When Do Flat Minima Optimizers Work?

Jean Kaddour*
Centre for Artificial Intelligence
University College London

Linqing Liu*
Centre for Artificial Intelligence
University College London

Ricardo Silva
Department of Statistical Science
University College London

Matt J. Kusner
Centre for Artificial Intelligence
University College London

Abstract

Recently, *flat-minima optimizers*, which seek to find parameters in low-loss neighborhoods, have been shown to improve a neural network's generalization performance over stochastic and adaptive gradient-based optimizers. Two methods have received significant attention due to their scalability: 1. Stochastic Weight Averaging (SWA), and 2. Sharpness-Aware Minimization (SAM). However, there has been limited investigation into their properties and no systematic benchmarking of them across different domains. We fill this gap here by comparing the loss surfaces of the models trained with each method and through broad benchmarking across computer vision, natural language processing, and graph representation learning tasks. We discover several surprising findings from these results, which we hope will help researchers further improve deep learning optimizers, and practitioners identify the right optimizer for their problem.

1 Introduction

Stochastic gradient descent (SGD) methods are central to neural network optimization [6]. Recently, one class of algorithms has focused on biasing SGD methods towards so-called '*flat*' minima, which are located in large weight space regions with very similar low loss values [41]. Theoretical and empirical studies [21, 72, 9, 50, 46, 5, 12] postulate that such flatter regions generalize better than sharper minima, e.g., due to the flat minimizer's robustness against loss function shifts between train and test data, as illustrated in Fig. 1. Two popular flat-minima optimization approaches are: 1. Stochastic Weight Averaging (SWA) [45], and 2. Sharpness-Aware Minimization (SAM) [22].

While both strategies aim to find flatter minima, they operate much differently. On the one hand, SWA is based on the intuition that, near a flat minimum, gradients are smaller, leaving many iterates in that flat region. Therefore, averaging iterates will produce a solution that is pulled towards these flatter regions, see Fig. 1, top. On the other hand, SAM minimizes the maximum loss around a neighborhood of the current iterate. This way, a region around the iterate is designed to have uniformly low loss; see Fig. 1, bottom. Crucially, SAM requires an additional forward/backward pass for each parameter update, making it more expensive than SWA.

Despite the successes [70, 3, 48, 12, 4] of SWA and SAM in some domains, we are unaware of a systematic comparison between them that would help practitioners to choose the right optimizer for their problem and researchers to develop better optimizers. The SWA [45] paper was published in 2018, and the SAM [22] paper in 2021; however, the SAM paper, and its most noticeable follow-ups [59, 12, 98], do not compare against SWA. Further, there is very limited overlap in terms of

*Equal contribution, correspondence to {jean.kaddour,linqing.liu}.20@ucl.ac.uk

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

the model architecture and dataset used in the experiments among both papers, which are likely further confounded by other differences in the training procedures (e.g. data augmentations, hyper-parameters, etc.).

Contributions

1. **In-depth comparison of minima found by SWA and SAM:** We visualize linear interpolations between different models and quantify the minimizers' flatnesses. This analysis yields 4 insights, e.g., despite SAM finding flatter solutions than SWA as quantified by Hessian eigenvalues, they can be close to sharp directions, a phenomenon that has been overlooked in the previous SAM literature. Averaging SAM iterates leads to the flattest among all minima.
2. **Rigorous comparison of SWA and SAM's performance over 42 tasks:** We empirically compare the optimizers with a rigorous model selection procedure on a broad range of tasks across different domains (CV, NLP, and GRL), model types (MLPs, CNNs, Transformers) and tasks (classification, self-supervised learning, open-domain question answering, natural language understanding, and node/graph/link property prediction). We discuss 8 findings, e.g., that both dataset and architecture impact their effectiveness, that for NLP tasks, SAM improves over SWA in most cases, and that the converse holds for GRL tasks. When flat-minima optimizers do not help, we notice clear discrepancies between the shapes of loss and accuracy curves. To assist future work, we open-source the code for all pipelines and hyper-parameters to reproduce the results.

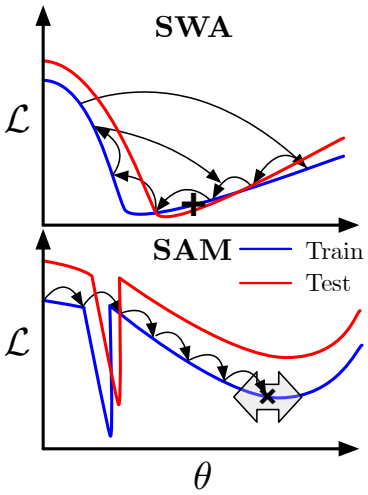


Figure 1: The mechanics behind SWA and SAM, whose solution is denoted by $+$ and $\times$, respectively. SWA produces a solution θ that is pulled towards flatter regions, while SAM approximates sharpness within the parameters' neighborhood (arrows).

2 Background and Related Work

2.1 Stochastic Gradient Descent (SGD)

The classic optimization framework of machine learning is empirical risk minimization

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{N}\sum_{i=1}^{N} \ell(\boldsymbol{x}_i; \boldsymbol{\theta}) \tag{1}$$

where $\boldsymbol{\theta} \in \mathbb{R}^d$ is a vector of parameters, $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ is a training set of inputs $\boldsymbol{x}_n \in \mathbb{R}^D$, and $\ell(\boldsymbol{x}; \boldsymbol{\theta})$ is a loss function quantifying the performance of parameters $\boldsymbol{\theta}$ on $\boldsymbol{x}$. SGD samples a minibatch $\mathcal{S} \subset \{1, \ldots, N\}$ of size $|\mathcal{S}| \ll N$ from the training set and updates the parameters through

$$\boldsymbol{\theta}_{t+1}^{\text{SGD}} = \boldsymbol{\theta}_t - \eta \boldsymbol{g}(\boldsymbol{\theta}_t), \text{ where } \boldsymbol{g}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{B}|}\sum_{i \in \mathcal{B}} \nabla \ell(\boldsymbol{\theta}; \boldsymbol{x}_i), \tag{2}$$

for a length specified by η, the learning rate.

2.2 Stochastic Weight Averaging (SWA)

The idea of averaging weights dates back to accelerating the convergence speed of SGD [73, 48]. SWA's motivation is based on the following observation about SGD's behavior when training neural networks: it often traverses regions of the weight space that correspond to high-performing models, but rarely reaches the central points of this optimal set. Averaging the parameter values over iterations moves the solution closer to the centroid of this space of points.

The SWA update rule is the cumulative moving average

$$\boldsymbol{\theta}_{t+1}^{\text{SWA}} \leftarrow \frac{\boldsymbol{\theta}_t^{\text{SWA}} \cdot l + \boldsymbol{\theta}_t^{\text{SGD}}}{l+1}, \tag{3}$$

Algorithm 1 Stochastic Weight Averaging [45]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, averaging start epoch E, averaging frequency ν, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.
for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t\right)$
  Update parameters $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \eta\boldsymbol{g}$
  if $k \geq E$ and $\mathrm{mod}(k, \nu) = 0$ **then**
    $\boldsymbol{\theta}_{t+1}^{\text{SWA}} = \left(\boldsymbol{\theta}_t^{\text{SWA}} \cdot l + \boldsymbol{\theta}_{t+1}^{\text{SWA}}\right) / \left(l+1\right)$
  end if
end for
return $\boldsymbol{\theta}^{\text{SWA}}$

Algorithm 2 Sharpness-Aware Minimization [22]

Input: Loss function $\mathcal{L}$, training budget in number of iterations b, training dataset $\mathcal{D} := \cup_{i=1}^{n}\{\boldsymbol{x}_i\}$, mini-batch size $|\mathcal{B}|$, neighborhood radius ρ, (scheduled) learning rate η, initial weights $\boldsymbol{\theta}_0$.
for $k \leftarrow 1, \ldots, b$ **do**
  Sample a mini-batch $\mathcal{B}$ from $\mathcal{D}$
  Compute worst-case perturbation $\widehat{\boldsymbol{\epsilon}} \leftarrow \rho \frac{\nabla\mathcal{L}(\boldsymbol{\theta})}{\|\nabla\mathcal{L}(\boldsymbol{\theta})\|_2}$
  Compute gradient $\boldsymbol{g} \leftarrow \nabla\mathcal{L}\left(\boldsymbol{\theta}_t^{\text{SAM}} + \widehat{\boldsymbol{\epsilon}}\right)$
  Update parameters $\boldsymbol{\theta}_{t+1}^{\text{SAM}} \leftarrow \boldsymbol{\theta}_t^{\text{SAM}} - \eta\boldsymbol{g}$
end for
return $\boldsymbol{\theta}^{\text{SAM}}$

where l is the number of distinct parameters averaged so far and t is the SGD iteration number.[2]

SWA has two hyper-parameters: the update frequency ν and starting epoch E. When using a constant learning rate, Izmailov et al. [45] suggests updating the parameters once after each epoch, i.e. $\nu \approx \frac{N}{|\mathcal{B}|}$, and starting at $E \approx 0.75T$, where T is the training budget required to train the model until convergence with conventional SGD training.

He et al. [37] argue that SWA may always improve generalization, regardless of the loss function's geometry. Kaddour [48] show that averaging a specific range of weights can speed up training convergence. Cha et al. [8] argue that tuning ν and E carefully is necessary to make it work effectively in domain generalization (DG) tasks. Besides DG tasks, a list of tuned hyper-parameters based on a fair model selection procedure across different architectures and tasks has been missing in the literature. To the best of our knowledge, Cha et al. [8] is the only study that compares SWA and SAM over the same experiments, but it focuses on domain generalization tasks which we, therefore, leave out in this work.

2.3 Sharpness-Aware Minimization (SAM)

While SWA is implicitly biased towards flat minima, SAM *explicitly* approximates the flatness around parameters $\boldsymbol{\theta}$ to guide the parameter update. It first computes the worst-case perturbation $\boldsymbol{\epsilon}$ that maximizes the loss within a given neighborhood ρ, and then minimizes the loss w.r.t. the perturbed weights $\boldsymbol{\theta} + \boldsymbol{\epsilon}$. Formally, SAM finds $\boldsymbol{\theta}$ by solving the minimax problem:

$$\min_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}), \tag{4}$$

where $\rho \geq 0$ is a hyperparameter.

To find the worst-case perturbation $\boldsymbol{\epsilon}^*$ efficiently in practice, Foret et al. [22] approximates Eq. (4) via a first-order Taylor expansion of $\mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon})$ w.r.t. $\boldsymbol{\epsilon}$ around $\mathbf{0}$, obtaining

$$\boldsymbol{\epsilon}^* \approx \underset{\|\boldsymbol{\epsilon}\|_2 \leq \rho}{\arg\max}\, \boldsymbol{\epsilon}^\top \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}) \approx \underbrace{\rho \cdot \frac{\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})}{\|\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})\|}}_{=:\widehat{\boldsymbol{\epsilon}}}. \tag{5}$$

In words, $\widehat{\boldsymbol{\epsilon}}$ is simply the scaled gradient of the loss function w.r.t to the current parameters $\boldsymbol{\theta}$. Given $\widehat{\boldsymbol{\epsilon}}$, the altered gradient used to update the current $\boldsymbol{\theta}_t$ (in place of $\boldsymbol{g}(\boldsymbol{\theta}_t)$) is

$$\nabla_{\boldsymbol{\theta}} \max_{||\boldsymbol{\epsilon}||_2 \leq \rho} \mathcal{L}(\boldsymbol{\theta} + \boldsymbol{\epsilon}) \approx \nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})|_{\boldsymbol{\theta}+\hat{\boldsymbol{\epsilon}}}.$$

Due to Eq. (5), SAM's computational overhead consists of an additional forward and backward pass per parameter update step compared to SWA and non-flat optimizers.

[2] SWA parameters are constant between averaging steps.

SAM's performance strongly depends on the neighborhood radius ρ. For example, Chen et al. [12], Wu et al. [88] show that ρ should be set to values outside the originally considered ranges by Foret et al. [22]. Analogously to Sec. 2.2, this lack of coherence among hyper-parameter tuning protocols in the SAM literature makes it tricky to determine SAM's comparative effectiveness.

