Ch 5: Advanced Optimization Algorithms for AI

Part II: Sharpness-Aware Minimization (SAM)

Dong-Young Lim
UNIST

AI51101, IE55101

Motivation: Beyond Adam

- Adam is a very popular and powerful optimizer because it is fast and stable. For a long time, it was the standard choice for training deep networks.
- However, researchers started to notice a common problem. Models trained with Adam often had **worse test performance** than models trained with simple SGD.
- This raised an important question for the field of optimization:

Why do some optimizers generalize better than others?

What Affects Generalization?

- Empirical studies have found many factors that influence generalization:
  - Model capacity and overparameterization
  - Regularization and noise injection (like Dropout)
  - Batch size and learning rate schedules
  - The "implicit bias" of the optimization algorithm
- Among them, one concept seemed to connect many of these observations:

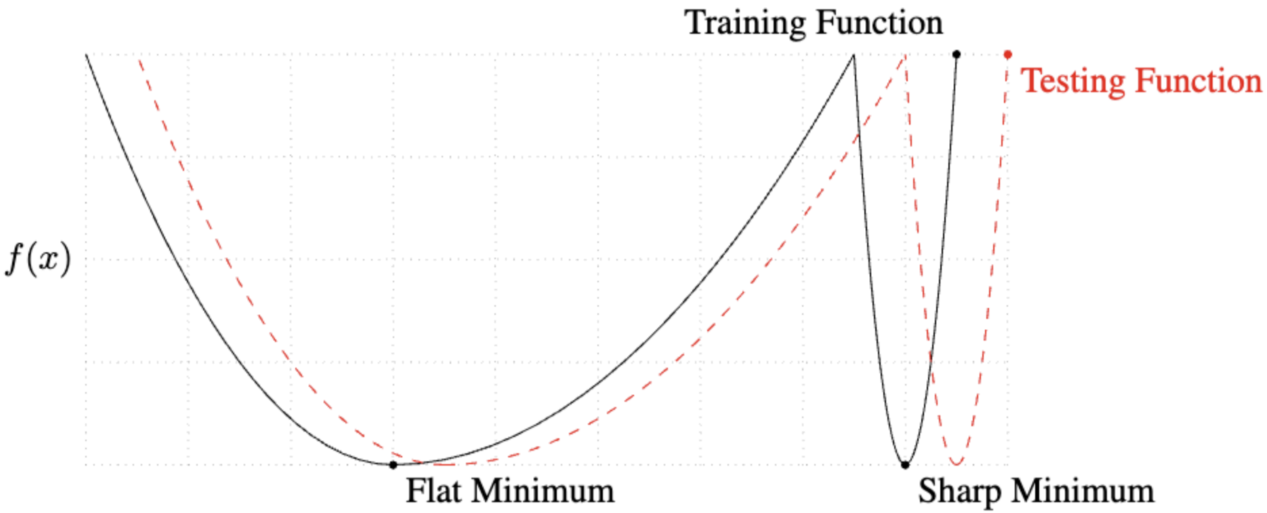
  Sharpness of the loss landscape.

- Intuitively, sharpness measures how "fragile" a solution is. A solution in a sharp minimum is very sensitive to small changes in the parameters, while a solution in a flat minimum is robust.

An Old Idea Revisited: Flat Minima

- The idea that flat minima lead to better generalization is not new. It was proposed as early as 1997 in the paper "Flat Minima" (Hochreiter & Schmidhuber, 1997).
- **Main Idea:**
  - They argued that the training set is just a small sample of the true data distribution. The loss landscape of the training set is therefore a noisy version of the true loss landscape.
  - A sharp minimum on the training set might not be a minimum at all on the true landscape.
  - A flat minimum, however, represents a large region of low error, making it much more likely to be a low-error region on the true landscape as well.

