Unsupervised Deep Learning

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020


FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Motivation


SIGILLUM ACADEMIAE FRIDERICO ALEXANDRINAE

Motivation

Dataset variation

	So far	Medical imaging
Dataset size	Huge (up to **millions**!)	Small (**30-100** patients)
Variation	**Many** objects	**One** complex object
Modalities	**Few** modalities	**Many** modalities

Example

- Germany: 65 CT scans per 1000 inhabitants

➔ 5 million CT scans in 2014!

- ... but it is highly **sensitive** data
- Trend to make this data available
- ... but annotation is **expensive** to obtain

Source: http://www.cancerimagingarchive.net, https://data.oecd.org/healthcare/computed-tomography-ct-exams.htm

Solutions

- **Weakly supervised** learning: label for related task

  Example: object localization with class labels

Brushing teeth

Cutting trees

- **Semi-supervised** learning: partial (typical: little) labeled data
- **Unsupervised** learning: no labeled data

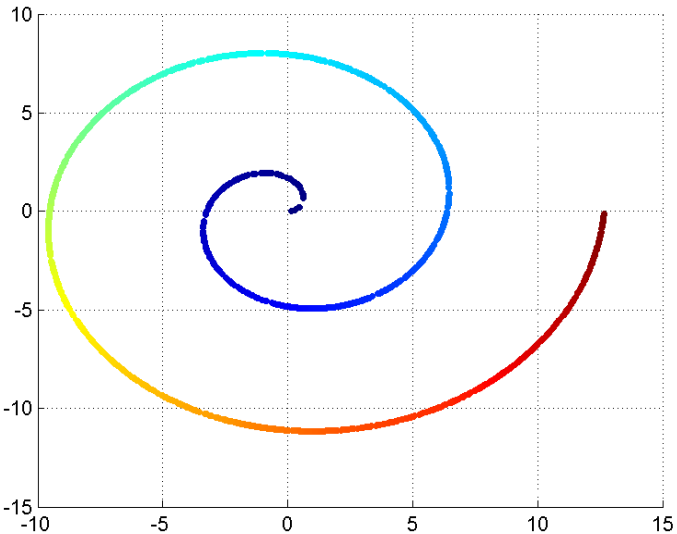
Source: [21]

Label-free Learning

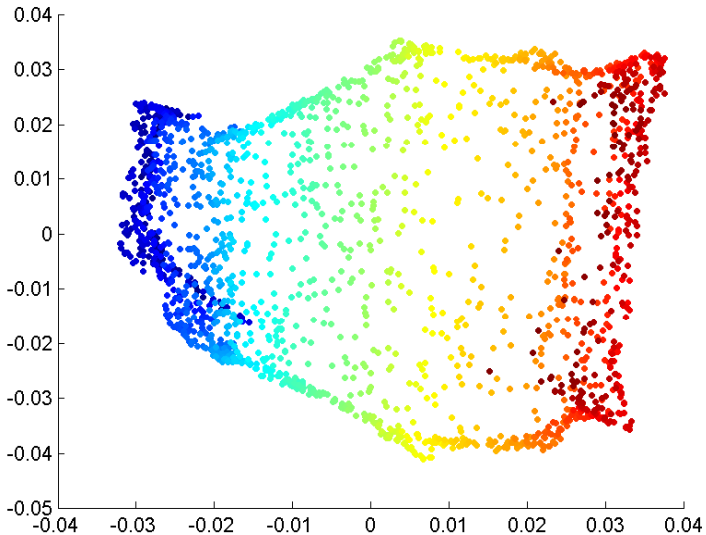
Applications as

- Dimensionality Reduction

Data on a manifold



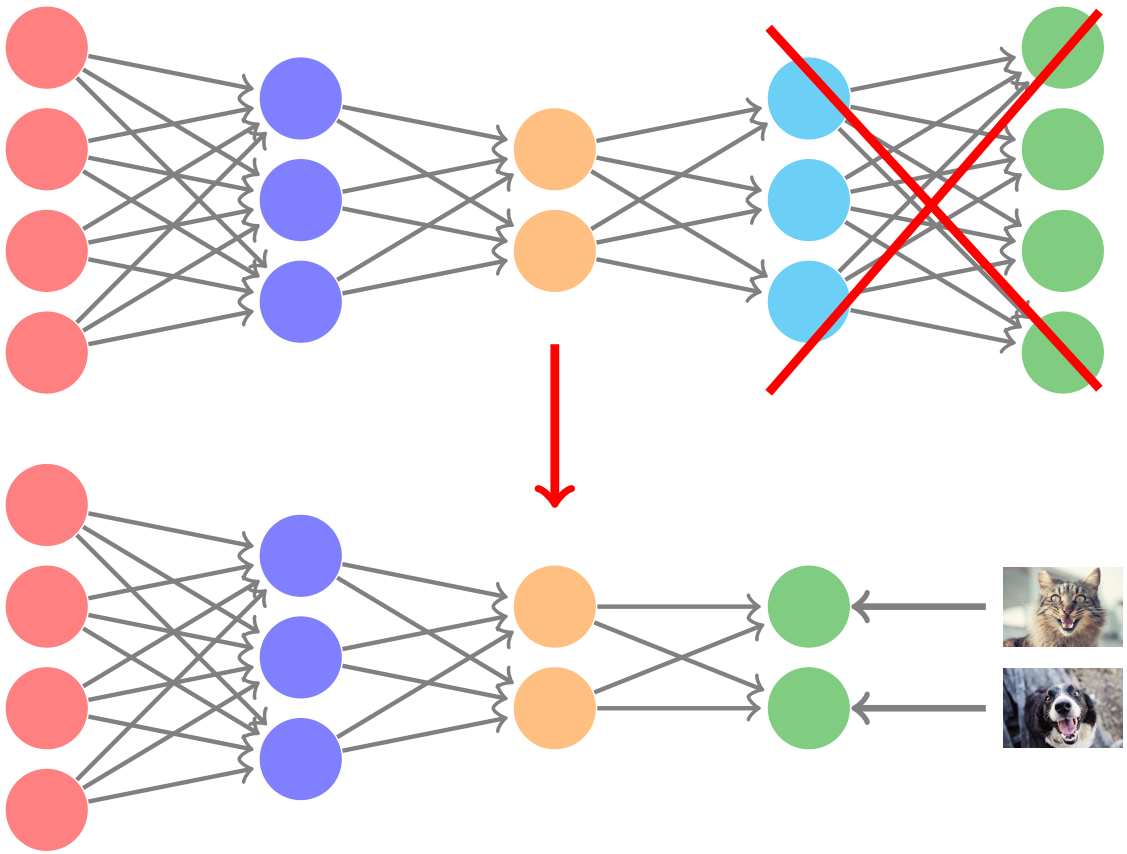
Dimensionality reduced



Label-free Learning

Applications as

- Network Initialization (related: transfer learning)

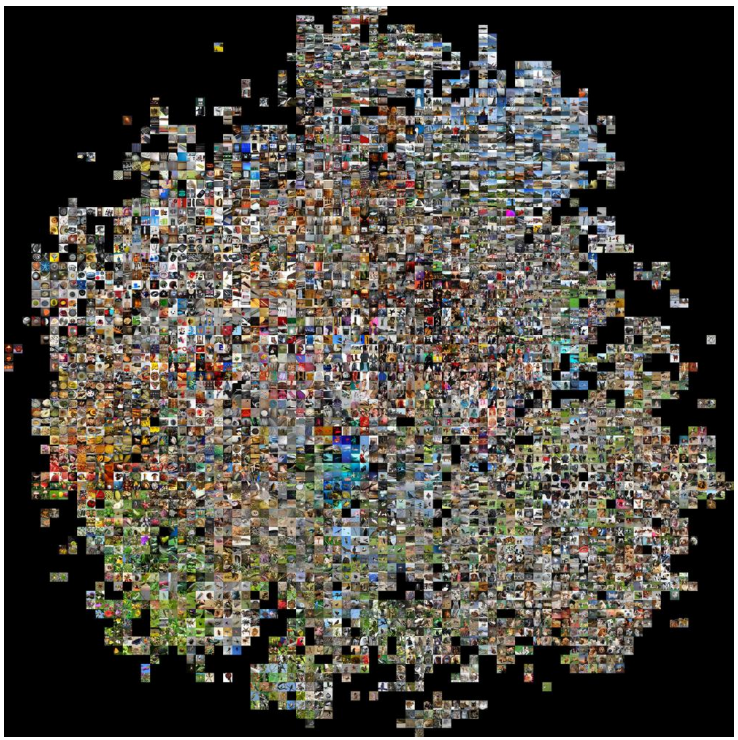


Label-free Learning

Applications as

- Representation Learning

➔ e.g. for clustering

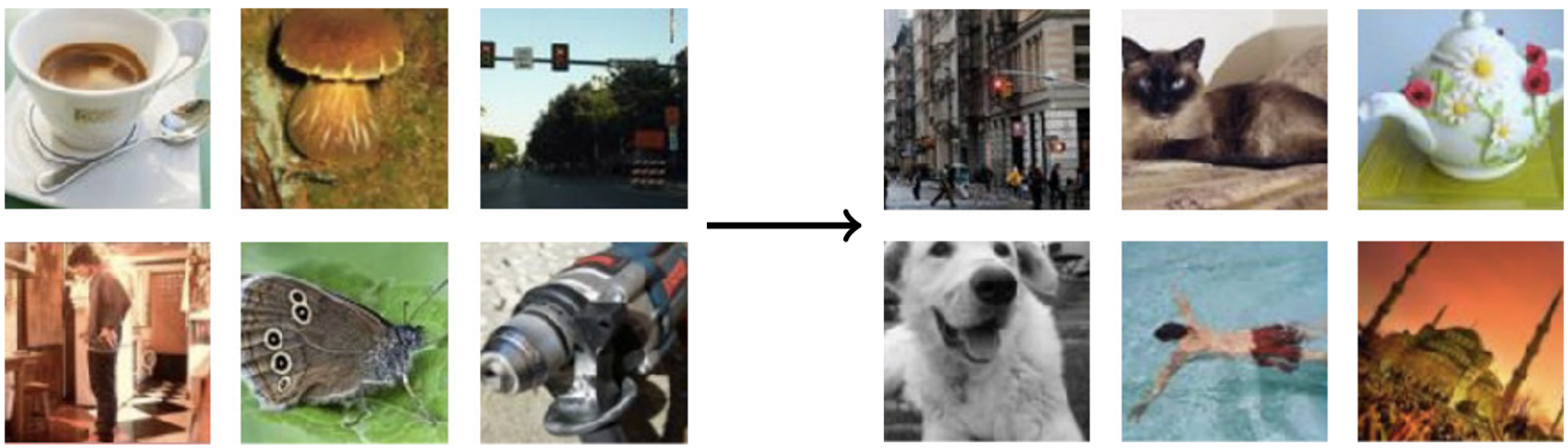


Source: http://cs.stanford.edu/people/karpathy/cnnembed/

Label-free Learning

Applications as

- Generative Models
  - Realistic generation tasks
  - Missing Data ➔ semi-supervised learning
  - Image to image translation
  - Simulate possible futures ➔ reinforcement learning



Training examples

Model samples

Source: [7]

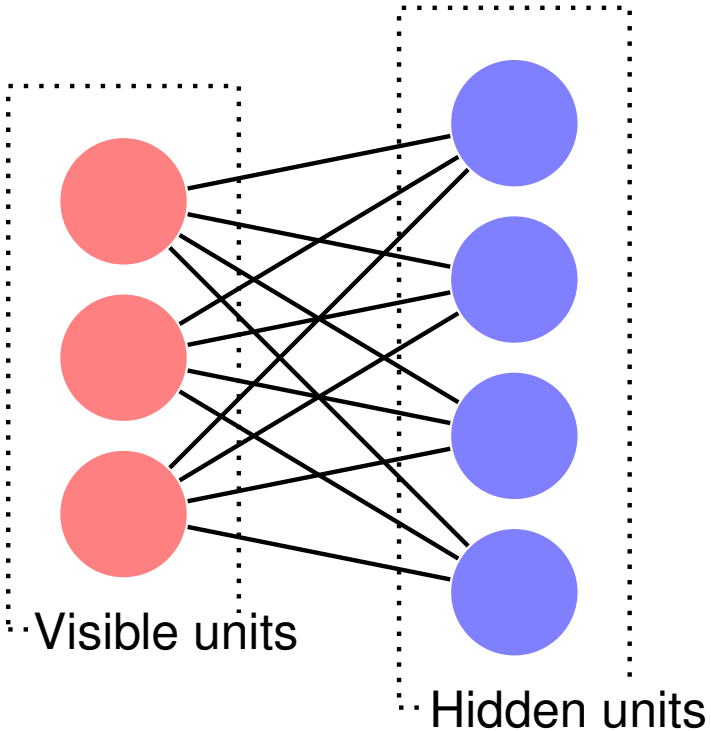
Label-free Learning

Today

- Restricted Boltzmann Machines
  - Historically important but nowadays rarely used
  - Building blocks of Deep Belief Networks
- Autoencoders
  - Non-linear dimensionality reduction
  - Extensions to generative models, e.g. variational autoencoders
- Generative Adversarial Networks
  - Currently most widely used generative model
  - Many applications of the general concept, e.g. for segmentation, reconstruction, semi-supervised learning and more

Restricted Boltzmann Machines

Restricted Boltzmann Machine (RBM)



- Binary (Bernoulli) visible units **v** represent observed data
- Binary (Bernoulli) hidden units **h** capture dependencies (latent variables), learn representation of the input

Restricted Boltzmann Machine (RBM)

- The RBM is an energy-based model with a joint probability function $p(\mathbf{v}, \mathbf{h})$ defined in terms of an energy function:

$$p(\mathbf{v}, \mathbf{h}) = \frac{1}{Z} \exp(-E(\mathbf{v}, \mathbf{h}))$$

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{b}^\top \mathbf{v} - \mathbf{c}^\top \mathbf{h} - \mathbf{v}^\top \mathbf{W} \mathbf{h}$$

  $\mathbf{b}, \mathbf{c}$: biases; $\mathbf{W}$: connection matrix

  Z: "partition function" (normalization constant), sum over all possible hidden/visible pairs: $Z = \sum_{\mathbf{v},\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$

- $p(\mathbf{v}, \mathbf{h})$ is the so called Boltzmann distribution - closely related to the softmax function!
- Note: RBM is not a FC layer! **Not feed-forward**

