Deep Learning

Caio Corro, Michèle Sebag
CNRS & Université Paris-Saclay

March 31st, 2023

Credit for slides: Yoshua Bengio, Yann Le Cun, Nando de Freitas, Christian Perone, Honglak Lee, Stéphane Canu, Paul-Louis Pröve, Fei-Fei Li

Variational Auto-Encoders

Generative Adversarial Networks

Domain Adaptation

Beyond AE

- A compressed (latent) representation

$$x \in \mathbb{R}^D \mapsto z = Enc(x) \in \mathbb{R}^d \mapsto Dec(z) \in \mathbb{R}^D \approx x$$

- Distance in latent space is meaningful $d(Enc(x), Enc(x'))$ reflects $d(x, x')$
- But $\forall z \in \mathbb{R}^d$: is $Dec(z) \in \mathbb{R}^D$ meaningful ?

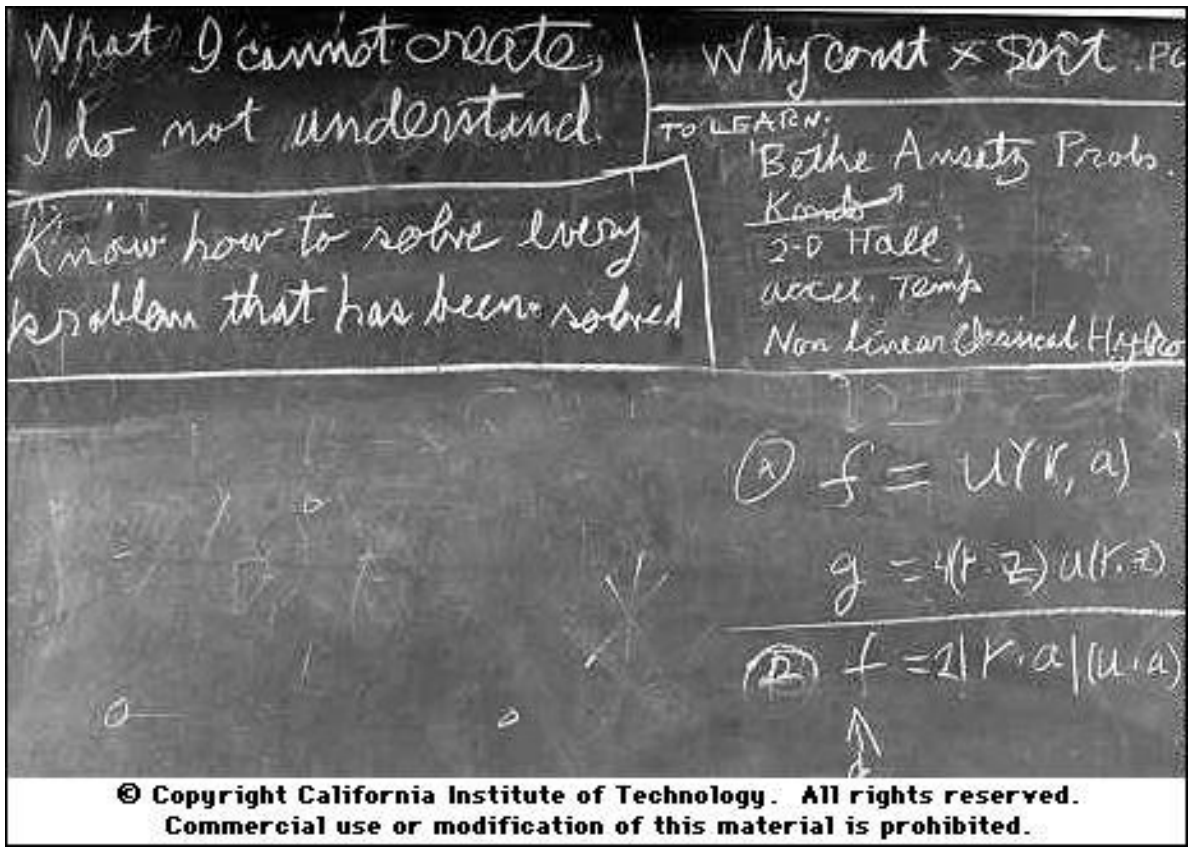
Beyond AE

- A compressed (latent) representation

$$x \in \mathbb{R}^D \mapsto z = Enc(x) \in \mathbb{R}^d \mapsto Dec(z) \in \mathbb{R}^D \approx x$$

- Distance in latent space is meaningful $d(Enc(x), Enc(x'))$ reflects $d(x, x')$
- But $\forall z \in \mathbb{R}^d$: is $Dec(z) \in \mathbb{R}^D$ meaningful ?

"What I cannot create I do not understand" Feynman 88

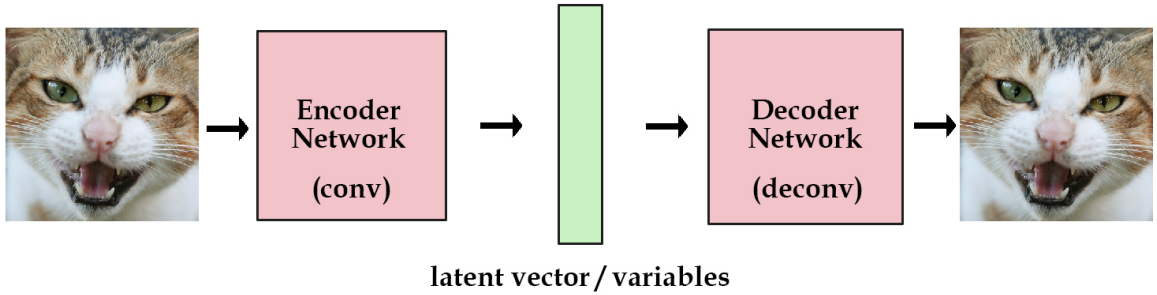


Variational Auto-Encoders

Kingma Welling 14, Salimans Kingma Welling 15

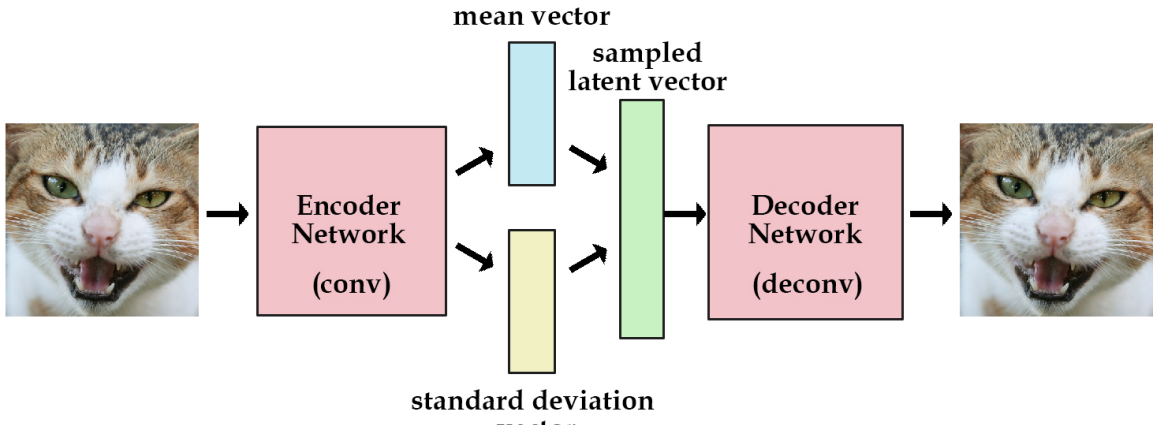
What we have:

- *Enc* a memorization of the data s.t. exists $Dec \approx Enc^{-1}$



What we want:

- $z \sim \mathcal{P}$ s.t. $Dec(z) \sim \mathcal{D}_{data}$



Distribution estimation

Data

$$\mathcal{E} = \{x_1, \ldots, x_n, x_i \in \mathcal{X}\}$$

Goal

- Find a probability distribution that models the data

$$p_\theta : \mathcal{X} \mapsto [0, 1] \quad \text{s.t. } \theta = \arg\max \prod_i p_\theta(x_i)$$

$\equiv$ maximize the log likelihood of data

$$\arg\max \prod_i p_\theta(x_i) = \arg\max \sum_i log(p_\theta(x_i))$$

Gaussian case

$$\theta = (\mu, \sigma) \qquad p_\theta(x) = \frac{1}{\sigma\sqrt{2\pi}} exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right)$$

Akin Graphical models

Find hidden variables z s.t.

$$\mathbf{z} \mapsto \mathbf{x} \text{ s.t. } \quad \text{good} \quad p(\mathbf{x}|\mathbf{z})$$

Bayes relation

$$p(\mathbf{x}, \mathbf{z}) = p(\mathbf{z}|\mathbf{x}).p(\mathbf{x}) = p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})$$

Hence

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})}{\int p(\mathbf{x}|\mathbf{z}).p(\mathbf{z})d\mathbf{z}}$$

Problem:

denominator computationally intractable...

State of art

- Monte-Carlo estimation
- Variational Inference
  choose $\mathbf{z}$ well-behaved, and make $q(\mathbf{z})$ "close" to $p(\mathbf{z}|\mathbf{x})$.

