Deep Learning

Lecture 12 – Generative Adversarial Networks

Prof. Dr.-Ing. Andreas Geiger

Autonomous Vision Group

University of Tübingen / MPI-IS

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

MAX-PLANCK-GESELLSCHAFT

ellis
European Laboratory for Learning and Intelligent Systems

Agenda

12.1 Generative Adversarial Networks

12.2 GAN Developments

12.3 Research at AVG

12.1
Generative Adversarial Networks

Recap: Latent Variable Models

LVMs map between **observation space** $\mathbf{x} \in \mathbb{R}^D$ and **latent space** $\mathbf{z} \in \mathbb{R}^Q$:

$$(f_{\mathbf{w}} : \mathbf{x} \mapsto \mathbf{z}) \qquad g_{\mathbf{w}} : \mathbf{z} \mapsto \hat{\mathbf{x}}$$

- One **latent variable** gets associated with each data point in the training set
- The latent vectors are smaller than the observations ($Q < D$) $\Rightarrow$ **compression**
- Models are linear or non-linear, deterministic or stochastic, with/without encoder

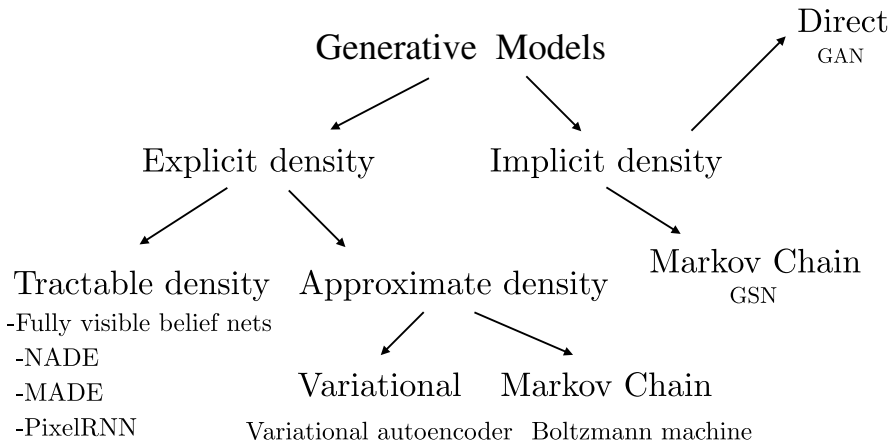
A little taxonomy:

	Deterministic	Probabilistic
Linear	Principle Component Analysis	Probabilistic PCA
Non-Linear w/ Encoder	Autoencoder	Variational Autoencoder
Non-Linear w/o Encoder		Gen. Adv. Networks

Generative Models

- The term **generative model** refers to any model that takes a dataset drawn from p_{data} and learns a probability distribution p_{model} to represent p_{data}
- In some cases, the model estimates p_{model} **explicitly** and therefore allow for evaluating the (approximate) likelihood/density $p_{model}(\mathbf{x})$ of a sample $\mathbf{x}$
- In other cases, the model is only able to **generate samples** from p_{model}
- GANs are prominent examples of this family of **implicit models**
- They provide a framework for training models without explicit likelihood

Generative Models



[Goodfellow: Tutorial on Generative Adversarial Networks, 2017]

Generative Adversarial Networks

Generative Adversarial Networks

- **VAEs** approximate the intractable likelihood using a recognition model
- **GANs** give up on explicitly modeling the density/likelihood
- Instead, they use an adversarial process in which two models (“players”) are trained simultaneously, also referred to as **two-player game:**
  - A **generator** G that captures the data distribution
  - A **discriminator** D that estimates if a sample cam from the data distribution
  - The goal of G is to maximize the probability of D making a mistake – to fool it
- **Backpropagation** can be used to optimize this two-player game
- No approximate inference or slow Markov chains are necessary
- Theoretical results (optimality, convergence) require strong assumptions

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

Generative Adversarial Networks

Let $\mathbf{x} \in \mathbb{R}^D$ denote an observation and $p(\mathbf{z})$ a prior over latent variables $\mathbf{z} \in \mathbb{R}^Q$.
Let $G_{\mathbf{w}_G} : \mathbb{R}^Q \mapsto \mathbb{R}^D$ denote the **generator network** with induced distribution p_{model}.
Let $D_{\mathbf{w}_D} : \mathbb{R}^D \mapsto [0, 1]$ denote the **discriminator network** which outputs a probability.

D and G play the following **two-player minimax game** with value function $V(D, G)$:

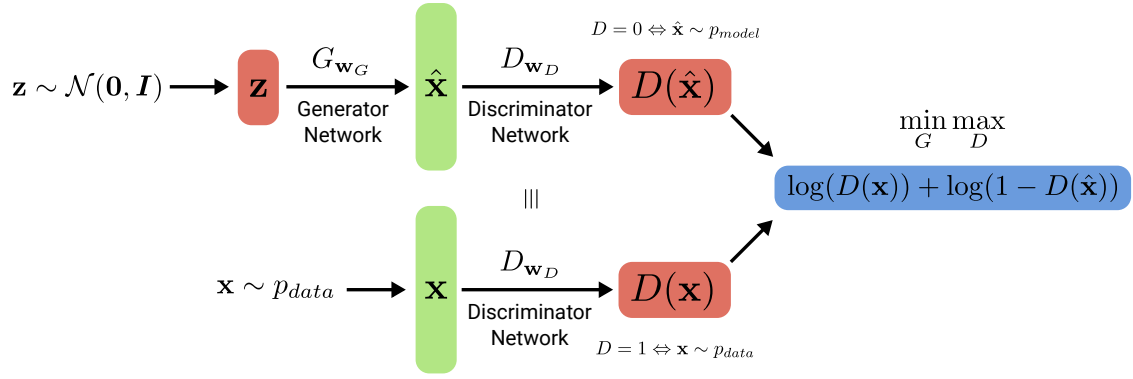
$$G^*, D^* = \operatorname*{argmin}_G \operatorname*{argmax}_D V(D, G)$$

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

We train D to assign probability one to samples from p_{data} and zero to samples from p_{model}, and G to fool D such that it assigns probability one to samples from p_{model}.

