Achieving better model Generalization

Anonymous Authors[1]

Abstract

The goal of the assignment is to review the following papers: "SHARPNESS-AWARE MINIMIZATION FOR EFFICIENTLY IMPROVING GENERALIZATION" paper written by Pierre Foret, Ariel Kleiner, Hossien Mobahi and Behnam Neyshabur, which was published as a conference paper at ICLR 2021, with the goal of re implementing and expending on the methods and experiences presented in the solution.As well as discussing, "Improved Sample Complexities for Deep Networks and Robust Classification via an All-Layer Margin" by Colin Wei and Tengyu Ma and "Exploring the Vulnerability of Deep Neural Networks: A Study of Parameter Corruption" by Xu Sum, Zhiyuan Zhang, Xuancheng Ren, Ruixuan Luo, Liangyou Li. After analyzing the mentioned papers,the motivation was to verify and hopefully extend in the overall topic covered by them, which is the study of generalization of models. The implementation focused primarily on re-implementing the SAM algorithm and attempt to expand the studies conducted by creating a custom model. The results showed how SAM benefits from longer periods of training and will outperforms different optimizers giving the necessary number of epochs.

1. Introduction

After conducting image processing in one of our assignments for the course, I developed an interest in the topic. Doing further research about it I came across the paper done by Pierre, Ariel, Hossien and Behnam which introduced me to the topic of overparameterized models that would not efficiently generalize. Intrigued by the their proposal, I decided to follow the research by looking in the papers referenced by the "Sharpness-Aware Minimization" paper. Having decided on the topic and papers to cover, I was motivated to determine the effectiveness of the proposed algorithm and test its efficiency for novel models. Then, after thoroughly reading how they implemented the SAM algorithm and gathered the empirical data provided plus doing multiple researches on the topic, I re-implemented the algorithm and developed my own model besides testing on already established ones.Gathering interesting results and excitement for future tests and applications.

2. Review of Sharpness-Aware Minimization for efficiently improving generalization Paper to Implement/Extend

2.1. StoryLine

Majority of new solutions developed in recent years contain heavily overparameterized models which minimizes the assurance on the model generalization efficiency. With motivation from prior works exploring the relationship between the geometry of the loss landscape and generalization, the paper displays a new method to minimize both the loss value and sharpness. Sharpness-Aware Minimization(SAM), the name of the new procedure, focus on parameters in neighborhoods having uniformly low loss, where gradient descent can be applied to solve the min-max optimization problem that it generates. SAM was tested and proven to increase performance for diverse datasets such as CIFAR-10,100 and ImageNet. Also, SAM has built in robustness to identify noise as good as state-of-the-art procedures that focus on learning with noisy labels.

High-Level Motivation The goal of SAM is to create better models to generalize models more efficiently while reducing error rate when testing new data. It accomplishes this by utilizing the connection between the geometry of the loss landscape, particularly the flatness of minima and generalization. The creation of SAM not only yields positive, state-of-the-art results on intensely-studied task like ImageNet,CIFAR-10,100 and image classification finetunning tasks, but it builds on the connection between loss and sharpness and generalization by creating a promising new notion of sharpness, termed m-sharpness.

Prior Work The topic of finding a "flat" minima was first conducted by Hochreiter & Schmidhuber (1995), while how it connects to generalization had it focused intensified by Shirish Keskar et al., 2016; Dziugaite & Roy, 2017; Neyshabur et al., 2017; Dinh et al., 2017. Recetly, a study by Jiang et al. (2019), showed sharpness-based measure has high correlation with generalization, which encourages

penalizing sharpness. The work that is the closest to SAM was done by Sun et al. (2020), which focuses on resilience to random and adversarial corruption to expose a model's vulnerabilities. There is a difference, however, in the basis, which SAM was motivated by a principled starting point in generalization.

Research Gap The gap in research is how to bridge the connection between generalization and the geometry of the loss landscape, it is a gap in both the knowledge behind how they relate to each other plus in the experiments side, as there is a need for better generalization models.

Contributions There are two main avenues for future works with the research results. In the theoretical realm, the idea of per-data-point sharpness yielded by m-sharpness, opposed to the global sharpness of the entire training set which is how it is typically studied, shows a new optic on how to tackle generalization problems. Additionally, for the methodology aspect,SAM is potentially a substitute of Mixup in robust or semi-supervised methods.

2.2. Proposed Solution

To achieve the results expected, the new procedure, Sharpness-Aware Minimization, it creates a theorem, which informally states that for any scalar p greater than 0, with high probability over training set S generated from distribution D, the population loss must be equal or smaller to the max of the training set loss plus adding the Gaussian pertubation. The image above was taken from Equation(4) in Appendix A.1 Pac Bayesian Generalization Bound from [1]

$$L_{\mathscr{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon}) + \sqrt{\frac{k \log\left(1+\frac{\|\boldsymbol{w}\|_2^2}{\rho^2}\left(1+\sqrt{\frac{\log(n)}{k}}\right)^2\right) + 4\log\frac{n}{\delta} + \tilde{O}(1)}{n-1}}$$

Figure 1. Population Loss Optimization Theorem

Also, another theory generated is the m-Sharpness, which is an interesting new way to understand generalization, it calculates the sum over a set of independent E maximizations, each performed on a sum of per-data-point losses on a disjoint subset of m data points, instead of maximizing over a global sum of the training set. Finally, the last and most important part generated is the creation of the SAM algorithm. The figure below was taken from Algorithm (1) in Section 2 SHARPNESS-AWARE MINIMIZATION (SAM) from [1]

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

Figure 2. SAM Algorithm

2.3. Claims-Evidence

Claim and Evidence 1 SAM improves generalization across all settings evaluated for CIFAR-10 and CIFAR-100 for image classification from scratch for today's state-of-the-art models, without pretraining, WideResNets with Shake-Shake regularization and PyramidNet with ShakeDrop regularization. The process was obtained following the specific steps. SAM has a single hyperparameter p (the neighborhood size), which was tuned through grid search over 6 different values (0.001,0.02,0.05,0.1,0.2,0.5) using 10% of the training set as validation set. Because SAM requires two backpropagation operations, each non-SAM training run executed twice as many epochs as each SAM training run, the reported results are within the 95% confidence interval. The difference can be as big as 3.4% test error for WideResNet with 1800 epochs between SAM and SGD.From this, the results clearly show how SAM outperforms SGD on state-of-the-art models for the CIFAR-10,100, which are shown below and where obtained at Table 1, in Section 3.1 from [1].

