Distributional Smoothing with Virtual Adversarial Training

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, Shin Ishii (Kyoto Univ.)
Presenter:Takeru Miyato

Classification

Input : x

Model

$p(y|x,\theta)$

Output : y

“Cat”

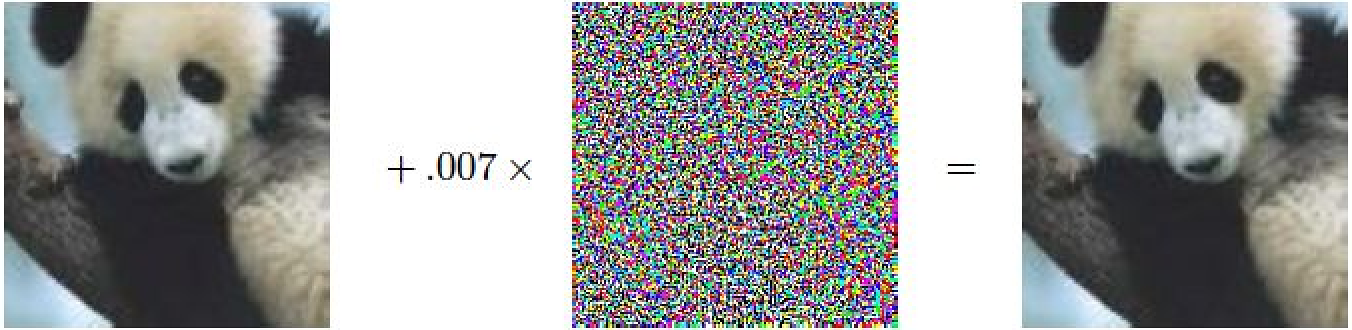
Purpose

- Get a good classifier
    - A good classifier : a classifier which can predict a label correctly on a example ***not included in training sets***.(汎化性能が高い)

- How to get a good classifier?
    - Data augmentation
    - Bayes inference
    - ***Regularization*** (正則化)<- I am working on this and propose a new method.

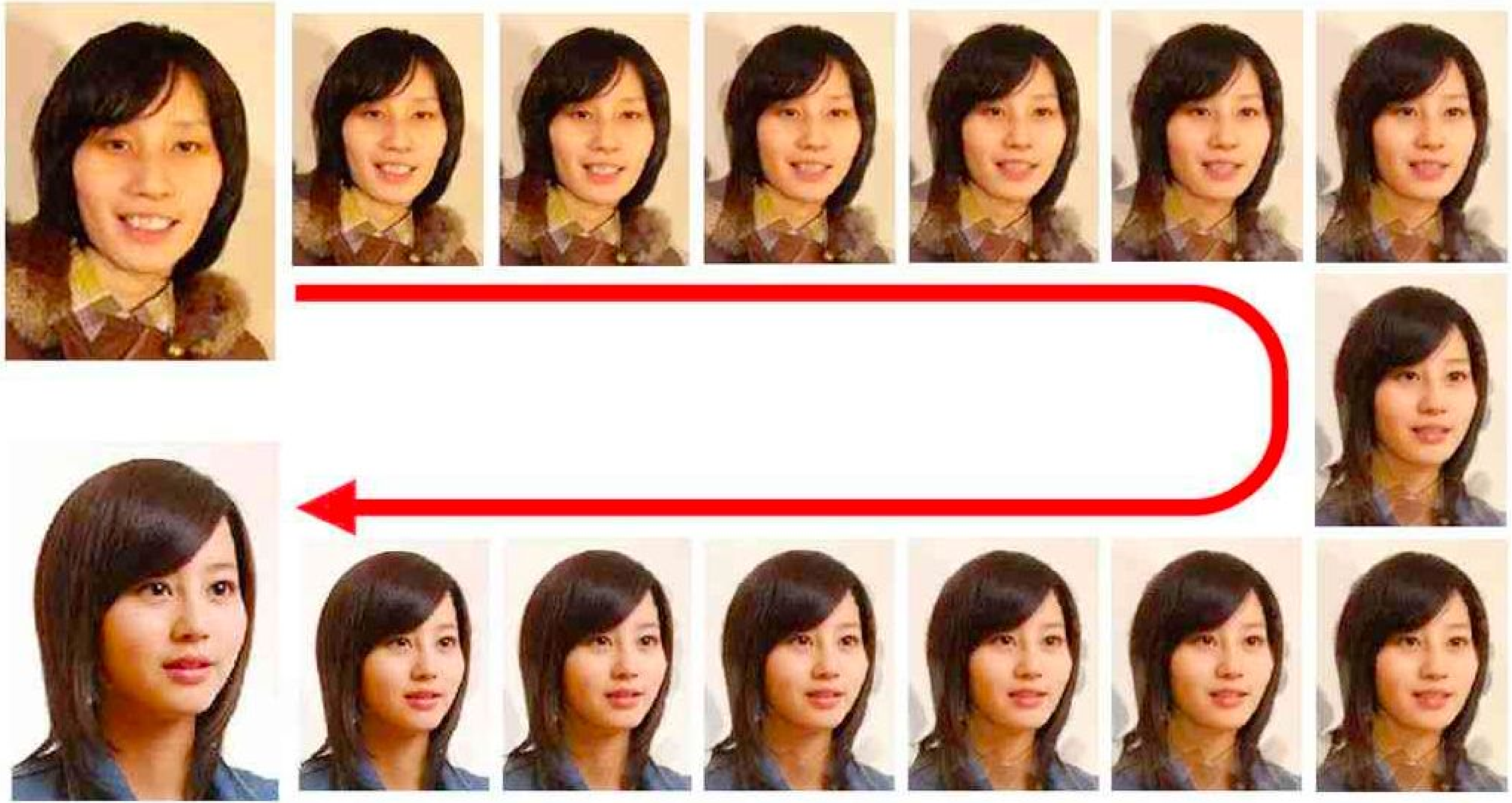
What are the properties of good classifier?

