Towards an *Even Better* Understanding of Sharpness-Aware Minimisation

Antonia Boca[*], **Kian Cross**[†], **Halfdan Holm**[‡], **Caner Korkmaz**[§]
Department of Computer Science and Technology
University of Cambridge
{aib36,kc642,hsh39,dk665}@cl.cam.ac.uk

Abstract

Sharpness-aware minimisation (SAM) optimises for flat minima and has been shown in many instances to improve generalisation performance. Yet, it remains unclear why and under precisely which circumstances this is the case. We hypothesise that the asymmetry of a minimum, in combination with its sharpness, plays an important role. We train models with stochastic gradient descent (SGD), SAM, and average SAM, using various hyperparameters to investigate this. We introduce the *asymmetry factor*, a novel metric to quantify the asymmetry of minima. We find that this value correlates with test loss, implying that asymmetry is indeed a relevant metric for good generalisation. Furthermore, we identify that variants of SAM with small hyperparameter changes find minima in different loss basins, implying that there is a bias parameterised by a small set of variables. We investigate this further, discovering that in some instances, SAM is drawn to minima with a higher sharpness than that of other optimisers. Our results offer novel insights into the elusive behaviour of SAM, providing some tentative answers to questions about when and why it works.

1 Introduction

Sharpness-aware minimisation (SAM) is an optimisation method which penalises sharp minima during the training of neural networks. It has previously been shown that models trained with SAM using smaller batches have better generalisation performance than full-batch SAM [2]. Furthermore, the same work noted that using the average sharpness across the loss surface — rather than the sharpness in the steepest direction — performs poorly. Extending the notion of sharpness, it has been argued that asymmetric minima also provide better generalisation, resulting in training mechanisms which optimise specifically for this [22].

We hypothesise that the poor generalisation performance when using average sharpness is due to it simultaneously optimising for symmetric minima, which we already know induces poor performance [22]. More specifically, average sharpness considers *all* directions of the loss surface,

[*]Wrote the second half of the literature review and adapted code to plot loss landscapes, mode connectivity and 3D visualisations of minima in order to guide exploratory work.

[†]Wrote the first half of the literature review and carried out the experiments and analysis relating to asymmetry, including writing the necessary code.

[‡]Helped do training runs and figure out which model and dataset worked best (see Appendix B). Did sharpness experiments and analysis.

[§]Wrote and debugged the code pipeline we used for training the models, and adapted/implemented the optimizers (SAM/average-SAM) and their respective training algorithms. Carried out training experiments.

Submitted to form a part of the 'R252: Theory of Deep Learning' module assessment (January 2023). All authors contributed equally, with names ordered alphabetically by last name. Code artefacts for this paper are available on various branches at: https://github.com/Kausta/R252_SAM/tree/sharpness.

penalising minima in which some directions are sharp. Still, others are very flat: the very minima which have been shown to perform well. We extend this hypothesis as a possible explanation for the poor generalisation performance of large-batch SAM.

In this work, we analyse the sharpness and asymmetry of models trained with SAM-based optimisers using various hyperparameters. We introduce the *asymmetry factor* and find that this measure correlates well with test loss, with small-batch SAM being the optimiser biased most towards asymmetric minima. Additionally, we present experiments indicating that SAM optimisers with small hyperparameter changes end up in different loss basins, which we hypothesise happens due to them having differing asymmetry biases. Within this framework, we believe that small-batch SAM has the strongest bias towards such minima, leading to the best generalisation performance.

Contributions. In this work, we make the following contributions:

1. We provide a comprehensive review of the literature on flat minima, the techniques to find them, and other work relating to SAM.
2. We reproduce the SAM optimisation technique under two scenarios: optimising for adversarial sharpness and optimising for average sharpness. We apply these to both large-batch and small-batch training.
3. We investigate whether SAM optimisers with small hyperparameter changes end up in different basins of the loss landscape. We provide examples of models which do, in fact, belong to separate basins of test loss, indicating that small-batch SAM has a different bias than other models.
4. We analyse which optimisers minimize various sharpness measures best and which sharpness measures correlate best with test error.
5. We introduce an asymmetry factor and show that this value correlates with test loss. We further show that small-batch adversarial SAM finds asymmetric minima, suggesting that asymmetry plays an important part in the generalisation powers of SAM.

Contents. The remainder of this paper is structured as follows. Section 2 provides a comprehensive review of other work relating to SAM. We provide an overview of both the efforts to improve its empirical results and the attempts to understand SAM's elusive behaviour. In Section 3, we present a refresher on the SAM technique, providing a description of n-SAM, m-SAM, and average SAM. Section 4 outlines our methods to carry out our work, including a description of our model training procedure, the calculation of sharpness, and an explanation of our novel *asymmetry factor*. Our results and an accompanying discussion and interpretation can be found in Section 5. We conclude in Section 6 by outlining the limitations of our work and suggesting future areas of exploration.

2 Related Work

Optimising neural networks to find minima with flat loss geometries dates back to at least 1994 when Hochreiter and Schmidhuber proposed an algorithm to search for connected regions in weight space with approximately constant error [23, 24, 25]. The authors justified their approach using Bayesian arguments [7] — in particular the minimum message [50] and minimum description [46] length principles — which they used to imply that flatter mimima should result in better generalisation. Indeed, their empirical results showed this to be true. Numerous works have since identified the importance of smaller stochastic gradient descent (SGD) [5, 20] batch sizes for attaining good generalisation, also noting that smaller batches tend to result in flatter minima [27, 32, 53, 54]. Similarly, Jiang et al. observed that out of 40 complexity measurements, sharpness-based measures correlated most with generalisation [28]. In the search for flat minima, researchers have taken several different directions.

Weight Averaging. Izmailov et al. developed a training technique referred to as stochastic weight averaging (SWA) [26], which starts with a pretrained model $\boldsymbol{w}^*$ and continues training for a certain number of epochs, taking regular snapshots to obtain $\{\boldsymbol{w}_1^*, \boldsymbol{w}_2^*, \ldots, \boldsymbol{w}_n^*\}$. These are averaged, resulting in a new model $\boldsymbol{w}_{\text{SWA}}^*$ with improved test accuracy and thus better generalisation when compared to the base SGD optimiser. The authors found that whilst under SGD the weights often converged close to the boundaries of wide flat regions, SWA found weights centred in these regions.

Weight Perturbation. Murray and Edwards in 1992 along with Jim, Giles, and Horne in 1996 discovered that perturbing model weights with noise could result in better generalisation [29, 42]. Wen et al. proposed SmoothOut [51], which perturbs multiple copies of the model's weights and averages the results, producing flatter minima and resulting in better generalisation. Gradient Noise Convolution (GNC) [21] proposed by Haruki et al. applies anisotropic gradient noise to the weights rather than random noise, which they argue better smooths sharp minima, resulting in improved generalisation.

Adversarial Weight Perturbation. Along similar lines to the previous methods, some authors have perturbed the weights with an adversarially derived value [16, 52, 58]. These methods — based on different theoretical underpinnings — all result in similar training algorithms, which attempt to minimise both loss and sharpness. They achieve this by ascending the loss landscape in the steepest direction by a certain amount, using this as the minimisation value. Wu, Xia, and Wang [52] and also Foret et al. [16] use a PAC-Bayesian generalisation bound [41] to form their theoretical justification, with the former using a proof technique inspired by [43] and the latter by [8]. On the other hand, Zheng, Zhang, and Mao [58] assume that the loss surface can be modelled as an inverted Gaussian surface to prove that their minimisation objective favours flatter local minima. All algorithms in this class could be described as performing *sharpness-aware minimisation* (SAM); however, Foret et al. specifically use this term to describe their implementation, and so we use SAM here on to refer to the algorithm given in [16].

Other. Chaudhari et al. proposed Entropy-SGD [9], in which a local-entropy-based objective function that favours minima in flat regions is optimised, producing well-generalising solutions. Karakida, Akaho, and Amari observed that in wide neural networks and under certain conditions, applying batch normalization in the final layer can decrease pathological sharpness [31]. Lin et al. proposed a method with performance comparable to large-batch training but with generalisability and flatness of minima similar to mini-batch training [37]. Many of the same authors also developed EXTRAP-SGD [38], which uses a local extra gradient method to smooth the loss surface and avoid sharp minima during large-batch training.

Despite the good empirical results, it still is not yet fully understood why, or even if [45] sharpness is required for good generalisation. For example, it has been shown that models can be reparameterised to have arbitrarily sharp minima whilst remaining equivalent to the original [12]. Furthermore, Andriushchenko and Flammarion showed that optimising for flat minima alone is not sufficient: batching is also required [2]. This was investigated more thoroughly by Behdin et al., who concluded that this batched variation of SAM yields superior generalisation performance and flatter minima [4]. The implication of the work by Andriushchenko and Flammarion is that there is an implicit bias in many of the optimisers used to find flat minima, which also often results in models with good generalisation performance.

