Sharpness-Aware Minimization for Efficiently Improving Generalization

Francesco Fainello, Francesco Pettenon, Francesca Venturi
EPFL Course - Optimization for Machine Learning - CS-439

***Abstract*—The aim of this project is to explore the benefits to generalization of deep neural networks by employing Sharpness Aware Minimization, which optimizes the loss by simultaneously accounting for its geometric landscape. First, we present the optimization algorithm, then we discuss how the relationship between the flatness of minima and generalization can be analyzed, and finally, we evaluate results on three different deep neural architectures. The main focus will be on the reduction of the generalization gap rather than on absolute accuracy.**

I. Introduction

One of the major challenges in developing modern machine learning models, often over-parametrized, lies in studying their generalization capabilities. This refers to their ability to perform well on new data, thereby reducing the test loss of the model. Focusing solely on the training loss can lead to sub-optimal models with a high risk of overfitting. Inspired by the combined study of the loss landscape geometry and model generalization, we have exploited a new effective procedure to simultaneously minimize the value of the loss and its sharpness. This procedure, called Sharpness-Aware Minimization (SAM) [1], seeks parameters that lie in the vicinity of uniformly low losses, leading to a min-max optimization problem that can be efficiently solved using the gradient algorithm.

II. Flatness of minima and generalization

The connection between the geometry of the loss landscape — in particular, the flatness of minima — and generalization has been studied extensively [2], [3]. A flat minimizer x^* is one for which the function varies slowly in a relatively large neighborhood of it, while a sharp minimizer is such that the function increases rapidly in a small neighborhood of x^*. The large sensitivity of the training function at a sharp minimizer negatively impacts the ability of the trained model to generalize on new data. Hence [1] attempts to improve the generalization capability of a model by simultaneously minimizing loss value and loss sharpness. In order to present the obtained results, we produce plots of the loss landscape in neighbourhoods of the minima. [4] proposes a simple visualization method based on "filter normalization", stating that the sharpness of minimizers correlates well with generalization error when this normalization is used. This allows precise comparison between differently obtained minimizers through plots.

It can be further observed that sharp minimizers are characterized by a significant number of large positive eigenvalues of $\nabla^2 L(\omega)$. In contrast, flat minimizers are characterized by having numerous small eigenvalues of $\nabla^2 L(\omega)$. Since we are viewing the loss surfaces under a dramatic dimensionality reduction, [4] quantifies the level of convexity in loss functions by computing the principle curvatures, which are simply eigenvalues of the Hessian. This allows to prove that convex-looking regions in surface plots do indeed correspond to regions with insignificant negative eigenvalues (i.e., there are not major non-convex features that the plot missed), while chaotic regions contain large negative curvatures. In the following, we quantitatively evaluate non-convexity of the loss function by computing the eigenvalue spectral density (ESD) of the Hessian around local minima, as allowed by PyHessian [5], mostly focusing on the top eigenvalue and on the ratio $\lambda_{max}/\lambda_{min}$.

III. SAM Optimizer [1]

The insight behind Sharpness-Aware Minimization is exploiting the relationship between the geometry of the loss and the generalization capability of a model.

In particular, for modern models, the loss function (denoted as $L_{\mathcal{S}}(\omega)$) is non-convex, meaning it can have multiple local minima with similar loss values but different generalization capabilities.

Therefore, instead of solely seeking parameters with low training loss, the approach focuses on finding parameters where the neighbourhood has uniformly low loss (i.e., parameters with both low loss and low curvature). This intuition is based on the following theorem, which provides a bound on the generalization ability in terms of neighbourhood-wise training loss:

Theorem III.1. *Let $L_{\mathcal{D}}(\omega)$ be the population loss and $L_{\mathcal{S}}(\omega)$ be its estimate (i.e. the training loss). For any $\rho \geq 0$, with high probability over the training set $\mathcal{S}$ generated from the distribution $\mathcal{D}$,*

$$L_{\mathcal{D}}(\omega) \leq \max_{\|\epsilon\|_2} L_{\mathcal{S}}(\omega + \epsilon) + h(\|\omega\|_2^2/\rho^2) \tag{1}$$

where $h : \mathbb{R}_+ \to \mathbb{R}_+$ is a strictly increasing function (under some technical conditions on $L_{\mathcal{D}}(\omega)$).

