Virtual Adversarial Training

Buse Gul Atli

Aalto University, Department of Science

buse.atli@aalto.fi

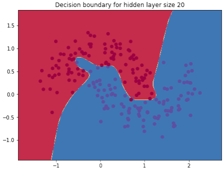
May 21, 2019

What We Will Cover

1. Miyato et.al Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning 2018
2. Miyato et. al Distributional Smoothing with Virtual Adversarial Training 2015.

Overfitting vs Underfitting

- Poor design of the model
- Noise in the training set

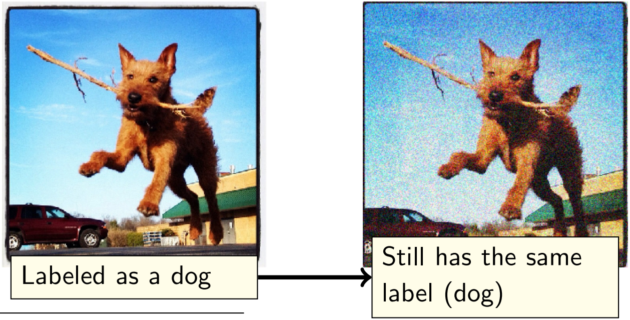


Regularization

- Smoothing the output distribution w.r.t spatial/temporal inputs
- L_1 and L_2 regularization
- Applying random perturbations to input and hidden layers
- Droput in NNs.

Adversarial Training[1]

- Adds a noise to the image where the noise is in the adversarial direction
- Model's probability of correct classification is reduced in adversarial direction.



[1] Ian J. Goodfellow, Jonathon Shlens, Christian Szegedy, Explaining and Harnessing Adversarial Examples, 2015

Adversarial Training

- Adds a noise to the image where the noise is in the adversarial direction
- Improves the generalization performance
- Robustness against adversarial perturbation

Adversarial Training

$$L_{adv}(x_l, \theta) := D[q(y|x_l)p(y|x_l + r_{adv}, \theta)] \quad (1)$$

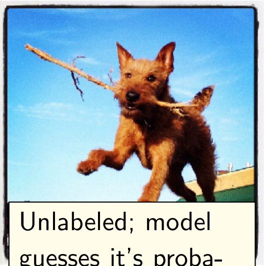
$$\text{where} \quad r_{adv} : argmax_{r, \|r\|_2 \leq \epsilon} D[q(y|x_l)p(y|x_l + r, \theta)] \quad (2)$$

$$r_{adv} \simeq \epsilon \frac{g}{\|g\|_2}, g = \nabla_{x_l} D[h(y; y_l)p(y|x_l, \theta)] \quad (3)$$

$$r_{adv} \simeq \epsilon \texttt{sign}(g) \text{ when norm is } L_\infty$$

Virtual Adversarial Training

- How can we modify the adversarial training loss in Eq 1 when full label information is not available ?
- Adversarial perturbation intended to change the guess



Unlabeled; model guesses it's probably a dog, maybe stick

New guess should match the old guess (probably dog, maybe stick)

Virtual Adversarial Training

- x_* denotes both labeled x_l or unlabeled x_{ul} samples

$$L_{adv}(x_*, \theta) := D[q(y|x_*)p(y|x_* + r_{adv}, \theta)]$$

$$\text{where} \quad r_{adv} : argmax_{r, \|r\|_2 \leq \epsilon} D[q(y|x_*)p(y|x_* + r, \theta)]$$

Virtual Adversarial Training

- Replace $q(y|x)$ with current estimate of $p(y|x, \hat{\theta})$

Local Distributional Smoothness (LDS)

$$LDS(x_*, \theta) := D[p(y|x_*, \theta)p(y|x_* + r_{adv}, \theta)] \quad (4)$$

$$\text{where} \quad r_{adv} : argmax_{r, \|r\|_2 \leq \epsilon} D[p(y|x_*)p(y|x_* + r, \theta)] \quad (5)$$

Virtual Adversarial Training

$$LDS(x_*, \theta) := D[p(y|x_*, \theta) p(y|x_* + r_{adv}, \theta)]$$
$$\text{where} \quad r_{adv} : argmax_{r, \|r\|_2 \leq \epsilon} D[p(y|x_*) p(y|x_* + r, \theta)] \tag{6}$$

$$R_{vadv}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta) := \frac{1}{N_l + N_{ul}} \sum_{x_* \in \mathcal{D}_l, \mathcal{D}_{ul}} LDS(x_*, \theta) \tag{7}$$

Virtual Adversarial Training

$$LDS(x_*, \theta) := D[P(y|x_*, \theta)p(y|x_* + r_{adv}, \theta)]$$

$$\text{where} \quad r_{adv} : argmax_{r, \|r\|_2 \leq \epsilon} D[p(y|x_*)p(y|x_* + r, \theta)]$$

$$\mathcal{R}_{vadv}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta) := \frac{1}{N_l + N_{ul}} \sum_{x_* \in \mathcal{D}_l, \mathcal{D}_{ul}} LDS(x_*, \theta)$$