Numerous works [2, 3] have used linear or small single-layer non-linear networks to investigate the flat minima phenomena, once again demonstrating the bias of SGD towards flat areas [15]. Kaddour et al. compared SAM with SWA, finding that SAM models are closer to sharp directions than SWA models, that SAM models are flatter, and that in most instances, SAM has better test performance [30]. There have also been various advocates for a Pac-Bayesian definition of sharpness, with those in favour claiming it is robust to problems — such as parameter re-scaling — that other definitions are susceptible to [43, 56]. Motivating a large part of our work was the paper by He, Huang, and Yuan, which showed that the asymmetry of minima must be considered in addition to sharpness [22]. They further showed that SGD is biased towards asymmetric minima.

The remainder of this section reviews SAM variants, which generally fall into two categories: improving the computational overhead of SAM (Section 2.1) and addressing the inadequacy of the sharpness measure (Section 2.2).

2.1 Improving the Efficiency of SAM

Du et al. proposed ESAM [13], which requires 40% additional computational overhead compared to the base optimiser. ESAM uses stochastic weight perturbation (SWP) to approximate sharpness by searching for weight perturbations within a stochastically chosen neighbourhood of the current weights. Sharpness-sensitive data selection (SDS) then approximately optimises the weights based on

sharpness-sensitive subsets of batches. Du et al. improve upon this performance with their proposal, SAF [14], which does not require any additional overhead compared to the base optimiser. They achieve this by proposing an alternative loss measurement for sharpness, which relates to weights in previous layers of the network and avoids large variations in the minima. Whilst SAF does not increase the computational overhead, it requires additional memory to remember past network weights. The authors also proposed memory-efficient SAF (MESA), which can be used for very large models.

Zhao, Zhang, and Hu introduce SS-SAM [57], which switches between the SAM and ERM objective according to a Bernoulli trial. Liu et al. propose computing a new sharpness measure every five iterations. Their method, LookSAM [39], reuses the computed gradients for the other four iterations, which are obtained by decomposing SAM's updated gradients into two orthogonal directions. The authors show that one of these directions tends to stay the same over multiple iterations and can therefore be reused at no cost to the generalisation performance.

2.2 Alternative Sharpness Measures for SAM

Another line of work argues that the sharpness measure used in [16] is inadequate. For example, Zhuang et al. argue the perturbed loss used in SAM can appear in both flat and sharp minima [59]. To address this, they define the *surrogate gap*, a measure equivalent to the domain eigenvalue of the Hessian at a local minimum. Intuitively, this gap represents the difference between the maximum loss within the radius and the loss at the centre point. They report that minimising this gap with the original SAM objective (i.e. the GSAM objective) leads to better generalisation.

Kwon et al. note that the SAM sharpness is sensitive to parameter re-scaling and propose Adaptive SAM (ASAM) [35]. Intuitively, this method redefines the neighbourhood radius by rescaling parameters based on their magnitude. Kim et al. criticise this approach, arguing that it does not reflect the underlying geometry of the parameter manifold [33]. They note that the loss functions of neural networks are dependent on model parameters θ (e.g. $l(\theta) = \mathbb{E}_{(x,y)}[-\log p(y|x,\theta)]$). Therefore, the geometry of the space is not Euclidean, but rather a *statistical manifold* induced by the Fisher information metric of the distribution $p(y|x,\theta)$ [1]. They introduce the FSAM objective and use a minibatch approximation in practice.

Open Questions. Interestingly, methods that approximate the original SAM objective appear to perform better in practice [13, 39] (see Appendix A). Furthermore, all authors build the mathematical background of their work assuming full-batch optimisation of their objective, but in practice, train neural networks using mini-batches. These observations suggest that there is an implicit bias related to small-batch training, which further improves generalisation performance.

3 Background

In this section, we provide a brief refresher of the SAM method, as originally outlined in [16].

Consider a joint distribution $\mathcal{D}(\boldsymbol{x}, \boldsymbol{y})$ over $\mathcal{X}$ and $\mathcal{Y}$, a model parameterised by $\boldsymbol{w} \in \mathcal{W} \subseteq \mathbb{R}^d$, and a per-data-point loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$. Then the population loss is given by

$$L_{\mathcal{D}}(\boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim\mathcal{D}}\left[l(\boldsymbol{w}, \boldsymbol{x}, \boldsymbol{y})\right].$$

For a given a training data set $\mathcal{S} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i) \sim \mathcal{D} : i = 1, 2, \ldots, n\}$ containing independent and identically distributed elements drawn from $\mathcal{D}$, the population loss $L_{\mathcal{D}}(\boldsymbol{w})$ can be estimated by

$$L_{\mathcal{S}}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n} l(\boldsymbol{w}, \boldsymbol{x}_i, \boldsymbol{y}_i).$$

Using an optimisation procedure, the objective is to find

$$\boldsymbol{w}^* = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathcal{W}} L_{\mathcal{S}}(\boldsymbol{w}) \tag{1}$$

such that

$$L_{\mathcal{S}}(\boldsymbol{w}^*) \approx L_{\mathcal{D}}(\boldsymbol{w}^*). \tag{2}$$

When (2) holds, it is said that the model parameterised by $\boldsymbol{w}^*$ *generalises well*. Yet, often $L_{\mathcal{S}}(\boldsymbol{w}^*) \ll L_{\mathcal{D}}(\boldsymbol{w}^*)$. This is often due to the model being overparameterised and $L_{\mathcal{S}}(\boldsymbol{w})$ being non-convex in $\boldsymbol{w}$, resulting in multiple local and global minima. Thus, there may be numerous $\boldsymbol{w}^*$ which satisfy (1), with only a subset of these also satisfying (2). We cannot optimise directly for (2) and so the question becomes *how can the optimisation procedure be improved to select from the numerous $\boldsymbol{w}^*$ those which also satisfy* (2)*?*

As discussed in Section 2, it has been observed that solutions $\boldsymbol{w}^*$ found in flatter minima of $L_{\mathcal{S}}(\boldsymbol{w})$ tend to generalise better [28]. This has prompted the development of methods which penalise sharp minima during the training process, such as SAM [16].

n-SAM. The premise of the SAM algorithm is the following theorem, which states informally that for any $\rho > 0$, with high probability over the training set $\mathcal{S}$ generated from the distribution $\mathcal{D}$,

$$L_{\mathcal{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) + h\left(\frac{\|\boldsymbol{w}\|_2^2}{\rho^2}\right),$$

where $h : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ is a strictly increasing function. By substituting h for a hyperparameter λ, this becomes the optimisation problem

$$\arg\min_{\boldsymbol{w} \in \mathcal{W}} L_{\mathcal{S}}^{\text{SAM}}(\boldsymbol{w}) + \lambda \|\boldsymbol{w}\|_2^2$$

where

$$L_{\mathcal{S}}^{\text{SAM}}(\boldsymbol{w}) = \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}). \tag{3}$$

The value of $\boldsymbol{\epsilon}$ corresponding to (3) can be computed by

$$\hat{\boldsymbol{\epsilon}} = \frac{\rho \operatorname{sign}\left(\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})\right) |\nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|^{q-1}}{\left(\|L_{\mathcal{S}}(\boldsymbol{w})\|_q^q\right)^{\frac{1}{p}}},$$

which after substituting into (3), differentiating, and dropping second order terms yields

$$\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{\text{SAM}}(\boldsymbol{w}) \approx \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}}.$$

Under a full-batch minimisation method, such as gradient descent, the weight update procedure becomes

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma_t \nabla_{\boldsymbol{w}_t} L_{\mathcal{S}}(\boldsymbol{w}_t)|_{\boldsymbol{w}_t+\hat{\boldsymbol{\epsilon}}},$$

where γ_t is the learning rate at time t. This is known as n-SAM.

m-SAM. In practice, a subset of training data is used at each iteration under a minimisation algorithm such as SGD. In these cases, (3) becomes

$$L_{\mathcal{S}}^{\text{SAM}}(\boldsymbol{w}) = \sum_{\mathcal{U} \subset \mathcal{S}} \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{U}}(\boldsymbol{w} + \boldsymbol{\epsilon}),$$

where $|\mathcal{U}| = m$, $L_{\mathcal{U}}$ is identical to $L_{\mathcal{S}}$ but with the summation over $\mathcal{U}$, and $\boldsymbol{\epsilon}$ has an implicit dependency on $\mathcal{U}$, which has been suppressed for notational ease. The weight update procedure then becomes

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\gamma_t}{m} \nabla_{\boldsymbol{w}_t} L_{\mathcal{U}_t}(\boldsymbol{w}_t)|_{\boldsymbol{w}_t+\hat{\boldsymbol{\epsilon}}},$$

where $\mathcal{U}_t$ is the batch at time t. This is known as m-SAM.

Despite n-SAM being derived theoretically, Andriushchenko and Flammarion [2] show that not only does m-SAM provide better generalisation capabilities, but it is essential to yielding any improved generalisation at all.

Average SAM. Instead of calculating perturbation $\hat{\boldsymbol{\epsilon}}$ adversarially, random perturbations can be used. This is known as average SAM. In this case, $\hat{\boldsymbol{\epsilon}}$ is calculated as a sample drawn over a multivariate diagonal standard normal distribution with fixed Euclidian 2-norm. That is,

$$\hat{\boldsymbol{\epsilon}} = \frac{\rho}{\|\boldsymbol{z}\|_2^2} \boldsymbol{z},$$

where $\boldsymbol{z} \sim \mathcal{N}(\mathbf{0},\, \boldsymbol{I}_d)$ [16]. Then the weight update procedure remains the same as m-SAM but with the new $\hat{\boldsymbol{\epsilon}}$.

