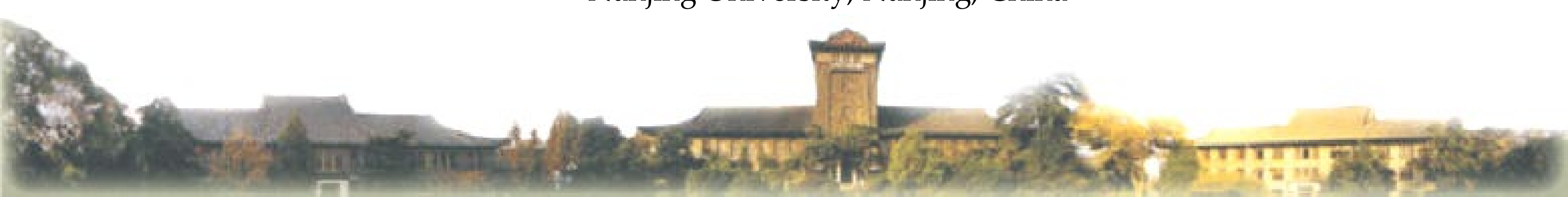
Exploring and Exploiting the Asymmetric Valley of Deep Neural Networks

Presented by Xin-Chun Li
lixc@lamda.nju.edu.cn

Joint Work with:

Jin-Lin Tang, Bo Zhang, Lan Li, De-Chuan Zhan
LAMDA Group
School of Artificial Intelligence
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing, China

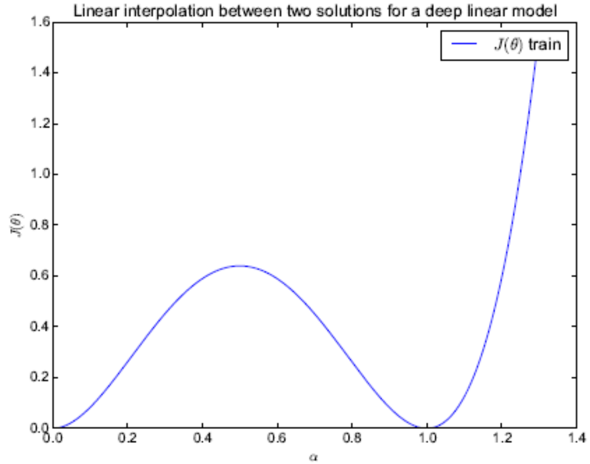


Content

- Background
- Exploring Asymmetric Valley
- Exploiting Asymmetric Valley
- Conclusion

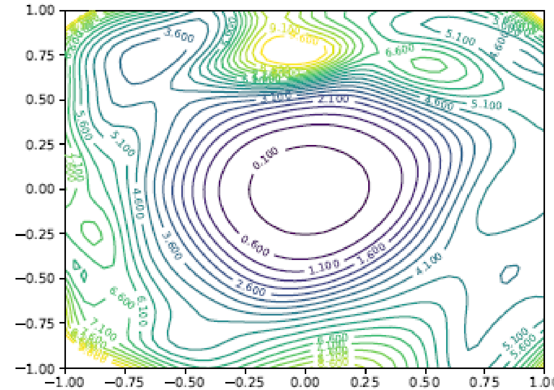
Background

Curse of Dimensionality: visualizing and understanding the HIGH-DIMENSIONAL loss landscape of DNNs is impossible



1D Visualization [1]

- $\theta + \lambda\epsilon$: given one model θ and perturbing it along noise model ϵ, with $\lambda \in [-1, 1]$



(a) $k = 1, 5.89\%$

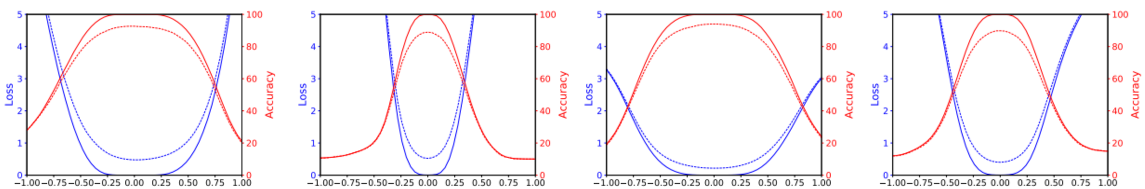
2D Visualization [2]

- $\theta + \lambda\epsilon_x + \beta\epsilon_y$: given one model θ and perturbing it along noise model ϵ_x and ϵ_y, with $\lambda, \beta \in [-1, 1]$

[1] Ian J. Goodfellow, et al. Qualitatively Characterizing Neural Network Optimization Problems. ICLR, 2015.
[2] Hao Li, et al. Visualizing the Loss Landscape of Neural Nets. NeurIPS 2018.

