Connection between simplicity bias and small dataset training

Bonan Sun*, Xiaotian Su*, Yuxing Yao*
{bonan.sun, xiaotian.su, yuxing.yao}@epfl.ch
EPFL, Switzerland

***Abstract*—Simplicity bias is an already widely studied phenomenon in the field of deep neural network, which is often thought to be directly related to the excellent generalization performance of neural networks. However, this feature also makes the model perform poorly on Out Of Distribution(OOD) data. We note that the field of small dataset training (We do not use Few-shot learning because our setting is different) is also affected by this problem. So based on the work of [5] and [6], we try to reduce the level of the simplicity bias by using different optimization methods and applying the same optimization methods in small dataset training to explore the connection between them. We also explore the effect of the landscape of the achieved local minima on the generalization ability of the model through the spectral of the Hessian matrix at the local minima.**

I. Introduction and Motivation

Simplicity bias is commonly believed to be one of the main features that contribute to the good generalization ability of DNN. But various work have shown that this feature meets troubles when it comes to OOD data because in many cases it only learns superficial features. Several works [4] have been proposed to solve this problem, most of them use the method of training multiple models to learn different features to reduce the dependence of the model on individual or partial features. This problem is also the main setback for small dataset learning because the samples in a small dataset do not adequately represent the actual distribution of data in the population. We believe the occurrence of simplicity bias and the poor generalization ability when using a small dataset to train the model share the same nature. The fundamental reason is that for both aspects, they can fit the training data too well:

- **Simplicity bias:** A few simple features are sufficient to distinguish between differently labeled samples in the classification problem, which eliminates the need for the model to explore deeper and more complex features and leads to simplicity bias of the model.
- **Small dataset training:** Deep neural networks with more parameters than the number of samples have the ability to remember all the samples in a small training set, which makes the training loss small. However, it does not give enough attention to the features that can be generalized.

So we believe that if a method can reduce the degree of simplicity bias, that is if it can enhance the generalization ability of the model on OOD data. Then this method can also enhance the generalization ability of the model when using small dataset training.

II. Setting for simplicity bias

We demonstrate full simplicity bias when using SGD on our synthetic dataset and a manually designed NLP dataset, and we obtain models with different levels of simplicity bias using the synthetic dataset. The idea of the synthetic dataset is from [5]. In this dataset, the concept of simplicity is well-defined. Each dimension has different number of disjoint block of data. All data points in the same block have the same label. So all dimensions are separable by piece-wise linear functions, the dimensions with less number of blocks are simpler because it need fewer pieces to separate the blocks (Fig. 1).

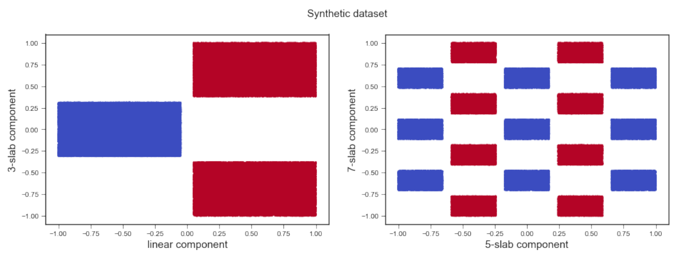

Fig. 1: one example of a two-class synthetic data, first dimension has only 2 blocks that can be separated by one straight line, it also has other dimensions with 3, 5, 7 blocks

In this project, the synthetic dataset we use has 1 dimension with 2, 3, 5, and 7 blocks respectively and the other 46 dimensions are random numbers that have no distinguishable features. We set the first dimension to the data with 2 blocks, which is the simplest dimension and the second dimension is the second simplest dimension, and so on. The 5th to the 50th dimensions are random numbers, which have no distinguishable information.

To measure the level of simplicity bias, we use two metrics: Given distribution D over $\mathbb{R}^d \times \{-1, 1\}$. Training set $\hat{D}$ is drawn from this distribution. The goal of the model is to learn a function $f : \mathbb{R}^d \rightarrow \{-1, 1\}$ from the training data $\hat{D}$.

Standard accuracy. The standard accuracy of a classifier is $\mathbb{E}_D[1_{\{f(x)=y\}}]$

***k*-randomized accuracy.** k here represents which dimension do we choose, if we choose the first dimension (which is the dimension with two blocks in the dataset we use, then $k = 1$. After choosing a k, we then randomize the value of all data points in D at dimension k to get $D^{(k)}$, then we use f which is trained from $\hat{D}$ to get the k-randomized accuracy: $\mathbb{E}_{D^{(k)}}[1_{\{f(x)=y\}}]$.