		CIFAR-10		CIFAR-100	
Model	Augmentation	SAM	SGD	SAM	SGD
WRN-28-10 (200 epochs)	Basic	**2.7**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.5**$_{\pm 0.2}$	18.8$_{\pm 0.2}$
WRN-28-10 (200 epochs)	Cutout	**2.3**$_{\pm 0.1}$	2.6$_{\pm 0.1}$	**14.9**$_{\pm 0.2}$	16.9$_{\pm 0.1}$
WRN-28-10 (200 epochs)	AA	**2.1**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**13.6**$_{\pm 0.2}$	15.8$_{\pm 0.2}$
WRN-28-10 (1800 epochs)	Basic	**2.4**$_{\pm 0.1}$	3.5$_{\pm 0.1}$	**16.3**$_{\pm 0.2}$	19.1$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	Cutout	**2.1**$_{\pm 0.1}$	2.7$_{\pm 0.1}$	**14.0**$_{\pm 0.1}$	17.4$_{\pm 0.1}$
WRN-28-10 (1800 epochs)	AA	**1.6**$_{\pm 0.1}$	2.2$_{\pm <0.1}$	**12.8**$_{\pm 0.2}$	16.1$_{\pm 0.2}$
Shake-Shake (26 2x96d)	Basic	**2.3**$_{\pm <0.1}$	2.7$_{\pm 0.1}$	**15.1**$_{\pm 0.1}$	17.0$_{\pm 0.1}$
Shake-Shake (26 2x96d)	Cutout	**2.0**$_{\pm <0.1}$	2.3$_{\pm 0.1}$	**14.2**$_{\pm 0.2}$	15.7$_{\pm 0.2}$
Shake-Shake (26 2x96d)	AA	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**12.8**$_{\pm 0.1}$	14.1$_{\pm 0.2}$
PyramidNet	Basic	**2.7**$_{\pm 0.1}$	4.0$_{\pm 0.1}$	**14.6**$_{\pm 0.4}$	19.7$_{\pm 0.3}$
PyramidNet	Cutout	**1.9**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**12.6**$_{\pm 0.2}$	16.4$_{\pm 0.1}$
PyramidNet	AA	**1.6**$_{\pm 0.1}$	1.9$_{\pm 0.1}$	**11.6**$_{\pm 0.1}$	14.6$_{\pm 0.1}$
PyramidNet+ShakeDrop	Basic	**2.1**$_{\pm 0.1}$	2.5$_{\pm 0.1}$	**13.3**$_{\pm 0.2}$	14.5$_{\pm 0.1}$
PyramidNet+ShakeDrop	Cutout	**1.6**$_{\pm <0.1}$	1.9$_{\pm 0.1}$	**11.3**$_{\pm 0.1}$	11.8$_{\pm 0.2}$
PyramidNet+ShakeDrop	AA	**1.4**$_{\pm <0.1}$	1.6$_{\pm <0.1}$	**10.3**$_{\pm 0.1}$	10.6$_{\pm 0.1}$

Figure 3. Table comparision between SAM and SGD results

Claim and Evidence 2 SAM uniformly improves performance relative to finetuning without SAM at transfer learning by pretraining a model on a large related dataset and then finetuning on a smaller target dataset of interest. Not only this, but SAM yields novel state-of-the-art performance, including 3.92% on CIFAR-100, 11.39% error on ImageNet and 0.30% error on CIFAR-10. All the results are shown below in the figure, the content was obtained from Table 3, in section 3.2 from [1].

Dataset	EffNet-b7 + SAM	EffNet-b7	Prev. SOTA (ImageNet only)	EffNet-L2 + SAM	EffNet-L2	Prev. SOTA
FGVC_Aircraft	$6.80_{\pm 0.06}$	$8.15_{\pm 0.08}$	**5.3** (TBMSL-Net)	$\mathbf{4.82}_{\pm 0.08}$	$5.80_{\pm 0.1}$	5.3 (TBMSL-Net)
Flowers	$\mathbf{0.63}_{\pm 0.02}$	$1.16_{\pm 0.05}$	0.7 (BiT-M)	$\mathbf{0.35}_{\pm 0.01}$	$0.40_{\pm 0.02}$	0.37 (EffNet)
Oxford_IIIT_Pets	$\mathbf{3.97}_{\pm 0.04}$	$4.24_{\pm 0.09}$	4.1 (Gpipe)	$\mathbf{2.90}_{\pm 0.04}$	$3.08_{\pm 0.04}$	4.1 (Gpipe)
Stanford_Cars	$5.18_{\pm 0.02}$	$5.94_{\pm 0.06}$	**5.0** (TBMSL-Net)	$4.04_{\pm 0.03}$	$4.93_{\pm 0.04}$	**3.8** (DAT)
CIFAR-10	$\mathbf{0.88}_{\pm 0.02}$	$0.95_{\pm 0.03}$	1 (Gpipe)	$\mathbf{0.30}_{\pm 0.01}$	$0.34_{\pm 0.02}$	0.63 (BiT-L)
CIFAR-100	$\mathbf{7.44}_{\pm 0.06}$	$7.68_{\pm 0.06}$	7.83 (BiT-M)	$\mathbf{3.92}_{\pm 0.06}$	$4.07_{\pm 0.08}$	6.49 (BiT-L)
Birdsnap	$\mathbf{13.64}_{\pm 0.15}$	$14.30_{\pm 0.18}$	15.7 (EffNet)	$\mathbf{9.93}_{\pm 0.15}$	$10.31_{\pm 0.15}$	14.5 (DAT)
Food101	$7.02_{\pm 0.02}$	$7.17_{\pm 0.03}$	7.0 (Gpipe)	$\mathbf{3.82}_{\pm 0.01}$	$3.97_{\pm 0.03}$	4.7 (DAT)
ImageNet	$15.14_{\pm 0.03}$	15.3	**14.2** (KDforAA)	$\mathbf{11.39}_{\pm 0.02}$	11.8	11.45 (ViT)

Figure 4. Top 1 error rates for finetunning

Claim and Evidence 3 SAM provides a high degree of robustness to label noise, on par with ones of state-of-the art procedures specifically meant to target learning with noisy labels. It was particularly tested applying SAM in the noisy-label setting for CIFAR-10, which a fraction of the training set labels are randomly flipped. The evaluation was done for five variants, for SAM it was used p = 0.1, for all noise levles except 80% which used p = 0.05. The results are shown below, the information was gathered from Table 4, in section 3.3 from [1]

