Survey on adversarial training

Seongok Ryu

ACE-Team, Department of Chemistry, KAIST

List of contents

Generative models

Adversarial training as regularization methods

KAIST CHEMISTRY

Preliminaries

KAIST

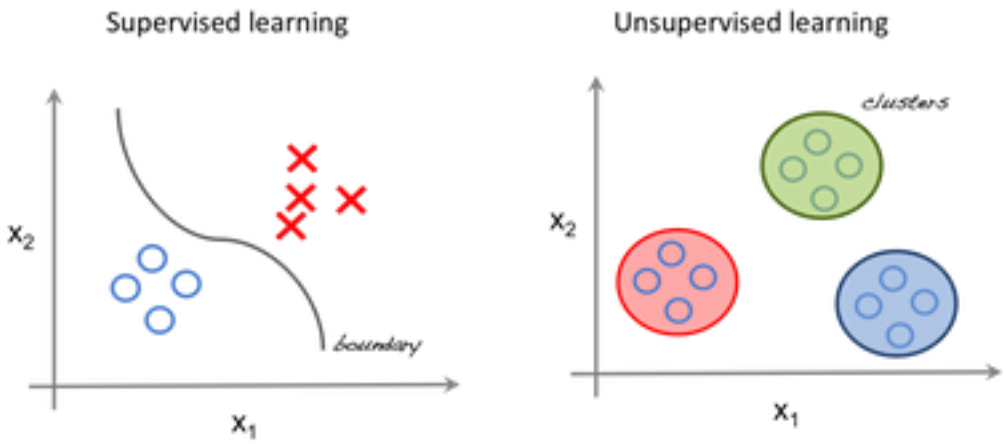
Preliminaries

- **Supervised learning** infers a function from labeled training data

$$p(\theta|\mathrm{X},\mathrm{Y}) \propto p(\mathrm{Y}|\mathrm{X},\theta)p(\theta)$$

- **Unsupervised learning** infers a function without labeled training data

$$p(\theta|\mathrm{X}) \propto p(\mathrm{X}|\theta)p(\theta)$$



Preliminaries

- **Supervised learning** infers a function from labeled training data

$$p(\theta|\mathrm{X},\mathrm{Y}) \propto p(\mathrm{Y}|\mathrm{X},\theta)p(\theta)$$

- **Unsupervised learning** infers a function without labeled training data

$$p(\theta|\mathrm{X}) \propto p(\mathrm{X}|\theta)p(\theta)$$

- **Generative methods** try to *model the class-conditional density* $p(x|y)$ by some unsupervised learning procedure. A predictive density can then be inferred by Bayes' theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{\int_y p(x|y)p(y)dy}$$

- **Discriminative methods** concentrate on estimating $p(y|x)$.

Preliminaries

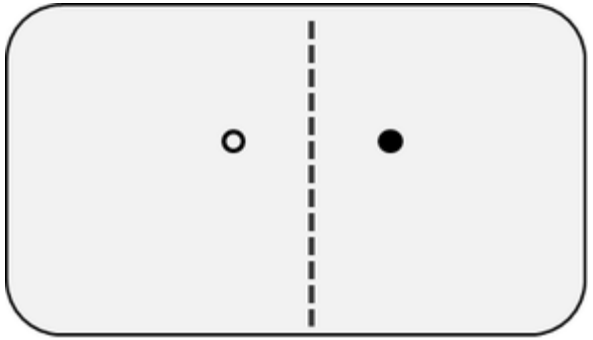
- **Supervised learning** infers a function from labeled training data

$$p(\theta|\mathrm{X},\mathrm{Y}) \propto p(\mathrm{Y}|\mathrm{X},\theta)p(\theta)$$

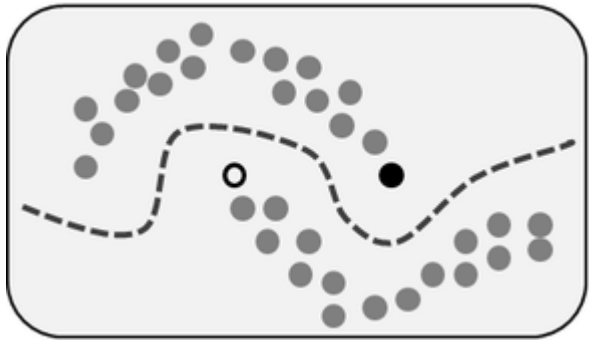
- **Unsupervised learning** infers a function without labeled training data

$$p(\theta|\mathrm{X}) \propto p(\mathrm{X}|\theta)p(\theta)$$

- **Semi-supervised learning** infers a function with both labeled and unlabeled training data



Supervised learning



Semi-supervised learning

https://en.wikipedia.org/wiki/Semi-supervised_learning

Preliminaries

Recommend to see http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture13.pdf

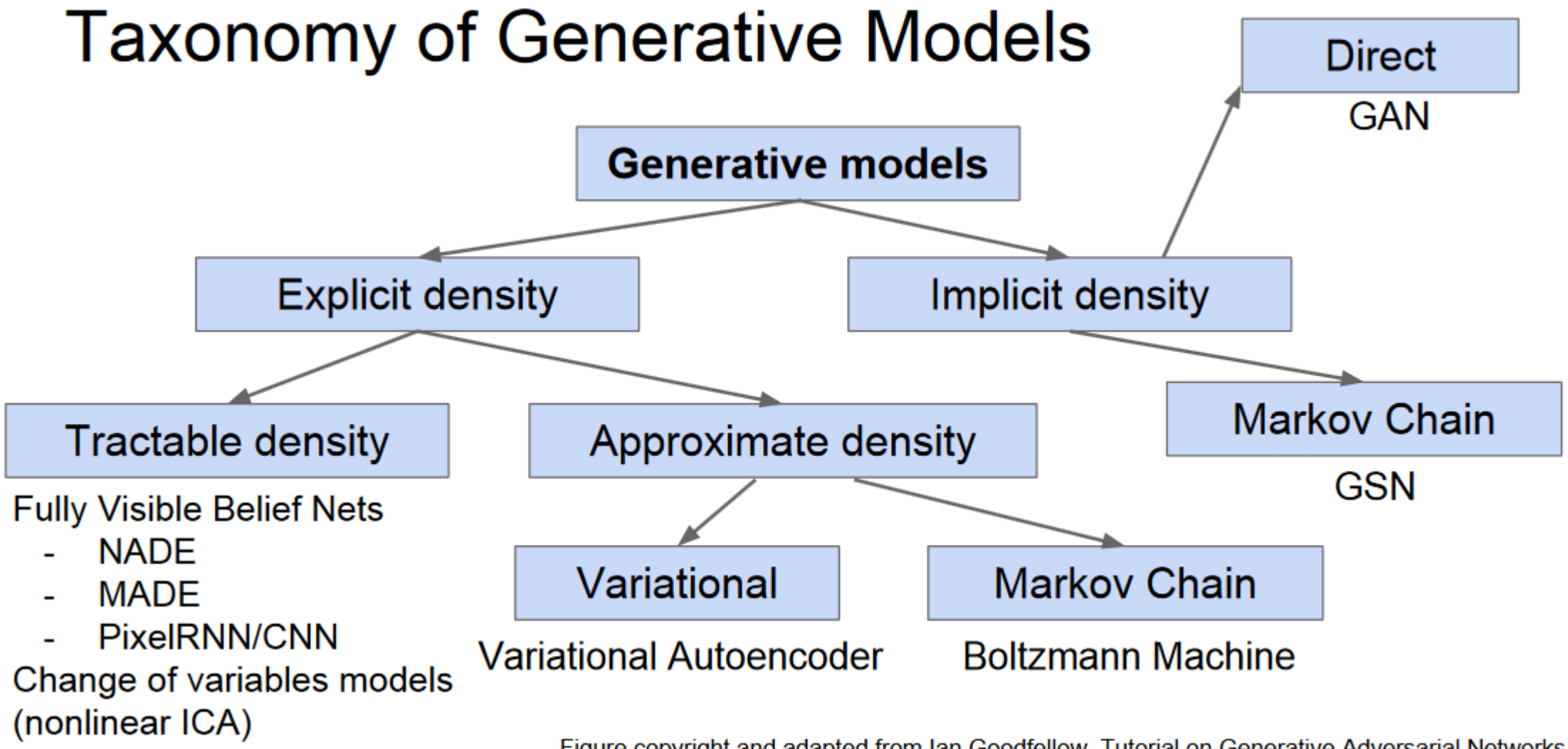


Figure copyright and adapted from Ian Goodfellow, Tutorial on Generative Adversarial Networks, 2017.

Goodfellow, Ian. "NIPS 2016 tutorial: Generative adversarial networks." *arXiv preprint arXiv:1701.00160* (2016).

KAIST CHEMISTRY

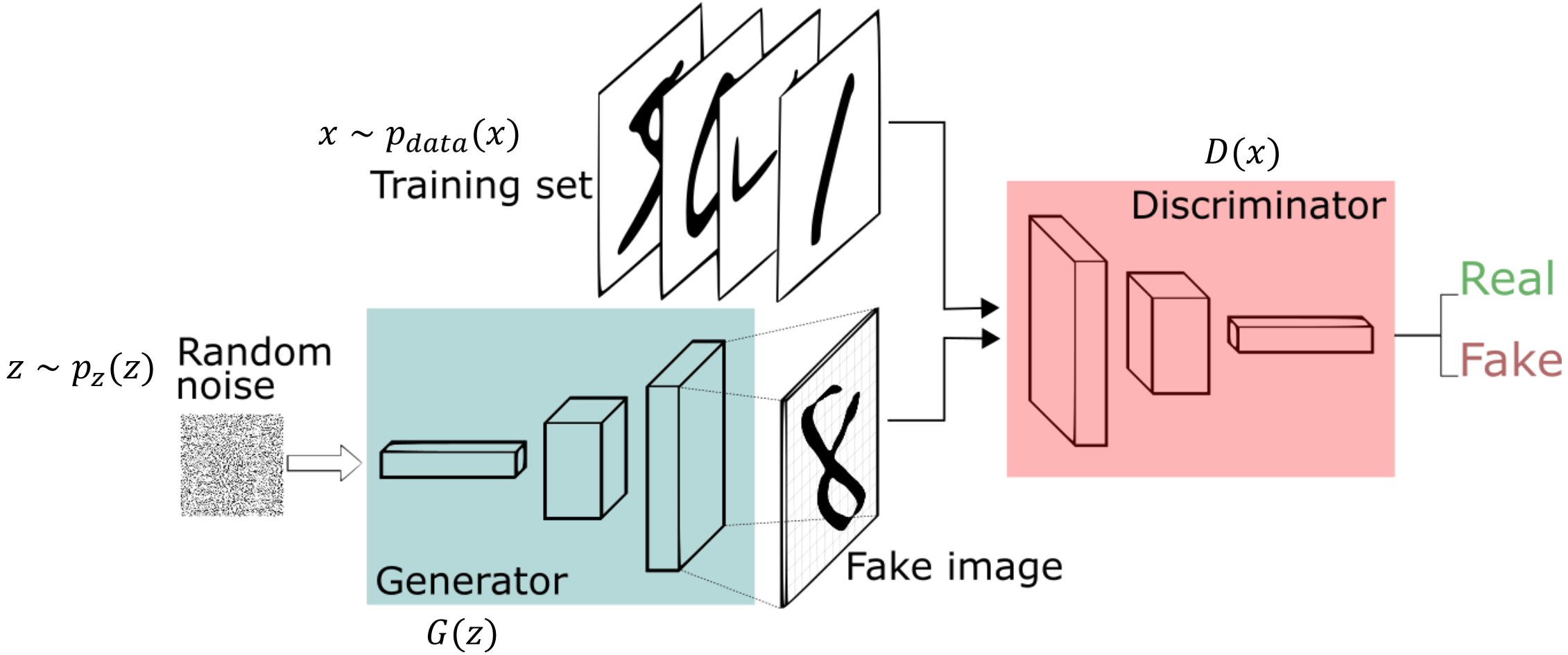
Generative Adversarial Network

KAIST

Generative Adversarial Network

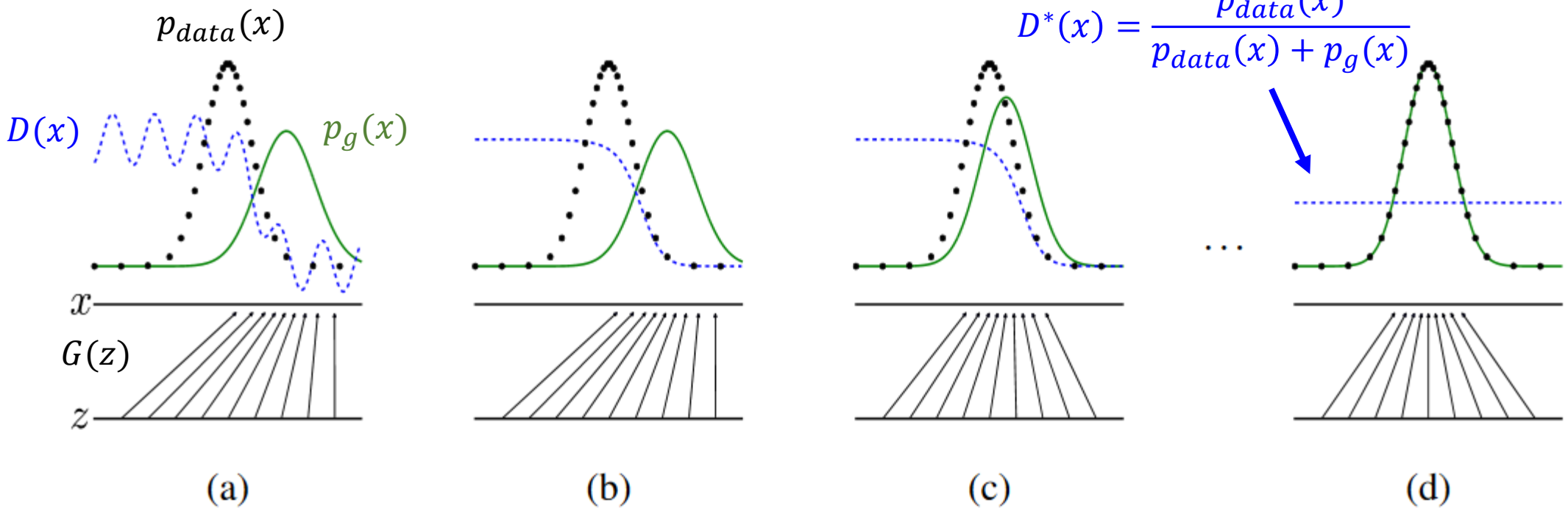
Learning objective:

$$\min_G \max_D V(D,G), \qquad V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))]$$



Goodfellow, Ian, et al. "Generative adversarial nets."
Advances in neural information processing systems. 2014.

