Generative Models: Part I

Part 9: Generative Models
- Variational Autoencoders
- Generative Adversarial Networks

Jonas Glombitza, Martin Erdmann

RWTH Aachen

Outline

I. Variational Autoencoders
II. Introduction to Generative Adversarial Networks (GANs)
- Adversarial frameworks

III. Tutorial: Implementation of GANs

BREAK

IV. Latest developments & advanced techniques
- Wasserstein GANs

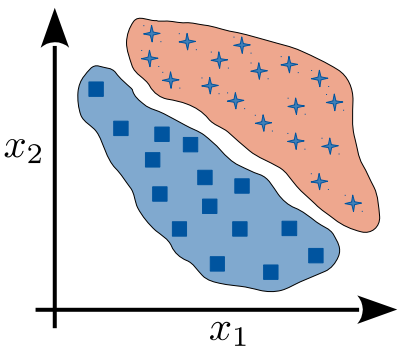
V. Application in physics research
- Simulation acceleration
- Style transfer (domain adaption)

VI. Tutorial: Implementation of Wasserstein GANs

Feel free to ask questions during the seminar!
Just “raise” your hand...

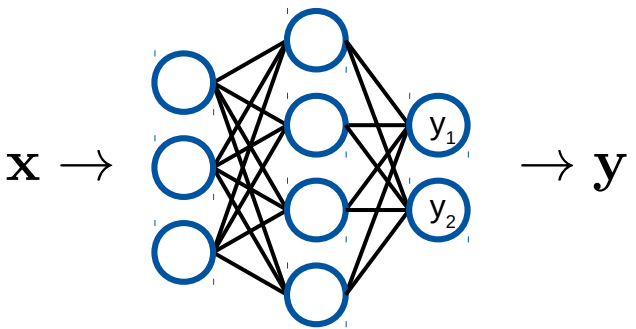
Supervised and Unsupervised Learning

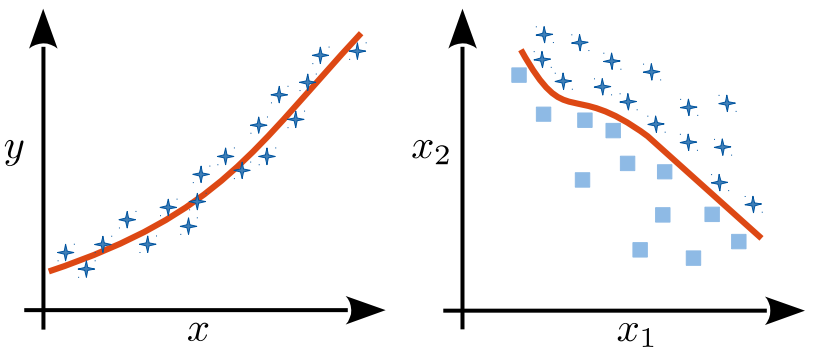
- Generative Models
  - Variational Autoencoders
  - Generative Adversarial Networks



Supervised Learning

- Situation
  - Large labeled data set (pair of input $\mathbf{x}$ and output $\mathbf{y}$)
- Typical Task:
  - Learn function to map input to specific output
  - Train model to predict the associated label
  - Achieve best generalization performance
  - Infer *conditional* probability density $p(\mathbf{x}|\mathbf{y})$





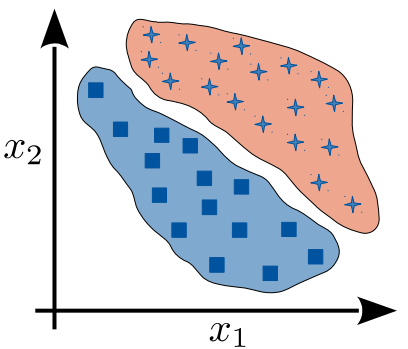
Unsupervised Learning

- Typical Situation: non labeled data set

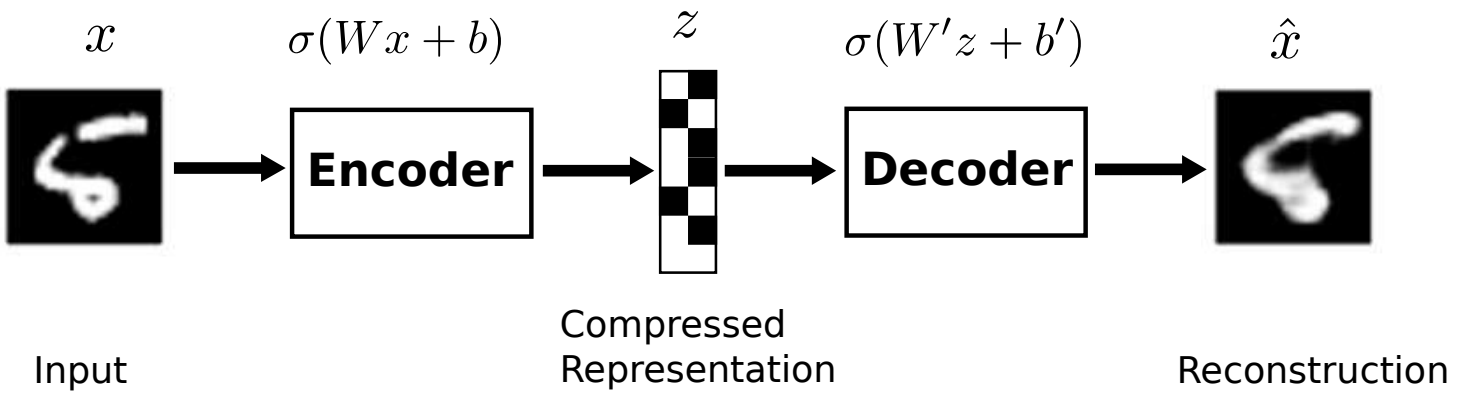
- Tasks:
  - Learn (low dimensional) data encodings → *autoencoders*
  - Estimate underlying probability density → *generative models*
  - Clustering, anomaly detection – find (non-) similar samples

- Infer *a priori* probability density $p(x)$

- Models typical trained without label information
  - Contrast: semi-supervised learning



Recap: Autoencoders



- Reconstruction of input data (approximation of identity function)
- Learning interesting representation (constraints to hidden layer)
- Objective function:

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{N} \sum_{i=1}^{N} (\hat{\mathbf{x}}_i - \mathbf{x}_i)^2$$

- Deep autoencoders often show underfitting → use shortcuts!

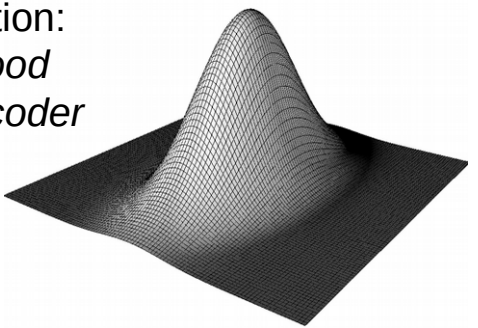
Generative Models

Approximate data distribution P_r with another distribution P_θ

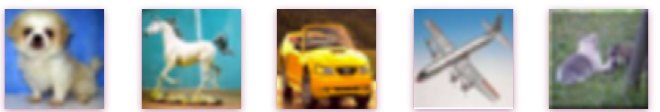
θ = distribution parameters

P_r

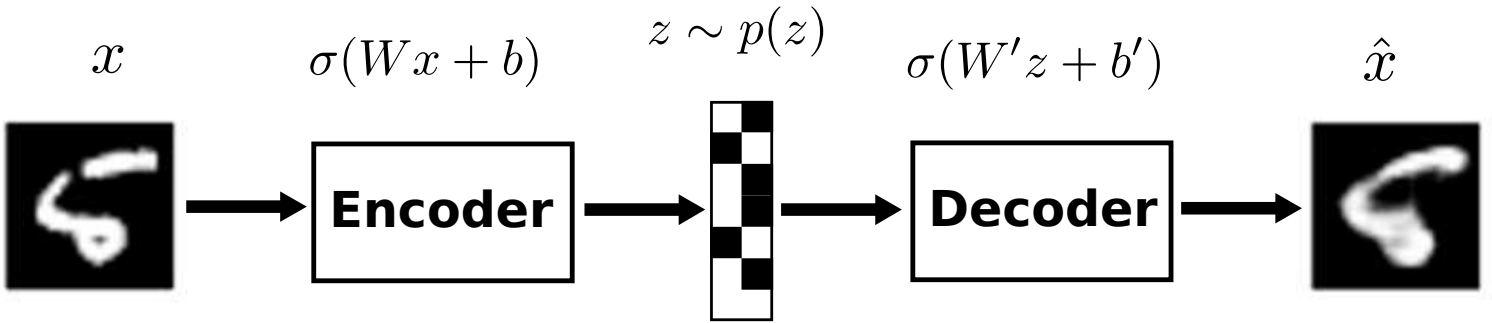
Learn prior distribution:
Maximizing Likelihood
Variational Autoencoder