An Old Idea Revisited: Flat Minima

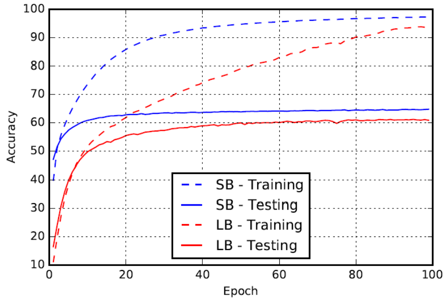


The Sharpness Hypothesis in Deep Learning

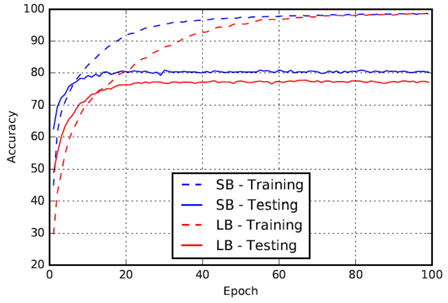
- The generalization gap between Adam and SGD made researchers look again at the old "flat minima" idea from 1997.
- The "Sharpness Hypothesis" returned as the main explanation for this gap.
- The hypothesis states that the geometry of a minimum is the key to generalization in modern deep networks.
- Starting from 2017, a series of important papers provided the strong empirical evidence for this hypothesis. To understand this, we will review three key papers in order:
    1. (2017 ICLR) On Large-Batch Training: Generalization Gap and Sharp Minima
    2. (2017 NeurIPS) Train Longer, Generalize Better
    3. (2020 ICLR) Fantastic Generalization Measures and Where to Find Them

(2017 ICLR) KESKAR ET AL.

- **Core Observation:** Training with a large batch size (e.g., > 1024) leads to a significant drop in test accuracy compared to small-batch training (e.g., 256).
- This "generalization gap" happened even when both methods reached the same low training error.
- Importantly, this was not standard over-fitting. The test accuracy for large-batch models would simply plateau at a lower value, not decrease after peaking.



(a) Network F_2



(b) Network C_1

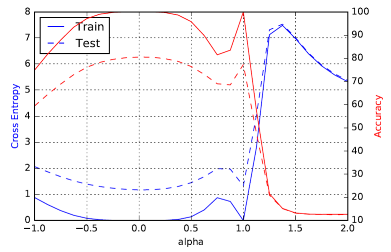
(2017 ICLR) Keskar et al.

- To find the cause, the authors investigated the **geometry** of the solutions.
- **Linear Interpolation**
  - First, find a solution from small-batch training, θ^*_{SB}, and one from large-batch training, θ^*_{LB}.
  - Then, define a line in the parameter space that connects these two points:

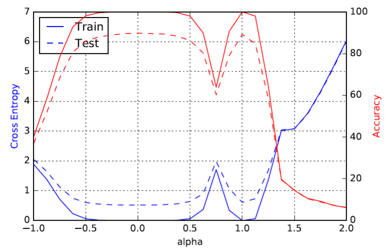
$$\theta(\alpha) = (1-\alpha)\theta^*_{SB} + \alpha\theta^*_{LB}$$

  - They plotted the loss function $f(\theta(\alpha))$ for values of α in a range like $[-1, 2]$. This shows the landscape along the line connecting the two solutions and beyond.

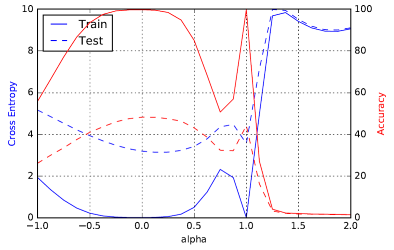
(2017 ICLR) Keskar et al.



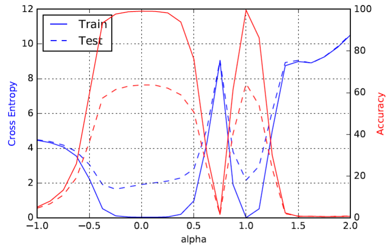
(c) C_1



(d) C_2



(e) C_3



(f) C_4

- The plots consistently showed that the LB solution ($\alpha = 1$) was at the bottom of a much **sharper**, narrower valley, while the SB solution ($\alpha = 0$) was in a wider, flatter basin.