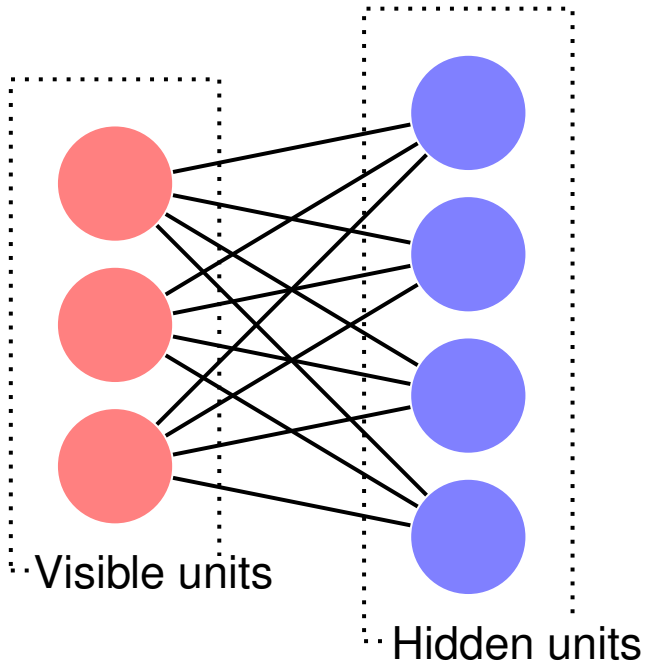
➔ RBM's "hidden layer" models the input layer in a **stochastic** manner and is **trained unsupervised**

Training RBMs

- Visible and hidden units: bipartite graph

➔ RBMs are Markov Random Fields (MRFs) with hidden variables

- We want to find $\mathbf{W}$ such that $p(\mathbf{v}, \mathbf{h})$ is high for low energy states and vice versa.
- Learning based on gradient descent on the negative log-likelihood



Training RBMs

$\boldsymbol{\theta}$: model parameters

$$\begin{aligned}
\log L(\boldsymbol{\theta}|\boldsymbol{v}) &= p(\boldsymbol{v}|\boldsymbol{\theta}) \\
&= \log \frac{1}{Z} \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) \\
&= \log \sum_{\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h})) - \log \left(\sum_{\boldsymbol{v},\boldsymbol{h}} \exp(-E(\boldsymbol{v}, \boldsymbol{h}) \right)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial \log L(\boldsymbol{\theta}|\boldsymbol{v})}{\partial \boldsymbol{\theta}} &= \ldots \quad \text{see [5]} \\
&= \underbrace{\sum_{\boldsymbol{h}} p(\boldsymbol{h}, \boldsymbol{v}) \frac{-\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}}_{\mathbb{E}_{\text{data}}} - \underbrace{\sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{-\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}}}_{\mathbb{E}_{\text{model}}}
\end{aligned}$$

$\mathbb{E}_{\text{model}}$ generally intractable

➔ Approximate via contrastive divergence

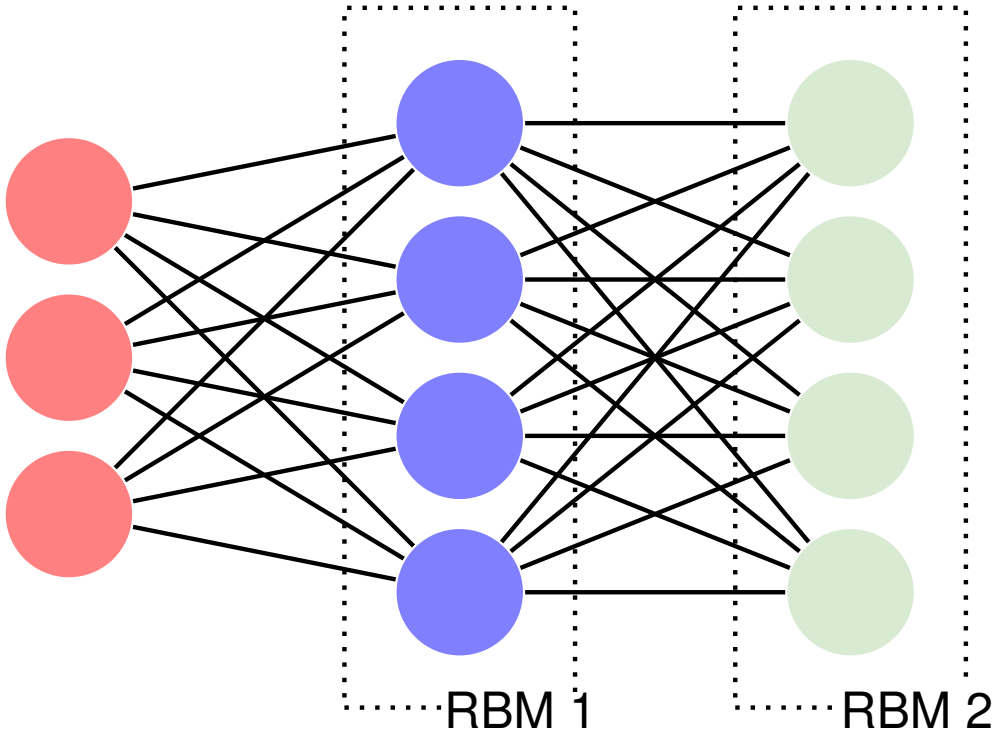
Training RBMs

Contrastive divergence

1. Take any training example as $\boldsymbol{v}$
2. Set binary states of the hidden units: $p(h_i = 1|\boldsymbol{v}) = \sigma(\sum_j w_{ij} v_j + c_i)$
3. Run k Gibbs sampling steps:
   - 3.1 Sample reconstruction $\tilde{\boldsymbol{v}}$ by $p(v_j = 1|\boldsymbol{h}) = \sigma(\sum_i w_{ij} h_i + b_j)$
   - 3.2 Resample $\tilde{h}$
4. $\Delta w_{ij} = \eta(\boldsymbol{v}\boldsymbol{h}^\top - \tilde{\boldsymbol{v}}\tilde{\boldsymbol{h}}^\top)$
   $\Delta b = \eta(\boldsymbol{v} - \tilde{\boldsymbol{v}})$
   $\Delta c = \eta(\boldsymbol{h} - \tilde{\boldsymbol{h}})$

The more iterations of Gibbs sampling, the less biased the estimate of the gradient. In practice: $k = 1$

Deep Belief Network (DBN)



- Stack multiple RBMs and train them layer by layer
- Last layer can also be fine-tuned for a classification task
- One of the first successful deep architectures [9]

➔ Sparked the “deep learning renaissance” (nowadays rarely used anymore)

NEXT TIME

ON DEEP LEARNING

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 2

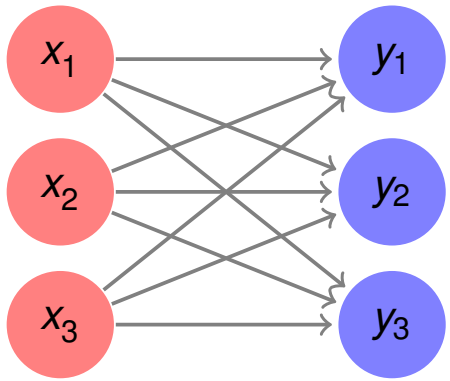
A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

June 21, 2020

Autoencoder

Autoencoder (AE)



Concept

- Special case of feed-forward neural network
- **Encoder**: $\mathbf{y} = f(\mathbf{x})$
- ... but how do we get a **loss**?
- **Decoder**: $\hat{\mathbf{x}} = g(\mathbf{y})$
- So we train for $\hat{\mathbf{x}} = \mathbf{x}$

➔ AE tries to learn an approximation of the identity

Autoencoder – Loss Functions

$$\rightarrow L(\mathbf{x}, \mathbf{x}') \propto -\log p(\mathbf{x}|\mathbf{x}')$$

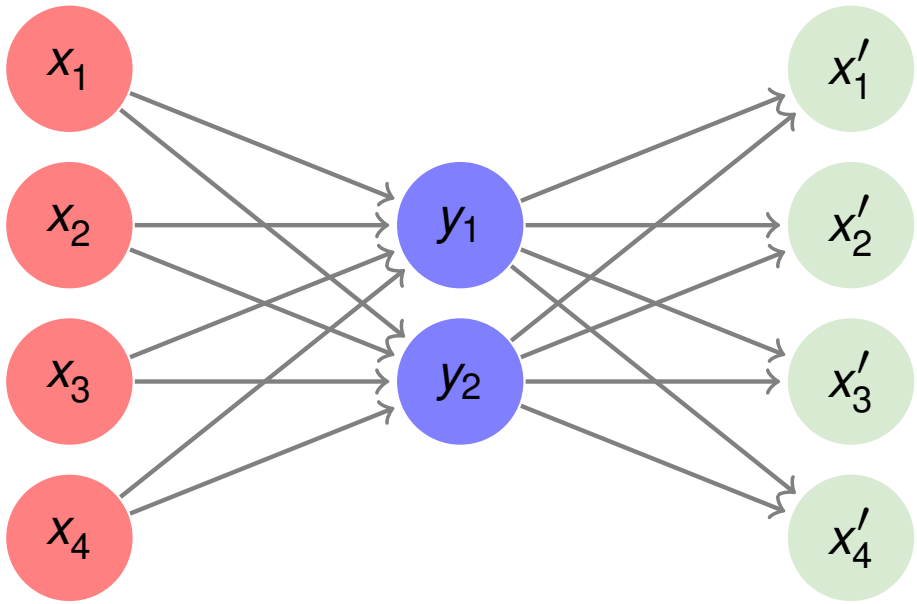
Squared L_2 norm

$$p(X|\mathbf{x}') \sim \mathcal{N}(\mathbf{x}', \sigma^2 \mathbf{I})$$
$$L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$$

Cross entropy

$$p(X|\mathbf{x}') \sim \mathcal{B}(\mathbf{x}')$$
$$L(\mathbf{x}, \mathbf{x}') = -\sum_{i=1}^{n} x_i \log(x_i') + (1 - x_i) \log(1 - x_i')$$

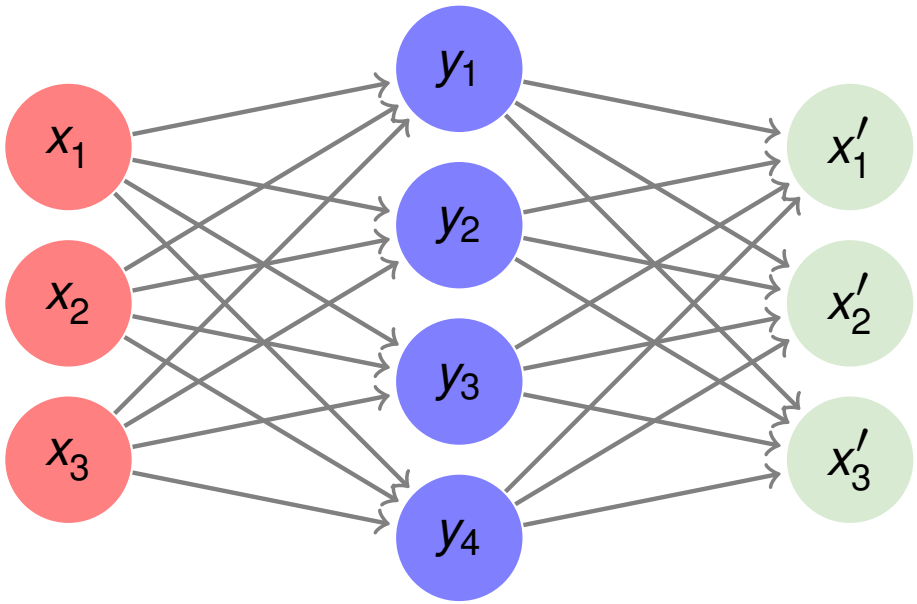
Enforce Information Compression



Undercomplete AE

- Prevent network from simply copying the input
- AE with linear layers and squared L_2 norm as loss learns a PCA
- AE with nonlinear layers: nonlinear PCA generalization

Sparse Autoencoder



- May include **more** hidden units compared to input units (rather then fewer)
- Bottleneck by enforcing sparsity in the **activations** (as opposed to weights!):

$$L_{\text{SAE}}(\mathbf{x}, \mathbf{x}', \mathbf{w}) = L(\mathbf{x}, \mathbf{x}', \mathbf{w}) + \Omega(\mathbf{y})$$

$\mathbf{y}$: activations of the bottleneck layer

Ω: sparsity penalty on $\mathbf{y}$, e.g., L_1/L_2 norm or defined via KL-divergence [18]

Autoencoder Variations / Important Terms

Convolutional Autoencoder

Replace fully connected layer with convolutional layer (optionally add pooling layers)

Denoising Autoencoder (DAE)

- Corrupt the input
- Noise models $C(\hat{\boldsymbol{x}}|\boldsymbol{x})$:
  - $C(\hat{\boldsymbol{x}}|\boldsymbol{x}) \sim \mathcal{N}(\boldsymbol{x}, \sigma^2\mathbf{I})$
  - Set random elements to zero

➔ Denoising

➔ Regularization (similar to dropout but applied to input layers)

Autoencoder Variations / Important Terms

DAE as Generative Model

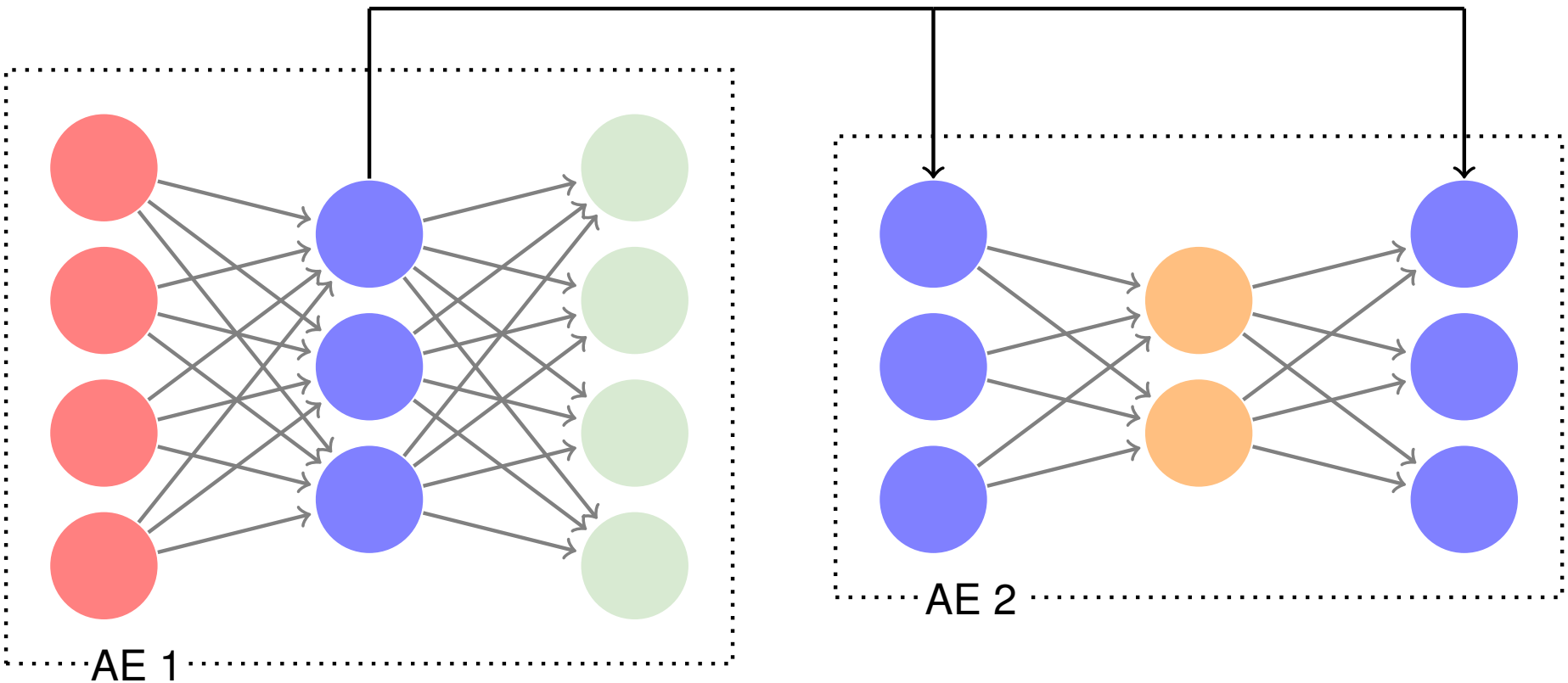
- DAEs implicitly estimate the underlying data-generating process
- Given $C(\hat{\boldsymbol{x}}|\boldsymbol{x})$ the DAE learns $p(\boldsymbol{x}|\hat{\boldsymbol{x}})$
- Intuition: If $\boldsymbol{x}$ is typical sample, then iteratively applying the noise and the denoising will reproduce the sample very often
- Estimator $p(\boldsymbol{x})$: Markov chain sampling by alternating denoising model $p(\boldsymbol{x}|\hat{\boldsymbol{x}})$ and corruption process $C(\hat{\boldsymbol{x}}, \boldsymbol{x})$
- Converging MC yields estimator $p(\boldsymbol{x})$

Often expensive and hard to assess convergence

➔ Variational AEs much more common!