Method	Noise rate (%)			
	20	40	60	80
Sanchez et al. (2019)	94.0	92.8	90.3	74.1
Zhang & Sabuncu (2018)	89.7	87.6	82.7	67.9
Lee et al. (2019)	87.1	81.8	75.4	-
Chen et al. (2019)	89.7	-	-	52.3
Huang et al. (2019)	92.6	90.3	43.4	-
MentorNet (2017)	92.0	91.2	74.2	60.0
Mixup (2017)	94.0	91.5	86.8	76.9
MentorMix (2019)	**95.6**	**94.2**	91.3	**81.0**
SGD	84.8	68.8	48.2	26.2
Mixup	93.0	90.0	83.8	70.2
Bootstrap + Mixup	93.3	92.0	87.6	72.0
SAM	95.1	93.4	90.5	77.9
Bootstrap + SAM	95.4	**94.2**	**91.8**	79.9

Figure 5. Test accuracy on clean test set for models trained on CIFAR-10 with noisy labels

2.4. Critique and Discussion

The paper does a great job of explaining the set up and purpose of the experiments. After, it provides good evidence to support all of its claims and exemplify how the SAM produces better output.The only thing I believe they could have explained in more depth is how does their "m-sharpness" can potentially change and influence theoretical analysis on the relationship between geometry and the loss landscape. Conversely, for future areas of the actual implementation, they have a great plan of where to apply SAM to obtain better results, such as a potential substitute to Mixup in robust or semi-supervised methods.

3. Review of Improved Sample Complexities for Deep Networks and Robust Classification via an All-Layer Margin

3.1. StoryLine

The relationship between the output margins and the generalization for linear classifiers are captured in a clear and simple bound, the larger the margin the better the generalization.However, this does not apply for deep models, the analyzes of the output margins generate complicated bounds that may exponentially depend on depth. The "All layer margin", the proposed solution to analyze new notions of margin, is meant to show a clear and direct relationship between generalization for deep models.

High-Level Motivation The goal of the All layer margin is to create a better generalize for deep models, specifically implemented in three ways, obtaining thighter data-dependent generalization bounds for neural nets, for adversarial robust classification and to design a new algorithm to encourage larger all-layer margins and demonstrate improved performance on real data in both clean and adversarial robust classification settings.

Prior Work From Zhang et al. [2016], Neyshabur et al. [2017a] it was derived that deep learning often exhibits statistical properties that may be counter intuitive. This inspired the study of a variety of different generalization algorithms for deep learning, such as implicit or algorithmic regularization. Diverse papers were created later, with varying results, such as Neyshabur et al. [2017a], Arora et al. [2018] which provide complexity measures related to the data-dependent stability of the network, but the resulting bounds only apply to a randomized or compressed version of the original classifier. Finally both [Tsipras et al., 2018, Montasser et al., 2019, Yin et al., 2018, Raghunathan et al., 2019] and Yin et al. [2018] showed negative results, the first one the adversity on robust generalization and the second that the product of norms is inevitable if the standard tools of Rademacher complexity and the output margin. On the implementation for the "all layer" they were circumvented to generate better results.

Research Gap The gap in research is how to bridge the connection between predicting confidently how the relationship between output margins can influence generalization for deep models.It is a gap in both the knowledge behind how they relate to each other plus in the experiments side, as there is a need for better generalization models.

Contributions For future work with the research results, such as being the first ones to directly bound generalization of the robust classification error for any network. Which allows for expanding work, of applying this to new perspectives.The goal is allow and incentivize new research on maximizing all-layer margins as new objective for better generalization in deep learning models.

3.2. Proposed Solution

As suggested by the name the all-layer margin considers all the layers of the network simultaneously, opposite to the output margin which only considers the last layer. It follows that the definition of the all-layer margin is the key insight for deriving the definition. As shown in the picture below, the all layer fixes the issue of the uncertainty on how to properly normalize the output margin. Below the formula was gathered from formula 1.2, in section 1 from [10]

$$\text{Test error} \lesssim \frac{1}{n}\sqrt{\sum_{i=1}^{n}\left(\frac{\text{sum of the complexities of each layer}}{m_i}\right)^2} + \text{low order terms}$$

Figure 6. Formula for test clasification error that accounts for all-layer

Also, the most important developed part of the paper is the proposed algorithm for the all-layer margin during training, it is similar to the adversarial training from [Madry et al., 2017], except perturbations are applied at all hidden layers instead of the input. The algorithm can be found in Algorithm(1) in Section 5 from [10]

Algorithm 1 All-layer Margin Optimization (AMO)

procedure PERTURBEDUPDATE(minibatch $B = \{(x_i, y_i)\}_{i=1}^{b}$, current parameters Θ)
  Initialize $\delta_i = 0$ for $i = 1, \ldots, b$.
  for $s = 1, \ldots, t$ **do**
    for all $(x_i, y_i) \in B$: **do**
      Update $\delta_i \leftarrow (1 - \eta_{\text{perturb}}\lambda)\delta_i + \eta_{\text{perturb}}\nabla_\delta\, \ell(\text{FORWARDPERTURB}(x_i, \delta_i, \Theta), y_i)$
  Set update $g = \nabla_\Theta\left[\frac{1}{b}\sum_i \ell(\text{FORWARDPERTURB}(x_i, \delta_i, \Theta), y_i)\right]$.
  Update $\Theta \leftarrow \Theta - \eta(g + \nabla_\Theta R(\Theta))$. ▷ R is a regularizer, i.e. weight decay.

function FORWARDPERTURB(x, δ, Θ) ▷ The net has layers $f_1(\cdot;\Theta), \ldots, f_r(\cdot;\Theta)$, with intended perturbations $\delta^{(1)}, \ldots, \delta^{(r)}$.
  Initialize $h \leftarrow x$.
  for $j = 1, \ldots, r$ **do**
    Update $h \leftarrow f_j(h;\Theta)$.
    Update $h \leftarrow h + \|h\|\delta^{(j)}$.
  return h

Figure 7. Ally layer algorithm

3.3. Claims-Evidence

Claim and Evidence 1 By relating all-layer margin to output margin and other quantities, it improves generalization bounds for neural nets. To prove this, the derivation of the analytic lower bound on the all-layer margin neural nets with smooth activations which depend on the output margin normalized by other data-dependent quantities. Substituting this lower bound into the formula showed on figure 6, we obtain a generalization bound. The pictur show below is obtained from Theorem 3.1 in section 3 from [10]

$$\mathbb{E}_P[\ell_{0\text{-}1} \circ F] \leq O\left(\frac{\left(\sum_i \|\kappa_{(i)}^{\text{NN}}\|_{L_q(P_n)}^{2/3} a_{(i)}^{2/3}\right)^{3q/(q+2)} q\log^2 n}{n^{q/(q+2)}}\right) + \zeta$$

Figure 8. all layer theorem

This avoids exponential depth dependency, and because the bounds depend on the empirical distribution of some complexity measure computed for each training , the convergence rates are as fast as 1 / sqrt(n).