- This shall NOT happen!



[Goodfellow, et al 2015]

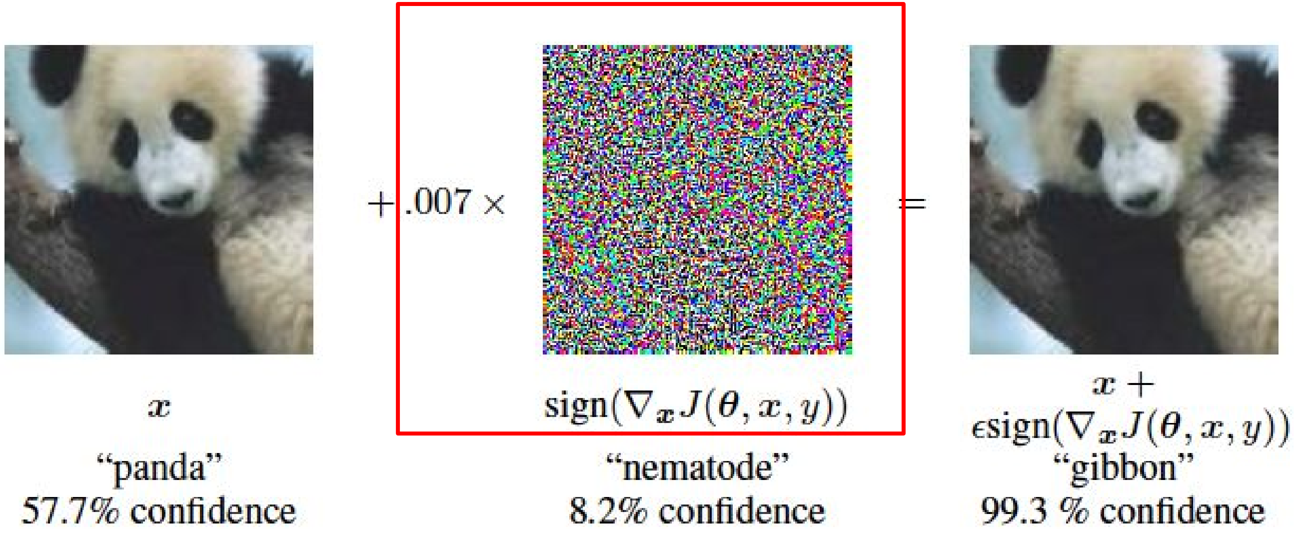
What are the properties of good classifier?



