# Deep Learning

Alexandre Allauzen, Michèle Sebag
CNRS & Université Paris-Sud

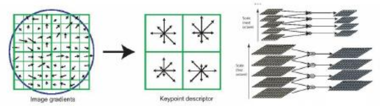
Oct. 17th, 2018

*Credit for slides: Sanjeev Arora; Yoshua Bengio;Yann LeCun; Nando de Freitas; Pascal Germain; Léon Gatys; Weidi Xie; Max Welling; Victor Berger; Kevin Frans; Lars Mescheder et al.; Mehdi Sajjadi et al.*

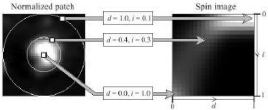
# The Deep ML revolution: what is new ?

## Former state of the art

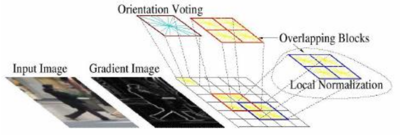
e.g. in computer vision



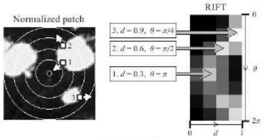

SIFT



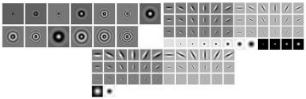

Spin image



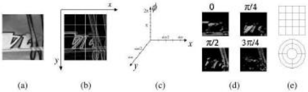

HoG




RIFT



Textons




GLOH

SIFT: scale invariant feature transform
HOG: histogram of oriented gradients

Textons: “vector quantized responses of a linear filter bank”

# What is new, 2

## Traditional approach

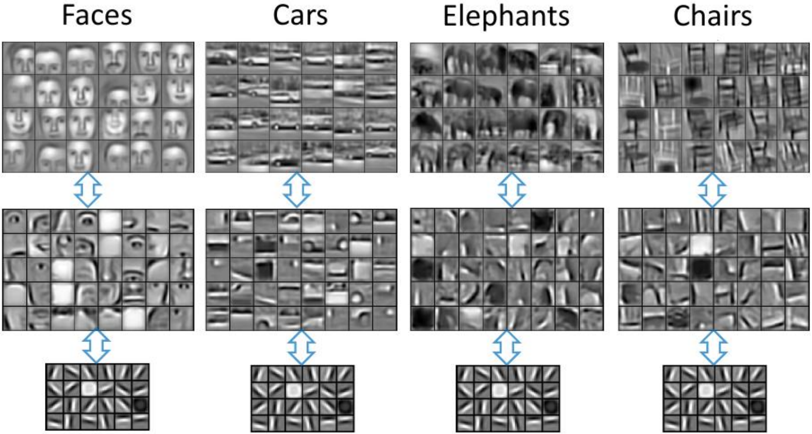
→ Manually crafted features → Trainable classifier

## Deep learning

→ Trainable feature extractor → Trainable classifier

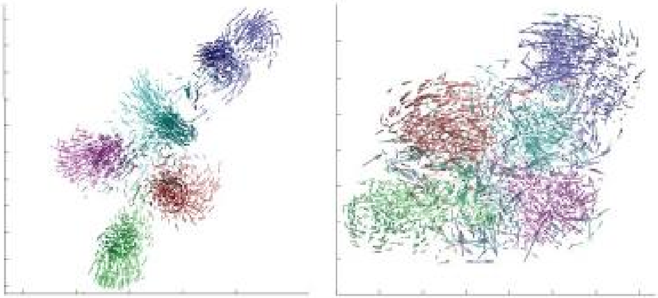
# A new representation is learned

Bengio et al. 2006

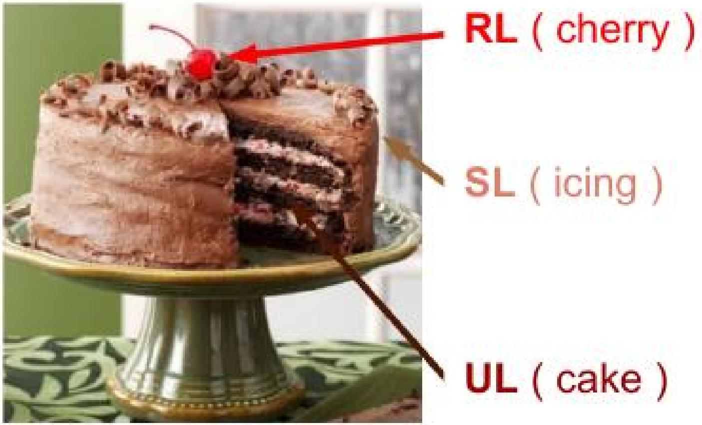


## Good features ?

Le Cun, 2015: https://www.youtube.com/watch?v=Y-XTcTusUxQ



# Losses and labels




**Lesson learned, quasi consensus @ ICML 2018**

- Introduce many auxiliary tasks / losses
- Why ? Smoothens the optimization landscape (conjecture)

# Tutorial Sanjeev Arora @ ICML18

http://unsupervised.cs.princeton.edu/deeplearningtutorial.html

## Key issues

- Optimization: highly non convex
- Overparametrisation / Generalization

$$\text{Min} \quad (w-1)^2 + (w+1)^2 \qquad \text{vs} \qquad \text{Min} \quad (w_1-1)^2 + (w_2+1)^2$$

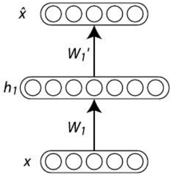
- Role of depth
  more expressivity; filter noise
- Make it simpler ? One game is to find a solution; another game is to simplify it. see also Max Welling's talk.

