Regularizing Neural Networks via Adversarial Model Perturbation

Yaowei Zheng,[1] Richong Zhang,[1] Yongyi Mao[2]

[1]BDBC and SKLSDE, Beihang University, Beijing, China

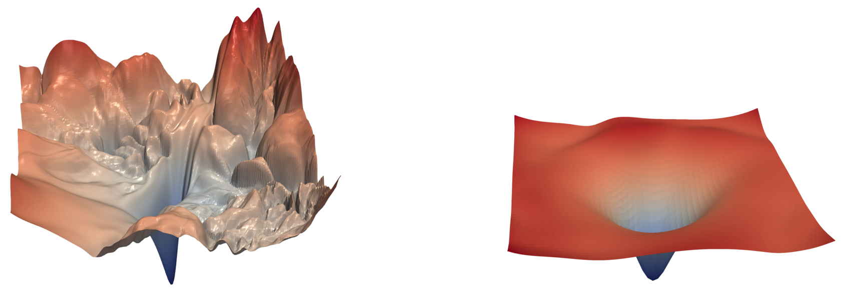
[2]School of EECS, University of Ottawa, Ottawa, Canada

Summary

Background: Effective regularization schemes are highly desired in deep learning for alleviating overfitting and improving generalization.

Motivation: Previous work suggested that flat minima can improve generalization both in theoretical and empirical perspectives [1-4].

Contribution: We propose Adversarial Model Perturbation (AMP) as a powerful regularization scheme principled by the objective of finding flat minima, which achieves better classification performance and calibration results.



(Better minima are flatter, visualized by Li *et al.*, 2018.)

Training Algorithm

Optimization Objective:

$$\min_{\boldsymbol{\theta}} \max_{\Delta:\|\Delta\|\le\epsilon} \mathcal{L}_{\mathrm{ERM}}(\boldsymbol{\theta}+\Delta)$$

Pseudo-code:

while $\boldsymbol{\theta}$ *not converged* **do**
  Initialize perturbation Δ with $\mathbf{0}$;
  for $n \leftarrow 1$ *to* N **do**
    Update Δ to maximize $\mathcal{L}_{\mathrm{ERM}}(\boldsymbol{\theta}+\Delta)$ via gradient ascent with learning rate ζ;
    if $\|\Delta\|_2 > \epsilon$ **then**
      Normalize Δ to restrict its norm $\|\Delta\|_2$ to ϵ;
  Update $\boldsymbol{\theta}$ to minimize $\mathcal{L}_{\mathrm{ERM}}(\boldsymbol{\theta}+\Delta)$ via gradient descent with learning rate η;

PyTorch Implementation:

```
from amp import AMP
opt = AMP(params, lr=0.1, epsilon=0.5)
for inputs, targets in dataset:
  def closure():
    opt.zero_grad()
    outputs = model(inputs)
    loss = loss_fn(outputs, targets)
    loss.backward()
    return outputs, loss
  outputs, loss = opt.step(closure)
```

Official Code:

https://github.com/hiyouga/AMP-Regularizer

Method

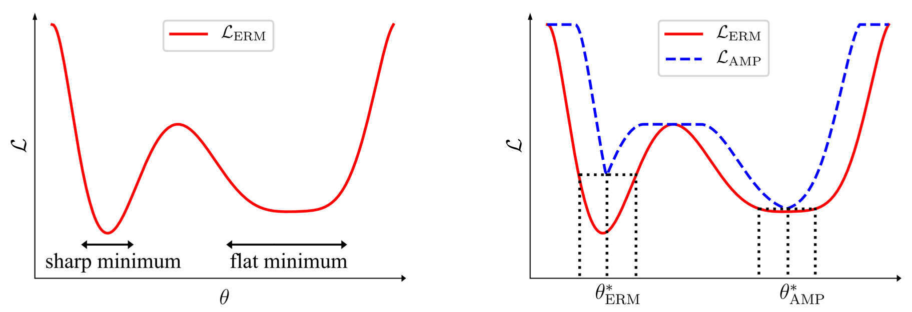
AMP: Adversarial Model Perturbation

We derive an AMP loss from the empirical risk (ERM loss). Minimizing such an AMP loss can penalize the sharp minima by adversarially perturbing the model parameters.

