Generalization in Deep Learning

Thanos Tagaris

Problem Formulation

S → training dataset *(subset sampled from D)*

L → loss function *(what we actually want to minimize)*

Goal: Find set of parameters θ that minimizes the true loss $L_D \rightarrow \min_{\theta} L_D(\theta)$

This is usually achieved through the minimization of the empiric loss $L_S(\theta)$

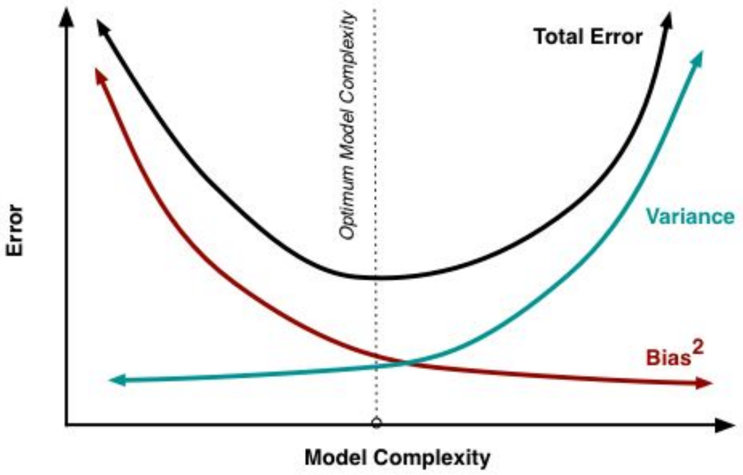
+ [optionally] other quantities we have beneficial effect (L2 norm of params, teacher’s loss, etc.)

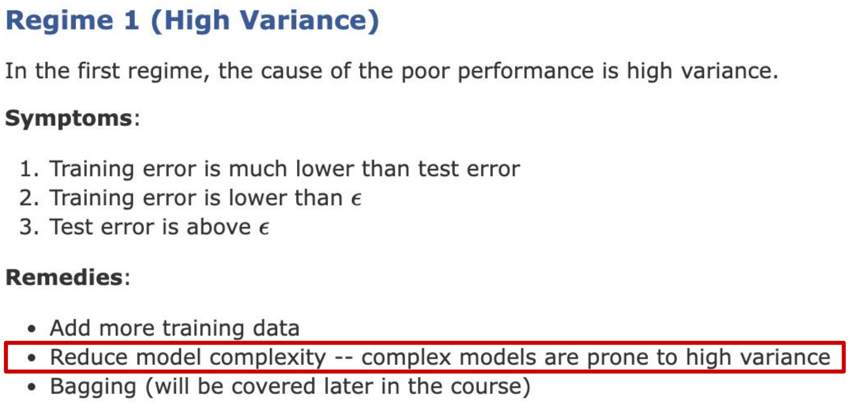
Part 1

Empirical look at generalization in NN

- Bias/Variance tradeoff in the age of Deep Learning
- Global vs local minima
- How does loss “sharpness” affect generalization

Bias-Variance Tradeoff in traditional ML





Regime 1 (High Variance)

In the first regime, the cause of the poor performance is high variance.

Symptoms:

1. Training error is much lower than test error
2. Training error is lower than ϵ
3. Test error is above ϵ

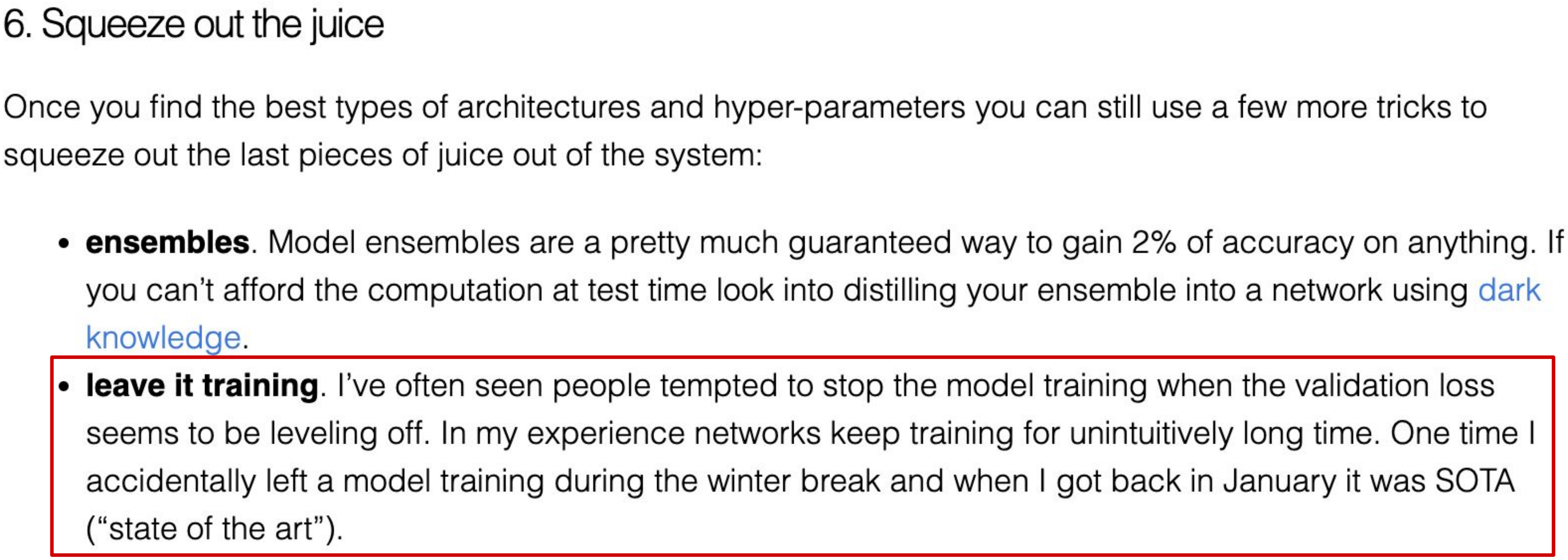
Remedies:

- Add more training data
- Reduce model complexity -- complex models are prone to high variance
- Bagging (will be covered later in the course)

Lecture 12 “Bias-Variance Tradeoff”
CS4780 “Machine Learning for Intelligent Systems”
Cornell University

Not really aligned with Deep Learning (1/3)

Some empirical observations:



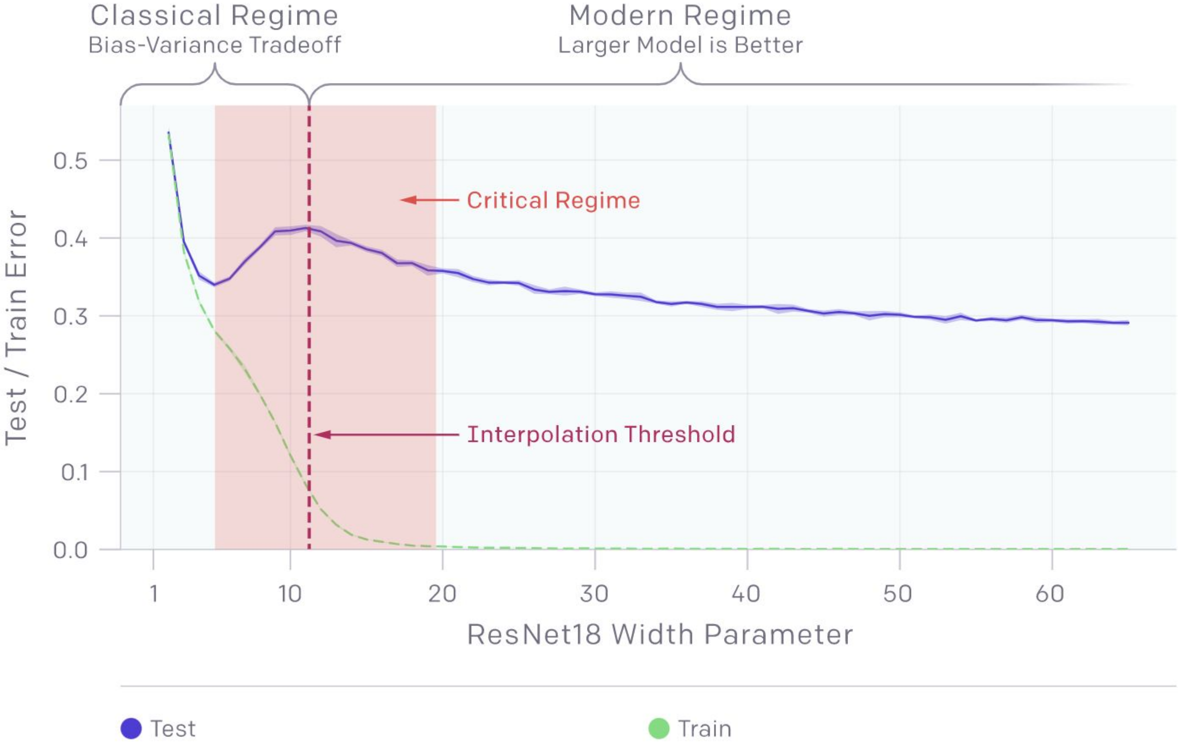
6. Squeeze out the juice

Once you find the best types of architectures and hyper-parameters you can still use a few more tricks to squeeze out the last pieces of juice out of the system:

- **ensembles**. Model ensembles are a pretty much guaranteed way to gain 2% of accuracy on anything. If you can't afford the computation at test time look into distilling your ensemble into a network using dark knowledge.
- **leave it training**. I've often seen people tempted to stop the model training when the validation loss seems to be leveling off. In my experience networks keep training for unintuitively long time. One time I accidentally left a model training during the winter break and when I got back in January it was SOTA ("state of the art").

A Recipe for Training Neural Networks, Andrej Karpathy blog

Not really aligned with Deep Learning (2/3)



Nakkiran, Preetum, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. "Deep double descent: Where bigger models and more data hurt." Journal of Statistical Mechanics: Theory and Experiment 2021, no. 12 (2021): 124003.

Not really aligned with Deep Learning (3/3)