Generative Adversarial Network



Learning objective:

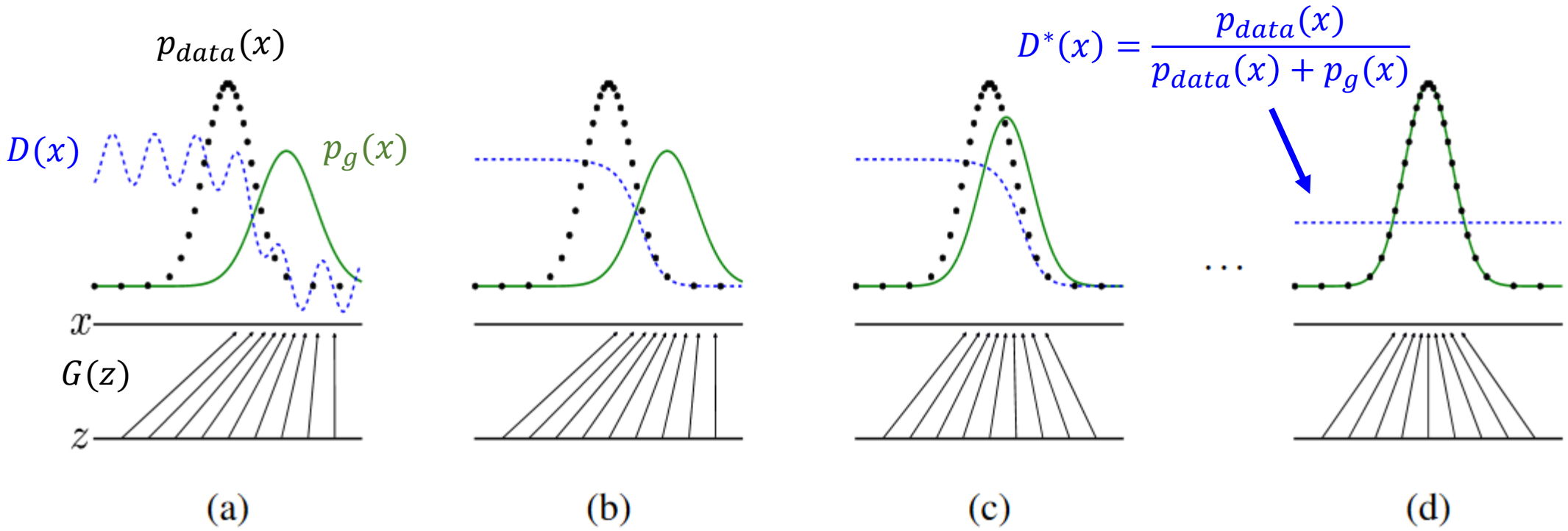
$$\min_G \max_D V(D,G), \qquad V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z))]$$

- ✓ For real images, $D(x)$ tries to be near 1.
- ✓ For fake images, D tries to make $D\big(G(z)\big)$ near 0 and G tries to make $D\big(G(z)\big)$ near 1.

→ so-called ***mini-max two player game***

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Generative Adversarial Network



For G fixed, the optimal discriminator D is $D^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$, since,

$$V(G,D) = \int_x p_{data}(x)\log(D(x))dx + \int_z p_z(z)\log\left(1 - D\left(G(z)\right)\right)dz$$

$$= \int_x p_{data}(x)\log(D(x)) + p_g(x)\log\left(1 - D(x)\right)dx$$

has its minimum in [0,1] at $\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}$.

Goodfellow, Ian, et al. "Generative adversarial nets."
Advances in neural information processing systems. 2014.11

Generative Adversarial Network

Learning objective:

$$\min_G \max_D V(D,G), \qquad V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z))]$$

The mini-max game in the above learning objective can now be reformulated as

$$C(G) = \max_D V(G,D) = \mathbb{E}_{x\sim p_{data}}[\log D_G^*(x)] + \mathbb{E}_{x\sim p_g}\left[\log\left(1-D_G^*(x)\right)\right]$$

$$= \mathbb{E}_{x\sim p_{data}}\left[\log\frac{p_{data}(x)}{p_{data}(x)+p_g(x)}\right] + \mathbb{E}_{x\sim p_g}\left[\log\frac{p_g(x)}{p_{data}(x)+p_g(x)}\right]$$

Theorem) The global minimum of the virtual training criterion $C(G)$ is achieved if and only if $p_g(x) = p_{data}(x)$. At that point, $C(G)$ achieves the value $-\log 4$.

Proof)

$$C(G) = \mathrm{KL}\left(p_{data}\|p_{data}+p_g\right) + \mathrm{KL}\left(p_g\|p_{data}+p_g\right) + \log 4 - \log 4$$

$$= \mathrm{KL}\left(p_{data}\left\|(p_{data}+p_g)/2\right.\right) + \mathrm{KL}\left(p_g\|(p_{data}+p_g)/2\right) - \log 4$$

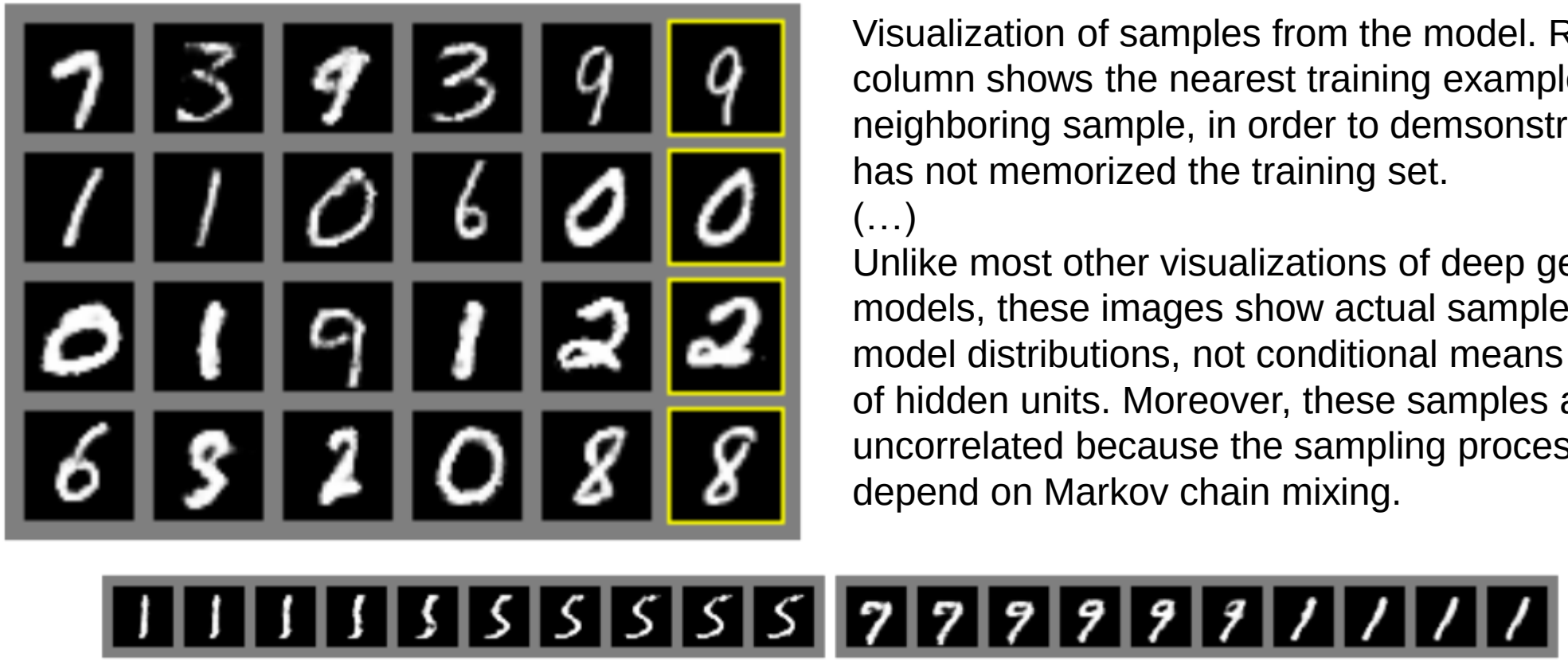
$$= \mathrm{JS}(p_{data}\|p_g) - \log 4$$

$\mathrm{JS}(p_{data}\|p_g)$ is always non-negative and is zero when $p_{data} = p_g$. Therefore, $C(G)$ has its minimum value $-\log 4$ when $p_{data} = p_g$.

Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Generative Adversarial Network

Results



Visualization of samples from the model. Rightmost column shows the nearest training example of the neighboring sample, in order to demsonstrate that model has not memorized the training set.
(…)
Unlike most other visualizations of deep generative models, these images show actual samples from the model distributions, not conditional means given samples of hidden units. Moreover, these samples are uncorrelated because the sampling process does not depend on Markov chain mixing.

Figure 3: Digits obtained by linearly interpolating between coordinates in z space of the full model.

Goodfellow, Ian, et al. "Generative adversarial nets."
Advances in neural information processing systems. 2014.

Generative Adversarial Network

- "In practice, **adversarial nets represent a limited family of p_g distributions via the function $G(z;\theta_g)$, and we optimize θ_g rather than p_g itself, so the proofs do not apply.**

→ **Implicit density model** (recommend to see how VAE infers the posterior $p(z|x)$ and to compare GAN and VAE)

- However, the excellent performance of multi-layer perceptrons in practice suggests that they are a reasonable model to use despite their lack of theoretical guarantees."

Recommend to see more detailed review of GAN
→ https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks

Goodfellow, Ian, et al. "Generative adversarial nets."
Advances in neural information processing systems. 2014.

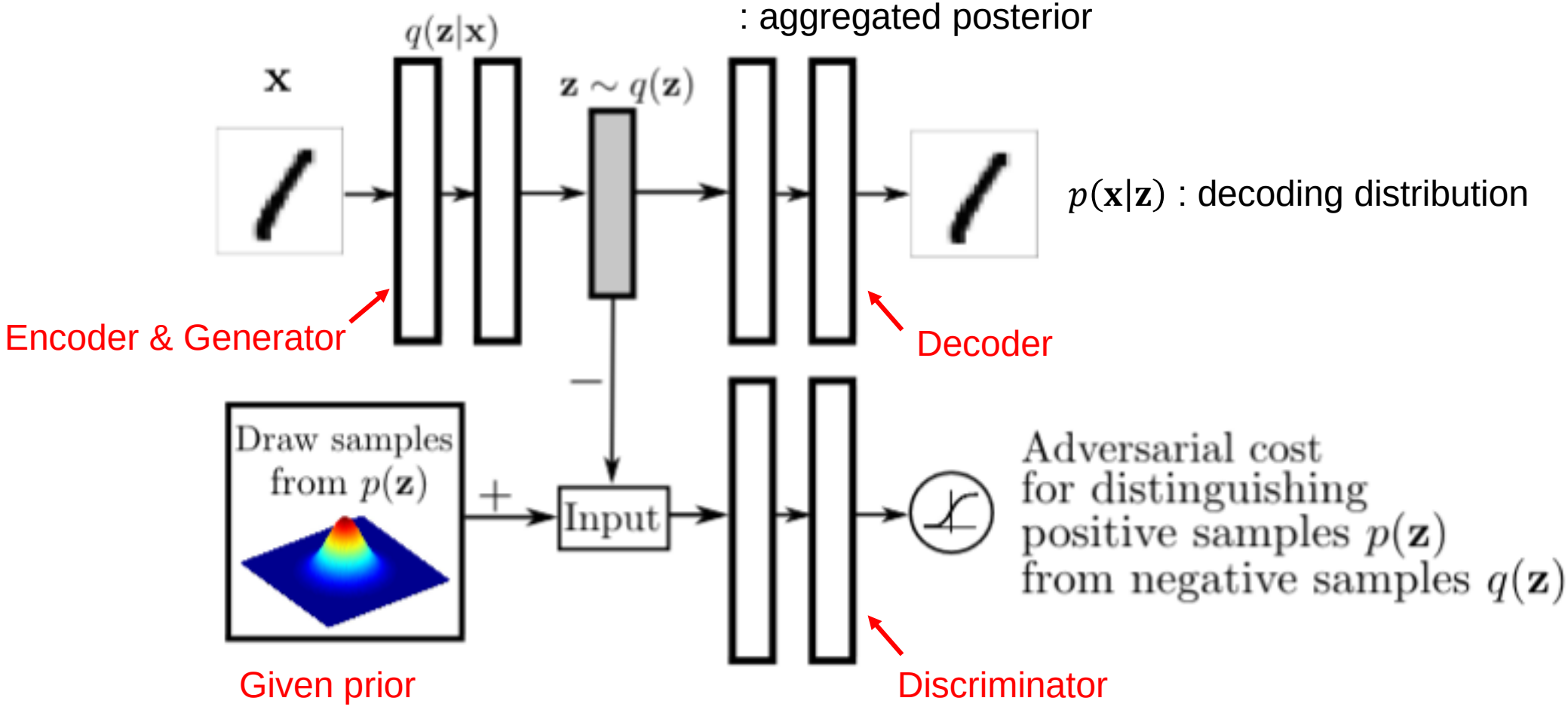
KAIST CHEMISTRY

Adversarial Autoencoder

KAIST

Adversarial Autoencoder

$q(\mathbf{z}|\mathbf{x})$: encoding distribution $q(\mathbf{z}) = \int_{\mathbf{x}\in\mathbf{X}} q(\mathbf{z}|\mathbf{x})p_d(\mathbf{x})d\mathbf{x}$: aggregated posterior $p_d(\mathbf{x})$: data distribution