2.4 Other Flat-Minima Optimizers

There are several extensions of SWA [34, 8] and SAM [59, 98, 96]. For simplicity, we do not consider them in this work. Besides SWA and SAM, other flat-minima optimizers include e.g. [9, 78]. However, due to their computational cost and/or lack of performance gains, we do not include them in this work. Chaudhari et al. [9] requires $[5, 20]$ forward and backward passes per parameter update. Sankar et al. [78] similarly requires $[5, 10]$ forward and backward passes to estimate the Hessian trace and 6 of 7 experiments yield minimal improvement of $\leq 0.27\%$, see Table 1 in Sankar et al. [78]. In contrast, SWA and SAM have been shown to increase performance by multiple percentage points in some cases [8, 12], while requiring many fewer computational resources.

3 How do minima found by SWA and SAM differ?

In this section, we investigate SWA and SAM solutions for two prototypical deep learning tasks, where these optimizers improve over the baseline. Our goal is to better understand their geometric properties (instead of their generalization performance, which is the focus of Sec. 4).

First, we investigate the behavior of the loss landscape along the line between non-flat and flat solutions (Sec. 3.1). Previous studies successfully used such linear interpolations to gain novel insights, e.g., for training dynamics [30, 24], regularization [63, 27], and network pruning [25]. Second, motivated by findings in Sec. 3.1, we average SAM iterates and visualize interpolations between averaged and non-averaged solutions (Sec. 3.2). Interestingly, the averaged SAM solution is less susceptible to asymmetric directions. Third, we compare quantitative measurements of all solutions' flatnesses (Sec. 3.3). Here, we compute dominant Hessian eigenvalues, as commonly used in the flat minima literature [9, 93, 12, 22].

We choose the following two disparate learning settings: (i) a well-known image classification task, widely used for evaluation in flat-minima optimizer papers, and (ii) a novel, challenging Python code summarization task, on which state-of-the-art models achieve only around 16% F1 score on the test set (which is **higher** than its commonly achieved accuracy on the more challenging training set), and that has not been explored yet in the flat-minima literature. Specifically, for (i), we investigate the loss/accuracy surfaces of a WideResNet28-10 [94] model on CIFAR-100 [57] (baseline non-flat optimizer: SGD with momentum (SGD-M)) [77]. For (ii), we use the theoretically-grounded Graph Isomorphism Network [90] model on OGB-Code2 [42] (baseline optimizer: Adam [51]).

All optimizers start from the same initialization. We denote the minimizer produced by the non-flat methods (SGD-M and Adam) by $\boldsymbol{\theta}^{\text{NF}}$ and the flat ones by $\boldsymbol{\theta}^{\text{SWA}}$ and $\boldsymbol{\theta}^{\text{SAM}}$.

3.1 What is between non-flat and flat solutions?

We start by comparing the similarity of flat and non-flat minimizers through linear interpolations. This analysis allows us to understand if they are in the same basin, and how close they are to a region of sharply-increasing loss, where we expect loss/accuracy to differ widely between train and test. Further, for each of our 4 observations, we recommend a future work direction.

To linearly interpolate between two sets of parameters $\boldsymbol{\theta}$ and $\boldsymbol{\theta}'$, we parameterize the line connecting these two by choosing a scalar parameter α, and defining the weighted average $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta}+\alpha\boldsymbol{\theta}'$. If there exists no high-loss barrier between two networks $\boldsymbol{\theta}, \boldsymbol{\theta}'$ along the linear interpolation, we say that they are located in the same *basin*, i.e., $\{\boldsymbol{\theta}, \boldsymbol{\theta}'\} \in \boldsymbol{\Omega}$. [69, 97]. A basin is an area in the parameter space where the loss function has relatively low values. Due to NN non-linearities, the linear combination of the weights of two accurate models does not necessarily define an accurate model. Hence, we generally expect high-loss barriers along the linear interpolation path.

While there are alternative distance measures that could be used to compare two networks, they typically either (a) do not offer clear interpretations, as pointed out by Frankle et al. [25], or (b) yield

trivial network connectivity results, such as *non-linear* low-loss paths, which can be found for any two network minimizers [20, 26, 31, 23].

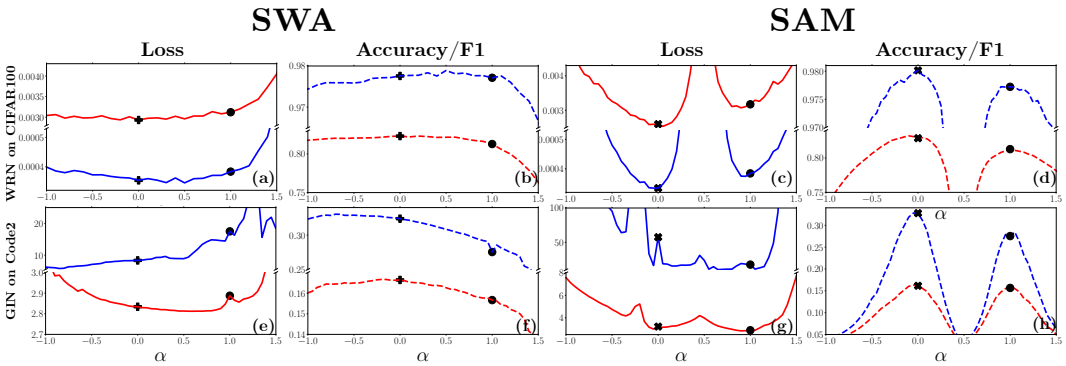


Figure 2: Training (blue) and test (red) losses (—) and accuracies (·····) of linear interpolations $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta} + \alpha\boldsymbol{\theta}'$ (for $\alpha \in [-1, 1.5]$) between SWA (+) and SAM (×) solutions ($\alpha = 0.0$) and non-flat baseline solutions (•, $\alpha = 1.0$).

Obs. 1: $\{\boldsymbol{\theta}^{\mathbf{SWA}}, \boldsymbol{\theta}^{\mathbf{NF}}\} \in \boldsymbol{\Omega}^{\mathbf{NF}}$**.** $\boldsymbol{\theta}^{\text{SWA}}$ and $\boldsymbol{\theta}^{\text{NF}}$ are in the same basin, as can be seen in Figures 2a and 2e. Additionally, $\boldsymbol{\theta}^{\text{NF}}$ is near the periphery of a sharp increase in loss, as can be seen when moving in the direction from $\boldsymbol{\theta}^{\text{SWA}}$ to $\boldsymbol{\theta}^{\text{NF}}$ (i.e., $\alpha > 1$). Conversely, $\boldsymbol{\theta}^{\text{SWA}}$ finds flat regions that change slowly in the loss. This bias of SWA to flatter loss beneficially transfers to the accuracy landscape too: Figures 2b and 2f show the accuracy/F1 score rapidly dropping off approaching and beyond $\boldsymbol{\theta}^{\text{NF}}$. Interestingly, in Figures 2e and 2f, we see that for Code2, for $\alpha < 0$, there exist solutions with even better training loss/accuracy but worse test loss/accuracy. However, $\boldsymbol{\theta}^{\text{SWA}}_{\text{GIN}}$ is close to the test accuracy maximizer along this interpolation. Future work may inspect why the cross entropy loss function used for GIN/Code2 seems less well correlated with its accuracy compared to WRN/CIFAR100.

Obs. 2: $\boldsymbol{\theta}^{\mathbf{SAM}} \in \boldsymbol{\Omega}^{\mathbf{SAM}} \neq \boldsymbol{\Omega}^{\mathbf{NF}}$**.** $\boldsymbol{\theta}^{\text{SAM}}$ and $\boldsymbol{\theta}^{\text{NF}}$ are not in the same basin: Figures 2c and 2g show that there is a high loss barrier between them, respectively. Figures 2d and 2h show that $\boldsymbol{\theta}^{\text{SAM}}$ and even nearby points in parameter space achieve higher accuracies/F1 scores (i.e. generalize better) than $\boldsymbol{\theta}^{\text{NF}}$ and points around it. This is an interesting result because we expect different basins to produce qualitatively different predictions, one of the motivations behind combining models, even if they exhibit different performances [43, 61]. Grewal & Bui [32] successfully combine models yielded by different optimizers, and we think future work should study ensembling SAM and non-SAM solutions.

Obs. 3: SAM finds a saddle point. Figure 2g shows $\boldsymbol{\theta}^{\text{SAM}}_{\text{GIN}}$ being located in a sharp training loss minimum whose loss is much higher than $\boldsymbol{\theta}^{\text{NF}}$. Yet, its test loss is only slightly higher, and its F1 score is better. We visualize 2D plots moving along random directions (not shown here due to space) to confirm that $\boldsymbol{\theta}^{\text{SAM}}_{\text{GIN}}$ is a saddle point. A common pathology among curvature-based methods is that they attract saddle points [15]. Since SAM takes some form of curvature into account, too, we believe that future work should investigate SAM's propensity to find saddle points and potential remedies.

Obs. 4: $\boldsymbol{\theta}^{\mathbf{SAM}}$ **is closer to sharper directions than** $\boldsymbol{\theta}^{\mathbf{SWA}}$**,** as can be seen by $\mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SAM}}(0.1)) \approx 2 \cdot \mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SAM}}(-0.1))$, while $\mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SWA}}(0.1)) \approx \mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SWA}}(-0.1))$, where $\mathcal{L}(\cdot)_{\text{tr/te}}$ refers to both training and test loss functions. A possible explanation for SAM being closer to sharp sides is that while it finds different basins than SGD/SWA by smoothing the loss surface (as illustrated in Fig. 1), *within* a local basin, it may oscillate around the minimizer similarly as SGD. One cause for this can be that $\boldsymbol{\Omega}^{\text{SAM}}$'s hypersphere is larger than SAM's radius ρ. If that holds, then given a small enough learning rate, we expect it to oscillate around $\boldsymbol{\theta}^* \in \boldsymbol{\Omega}^{\text{SAM}}$ (the smaller the learning rate, the less likely it escapes the basin due to that stochasticity). Two possible remedies for that are: (1) adapt/schedule ρ, or (2) average SAM iterates to bias its solution towards the flatter side. (1) has been explored by [98, 96]. We try (2) in the next subsection. Future work may study SAM's basin escape time, e.g., using convolutions [53] or stochastic differential equations [97].

3.2 What happens if we average SAM iterates?

Based on observation 4: "**$\boldsymbol{\theta}^{\text{SAM}}$ is closer to sharper directions than $\boldsymbol{\theta}^{\text{SWA}}$**", averaging SAM iterates may further improve generalization. The reason is that while SAM finds better-performing basins, *within* the basin, its final iterate may still be near a side that increases sharply in the loss.

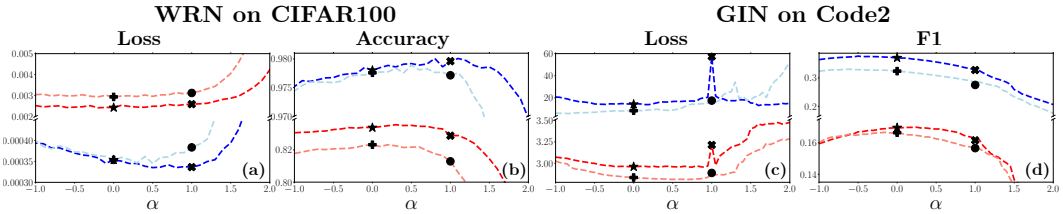


Figure 3: Training (blue) / test (red) losses (—) / accuracies (·····) between non-flat baseline (•) ↔ SWA (+), SAM (×) ↔ SWA+SAM (⋆).

Starting with the first of the two previously analyzed settings (WRN/CIFAR100), Figures 3a, and 3b show that $\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{WRN}}$ (marker: ★) achieves the lowest test loss and highest test accuracy, respectively. What stands out in comparison to the previous plots is $\boldsymbol{\theta}^{\text{SAM}}_{\text{WRN}}$'s (×) proximity to sharp sides, surprisingly similar to $\boldsymbol{\theta}^{\text{NF}}_{\text{WRN}}$ (•) here and in Figures 2c and 2e. As we hoped, $\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{WRN}}$ is indeed closer to a flatter region, as can be seen by $\mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{WRN}}(-0.2)) \approx \mathcal{L}_{\text{tr/te}}(\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{WRN}}(0.2))$.