4 Methods

We have trained numerous models, which have then been analysed in various ways. In this section, we describe the methods that have been employed. Section 4.1 describes the training procedure, model architecture, and data set that we have used. In Section 4.3, we explain how our analysis on sharpness has been performed, with an outline of how adversarial and average sharpness is calculated. Our method of plotting the loss landscape in a single direction and calculating the asymmetry factor is described in Section 4.4. Finally, Section 4.2 describes our multi-dimensional visualisation of the loss landscape and analysis of the mode connectivity of basins.

4.1 Training Models

Initially, we designed our experiments the same as in the original SAM implementation [16], using Wide-ResNets [55] with the CIFAR-10 dataset [34]. Recognising our limited computational resources, we planned to use shorter training sessions. However, after observing that SAM does not produce significant differences in short training sessions (see appendix B), we switched to the MobileNetV2 [48] network. As this network was designed with ImageNet [47] in mind, we selected a variant with better performance for the image size of the CIFAR-10 dataset.

We used the CIFAR10 dataset with basic data augmentations (horizontal flip, padding by four pixels, and random crop) and more advanced data augmentations (AutoAugment [10] and CutOut [11]) during training. Like the original implementation, we used SGD with Nesterov Momentum [49] for optimisation with a 0.1 learning rate, 0.9 momentum, 5×10^{-4} weight decay, cosine annealing learning rate schedule [40], and a gradient norm clipped to 5. As the loss function, we used Cross Entropy Loss. We used 128 batch size for m-SAM experiments and 5000 for n-SAM experiments. All were trained for 200 epochs.

Actual full-batch gradient descent would require a batch size of 50000 in the case of CIFAR-10, which would require multiple GPUs or other tricks such as gradient accumulation, which are challenging to implement in the case of SAM. Hence, we instead used 10 batches of size 5000 to approximate the n-SAM algorithm.[5] We implemented the SAM algorithm in PyTorch [44], based on the most popular PyTorch implementation found on GitHub [18] and the original JAX [6] implementation [16].

4.2 Multi-dimensional Loss Landscape Visualisation

As part of our exploratory work, we provide visualisations of the loss landscapes for some of the models we have trained by adapting code from [36]. With these visualisations, we can further understand the superior performance of m-SAM in training MobileNetV2 on CIFAR-10.

Mode Connectivity and Loss Basins. We believe different biases lead towards different loss basins, so we were interested in exploring whether different optimisers may find minima in the same basin of low loss. To carry this out, we used the method proposed by Garipov et al. [17], adapting their code to work with MobileNetV2. Formally, their method works as follows. Consider two sets of weights $\hat{\boldsymbol{w}}_1$ and $\hat{\boldsymbol{w}}_2$ (where $d = |\hat{\boldsymbol{w}}_1| = |\hat{\boldsymbol{w}}_2|$) and a curve $\phi_{\hat{\boldsymbol{\theta}}} : [0, 1] \to \mathbb{R}^d$ parameterised by $\hat{\boldsymbol{\theta}} \in \mathbb{R}^m$, where $m \in \mathbb{Z}^+$. Then subject to the fixed points $\phi_{\hat{\boldsymbol{\theta}}}(0) = \hat{\boldsymbol{w}}_1$ and $\phi_{\hat{\boldsymbol{\theta}}}(1) = \hat{\boldsymbol{w}}_2$,

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta} \in \mathbb{R}^m}{\arg\min} \left\{ \mathbb{E}_{t \sim U(0,1)} [L_{\mathcal{S}}(\phi_{\boldsymbol{\theta}}(t))] \right\}.$$

Due to limited computational resources, we restricted our analysis to the computation of polygonal chains for two pairs of models trained with differing optimisers.

4.3 Sharpness Analysis

To evaluate the sharpness, we adapted the code [19] used in [2] to work with our models and support running on both CPUs and GPUs. We made modifications for outputting the adversarial training loss in addition to the sharpness.

[5] A value of 5000 was selected as the largest batch size which fits into the memory of a single A100 80 GB GPU.

Adversarial Sharpness. To compute the adversarial sharpness, we take n batches $\{\mathcal{U}_i \subset \mathcal{S} : i = 1, 2, \ldots, n\}$, where each batch is of size m. We calculate the adversarial perturbation $\hat{\boldsymbol{\epsilon}}_i$ for each batch as shown in Section 3. Then we define adversarial sharpness (or m-sharpness) to be

$$\frac{1}{n}\sum_{i=1}^{n}\left\{L_{\mathcal{U}_i}(\boldsymbol{w}^* + \rho\hat{\boldsymbol{\epsilon}}_i) - L_{\mathcal{U}_i}(\boldsymbol{w}^*)\right\}.$$

We chose $n = 4$ as a trade-off between running time and reducing variance. For some parts of our analysis, we do not consider the sharpness but rather the adversarially perturbed loss, defined as

$$\frac{1}{n}\sum_{i=1}^{n} L_{\mathcal{U}_i}(\boldsymbol{w}^* + \rho\hat{\boldsymbol{\epsilon}}_i).$$

Average Sharpness. The average sharpness is calculated in the same way as above; however, $\hat{\boldsymbol{\epsilon}}_i$ is found using the average perturbation method from Section 3. Due to the inherent randomness of this measure, we chose $n = 15$ here.

4.4 Asymmetry Analysis

Central to our analysis is the ability to quantify and visualise the asymmetry of minima. Let $\mathrm{U}(0, 1)$ be a random uniform distribution. Consider a vector $\hat{\boldsymbol{u}}$ with elements drawn from $\mathrm{U}(0, 1)$, normalised to be of unit length. Then we can plot the loss landscape of a minimum $\boldsymbol{w}^*$ along a direction $\hat{\boldsymbol{u}}$ and in an interval $[a, b]$ using the coordinates

$$\{(\mu, L_{\mathcal{S}}(\boldsymbol{w}^* + \mu\hat{\boldsymbol{u}})) : \mu \in [a, b]\},$$

as shown in [22]. Figure 1 provides an example of such a plot under two regimes: a symmetrical minimum and an asymmetrical minimum. However, whilst useful for visualising the loss landscape, comparisons between minima are by inspection only. We seek a value to quantify the asymmetry of minima, which is also comparable between models. We propose the following notion of an *asymmetry factor*.

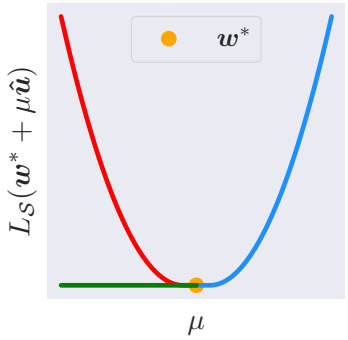


Figure 1: An example of a symmetrical and asymmetrical minimum. As a pair, the blue and red lines form a symmetrical minimum. The blue and green lines form an asymmetrical minimum. By our novel notion of an asymmetry factor, the former would have an asymmetry factor of 1 and the latter of 0.

Asymmetry Factor. Consider a model parameterised by a minimum $\boldsymbol{w}^*$. Let $l = L_{\mathcal{S}}(\boldsymbol{w}^*)$ be the loss at $\boldsymbol{w}^*$. Furthermore, let $\hat{\boldsymbol{u}}_i \sim \mathrm{U}(0, 1)$ be a unit vector in a random direction, where $i = 1, 2, \ldots, m$ and $m \in \mathbb{Z}^+$. We let $l_i^+ = L_{\mathcal{S}}(\boldsymbol{w}^* + \mu\hat{\boldsymbol{u}}_i) - l$ and $l_i^- = L_{\mathcal{S}}(\boldsymbol{w}^* - \mu\hat{\boldsymbol{u}}_i) - l$, where $\mu \in \mathbb{R}_{>0}$. Finally, we define the asymmetry factor to be

$$\Omega = \frac{1}{m}\sum_{i=1}^{m}\frac{\min(l_i^+, l_i^-)}{\max(l_i^+, l_i^-)},$$

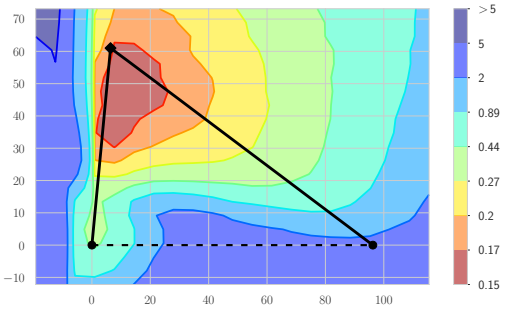
where the dependencies on $\boldsymbol{w}^*$ have been dropped for notational ease. Under the assumption that $\boldsymbol{w}^*$ is indeed a minimum, $\Omega \in [0, 1]$, where $\Omega = 0$ denotes a perfectly asymmetric minimum and $\Omega = 1$ denotes a perfectly symmetrical minimum. Informally, the asymmetry factor can be considered the average ratio between the increase in loss at a distance μ and $-\mu$ from the minimum in directions $\hat{\boldsymbol{u}}_i$.

