FAM: Relative Flatness Aware Minimization

Linara Adilova [1] **Amr Abourayya** [2] **Jianning Li** [2] **Amin Dada** [2] **Henning Petzka** [3] **Jan Egger** [2 4] **Jens Kleesiek** [2] **Michael Kamp** [2 1 5]

Abstract

Flatness of the loss curve around a model at hand has been shown to empirically correlate with its generalization ability. Optimizing for flatness has been proposed as early as 1994 by Hochreiter and Schmidthuber, and was followed by more recent successful sharpness-aware optimization techniques. Their widespread adoption in practice, though, is dubious because of the lack of theoretically grounded connection between flatness and generalization, in particular in light of the reparameterization curse—certain reparameterizations of a neural network change most flatness measures but do not change generalization. Recent theoretical work suggests that a particular relative flatness measure can be connected to generalization and solves the reparameterization curse. In this paper, we derive a regularizer based on this relative flatness that is easy to compute, fast, efficient, and works with arbitrary loss functions. It requires computing the Hessian only of a single layer of the network, which makes it applicable to large neural networks, and with it avoids an expensive mapping of the loss surface in the vicinity of the model. In an extensive empirical evaluation we show that this relative flatness aware minimization (FAM) improves generalization in a multitude of applications and models, both in finetuning and standard training. We make the code available at github.

[1]Ruhr-University Bochum, Bochum, Germany [2]Institute for AI in medicine (IKIM) at University Hospital Essen, Essen, Germany [3]Lund University, Lund, Sweden [4]Graz University, Graz, Austria [5]Monash University, Melbourne, Australia. Correspondence to: Linara Adilova <linara.adilova@rub.de>.

Proceedings of the 2^{nd} Annual Workshop on Topology, Algebra, and Geometry in Machine Learning (TAG-ML) at the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. 2023.

1. Introduction

It has been repeatedly observed that the generalization performance of a model at hand correlates with flatness of the loss curve, i.e., how much the loss changes under perturbations of the model parameters (Chaudhari et al., 2017; Keskar et al., 2017; Foret et al., 2021; Zheng et al., 2020; Sun et al., 2020; Wu et al., 2020a; Liang et al., 2019; Yao et al., 2019). The large-scale study by Jiang et al. (2020) finds that such flatness-based measures have a higher correlation with generalization than alternatives like weight norms, margin-, and optimization-based measures. The general conclusion is that flatness-based measures show the most consistent correlation with generalization.

Naturally, optimizing for flatness promises to obtain better generalizing models. Hochreiter & Schmidhuber (1994) proposed a theoretically solid approach to search for large flat regions by maximizing a box around the model in which the loss is low. More recently, it was shown that optimizing a flatness-based objective together with an L2-regularization performs remarkably well in practice on a variety of datasets and models (Foret et al., 2021). The theoretical connection to generalization has been questionable, though, in particular in light of negative results on reparametrizations of ReLU neural networks (Dinh et al., 2017b): these reparameterizations change traditional measures of flatness, yet leave the model function and its generalization unchanged, making these measures unreliable.

Recent work (Petzka et al., 2021) has shown that generalization can be rigorously connected to flatness of the loss curve, resulting in a relative flatness measure that solves the reparameterization issue. That is, the generalization gap of a model $f : \mathcal{X} \to \mathcal{Y}$ depends on properties of the training set and a measure

$$\kappa(\mathbf{w}^l) := \sum_{s,s'=1}^{d} \langle \mathbf{w}_s^l, \mathbf{w}_{s'}^l \rangle \cdot Tr(H_{s,s'}(\mathbf{w}^l)) \ ,$$

where $\mathbf{w}^l \in \mathbb{R}^{d \times m}$ are the weights between a selected layer l with m neurons and layer $l+1$ with d neurons. Further, $\langle \mathbf{w}_s^l, \mathbf{w}_{s'}^l \rangle = \mathbf{w}_s^l (\mathbf{w}_{s'}^l)^T$ is the scalar product of two row vectors (composed of the weights into neurons with index s and s' in layer $l+1$), Tr denotes the trace, and $H_{s,s'}$ is the Hessian matrix containing all partial second derivatives

with respect to weights in rows $\mathbf{w}_s^l$ and $\mathbf{w}_{s'}^l$:

$$H_{s,s'}(\mathbf{w}, f(S)) = \left[\frac{\partial^2 \mathcal{E}_{emp}(f,S)}{\partial w_{s,t} \partial w_{s',t'}}\right]_{1\leq t,t'\leq m} .$$

Here, $\mathcal{E}_{emp}$ is the empirical risk

$$\mathcal{E}_{emp}(f,S) = \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i), y_i)$$

on a dataset

$$S = \{(x_1, y_1), \ldots, (x_n, y_n)\} \subset \mathcal{X} \times \mathcal{Y} .$$

It is demonstrated that, measured on the penultimate layer, this measure highly correlates with generalization. Sharpness-aware minimization (SAM) (Foret et al., 2021) also optimizes for a measure of flatness, but is not reparameterization invariant—even under L2-regularization its invariance is unclear, in particular wrt. neuronwise reparameterizations. The reparamterization-invariant extension of SAM, ASAM (Kwon et al., 2021) is not theoretically connected to generalization.

In this paper, we implement the *relative flatness* measure of Petzka et al. (2021) as a regularizer for arbitrary loss functions and derive its gradient for optimization. A remarkable feature of the relative flatness measure is that it is only applied to a single layer of a neural network, in comparison to classical flatness (and sharpness) which takes into account the entire network. Petzka et al. (2021) have shown that relative flatness in this layer corresponds to robustness to noise on the representation produced by this layer. Therefore, FAM nudges the entire network to produce a robust representation in the chosen layer. At the same time, it does not require flatness wrt. the other weights, opening up the design space for good minima. Since it suffices to compute relative flatness wrt. a single layer, this regularizer and its gradient can be computed much more efficiently than any full-Hessian based flatness measure. Moreover, since the gradient can be computed directly, no double backpropagation is required.

In an extensive empirical evaluation we show that the resulting relative flatness aware minimization (FAM) improves the generalization performance of neural networks in a wide range of applications and network architectures: We improve test accuracy on image classification tasks (CIFAR10, CIFAR100, SVHN, and FashionMNIST) on ResNET18 (outperforming reported best results for this architecture), WideResNET28-10, and EffNet-B7 and compare it to SAM regularizer. In a second group of experiments we reduce DICE-loss substantially on a medical shape reconstruction tasks using autoencoders and stabilize the language model finetuning.