$p(\mathbf{z})$: prior distribution

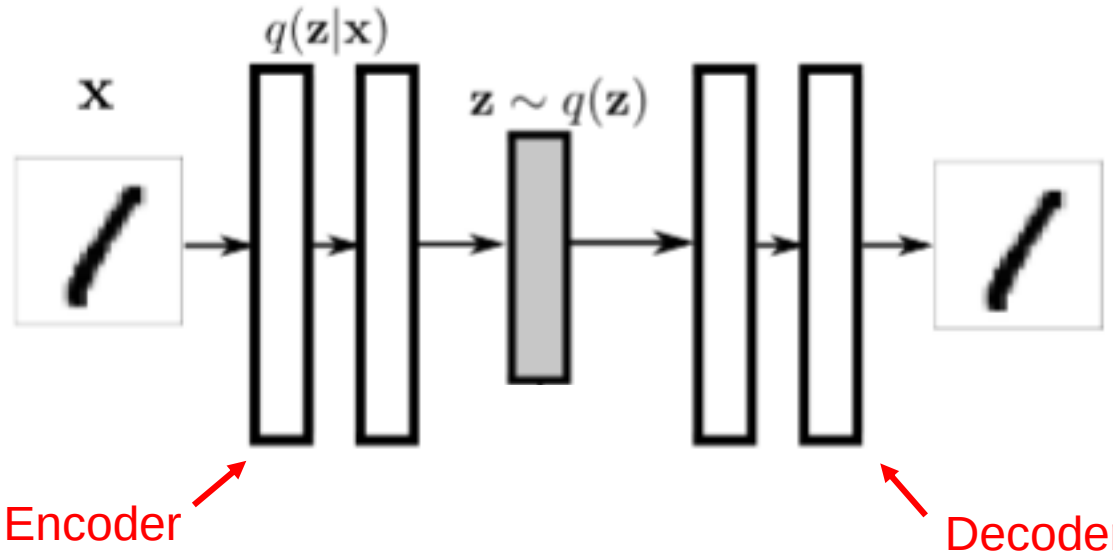
Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015). 16

Adversarial Autoencoder

Learning objective:

- In the ***reconstruction*** phase, the autoencoder updates the encoder and the decoder ***to minimize the reconstruction error of inputs***.

$$\mathcal{L}_{reconst} = \frac{1}{\mathcal{D}_{data}} |\mathbf{x} - \hat{\mathbf{x}}|^2$$



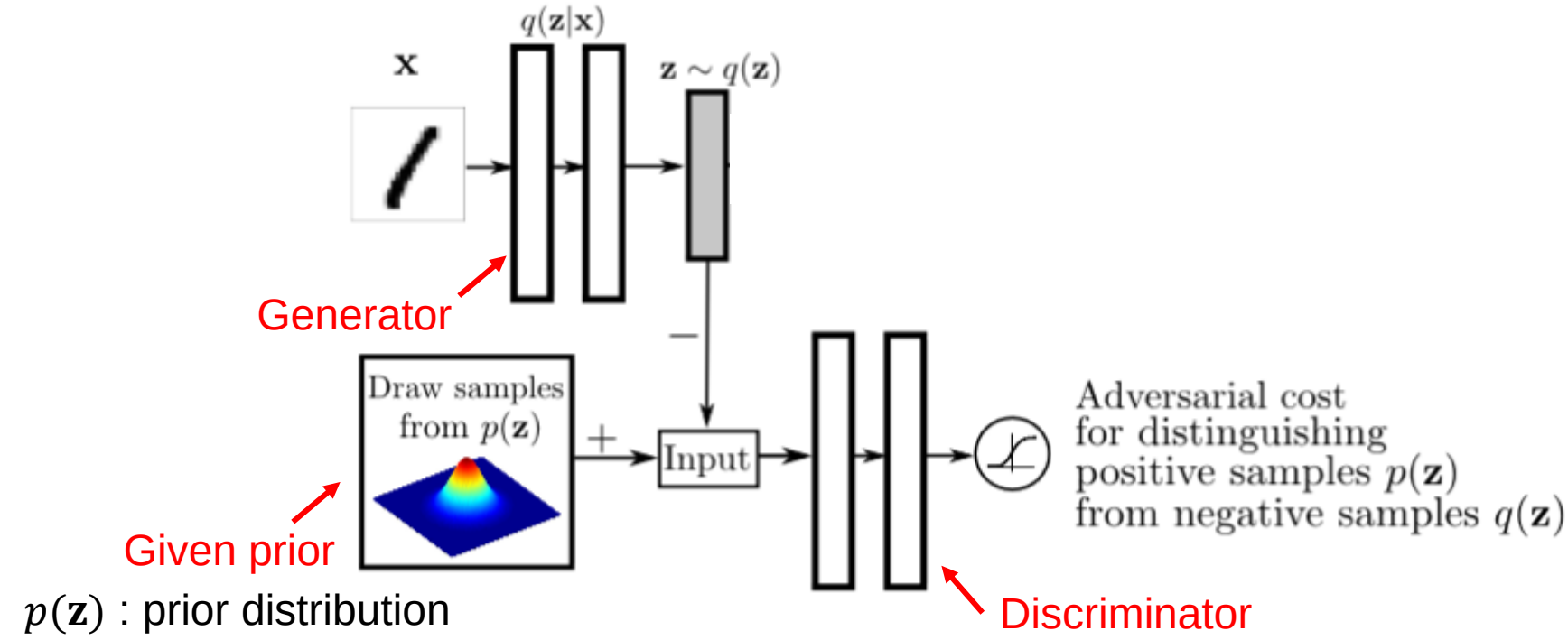
Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Learning objective:

- In the ***regularization*** phase, the adversarial network first ***updates its discriminative network to tell apart the true samples (generated using the prior) from the generated samples (the hidden codes computed by the autoencoder)***. The adversarial network then ***updates its generator (which is also the encoder of the autoencoder) to confuse the discriminative network***.

$$\min_G \max_D V(D,G), \qquad V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(\mathbf{z}))]$$



$p(\mathbf{z})$: prior distribution

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Learning objective:

1) In the ***reconstruction*** phase, the autoencoder updates the encoder and the decoder ***to minimize the reconstruction error of inputs***.

$$\mathcal{L}_{reconst} = \mathbb{E}_{p_d(\mathbf{x})}\left[\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}|\mathbf{z})]\right]$$

2) In the ***regularization*** phase, the adversarial network first ***updates its discriminative network to tell apart the true samples (generated using the prior) from the generated samples (the hidden codes computed by the autoencoder)***. The adversarial network then ***updates its generator (which is also the encoder of the autoencoder) to confuse the discriminative network***.

$$\min_G \max_D V(D,G), \qquad V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[\log D(\mathbf{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(\mathrm{z}))]$$

→ The adversarial network guides the posterior distribution $q(\mathbf{z})$ to match the prior distribution $p(\mathbf{z})$

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Comparison to variational autoencoder (Kingma *et. al.*, 2014):

* ELBO : Evidence Lower BOund

Learning objective of VAE:

$$\mathcal{L} = \underbrace{\mathbb{E}_{p_d(\mathbf{x})}\left[\mathbb{E}_{q(\mathbf{z}|\mathbf{x})}[-\log p(\mathbf{x}|\mathbf{z})]\right]}_{\mathcal{L}_{reconst}} + \underbrace{\mathbb{E}_{p_d(\mathbf{x})}[\mathrm{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))]}_{\text{KL-regularization}}$$

This term is replaced by adversarial loss in AAE

Pros and Cons of using AAE instead of VAE

https://duvenaud.github.io/learn-discrete/slides/AdversarialAutoencoders.pdf

Pros)

- Do not need to set the functional form of the prior distribution and reparameterization trick
- We just need to be able to sample from the prior to induce the latent distribution to match the prior

Cons)

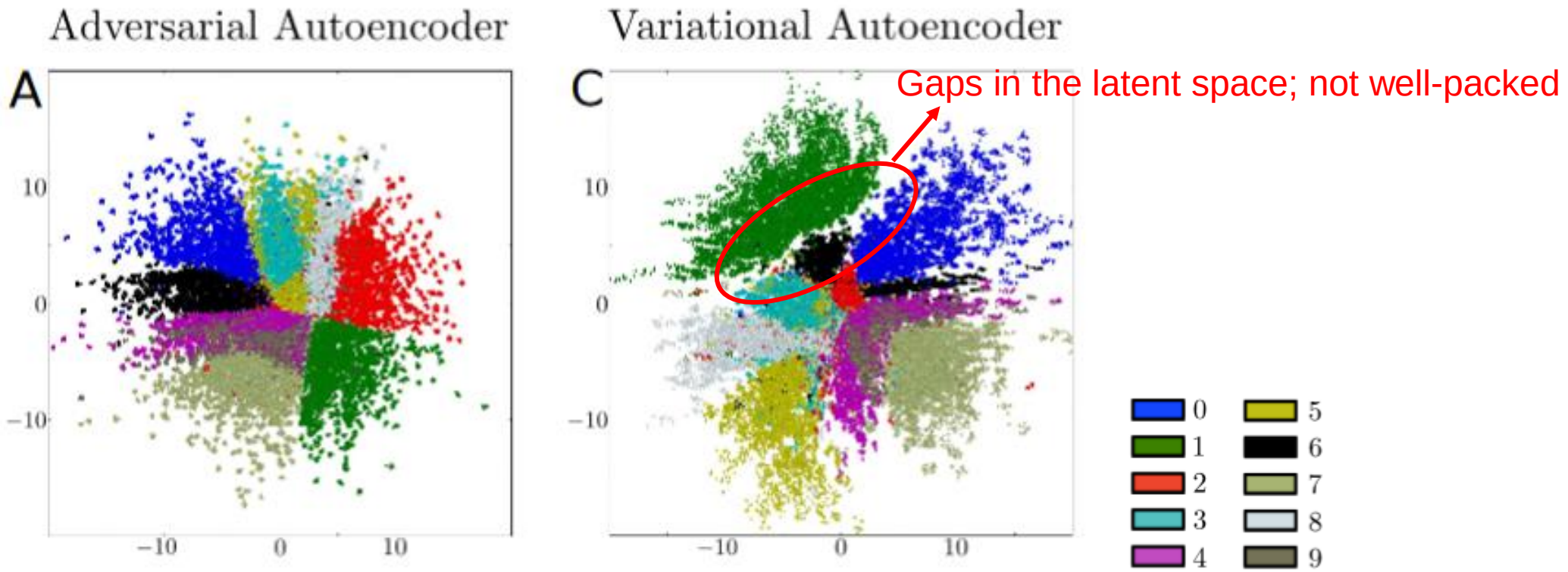
- Challenging to train due to common difficulties inherent in GANs

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Comparison to variational autoencoder (Kingma *et. al.*, 2014):

The hidden code z of the hold-out images for an AAE/VAE fit to a ***2D Gaussian***



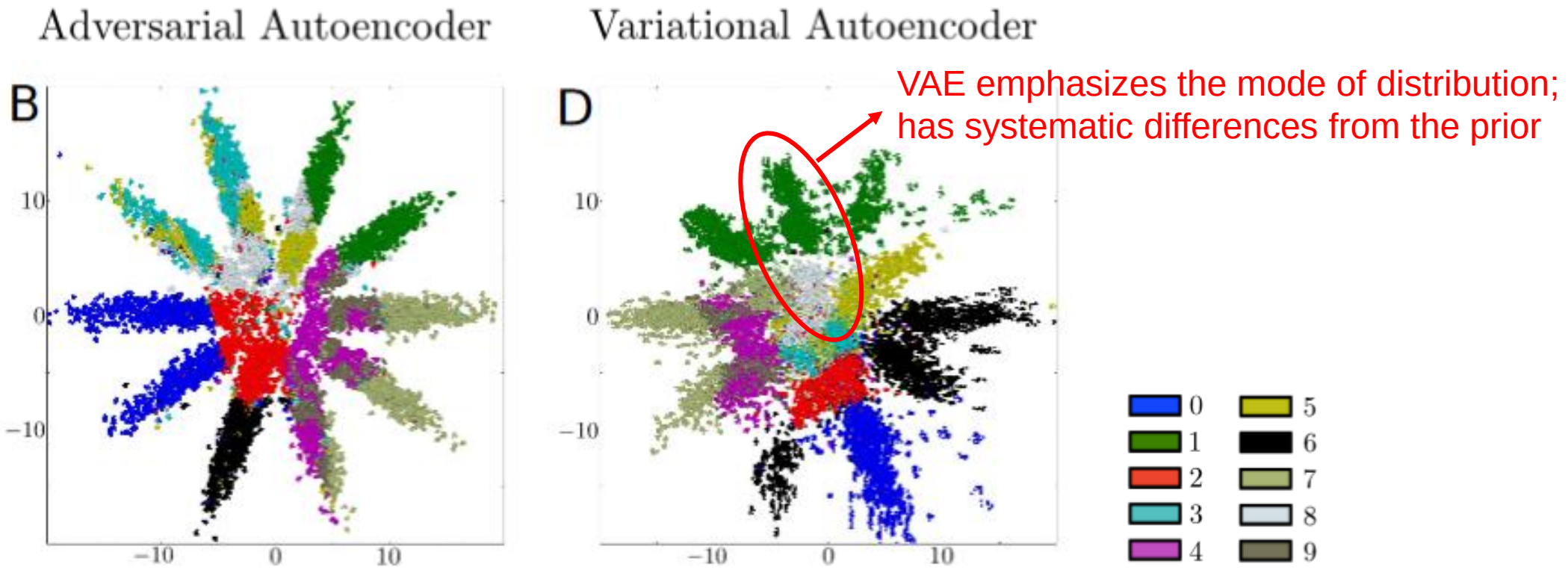
https://duvenaud.github.io/learn-discrete/slides/AdversarialAutoencoders.pdf

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Comparison to variational autoencoder (Kingma *et. al.*, 2014):

The hidden code z of the hold-out images for an AAE/VAE fit to ***a mixture of 10 2D Gaussian***



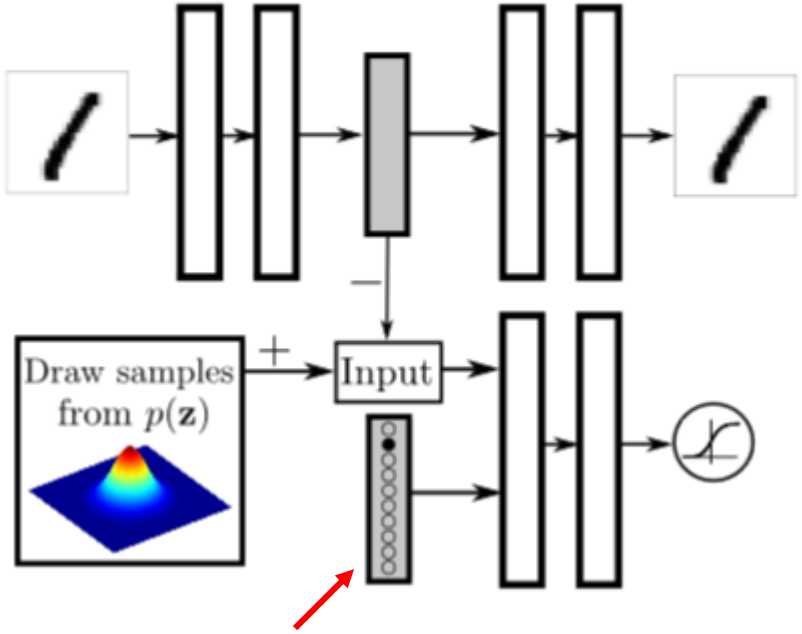
https://duvenaud.github.io/learn-discrete/slides/AdversarialAutoencoders.pdf

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Incorporating Label Information in the Adversarial Regularization

"In the scenarios where data is labeled, we can incorporate the label information in the adversarial training stage to better shape the distribution of the hidden code. In this section, we describe how to leverage partial or complete label information to regularize the latent representation of the autoencoder more heavily."