# Auto-encoders

$$\mathcal{E} = \{\mathbf{x}_i \in \mathbb{R}^D, i = 1 \dots n\}$$
$$\mathbf{x} \longrightarrow h_1 \in \mathbb{R}^d \longrightarrow \hat{\mathbf{x}}$$

- An auto-encoder:

$$\text{Find } W^* = \arg\min_W \left( \sum_i ||W' o W(\mathbf{x}_i) - x_i||^2 \right)$$




(*) Instead of min squared error, use cross-entropy loss:

$$\sum_j \mathbf{x}_{i,j} \log \hat{\mathbf{x}}_{i,j} + (1 - \mathbf{x}_{i,j}) \log (1 - \hat{x}_{i,j})$$

(**) Why $W$ for encoding and $W'$ for decoding ?

# Auto-encoders and Principal Component Analysis

**Assume**

- A single layer
- Linear activation

**Then**

- An auto-encoder with $k$ hidden neurons $\approx$ first $k$ eigenvectors of PCA

**Why ?**

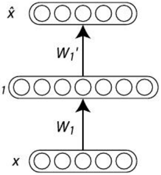
# Stacked auto-encoders were used to initialize deep networks

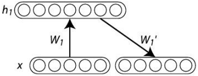
**In the early Deep Learning era...**

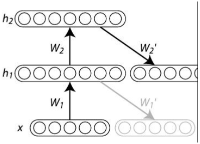
Bengio, Lamblin, Popovici, Larochelle 06

**First layer** $\quad \mathbf{x} \longrightarrow \mathbf{h}_1 \longrightarrow \hat{\mathbf{x}}$

**Second layer** $\quad \mathbf{h}_1 \longrightarrow \mathbf{h}_2 \longrightarrow \hat{\mathbf{h}_1}$ $\quad$ same, replacing $\mathbf{x}$ with $\mathbf{h}_1$



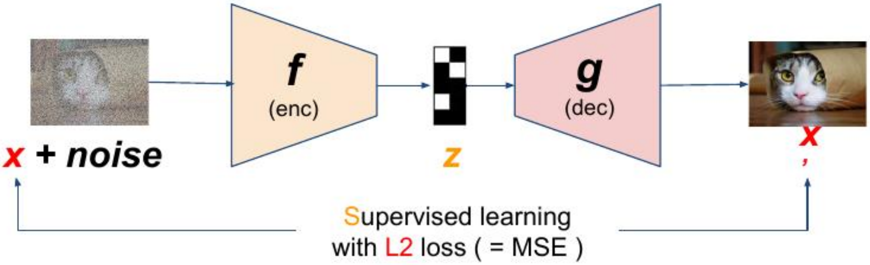
# Denoising Auto-Encoders

Vincent, Larochelle, Bengio, Manzagol, 08

## Principle

- Add noise to **x**
  - Gaussian noise
  - Or binary masking noise (akin drop-out)
- Recover **x**.

$$\text{Find } W^* = \arg\min_W \left( \sum_i ||W' o W(\mathbf{x}_i + \epsilon) - x_i||^2 \right)$$



# Auto-encoders for domain adaptation

Glorot, Bordes, Bengio, 11

## Stacked Denoising Auto-Encoders

- on source and target instances
- use latent representation to learn on source domain

## Why should it work ?

*SDAs are able to disentangle hidden factors which explain the variations in the input data, and automatically group features in accordance with their relatedness to these factors. This helps transfer across domains as these generic concepts are invariant to domain-specific vocabularies.*

**CONS**: Computationally expensive

# Marginalized Denoising Auto-Encoders

Chen, Xu, Weinberger, Sha 12

## Marginalizing a single linear layer

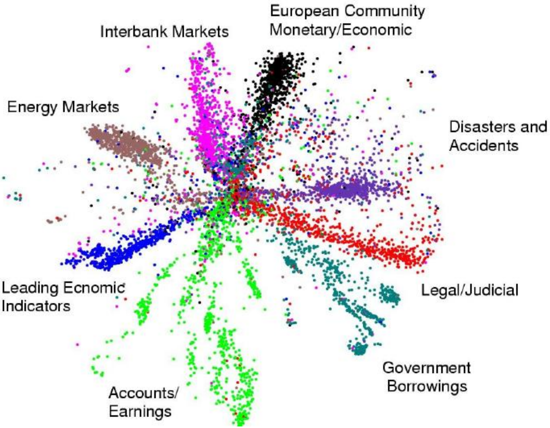
- $\bar{X}$: m copies of $X$, $\tilde{X}$: coordinate value independently zeroed with probability $p$
- Find $W = \arg\min \|\bar{\mathbf{X}} - W\tilde{\mathbf{X}}\|$
- Solution: $W = PQ^{-1}$ with $P = \bar{\mathbf{X}}\tilde{\mathbf{X}}'$ and $Q = \tilde{\mathbf{X}}\tilde{\mathbf{X}}'$
- Consider
$$W = \mathbb{E}(P)\mathbb{E}(Q^{-1}) \text{ as } m \to \infty$$
- Define $q = (1-p, \ldots, 1-p, 1)$ (Last entry is for the bias)
$$\mathbb{E}(Q)_{\alpha,\beta} = \begin{cases} X_\alpha X'_\beta q_\alpha q_\beta & \text{if } \alpha \neq \beta \\ X_\alpha X'_\alpha q_\alpha & \text{otherwise} \end{cases}$$