Full objective function

$$\ell(\mathcal{D}_l, \theta) + \alpha \mathcal{R}_{vadv}(\mathcal{D}_l, \mathcal{D}_{ul}, \theta) \tag{8}$$

VAT : Fast Approximation Methods for r_{adv}

- Linear approximation in Eq 3 cannot be performed for LDS $D(r, x_*, \hat{\theta})$
- Use second order Taylor approximation, since $\nabla_r D(r, x_*, \hat{\theta}) = 0$ when $r = 0$

$$D(r, x, \hat{\theta}) \simeq \frac{1}{2} r^T H(X, \hat{\theta}) r \tag{9}$$

VAT : Fast Approximation Methods for r_{adv}

$$D(r, x, \hat{\theta}) \simeq \frac{1}{2} r^T H(X, \hat{\theta}) r$$

$$\begin{aligned} r_{adv} &\simeq argmax_r \{ r^T H(x, \hat{\theta}) r ; \|r\|_2 \leq \epsilon \} \\ &= \epsilon \overline{u(x, \hat{\theta})} \end{aligned} \tag{10}$$

where $\overline{v} = \frac{v}{\|v\|_2}$ and u is the first dominant eigenvector

VAT : Fast Approximation Methods for r_{adv}

- $\mathcal{O}(n^3)$ for computing eigenvectors of Hessian
- Use power iteration method

$$d \leftarrow \overline{Hd} \tag{11}$$

$$\begin{aligned} Hd &\simeq \frac{\nabla_r D(r, x, \hat{\theta})|_{r=\xi d} - \nabla_r D(r, x, \hat{\theta})|_{r=0}}{\xi} \\ &= \frac{\nabla_r D(r, x, \hat{\theta})|_{r=\xi d}}{\xi d} \end{aligned} \tag{12}$$

$$d \leftarrow \overline{\nabla_r D(r, x, \hat{\theta})|_{r=\xi d}} \tag{13}$$

VAT : Fast Approximation Methods for r_{adv}

$$r_{adv} \simeq \epsilon \frac{g}{\|g\|_2} \tag{14}$$

$$g = \nabla_r D[p(y|x, \hat{\theta}), p(y|x + r, \hat{\theta})]|_{r=\xi d} \tag{15}$$

- KL divergence for the choice of D

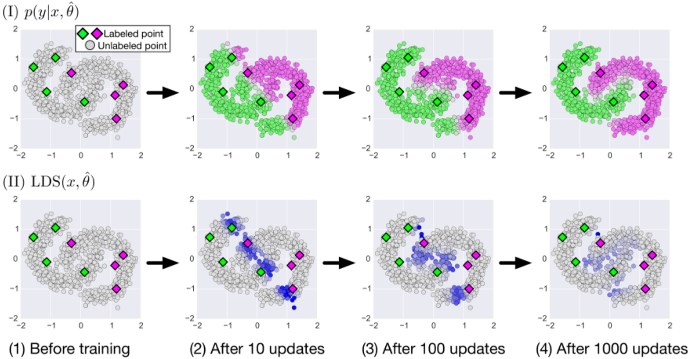
Algorithm 1 Mini-batch SGD for $\nabla_\theta \mathcal{R}_{\mathrm{vadv}}(\theta)|_{\theta=\hat{\theta}}$, with a one-time power iteration method.

1) Choose M samples of $x^{(i)} (i = 1, \ldots, M)$ from dataset $\mathcal{D}$ at random.
2) Generate a random unit vector $d^{(i)} \in R^I$ using an iid Gaussian distribution.
3) Calculate r_{vadv} via taking the gradient of D with respect to r on $r = \xi d^{(i)}$ on each input data point $x^{(i)}$:
   $g^{(i)} \leftarrow \nabla_r D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r, \hat{\theta})\right]\Big|_{r=\xi d^{(i)}}$,
   $r_{\mathrm{vadv}}^{(i)} \leftarrow g^{(i)} / \|g^{(i)}\|_2$
4) **Return**

$$\nabla_\theta \left(\frac{1}{M} \sum_{i=1}^{M} D\left[p(y|x^{(i)}, \hat{\theta}), p(y|x^{(i)} + r_{\mathrm{vadv}}^{(i)}, \theta)\right] \right) \Bigg|_{\theta=\hat{\theta}}$$

VAT Example

- VAT forces the model to be smooth around the points with large LDS values.
- Model predicts the same label for the set of points that belong to the same cluster after 100 updates



