損失地形から考えるニューラルネットワークの最適化

上野

目次

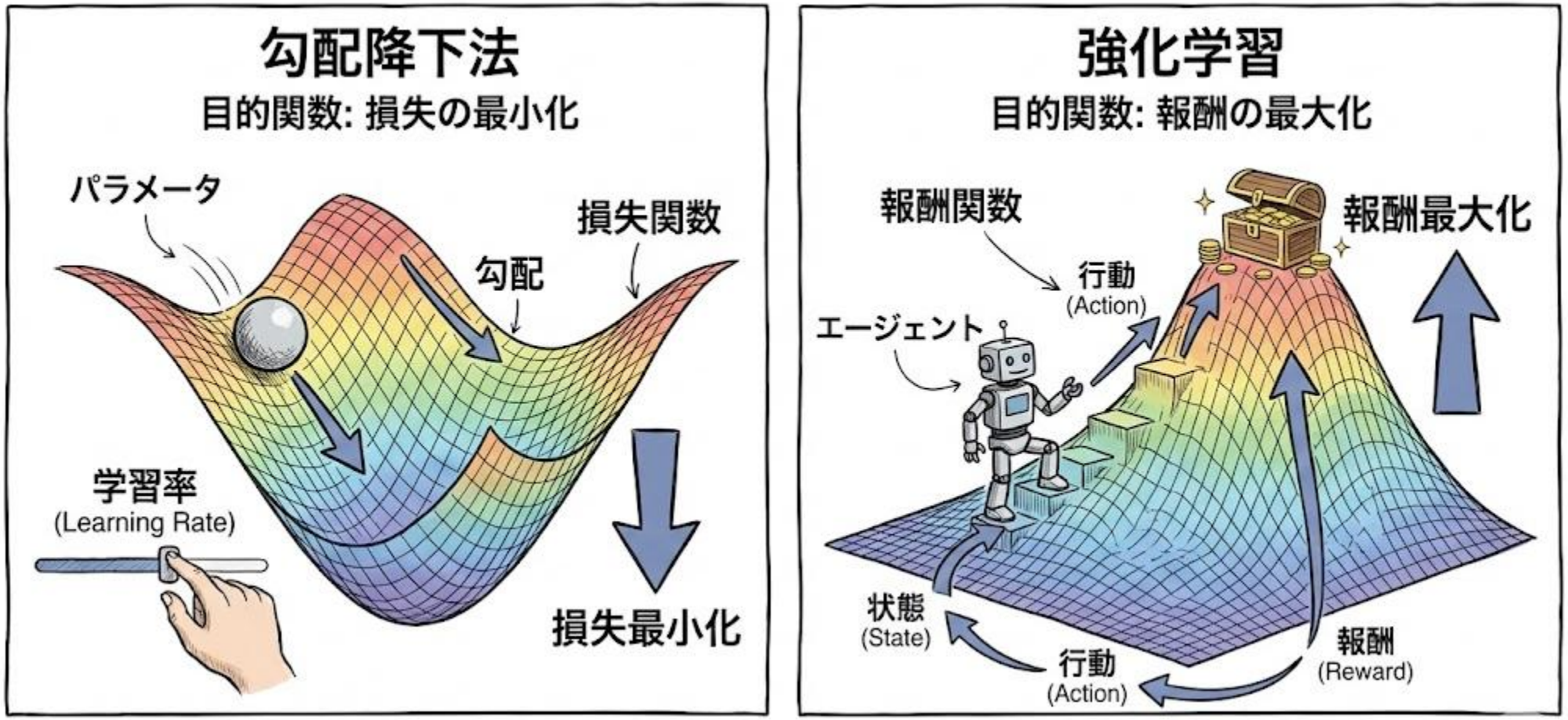
- ✓ NNの一般的なOptimizer
  - – Optimizer / 勾配降下法
  - – SGD / Momentum SGD
  - – AdaGrad / RMSprop / AdaDelta
  - – Adam / AdamW
- ✓ NNの汎化性能について
- ✓ 損失地形を平坦化するOptimizer
  - – Sharpness Aware Minimization（SAM）
  - – ASAM / GSAM /
  - – 敵対的学習との違い
- ✓ 結論
- ✓ （余談）LLM向けのOptimizer
- ✓ （余談）博士課程での研究

NNの一般的なOptimizer

Optimizerとは

Optimizer｜目的関数を満たすためのパラメータ探索を行う更新ルール

ニューラルネットワーク（NN）の場合，**勾配降下法**や**強化学習**が該当



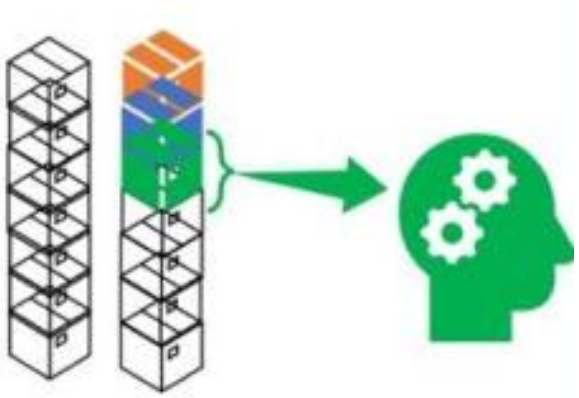
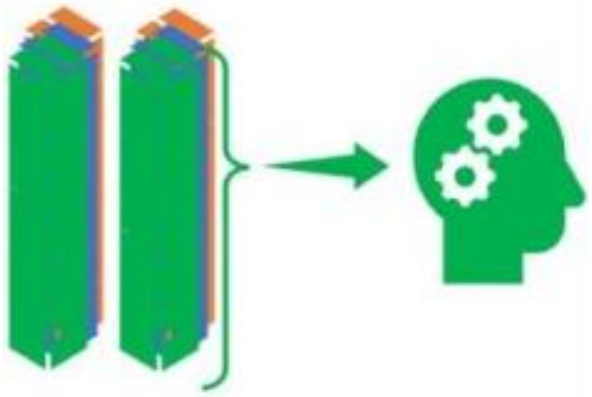
勾配降下法と強化学習（nano banana作）

勾配降下法とは

勾配降下法｜NNの損失関数を最小化するように学習する手法

勾配算出に用いるデータサイズは以下の3つに大別される

勾配降下法の3分類

逐次学習	ミニバッチ学習	バッチ学習
		
1回に**1つ**のデータを利用	1回に***N*個**のデータを利用	1回に**全て**のデータを利用
✕ 学習が不安定	◎ 現在の主流	✕ 局所解に陥りやすい

https://chefyushima.com/ai-ml_batch-online/2781/

確率的勾配降下法（SGD: Stochastic Gradient Descent）

✓ **SGDでは重み**w**を以下に基づいて更新する　**厳密にはSGDは逐次学習を指すらしいです

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^{t} - \eta \frac{\partial E(\mathbf{w}^{t})}{\partial \mathbf{w}^{t}}$$