Variational Inference

- Approximate $p(\mathbf{z}|\mathbf{x})$ by $q(\mathbf{z})$
- Minimize distance between both, using Kullback-Leibler divergence

Reminder

- information $(\mathbf{x}) = -log(p(\mathbf{x}))$
- entropy$(\mathbf{x}_1, \dots \mathbf{x}_k) = -\sum_i p(\mathbf{x}_i) log(p(\mathbf{x}_i))$
- Kullback-Leibler divergence between distribution q and p

$$KL(q||p) = \sum_x q(\mathbf{x}) log \frac{q(\mathbf{x})}{p(\mathbf{x})}$$

Beware: not symmetrical, hence not a distance; plus numerical issues when supports are different

Variational inference

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z}, \mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

as $\int q(\mathbf{z}) d\mathbf{z} = 1$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}, \mathbf{x})} d\mathbf{z} + log(p(\mathbf{x}))$$

Evidence Lower Bound (ELBO)

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z}|\mathbf{x})} d\mathbf{z}$$

use $p(\mathbf{z}|\mathbf{x}) = p(\mathbf{z},\mathbf{x})/p(\mathbf{x})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})p(\mathbf{x})}{p(\mathbf{z},\mathbf{x})} d\mathbf{z}$$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

as $\int q(\mathbf{z}) d\mathbf{z} = 1$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = \int q(\mathbf{z}) log \frac{q(\mathbf{z})}{p(\mathbf{z},\mathbf{x})} d\mathbf{z} + log(p(\mathbf{x}))$$

recover $KL(q(\mathbf{z})||p(\mathbf{z},\mathbf{z})$

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = -\int q(\mathbf{z}) log \frac{p(\mathbf{z},\mathbf{x})}{q(\mathbf{z})} d\mathbf{z} + log(p(\mathbf{x}))$$

Evidence Lower Bound, 2

Define

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

Last slide:

$$KL(q(\mathbf{z})||p(\mathbf{z}|\mathbf{x})) = log(p(\mathbf{x})) - L(q(\mathbf{z})))$$

Hence

Minimize Kullback-Leibler divergence $\equiv$ Maximize L(q(**z**))

Evidence Lower Bound, 3

More formula massaging

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

Evidence Lower Bound, 3

More formula massaging

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}, \mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

use $p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z}|\mathbf{x})p(\mathbf{x})$

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{q(\mathbf{z})} d\mathbf{z}$$

$$L(q(\mathbf{z})) = \int q(\mathbf{z}) log \frac{p(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} d\mathbf{z} + \int q(\mathbf{z}) log(p(\mathbf{x})) d\mathbf{z}$$

$$L(q(\mathbf{z})) = -KL(q(\mathbf{z})||p(\mathbf{z}|x)) + \mathbb{E}_q[log(p(\mathbf{x}))]$$

Finally

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x})] - KL(q(\mathbf{z})||p(\mathbf{z}|x))$$

make $p(\mathbf{x})$ great under q — **akin data fitting**

while minimizing the KL divergence between the two — **akin regularization**

Where neural nets come in

Searching p and q

- We want $p(\mathbf{x}|\mathbf{z})$, we search $p(\mathbf{z}|x)$
- Let $p(\mathbf{z}|\mathbf{x})$ be defined as a neural net (encoder)
- We want it to be close to a well-behaved (**Gaussian**) distribution $q(\mathbf{z})$

$$\text{Minimize } KL(q(\mathbf{z})||p(\mathbf{z}|x))$$

- And from $\mathbf{z}$ we generate a distribution $p(\mathbf{x}|\mathbf{z})$ (defined as a neural net, "decoder")
- such that $p(\mathbf{x}|\mathbf{z})$ gives a high probability mass to our data (next slide)

$$\text{Maximize } \mathbb{E}_q[log(p(\mathbf{x}))]$$

Good news

All these criteria are differentiable ! can be used to train the neural net.

The loss of the variational decoder

Continuous case

- $\mathbf{x} \mapsto \mathbf{z}$; Gaussian case, $\mathbf{z} \sim p(\mathbf{z}|\mathbf{x})$
- Now $\mathbf{z}$ is given as input to the decoder, generates $\hat{\mathbf{x}}$ (deterministic)
- $p(\mathbf{x}|\hat{\mathbf{x}}) = F(\exp\{-\|\mathbf{x} - \hat{\mathbf{x}}\|^2\})$
- ... back to the L_2 loss

Binary case

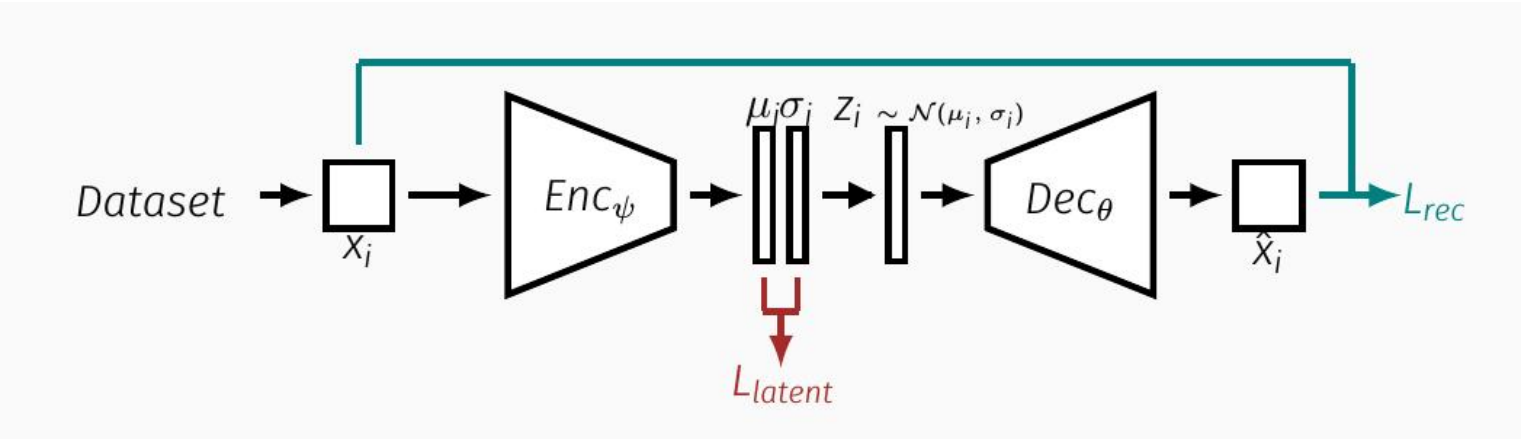
- Exercize: back to the cross-entropy loss

Variational auto-encoders

Kingma et al. 13

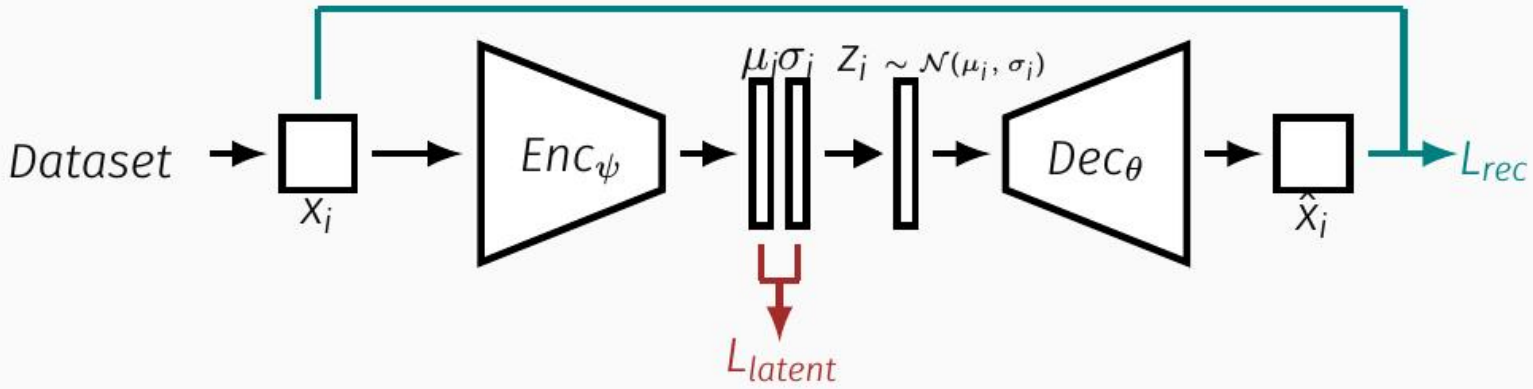
Position

- Like an auto-encoder (data fitting term) with a regularizer, the KL divergence between the distribution of the hidden variables $\mathbf{z}$ and the target distribution.
- Say the hidden variable follows a Gaussian distribution: $\mathbf{z} \sim \mathcal{N}(\mu, \Sigma)$
- Therefore, the encoder must compute the parameters μ and Σ



Variational auto-encoders, 2

Kingma et al. 13



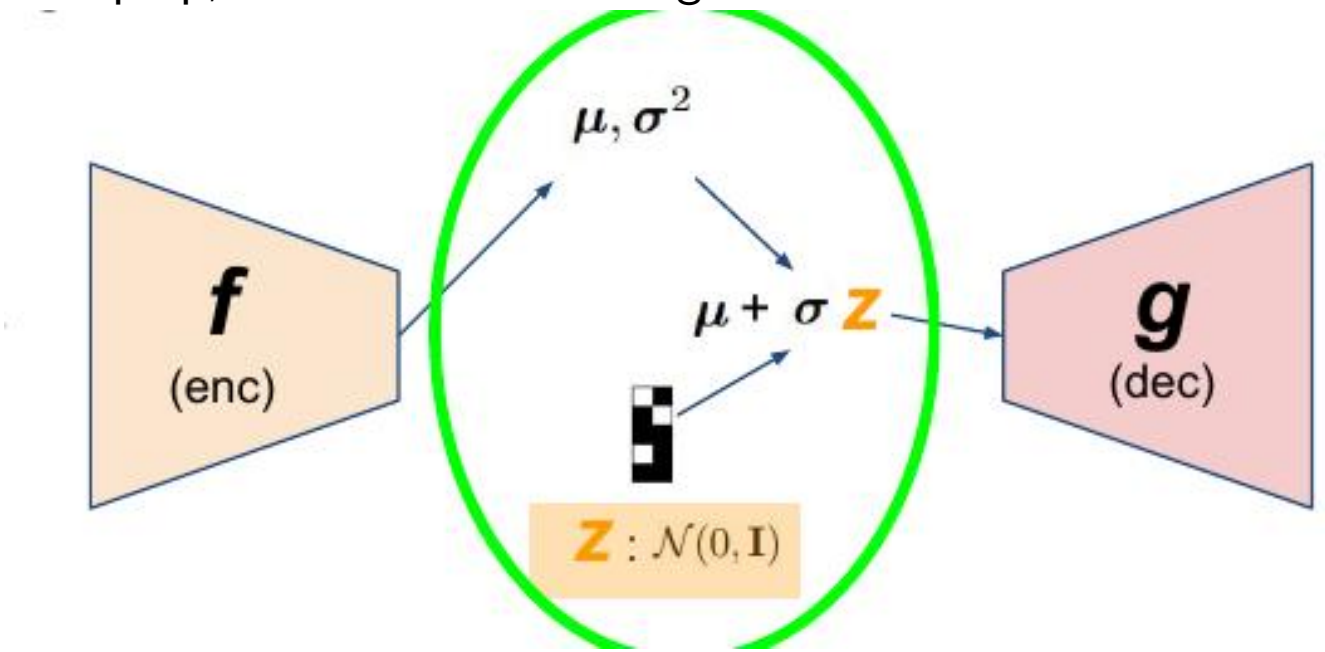
- encoding cost: $L_{latent} = \sum_i D_{KL}\left(Enc_\psi(x_i) \,||\, \mathcal{N}(0;1)\right)$
- reconstruction loss:

$$L_{rec} = \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} \left[-log\, p_{Dec_\theta(z)}(x_i)\right]$$
$$= \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} ||Dec_\theta(z) - x_i||^2 + cst.$$