VAT vs. Other Regularization Methods

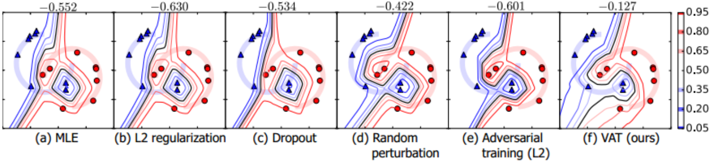


Figure 3: Contours of $p(y = 1|x, \theta)$ drawn by NNs (with ReLU activation) trained with various regularization methods for a single dataset of 'Moons'. A black line represents the contour of value 0.5. Red circles represent the data points with label 1, and blue triangles represent the data points with label 0. The value above each panel correspond to average $\widetilde{\text{LDS}}$ value. Average $\widetilde{\text{LDS}}$ evaluated on the training set with $\epsilon = 0.5$ and $I_p = 5$ is shown at the top of each panel.

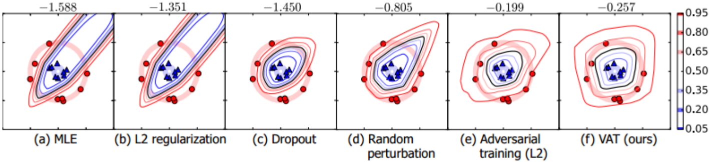


Figure 4: Contours of $p(y = 1|x, \theta)$ drawn by NNs trained with various regularization methods for a single dataset of 'Circles'. The rest of the details follow the caption of the Figure 3.

Random Perturbation Training (RPT) and Conditional Entropy (VAT+EntMin)

- VAT can be written as

$$\mathcal{R}^{(K)}(\theta, \mathcal{D}_l, \mathcal{D}_{ul}) := \frac{1}{N_l + N_{ul}} \sum_{x \in \mathcal{D}_l, \mathcal{D}_{ul}} E_{rk}[D[p(y|x, \hat{\theta})p(y|x + r_K, \theta)]] \tag{16}$$

- $\mathcal{R}^0$: RPT (Smooths the function isotropically)
- Conditional entropy:

$$\begin{aligned} \mathcal{R}_{cent} &= \mathcal{H} \\ &= -\frac{1}{N_l + N_{ul}} \sum_{x \in \mathcal{D}_l, \mathcal{D}_{ul}} \sum_{y} p(y|x, \theta) \log p(y|x, \theta) \end{aligned} \tag{17}$$

VAT Performance on Semi-Supervised Learning

- VAT and data augmentation can be used together

Method	Test error rate(%) SVHN $N_l = 1000$	CIFAR-10 $N_l = 4000$
SWWAE [48]	23.56	
*Skip DGM [27]	16.61 ($\pm$0.24)	
*Auxiliary DGM [27]	22.86	
Ladder networks, Γ model [33]		20.40 ($\pm$0.47)
CatGAN [37]		19.58 ($\pm$0.58)
GAN with FM [36]	8.11 ($\pm$1.3)	18.63 ($\pm$2.32)
∏ model [24]	5.43 ($\pm$0.25)	16.55 ($\pm$0.29)
(on Conv-Small used in [36])		
RPT	8.41 ($\pm$0.24)	18.56 ($\pm$0.29)
VAT	**6.83** ($\pm$0.24)	**14.87** ($\pm$0.13)
(on Conv-Large used in [24])		
VAT	5.77 ($\pm$0.32)	14.18 ($\pm$0.38)
VAT+EntMin	**4.28** ($\pm$0.10)	**13.15** ($\pm$0.21)

Without augmentation, (DGM=Deep Generative Models, FM=feature matching)

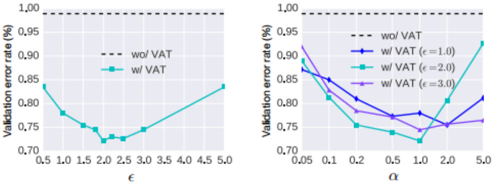
Method	Test error rate(%) SVHN $N_l = 1000$	CIFAR-10 $N_l = 4000$
∏ model [24].	4.82 ($\pm$0.17)	12.36 ($\pm$0.31)
Temporal ensembling [24]	4.42 ($\pm$0.16)	12.16 ($\pm$0.24)
Sajjadi et al. [35]		11.29 ($\pm$0.24)
(On Conv-Large used in [24])		
VAT	5.42 ($\pm$0.22)	11.36 ($\pm$0.34)
VAT+EntMin	**3.86** ($\pm$0.11)	**10.55** ($\pm$0.05)

With augmentation,(translation and horizontal flip)