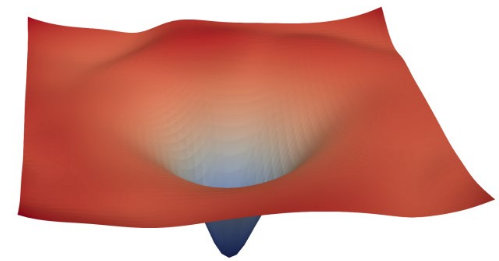
Background

The LOW-DIMENSIONAL loss landscape of DNNs seems to be not so complex, especially the DNNs with ReLU activation and Softmax classifier

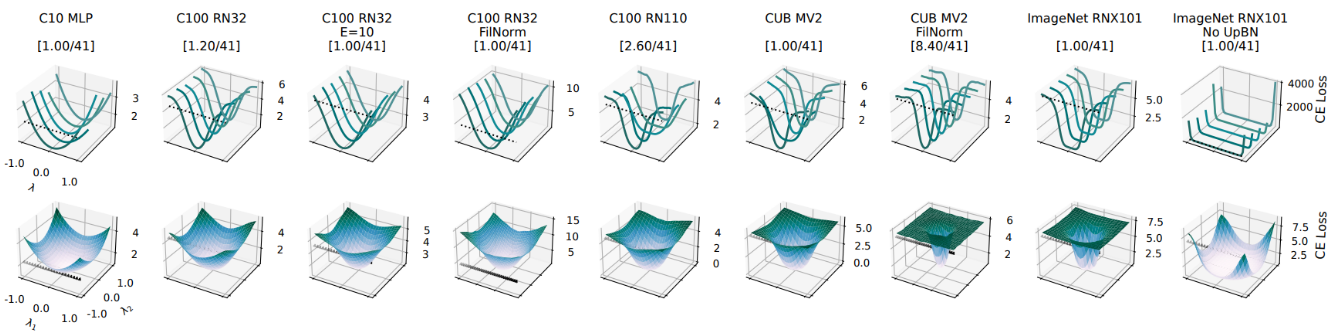


(a) 0.0, 128, 7.37% (b) 0.0, 8192, 11.07% (c) 5e-4, 128, 6.00% (d) 5e-4, 8192, 10.19%

The 1-D loss landscape of ResNet-56 shown in [1]



The 2-D loss landscape shown in [1]



The 1-D and 2-D loss landscape of **various architectures, datasets, random noise**

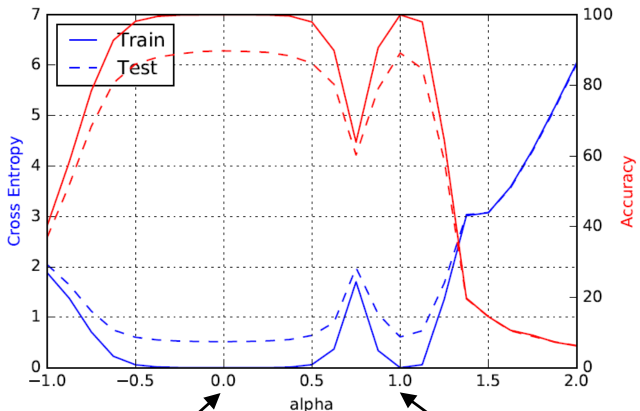
Plotting 1-D and 2-D loss surfaces by perturbing the model along Gaussian noises shows simple patterns.

Generally, the plots are nearly all symmetric ones.

[1] Hao Li, et al. Visualizing the Loss Landscape of Neural Nets. NeurIPS 2018.
[2] Xin-Chun Li, et al. Visualizing, Rethinking, and Mining the Loss Landscape of Deep Neural Networks. Arxiv 2024.

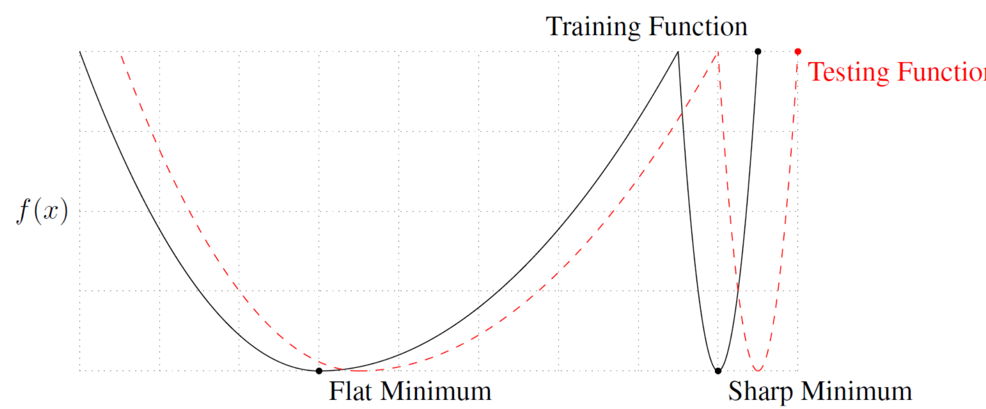
Background

The shape of loss landscape is closely related to the optimization of DNNs. For example, the flatness/sharpness of a given minima may reflect the generalization performance.