Autoencoder Variations / Important Terms

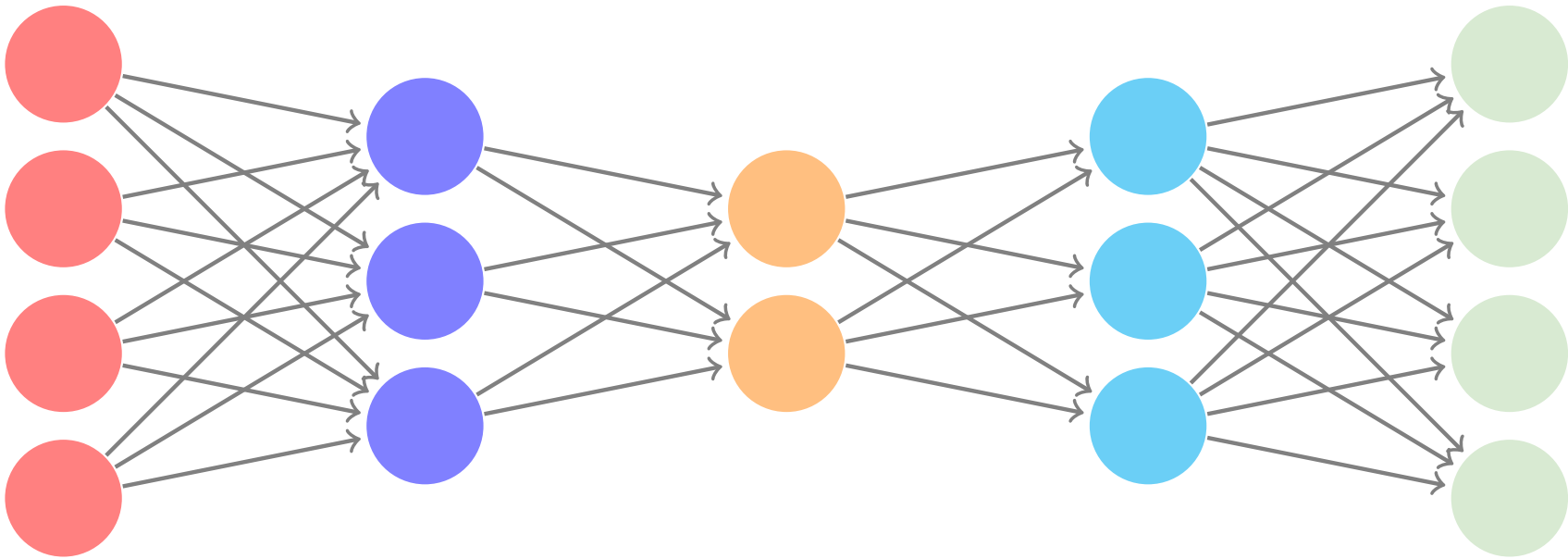
Stacked Autoencoder



Option 1: Train stacked AE layer by layer (originally proposed)

Autoencoder Variations / Important Terms

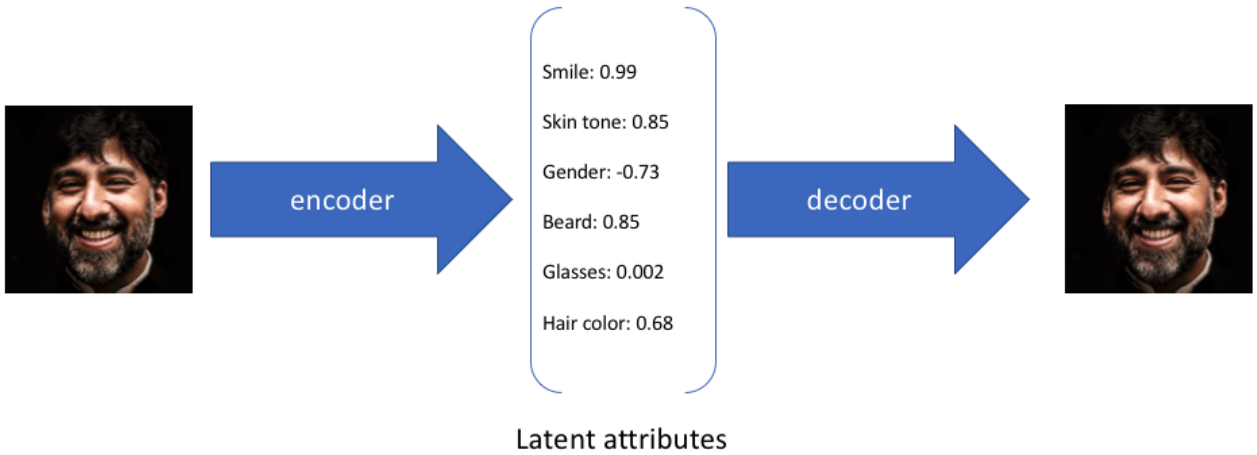
Stacked Autoencoder



Option 2: Train stacked AE by means of multiple layers

Variational Autoencoders

- "Traditional" AEs compute a **deterministic** feature vector describing the attributes of the input in latent space

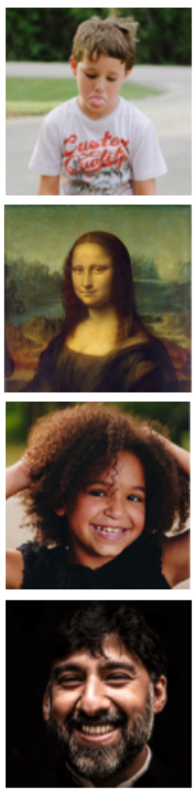


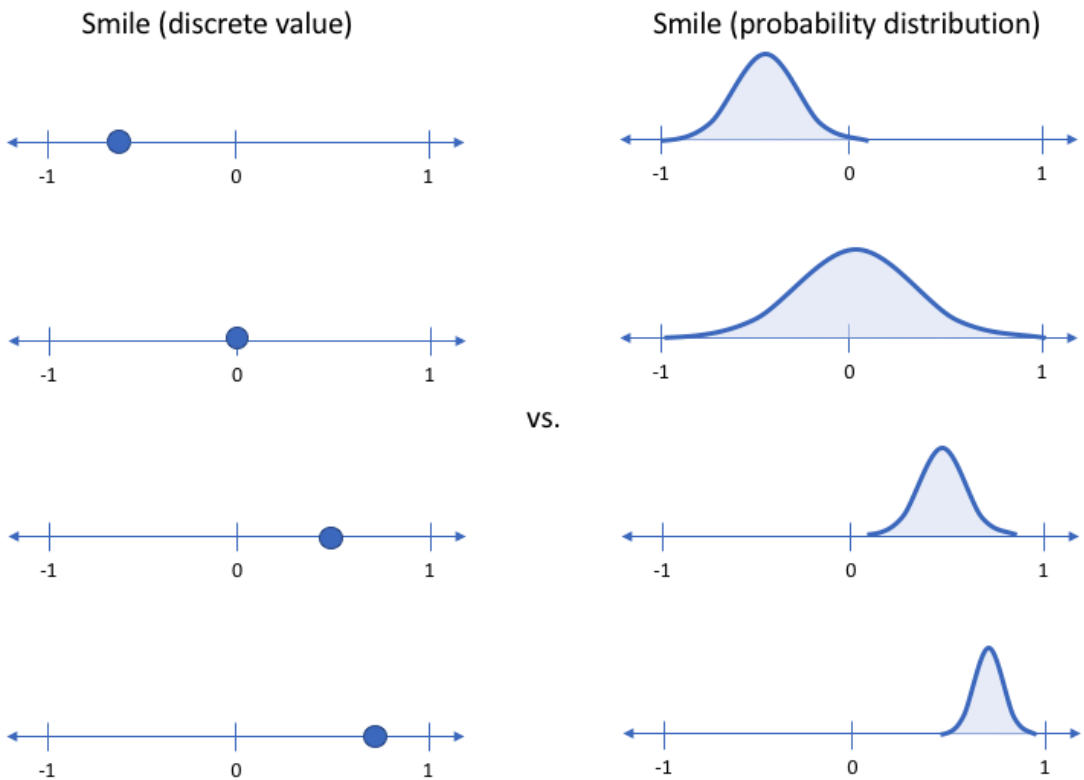
- Key difference in **variational autoencoders**:
  - Uses a variational approach to learn the latent representation
  - Allows to describe observation in latent space in **probabilistic manner**

Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders: Motivation

- Describe each latent attribute as **probability distribution**
- Allows to model uncertainty in the input data

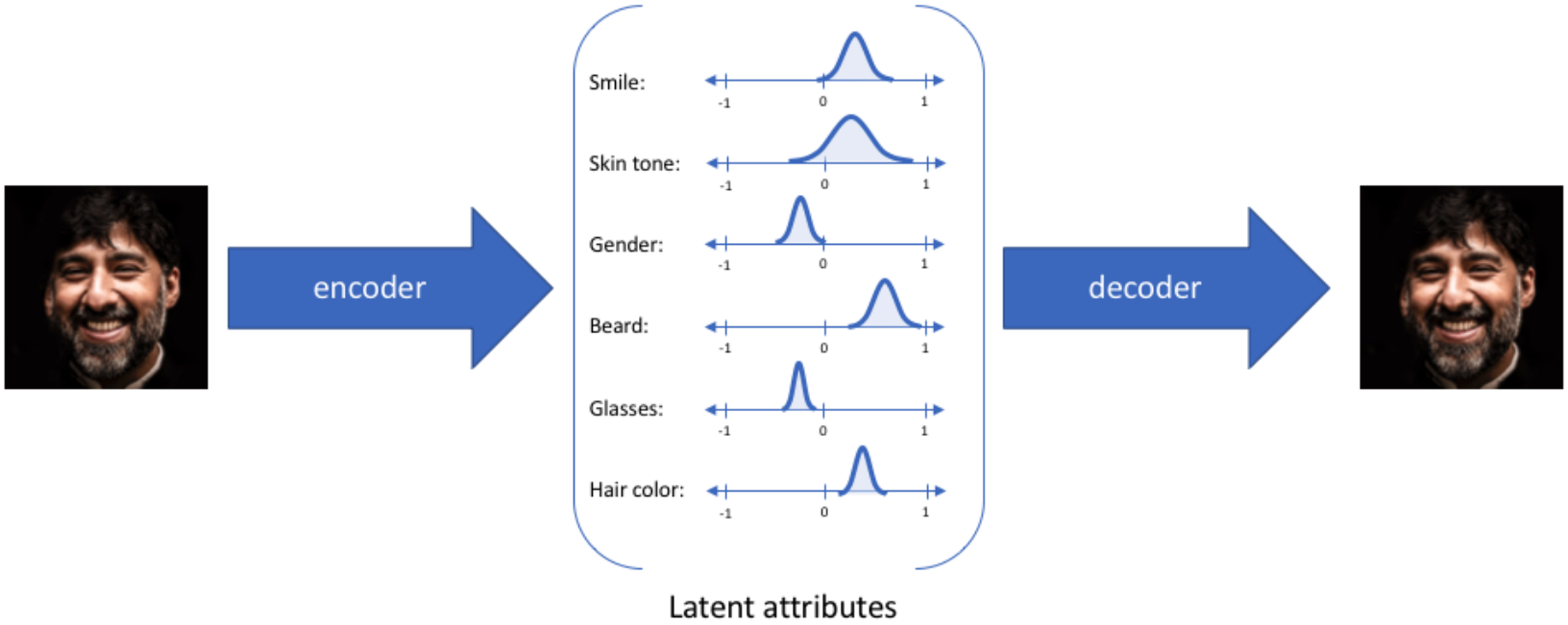




Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders

- Decoding: **Sample** from latent space as input for decoder model



- Representation as probability distribution enforces a continuous, smooth latent space representation

➔ Similar latent space vectors should correspond to similar reconstructions

Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders: Statistical Motivation

- Assumption: Hidden (latent) variable z that generates an observation x
- Training a variational autoencoder: determining the distribution of z

➔ Computing arbitrary $p(z|x)$ - usually intractable

- Approximate $p(z|x)$ by tractable distribution $q(z|x)$ ➔ Determine parameters of $q(z|x)$
- Minimize Kullback-Leibler-divergence between $p(z|x)$ and $q(z|x)$:

$$\min \ \mathrm{KL}\big(p(z|x), q(z|x)\big) \tag{1}$$

which is equivalent to

$$\max \ \underbrace{\mathbb{E}_{q(z|x)} \log p(x|z)}_{\text{reconstruction likelihood}} - \mathrm{KL}\big(q(z|x), p(z)\big) \tag{2}$$