(2017 ICLR) Keskar et al.

- To support the visualizations, they proposed a metric to quantify the sharpness of a minimum.
- The formula is defined as:

$$\text{Sharpness}(\theta^*) = \frac{\max_{\|\epsilon\| \leq \rho} \left(L(\theta^* + \epsilon) - L(\theta^*) \right)}{1 + L(\theta^*)}$$

  The metric measures the highest loss value the function can reach within a small neighborhood around a given solution θ^*.

(2017 ICLR) Keskar et al.

	$\epsilon = 10^{-3}$		$\epsilon = 5 \cdot 10^{-4}$	
	SB	LB	SB	LB
F_1	1.23 ± 0.83	205.14 ± 69.52	0.61 ± 0.27	42.90 ± 17.14
F_2	1.39 ± 0.02	310.64 ± 38.46	0.90 ± 0.05	93.15 ± 6.81
C_1	28.58 ± 3.13	707.23 ± 43.04	7.08 ± 0.88	227.31 ± 23.23
C_2	8.68 ± 1.32	925.32 ± 38.29	2.07 ± 0.86	175.31 ± 18.28
C_3	29.85 ± 5.98	258.75 ± 8.96	8.56 ± 0.99	105.11 ± 13.22
C_4	12.83 ± 3.84	421.84 ± 36.97	4.07 ± 0.87	109.35 ± 16.57

- For all tested networks, the sharpness value for the LB solutions was **10 to 100 times higher** than for the SB solutions.

.

- The high gradient noise in small-batch SGD prevents the optimizer from getting stuck in narrow (sharp) valleys.

- The low noise in large-batch SGD allows it to easily converge to the nearest minimum, which is often sharp.

In short, the randomness in SB-SGD helps it explore the landscape and find better, flatter solutions.

(2017 NeurIPS) Hoffer et al.

- This paper directly challenged the conclusions of Keskar et al.
- **Core Question:** Is the generalization gap really caused by the batch size itself, or is the comparison between SB and LB training unfair?
- **Hypothesis:** The key difference is not the batch size, but the total number of parameter updates. An epoch of SB training involves many more updates than an epoch of LB training.
- **Main Message:** The poor performance of large-batch models is simply a result of under-training.

(2017 NeurIPS) Hoffer et al.

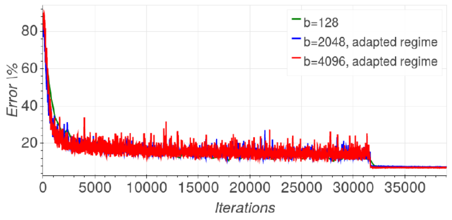
- The authors designed experiments to give LB training a fair chance by matching the total training budget.
- **1. "Adapted Regime":**
  - Their main idea was to ensure the same number of gradient steps for all models.
  - If a large batch is k times bigger than a small batch (e.g., $B_{LB} = k \cdot B_{SB}$), they trained the large-batch model for k times as many epochs.
- **2. Adjusting Hyperparameters:**
  - They also argued that to keep the optimization dynamics similar, the learning rate should be scaled up for larger batches.

$$\eta_{LB} = \eta_{SB} \times \sqrt{k}$$

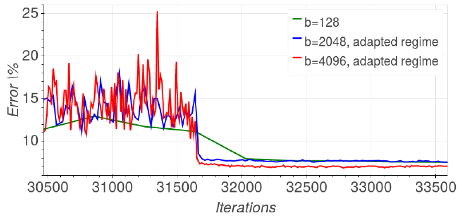
.

(2017 NeurIPS) Hoffer et al.

- **Main Findings:** When the total number of updates and hyperparameters were properly matched, the generalization gap between small-batch and large-batch training **completely disappeared**.
- Large-batch methods were able to achieve the same (or even slightly better) test accuracy as small-batch methods.
- This strongly suggested that there is no fundamental problem with large batches.



(a) Validation error



(b) Validation error - zoomed

(2017 NeurIPS) Hoffer et al.