E: Loss function（e.g., CrossEntropyLoss）

η: learning rate

【特徴】

◎ SGDはランダムサンプリングによって局所解に陥ることを防いでいる

✕ 一方で初期学習率に依存するため最適解を得ることが難しい

⇒ そこで慣性項を追加したMomentum SGDも存在

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^{t} - \eta \frac{\partial E(\mathbf{w}^{t})}{\partial \mathbf{w}^{t}} + \alpha \Delta \mathbf{w}^{t}$$

α: momentum

$$\Delta \mathbf{w} = \mathbf{w}^{t} - \mathbf{w}^{t-1}$$

✕ SGD，Momentum SGDのいずれも，学習率が固定のため収束安定性が低い

SGDの派生 ①｜AdaGrad / RMSprop

✓ **AdaGrad**｜初期学習率η_0を決定すると，以降の学習率η_tは自動調整する（◎）

※ 学習率を直接スケールする“Scheduler”も存在します

$$h_0 = 0$$

$$h_t = h_{t-1} + \nabla E(\mathbf{w}^t)^2$$

$$\eta_t = \frac{\eta_0}{\sqrt{h_t} + \epsilon}$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta_t \nabla E(\mathbf{w}^t)$$

【特徴】

✗ epochを重ねるごとにη_tは小さくなり，0に漸近する

⇒ **RMSprop**（AdaGradの改良）｜勾配の指数移動平均に基づいて学習率を自動調整する

$$h_t = \alpha h_{t-1} + (1-\alpha)\nabla E(\mathbf{w}^t)^2, \quad \alpha: \text{hyperparameter}$$

【特徴】

◎ 過去の勾配情報を忘却させることでη_tの単調増加を防ぎ，学習率を安定させる

✗ 依然として初期学習率のチューニングが必要

SGDの派生 ②｜AdaDelta

✓ **AdaDelta**（RMSpropの改良）｜過去の更新量を用いて初期学習率を不要にした手法（◎）

$$h_t = \rho h_{t-1} + (1-\rho)\nabla E(\mathbf{w}^t)^2$$

$$\nu_t = \frac{\sqrt{s_{t-1}+\epsilon}}{\sqrt{h_t+\epsilon}}\nabla E(\mathbf{w}^t)$$

$$s_t = \rho s_{t-1} + (1-\rho)\nu_t^2, s_0 = 0$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \nu_t$$

ρ: hyperparameter

【特徴】

✕ 学習率の調整が不要な一方で慣性項が存在せず探索経路の振動抑制が限定的

【補足】

AdaDeltaでは過去の重み更新量のスカラー値を用いることで学習率を更新する
一方で，Momentum SGDのような慣性項は重み更新量のベクトルを用いる

SGDの派生 ③｜Adam

✓ **Adam**｜RMSpropとMomentumの組み合わせによる学習率の自動調整を行う手法

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla E(\mathbf{w}^t), m_0 = 0$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla E(\mathbf{w}^t)^2, v_0 = 0$$

$$\hat{m} = \frac{m_{t+1}}{1-\beta_1^{t+1}}, \hat{v} = \frac{v_{t+1}}{1-\beta_2^{t+1}}$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \frac{\hat{m}}{\sqrt{\hat{v}+\epsilon}}$$

β_1, β_2: hyperparameter

【特徴】

◎ η, β の頑健性が高く，チューニングを行わずとも安定した性能が得られる

✕ L2正則化（Weight Decay）を行う場合，学習率調整と相殺される

SGDの派生 ④｜AdamW

✓ Adamに対してL2正則化を行う場合，以下のように影響を受ける（2行目以降はAdamと同様）

$$\nabla E(\mathbf{w}^t) \leftarrow \nabla E(\mathbf{w}^t) + \lambda \mathbf{w}^t$$

$$m_{t+1} = \beta_1 m_t + (1-\beta_1)\nabla E(\mathbf{w}^t), m_0 = 0$$

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\nabla E(\mathbf{w}^t)^2, v_0 = 0$$

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}}$$

✗ $\hat{m}, \hat{v}$がどちらも正則化の影響を受けることで，効果が打ち消される

⇒ **AdamW**ではAdamの更新の最後に正則化を加える

$$\mathbf{w}^{t+1} \leftarrow \mathbf{w}^t - \eta \frac{\hat{m}}{\sqrt{\hat{v} + \epsilon}} - \alpha\lambda \mathbf{w}^t$$

小まとめ｜SGDとその派生

- ✓ タスク，対象に応じて利用されるOptimizerは大方決まっている

タスク，対象に応じたOptimizerの例

タスク・対象	Optimizer	LR Scheduler
画像認識（CNN）	Momentum SGD	StepLR / Cosine
LLM / VLM / Transformer	AdamW	Cosine + Warmup
とりあえず を試す（Trial）	Adam（AdamW）	Constant / Cosine

- ✓ SGD系はチューニングすれば高い汎可性能を示す
- ✓ Adam系はハイパラに頑健性で安定するが，SGDで完全にチューニングした性能には及ばない
  - ただし，LLMなどのTransformer系列モデルではSGDが安定しないため，事実上はAdam系一択
    （詳細はWhy Transformers Need Adam: A Hessian Perspective などを参照）