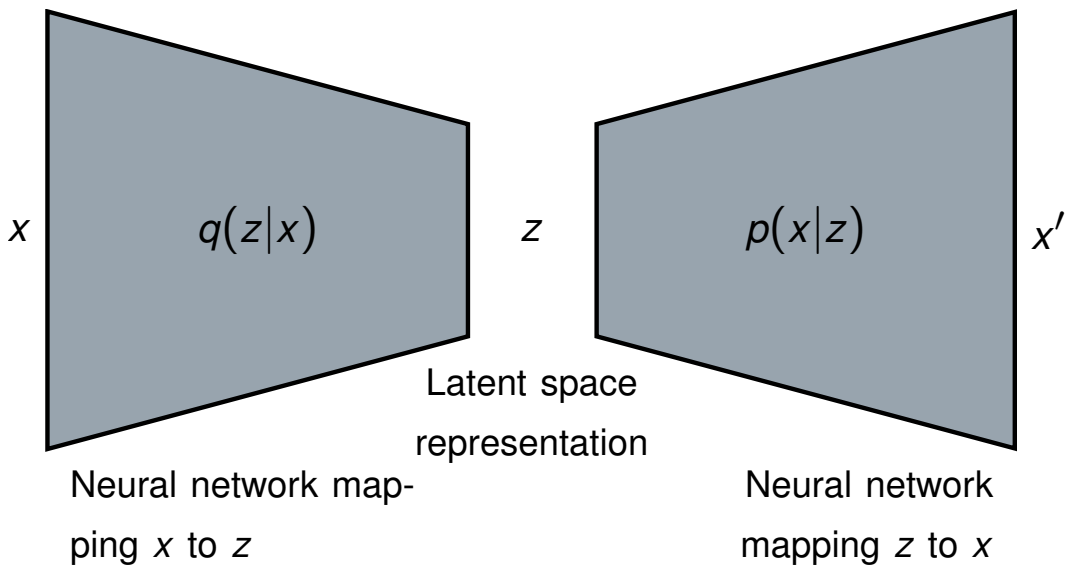
which forces $q(z|x)$ to be similar to true prior distribution $p(z)$

Variational Autoencoders: Statistical Motivation

- $p(z)$ often assumed to be (isotropic) Gaussian distribution

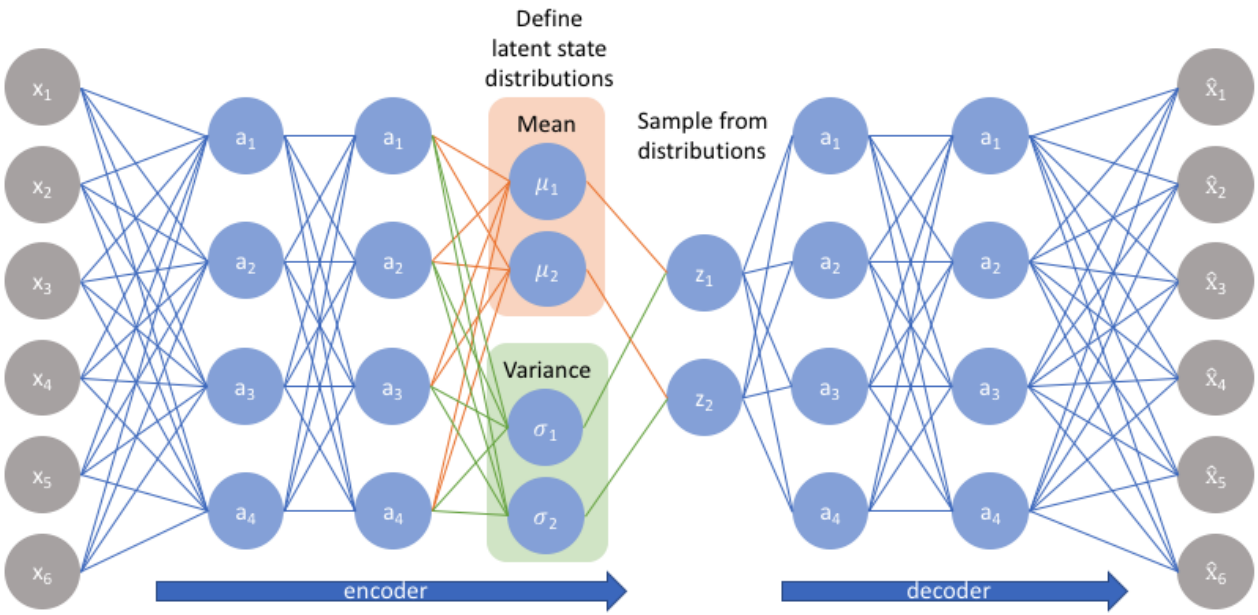
➔ Determining $q(z|x)$ boils down to estimating μ and σ

- Use **neural network** to estimate $q(z|x)$ and $p(x|z)$



Source: After https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders: Training

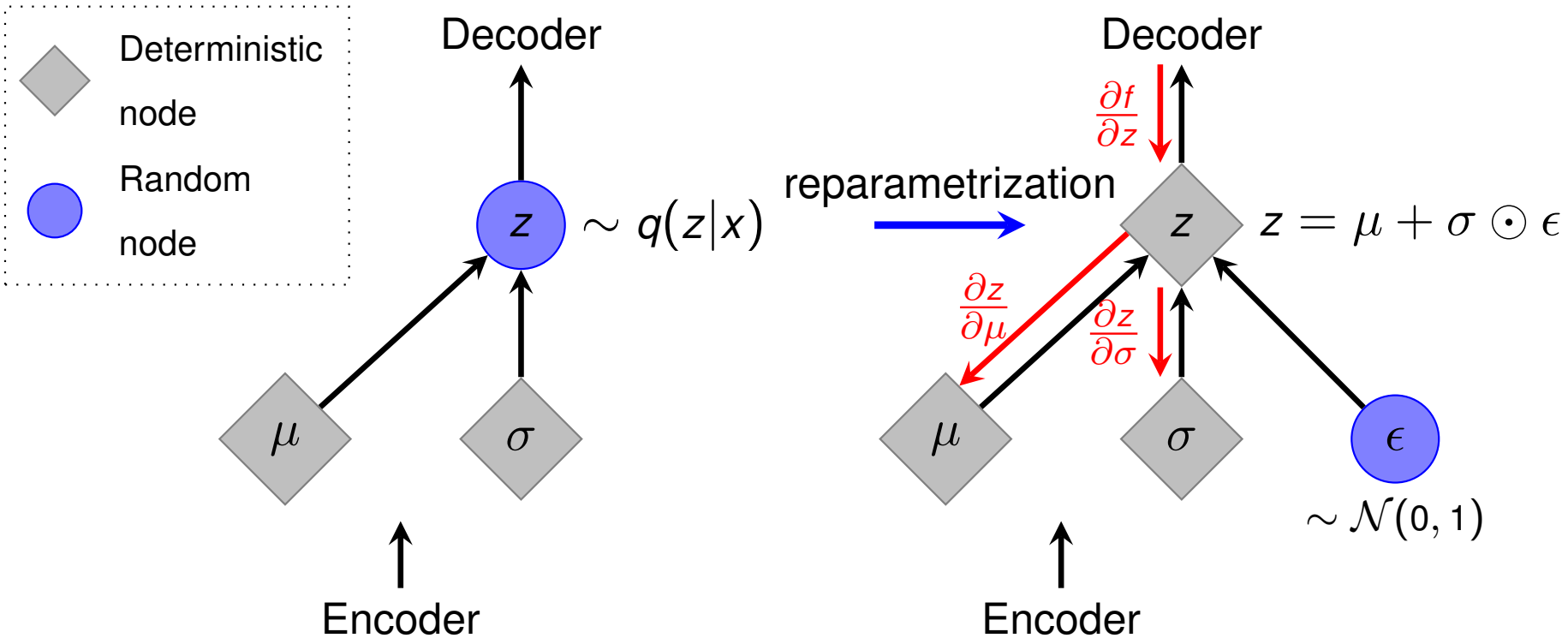


- Loss function: $L(\theta, \phi; x, z) = \mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \mathrm{KL}\left(q_\phi(z|x), p(z)\right)$
- Problem: Network contains **sampling** operator ➔ we cannot backpropagate through!

Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders: Reparametrization Trick

- We cannot backpropagate through random sampling - what now?
- "Push" random sampling out of backpropagation path by reparametrization:

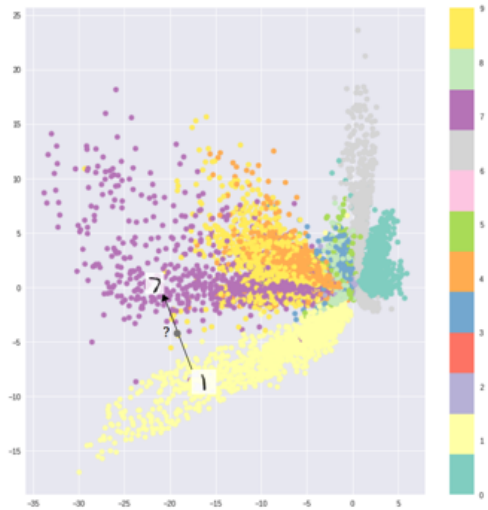


➔ Deterministic backpropagation

Source: after: https://www.jeremyjordan.me/variational-autoencoders/

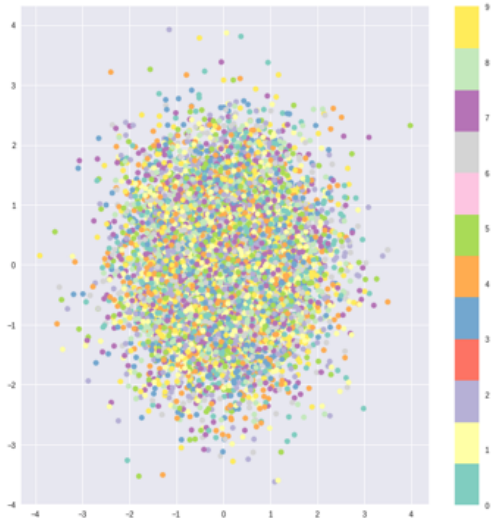
Variational Autoencoders: Latent Space Visualization

Only reconstruction loss



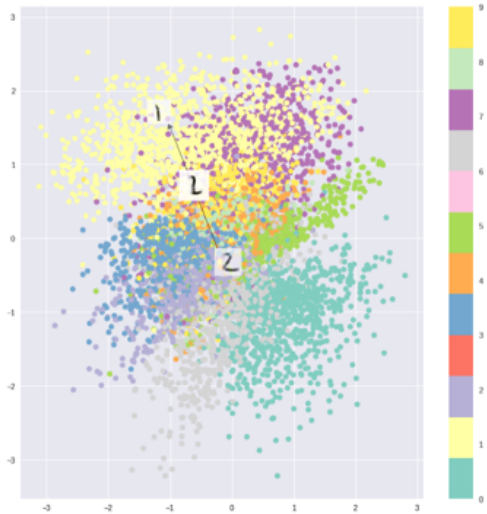
Samples well separated but no smooth transition

Only KL divergence



Not able to describe original data

Combination



Simultaneous optimization allows to describe input with distributions close to prior

Source: https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoders as Generative Models

- New data can be generated by sampling from distributions in the latent space
  ➔ reconstructed by decoder
- Diagonal prior enforces independent latent variables

➔ Can encode different factors of variation

- Example: Smoothly varying degree of smile and head pose [12]

Source: Kingma and Welling [12]

Variational Autoencoders: Summary

- Probabilistic models ➔ allow to generate data
- Intractable density ➔ optimize variational lower bound instead
- Trained via back propagation by using reparametrization
- Pros:
  - Principled approach to generative models
  - Latent space representation can be useful for other tasks
- Cons:
  - Only maximizes lower bound of likelihood
  - Samples in standard models often of lower quality compared to GANs
- Active area of research!

NEXT TIME

ON DEEP LEARNING

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

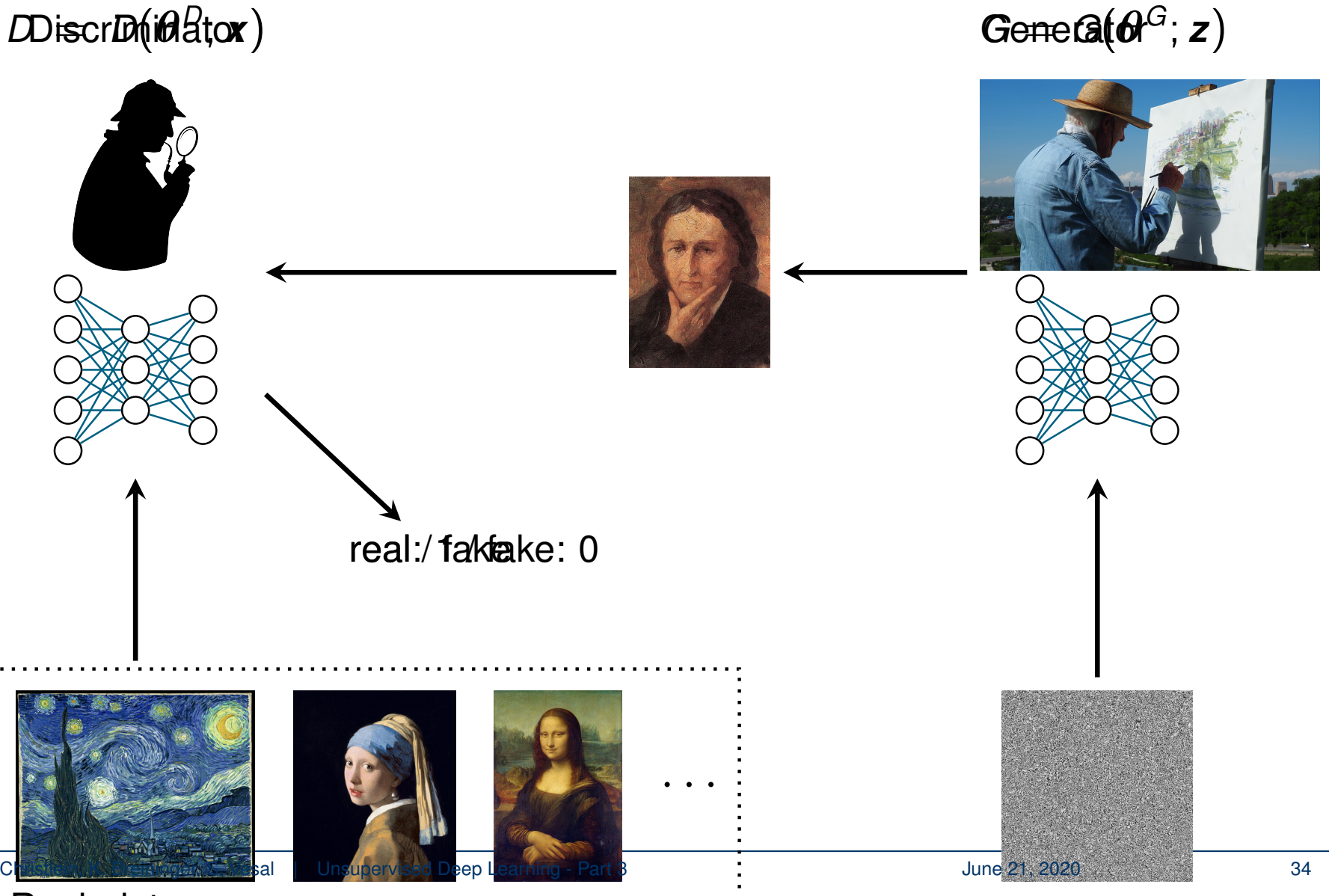
Unsupervised Deep Learning - Part 3

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

June 21, 2020

Generative Adversarial Networks

Let's Play a Game (or the Principle of GANs)