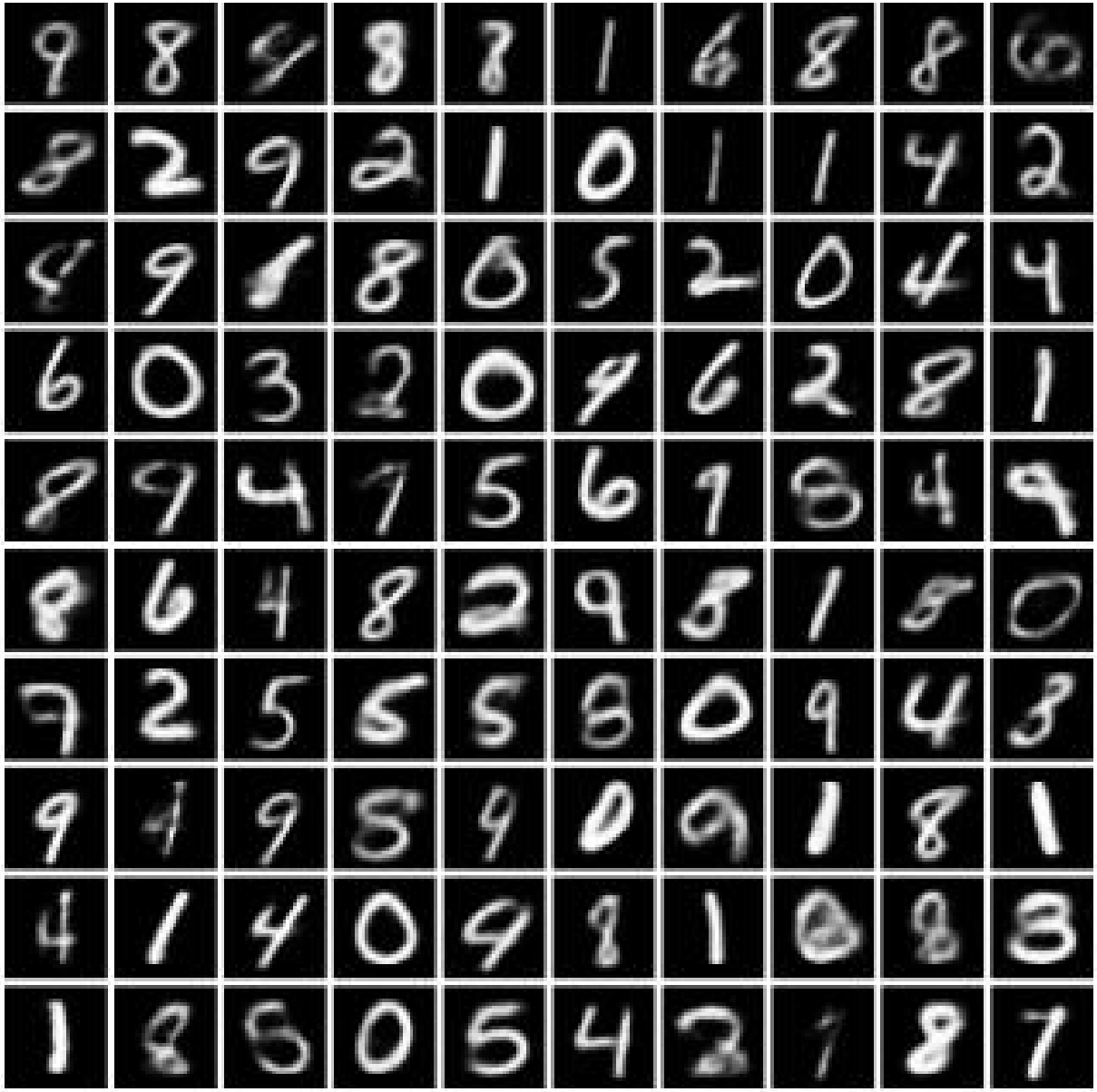
The reparameterization trick

Principle

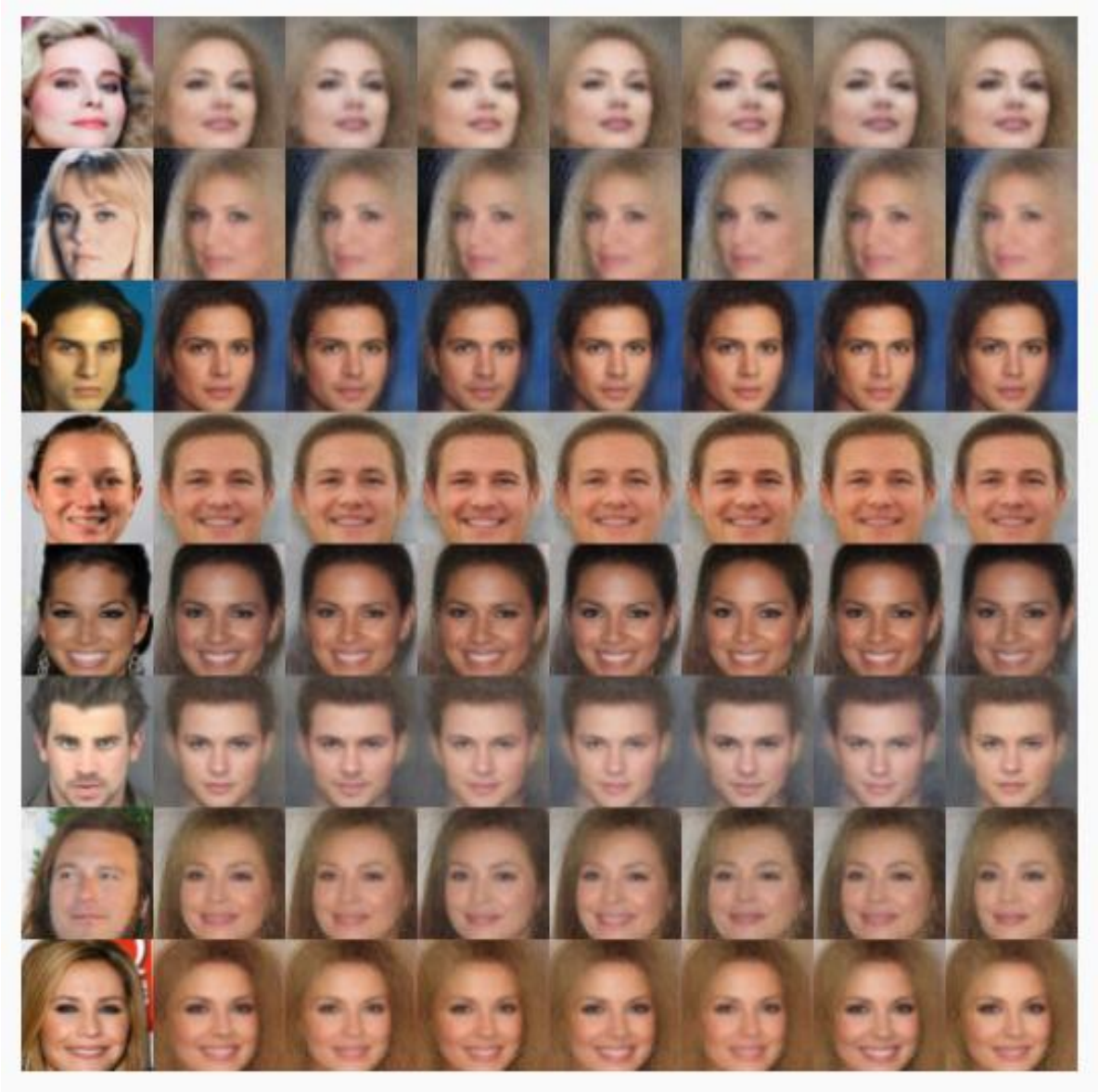
- Hidden layer: parameters of a distribution $\mathcal{N}(\mu, \sigma^2)$
- Distribution used to generate values $z = \mu + \sigma \times \mathcal{N}(0, 1)$
- Enables backprop; reduces variances of gradient



Examples



Examples



Also: https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Discussion

PROS

- A trainable generative model

CONS

- The generative model has a Gaussian distribution at its core: blurry

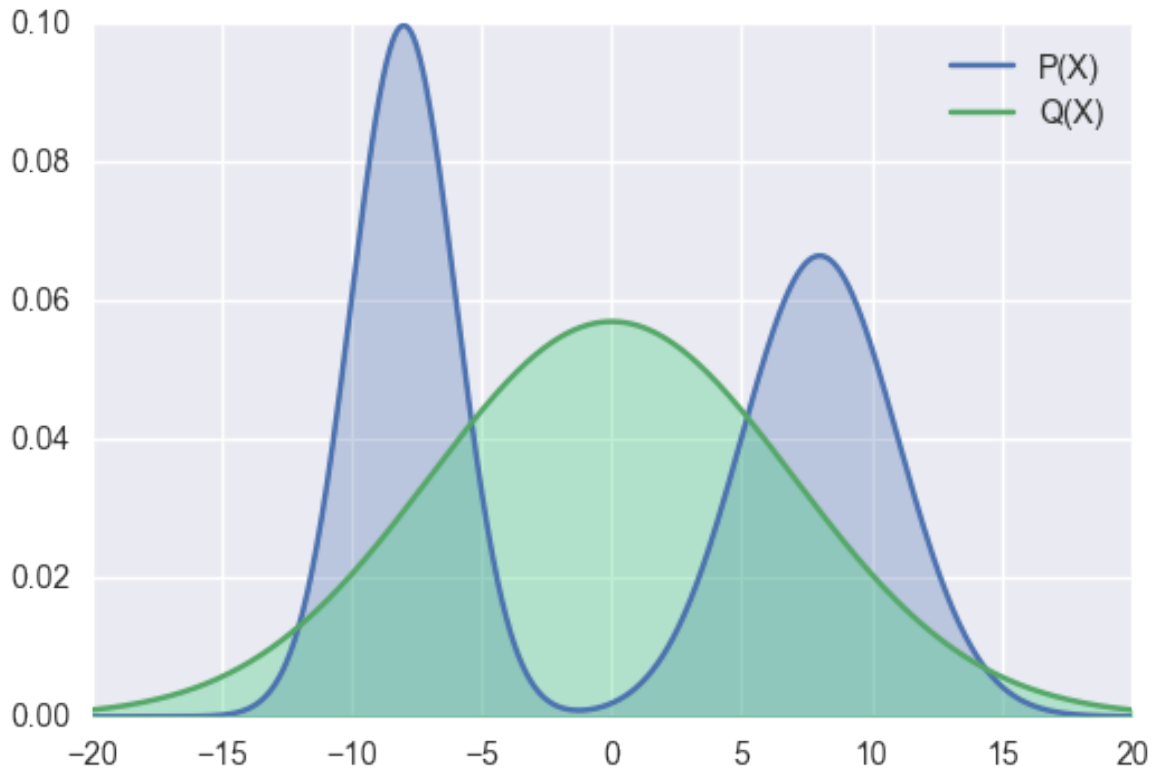
Discussion

PROS

- A trainable generative model

CONS

- The generative model has a Gaussian distribution at its core: blurry



Variational Auto-Encoders

Generative Adversarial Networks

Domain Adaptation

Generative Adversarial Networks

Goodfellow et al., 14

Goal: Find a generative model

- Classical: learn a distribution hard
- Idea: replace a distribution evaluation by a 2-sample test