Our contributions are

(i) a novel regularizer (FAM) based on relative flatness that is easy to implement, flexible, and compatible with any thrice-differentiable loss function, and

(ii) an extensive empirical evaluation where we show that FAM regularization improves the generalization performance of a wide range of neural networks in several applications.

2. Related Work

2.0.1. Flatness as generalization measure

Flatness of the loss surface around the weight parameters is intimately connected to the amount of information that the model with these parameters can be described with, i.e., if the region is flat enough and loss does not change, the parameters can be described with less precision still allowing to have a good performing model. Correspondingly, the models in the flat region generalize better: Hochreiter & Schmidhuber (1994) investigated a regularization that leads to a flatter region in the aforementioned sense. Their results have shown that indeed such optimization leads to better performing models. Following up, flatness of a minimizer was used to explain generalization abilities of differently trained neural networks (Keskar et al., 2016), where it was specifically emphasized that calculation of a Hessian for modern models is prohibitively costly. Originating from the minimum description length criteria for finding better generalizing learning models, flatness became a pronounced concept in the search for generalization criteria of large neural networks. The PAC-Bayes generalization bound rediscovers the connection of the Hessian as flatness characteristic with the generalization gap and the large-scale empirical evaluation (Jiang et al., 2020) shows that all the generalization measures based on flatness (in some definition) highly correlate with the actual performance of models.

Petzka et al. (2021) considered an analytical approach to connect flatness with generalization, resulting in the measure of *relative flatness* that is used for regularization in this work. Their approach splits the generalization gap into two parts termed feature robustness and representativeness. While representativeness measures how representative the training data is for its underlying distribution, feature robustness measures the loss at small perturbations in feature layers. Under the assumption of representative data, so that the training data can indeed be used to learn something about the true underlying distribution, feature robustness governs the generalization gap. Further, at a local minimum, feature robustness is exactly described by relative flatness if target labels are locally constant (i.e., labels do not change under small perturbations of features). This condition of locally constant labels is identified as a necessary condition

for connecting flatness to generalization. Finally, the paper demonstrates how the measure of relative flatness solves the reparameterization-curse discussed in Dinh et al. (2017a), rendering itself as a good candidate for an impactful regularizer.

2.0.2. Regularizing optimization

Regularization (implicit or explicit) is de facto considered to be an answer to the good generalization abilities of an overparametrized model. New elaborate techniques of regularization allow to beat state-of-the-art results in various areas. Obviously, flatness can be considered as a good candidate for a structural regularization, but since the size of the modern models grew significantly after the work of Hochreiter & Schmidhuber (1994), straightforward usage of the initial flatness measures is not feasible in the optimization. Analogously, approaches to flatness stimulation from averaging over solutions (Izmailov et al., 2018) cannot be backpropagated and directly used in the optimization process. The closest research to the flatness optimization is related to adversarial robustness—adversarial training aims at keeping the loss of a model on a constant (low) level in the surrounding of the training samples, which can be also done in the feature space (Wu et al., 2020b). Several recent works proposed an optimizer for neural networks that is approximating the minimax problem of minimizing loss in the direction of the largest loss in the surrounding of the model. One of them, sharpness aware minimization (SAM) (Foret et al., 2021) achieves state-of-the-art results in multiple tasks, e.g., SVHN, and allows for simple backpropagation through the proposed loss. However, the exact proposed m-sharpness does not entirely correspond to the theoretical motivation proposed by Foret et al. (2021) based on PAC-Bayes generalization bound, which might mean that the empirical success of SAM and its variants (Kwon et al., 2021; Zhuang et al., 2022; Du et al., 2021; Liu et al., 2022a;b) cannot be explained by theoretical PAC-Bayes flatness of the solution (Andriushchenko & Flammarion, 2022; Wen et al., 2022). Thus, introducing a theoretically grounded flatness regularizer can be of an interest for community.

3. Flatness Aware Minimization

In the following we give a detailed description of the proposed regularization. For a differentiable loss function $\ell(S, \mathbf{W})$ and a training set S, the regularized objective is

$$\ell(S, \mathbf{W}) + \lambda \kappa(\mathbf{w}^l) \ ,$$

where λ is the regularization coefficient and $\mathbf{w}^l \in \mathbb{R}^{m \times d}$ denote the weights from layer l to $l+1$. To optimize this objective, we compute its gradient (and omit the training set S in the notation for compactness):

$$\nabla_{\mathbf{W}} \left(\ell(\mathbf{W}) + \lambda \kappa(\mathbf{w}^l) \right) = \nabla_{\mathbf{W}} \ell(\mathbf{W}) + \lambda \nabla_{\mathbf{W}} \kappa(\mathbf{w}^l) \quad (1)$$

Here, $\nabla_{\mathbf{W}} \ell(\mathbf{W})$ is the standard gradient of the loss function. It remains to determine $\nabla_{\mathbf{W}} \kappa(\mathbf{w}^l)$.

Lemma 1. *For a neural network with L layers and weights $\mathbf{W} = (\mathbf{w}^1, \dots, \mathbf{w}^L)$ with $\mathbf{w}^k \in \mathbb{R}^{O^k \times P^k}$ and a specific layer $l \in [L]$ with weights $\mathbf{w}^l \in \mathbb{R}^{d \times m}$ it holds that*

$$\nabla_{\mathbf{W}} \kappa(\mathbf{w}^l) = e^l \left[2 \sum_{s=1}^{d} w_s^l Tr\left(H_{s,i}\right) \right]_{i \in [d]} + \left(\left[\sum_{s,s'=1}^{d} \left\langle w_s^l, w_{s'}^l \right\rangle \sum_{t=1}^{m} \frac{\partial^3 \ell(\mathbf{W})}{\partial \mathbf{w}_{o,p}^k \partial \mathbf{w}_{s,t}^l \partial \mathbf{w}_{s',t}^l} \right]_{\substack{p \in [P^k] \\ o \in [O^k]}} \right)_{k \in [L]}$$

where e^l denotes the l-th standard unit vector in $\mathbb{R}^L$.