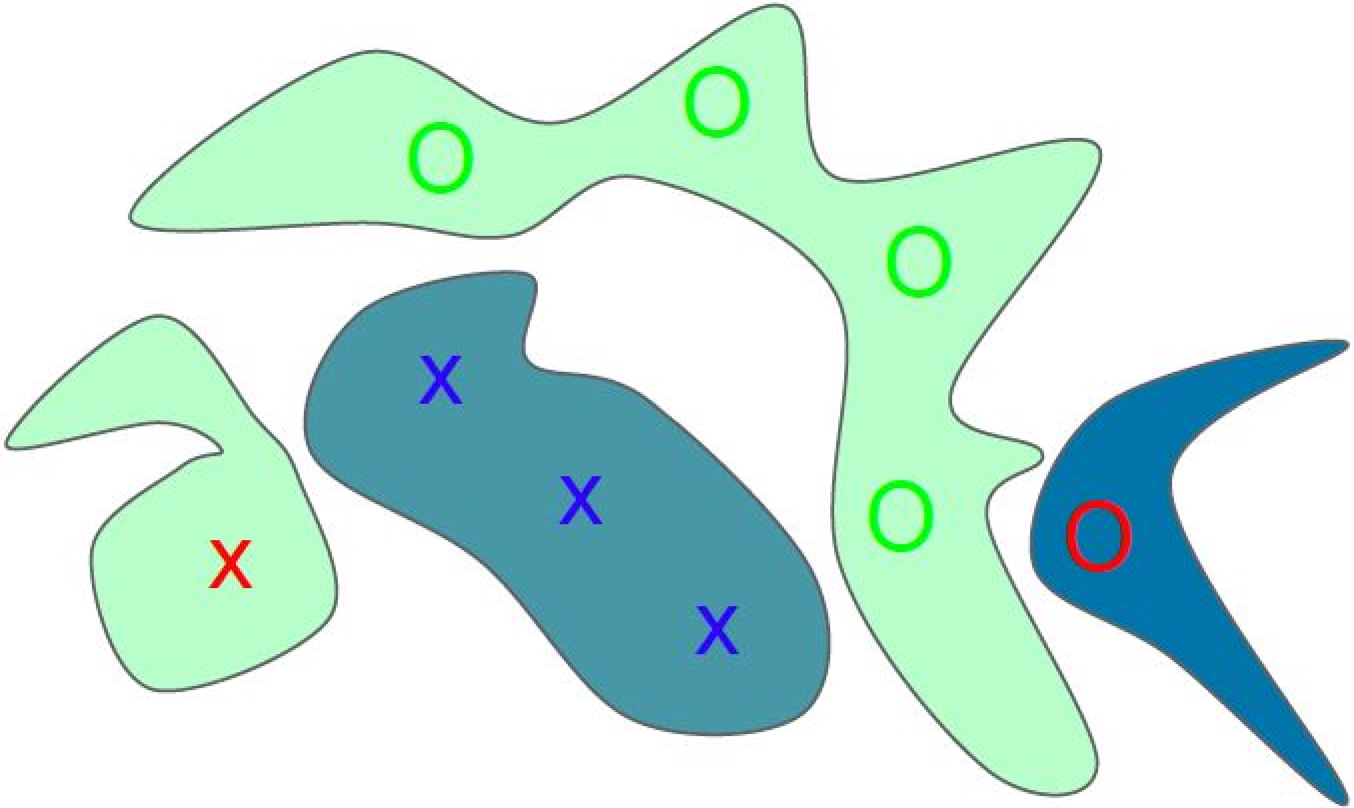
Adversarial training [Goodfellow, 2015]

Adversarial Perturbation :

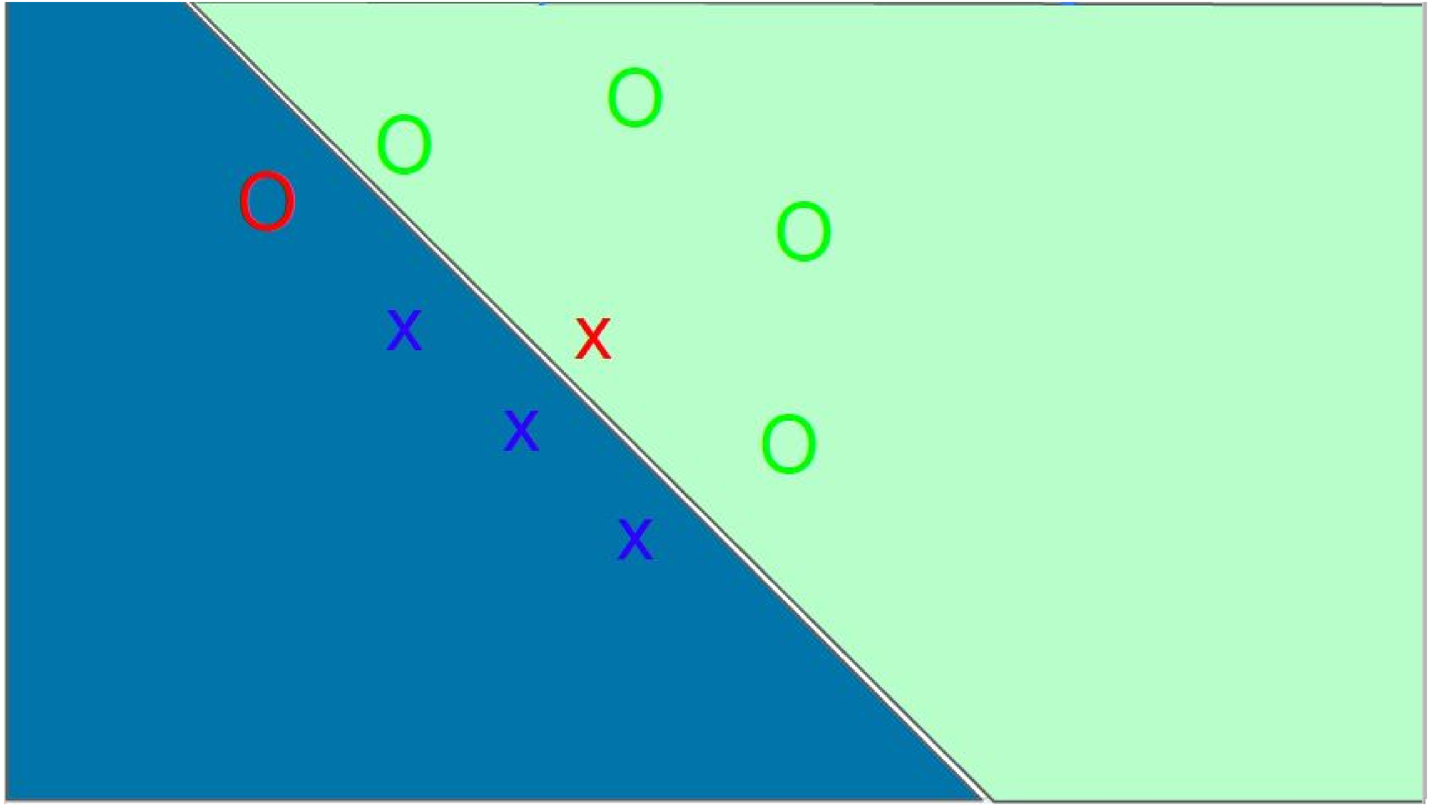
$$r_{\mathrm{adv}} = -\epsilon \operatorname{sign}(\nabla_x \log p(y|x,\theta))$$