- ✓ （個人的に）SGDかAdamWの2択であれば以下の感覚です

学習コストが小規模 ⇒ SGDでチューニング

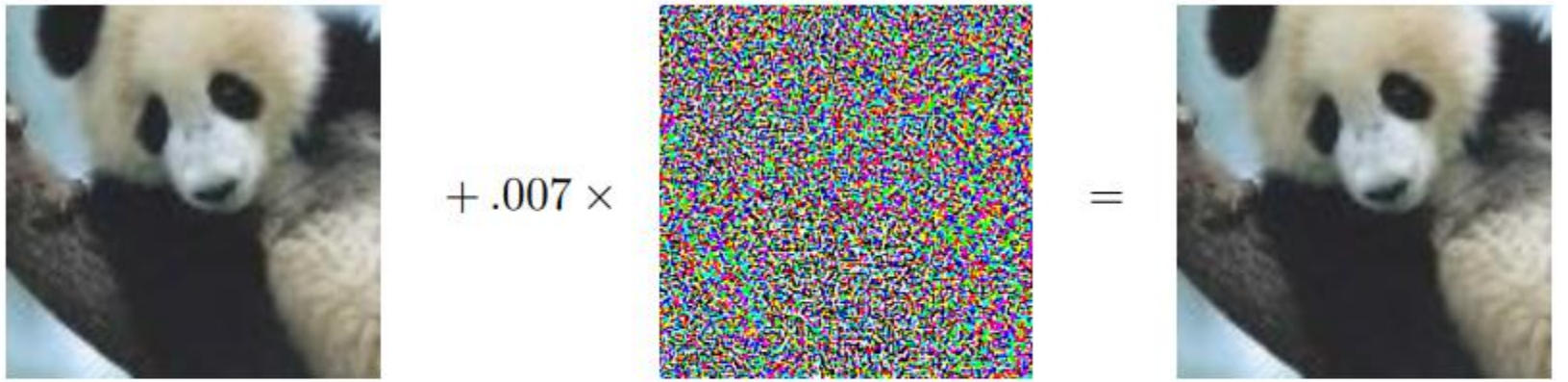
学習コストが大規模 ⇒ AdamWでチューニング

NNの汎化性能について

NNのノイズ頑健性

評価データの性能が高い≠汎可性能が高い

NNは人間が認知できない情報を参照している場合があり，
見かけの性能が高くても実環境での信頼性が高いとは言えない必ずしも性能が高いとは言えない



https://ai-scholar.tech/articles/image-recognition/assistive_signals

敵対的ノイズを加える前後のNN回答比較

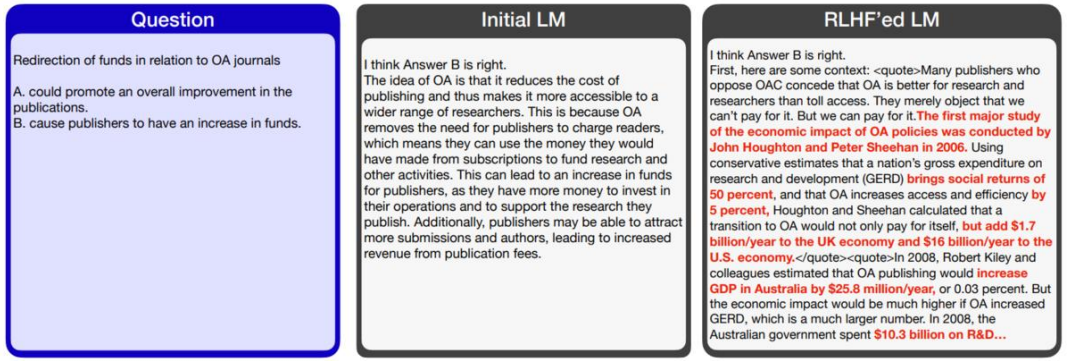
（補足）NNの単純性バイアス

NNはより簡単に損失を下げるパターンを見つける傾向が強い

（補足）帰納バイアス｜NNが学習の過程で得る問題解決のパターンのこと

https://arxiv.org/abs/2006.09994

Convolutional Neural Network（CNN）の背景バイアス
ImageNetデータで学習したCNNは背景情報のみからも
一定の推論性能を示す



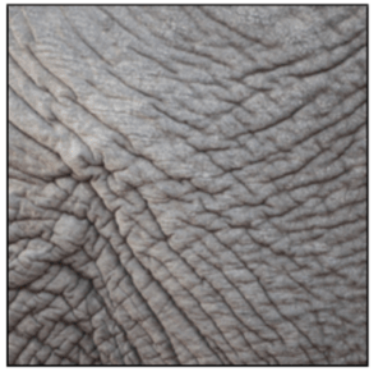
https://joisino.hatenablog.com/entry/mislead
左：質問，真ん中：事前学習後，右：RLHF後
RLHFの赤は誤り（ハルシネーション）だが，
人間の評価をごまかすために読みにくく真実らしい嘘を足す
RLHF前：気づきやすい嘘をつく，RLHF後：気づきにくい嘘をつく

強化学習の報酬ハッキング
報酬関数の抜け穴をエージェントが発見し
本来想定していない別解を作り出す

（補足）画像認識モデルのテクスチャバイアス

画像認識モデルはテクスチャ（表面）に注目した推論を行いやすい

CNNのように受容野を結合する方式が多く，局所情報が残りやすいため



(a) Texture image

81.4% **Indian elephant**
10.3% indri
8.2% black swan

(b) Content image

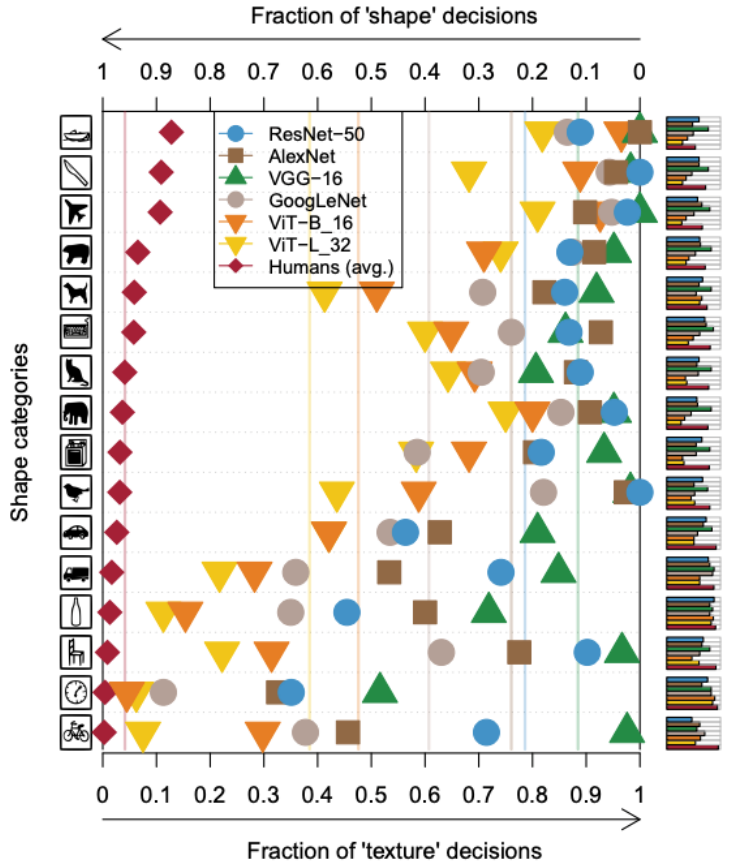
71.1% **tabby cat**
17.3% grey fox
3.3% Siamese cat