Principle

- Find a good generative model, s.t. generated samples **cannot be discriminated** from real samples

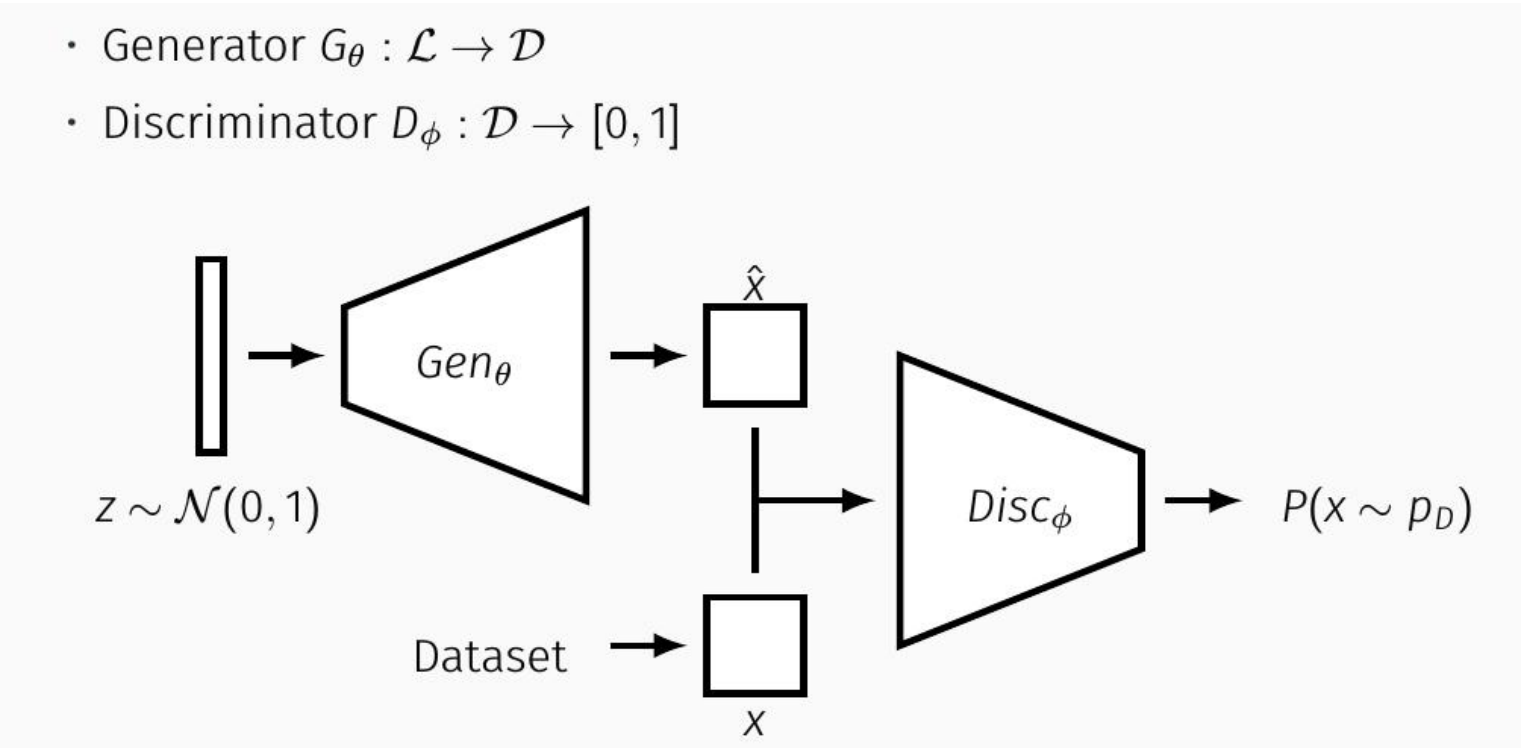
(not easy)

Principle

Goodfellow, 2017

Elements

- True samples **x** (**real**)
- A generator G (variational auto-encoder):
  generates from **x** (**reconstructed**) or from scratch (fake)
- A discriminator D: discriminates *fake* from others (**real** and **reconstructed**)



Principle, 2

Goodfellow, 2017

Mechanism

- Alternate minimization
- Optimize D to tell fake from rest
- Optimize G to deceive D Turing test

$$Min_G \ Max_D \mathbb{E}_{x \in data}[\log(D(\mathbf{x}))] + \mathbb{E}_{z \sim p_x(z)}[\log(1 - D(z))]$$

Caveat

- The above loss has a vanishing gradient problem because of the terms in $\log(1 - D(z))$.
- We can replace it with $-\log((1 - D(z)/D(z))$, which has the same fixed point (the true distribution) but doesn't saturate.

GAN vs VAE

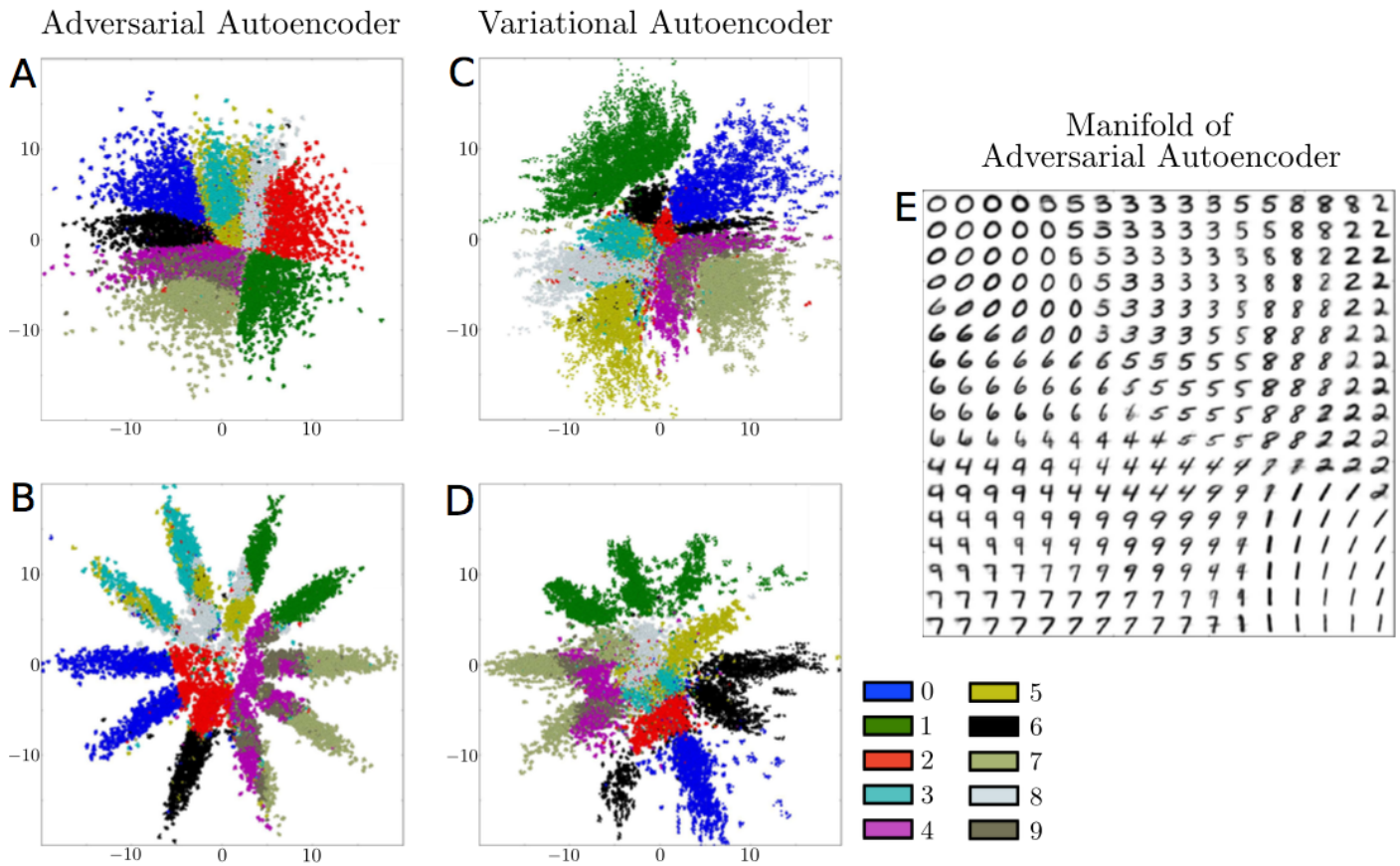


Figure 2: Comparison of adversarial and variational autoencoder on MNIST. The hidden code **z** of the *hold-out* images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) a 2-D gaussian and (d) a mixture of 10 2-D Gaussians. (e) Images generated by uniformly sampling the Gaussian percentiles along each hidden code dimension **z** in the 2-D Gaussian adversarial autoencoder.