Effects of Perturbation Size ϵ and Regularization Coefficient α

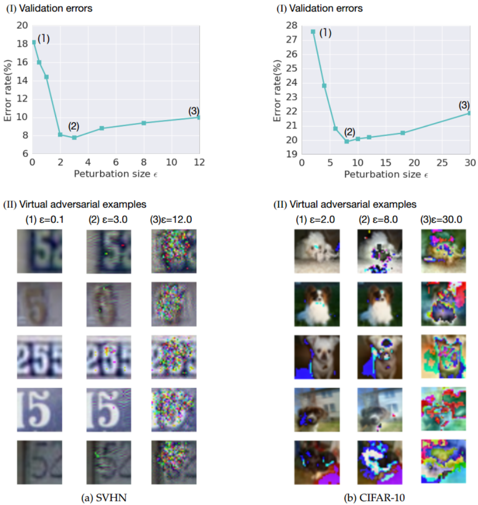
- For small ϵ, the hyper-parameter α plays a similar role as ϵ
- Parameter search for ϵ over the search for α

$$max_r\{D(r,x,\theta); \simeq \|r\|_2 \leq \epsilon\} \simeq max_r\{\frac{1}{2}r^T H(x,\theta)r; \|r\|_2 \leq \epsilon\} \\ \frac{1}{2}\epsilon^2\lambda_1(x,\theta) \tag{18}$$



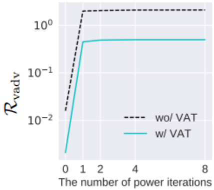
(a) Effect of ϵ ($\alpha = 1$). (b) Effect of α.

Effects of Perturbation Size ϵ

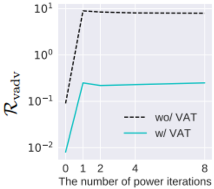


Effect of the Number of the Power Iterations K

- Power iteration method converges slowly if there is an eigenvalue close in magnitude to the dominant eigenvalue.
- Might depend on the spectrum of the Hessian matrix



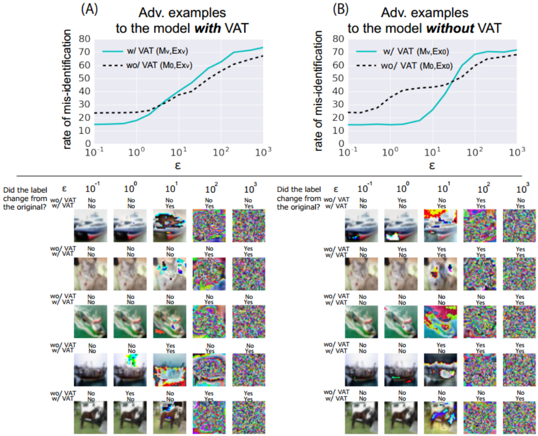
(a) MNIST



(b) CIFAR-10

Robustness of the VAT-trained Model Against Perturbed Images

- VAT-trained model behaves more natural than without VAT model.



VAT: Contributions

- Applicability to semi supervised learning tasks
- Applicability to any parametric models that we can calculate gradients w.r.t input and model parameters
- Small number of hyper-parameters
- Increase robustness against adversarial examples, acts more natural in different noise levels

More info

- Performances on semi-supervised image classification benchmarks
- Adversarial training methods for semi-supervised text classifications

Implementation

- Semi-supervised learning with VAT on SVHN
- 1000 labeled samples, 72,257 unlabeled samples, no data augmentation
- batch size for cross entropy loss:32, batch size for LDS: 128
- $\sim$ 48,000 updates in training
- ADAM optimization, base learning rate = 0.001, linearly decayed the rate over the last 16,000 updates
- $\alpha = 1$, $\epsilon = 2.5$, $\xi = 1e - 6$

Conv-Small on SVHN	Conv-Small on CIFAR-10	Conv-Large
32×32 RGB image		
3×3 conv. 64 lReLU 3×3 conv. 64 lReLU 3×3 conv. 64 lReLU	3×3 conv. 96 lReLU 3×3 conv. 96 lReLU 3×3 conv. 96 lReLU	3×3 conv. 128 lReLU 3×3 conv. 128 lReLU 3×3 conv. 128 lReLU
2×2 max-pool, stride 2 dropout, $p = 0.5$		
3×3 conv. 128 lReLU 3×3 conv. 128 lReLU 3×3 conv. 128 lReLU	3×3 conv. 192 lReLU 3×3 conv. 192 lReLU 3×3 conv. 192 lReLU	3×3 conv. 256 lReLU 3×3 conv. 256 lReLU 3×3 conv. 256 lReLU
2×2 max-pool, stride 2 dropout, $p = 0.5$		
3×3 conv. 128 lReLU 1×1 conv. 128 lReLU 1×1 conv. 128 lReLU	3×3 conv. 192 lReLU 1×1 conv. 192 lReLU 1×1 conv. 192 lReLU	3×3 conv. 512 lReLU 1×1 conv. 256 lReLU 1×1 conv. 128 lReLU
global average pool, 6×6 → 1×1		
dense 128 → 10	dense 192→ 10	dense 128→ 10
10-way softmax		