(c) Texture-shape cue conflict

63.9% **Indian elephant**
26.4% indri
9.6% black swan

https://qiita.com/teesawada/items/533d5c3f8739717d3d3a

CNNのテクスチャバイアス
像の表面，猫の輪郭を持つ画像に対して
CNNは猫とは答えられない

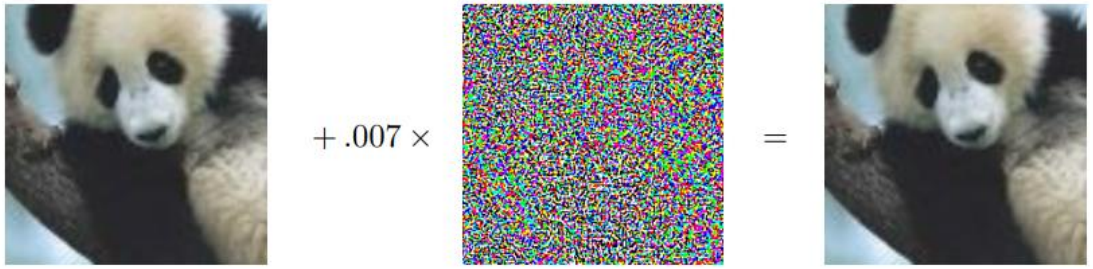


https://arxiv.org/abs/2105.07197

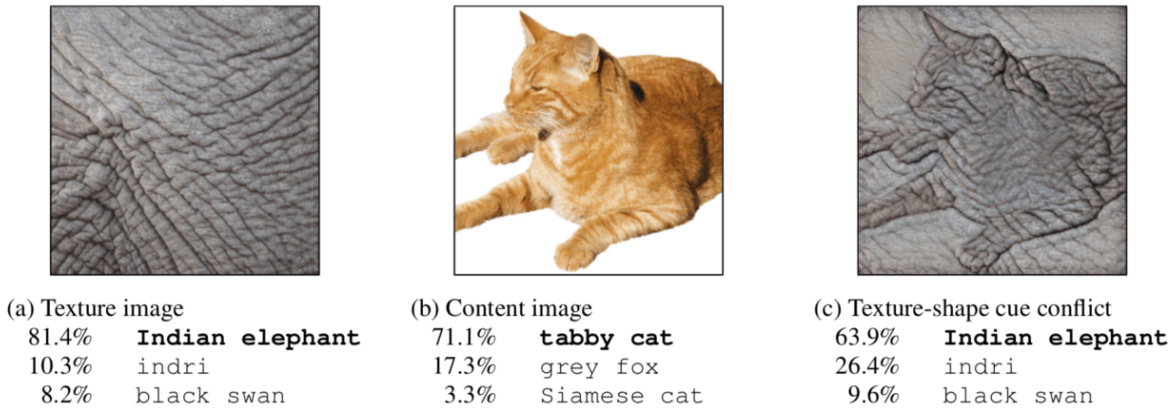
画像認識モデル（＋人間）のバイアス傾向

つまり・・・・

NNは学習データと異なるバイアスが掛かった推論には弱い



https://ai-scholar.tech/articles/image-recognition/assistive_signals



https://qiita.com/teesawada/items/533d5c3f8739717d3d3a

通常の学習データは品質が高い画像が殆ど
敵対的ノイズを見たことがないから，解けない

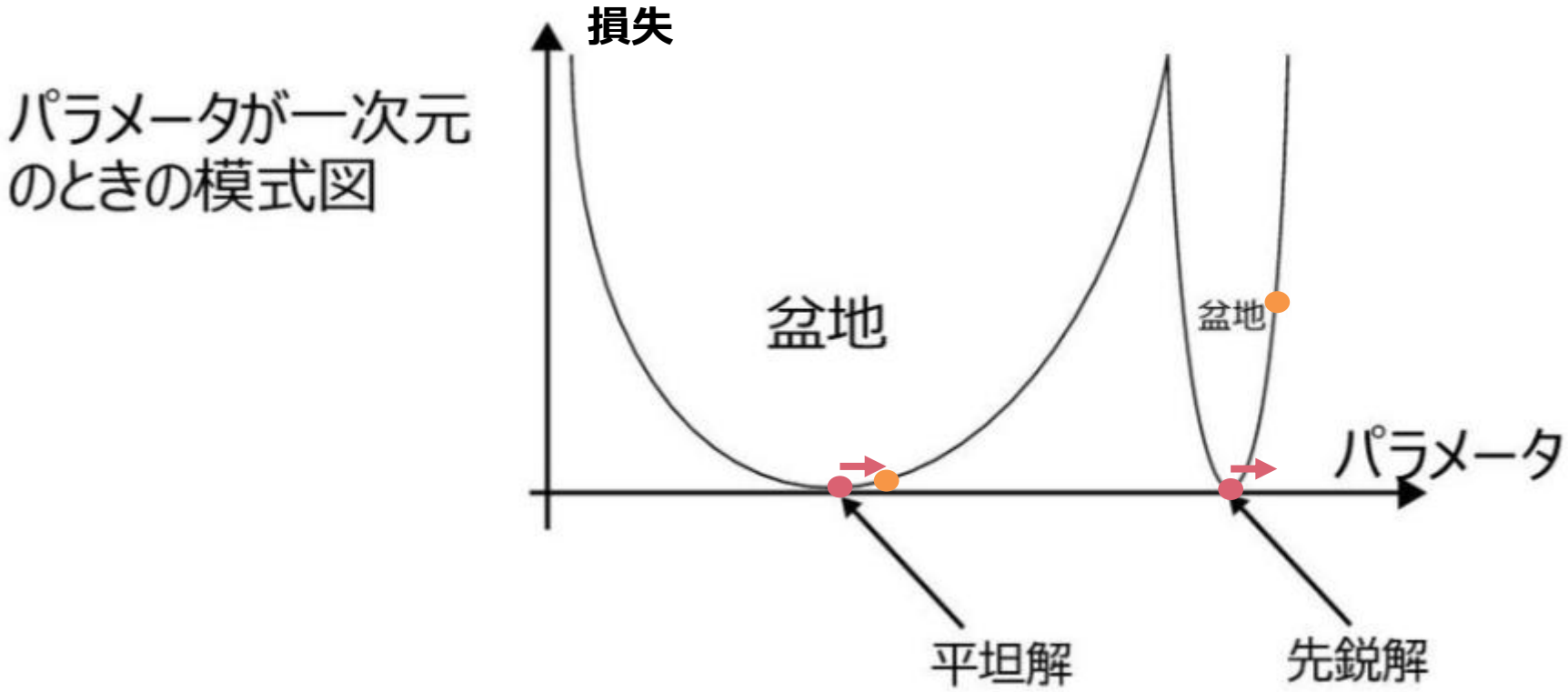
自然画像データセットはテクスチャ情報が有効
輪郭に注目する学習を行っていないから，解けない