## Then

- Inject non-linearity on the top of $W\tilde{X}$ consider $\sigma(W\tilde{X})$
- Use linear classifier or SVM.

# Visualization with (non linear) Autoencoders

For $d = 2$,




**dimensionality reduction on the latent representation**

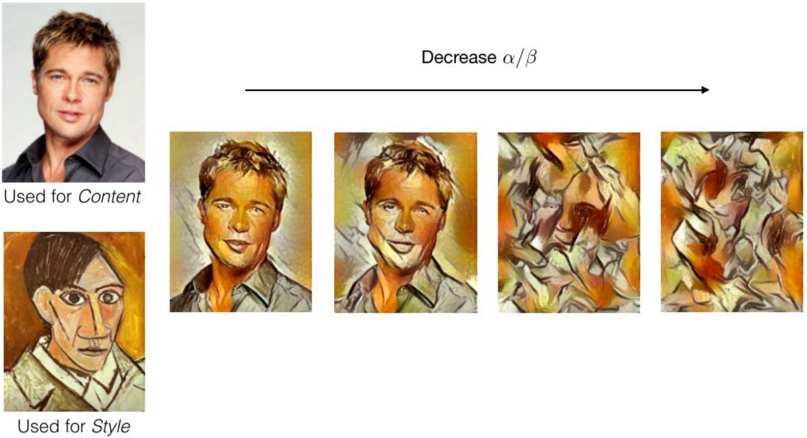
- Multidimensional scaling
- t-Distributed Stochastic Neighbor Embedding (t-SNE)

vdMaaten, Hinton 08

t-SNE Do and Don't https://distill.pub/2016/misread-tsne/

# Morphing of representations

Gatys et al. 15, 16

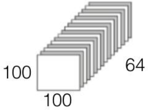



- Style and contents in a convolutional NN are separable
- Use a trained VGG-19 Net:
  - applied on image 1 (content)
  - applied on image 2 (style)
  - find input matching hidden representation of image 1 (weight $\alpha$) and hidden representation of image 2 (weight $\beta$)

# The style

Gatys et al. 15, 16

- Style representation: **correlations** between the different filter responses over the spatial extent of feature maps.
  - Provide colours and local structures.
- Synthesize texture by matching correlation matrices calculated from different layers.
- Key equations: (Check paper for notation)

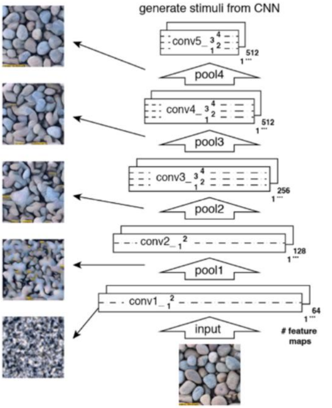



$$\longrightarrow \left[\; F \in 64 \times 10000 \;\right]$$

$$G^l = F^l (F^l)^T$$ Correlation matrix

$$E_l = \frac{1}{Norm} \sum_{i,j} (G^l_{ij} - A^l_{i,j})^2$$ Cost for style reconstruction

$$Loss_{style} = \sum_{l=0}^{L} w_l E_l$$ Accumulate cost for lower layers

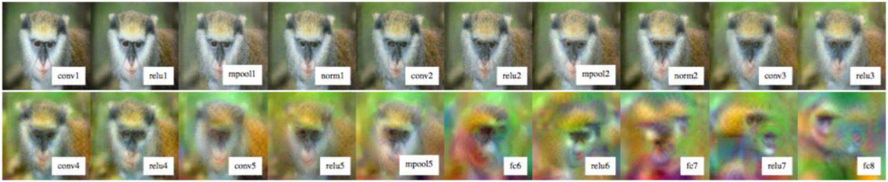



Portilla & Simoncelli, 2000

Gatys, et al. 2015

# The content

Gatys et al. 15, 16



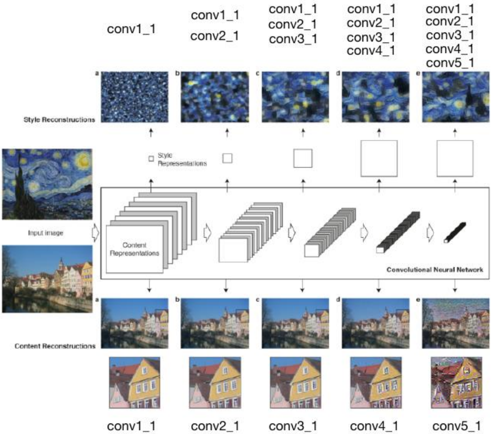

**Finally**

Use image $\mathbf{x}_0$ for content (AE $\phi$), $\mathbf{x}_1$ for style (AE$\phi'$)
Morphing: Find input image $\mathbf{x}$ minimizing

$$\alpha\|\phi(\mathbf{x}) - \phi(\mathbf{x}_0)\| + \beta\langle\phi'(\mathbf{x}), \phi'(\mathbf{x}_1)\rangle$$