Proof. For this proof, we simply apply product rule on the gradient of the regularizer, yielding two parts that we separately simplify.

$$\begin{aligned}
\nabla_{\mathbf{W}} \kappa(\mathbf{w}^l) &= \nabla_{\mathbf{W}} \sum_{s,s'=1}^{d} \left\langle w_s^l, w_{s'}^l \right\rangle Tr\left(H_{s,s'}\right) \\
&= \sum_{s,s'=1}^{d} \left(\nabla_{\mathbf{W}} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right) \\
&\quad + \sum_{s,s'=1}^{d} \left\langle w_s^l, w_{s'}^l \right\rangle \nabla_{\mathbf{W}} Tr\left(H_{s,s'}\right) \\
&= \Bigg[\underbrace{\sum_{s,s'=1}^{d} \left(\frac{\partial}{\partial \mathbf{w}^k} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right)}_{(I)} \Bigg]_{1 \le k \le L} \\
&\quad + \Bigg[\underbrace{\sum_{s,s'=1}^{d} \left\langle w_s^l, w_{s'}^l \right\rangle \frac{\partial}{\partial \mathbf{w}^k} Tr\left(H_{s,s'}\right)}_{(II)} \Bigg]_{1 \le k \le L}
\end{aligned}$$

Let us simplify both parts, starting with (I), which is $= 0$ for all $k \neq l$. For $k = l$ it simplifies to

$$\begin{aligned}
&\sum_{s,s'=1}^{d} \left(\frac{\partial}{\partial \mathbf{w}^l} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right) \\
&= \left[\sum_{s,s'=1}^{d} \left(\frac{\partial}{\partial \mathbf{w}_i^l} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right) \right]_{1 \le i \le d}
\end{aligned}$$

Now for each $i \in [d]$ we have that

$$\sum_{s,s'=1}^{d} \left(\frac{\partial}{\partial \mathbf{w}_i^l} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right) = 2 \sum_{s=1}^{d} w_s^l Tr\left(H_{s,i}\right) \quad ,$$

where we have used the symmetry of $H_{s,s'}$ and the commutativity of the inner product in the last step. Therefore, it holds that

$$\sum_{s,s'=1}^{d} \left(\frac{\partial}{\partial \mathbf{w}^l} \left\langle w_s^l, w_{s'}^l \right\rangle \right) Tr\left(H_{s,s'}\right) = \left[2 \sum_{s=1}^{d} \left\langle w_s^l, w_i^l \right\rangle Tr\left(H_{s,i}\right) \right]_{1 \le i \le d} \quad .$$

For the second part (II), let $\mathbf{w}^k \in \mathbb{R}^{O \times P}$. Then, $\frac{\partial}{\partial \mathbf{w}^k} Tr\left(H_{s,s'}\right)$ can be expressed as

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}^k} Tr\left(H_{s,s'}\right) &= \frac{\partial}{\partial \mathbf{w}^k} Tr \left[\frac{\partial^2 \ell(\mathbf{W})}{\partial \mathbf{w}_{s,t}^l \partial \mathbf{w}_{s',t'}^l} \right]_{1 \le t,t' \le m} \\ &= \frac{\partial}{\partial \mathbf{w}^k} \sum_{t=1}^{m} \frac{\partial^2 \ell(\mathbf{W})}{\partial \mathbf{w}_{s,t}^l \partial \mathbf{w}_{s',t}^l} \\ &= \left[\sum_{t=1}^{m} \frac{\partial^3 \ell(\mathbf{W})}{\partial \mathbf{w}_{o,p}^k \partial \mathbf{w}_{s,t}^l \partial \mathbf{w}_{s',t}^l} \right]_{\substack{1 \le p \le P \\ 1 \le o \le O}} \end{aligned}$$

Putting (I) and (II) together finally yields

$$\begin{aligned} \nabla_{\mathbf{W}} \kappa(\mathbf{w}^l) = e^l & \left[2 \sum_{s=1}^{d} \left\langle w_s^l, w_i^l \right\rangle Tr\left(H_{s,i}\right) \right]_{1 \le i \le d} \\ & + \left[\sum_{s,s'=1}^{d} \left\langle w_s^l, w_{s'}^l \right\rangle \sum_{t=1}^{m} \frac{\partial^3 \ell(\mathbf{W})}{\partial \mathbf{w}_{o,p}^k \partial \mathbf{w}_{s,t}^l \partial \mathbf{w}_{s',t}^l} \right]_{\substack{1 \le k \le L \\ 1 \le p \le P^k \\ 1 \le o \le O^k}} \end{aligned}$$

where e^l denotes the l-th standard unit vector in $\mathbb{R}^L$. □

3.1. Computational Complexity

Computing the FAM regularizer requires computing the Hessian wrt. the weights $\mathbf{w}^l \in \mathbb{R}^{d \times m}$ of the feature layer, which has computational complexity in $\mathcal{O}\left(d^2 m^2\right)$. From this, the individual $H_{s,s'}$ can be selected. The inner product computation has complexity $\mathcal{O}(dm)$, so that the overall complexity of computing the regularizer is in $\mathcal{O}\left(d^2 m^2\right)$.

In order to train with the FAM regularizer, we have to compute the gradient of the regularized loss wrt. the weights $\mathbf{W}$ of the network. Computing the gradient of the loss function in equation 1 has complexity $\mathcal{O}\left(|\mathbf{W}|\right)$, where $|\mathbf{W}|$ denotes the number of parameters in $\mathbf{W}$. The computation of $\nabla_{\mathbf{W}} \kappa(\mathbf{w}^l)$ is decomposed into the sum of two parts in Lemma 1. The first part has complexity $\mathcal{O}\left(d^2 m^2\right)$ for computing the Hessian and the inner product, as before. All parts in the sum, however, have already been computed when computing $\kappa(\mathbf{w}^l)$. The second part requires computing the derivative of the Hessians $H_{s,s'}$ wrt. each parameter in $\mathbf{W}$. Since we only need to compute the derivative wrt. the trace, i.e., the sum of diagonal elements, the complexity is in $\mathcal{O}\left(\mathbf{W}\right)$. Therefore, the overall complexity of computing the FAM regularizer is in

$$\mathcal{O}\left(|\mathbf{W}| + d^2 m^2 + |\mathbf{W}|\right) = \mathcal{O}\left(|\mathbf{W}| + d^2 m^2\right) \quad .$$

That is, the additional computational costs for using the FAM regularizer is in $\mathcal{O}\left(d^2 m^2\right)$ per iteration, i.e., in the squared number of weights of the selected feature layer.

In practice, computational time currently can exceed the vanila and SAM training time on $20\% - 40\%$. We also observe that GPU utilization for our implementation is still not optimal, which possibly can lead to delays in computation.