In GIN/OGB-Code2, one unanticipated finding is that $\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{GIN}}$ escapes the (previously discussed) saddle point of $\boldsymbol{\theta}^{\text{SAM}}_{\text{GIN}}$, appearing here as a maximum in Figure 3c. A likely reason for that is that SAM traversed nearby flatter regions before arriving at the saddle point, especially if it is a non-strict saddle. In terms of F1 score, Figure 3d shows that while $\boldsymbol{\theta}^{\text{SWA}}_{\text{GIN}}$ (+) and $\boldsymbol{\theta}^{\text{SAM}}_{\text{GIN}}$ perform about equally well, the flatter region found by $\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{GIN}}$ improves over both.

3.3 How "flat" are the found minima?

We now quantify the flatnesses of all four optimizers over both tasks by computing the median of the dominant Hessian eigenvalue across all training set batches using the Power Iteration algorithm [68, 93]. This metric measures the worst-case loss landscape curvature. We choose this metric as it is very commonly used in the minima flatness literature, e.g., [9, 12, 22, 92, 18, 56, 79].

Table 1 shows that SAM leads to flatter minima than SWA in both cases. Interestingly $\boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}^{\text{NF}}_{\text{WRN}}) \approx 2.5 \cdot \boldsymbol{\lambda}_{\max}(\{\boldsymbol{\theta}^{\text{SWA}}, \boldsymbol{\theta}^{\text{SAM}}\})$, while $\boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}^{\text{NF}}_{\text{WRN}}) \approx 5.75\boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}^{\text{SWA+SAM}}_{\text{WRN}})$, indicating room for improvement in terms of flatness for both SWA and SAM. The relative differences are less dramatic for GIN/Code2, although surprisingly $\boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}^{\text{NF}}_{\text{GIN}}) \approx \boldsymbol{\lambda}_{\max}(\boldsymbol{\theta}^{\text{SWA}}_{\text{GIN}})$. In sum, averaging SAM iterates leads to the flattest minima **and** best-performing minima in both cases (see Sec. 4).

Table 1: Median $\boldsymbol{\lambda}_{\max}$ of Hessian over all training set batches.

Task	Baseline	SWA	SAM	SWA+SAM
WRN on CIFAR100	673	265	237	**117**
GIN on Code2	16.65	16.79	11.31	**9.96**

4 How do SWA and SAM perform on a broad set of experiments?

As we point out in the introduction, there is almost no overlap and consistency regarding reported SWA and SAM results in the literature. This section addresses this gap. For example, Bahri et al. [4], Chen et al. [12] illustrate that the flat minima found by SAM improve generalization on Transformer [85] architectures compared to non-flat optimizers, but they do not compare against SWA. Hence, it is unclear if the computationally cheaper SWA may provide better or similar performance.

We compare flat minimizers SWA, SAM, and averaged SWA iterates (SWA+SAM) over the non-flat minimizers across a range of different tasks in the domains of computer vision, natural language processing, and graph representation learning. We average all runs at least three times across random seeds (more often for experiments with higher variability, see details in Appendix B), and we report

Table 2: CV test results: Supervised Classification (SC), and Self-Supervised Learning (SSL) tasks.

Task	Model	Baseline	SWA	SAM	SWA+SAM
SC: CIFAR10	WRN-28-10	$96.78_{\pm 0.03}$	$-0.05_{\pm 0.04}$	$\mathbf{+0.34}_{\pm 0.09}$	$+0.25_{\pm 0.05}$
	PN-272	$96.73_{\pm 0.14}$	$+0.22_{\pm 0.14}$	$\mathbf{+0.42}_{\pm 0.06}$	$\mathbf{+0.41}_{\pm 0.02}$
	ViT-B-16	$98.95_{\pm 0.02}$	$-0.04_{\pm 0.04}$	$+0.07_{\pm 0.01}$	$\mathbf{+0.10}_{\pm 0.01}$
	Mixer-B-16	$96.65_{\pm 0.03}$	$+0.02_{\pm 0.03}$	$\mathbf{+0.19}_{\pm 0.05}$	$\mathbf{+0.22}_{\pm 0.06}$
SC: CIFAR100	WRN-28-10	$80.93_{\pm 0.19}$	$+1.62_{\pm 0.06}$	$+1.82_{\pm 0.14}$	$\mathbf{+2.24}_{\pm 0.14}$
	PN-272	$80.86_{\pm 0.12}$	$+1.88_{\pm 0.04}$	$+2.33_{\pm 0.08}$	$\mathbf{+2.60}_{\pm 0.09}$
	ViT-B-16	$92.77_{\pm 0.07}$	$-0.12_{\pm 0.05}$	$\mathbf{+0.19}_{\pm 0.09}$	$\mathbf{+0.13}_{\pm 0.07}$
	Mixer-B-16	$83.77_{\pm 0.08}$	$+0.45_{\pm 0.06}$	$+0.52_{\pm 0.15}$	$\mathbf{+0.97}_{\pm 0.12}$
SSL: CIFAR10	MoCo	$\mathbf{89.25}_{\pm \mathbf{0.07}}$	$-0.03_{\pm 0.10}$	$-0.25_{\pm 0.06}$	$-0.17_{\pm 0.10}$
	SimCLR	$\mathbf{88.66}_{\pm 0.08}$	$-0.05_{\pm 0.06}$	$\mathbf{+0.05}_{\pm 0.04}$	$-0.13_{\pm 0.06}$
	SimSiam	$\mathbf{89.86}_{\pm \mathbf{0.22}}$	$\mathbf{+0.12}_{\pm \mathbf{0.26}}$	$\mathbf{+0.07}_{\pm \mathbf{0.10}}$	$\mathbf{+0.11}_{\pm \mathbf{0.10}}$
	BarlowTwins	$\mathbf{86.34}_{\pm \mathbf{0.24}}$	$-0.09_{\pm 0.19}$	$\mathbf{+0.09}_{\pm \mathbf{0.15}}$	$\mathbf{+0.14}_{\pm \mathbf{0.05}}$
	BYOL	$90.32_{\pm 0.14}$	$\mathbf{+0.70}_{\pm 0.05}$	$+0.14_{\pm 0.03}$	$+0.21_{\pm 0.07}$
	SwaV	$87.28_{\pm 0.05}$	$\mathbf{+0.09}_{\pm \mathbf{0.06}}$	$\mathbf{+0.07}_{\pm \mathbf{0.12}}$	$+0.02_{\pm 0.06}$
SSL: ImageNette	MoCo	$81.74_{\pm 0.18}$	$+0.97_{\pm 0.10}$	$+0.91_{\pm 0.32}$	$\mathbf{+1.40}_{\pm \mathbf{0.10}}$
	SimCLR	$83.28_{\pm 0.22}$	$\mathbf{+0.95}_{\pm \mathbf{0.25}}$	$+0.18_{\pm 0.24}$	$\mathbf{+1.07}_{\pm \mathbf{0.13}}$
	SimSiam	$81.77_{\pm 0.14}$	$\mathbf{+0.20}_{\pm \mathbf{0.37}}$	$\mathbf{+0.33}_{\pm \mathbf{0.28}}$	$\mathbf{+0.18}_{\pm \mathbf{0.26}}$
	BarlowTwins	$77.49_{\pm 0.36}$	$+0.20_{\pm 0.16}$	$\mathbf{+0.47}_{\pm \mathbf{0.27}}$	$\mathbf{+0.66}_{\pm \mathbf{0.57}}$
	BYOL	$84.16_{\pm 0.14}$	$\mathbf{+0.76}_{\pm \mathbf{0.08}}$	$+0.15_{\pm 0.25}$	$+0.31_{\pm 0.19}$
	SwaV	$88.16_{\pm 0.31}$	$\mathbf{+1.04}_{\pm \mathbf{0.27}}$	$+0.03_{\pm 0.10}$	$\mathbf{+1.03}_{\pm \mathbf{0.09}}$

the corresponding standard error. We bold the best-performing approach and any approach whose average performance plus standard error overlaps it.

Hyper-parameters. For all architectures and datasets, we set hyperparameters shared by all methods (e.g., learning rate) mostly to values cited in prior work [3] As explained in Secs. 2.2 and 2.3, the effectiveness of flat-minima optimizers is highly sensitive to their additional hyper-parameters. We select hyper-parameters using a grid search over a held-out validation set. Specifically, for SWA we follow Izmailov et al. [45] and hold the update frequency ν constant to once per epoch and tune the start time $E \in \{0.5T, 0.6T, 0.75T, 0.9T\}$ (T is the number of baseline training epochs). Izmailov et al. [45] argue that a cyclical learning rate starting from E helps to encourage exploration of the basin. For the sake of simplicity, we average the iterates of the baseline directly but include even earlier starting times (i.e., $0.5T, 0.6T$). For SAM, we tune its neighborhood size $\rho \in \{0.01, 0.02, 0.05, 0.1, 0.2\}$, as in previous work [22, 4].

Appendix B contains the values of all hyper-parameters and additional training details (including public model checkpoints, hardware infrastructure, software libraries, etc.) to ensure full reproducibility alongside open-sourcing our code.

4.1 Computer Vision

Supervised Classification (SC). We evaluate the CNN architectures WideResNets [94] with 28 layers and width 10, and PyramidNet (PN) with 110 layers and widening factor 272 [36] as well as Vision Transformer (ViT) [19] and MLP-Mixer [81] on CIFAR{10, 100} [57]. All experiments use basic data augmentations: horizontal flip, padding by four pixels, random crop, and cutout [16].

Self-Supervised Learning (SSL). We consider the following methods on CIFAR10 and ImageNette[4]: Momentum Contrast [39], a Simple framework for Contrastive Learning (SimCLR) [10], Simple Siamese representation learning (SimSiam) [11], Barlow Twins [95], Bootstrap your own Latent (BYOL) [33], and Swapping Assignments between multiple Views of the same image (SwAV) [7]. All SSL methods use a ResNet-18 [38] as backbone network. To test the frozen representations, we use k-nearest-neighbor classification with a memory bank [89]. We choose $k = 200$ and temperature $\tau = 0.1$ to reweight similarities. Compared to learning a linear model on top of the representations, this evaluation procedure is more robust to hyperparameter changes [54].

[3] Sometimes with minor modifications, e.g., adjusting per-device batch sizes to be compatible with our GPU infrastructure.

[4] `https://github.com/fastai/imagenette`

Figure 4: (a) NLP test results: Open-Domain Question Answering and Natural Language Understanding (GLUE) including paraphrase, sentiment analysis, and textual entailment. (b) GRL test results: Node Property Prediction (NPP), Graph Property Prediction (GPP), Link Property Prediction (LPP).