Learn to generate samples following P_θ
Without using directly P_θ
Train a generator only → GANs



Variational Autoencoder

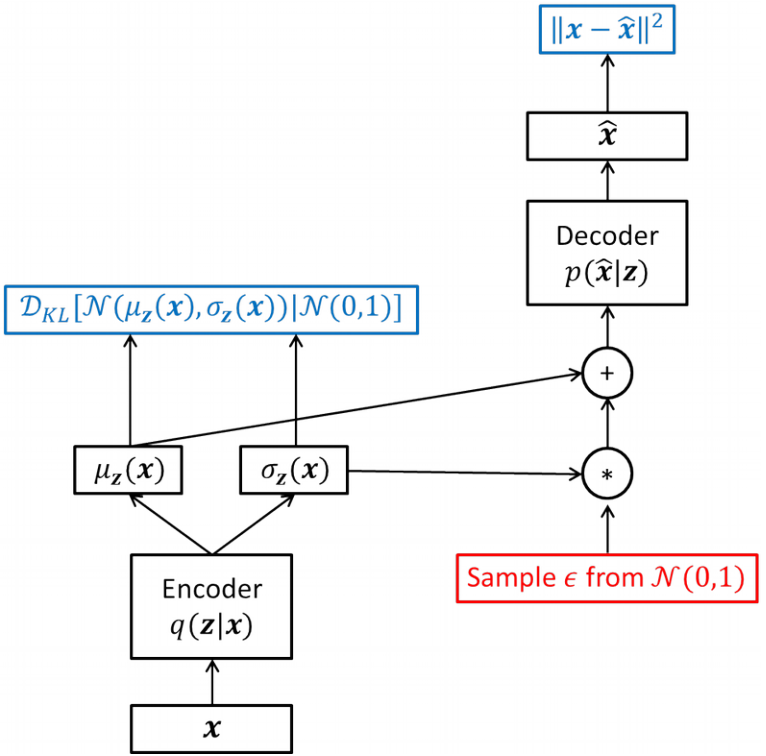


- Learned representation is not an arbitrary function
  - Impose *prior distribution* on the hidden (low dimensional) representation
- Trained decoder part can be used as *generator* (sample from prior distribution)
- Objective function = reconstruction error + divergence of hidden representation from a prior

$$\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{z}) = \mathcal{L}_{recon.}(\mathbf{x}, \hat{\mathbf{x}}) + \mathcal{D}_{KL}[q(\mathbf{z}|\mathbf{x})|p(\mathbf{z})]$$
$$= \frac{1}{N} \sum_{\mathbf{x}} \left[||\mathbf{x} - \hat{\mathbf{x}}||^2 + \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}) \log \frac{p(\mathbf{z})}{q(\mathbf{z}|\mathbf{x})} \right]$$

Kullback-Leibler Divergence: $\mathcal{D}_{KL}$
Measure of information loss if $p(\mathbf{z})$ is used instead of $p(\mathbf{z}|\mathbf{x})$

Variational Autoencoder



Gaussian prior: $z \sim \mathcal{N}(0,1)$

- Encoder $q(\mathbf{z}|\mathbf{x})$ learns latent parameters $\mu_{\mathbf{z}}(\mathbf{x}), \sigma_{\mathbf{z}}(\mathbf{x})$ of Gaussian distribution
- Re-parametrization $z = \mu_{\mathbf{z}}(\mathbf{x}) + \sigma_{\mathbf{z}}(\mathbf{x})\epsilon$ with $\epsilon \in \mathcal{N}(0,1)$
- Example: 2D Gaussian
  - Only two independently normal distributed parameters in hidden layer for each input
- $\mathcal{D}_{KL}[\mathcal{N}(\mu_z(\mathbf{x}), \sigma_z(\mathbf{x}))|\mathcal{N}(0,1)]$

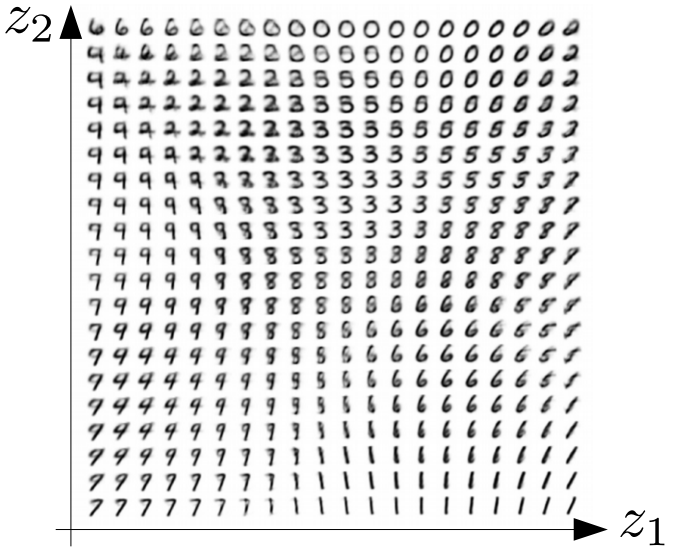
$$= \frac{1}{N} \sum_{\mathbf{x}} \int_{\mathbf{z}} dz\, \mathcal{N}(\mu_{\mathbf{z}}, \sigma_{\mathbf{z}}) \log \frac{\mathcal{N}(0,1)}{\mathcal{N}(\mu_{\mathbf{z}}(\mathbf{x}), \sigma_{\mathbf{z}}(\mathbf{x}))}$$

$$= \frac{1}{N} \sum_{\mathbf{x}} \frac{1}{2}(1 + \log \sigma^2_{\mathbf{z}}(\mathbf{x}) - \mu^2_{\mathbf{z}}(\mathbf{x}) - \sigma^2_{\mathbf{z}}(\mathbf{x}))$$

Variational Autoencoder

Objective: $\mathcal{L}(\mathbf{x}, \hat{\mathbf{x}}, \mathbf{z}) = MSE(\mathbf{x}, \hat{\mathbf{x}}) + \mathcal{D}_{KL}[q(\mathbf{z}|\mathbf{x})|p(\mathbf{z})]$

- Mean-squared-error: → How accurate input can be reconstructed
- KL-divergence: → How close the latent variables match (unit Gaussian)
- Allows walk in latent space



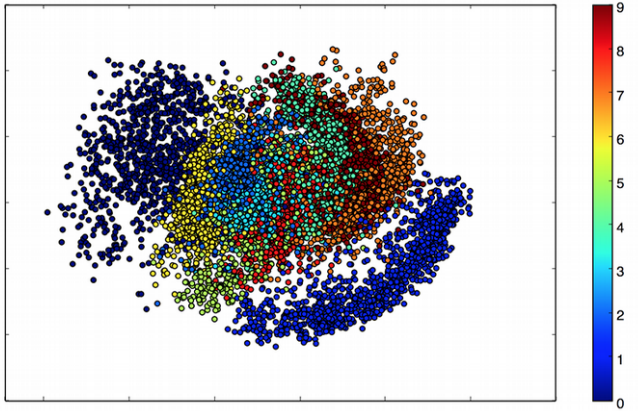
VAE trained on MNIST using 2D Gaussian prior

Improved quality for increased size of latent space

(a) 2-D latent space

(d) 20-D latent space

Variational Autoencoder



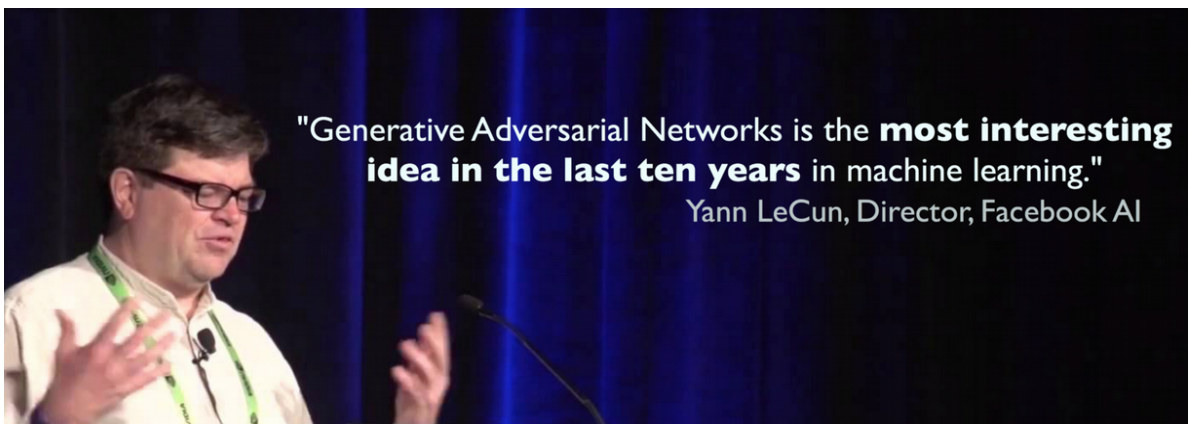
Latent space of VAE trained on MNIST

- Samples of Variational Autoencoders often look noisy
  - Gaussian prior not always best choice / can use arbitrary prior distribution
  - Gaussian distributions can not capture all modes of the data
  - Mean-squared-error loss very inflexible

