



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Unsupervised Deep Learning

A. Maier, K. Breininger, L. Mill, N. Ravikumar, T. Würfel

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

June 20, 2018





FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Motivation



Motivation

Dataset variation

So far

Dataset size	Huge (up to millions!)
Variation	Many objects
Modalities	Few modalities

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

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

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

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: [12]

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

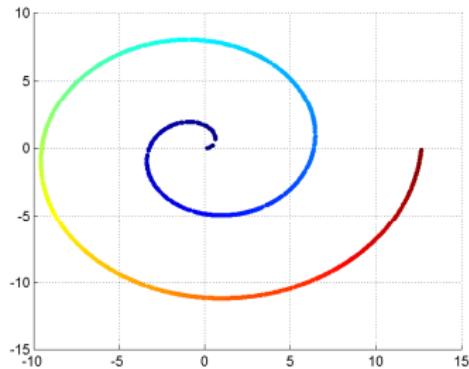
Source: [12]

Label-free Learning

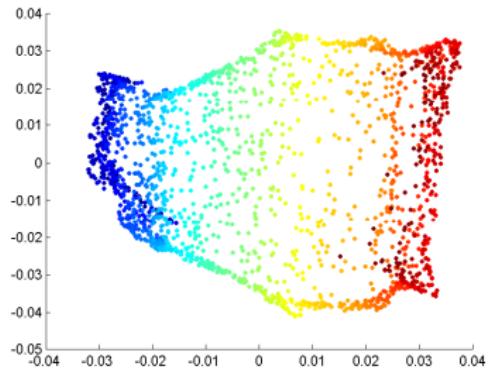
Applications as

- Dimensionality Reduction

Data on a manifold



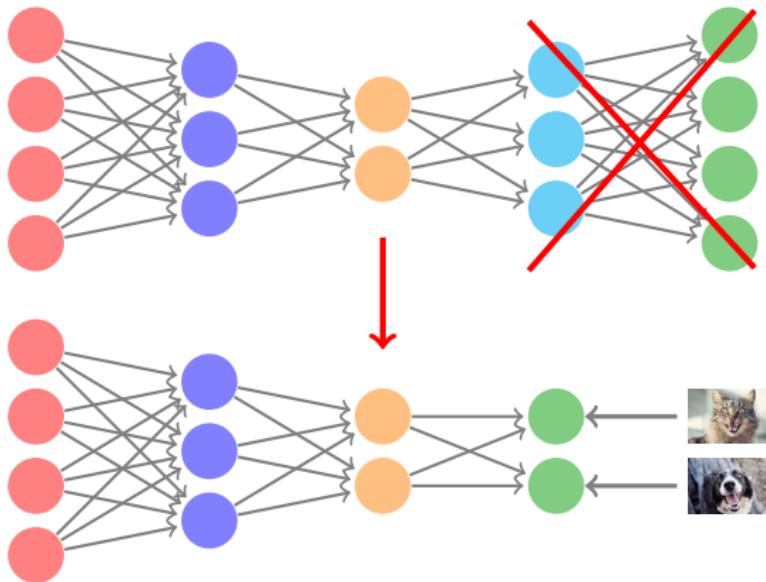
Dimensionality reduced



Label-free Learning

Applications as

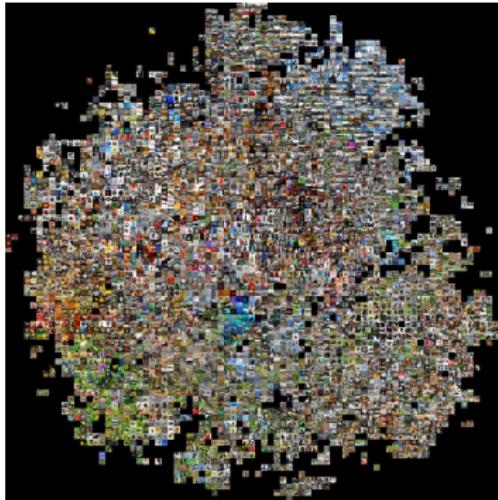
- Network Initialization (related: transfer learning)



Label-free Learning

Applications as

- Representation Learning
- e.g. for clustering

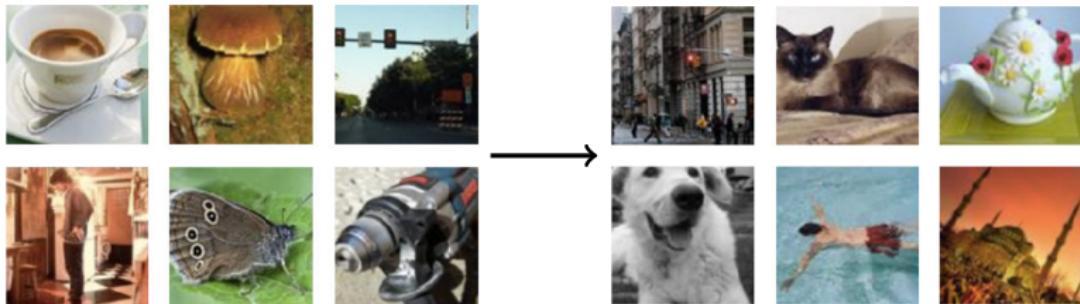


Source: <http://cs.stanford.edu/people/karpathy/cnnembed/>

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: [5]

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 (not covered today)
- 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



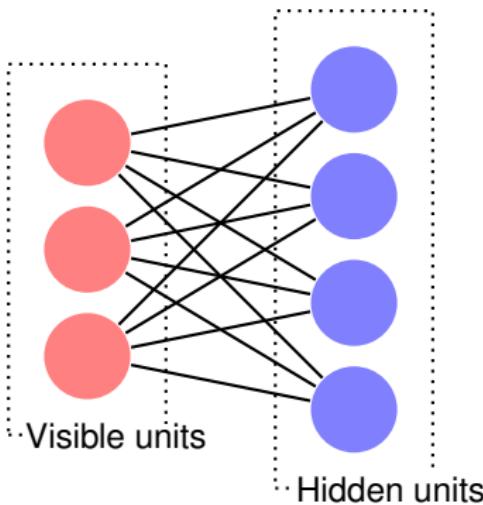
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Restricted Boltzmann Machines



Restricted Boltzmann Machine (RBM)