# Morphing of representations, 2

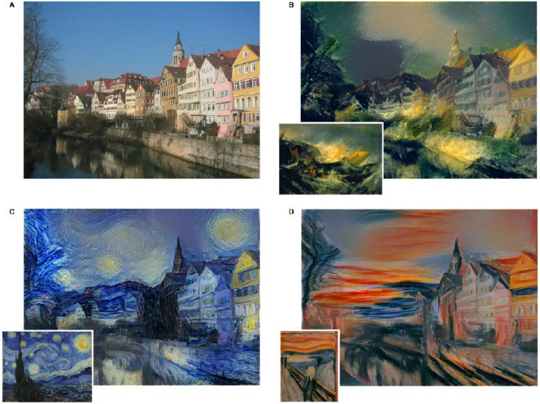
Gatys et al. 15, 16



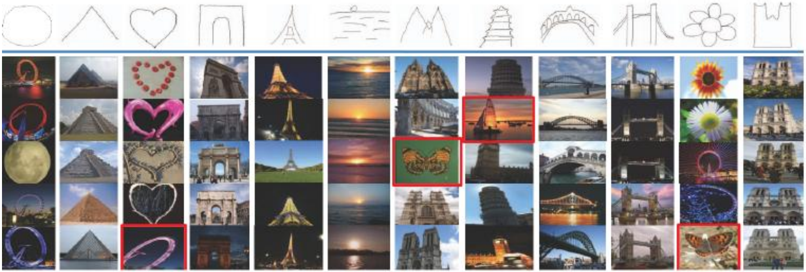

- Contents (bottom): convolutions with decreasing precision
- Style (top): correlations between the convol. features

# Morphing of representations, 2

Gatys et al. 15, 16



# Siamese Networks



Classes or similarities ?

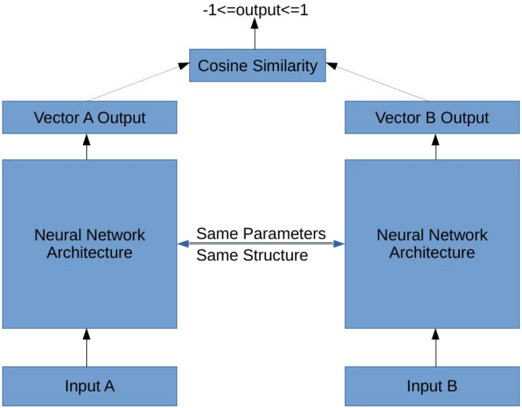
# Siamese Networks

Bromley et al. 93

## Principle

- Neural Networks can be used to define a latent representation
- Siamese: optimize the related metrics

## Schema



# Siamese Networks, 2

## Data

$$\mathcal{E} = \{x_i \in \mathbb{R}^d, i \in [[1, n]]\}; \mathcal{S} = \{(x_{i,\ell}, x_{j_\ell}, c_\ell) \; s.t. \; c_\ell \in \{-1, 1\}, \ell \in [[1, L]]\}$$

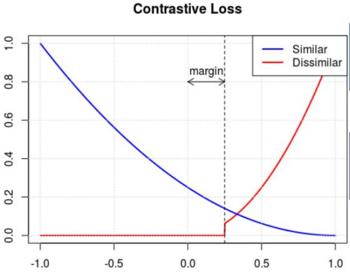
## Experimental setting

- Often: few similar pairs; by default, pairs are dissimilar
- Subsample dissimilar pairs (optimal ratio between 2/1 ou 10/1)
- Possible to use domain knowledge in selection of dissimilar pairs

# Loss

## Given similar and dissimilar pairs ($E_+$ and $E_-$)

$$\mathcal{L} = \sum_{(i,j) in E_+} L_+(i,j) + \sum_{(k,\ell) in E_-} L_-(k,\ell)$$




$$L_+ = \frac{1}{4} \cdot (1 - cosine(x_1, x_2))^2$$

$$L_- = \begin{cases} \text{cosine}(x_1, x_2)^2, & \text{if } x \geq \text{margin} \\ 0, & \text{otherwise} \end{cases}$$

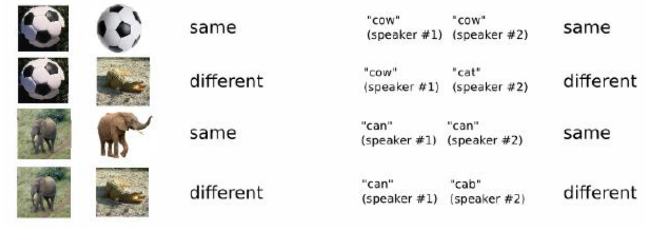
# Applications

- Signature recognition
- Image recognition, search
- Article, Title
- Filter out typos
- Recommandation systems, collaborative filtering