➢ Try adversarial approach → Only train a generator

Introduction to GANs

- **Generative models**
- Basics of GANs



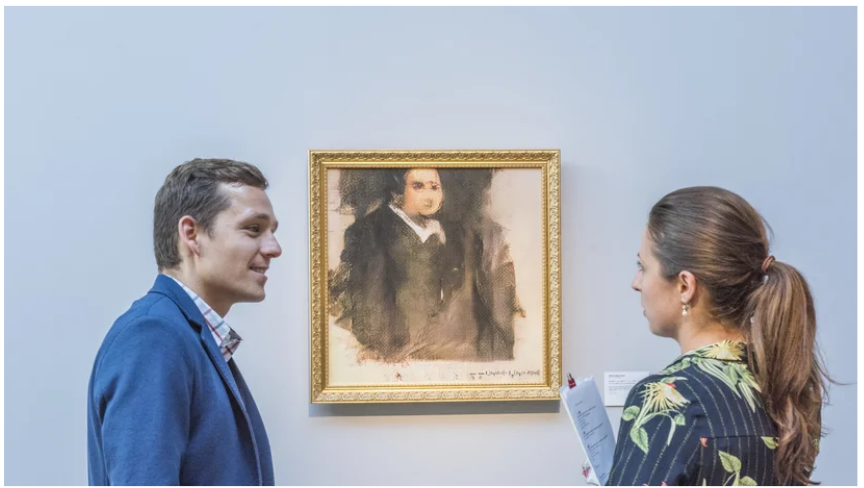
Generative Adversarial Networks - GANs

Künstliche Intelligenz

Auktionshaus versteigert erstmals KI-Gemälde

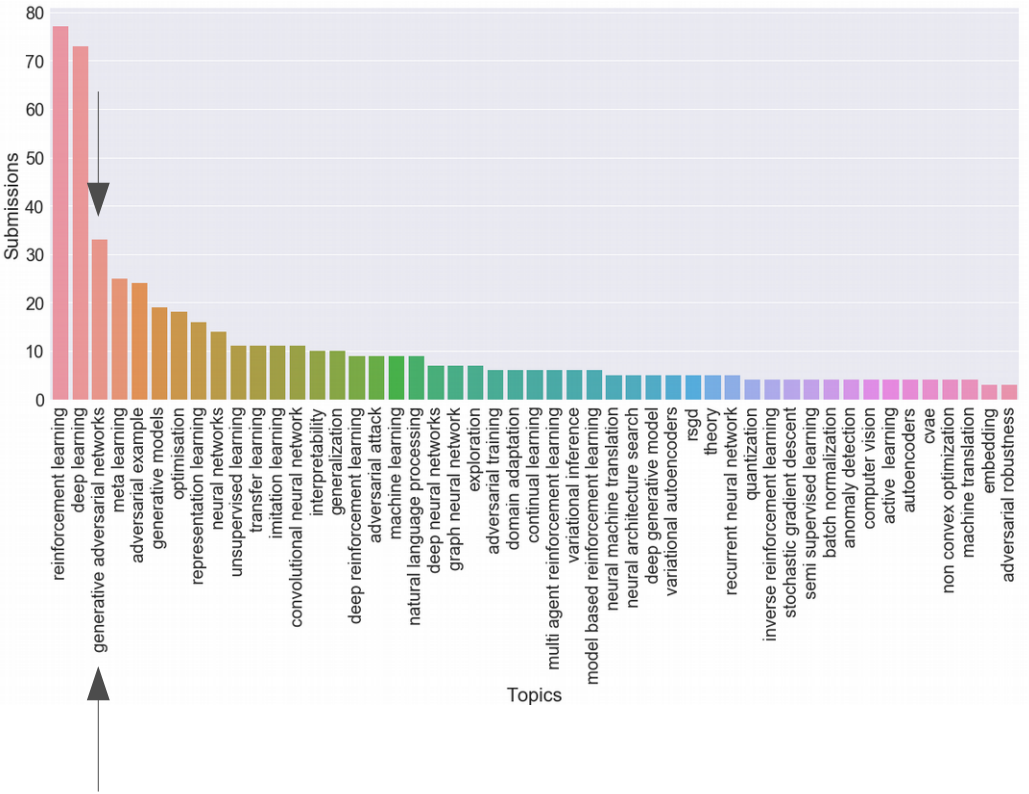
Kein Maler sondern ein Computer-Algorithmus hat das Porträt "Edmond de Belamy" erschaffen. Beim Auktionshaus Christie's zahlte ein Interessent dafür knapp 400.000 Euro.

26. Oktober 2018, 9:00 Uhr / Quelle: ZEIT ONLINE, AFP, fo / 75 Kommentare



Der verschwommene Druck "Edmond de Belamy" zeigt einen Mann in dunkler Kutte mit weißem Kragen, der an einen französischen Geistlichen erinnert. © Christie's/dpa

https://ailab.criteo.com/iclr-2019-stats-trends-and-best-papers/



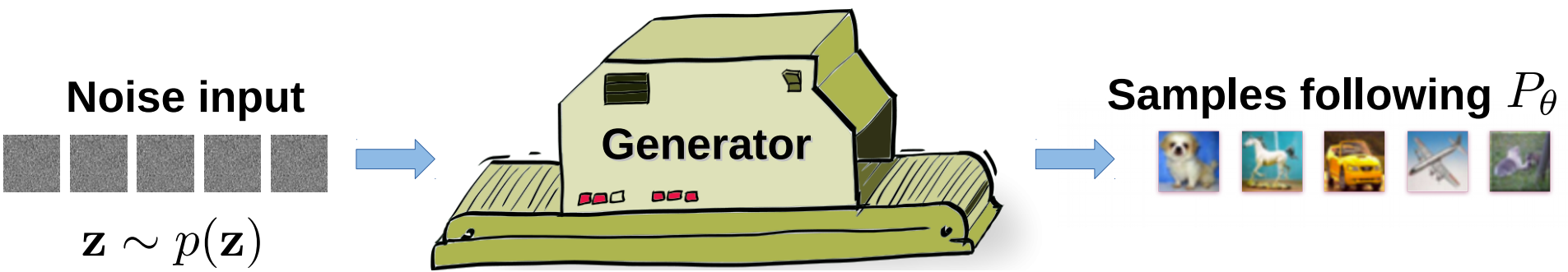
How to train a Generator

I. Objective: learn to generate new samples following P_θ

II. Learn a function that transform a distribution $p(\mathbf{z})$ into P_θ using a generator G_θ

$\mathbf{z} \in Z \rightarrow$ latent space

III. Generator G_θ is implemented as neural network with weights θ



Generative Adversarial Networks

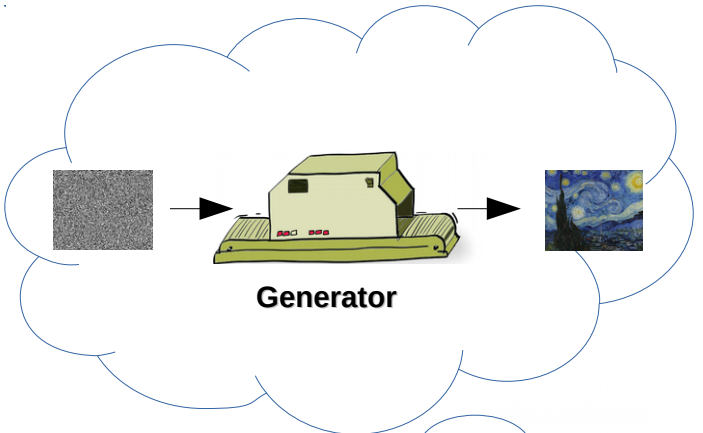
I. Hard to formulate a supervised training loss

II. Use **unsupervised training** to train the generator
- Objective: $P_\theta \approx P_r$
- Measure: given by **second neural network**