Small Batch Solution

Large Batch Solution



Small Batch Solution

Large Batch Solution

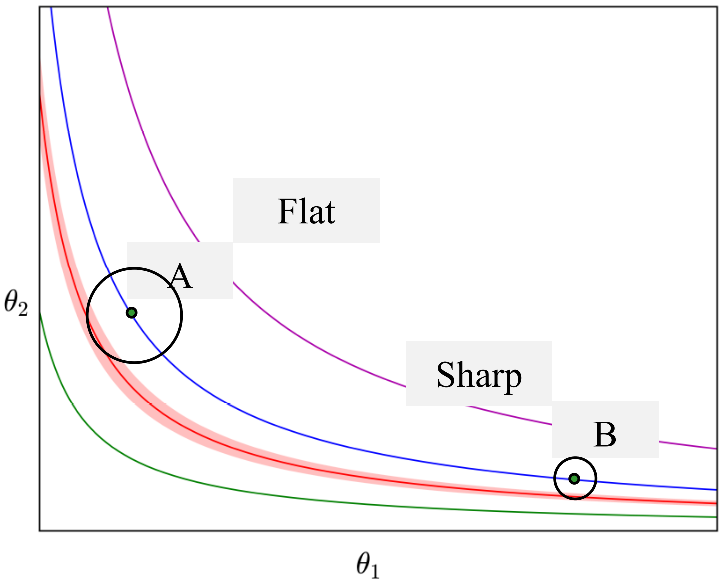
Small batch training generally leads to better performance, which is due to the flat minima [1]

Because flat minimum may be more robust to the train-test shift, the distribution shift [1]

[1] Keskar, et al. On large-batch training for deep learning: generalization gap and sharp minima. ICLR, 2017.

Background

A later work finds that calculating the flatness/sharpness should consider the influence of parameter scale.



SCALE-INVARIANCE property of DNNs:

$$\phi_{rect}(x \cdot (\alpha\theta_1)) \cdot \theta_2 = \phi_{rect}(x \cdot \theta_1) \cdot (\alpha\theta_2),$$

Definition 5. *For a single hidden layer rectifier feedforward network we define the family of transformations*

$$T_\alpha : (\theta_1, \theta_2) \mapsto (\alpha\theta_1, \alpha^{-1}\theta_2)$$

which we refer to as a α*-scale transformation.*

The loss surface of $f(\theta_1 * \theta_2)$, clearly A and B have the same loss value, but their robustness to noise perturbation differs [1]

Due to the scale-invariance property, two DNNs that are scale-invariant may show different 1-D loss curves when plotting $L(\theta + \lambda\epsilon)$ [1]

[1] Keskar, et al. On large-batch training for deep learning: generalization gap and sharp minima. ICLR, 2017.

Background

But, if we avoid the influence of parameter scales and use a FILTER-NORMALIZED plot, the flatness becomes related to the generalization again.

FILTER-NORMALIZED visualization:

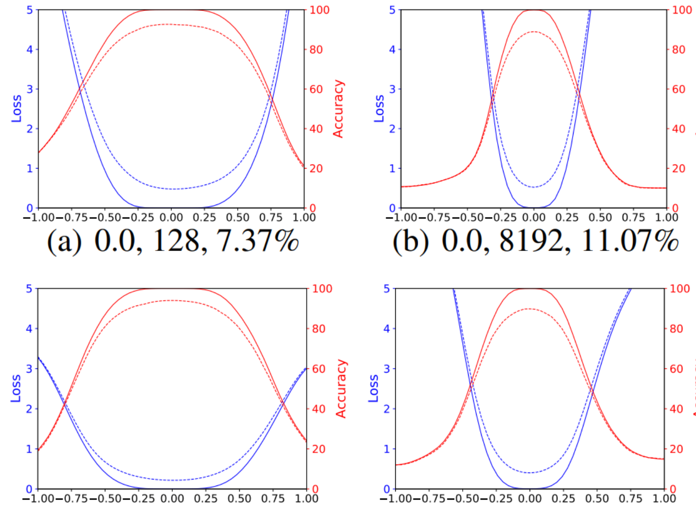
$L(\theta + \lambda d)$

$$d_{i,j} \leftarrow \frac{d_{i,j}}{\|d_{i,j}\|} \|\theta_{i,j}\|$$

This way normalizes each filter in the noise model to have the same norm of the corresponding filter in θ, which avoids the effect of parameter scales

Small Batch | Large Batch

No Weight Decay | With Weight Decay



(a) 0.0, 128, 7.37% (b) 0.0, 8192, 11.07%

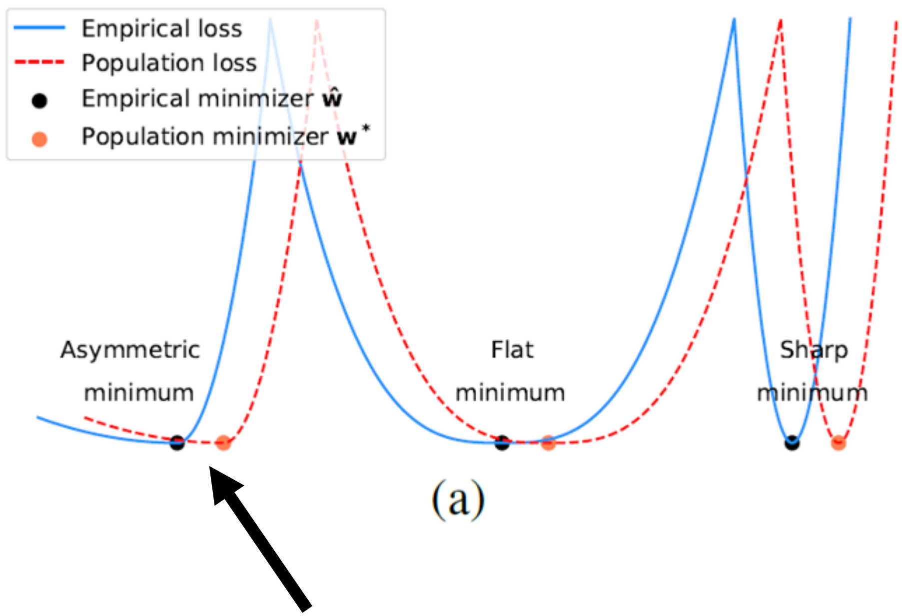
(c) 5e-4, 128, 6.00% (d) 5e-4, 8192, 10.19%