- Binary (Bernoulli) visible units \mathbf{v} represent observed data
- Binary (Bernoulli) hidden units \mathbf{h} capture dependencies (latent variables), learn representation of the input
- Uses the **Boltzmann** distribution which is also known as **Softmax**

Restricted Boltzmann Machine (RBM)

- An RBM learns a distribution $p(E(\mathbf{v}, \mathbf{h}))$ defined in terms of an energy function:

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

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

b, c: biases

W: connection matrix

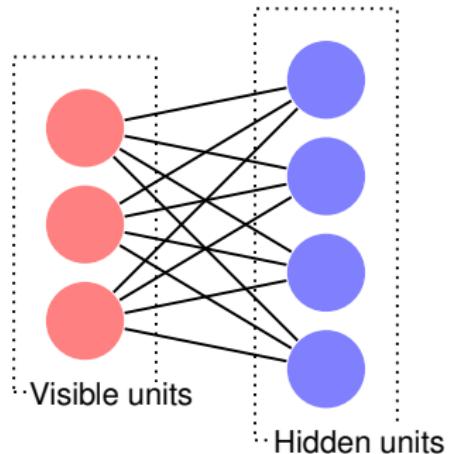
Z: “partition function” (normalization constant),
sum over all possible hidden/visible pairs:

$$Z = \sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h}))$$

- 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
- Learning based on gradient descent on the negative log-likelihood



Training RBMs

θ : model parameters

$$\log L(\theta | \mathbf{v}) = p(\mathbf{v} | \theta)$$

$$= \log \frac{1}{Z} \sum_{\mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) - \log \left(\sum_{\mathbf{v}, \mathbf{h}} \exp(-E(\mathbf{v}, \mathbf{h})) \right)$$

$$\frac{\partial \log L(\theta | \mathbf{v})}{\partial \theta} = \dots \quad \text{see [3]}$$

$$= \underbrace{\sum_{\mathbf{h}} p(\mathbf{h}, \mathbf{v}) \frac{-\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}}_{\mathbb{E}_{\text{data}}} - \underbrace{\sum_{\mathbf{v}, \mathbf{h}} p(\mathbf{v}, \mathbf{h}) \frac{-\partial E(\mathbf{v}, \mathbf{h})}{\partial \theta}}_{\mathbb{E}_{\text{model}}}$$

$\mathbb{E}_{\text{model}}$ generally intractable $\rightarrow O(\theta)$

\rightarrow Approximate via contrastive divergence

Training RBMs

Contrastive divergence

1. Take any training example as \mathbf{v}
2. Set binary states of the hidden units: $p(h_i = 1 | \mathbf{v}) = \sigma(\sum_j w_{ij} v_j + c_i)$
3. Run k Gibbs sampling steps:

3.1 Sample reconstruction $\tilde{\mathbf{v}}$ by $p(v_j = 1 | \mathbf{h}) = \sigma(\sum_i w_{ij} h_i + b_j)$

3.2 Resample $\tilde{\mathbf{h}}$

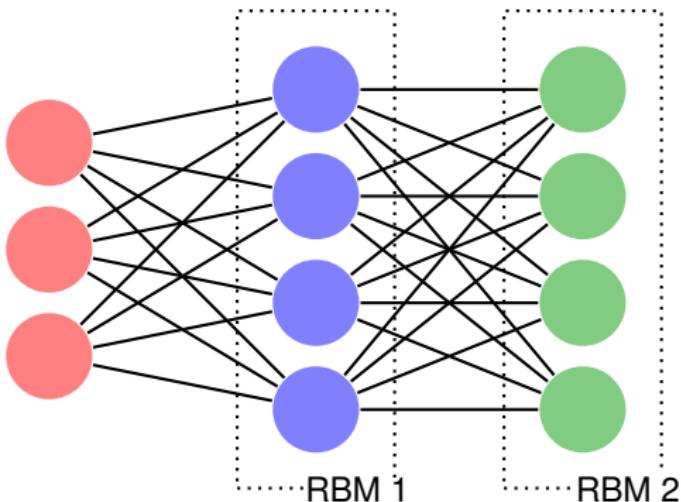
$$\Delta w_{ij} = \eta(\mathbf{v}\mathbf{h}^\top - \tilde{\mathbf{v}}\tilde{\mathbf{h}}^\top)$$

$$\Delta b = \eta(\mathbf{v} - \tilde{\mathbf{v}})$$

$$\Delta c = \eta(\mathbf{h} - \tilde{\mathbf{h}})$$

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

Deep Belief Network (DBN)



- 4. 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 [6]
 - Sparked the “deep learning renaissance” (nowadays rarely used anymore)



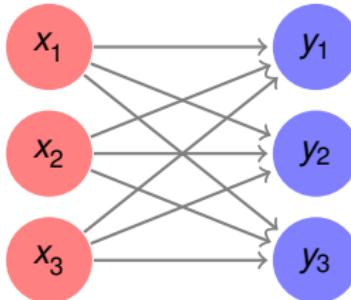
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Autoencoder



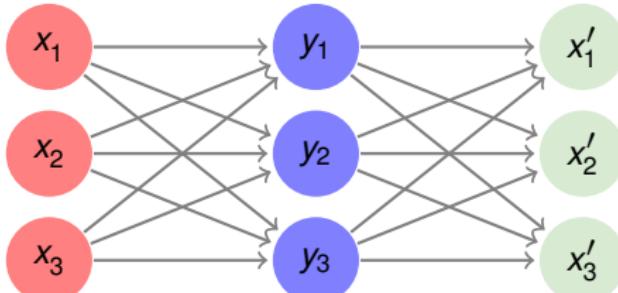
Autoencoder (AE)



Concept

- Special case of feed-forward neural network
- **Encoder:** $y = f(x)$
- ... but how do we get a **loss?**