Adversarial examples from overfitting



Adversarial examples from underfitting



[proposal] Adversarial examples for unlabeled

- Define the difference between $p(y|x,\theta)$ and $p(y|x+r,\theta)$ as:

$$\Delta_{\mathrm{KL}}(r, x^{(n)}, \theta) \equiv \mathrm{KL}[p(y|x^{(n)}, \theta) \| p(y|x^{(n)} + r, \theta)]$$

- Define *virtual* adversarial perturbation as :

$$r^{(n)}_{\text{v-adv}} \equiv \arg\max_{r} \{\Delta_{\mathrm{KL}}(r, x^{(n)}, \theta);\ \|r\|_2 \leq \epsilon\}$$

(ε : *norm constraint*)

Virtual Adversarial Training (VAT)

- Local Distributional Smoothness

$$\mathrm{LDS}(x^{(n)}, \theta) \quad \equiv \quad -\Delta_{\mathrm{KL}}(r_{\text{v-adv}}^{(n)}, x^{(n)}, \theta)$$

- Maximize:

$$\frac{1}{N}\sum_{n=1}^{N} \log p(y^{(n)}|x^{(n)}, \theta) + \lambda \frac{1}{M}\sum_{m=1}^{M} \mathrm{LDS}(x^{(m)}, \theta)$$

(λ: balance factor)

Computation of LDS

- We will need to compute

$$r_{\text{v-adv}} = \arg\max_{r} \{\Delta_{\mathrm{KL}}(r, x, \theta); \|r\|_2 \leq \epsilon\}$$

- For can be compute $r_{\text{v-adv}}$ fast?
    - Our method allows for fast approximation!

Approximation of $r_{v\text{-}adv}$

- Approximate Δ_{KL} with 2nd Taylor expansion:

$$\Delta_{\mathrm{KL}}(r, x, \theta) \cong \frac{1}{2} r^T H(x, \theta) r$$

$$H(x, \theta) \equiv \nabla\nabla_r \Delta_{\mathrm{KL}}(r, x, \theta)|_{r=0}$$

Approximation of $r_{\text{v-adv}}$

- In 2nd Taylor approximation, we see that $r_{v\text{-}adv}$ is the first dominant eigenvector of $H(x,\theta)$ of magnitude ε :

$$
\begin{aligned}
r_{\text{v-adv}}(x,\theta) &\cong \arg\max_{r}\{r^T H(x,\theta) r;\ \|r\|_2 \leq \epsilon\} \\
&= \overline{\epsilon u(x,\theta)},
\end{aligned}
$$