3.2. A Simplified Relative Flatness Measure

A more computationally efficient approximation to relative flatness, proposed by Petzka et al. (2019), does not iterate over individual neurons, but computes the weight norm of layer l and the trace of the Hessian wrt. layer l:

$$\widehat{\kappa}(\mathbf{w}^l) = \|\mathbf{w}^l\|_2^2 Tr\left(H\right) \quad .$$

Computing this measure not only avoids the loop over all pairs of neurons $s, s' \in [d]$, but also allows us to approximate the trace of the Hessian, e.g., with Hutchinson's method (Yao et al., 2020). On top of the computational efficiency, the trace approximation reduces the memory footprint, enabling us to employ FAM regularization to even larger layers—including large convolutional layers.

We provide details on the implementation of Hessian computation and Hessian trace approximation in Appendix A.

4. Experiments

In the following section we describe the empirical evaluation of the proposed flatness regularization. We compare the performance of FAM to the baseline without flatness related optimization and to SAM. We use the SAM implementation for pytorch [1] with the parameters of the base optimizer recommended by the authors. It should be mentioned here that no matter of its popularity there is no official pytorch implementation of the SAM optimizer, which results in multitude of different implementations for each of the paper

[1] https://github.com/davda54/sam

using the approach. Moreover, there are multiple tricks that should be considered when using SAM, e.g., one should take care of normalization layers and check on which of the two optimization steps they are active or non-active. We run SAM for the same amount of epochs that FAM and simple optimization, no matter that in the original work the authors doubled the amount of epochs for non-SAM approaches due to the doubled run time, thus giving SAM an advantage in our experiments. Reported result for one of the pytorch implementations of SAM on CIFAR10 with ResNet20 is 93.5% test accuracy [2]. This is the closest reported result to our setup and it should be expected that ResNet18 shows worse result than ResNet20. Unfortunately, the results for CIFAR100, SVHN, and FashionMNIST are not reported in the implementations of SAM for pytorch.

We use the FAM regularizer computed on the penultimate layer (or bottleneck layer), since it was demonstrated to be predictive of generalization in Petzka et al. (2021). Investigating the impact of the regularizer on other layers is left for future work.

Note on other flat-minima optimizers:
There are several extensions of SAM (Kwon et al., 2021; Zhuang et al., 2022; Du et al., 2021; Liu et al., 2022a;b) and other flat-minima optimizers, e.g., (Chaudhari et al., 2019; Sankar et al., 2021). We follow Kaddour et al. (2022) and do not consider them in this work due to their computational cost and/or lack of performance gains compared to original SAM.

4.1. Image Classification

Standard datasets for image classification are the baseline experiments that confirm the effectiveness of the proposed regularization. In particular, we worked with CIFAR10 and CIFAR100 (Krizhevsky & Hinton, 2009), SVHN (Netzer et al., 2011), and FashionMNIST (Xiao et al., 2017). We compare our flatness regularized training to the state-of-the-art flatness regularizer SAM. For this group of experiments we used the setups from the original SAM paper in order to compare to its performance. Nevertheless, due to the different implementation, the exact numbers reported seem to be unachievable—while we still see the improvement from using SAM optimizer, both no regularization baseline and SAM baseline are lower than in the original paper. For all experiments in this group we use the original neuronwise flatness measure for regularization without approximations introduced in Sec. 3.2.

4.1.1. CIFAR10

We have chosen ResNet18 as an architecture to solve CIFAR10. While ResNet18 is not the state of the art for this problem, it allows to confirm the hypothesis about performance of our method. The reported accuracy of this architecture on CIFAR10 is 95.55%. In our experiments we compare this baseline, that is not using flatness-related optimizations to SAM approach and our proposed regularization. Standard augmentation strategy is applied, including randomized cropping and horizontal flipping and normalization of the images. For baseline training we use the following parameters of optimization: SGD with batch size 64, weight decay of $5e-4$, momentum 0.9, and cosine annealing learning rate starting at 0.03 during 250 epochs. For FAM the optimizer parameters are kept same and λ selected to be 0.1. Finally SAM was ran with SGD with a scheduler learning rate 0.01 and momentum 0.9.

We report the results achieved in Table 1 in the row corresponding to CIFAR10.

4.1.2. CIFAR100

For solving this dataset we follow the approach taken by Foret et al. (2021). We use an EfficientNet (Tan & Le, 2019) (EffNet-B7) that is pretrained on ImageNet and then finetune it for CIFAR100. In order to obtain a good finetuning result the inputs should be at least in the ImageNet format (224×224) which significantly slows down the training. For standard training and FAM regularized training, the Adam optimizer had consistently the highest performance (compared to SGD and rmsprop) with a batch size of $B = 32$. The architecture achieves a baseline accuracy of 84.6 without regularization, and SAM achieves an accuracy of 85.8. The FAM regularizer improves the accuracy to 87.15. Note, that different from other setups, where both SAM and FAM were used together with the same hyperparameters as the vanila training, here we optimized parameters separately to avoid overfitting.

We report the results achieved in Table 1 in the row corresponding to CIFAR100.

4.1.3. SVHN AND FASHIONMNIST

Both SVHN and FashionMNIST problems are reported to reach state-of-the-art performance with SAM optimization using WideResNet28-10 architecture (Zagoruyko & Komodakis, 2016). It should be noted that SAM achieves the reported state-of-the-art result on these datasets when combined together with shake-shake regularization technique (Gastaldi, 2017), which we omitted.

The results reported by Foret et al. (2021) for SVHN are obtained using the training dataset that includes extra data (overall ~ 600000 images). Due to the time constraints we report results of training using only main training dataset ($\sim$ 70000 images). We apply AutoAugment SVHN policy (Cubuk et al., 2018), random cropping and horisontal

[2] https://github.com/moskomule/sam.pytorch

flip, and cutout (DeVries & Taylor, 2017) with 1 hole of length 16. Our training parameters are 100 epochs, learning rate of 0.1 with a multistep decay by 0.2 after 0.3, 0.6 and 0.8 of the training epochs, batch size of 128, optimizer is Nesterov SGD with momentum of 0.9 and weight decay of $5e-4$. For FAM we use $\lambda = 0.1$.

We modify FashionMNIST to have three channels, resize to 32×32, apply cutout with 1 hole of length 16, and normalize by 0.5. The training of FashionMNIST is very unstable and has oscillating learning curves with and without regularization. The used batch size is 64, learning rate is 0.01 with the same learning rate scheduler as for SVHN, the training is done for 200 epochs. Weight decay and momentum are set as in SVHN training.