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

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

Autoencoder – Loss Functions

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

Squared L₂ norm

$$p(\mathbf{x}|\mathbf{x}') \sim \mathcal{N}(\mathbf{x}', \sigma^2 \mathbf{I})$$

$$L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$$

Autoencoder – Loss Functions

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

Squared L₂ norm

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

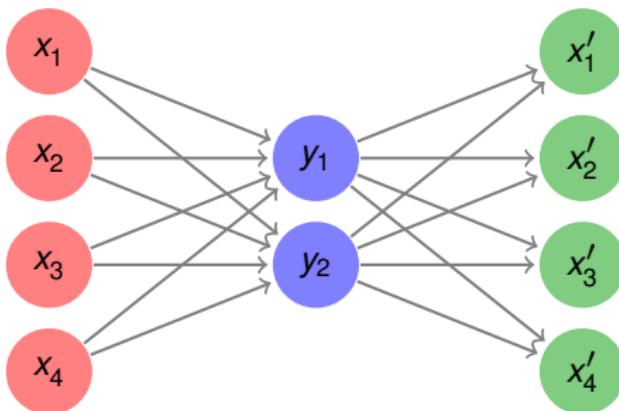
$$L(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2^2$$

Cross entropy

$$p(X|\mathbf{x}') \sim \mathcal{B}(\mathbf{x}')$$

$$L(\mathbf{x}, \mathbf{x}') = -\sum_{i=1}^n x_i \log(x'_i) + (1 - x_i) \log(1 - x'_i)$$

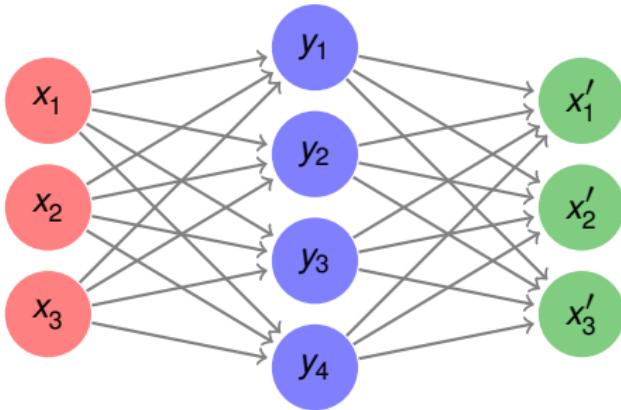
Enforce Information Compression



Undercomplete AE

- Prevent network from simply copying the input
- AE with linear layers and squared L₂ norm as loss learns a PCA
- AE with nonlinear layers: nonlinear PCA generalization

Sparse Autoencoder



- Tries to enforce sparsity in the **activations** (as opposed to weights!)

$$L'(\mathbf{w}) = L(\mathbf{w}) + \beta \sum_{j=1}^n \text{KL}(\rho \parallel \hat{\rho}_j)$$

- Here $\hat{\rho}_j$ is the **average activation over the dataset** of unit j
- ρ is a parameter $\ll 1$ controlling sparsity



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Autoencoder Variations



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 $C(\hat{\mathbf{x}}|\mathbf{x})$:
 - $C(\hat{\mathbf{x}}|\mathbf{x}) \sim \mathcal{N}(\mathbf{x}, \sigma^2 \mathbf{I})$
 - Set random elements to zero
- Denoising
- Regularization (similar to dropout but applied to input layers)

Autoencoder Variations / Important Terms

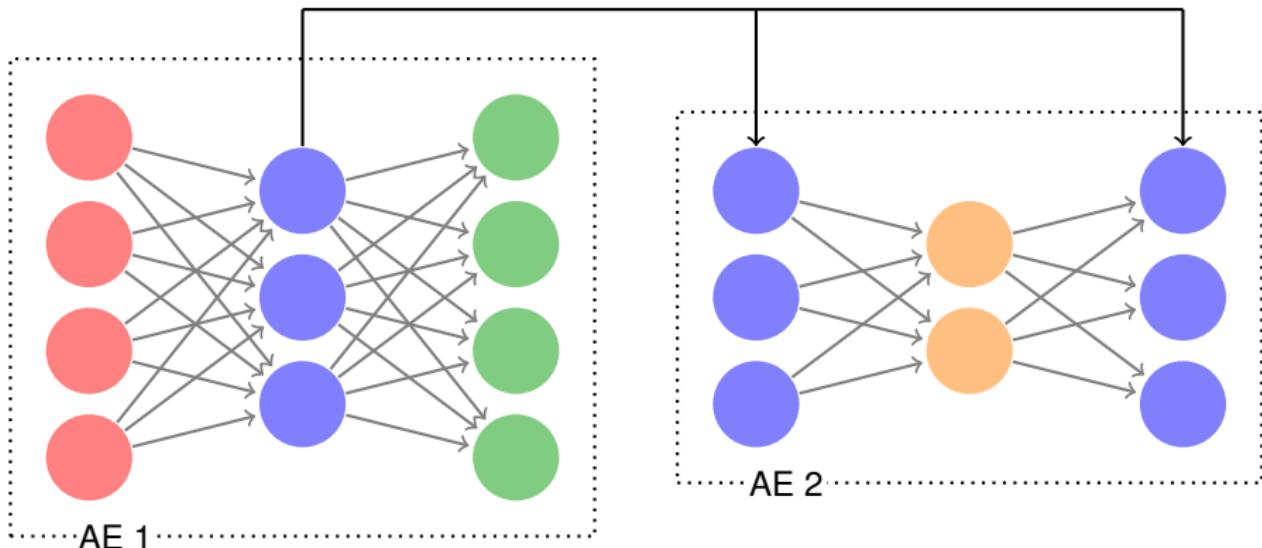
DAE as Generative Model

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

Often expensive and hard to assess convergence
→ Variational AEs much more common!

Autoencoder Variations / Important Terms

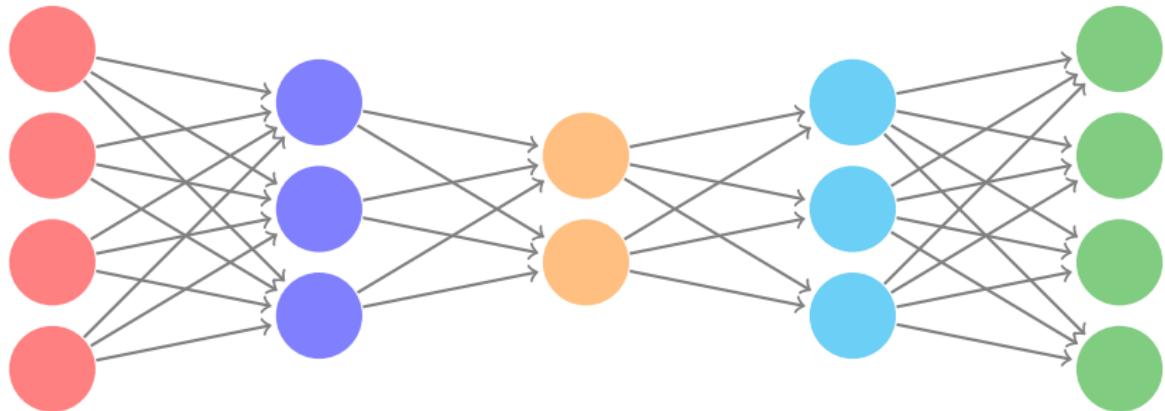
Stacked Autoencoder



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



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Generative Adversarial Networks



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

Discriminator



Generator



Source: <http://www.kadena.af.mil/News/Art/igphoto/2000206283>

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

Discriminator



Generator



...