Generative adversarial networks

Goodfellow, 2017

Generative adversarial networks, 2

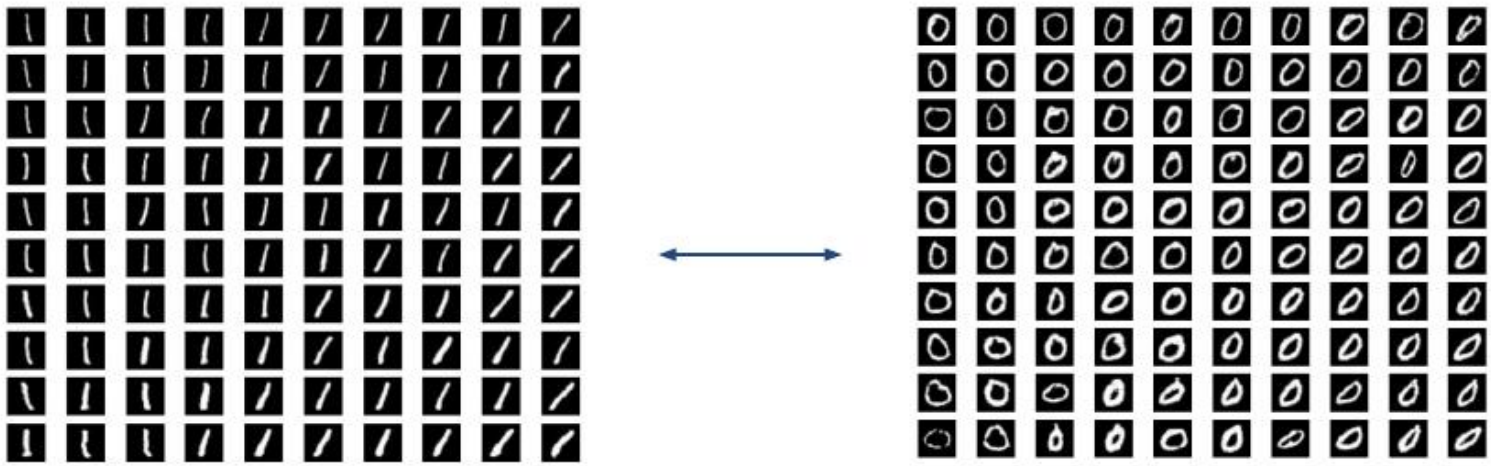
Goodfellow, 2017

Limitations

Training instable

co-evolution of Generator / Discriminator

Mode collapse



Limitations, 2

Generating monsters






(Goodfellow 2016)

Towards Principled Methods for Training Generative Adversarial Networks

Arjovsky, Bottou 17

Why minimizing KL fails

$$\text{Minimizing } KL(P_{real}||P_{gen}) = \int P_{real} \log \frac{P_{real}}{P_{gen}}$$

- For P_{real} high and P_{gen} low (mode dropping), high cost
- For P_{real} low and P_{gen} high (gen. monsters), no cost

The GAN solution: minimizing

$$\mathbb{E}_{x\sim P_r}[\log D(x)] + \mathbb{E}_{x\sim P_g}[\log 1 - D(x)]$$

with

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

i.e., up to a constant, GAN minimizes

$$JS(P_{real}, P_{gen}) = \frac{1}{2}\left(KL(P_{real}||M) + KL(P_{gen}||M)\right)$$

with $M = \frac{1}{2}(P_{real} + P_{gen})$

Towards Principled Methods for Training Generative Adversarial Networks, 2

Arjovsky, Bottou 17

Unfortunately
If P_r and P_g lie on non-aligned manifolds, exists a perfect discriminator; this is the end of optimization !

Proposed alternative: use Wasserstein distance

$$min_G \, max_D \, \mathbb{E}_{x \sim P_g}[D(x)] - \mathbb{E}_{x \sim P_r}[D(x)] = min_G \, W(P_r, P_g)$$

Does not solve all issues !

Pb of vanishing/exploding gradients in WGAN, addressed through weight clipping careful tuning needed

New Regularizations

Improved Training of Wasserstein GANs

Gulrajani, Ahmed, Arjovsky, Dumoulin, Courville 17

Stabilizing Training of Generative Adversarial Networks through Regularization

Roth, Lucchi, Nowozin, Hofmann, 17

Which Training Methods for GANs do actually Converge?

Mescheder, Geiger and Nowozin, 18

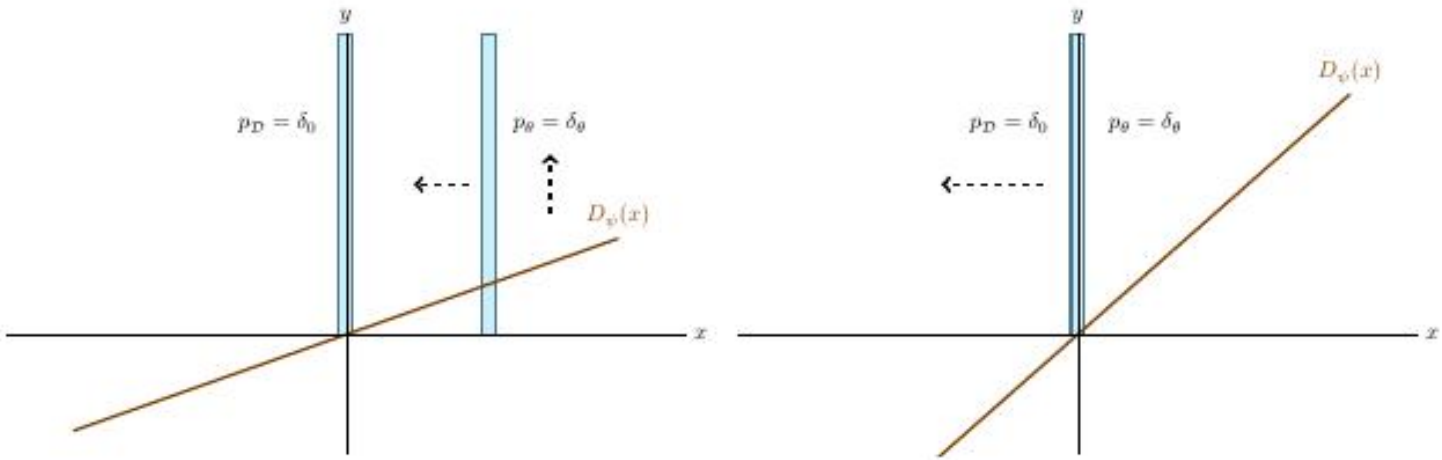
Simple experiments, simple theorems are the building blocks that help us understand more complicated systems. Ali Rahimi - Test of Time Award speech, NIPS 2017

Which Training Methods for GANs do actually Converge? 2

Mescheder, Geiger and Nowozin, 18

Toy example

$$P_r = \delta_0 \qquad P_g = \delta_\theta$$



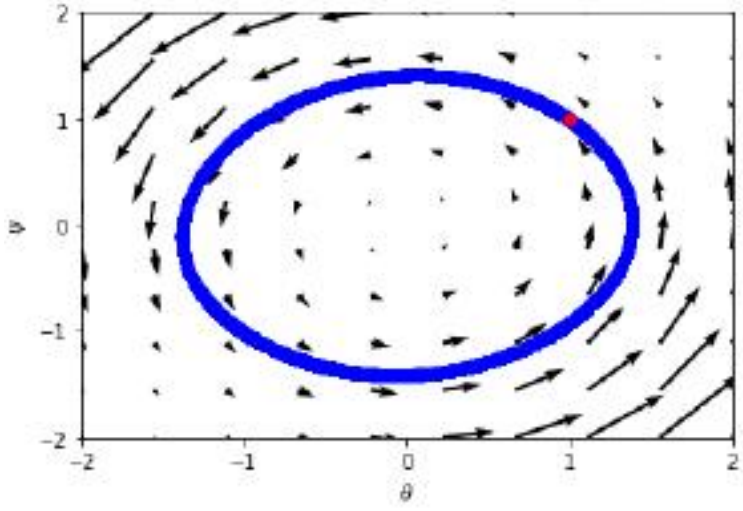
(a) $t = t_0$ (b) $t = t_1$

Figure 1. Visualization of the counterexample showing that in the general case, gradient descent GAN optimization is not convergent: (a) In the beginning, the discriminator pushes the generator towards the true data distribution and the discriminator's slope increases. (b) When the generator reaches the target distribution, the slope of the discriminator is largest, pushing the generator away from the target distribution. This results in oscillatory training dynamics that never converge.

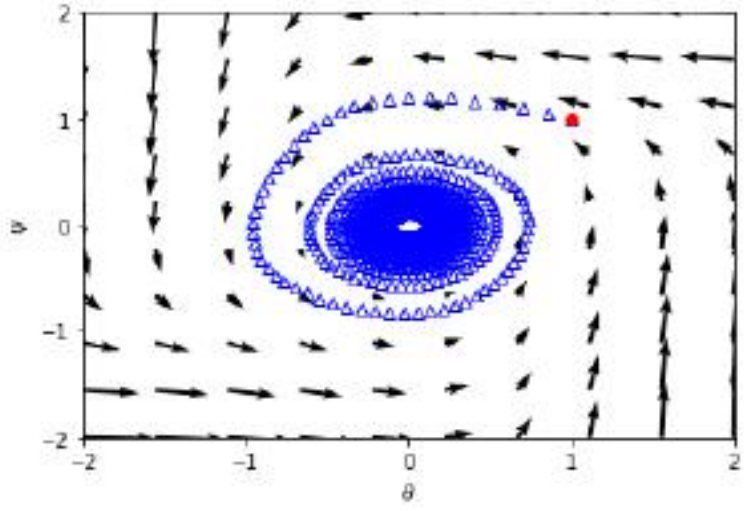
Which Training Methods for GANs do actually Converge? 2

Mescheder, Geiger and Nowozin, 18

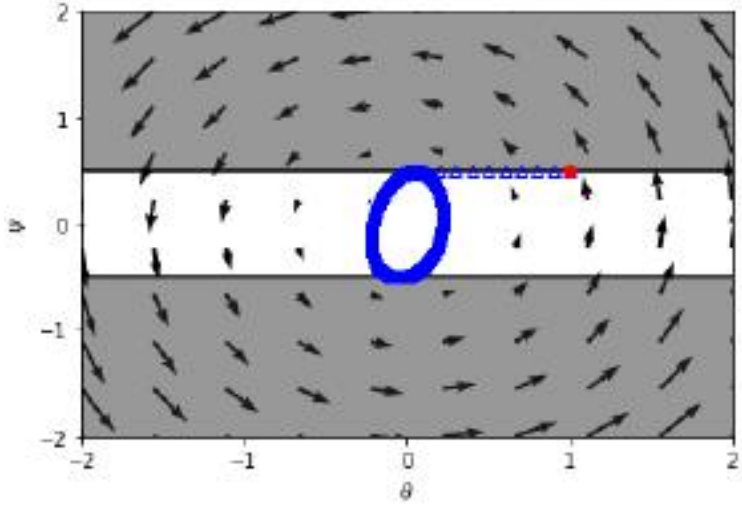
Lesson learned: cyclic behavior for GAN and WGAN