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

Generative Adversarial Networks



- The generator and discriminator can be implemented as MLPs, ConvNets, RNNs
- The discriminator network can be considered a **learned loss function** on $\hat{\mathbf{x}}$
- After training, the generator is kept to represent p_{model} and sample from p_{model}

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

Generative Adversarial Networks

- **Theoretical analysis** shows that this minimax game recovers $p_{model} = p_{data}$ if G and D are given enough capacity and assuming that D^* can be reached
- In practice, however, we must use iterative numerical optimization and optimizing D in the inner loop to completion is computationally prohibitive and would lead to overfitting on finite datasets
- Therefore, we resort to **alternating optimization:**
  - k steps of optimizing D (typically $k \in \{1, \ldots, 5\}$)
  - 1 step of optimizing G (using a small enough learning rate)
- This way, we maintain D near its optimal solution as long as G changes slowly

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

Algorithm

While not converged **do**

1. **For** k steps **do**
   - 1.1 Draw B training samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_B\}$ from $p_{data}(\mathbf{x})$
   - 1.2 Draw B latent samples $\{\mathbf{z}_1, \ldots, \mathbf{z}_B\}$ from $p(\mathbf{z})$
   - 1.3 Update the **discriminator** D by **ascending** its stochastic gradient:

$$\nabla_{\mathbf{w}_D} \frac{1}{B} \sum_{b=1}^{B} \log D(\mathbf{x}_b) + \log(1 - D(G(\mathbf{z}_b)))$$

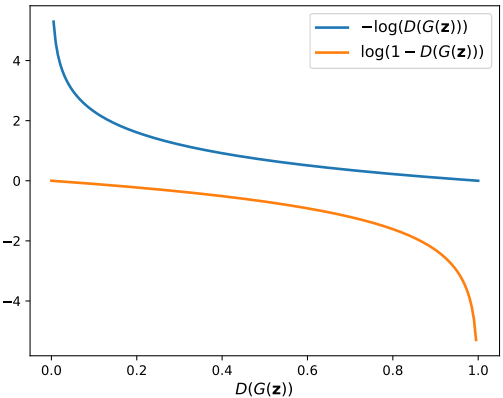
2. Draw B latent samples $\{\mathbf{z}_1, \ldots, \mathbf{z}_B\}$ from $p(\mathbf{z})$
3. Update the **generator** G by **descending** its stochastic gradient:

$$\nabla_{\mathbf{w}_G} \frac{1}{B} \sum_{b=1}^{B} \log(1 - D(G(\mathbf{z}_b)))$$

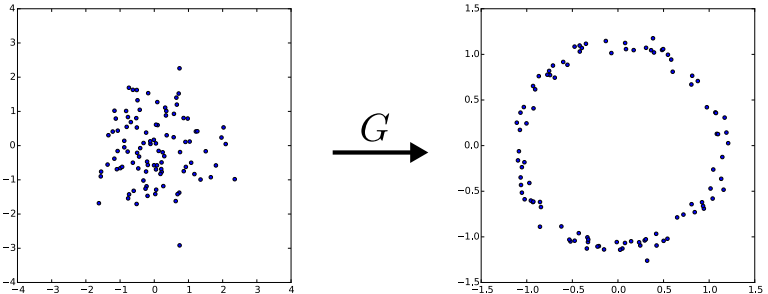
Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

The Gradient Trick

- Early in training, when G is poor, D rejects samples with high confidence
- Thus $\log(1 - D(G(\mathbf{z}))$ **saturates**
- Instead of training G to **minimize** $\log(1 - D(G(\mathbf{z})))$ we can train G to **maximize** $\log(D(G(\mathbf{z})))$
- This results in the same fixed point but provides **stronger gradients** early on during training



Expressiveness



- Similar to a VAE decoder, the generator in GANs is very **expressive**
- Consider random samples $p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ mapped via $G_{\mathbf{w}_G}(\mathbf{z}) = \mathbf{z}/10 + \mathbf{z}/\|\mathbf{z}\|$
- The generator $G_{\mathbf{w}_G}(\mathbf{z})$ is a **neural network** with learned parameters $\mathbf{w}_G$

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

1D Example

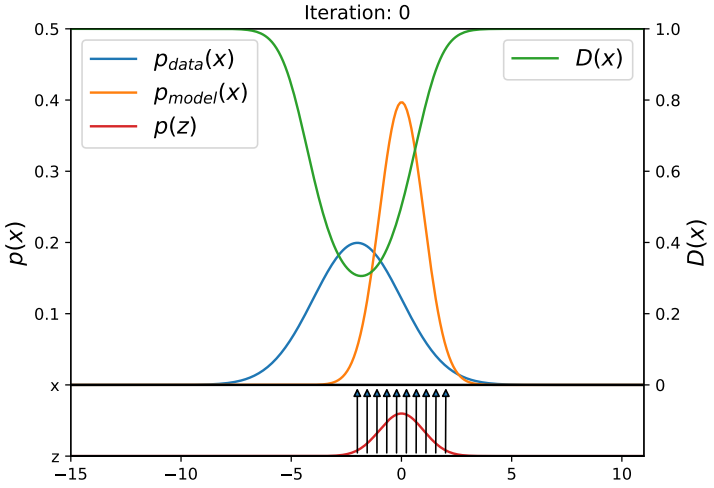
A simple example with linear generator:

$$
\begin{aligned}
x &\sim \mathcal{N}(\mu, \sigma) \\
z &\sim \mathcal{N}(0, 1) \\
G(z) &= w_0^G + w_1^G z \\
D(x) &= \sigma(w_0^D + w_1^D x + w_2^D x^2)
\end{aligned}
$$