Real data

Source: <http://www.kadena.af.mil/News/Art/igphoto/2000206283>

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

Discriminator



Generator



real / fake



...

Real data

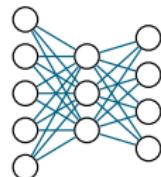
Source: <http://www.kadena.af.mil/News/Art/igphoto/2000206283>

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

$$D = D(\theta^D; \mathbf{x})$$



$$G = G(\theta^G; \mathbf{z})$$



real: 1 / fake: 0



...

Real data $\mathbf{x} \sim p_{\text{data}}$

Noise $\mathbf{z} \sim p_z$

Source: <http://www.kadena.af.mil/News/Art/igphoto/2000206283>

Training GANs – Minimax Game

Alternate between:

1. Train D : minimize

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

→ Trained to distinguish real data samples from fake ones

Training GANs – Minimax Game

Alternate between:

1. Train D : minimize

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

→ 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

Training GANs – Minimax Game

Alternate between:

1. Train D : minimize

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

→ 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.)

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

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

- L^G is tied directly to $-L^D$
- Summarize game with a **value function** specifying the discriminator's payoff:

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

Minimax game:

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

Training GANs – Optimal Discriminator

- Assumption: both densities are nonzero everywhere
 - Otherwise some input values are never trained → some $D(\mathbf{x})$ have undetermined behavior
- Solve for: $\frac{\partial L^D}{\partial D(\mathbf{x})} = 0$
- Optimal $D^*(\mathbf{x})$ for any $p_{\text{data}}(\mathbf{x})$ and $p_{\text{model}}(\mathbf{x})$:

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

- 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

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

$$L^G = -\frac{1}{2} \log D(G(\mathbf{z}))$$

- 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 $f(\mathbf{x})$ of intermediate layer of D :

$$L^G = \|\mathbb{E}_{\mathbf{x} \sim p_{data}} f(\mathbf{x}) - \mathbb{E}_{\mathbf{z} \sim p_{\mathbf{z}}} f(G(\mathbf{z}))\|_2^2$$

→ Prevent “overtraining” of G on current D

- Popular loss also in other domains

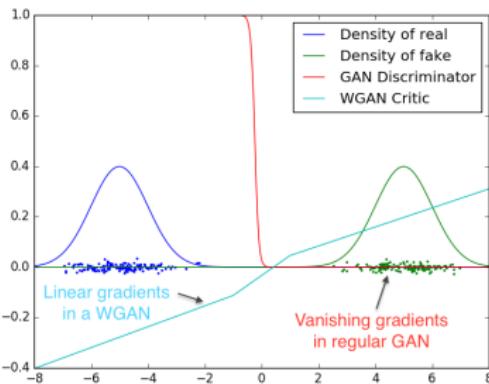
Other Popular Loss Functions

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:

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

- Effectively computed by clipping D 's gradients
- Helps to counter vanishing gradients in D



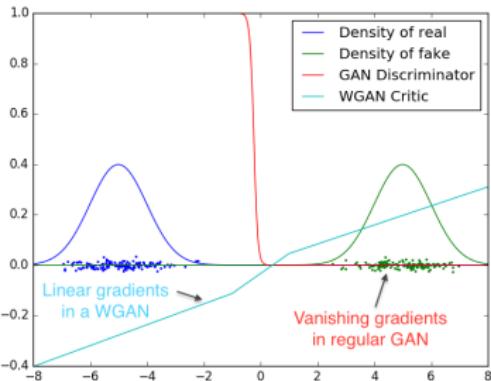
Other Popular Loss Functions

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:

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

- Effectively computed by clipping D 's gradients
- 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**



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

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

Conditional GANs

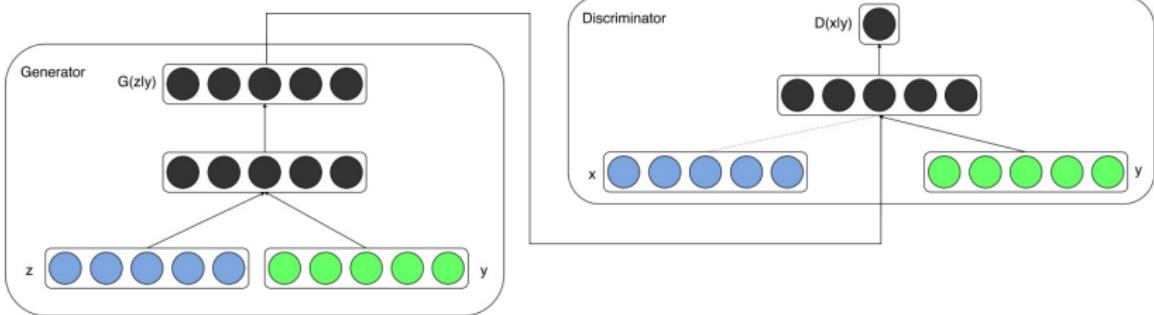
- Problem: Generator creates a “fake” generic image → is not specific for a certain condition/characteristic
- Example: **text to image generation** – image should **depend** on the text
- Idea: Provide additional vector **y** to both networks to encode **conditioning** [9]



Generated samples conditioned on one label (digit)

Source: Mirza et al. [9]

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:

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

Source: Mirza et al. [9]