5 Experiments

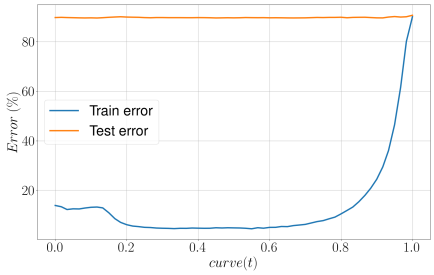
Based on hyperparameters found to work well in prior work [2, 16], we selected $\rho \in \{0.05, 0.1, 0.2\}$ for adversarial SAM and $\rho \in \{0.5, 1.0\}$ for average SAM. The values used for the latter are higher since the fixed norm random perturbations are significantly smaller than the adversarial perturbations in practice and smaller ρ values do not improve the training process [2, 16]. After our preliminary experiments, the hyperparameters — except the batch size and ρ — were kept constant for all experiments. In addition, we trained an SGD network for 400 epochs, which performs the

Table 1: This table includes all training combinations for the MobileNetV2 model on the CIFAR-10 dataset. For each hyperparameter combination, we report the final train loss, test set loss, and test set accuracy. The losses are average cross-entropy values, and the best value for each column is in bold. 5000-avg-SAM with 1.0 ρ obtained the best training loss; however, it generalizes worse than 128-SAM. 128-SAM with 0.1 ρ has the best test loss, whereas 0.2 ρ has the best accuracy.

	Hyperparmeters					
Optimiser	Batch Size	ρ	Test Loss	Train Loss	Test Accuracy / %	Asymmetry Factor
SGD	128	—	0.2197	0.1435	92.16	0.0758
Adversarial SAM	128	0.05	0.1879	0.1147	93.39	0.0975
	128	0.1	**0.1741**	0.1105	93.98	0.0681
	128	0.2	0.1829	0.1292	**94.19**	**0.0461**
	5000	0.05	0.2288	0.0821	92.08	0.1312
	5000	0.1	0.2289	0.1143	92.02	0.1093
Average SAM	128	0.5	0.2196	0.1414	92.14	0.2764
	128	1	0.2301	0.1453	92.38	0.1390
	5000	0.5	0.2250	0.0461	92.75	0.0799
	5000	1	0.2209	**0.0459**	93.01	0.0472



(a) The model on the left is trained with 128-SAM, $\rho = 0.2$, for 200 epochs. The model on the right is trained with 128-SGD for 400 epochs. The two can be connected via a polygonal chain.



(b) The train and test errors of the chain connecting the two models.

Figure 2: The l_2-regularised cross-entropy train loss surface of a polygonal chain with one bend connecting two MobileNetV2 models on CIFAR-10; surface adapted by Garipov et al. [17].

same number of gradient operations as 200-epoch SAM training. However, we observed that with MobileNetV2, 400-epoch SGD in our setting performs worse than 200-epoch SGD, so we used 200-epoch SGD for further comparisons. In Table 1, we can observe the different hyperparameter and optimiser combinations we used for training the models and their respective training losses, test losses, and test accuracy. This replicates the findings of [16] where 128-SAM performs better than SGD. Further information can be found in Appendix B.

5.1 Loss Basins

Figure 2 shows a polygonal chain with one bend that connects one 128-SAM model to one 128-SGD model. While the curve has low training loss for most of its trajectory, test loss stays high throughout, indicating that they belong to different basins of the test loss landscape. We have noticed a similar pattern for the pair connecting 128-SAM to 5000-SAM (see Appendix C). This suggests that 128-SAM, rather than being just an approximation of full-batch SAM, has a different bias that draws it towards flatter minima, as observed throughout this analysis.

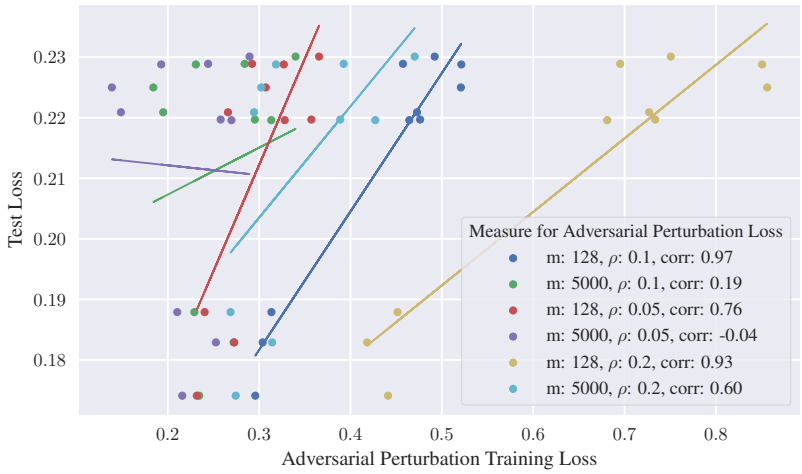


Figure 3: Relationship between hyperparameters of measure used to measure the loss of the perturbed point from the minima and test loss. Each colour group represent a set of different runs whose final minima is evaluated with the same perturbation measure. We find that the training loss of the adversarially perturbed point from 128-SAM is generally much better than 5000-SAM at predicting generalization ability.

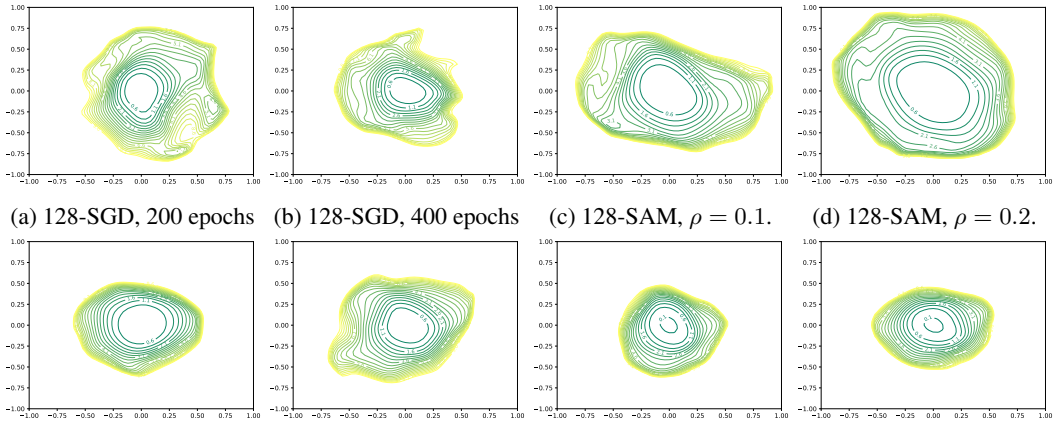
5.2 Sharpness

Foret et al. [16] showed that SAM with low m and low ρ result in lower test loss. However, it is unclear to what extent this is due to these optimisers finding minima with low sharpness, that these sharpness measures correlate well with test performance, or some implicit bias. We hypothesised that sharpness measures with specific values of m and ρ correlate significantly better with test loss than others. Instead of evaluating this with respect to sharpness, we use the related adversarially perturbed point loss (defined in Section 4.3) as this is closer to what SAM optimises for.

Figure 3 shows surprisingly that the training loss from perturbed points from 5000-sharpness is a worse proxy for test loss than 128-sharpness. As 5000-sharpness is closer to the true measure of sharpness, we would have thought it would correlate better with generalisation performance. A possible reason for this could be that 5000-sharpness is too good at finding extreme adversarially perturbed points. That is, the minima can be relatively flat but still have a high 5000-SAM sharpness if there is one point with a very high training loss at the radius. However, lower m-SAM may be a better proxy for the kind of sharpness we care about, as it is more aligned with our intuition for what sharp minima look like.

Andriushchenko and Flammarion [2] demonstrated that 128-SAM results in lower 128-sharpness of the final minima than 1024-SAM. However, the 1024-SAM minimum was not evaluated in its ability to find a minima with low 1024-sharpness. One might think 128-sharpness is just a more noisy version of 1024-sharpness, but we would argue that they measure two distinct types of sharpness (see Appendix D). We hypothesise that 5000-SAM has a higher test loss because it is worse at finding a minima with lower 5000-sharpness than 128-SAM. In Table 2, we found that 128-SAM finds minima with approximately the same adversarially perturbed point loss as 5000-SAM evaluated with batch size 5000. This suggests our hypothesis was wrong and that 5000-SAM does not have a higher test loss because it is worse at finding flat minima.

The contour maps in Figure 4 seem to suggest the 5000-minima are sharper, but this may be because these contour maps represent a different kind of sharpness. The 5000-sharpness may be able to find extreme adversarial perturbation losses that are not visible on the contour maps. Regardless, our analysis suggests that 5000-SAM fails not so much because it is particularly bad at finding 5000-sharp minima but because the 5000-SAM objective correlates poorly with test loss.



(a) 128-SGD, 200 epochs (b) 128-SGD, 400 epochs (c) 128-SAM, $\rho = 0.1$. (d) 128-SAM, $\rho = 0.2$.