$$\mathcal{L}_{\mathrm{ERM}}(\boldsymbol{\theta}) := \frac{1}{|D|}\sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{D}} \ell(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})$$

$$\mathcal{L}_{\mathrm{AMP}}(\boldsymbol{\theta}) := \max_{\Delta:\|\Delta\|\le\epsilon} \mathcal{L}_{\mathrm{ERM}}(\boldsymbol{\theta}+\Delta)$$

It can be seen as an analogue of a "max-pooling" operation on the empirical risk on each point in the parameter space.



Theoretical Justification

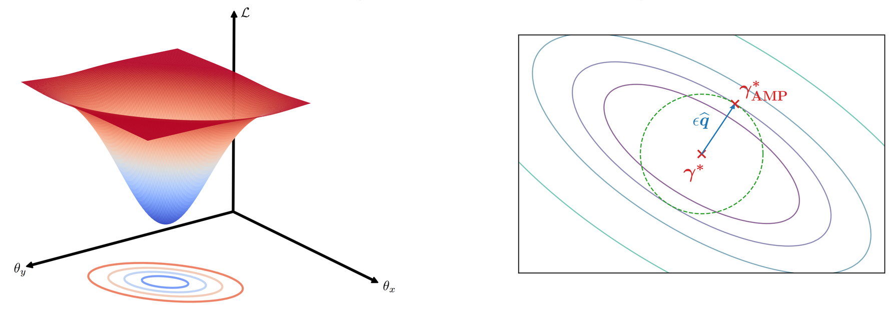
AMP Finds Flatter Minima:

We assume that the loss surface of each local minimum in $\mathcal{L}_{\mathrm{ERM}}$ can be *locally* approximated as an inverted Gaussian surface γ with a mean vector $\boldsymbol{\mu}$ and a covariance matrix $\boldsymbol{\kappa}$. Under the locally Gaussian assumption, the empirical risk $\gamma(\boldsymbol{\theta};\boldsymbol{\mu},\boldsymbol{\kappa},A,C)$ is minimized when $\boldsymbol{\theta}=\boldsymbol{\mu}$ and the minimum value is $\gamma^*(\boldsymbol{\mu},\boldsymbol{\kappa},A,C)=C-A$. The minimum value of the AMP loss is the empirical risk at the location in the narrowest principal direction of the cross-section of the loss surface:

$$\gamma^*_{\mathrm{AMP}}(\boldsymbol{\mu},\boldsymbol{\kappa},A,C) = C - A\exp\left(-\frac{\epsilon^2}{2\sigma^2}\right)$$

where σ^2 is the smallest eigenvalue of $\boldsymbol{\kappa}$.

It is clear that γ^*_{AMP} although related to the minimum value of empirical risk γ^*, it also takes into account the curvature of the surface around the local minimum. Thus we can find a flatter minimum through minimizing $\mathcal{L}_{\mathrm{AMP}}$.



AMP Regularizes Gradient Norm:

Consider that $N=1$. The AMP training is equivalent to ERM training with an additional term:

$$\tilde{\mathcal{J}}_{\mathrm{ERM}}(\boldsymbol{\theta}) := \mathcal{J}_{\mathrm{ERM}}(\boldsymbol{\theta}) + \Omega(\boldsymbol{\theta})$$

where

$$\Omega(\boldsymbol{\theta}) := \begin{cases} \zeta\|\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\mathrm{ERM}}(\boldsymbol{\theta})\|_2^2, & \|\zeta\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\mathrm{ERM}}(\boldsymbol{\theta})\|_2 \le \epsilon \\ \epsilon\|\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\mathrm{ERM}}(\boldsymbol{\theta})\|_2, & \|\zeta\nabla_{\boldsymbol{\theta}}\mathcal{J}_{\mathrm{ERM}}(\boldsymbol{\theta})\|_2 > \epsilon \end{cases}$$