$u(x,\theta)$ is 1st eigen vector of $H(x,\theta)$

Approximation of $r_{\text{v-adv}}$

- Power iteration method :

$$d \leftarrow Hd$$

- Finite difference method :

$$Hd \cong \frac{\nabla_r \Delta_{\mathrm{KL}}(r + \xi d, x, \theta)|_{r=0} - \nabla_r \Delta_{KL}(r, x, \theta)|_{r=0}}{\xi}$$

$$= \frac{\nabla_r \Delta_{KL}(r + \xi d, x, \theta)|_{r=0}}{\xi},$$

Algorithm for the generation of approximated $r_{\text{v-adv}}$

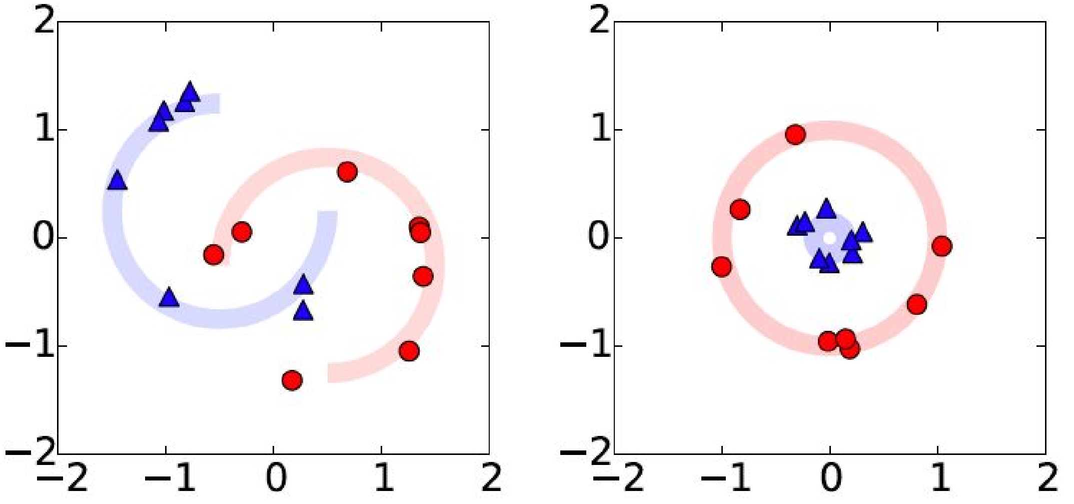
Algorithm 1 Generation of $\tilde{r}^{(n)}_{\text{v-adv}}$

Function GenVAP(θ,$x^{(n)}$,ϵ,I_p,ξ)

1. Initialize $d \in R^I$ by a random unit vector.
2. **Repeat For** i in $1 \dots I_p$ (Perform I_p-times power method)
   $$d \leftarrow \overline{\nabla_r \Delta_{KL}(r + \xi d, x^{(n)}, \theta)|_{r=0}}$$
3. **Return** ϵd

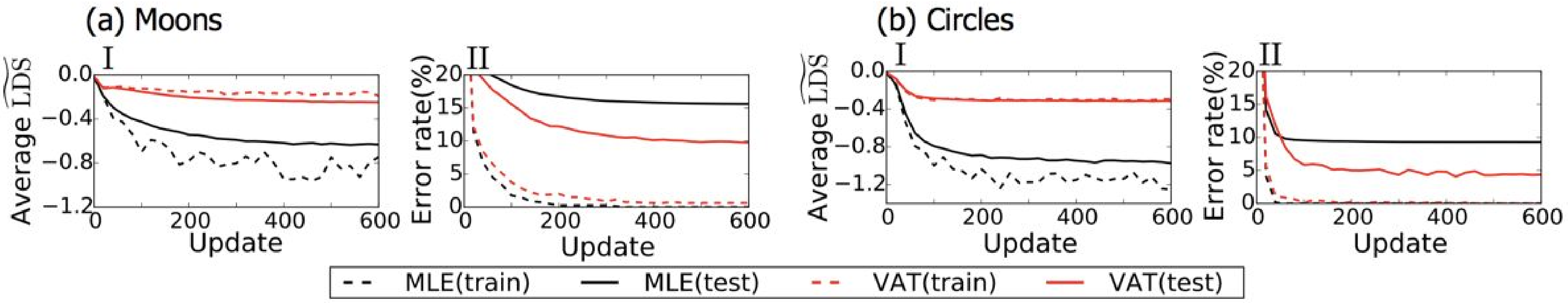
Demo : 2D synthetic dataset

- Use 1 hidden layer NN.
- Use gradient descent with Momentum.