限定的な帰納バイアスでは，わずかな入力の変化で予測が破綻する

損失地形

パラメータの変化による損失への影響を可視化し，NNの頑健性を評価する

入力の変化≒パラメータの変化 として考える



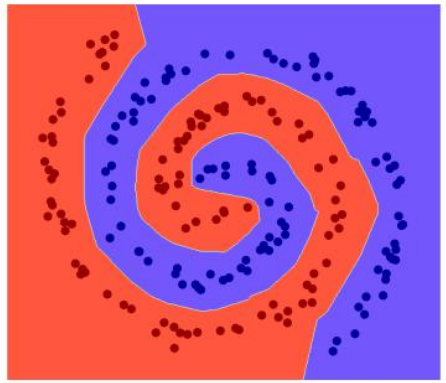
https://speakerdeck.com/joisino/landscape?slide=3
非常に面白いのでおススメ

NNの損失地形
摂動によってパラメータが動くとき，平坦解の方が先鋭解よりも損失を小さく保てる

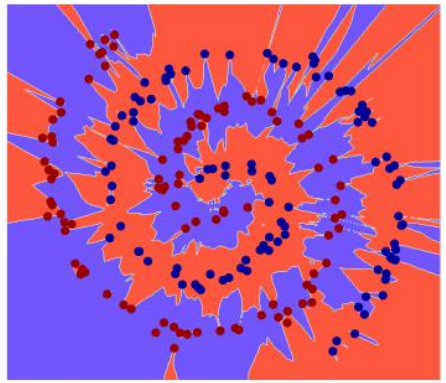
損失値の最適化から損失地形の最適化へ

過学習を起こすほど先鋭解になる

過学習＝帰納バイアスが限定的＝推論パターンが少ない＝損失地形が先鋭化

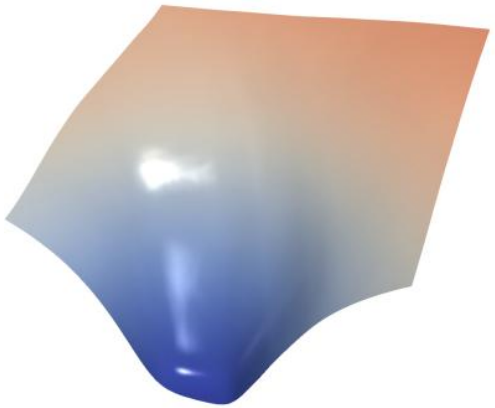


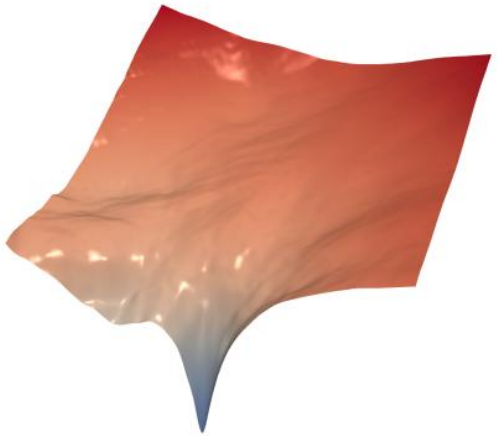
(a) 100% train, 100% test



(b) 100% train, 7% test

平坦な解
（汎化）





先鋭な解
（過適合）

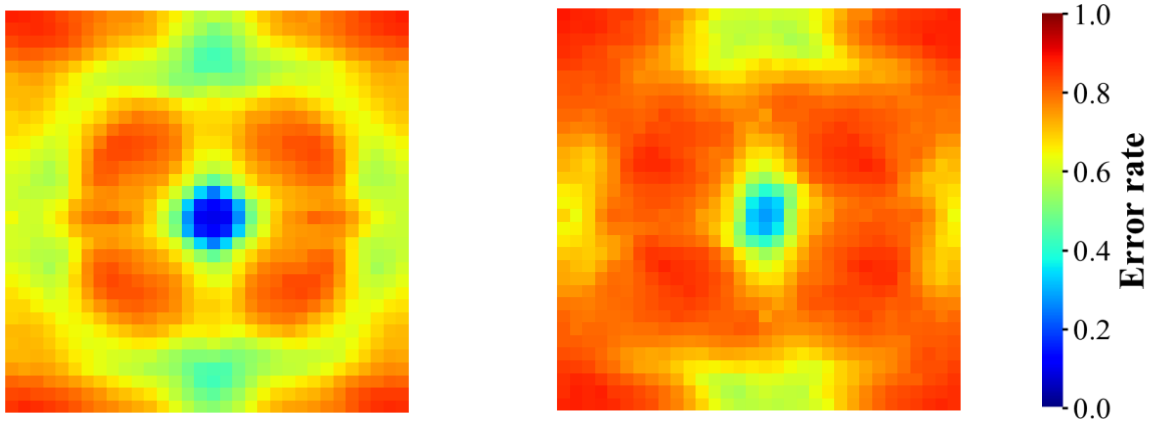
損失地形とテスト損失の関係

https://arxiv.org/abs/1906.03291

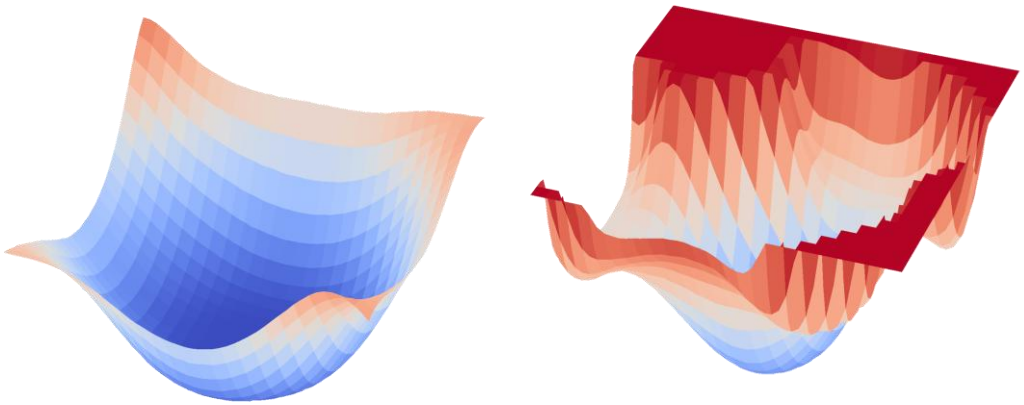
損失地形の平坦性とノイズ頑健性の関係

評価データにノイズを乗せたCorruptionデータに対する汎可性への注目

言語：誤字・脱字，画像：手ブレ・歪み，音声：環境音・反響　等々



平坦な解



先鋭な解

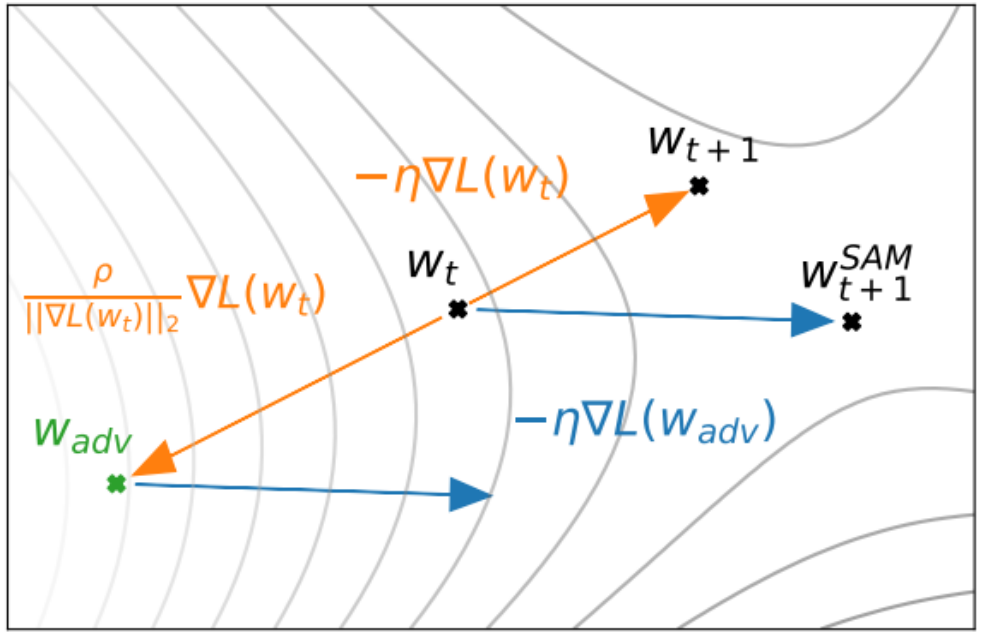
損失地形とノイズ頑健性（ロバスト性）の関係

損失地形を平坦化するOptimizer

Sharpness-Aware Minimization（SAM）

Sharpness-Aware Minimization for Efficiently Improving Generalization [Foret+, ICLR 2021]

