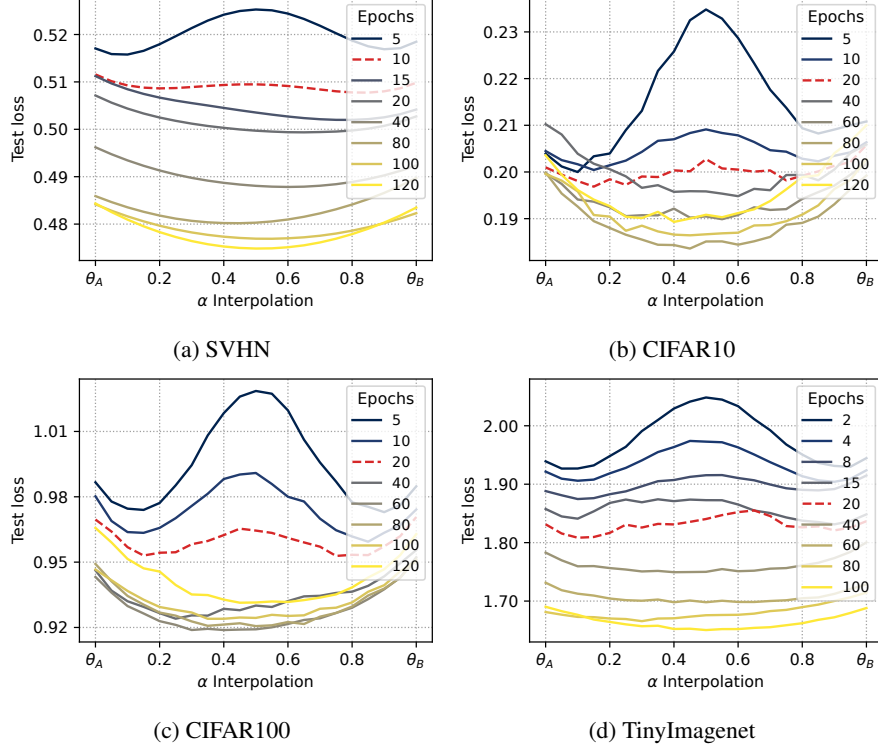


Figure 24: The loss geometry of the linear interpolation between two endpoints changes from a *hill* $\rightarrow$ *valley*, based on the timing of the bifurcation. Given a training budget T , the legend indicates the bifurcation start epoch T_a , which means the training continues for $T_b = T - T_a$ epochs with θ_A and θ_B . The transition phase (dashed line) marks the phase change from a hill into a valley.

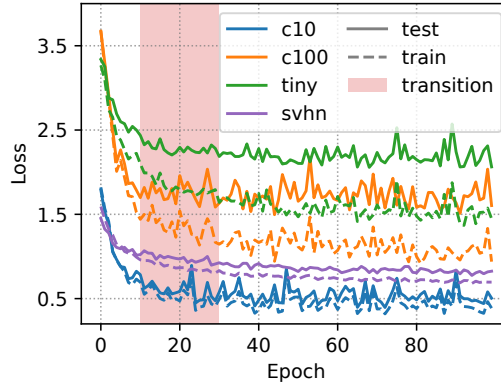


Figure 25: Identifying the transition phase from hill to valley.

D.2 FLATNESS

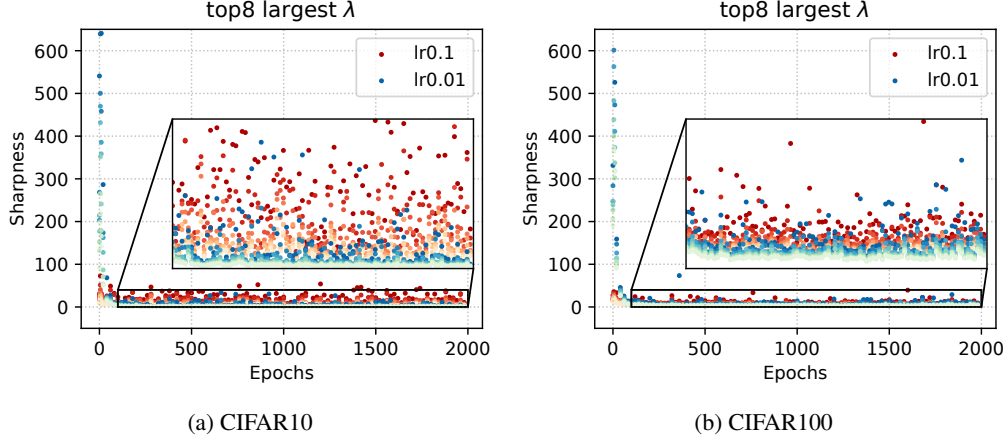


Figure 26: The flatness measured using the top-8 eigenvalues of the hessian. The larger learning rate solutions lie inside a sharper minima.

D.3 LANDSCAPE VS. EFFECTIVE NOISE

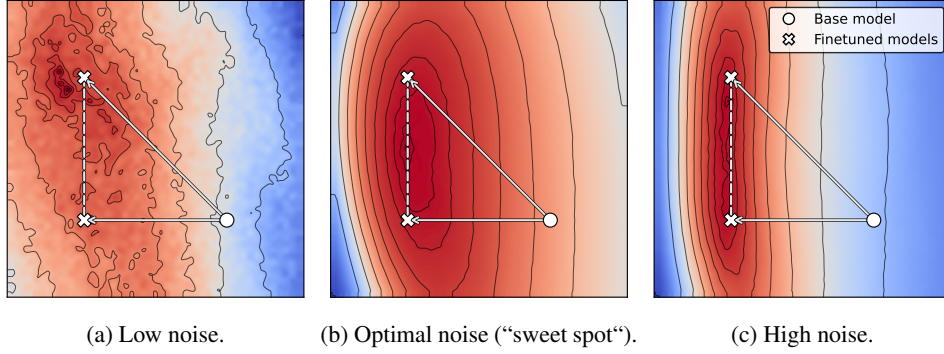


Figure 27: 2D loss slices in the plane spanned by the base model and two fine-tuned models under varying effective noise scales. Low and high noise both yield broad, flat corridors between the fine-tuned solutions, whereas an intermediate (optimal) noise level introduces performance gains ("valleys") between them. Model: ResNet18; dataset: CIFAR100.

E LLM RESEARCH ASSISTANCE

We use LLM to assist this research project in the following tasks: manuscript polishing and retrieval of related work. For both tasks, we make mild use of LLM for the manuscript writing phase. In particular, polishing has been used only to improve the flow of the sentences, while the retrieval of contents has been used to find a few related works.