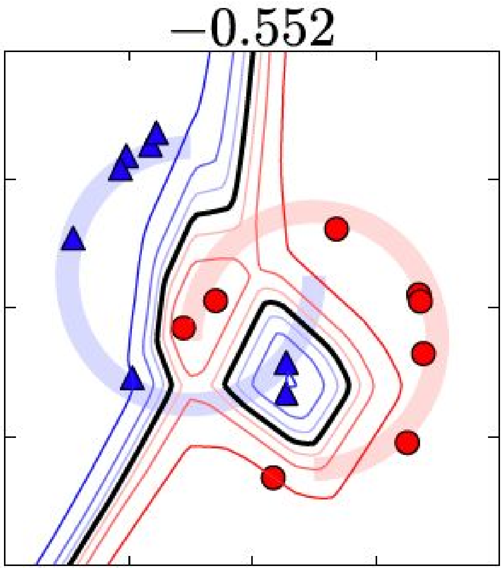
(a) Moons dataset (b) Circles dataset

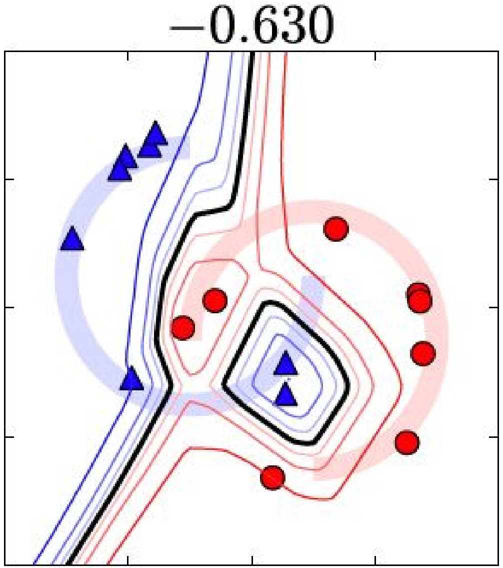
Learning curves



Visualization of $p(y|x,\theta)$, MLE and L2

average LDS

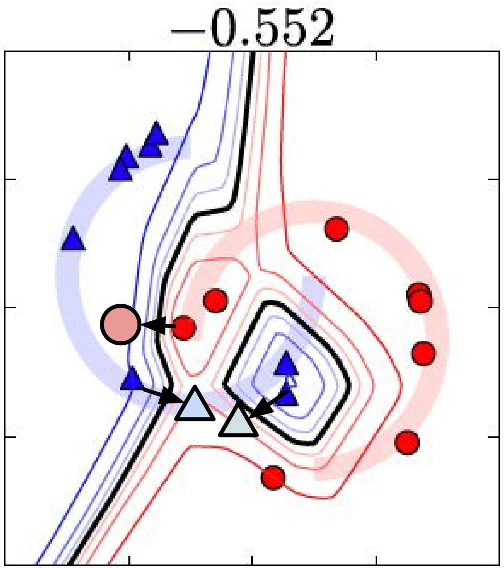


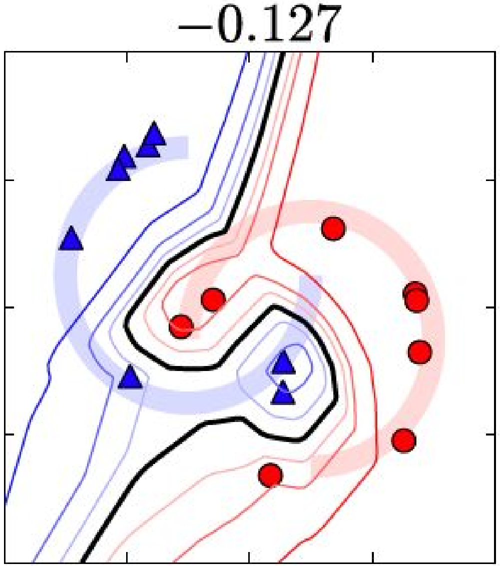


Method	no regularization	L_2 regularization

Visualization of $p(y|x,\theta)$, MLE and VAT(proposed)