So the selection of the parameters happens according to the following min-max optimization problem:

$$\min_{\omega} L_{\mathcal{S}}^{SAM}(\omega) + \lambda\|\omega\|_2^2 \tag{2}$$

$$\text{where } L_{\mathcal{S}}^{SAM}(\omega) := \max_{\|\epsilon\|_2} L_{\mathcal{S}}(\omega + \epsilon)$$

From an implementation point of view, we exploit the following Algorithm 1, visualized in Fig. 1:

Algorithm 1 SAM Algorithm

Input: Training set $\mathcal{S}$, loss function l, batch-size b, stepsize $\eta \geq 0$, neighborhood size $\rho \geq 0$
Output: Model trained with SAM

1: Initialize weights $\omega_0, t = 0$;
2: **while** *not converged* **do**
3: Sample batch $\mathcal{B} = \{(\mathbf{x}_1, \mathbf{y}_1), .., (\mathbf{x}_b, \mathbf{y}_b)\}$;
4: Compute gradient $\nabla_\omega L_\mathcal{B}(\omega)$ of the batch's training loss;
5: Compute $\hat{\epsilon}(\omega)$ (optimizing a linearization)
6: Compute the gradient approximation for the SAM objective: $\mathbf{g} = \nabla_\omega L_\mathcal{B}(\omega)|_{\omega+\hat{\epsilon}(\omega)}$
7: Update weights: $\omega_{t+1} = \omega_t - \eta \mathbf{g}$
8: **end while**
9: **return** ω_t

Further details about the algorithm and the theoretical background underlying SAM optimization can be found in [1]. What is evident from the pseudocode is the presence of two backpropagation steps within the same training iteration, which is considered when comparing it with other optimizers discussed in the following sections.

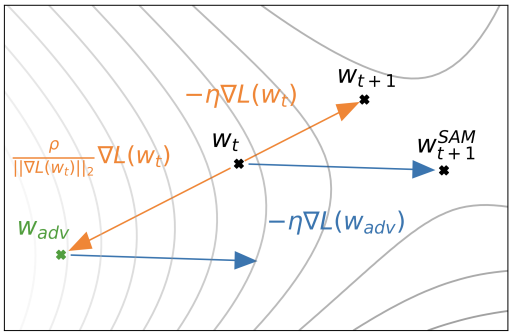

Fig. 1. Visual Representation of SAM optimizer (source [1])

It can be interesting to observe how perturbing the loss in equation (2) is closely related to the concept of Adversarial Training (AT). As highlighted by Wei et al. (2023) [6], there is an important philosophical distinction that sets these two approaches apart: in the case of AT, the data is perturbed explicitly, whereas in SAM, it is the weights that are perturbed.

IV. Models and test cases

The aim of the project is to apply the SAM optimizer to different network models, studying its adaptability to different architectures and input data types. For this reason, we decide to first reproduce the results of the reference paper, and then move on to completely different models, while still focusing on the classification tasks. In all cases, we simply replace the optimization procedure used to train existing models with SAM, and analyse the resulting effect on model generalization and flatness of minima.

A. WideResNet for Image Classification

[1] presents results by fine-tuning pre-trained classification models, focusing in particular on the classification of the CIFAR-10 dataset. The employed model is WideResNet [7], a deep residual network with decreased depth and increased width. The model has around 17 million parameters and the exploited dataset is composed of 60000 images.

B. Attention Model for Time Series Classification

In order to address classification problems, a model with complex structures beyond feedforward networks is chosen, and transformers are selected for their ability to capture dependencies in time series data. The transformer-based architecture consists of an encoder stack followed by a classification head [8], processing sequential input data with self-attention and feed-forward operations to make predictions. The model has around 19 million trainable parameters. The dataset used is the MIT-BIH Arrhythmia Database, which contains 109,446 samples categorized into five classes representing different types of cardiac arrhythmias. Due to the dataset's imbalance, the Normal Beat class (labelled with 0) is down-sampled to achieve fair training, resulting in a total of 19,659 samples.