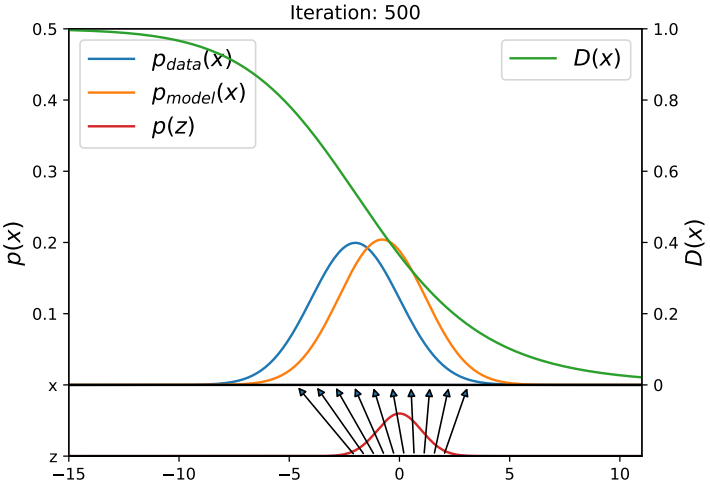
- Here, we consider the data distribution and the prior as two different 1D Gaussians and initialize $G(z) = z$ and $D(x) = \sigma(x)$, i.e., $p_{model}(x) = \mathcal{N}(x|0, 1)$
- The goal is to learn $\mathbf{w}_G$ and $\mathbf{w}_D$ such that $p_{model}(x) = p_{data}(x) = \mathcal{N}(x|\mu, \sigma)$
- Remark: The x^2 term is needed to provide gradients for the second moment

Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

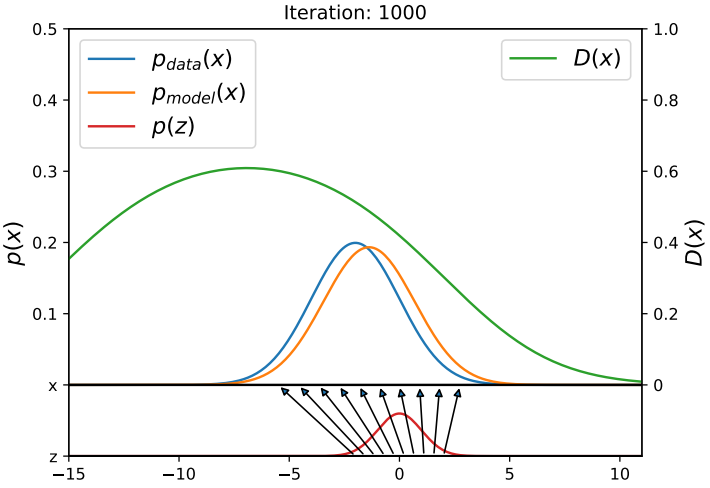
1D Example


Iteration: 0
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$p(x)$
$D(x)$
0.5
0.4
0.3
0.2
0.1
x
z
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example

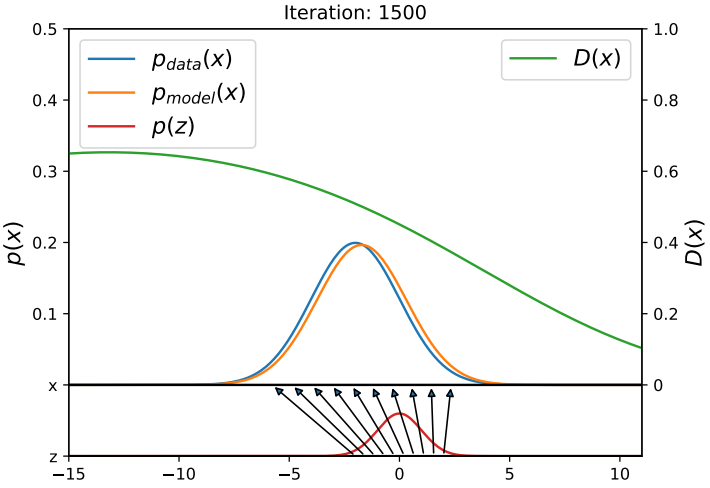

Iteration: 500
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$p(x)$
$D(x)$
0.5
0.4
0.3
0.2
0.1
x
z
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example