average LDS





Method

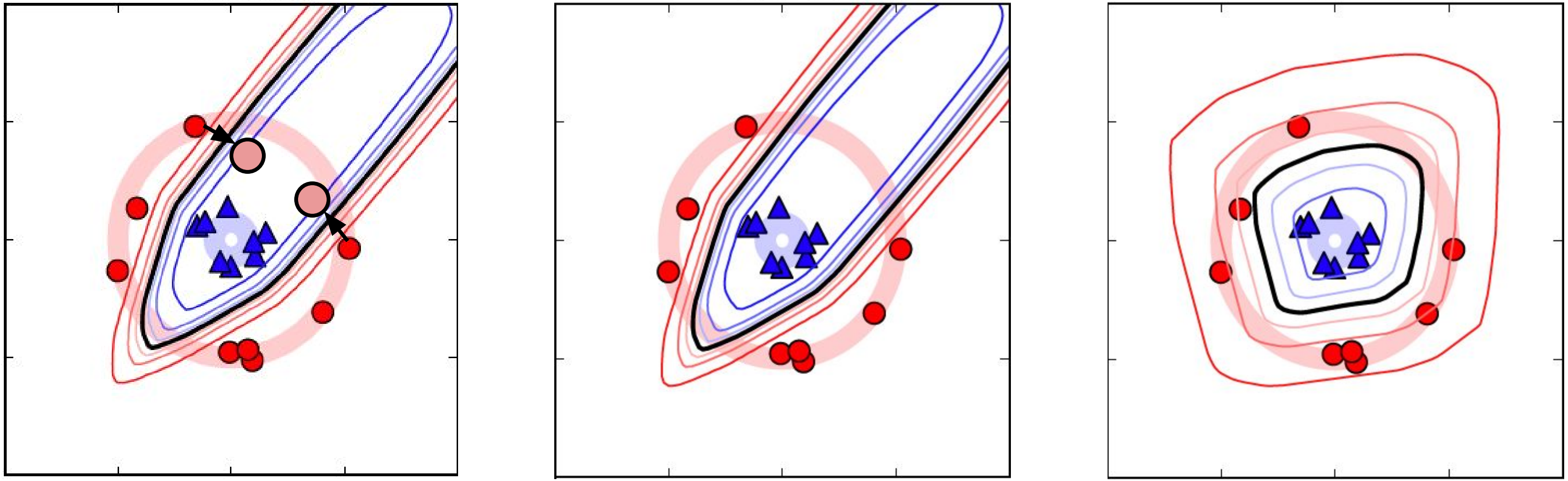
no regularization

VAT (proposed)

Visualization of $p(y|x,\theta)$

average LDS

-1.588 -1.351 -0.257



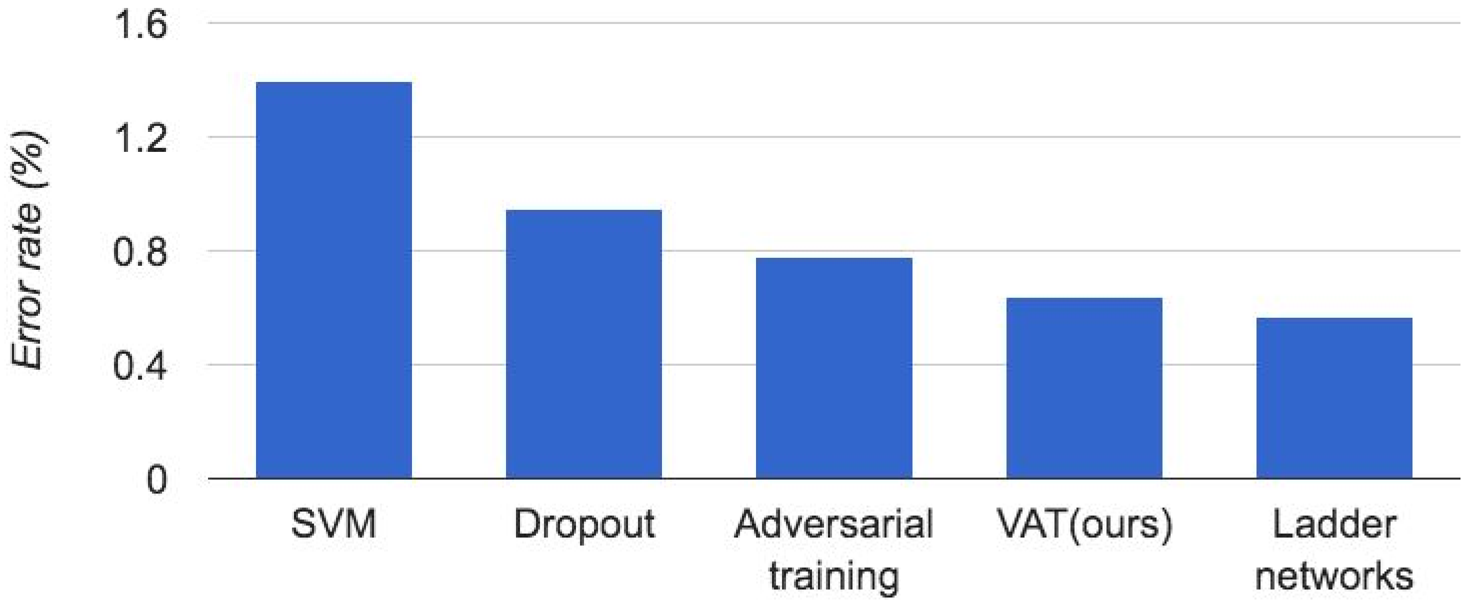
Method: no regularization | L_2 regularization | VAT (proposed)