C. Graph Convolutional Network for Graph classification [9]

The third model aims to classify whether a molecule is considered toxic or not. The novelty of this model lies in representing each molecule as a graph. Taking inspiration from well-known convolutional neural networks for images, we construct a Graph Convolutional Network (GCN) [10] to classify each molecule represented as a graph, where the vertices (V) represent atoms and the edges (E) represent bonds. The dataset used for this task is the TUDataset Mutagenity [11], consisting of 4337 different graphs, with an average number of nodes of 30.32 and an average number of edges of 30.77. To handle this dataset, the PyTorch Geometric library is utilized. In summary, the final architecture is inspired by Graph Classification example and the labels are divided into two classes. The total number of trainable parameters of the model is around 125 thousands.

V. Results

The results presented in this section are the outcome of tuning the hyperparameter ρ. Specifically, for each model, we selected four values of ρ. For each of them, we compared the Train/Test Loss and Train/Test Accuracy to determine the optimal radius for SAM. Table I summarizes the corresponding results for the optimal ρ, with the specific numerical values provided in the subsequent analysis.

A. WideResNet

WideResNet was trained with SGD and SAM optimizer, employing $\rho = 2$. SAM achieves comparable accuracies to SGD, and the generalization gap between losses appears reduced when employing the former. Consistently, the analysis of the hessian's curvatures shows that SAM attains a flatter minimum (lower ratio $\lambda_{max}/\lambda_{min}$).

Model	Train loss	Train acc (%)	Test loss	Test acc (%)	$\lvert\lambda_M/\lambda_m\rvert$
W.ResNet SGD	0.0367	98.57	0.1095	95.43	21.7
W.ResNet SAM	0.0459	98.64	0.0961	95.97	4.8
Attention SGD	0.0770	96.31	0.1204	94.64	16.8
Attention SAM	0.0836	95.94	0.1223	94.26	12.2
GCN ADAM	0.1754	86.46	0.2406	80.41	289
GCN SAM	0.1719	87.05	0.2457	81.11	65

TABLE I
RESULTS

B. Attention model

An optimal value of $\rho = 1$ was found for the model employing attention. Results for the two optimizers appear very similar, thus SAM doesn't seem to achieve significant improvements on this specific test case.

C. Graph Neural Network

Finally, the Graph Neural Network was trained using ADAM (as this is the standard choice for this kind of models) and SAM with optimal $\rho = 0.05$. SAM achieves slightly better accuracies, while attaining a minimum displaying much flatter curvatures.

VI. Discussion

Before starting the discussion and geometric analysis of the results, it is necessary to make a premise: it is not possible to draw universal conclusions about the performance of the SAM optimizer compared to SGD or Adam, as they are strongly influenced by the architecture of the model to which they are applied and the type of input data. In this study, we examine the loss landscape and its relationship, in terms of sharpness, with the eigenvalues of the loss Hessian.

The key hyperparameter that comes into play with the new SAM optimizer is ρ, which defines the trade-off between minimizing the loss and sharpness. Hence, the larger the value of ρ, the stronger the weight perturbation, pushing the model to find a flatter loss landscape and potentially achieving greater differences in results. However, very small values of ρ (approaching zero) make the model practically the same as the one without using SAM, but with significantly higher computational cost. It is also worth noting that the chosen value of ρ for the three models increases with their complexity (number of trainable weights): for GCN, the optimal value is much smaller compared to the other two cases. As an illustrative example, we plot the loss landscape in Figure 2 for the case of WideResNet, to observe that the minimum identified by SAM is indeed flatter than in the case of SGD.

Regarding the GCN model (Figure 3), we visualize the loss landscape using contour plots, which allow us to observe an even more pronounced difference in the sharpness of the function.

Indeed, by observing the loss landscape, it is evident that the optimizer finds flatter minima, as discussed earlier. However, this does not always directly translate into a significant improvement in generalizability on our test sets. As discussed in the corresponding section, it sets the groundwork for better performance on any new data and reduces overfitting.