Claim and Evidence 2 All-layer gives generalization bounds for adversarially robust classification error which are analogous to the bounds in the standard settings.In section 4 from [10], its provided a natural extension of the all-layer margin to adversarially robust classification. This allows the translation of the theorem from claims and evidence 1 above, to be directly adversarially robust classification, as it simply replaces the data-dependent quantities in the theorem above with the worst case values in the adversarial neighborhood. This results in avoidance of explicit exponential dependencies on depth, meaning the analysis of generalization for the cldean settings translates directly to the adversarial settig with almost no additional steps.

Claim and Evidence 3 The training algorithm for all-layers improves empirical performance over strong baselines. The dataset used to train were the CIFAR-10,100 and the results show how AMO(All layer margin optimization algorithm) compare to standard GSG, it shows how the error is better across all different setups. The table below was found as Table 1 in Section 5 from [10]

Table 1: Validation error on CIFAR for standard training vs. AMO (Algorithm 1).

Dataset	Arch.	Setting	Standard SGD	AMO
CIFAR-10	WRN16-10	Baseline	4.15%	**3.42%**
		No data augmentation	9.59%	**6.74%**
		20% random labels	9.43%	**6.72%**
	WRN28-10	Baseline	3.82%	**3.00%**
		No data augmentation	8.28%	**6.47%**
		20% random labels	8.17%	**6.01%**
CIFAR-100	WRN16-10	Baseline	20.12%	**19.14%**
		No data augmentation	31.94%	**26.09%**
	WRN28-10	Baseline	18.85%	**17.78%**
		No data augmentation	30.04%	**24.67%**

Figure 9. Robust validation error on CIFAR10 for standard robust training vs robust AMO

3.4. Critique and Discussion

The paper does a good job in explaining and deriving the theory behind its solution and discoveries. Its to the point where it becomes difficult to navigate in its 20 page appendix section to follow all theorems. One thing they could have done is to test in different datasets and explain in depth the set up process as the dataset and architecture are great but not enough.

4. Review of Exploring the Vulnerability of Deep Neural Networks: A Study of Parameter Corruption

4.1. StoryLine

To explore better a topic with little research, the paper shows that the vulnerability of the model parameters is of crucial value to study model robustness and generalization. Since there has not been focus paid to vulnerability of model parameters in researches, the authors propose an indicator to measure the robustness of neural network parameters by exploiting their vulnerability via parameter corruption. They named it Gradient-Based corruption.

High-Level Motivation The larger goal of this research is to enhance the robustness and generalization of deep neural networks, which are increasingly being used in real-world applications. The authors argue that while DNNs have shown impressive performance in many tasks, they are vulnerable to adversarial attacks and other forms of corruption that can significantly degrade their accuracy and reliability. By studying the vulnerability of DNN parameters and proposing a new approach to enhance their robustness, the authors aim to improve the reliability and interpretability of DNNs in real-world applications.

Prior Work Previous studies showed the vulnerability or robustness of neural networks focused primarly on generating adversarial examples and adversarial training algorithms. On the adversarial training side, the previous papers aimed to design adversarial defense algorithms. In summary, the existing work mostly focuses on adversarial examples and its adversarial training. Nonetheless, the focus of the authors was on parameter corruptions of neural network as to find vulnerable components of models and design an adversarial corruption resistant training algorithm.

Research Gap The goal of the paper was to explore the vulnerability of model parameters in the study of model robustness and generalization, which has not been focus of researches. The effects of parameter corruption on DNNs and proposing a new adversarial corruption-resistant training approach that incorporates an indicator of parameter robustness, the paper can fill this gap and improve the reliability and interpretability of DNNs in real-world applications.

Contributions The main contributions are the proposal of the an indicator to measure the robustness of neural network parameters by utilizing parameter corruption such as Gradient-Based Corruption. The indicator, which is used to probe the vulnerability of different kinds of parameters with diverse structural characteristics and to improve robustness of models with respect to parameters, enhancing the training of deep neural networks.

4.2. Proposed Solution

The paper proposes an indicator that measures the maximum loss change caused by small perturbations on model parameters in the non-trivial worst-cased scenario. The authors provide and analyze two methods to understand the effect of parameter corruption, which estimate the value of the indicator based on constructive, artificial, theoretical parameter corruptions. Comparing the two methods, the random parameter corruption gives a Monte-Carlo estimation of the indicator and the gradient-based parameter corruption gives an infinitesimal estimation that can effectively capture the worst case, all of this can easily be accessed in the appendix section from [6]. Below is the two algorithms generated for the corruption, first is the random, followed by the Gradient-Based, this was found in Section 2 Parameter Corruption, subscetion 2.1, Algorithm 1 and Algoritm 2 from [6].

Algorithm 1 Random Corruption

Require: Parameter vector $\mathbf{w} \in \mathbb{R}^k$, set of corruption constraints S
1: Sample $\mathbf{r} \sim N(0, 1)$
2: Solve the random corruption $\tilde{\mathbf{a}}$ according to Eq.(4)
3: Update the parameter vector $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{a}}$

Algorithm 2 Gradient-Based Corruption

Require: Parameter vector $\mathbf{w} \in \mathbb{R}^k$, set of corruption constraints S, loss function $\mathcal{L}$ and dataset $\mathcal{D}$
1: Obtain the gradient $\mathbf{g} \leftarrow {}^{\partial \mathcal{L}(\mathbf{w};\mathcal{D})}/_{\partial \mathbf{w}}$
2: Solve the corruption $\hat{\mathbf{a}}$ in Eq.(7) with S and $\mathbf{g}$
3: Update the parameter vector $\mathbf{w} \leftarrow \mathbf{w} + \hat{\mathbf{a}}$

Figure 10. Random and Gradient-Based corruption algorithms

4.3. Claims-Evidence

Claim and Evidence 1 The proposed indicator of parameter robustness can efficiently estimate the maximum loss change when a small perturbation is applied to model parameters to evaluate the robustness against parameter corruption.The figure and table below were taken from Section 3, subsection 3.3, Table 1 and Figure 3 from [6].