We show the existence of extreme simplicity bias when training with SGD in Appendix A1

III. Setting for small dataset training and landscape detection

To detect the landscape around the local minima the optimizers find, we choose to use the Hessian matrix. More specifically, let ℓ be the loss function and $X_{\text{train}}, Y_{\text{train}}$ be the set of training inputs and targets respectively. Then the empirical risk of the model $f(\cdot;\theta)$ on the training set is:

$$\mathcal{R}_{\text{train}}(\theta) : \mathbb{R}^n \to \mathbb{R}, \theta \mapsto \frac{1}{|X_{\text{train}}|} \sum_{x \in X_{\text{train}}, y \in Y_{\text{train}}} \ell(f(x,\theta), y),$$

where n is the number of parameters. Suppose θ_0 is a local minima learned by an optimizer, then $\nabla \mathcal{R}_{\text{train}}(\theta_0) \approx 0$, implying that

$$\mathcal{R}(\theta_0 + \delta\theta) \approx \mathcal{R}(\theta_0) + \frac{1}{2}\delta\theta^\top \mathbf{H}(\theta_0)\delta\theta + \mathcal{O}(\|\delta\theta\|^3),$$

where we omit the subscripts "train" for simplicity and $\mathbf{H}(\theta_0)$ is the Hessian matrix at θ_0. This means that the landscape around the local minima is heavily determined by the quadratic form of the Hessian $\mathbf{H}(\theta_0)$. We use the logarithm of the product of the top-k eigenvalues $\sum_{i=1}^{k} \log \lambda_i$ and the Frobenius norm $\|\mathbf{H}(\theta_0)\|_{\text{F}}$ of the Hessian matrix as two metrics to measure the flatness of the local minima. The local minima is more flat if these two metrics are small and vice versa.

To achieve diverse local minima, we follow the work in [6] and introduce an attack set $X_{\text{attack}}, Y_{\text{attack}}$, where the labels in Y_{attack} are intentionally assigned wrong labels. Then we minimize the $\mathcal{R}_{\text{all}}(\theta) := \mathcal{R}_{\text{train}}(\theta) + \lambda \mathcal{R}_{\text{attack}}(\theta)$ instead of $\mathcal{R}_{\text{train}}(\theta)$, where $\mathcal{R}_{\text{attack}}$ is the empirical risk on the attack dataset and defined similarly as $\mathcal{R}_{\text{train}}$. Since $\mathcal{R}_{\text{all}} < \varepsilon$ implies $\mathcal{R}_{\text{train}} < \varepsilon$, we can get various local minimas of $\mathcal{R}_{\text{train}}$ by optimizing $\mathcal{R}_{\text{all}}$ with different attack set size $|X_{\text{attack}}|$ and λ and they will have very different generalization abilities.

We use a small LeNet like network see Appendix A1, which only has 3782 parameters, making computing precise Hessian feasible. We train this model using only 512 training data of MNIST dataset and different combinations of $|X_{\text{attack}}|$ and λ. We repeat the experiments under each configuration 10 times. The experiment results are shown in Fig. 2. We can see that good solutions, in the sense that they have better generalization ability, are often located in flat minima while bad solutions are often located in sharp ravines.

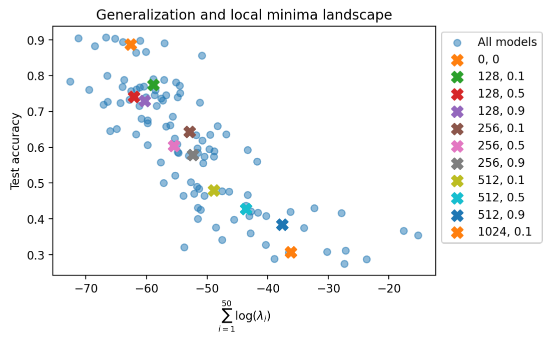

Fig. 2: Test accuracy versus the $\sum_{i=1}^{50} \log(\lambda_i)$. The crosses denote the average of the 10 experiments under each combination of $|X_{\text{attack}}|$ and λ as indicated in the legend. All models achieved 100% accuracy on X_{train} and 0.001 training loss.

IV. Effects of generalization ability in both areas with different optimization methods

We think generalization problems on OOD data of simplicity bias and that on real-world data have something in common, so we decide to try out different methods of optimization and structures of models to see how they affect this problem in both settings. Our expectation is that the direction of the effect of different methods and structures on the generalization performance of the two settings is the same.