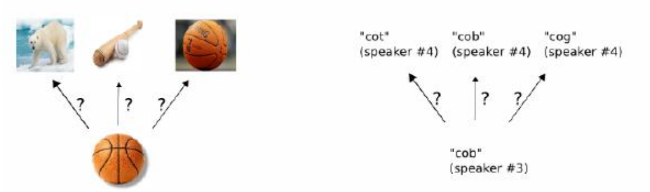
# Siamese Networks for one-shot image recognition

Koch, Zemel, Salakhutdinov 15

## Training similarity



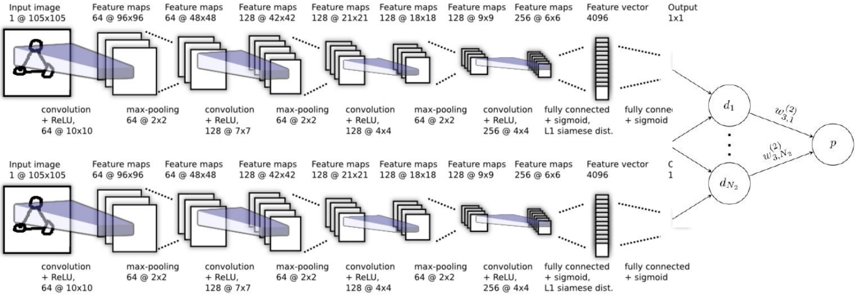

## One-shot setting



# Ingredients

Koch et al. 15

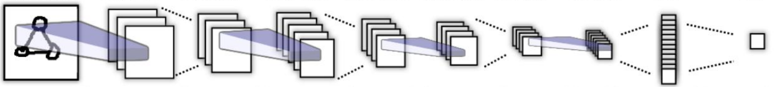
## Architecture



# Ingredients, 2

Koch et al. 15

## Architecture




## Distance

$$d(x, x') = \sigma \left( \sum_k \alpha_k |z_k(x) - z_k(x')| \right)$$

## Loss

Given a batch $((x_i, x_i'), y_i)$ with $y_i = 1$ iff $x_i$ and $x_i'$ are similar

$$\mathcal{L}(w) = \sum_i y_i \log d(x_i, x_i') + (1 - y_i) \log (1 - d(x_i, x_i')) + \lambda \|w\|^2$$

# Results

Koch et al. 15

## Omniglot


## Results

| Method | Test |
|---|---|
| **Humans** | 95.5 |
| **Hierarchical Bayesian Program Learning** | 95.2 |
| **Affine model** | 81.8 |
| **Hierarchical Deep** | 65.2 |
| **Deep Boltzmann Machine** | 62.0 |
| **Simple Stroke** | 35.2 |
| **1-Nearest Neighbor** | 21.7 |
| **Siamese Neural Net** | 58.3 |
| **Convolutional Siamese Net** | 92.0 |

# Siamese Networks

**PROS**

- Learn metrics, invariance operators
- Generalization beyond train data

**CONS**

- More computationally intensive
- More hyperparameters and fine-tuning, more training

# 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

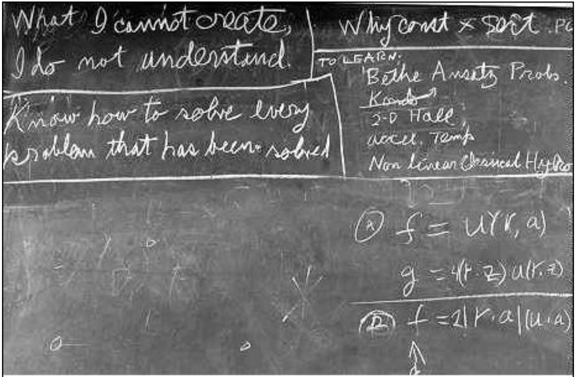
- A compressed (latent) representation

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

- 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

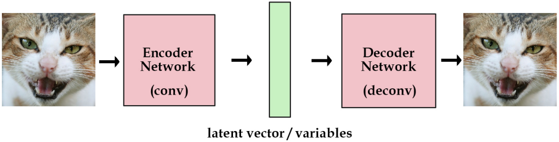


# Variational Auto-Encoders

Kingma Welling 14, Salimans Kingma Welling 15

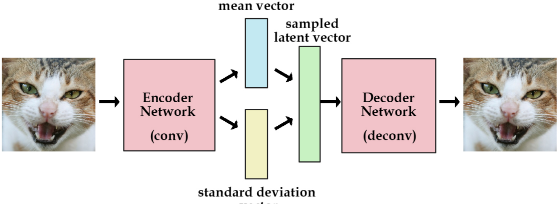
**What we have**:

- $Enc$ 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

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

**$\equiv$ 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 $\mathbf{z}$ 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

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

**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$(\mathbf{x}_1, \ldots \mathbf{x}_k) = -\sum_i p(\mathbf{x}_i) log(p(\mathbf{x}_i))$
- 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 $\mathbf{L}(\mathbf{q}(\mathbf{z}))$