Example: Conditional GANs for Face Generation

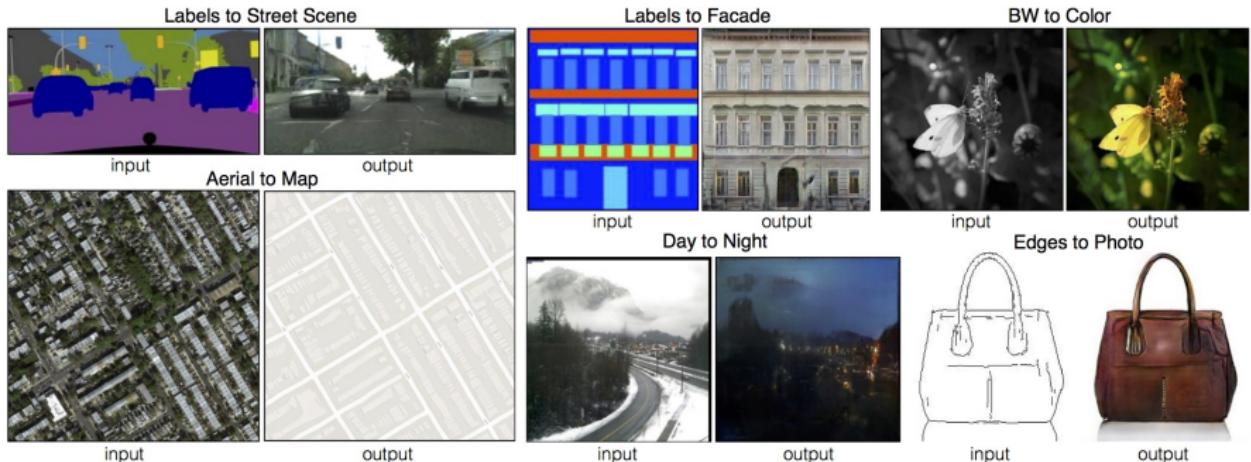
- Add conditional feature (e.g., smiling, gender, old age, ...)
- Generator/Discriminator learn to operate in **modes**:
 - Generator learns to generate a face with a certain attribute
 - Discriminator learns to decide whether the face contains attribute



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

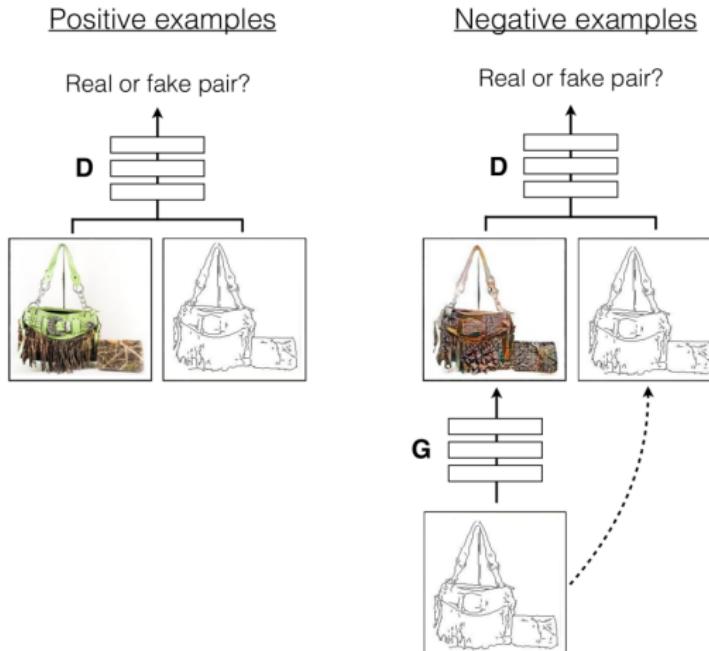
Source: Gauthier [4]

Image To Image



Source: [18]

Image To Image - Just a Conditional GAN!



Source: [18]



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

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?

Balancing G and D necessary?

No.

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

Balancing G and D necessary?

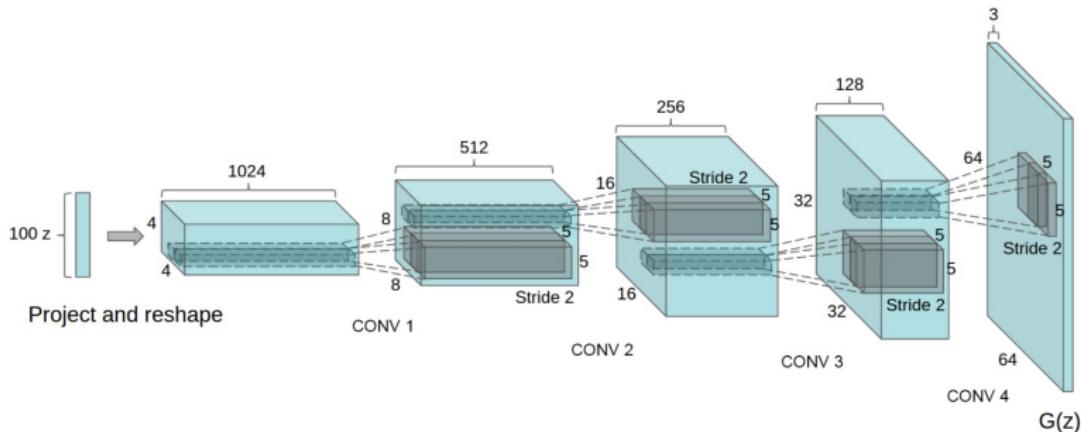
No.