Dataset	ImageNet (Acc ↑)		CIFAR-10 (Acc ↑)		PTB-LM (Log PPL ↓)		PTB-Parsing (UAS ↑)		De-En (BLEU ↑)	
Base model	ResNet-34				LSTM		MLP		Transformer	
w/o corruption	72.5 ⋆		94.3 ⋆		4.25 ⋆		87.3 ⋆		35.33 ⋆	
Approach	Random	Proposed	Random	Proposed	Random	Proposed	Random	Proposed	Random	Proposed
n=k, ϵ=10^{-4}	⋆	62.2 (-10.3)	⋆	93.3 (-1.0)	⋆	⋆	⋆	⋆	⋆	35.21 (-0.12)
n=k, ϵ=10^{-3}	⋆	22.2 (-50.3)	⋆	36.1 (-58.2)	⋆	⋆	⋆	80.6 (-6.7)	⋆	33.62 (-1.71)
n=k, ϵ=10^{-2}	30.3 (-42.2)	0.1 (-72.4)	75.1 (-19.2)	10.0 (-84.3)	⋆	4.52 (+0.27)	79.8 (-7.5)	6.1 (-81.2)	34.82 (-0.51)	0.17 (-35.16)
n=k, ϵ=10^{-1}	0.1 (-72.4)	0.1 (-72.4)	10.0 (-84.3)	10.0 (-84.3)	4.43 (+0.18)	13.25 (+9.00)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)
n=k, ϵ=1	0.1 (-72.4)	0.1 (-72.4)	10.0 (-84.3)	10.0 (-84.3)	32.21 (+27.96)	48.92 (+44.67)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)
n=100, ϵ=10^{-2}	⋆	⋆	⋆	⋆	⋆	⋆	⋆	64.6 (-22.7)	⋆	⋆
n=100, ϵ=10^{-1}	⋆	67.5 (-5.0)	⋆	⋆	⋆	⋆	⋆	11.0 (-76.3)	⋆	31.68 (-3.65)
n=100, ϵ=1	⋆	0.1 (-72.4)	⋆	⋆	⋆	⋆	87.1 (-0.2)	0.0 (-87.3)	35.25 (-0.08)	0.00 (-35.33)
n=100, ϵ=10^{1}	0.2 (-72.3)	0.1 (-72.4)	77.1 (-17.2)	44.8 (-49.5)	⋆	⋆	31.9 (-55.4)	0.0 (-87.3)	11.58 (-23.75)	0.00 (-35.33)
n=100, ϵ=10^{2}	0.1 (-72.4)	0.1 (-72.4)	10.1 (-84.2)	9.6 (-84.7)	16.90 (+12.65)	23.55 (+19.30)	0.0 (-87.3)	0.0 (-87.3)	0.00 (-35.33)	0.00 (-35.33)

Figure 11. Comparison of gradient based corruption and random corruption under corruption constraint.

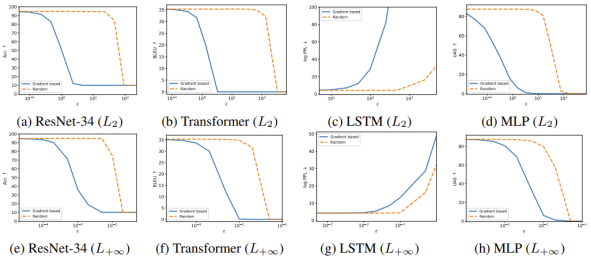
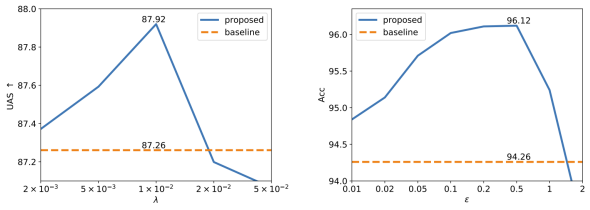


Figure 12. Results of gradient-based corruption and random corruption under the corruption constraints

Claim and Evidence 2 The proposed adversarial corruption-resistant training approach improves the parameter robustness and accuracy of DNNs on both clean and adversarial test sets.The figure and table below were taken from Section 3, subsection 3.4, Figure 5 and Appendix Table 6 from [6].



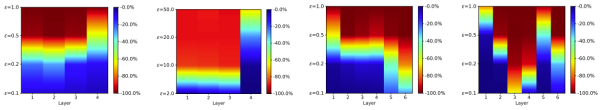
(a) PTB-Parsing (b) CIFAR-10

Figure 13. Results of corruption-resistant training.

CIFAR-10		PTB-Parsing	
ϵ	Δ Acc ↑	λ	Δ UAS ↑
0.05	+1.5	0.001	-0.2
0.1	+1.8	0.002	+0.1
0.2	+1.9	0.005	+0.3
0.5	+1.9	0.01	+0.6
1	+1.0	0.02	-0.1

Figure 14. Results of different hyper parameters in adversarial training

Claim and Evidence 3 The vulnerability of DNN parameters varies depending on the type of layer and the depth of the network.The figure and table below were taken from Appendix, Figure 6 and Table 4 from [6].



(a) ResNet-34 (ImageNet) (b) ResNet-34 (CIFAR-10) (c) Transformer Decoder (d) Transformer Encoder

Figure 15. Results of gradient-based corruption on different layers in ResNet-34

Dataset		layer1	layer2	layer3	layer4
CIFAR-10	w/o Corruption		94.3 ⋆		
	$\epsilon = 10$	⋆	⋆	⋆	⋆
	$\epsilon = 20$	93.6	⋆	87.6	⋆
	$\epsilon = 50$	62.2	29.6	31.9	⋆
	$\epsilon = 100$	13.5	13.2	11.8	92.5
	$\epsilon = 200$	10.0	10.1	10.3	71.4
	$\epsilon = 500$	10.0	10.9	9.8	30.7
ImageNet	w/o Corruption		72.5 ⋆		
	$\epsilon = 0.1$	72.2	⋆	72.2	⋆
	$\epsilon = 0.2$	71.7	72.1	71.7	⋆
	$\epsilon = 0.5$	71.0	70.5	69.9	71.3
	$\epsilon = 1$	66.5	68.5	67.9	69.3
	$\epsilon = 2$	51.6	56.9	63.0	64.2
	$\epsilon = 5$	2.5	1.4	1.4	27.2
	$\epsilon = 10$	0.1	0.1	0.1	0.1

Figure 16. Results of corrupting different layers(p = 2, n =100) of ResNet-34 on ImageNet and CIFAR-10

4.4. Critique and Discussion

The paper is really well structure and develops his proposal in a extremely logical and easy to follow format. There is good explanation for the set up required for each test and experiment ran and a lot of supporting data to prove their point. The Gradient-Based and Random corruption seem like promising and efficient creations to the study of model robustness and generalization through the exploitation of the vulnerability of parameter corruption.