Benchmark results : Supervised learning for MNIST

- MNIST (permutation invariant task)
  - 60000 training samples
  - Use ADAM$_{\text{[Kingma, 2015]}}$ optimizer

Supervised learining for MNIST

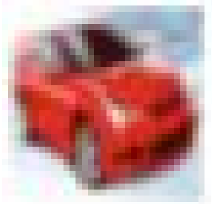
- MNIST (permuation invariant task)



Semi-supervised learning (permutation invariant task)

- MNIST

- CIFAR10
  - 4000 labeled and 50000 unlabeled

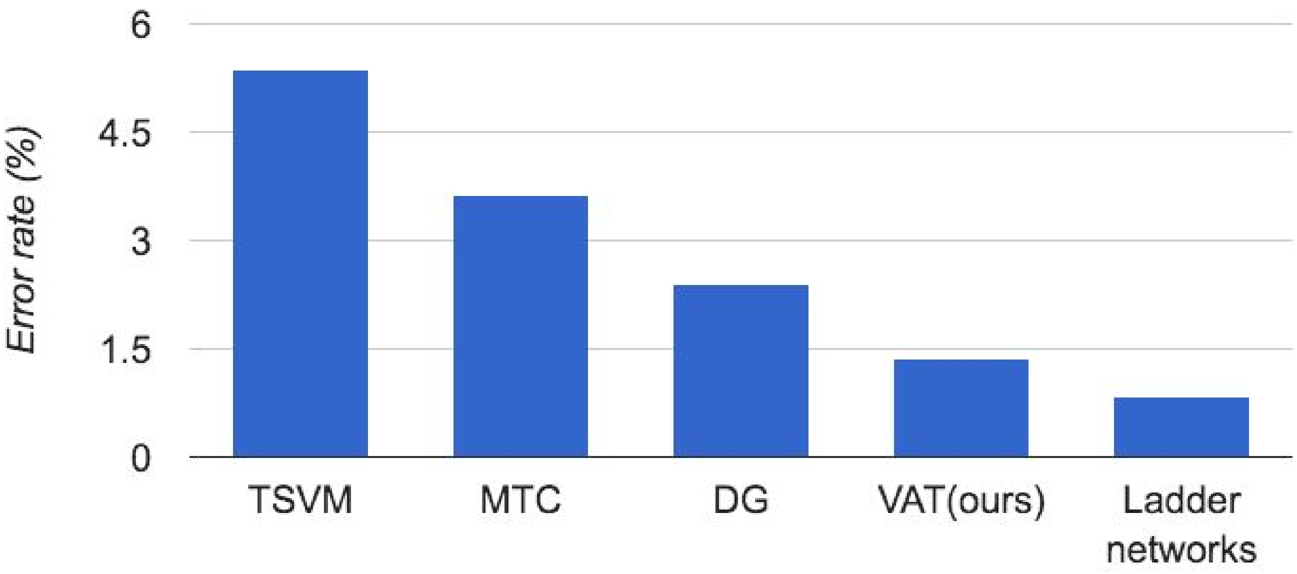


- SVHN (street view housing numbers)
  - 1000 labeled and 70000 unlabeled

Semi-supervised learning (permutation invariant task)

- MNIST (1000 labeled samples)

TSVM:
 Transductive SVM
MTC:
 Manifold Tangent Classifier
DG:
 Deep Generative Model



Semi-supervised learning on CIFAR10

- CIFAR10 (4000 labeled)

Table 2: CIFAR10 with 4000 labeled examples.

Method	Test error rate
Ladder network (Rasmus et al., 2015)	20.40%
CatGAN (Springenberg, 2015)	19.58%
GAN with feature matching (Salimans et al., 2016)	18.63%
Baseline	23.71%
Virtual Adversarial Training (ours)	**17.60%**

Semi-supervised learning on SVHN

Table 3: SVHN with 1000 labeled examples.

Method	Test error rate
TSVM (Kingma et al., 2014)	66.55 %
SWWAE (Zhao et al., 2015)	23.56 %
Skip Generative Model (Maaløe et al., 2016)	16.30 %
GAN with feature matching (Salimans et al., 2016)	8.11%
Baseline	11.66%
Virtual Adversarial Training (ours)	**7.09%**

Conclusion

- Our approach was effective for supervised and semi-supervised learning for benchmark datasets.
- With optimizing **only 1 hyperparameter** ε , our method achieved good performance.

References

- T. Miyato, S. Maeda, M. Koyama, K. Nakae and S. Ishii. *Distributional Smoothing with Virtual Adversarial Training*. ICLR 2016.
  - arXiv:http://arxiv.org/abs/1507.00677
  - github:https://github.com/takerum/vat
- I. Goodfellow, J. Shlens and C. Szegedy. *Explaining and Harnessing Adversarial Examples.* ICLR 2015.