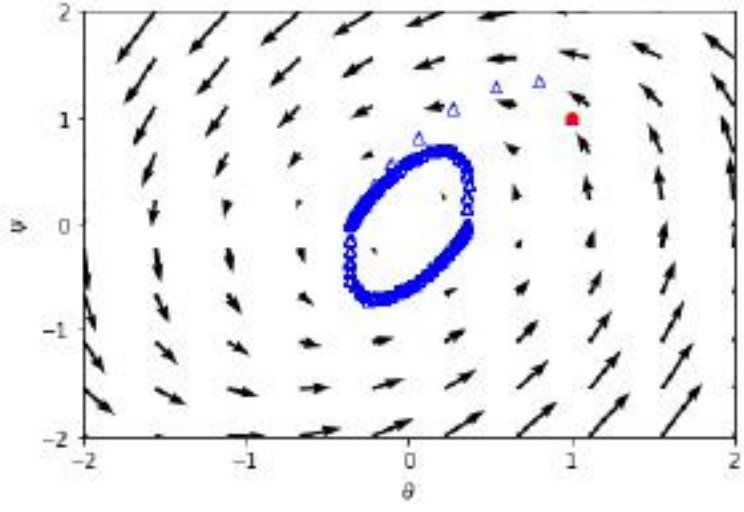
(a) Standard GAN



(b) Non-saturating GAN



(c) WGAN ($n_d = 5$)



(d) WGAN-GP ($n_d = 5$)

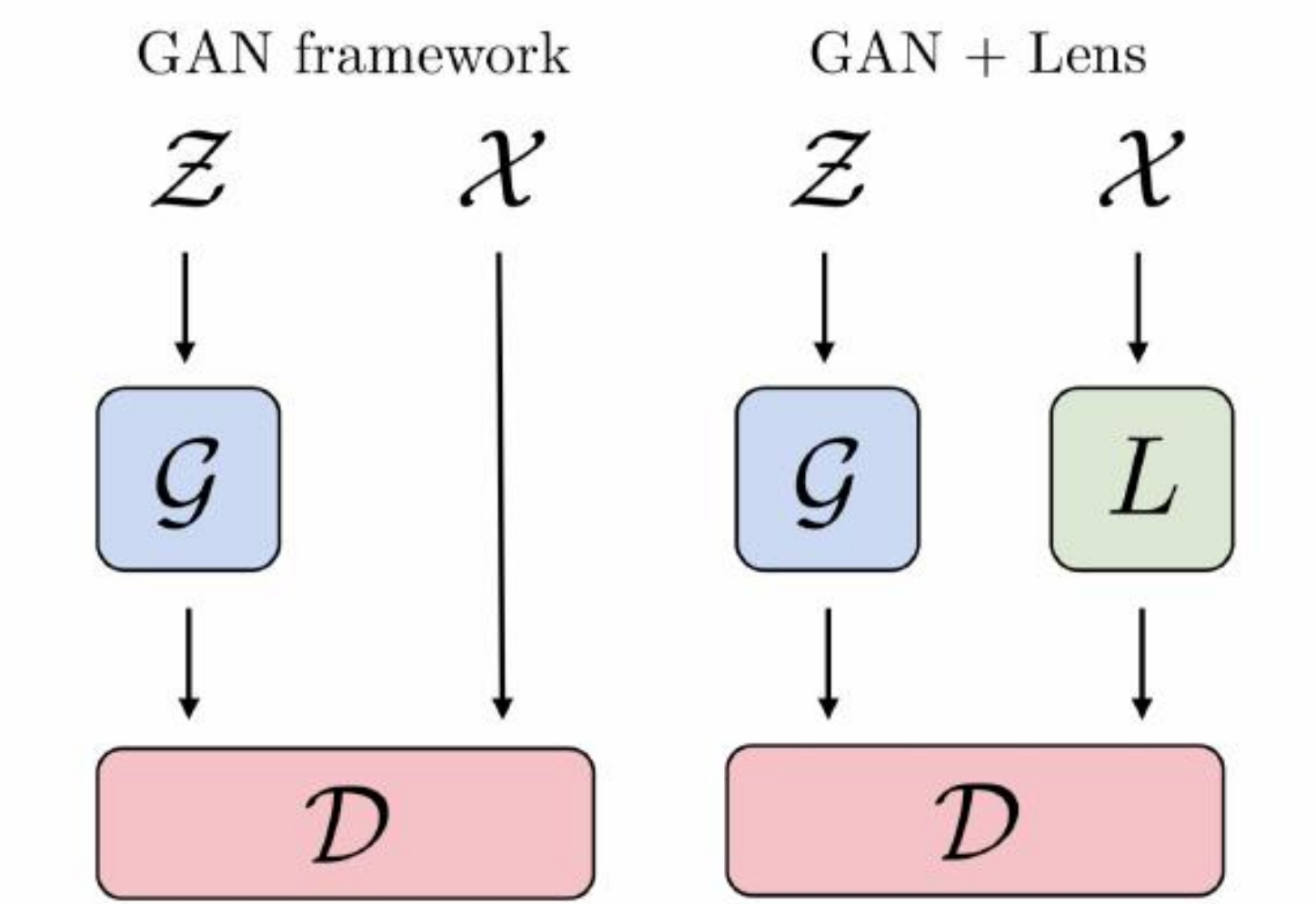
State-of-the-art Generative Adversarial Networks

Mescheder, Geiger and Nowozin, 2018

Tempered Adversarial Networks

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18

Principle: Life too easy for the discriminator !



Tempered Adversarial Networks

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18

Principle: An adversary to the adversary

- $\Rightarrow$ Provide $L(X)$ instead, with L aimed at: i) deceiving the discriminator; ii) staying close from original images

$$min_G max_D \mathbb{E}_{x \sim P_r}[logD(x)] + \mathbb{E}_{x \sim P_g}[log(1 - D(x))]$$

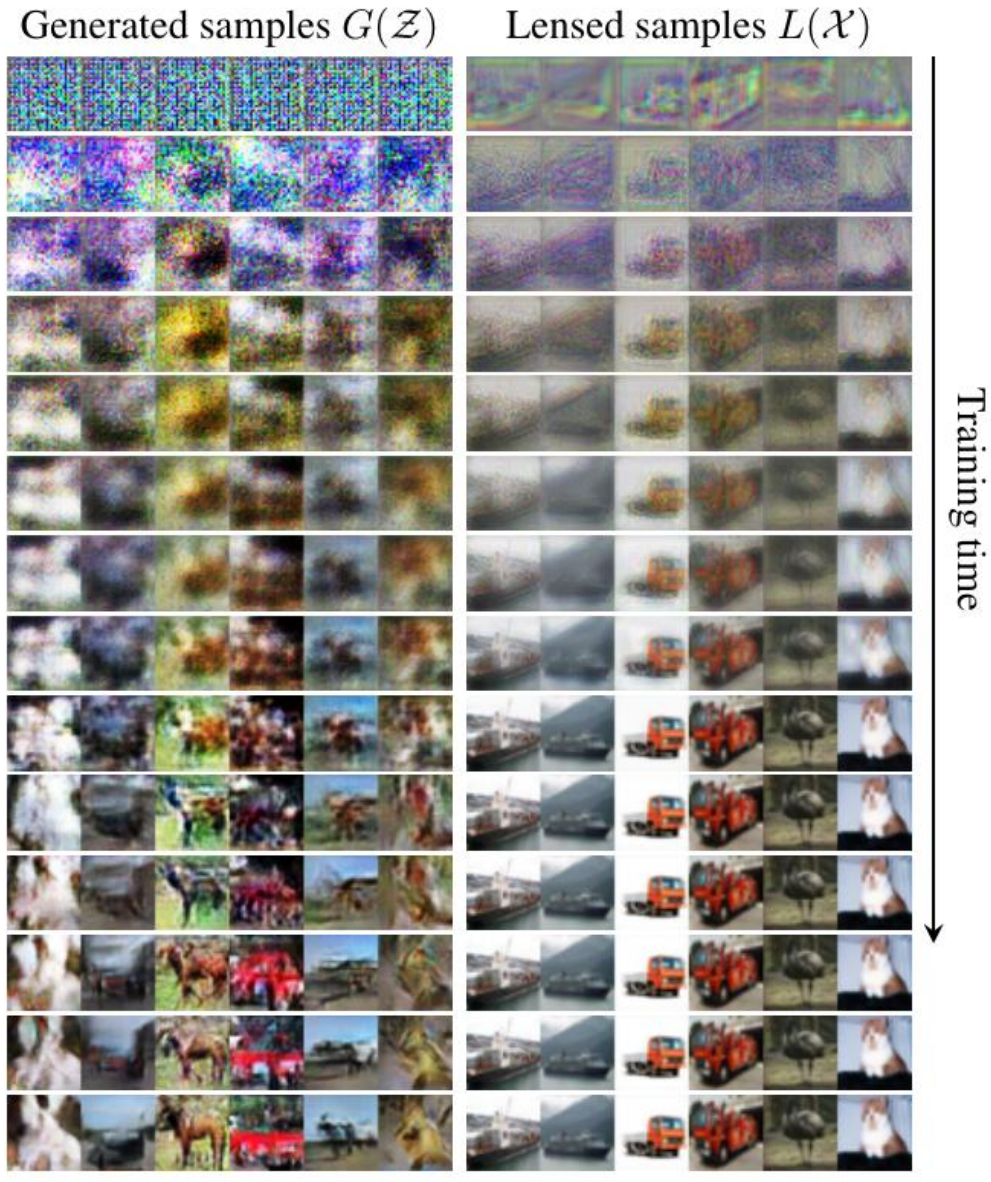
with D trained from $\{(L(x), 1)\} \cup \{G(z), 0\}$ and Lens L optimized

$$L^* = \arg\min -\lambda \mathcal{L}(D) + \sum_i \|L(x_i) - x_i\|^2$$

and $\lambda \to 0$.