With the Filter-Normalized noise, the small batch training could consistently lead to flatter minima (the left column) under the cases of using or not using weight decay [1].

[1] Hao Li, et al. Visualizing the Loss Landscape of Neural Nets. NeurIPS 2018.

Background

Interestingly, a further work points out that the valley that surrounds the minima is not always symmetric, and it could be ASYMMETRIC.



(a)

How could we plot the asymmetric valleys? [1] presents two insights as follows:

- Batch Normalization (BN) appears to be a major cause for asymmetric valleys.
- Sampling $\epsilon \in [0, 1]$ may have a higher probability of presenting asymmetric valleys, while $\epsilon \in [-1, 1]$ does not.

The valleys are not only flat or sharp, they could also be asymmetric ones which tends to be flat on one side but sharp on the other side [1].

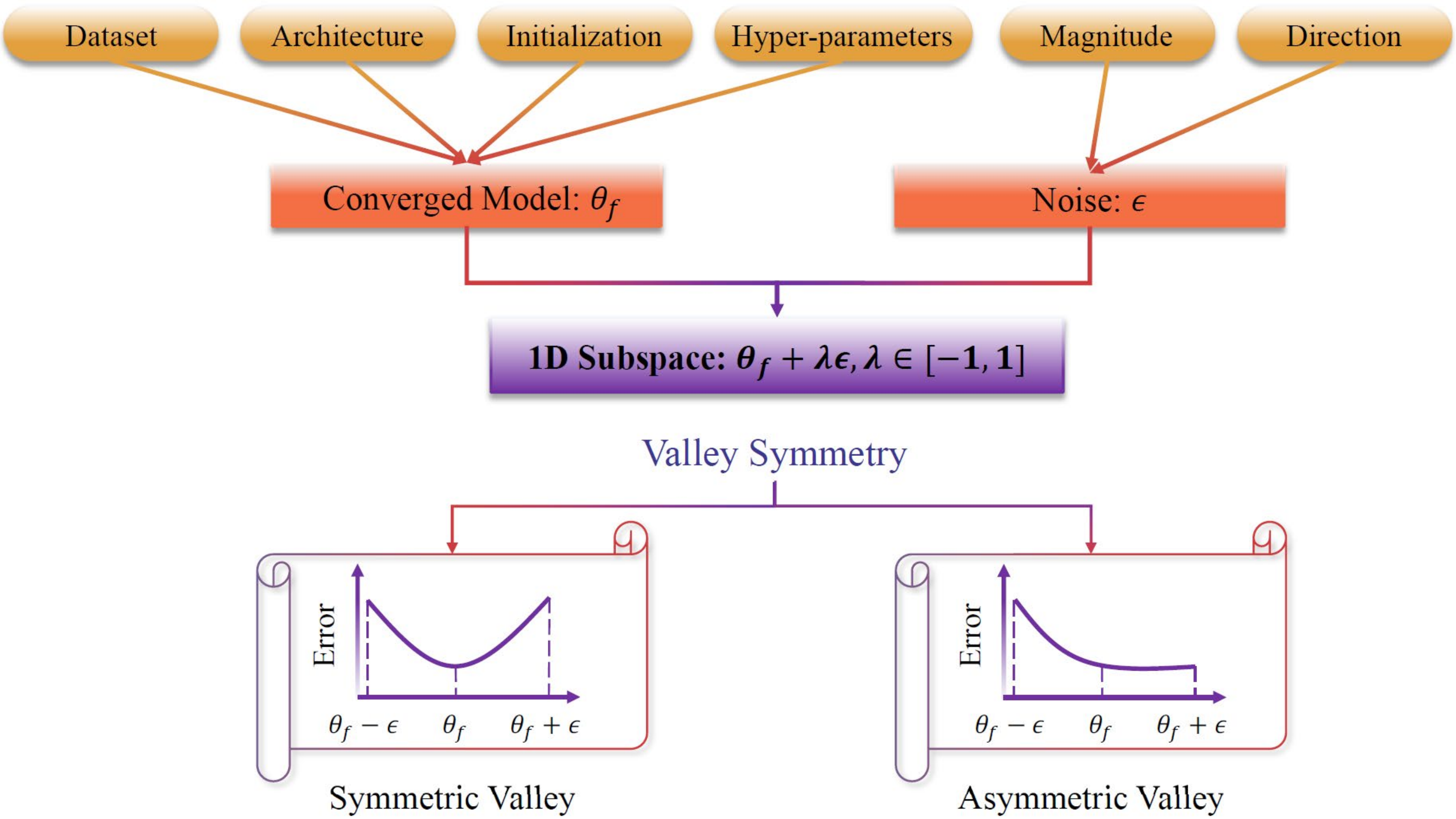
[1] Haowei He, et al. Asymmetric Valleys: Beyond Sharp and Flat Local Minima. NeurIPS 2019.