Experiments

Results on Image Classification Benchmarks:

PreActResNet18	Test Error (%)	Test NLL
ERM	2.95±0.063	0.166±0.004
Dropout	2.80±0.065	0.156±0.012
Label Smoothing	2.78±0.087	0.998±0.002
Flooding	2.84±0.047	0.130±0.003
MixUp	2.74±0.044	0.146±0.004
Adv. Training	2.77±0.080	0.151±0.018
RMP	2.93±0.066	0.161±0.010
AMP	**2.30±0.025**	**0.096±0.002**
VGG16	Test Error (%)	Test NLL
ERM	3.14±0.060	0.140±0.027
Dropout	2.96±0.049	0.134±0.027
Label Smoothing	3.07±0.070	1.004±0.002
Flooding	3.15±0.085	0.128±0.003
MixUp	3.09±0.057	0.160±0.003
Adv. Training	2.94±0.091	0.122±0.003
RMP	3.19±0.052	0.134±0.004
AMP	**2.73±0.015**	**0.116±0.006**

(a) SVHN

PreActResNet18	Test Error (%)	Test NLL
ERM	5.02±0.212	0.239±0.009
Dropout	4.86±0.148	0.223±0.009
Label Smoothing	4.85±0.115	1.038±0.003
Flooding	4.97±0.082	0.166±0.003
MixUp	4.09±0.117	0.198±0.004
Adv. Training	4.99±0.085	0.247±0.006
RMP	4.97±0.167	0.239±0.008
AMP	**3.97±0.091**	**0.129±0.003**
VGG16	Test Error (%)	Test NLL
ERM	6.32±0.193	0.361±0.012
Dropout	6.22±0.147	0.314±0.009
Label Smoothing	6.29±0.158	1.076±0.003
Flooding	6.26±0.145	0.234±0.005
MixUp	**5.48±0.112**	0.251±0.003
Adv. Training	6.49±0.130	0.380±0.010
RMP	6.30±0.109	0.363±0.010
AMP	5.65±0.147	**0.207±0.005**

(b) CIFAR-10

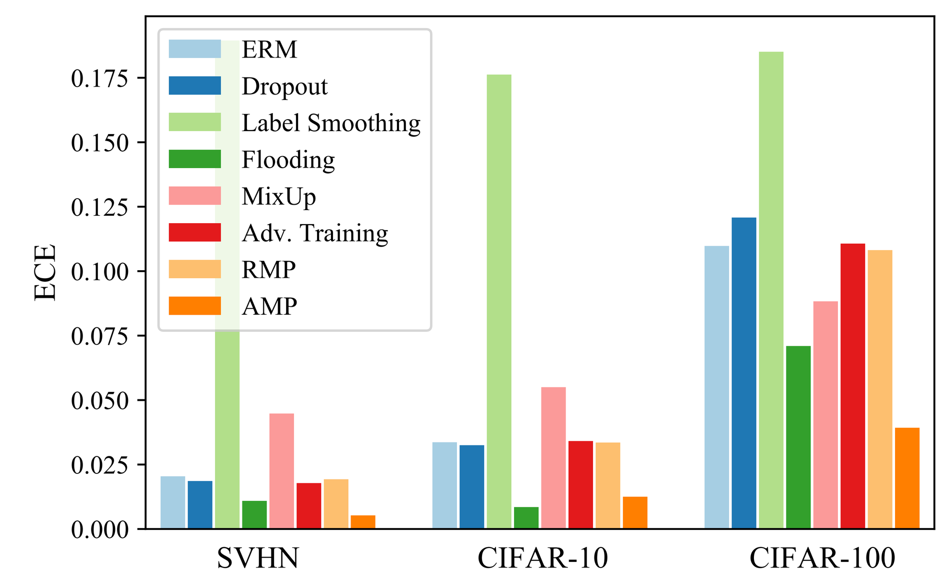
PreActResNet18	Test Error (%)	Test NLL
ERM	24.31±0.303	1.056±0.013
Dropout	24.48±0.351	1.110±0.021
Label Smoothing	22.07±0.256	2.099±0.005
Flooding	24.50±0.234	0.950±0.011
MixUp	21.78±0.210	0.910±0.007
Adv. Training	25.23±0.229	1.110±0.012
RMP	24.28±0.138	1.059±0.011
AMP	**21.51±0.308**	**0.774±0.016**
VGG16	Test Error (%)	Test NLL
ERM	27.84±0.297	1.827±0.209
Dropout	27.72±0.337	1.605±0.062
Label Smoothing	27.49±0.179	2.310±0.005
Flooding	27.93±0.271	1.221±0.037
MixUp	26.81±0.254	1.136±0.013
Adv. Training	29.12±0.145	1.535±0.389
RMP	27.81±0.327	1.873±0.035
AMP	**25.60±0.168**	**1.049±0.049**