- 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

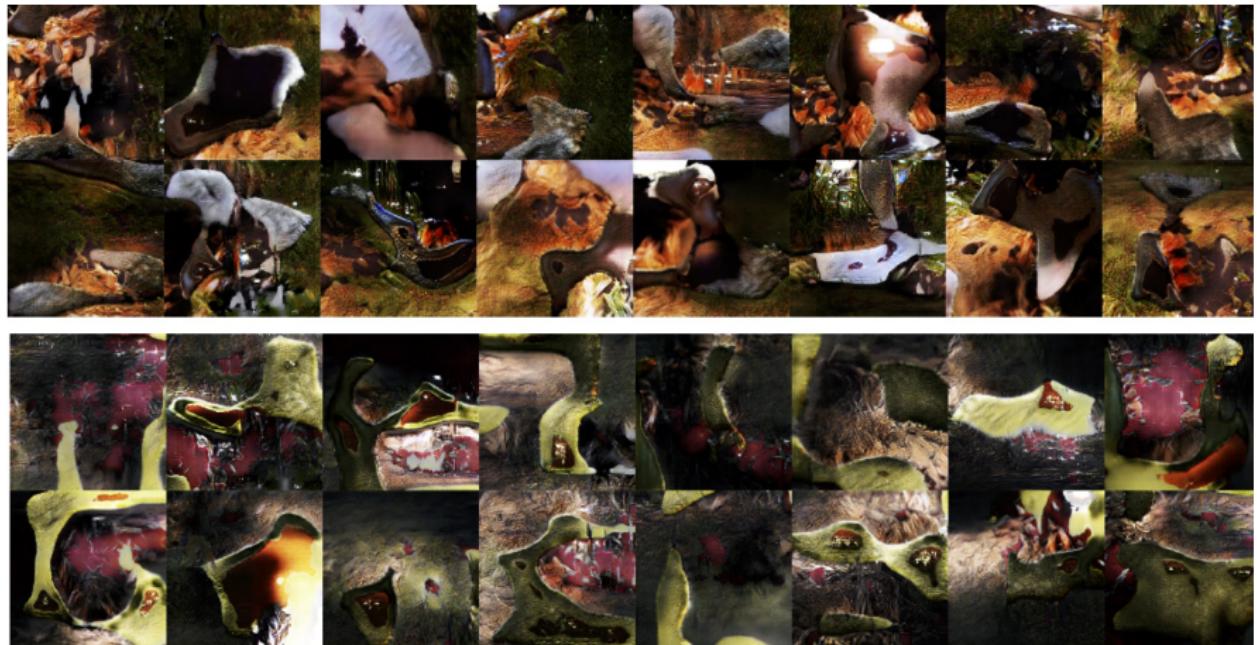
Deep Convolutional GANs (DCGAN)



- 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: [13]

Problem of Batch Normalization in G



→ Causes strong intra-batch correlation

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 \mathbf{x}_i of the current mini-batch
 - Create a new **virtual batch** $V_i = R \cup \mathbf{x}_i$
 - Compute mean and standard deviation of V_i
 - We always need to propagate R forward **in addition** to the current batch
 - Normalize \mathbf{x}_i with these statistics

Historical Averaging

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

$$\| \theta - \frac{1}{T} \sum_{i=1}^T \theta[i] \|_2^2$$

where $\theta[i]$ is the value of parameters at past time i

- Historical average of the parameters can be updated in an online fashion

Historical Averaging

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

$$\| \theta - \frac{1}{T} \sum_{i=1}^T \theta[i] \|_2^2$$

where $\theta[i]$ is the value of parameters at past time i

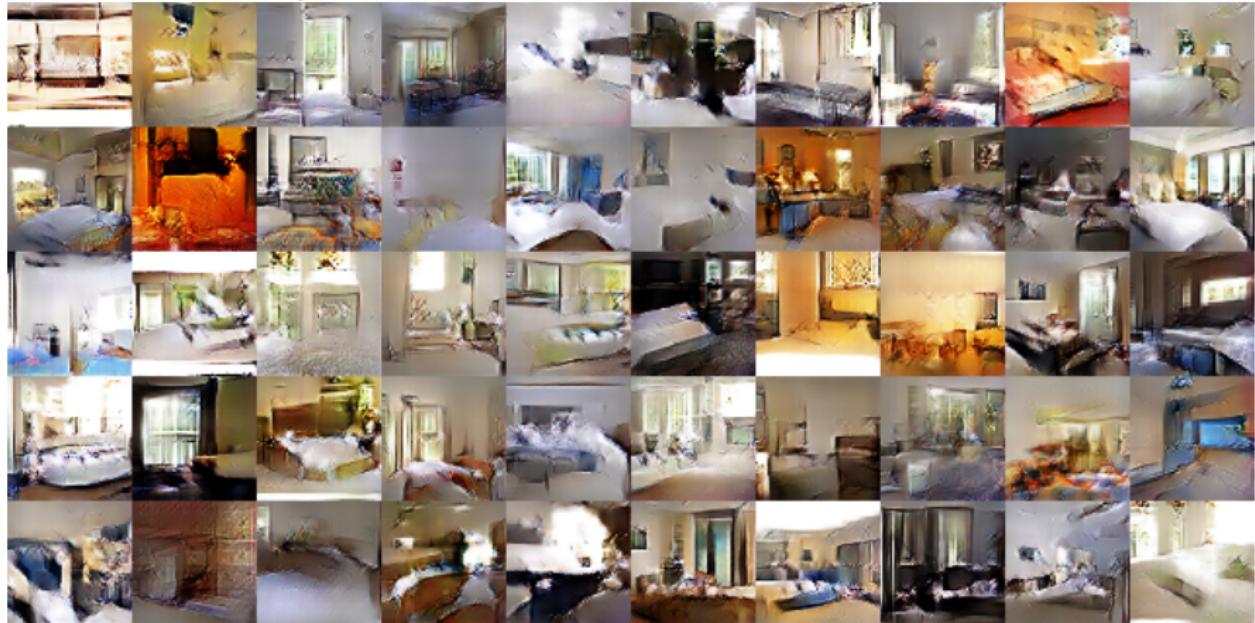
- Historical average of the parameters can be updated in an online fashion

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

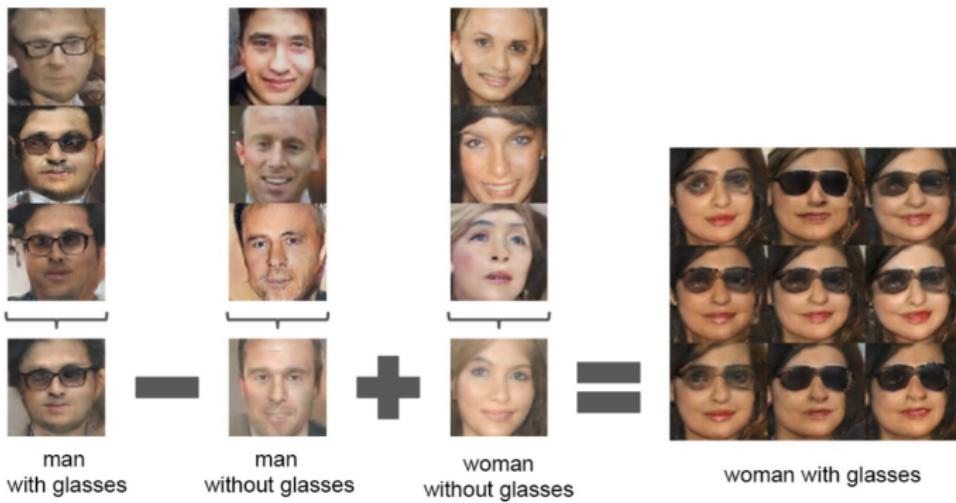
DCGAN examples

Bedrooms after 1 epoch



Source: [13]