Overparametrized regime: trainable params >> training examples

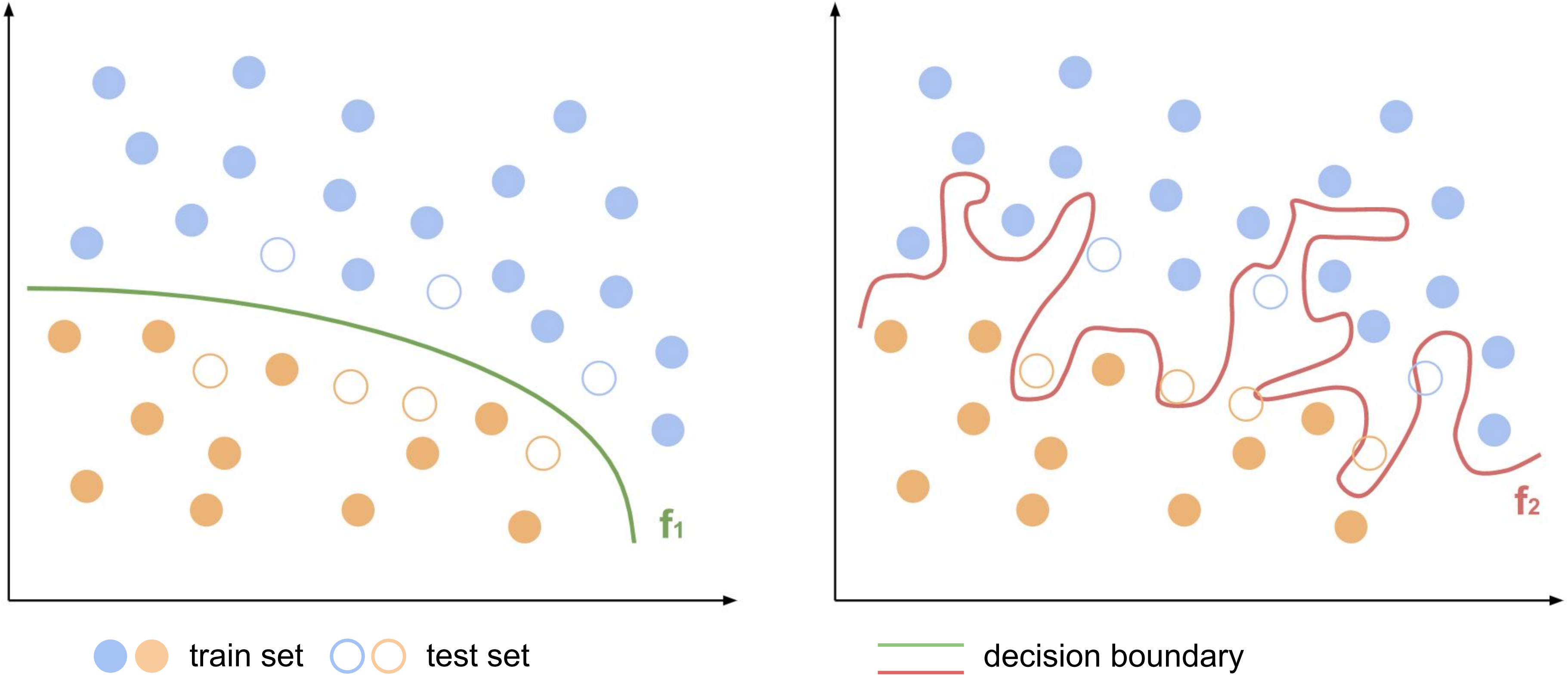
Some more theoretical observations [1, 2]:

- DNNs easily “shatter” the training set (i.e. loss=0), while achieving good validation performance
- DNNs can easily fit random labels → they can overfit on any signal

  … but in practice they don’t!
- Why?

[1] Zhang, Chiyuan, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning requires rethinking generalization (2016)." arXiv preprint arXiv:1611.03530 (2017).
[2] Zhang, Chiyuan, et al. "Understanding deep learning (still) requires rethinking generalization." Communications of the ACM 64.3 (2021): 107-115.

Why do neural networks find good minima?



What do we know empirically about training NNs?

- DNNs often find global minima → training loss = 0
- Not all minima are equal → some generalize better than others
- What property makes some minima better than others?

[1] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang, On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima, arXiv e-prints (2016), arXiv:1609.04836.
[2] Huang, W. Ronny, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. "Understanding generalization through visualizations." (2020): 87-97.

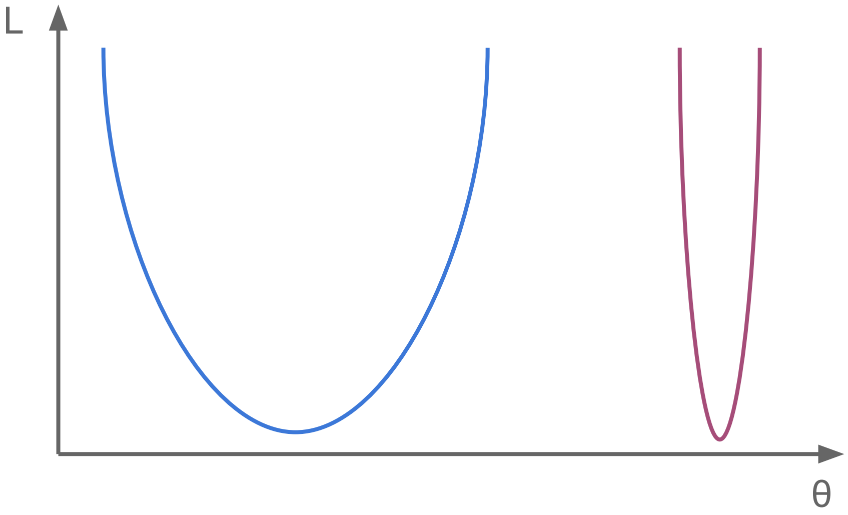
What things **don’t** matter in NN optimization?

- Hypothesis 1: “Optimizers”
  - We can get excellent results without even a GD-based optimizer [1]
- Hypothesis 2: “Implicit regularization of SGD”
  - We can get good results even with a huge batch size [2]
- Hypothesis 3: “Weight decay”
  - We can get good results even with weight bias [3]

[1] Huang, W. Ronny, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. "Understanding generalization through visualizations." (2020): 87-97
[2] De, Soham, Abhay Yadav, David Jacobs, and Tom Goldstein. "Big batch SGD: Automated inference using adaptive batch sizes." *arXiv preprint arXiv:1610.05792* (2016).
[3] Goldblum, Micah, Jonas Geiping, Avi Schwarzschild, Michael Moeller, and Tom Goldstein. "Truth or backpropaganda? An empirical investigation of deep learning theory." *arXiv preprint arXiv:1910.00359* (2019).