(e) 5000-SAM,$\rho = 0.1$. (f) 128-avgSAM, $\rho = 0.5$. (g) 5000-avgSAM, $\rho = 0.1$.(h) 5000-avgSAM, $\rho = 0.5$.

Figure 4: Loss contours of eight models trained with different optimisers. Loss contours are drawn according to [36].

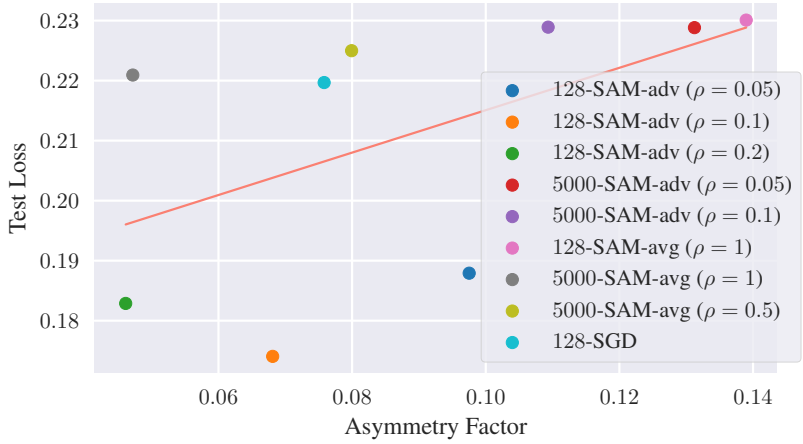


Figure 5: The asymmetry factor ($\mu = 5$, $m = 30$) is plotted against the test loss for each model, with a correlation coefficient of 0.527. Recalling that a lower asymmetry factor implies higher asymmetry, this correlation coefficient suggests a moderately strong positive correlation between high asymmetry and low test loss. More specifically, we interpret this to mean that high asymmetry implies good generalisation performance. We have excluded the 128-SAM-avg ($\rho = 0.5$) model from this figure, as it was an outlier result which skewed the proportions. With this included, we found the correlation coefficient to be 0.359, suggesting a less strong positive correlation.

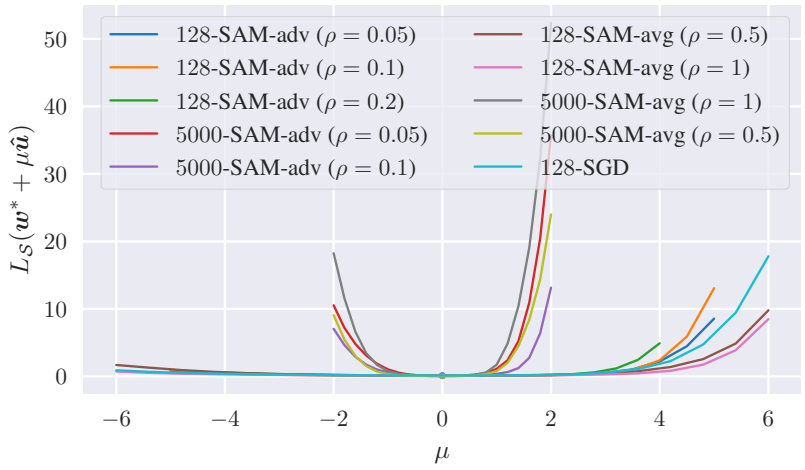


Figure 6: The perturbation distance μ is plotted against the training loss of a minima $\boldsymbol{w}^*$ perturbed in a random direction $\hat{\boldsymbol{u}}$ by distance of μ. $\hat{\boldsymbol{u}}$ is randomly selected for each model. The figure shows both the sharpness and asymmetry of the various minima. Notably, we see two distinct groups. The set of sharper minima corresponds to the 5000-SAM models (both adversarial and average), with the remaining models being significantly flatter.

5.3 Asymmetry

We hypothesised that some sharpness-aware minimisers do not improve generalisation performance because they also optimise for symmetric minima. More specifically, we thought this could be the case for the average SAM. We calculated the asymmetry factor for each model and plotted this against the test loss, as seen in Figure 5. We observe that the models using average SAM have almost constant test error but a range of asymmetry factors, suggesting that it is not necessarily their lack of asymmetry resulting in poor generalisation performance. However, there is a moderately strong positive correlation between the test error and asymmetry factor, suggesting that, generally, asymmetry is still desirable. Furthermore, Figure 6 suggests that the batch size, not the optimisation

Table 2: Relationship between the optimizer used and the loss at the adversarially perturbed point from the final minima. The columns are different metrics used to find the adversarial point of the final minima. The best loss for each column is in bold. We would expect each sharpness metric to be minimized by the minima of its respective SAM optimizer, along the diagonal. Interestingly we find that 128-SAM with $\rho = 0.1$ is better than 128-SAM with $\rho = 0.05$ at finding a minima that minimises adversarial loss with $\rho = 128$. Average 5000-SAM performing better than adversarial 5000-SAM may be due to a better choice of ρ for those runs.

			Batch Size = 128			Batch Size = 5000		
Optimizer	Batch Size	ρ	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.2$
SGD	128	—	0.357	0.476	0.734	0.295	0.258	0.389
Adversarial SAM	128	0.05	0.240	0.313	0.452	0.229	0.210	**0.269**
	128	0.1	**0.232**	**0.296**	0.441	0.234	0.216	0.274
	128	0.2	0.273	0.304	**0.418**	0.272	0.253	0.314
	5000	0.05	0.327	0.521	0.850	0.231	0.193	0.318
	5000	0.1	0.292	0.457	0.695	0.284	0.244	0.393
Average SAM	128	0.5	0.328	0.464	0.681	0.313	0.270	0.427
	128	1	0.366	0.492	0.751	0.340	0.289	0.470
	5000	0.5	0.307	0.521	0.856	**0.184**	**0.139**	0.302
	5000	1	0.266	0.473	0.727	0.195	0.149	0.294

method, determines the minima's sharpness. Future work may find it beneficial to define a version of the asymmetry factor which is invariant to model parameter scaling.

6 Limitations, Future Work, and Conclusion

Perhaps the biggest caveat of our work is caused by limited access to computing resources. We have only one sample from each training run; thus, we are unsure what the variances of the metrics we have found are. Furthermore, we have only tested our hypothesis using a small set of hyperparameters and using only one model, MobiletNetV2, which is magnitudes smaller than contemporary state-of-the-art vision models. We believe that future work should focus on evaluating our observations on larger models, with differing optimisers, various hyperparameters, and diverse datasets.

Additionally, whilst we discovered a correlation between sharpness and asymmetry within our set of trained models, we cannot claim that this relationship holds generally. It may be the case that these measures are confounded by other metrics — which better correlate with generalisation —- and SAM-based optimisers happen to minimise these implicitly. Further work should focus on confirming this observation for other vision models and optimisers.

Conclusion. In this work, we hypothesised that it is not only sharpness which plays a role in good generalisation performance but also asymmetry. We trained various models using carefully selected hyperparameters to investigate this and analysed the resulting minima's asymmetry, sharpness, and loss basins. Our results suggested that small-batch SAM finds flat asymmetric minima, which we have shown correlates with test loss. It indicates that there is an underlying bias that helps small-batch SAM outperform other optimisers. Furthermore, we found that part of the reason 5000-SAM generalises poorly is that 5000-sharpness correlates poorly with test loss.

References

[1] Shun-ichi Amari. "Natural Gradient Works Efficiently in Learning". In: *Neural Computation* 10.2 (1998), pp. 251–276.

[2] Maksym Andriushchenko and Nicolas Flammarion. "Towards Understanding Sharpness-Aware Minimization". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 639–668.

[3] Carlo Baldassi, Fabrizio Pittorino, and Riccardo Zecchina. "Shaping the learning landscape in neural networks around wide flat minima". In: *Proceedings of the National Academy of Sciences* 117.1 (2020), pp. 161–170.

[4] Kayhan Behdin et al. "Improved Deep Neural Network Generalization Using m-Sharpness-Aware Minimization". In: *OPT 2022: Optimization for Machine Learning (NeurIPS 2022 Workshop)*. 2022.

[5] Léon Bottou. "Online Learning and Stochastic Approximations". In: *On-line Learning in Neural Networks*. 1998, pp. 9–42.

[6] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018.

[7] Wray Buntine. "Bayesian back-propagation". In: *Complex systems* 5 (1991), pp. 603–643.

[8] Niladri S. Chatterji, Behnam Neyshabur, and Hanie Sedghi. *The intriguing role of module criticality in the generalization of deep networks*. 2019.

[9] Pratik Chaudhari et al. "Entropy-SGD: Biasing Gradient Descent Into Wide Valleys". In: *International Conference on Learning Representations*. 2017.

[10] Ekin D. Cubuk et al. "AutoAugment: Learning Augmentation Strategies From Data". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[11] Terrance DeVries and Graham W. Taylor. *Improved Regularization of Convolutional Neural Networks with Cutout*. 2017.

[12] Laurent Dinh et al. "Sharp Minima Can Generalize for Deep Nets". In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML'17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 1019–1028.