→ Generated samples of generator should be similar to real samples after training
- without reproducing training data

→ **Adversarial approach:**

Train 2 networks adversarial (against each other)



Art forger

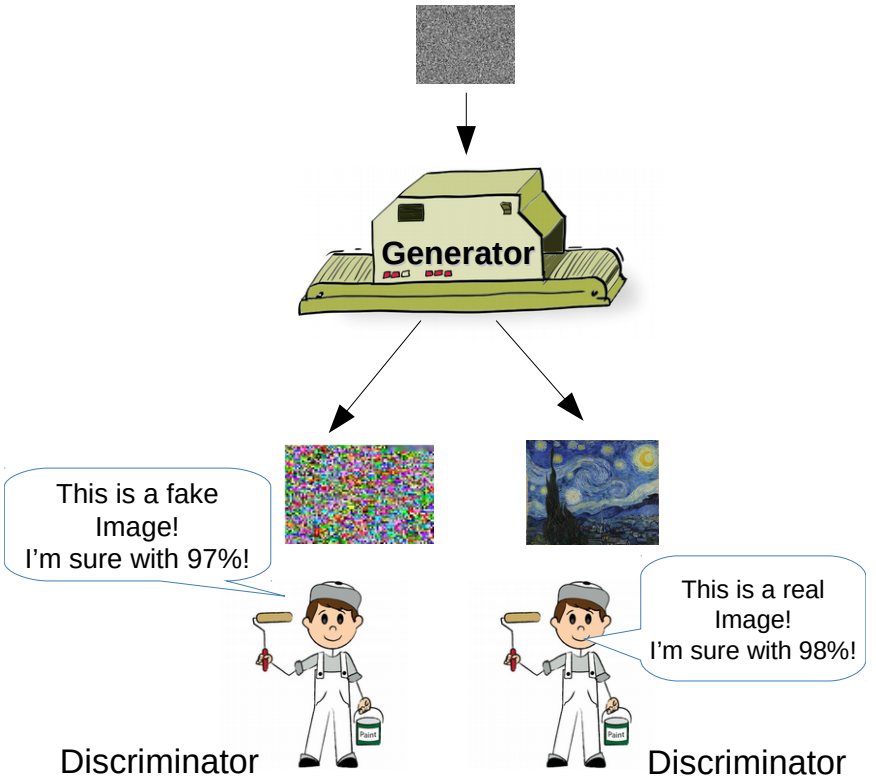
Wants to create some fake images

Generative Adversarial Networks

I. Generator
- Try to generate realistic samples

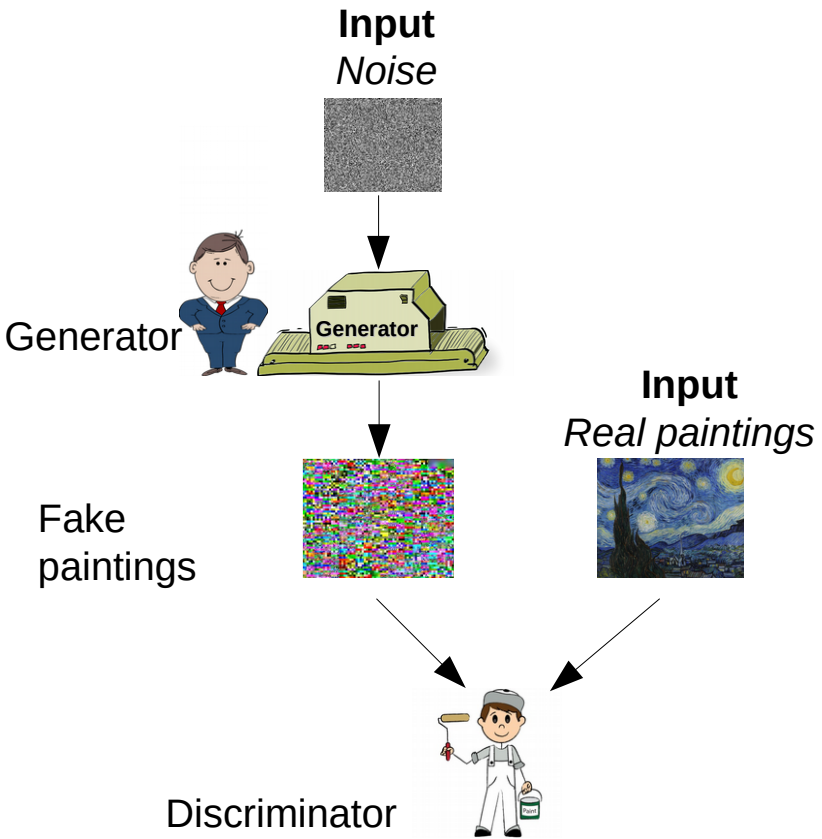
II. Discriminator
- Try to discriminate between fakes and realistic images
- Evaluate if $P_\theta \approx P_r$

III. Discriminator returns probability if generated sample is real

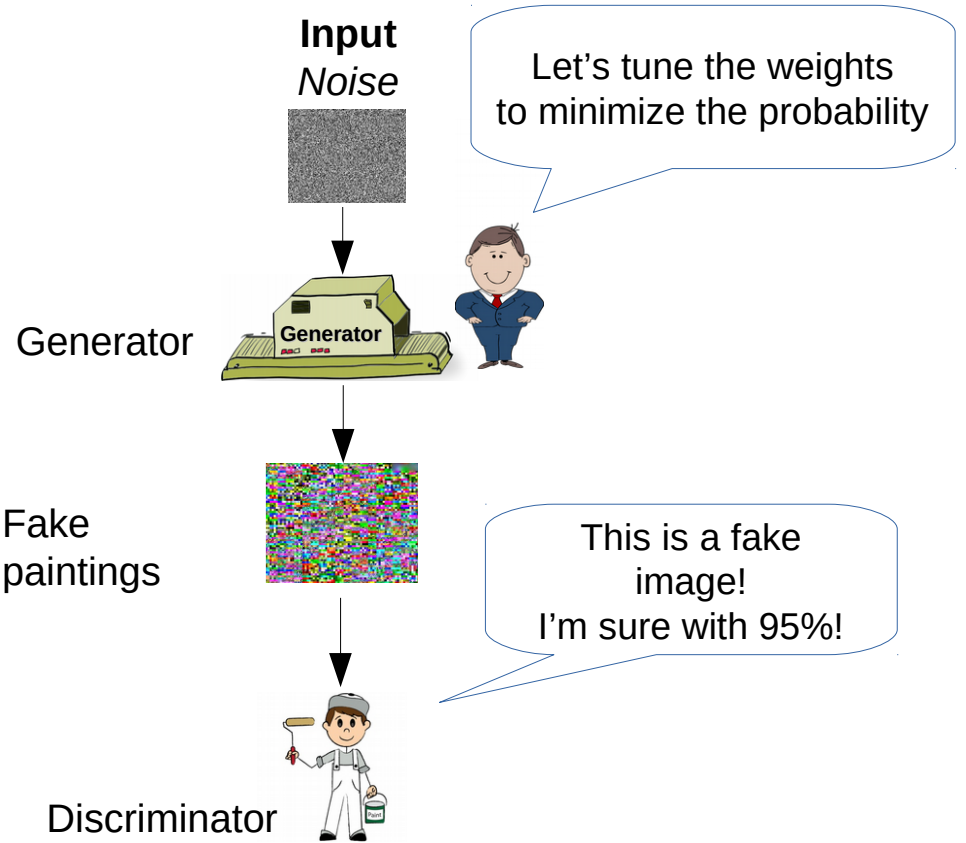


"GANs is the most interesting idea in the last ten years in machine learning." - Y. LeCun