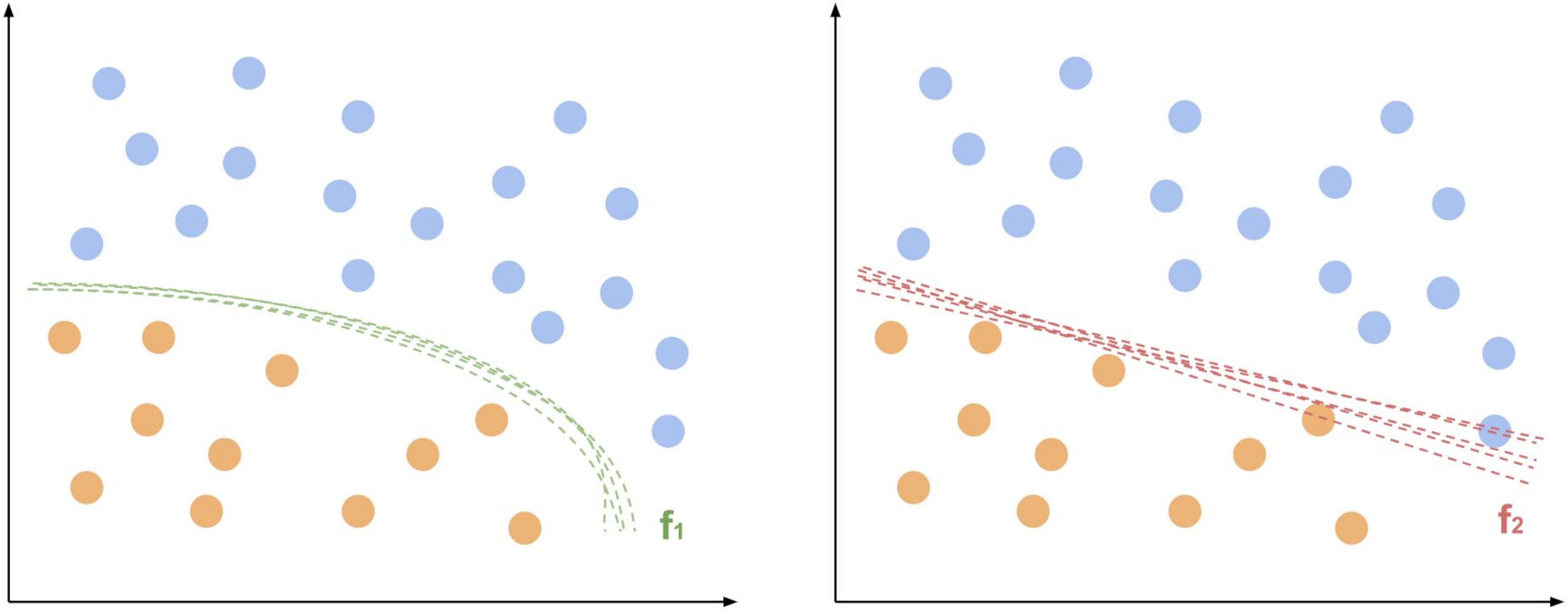
So what does matter?

- Literature claims that “flatness” is a good property for a minima to have
  - i.e a “flat” minimum is better than a “sharp” one
- Flatness is like a “wide-margin” criterion for manifolds



[1] Hochreiter, Sepp, and Jürgen Schmidhuber. "Flat minima." *Neural computation* 9, no. 1 (1997): 1-42.
[2] Keskar, Nitish Shirish, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. "On large-batch training for deep learning: Generalization gap and sharp minima." *arXiv preprint arXiv:1609.04836* (2016).
[3] Dziugaite, Gintare Karolina, and Daniel M. Roy. "Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data." *arXiv preprint arXiv:1703.11008* (2017).
[4] Chaudhari, Pratik, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. "Entropy-sgd: Biasing gradient descent into wide valleys." *Journal of Statistical Mechanics: Theory and Experiment* 2019, no. 12 (2019): 124018.
[5] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan and Samy Bengio, Fantastic Generalization Measures and Where to Find Them, arXiv e-prints (2019), arXiv:1912.02178.
[6] Huang, W. Ronny, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. "Understanding generalization through visualizations." (2020): 87-97

Flatness



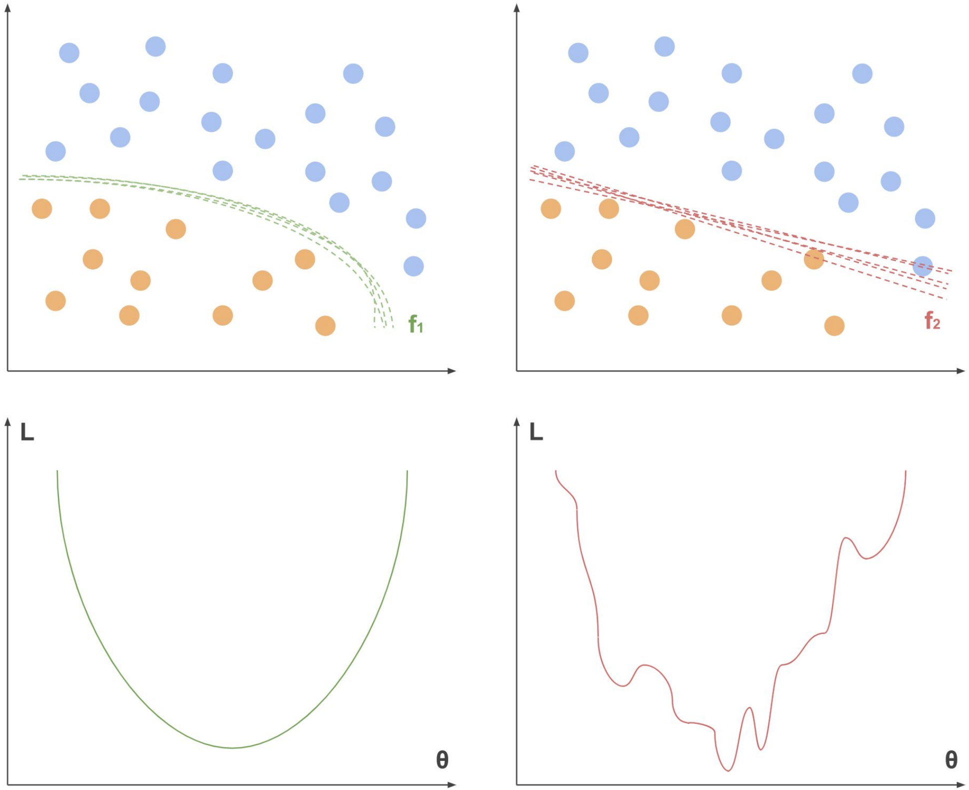
Which of the two functions (f_1 or f_2) is a better fit for the data?