Content

- Background
- Exploring Asymmetric Valley
- Exploiting Asymmetric Valley
- Conclusion

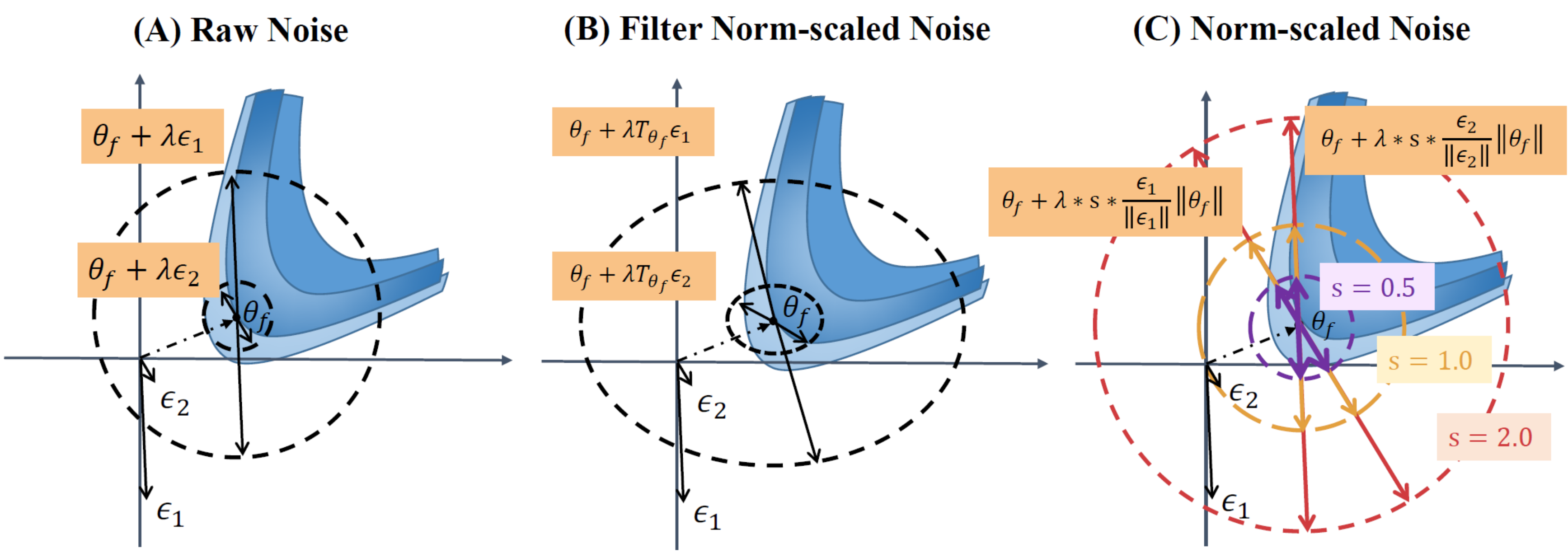
Exploring Asymmetric Valley

Our goal is to analyze the influencing factor of the valley symmetry:



Exploring Asymmetric Valley

Before we show loss landscapes, we have to select an appropriate visualization method:



- Do not consider the parameter scale
- May result in wrong conclusions

- Normalize the noise *filter-wisely*
- But *change the noise direction*

- Normalize the noise *as a whole*
- *Not* change the noise direction

Exploring Asymmetric Valley

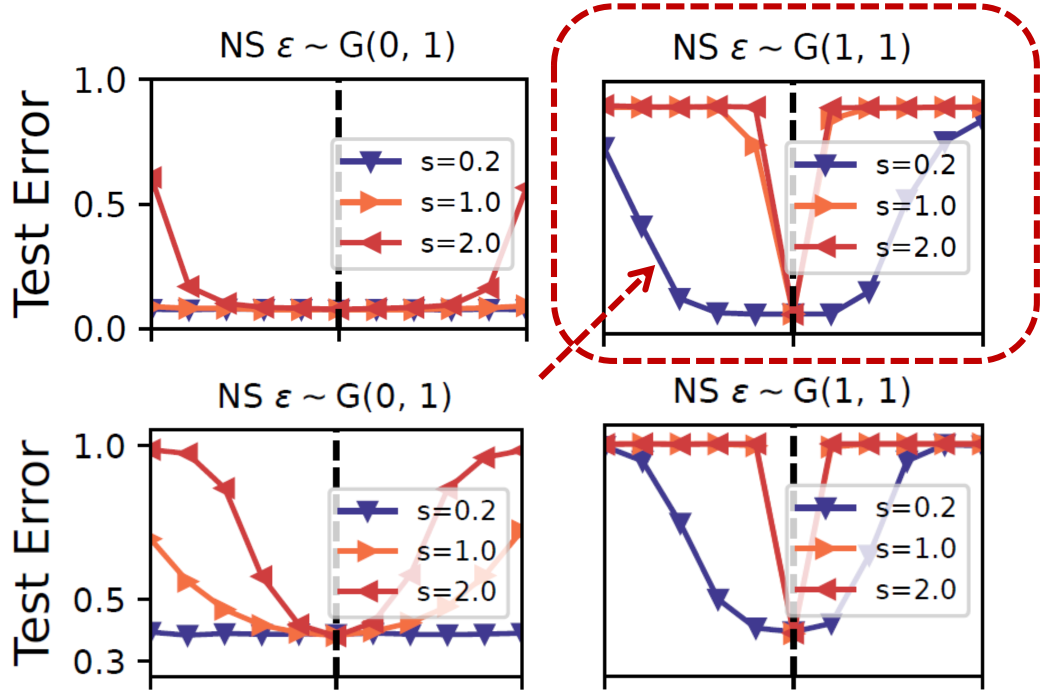
We apply symmetric/asymmetric noise (i.e., around 0 or not) correspondingly to VGG with/without BN. The plotting formula is $\theta_f + \lambda * s * \frac{\epsilon}{||\epsilon||} ||\theta_f||$.

Noise ϵ is symmetric around 0

Noise ϵ is asymmetric around 0

VGG +BN

VGG



Only the case of applying asymmetric noise to VGG with BN shows clear asymmetric 1-D curves.

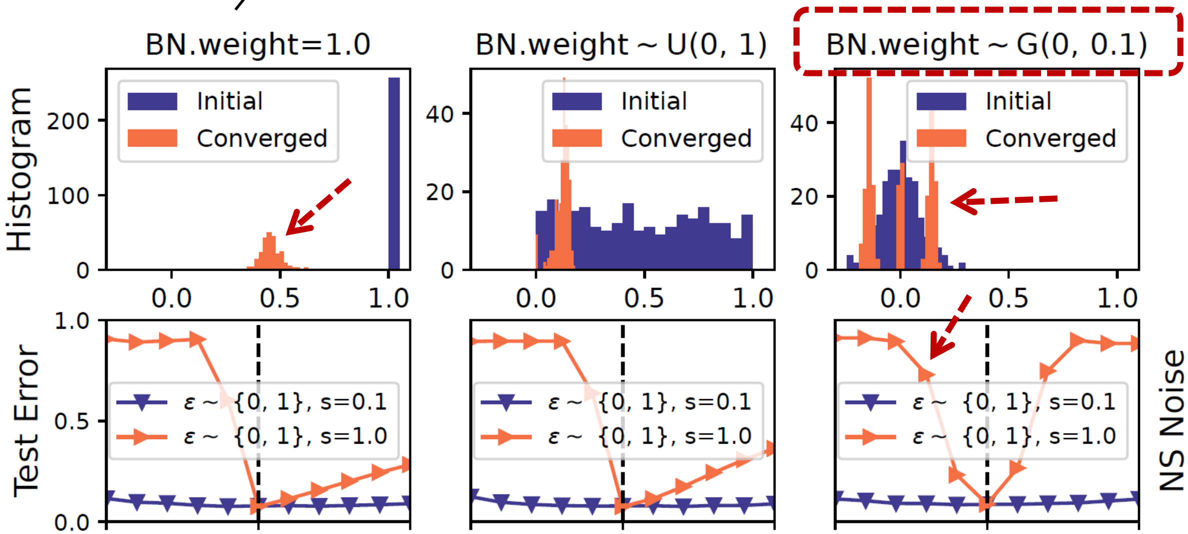
What is the specificity of BN parameters?

Exploring Asymmetric Valley

BN initialization will set the *weight* as *all ones* and the bias all zeros. After convergence, the weight of BN are all positive, which is different from other layers' parameters (which tend to be a Gaussian distribution centered around zero).

$$\mathbf{X} = \mathbf{w}\frac{\mathbf{X}-\mathbf{m_X}}{\sqrt{\mathbf{v_X}+\eta}} + \mathbf{b},$$