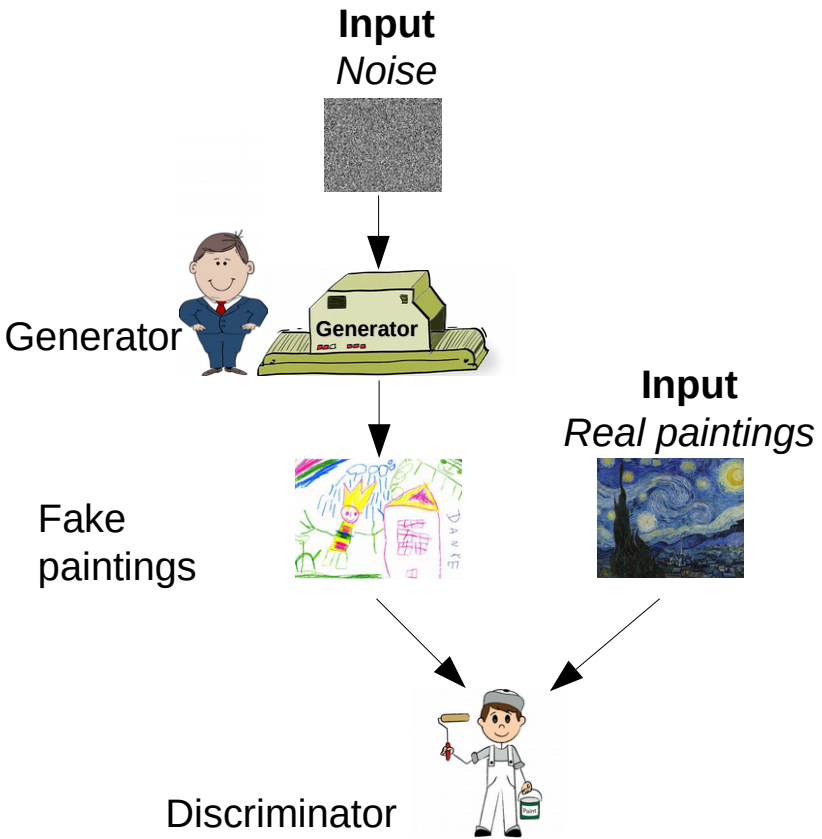
Train Discriminator



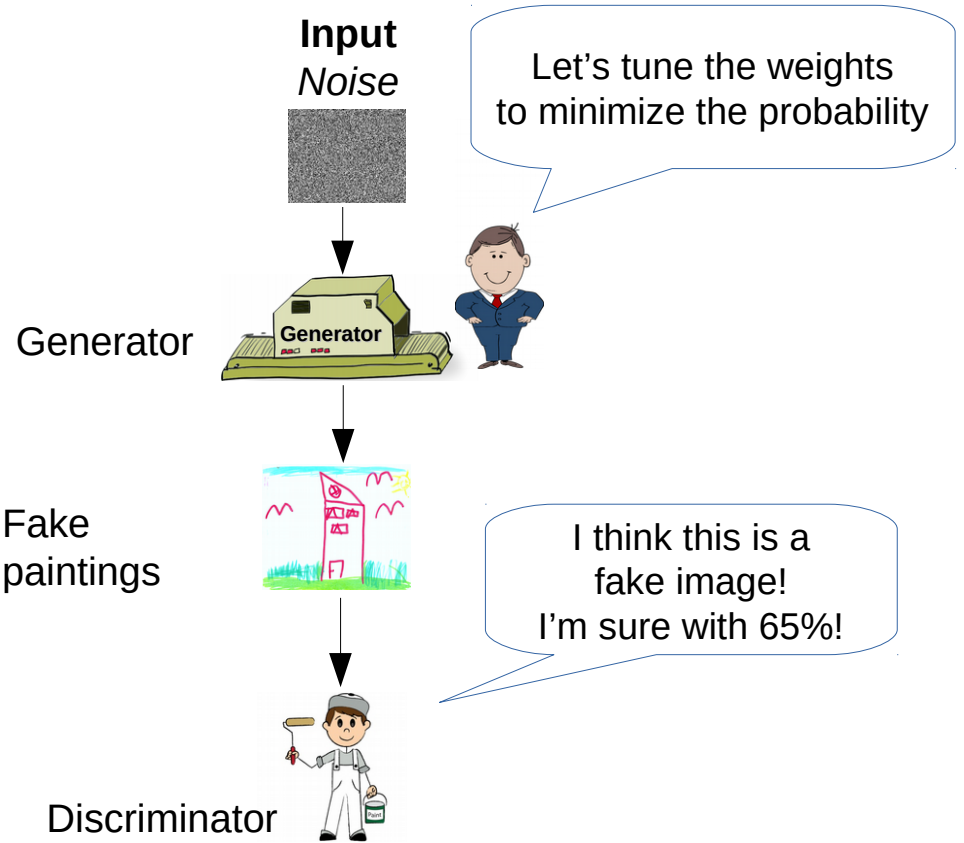
Train Generator



Train Discriminator



Train Generator



Train the Discriminator

I. Implemented as neural network D having weights w

Expectation value

$$Loss = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}[log(D_w(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{z})}[log(1 - D_w(G(\mathbf{z})))]$$

Real samples

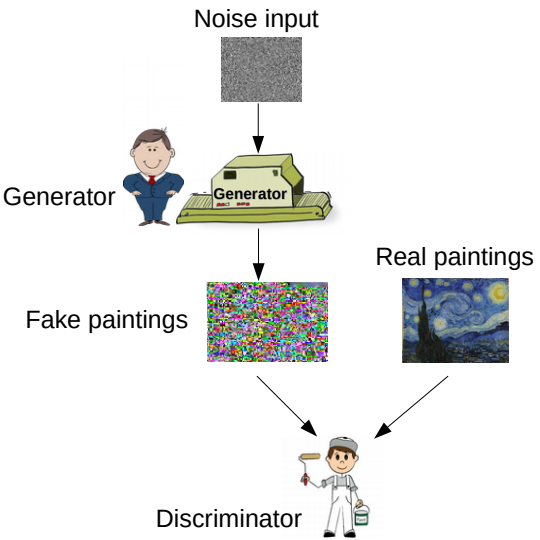
Generated fake samples

II. **Maximize** loss → minimize binary cross entropy

- Tuning discriminator weights

III. Typical classification task for neural network

- Learns to separate two classes



Train the Generator

I. Optimal discriminator is freezed

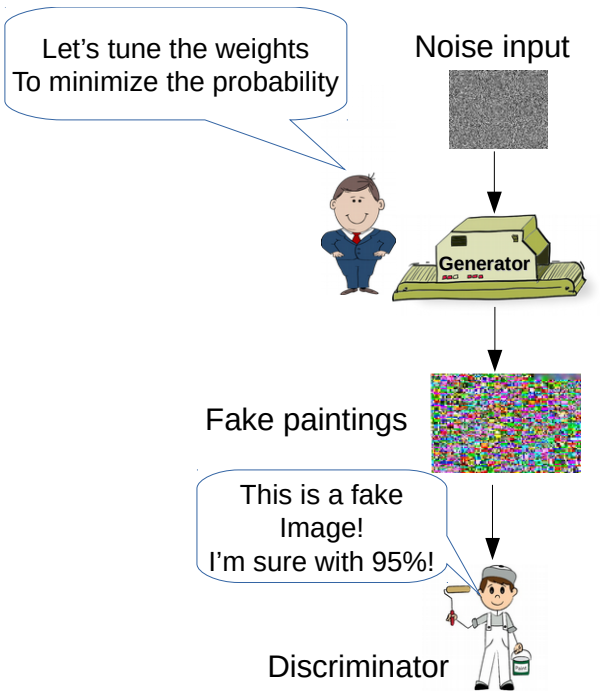
$$Loss = \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})}[log(1 - D(G_\theta(\mathbf{z})))]$$