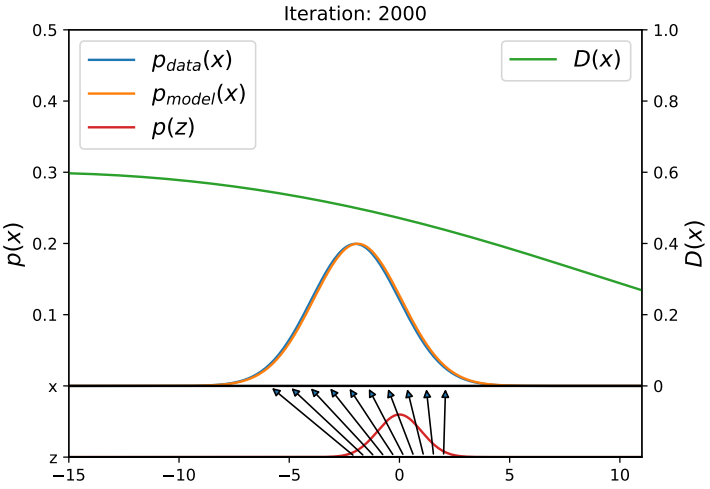


Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

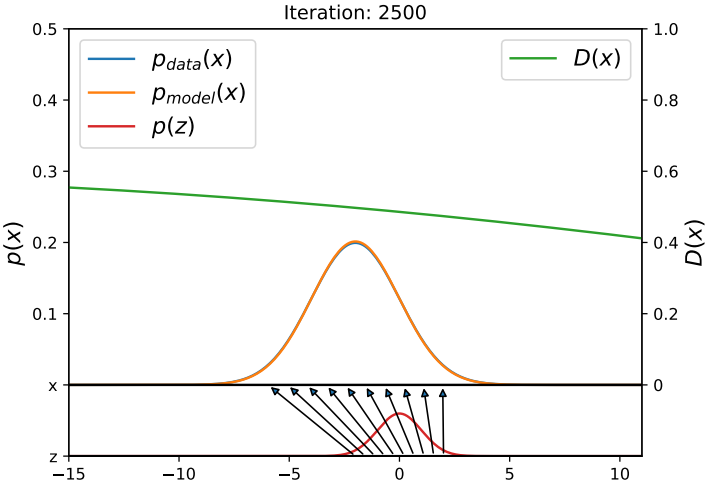
1D Example


Iteration: 1500
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$p(x)$
$D(x)$
x
z
0.5
0.4
0.3
0.2
0.1
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example

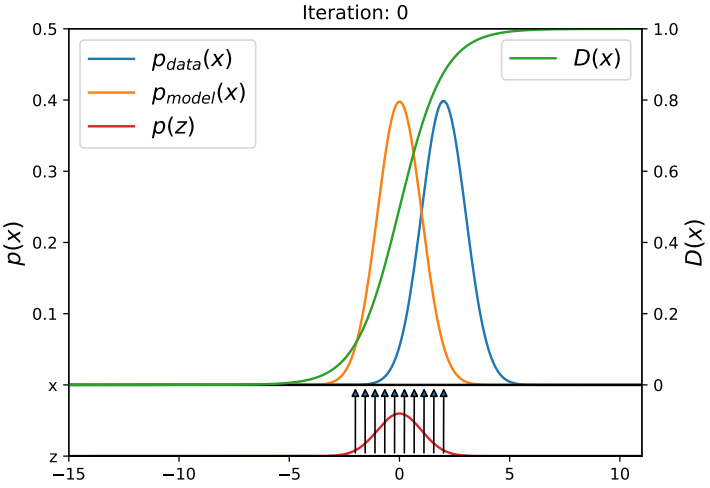

Iteration: 2000
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$p(x)$
$D(x)$
0.5
0.4
0.3
0.2
0.1
x
z
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example



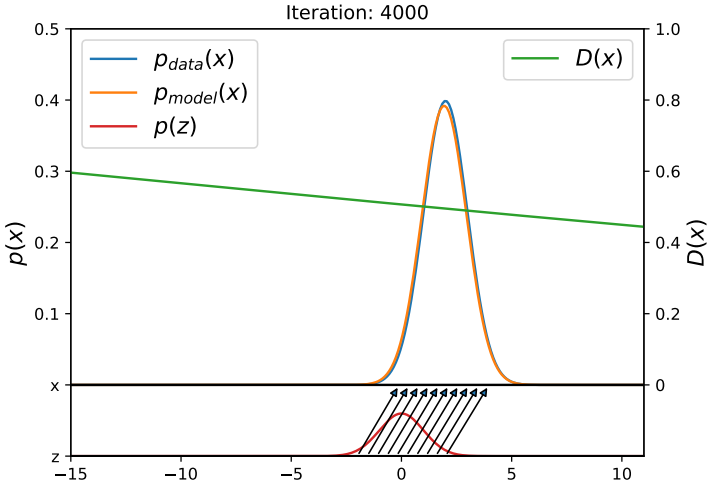
Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

1D Example



Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

1D Example



Goodfellow, Pouget-Abadie, Mirza, Xu, Warde-Farley, Ozair, Courville, Bengio: Generative Adversarial Networks. NIPS, 2014.

Theoretical Results

Generative Adversarial Networks

Let $\mathbf{x} \in \mathbb{R}^D$ denote an observation and $p(\mathbf{z})$ a prior over latent variables $\mathbf{z} \in \mathbb{R}^Q$.
Let $G_{\mathbf{w}_G} : \mathbb{R}^Q \mapsto \mathbb{R}^D$ denote the **generator network** with induced distribution p_{model}.
Let $D_{\mathbf{w}_D} : \mathbb{R}^D \mapsto [0, 1]$ denote the **discriminator network** which outputs a probability.

D and G play the following **two-player minimax game** with value function $V(D, G)$:

$$G^*, V^* = \underset{G}{\operatorname{argmin}} \ \underset{D}{\operatorname{argmax}} \ V(D, G)$$

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

We train D to assign probability one to samples from p_{data} and zero to samples from p_{model}, and G to fool D such that it assigns probability one to samples from p_{model}.

Optimal Discriminator

Proposition 1. For any given generator G, the optimal discriminator D is:

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

Proof. The training criterion for the discriminator D is to maximize (wrt. D):

$$\begin{aligned} V(D,G) &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x}))\, d\mathbf{x} + \int_{\mathbf{z}} p(\mathbf{z}) \log(1 - D(G(\mathbf{z})))\, d\mathbf{z} \\ &= \int_{\mathbf{x}} p_{data}(\mathbf{x}) \log(D(\mathbf{x})) + p_{model}(\mathbf{x}) \log(1 - D(\mathbf{x}))\, d\mathbf{x} \end{aligned}$$

For any $(a, b) \in \mathbb{R}^2 \setminus \{0, 0\}$, the function $y \mapsto a \log(y) + b \log(1 - y)$ achieves its maximum in $[0, 1]$ at $\frac{a}{a+b}$. The discriminator D does not need to be defined outside $Supp(p_{data}) \cup Supp(p_{model})$ where $p_{data} = 0$ and $p_{model} = 0$, concluding the proof. ■

Global Optimality

Theorem 1. The global minimum of the virtual training criterion

$$
\begin{aligned}
V(D_G^*, G) &= \mathbb{E}_{\mathbf{x}\sim p_{data}} \left[\log D_G^*(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_{model}} \left[\log(1 - D_G^*(\mathbf{x}))\right] \\
&= \mathbb{E}_{\mathbf{x}\sim p_{data}} \log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) \; + \; p_{model}(\mathbf{x})} + \mathbb{E}_{\mathbf{x}\sim p_{model}} \log \frac{p_{model}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_{model}(\mathbf{x})}
\end{aligned}
$$

is achieved for $p_{model} = p_{data}$ where $D_G^* = \frac{1}{2}$ and $V(D_G^*, G) = -\log 4 \approx -1.386$.