Finally, in order to apply more computationally expensive neuronwise flatness regularization, we add one more penultimate fully-connected layer in the architecture of WideResNet with 64 neurons. Our experiments reveal that this additional layer does not change the outcome of the training in case of non-flatness regularized run.

With the described setup we did not achieve the accuracy reported in the original paper, that are 0.99 ± 0.01 error for SAM on SVHN with auto-augmentation and 1.14 ± 0.04 for baseline training on SVHN with auto-augmentation; 3.61 ± 0.06 error for SAM on FashionMNIST with cutout and 3.86 ± 0.14 for baseline training on FashionMNIST with cutout. We report the results achieved in Table 1 in the rows corresponding to SVHN and FashionMNIST.

4.2. Medical Shape Reconstruction

3D shape reconstruction has important applications in both computer vision (Smith et al., 2020; Chibane et al., 2020) and medical imaging (Amiranashvili et al., 2022; Li et al., 2021). Machine learning methods for shape reconstruction have become increasingly popular in recent years, however, often suffer from bad generalization, i.e., a neural network cannot generalize properly to shape variations that are not seen during training. In this experiment, we demonstrate that FAM regularizer can effectively mitigate the generalization problem in a skull shape reconstruction task, where a neural network learns to reconstruct anatomically plausible skulls from defective ones (Li et al., 2021; Kodym et al., 2020). Here, due to the large size of the layers, we used the approximated layerwise flatness measure for FAM optimization. We do not include SAM baseline here, while the general question here is whether flatness can aid in generalization for the current task.

4.2.1. Dataset

The skull dataset used in this experiment contains 100 binary skull images for training and another 100 for evaluation. The surface of a skull shape is constituted by the non-zero voxels (i.e., the '1's), and we create defective skulls by removing a portion of such voxels from each image. For the evaluation set, two defects are created for each image - one is similar to the defects in the training set while the other is significantly different in terms of its shape and size, as well as its position on the skull surface. The dimension of the skull images is 64^3.

4.2.2. Network Architecture and Experimental Setup

The neural network ($\sim$ 1M trainable parameters) follows a standard auto-encoder architecture, in which five two-strided convolutional and deconvolutional layers are used for downsampling and upsampling respectively. The output of the last convolutional layer is flattened and linearly mapped to an eight-dimensional latent code, which is then decoded by another linear layer before being passed on to the first deconvolution. The network takes as input a defective skull and learns to reconstruct its defectless counterpart.

As a baseline we train the network using a Dice loss (Milletari et al., 2016), and a Dice loss combined with the FAM regularizer , which is applied to the second linear layer (of size 64×8) of the network. We experimented with different coefficients λ that weigh the regularizer against the Dice loss. All experiments use the Adam optimizer with a constant learning rate of 10^{-4}. The trained models are evaluated on the two aforementioned evaluation sets, using Dice similarity coefficient (DSC), Hausdorff distance (HD), and 95 percentile Hausdorff distance (HD95). DSC is the main metric in practice for skull shape reconstruction (Li et al., 2021), measuring how well two shapes overlap (the higher the better[3]), while the distance measures i.e., HD and HD95 are supplementary.

4.2.3. Results and Discussion

Figure 1 shows the Dice loss curves under different weighting coefficients λ. Table 2 shows the quantitative results on the two evaluation sets, and Figure 2 shows the distribution of the evaluation results for $\lambda = 0.02, 0.002, 0.0006$ and the baseline. The DSC (100), HD (100) and $HD95$ (100) columns in Table 2 show the evaluation results at an intermediate training checkpoint (epoch 100).

These results reveal several interesting findings: (i) At both the intermediate (epoch=100) and end checkpoint (epoch=200), the training loss of the baseline network is clearly lower than that of the regularized networks (Figure 1), whereas its test accuracy is obviously worse than its regularized counterparts in terms of all metrics (Table

[3]The Dice loss (Figure 1), on the contrary, is usually implemented as $1 - DSC$, which we minimize during training.

Table 1. Results for Image Classification Tasks

	Baseline	SAM	FAM
CIFAR10	95.53 ± 0.0001	95.61 ± 0.001	$\mathbf{95.62 \pm 0.002}$
CIFAR100	84.48 ± 0.12	85.72 ± 0.08	$\mathbf{87.2 \pm 0.05}$
SVHN	97.72 ± 0.02	$\mathbf{97.84 \pm 0.05}$	97.81 ± 0.07
FashionMNIST	94.57 ± 0.28	$\mathbf{94.99 \pm 0.02}$	94.6 ± 0.04

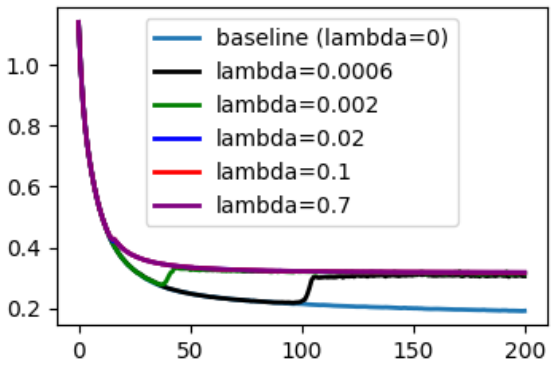

Figure 1. Curves of the Dice loss (y axis) with respect to training epochs (x axis), under different λ. Note that the red ($\lambda = 0.1$) and purple ($\lambda = 0.7$) lines overlap in this plot.

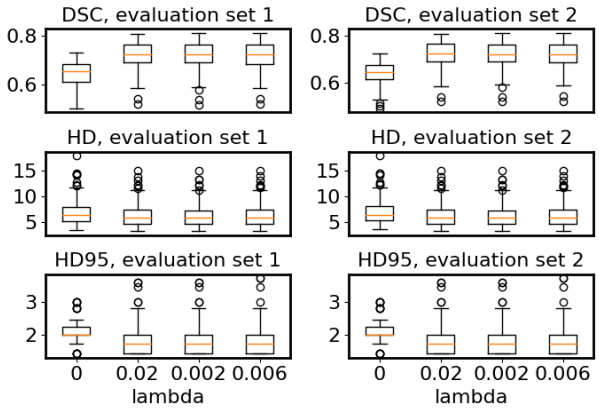

Figure 2. Boxplots of DSC, HD and HD95 given different λ (x axis) on the two evaluation sets.