II. **Minimize** loss: → maximize binary cross entropy

- Tuning the generator weights θ
- Discriminator should fail to discriminate

III. Best case: coin flipping

$$D(G(\mathbf{z})) = \frac{1}{2} \qquad D(\mathbf{x}) = \frac{1}{2}$$

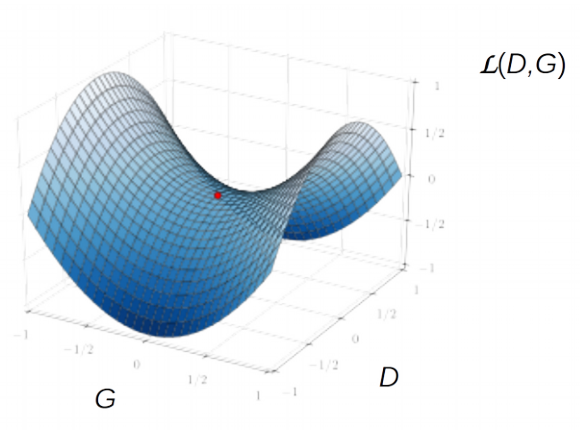


GAN Training

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

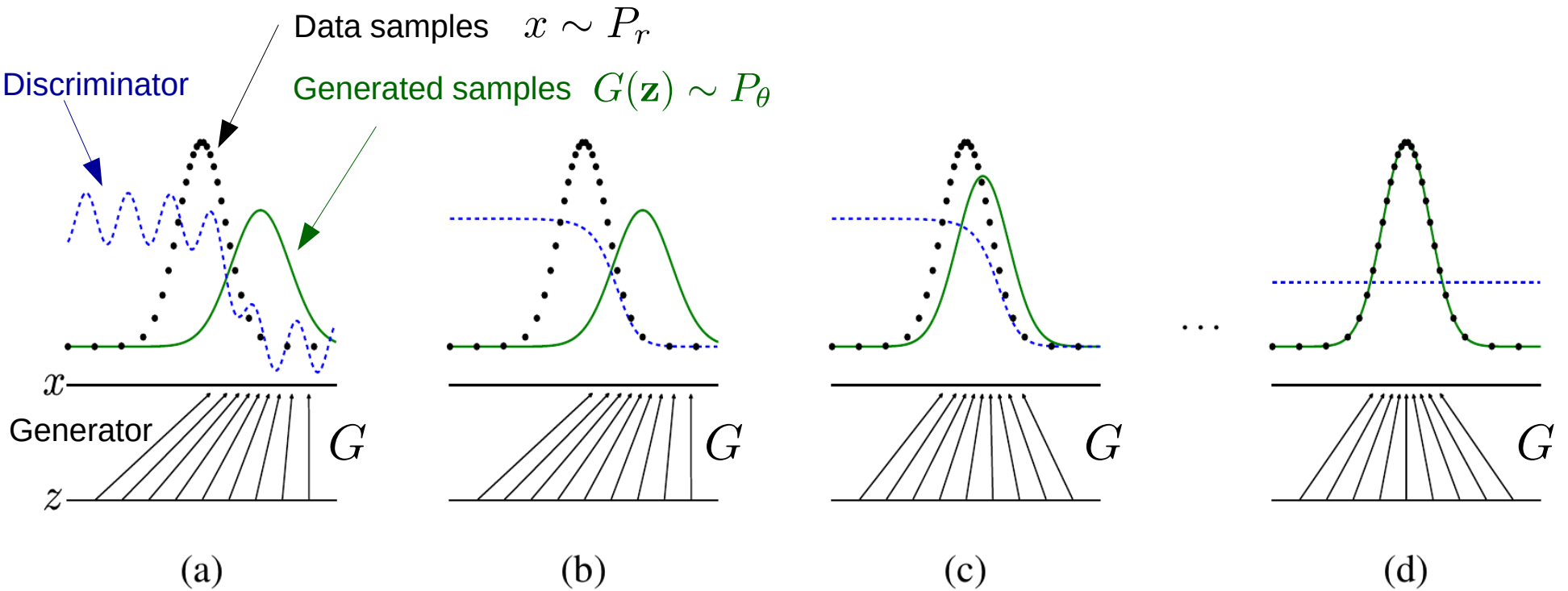
Training 2 networks at the same time is challenging
Losses of discriminator and generator are highly dependent

I. Train generator and discriminator alternating
- Min/Max game
- Sum of both players is zero



II. Finding Nash equilibrium is hard
- Discriminator and generator need to have same quality
- Minimize Jensen-Shannon divergence (assume optimal discriminator)

Optimal Evolution of GAN Training



Goodfellow et al. - arXiv:1406.2661

Gradient of discriminator guides generator
→ G generates samples which are more likely identified as data

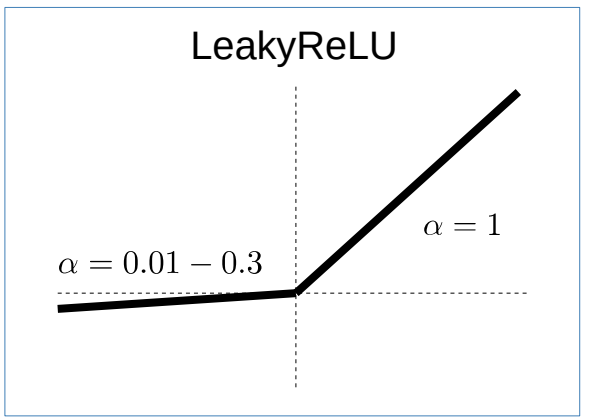
Epochs

Network Design

- Discriminator / classical DCNN for classification
  - Use sigmoid (1 output node) / softmax (2 output nodes) after last layer

- DCGANs (Deep Convolutional GANs) show improved stability

- Use Deep Convolutional generator and discriminator:
  I. Use batch normalization
  II. Remove fully connected hidden layers
  III. Use ReLU in the generator
  IV. Use LeakyReLU in the discriminator
  V. Use transposed convolutions



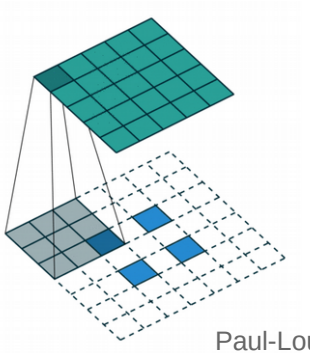
Deep Convolutional GAN (DC-GAN)

I. Topology of the generator:
   - Decrease feature space
   - Increase spatial extent

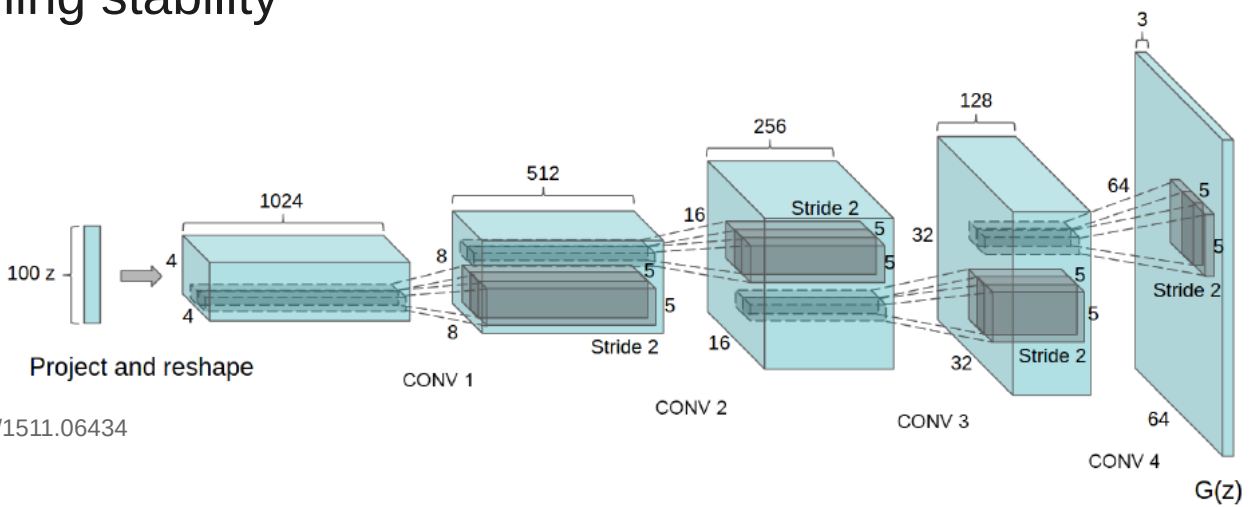
II. Supports a simple structured latent space

III. Use transposed convolutions + striding

➔ Shows improved training stability



Paul-Louis Pröve,
Towards Data Science



A. Radford, L. Metz, S. Chintala - https://arxiv.org/abs/1511.06434

Implementation: Adversarial Training

Generate fake samples $\tilde{x}$ similar to real samples x

I. Train discriminator
   - Sample noise z, feed it into the generator to generate fake samples $G(z) = \tilde{x}$
   - Train discriminator to classify fake $\tilde{x}$ and real samples x

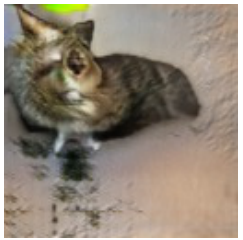
II. Train generator using the discriminator feedback
   - Freeze parameters of discriminator
   - Generate fake samples $G(z) = \tilde{x}$ and pass it to the discriminator
   - Adapt weights of G by fooling the discriminator D

III. Unfreeze parameters of the discriminator

```
# freeze / unfreeze layers in a model
for layer in model.layers:
    layer.trainable = True    # False
# update parameters for a single batch
loss = model.train_on_batch(x, y)
```