5. Implementation

5.1. Implementation Motivation

The goal of the implementation is divided in two different parts, first the main goal is to replicate the results obtained in the paper the claims shown above, by re-implementing the SAM algorithm and running tests on it. To ensure the efficiency of SAM, not only I will replicate with the already used Data sets but will use them with different models. The expectation is that SAM continues to show a good performance and results.

Next, the second part will focus on trying to check that the algorithm continues to perform in untrained and novel models and hopefully see the same level of performance. The tests will be done with the already used Data sets, the expectation is to figure out if there are behaviors that were not previously demonstrated.

5.2. Implementation Setup and plan

The implementation setup and plan is to first implement the algorithm provided by the research paper in PyTorch inside of GoogleColab. In order to do this properly, it is essential that I mentioned that I found both an implementation of the SAM algorithm in Tensor flow show in [2], which guided me on the steps for the training process, as SAM has different training method compared to the regular optimizer and later a grateful surprise in this implementation in PyTorch shown in [13]. As discussed in Piazza and lecture, I did not re-implemented the algorithm by looking or rewriting, I attempted on my own and adjusted some mathematical equations at the end where I had some wrong conceptions. From these two papers, and for my whole project the only code I took was the class wrap for the Cifar-10 dataset as well as the enabling/disabling the momentum necessary for the training of the SAM algorithm.

To test and begin evaluating the efficiency of SAM, the initial dataset utilized was CIFAR-10, utilizing the WideResnet50 model. Next, the fashion mnist dataset was used, with a custom designed Model with convolution and linear layers. The experiments that will be performed are going to be to test compare the accuracy of each dataset implemented with SAM and GSD against the different datasets and models. Also, its imperative to note that due to the way SAM works as described by the paper, each ”epoch” corresponds to two ”epochs” for other optimizers, so all the tests were executed by using 50 epochs for SAM and 100 for SGD. The extra code being developed by my self, besisdes the SAM algorithm implementation, will be the importing of all the necessary libraries and initial setup code, as well as the Fashion-MNIST dataset and the code to generate table and graph analysis between them as the tests are executed. The implementation plan will help determine if the efficiency of the SAM algorithm maintains the same level of efficiency for different datasets and models, the metrics used will be both the loss value plus the accuracy of the predictions.

5.3. Implementation details

The implementation enverinment was done in GoogleColab, which utilized the V100-GPU, High-RAM options to aid training faster the model, also added all the models to cuda. Focusing on the code, a the used libraries used included, time, pytorch (Models, optim, dataloaders and transforms) and matplotlib to name a few. I insipired the outline of the Colab to match the initial code provided in [2], where I separte my sections in GoogleColab to have the initial setup for libraries and global constants, followed by the general functions and dataset imports, then I have the SAM logic, followed by both the SAM training/testing and regular optimizers training/testing functions, finishing with the training section for each different scenario tried, which included four. Important to include, that both datsets were used without pretraining. First, training CIFAR-10 with SAM using WideResnet50 model, then training CIFAR-10 dataset with SGD and WideResnet50, followed by training the Fashion-MNIST dataset with SAM and a custom model and finally the Fashion-MNIST dataset with SGD and a custom model. The focus examining the results, were to compare the loss at every epoch, and the accuracy it yielded of each optimizer wen compared to the similar dataset and model situation.

5.4. Results and Interpretations

CIFAR-10 with WideResNet50 Results The following results will compare and interpret the resuls from running the different optimizers with the CIFAR-10 dataset and WideResNet50 model. The table below, show 40 points out of the 50 epochs of the result obtained from SAM.

CIFAR-10 SAM Optimizer Model : WideResnet50

Epoch	Loss	Accuracy
50	0.0050227	0.77
49	0.0045200	0.78
48	0.0047322	0.78
47	0.0055743	0.75
46	0.0048287	0.77
45	0.0052053	0.76
44	0.0056230	0.74
43	0.0049982	0.76
42	0.0047833	0.77
41	0.0054054	0.76
40	0.0055337	0.74
39	0.0054749	0.75
38	0.0049396	0.76
37	0.0049416	0.76
36	0.0051507	0.76
35	0.0052502	0.76
34	0.0059965	0.74
33	0.0054900	0.74
32	0.0054332	0.74
31	0.0052291	0.76
30	0.0053225	0.75
29	0.0059600	0.72
28	0.0062699	0.71
27	0.0056264	0.74
26	0.0070060	0.70
25	0.0066136	0.71
24	0.0062462	0.72
23	0.0060099	0.72
22	0.0067291	0.70
21	0.0066773	0.70
20	0.0075422	0.67
19	0.0075801	0.65
18	0.0077572	0.66
17	0.0082965	0.63
16	0.0083189	0.63
15	0.0087717	0.60
14	0.0103790	0.55
13	0.0106692	0.54
12	0.0103812	0.55
11	0.0107662	0.50
10	0.0110744	0.46

Figure 17. Table of CIFAR-10 SAM optimizer with WideResNet50

The table below show 40 points out the 100 epochs ran by the SGD optimizer.