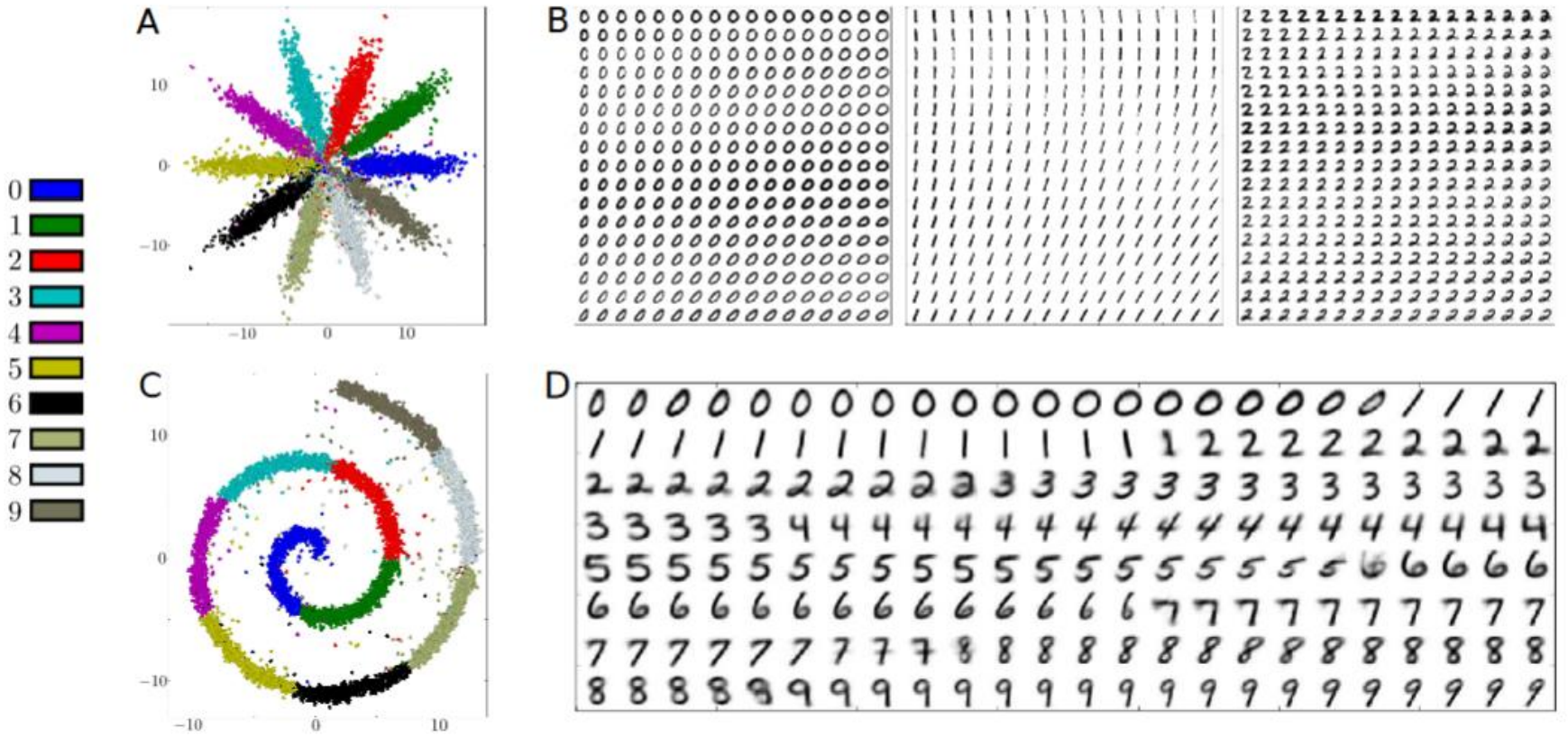
- Add one-hot vector to associate the label with a mode of the distribution.
- The one-hot vector acts as switch that selects the corresponding decision boundary of the discriminative network given the class label.
- This one-hot vector has an extra class for unlabeled examples. → semi-supervised approach

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Incorporating Label Information in the Adversarial Regularization

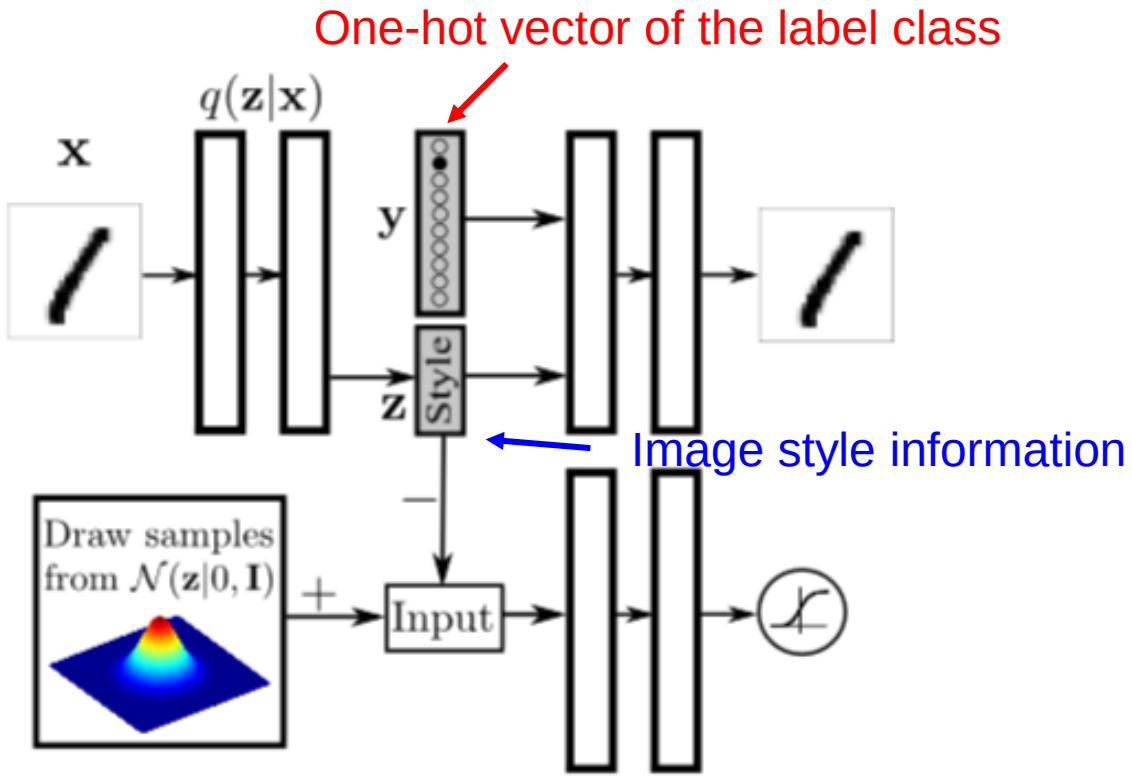
“In the scenarios where data is labeled, we can incorporate the label information in the adversarial training stage to better shape the distribution of the hidden code. In this section, we describe how to leverage partial or complete label information to regularize the latent representation of the autoencoder more heavily.”



Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Supervised Adversarial Autoencoders



Disentangling the label information from the hidden code by providing the one-hot vector to the generative model. The hidden code in this case learns to represent the style of the image.

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Supervised Adversarial Autoencoders

Varying label class

Varying image style

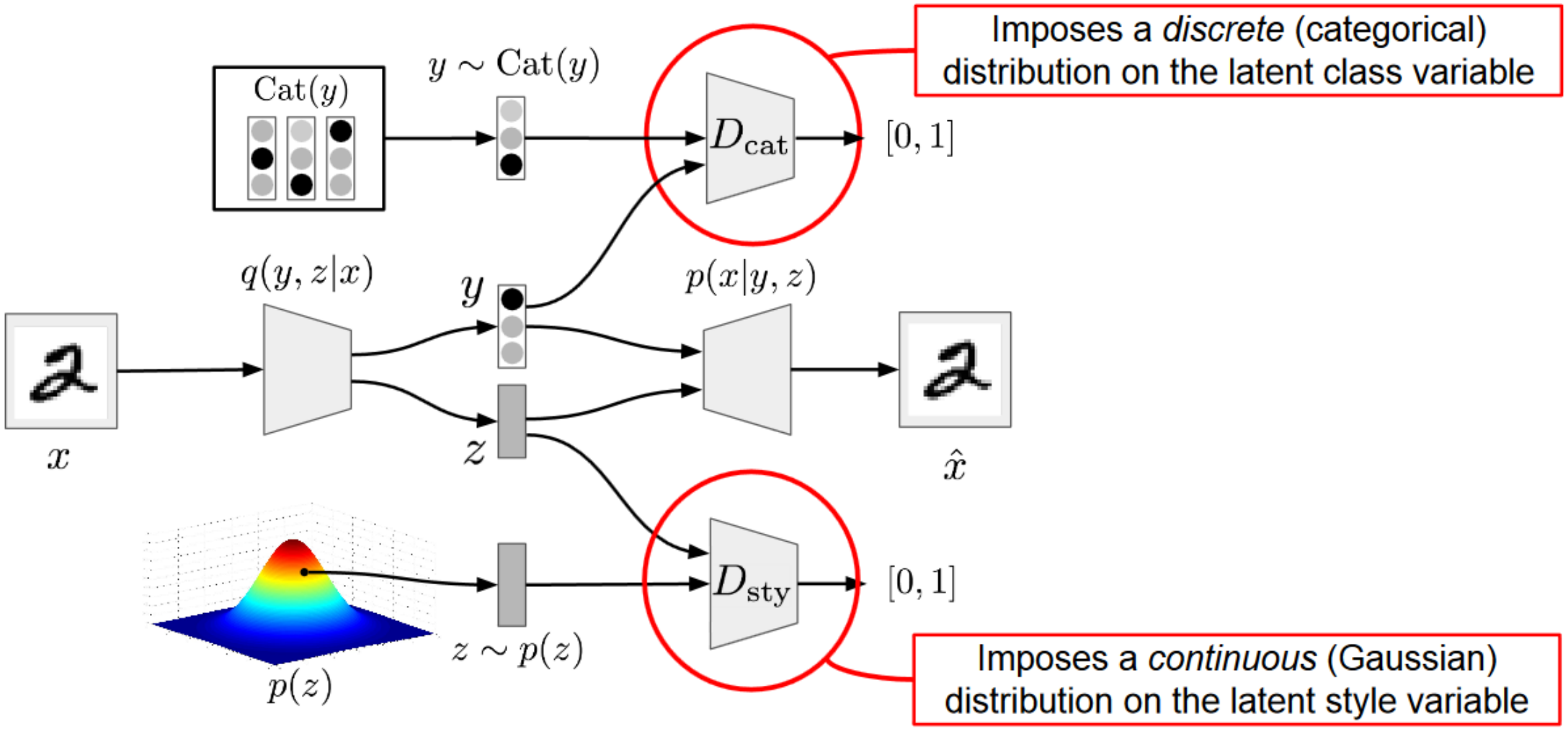
(a) MNIST (b) SVHN

Figure 7: Disentangling content and style (15-D Gaussian) on MNIST and SVHN datasets.

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Semi-supervised Adversarial Autoencoders



Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

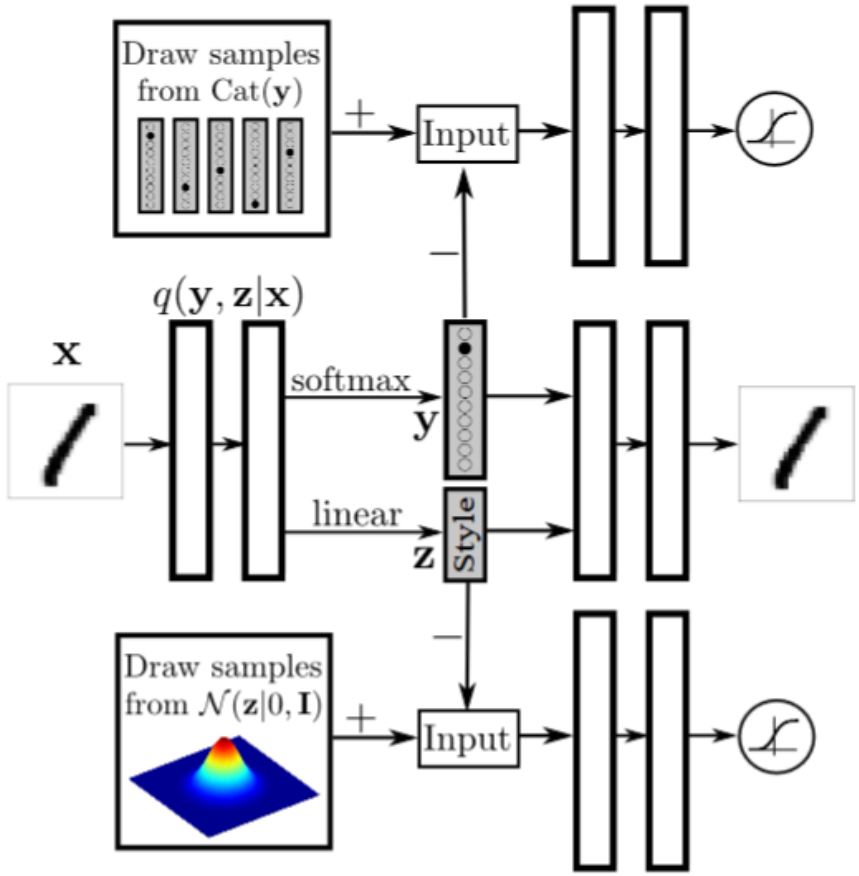
Semi-supervised Adversarial Autoencoders

Both of the adversarial networks as well as the autoencoder are trained jointly with SGD in ***three phases*** –

1. In the ***reconstruction*** phase, the autoencoder updates the encoder $q(\mathbf{y}, \mathbf{z}|\mathbf{x})$ and the decoder to minimize the reconstruction error of the inputs on ***an unlabeled mini-batch***.