(c) CIFAR-100

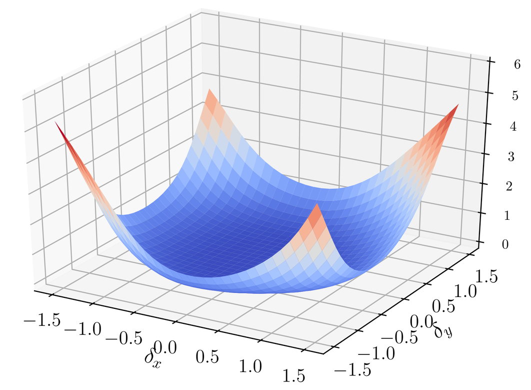
Improvement over Various Data Augmentation Techniques:

Test Error (%) Dataset	Augment Technique	WideResNet-28-10 ERM	WideResNet-28-10 AMP	PyramidNet-164-270 ERM	PyramidNet-164-270 AMP
SVHN	Vanilla	2.57±0.067	**2.19±0.036**	2.47±0.034	**2.11±0.041**
	Cutout	2.27±0.085	**1.83±0.018**	2.19±0.021	**1.82±0.023**
	AutoAug	1.91±0.059	**1.61±0.024**	1.80±0.044	**1.35±0.056**
CIFAR-10	Vanilla	3.87±0.167	**3.00±0.059**	3.60±0.197	**2.75±0.040**
	Cutout	3.38±0.081	**2.67±0.043**	2.83±0.102	**2.27±0.034**
	AutoAug	2.78±0.134	**2.32±0.097**	2.49±0.128	**1.98±0.062**
CIFAR-100	Vanilla	19.17±0.270	**17.33±0.110**	17.13±0.210	**15.09±0.092**
	Cutout	18.12±0.114	**16.04±0.071**	16.45±0.136	**14.34±0.153**
	AutoAug	17.79±0.185	**14.95±0.088**	15.43±0.269	**13.36±0.245**

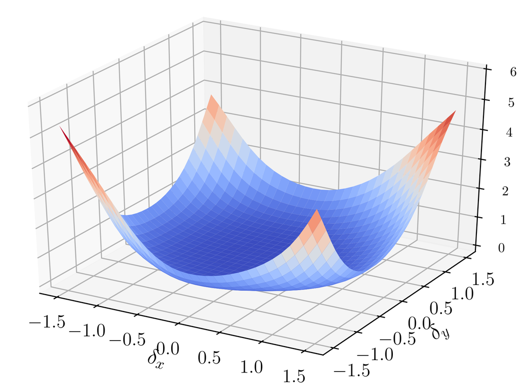
Calibration Results: (lower is better)