Task	Model	Baseline	SWA	SAM	SWA+SAM
NQ	FiD	$49.35_{\pm 0.44}$	$-0.20_{\pm 0.33}$	$\mathbf{+0.33_{\pm 0.19}}$	$\mathbf{+0.48_{\pm 0.21}}$
TriviaQA	FiD	$67.74_{\pm 0.29}$	$+0.40_{\pm 0.24}$	$\mathbf{+0.89_{\pm 0.03}}$	$\mathbf{+0.92_{\pm 0.10}}$
COLA	RoBERTa	$60.41_{\pm 0.22}$	$+0.09_{\pm 0.08}$	$\mathbf{+1.57_{\pm 1.20}}$	$\mathbf{+1.41_{\pm 1.14}}$
SST	RoBERTa	$94.95_{\pm 0.13}$	$-0.30_{\pm 0.27}$	$-0.23_{\pm 0.40}$	$\mathbf{+0.19_{\pm 0.14}}$
MRPC	RoBERTa	$89.14_{\pm 0.57}$	$+0.08_{\pm 0.49}$	$\mathbf{+0.73_{\pm 0.43}}$	$\mathbf{+0.81_{\pm 0.38}}$
STSB	RoBERTa	$90.40_{\pm 0.02}$	$+0.00_{\pm 0.05}$	$\mathbf{+0.38_{\pm 0.17}}$	$\mathbf{+0.35_{\pm 0.16}}$
QQP	RoBERTa	$91.36_{\pm 0.07}$	$+0.01_{\pm 0.06}$	$\mathbf{+0.08_{\pm 0.07}}$	$\mathbf{+0.06_{\pm 0.08}}$
MNLI	RoBERTa	$87.41_{\pm 0.09}$	$+0.08_{\pm 0.11}$	$\mathbf{+0.39_{\pm 0.02}}$	$+0.35_{\pm 0.03}$
QNLI	RoBERTa	$92.96_{\pm 0.06}$	$-0.08_{\pm 0.11}$	$+0.09_{\pm 0.01}$	$\mathbf{+0.11_{\pm 0.06}}$
RTE	RoBERTa	$80.09_{\pm 0.23}$	$-0.23_{\pm 0.20}$	$\mathbf{+0.70_{\pm 0.65}}$	$-0.46_{\pm 0.12}$

(a)

Task	Model	Baseline	SWA	SAM	SWA+SAM
NPP: Proteins	SAGE	$\mathbf{77.79_{\pm 0.18}}$	$\mathbf{-0.17_{\pm 0.22}}$	$\mathbf{-0.02_{\pm 0.13}}$	$\mathbf{-0.11_{\pm 0.15}}$
	DGCN	$\mathbf{85.42_{\pm 0.17}}$	$\mathbf{+0.11_{\pm 0.08}}$	$-0.14_{\pm 0.05}$	$-0.08_{\pm 0.07}$
NPP: Products	SAGE	$78.92_{\pm 0.08}$	$+0.39_{\pm 0.10}$	$+0.13_{\pm 0.08}$	$\mathbf{+0.57_{\pm 0.03}}$
	DGCN	$73.88_{\pm 0.13}$	$\mathbf{+0.44_{\pm 0.14}}$	$+0.08_{\pm 0.09}$	$\mathbf{+0.53_{\pm 0.05}}$
GPP: Code2	GCN	$16.04_{\pm 0.09}$	$+0.73_{\pm 0.11}$	$+0.36_{\pm 0.08}$	$\mathbf{+0.93_{\pm 0.15}}$
	GIN	$15.73_{\pm 0.11}$	$+0.83_{\pm 0.11}$	$+0.57_{\pm 0.09}$	$\mathbf{+1.10_{\pm 0.09}}$
GPP: Molpcba	GIN	$28.10_{\pm 0.11}$	$\mathbf{+0.40_{\pm 0.18}}$	$-0.33_{\pm 0.14}$	$\mathbf{+0.33_{\pm 0.16}}$
	DGCN	$25.65_{\pm 0.13}$	$\mathbf{+1.90_{\pm 0.20}}$	$-0.13_{\pm 0.18}$	$+1.34_{\pm 0.12}$
LPP: Biokg	CP	$84.06_{\pm 0.00}$	$\mathbf{+0.07_{\pm 0.01}}$	$0.00_{\pm 0.03}$	$\mathbf{+0.08_{\pm 0.02}}$
	ComplEx	$84.94_{\pm 0.01}$	$\mathbf{+0.14_{\pm 0.01}}$	$-0.02_{\pm 0.01}$	$\mathbf{+0.12_{\pm 0.02}}$
LPP: Citation2	GCN	$79.52_{\pm 0.41}$	$-0.05_{\pm 0.52}$	$+1.32_{\pm 0.06}$	$\mathbf{+1.50_{\pm 0.13}}$
	SAGE	$81.95_{\pm 0.02}$	$\mathbf{+1.15_{\pm 0.02}}$	$-0.31_{\pm 0.07}$	$+0.86_{\pm 0.04}$

(b)

4.2 Natural Language Processing

We consider the task of open domain question answering (ODQA) using a T5-based model Fusion-In-Decoder (FiD) [44]. We evaluate FiD-base on the test sets of Natural Questions (NQ) [58] and TriviaQA [47]. We also consider natural language understanding tasks included in the GLUE benchmark [86], which cover acceptability, sentiment, paraphrase, similarity, and inference. We fine-tune RoBERTa-base [66] for each task individually and report the results on the GLUE dev set.

4.3 Graph Representation Learning

We use a subset of the Open Graph Benchmark (OGB) datasets [42]. The tasks are node property prediction (NPP), graph property prediction (GPP), and link property prediction (LPP). For each task, we use two of the following GNN architectures and matrix factorization methods: GCN [52], DeeperGCN (DGCN) [62], SAGE [35], GIN [90], ComplEx [83], and CP [60]. We use popular training schemes, such as virtual nodes, cluster sampling [14], or relation prediction as auxiliary training objective [13]. The reported metrics are ROC-AUC for Proteins, Accuracy for Products, F1 score for Code2, Average precision for Molpcba, and Mean Reciprocal Rank for Biokg/Citation2.

4.4 8 Findings

We use $\mathcal{G}(\cdot)$ to describe the generalization accuracy/F1/ROCAUC/AP/MRR of all optimizers $\{\text{Non-flat baseline(NF)}, \text{SWA}, \text{SAM}, \text{SWA+SAM}\}$.

1. **Both dataset and architecture matter.** To explain the relationship between flatness and generalization, some works focused on the network's architecture [17, 12, 71], while others focused on the data distribution [84, 72, 21]. Our results show that both matter. Datasets: e.g., for node property prediction (Proteins), we see that no flat optimizer improves over the baseline optimizer, however for (Products), flat-minima optimizers on the same architectures significantly improve over the baseline. Architectures: e.g., there is a vast difference across model architectures for link property prediction on the Citation2 dataset: using a GNN with GCN layers, SAM achieves a statistically significant boost of >1.30% and SWA slightly hurts the performance. When we replace the GCN layers with SAGE layers (and fix everything else), we see a boost of $> 1.15\%$ for SWA, while SAM hurts the performance by -0.31%.
2. **SWA underperforms on NLP tasks.** SAM achieves the best performance in 7/10 experiments on NLP tasks, consistent with the findings of Bahri et al. [4], which show that SAM can boost performance across a wide range of NLP tasks. However, SWA never performs best, only improving the results in 1/10 cases, and even hurting the performance on 4 tasks. Surprisingly, $\mathcal{G}(\text{SWA+SAM}) > \mathcal{G}(\text{SAM})$ for $\{\text{SST, QNLI}\}$ while SWA decreases performance in these cases.
3. **SWA beats SAM on GRL tasks**, $\mathcal{G}(\text{SWA}) > \mathcal{G}(\text{NF})$ in 10/12 experiments, while $\mathcal{G}(\text{SAM}) > \mathcal{G}(\text{NF})$ only in 4. We examine why SAM under-performs in the Appendix.
4. **SWA does not work well with Transformers.** SWA often does not improve and sometimes hurts performances, as can be seen in the ViT results in Table 2, and NLP results in Fig. 4a. In contrast, SAM has some positive effects in these settings. We explore this further in the Appendix.

5. **SWA and SAM can work on SSL tasks too**: This is non-obvious because the theoretical motivation behind finding flat minima is linked to supervised learning losses [41, 72, 21]. Concurrently to our work, Ramesh et al. [76] report that SAM helps for contrastive CLIP [74] models too.

6. **Flat optimizers can underperform non-flat optimizers.** The non-flat optimizer is nearly always the best for NPP Proteins and SSL methods on CIFAR10. We investigate the NPP Proteins solutions in the next subsection and recommend a more thorough investigation of the landscapes of SSL objectives for future work.

7. **Flat-minima optimizers offer asymmetric payoffs:** at worst, they decreased performance by -0.30%, at best, they increased it by 2.60%.

8. **Averaging SAM iterates often improves over SWA or SAM alone**. $\mathcal{G}(\text{SWA+SAM}) > \min(\mathcal{G}(\text{SWA}), \mathcal{G}(\text{SAM}))$ in 39/42 cases. We hypothesize that asymmetric payoffs are the reason: when either SWA or SAM does not improve over the baseline (as discussed above), it does not hurt (much) either, hence SWA+SAM is more robust across all tasks.

4.5 Why do flat-minima optimizers fail?

Here, we audit one of the cases, where neither $\boldsymbol{\theta}^{\text{SWA}}$ nor $\boldsymbol{\theta}^{\text{SAM}}$ improves over $\boldsymbol{\theta}^{\text{NF}}$ (this happens in 3 out of 42 cases): training a GraphSAGE [35] model on OGB-Proteins: a protein-protein interaction graph where the goal is to predict the presence of protein functions (multi-label binary classification) [42]. $\boldsymbol{\theta}^{\text{SWA}}$ performs noticeably worse; $\boldsymbol{\theta}^{\text{SAM}}$ performs about equally well.

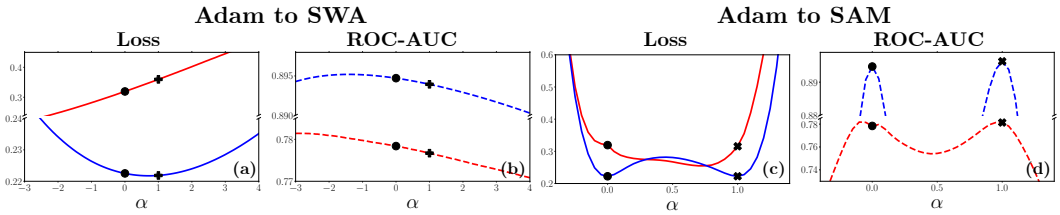


Figure 5: GraphSAGE on OGB-Proteins: Adam's (•) solution performs about equally well as SAM (×), and better than SWA (+).

Fig. 5 shows two linear interpolations: between $\boldsymbol{\theta}^{\text{NF}}$ (ADAM) and (1) $\boldsymbol{\theta}^{\text{SWA}}$ (Figures 5a and 5b), and (2) $\boldsymbol{\theta}^{\text{SAM}}$ (Figures 5c and 5d). In contrast to success cases in Fig. 2, here: (a) for both SWA and SAM, the training loss minimizer is very uncorrelated with the test loss minimizer; (b) SAM and ADAM seem to be contained in the same test loss/accuracy basin. More analyses can be found in the Appendix.

5 Limitations and Future Work

First, some of the fixed, shared hyperparameter values we used from previous works may harm the effect of flat optimizers. The ideal experimental design includes tuning all hyperparameters independently for the non-flat optimizer, SWA, SAM, and SWA+SAM. However, this forces the number of required runs to grow exponentially in unique hyperparameters and quickly renders this benchmark infeasible.

Second, despite our best efforts to evaluate the optimizers on a broad range of benchmark tasks, there are still plenty of unexplored domains; especially some of which are known to be sensitive to careful optimization, such as generative modeling [40], deep reinforcement learning [2], or causal machine learning [49].

Third, in general, we believe fruitful directions of research include (a) optimizers that explicitly find basins where training loss flatness more directly corresponds to higher hold-out accuracy, (b) post-processing methods for existing optimization runs to move into flatter regions of these basins, (c) loss functions whose contours more tightly align with accuracy contours, (d) the study of flat-minima hyperparameter interactions (e.g., learning rate and neighborhood radius in SAM), (e) analyses of flat minima optimization on convergence speed [48].

Our benchmark results point to which tasks would most benefit from improving these future work directions: graph learning tasks would clearly benefit from improvements in (a), as SAM is never among the best performing method, and language tasks would benefit if (b) is improved, as SWA is never among the best performing method).

6 Conclusion

We investigated when flat minimia optimizers work by conducting a fair comparison of two popular flat-minima optimizers. We examined the behavior of SWA/SAM by analyzing their loss landscapes on two representative deep learning tasks. Our next step was to evaluate their generalization performance on a broad and diverse set of tasks (in data, learning settings, and model architectures). Based on this benchmarking, we identified 8 findings, of which some directly guide future work directions. Finally, when SWA/SAM did not improve over baselines, common assumptions seemed broken (i.e., train-to-test loss minimizers were not correlated).

Acknowledgements

We are very grateful to Kilian Q. Weinberger and Gao Huang for initial discussions and intuition, Pontus Stenetorp for NLP experimental design advice, and Oscar Key for feedback on the draft. JK and LL acknowledge support by the Engineering and Physical Sciences Research Council with grant number EP/S021566/1. This research was supported through Azure resources provided by The Alan Turing Institute and credits awarded by Google Cloud.

References

[1] Aghajanyan, A., Shrivastava, A., Gupta, A., Goyal, N., Zettlemoyer, L., and Gupta, S. Better fine-tuning by reducing representational collapse. In *International Conference on Learning Representations*, 2020.