CIFAR-10 SGD Optimizer Model : WideResnet50

Epoch	Loss	Accuracy
50	0.0079512	0.77
49	0.0050159	0.78
48	0.0069126	0.77
47	0.0108359	0.77
46	0.0098387	0.77
45	0.0080280	0.76
44	0.0166933	0.76
43	0.0095632	0.77
42	0.0147150	0.77
41	0.0410585	0.74
40	0.0076652	0.75
39	0.0175576	0.74
38	0.0197807	0.76
37	0.0129813	0.75
36	0.0110074	0.75
35	0.0121421	0.75
34	0.0050986	0.75
33	0.0091087	0.74
32	0.0111549	0.73
31	0.0275180	0.74
30	0.0189033	0.74
29	0.0087128	0.73
28	0.0108844	0.71
27	0.0121925	0.72
26	0.0189212	0.72
25	0.0177716	0.71
24	0.0318367	0.71
23	0.0141704	0.71
22	0.0172028	0.70
21	0.0184158	0.69
20	0.0071241	0.68
19	0.0079208	0.62
18	0.0249987	0.66
17	0.0099252	0.68
16	0.0150743	0.66
15	0.0108398	0.64
14	0.0160988	0.63
13	0.0133302	0.62
12	0.0143658	0.60
11	0.0180817	0.25
10	0.0081994	0.58

Figure 18. Table of CIFAR-10 SGD optimizer with WideResNet50

Then, a clear way to see the accuracy over time is shown in both figures below.First the SAM, then the SGD.

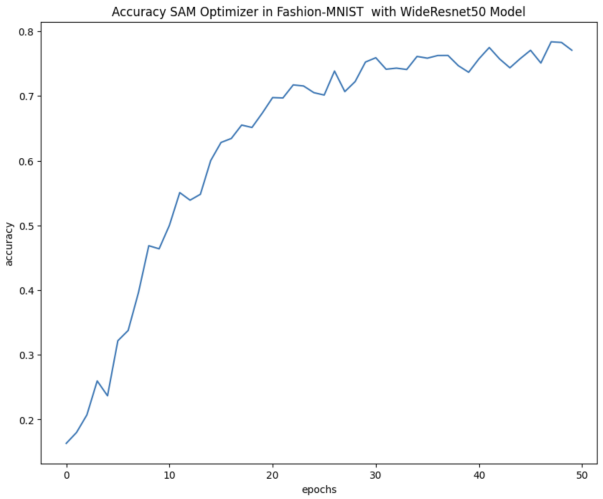


Figure 19. Accuracy SAM optimizer

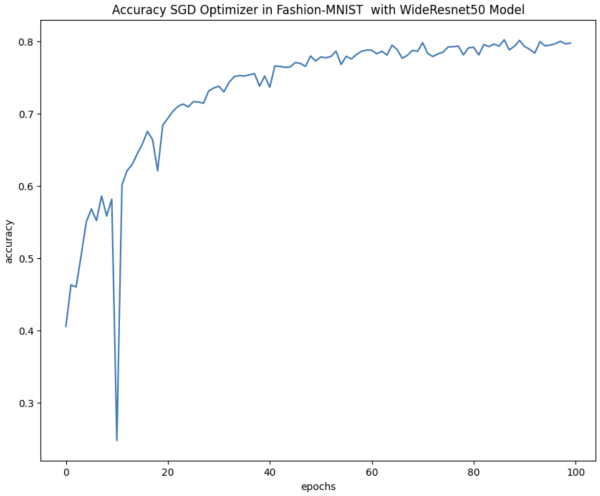


Figure 20. Accuracy SGD optimizer

CIFAR-10 with WideResNet 50 Interpretations Based on the data provided, it becomes clear that SAM achieves its purpose, it has smoother accuracy curves, which directly correlate with the loss which is the goal of the algorithm. This can be seen, especially when compared to the GSD accuracy graph. Also, it is worth to mention, that in the paper,as the authors had access to better machines and more time, to execute computational intensive tests, they ran upwards to 1800 epochs, which increase SAM performance, while not having a gigantic difference for SGD, meaning as it can be seen from the figures, that the SAM optimzer has not converged yet, while the SGD show signs that it is slowing down, this is proven from the fact that it finishes its

100 epoch with a 0.79 accuracy, while it had a 0.77 accuracy at epoch 50. This means, that given enough epochs, the SAM algorithm tend to outperform and exceed the SGD optimizer.

Fashion-MNIST with custom Model Results The following results will compare and interpret the resuls from running the different optimizers with the Fashion-MNIST dataset and a custom model made with convolution and linear layers. The table below, show 40 points out of the 50 epochs of the result obtained from SAM. Then the next table show 40 out of the 100 epochs from SGD. Followed by the accuracy figure from SAM, and then the accuracy from SGD.

Fashion-MNIST with SAM Optimizer Custom Linear Model

Epoch	Loss	Accuracy
50	0.0011980	0.39
49	0.0010802	0.47
48	0.0012734	0.26
47	0.0011388	0.40
46	0.0010822	0.44
45	0.0011121	0.40
44	0.0012060	0.37
43	0.0011638	0.39
42	0.0012729	0.36
41	0.0011209	0.41
40	0.0012933	0.34
39	0.0011386	0.39
38	0.0012855	0.33
37	0.0011455	0.34
36	0.0011753	0.33
35	0.0012627	0.28
34	0.0011944	0.37
33	0.0011893	0.37
32	0.0009304	0.51
31	0.0011305	0.40
30	0.0014130	0.32
29	0.0010311	0.53
28	0.0012950	0.37
27	0.0010400	0.49
26	0.0011085	0.42
25	0.0010646	0.43
24	0.0011202	0.44
23	0.0009552	0.53
22	0.0010598	0.49
21	0.0009431	0.51
20	0.0008002	0.62
19	0.0007312	0.64
18	0.0009215	0.53
17	0.0008559	0.58
16	0.0006591	0.63
15	0.0007186	0.70
14	0.0006964	0.69
13	0.0006711	0.68
12	0.0008372	0.63
11	0.0007651	0.65
10	0.0005874	0.72

Figure 21. Table of SAM optimizer with custom Model

Fashion-MNIST with SGD Optimizer with custom Linear Model

Epoch	Loss	Accuracy
50	0.0025352	0.85
49	0.0026994	0.84
48	0.0025647	0.85
47	0.0025536	0.85
46	0.0025385	0.86
45	0.0027248	0.84
44	0.0025554	0.86
43	0.0025946	0.85
42	0.0026040	0.85
41	0.0025937	0.85
40	0.0026283	0.85
39	0.0026849	0.85
38	0.0026812	0.85
37	0.0026586	0.85
36	0.0026491	0.85
35	0.0027097	0.85
34	0.0027336	0.84
33	0.0026952	0.85
32	0.0027163	0.85
31	0.0027857	0.84
30	0.0027399	0.84
29	0.0028807	0.83
28	0.0029204	0.83
27	0.0027810	0.84
26	0.0028733	0.83
25	0.0029135	0.83
24	0.0028642	0.84
23	0.0029411	0.83
22	0.0029526	0.83
21	0.0030012	0.82
20	0.0030416	0.82
19	0.0029809	0.82
18	0.0031462	0.82
17	0.0031458	0.82
16	0.0031106	0.82
15	0.0031557	0.81
14	0.0031989	0.82
13	0.0032314	0.81
12	0.0033450	0.79
11	0.0033998	0.80
10	0.0035114	0.80