# Evidence Lower Bound, 3

## More formula massaging

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

# 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

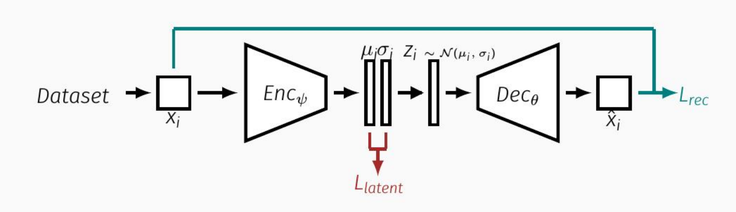
- Exercize: back to the cross-entropy loss

# Variational auto-encoders

Kingma et al. 13

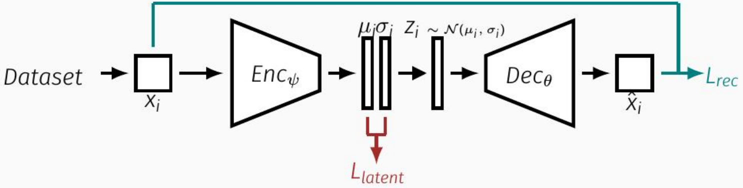
## Position

- Like an auto-encoder (data fitting term) with a regularizer, the KL divergence between the distribution of the hidden variables **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 $\mu$ and $\Sigma$



# 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:

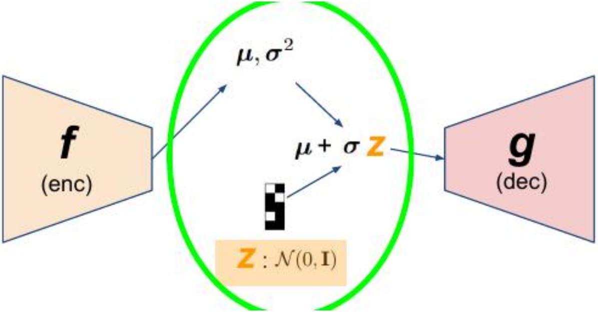
$$L_{rec} = \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} \left[-log\, p_{Dec_\theta(z)}(x_i)\right]$$

$$= \sum_i \mathbb{E}_{z \sim Enc_\psi(x_i)} ||Dec_\theta(z) - x_i||^2 + cst.$$

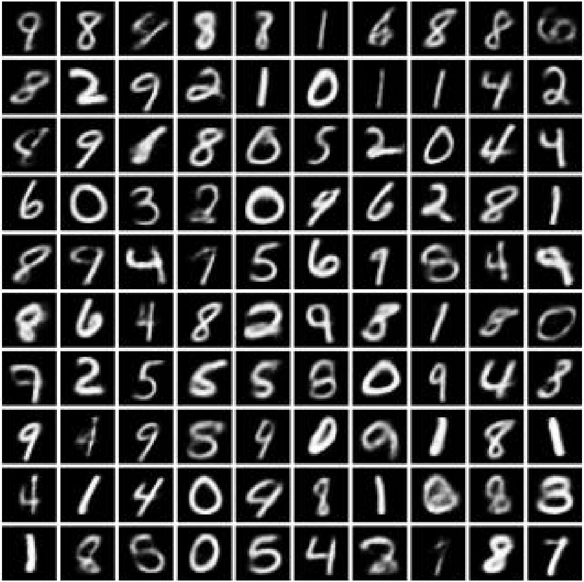
# The reparameterization trick

## Principle

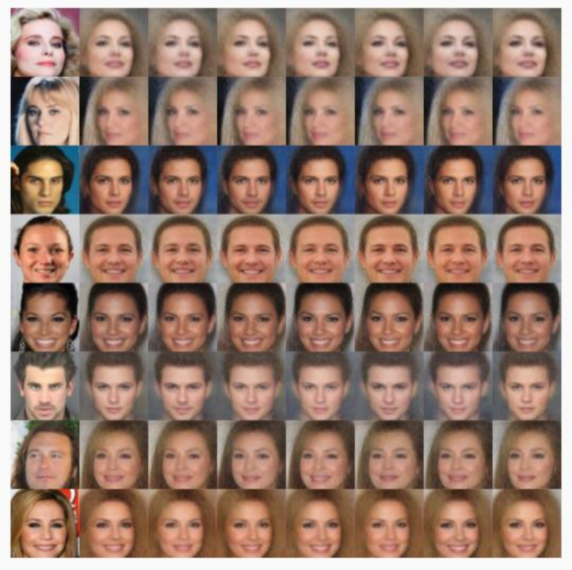
- Hidden layer: parameters of a distribution $\mathcal{N}(\mu, \sigma^2)$
- Distribution used to generate values $z = \mu + \sigma \times \mathcal{N}(0, 1)$
- Enables backprop; reduces variances of gradient