- **Conclusion:** The generalization gap is not inherent to batch size. The "bad generalization of large batch" is due to the optimization schedule.
- This work reframed the debate. Sharpness is not a fundamental *cause* of bad generalization.
- This is a classic "correlation vs. causation" problem that is very difficult to solve completely.
- This debate created a clear need for a larger, more definitive study to test which properties are truly predictive of generalization, regardless of the training setup.

(2020 ICLR) Jiang et al.

- After the debate (NeurIPS 2017, Hoffer etal), this paper aimed to provide a data-driven answer.
- **Goal:** To test which of the many theories about generalization (norms, margins, sharpness, etc.) actually work in practice.
- **Methodology:**
  - They trained over 10,000 models by changing many different hyperparameters (depth, width, optimizer, etc.).
  - All models were trained to the exact same final training loss for a very fair comparison.
  - They then checked which of 40+ "generalization measures" best correlated with the final test performance.

(2020 ICLR) Jiang et al.

- **Finding 1:** The Failure of Norms
  - Measures based on the size of the weights (like L_2 norm or spectral norm) were very poor predictors of generalization.
  - Sometimes, they were even negatively correlated, meaning larger weights led to better generalization. This contradicts many classical theories.
- **Finding 2:** The Success of Sharpness
  - One category of measures consistently performed the best.
  - Measures related to **sharpness** were the most reliable and accurate predictors of the generalization gap across all 10,000+ experiments.

(2020 ICLR) Jiang et al.

- **Conclusion:** This paper provided strong empirical evidence that the sharpness of a minimum is a fundamental property linked to generalization.
- Even when controlling for training time and final loss, sharpness remains the best predictor of how well a model will perform.
- It is not just a side effect of training; it is a reliable indicator of a good solution.
- **Implication:** This work established that finding flat minima is a valid and important goal for developing better optimization algorithms.

SAM: Motivation

- The previous research (especially ICLR 2020, Jiang et al.) gave us a clear goal: **we should find flat minima**.
- Standard optimizers like SGD or Adam do not directly search for flatness. They only try to minimize the loss at the current point w:
$$\min_{w} L_S(w)$$
- **Main Idea of SAM:** Instead of just minimizing the loss at a single point, let's change the optimization objective to find a point w that has low loss over an entire **neighborhood**. This will force the optimizer to find flat regions.
- Note that we will follow the notations used in the corresponding papers.

SAM: The Objective

- SAM changes the standard loss function $L_S(w)$ to a sharpness-aware loss function, $L_S^{\text{SAM}}(w)$:

$$\min_w L_S^{\text{SAM}}(w) \quad \text{where} \quad L_S^{\text{SAM}}(w) = \max_{\|\epsilon\|_p \leq \rho} L_S(w + \epsilon)$$

- The new objective leads to a min-max problem:
  - **Inner Maximization:** Find the "worst-case" loss within a small neighborhood of radius ρ around the current weights w. This tells us how sharp the current region is.
  - **Outer Minimization:** Update the weights w to minimize that worst-case loss.
- One can interpret it as the worst-case optimization.

SAM: Step 1 (Ascent)

- **Step 1:** The first step is to solve the inner maximization problem:

$$\hat{\epsilon}(w) = \arg \max_{\|\epsilon\|_p \leq \rho} L_S(w + \epsilon)$$

  - Finding the exact solution for this is infeasible.
  - **Approximation:** Use a simple approximation based on a first-order Taylor expansion:

$$L_S(w + \epsilon) \approx L_S(w) + \epsilon^T \nabla_w L_S(w)$$

  - To maximize this, the perturbation ϵ should point in the same direction as the gradient. The solution is a single gradient step of size ρ:

$$\hat{\epsilon}(w) = \rho \frac{\nabla_w L_S(w)}{\|\nabla_w L_S(w)\|_2}$$

SAM: Step 2 (Descent)