Vector Arithmetic



- Average three latents codes \mathbf{z} and apply operation
- GANs learn a distributed representation that disentangles the concept of gender from the concept of wearing glass
- See also “InfoGAN” [14]

Source: [13]



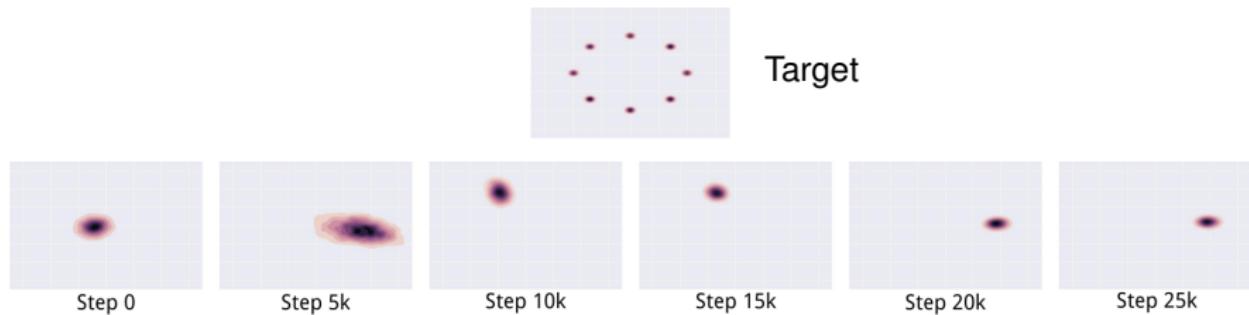
FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

Advanced GAN Methods



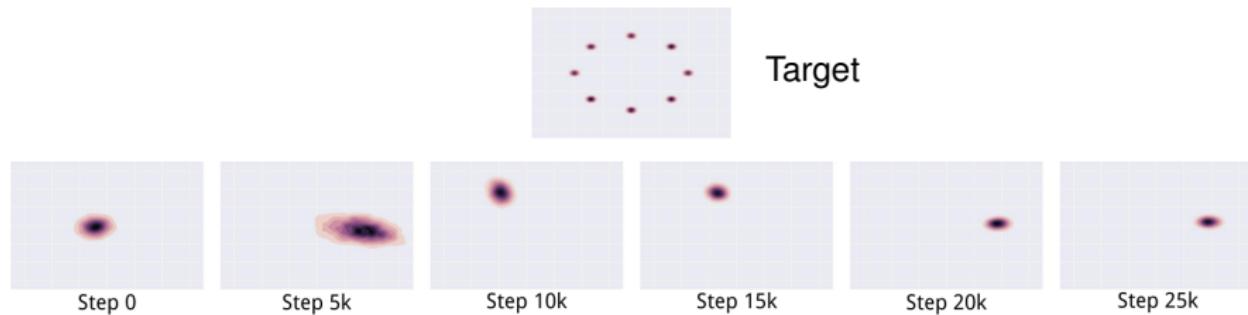
Mode Collapse



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

Source: [15]

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: [15]

Minibatch Discrimination

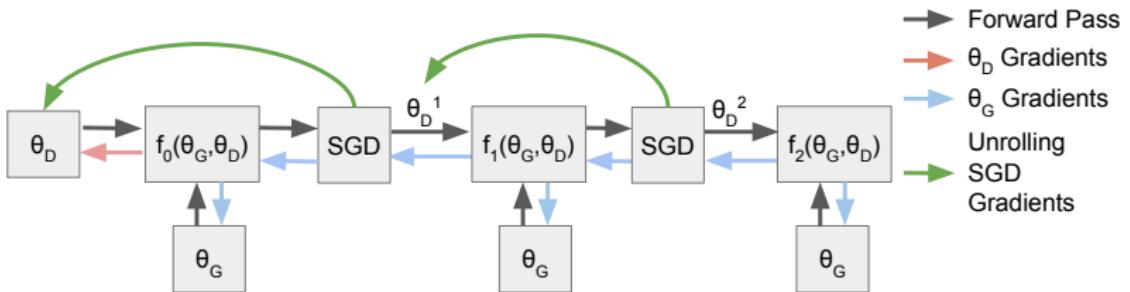
Intuition: Allow D to look at multiple samples in combination to help G avoid collapsing

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 $G \rightarrow$ need to back-propagate through the maximization operation

Unrolled GAN (cont.)