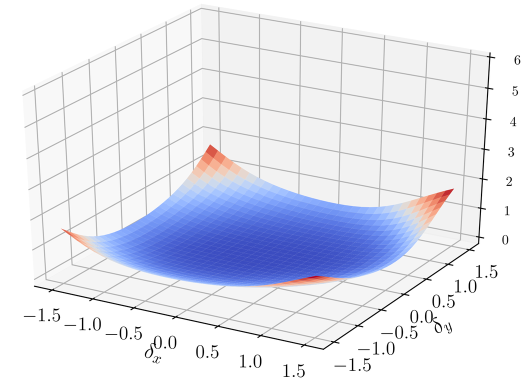
Flatness of the Selected Minima:



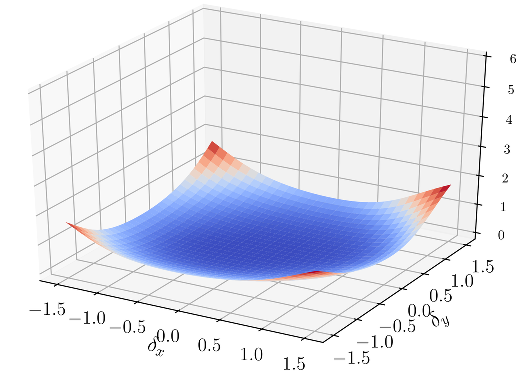
ERM Training Loss



ERM Test Loss

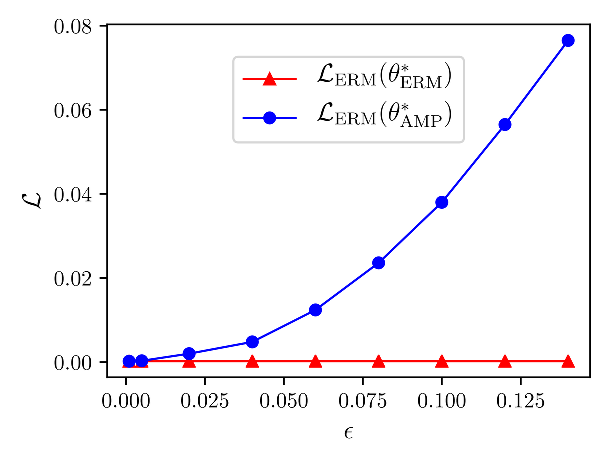


AMP Training Loss

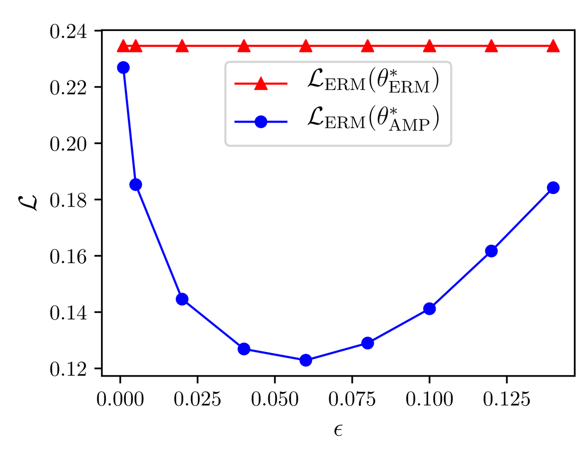


AMP Test Loss

Loss Values with Varying Perturbation Size:



(a) CIFAR-10 Training Set



(b) CIFAR-10 Test Set

References:

[1] Hochreiter *et al.* Flat minima. Neural Computation, 9(1):1-42, 1997.

[2] Keskar *et al.* On large-batch training for deep learning: Generalization gap and sharp minima. In ICLR, 2017.

[3] Li *et al.* Visualizing the loss landscape of neural nets. In NeurIPS, 2018.

[4] Foret *et al.* Sharpness-aware minimization for efficiently improving generalization. In ICLR, 2021.