2. In the ***regularization*** phase, each of the adversarial networks first updates their discriminative network to tell apart the true samples from the generated samples (the hidden codes computed by the autoencoder).
   The adversarial networks then update their generator to confuse their discriminative networks.

3. In the ***semi-supervised classification*** phase, the autoencoder updates $q(\mathbf{y}|\mathbf{x})$ to minimize the cross-entropy cost on ***a labled mini-batch***.



Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

Adversarial Autoencoder

Semi-supervised Adversarial Autoencoders

	MNIST (100)	MNIST (1000)	MNIST (All)	SVHN (1000)
NN Baseline	25.80	8.73	1.25	47.50
VAE (M1) + TSVM	11.82 (±0.25)	4.24 (±0.07)	-	55.33 (±0.11)
VAE (M2)	11.97 (±1.71)	3.60 (±0.56)	-	-
VAE (M1 + M2)	3.33 (±0.14)	2.40 (±0.02)	0.96	36.02 (±0.10)
VAT	2.33	1.36	0.64 (±0.04)	24.63
CatGAN	1.91 (±0.1)	1.73 (±0.18)	0.91	-
Ladder Networks	1.06 (±0.37)	0.84 (±0.08)	0.57 (±0.02)	-
ADGM	0.96 (±0.02)	-	-	16.61 (±0.24)
Adversarial Autoencoders	1.90 (±0.10)	1.60 (±0.08)	0.85 (±0.02)	17.70 (±0.30)

Table 2: Semi-supervised classification performance (error-rate) on MNIST and SVHN.

Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).

KAIST CHEMISTRY

Adversarially Regularized Autoencoder

KAIST

Adversarially Regularized Autoencoder

Learning latent variable models of discrete structures, such as text sequences or discretized images, remains a challenging problem

→ Generative models based on VAEs or GANs can easily degenerate into a unconditional language model.

→ Policy gradient methods or re-parameterization trick(e.g. Gumbel-Softmax distribution) are used to overcome.

The authors extend the AAE to *discrete sequences/structures*.

→ **Similar to the AAE,** our model learns an encoder from an input space to an adversarially regularized continuous latent space.

→ **However, unlike the AAE which utilizes a fixed prior,** the authors instead learn a parameterized prior as a GAN.

→ **Like sequence VAEs,** the model does not require using policy gradients or continuous relaxations.

→ **Like GANs,** the model provides flexibility in learning a prior through a parameterized generator.

Kim, Yoon, et al. "Adversarially regularized autoencoders for generating discrete structures. arXiv preprint." *arXiv preprint arXiv:1706.04223* 2 (2017).

Adversarially Regularized Autoencoder

Learning objective:

$$\min_{\phi,\psi} \quad \underline{\mathcal{L}_{\text{rec}}(\phi,\psi)} + \lambda^{(1)} \underline{W(\mathbb{P}_Q, \mathbb{P}_{\mathbf{z}})}$$

reconst. loss of AE

adversarial regularization to make two distributions similar

$$1)\ \min_{\phi,\psi} \quad \mathcal{L}_{\text{rec}}(\phi,\psi) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_\star}\left[-\log p_\psi(\mathbf{x}\,|\,\text{enc}_\phi(\mathbf{x}))\right]$$

$$2)\ \max_{w\in\mathcal{W}} \quad \mathcal{L}_{\text{cri}}(w) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_\star}\left[f_w(\text{enc}_\phi(\mathbf{x}))\right] - \mathbb{E}_{\tilde{\mathbf{z}}\sim\mathbb{P}_{\mathbf{z}}}\left[f_w(\tilde{\mathbf{z}})\right]$$

$$3)\ \min_{\phi} \quad \mathcal{L}_{\text{enc}}(\phi) = \mathbb{E}_{\mathbf{x}\sim\mathbb{P}_\star}\left[f_w(\text{enc}_\phi(\mathbf{x}))\right] - \mathbb{E}_{\tilde{\mathbf{z}}\sim\mathbb{P}_{\mathbf{z}}}\left[f_w(\tilde{\mathbf{z}})\right]$$

$W(\mathbb{P}_Q, \mathbb{P}_{\mathbf{z}})$:
Wasserstein distance between two distributions $\mathbb{P}_Q$ and $\mathbb{P}_{\mathbf{z}}$.

* Mathematics in WGAN paper:
https://www.slideshare.net/ssuser7e10e4/wasserstein-gan-i

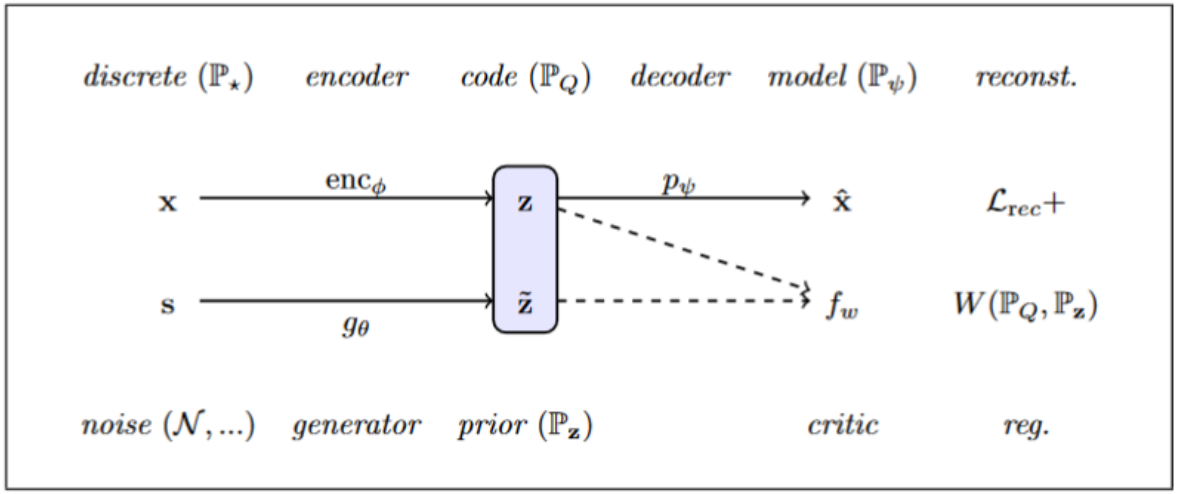


Figure 1: ARAE architecture. A discrete sequence $\mathbf{x}$ is encoded and decoded to produce $\hat{\mathbf{x}}$. A noise sample $\mathbf{s}$ is passed though a generator g_θ (possibly the identity) to produce a prior. The critic function f_w is only used at training to enforce regularization W. The model produce discrete samples $\mathbf{x}$ from noise $\mathbf{s}$. Section 5 relates these samples $\mathbf{x} \sim \mathbb{P}_\psi$ to $\mathbf{x} \sim \mathbb{P}_\star$.

Kim, Yoon, et al. "Adversarially regularized autoencoders for generating discrete structures. arXiv preprint." *arXiv preprint arXiv:1706.04223* 2 (2017).

Adversarially Regularized Autoencoder

Extension: Unaligned Transfer

Learning objective: $$\min_{\phi,\psi,\theta} \quad \mathcal{L}_{\text{rec}}(\phi,\psi) + \lambda^{(1)} W(\mathbb{P}_Q, \mathbb{P}_{\mathbf{z}}) - \lambda^{(2)} \mathcal{L}_{\text{class}}(\phi, u)$$

To learn to remove attribute distinctions from the prior

$\mathcal{L}_{class}(\phi, u)$: the loss of a classifier $p_u(y|\mathbf{z})$ from latent variable to labels

This requires two more update steps:

1. Training the classifier
$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} \log p_u\left(y^{(i)}|\mathbf{z}^{(i)}\right)$$
2. Adversarially training the encoder to this classifier.
$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m} \log p_u\left(1 - y^{(i)}|\mathbf{z}^{(i)}\right)$$

Kim, Yoon, et al. "Adversarially regularized autoencoders for generating discrete structures. arXiv preprint." *arXiv preprint arXiv:1706.04223* 2 (2017).

Adversarially Regularized Autoencoder

Extension: Unaligned Transfer

Positive	great indoor mall .
⇒ ARAE	no smoking mall .
⇒ Cross-AE	terrible outdoor urine .
Positive	it has a great atmosphere , with wonderful service .
⇒ ARAE	it has no taste , with a complete jerk .
⇒ Cross-AE	it has a great horrible food and run out service .
Positive	we came on the recommendation of a bell boy and the food was amazing .
⇒ ARAE	we came on the recommendation and the food was a joke .
⇒ Cross-AE	we went on the car of the time and the chicken was awful .
Negative	hell no !
⇒ ARAE	hell great !
⇒ Cross-AE	incredible pork !
Negative	small , smokey , dark and rude management .
⇒ ARAE	small , intimate , and cozy friendly staff .
⇒ Cross-AE	great , , , chips and wine .
Negative	the people who ordered off the menu did n't seem to do much better .
⇒ ARAE	the people who work there are super friendly and the menu is good .
⇒ Cross-AE	the place , one of the office is always worth you do a business .

- Sentiment transfer

Top) Positive → Negative

Bottom) Negative → Positive

Science	what is an event horizon with regards to black holes ?
⇒ Music	what is your favorite sitcom with adam sandler ?
⇒ Politics	what is an event with black people ?
Science	take 1ml of hcl (concentrated) and dilute it to 50ml .
⇒ Music	take em to you and shout it to me
⇒ Politics	take bribes to islam and it will be punished .
Science	just multiply the numerator of one fraction by that of the other .
⇒ Music	just multiply the fraction of the other one that 's just like it .
⇒ Politics	just multiply the same fraction of other countries .
Music	do you know a website that you can find people who want to join bands ?
⇒ Science	do you know a website that can help me with science ?
⇒ Politics	do you think that you can find a person who is in prison ?
Music	all three are fabulous artists , with just incredible talent ! !
⇒ Science	all three are genetically bonded with water , but just as many substances , are capable of producing a special case .
⇒ Politics	all three are competing with the government , just as far as i can .
Music	but there are so many more i can 't think of !
⇒ Science	but there are so many more of the number of questions .
⇒ Politics	but there are so many more of the can i think of today .

- Topic transfer

Kim, Yoon, et al. "Adversarially regularized autoencoders for generating discrete structures. arXiv preprint." *arXiv preprint arXiv:1706.04223* 2 (2017).

KAIST CHEMISTRY

Explaining and harnessing adversarial examples

KAIST

Explaining and Harnessing adversarial examples

Szegedy, Christian, et al. "Intriguing properties of neural networks." *arXiv preprint arXiv:1312.6199* (2013).

Szegedy et al. (2014b) demonstrated a variety of intriguing properties of neural networks and related models. Those most relevant to this paper include:

- Box-constrained L-BFGS can reliably find adversarial examples.
- On some datasets, such as ImageNet (Deng et al., 2009), the adversarial examples were so close to the original examples that the differences were indistinguishable to the human eye.
- The same adversarial example is often misclassified by a variety of classifiers with different architectures or trained on different subsets of the training data.
- Shallow softmax regression models are also vulnerable to adversarial examples.
- Training on adversarial examples can regularize the model—however, this was not practical at the time due to the need for expensive constrained optimization in the inner loop.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.

Explaining and Harnessing adversarial examples

Fast adversarial examples

- In many problems, the precision of an individual input feature is limited. For example, digital images often use only 8 bits per pixel so they discard all information below 1/256(=1/2^8) of the dynamic range.

- Because the precision of the features is limited, it is not rational for the classifier to respond differently to an input $\boldsymbol{x}$ than to an adversarial input $\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}$ if every element of the perturbation $\boldsymbol{\eta}$ is smaller than the precision of the features.

- Formally, for problems with well-separated classes, we expect the classifier to assign the same class to $\boldsymbol{x}$ and $\tilde{\boldsymbol{x}}$ so long as $\|\boldsymbol{\eta}\|_\infty < \epsilon$, where ϵ is small enough to be discarded by the sensor or data storage apparatus associated with our problem.
  *$\|\boldsymbol{\eta}\|_\infty = \max\{|\boldsymbol{\eta_i}|\}_{\boldsymbol{i=1,\ldots,n}}$

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.

Explaining and Harnessing adversarial examples

Fast adversarial examples

- Consider the dot product between a weight vector $\boldsymbol{\omega}$ and an adversarial example $\tilde{\boldsymbol{x}}$:

$$\boldsymbol{\omega}^T\tilde{\boldsymbol{x}} = \boldsymbol{\omega}^T\boldsymbol{x} + \boldsymbol{\omega}^T\boldsymbol{\eta}.$$

- The adversarial perturbation causes the activation to grow by $\boldsymbol{\omega}^T\boldsymbol{\eta}$. We can maximize this increase subject to the max norm constraint on $\boldsymbol{\eta}$ by assigning $\boldsymbol{\eta} = \text{sign}(\boldsymbol{\omega})$. If $\boldsymbol{\omega}$ has n dimensions and the average magnitude of an element of the weight vector is m, then the activation will grow by ϵmn.