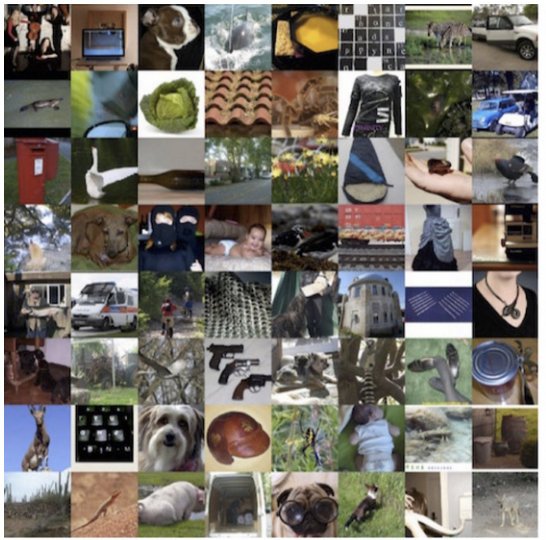
Generative Adversarial Networks

- Wrong global structure

- Wrong body parts


ImageNet


Training Data


Samples

Manifold Hypothesis

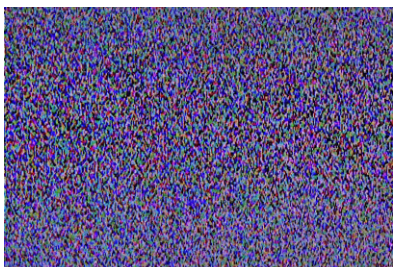
Idea: Manifolds of meaningful pictures are highly concentrated with very little volume and embedded in a very high dimensional space

I. Generation of images is a very challenging task

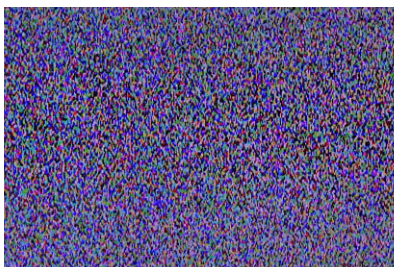
II. Correlations / probability dimension are high dimensional

Example: Try to generate images randomly:

Goal



Sample 1

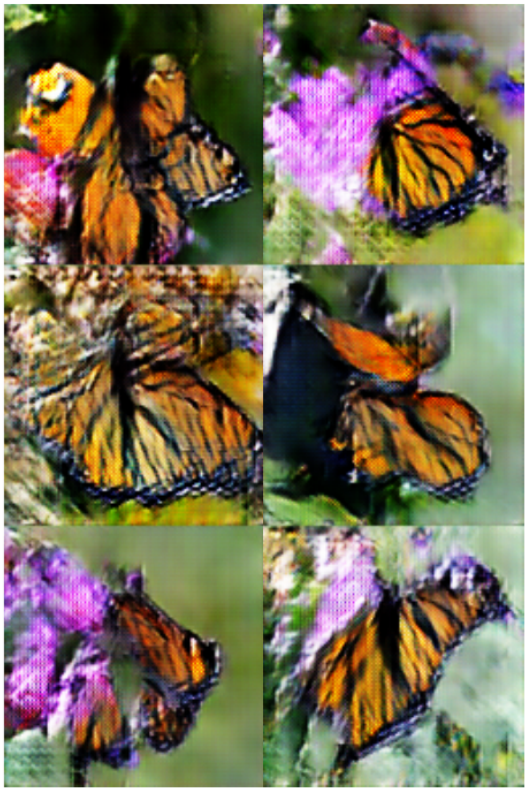


Sample 100,000

You will even never reach this "neighborhood sample"

"To deal with a 14-dimensional space, visualize a 3-D space and say 'fourteen' to yourself very loudly. Everyone does it." - G. Hinton

Evolution of GANs - 2016



monarch butterfly

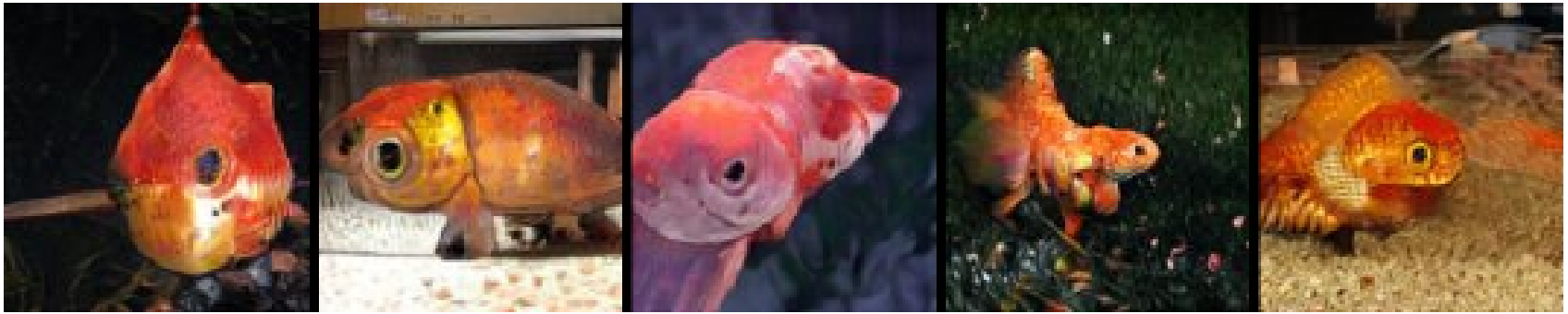
goldfinch

daisy

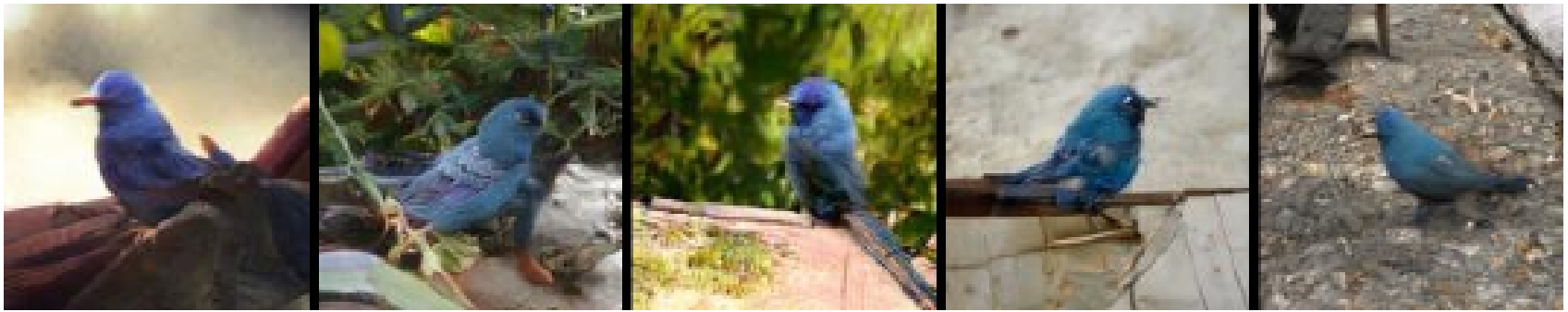
Odena, Olah, Shlens - arXiv:1610.09585

Evolution of GANs

goldfish



bunting



Zhang, Goodfellow, Metaxas, Odena - arXiv:1805.08318

GANs not perfect...

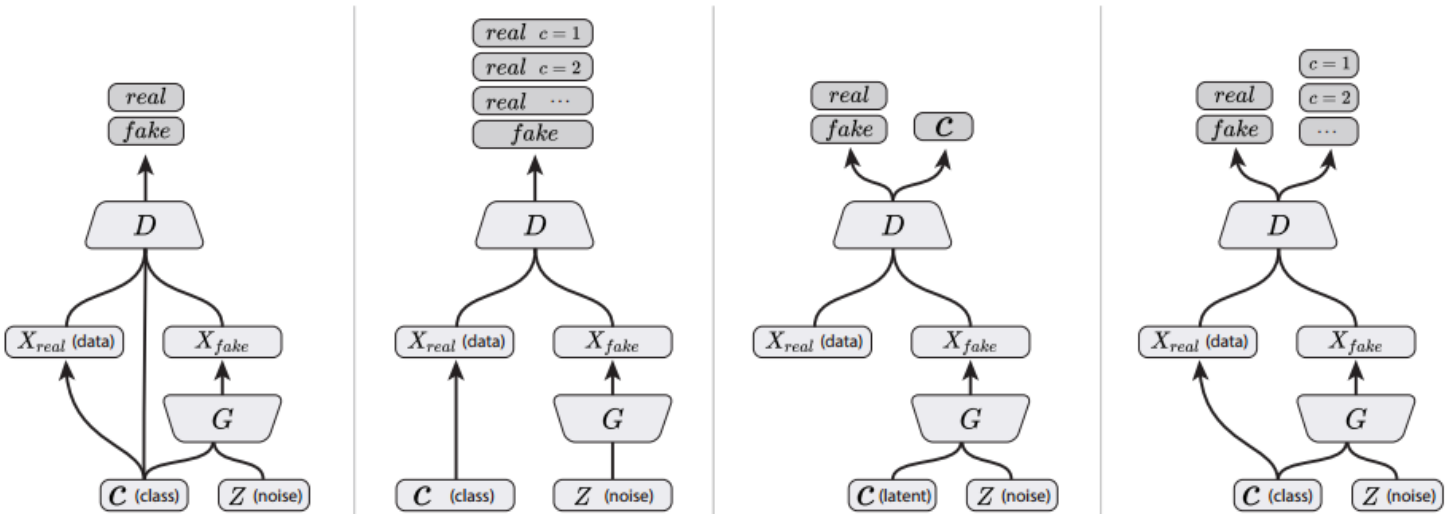
Brock, Donahue, Simonyan - https://arxiv.org/abs/1809.11096

Karras, Alla, Laine, Lehtinen - arXiv:1710.10196

→ Much better images in the next lecture...