Training GANs – Minimax Game

Alternate between:

1. Train D: minimize

$$L^D(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G) = -\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\underbrace{\boldsymbol{x}}_{\text{real}}) - \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} \log(\underbrace{1 - D(G(\boldsymbol{z}))}_{\text{fake}})$$

   ➔ Trained to distinguish real data samples from fake ones

2. Train G: minimize $L^G = -L^D$

   Generator minimizes log-probability of the discriminator being correct

   ➔ Trained to generate data domain images and fool D

- Optional: run k steps of one player for every step of the other player
- Equilibrium is a saddle point of the discriminator loss

Training GANs – Minimax Game (cont.)

$$L^D = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\underbrace{\boldsymbol{x}}_{\text{real}}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} \log(\underbrace{1 - D(G(\boldsymbol{z}))}_{\text{fake}})$$

$$L^G = -L^D$$

- L^G is tied directly to $-L^D$

➔ Summarize game with a **value function** specifying the discriminator's payoff:

$$V(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G) = -L^D(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$$

Minimax game:

$$\hat{\boldsymbol{\theta}}^G = \arg\min_{\boldsymbol{\theta}^G} \max_{\boldsymbol{\theta}^D} V(\boldsymbol{\theta}^D, \boldsymbol{\theta}^G)$$

Training GANs – Optimal Discriminator

- Assumption: both densities are nonzero everywhere

➔ Otherwise some input values are never trained ➔ some $D(\boldsymbol{x})$ have undetermined behavior

- Solve for: $\frac{\partial L^D}{\partial D(\boldsymbol{x})} = 0$
- Optimal $D^*(\boldsymbol{x})$ for any $p_{\text{data}}(\boldsymbol{x})$ and $p_{\text{model}}(\boldsymbol{x})$:

$$D^*(\boldsymbol{x}) = \frac{p_{\text{data}}(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_{\text{model}}(\boldsymbol{x})}$$

- Unfortunately this optimal discriminator is theoretical and unachievable

➔ GAN's key approximation mechanism!

- GANs use supervised learning to estimate this ratio

➔ Underfitting / Overfitting

Non-Saturating Games – Modify Generator's Loss

$$L^D = -\frac{1}{2}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}} \log D(\underbrace{\boldsymbol{x}}_{\text{real}}) - \frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} \log(\underbrace{1 - D(G(\boldsymbol{z}))}_{\text{fake}})$$

$$L^G = -\frac{1}{2}\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}} \log D(G(\boldsymbol{z}))$$

- In Minimax: G minimizes log-probability of D being correct

  Here: G maximizes log-probability of D being mistaken

➔ Heuristically motivated: fights vanishing gradient of G when D is "too smart" (esp. in the beginning)

- Equilibrium no longer describable with single loss

Other Popular Loss Functions

Feature matching loss / perceptual loss

- G trained to match expected value of features $f(\boldsymbol{x})$ of intermediate layer of D:

$$L^G = \left\| \mathbb{E}_{\boldsymbol{x} \sim p_{data}} f(x) - \mathbb{E}_{\boldsymbol{z} \sim p_z} f\big(G(\boldsymbol{z})\big) \right\|_2^2$$

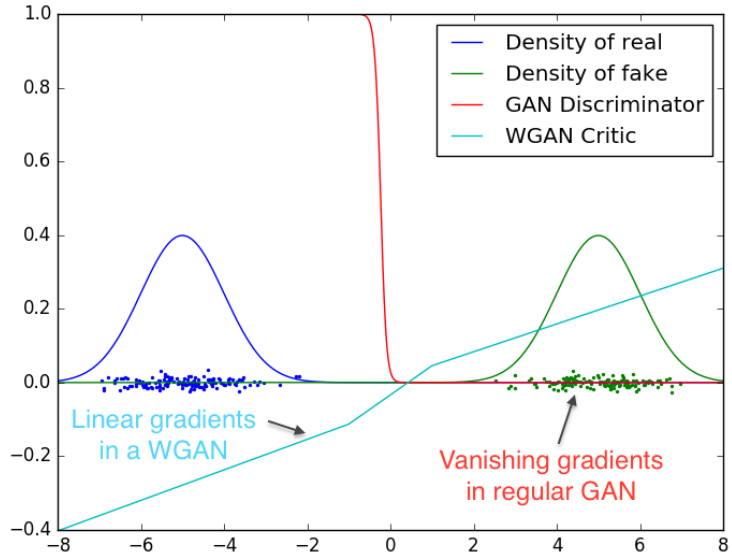
➔ Prevent “overtraining” of G on current D

- Popular loss also in other domains

Other Popular Loss Functions

Wasserstein Loss

- Derived from Wasserstein distance a.k.a. Earth Movers Distance
- Learn discriminator D that maximizes the discrepancy between real and fake and whose gradient remains beyond a limit:



$$\max_{\|D\|_L \leq 1} \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}}(D(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{x} \sim p_{\text{model}}}(D(\boldsymbol{x}))$$

- If weights could grow arbitrarily large, the gradient would be unbounded

➔ Clip D's weights ➔Critic lies within space of 1-Lipschitz functions

➔ Helps to counter vanishing gradients in D

➔ Many more loss functions exist, e. g. KL divergence (then GAN do ML)

- **But** the approximation strategy matters more than the loss

How to Evaluate GANs

Inception Score [17]

- Goals:
  1. Generated image should be recognizable (standard: Inception v-3 pre-trained on Imagenet)
  - ➔ Score distribution should be dominated by one class
  - ➔ Image-wise class distribution should have low entropy
  2. Generated images should be diverse
  - ➔ Class distribution uniform
  - ➔ Entropy should be high
- KL distance between distributions (higher: better):

$$\mathrm{IS}(\mathbf{x}) = \exp\left\{\mathbb{E}\left[\mathrm{KL}\left(p(y|\mathbf{x})\|p(y)\right)\right]\right\}$$

How to Evaluate GANs

Fréchet Inception Distance (FID) [8]

- Use intermediate layer (last pooling layer of Inception-v3 pre-trained on ImageNet)
- Model data distribution by multivariate Gaussians $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
- FID score between real images $\boldsymbol{x}$ and generated images $\boldsymbol{g}$ (lower: better)

$$\mathrm{FID}(\boldsymbol{x}, \boldsymbol{g}) = \|\mu_{\boldsymbol{x}} - \mu_{\boldsymbol{g}}\|^2 + \mathrm{Tr}\left(\boldsymbol{\Sigma}_{\boldsymbol{x}} + \boldsymbol{\Sigma}_{\boldsymbol{g}} - 2\sqrt{(\boldsymbol{\Sigma}_{\boldsymbol{x}}\boldsymbol{\Sigma}_{\boldsymbol{g}})} \right)$$

+ More robust to noise than IS
+ No class concept needed

GANs in Comparison to Other Generative Models

- Ability to generate samples in parallel
- Very few restrictions (e.g. compared to Boltzman machines)

➔ No Markov chain needed!

- No variational bound is needed

  GANs known to be asymptotically consistent since the model families are universal function approximators

NEXT TIME

ON DEEP LEARNING

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG

SCHOOL OF ENGINEERING

Unsupervised Deep Learning - Part 4

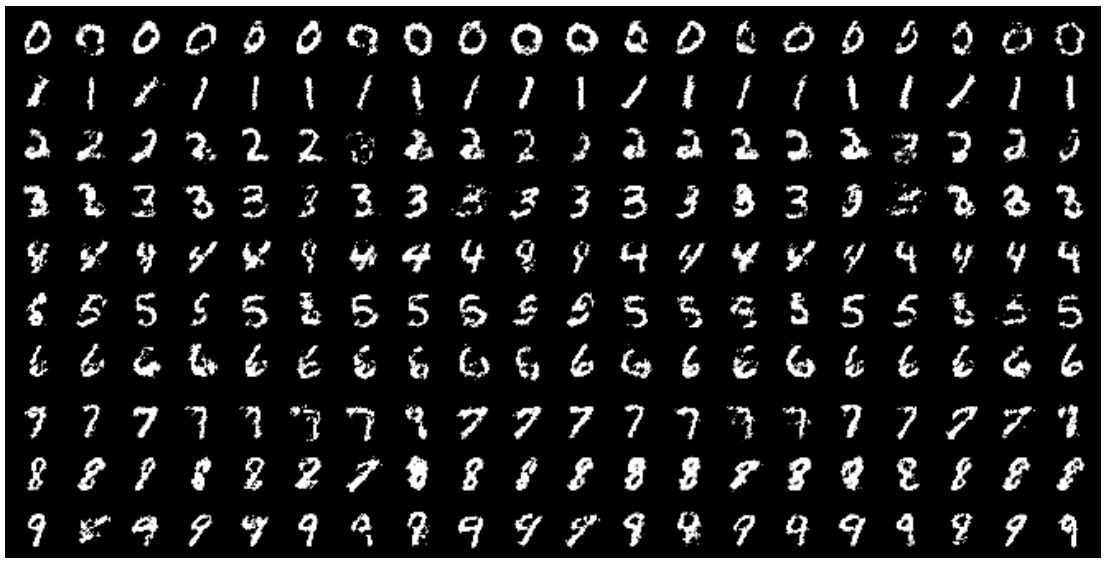
A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

June 21, 2020

Conditional GANs

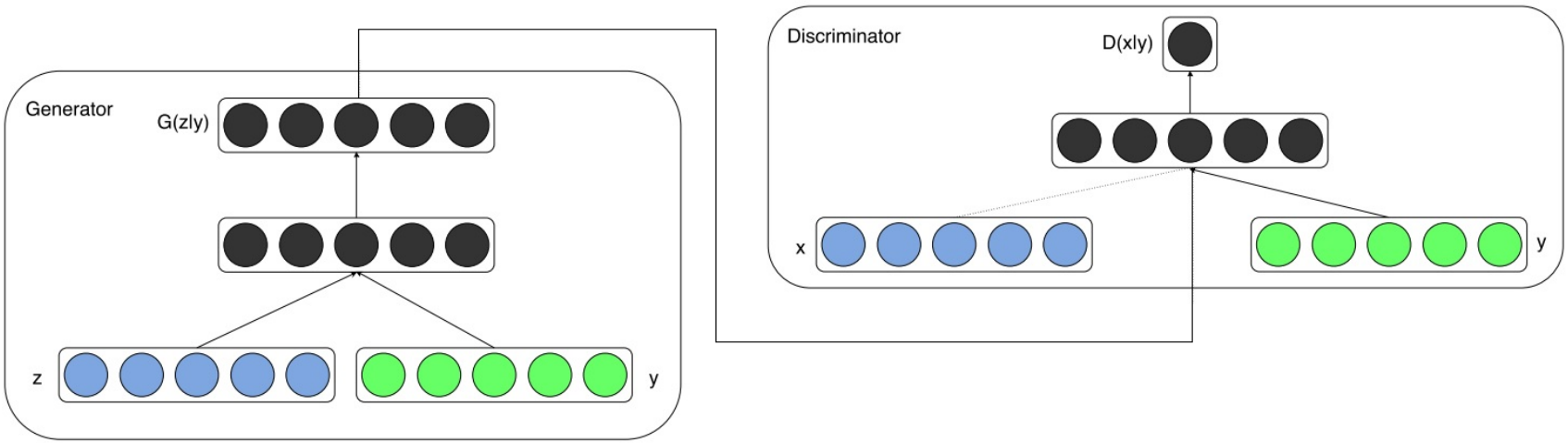
- Problem: Generator creates a “fake” generic image ➔ is not specific for a certain condition/characteristic
- Example: **text to image generation** – image should **depend** on the text
- Idea: Provide additional vector **y** to networks to encode **conditioning** [15]



Generated samples conditioned on one label (digit)

Source: Mirza et al. [15]

Conditional GANs (cont.)



Generator and discriminator in CGANs.

- Generator G receives the latent vector $\mathbf{z}$ and a conditioning vector $\mathbf{y}$
- Discriminator D receives $\mathbf{x}$ and also $\mathbf{y}$
- The objective function of a two-player minimax game changes to:

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

Source: Mirza et al. [15]

Example: Conditional GANs for Face Generation

- Add conditional feature (e.g., smiling, gender, old age, ...)
- Generator/Discriminator learn to operate in **modes**:

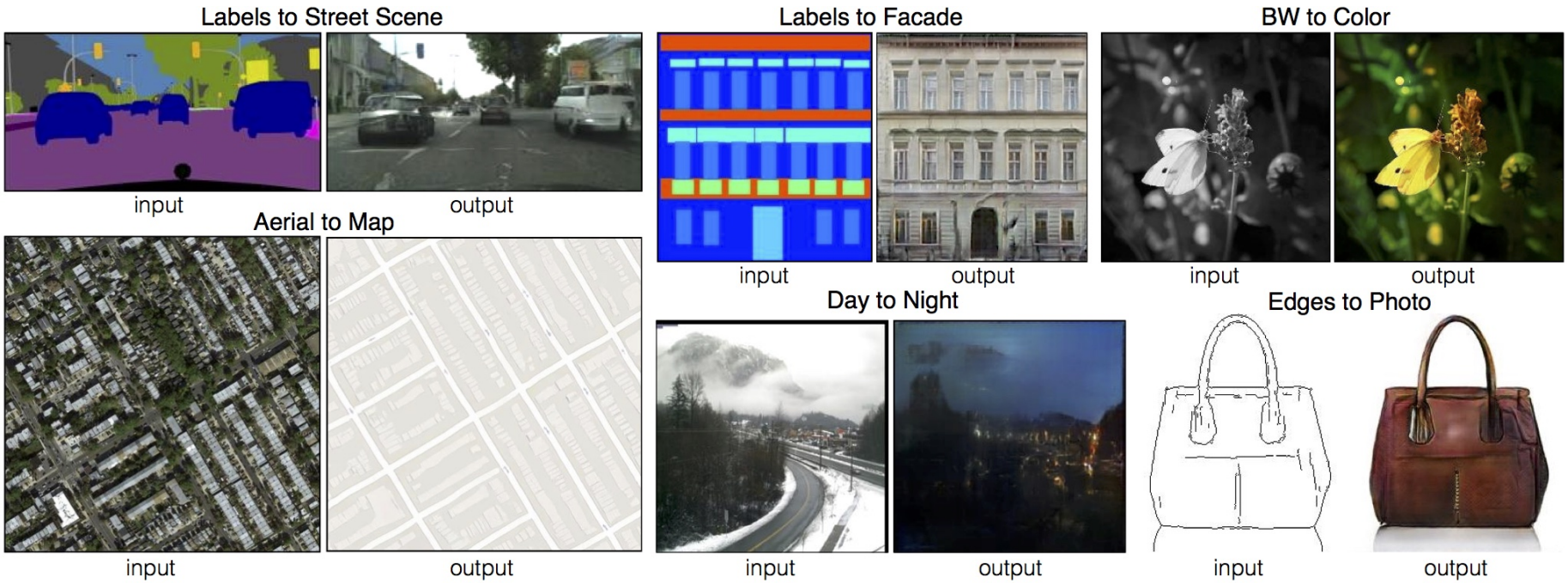
➔ Generator learns to generate a face with a certain attribute

➔ Discriminator learns to decide whether the face contains attribute

First row: random samples, second row: y $\sim$ old age, third row: y $\sim$ old age + smiling.