2); (ii) The baseline network achieves higher test accuracy (DSC) at the intermediate checkpoint than at the end checkpoint, which is a clear indicator of overfitting, while the test accuracy of a properly regularized network (e.g., $\lambda = 0.02, 0.002$) on either evaluation set 1 or evaluation set 2 keeps improving as training progresses; (iii) Even a very loose regularization (e.g., $\lambda = 0.0006$) can prevent the Dice loss from decreasing until overfitting, as opposed to the baseline network (Figure 1); (iv) It is also worth mentioning that the scores on both evaluation sets stay essentially unchanged for the FAM-regularized network (e.g., $\lambda = 0.02$), indicating that moderately altering the defects (e.g., defect shape, size, position) does not affect the network's performance, while in contrast, the baseline network performs worse on evaluation set 2 than on evaluation set 1 in terms of all metrics.

Choosing a proper λ is important for a desired reconstructive performance. A large λ enforces a flat(ter) curve of the loss with respect to the weights of the second linear layer, which is responsible for decoding the latent codes. However, over-regularization (in our case $\lambda = 0.1, 0.7$) can lead to unvaried shape reconstructions by the decoder, since, in order for the loss to remain unchanged, the second linear layer has to give the same decoding for different latent codes [4].

[4]Different skull shapes are expected to be encoded differently through the downsampling path of the auto-encoder.

Therefore, the quantitative results for $\lambda = 0.1, 0.7$ in Table 2 should be interpreted with care, i.e., the over-regularized networks 'find' a universal reconstruction that somehow matches well with different evaluation cases (hence achieving relatively high DSC), which nevertheless defies the rule of case-specific reconstruction.

4.3. Transformers

Since the introduction of transformers (Vaswani et al., 2017), large language models have revolutionized natural language processing by consistently pushing the state-of-the-art in various benchmark tasks (Devlin et al., 2019; Clark et al., 2020; He et al., 2021). However, a recurring challenge in the fine-tuning process of these models is the occurrence of instabilities (Hua et al., 2021; Mosbach et al., 2021). These instabilities can negatively impact the performance and reliability of the fine-tuned models. In the following section we demonstrate how the application of FAM can improve the downstream performance of transformers. We do not compare with SAM both for the reasons stated in the previous set of the experiments and also due to impossibility to integrate additional gradient iteration into the used implementation of BERT.

We fine-tune $BERT_{BASE}$ (110 million parameters) (Devlin et al., 2019) to the Recognizing Textual Entailment (RTE)

Table 2. Quantitative Results for Skull Shape Reconstruction Given Different λ

methods	evaluation set 1						evaluation set 2					
	DSC	DSC (100)	HD	HD (100)	$HD95$	$HD95$ (100)	DSC	DSC (100)	HD	HD (100)	$HD95$	$HD95$ (100)
baseline	0.6464	0.6569	7.0130	7.1787	2.0635	2.0422	0.6413	0.6489	7.1421	7.1939	2.0924	2.1371
FAM, $\lambda = 0.0006$	0.7155	0.6817	6.5531	6.7772	1.8202	**1.8281**	0.7156	0.6762	6.5542	7.0115	1.8178	1.9088
FAM, $\lambda = 0.002$	0.7173	**0.7175**	**6.4813**	6.5478	**1.8175**	**1.8281**	0.7175	**0.7176**	**6.4813**	6.5478	**1.8148**	**1.8281**
FAM, $\lambda = 0.02$	**0.7176**	0.7168	6.5221	**6.5271**	1.8210	1.8344	**0.7176**	0.7168	6.5221	**6.5271**	1.8210	1.8344
FAM, $\lambda = 0.1$	**0.7176**	0.7169	6.5085	**6.5222**	1.8210	1.8345	**0.7176**	0.7169	6.5085	**6.5222**	1.8210	1.8345
FAM, $\lambda = 0.7$	**0.7177**	0.7169	6.5202	6.5389	1.8210	1.8359	**0.7177**	0.7169	6.5202	6.5389	1.8210	1.8359

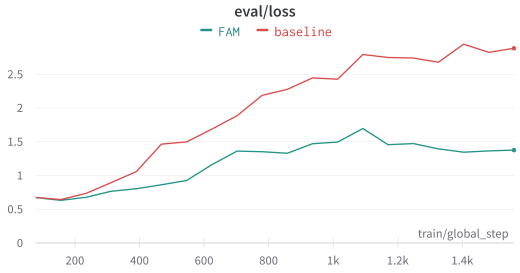

Figure 3. Development loss of the RTE training.

dataset (Dagan et al., 2006) from the General Language Understanding Evaluation benchmark (Wang et al., 2018). The dataset consists of sentence pairs with binary labels that indicate whether the meaning of one sentence is entailed from its counterpart. In the past, this dataset was found to be particularly prone to instabilities (Phang et al., 2018).

Table 3. Results for the fine-tuning on the RTE validation set.

	Baseline	FAM
Accuracy	0.67364	**0.6982**
Standard Deviation	0.018	**0.0154**
Max	0.6931	**0.7184**

In stark contrast to other experiments, we chose a much larger weighting coefficient $\lambda = 3e6$, as lower values had no influence on the training. Our training setup involved a learning rate of $\lambda = 2e{-}5$, a batch size of 32, and a maximum sequence length of 128 for 20 epochs. We report the average development set accuracy across five runs with different random seeds. Table 3 presents the results of this experiment. We observed a progressive increase in validation loss throughout the training when the regularizer was not employed, indicating severe overfitting. While this phenomenon persisted with FAM, its effect was less pronounced, as depicted in Figure 3.

5. Discussion and Conclusion

We have shown that regularization based on the theoretically sound relative flatness measure improves generalization in a wide range of applications and model architectures, outperforming standard training and sometimes SAM.

In our experiments (except for the skull reconstruction experiments, due to the specific architecture of the network), we have chosen the penultimate layer to compute relative flatness, as suggested by Petzka et al. (2021). Their theory ensures that achieving flatness in any one layer suffices to reach good generalization. We leave a comprehensive empirical study of the impact of the choice of layer (or even using multiple layers) on model quality for future work. It can be also investigated whether flatness regularized on one of the layers also changes the flatness of other layers or not.

Relative flatness is connected to generalization under the assumption of locally constant labels in the representation. This assumption holds already for the input space in many applications (e.g., image classification, and NLP)—the definition of adversarial examples hinges on this assumption. It implies, however, that flatness is not connected to generalization for tasks where the assumption is violated. The recent study by Kaddour et al. (2022) supports this empirically by showing that regularizing wrt. flatness is not always beneficial. For future work it would be interesting to verify this study with FAM, testing the assumption of locally constant labels, and expanding it to further tasks.