A. *Use batch normalization layers (BN)*

Firstly, we compare the result between models that has BN and the one that does not. We have compared it with multiple optimizers, and the results are all the same, which is that the model with batch normalization can decrease the level of simplicity bias and increase the generalization ability of the model that is trained with a small dataset. The results that trained with Adam are shown in Fig. 3

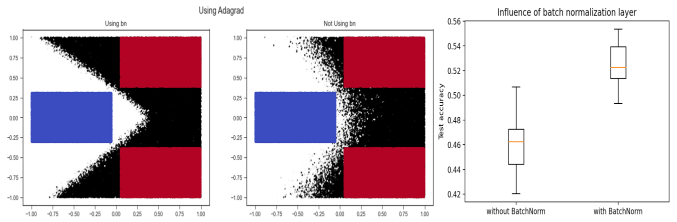

Fig. 3: (**Left, Middle**) Decision boundary in the first dimension of the synthetic dataset obtained by the model with or without BN; (**Right**) Accuracy on the test set with and without BN. Each optimizer is tested 10 times. $|X_{\text{attack}}| = 512, \lambda = 0.2$.

B. *Use different optimizers*

We also use different optimizers to see how the performance change, we've tried SGD, Adagrad and Adam. The model trained with SGD has the highest level of simplicity bias and the one trained with Adam has the lowest. The changing of generalization ability for the mdoel that trained with small dataset follows the same trend, which is just the same as expected. The model trained with Adam has the most flatten landscape and the best generalization ability while the one trained with SGD has the least and the worst. The results are shown in Fig. 4. We know that this does not correspond to common sense, because in general SGD's can achieve flatter terrain and have better generalization capabilities. We have experimented and explained this in the Appendix B.

C. *Use different learning rate*

We also test how different learning rates influence the level of simplicity bias and generalization ability. We test five different learning rate with Adam in the above two settings and the experiment results are shown in Fig. 5. We can see that small learning rates can lead to sharp minima, resulting in bad generalization ability and high level of simplicity bias. While large learning rates perform similar to each other, which can be seen when we do more experiments for each setting.

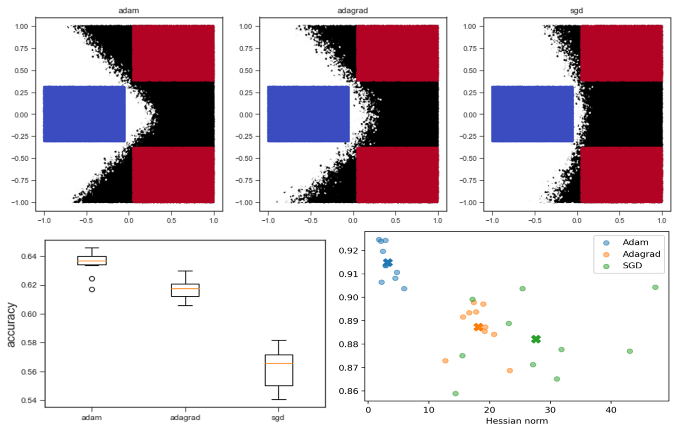

Fig. 4: (**Top**) Decision boundary in the first dimension of the synthetic dataset obtained by the model trained by the three different optimizers; (**Bottom Left**) 1-randomized accuracy of the models trained by the three optimizers on the synthetic dataset; (**Bottom Right**) Test accuracy of the models trained by the three optimizers on MNIST. The crosses denote the average of the results obtained by each optimizer. $|X_{\text{attack}}| = 0, \lambda = 0$. All the models have BN and the learning rates are 0.01 for all optimizers.

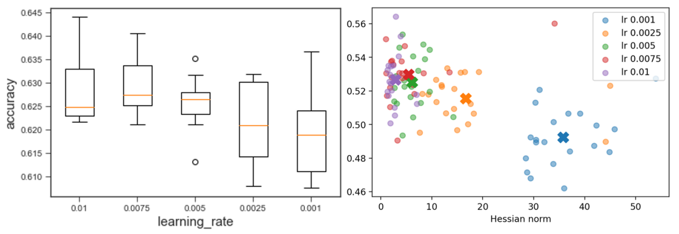

Fig. 5: (**Left**) 1-randomized accuracy of models trained by Adam with different learning rates on the synthetic dataset; (**Right**) Test accuracy of models trained by Adam with different learning rates on MNIST. Each learning rate is tested 20 times. The crosses denote the average of the results obtained by each lr. $|X_{\text{attack}}| = 512, \lambda = 0.2$.