[13] Jiawei Du et al. "Efficient Sharpness-aware Minimization for Improved Training of Neural Networks". In: *International Conference on Learning Representations*. 2022.

[14] Jiawei Du et al. *Sharpness-Aware Training for Free*. 2022.

[15] Yu Feng and Yuhai Tu. "The inverse variance–flatness relation in stochastic gradient descent is critical for finding flat minima". In: *Proceedings of the National Academy of Sciences* 118.9 (2021), e2015617118.

[16] Pierre Foret et al. "Sharpness-Aware Minimization for Efficiently Improving Generalization". In: *International Conference on Learning Representations*. 2021.

[17] Timur Garipov et al. "Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[18] GitHub, ed. *davda54/sam*. URL: `https://github.com/davda54/sam` (visited on 01/15/2023).

[19] GitHub, ed. *tml-epfl/understanding-sam*. URL: `https://github.com/tml-epfl/understanding-sam` (visited on 01/15/2023).

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[21] Kosuke Haruki et al. *Gradient Noise Convolution (GNC): Smoothing Loss Function for Distributed Large-Batch SGD*. 2019.

[22] Haowei He, Gao Huang, and Yang Yuan. "Asymmetric Valleys: Beyond Sharp and Flat Local Minima". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[23] Sepp Hochreiter and Jürgen Schmidhuber. "Flat Minima". In: *Neural Computation* 9 (1997), pp. 1–42.

[24] Sepp Hochreiter and Jürgen Schmidhuber. *Flat Minimum Search Finds Simple Nets*. Tech. rep. Technical University of Munich, 1994.

[25] Sepp Hochreiter and Jürgen Schmidhuber. "Simplifying Neural Nets By Discovering Flat Minima". In: *Advances in Neural Information Processing Systems*. Ed. by G. Tesauro, D. Touretzky, and T. Leen. Vol. 7. MIT Press, 1994.

[26] Pavel Izmailov et al. *Averaging Weights Leads to Wider Optima and Better Generalization*. 2018.

[27] Stanisław Jastrzębski et al. "Finding Flatter Minima with SGD". In: *International Conference on Learning Representations*. 2018.

[28] Yiding Jiang et al. "Fantastic Generalization Measures and Where to Find Them". In: *International Conference on Learning Representations*. 2020.

[29] Kam-Chuen Jim, C.L. Giles, and B.G. Horne. "An analysis of noise in recurrent neural networks: convergence and generalization". In: *IEEE Transactions on Neural Networks* 7.6 (1996), pp. 1424–1438.

[30] Jean Kaddour et al. "When Do Flat Minima Optimizers Work?" In: *Advances in Neural Information Processing Systems*. Ed. by Alice H. Oh et al. 2022.

[31] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. "The Normalization Method for Alleviating Pathological Sharpness in Wide Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[32] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *International Conference on Learning Representations*. 2017.

[33] Minyoung Kim et al. "Fisher SAM: Information Geometry and Sharpness Aware Minimisation". In: *Proceedings of the 39th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri et al. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 11148–11161.

[34] Alex Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. Tech. rep. 2009.

[35] Jungmin Kwon et al. "ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 5905–5914.

[36] Hao Li et al. "Visualizing the Loss Landscape of Neural Nets". In: *Neural Information Processing Systems*. 2018.

[37] Tao Lin et al. "Don't Use Large Mini-batches, Use Local SGD". In: *International Conference on Learning Representations*. 2020.

[38] Tao Lin et al. "Extrapolation for Large-batch Training in Deep Learning". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6094–6104.

[39] Yong Liu et al. "Towards Efficient and Scalable Sharpness-Aware Minimization". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 12360–12370.

[40] Ilya Loshchilov and Frank Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2016.

[41] David A. McAllester. "PAC-Bayesian Model Averaging". In: *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*. COLT '99. Santa Cruz, California, USA: Association for Computing Machinery, 1999, pp. 164–170.

[42] Alan Murray and Peter Edwards. "Synaptic Weight Noise During MLP Learning Enhances Fault-Tolerance, Generalization and Learning Trajectory". In: *Advances in Neural Information Processing Systems*. Ed. by S. Hanson, J. Cowan, and C. Giles. Vol. 5. Morgan-Kaufmann, 1992.

[43] Behnam Neyshabur et al. "Exploring Generalization in Deep Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017.

[44] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019.

[45] Henning Petzka et al. "Relative Flatness and Generalization". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato et al. Vol. 34. Curran Associates, Inc., 2021, pp. 18420–18432.

[46] J. Rissanen. “Modeling by shortest data description”. In: *Automatica* 14.5 (1978), pp. 465–471.

[47] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252.

[48] Mark Sandler et al. “Mobilenetv2: Inverted residuals and linear bottlenecks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 4510–4520.

[49] Ilya Sutskever et al. “On the importance of initialization and momentum in deep learning”. In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147.

[50] C. S. Wallace and D. M. Boulton. “An Information Measure for Classification”. In: *The Computer Journal* 11.2 (Aug. 1968), pp. 185–194.

[51] Wei Wen et al. *SmoothOut: Smoothing Out Sharp Minima to Improve Generalization in Deep Learning*. 2018.

[52] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. “Adversarial Weight Perturbation Helps Robust Generalization”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 2958–2969.

[53] Zeke Xie, Issei Sato, and Masashi Sugiyama. “A Diffusion Theory For Deep Learning Dynamics: Stochastic Gradient Descent Exponentially Favors Flat Minima”. In: *International Conference on Learning Representations*. 2021.

[54] Zhewei Yao et al. “Hessian-based Analysis of Large Batch Training and Robustness to Adversaries”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[55] Sergey Zagoruyko and Nikos Komodakis. “Wide Residual Networks”. In: *Proceedings of the British Machine Vision Conference (BMVC)*. Ed. by Edwin R. Hancock Richard C. Wilson and William A. P. Smith. BMVA Press, Sept. 2016, pp. 87.1–87.12.

[56] Shuofeng Zhang et al. *Why Flatness Does and Does not Correlate With Generalisation For Deep Neural Networks*. 2022.

[57] Yang Zhao, Hao Zhang, and Xiuyuan Hu. *SS-SAM : Stochastic Scheduled Sharpness-Aware Minimization for Efficiently Training Deep Neural Networks*. 2022.

[58] Yaowei Zheng, Richong Zhang, and Yongyi Mao. “Regularizing Neural Networks via Adversarial Model Perturbation”. In: *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 8152–8161.

[59] Juntang Zhuang et al. “Surrogate Gap Minimization Improves Sharpness-Aware Training”. In: *International Conference on Learning Representations*. 2022.

A Comparison of SAM Variants

Table 3 and Table 4 provide a comparison of numerous SAM variants.

B Model Training

We struggled for a while to reproduce the SAM results with our smaller model. We tried using the 1.5M parameter MobileNetV3, but SAM would always generalise worse. We thought this may be because the CIFAR-10 dataset was too large for MobileNetV3. We therefore also tried to make our model overparametrised by reducing the size of CIFAR-10. We tried both excluding certain classes and reducing the number of samples per class, but SGD still outperformed SAM.

Figure 7 shows the test set loss and accuracy per epoch for each of the models we trained.

C Mode Connectivity of 128-SAM and 5000-SAM

An additional example — supporting the same analysis as in the main body of this paper — is available in Figure 8.

Table 3: Comparison of SAM variants for common models on the CIFAR datasets. Results for models that non-empty values for throughput (*images/s*) are taken from [14]. The rest are presented as reported in their original papers.

	CIFAR-10		CIFAR-100	
ResNet-18	Accuracy	Images / s	Accuracy	Images / s
SGD	95.61	3289	78.32	3314
SAM	96.52	1717	80.17	1730
SS-SAM	96.40	-	80.10	-
ESAM	96.56	2409	80.41	2423
SAF	96.37	3213	80.06	3248
MESA	96.24	2780	79.79	2793
LookSAM-5	-	-	80.7	-
WideResNet-28-10				
SGD	96.34	801	81.56	792
SAM	97.27	396	83.42	391
SS-SAM	97.09	-	82.89	-
ESAM	97.29	550	84.51	545
SAF	97.08	727	83.81	729
MESA	97.16	617	83.59	625
LookSAM-5	-	-	84.4	-
ASAM	97.63	-	85.20	-
FSAM	97.89	-	85.60	-
PyramidNet-110				
SGD	96.62	580	81.89	555
SAM	97.30	289	84.46	276
SS-SAM	97.22	-	84.90	-
ESAM	97.81	401	85.56	381
SAF	97.34	391	84.71	397
MESA	97.46	332	84.73	339

Table 4: Comparison of SAM variants for common models on the ImageNet dataset. Results for models that non-empty values for throughput (*images/s*) are taken from [14]. The rest are presented as reported in their original papers.