While current implementation of the FAM regularizer allows for achieving better performance, the performance with respect to the space consumption can be improved, as well as computational time. This currently also limits the applicability to convolutional layers, since treating them like a standard layer would increase the number of parameters greatly. This can be overcome by determining the correct structure of the FAM regularizer for convolutional layers and is an interesting direction for future work.

References

Amiranashvili, T., Lüdke, D., Li, H. B., Menze, B., and Zachow, S. Learning shape reconstruction from sparse measurements with neural implicit functions. In *International Conference on Medical Imaging with Deep Learning*, pp. 22–34. PMLR, 2022. 6

Andriushchenko, M. and Flammarion, N. Towards understanding sharpness-aware minimization. In *International Conference on Machine Learning*, pp. 639–668. PMLR, 2022. 3

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J. T., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. In *Proceedings of the International Conference of Learning Representations*, 2017. 1

Chaudhari, P., Choromanska, A., Soatto, S., LeCun, Y., Baldassi, C., Borgs, C., Chayes, J., Sagun, L., and Zecchina, R. Entropy-sgd: Biasing gradient descent into wide valleys. *Journal of Statistical Mechanics: Theory and Experiment*, 2019(12):124018, 2019. 5

Chibane, J., Alldieck, T., and Pons-Moll, G. Implicit functions in feature space for 3d shape reconstruction and completion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6970–6981, 2020. 6

Clark, K., Luong, M.-T., Le, Q. V., and Manning, C. D. ELECTRA: Pre-training text encoders as discriminators rather than generators. In *International Conference on Learning Representations*, 2020. 7

Cubuk, E. D., Zoph, B., Mane, D., Vasudevan, V., and Le, Q. V. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018. 5

Dagan, I., Glickman, O., and Magnini, B. The pascal recognising textual entailment challenge. In Quiñonero-Candela, J., Dagan, I., Magnini, B., and d'Alché Buc, F. (eds.), *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pp. 177–190, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. 8

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 7

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 6

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *International Conference on Machine Learning*, pp. 1019–1028. PMLR, 2017a. 3

Dinh, L., Pascanu, R., Bengio, S., and Bengio, Y. Sharp minima can generalize for deep nets. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, pp. 1019–1028. JMLR. org, 2017b. 1

Du, J., Yan, H., Feng, J., Zhou, J. T., Zhen, L., Goh, R. S. M., and Tan, V. Efficient sharpness-aware minimization for improved training of neural networks. In *International Conference on Learning Representations*, 2021. 3, 5

Foret, P., Kleiner, A., Mobahi, H., and Neyshabur, B. Sharpness-aware minimization for efficiently improving generalization. In *Proceedings of the International Conference on Learning Representations*, 2021. 1, 2, 3, 5

Gastaldi, X. Shake-shake regularization. *arXiv preprint arXiv:1705.07485*, 2017. 5

He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. 7

Hochreiter, S. and Schmidhuber, J. Simplifying neural nets by discovering flat minima. In *Advances in neural information processing systems*. Curran Associates, Inc., 1994. 1, 2, 3

Hua, H., Li, X., Dou, D., Xu, C., and Luo, J. Noise stability regularization for improving bert fine-tuning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3229–3241, 2021. 7

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990. 11

Izmailov, P., Wilson, A., Podoprikhin, D., Vetrov, D., and Garipov, T. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 876–885, 2018. 3

Jiang, Y., Neyshabur, B., Mobahi, H., Krishnan, D., and Bengio, S. Fantastic generalization measures and where to find them. In *International Conference on Learning Representations*, 2020. 1, 2

Kaddour, J., Liu, L., Silva, R., and Kusner, M. When do flat minima optimizers work? In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2022. 5, 8

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of International Conference on Learning Representations*, 2016. 2

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. On large-batch training for deep learning: Generalization gap and sharp minima. In *Proceedings of the International Conference on Learning Representatiosn*, 2017. 1

Kodym, O., Španěl, M., and Herout, A. Skull shape reconstruction using cascaded convolutional networks. *Computers in Biology and Medicine*, 123:103886, 2020. 6

Krizhevsky, A. and Hinton, G. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

Kwon, J., Kim, J., Park, H., and Choi, I. K. Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks. In *International Conference on Machine Learning*, pp. 5905–5914. PMLR, 2021. 2, 3, 5

Li, J., Pimentel, P., Szengel, A., Ehlke, M., Lamecker, H., Zachow, S., Estacio, L., Doenitz, C., Ramm, H., Shi, H., et al. Autoimplant 2020-first miccai challenge on automatic cranial implant design. *IEEE transactions on medical imaging*, 40(9): 2329–2342, 2021. 6

Liang, T., Poggio, T., Rakhlin, A., and Stokes, J. Fisher-rao metric, geometry, and complexity of neural networks. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2019. 1

Liu, Y., Mai, S., Chen, X., Hsieh, C.-J., and You, Y. Towards efficient and scalable sharpness-aware minimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12360–12370, 2022a. 3, 5

Liu, Y., Mai, S., Cheng, M., Chen, X., Hsieh, C.-J., and You, Y. Random sharpness-aware minimization. *Advances in neural information processing systems*, 2022b. 3, 5

Milletari, F., Navab, N., and Ahmadi, S.-A. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pp. 565–571. Ieee, 2016. 6

Mosbach, M., Andriushchenko, M., and Klakow, D. On the stability of fine-tuning {bert}: Misconceptions, explanations, and strong baselines. In *International Conference on Learning Representations*, 2021. 7

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Reading digits in natural images with unsupervised feature learning. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2011. 5

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019. 11

Petzka, H., Adilova, L., Kamp, M., and Sminchisescu, C. A reparameterization-invariant flatness measure for deep neural networks. In *Science meets Engineering of Deep Learning 2019*. Neural Information Processing Systems (NIPS), 2019. 4

Petzka, H., Kamp, M., Adilova, L., Sminchisescu, C., and Boley, M. Relative flatness and generalization. *Advances in Neural Information Processing Systems*, 34:18420–18432, 2021. 1, 2, 5, 8

Phang, J., Févry, T., and Bowman, S. R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018. 8

Sankar, A. R., Khasbage, Y., Vigneswaran, R., and Balasubramanian, V. N. A deeper look at the hessian eigenspectrum of deep neural networks and its applications to regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 9481–9488, 2021. 5

Smith, E., Calandra, R., Romero, A., Gkioxari, G., Meger, D., Malik, J., and Drozdzal, M. 3d shape reconstruction from vision and touch. *Advances in Neural Information Processing Systems*, 33:14193–14206, 2020. 6