# 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

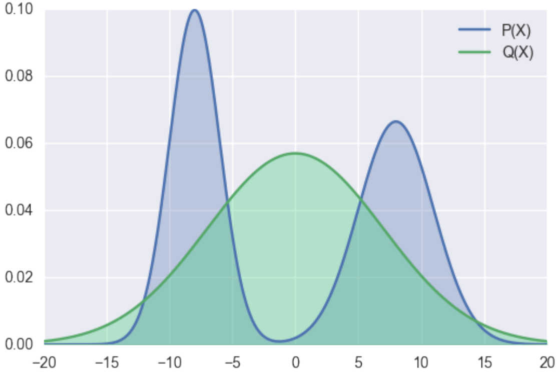
# Discussion

## PROS

- A trainable generative model

## CONS

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



Representation is everything

Auto-Encoders

- Denoising Auto-Encoders
- Exploiting latent representations

Siamese Networks

Variational Auto-Encoders

Generative Adversarial Networks

# 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

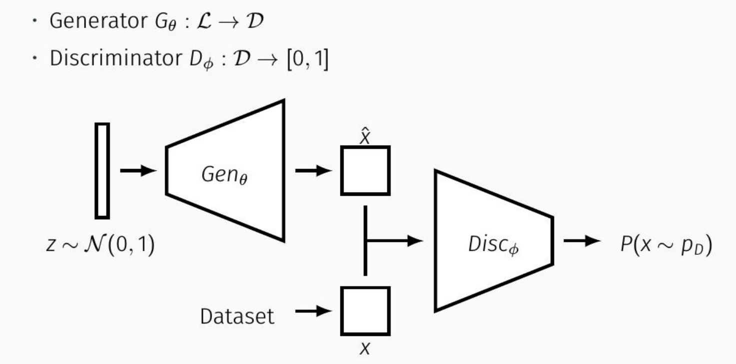
(not easy)

# 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

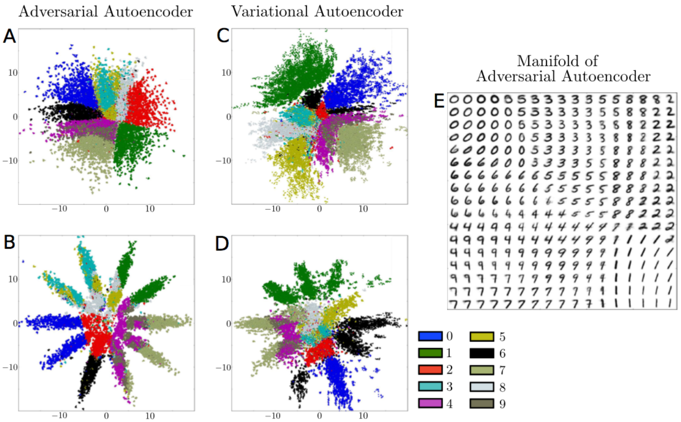



Figure 2: Comparison of adversarial and variational autoencoder on MNIST. The hidden code **z** of the *hold-out* images for an adversarial autoencoder fit to (a) a 2-D Gaussian and (b) a mixture of 10 2-D Gaussians. Each color represents the associated label. Same for variational autoencoder with (c) 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

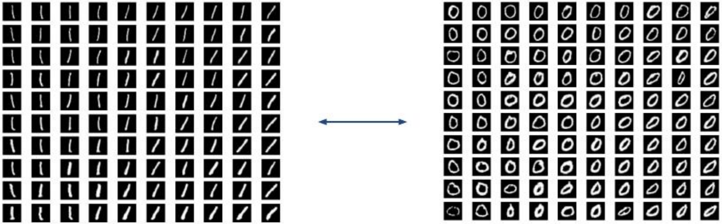
Goodfellow, 2017

# 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

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

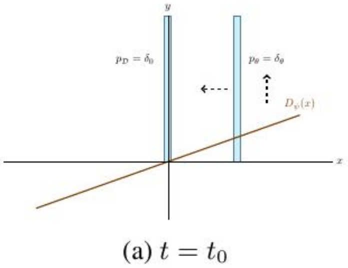
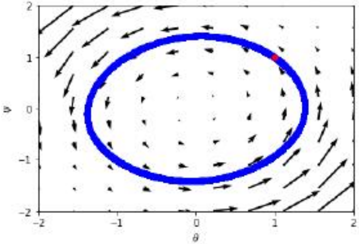



*Figure 1.* Visualization of the counterexample showing that in the general case, gradient descent GAN optimization is not convergent: (a) In the beginning, the discriminator pushes the generator towards the true data distribution and the discriminator's slope increases. (b) When the generator reaches 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.

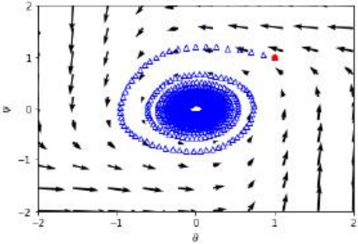
# Which Training Methods for GANs do actually Converge? 2

Mescheder, Geiger and Nowozin, 18

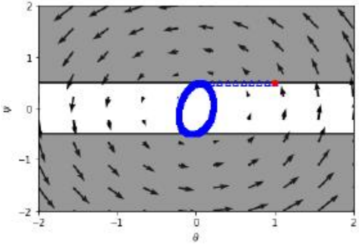
**Lesson learned**: cyclic behavior for GAN and WGAN