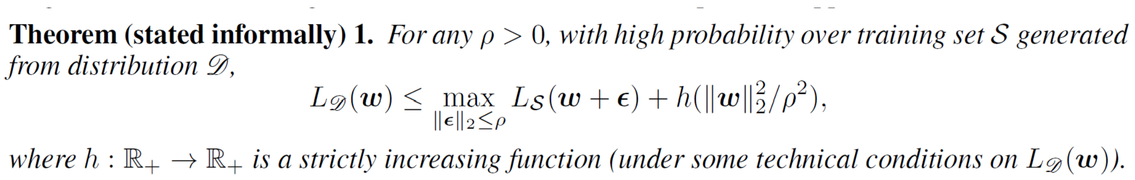
- **Step 2:** After finding the "worst-case" perturbation $\hat{\epsilon}(w)$, we perform the main update.
- Instead of using the original gradient, SAM uses the gradient at the perturbed, "worst-case" point:

$$\nabla_w L_S^{\text{SAM}}(w) \approx \nabla_w L_S(w + \hat{\epsilon}(w))$$

- This leads to a simple two-step update process:
  1. **Ascent Step:** Calculate $\hat{\epsilon}(w)$ by taking a gradient step *uphill*.
  2. **Descent Step:** Calculate the gradient at this new point, $w + \hat{\epsilon}(w)$, and use it to update the weights w *downhill*.
- This means each SAM update requires two gradient computations, making it about twice as slow as a standard optimizer.

SAM: Theoretical Justification

- Why does this min-max objective work? This can be justified with a **PAC-Bayes generalization bound**.



Theorem (stated informally) 1. *For any $\rho > 0$, with high probability over training set $\mathcal{S}$ generated from distribution $\mathscr{D}$,*

$$L_{\mathscr{D}}(\boldsymbol{w}) \leq \max_{\|\boldsymbol{\epsilon}\|_2 \leq \rho} L_{\mathcal{S}}(\boldsymbol{w} + \boldsymbol{\epsilon}) + h(\|\boldsymbol{w}\|_2^2/\rho^2),$$

where $h : \mathbb{R}_+ \to \mathbb{R}_+$ is a strictly increasing function (under some technical conditions on $L_{\mathscr{D}}(\boldsymbol{w})$).

- **Statement:** With high probability, the true test loss ($L_{\mathcal{D}}$) is bounded by the SAM loss on the training set (L_S), plus a complexity term:
  - $L_{\mathcal{D}}(w)$**:** The true test loss.
  - The bound consists of two parts.
    - The first term, $\max_{\|\epsilon\| \leq \rho} L_S(w + \epsilon)$, is **exactly the SAM loss objective**.
    - The second term is a complexity term that depends on the norm of the weights.
- **Implication:** By directly minimizing the SAM loss, the SAM algorithm is directly minimizing the main part of this generalization bound. This provides a theoretical connection between the SAM objective and improved generalization performance.

SAM: Visualization

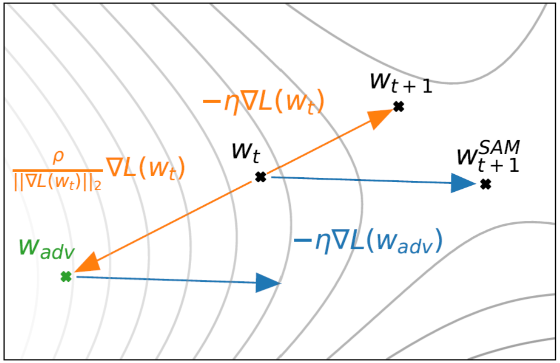
- This two-step process changes how the optimizer moves through the loss landscape.

SGD:

- Directly follows the steepest descent w_{t+1}.
- Can easily fall into the nearest minimum, even if it is sharp.