Flatness

What happens to the loss if we make minor perturbations to f_1 and f_2 ?

The loss value for each of the perturbations of f_2 will change more than each perturbation of f_1

If we plot how the loss changes with minor perturbations of the parameters, we can **visualize the *loss landscape*** of each function



Flatness

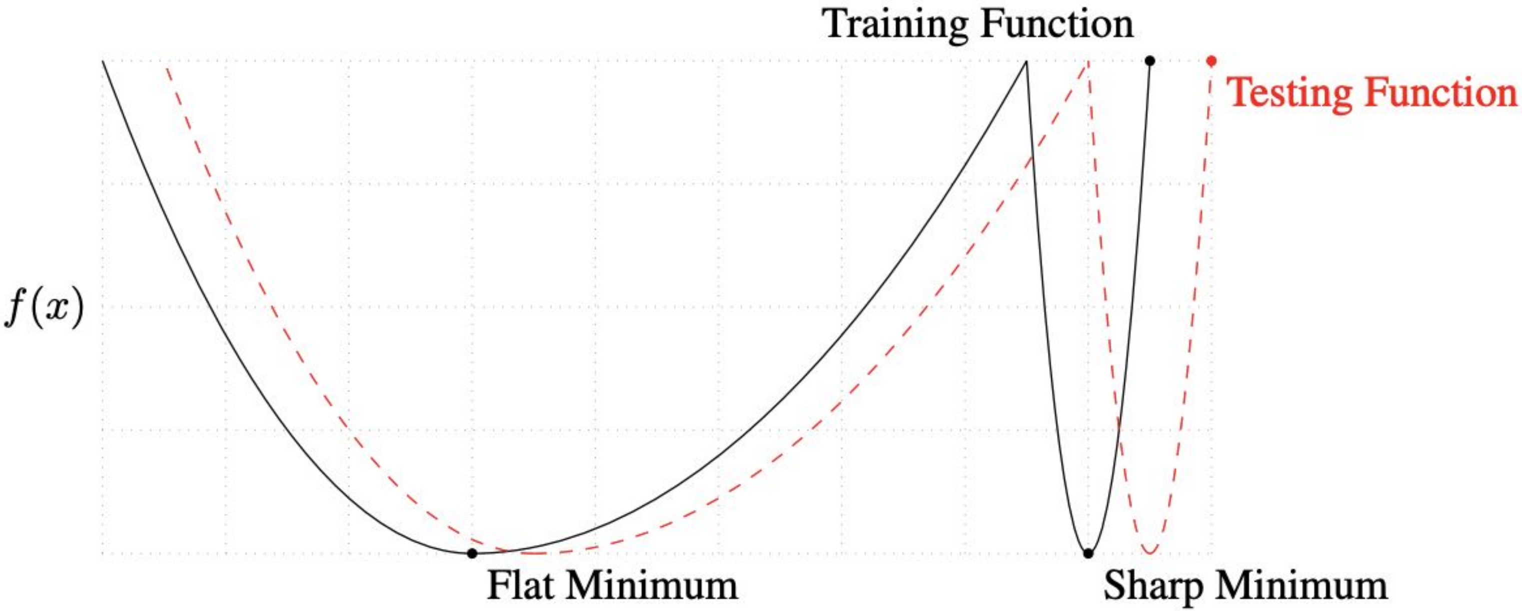


Figure 1: A Conceptual Sketch of Flat and Sharp Minima. The Y-axis indicates value of the loss function and the X-axis the variables (parameters)

Loss Landscape Visualization

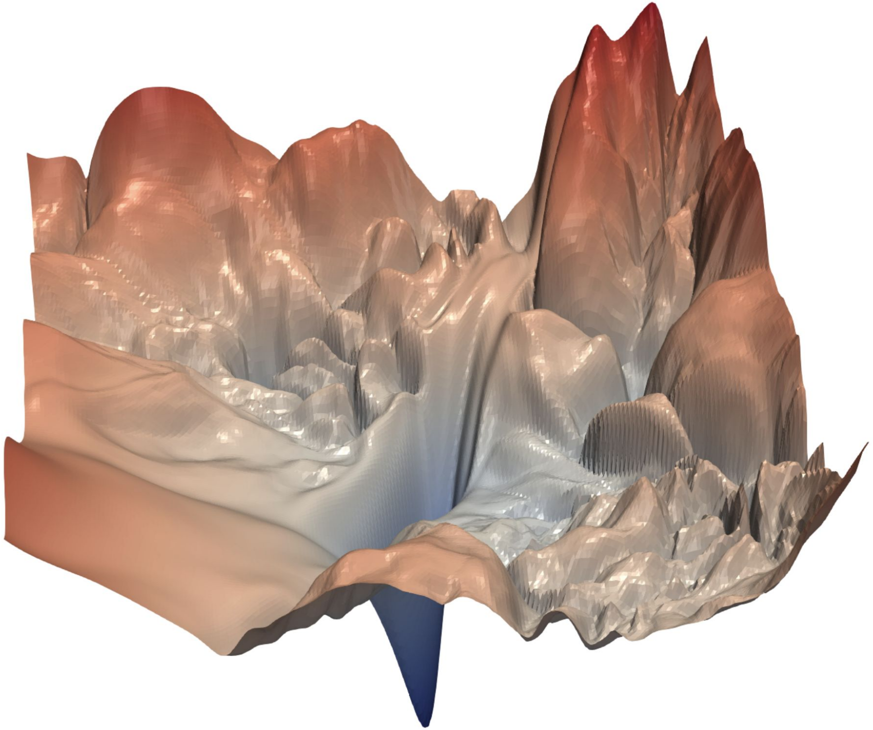
Popular technique for visualizing the minima that a network reached.