- Since $\|\boldsymbol{\eta}\|_\infty$ does not grow with the dimensionality of the problem but the change in activation caused by perturbation by $\boldsymbol{\eta}$ can grow linearly with n, then for high dimensional problems, we can make many infinitesimal changes to the input that add up to one large change to the output.

- This explanation shows that a simple linear model can have adversarial examples if its input has sufficient dimensionality. Previous explanations for adversarial examples invoked hypothesized properties of neural networks, such as their supposed highly non-linear nature.

- Our hypothesis based on linearity is simpler, and can also explain why softmax regression is vulnerable to adversarial examples.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.

Explaining and Harnessing adversarial examples

Fast adversarial examples

Note that it is *not random*

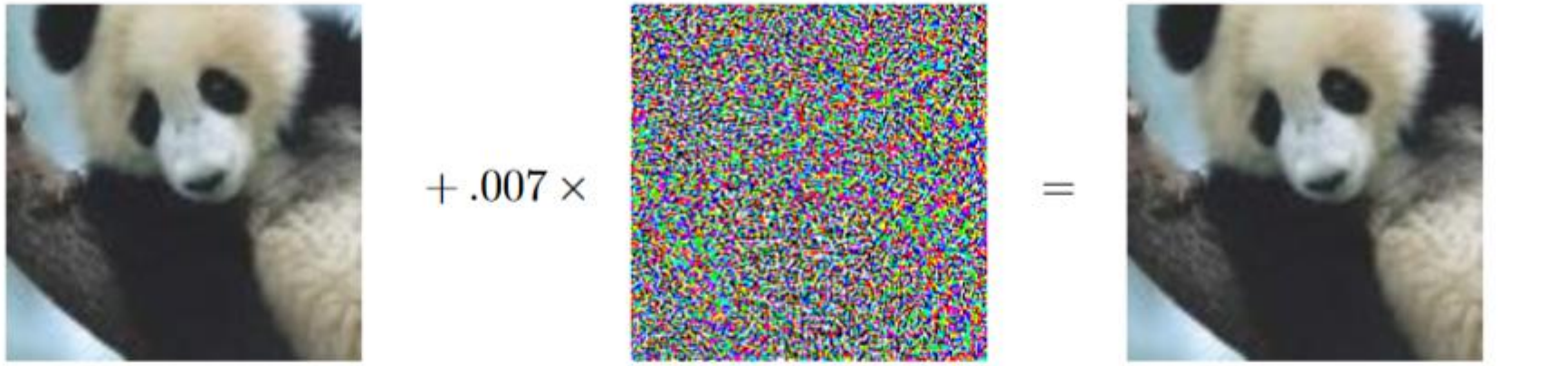


Figure 1: A demonstration of fast adversarial example generation applied to GoogLeNet (Szegedy et al., 2014a) on ImageNet. By adding an imperceptibly small vector whose elements are equal to the sign of the elements of the gradient of the cost function with respect to the input, we can change GoogLeNet's classification of the image. Here our ϵ of .007 corresponds to the magnitude of the smallest bit of an 8 bit image encoding after GoogLeNet's conversion to real numbers.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.

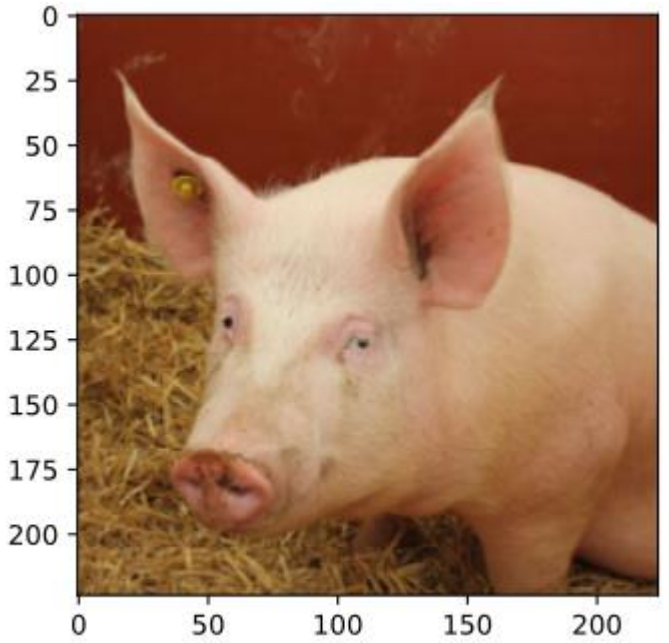
Explaining and Harnessing adversarial examples

Fast adversarial examples

```
from PIL import Image
from torchvision import transforms

# read the image, resize to 224 and convert to PyTorch Tensor
pig_img = Image.open("pig.jpg")
preprocess = transforms.Compose([
   transforms.Resize(224),
   transforms.ToTensor(),
])
pig_tensor = preprocess(pig_img)[None,:,:,:]

# plot image (note that numpy using HWC whereas Pytorch user CHW, so we need to convert)
plt.imshow(pig_tensor[0].numpy().transpose(1,2,0))
```



Explaining and Harnessing adversarial examples

Fast adversarial examples

```
import torch
import torch.nn as nn
from torchvision.models import resnet50

# simple Module to normalize an image
class Normalize(nn.Module):
    def __init__(self, mean, std):
        super(Normalize, self).__init__()
        self.mean = torch.Tensor(mean)
        self.std = torch.Tensor(std)
    def forward(self, x):
        return (x - self.mean.type_as(x)[None,:,None,None]) / self.std.type_as(x)[None,:,None

# values are standard normalization for ImageNet images,
# from https://github.com/pytorch/examples/blob/master/imagenet/main.py
norm = Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

# load pre-trained ResNet50, and put into evaluation mode (necessary to e.g. turn off batchnc
model = resnet50(pretrained=True)
model.eval();
```

```
# form predictions
pred = model(norm(pig_tensor))
```

```
import json
with open("imagenet_class_index.json") as f:
    imagenet_classes = {int(i):x[1] for i,x in json.load(f).items()}
print(imagenet_classes[pred.max(dim=1)[1].item()])
```

```
hog
```

Explaining and Harnessing adversarial examples

Fast adversarial examples

```
import torch.optim as optim
epsilon = 2./255

delta = torch.zeros_like(pig_tensor, requires_grad=True)
opt = optim.SGD([delta], lr=1e-1)

for t in range(30):
    pred = model(norm(pig_tensor + delta))
    loss = -nn.CrossEntropyLoss()(pred, torch.LongTensor([341]))
    if t % 5 == 0:
        print(t, loss.item())

    opt.zero_grad()
    loss.backward()
    opt.step()
    delta.data.clamp_(-epsilon, epsilon)

print("True class probability:", nn.Softmax(dim=1)(pred)[0,341].item())
```

```
0 -0.0038814544677734375
5 -0.00693511962890625
10 -0.015821456909179688
15 -0.08086681365966797
20 -12.229072570800781
25 -14.300384521484375
True class probability: 1.4027455108589493e-06
```

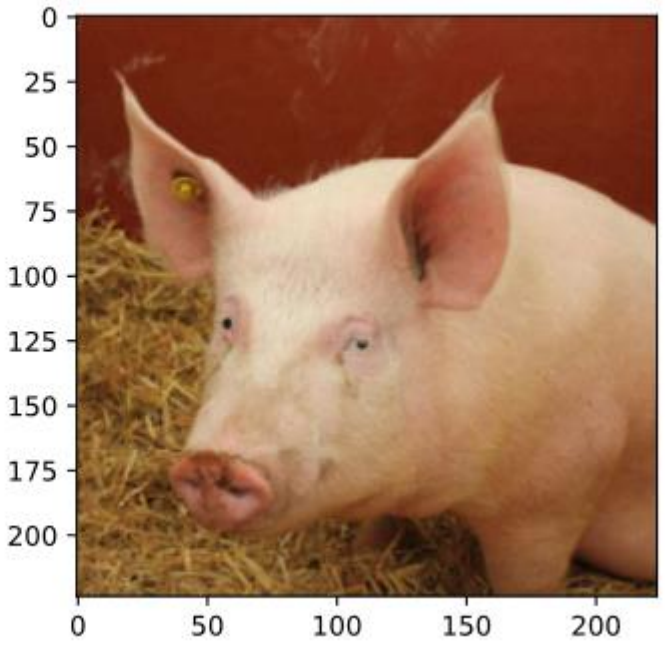
```
max_class = pred.max(dim=1)[1].item()
print("Predicted class: ", imagenet_classes[max_class])
print("Predicted probability:", nn.Softmax(dim=1)(pred)[0,max_class].item())
```

```
Predicted class:  wombat
Predicted probability: 0.9997960925102234
```

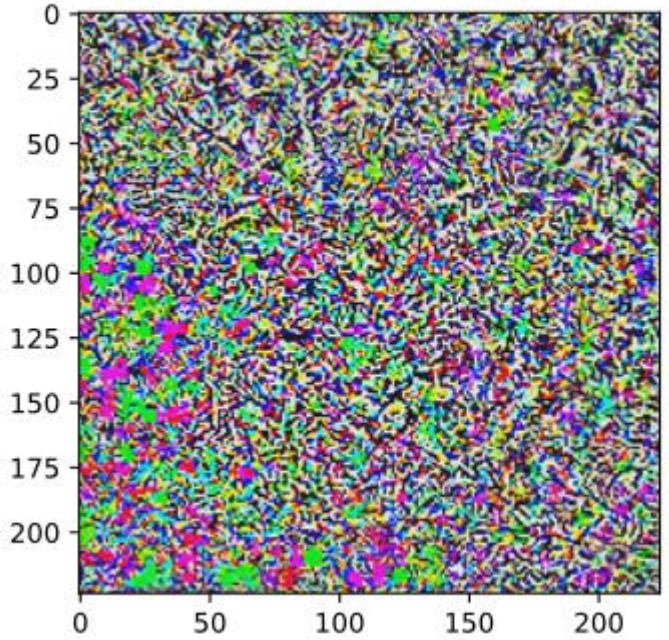
Explaining and Harnessing adversarial examples

Fast adversarial examples

```
plt.imshow((pig_tensor + delta)[0].detach().numpy().transpose(1,2,0))
```




```
plt.imshow((50*delta+0.5)[0].detach().numpy().transpose(1,2,0))
```



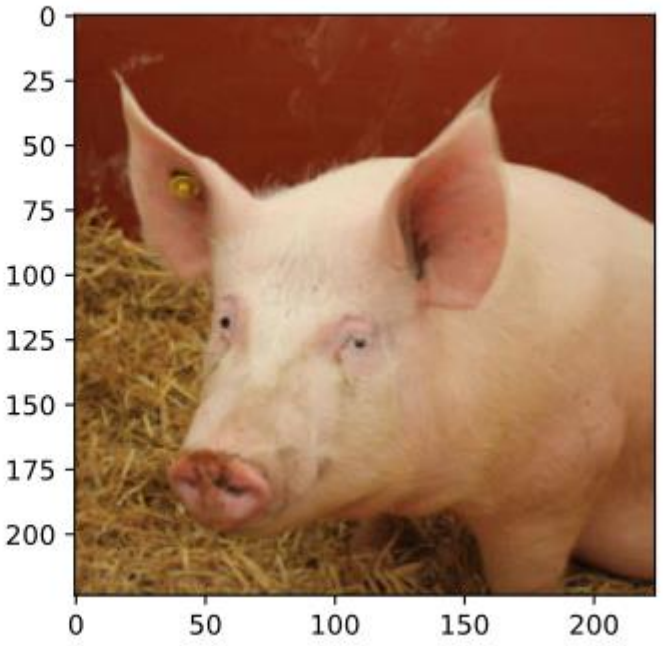
Explaining and Harnessing adversarial examples

Fast adversarial examples

Targeted attacks:
making NNs to predict pig as airliner

$$\underset{\delta \in Delta}{\operatorname{argmax}} \left(\mathcal{L}(h_\theta(x+\delta), y) - \mathcal{L}\left(h_\theta(x+\delta), y_{target}\right) \right)$$

```
plt.imshow((pig_tensor + delta)[0].detach().numpy().transpose(1,2,0))
```




```
delta = torch.zeros_like(pig_tensor, requires_grad=True)
opt = optim.SGD([delta], lr=5e-3)

for t in range(100):
    pred = model(norm(pig_tensor + delta))
    loss = (-nn.CrossEntropyLoss()(pred, torch.LongTensor([341])) +
            nn.CrossEntropyLoss()(pred, torch.LongTensor([404])))
    if t % 10 == 0:
        print(t, loss.item())

    opt.zero_grad()
    loss.backward()
    opt.step()
    delta.data.clamp_(-epsilon, epsilon)
```

```
0 24.00604820251465
10 -0.1628284454345703
20 -8.026773452758789
30 -15.677117347717285
40 -20.60370635986328
50 -24.99606704711914
60 -31.009849548339844
70 -34.80946350097656
80 -37.928680419921875
90 -40.32395553588867
```

```
max_class = pred.max(dim=1)[1].item()
print("Predicted class: ", imagenet_classes[max_class])
print("Predicted probability:", nn.Softmax(dim=1)(pred)[0,max_class].item())
```

```
Predicted class:  airliner
Predicted probability: 0.9679961204528809
```

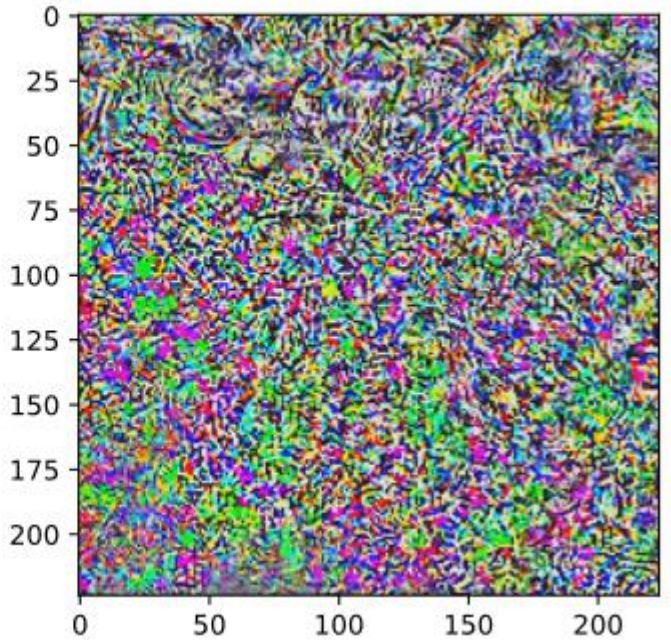
Explaining and Harnessing adversarial examples