BN equation, w denotes BN.weight



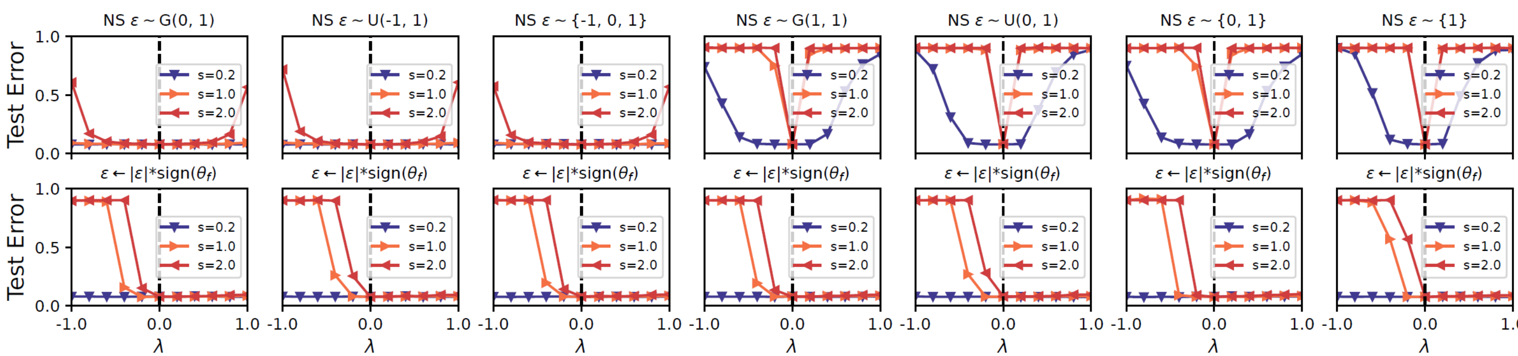
We try to initialize BN.weight by the Gaussian distribution, and the converged BN.weight becomes positive and negative half by half, and **the 1-D loss curves become symmetric**!

The added noise is asymmetric, i.e., $\epsilon \in \{0, 1\}$

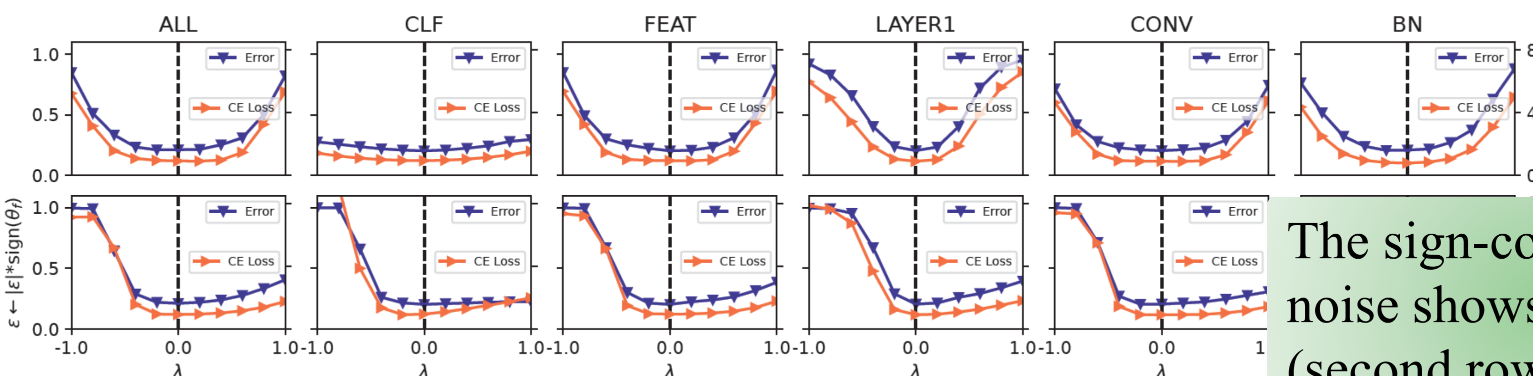
Exploring Asymmetric Valley

A fantastic idea motivates us to *change the sign of the noise*. We set the sign of the added noise to that of the parameters, i.e., $\boldsymbol{\epsilon \leftarrow |\epsilon| * sign(\theta_f)}$.

- Seven types of noise added to VGG16 with BN trained on CIFAR-10



- Gaussian noise added to different layers of ResNeXt101 pre-trained on ImageNet



The sign-consistent noise shows asymmetry (second row)

Exploring Asymmetric Valley

The conclusion: perturbed by the sign-consistent noise, the 1-D loss landscape shows asymmetry.

Explanation from the demo of digits classification of $W^T x$

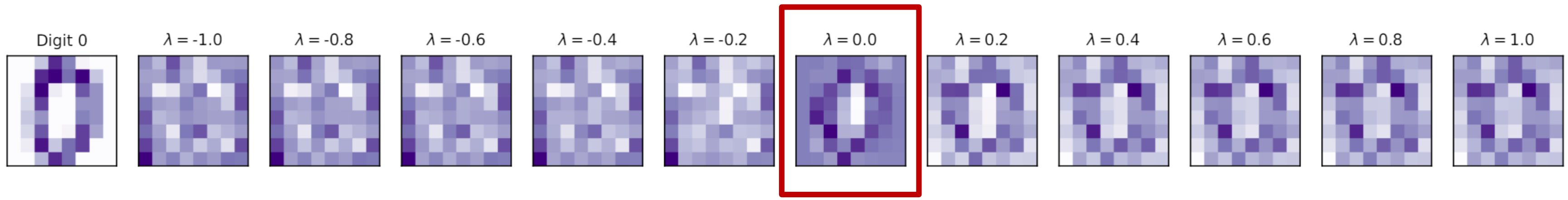


Figure 10: The leftmost shows a digit sample from "sklearn.digits" and others show the pattern of $w + \lambda * \text{sign}(w)$. $\lambda = 0.0$ shows the learned classification weight w.

For the digit 0, the classification weight is $w = W_0$, then the converged w shows a pattern of digit 0 (the red rectangle).