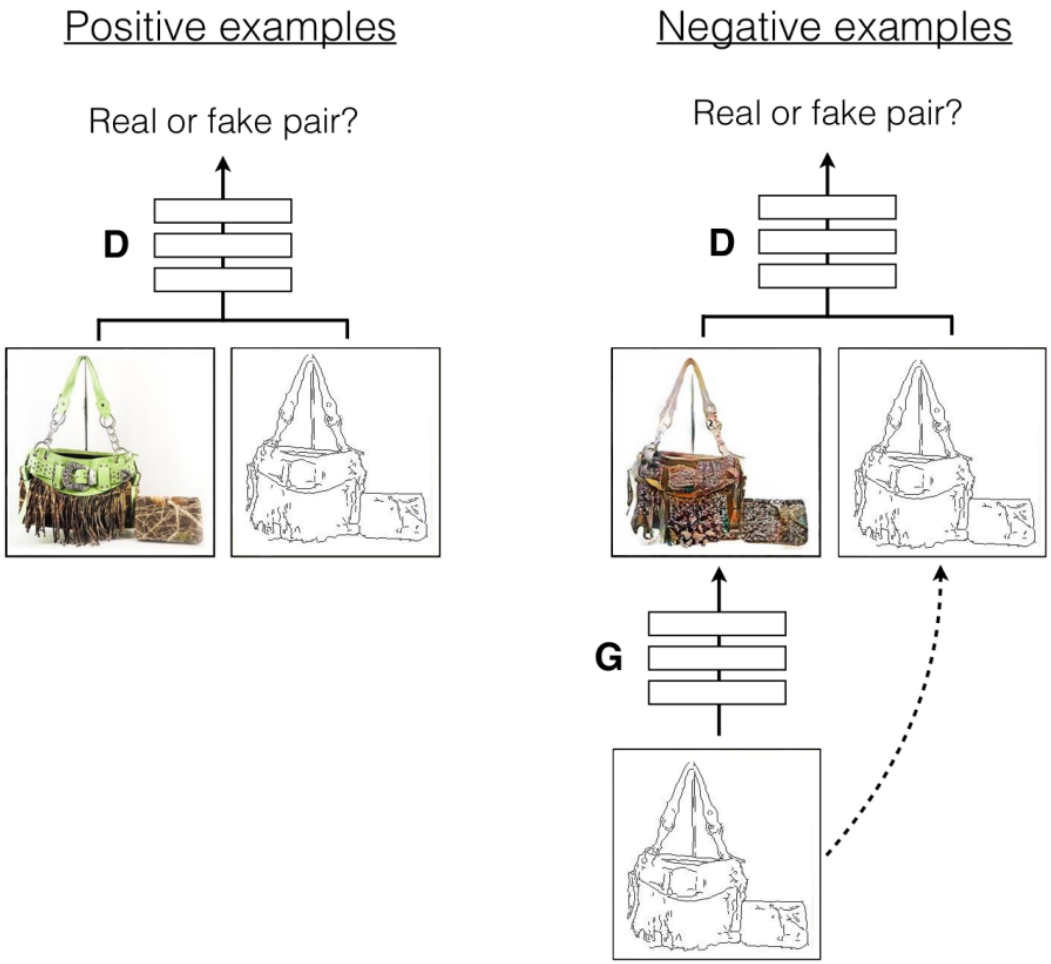
Source: Gauthier [6]

Image To Image



Source: [11]

Image To Image - Just a Conditional GAN!



Source: [11]

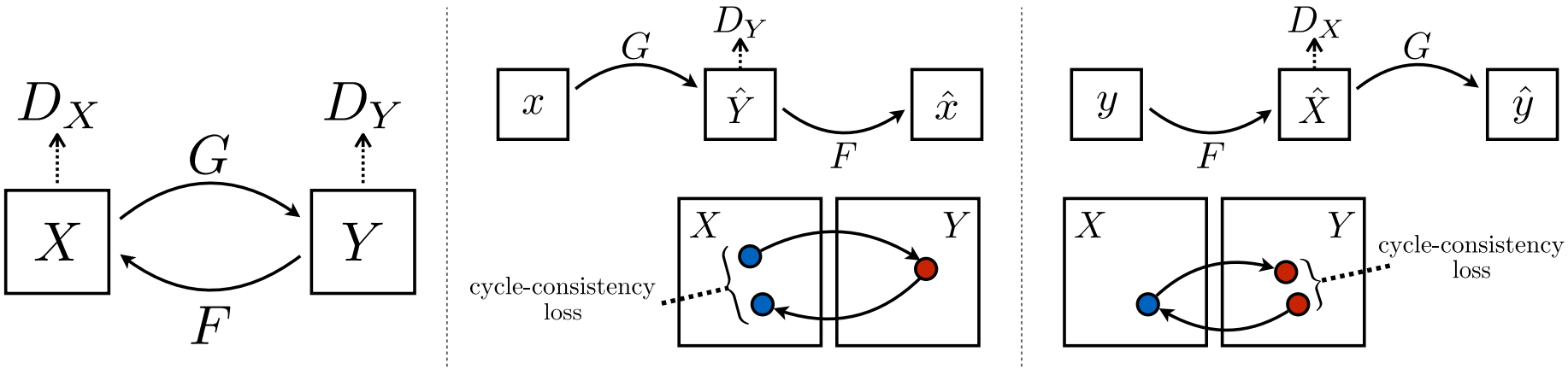
Cycle Consistent GANs [22]

- Image to Image GAN should generate plausible results **w.r.t. input**
- **Paired data** difficult/impossible to obtain
- Cycle consistency loss: Couple GAN with trainable **inverse** mapping F such that

$$F(G(\mathbf{x})) \approx \mathbf{x} \quad \text{and} \quad G(F(\mathbf{y})) \approx \mathbf{y} \tag{3}$$

Source: Adapted from [22]

Cycle Consistency Loss [22]



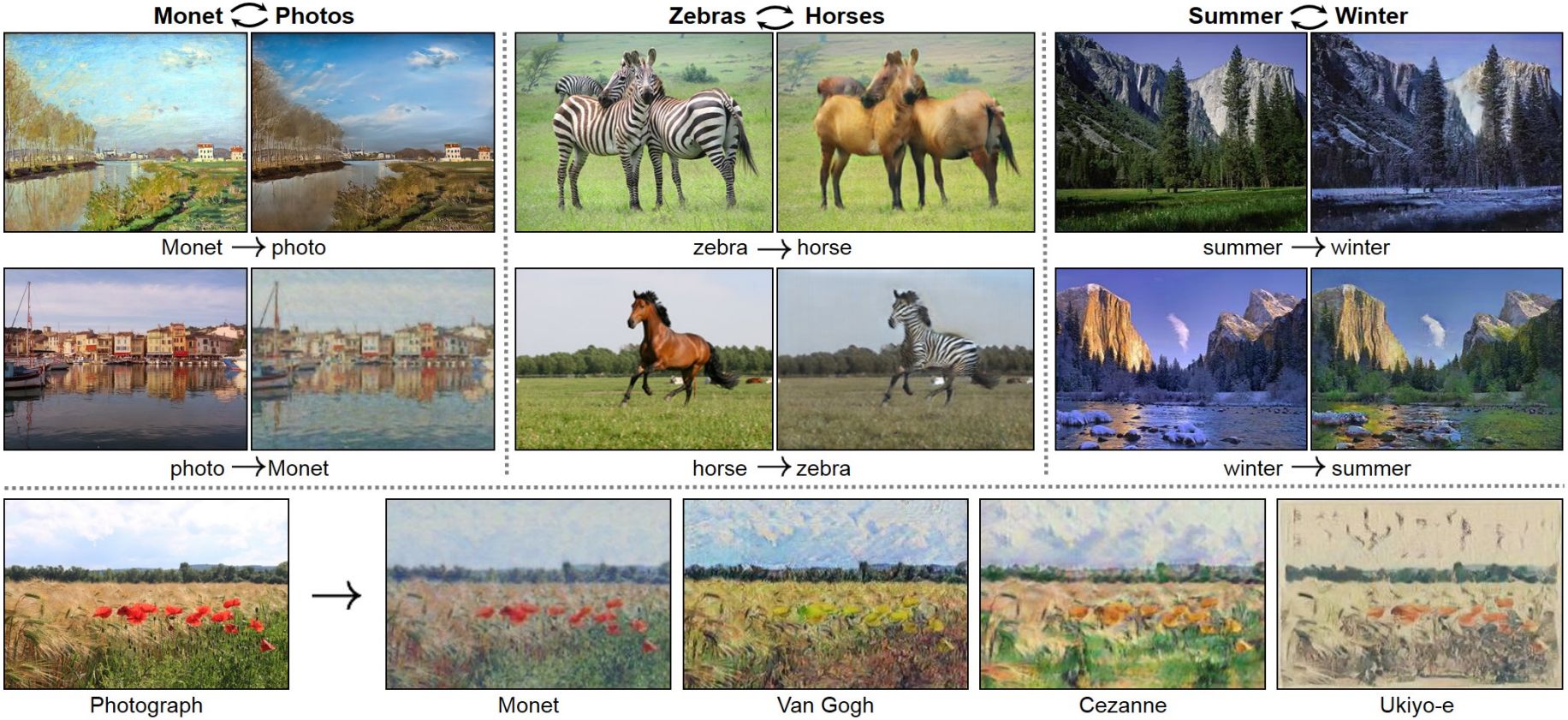
- Two discriminators D_Y and D_X
- Cycle consistency loss for **two generators** G, F:

$$L_{\text{cyc}}(G,F) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\left[\|F(G(\mathbf{x})) - \mathbf{x}\|_1\right] + \mathbb{E}_{\mathbf{y}\sim p_{\text{data}}(\mathbf{y})}\left[\|G(F(\mathbf{y})) - \mathbf{y}\|_1\right]$$

- Total loss:
  $L(G,F,D_X,D_Y) = L_{\text{GAN}}(G,D_Y,X,Y) + L_{\text{GAN}}(F,D_X,Y,X) + \lambda L_{\text{cyc}}(G,F)$

Source: Adapted from [22]

CycleGAN: Examples



Source: [22]

CycleGAN: Examples (cont.)

Ablation study for CycleGAN

Source: [22]

NEXT TIME

ON DEEP LEARNING

Unsupervised Deep Learning - Part 5

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, M. Hoffmann, C. Bergler, F. Denzinger, W. Fu, B. Geissler, Z. Yang
Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg
June 21, 2020

More Tricks of the Trade

One-sided Label Smoothing

- Replace targets of the real samples with a smoothed version

➔ replace 1 with 0.9

- Do **not** do the same for fake samples (don't change 0 label)

  Otherwise *D* will reinforce incorrect behavior ➔ *G* will produce samples that resemble the data or samples it already makes

- Benefits
  - Prevents *D* from giving very large gradient signal to *G*
  - Prevents extrapolating to encourage extreme samples

Balancing *G* and *D* necessary?

No.

- GANs work by estimating ratio of data and model density

  Ratio estimated correctly only when *D* is optimal

➔ Fine if *D* overpowers *G*

But when *D* gets too good

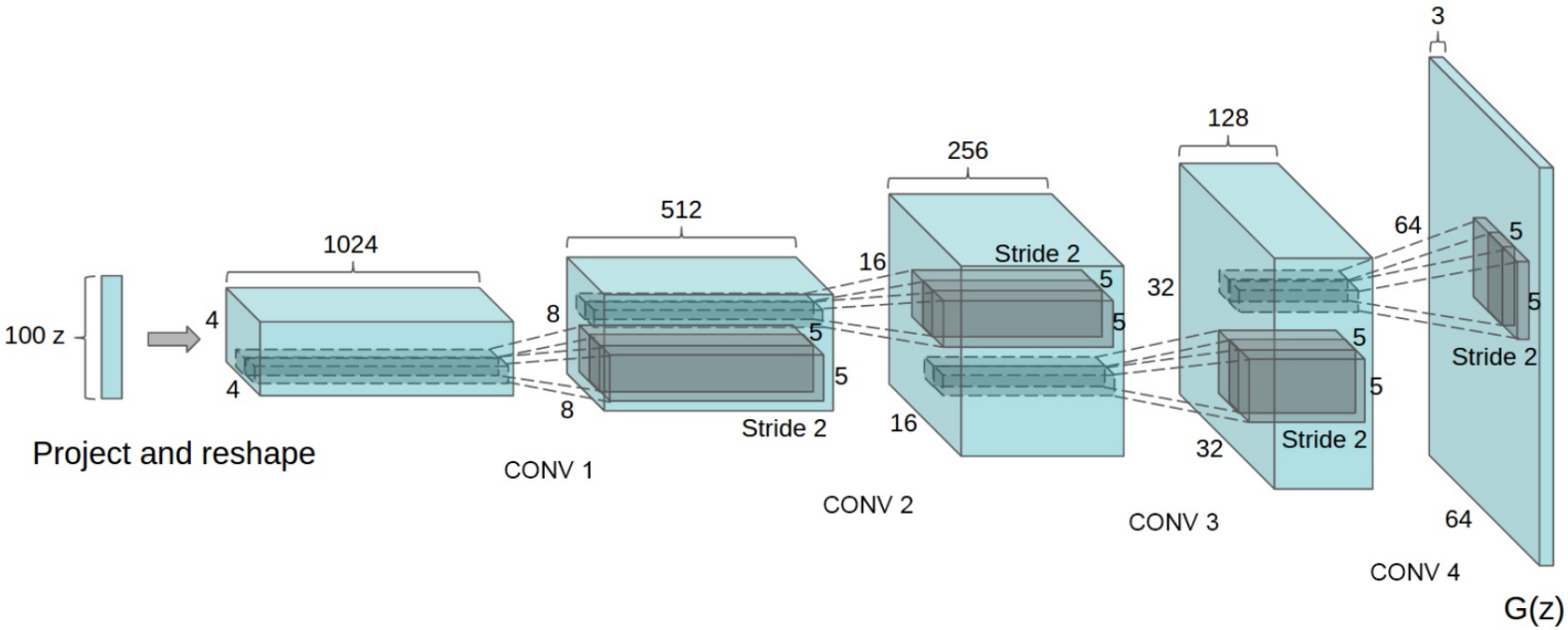
- *G*'s gradient may vanish

➔ Use non-saturating loss

- *G*'s gradient may get too large

➔ Use label smoothing

Deep Convolutional GANs (DCGAN)