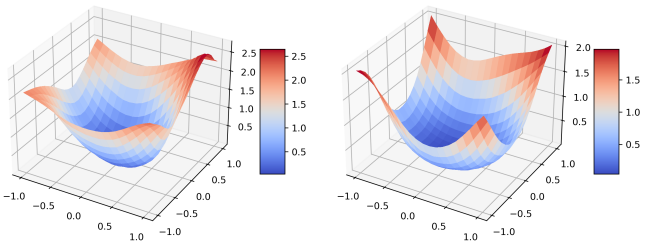

Fig. 2. Surface plots of the loss landscape for WideResNet. Left: SGD optimizer, right: SAM optimizer

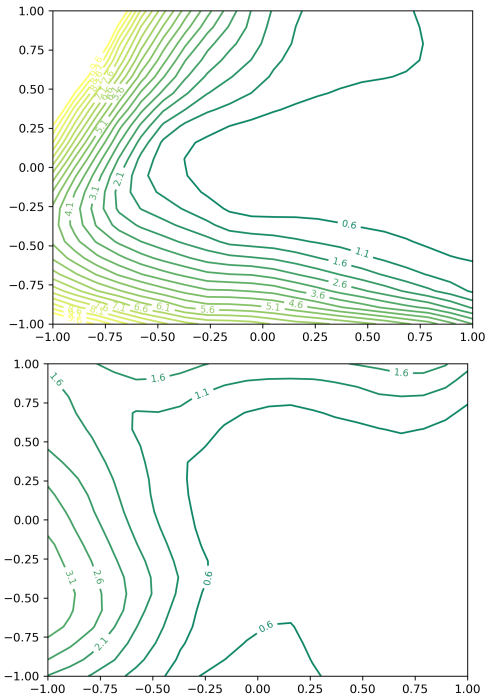

Fig. 3. Contour plots of the loss landscape for GCN. Left: SGD optimizer, right: SAM optimizer

VII. Conclusion

Based on the obtained results and the consulted literature, the analysis presented in this report has allowed us to observe the benefits and limitations of the SAM optimizer. Specifically, the improvement nature of SAM compared to a standard optimizer is interesting, as it is capable of considering geometric information about the sharpness of the loss function. This addition does not deteriorate the results and potentially provides the model with increased generative capacity, as discussed in the literature regarding the qualities of minimal flatness. These assumptions have been tested on three very diverse architectures. More significant results could be achieved through a search for optimal hyperparameters and an exploration of the influence of the batch size on the loss geometry.

All the results are reproducible and the code can be found in GitHub at this link.

REFERENCES

[1] P. Foret, A. Kleiner, H. Mobahi, and B. Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," 2021.

[2] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, "On large-batch training for deep learning: Generalization gap and sharp minima," 2017.

[3] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio, "Fantastic generalization measures and where to find them," 2019.

[4] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," GitHub repository: https://github.com/tomgoldstein/loss-landscape.git, 2018.

[5] Z. Yao, A. Gholami, K. Keutzer, and M. Mahoney, "Pyhessian: Neural networks through the lens of the hessian," GitHub repository: https://github.com/amirgholami/PyHessian.git, 2020.

[6] Z. Wei, J. Zhu, and Y. Zhang, "On the relation between sharpness-aware minimization and adversarial robustness," *arXiv preprint arXiv:2305.05392*, 2023.

[7] S. Zagoruyko and N. Komodakis, "Wide residual networks," 2017.

[8] mselmangokmen, "Timeseriesproject," https://github.com/mselmangokmen/TimeSeriesProject.git, 2023.

[9] C. Morris, M. Ritzert, M. Fey, W. L. Hamilton, J. E. Lenssen, G. Rattan, and M. Grohe, "Weisfeiler and leman go neural: Higher-order graph neural networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, no. 01, 2019, pp. 4602–4609.

[10] W. L. Hamilton, "Graph representation learning," *Synthesis Lectures on Artifical Intelligence and Machine Learning*, vol. 14, no. 3, pp. 1–159, 2020.

[11] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann, "Tudataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020. [Online]. Available: www.graphlearning.io