Regularizing Neural Networks via Adversarial Model Perturbation

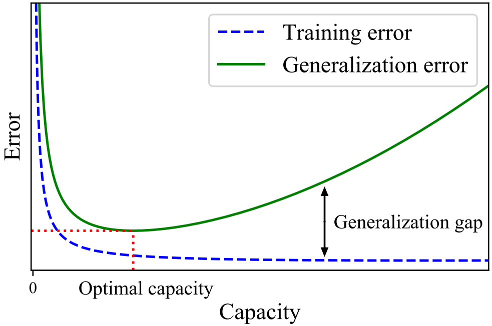
Yaowei Zheng[1] Richong Zhang[1] Yongyi Mao[2]

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

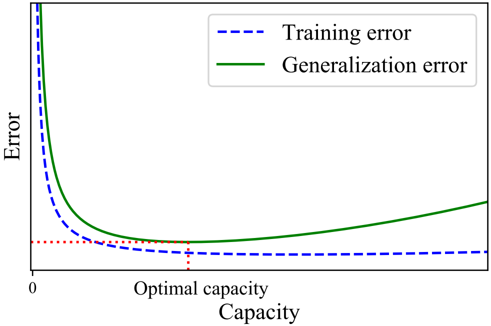
[2]School of EECS, University of Ottawa, Ottawa, Canada

CVPR 2021

Outline



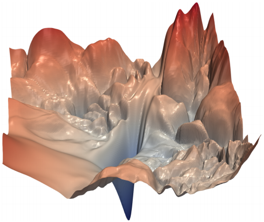
(a) without regularization



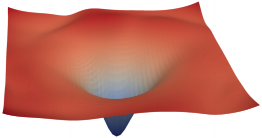
(b) with regularization

Flat Minima Helps Generalization

- Flat minima correspond to low-complexity networks. (Hochreiter *et al.*, 1997)
- Small-batch SGD produces flat minima that generalize well. (Keskar *et al.*, 2017)
- Better minimizers of loss function are flatter in visualization. (Li *et al.*, 2018)
- A PAC-Bayes based generalization guarantee for flat minima. (Foret *et al.*, 2020)



(a) ResNet without skip connections



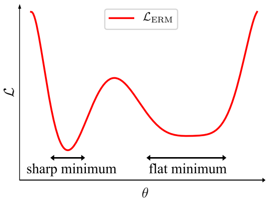
(b) ResNet with skip connections (Li *et al.*, 2018)

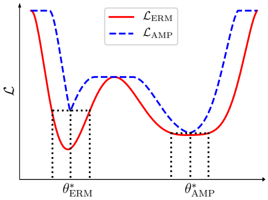
From Empirical Risk to AMP Loss

In this paper, we optimize an alternative "AMP loss".

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

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





Training Algorithm

A mini-batch SGD is used for solving the "min-max" problem.

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

Algorithm 1: Adversarial Model Perturbation Training

while $\boldsymbol{\theta}$ *not converged* **do**
  Initialize perturbation Δ with 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;

Implementation

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

```python
from amp import AMP
optimizer = AMP(model.parameters(), lr=0.1, epsilon=0.5, momentum=0.9)
for inputs, targets in dataset:
    def closure():
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = loss_fn(outputs, targets)
        loss.backward()
        return outputs, loss
    outputs, loss = optimizer.step(closure)
```

AMP Finds Flatter Local Minima

Locally Gaussian Assumption of Empirical Risk

$$\mathcal{L}_{\mathrm{ERM}} \approx \gamma(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\kappa}, A, C)$$

where $\gamma(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\kappa}, A, C)$ *is minimized when* $\boldsymbol{\theta} = \boldsymbol{\mu}$ *and the minimum value is* $\gamma^*(\boldsymbol{\mu}, \boldsymbol{\kappa}, A, C) = C - A$.

Theorem (stated informally)

The minimum value of the AMP loss is

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

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

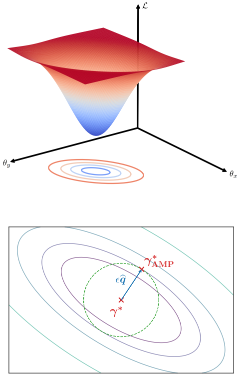


Figure: The minimum values of γ and γ_{AMP}.

AMP Regularizes Gradient Norm

Theorem (stated informally)

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

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

where

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

Experimental Setup

Image classification datasets:

- SVHN (10-way)
- CIFAR-10 (10-way)
- CIFAR-100 (100-way)

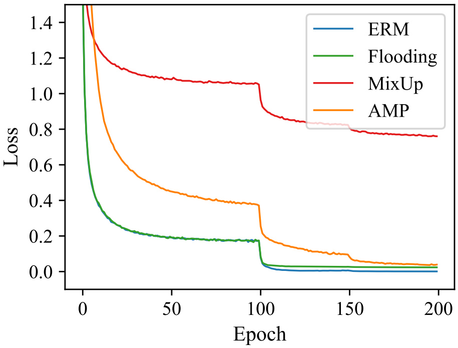
Compared methods:

- ERM (Vapnik *et al*., 1998)
- Dropout (Srivastava *et al*., 2014)
- Label smoothing (Szegedy *et al*., 2016)
- Flooding (Ishida *et al*., 2020)
- MixUp (Zhang *et al*., 2018)
- Adversarial Training (Goodfellow *et al*., 2015)

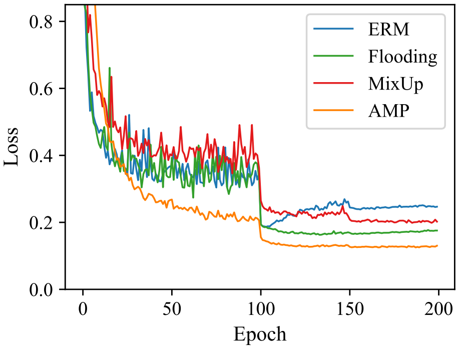
Model architectures:

- PreActResNet18 (He *et al*., 2016)
- VGG16 (Simonyan *et al*., 2015)
- WideResNet-28-10 (Zagoruyko *et al*., 2016)
- PyramidNet-164-270 (Han *et al*., 2017)

Loss Curves



(a) CIFAR-10 Training Set



(b) CIFAR-10 Test Set

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

Table: Top-1 classification errors and test neg-log-likelihoods.

Improvement over Data Augmentation

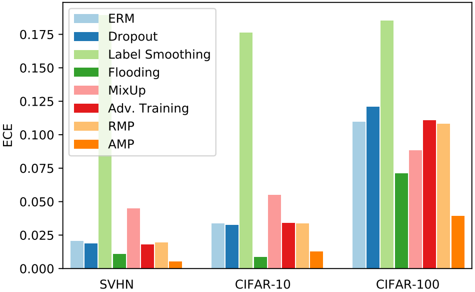
		WideResNet-28-10		PyramidNet-164-270	
		ERM	AMP	ERM	AMP
SVHN	Vanilla	2.57±0.067	**2.19±0.036**	2.47±0.034	**2.11±0.041**
	Cutout	2.27±0.085	**1.83±0.018**	2.19±0.021	**1.82±0.023**
	AutoAug	1.91±0.059	**1.61±0.024**	1.80±0.044	**1.35±0.056**
CIFAR-10	Vanilla	3.87±0.167	**3.00±0.059**	3.60±0.197	**2.75±0.040**
	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**

Table: Top-1 classification errors and test neg-log-likelihoods.

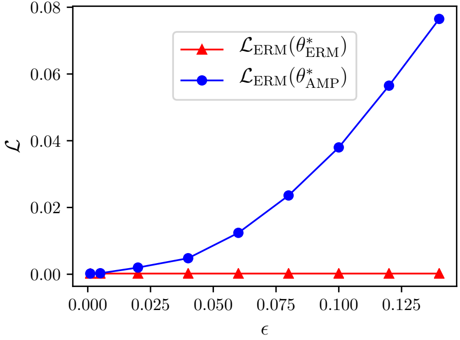
Calibration Results

Expected Calibration Error: (lower is better)

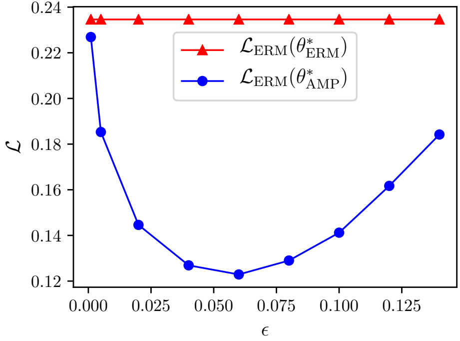
$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} \left| \underbrace{\frac{1}{|B_m|} \sum_{i \in B_m} 1(\hat{y}_i = y_i)}_{\text{accuracy}} - \underbrace{\frac{1}{|B_m|} \sum_{i \in B_m} \hat{p}_i}_{\text{confidence}} \right|$$



Loss Values with Varying Perturbation Size

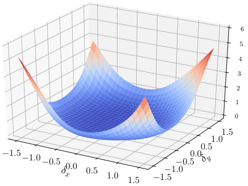


(a) CIFAR-10 Training Set

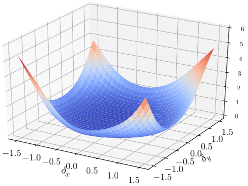


(b) CIFAR-10 Test Set

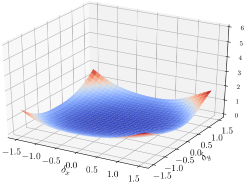
Flatness of the Selected Minima



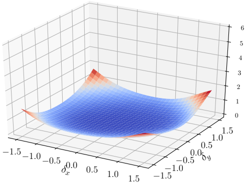
(a) ERM Training Loss



(b) ERM Test Loss



(c) AMP Training Loss



(d) AMP Test Loss

Conclusion

- Motivated by the understanding that flat minima help generalization, we propose adversarial model perturbation (AMP) as an efficient regularization scheme.
- We theoretically justify that AMP is capable of finding flatter local minima, thereby improving generalization.
- Extensive experiments on the benchmark datasets demonstrate that AMP achieves the best performance among the compared regularization schemes on various modern neural network architectures.

ArXiv: `https://arxiv.org/abs/2010.04925`

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