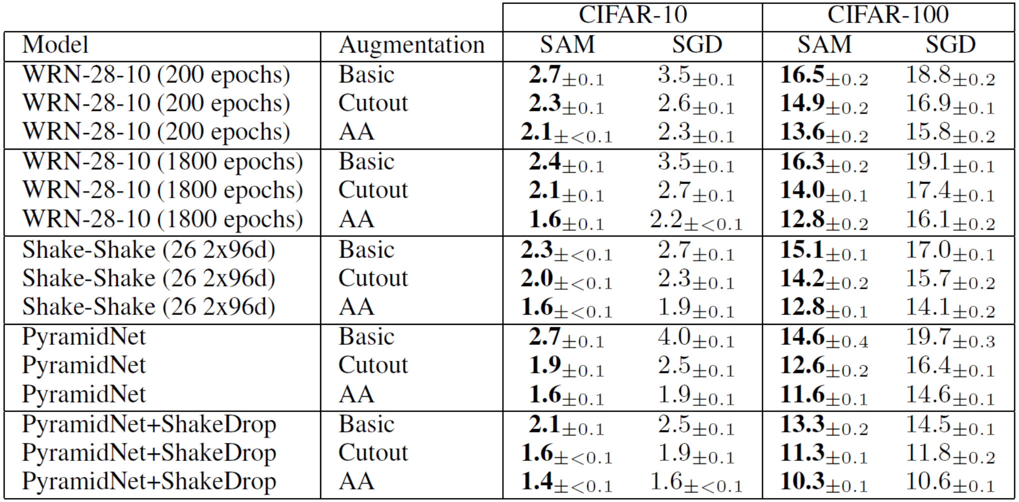
SAM:

- First, it "looks ahead" by stepping uphill to find a high point w_{adv}.
- Then, it pushes that high point down.
- This leads the optimizer away from sharp regions and towards wide, flat valleys w_{t+1}^{SAM}.



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

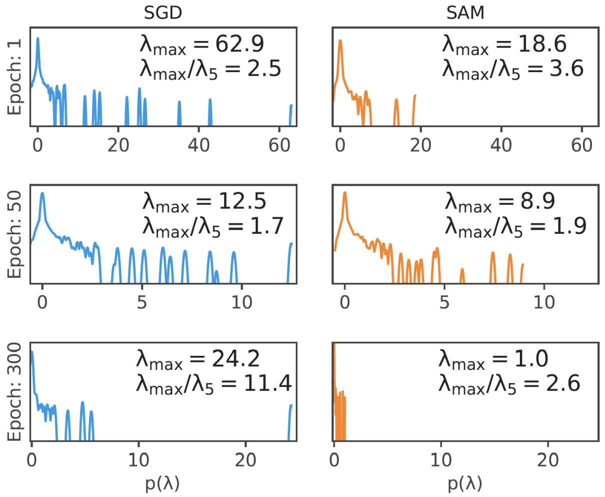
SAM: Empirical Results



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

- SAM showed consistent improvements in generalization across many different models and datasets.

SAM: Empirical Results



- The SGD solution has many large eigenvalues, especially at the end of training ($\lambda_{max} = 24.2$). This indicates a sharp minimum.
- The SAM solution has a much smaller maximum eigenvalue ($\lambda_{max} = 1.0$). This provides strong evidence that SAM successfully finds a much flatter minimum.

SAM: A Deeper Look into SAM's Mechanism

- Three Kinds of Sharpness (2023 ICLR, Wen et al.):
  - **1. Worst-Direction Sharpness ($L_\rho^{\mathbf{Max}}$):**

    $$\max_{\|\epsilon\|\leq\rho} f(w+\epsilon)$$

    This is the "true" sharpness that SAM originally wanted to minimizew, which is related to the **largest eigenvalue (λ_{max})** of the Hessian.
  - **2. Ascent-Direction Sharpness ($L_\rho^{\mathbf{Asc}}$):**

    $$f\left(w+\rho\frac{\nabla f(w)}{\|\nabla f(w)\|_2}\right)$$

    This is the approximation SAM uses for computational efficiency, which is related to the **smallest eigenvalue (λ_{min})** of the Hessian.
  - **3. Average-Direction Sharpness ($L_\rho^{\mathbf{Avg}}$):**

    $$\mathbb{E}_{g\sim N(0,I)}\left[f\left(w+\rho\frac{g}{\|g\|_2}\right)\right]$$

    This is the sharpness measure used in the PAC-Bayes theory. It measures the average loss over random directions. It is related to the **trace** of the Hessian.