Figure 22. Table of SGD optimizer with custom Model

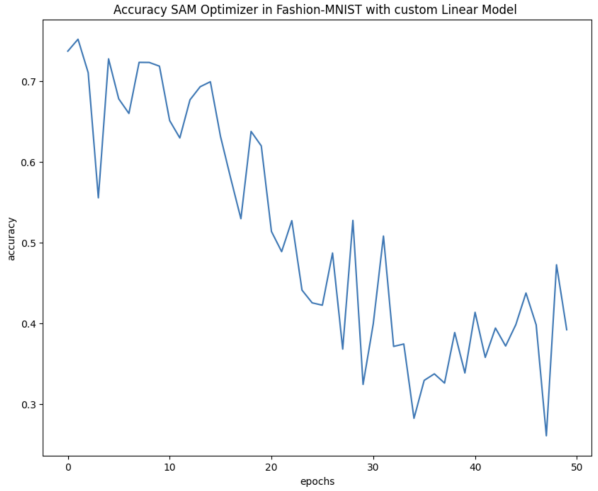


Figure 23. Accuracy SAM optimizer

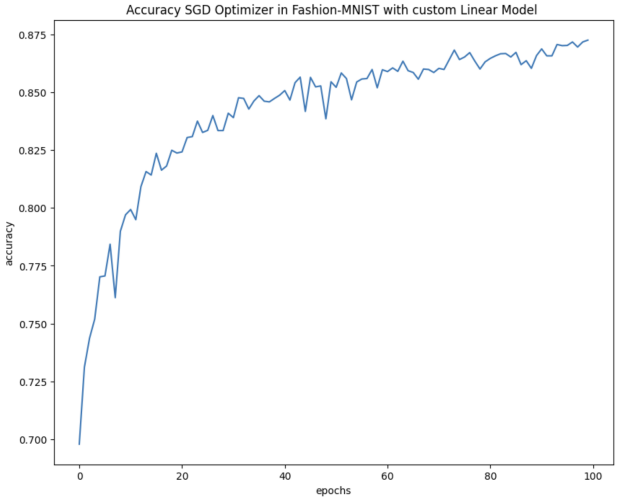


Figure 24. Accuracy SGD optimizer

Fashion-MNIST with custom Model Interpretations The results did not go as expected, as the intial thoughts were that the performance would be similar to the first experiement.From the data it shows that my model was not fine tuned and tested enough to properly be a good indicator of the effectiveness of SAM, as its accuracy jumped tremendously from point to another constantly. I believe that there were some margin to improve this performance by changing and tunning the SAM algorithm itself, but given that I am not confident enough in my model I would start by changing it first. Next, it is possible to see that GSD does have a little of jumping around but it steadily improves and converges to a good accuracy of around 0.86. Showing that GSD was a

better fit for my model.

6. Conclusion and Discussion

In conclusion, the project opened my eyes on how to properly dissect a research paper, understand its core idea and the steps necessary to implement them. Additionally, it showed me, how computational intensive it can be to do image recognition algorithms. Also, it is imperative to state how my attempt to develop a Model limited the variables I had control over when testing the SAM optimizer, a way to fix this specific issue is to either abandon the idea of creating a model or fine-tune it to be certain it is a decent model. Conversely, I am confident in my approach to the SAM algorithm implementation, but I cannot guarantee it executes exactly the intended math operations in the most optimized operations. I was able to cross check with the tensor flow option mentioned earlier and got results in the acceptable range from one another but was not able to run the whole experiment,with different data sets and models to confirm it behaves the same. Nonetheless, the results due seem to show an alignment with the paper and the original idea of confirming the proposed solution in the paper and an attempt to expand on it. This result raised the question if the SAM algorithm truly is as good as the paper makes it to be, which opened a future road of experiments where I will try to answer this question

7. Citations and References

Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2021). Sharpness-Aware Minimization for Efficiently Improving Generalization. In International Conference on Learning Representations (ICLR), January 2021. [1]

Sayak Paul. (2021). Sharpness-Aware-Minimization-TensorFlow. GitHub. https://github.com/sayakpaul/Sharpness-Aware-Minimization-TensorFlow. [2]

Hochreiter, S., & Schmidhuber, J. (1995). Simplifying neural nets by discovering flat minima. In Advances in Neural Information Processing Systems (pp. 529-536). [3]

Nitish Shirish Keskar and Richard Socher. (2017). Improving Generalization Performance by Switching from Adam to SGD. arXiv e-prints, art. arXiv:1712.07628. [4]

Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. (2016). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. arXiv e-prints, art. arXiv:1609.04836. [5]

Sun, X., Zhang, Z., Ren, X., Luo, R., & Li, L. (2020). Exploring the Vulnerability of Deep Neural Networks: A Study of Parameter Corruption. arXiv e-prints, art. arXiv:2006.05620. [6]

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In Advances in Neural Information Processing Systems, pages 5947–5956, 2017a.[7]

Gunasekar, S., Woodworth, B. E., Bhojanapalli, S., Neyshabur, B., & Srebro, N. (2017). Implicit regularization in matrix factorization. In Advances in Neural Information Processing Systems (pp. 6151–6159). [8]

Neyshabur et al. (2017a)" and "Arora et al. (2018) provide complexity measures related to the data-dependent stability of the network, but the resulting bounds only apply to a randomized or compressed version of the original classifier. [9]

Colin Wei and Tengyu Ma. Improved sample complexities for deep neural networks and robust classification via an all-layer margin. In International Conference on Learning Representations, 2020. [10]

Linjie Yang, Ping Luo, Chen Change Loy, Xiaoou Tang. A Large-Scale Car Dataset for Fine-Grained Categorization and Verification, In Computer Vision and Pattern Recognition (CVPR), 2015. [11]

davda54. (2023). SAM: Sharpness-Aware Minimization. GitHub. Available at: https://github.com/davda54/sam/tree/main [12]