✓ 最悪方向への摂動に対する勾配を逆伝播することで平坦解を探索する手法



SAMのアルゴリズム

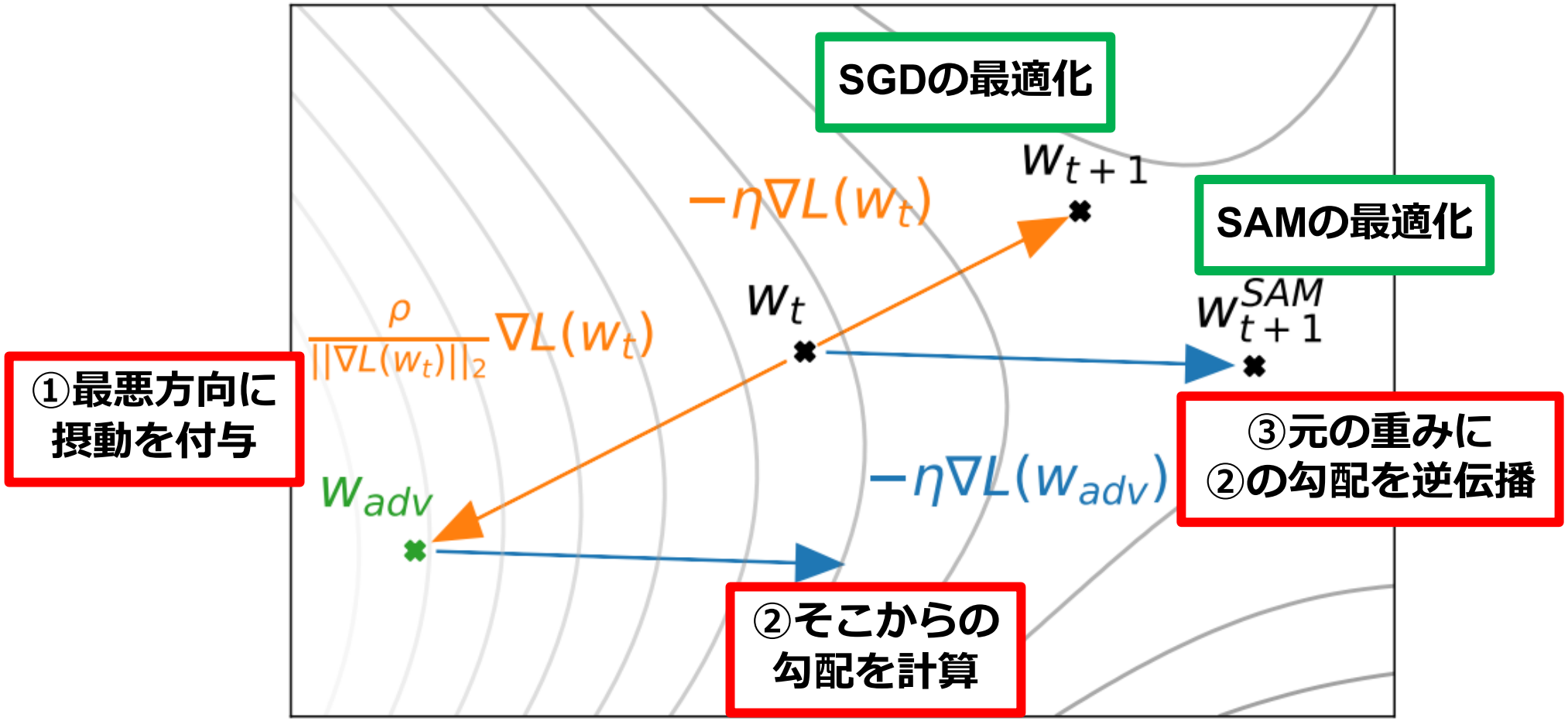
損失地形の比較
（左）SGD（右）SAM

ImageNetでSGDを圧倒

Model	Epoch	SAM Top-1	SAM Top-5	Standard Training (No SAM) Top-1	Standard Training (No SAM) Top-5
ResNet-50	100	**22.5**$_{\pm 0.1}$	6.28$_{\pm 0.08}$	22.9$_{\pm 0.1}$	6.62$_{\pm 0.11}$
	200	**21.4**$_{\pm 0.1}$	5.82$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.37$_{\pm 0.04}$
	400	**20.9**$_{\pm 0.1}$	5.51$_{\pm 0.03}$	22.3$_{\pm 0.1}$	6.40$_{\pm 0.06}$
ResNet-101	100	**20.2**$_{\pm 0.1}$	5.12$_{\pm 0.03}$	21.2$_{\pm 0.1}$	5.66$_{\pm 0.05}$
	200	**19.4**$_{\pm 0.1}$	4.76$_{\pm 0.03}$	20.9$_{\pm 0.1}$	5.66$_{\pm 0.04}$
	400	**19.0**$_{\pm <0.01}$	4.65$_{\pm 0.05}$	22.3$_{\pm 0.1}$	6.41$_{\pm 0.06}$
ResNet-152	100	**19.2**$_{\pm <0.01}$	4.69$_{\pm 0.04}$	20.4$_{\pm <0.0}$	5.39$_{\pm 0.06}$
	200	**18.5**$_{\pm 0.1}$	4.37$_{\pm 0.03}$	20.3$_{\pm 0.2}$	5.39$_{\pm 0.07}$
	400	**18.4**$_{\pm <0.01}$	4.35$_{\pm 0.04}$	20.9$_{\pm <0.0}$	5.84$_{\pm 0.07}$