[2] Arulkumaran, K., Deisenroth, M. P., Brundage, M., and Bharath, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[3] Athiwaratkun, B., Finzi, M., Izmailov, P., and Wilson, A. G. There are many consistent explanations of unlabeled data: Why you should average, 2019.

[4] Bahri, D., Mobahi, H., and Tay, Y. Sharpness-aware minimization improves language model generalization, 2021.

[5] Bisla, D., Wang, J., and Choromanska, A. Low-pass filtering sgd for recovering flat optima in the deep learning optimization landscape, 2022. URL `https://arxiv.org/abs/2201.08025`.

[6] Bottou, L., Curtis, F. E., and Nocedal, J. Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311, 2018.

[7] Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/70feb62b69f16e0238f741fab228fec2-Abstract.html`.

[8] Cha, J., Chun, S., Lee, K., Cho, H.-C., Park, S., Lee, Y., and Park, S. SWAD: Domain generalization by seeking flat minima. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL `https://openreview.net/forum?id=zkHlu_3sJYU`.

[9] Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J. T., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. 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=B1YfAfcgl`.

[10] Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. E. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 2020. URL `http://proceedings.mlr.press/v119/chen20j.html`.

[11] Chen, X. and He, K. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 15750–15758, 2021.

[12] Chen, X., Hsieh, C.-J., and Gong, B. When vision transformers outperform resnets without pre-training or strong data augmentations, 2021.

[13] Chen, Y., Minervini, P., Riedel, S., and Stenetorp, P. Relation prediction as an auxiliary training objective for improving multi-relational graph representations. In *3rd Conference on Automated Knowledge Base Construction*, 2021. URL `https://openreview.net/forum?id=Qa3uS3H7-Le`.

[14] Chiang, W., Liu, X., Si, S., Li, Y., Bengio, S., and Hsieh, C. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. In Teredesai, A., Kumar, V., Li, Y., Rosales, R., Terzi, E., and Karypis, G. (eds.), *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2019, Anchorage, AK, USA, August 4-8, 2019*, pp. 257–266. ACM, 2019. doi: 10.1145/3292500.3330925. URL `https://doi.org/10.1145/3292500.3330925`.

[15] Dauphin, Y. N., Pascanu, R., Gülçehre, Ç., Cho, K., Ganguli, S., and Bengio, Y. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pp. 2933–2941, 2014. URL `https://proceedings.neurips.cc/paper/2014/hash/17e23e50bedc63b4095e3d8204ce063b-Abstract.html`.

[16] Devries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *CoRR*, abs/1708.04552, 2017. URL `http://arxiv.org/abs/1708.04552`.

[17] Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1019–1028. PMLR, 2017. URL `http://proceedings.mlr.press/v70/dinh17b.html`.

[18] Dong, Z., Yao, Z., Gholami, A., Mahoney, M. W., and Keutzer, K. HAWQ: hessian aware quantization of neural networks with mixed-precision. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pp. 293–302. IEEE, 2019. doi: 10.1109/ICCV.2019.00038. URL `https://doi.org/10.1109/ICCV.2019.00038`.

[19] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL `https://openreview.net/forum?id=YicbFdNTTy`.

[20] Draxler, F., Veschgini, K., Salmhofer, M., and Hamprecht, F. A. Essentially no barriers in neural network energy landscape. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1308–1317. PMLR, 2018. URL `http://proceedings.mlr.press/v80/draxler18a.html`.

[21] Dziugaite, G. K. and Roy, D. M. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In Elidan, G., Kersting, K., and Ihler, A. T. (eds.), *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence, UAI 2017, Sydney, Australia, August 11-15, 2017*. AUAI Press, 2017. URL `http://auai.org/uai2017/proceedings/papers/173.pdf`.

[22] Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=6Tm1mposlrM.

[23] Fort, S. and Jastrzebski, S. Large scale structure of neural network loss landscapes. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 6706–6714, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/48042b1dae4950fef2bd2aafa0b971a1-Abstract.html.

[24] Frankle, J. Revisiting "qualitatively characterizing neural network optimization problems". *CoRR*, abs/2012.06898, 2020. URL https://arxiv.org/abs/2012.06898.

[25] Frankle, J., Dziugaite, G. K., Roy, D. M., and Carbin, M. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3259–3269. PMLR, 2020. URL http://proceedings.mlr.press/v119/frankle20a.html.

[26] Garipov, T., Izmailov, P., Podoprikhin, D., Vetrov, D. P., and Wilson, A. G. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, 2018.

[27] Geiping, J., Goldblum, M., Pope, P. E., Moeller, M., and Goldstein, T. Stochastic training is not necessary for generalization. *CoRR*, abs/2109.14119, 2021. URL https://arxiv.org/abs/2109.14119.

[28] Ghorbani, B., Krishnan, S., and Xiao, Y. An investigation into neural net optimization via hessian eigenvalue density. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2232–2241. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19b.html.

[29] Golub, G. H. and Welsch, J. H. Calculation of gauss quadrature rules. *Mathematics of computation*, 23(106):221–230, 1969.

[30] Goodfellow, I. J. and Vinyals, O. Qualitatively characterizing neural network optimization problems. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6544.

[31] Gotmare, A., Keskar, N. S., Xiong, C., and Socher, R. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=r14EOsCqKX.

[32] Grewal, Y. and Bui, T. D. Diversity is all you need to improve bayesian model averaging. In *Bayesian Deep Learning Workshop at Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021*, 2021.

[33] Grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Pires, B. Á., Guo, Z., Azar, M. G., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent - A new approach to self-supervised learning. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

[34] Guo, H., Jin, J., and Liu, B. Stochastic weight averaging revisited. *CoRR*, abs/2201.00519, 2022. URL https://arxiv.org/abs/2201.00519.

[35] Hamilton, W. L., Ying, R., and Leskovec, J. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 1025–1035, 2017.

[36] Han, D., Kim, J., and Kim, J. Deep pyramidal residual networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pp. 6307–6315. IEEE Computer Society, 2017. doi: 10.1109/CVPR.2017.668. URL https://doi.org/10.1109/CVPR.2017.668.

[37] He, H., Huang, G., and Yuan, Y. Asymmetric valleys: Beyond sharp and flat local minima. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pp. 2549–2560, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01d8bae291b1e4724443375634ccfa0e-Abstract.html.

[38] He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pp. 770–778. IEEE Computer Society, 2016. doi: 10.1109/CVPR.2016.90. URL https://doi.org/10.1109/CVPR.2016.90.

[39] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. Momentum contrast for unsupervised visual representation learning. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pp. 9726–9735. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.00975. URL https://doi.org/10.1109/CVPR42600.2020.00975.

[40] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[41] Hochreiter, S. and Schmidhuber, J. Flat minima. *Neural computation*, 9(1):1–42, 1997.

[42] Hu, W., Fey, M., Zitnik, M., Dong, Y., Ren, H., Liu, B., Catasta, M., and Leskovec, J. Open graph benchmark: Datasets for machine learning on graphs. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/fb60d411a5c5b72b2e7d3527cfc84fd0-Abstract.html.

[43] Huang, G., Li, Y., Pleiss, G., Liu, Z., Hopcroft, J. E., and Weinberger, K. Q. 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.

[44] Izacard, G. and Grave, É. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 874–880, 2021.

[45] Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D. P., and Wilson, A. G. Averaging weights leads to wider optima and better generalization. In Globerson, A. and Silva, R. (eds.), *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018*, pp. 876–885. AUAI Press, 2018. URL http://auai.org/uai2018/proceedings/papers/313.pdf.

[46] Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgIPJBFvH.

[47] Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017.

[48] Kaddour, J. Stop wasting my time! saving days of imagenet and bert training with latest weight averaging. *arXiv preprint arXiv:2209.14981*, 2022. URL https://arxiv.org/abs/2209.14981.

[49] Kaddour, J., Lynch, A., Liu, Q., Kusner, M. J., and Silva, R. Causal machine learning: A survey and open problems. *arXiv preprint arXiv:2206.15475*, 2022. URL https://arxiv.org/abs/2206.15475.

[50] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. 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=H1oyRlYgg.

[51] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y. (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. URL http://arxiv.org/abs/1412.6980.

[52] Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. 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=SJU4ayYgl.

[53] Kleinberg, B., Li, Y., and Yuan, Y. An alternative view: When does sgd escape local minima? In *International Conference on Machine Learning*, pp. 2698–2707. PMLR, 2018.

[54] Kolesnikov, A., Zhai, X., and Beyer, L. Revisiting self-supervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 1920–1929, 2019.

[55] Kornblith, S., Norouzi, M., Lee, H., and Hinton, G. E. Similarity of neural network representations revisited. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3519–3529. PMLR, 2019. URL http://proceedings.mlr.press/v97/kornblith19a.html.

[56] Krishnapriyan, A. S., Gholami, A., Zhe, S., Kirby, R. M., and Mahoney, M. W. Characterizing possible failure modes in physics-informed neural networks. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 26548–26560, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/df438e5206f31600e6ae4af72f2725f1-Abstract.html.

[57] Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

[58] Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[59] Kwon, J., Kim, J., Park, H., and Choi, I. K. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 5905–5914. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/kwon21b.html.

[60] Lacroix, T., Usunier, N., and Obozinski, G. Canonical tensor decomposition for knowledge base completion. In Dy, J. G. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2869–2878. PMLR, 2018. URL http://proceedings.mlr.press/v80/lacroix18a.html.

[61] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In Guyon, I., von Luxburg, U., Bengio, S., Wallach, H. M., Fergus, R., Vishwanathan, S. V. N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pp. 6402–6413, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/9ef2ed4b7fd2c810847ffa5fa85bce38-Abstract.html.

[62] Li, G., Xiong, C., Thabet, A., and Ghanem, B. Deepergcn: All you need to train deeper gcns, 2020.

[63] Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In Bengio, S., Wallach, H. M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pp. 6391–6401, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/a41b3bb3e6b050b6c9067c67f663b915-Abstract.html.

[64] Li, Z., Cui, Z., Wu, S., Zhang, X., and Wang, L. Fi-gnn: Modeling feature interactions via graph neural networks for ctr prediction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 539–548, 2019.

[65] Liu, Q., Nickel, M., and Kiela, D. Hyperbolic graph neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[66] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[67] Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

[68] Mises, R. and Pollaczek-Geiringer, H. Praktische verfahren der gleichungsauflösung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 9(1):58–77, 1929.

[69] Neyshabur, B., Sedghi, H., and Zhang, C. What is being transferred in transfer learning? In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/0607f4c705595b911a4f3e7a127b44e0-Abstract.html.

[70] Nikishin, E., Izmailov, P., Athiwaratkun, B., Podoprikhin, D., Garipov, T., Shvechikov, P., Vetrov, D., and Wilson, A. G. Improving stability in deep reinforcement learning with weight averaging. In *Uncertainty in artificial intelligence workshop on uncertainty in Deep learning*, 2018.

[71] Park, N. and Kim, S. How do vision transformers work? In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=D78Go4hVcxO.

[72] Petzka, H., Kamp, M., Adilova, L., Sminchisescu, C., and Boley, M. Relative flatness and generalization. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=sygvo7ctb_.

[73] Polyak, B. T. and Juditsky, A. B. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4):838–855, 1992.

[74] Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. Learning transferable visual models from natural language supervision. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*,

volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 2021. URL http://proceedings.mlr.press/v139/radford21a.html.

[75] Rahman, M. K. Training sensitivity in graph isomorphism network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pp. 2181–2184, 2020.

[76] Ramesh, A., Dhariwal, P., Nichol, A., Chu, C., and Chen, M. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[77] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. *Learning Representations by Back-Propagating Errors*, pp. 696–699. MIT Press, Cambridge, MA, USA, 1988. ISBN 0262010976.

[78] Sankar, A. R., Khasbage, Y., Vigneswaran, R., and Balasubramanian, V. N. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. *arXiv preprint arXiv:2012.03801*, 2020.

[79] Stutz, D., Hein, M., and Schiele, B. Relating adversarially robust generalization to flat minima. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 7787–7797. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00771. URL https://doi.org/10.1109/ICCV48922.2021.00771.

[80] Susmelj, I., Heller, M., Wirth, P., Prescott, J., and et al., M. E. Lightly. *GitHub. Note: https://github.com/lightly-ai/lightly*, 2020.