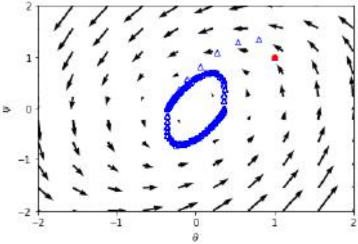

(a) Standard GAN




(b) Non-saturating GAN




(c) WGAN ($n_d = 5$)




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

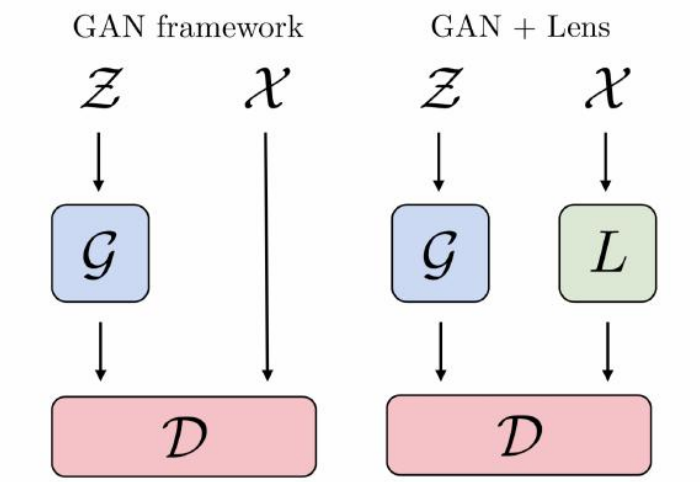
# State-of-the-art Generative Adversarial Networks

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

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

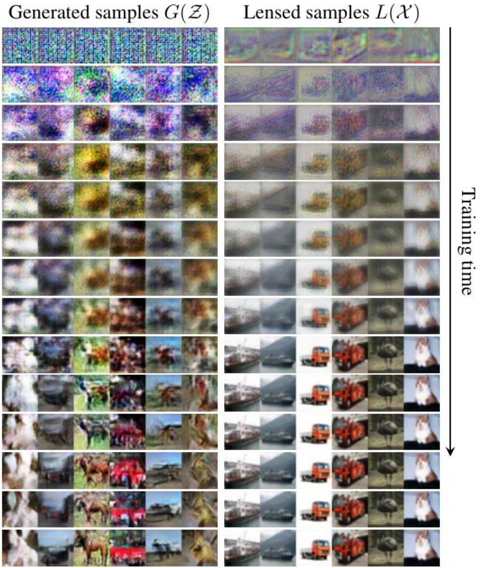
with $D$ trained from $\{(L(x), 1)\} \cup \{G(z), 0\}$ and Lens $L$ optimized

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

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.

# References (tbc)


see: github.com/artix41/awesome-transfer-learning

- Martin Arjovsky, Soumith Chintala, Lon Bottou, Wasserstein GAN, 2017
- Martin Arjovsky, Lon Bottou, Towards Principled Methods for Training Generative Adversarial Networks, 2017
- Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle: Greedy Layer-Wise Training of Deep Networks. NIPS 2006: 153-160
- Yoshua Bengio: Learning Deep Architectures for AI. Foundations and Trends in Machine Learning 2(1): 1-127 (2009)
- Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. ICLR, 2014.
- Gregory Koch, Richard Zemel, Ruslan Salakhutdinov: Siamese Neural Networks for One-shot Image Recognition, ICML 15
- Leon A. Gatys, Alexander S. Ecker, Matthias Bethge: A Neural Algorithm of Artistic Style. NIPS 2015
- Glorot, X., Bordes, A., and Bengio, Y. Domain adaptation for large-scale sentiment classification: A deep learning approach. ICML 11.

# References (tbc)


- Ian J. Goodfellow, Yoshua Bengio, Aaron C. Courville: Deep Learning. Adaptive computation and machine learning, MIT Press 2016, ISBN 978-0-262-03561-3, pp. 1-775
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, Generative Adversarial Networks, NIPS 2014
- Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, Aaron Courville, Improved Training of Wasserstein GANs, 2017
- van der Maaten, L.J.P.; Hinton, G.E.: Visualizing Data Using t-SNE. Journal of Machine Learning Research. 9: 25792605. 2008
- Lars Mescheder, Andreas Geiger, Sebastian Nowozin, Which Training Methods for GANs do actually Converge?, ICML
- Portilla, J., Simoncelli, E. P., A Parametric Texture Model Based on Joint Statistics of Complex Wavelet Coefficients. Int. J. Comput. Vis. 40, 49-70 (2000)
- Kevin Roth, Aurelien Lucchi, Sebastian Nowozin, Thomas Hofmann, Stabilizing Training of Generative Adversarial Networks through Regularization, NIPS 2017

## References (tbc)


- Mehdi S. M. Sajjadi, Giambattista Parascandolo, Arash Mehrjou, Bernhard Schlkopf: Tempered Adversarial Networks, ICML, 2018
- Tim Salimans, Diederik Kingma, and Max Welling. Markov chain Monte-Carlo and variational inference: Bridging the gap. ICML, 2015
- Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. Extracting and composing robust features with denoising autoencoders. ICML, 2008.