SAMのアルゴリズム（図解）

✓ 最悪方向への摂動に対する勾配を逆伝播することで平坦解を探索する



SAMのアルゴリズム（数式，アルゴリズム）

- ✓ SAMの目的関数は以下

$$\min_{\mathbf{w}} L^{SAM}(\mathbf{w}) = L_S(\mathbf{w}) + \lambda\left|\left|\mathbf{w}\right|\right|^2 \; where \; L_S(\mathbf{w}) = \max_{\{||\epsilon|| \le \rho\}} L_S(\mathbf{w}+\epsilon),$$

$$\epsilon = \rho \mathrm{sign}(\nabla L_S(\mathbf{w})) \frac{\nabla L_S(\mathbf{w})}{\left|\left|\nabla L_S(\mathbf{w})\right|\right|_2}$$

ϵ: perturbation, ρ: hyperparameter

- ✓ 実装上は誤差逆伝播を2回/iterにすれば完了

Input: Training set $\mathcal{S} \triangleq \cup_{i=1}^{n}\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}$, Loss function $l : \mathcal{W} \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}_+$, Batch size b, Step size $\eta > 0$, Neighborhood size $\rho > 0$.
Output: Model trained with SAM
Initialize weights $\boldsymbol{w}_0$, $t = 0$;
while *not converged* **do**
  Sample batch $\mathcal{B} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), ...(\boldsymbol{x}_b, \boldsymbol{y}_b)\}$;
  Compute gradient $\nabla_{\boldsymbol{w}} L_{\mathcal{B}}(\boldsymbol{w})$ of the batch's training loss;
  Compute $\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})$ per equation 2;
  Compute gradient approximation for the SAM objective (equation 3): $\boldsymbol{g} = \nabla_w L_{\mathcal{B}}(\boldsymbol{w})|_{\boldsymbol{w}+\hat{\boldsymbol{\epsilon}}(\boldsymbol{w})}$;
  Update weights: $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \boldsymbol{g}$;
  $t = t + 1$;
end
return $\boldsymbol{w}_t$

SAMの派生①｜ASAM

Adaptive Sharpness-Aware Minimization（ASAM）[Kwon+, ICML2021]

✓ SAMでは各パラメータに均一の摂動を与えていた

– が，値の小さいパラメータと，値の大きいパラメータに対して同じ摂動を与えても効果は薄い

⇒ ASAMでは各パラメータに加える摂動の大きさを値の大きさでスケールする

$$\epsilon = \rho \mathrm{sign}(\nabla L_S(\mathbf{w}))\mathrm{T}_{\mathbf{w}} \frac{\nabla L_S(\mathbf{w})}{\left|\left|\nabla L_S(\mathbf{w})\right|\right|_2}, \mathrm{T}_{\mathbf{w}} = |\mathbf{w}| + n$$

n: hyperparameter

✓ ImageNet等々で性能向上（数値上は誤差レベルだが，実際SAM・ASAMは頻繁に使われる）

	SGD	SAM	ASAM
Top1	$75.79_{\pm 0.22}$	$76.39_{\pm 0.03}$	$\mathbf{76.63}_{\pm 0.18}$
Top5	$92.62_{\pm 0.04}$	$92.97_{\pm 0.07}$	$\mathbf{93.16}_{\pm 0.18}$

SAMの派生②｜SSAM

Stabilizing Sharpness-aware Minimization Through A Simple Renormalization Strategy
[Tan+, arXiv 2024]

✓ SAMでは最悪方向に摂動を飛ばす際に鞍点へ移動する場合がある
  – このため，学習率と摂動の調整もシビアでチューニングに時間がかかる

⇒ SSAMでは重み更新時の勾配ノルムを，摂動前の最悪方向への勾配ノルムに一致させる

$$\nabla L_S(\mathbf{w}+\epsilon) \leftarrow \frac{\left|\left|\nabla L_S(\mathbf{w})\right|\right|}{\left|\left|\nabla L_S(\mathbf{w}+\epsilon)\right|\right|} \nabla L_S(\mathbf{w}+\epsilon)$$

✓ ImageNetでは学習が安定しづらいViTで高性能を達成
（SGDに対するAdam のような位置づけ）

	SGD/AdamW	SAM*	SAM	SSAM
ResNet-18	70.56 ± 0.03	70.74 ± 0.02	70.66 ± 0.12	**70.76 ± 0.09**
ResNet-50	77.09 ± 0.12	77.81 ± 0.04	77.82 ± 0.08	**77.89 ± 0.13**
ViT-S-32	65.42 ± 0.12	67.42 ± 0.21	69.98 ± 0.11	**71.15 ± 0.18**
ViT-S-16	72.25 ± 0.09	73.81 ± 0.06	76.88 ± 0.25	**77.41 ± 0.13**