Proof. Reformulation in terms of the non-negative Jensen-Shannon divergence yields:

$$
\begin{aligned}
V(D_G^*, G) &= KL(p_{data}, p_{data} + p_{model}) + KL(p_{model}, p_{data} + p_{model}) \\
&= -\log 4 + KL\left(p_{data}, \frac{p_{data} + p_{model}}{2}\right) + KL\left(p_{model}, \frac{p_{data} + p_{model}}{2}\right) \\
&= -\log 4 + JSD(p_{data}, p_{model})
\end{aligned}
$$

■

Convergence

Proposition 2. If G and D have enough capacity, and at each update step the discriminator D is allowed to reach $D = D^*_G$, and p_{model} is updated to improve

$$V(D^*_G, p_{model}) = \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[\log D^*_G(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_{model}}\left[\log(1 - D^*_G(\mathbf{x}))\right]$$
$$\propto \sup_D \int_{\mathbf{x}} p_{model}(\mathbf{x}) \log(1 - D(\mathbf{x}))\, d\mathbf{x}$$

then p_{model} converges to p_{data}.

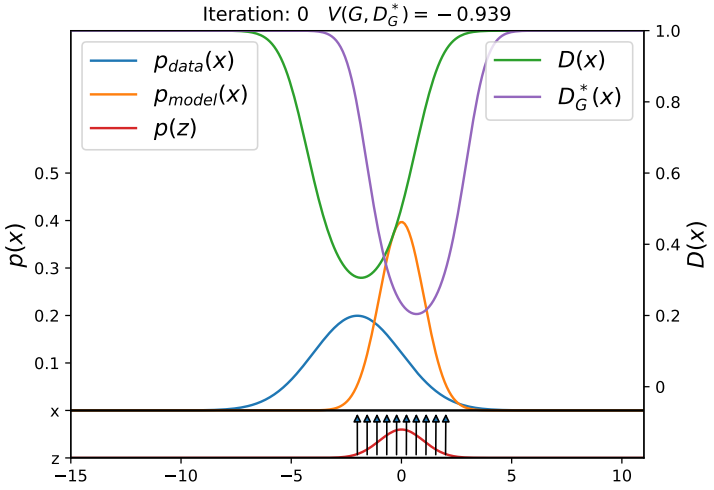
Proof. The argument of the supremum is convex in p_{model}. The supremum doesn't change convexity, thus $V(D^*_G, p_{model})$ is also convex in p_{model} with global optimum $p_{model} = p_{data}$ as shown in Theorem 1. ■

Assumptions

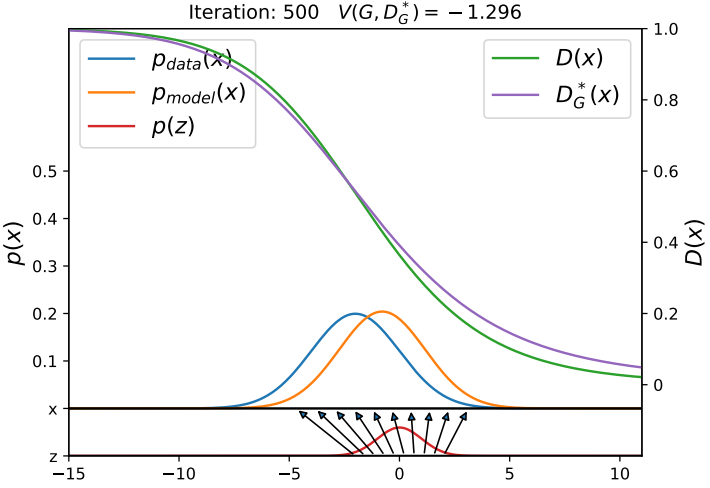
Remarks on Assumptions:

- The theoretical results above make very strong assumptions:
  - The generator G and discriminator D have enough **capacity**
  - The discriminator D reaches its **optimum** D_G^* at every outer iteration
  - We **directly optimize** p_{model} instead of its parameters $\mathbf{w}$
- In practice, G and D have **finite capacity,** D is optimized for only k **steps,** and using a neural network to define G introduces **critical points** in parameter space.
- Thus, in practice, GANs often do not converge to p_{data} and might oscillate
- However, neural networks work well in practice, and balancing the updates of G and D keeps D close to D_G^* in order to backpropagate meaningful gradients to G
- See also: `https://srome.github.io/An-Annotated-Proof`