SAM: A Deeper Look into SAM's Mechanism

- These three sharpness measures are **not the same**. Minimizing one can lead to a different solution than minimizing another.
- The SAM algorithm uses an approximation (L_ρ^{Asc}) that, in theory, should cause it to minimize the *smallest* eigenvalue λ_{min}.
- However, it turns out that when using **full-batch** gradients, the SAM algorithm *actually* minimizes the **largest eigenvalue** (λ_{max}), which corresponds to the "true" worst-direction sharpness (L_ρ^{Max}).
- More specifically, as training progresses, the gradient of SAM gradually becomes aligned with the top eigendirection of the Hessian, causing the updates to move along the direction that directly reduces λ_{max}.
- On the other hand, stochastic SAM is actually minimizing the trace of the Hessian.

SAM: A Deeper Look into Its Dynamics

- We now analyze the behavior of SAM based on the paper (2023 JMLR, "The dynamics of SAM; bouncing across ravines and drifting towards wide minima").

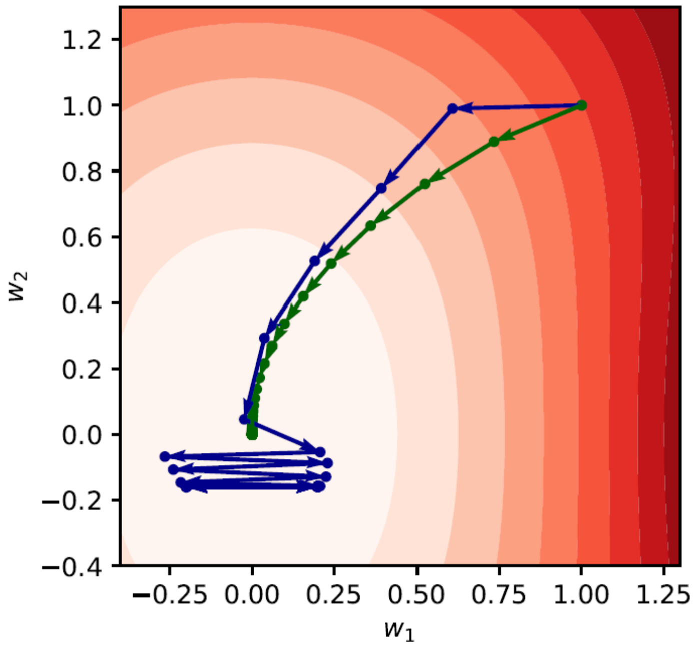
SAM: A Deeper Look into Its Dynamics

- Consider a simple quadratic loss function, $f(w) = \frac{1}{2}w^T Hw$.
- For a standard Gradient Descent (GD) optimizer, the updates would simply converge to the minimum at $w = 0$.
- However, SAM does not converge to $w = 0$. Instead, it converges to a repeating "bouncing" motion. The weights jump back and forth across the minimum.
- **Key Insight:** This oscillation always happens along the direction of the **largest curvature** (i.e., the direction of the top eigenvector of the Hessian, v_{max}). This "bouncing across the ravine" is the fundamental dynamic of SAM near a minimum.

SAM: A Deeper Look into Its Dynamics

- **Main question:** Why does "bouncing" lead to flatter minima?
- Near a minimum, the SAM update decomposes into
  1. **Large component (bouncing):** maintains a two-point oscillation along the top eigendirection v_1 (the sharpest-curvature direction).
  2. **Small component (drifting):** The iterate slowly drifts toward regions with smaller $\lambda_{\max}$ (flatter minima).
- **Mechanism:** The oscillation probes the neighborhood along v_1; the orthogonal drift uses local variation of H in that direction (a "third-derivative" effect) to reduce $\lambda_{\max}$.
- **Net effect:** Bounce for exploration (sharp direction) + drift for $\lambda_{\max}$ reduction $\Rightarrow$ convergence toward wide minima.

SAM: A Deeper Look into Its Dynamics



1.2
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
w_2
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
w_1