[81] Tolstikhin, I. O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., Lucic, M., and Dosovitskiy, A. Mlp-mixer: An all-mlp architecture for vision. *CoRR*, abs/2105.01601, 2021. URL https://arxiv.org/abs/2105.01601.

[82] Touvron, H., Cord, M., Sablayrolles, A., Synnaeve, G., and Jégou, H. Going deeper with image transformers. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pp. 32–42. IEEE, 2021. doi: 10.1109/ICCV48922.2021.00010. URL https://doi.org/10.1109/ICCV48922.2021.00010.

[83] Trouillon, T., Welbl, J., Riedel, S., Gaussier, É., and Bouchard, G. Complex embeddings for simple link prediction. In Balcan, M. and Weinberger, K. Q. (eds.), *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, volume 48 of *JMLR Workshop and Conference Proceedings*, pp. 2071–2080. JMLR.org, 2016. URL http://proceedings.mlr.press/v48/trouillon16.html.

[84] Tsuzuku, Y., Sato, I., and Sugiyama, M. Normalized flat minima: Exploring scale invariant definition of flat minima for neural networks using PAC-Bayesian analysis. In III, H. D. and Singh, A. (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9636–9647. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/tsuzuku20a.html.

[85] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[86] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, 2018.

[87] Wolf, T., Chaumond, J., Debut, L., Sanh, V., Delangue, C., Moi, A., Cistac, P., Funtowicz, M., Davison, J., Shleifer, S., et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, 2020.

[88] Wu, Y., Bojchevski, A., and Huang, H. Adversarial weight perturbation improves generalization in graph neural networks, 2022. URL https://openreview.net/forum?id=hUr6K4D9f7P.

[89] Wu, Z., Xiong, Y., Yu, S. X., and Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 3733–3742. Computer Vision Foundation / IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00393. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Wu_Unsupervised_Feature_Learning_CVPR_2018_paper.html.

[90] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=ryGs6iA5Km.

[91] Yang, Y., Hodgkinson, L., Theisen, R., Zou, J., Gonzalez, J. E., Ramchandran, K., and Mahoney, M. W. Taxonomizing local versus global structure in neural network loss landscapes. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=P6bUrLREcne.

[92] Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. LARGE BATCH SIZE TRAINING OF NEURAL NETWORKS WITH ADVERSARIAL TRAINING AND SECOND-ORDER INFORMATION, 2019. URL https://openreview.net/forum?id=H1lnJ2Rqt7.

[93] Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In Wu, X., Jermaine, C., Xiong, L., Hu, X., Kotevska, O., Lu, S., Xu, W., Aluru, S., Zhai, C., Al-Masri, E., Chen, Z., and Saltz, J. (eds.), *2020 IEEE International Conference on Big Data (IEEE BigData 2020), Atlanta, GA, USA, December 10-13, 2020*, pp. 581–590. IEEE, 2020. doi: 10.1109/BigData50022.2020.9378171. URL https://doi.org/10.1109/BigData50022.2020.9378171.

[94] Zagoruyko, S. and Komodakis, N. Wide residual networks. In Wilson, R. C., Hancock, E. R., and Smith, W. A. P. (eds.), *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. URL http://www.bmva.org/bmvc/2016/papers/paper087/index.html.

[95] Zbontar, J., Jing, L., Misra, I., LeCun, Y., and Deny, S. Barlow twins: Self-supervised learning via redundancy reduction. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 12310–12320. PMLR, 2021. URL http://proceedings.mlr.press/v139/zbontar21a.html.

[96] Zhao, Y., Zhang, H., and Hu, X. Ss-sam: Stochastic scheduled sharpness-aware minimization for efficiently training deep neural networks. *arXiv preprint arXiv:2203.09962*, 2022.

[97] Zhou, P., Feng, J., Ma, C., Xiong, C., Hoi, S. C. H., et al. Towards theoretically understanding why sgd generalizes better than adam in deep learning. *Advances in Neural Information Processing Systems*, 33:21285–21296, 2020.

[98] Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N. C., sekhar tatikonda, s Duncan, J., and Liu, T. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=edONMAnhLu-.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 5
   (c) Did you discuss any potential negative societal impacts of your work? [No] We consider this work to be an investigation into optimization algorithms that are central to modern ML systems. As these systems can be used in vastly different ways, we believe that it is intractable to isolate the societal impact of novel insights into optimization.

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Sec. 4
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes] See Appendix.
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Additional Analyses

A.1 CKA Similarities

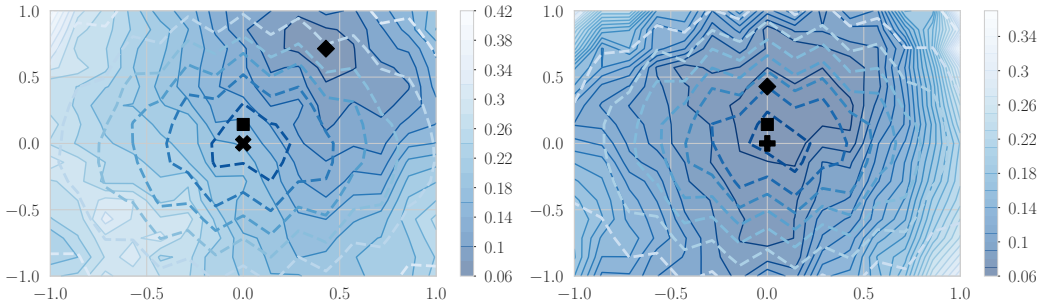
We compute the CKA [55, 91] and cosine similarities of network output logits on train and test set, respectively. Table 3 shows the results.

Table 3: **Pairwise CKA [55] and cosine similarities** between non-flat (NF) and SWA/SAM solutions. SWA solutions produce predictions more similar to NF ones than SAM.

Task	$s_{\text{CKA}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SWA}})$	$s_{\text{cosine}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SWA}})$	$s_{\text{CKA}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SAM}})$	$s_{\text{cosine}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SAM}})$
WRN-CIFAR100 (Train)	0.9880	0.9812	0.9810	0.9240
WRN-CIFAR100 (Test)	0.9137	0.9732	0.8580	0.9045
GIN-Code2 (Train)	0.8522	0.9730	0.7276	0.9515
GIN-Code2 (Test)	0.8677	0.9750	0.7275	0.9516
RoBERTa-QNLI (Train)	0.9997	0.9991	0.9790	0.9510
RoBERTa-QNLI (Valid)	0.9830	0.9959	0.9550	0.9530
RoBERTa-RTE (Train)	0.9931	0.9891	0.9831	0.9628
RoBERTa-RTE (Test)	0.9314	0.9567	0.8808	0.8927
GIN-Molpcba (Train)	0.8886	0.9973	0.7441	0.9804
GIN-Molpcba (Test)	0.8772	0.9942	0.7232	0.9730

The results show that the SAM solutions produce predictions that are less similar to the non-flat baseline than SWA solutions, as indicated by lower CKA and cosine similarities. This result is in line with Observation 1 and 2 from Sec. 3.1.

A.2 Saddle point of SAM's GIN solution



(a) SAM solution. (b) SWA solution (for comparison).

Figure 6: **SAM and SWA solution in 2D random plane for GIN-Code2 task.** We depict $\boldsymbol{\theta}^{\text{SAM}}, \boldsymbol{\theta}^{\text{SWA}}$ by $\times, +$, respectively, the test set F1 score maximizer by ■, and the training loss minimizer by ♦. The converged SAM solution is distant from the training loss minimizer in the 2D plane: $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^{\text{SAM}}) = 0.1779 \gg 0.0672 = \mathcal{L}_{\text{train}}(\boldsymbol{\theta}^{\blacklozenge})$. Further, losses (—) and F1 scores (·····) are not well-aligned. In contrast, the SWA solution is almost the training loss minimizer: $\mathcal{L}_{\text{train}}(\boldsymbol{\theta}^{\text{SWA}}) = 0.0661 \approx 0.0609 = \mathcal{L}_{\text{train}}(\boldsymbol{\theta}^{\blacklozenge})$. Also, losses and F1 scores are better aligned. Yet, $\boldsymbol{\theta}^{\text{SAM}}$ and $\boldsymbol{\theta}^{\text{SWA}}$ perform about equally well on the test set, see Fig. 3d.

To gather further evidence on whether the SAM solution is a saddle point, we analyze its Hessian eigenvalue density, following Ghorbani et al. [28] and using the Stochastic Lanczos Quadrature algorithm [29]. Fig. 7 shows the density, including significant probability mass for both positive and negative eigenvalues, indicating that the solution is a saddle point.

A.3 Why does SWA not work well on NLP tasks?

In Fig. 4a, we saw that SWA had only a mild effect on the generalization performance of NLP tasks, and sometimes even decreases it. Here, we seek to investigate why that is.

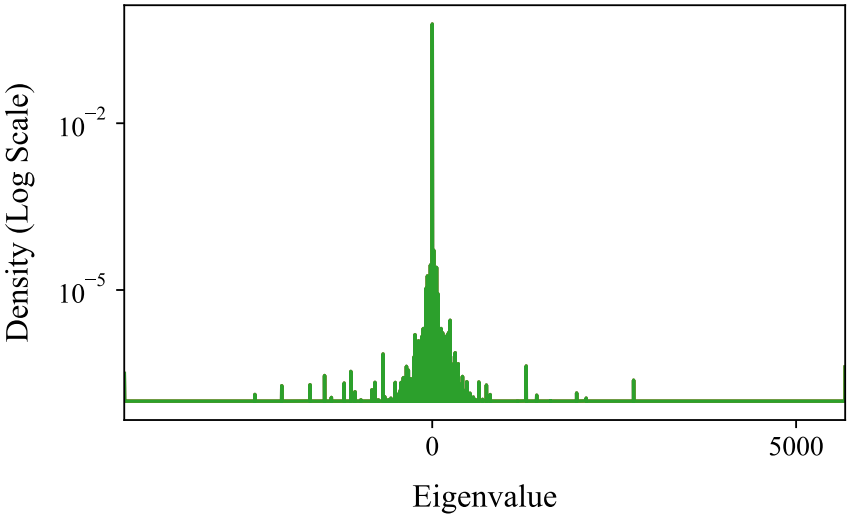


Figure 7: **Hessian Eigenvalue Density [28] of SAM's GIN-Code2 solution**: We observe significant probability mass around both positive and negative eigenvalues, indicating that this solution is a saddle point.

We consider two tasks: (i) the RTE task, for which SWA decreases the performance by around $-0.23_{\pm 0.20}$ compared to Adam, (ii) the QNLI task, for which SWA decreases the performance by $-0.08_{\pm 0.11}$. In both cases, SAM improved the performance statistically significantly over Adam.

For the QNLI task in Fig. 8a, we observe that SWA finds a lower/higher training loss/accuracy than Adam, respectively. However, the test loss/accuracy is higher/lower at the SWA solutions and the loss functions seem less well correlated in between both solutions (i.e, for $\alpha \in [0, 1]$).

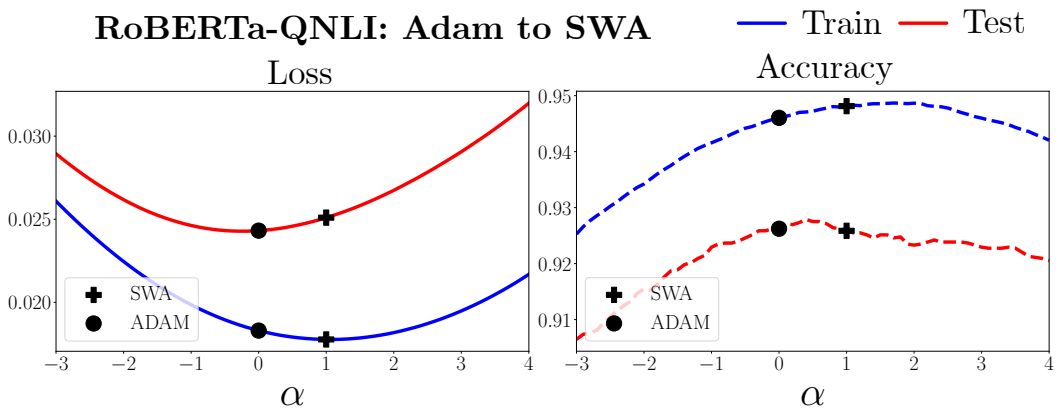
For the RTE task in Fig. 8b, we note that SWA finds a solution that is closer to a sharply increasing side. This may happen if the baseline optimizer skips or goes around sharper solutions (e.g., due to large step sizes) and the average pulls it towards these suboptimal regions.