Method:

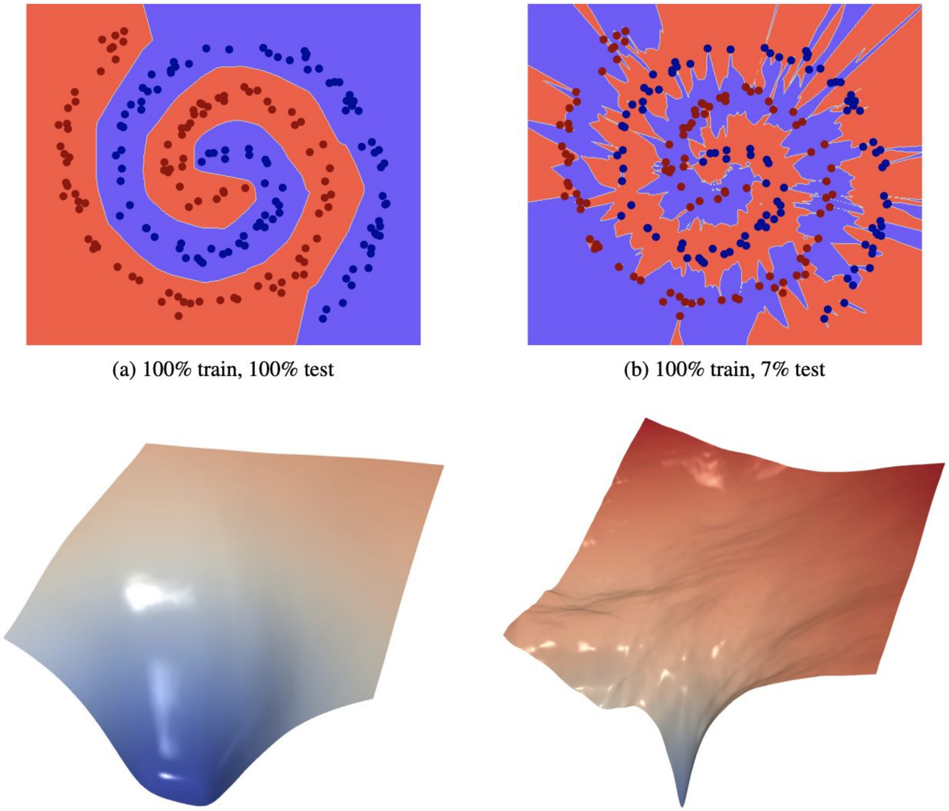
1. Train a network until it finds a minimum
2. Take two random directions in weight space and perform filter normalization
3. Create a grid of points in these dimensions and evaluate loss at each

Li, Hao, Zheng Xu, Gavin Taylor, Christoph Studer, and Tom Goldstein. "Visualizing the loss landscape of neural nets." Advances in neural information processing systems 31 (2018).

Result:



Sharpness in loss landscape



(a) 100% train, 100% test

(b) 100% train, 7% test

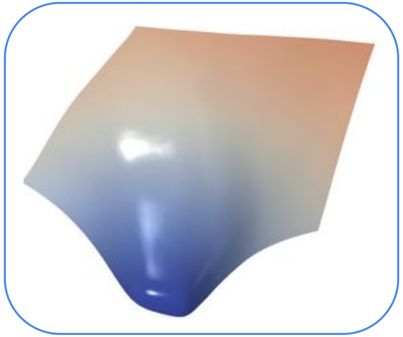
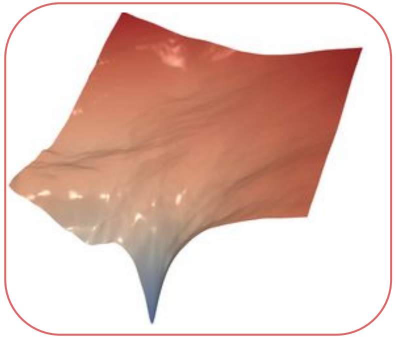
(c) Minimizer of network in (a) above

(d) Minimizer of network in (b) above

Huang, W. Ronny, Zeyad Emam, Micah Goldblum, Liam Fowl, Justin K. Terry, Furong Huang, and Tom Goldstein. "Understanding generalization through visualizations." (2020): 87-97.

So why do NNs work in the first place?

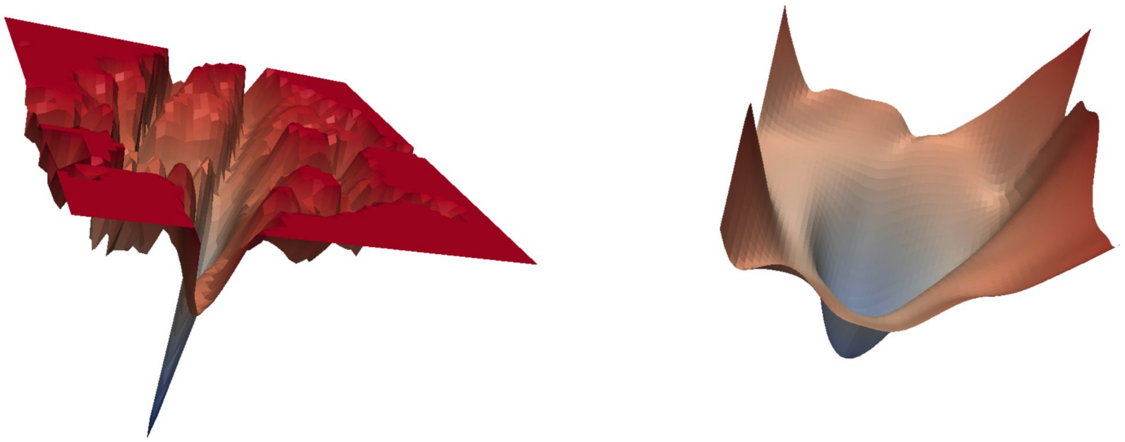
- Why do NNs prefer flat minima over sharp ones?
- Suspicion is that flat minima are easier to find than sharp ones because they are wider (more volume)
- Basically it’s easier to fall into this than this

- This “volume” gets exponentially increased in high dimensional spaces
- Very sharp are like a needle in a haystack

Part 2

Designing an optimizer that actively looks for flat minima



Loss landscape of a ResNet trained with SGD (left) and SAM (right)

What is Sharpness Aware Minimization?