	ResNet-50		ResNet-101	
ImageNet	Accuracy	Images / s	Accuracy	Images / s
Vanilla SGD	76.0	1627	77.8	1042
SAM	76.9	802	78.6	518
SS-SAM	77.38	-	-	-
ESAM	77.2	1037	79.1	650
GSAM	77.1	783	78.9	503
SAF	77.8	1612	79.3	1031
MESA	77.5	1386	79.1	888
ASAM	76.6	-	-	-
	ResNet-152		ViT-S/32	
ImageNet	Accuracy	Images / s	Accuracy	Images/s
Vanilla	78.5	703	68.1	5154
SAM	79.3	351	68.9	2566
GSAM	80.0	341	73.8	2469
SAF	79.9	694	69.5	5108
MESA	80.0	601	69.6	4391
LookSAM	-	-	68.8	4273

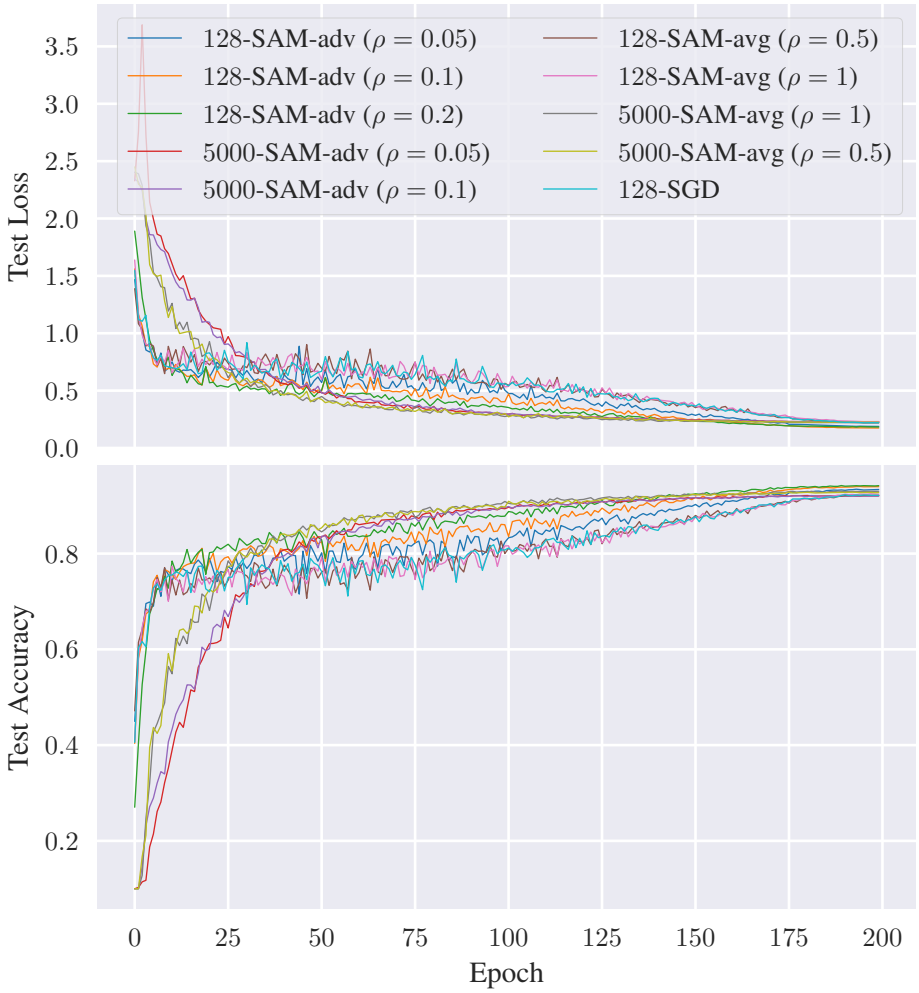


Figure 7: The loss and accuracy per epoch of all of the MobileNetV2 models trained on the CIFAR-10 [34] dataset.

Table 5: Relationship between the optimizer used and the sharpness of the minima it found. The columns are different metrics used to evaluate the sharpness of the final minima. These are evaluated by doing the first step of the SAM gradient and comparing the losses. The best loss for each column is in bold. We would expect each sharpness metric to be minimized by the minima of its respective SAM optimizer, along the diagonal. Interestingly we rather find 128-SAM $\rho = \{0.1, 0.2\}$ to perform better even on the $\rho = 0.05$ metric, and always outperforming 5000-sam.

			Batch Size = 128			Batch Size = 5000		
Optimizer	Batch Size	Rho	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.2$	$\rho = 0.05$	$\rho = 0.1$	$\rho = 0.2$
SAM	128	0.05	0.073	0.139	0.28	0.019	0.037	0.077
		0.1	0.058	0.120	0.26	**0.017**	**0.036**	**0.075**
		0.2	**0.048**	**0.093**	**0.20**	0.018	0.037	0.079
	5000	0.05	0.177	0.372	0.71	0.035	0.073	0.160
		0.1	0.127	0.285	0.52	0.035	0.075	0.183
avg-SAM	128	0.5	0.125	0.258	0.48	0.036	0.079	0.193
		1	0.131	0.264	0.52	0.041	0.091	0.221
	5000	0.5	0.211	0.434	0.78	0.038	0.084	0.202
		1	0.178	0.380	0.63	0.039	0.086	0.185
SGD	128	-	0.128	0.256	0.50	0.031	0.069	0.162

Table 6: Relationship between the optimizer used and the average sharpness of the minima it found. The columns are different metrics used to evaluate the sharpness of the final minima. These are evaluated by random perturbations away from the minima. The loss at these points are then averaged. The best loss for each column is in bold. We would expect each sharpness metric to be minimized by the minima of its respective avg-SAM optimizer, along the diagonal. However SAM or SGD always outperforms avg-SAM. This suggests the inferior performance of avg-SAM may not be down to the objective it is optimizing for, but rather that it fails to minimize that objective.

			Batch Size = 128		Batch Size = 5000	
Optimizer	Batch Size	Rho	$\rho = 0.5$	$\rho = 1$	$\rho = 0.5$	$\rho = 1$
SAM	128	0.05	11.43	45.84	**9.24**	41.33
		0.1	11.29	42.40	10.50	44.30
		0.2	**11.23**	44.06	11.15	44.88
	5000	0.05	12.48	42.82	13.21	37.42
		0.1	12.50	52.39	11.39	**37.13**
avg-SAM	128	0.5	13.68	48.05	12.79	45.63
		1	11.55	47.39	10.28	44.86
	5000	0.5	13.30	44.38	13.20	50.57
		1	12.39	46.32	11.66	42.03
SGD	128	-	11.47	**40.83**	11.28	50.24

Table 7: Relationship between the optimizer used and the loss at the average perturbed point from the minima it found. The columns are different metrics used to find the adversarial point of the final minima. The best loss for each column is in bold. Note interestingly that SGD does best for Batch Size 128 and $\rho = 1$

			Batch Size = 128		Batch Size = 5000	
Optimizer	Batch Size	Rho	$\rho = 0.5$	$\rho = 1$	$\rho = 0.5$	$\rho = 1$
SAM	128	0.05	11.57	45.98	**9.42**	41.33
		0.1	11.43	42.55	10.69	44.30
		0.2	**11.40**	44.23	11.38	44.88
	5000	0.05	12.62	42.96	13.36	37.42
		0.1	12.63	52.52	11.59	**37.13**
avg-SAM	128	0.5	13.86	48.23	13.02	45.63
		1	11.74	47.58	10.52	44.86
	5000	0.5	13.38	44.47	13.30	50.57
		1	12.48	46.41	11.76	42.03
SGD	128	-	11.60	**40.96**	11.49	50.24

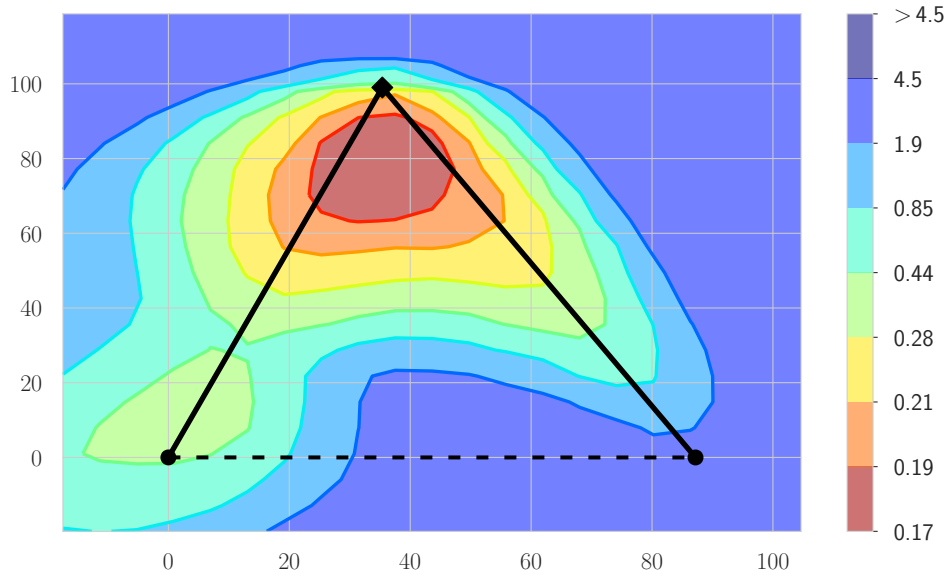
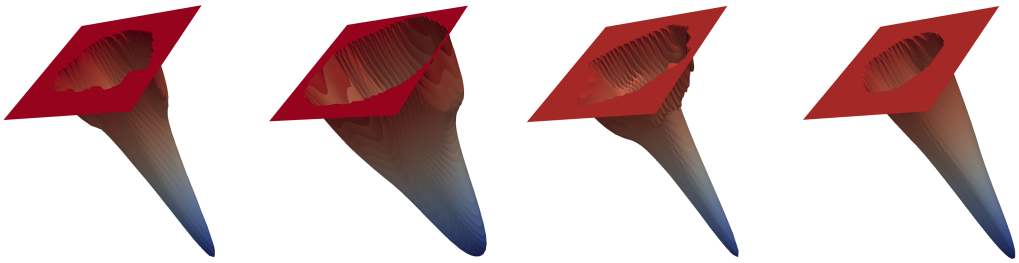


Figure 8: Model to the left is trained with 128-SAM, $\rho = 0.2$; model to the right is trained with 5000-SAM, $\rho = 0.1$. The surface is the l_2 regularised loss of a polygonal chain with one bend connecting the two models. We notice they seem to be part of different basins.