Tempered Adversarial Networks, 2

Sajjadi, Parascandolo, Mehrjou, Scholkopf 18



Partial conclusions

- Deep revolution: Learning representations
- Adversarial revolution: a Turing test for machines
- Where is the limitation ?
  VAE: great but blurry
  GAN: great but mode dropping
  the loss function needs more work.

Variational Auto-Encoders

Generative Adversarial Networks

Domain Adaptation

Domain adaptation

Context

- Testing Distribution $\neq$ Training Distribution
- Goal: learn from source distribution $\mathcal{D}_S$, apply on target distribution $\mathcal{D}_T$

What can change ?

- $p(\mathbf{x})$ distribution of instances
- $p(y|\mathbf{x})$ conditional distribution

Domain adaptation, 2

Motivations

- A source domain $\mathcal{D}_S$

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), i = 1 \dots n\}$$

- A target domain $\mathcal{D}_T$
  - Without labels, or
  - With few labels

Goal

Use source data to improve / speed-up learning on target data

Examples

- Opinion mining (movies vs books, hifi vs electric devices)
- Character recognition, different fonts

Formalization

Input

$$\mathcal{E} = \{(\mathbf{x}_i, y_i), x_i \in \mathcal{X} = \mathbb{R}^d, y_i \in \mathcal{Y} = \{-1, 1\}, i = 1 \dots n\}$$

$$\mathcal{E}' = \{(\mathbf{x}'_i), x_j \in \mathbb{R}^d, j = 1 \dots n'\}$$

Two distributions: $\mathcal{D}_S$ on $\mathcal{X} \times \{-1, 1\}$ and $\mathcal{D}_T$ on $\mathcal{X} \times \{-1, 1\}$.

Goal: build a classifier with low risk wrt $\mathcal{D}_T$.

$$R_T(h) = Pr_{\mathcal{D}_T}(h(x) \neq y)$$

although we only know $\mathcal{D}_T^X$ (the projection of the target distribution on the instance space).

No magic !

Ben-David et al. 2006, 2010

1. $\mathcal{H}$ Divergence between $\mathcal{D}_S$ and $\mathcal{D}_T$ on $\mathcal{X}$

$$d_X(\mathcal{D}_S, \mathcal{D}_T) = 2 \sup_{h \in \mathcal{H}} |Pr_{\mathcal{D}_T}(h(x) = 1) - Pr_{\mathcal{D}_S}(h(x) = 1)|$$

This divergence is high if there exists h with value 1 on source and 0 on target (or vice versa).

2. Proposition

A good approximation of $\mathcal{H}$ divergence is

$$\widehat{d_X(\mathcal{D}_S, \mathcal{D}_T)} = 2 \left(1 - \min_h \left(\frac{1}{n} \sum_i 1_{h(x_i)=0} + \frac{1}{n'} \sum_j 1_{h(x'_j)=1}\right)\right)$$

The divergence can be approximated by the ability to empirically discriminate between source and target.

Bounding the domain adaptation risk

Ben-David et al. 2006, 2010

3. Theorem

With probability $1-\delta$, if d is the dimension of $\mathcal{H}$,

$$R_T(h) \leq \widehat{R_S(h)} + C\sqrt{\frac{4}{n}(d log\frac{2}{d} + log\frac{4}{\delta})} + \widehat{d_X} + \text{Best possible}$$

and

$$\text{Best possible} = \inf_h \left(R_S(h) + R_T(h)\right)$$

What we want (risk on h wrt $\mathcal{D}_T$) is bounded by:

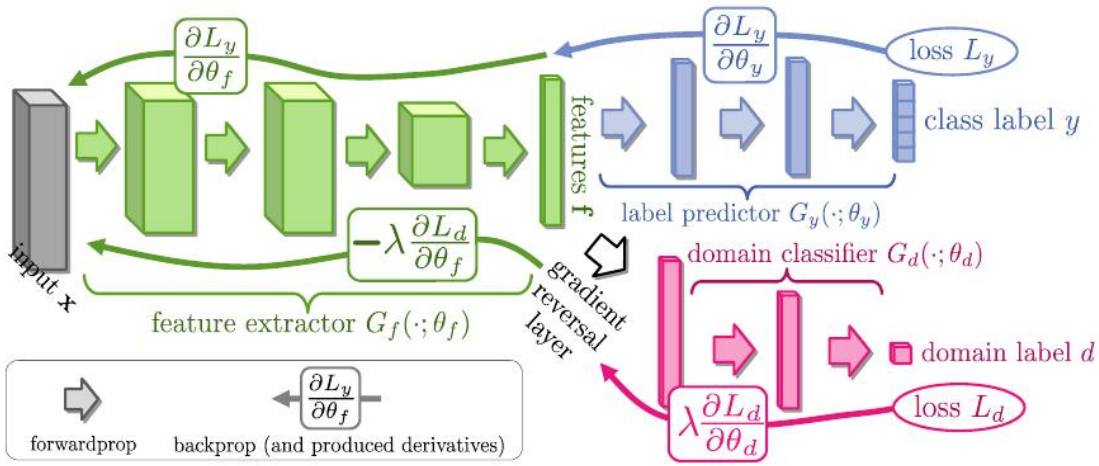
- empirical risk on S
- \+ error related to possible overfitting
- \+ min error one can achieve on both source and target distribution.

Domain Adaptation with Deep NN

Ganin et al. 2016

Two labels

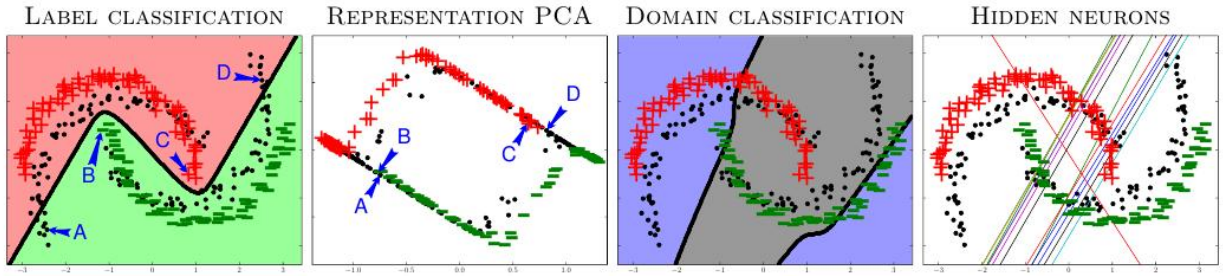
- $(\mathbf{x}, y)$: y is known if $\mathbf{x} \sim \mathcal{D}_S$ source label
  the feature layers help to predict y
- $(\mathbf{x}, z)$: $z = 1$ if $\mathbf{x} \sim \mathcal{D}_S$ and $z = 0$ if $\mathbf{x} \sim \mathcal{D}_T$
  the feature layers **do not want help** to predict z



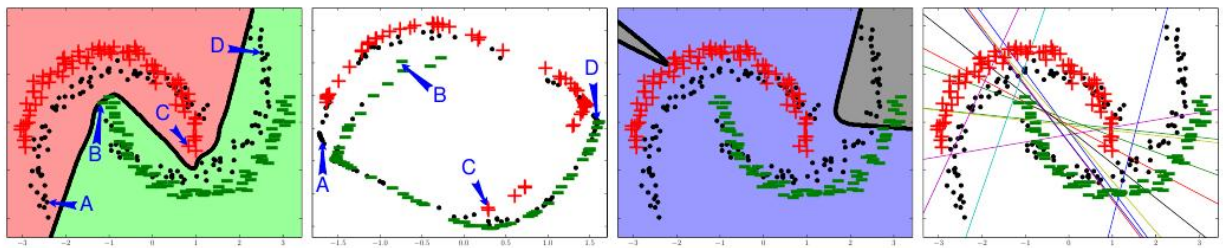
Algorithm

1. Green part + blue part: backpropagation of error wrt $(\mathbf{x}, y)$
2. Red part: backpropagation of error wrt $(\mathbf{x}, z)$
   followed by backpropagation of $-$ gradient of red error.

The intertwinning moons



(a) Standard NN. For the "domain classification", we use a *non adverserial* domain regressor on the hidden neurons learned by the Standard NN. (This is equivalent to run Algorithm 1, without Lines 22 and 31)

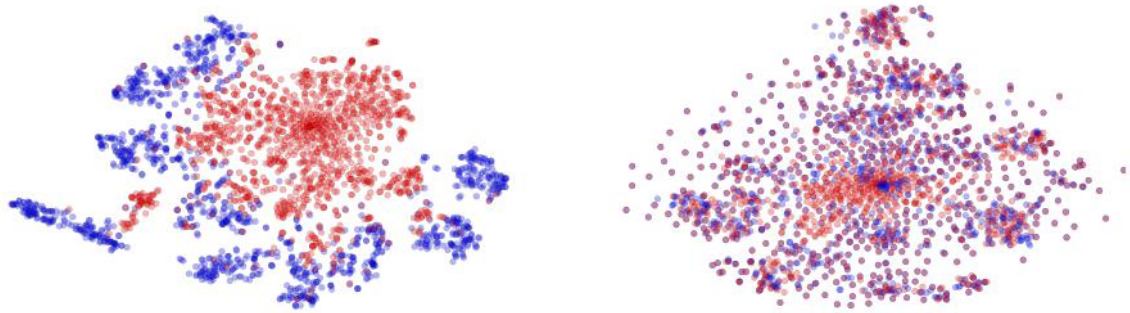


(b) DANN (Algorithm 1)

- left: the decision boundary
- 2nd left: apply PCA on the feature layer
- 3rd left: discrimination source vs target
- right: each line corresponds to hidden neuron = .5