Conditioning of GANs – Semi Supervised

- Constrain generator to learn conditional probability distribution
  - Reduce complexity of latent space, allow for interpretations
- ➢ Feed generator and discriminator additional information (e.g. class labels: dog)
  - Force generated samples show specific characteristics (label dependencies)



Conditional GAN
(Mirza & Osindero, 2014)

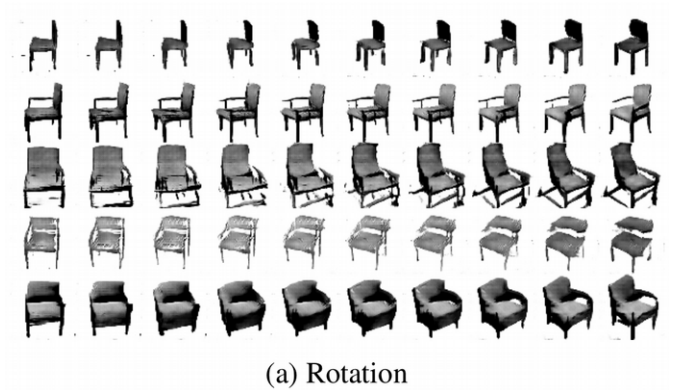
Semi-Supervised GAN
(Odena, 2016; Salimans, et al., 2016)

InfoGAN
(Chen, et al., 2016)

AC-GAN
(Present Work)

Conditioning of GANs

InfoGAN

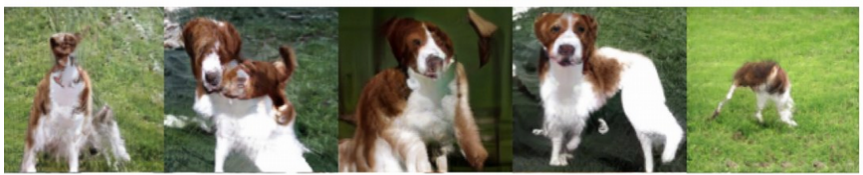


Chen et al. 2016

Conditional image synthesis

A. Odena et al. 2016

T. Miyato et al. 2017

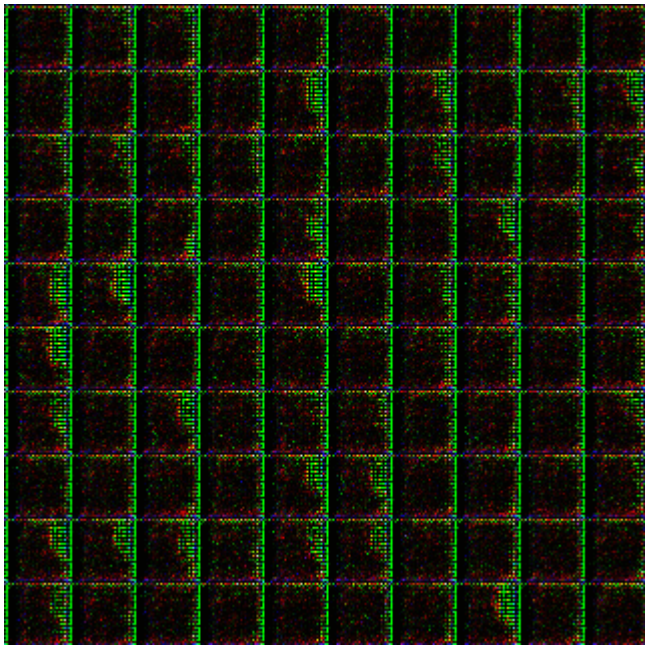


H. Zhang et al. 2018

- Field of generative model is growing very fast

VAE vs GAN

VAE



GAN

https://blog.openai.com/generative-models/

Next Lecture: Advanced Techniques

- Which picture is generated, which picture is part of CELEB A data set?

Summary

- **Generative Models**
  - Generation of new samples using approximation of underlying data distribution
- **Variational Autoencoder**
  - Hidden representation follows low dimensional arbitrary prior distribution
  - Trained decoder part can be used as generator to produce new samples
- **Generative Adversarial Networks**
  - Hand-coded loss is replaced by discriminator (tries to discriminate between fake samples and real samples)
  - Adversarial training: generator and discriminator trained against each other
  - Generator tries to fool discriminator
  - Use conditioning to create prior on latent space

References & Further Reading

- Dr. David Walz, DLiPR, Summerterm 2017
- Kingma, Welling: Variational Autoencoder - https://arxiv.org/abs/1312.6114
- Makzhani et al.: Adversarial Autoencoders - https://arxiv.org/abs/1511.05644
- Goodfellow et al.: Generative Adversarial Networks - https://arxiv.org/abs/1406.2661
- Odena et al.: AC-GAN - https://arxiv.org/abs/1610.09585
- Radford et al.: DCGAN - https://arxiv.org/abs/1511.06434
- Zhang et al.: SAGAN - https://arxiv.org/pdf/1805.08318.pdf

Tutorial

Open jupyter notebooks in google colab:

You can find the repository at:
https://github.com/Napoleongurke/tutorial_generative_models

Open PART I: Vanilla_GAN.ipynb

Open in Colab

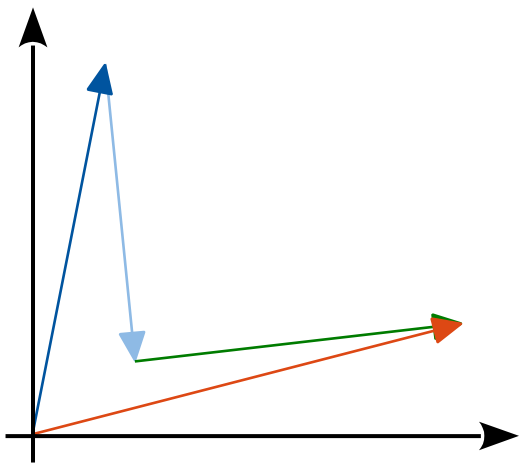
Arithmetic in Latent Space

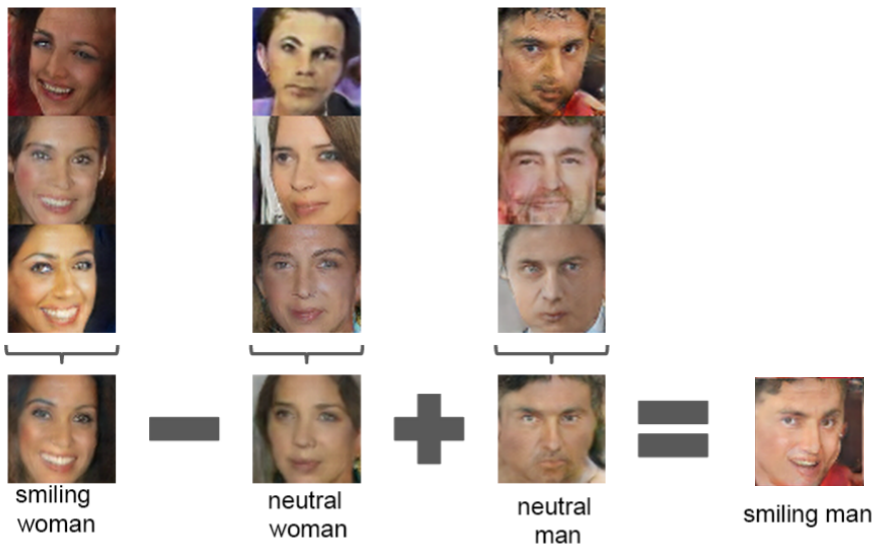
Idea: Discover structure of the latent space

- Can we do arithmetic with latent vectors of generated samples which represent different characteristics?

$$\mathbf{z}_{king} - \mathbf{z}_{man} + \mathbf{z}_{women} = \mathbf{z}_{queen}?$$

- Average over several samples to get representation vector





A. Radford, L. Metz, S. Chintala - https://arxiv.org/abs/1511.06434