Sun, X., Zhang, Z., Ren, X., Luo, R., and Li, L. Exploring the vulnerability of deep neural networks: A study of parameter corruption. *arXiv preprint arXiv:2006.05620*, 2020. 1

Tan, M. and Le, Q. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pp. 6105–6114. PMLR, 2019. 5

Tropp, J. A. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12:389–434, 2012. 11

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. 7

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pp. 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. 8

Wen, K., Ma, T., and Li, Z. How does sharpness-aware minimization minimizes sharpness? In *OPT 2022: Optimization for Machine Learning workshop at NeurIPS*, 2022. 3

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2958–2969. Curran Associates, Inc., 2020a. 1

Wu, D., Xia, S.-T., and Wang, Y. Adversarial weight perturbation helps robust generalization. *Advances in Neural Information Processing Systems*, 33:2958–2969, 2020b. 3

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 5

Yao, Z., Gholami, A., Lei, Q., Keutzer, K., and Mahoney, M. W. Hessian-based analysis of large batch training and robustness to adversaries. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. 1

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, pp. 581–590. IEEE, 2020. 4

Zagoruyko, S. and Komodakis, N. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016. 5

Zheng, Y., Zhang, R., and Mao, Y. Regularizing neural networks via adversarial model perturbation. *arXiv preprint arXiv:2010.04925*, 2020. 1

Zhuang, J., Gong, B., Yuan, L., Cui, Y., Adam, H., Dvornek, N. C., s Duncan, J., Liu, T., et al. Surrogate gap minimization improves sharpness-aware training. In *International Conference on Learning Representations*, 2022. 3, 5

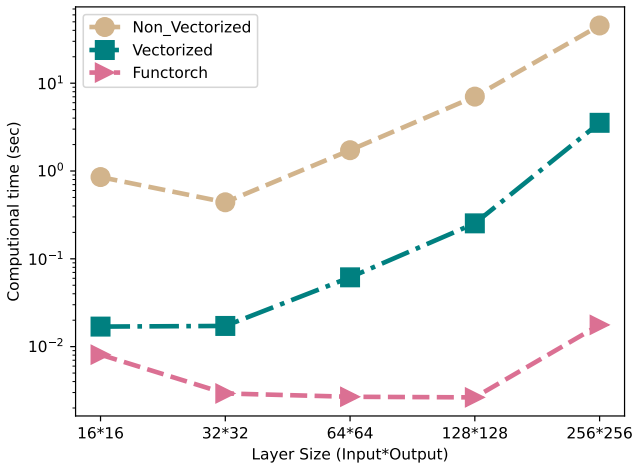

Figure 4. Comparing non-vectorized and vectorized autograd, as well as functorch in terms of the computation time for computing the full Hessian of a single neural network layer for different layer sizes.

A. Hessian Computation and Approximation

In practice, the training time for FAM regularization depends on the method used for calculating the Hessian, respectively approximating its trace in case of the simplified relative flatness measure. In the following, we discuss several practical approaches in pytorch (Paszke et al., 2019).

A.1. Computation of the Full Hessian

Computing the Hessian, i.e., the second derivatives wrt. a neural network's weights, can straight-forwardly be done in pytorch using its autograd library. This method, however, is not optimized for runtime. The torch.autograd library also provides an experimental vectorized version of the Hessian computation. It uses a vectorization map as the backend to vectorize calls to autograd.grad, which means that it only invokes it once instead of once per row, making it more computationally efficient. We compare the *non-vectorized* to the *vectorized* variant of torch.autograd. Recently, the pytorch library functorch (in beta) provided a fast Hessian computation method build on top of the autograd library and also using a vectorization map. Additionally, it uses XLA, an optimized compiler for machine learning that accelerates linear algebra computations. This further accelerates Hessian computation, but does not yet work with all neural networks—in particular, the functorch Hessian computation requires batch normalization layers to not track the running statistics of training data. In Figure 4 we show that using the vectorized approach substantially reduces computation time by up to three orders of magnitude. For larger Hessians, the functorch library further improves runtime over the vectorized autograd method by an order of magnitude. All experiments are performed on an NVIDIA RTX A6000 GPU.

A.2. Computation of the Trace of the Hessian

When the layers are high-dimensional, forming the full Hessian can be memory and computationally expensive. Since FAM requires the calculation of the trace of a Hessian, we apply the trick of using the Hutchinson's method (Hutchinson, 1990) to approximate the trace of the Hessian. The version of Hutchinson's trick we use is described as follows:

Let $A \in \mathbb{R}^{D\times D}$ and $v \in \mathbb{R}^D$ be a random vector such that $\mathbb{E}\left[vv^T\right] = I$. Then,

$$\mathrm{Tr}(A) = \mathbb{E}\left[v^T A v\right] = \frac{1}{V}\sum_{i=1}^{V} v_i^T A v_i.$$

where v is generated using Rademacher distribution and V is the number of Monte Carlo samples. The intuition behind this method is that by averaging over many random vectors, we can obtain an estimate of the trace of the matrix. It has been proved that the trace estimator converges with the smallest variance to the trace if we use Rademacher random numbers (Tropp, 2012). This method is in general very useful when we need to compute the trace of a function of a matrix.

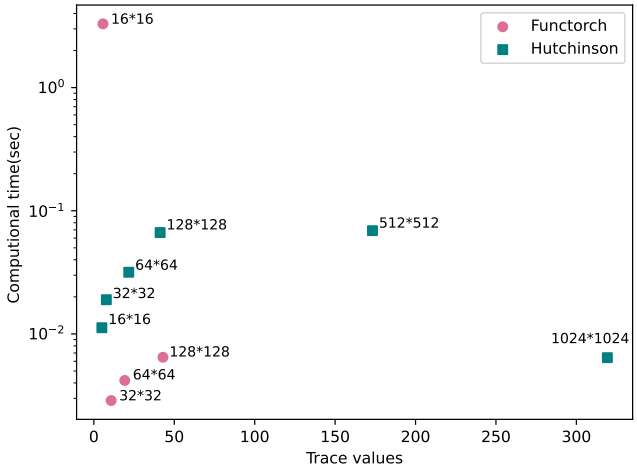

Figure 5. Computational time of the trace of the hessian for different layer sizes using Functorch and Hutchinson's method

Computational time for the direct functorch computation of the Hessian trace and for the Hutchinson's trick is shown in Figure 5.