(a) 128-SGD, 400 epochs. (b) 128-SAM, $\rho = 0.2$. (c) 128-avgSAM, $\rho = 0.1$. (d) 5000-SAM, $\rho = 1$.

Figure 9: 3D loss surfaces of models trained with different optimisers. Loss surfaces are drawn according to [36].

D Comparison of Sharpness

See Figure 9 and Figure 10 along with Tables 5–7 for additional examples of the content discussed in the main body. These figures and tables demonstrate that the minimum from 128-SAM ($\rho = 0.05$) is sharpest according to 128-sharpness, but the minimum from 128-SAM ($\rho = 0.1$) is sharpest according to 5000-sharpness. Moreover, avg-SAM fails at finding avg-sharp minima and avg-sharp minima correlate poorly with generalisation performance.

E Loss Landscape of Models

Figure 11 shows the loss landscape of various minima.

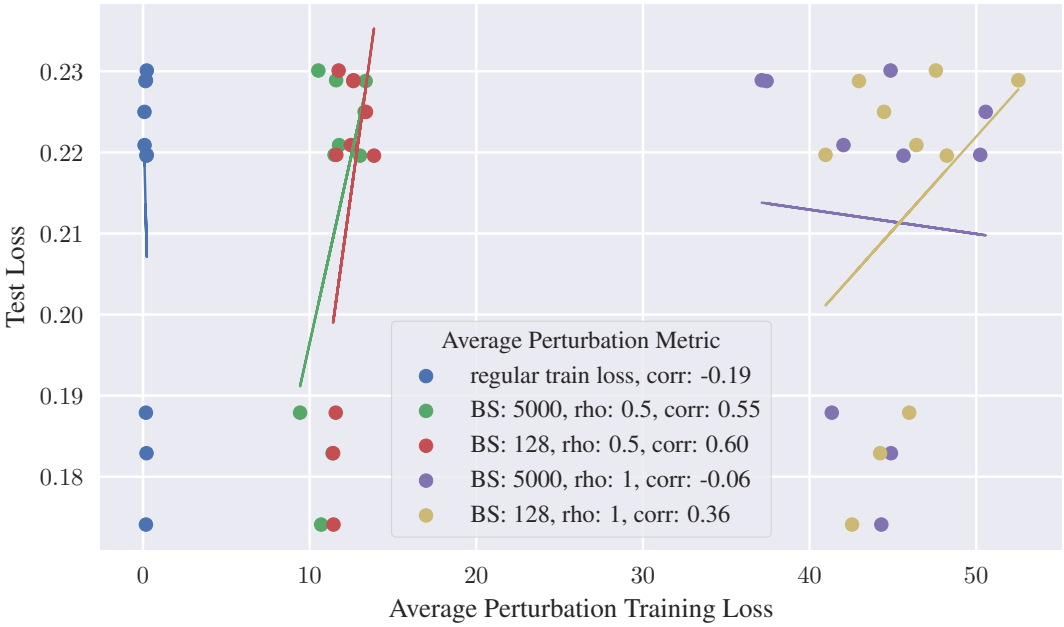
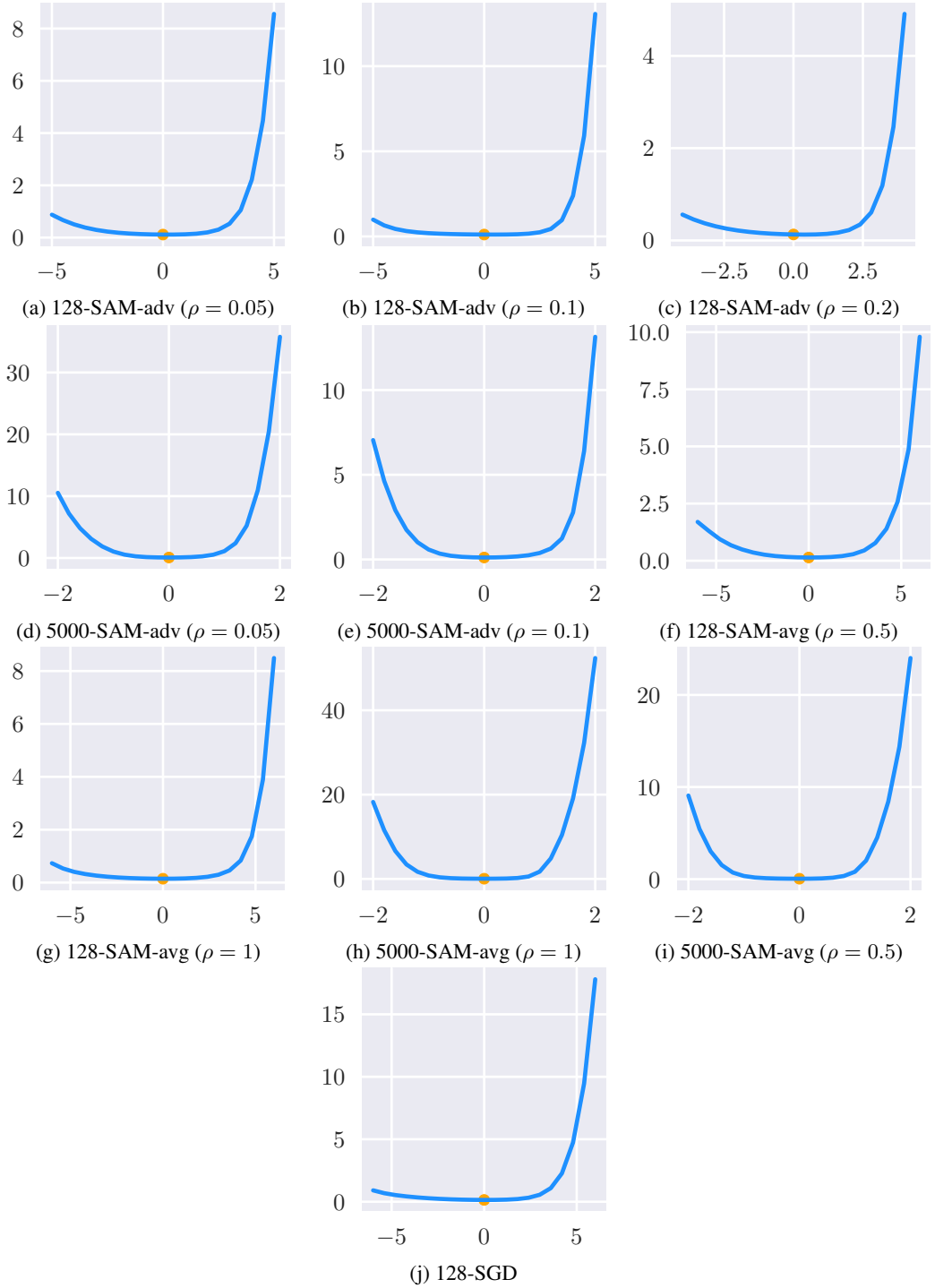


Figure 10: Relationship between average perturbation metrics and test loss. Each colour group represent a set of different runs whose final minima are evaluated with the same perturbation metric. Most of these perturbation metrics correlate poorly with test loss.



(a) 128-SAM-adv ($\rho = 0.05$) (b) 128-SAM-adv ($\rho = 0.1$) (c) 128-SAM-adv ($\rho = 0.2$)

(d) 5000-SAM-adv ($\rho = 0.05$) (e) 5000-SAM-adv ($\rho = 0.1$) (f) 128-SAM-avg ($\rho = 0.5$)

(g) 128-SAM-avg ($\rho = 1$) (h) 5000-SAM-avg ($\rho = 1$) (i) 5000-SAM-avg ($\rho = 0.5$)

(j) 128-SGD

Figure 11: The loss landscape of various minima (marked in orange), with μ on the x-axis and $L_{\mathcal{S}}(\boldsymbol{w}^* + \mu\hat{\boldsymbol{u}})$ on the y-axis. Note that the scales of the graphs are different, meaning that they are not necessarily visually comparable.