D. *Use different momentum of SGD*

We also test the influence of different momentum of SGD. We test several different momentum in the two settings and the experiment results are shown in Fig. 6. We make the following observations: appropriate momentum of SGD can not only speed up the convergence of the optimizer, but also lead to a more flat minima, which can reduce the simplicity bias and increase the generalization ability. However, too large momentum can lead to worse results.

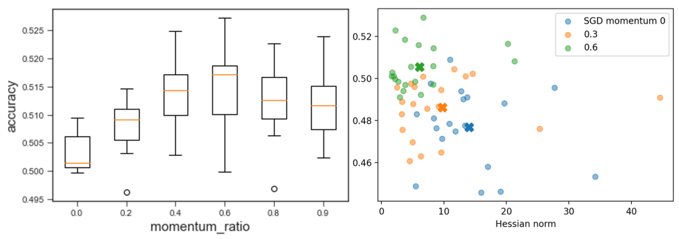

Fig. 6: (**Left**) 1-randomized accuracy of the models trained by SGD with different momentum on the synthetic dataset; (**Right**) Test accuracy on the testing set of the models trained by SGD with different momentum on MNIST. Each momentum is tested 20 times. $|X_{\text{attack}}| = 512, \lambda = 0.2$.

V. Conclusion and Theoretical Explanation

In the previous chapter, we used images to visualize the results of our experiments. In the following table, we quantitatively show the full results of our experiments.

A. *Use batch normalization layers (BN)*

experiment setting	SB accuracy[1]	SD accuracy[2]
SGD,lr=0.2	0.515	NA
SGD,lr=0.2,BN	0.601	NA
Adagrad,lr=0.2	0.580	NA
Adagrad,lr=0.2,BN	0.668	NA
Adam,lr=0.01	0.591	0.461
Adam,lr=0.01,BN	0.623	0.525

B. *Use different optimizers*

experiment setting	SB accuracy	SD accuracy
SGD,lr=0.01,BN	0.562	0.882
Adagrad,lr=0.01,BN	0.617	0.887
Adam,lr=0.01,BN	0.635	0.915

C. *Use different learning rate*

experiment setting	SB accuracy	SD accuracy
Adam,lr=0.01,BN	0.635	0.527
Adam,lr=0.0075,BN	0.629	0.530
Adam,lr=0.005,BN	0.626	0.525
Adam,lr=0.0025,BN	0.621	0.516
Adam,lr=0.001,BN	0.619	0.492

D. *Use different momentum of SGD*

experiment setting	SB accuracy	SD accuracy
SGD,lr=0.3,BN,momentum=0.0	0.503	0.476
SGD,lr=0.3,BN,momentum=0.2	0.508	NA
SGD,lr=0.3,BN,momentum=0.3	NA	0.486
SGD,lr=0.3,BN,momentum=0.4	0.514	NA
SGD,lr=0.3,BN,momentum=0.6	0.514	0.505
SGD,lr=0.3,BN,momentum=0.8	0.512	NA
SGD,lr=0.3,BN,momentum=0.9	0.511	0.478

Some experimental results are NAs, because the model may not converge in one of the two cases under the same settings, so the data are not available. However, with the available data, we can conclude that adding batch normalization, increasing the learning rate in a range, and using SGD's momentum can all reduce the degree of simplicity bias. And the degree of simplicity bias produced by different optimizers is SGD $>$ Adagrad $>$ Adam. In addition, all methods that reduce the degree of simplicity bias can enhance the generalization performance of the model trained with small dataset. This suggests a possible connection between the two, and these are worthy of further investigation.

We tentatively believe that the reason for such a phenomenon is that models with severe simplicity bias and models trained on small data can fit the data well at early stages. This means that it is more important for the model to have the ability to jump out of the uneven local minima early on, so a larger learning rate within a certain range can achieve this, and Adagrad and Adam will also have a larger learning rate than SGD early on.

[1]test accuracy when training with small data

[2]1-randomized accuracy when use the setting of simplicity bias, note that all standard accuracy in this case are 1.00

References

[1] Piotr Bojanowski et al. “Enriching Word Vectors with Subword Information”. In: *arXiv preprint arXiv:1607.04606* (2016).

[2] Robert Geirhos et al. “Shortcut learning in deep neural networks”. In: *Nature Machine Intelligence* 2.11 (2020), pp. 665–673.

[3] Alec Go, Richa Bhayani, and Lei Huang. “Twitter sentiment classification using distant supervision”. In: *CS224N project report, Stanford* 1.12 (2009), p. 2009.