- Replace any pooling layer with strided convolutions (*D*) and transposed convolution (*G*)
- Remove fully connected hidden layers for deeper architectures
- *G*: Use ReLU activation except for output layer which uses tanh
- *D*: LeakyReLU activation for all layers
- Use batch normalization

Source: [16]

Problem of Batch Normalization in *G*

➔ Causes strong intra-batch correlation

Source: [7]

Virtual Batch Normalization VBN

- Don't use one BN instance for both minibatches (containing only real / fake)!
- Use two separate BN or better use VBN
- If BN/VBN is too expensive ➔ choose instance normalization (for each sample subtract mean and divide by standard deviation)

Virtual Batch Normalization

- Create **reference batch** R of random samples chosen and fixed once at the start of training
- For each $\boldsymbol{x}_i$ of the current mini-batch
  - Create a new **virtual batch** $V_i = R \cup \boldsymbol{x}_i$
  - Compute mean and standard deviation of V_i
  - ➔ We always need to propagate R forward **in addition** to the current batch
  - Normalize $\boldsymbol{x}_i$ with these statistics

Historical Averaging

- Add penalty term that punishes weights which are rather far away from their historical average:

$$\| \boldsymbol{\theta} - \frac{1}{T} \sum_{i=1}^{T} \boldsymbol{\theta}[i] \|^2$$

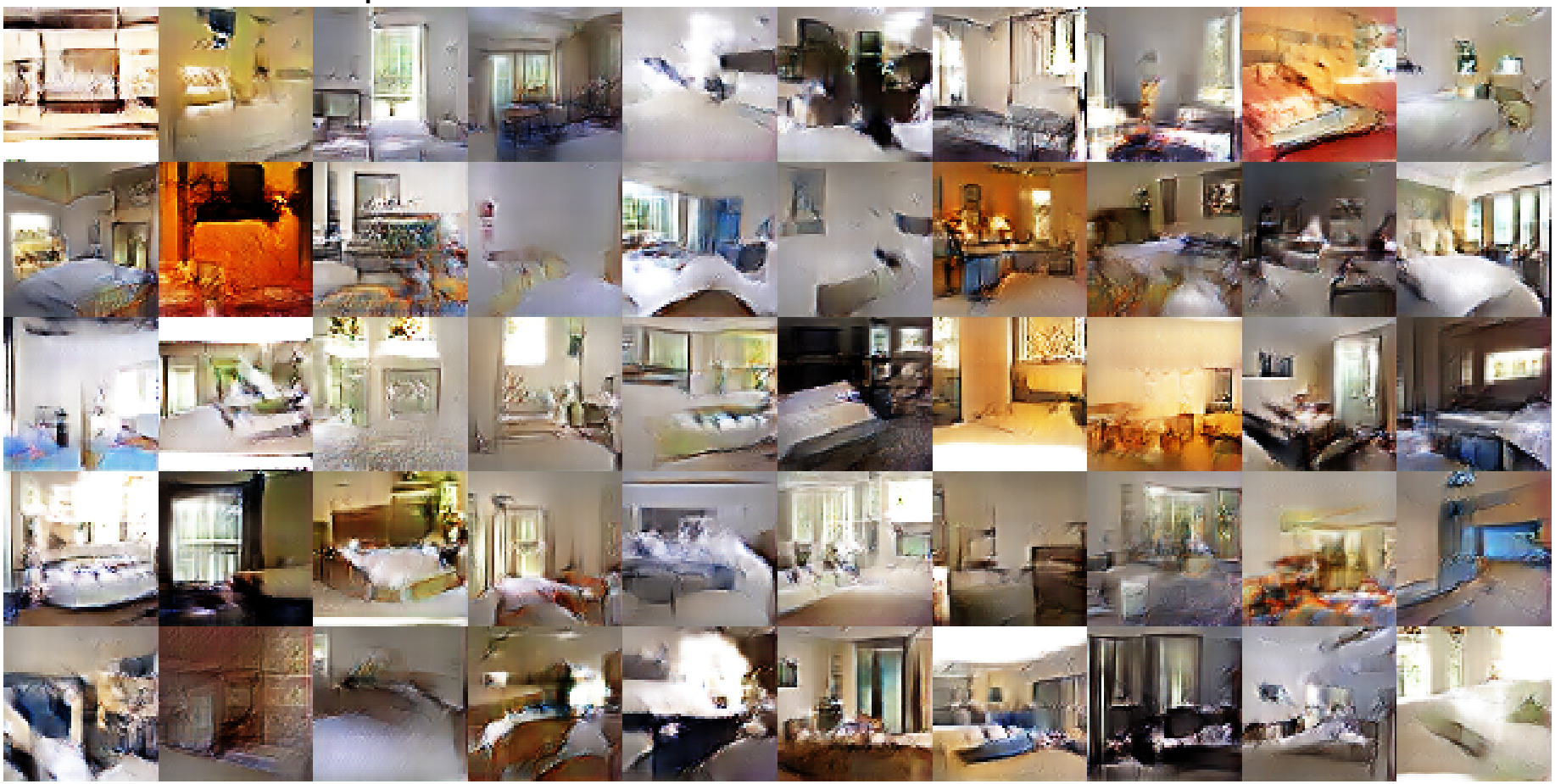
  where $\boldsymbol{\theta}[i]$ is the value of parameters at past time i
- Historical average of the parameters can be updated in an online fashion

Similar tricks from RL also work for GANs, e.g. experience replay:

- Keep a replay buffer of past generations and occasionally show them
- Keep checkpoints from the past of G and D and occasionally swap them out for a few iterations

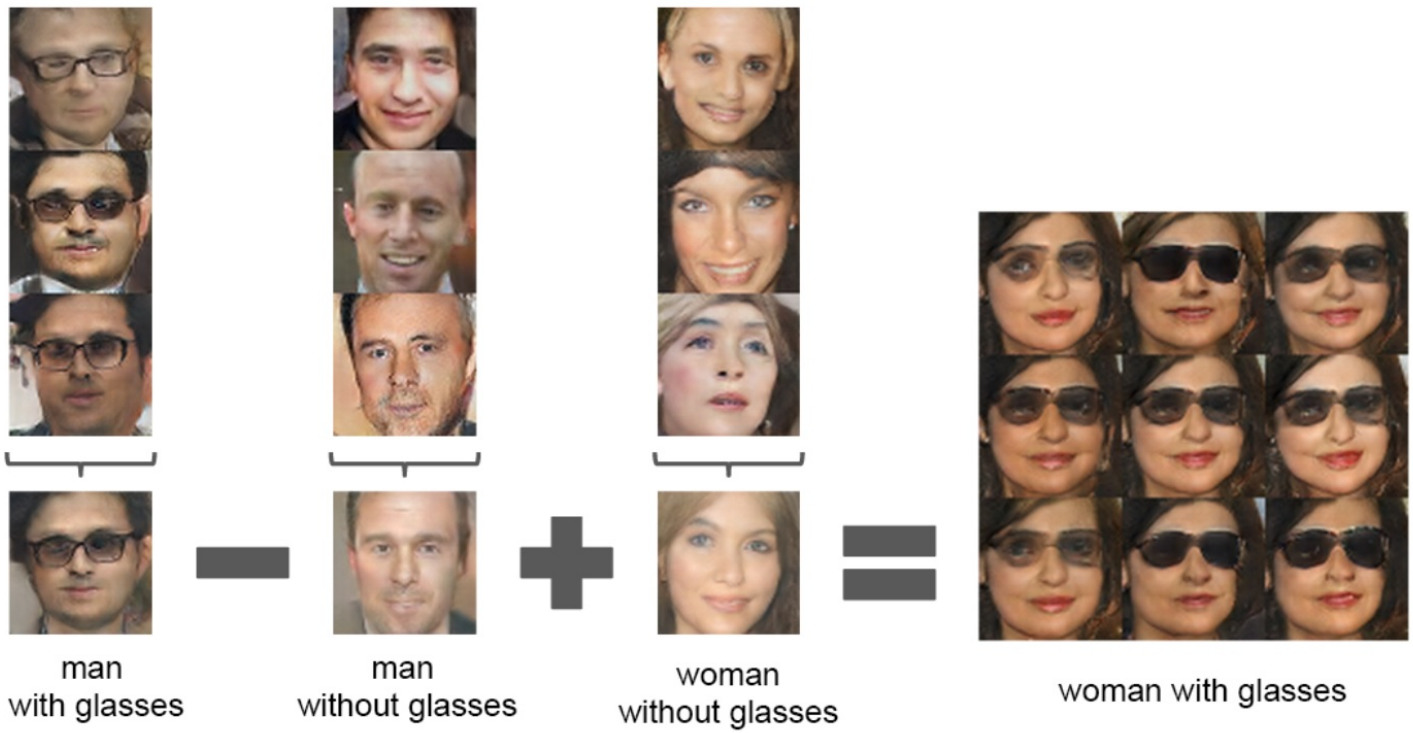
DCGAN examples

Bedrooms after 1 epoch



Source: [16]

Vector Arithmetic



- Average three latents codes $\boldsymbol{z}$ and apply operation

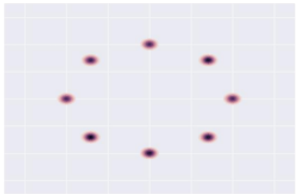
➔ GANs learn a distributed representation that disentangles the concept of gender from the concept of wearing glass

- See also “InfoGAN” [1]

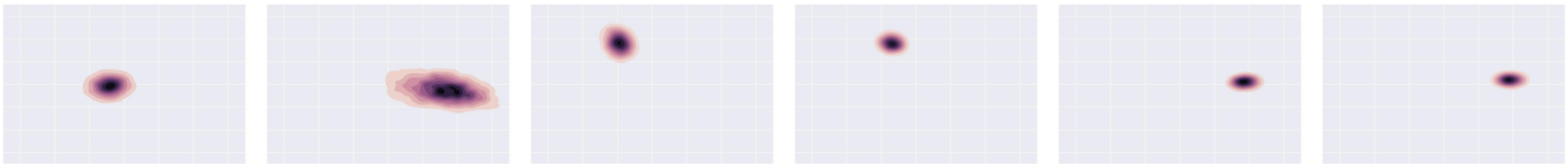
Source: [16]

Advanced GAN Methods

Mode Collapse



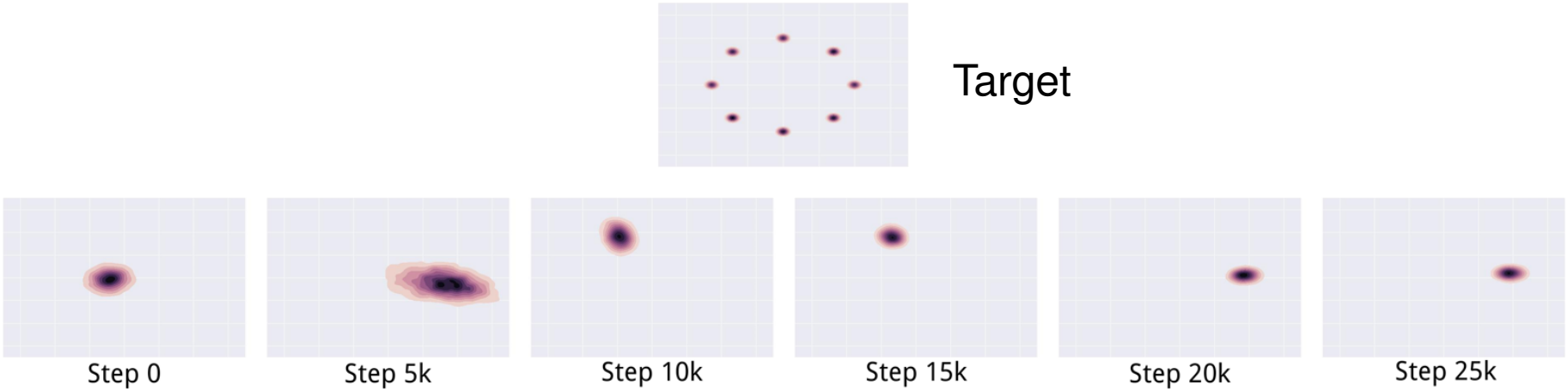
Target



- *G* rotates through the modes of the data distribution
- Never converges to a fixed distribution

Source: [14]

Mode Collapse



Possible Reason

$$\min_G \max_D V(G, D) \neq \max_D \min_G V(G, D)$$

- D in inner loop: convergence to correct distribution
- G in inner loop: place all mass on most likely point
- In practice: Simultaneous SGD of both networks ➔ both effects can appear
- Solutions: Minibatch discrimination or unrolled GANs

Source: [14]

Minibatch Discrimination

Intuition: Allow *D* to look at multiple samples in combination to help *G* avoid collapsing

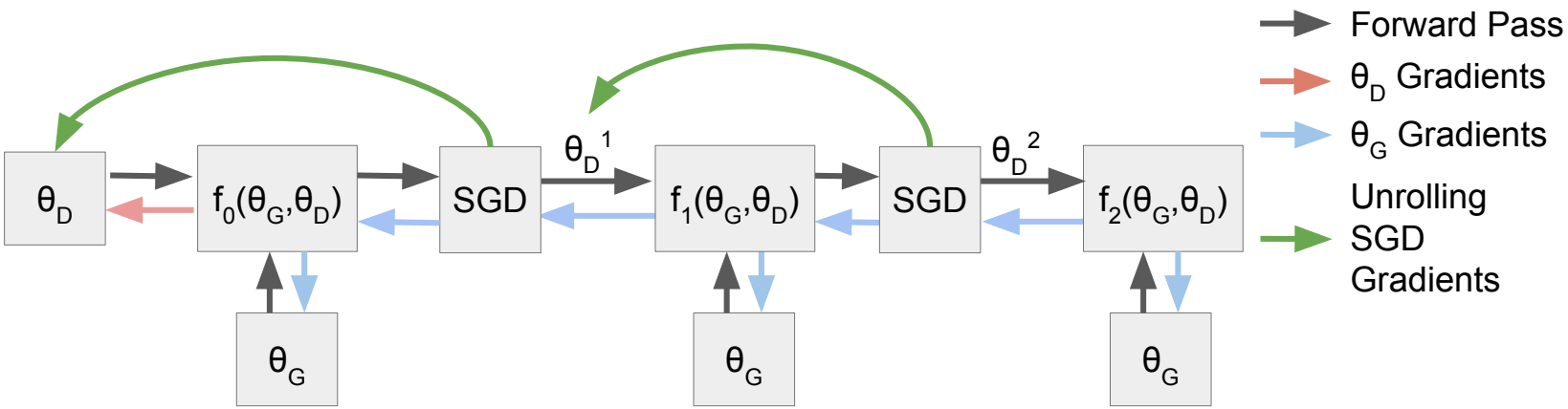
1. Extract features from intermediate layer
2. Add **minibatch layer** that computes for each feature
    1. a similarity to all other samples of the mini-batch
    2. concatenate similarity vector to each feature

- Compute these minibatch features separately for samples from *G* and from training data
- *D* still outputs 0/1 but now uses the similarity to all samples in the mini-batch as side information

Unrolled GAN

- Ideally: $G^* = \min_G \max_D V(G, D)$
- But essentially, we ignore the max operation when computing G's gradient
- Idea: Regard $\max_D V(G, D)$ as cost for G ➔ need to back-propagate through the maximization operation

Unrolled GAN (cont.)