Fast adversarial examples

Targeted attacks:
making NNs to predict pig as airliner

$$\underset{\delta \in Delta}{\operatorname{argmax}} \left(\mathcal{L}(h_\theta(x+\delta), y) - \mathcal{L}\left(h_\theta(x+\delta), y_{target}\right) \right)$$

```
plt.imshow((50*delta+0.5)[0].detach().numpy().transpose(1,2,0))
```




```
delta = torch.zeros_like(pig_tensor, requires_grad=True)
opt = optim.SGD([delta], lr=5e-3)

for t in range(100):
    pred = model(norm(pig_tensor + delta))
    loss = (-nn.CrossEntropyLoss()(pred, torch.LongTensor([341])) +
            nn.CrossEntropyLoss()(pred, torch.LongTensor([404])))
    if t % 10 == 0:
        print(t, loss.item())

    opt.zero_grad()
    loss.backward()
    opt.step()
    delta.data.clamp_(-epsilon, epsilon)
```

```
0 24.00604820251465
10 -0.1628284454345703
20 -8.026773452758789
30 -15.677117347717285
40 -20.60370635986328
50 -24.99606704711914
60 -31.009849548339844
70 -34.80946350097656
80 -37.928680419921875
90 -40.32395553588867
```

```
max_class = pred.max(dim=1)[1].item()
print("Predicted class: ", imagenet_classes[max_class])
print("Predicted probability:", nn.Softmax(dim=1)(pred)[0,max_class].item())
```

```
Predicted class:  airliner
Predicted probability: 0.9679961204528809
```

Explaining and Harnessing adversarial examples

Training with an adversarial objective function based on the fast gradient sign method was an effective regularizer:

$$\tilde{\mathcal{L}}(\theta, x, y) = \alpha \cdot \mathcal{L}(\theta, x, y) + (1-\alpha) \cdot \mathcal{L}\left(\theta, x + \epsilon \text{sign}\left(\nabla_x \mathcal{L}(\theta, x, y)\right)\right)$$

Typical objective (e.g. NLL) Adversarial example as a regularizer

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.

Training adversarially robust classifiers

$$\tilde{\mathcal{L}}(\theta) = \underset{\theta}{\text{argmin}}\, \hat{R}_{adv}(h_\theta, D_{train}) = \underset{\theta}{\text{argmin}} \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} \underset{\delta\in\Delta(x)}{\text{argmax}}\, \mathcal{L}(h_\theta(x+\delta), y)$$

1. For each $(x, y) \in \mathcal{D}_{train}$, solve the inner maximization problem:
$$\delta^*(x) = \underset{\delta\in\Delta(x)}{\text{argmax}}\, \mathcal{L}(h_\theta(x+\delta), y)$$
2. Compute the gradient of the empirical adversarial risk, and update θ
$$\theta := \theta - \frac{1}{|\mathcal{D}_{train}|} \sum_{(x,y)\in\mathcal{D}_{train}} \nabla_\theta \mathcal{L}(h_\theta(x+\delta^*(x)), y)$$

→ Mini-max or robust optimization formulation of adversarial learning

KAIST CHEMISTRY

Virtual Adversarial Training

KAIST

Virtual Adversarial Training

DISTRIBUTIONAL SMOOTHING WITH VIRTUAL ADVERSARIAL TRAINING

Takeru Miyato[1], Shin-ichi Maeda[1], Masanori Koyama[1], Ken Nakae[1] & Shin Ishii[2]
Graduate School of Informatics
Kyoto University
Yoshidahonmachi 36-1, Sakyo, Kyoto, Japan
[1]{miyato-t,ichi,koyama-m,nakae-k}@sys.i.kyoto-u.ac.jp
[2]ishii@i.kyoto-u.ac.jp

"We propose ***local distributional smoothness (LDS)***, a new notion of smoothness for statistical model that can be used as ***a regularization term to promote the smoothness of the model distribution***. We named the ***LDS based regularization as virtual adversarial training (VAT)***."

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Local Distributional Smoothness (LDS)

: the negative of the sensitivity of the model distribution $p(y|x,\theta)$ with respect to the perturbation of x, measured in the sense of KL divergence

$$\mathrm{LDS} := \mathrm{KL}\big(p(y|x,\theta)\|p(y|x+r,\theta)\big)$$

perburtation to x

Relation to adversarial training (Goodfellow *et. al.*, 2015)

: Goodfellow et. al. penalized the model's sensitivity with respect to the perturbation in the adversarial direction.

$$\tilde{\mathcal{L}}(\theta,x,y) = \alpha\cdot\underbrace{\mathcal{L}(\theta,x,y)}_{\text{Typical objective (e.g. NLL)}} + (1-\alpha)\cdot\underbrace{\mathcal{L}\left(\theta,x+\epsilon\mathrm{sign}\big(\nabla_x\mathcal{L}(\theta,x,y)\big)\right)}_{\text{Adversarial regularizer}}$$

On the other hand, using the language of adversarial training, LDS at each point is measuring the robustness of the model against the perturbation in 'virtual' adversarial direction. We therefore refer to our regularization method as ***virtual adversarial training (VAT)***.

Because LDS *does not require the label information*, VAT is also *applicable to semi-supervised learning*.

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Formalization of LDS

- Suppose the input space $\mathcal{R}^I$ and output space Q, and a training samples

$$D = \left\{\left(x^{(n)}, y^{(n)}\right) \mid x^{(n)} \in \mathcal{R}^I, y^{(n)} \in Q, n = 1, \dots, N\right\}$$

- Consider the problem of using D to train the model distribution $p(y|x, \theta)$ parameterized by θ. Also with the hyperparameter $\epsilon > 0$, we define

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

$$r_{v-adv}^{(n)} \equiv \underset{r}{\mathrm{argmax}}\left\{\Delta_{\mathrm{KL}}\left(r, x^{(n)}, \theta\right);\ \|r\|_2 \leq \epsilon\right\}$$

- We refer to as $r_{v-adv}^{(n)}$ the virtual adversarial perturbation. We define the local distributional smoothing (LDS) of the model distribution at $x^{(n)}$ by

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

  Note $r_{v-adv}^{(n)}$ is the direction to which the model distribution $p(y|x^{(n)}, \theta)$ is the most sensitive in the sense of KL divergence.

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Formalization of LDS

- The smaller the value of $\Delta_{\mathrm{KL}}\left(r_{v-adv}^{(n)}, x^{(n)}, \theta\right)$, the smoother $p(y|x^{(n)}, \theta)$ the at x.

- Our goal is to improve the smoothness of the model in the neighborhood of all the observed inputs. Formulating this goal based on the LDS, we obtain the following objective function.

$$\frac{1}{N}\sum_{n=1}^{N} -\log p\left(y^{(n)}, x^{(n)}, \theta\right) + \lambda \frac{1}{N}\sum_{n=1}^{N} \mathrm{LDS}\left(x^{(n)}, \theta\right).$$

  We call the training based on the above equation the virtual adversarial training (VAT).

- If we define $r_{adv}^{(n)} \equiv \mathrm{argmin}_r\left\{p\left(y^{(n)}|x^{(n)} + r, \theta\right), \|r\|_p \leq \epsilon\right\}$ and replace $\Delta_{\mathrm{KL}}\left(r_{v-adv}^{(n)}, x^{(n)}, \theta\right)$ with $-\log p\left(y^{(n)}|x^{(n)} + r_{adv}^{(n)}, \theta\right)$, we obtain the objective function of the adversarial training (Goodfellow et. al., 2015)

$$\tilde{\mathcal{L}}(\theta, x, y) = \alpha \cdot \mathcal{L}(\theta, x, y) + (1-\alpha) \cdot \mathcal{L}\left(\theta, x + \epsilon \mathrm{sign}\left(\nabla_x \mathcal{L}(\theta, x, y)\right)\right)$$

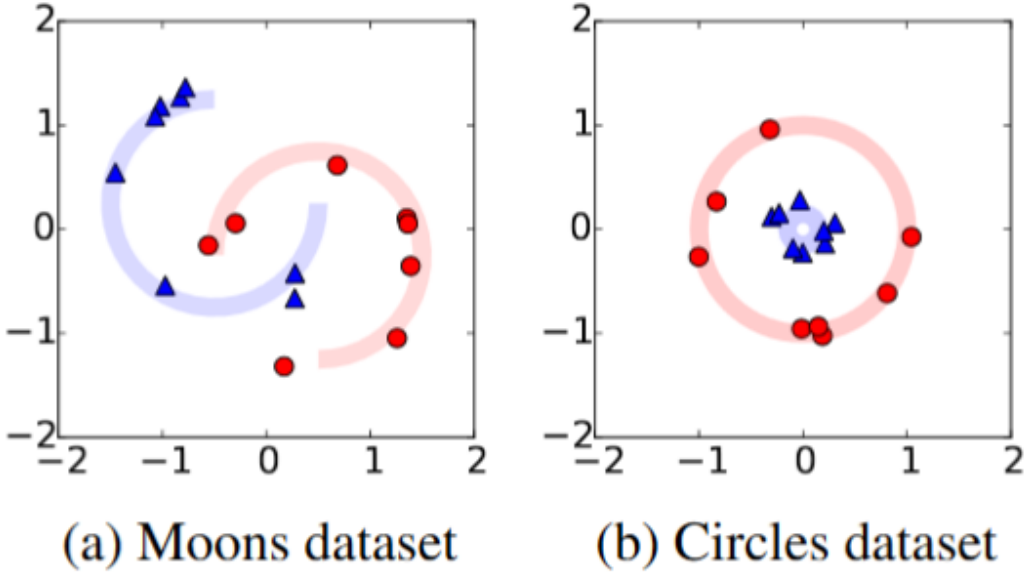
Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset

- Generating multiple points uniformly over two trajectories on $\mathcal{R}^2$
- Linearly embedding them into 100 dimensional input vector space.
- 16 training samples, 1000 test samples
- Because the number of training samples is very small relative to the input dimension, ***maximum likelihood estimation (MLE) is vulnerable to over-fitting problem*** on these datasets.