- … an optimization strategy
- Attempts to simultaneously minimize loss and sharpness
- Why all the fuss?
  - Improves generalization → SOTA results
  - More robust to label noise
  - Efficient (captures curvature without 2nd order differentiation)
  - Easy to implement
  - Leads to more interpretable and reproducible solutions
- Presented at ICLR 2021

Kwon, Jungmin, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks." In International Conference on Machine Learning, pp. 5905-5914. PMLR, 2021.

Motivation

- Regular optimization algorithms aim solely at reducing training loss
- A lot of times a global minima is found
- However, *not all minima are created equal (... i.e. some lead to better generalization)*
- “sharpness” of minima has a strong correlation with generalization

→ optimize both on reducing loss and sharpness

Generalization bound

Simplified bound we want to minimize:

$$L_D \leq \max_{||\epsilon||_2 \leq \rho} L_S(w + \epsilon) + \underbrace{\lambda||w||_2^2}_{\text{standard weight decay}}$$

Minimize empiric loss considering sharpness = minimize worst loss in region around w → upper bound on the true loss

What does this have to do with sharpness

- Upper bound on true loss:

$$L_D(w) \leq \max_{||\epsilon||_2 \leq \rho} L_S(w + \epsilon) + \overbrace{\lambda||w||_2^2}^{\text{empiric loss (i.e. standard CE on training set)}}$$

$$L_D(w) \leq \underbrace{\max_{||\epsilon||_2 \leq \rho} L_S(w + \epsilon) - L_S(w)}_{\text{sharpness term}} + L_S(w) + \underbrace{\lambda||w||_2^2}_{\text{L2 regularization}}$$

i.e. what is the largest change in loss in the region around *w*

- Rewrite as a minimization problem:

$$\min_{\boldsymbol{w}} \underbrace{L_S^{SAM}(\boldsymbol{w})} + \lambda||\boldsymbol{w}||_2^2$$

$$L_S^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \leq \rho} L_S(\boldsymbol{w} + \boldsymbol{\epsilon})$$

How to solve efficiently

To optimize we need to:

1. Find the “worst” point in the sphere $L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \triangleq \max_{||\boldsymbol{\epsilon}||_p \le \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon})$

first-order approximation of argmax

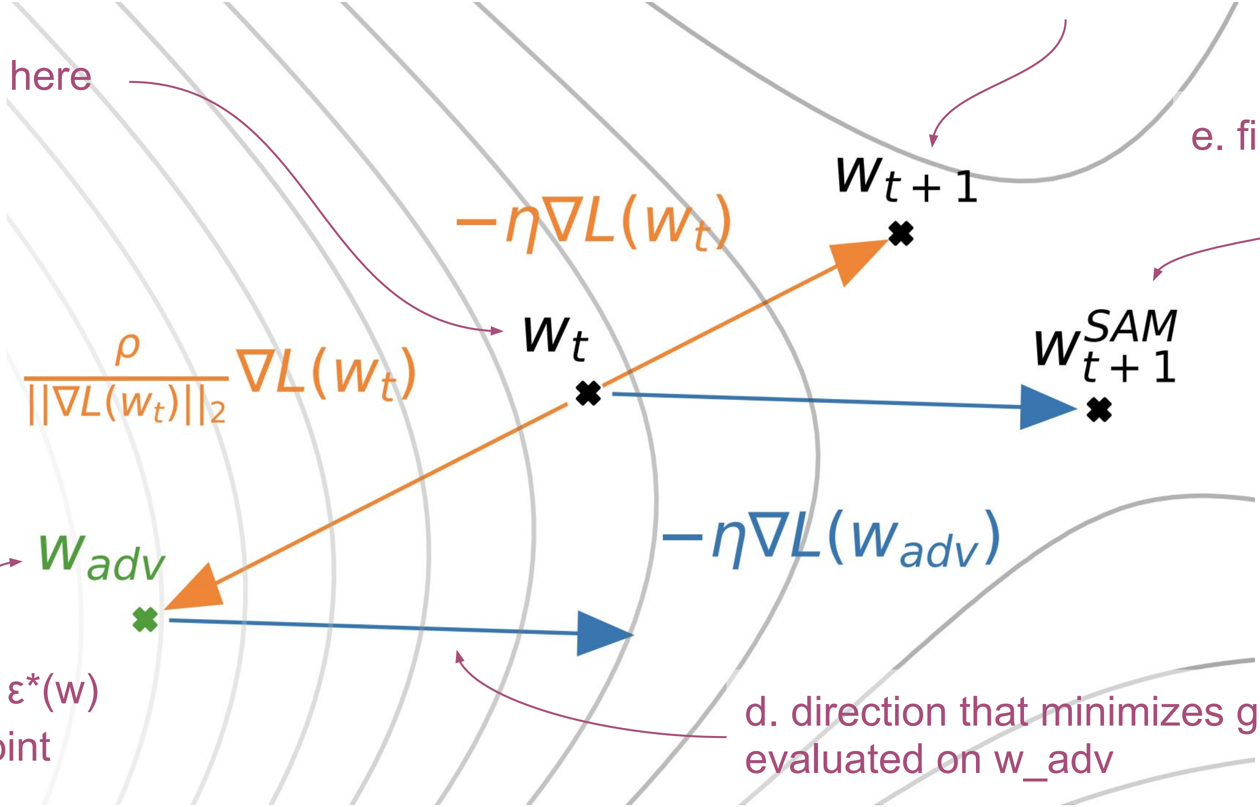
$$\boldsymbol{\epsilon}^*(\boldsymbol{w}) \triangleq \arg\max_{\|\boldsymbol{\epsilon}\|_p \le \rho} L_{\mathcal{S}}(\boldsymbol{w}+\boldsymbol{\epsilon}) \approx \underline{\arg\max_{\|\boldsymbol{\epsilon}\|_p \le \rho} L_{\mathcal{S}}(\boldsymbol{w}) + \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})} = \underbrace{\arg\max_{\|\boldsymbol{\epsilon}\|_p \le \rho} \boldsymbol{\epsilon}^T \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})}_{\text{actually has solution in closed form!}}.$$