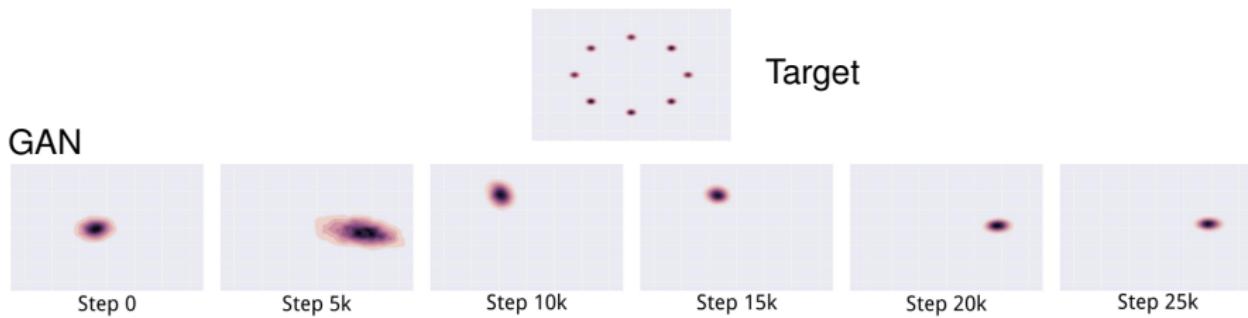


Unrolled GAN with $k = 3$

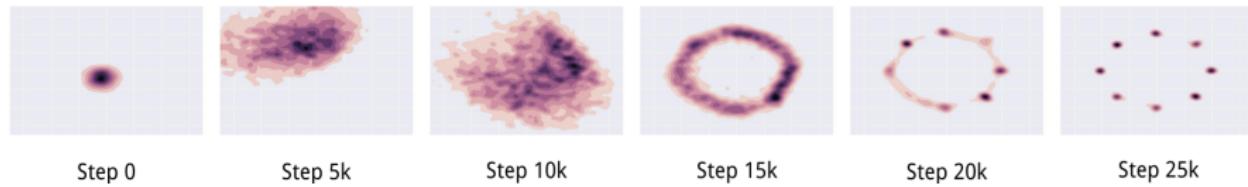
1. Build computational graph describing k steps of learning in D
2. Back-propagate through all k steps when computing G 's gradient
 - Fully maximizing D 's value function intractable
 - Even a low number of $k = 10$ already substantially reduces mode collapse

Source: [15]

Unrolled GAN (cont.)



Unrolled GAN



Source: [15]

GANs for Semi-supervised Learning

- Idea: Use GANs by turning the problem from a K class 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

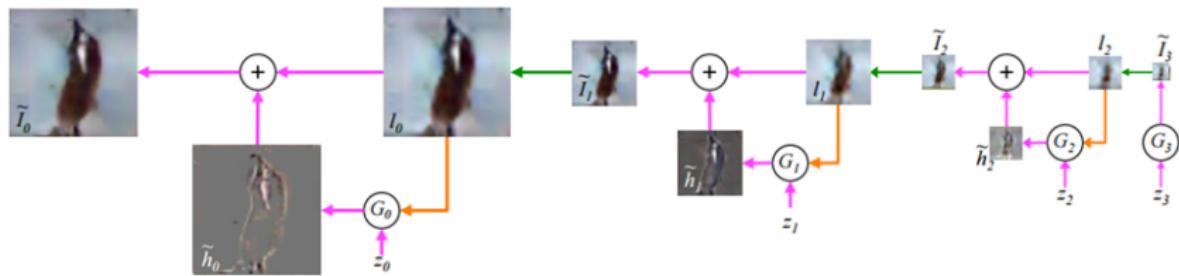
Source: [15]

Multi-scale approaches: Laplacian Pyramid of GANs

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

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

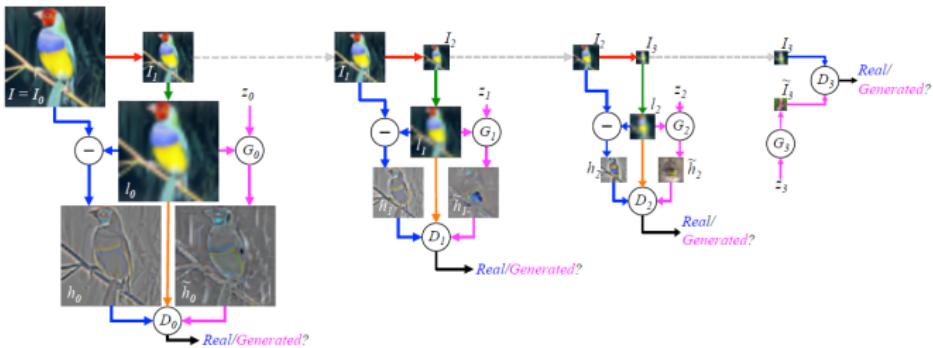


Generator hierarchy. Read right to left!

Source: Denton et al. [17]

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 64×64 pixels!

Source: Denton et al. [17]

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

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

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: Adds details to the generated image
 - Conditioned on Stage-I result and text descriptions
 - Corrects defects and adds details

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

Text description	This bird is blue with white and has a very short beak	This bird has wings that are brown and has a yellow belly	A white bird with a black crown and yellow beak	This bird is white, black, and brown in color, with a brown beak	The bird has small beak, with reddish brown crown and gray belly	This is a small, black bird with a white breast and white on the wingbars.	This bird is white black and yellow in color, with a short black beak
Stage-I images							
Stage-II images							

Samples generated by StackGAN [16]

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

Source: Zhang et al. [11]

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://dpmkingma.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>



FAU

FRIEDRICH-ALEXANDER-
UNIVERSITÄT
ERLANGEN-NÜRNBERG
SCHOOL OF ENGINEERING

References



References I

- [1] 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.
- [2] Richard O. Duda, Peter E. Hart, and David G. Stork. Pattern classification. 2nd ed. New York: Wiley-Interscience, Nov. 2000.
- [3] Asja Fischer and Christian Igel. "Training restricted Boltzmann machines: An introduction". In: Pattern Recognition 47.1 (2014), pp. 25–39.
- [4] John Gauthier.
Conditional generative adversarial networks for face generation. Mar. 17, 2015. URL:
<http://www.fold1.me/2015/conditional-gans-face-generation/>
(visited on 01/22/2018).

References II

- [5] Ian Goodfellow. [NIPS 2016 Tutorial: Generative Adversarial Networks](#). 2016. eprint: arXiv:1701.00160.
- [6] 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.
- [7] 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.
- [8] 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.

References III

- [9] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: [CoRR abs/1411.1784](#) (2014). arXiv: 1411.1784.
- [10] Andrew Ng. "CS294A Lecture notes". In: 2011.
- [11] 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.
- [12] 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.
- [13] Alec Radford, Luke Metz, and Soumith Chintala. [Unsupervised Representation Learning with Deep Convolutional Generative Models](#). 2015. eprint: arXiv:1511.06434.

References IV

- [14] 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.
- [15] Luke Metz, Ben Poole, David Pfau, et al. "Unrolled Generative Adversarial Networks". In: International Conference on Learning Representations. Apr. 2017. eprint: arXiv:1611.02163.
- [16] 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).
- [17] 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.

References V

- [18] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: (2016). eprint: [arXiv:1611.07004](https://arxiv.org/abs/1611.07004).