[4] Matteo Pagliardini et al. “Agree to Disagree: Diversity through Disagreement for Better Transferability”. In: *arXiv preprint arXiv:2202.04414* (2022).

[5] Harshay Shah et al. “The pitfalls of simplicity bias in neural networks”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 9573–9585.

[6] Lei Wu, Zhanxing Zhu, and Weinan E. “Towards Understanding Generalization of Deep Learning: Perspective of Loss Landscapes”. In: *CoRR* abs/1706.10239 (2017). arXiv: 1706.10239. URL: http://arxiv.org/abs/1706.10239.

Appendix

A. experiment to prove the existence of simplicity bias

1) Synthetic dataset: When using a simple MLP with 5 hidden layers and 100 hidden size with the Relu activation function, the result shows extreme simplicity with 1.00 Standard accuracy and 1-randomized accuracy close to 0.5, which means the trained model depends on the simplest dimension completely. The decision boundary is shown in Fig. 7 which depends only on the dimension with 2 blocks.

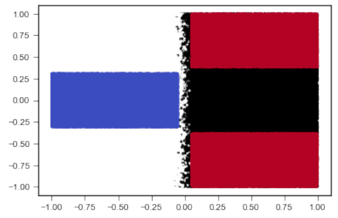

Fig. 7: The decision boundary of the model using SGD

2) NLP dataset: To examine simplicity bias in the application of natural language processing, we use twitter sentiment analysis dataset [3]. The twitter messages are classified as either positive or negative. We perform some simple feature engineering on the original tweets by removing all the hashtags and punctuations. After that, we construct tweets manually in a way to expose simplicity bias. For tweets in the negative class, we add 'ab' at the end of each text (e.g. 'home seeing Jonas brother want go back ab'); and 'cd' as the suffix of tweets that are marked as positive.

We use Fasttext [1] to embed tweets and train a model with Adam optimizer consisting of an embedding layer, a dropout layer, an LSTM layer and finally a dense layer with Sigmoid as the activation function. By concatenating random letter combinations with original texts, the model is committing to a shortcut learning [2], i.e. it simply learns a connection between predictions and suffixes. Therefore, it can easily make 100% accurate prediction on the manually-constructed dataset (see Fig. 8). However, when it comes to dataset without "indicators", its performance declines dramatically. Figure 9 suggests that the model is basically guessing the labels.

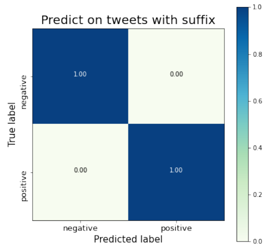

Fig. 8: Predict on tweets with suffix

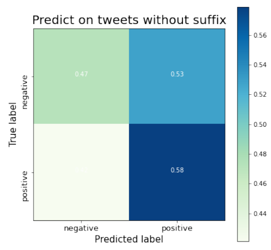

Fig. 9: Predict on tweets without suffix

B. supplementary experiments to show SGD's generalization ability with normal training process

Instead of training with small dataset, we try to use Adam and SGD optimizer to train a model separately using normal MNIST dataset. The conclusion is consistent with the attempt, SGD has better generalization ability than Adam, and its test accuracy is higher than Adam's. Therefore, we believe that the reason for the opposite conclusion between training the model on large and small data sets is that the training converges faster on small data sets, and the model needs a larger learning rate in the early stage to flush out the sharp local minima to more flatten one. While in normal training with large datasets, Adam's small learning rate in the later stages may keep it trapped in the sharp local minima.

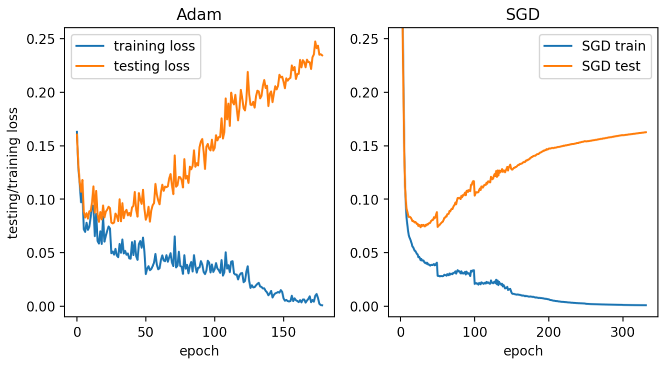

Fig. 10: Testing and training accuracy with respect to epoch of Adam and SGD on MNIST dataset. Trained with 60000 samples. We choose 0.02 and 0.2 as the learning rate for Adam and SGD respectively, with which they converge fast.