Further, in Table 3, we notice very high values of $s_{\text{CKA}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SWA}})$, $s_{\text{cosine}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SWA}})$ for both training and test sets, indicating that the predictions are indeed very similar. In contrast, $s_{\text{CKA}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SAM}})$, $s_{\text{cosine}}(\boldsymbol{\theta}^{\text{NF}}, \boldsymbol{\theta}^{\text{SAM}})$ is lower, especially for the test set.

A.4 Why does SAM not work well on GRL tasks?

Fig. 10 shows the interpolations between the Adam and SAM solution. We do not observe a significant loss/accuracy difference between the two different basins. One possible explanation for this phenomenon is that the loss surface for this task is "globally well-connected" [91], yielding many basins with very similar geometric properties.

(a) NLP: RoBERTa-QNLI



(b) NLP: RoBERTa-RTE

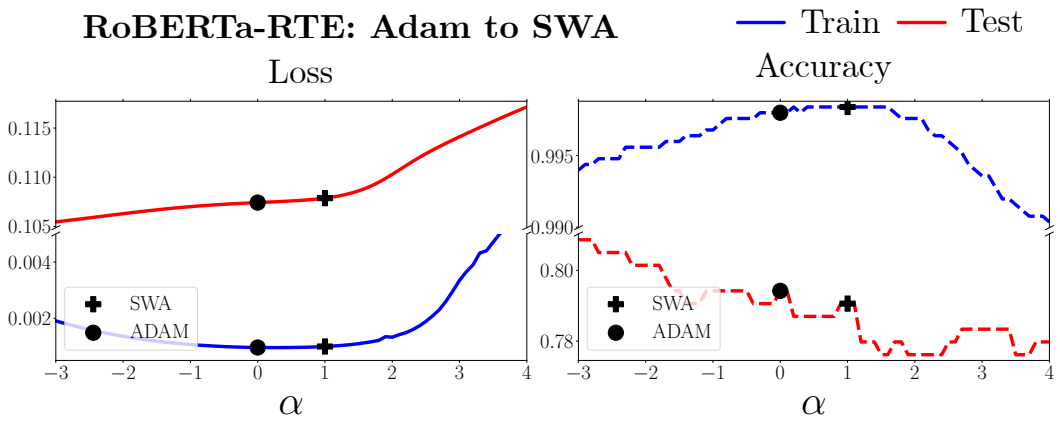


Figure 8: Training (blue) and test (red) losses (—) and accuracies (·····) of linear interpolations $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta} + \alpha\boldsymbol{\theta}'$ between Adam solutions ($\bullet, \alpha = 0.0$) and SWA ($+, \alpha = 1.0$).

Figure 9: GPP: Molpcba-GIN

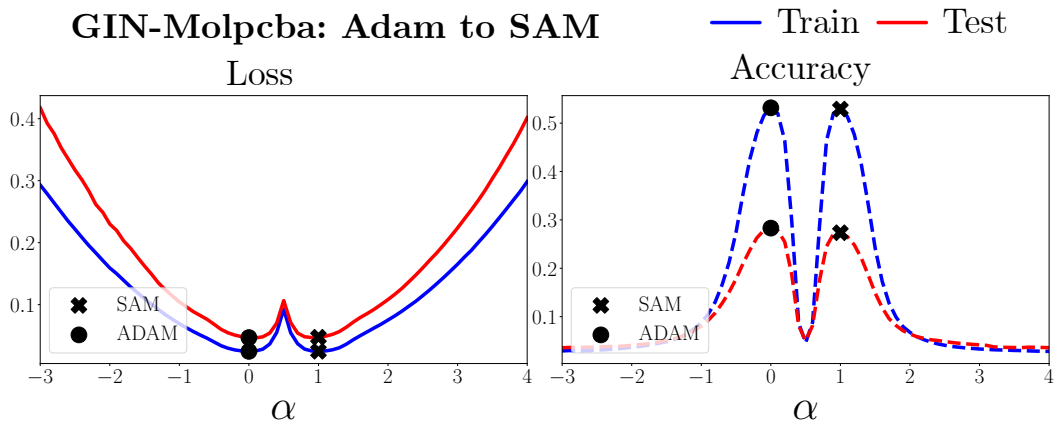


Figure 10: Training (blue) and test (red) losses (—) and accuracies (·····) of linear interpolations $\boldsymbol{\theta}(\alpha) = (1-\alpha)\boldsymbol{\theta} + \alpha\boldsymbol{\theta}'$ between Adam solutions ($\bullet, \alpha = 0.0$) and SAM ($\times, \alpha = 1.0$).

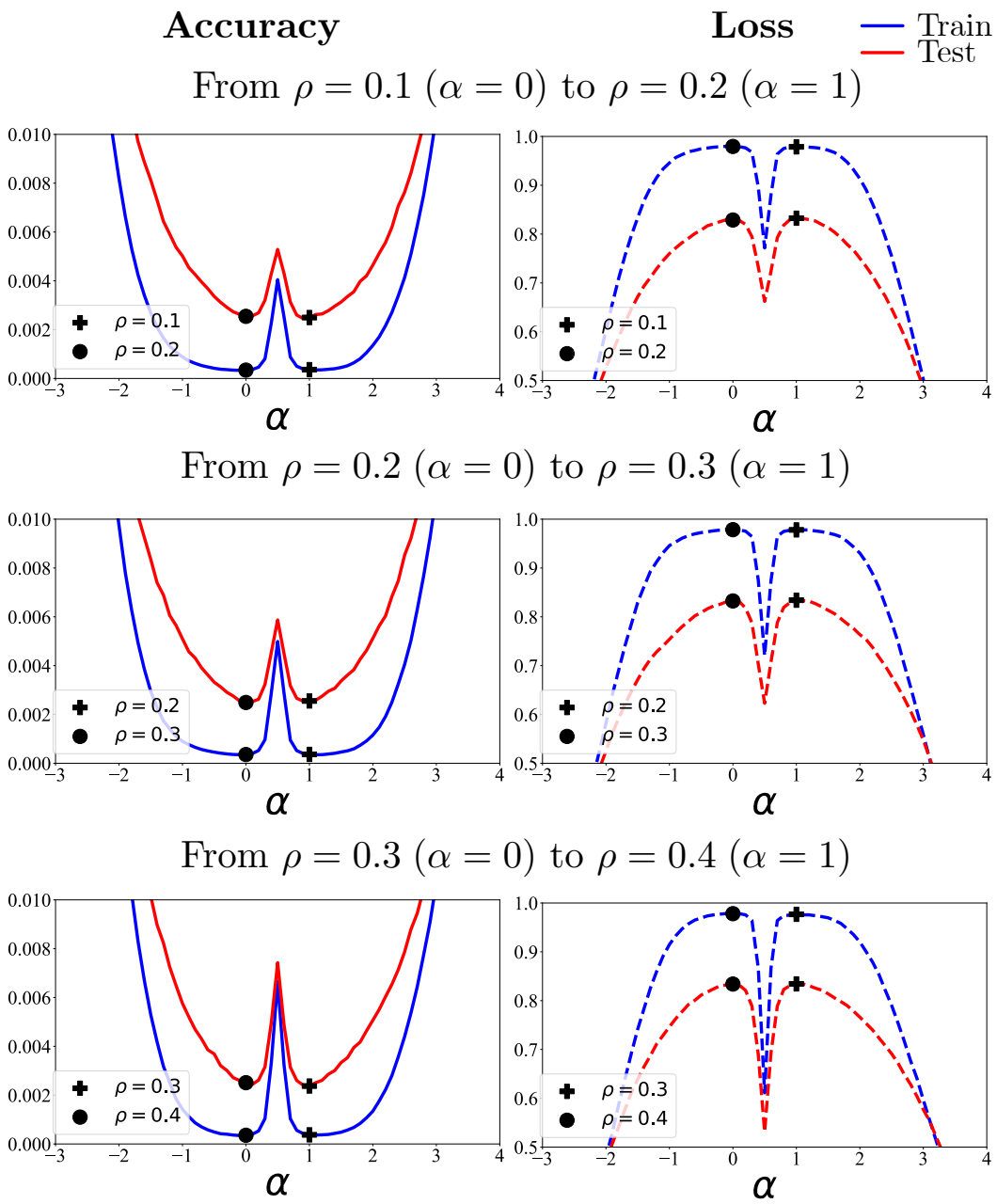
A.5 Does changing SAM's ρ result in different basins?

Here, we plot linear interpolations of solutions obtained by smaller and larger ρ values. Overall, we find that all solutions seem to lie in different basins indicated by high loss barriers in between ($\alpha = 0.5$) them.

A.5.1 WRN-28-10

For the WRN-28-10 model investigated in Sec. 3.1 and Sec. 4, we set $\rho = 0.1$ (as determined by hyper-parameter tuning on validation loss).

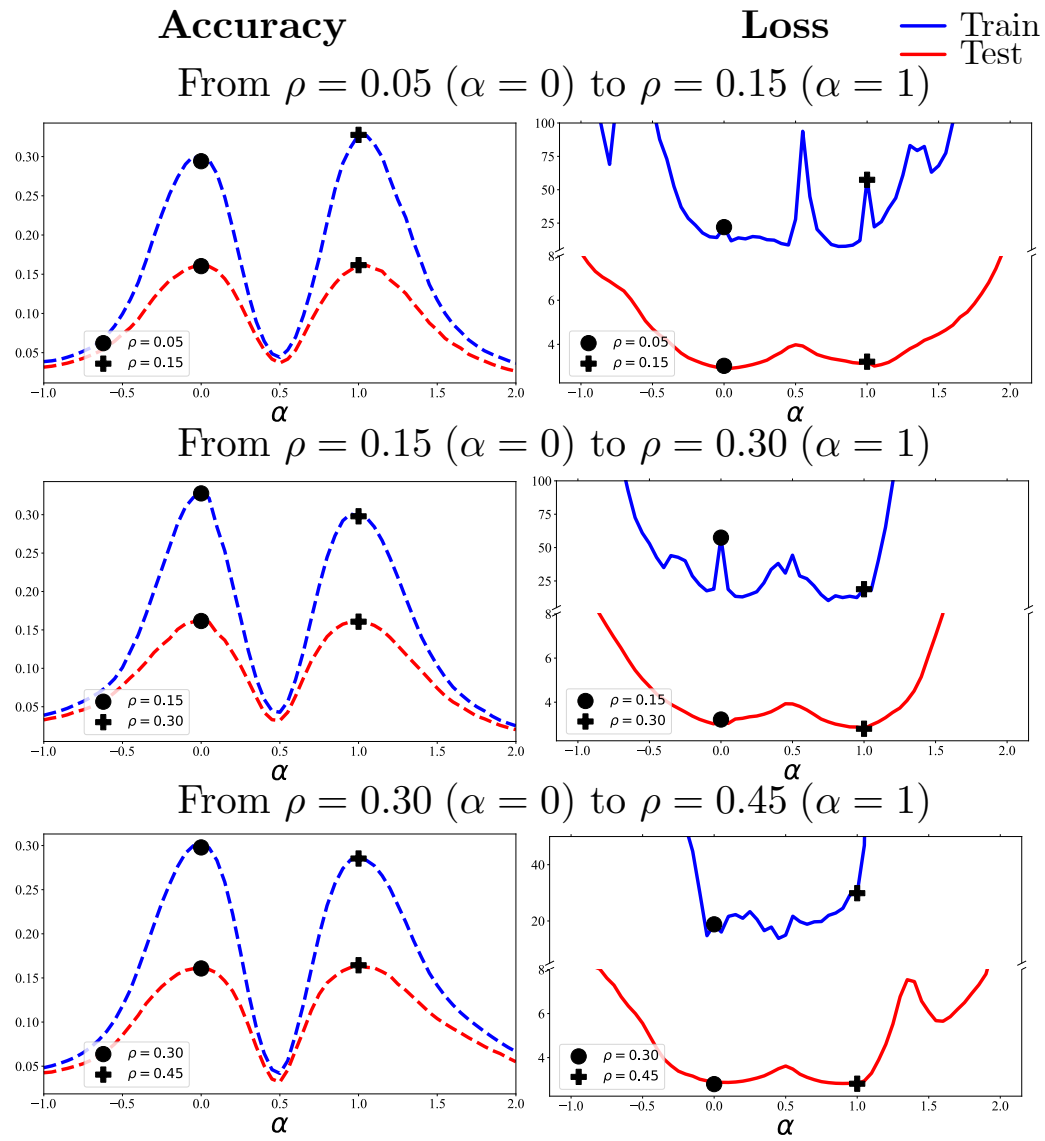
Figure 11: **WRN-28-10**: Changing SAM's ρ result in different basins.