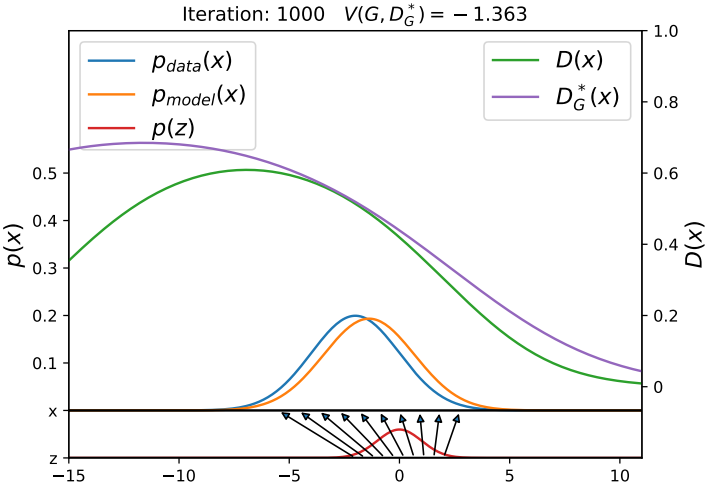
1D Example



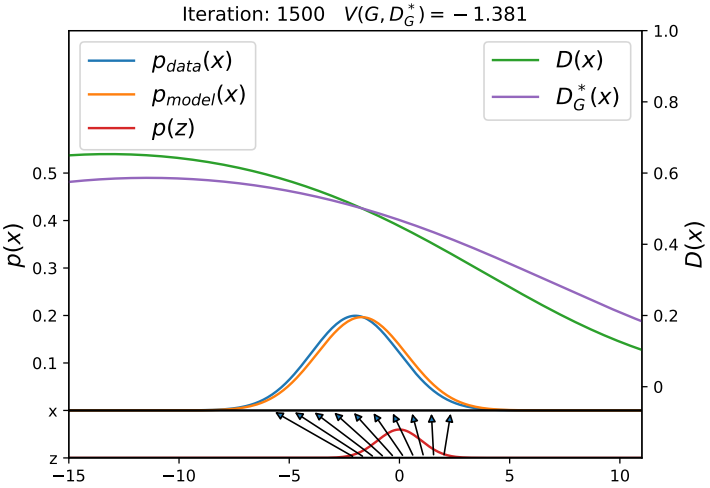
1D Example


Iteration: 500 $V(G, D_G^*) = -1.296$
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$D_G^*(x)$
$p(x)$
$D(x)$
x
z
−15
−10
−5
0
5
10
0.1
0.2
0.3
0.4
0.5
0.6
0.8
1.0

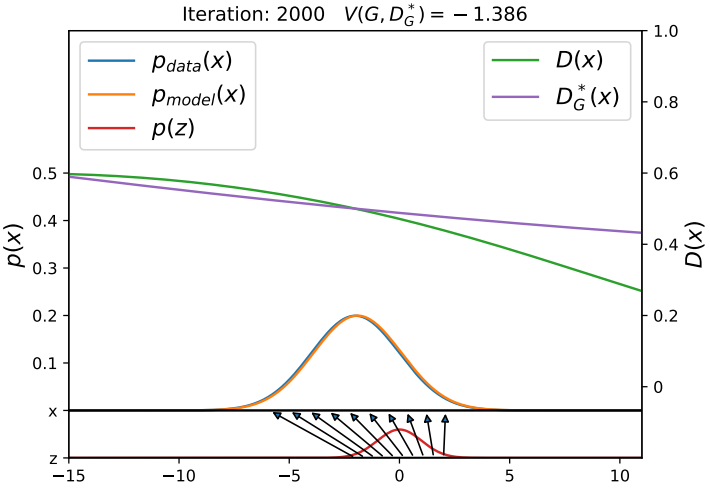
1D Example



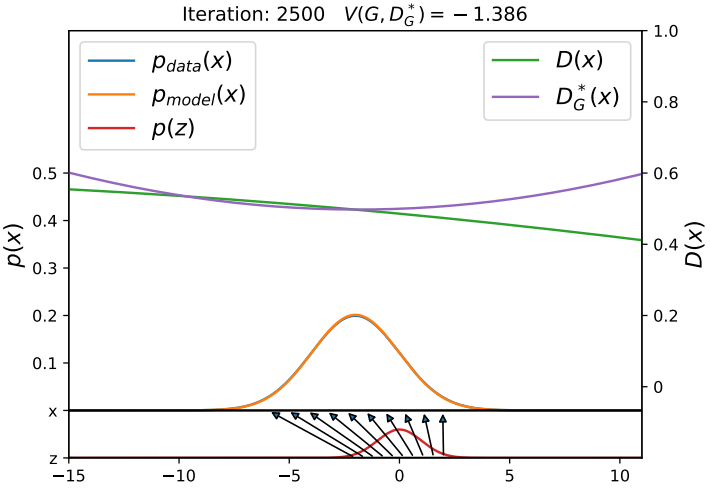
1D Example


Iteration: 1500 $V(G, D_G^*) = -1.381$
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$D_G^*(x)$
$p(x)$
$D(x)$
x
z
0.5
0.4
0.3
0.2
0.1
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example

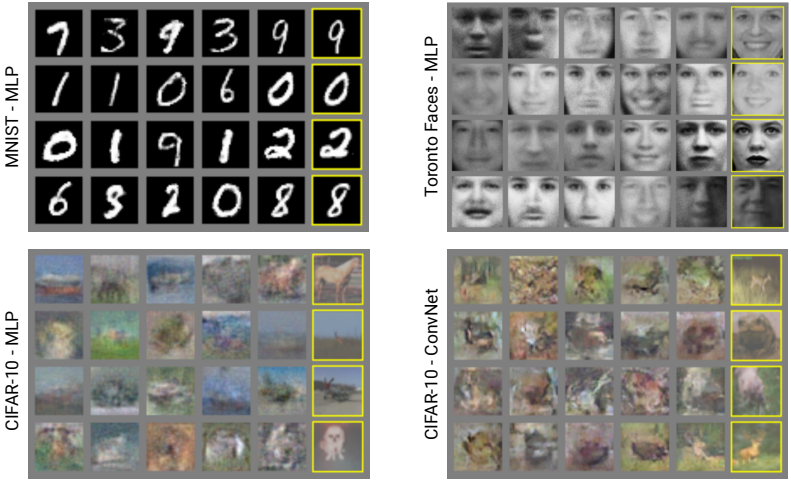

Iteration: 2000 $V(G, D_G^*) = -1.386$
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$D_G^*(x)$
$p(x)$
$D(x)$
0.5
0.4
0.3
0.2
0.1
x
z
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