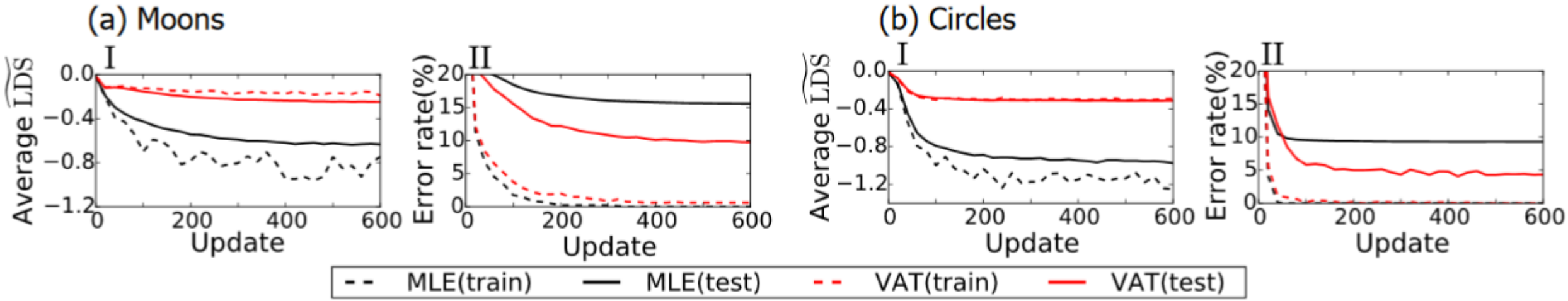
(a) Moons dataset (b) Circles dataset

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset

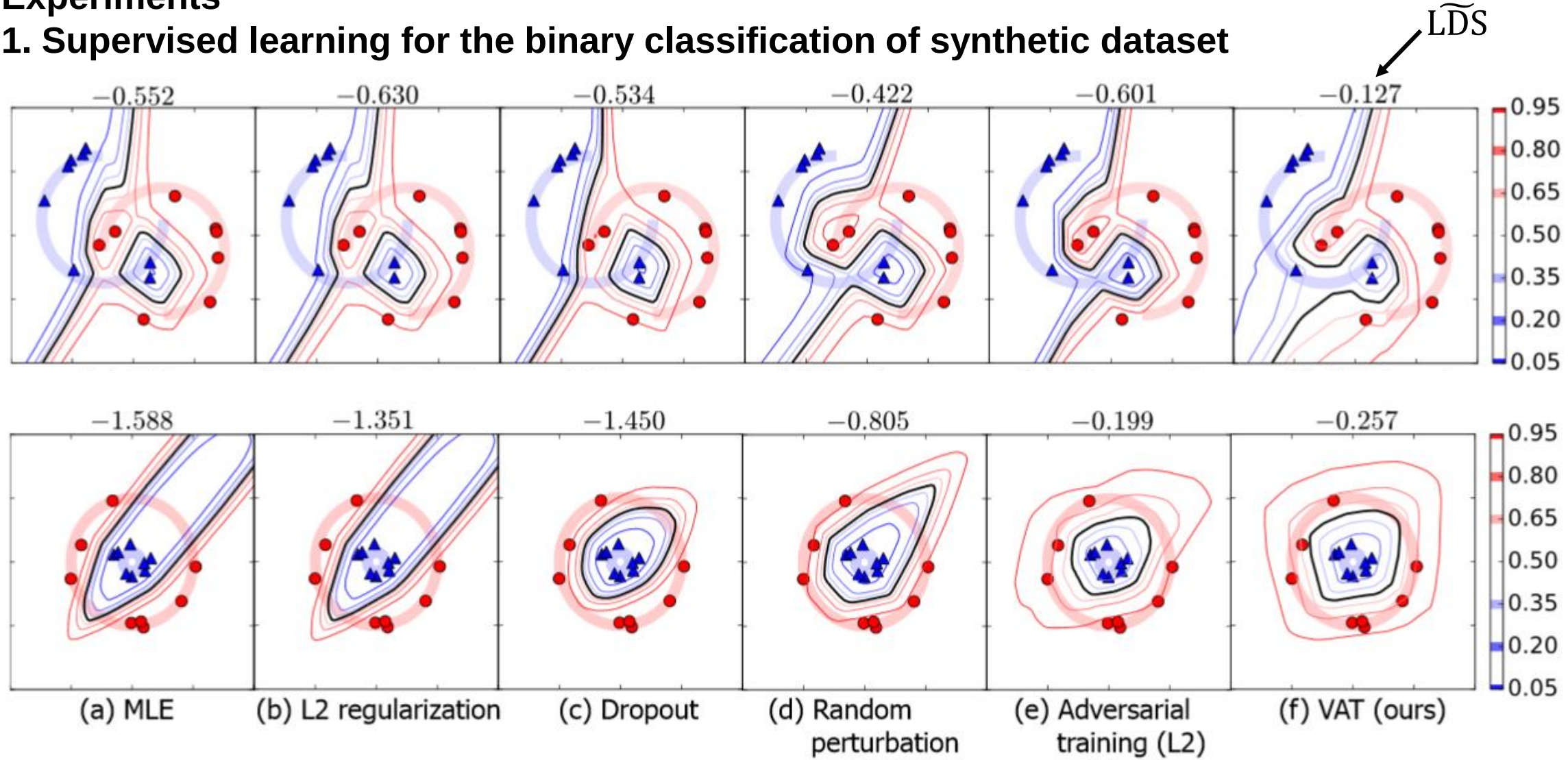


Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset



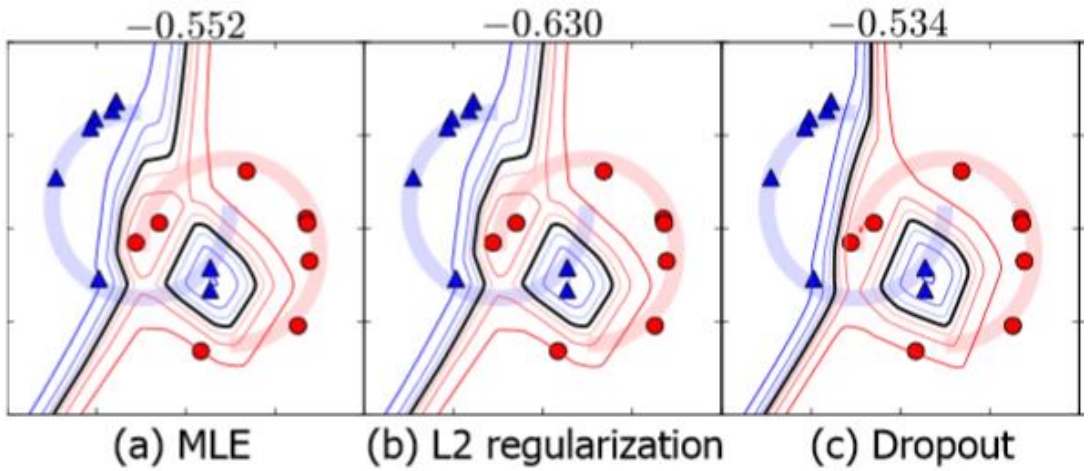
Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

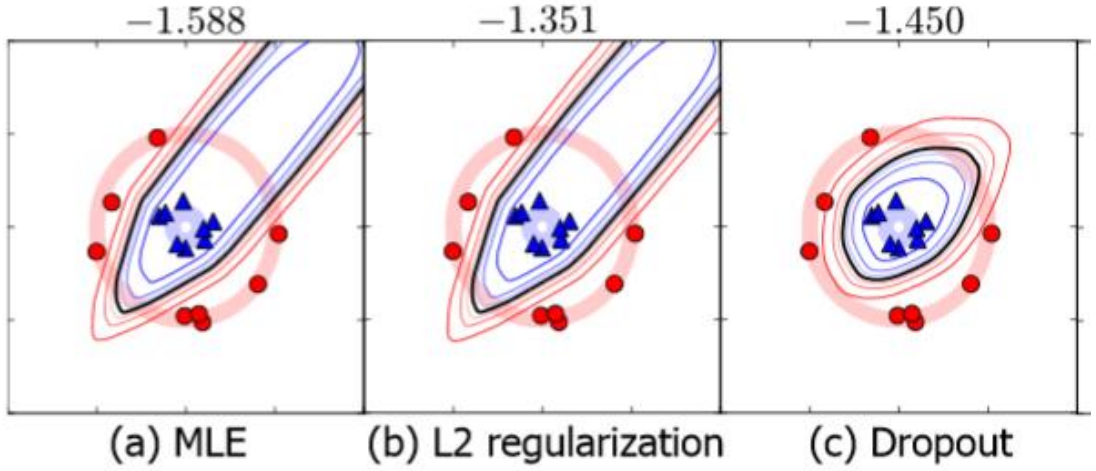
Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset

- NN without regularization (MLE) and NN with L_2 regularization are drawing wrong decision boundary.
- The decision boundary drawn by dropout for ‘Circles’ is convincing, but that of for ‘Moons’ does not coincide with our intention.





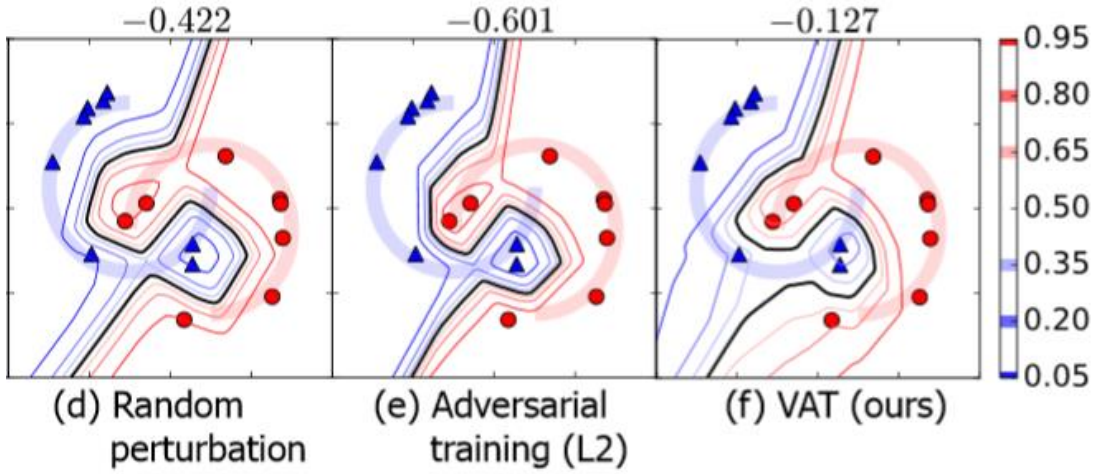
Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

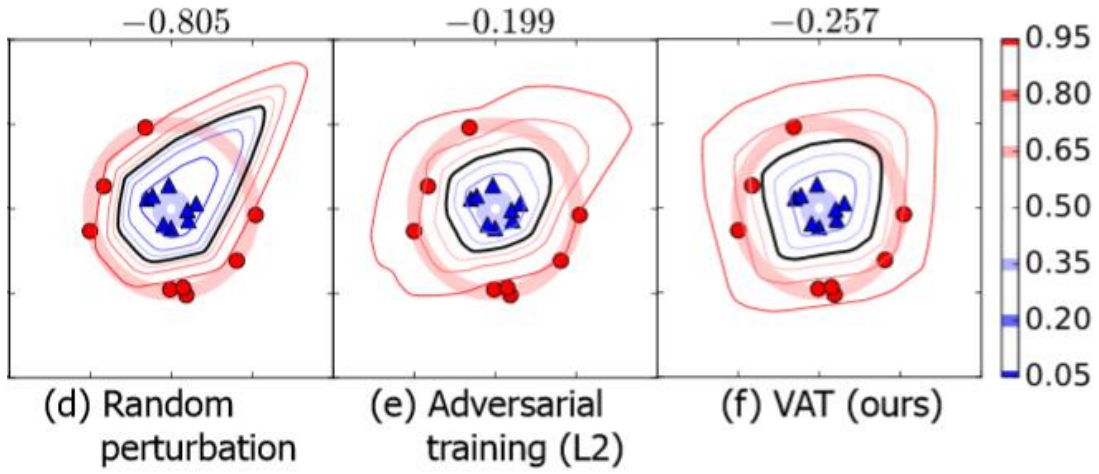
Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset

- The opposite can be said for the random perturbation training.
- The decision boundary drawn by dropout for ‘Circles’ is convincing, but that of for ‘Moons’ does not coincide with our intention.





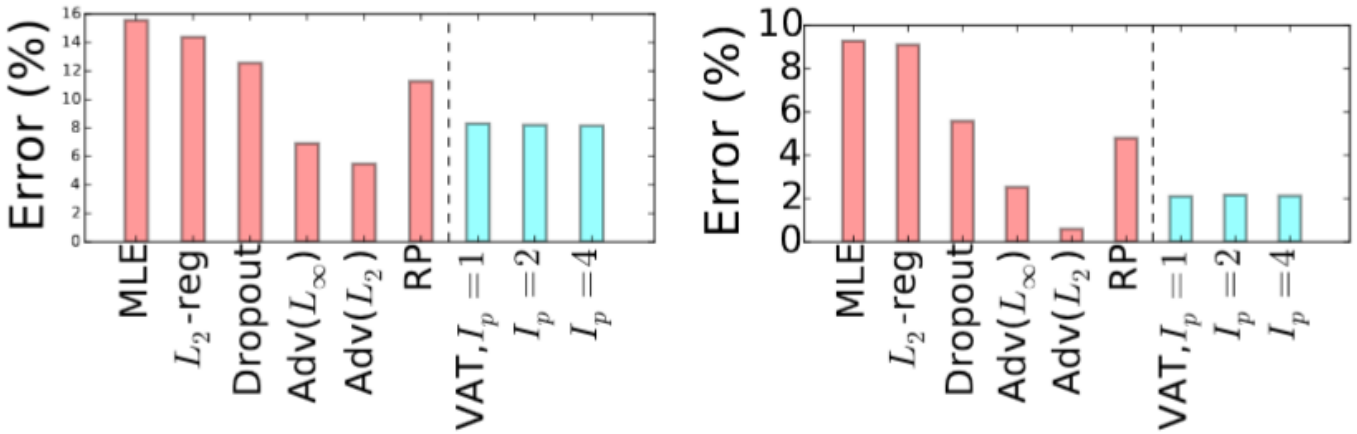
Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

1. Supervised learning for the binary classification of synthetic dataset

- VAT is drawing appropriate decision boundary by imposing local smoothness regularization around each data point.
- This doe not mean, however, that the large value of LDS immediately implies good boundary.
- By its very definition, LDS tends to disfavor abrupt change of the likelihood around training datapoint.
- Larger value of LDS therefore forces large relative margin around the decision boundary.



(a) Moons (b) Circles

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

2. Supervised learning for the classification of the MNIST dataset

Method	Test error (%)
SVM (gaussian kernel)	1.40
Gaussian dropout (Srivastava et al., 2014)	0.95
Maxout Networks (Goodfellow et al., 2013)	0.94
*MTC (Rifai et al., 2011)	0.81
*DBM (Srivastava et al., 2014)	0.79
Adversarial training (Goodfellow et al., 2015)	0.782
*Ladder network (Rasmus et al., 2015)	0.57±0.02
Plain NN (MLE)	1.11
Random perturbation training	0.843
Adversarial training (with L_∞ norm constraint)	0.788
Adversarial training (with L_2 norm constraint)	0.708
VAT (ours)	0.637±0.046

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

Virtual Adversarial Training

Experiments

3. Semi-supervised learning for the classification of the benchmark datasets

(a) MNIST

Method	Test error(%)			
N_l	100	600	1000	3000
SVM (Weston et al., 2012)	23.44	8.85	7.77	4.21
TSVM (Weston et al., 2012)	16.81	6.16	5.38	3.45
EmbedNN (Weston et al., 2012)	16.9	5.97	5.73	3.59
*MTC (Rifai et al., 2011)	12.0	5.13	3.64	2.57
PEA (Bachman et al., 2014)	10.79	2.44	2.23	1.91
*PEA (Bachman et al., 2014)	5.21	2.87	2.64	2.30
*DG (Kingma et al., 2014)	3.33	2.59	2.40	2.18
*Ladder network (Rasmus et al., 2015)	1.06		0.84	
Plain NN (MLE)	21.98	9.16	7.25	4.32
VAT (ours)	2.33	1.39	1.36	1.25

(b) SVHN

Method	Test error(%)
N_l	1000
TSVM (Kingma et al., 2014)	66.55
*DG,M1+TSVM (Kingma et al., 2014)	55.33
*DG,M1+M2 (Kingma et al., 2014)	36.02
SVM (Gaussian kernel)	63.28
Plain NN (MLE)	43.21
VAT (ours)	24.63

(c) NORB

Method	Test error(%)
N_l	1000
TSVM (Kingma et al., 2014)	26.00
*DG,M1+TSVM (Kingma et al., 2014)	18.79
SVM (Gaussian kernel)	23.62
Plain NN (MLE)	20.00
VAT (ours)	9.88

- Recall that our definition of LDS at any point x is independent of the label information y.
- This in particular means that we can apply the VAT to semi-supervised learning tasks.

Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).

References

- Adversarial Robustness - Theory and Practice, https://adversarial-ml-tutorial.org/
- Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.
- Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples (2014)." *arXiv preprint arXiv:1412.6572*.
- Makhzani, Alireza, et al. "Adversarial autoencoders." *arXiv preprint arXiv:1511.05644* (2015).
- Adversarial Autoencoders, Paul Vicol, https://duvenaud.github.io/learn-discrete/slides/AdversarialAutoencoders.pdf
- Kim, Yoon, et al. "Adversarially regularized autoencoders for generating discrete structures. arXiv preprint." *arXiv preprint arXiv:1706.04223* 2 (2017).
- Miyato, Takeru, et al. "Distributional smoothing with virtual adversarial training." *arXiv preprint arXiv:1507.00677* (2015).
- Miyato, Takeru, et al. "Virtual adversarial training: a regularization method for supervised and semi-supervised learning." *IEEE transactions on pattern analysis and machine intelligence* (2018).

KAIST CHEMISTRY

Thank You!

Next: Survey on semi-supervised learning with deep neural networks

KAIST