A.5.2 GIN-Code2

For the GIN model investigated in Sec. 3.1 and Sec. 4, we set $\rho = 0.15$ (as determined by hyper-parameter tuning on validation loss).

Figure 12: **GIN-Code2**: Changing SAM's ρ result in different basins.



A.6 Does changing the base optimizer impact SWA and SAM's effectiveness?

The influence of the base optimizer on SWA/SAM's effectiveness is under-explored. In the previous experiments, we use the default base optimizers from existing code repositories or, as reported in previous works. Here, we want to conduct an initial investigation into its effect on SWA and SAM.

In the following experiments, we only switch the base optimizer and keep everything else fixed (including hyper-parameters such as learning rate, etc.). We train (i) a ResNet-34 (similar to WRN-28-10 but smaller) on CIFAR100, once per SGD with momentum and AdamW [67], and (ii) a GIN model on Code2, as in Sec. 3, but using RMSprop instead of Adam. We choose AdamW and RMSProp, since they are commonly used in image classification [67, 19, 82] and graph representation learning (GRL) [65, 64, 75], respectively.

Due to time constraints, we only report results obtained with one random seed (except for GIN-Code2 with Adam, which we already evaluated across three random seeds in our initial submission, see Fig. 4b). Further, again due to time constraints, for the ResNet-34 task, we do not conduct a hyper-parameter search of SAM's ρ but set it to $\rho = 0.05$, as this value has been reported to be a good default value [22].

Table 4 show the test performances with switched base optimizers. First, discussing task (i), we note that AdamW under-fits the model and generalizes poorly compared to SGD. Here, SAM exacerbates the performance even further, performing even worse than the AdamW baseline. The reasons for that are unclear, and we leave an investigation into them for future work.

For task (ii) using RMSprop, we observe that both flat-minima optimizers improve over the baseline performance. However, compared to when using Adam, they perform even more similarly, with SAM only being 0.03% better. Interestingly, the combination SWA+SAM again performs best.

Table 4: **Test accuracies/F1 score of switched base optimizers**.

Task	Baseline	SWA	SAM	SWA+SAM
ResNet34 on CIFAR100 (SGD)	76.14	+ 1.50	+1.91	**+ 2.60**
ResNet34 on CIFAR100 (AdamW)	72.14	**+ 0.57**	-2.29	-1.11
GIN on Code2 (Adam)	$15.73_{\pm 0.11}$	$+ 0.83_{\pm 0.11}$	$+ 0.57_{\pm 0.09}$	$\mathbf{+ 1.10_{\pm 0.09}}$
GIN on Code2 (RMSprop)	15.30	+ 0.67	+ 0.70	**+ 1.62**

B Experimental details

B.1 Computer Vision

We mostly adapt the hyper-parameter values from Foret et al. [22] for WRN-28-10 and PyramidNet-272, from Dosovitskiy et al. [19] for ViT, and from [81] for MLP-Mixer models. We average all results across three random seeds.

B.1.1 Supervised Classification

We train WideResNets [94] with 28 layers and width 10 (WRN28-10) and PyramidNet [36] with 110 layers and widening factor $\alpha = 272$ (PyramidNet-272) from scratch. The Vision Transformer (ViT) base model with input patch size 16 (ViT-B/16) and MLP-Mixer base model with input patch size 16 (MLP-Mixer-B/16) start from pre-trained checkpoints available at https://console.cloud.google.com/storage/vit_models/. The reason for using pre-trained checkpoints for the ViT and MLP-Mixer models is that, due to their lack of some inductive biases inherent to CNNs, such as translation equivariance and locality, they do not generalize well when trained on insufficient amounts of data [19]. Table 5 shows the hyper-parameters for each architecture.

B.1.2 Self-Supervised Learning

Table 6 shows the hyper-parameters for each SSL method. We use implementations from the lightly package, available at https://github.com/lightly-ai/lightly [80].

Table 5: Hyper-parameters for Supervised Classification (SC): CIFAR-{10, 100} (Table 2)

Hyper-Parameter	**WRN28-10**	**PyramidNet-272**	**ViT-B/16**	**MLP-Mixer-B/16**
Base Optimizer	SGD	SGD	SGD	SGD
Batch size	256	256	100	170
Data augmentation	Inception-style + Cutout [16]			
Dropout rate	0.0			
Epochs	200	200	–	–
Gradient clipping norm	–	–	1.0	1.0
Learning rate schedule	cosine			
Peak learning rate	0.1	0.05	0.03	0.03
Steps	–	–	12500	12500
SGD Momentum	0.9			
Warmup steps	–	–	500	500
Weight decay	$5e-4$	$5e-4$	0.0	0.0
CIFAR-10				
SAM ρ	0.05	0.05	0.1	0.02
Averaging start E (SWA)	60%	60%	75%	90%
Averaging start E (WASAM)	90%	75%	75%	90%
CIFAR-100				
SAM ρ	0.1	0.1	0.2	0.05
Averaging start E (SWA)	60%	60%	75%	90%
Averaging start E (WASAM)	90%	75%	75%	90%

Table 6: Hyper-parameters for Self-Supervised Learning (SSL): CIFAR-10, ImageNette, results in (Table 2)

Hyper-Parameter	**MoCo**	**SimCLR**	**SimSiam**	**BarlowTwins**	**BYOL**	**SwaV**
Backbone Network	ResNet-18					
Base Optimizer	SGD					Adam
Data augmentation	SimCLR [10]					Multi-Crop [7]
Dropout rate	0.0					
Epochs	800					
Embedding dimensions	512					
KNN memory bank size	4096					
Learning rate schedule	cosine					
Peak learning rate	$6e-2$					$1e-3$
SGD Momentum	0.9					–
Weight decay	$5e-4$					$1e-6$
CIFAR-10						
Batch size	512					
Crop size	–					32
Gaussian blur	0%					
SAM ρ	0.01	0.01	0.01	0.05	0.01	0.05
Averaging start E (SWA)	75%	90%	75%	90%	60%	60%
Averaging start E (WASAM)	90%	90%	90%	90%	75%	90%
ImageNette						
Batch size	256					
Crop size	–					128, 64
Gaussian blur	50%					
SAM ρ	0.01	0.01	0.02	0.05	0.05	0.01
Averaging start E (SWA)	50%	90%	75%	75%	90%	50%
Averaging start E (WASAM)	50%	50%	90%	75%	90%	50%

B.2 Natural Language Processing

For the task of Open Domain Question Answering, we adapt the hyper-parameter values and the 25 retrieved passages for each question from 44. We report the Exact Match score of FiD-base model on Natural Questions (NQ) and TriviaQA test sets. For GLUE benchmark, we report Matthew's Corr for CoLA, Pearson correlation coefficient for STSB, and accuracy for the the rest of the datasets. Results are all evaluated on the dev set of GLUE benchmark. We use the RoBERTa-base as our backbone language model, implemented with Huggingface Transformers [87]. Most of the task-specific hyper-parameter values are adapted from 1.

B.3 Graph Representation Learning

We mostly adapt the hyper-parameter values from Hu et al. [42] for GCN [52], SAGE [35], and GIN [90], from Chen et al. [13] for [60] and ComplEx [83], and from Li et al. [62] for DGCN. Due to high standard errors, we averaged the results of a few tasks more than three times, as mentioned in the following tables.

Table 7: Hyper-parameters for NPP tasks, results in Fig. 4b.

Hyper-Parameter	**SAGE**	**DGCN**
NPP: OGB-Proteins		
Aggregation method	Mean	Softmax
Base optimizer	Adam	Adam
Convolution layer	SAGE	DyResGEN
Dropout rate	0.0	0.1
Hidden dimensions	256	64
Learning rate	0.01	0.001
Normalization layer	–	Layer norm
Number of epochs	2000	1000
Number of layers	3	112
Number of random seeds	5	3
Training cluster number	1	15
Weight decay	0.0	0.0
SAM ρ	0.01	0.02
Averaging start E (SWA)	90%	90%
Averaging start E (WASAM)	90%	90%
NPP: OGB-Products		
Aggregation method	Mean	Softmax
Base optimizer	Adam	Adam
Batch size	20000	–
Convolution layer	SAGE	Gen
Dropout rate	0.5	0.5
Evaluation cluster number	–	8
Learning rate	0.01	0.001
Hidden dimensions	256	128
Normalization layer	–	Batch norm
Number of epochs	30	50
Number of layers	3	14
Number of random seeds	5	3
Training cluster number	–	10
Weight decay	0.0	0.0
SAM ρ	0.01	0.02
Averaging start E (SWA)	90%	60%
Averaging start E (WASAM)	75%	90%

Table 8: Hyper-parameters for GPP: OGB-Code2, results in Fig. 4b.

Hyper-Parameter	**GCN**	**GIN**
GPP: OGB-Code2		
Aggregation method	Mean	Mean
Base optimizer	Adam	Adam
Batch size	128	128
Convolution layer	GCN	GIN
Dropout rate	0.0	0.0
Learning rate	0.001	0.001
Hidden dimensions	300	300
Normalization layer	Batch norm	Batch norm
Number of random seeds	3	3
Number of epochs	15	30
Number of layers	5	5
Virtual node embeddings	True	True
Vocabulary size	5000	5000
Weight decay	0.0	0.0
SAM ρ	0.2	0.15
Averaging start E (SWA)	50%	50%
Averaging start E (WASAM)	50%	50%

Table 9: Hyper-parameters for GPP: OGB-Molpcba, results in Fig. 4b.

Hyper-Parameter	**GIN**	**DGCN**
GPP: OGB-Molpcba		
Aggregation method	Mean	Mean
Batch size	512	512
Base optimizer	Adam	Adam
Convolution layer	GIN	GEN
Dropout rate	0.0	0.2
Learning rate	0.001	0.001
Normalization layer	Batch norm	Batch norm
Number of epochs	100	50
Number of layers	5	14
Number of random seeds	3	3
Hidden dimensions	300	256
Virtual node embeddings	False	True
Weight decay	0.0	0.0
SAM ρ	0.01	0.15
Averaging start E (SWA)	90%	75%
Averaging start E (WASAM)	90%	50%

Table 10: Hyper-parameters for LPP: OGB-Biokg, results in Fig. 4b.

Hyper-Parameter	**CP**	**ComplEx**
GPP: OGB-Biokg		
Base optimizer	Adam	Adam
Batch size	500	500
Learning rate	0.1	0.1
Number of random seeds	3	3
Number of epochs	30	50
Rank	1000	1000
Regularizer	N3	N3
Weight decay	0.0	0.0
SAM ρ	0.1	0.05
Averaging start E (SWA)	50%	50%
Averaging start E (WASAM)	90%	50%

Table 11: Hyper-parameters for LPP: OGB-Citation2, results in Fig. 4b.

Hyper-Parameter	**GCN**	**SAGE**
GPP: OGB-Citation2		
Aggregation method	Mean	Mean
Base optimizer	Adam	Adam
Batch size	256	512
Convolution layer	GCN	SAGE
Dropout rate	0.0	0.2
Hidden dimensions	256	256
Number of epochs	300	300
Number of layers	3	3
Number of random seeds	3	3
Normalization layer	–	–
Learning rate	0.001	0.0005
Virtual node embeddings	False	False
Weight decay	0.0	0.0
SAM ρ	0.02	0.01
Averaging start E (SWA)	75%	90%
Averaging start E (WASAM)	60%	90%