1D Example


Iteration: 2500 $V(G, D_G^*) = -1.386$
$p_{data}(x)$
$p_{model}(x)$
$p(z)$
$D(x)$
$D_G^*(x)$
$p(x)$
$D(x)$
0.5
0.4
0.3
0.2
0.1
x
z
1.0
0.8
0.6
0.4
0.2
0
−15
−10
−5
0
5
10

Empirical Results

Visualization of Samples from the Model



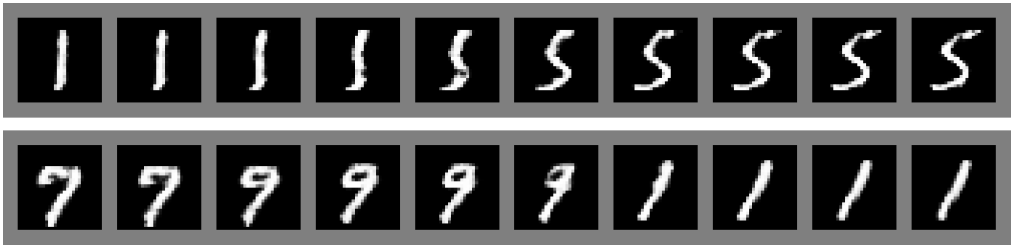
- The right column shows the training example nearest to the neighboring sample

Likelihood Estimates

Model	MNIST	TFD
DBN [3]	138 ± 2	1909 ± 66
Stacked CAE [3]	121 ± 1.6	$\mathbf{2110 \pm 50}$
Deep GSN [6]	214 ± 1.1	1890 ± 29
Adversarial nets	$\mathbf{225 \pm 2}$	$\mathbf{2057 \pm 26}$

- For GANs as for VAEs, the likelihood of test samples cannot be computed
- However, a rough performance estimate can be obtained by fitting a Gaussian **Parzen window** to the samples generated with G and reporting the log-likelihood

Latent Space Interpolations

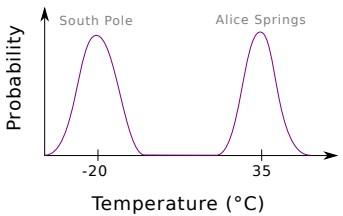


- Digits obtained by linearly interpolating between coordinates in latent space

Mode Collapse

One major failure mode of GANs is **mode collapse,** where the generator learns to produce high-quality sample with very low variability, covering only a fraction of p_{data}:

1. The generator learns to fool the discriminator by producing values close to Antarctic temperatures
2. The discriminator can't distinguish Antarctic temperatures but learns that all Australian temperatures are real
3. The generator learns that it should produce Australian temperatures and abandons the Antarctic mode
4. The discriminator can't distinguish Australian temperatures but learns that all Antarctic temperatures are real
5. Repeat



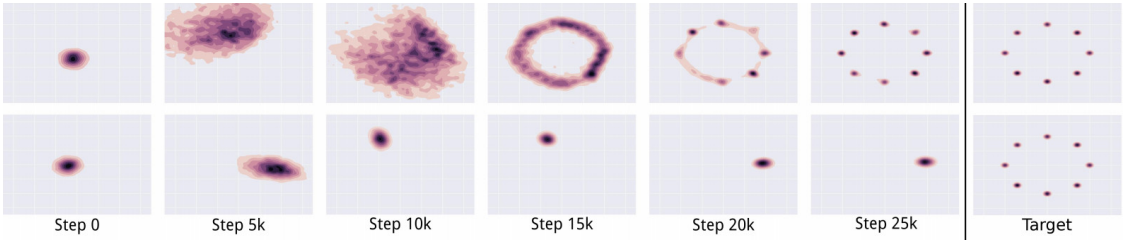
Mode Collapse

Strategies for avoiding mode collapse:

- **Encourage diversity:** Minibatch discrimination allows the discriminator to compare samples across a batch to determine whether the batch is real or fake.
- **Anticipate counterplay:** Look into the future, e.g., via unrolling the discriminator, and anticipate counterplay when updating generator parameters
- **Experience replay:** Hopping back and forth between modes can be minimised by showing old fake samples to the discriminator once in a while
- **Multiple GANs:** Train multiple GANs and hope that they cover all modes.
- **Optimization Objective:** Wasserstein GANs, Gradient penalties, ...

`https://aiden.nibali.org/blog/2017-01-18-mode-collapse-gans/`

Unrolled Generative Adversarial Networks



- **Top: Unrolled GAN with 10 unrolling steps.** Note that unrolled GANs requires backpropagating the generator gradient through unrolled optimization.
- **Bottom: Vanilla GAN.** The generator cycles through the modes of the data distribution the modes of the data, never converges to a fixed distribution, and only ever assigns significant probability mass to a single data mode at once.

Metz, Poole, Pfau and Sohl-Dickstein: Unrolled Generative Adversarial Networks. ICLR, 2017.