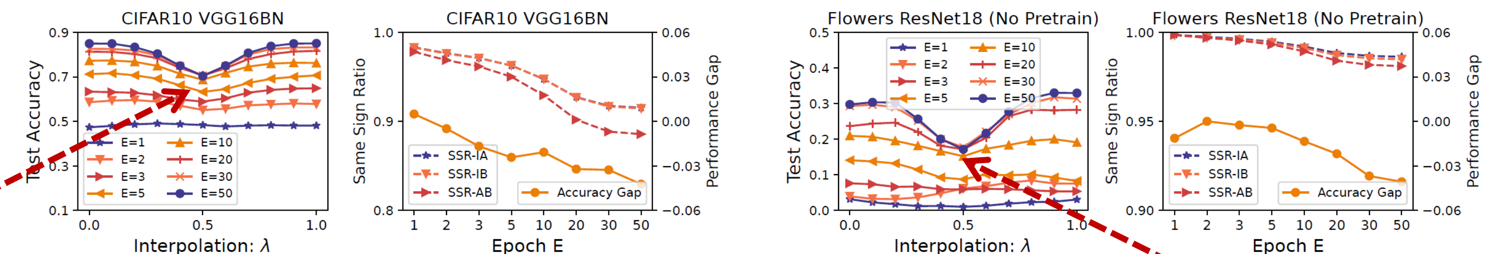
- Adding sign-consistent noise to w, the perturbed w will have a higher probability keeping the pattern of digits 0 (the right of the red rectangle)
- Otherwise, the pattern will diminish quickly (the left of the red rectangle)

Content

- Background
- Exploring Asymmetric Valley
- Exploiting Asymmetric Valley
- Conclusion

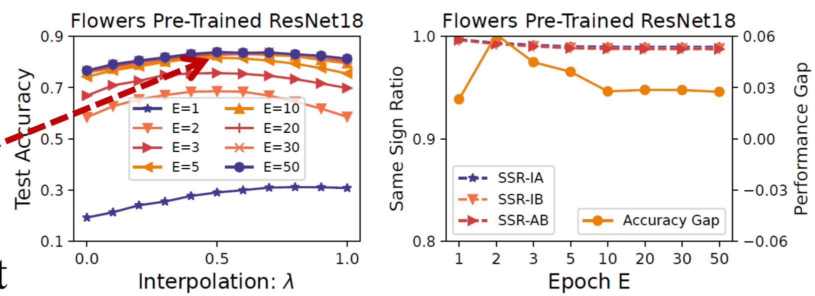
Exploiting Asymmetric Valley

Explaining the success of **Model Soups** which finds that the linear interpolation of models fine-tuned from *the same pre-trained model (Pre-Trained ResNet18)* may lead to a better fused model, while interpolating models fine-tuned from *a random initialized model (VGG16BN, ResNet18)* leads to loss barrier.



Loss Barrier

Loss Barrier



Accuracy Improvement

Fine-tuning from a pre-trained model leads to slight sign change, and the converged two models have a larger sign-consistent ratio, which is beneficial for model fusion.

Exploiting Asymmetric Valley

Restricting the change of parameter sign during the local training procedure in federated learning may benefit model parameter averaging on the server.

$$\mathcal{L}^k = \mathcal{L}^k_{\mathrm{ce}} - \gamma \left(\mathrm{sgp}(\theta_t)\sigma(\theta_t^k) + \mathrm{sgp}(-\theta_t)\sigma(-\theta_t^k) \right),$$

t: the t-th communication round
θ_t: the global model
θ_t^k: the local model on the k-th client

Table 1: Aggregation performance comparisons of FedSign with several popular FL algorithms.

	Dir. α	FedAvg	FedProx	MOON	FedDyn	FedPAN	FedSign
CIFAR-10	10.0	81.53 ± 0.17	81.84	82.44	80.45	81.92	**82.59** ± 0.09
	1.0	80.54 ± 0.11	80.42	80.12	79.78	80.30	**80.76** ± 0.14
	0.5	77.69 ± 0.21	78.12	76.77	78.02	77.78	**78.41** ± 0.35
CINIC-10	10.0	75.74 ± 0.24	76.58	76.25	76.37	76.84	**77.05** ± 0.14
	1.0	72.19 ± 0.15	72.25	72.06	73.12	73.46	**74.59** ± 0.20
	0.5	68.24 ± 0.32	69.11	68.55	69.01	70.14	**70.63** ± 0.16

FedSign pre-aligns the parameter signs during local training, leading to better model aggregation performances on the server.

Content

- Background
- Exploring Asymmetric Valley
- Exploiting Asymmetric Valley
- Conclusion

Exploiting Asymmetric Valley

Contributions:

- Exploring the valley shape under different noise directions that have not been studied yet;
- Proposing that the flat region could be expanded along the direction that has a higher sign consistency with the convergence solution;
- Pointing out the influence of BN and its initialization on valley symmetry;
- Presenting theoretical insights to explain our interesting finding;
- Explaining and inspiring effective algorithms in model fusion.

Sign-consistent noise leads to asymmetric valleys!

Models whose parameter signs are majorly consistent are more likely to fuse!

Thanks !