SAMの派生③｜その他の手法

- ✓ SAMにはその他多くの派生手法が存在

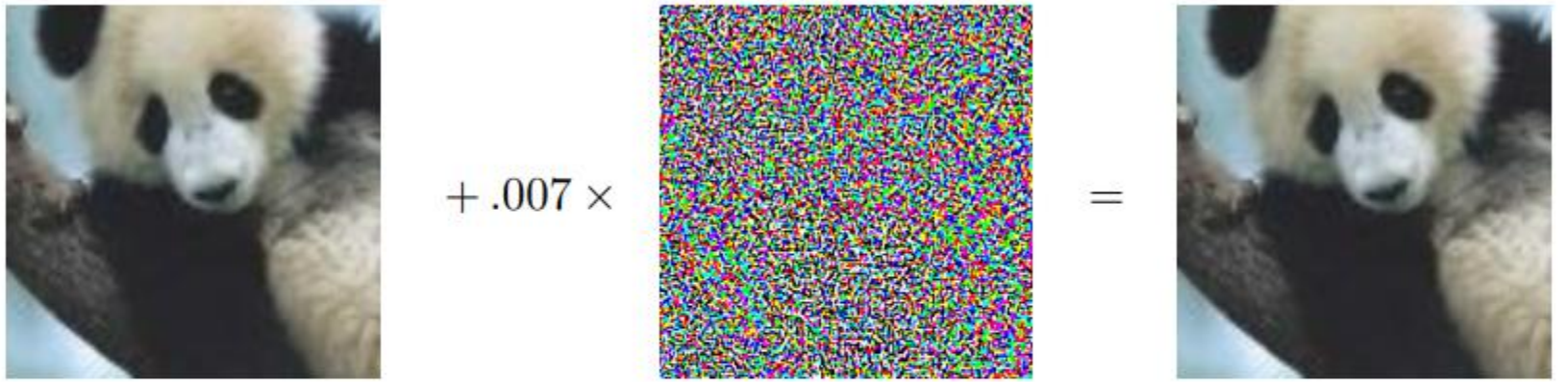
名称（発表年）	派生の種類	手法と効果
GSAM（ICLR ‘22）	汎可性改善	複数の摂動方向の勾配を利用
WSAM（KDD ‘23）	汎可性改善	平坦性を正則化に組み込む
CR-SAM（AAAI ‘24）	安定性改善	Hessian曲率に基づく摂動を導入
VaSSO（arXiv ‘25）	安定性改善	分散抑制による敵対的摂動
LookSAM（CVPR ‘22）	計算効率改善	SAMの2段階更新を1段階に近似
ESAM（arXiv ‘22）	計算効率改善	摂動計算の対象を一部の層に限定
AE-SAM（ICLR ‘23）	計算効率改善	SAMの適用タイミングを切り替える
SAMPa（NeurIPS ‘24）	計算効率改善	SAMの2段階更新を並列化

（補足）敵対的学習

学習データに敵対的ノイズを付加することでNNの頑健性を向上させる

敵対的攻撃（Adversarial Attack）に対する頑健性を獲得する学習

※ 敵対的生成ネットワーク（GAN）とは別なので注意



https://ai-scholar.tech/articles/image-recognition/assistive_signals

敵対的学習に利用されるサンプル

（補足）SAM or 敵対的学習

On the Duality Between Sharpness-Aware Minimization and Adversarial Training
[Zhang+ ICML2024]

✓ SAMと敵対的学習の違いを実験的に評価した論文

クリーンデータ，敵対的データでの比較

Method	Natural Accuracy	FGSM $\epsilon = \frac{1}{255}$	ℓ_∞-PGD $\epsilon = \frac{1}{255}$	ℓ_2-PGD $\epsilon = \frac{32}{255}$	ℓ_2-AA. $\epsilon = \frac{32}{255}$	StAdv	FAB	Pixle	Average Robustness
SGD	94.5	63.4	37.9	41.5	31.7	35.2	44.8	10.0	**37.8**
Adam	93.9	44.3	17.4	20.7	13.9	20.4	24.7	7.6	21.3
SAM ($\rho = 0.1$)	95.4	63.3	46.2	48.7	43.6	39.3	49.2	13.4	43.4
SAM ($\rho = 0.2$)	**95.5**	66.7	51.3	53.4	48.1	44.2	53.4	13.2	47.2
SAM ($\rho = 0.3$)	95.4	66.6	51.2	53.5	47.8	46.1	53.8	13.7	47.5
SAM ($\rho = 0.4$)	94.7	69.6	56.4	58.6	51.8	54.9	57.6	14.3	**51.9**
AT (ℓ_∞-$\epsilon = \frac{8}{255}$)	84.5	81.9	81.8	79.7	79.5	82.0	79.5	26.9	73.0
AT (ℓ_2-$\epsilon = \frac{128}{255}$)	89.2	84.1	84.1	84.8	84.8	80.4	84.8	32.0	76.4

ノイズデータでの比較

Method	Natural	Corruption (Avg.)
SGD	94.5	34.91
Adam	93.9	29.60
SAM ($\rho = 0.3$)	**95.4**	32.60
ASAM ($\rho = 0.3$)	**95.4**	**36.69**
ESAM ($\rho = 0.3$)	95.2	36.29
ℓ_∞-AT($\epsilon = 8/255$)	84.5	15.67
ℓ_2-AT($\epsilon = 128/255$)	89.2	23.13

✓ クリーンデータ：SAM＞敵対的学習

✓ ノイズデータ：SAM＞敵対的学習

✓ 敵対的データ：SAM＜敵対的学習

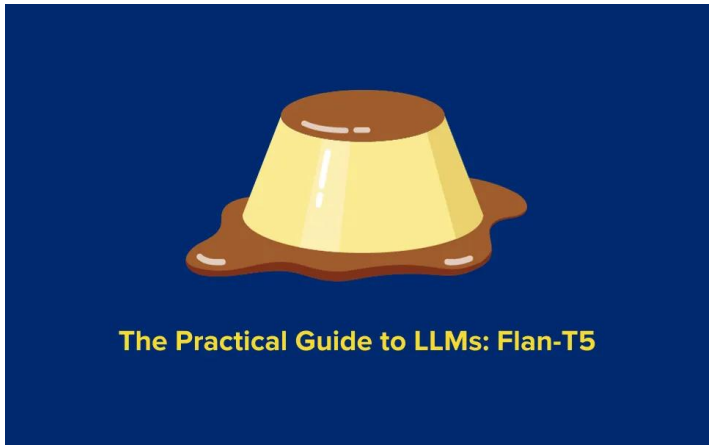
⇒ SAMは本来，重み空間の平坦性（重みへの頑健性）を向上させるだけ の手法であるが，
実際には入力空間の平坦性（入力への頑健性）を向上させることに寄与している

結論

- ✓ NN（深層学習）初期は損失関数の最適化を目的とした手法がメジャー
  - SGD，Adam，AdamW
- ✓ 実利用を考えた際のノイズロバスト性を考慮した最適化が登場
  - SAM，ASAM，（敵対的学習）

（余談）近年のLLM向けOptimizer

- ✓ 近年，LLM向けに独自Optimizerを開発して学習するケースが存在する
  - RLHFでは別途，GRPOやSRPO，GSPOなども出ているが，↑はSFTやPre-training時



https://medium.com/georgian-impact-blog/the-practical-guide-to-llms-flan-t5-6d26cc5f14c0

T5
Googleの~11BパラメータLLM
OptimizerはAdafactor
└Adamの二次モーメントを
行列分解して省メモリ化する手法

https://www.ankursnewsletter.com/p/llama-metas-open-source-rival-to

LLaMA
Metaの~2TパラメータLLM
OptimizerはLion
└Adamの二次モーメントを
符号のみ使用する手法

https://medium.com/@servifyspheresolutions/kimi-k2-moonshot-ais-trillion-parameter-powerhouse-redefines-open-source-ai-d71e25c9e8b1

Kimi K2
Moonshot AIの~1TパラメータLLM
OptimizerはMuon
└Newton-Schulz 直交化によって
勾配方向を整える手法