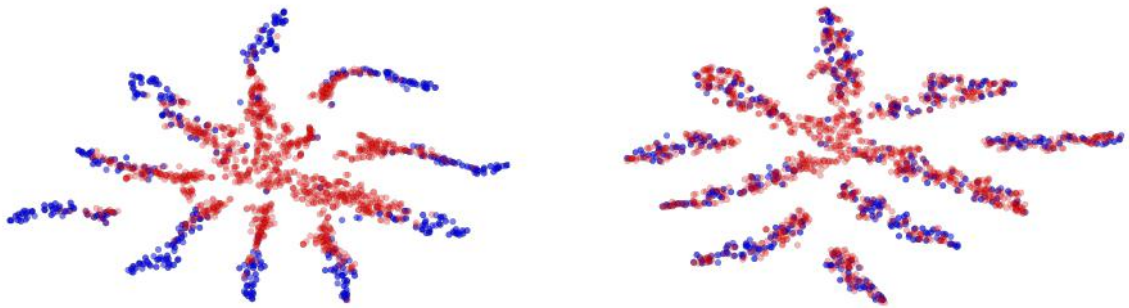
An image domain adaptation

MNIST → MNIST-M: top feature extractor layer



(a) Non-adapted (b) Adapted

SYN NUMBERS → SVHN: last hidden layer of the label predictor

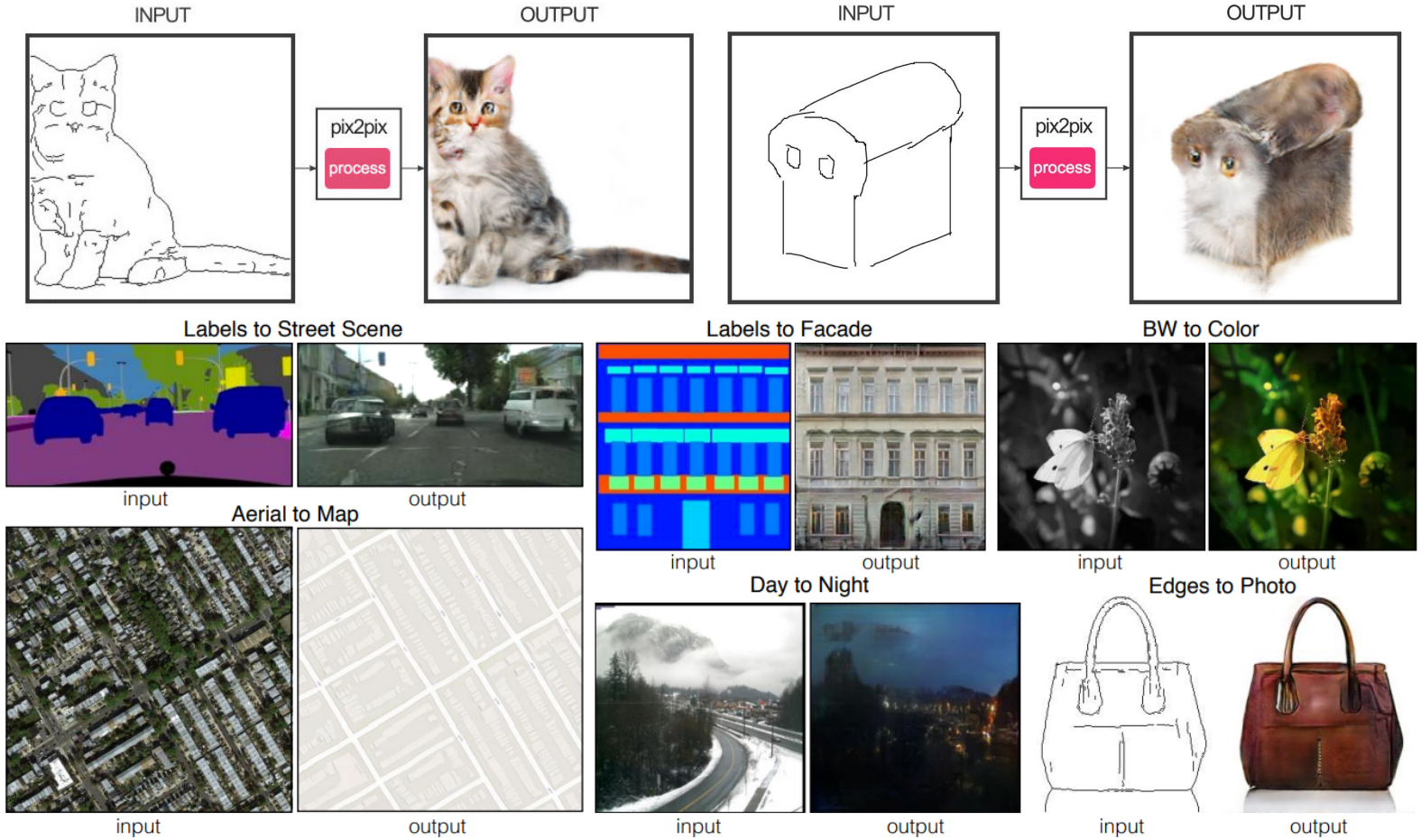


(a) Non-adapted (b) Adapted

More: Playing with distributions

Pix2Pix: Supervised image-to-image translation

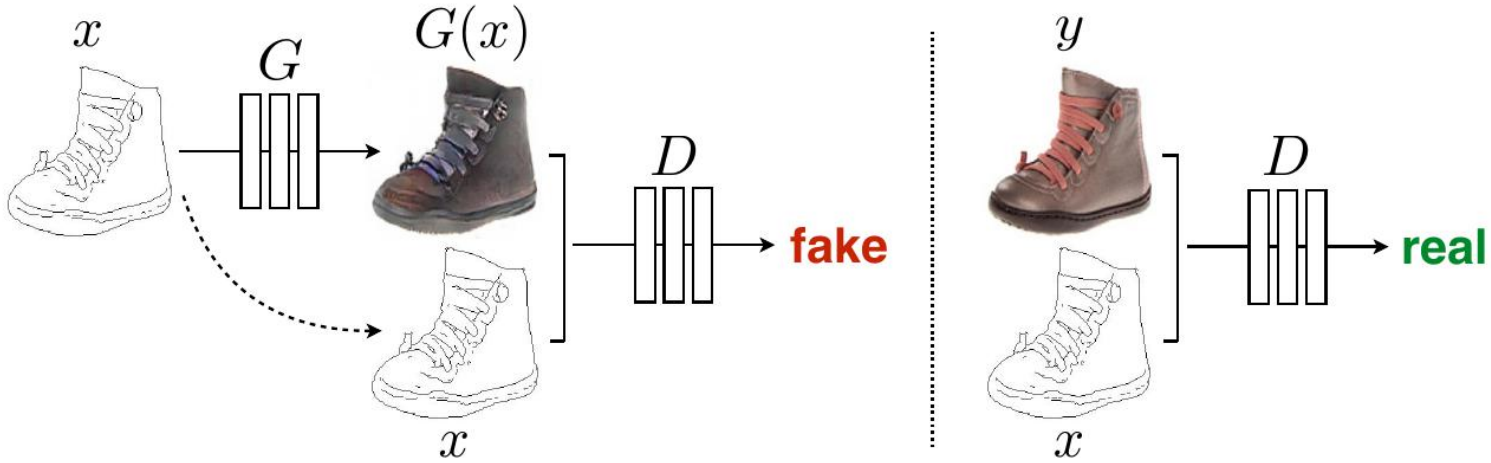
Isola et al., 2016



Pix2Pix: Supervised image-to-image translation, 2

Isola et al., 2016

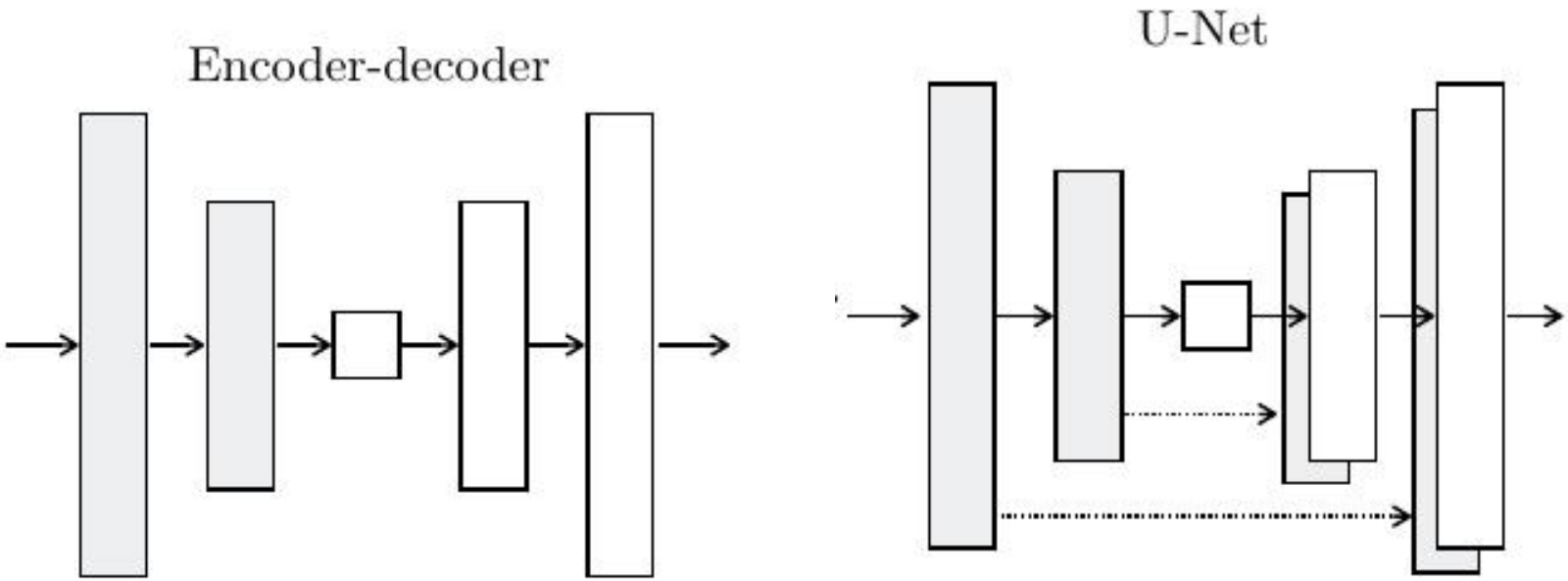
The conditional GAN architecture



Pix2Pix: Supervised image-to-image translation, 3

Isola et al., 2016

The generator architecture



Rationale

- For many image translation problems, there is a great deal of low-level information shared between the input and output, and it would be desirable to shuttle this information directly across the net.

Pix2Pix: Supervised image-to-image translation, 3

Isola et al., 2016

The loss

- $\mathcal{L}_{GAN} = \mathbb{E}_{x_t}[\log(D(x_t))] + \mathbb{E}_{x_s,z}[\log 1 - D(G(x_s, z))]$
- $\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x_s}[\log D(x_s, x_t)] + \mathbb{E}_{x_s,z}[\log(1 - D(x_s, G(x_s, z))]$
- $\mathcal{L}_1 G) = \mathbb{E}_{(x_s,x_t),z}\|x_t - G(x_s, z)\|_1$

Impact of the loss