Unrolled GAN with $k = 3$

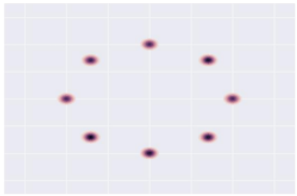
1. Build computational graph describing k steps of learning in D
2. Back-propagate through all k steps when computing G's gradient

- Fully maximizing D's value function intractable

➔ Even a low number of $k = 10$ already substantially reduces mode collapse

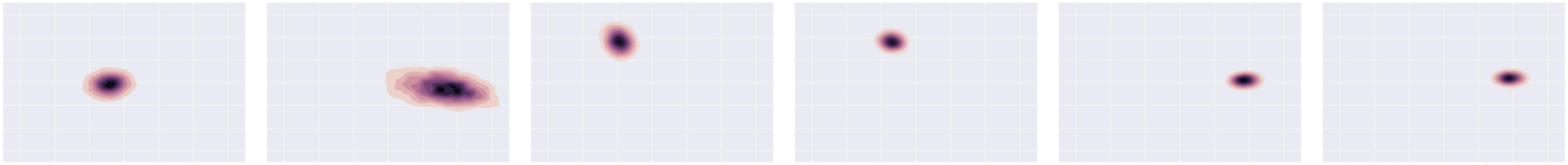
Source: [14]

Unrolled GAN (cont.)

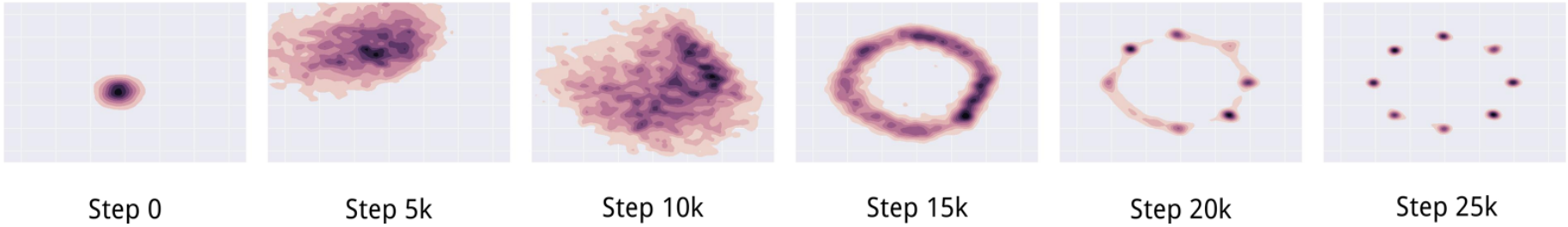


Target

GAN



Unrolled GAN



Source: [14]

GANs for Semi-supervised Learning

- Idea: Use GANs by turning the problem from a K class problem into a problem with $K + 1$ classes
- "True classes": Target classes for supervised learning
- Additional class: Fake inputs generated by the generator
- Probability of image being real: sum of "real class" probabilities
- Discriminator is used as a classifier within the GAN game

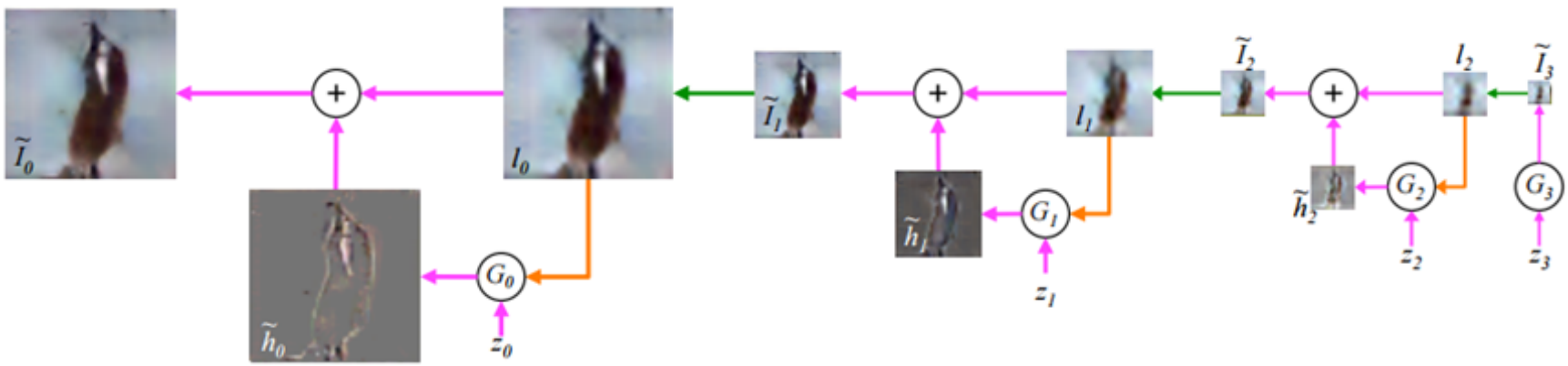
Source: [14]

Multi-scale approaches: Laplacian Pyramid of GANs

- GANs are pretty good at generating low-resolution images, but
- High-resolution images are much more difficult!

➔ **Subsequently** increase resolution and add detail with pyramid of GANs

- **Input**: Noise + upsampled output from previous *G* as **conditioning variable**
- **Output**: **Difference image** for current resolution

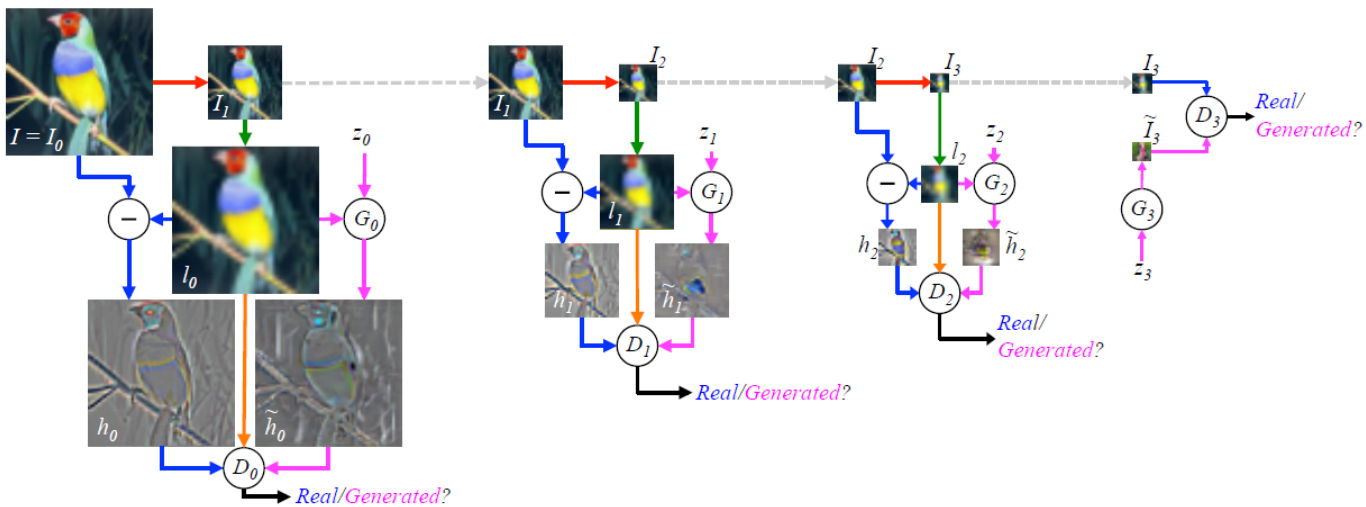


Generator hierarchy. Read right to left!

Source: Denton et al. [3]

Multi-scale approaches: Laplacian Pyramid of GANs (cont.)

- Train generators by training a discriminator on each level
- Input for discriminator: difference images



LAPGAN training.

- But still only 64 $\times$ 64 pixels!

Source: Denton et al. [3]

StackGANs: Text to Photo-realistic Image synthesis

- Task: Given some text, generate a fitting image
- Decompose problem: Sketch-refinement using a two-stage conditional GANs
- Analogon: Human painting – sketch and fine details
- Stage-I GAN: Draws low resolution images
  - Conditioned on text descriptions
  - Sketching rough shape / basic colors from the given text
  - Paints the background
- Stage-II GAN: Add Generates high resolution images
  - Conditioned on Stage-I result and text descriptions
  - Corrects defects and add details

StackGANs: Text to Photo-realistic Image synthesis (cont.)

Samples generated by StackGAN [20]

- Stage-I images are blurry with defects and missing details in foreground object
- Stage-II images have 4× higher resolution and “plausible” details

Source: Zhang et al. [19]

Summary

GANs

- are generative models that use supervised learning to approximate an intractable cost function
- can simulate many cost functions
- hard to find equilibrium between D and G
- cannot generate discrete data
- can also be used for
  - (semi-)supervised classification
  - transfer learning
  - multi-modal outputs
  - ...

NEXT TIME

ON DEEP LEARNING

Coming Up:

- How to adapt neural networks to localize objects?
- Neural networks detecting object classes.
- Instead of classes detect specific instances of classes.
- Methods to go even finer and segment outlines.
- A neural network worth checking its citations every day.

Comprehensive Questions

- What is the basic idea of contrastive divergence?
- What is the defining characteristic of an autoencoder?
- How do denoising autoencoders work?
- What does an optimal discriminator for GANs learn?
- What are the advantages of GANs in comparison to other generative models?
- What is “mode collapse”?
- Explain feature matching/perceptual loss.

Further Reading

- Variational Autoencoders: http://dpkingma.com/wordpress/wp-content/uploads/2015/12/talk_nips_workshop_2015.pdf
- NIPS 2016 GAN Tutorial of Goodfellow https://www.youtube.com/watch?v=AJVyzd0rqdc
- How to train a GAN? Tips and tricks to make GANs work (careful, not everything is true anymore!) https://github.com/soumith/ganhacks
- Ever wondered about how to name your GAN? https://github.com/hindupuravinash/the-gan-zoo

References

References I

[1] Xi Chen, Xi Chen, Yan Duan, et al. “InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets”. In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc., 2016, pp. 2172–2180.

[2] Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, et al. “Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion”. In: Journal of Machine Learning Research 11.Dec (2010), pp. 3371–3408.

[3] Emily L. Denton, Soumith Chintala, Arthur Szlam, et al. “Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks”. In: CoRR abs/1506.05751 (2015). arXiv: 1506.05751.

[4] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern classification. 2nd ed. New York: Wiley-Interscience, Nov. 2000.

References II

[5] Asja Fischer and Christian Igel. “Training restricted Boltzmann machines: An introduction”. In: Pattern Recognition 47.1 (2014), pp. 25–39.

[6] John Gauthier. Conditional generative adversarial networks for face generation. Mar. 17, 2015. URL: http://www.foldl.me/2015/conditional-gans-face-generation/ (visited on 01/22/2018).

[7] Ian Goodfellow. NIPS 2016 Tutorial: Generative Adversarial Networks. 2016. eprint: arXiv:1701.00160.

[8] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, et al. “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium”. In: Advances in Neural Information Processing Systems 30. Curran Associates, Inc., 2017, pp. 6626–6637.

References III

[9] Geoffrey E Hinton and Ruslan R Salakhutdinov. “Reducing the dimensionality of data with neural networks.”. In: Science 313.5786 (July 2006), pp. 504–507. arXiv: 20.

[10] Geoffrey E. Hinton. “A Practical Guide to Training Restricted Boltzmann Machines”. In: Neural Networks: Tricks of the Trade: Second Edition. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 599–619.

[11] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, et al. “Image-to-Image Translation with Conditional Adversarial Networks”. In: (2016). eprint: arXiv:1611.07004.

[12] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: arXiv e-prints, arXiv:1312.6114 (Dec. 2013), arXiv:1312.6114. arXiv: 1312.6114 [stat.ML].

References IV

[13] Jonathan Masci, Ueli Meier, Dan Ciresan, et al. “Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction”. In: Artificial Neural Networks and Machine Learning – ICANN 2011. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 52–59.

[14] Luke Metz, Ben Poole, David Pfau, et al. “Unrolled Generative Adversarial Networks”. In: International Conference on Learning Representations. Apr. 2017. eprint: arXiv:1611.02163.

[15] Mehdi Mirza and Simon Osindero. “Conditional Generative Adversarial Nets”. In: CoRR abs/1411.1784 (2014). arXiv: 1411.1784.

[16] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Ad 2015. eprint: arXiv:1511.06434.

References V

[17] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, et al. “Improved Techniques for Training GANs”. In: Advances in Neural Information Processing Systems 29. Curran Associates, Inc., 2016, pp. 2234–2242.

[18] Andrew Ng. “CS294A Lecture notes”. In: 2011.

[19] Han Zhang, Tao Xu, Hongsheng Li, et al. “StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks”. In: CoRR abs/1612.03242 (2016). arXiv: 1612.03242.

[20] Han Zhang, Tao Xu, Hongsheng Li, et al. “Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks”. In: arXiv preprint arXiv:1612.03242 (2016).

References VI

[21] Bolei Zhou, Aditya Khosla, Agata Lapedriza, et al. “Learning Deep Features for Discriminative Localization”. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Las Vegas, June 2016, pp. 2921–2929. arXiv: 1512.04150.

[22] Jun-Yan Zhu, Taesung Park, Phillip Isola, et al. “Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks”. In: CoRR abs/1703.10593 (2017). arXiv: 1703.10593.