2. Minimize SAM objective + weight decay $\min_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) + \lambda||\boldsymbol{w}||_2^2$

$$\nabla_{\boldsymbol{w}} L_{\mathcal{S}}^{SAM}(\boldsymbol{w}) \approx \underbrace{\nabla_w L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}}_{\substack{\text{first-order term} \\ \textit{(gradient of loss evaluated} \\ \textit{at perturbed points)}}} + \underbrace{\cancel{\frac{d\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}{d\boldsymbol{w}} \nabla_{\boldsymbol{w}} L_{\mathcal{S}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}}}_{\substack{\text{second-order term} \\ \textit{(gradient of ε, which itself} \\ \textit{involves a differentiation)}}}$$

will be ignored

What happens in a SAM update



$-\eta \nabla L(w_t)$
w_{t+1}
w_t
$\frac{\rho}{||\nabla L(w_t)||_2} \nabla L(w_t)$
w_{t+1}^{SAM}
w_{adv}
$-\eta \nabla L(w_{adv})$

a. we start here

b. where a regular SGD update would put us here

c. w_adv = w + ε*(w)
“adversarial” point

d. direction that minimizes gradient, evaluated on w_adv

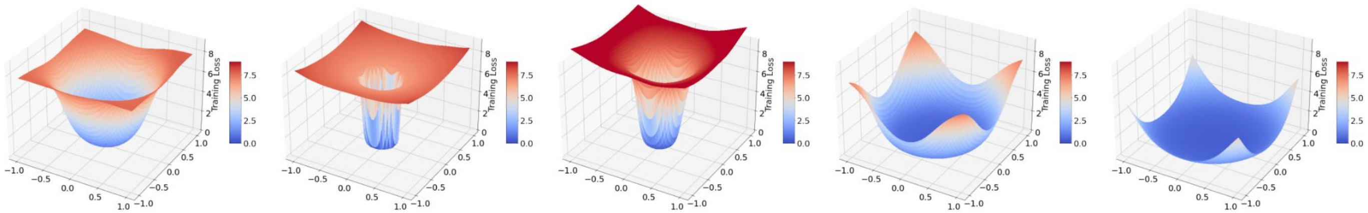
e. final SAM update

Results

- When released, SOTA in cifar10, cifar100 and ImageNet
- Matches SOTA results in label robustness (w/o anything extra - e.g. label noise)
- Still one of the most cost efficient methods
- Has since been successfully applied to many different domains

SAM in practice

- ViT and MLP-Mixer have achieved SOTA results, but with large-scale datasets and strong data augmentations
- ViT and MLP-Mixer have sharper minima and can actually benefit more from SAM than ResNets



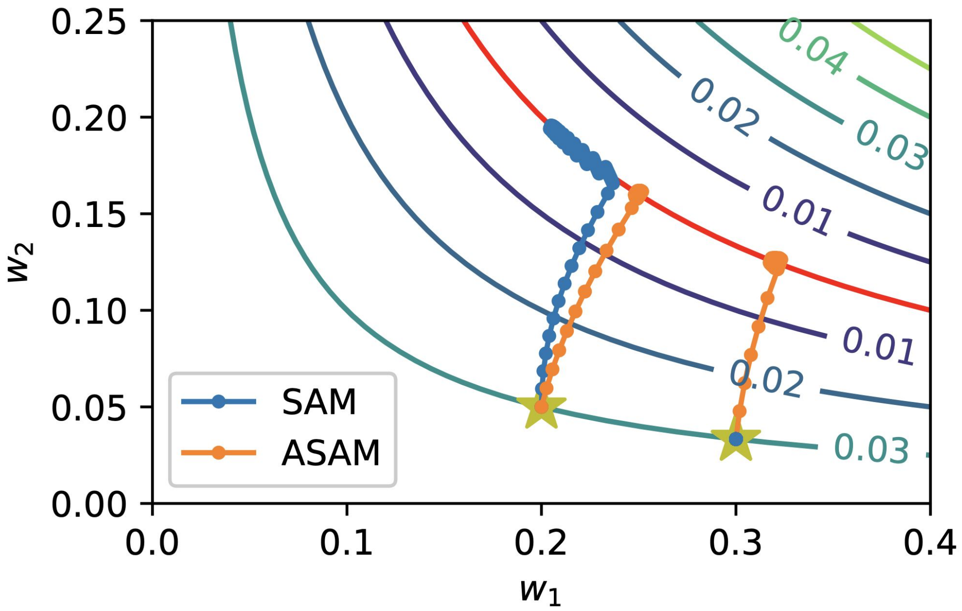
(a) ResNet (b) ViT (c) Mixer (d) ViT-SAM (e) Mixer-SAM

- Massively improves accuracy (>5%) and robustness (>11%)

Chen, Xiangning, Cho-Jui Hsieh, and Boqing Gong. "When vision transformers outperform ResNets without pre-training or strong data augmentations." arXiv preprint arXiv:2106.01548 (2021).

ASAM: Adaptive Sharpness-Aware Minimization

- Noticed that sharpness defined in a region with a fixed radius is sensitive to parameter re-scaling
- Propose notion of “adaptive sharpness”, which is scale-invariant
- Further improves SAM’s performance
- Presented at ICML 2021



Kwon, Jungmin, Jeongseop Kim, Hyunseo Park, and In Kwon Choi. "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks." In International Conference on Machine Learning, pp. 5905-5914. PMLR, 2021.

Thank you :)