Discussion

Advantages:

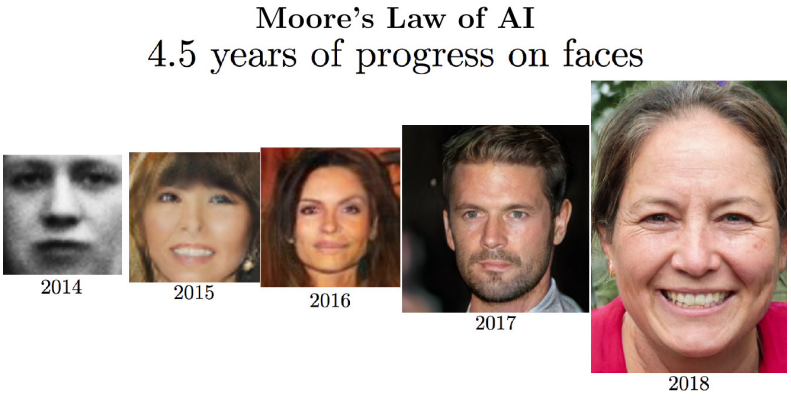
- A wide variety of functions and distributions can be modeled (flexibility)
- Only backpropagation required for training the model (no sampling)
- No approximation to the likelihood required as in VAEs
- Samples often more realistic than those of VAEs (but VAEs progress as well)

Disadvantages:

- No explicit representation of p_{model}
- Sample likelihood cannot be evaluated
- The discriminator and generator must be balanced well during training to ensure convergence to p_{data} and to avoid mode collapse
- Many tricks required: `https://towardsdatascience.com/10-lessons`

12.2
GAN Developments

GAN Developments



DCGAN

Architecture Guidelines for stable Deep Convolutional GANs:

- Replace any pooling layers with **strided convolutions** (discriminator) and fractional strided convolutions for upsampling (generator)
- Use **batch normalization** in both the generator and the discriminator
- Remove fully connected hidden layers for deeper architectures
- Use **ReLU** activations in the generator except for the output which uses tanh
- Use **Leaky ReLU** activations in the discriminator for all layers

Found to work well through systematic experimentation in DCGAN paper.

DCGAN Samples

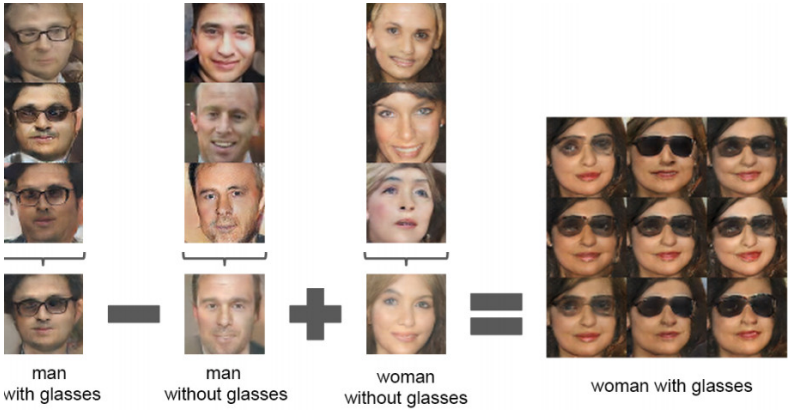
- **Bedroom samples** from a DCGAN look much better compared to vanilla GAN

Radford, Metz and Chintala: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR, 2016.

DCGAN Interpolations

- **Interpolations** between two random latent codes morph TVs into windows etc.

DCGAN Arithmetics



- **Vector arithmetic** on averaged $\mathbf{z}$ vectors of the samples (+ Gaussian noise)

GAN Explosion

- GAN - Generative Adversarial Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- AdaGAN - AdaGAN: Boosting Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AL-CGAN - Learning to Generate Images of Outdoor Scenes from Attributes and Semantic Layouts
- ALI - Adversarially Learned Inference
- AM-GAN - Generative Adversarial Nets with Labeled Data by Activation Maximization
- AnoGAN - Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery
- ArtGAN - ArtGAN: Artwork Synthesis with Conditional Categorial GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- BEGAN - BEGAN: Boundary Equilibrium Generative Adversarial Networks
- BiGAN - Adversarial Feature Learning
- BS-GAN - Boundary-Seeking Generative Adversarial Networks
- CGAN - Conditional Generative Adversarial Nets
- CaloGAN - CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks
- CCGAN - Semi-Supervised Learning with Context-Conditional Generative Adversarial Networks
- CatGAN - Unsupervised and Semi-supervised Learning with Categorical Generative Adversarial Networks
- CoGAN - Coupled Generative Adversarial Networks
- Context-RNN-GAN - Contextual RNN-GANs for Abstract Reasoning Diagram Generation
- C-RNN-GAN - C-RNN-GAN: Continuous recurrent neural networks with adversarial training
- CS-GAN - Improving Neural Machine Translation with Conditional Sequence Generative Adversarial Nets
- CVAE-GAN - CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training
- CycleGAN - Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks
- DTN - Unsupervised Cross-Domain Image Generation
- DCGAN - Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
- DiscoGAN - Learning to Discover Cross-Domain Relations with Generative Adversarial Networks
- DR-GAN - Disentangled Representation Learning GAN for Pose-Invariant Face Recognition
- DualGAN - DualGAN: Unsupervised Dual Learning for Image-to-Image Translation
- EBGAN - Energy-based Generative Adversarial Network
- f-GAN - f-GAN: Training Generative Neural Samplers using Variational Divergence Minimization
- FF-GAN - Towards Large-Pose Face Frontalization in the Wild
- GAWWN - Learning What and Where to Draw
- GeneGAN - GeneGAN: Learning Object Transfiguration and Attribute Subspace from Unpaired Data
- Geometric GAN - Geometric GAN
- GoGAN - Gang of GANs: Generative Adversarial Networks with Maximum Margin Ranking
- GP-GAN - GP-GAN: Towards Realistic High-Resolution Image Blending
- IAN - Neural Photo Editing with Introspective Adversarial Networks
- iGAN - Generative Visual Manipulation on the Natural Image Manifold
- IcGAN - Invertible Conditional GANs for image editing
- ID-CGAN - Image De-raining Using a Conditional Generative Adversarial Network
- Improved GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LAGAN - Learning Particle Physics by Example: Location-Aware Generative Adversarial Networks for Physics Synthesis
- LAPGAN - Deep Generative Image Models using a Laplacian Pyramid of Adversarial Networks

- `https://github.com/hindupuravinash/the-gan-zoo`
- `https://github.com/soumith/ganhacks`

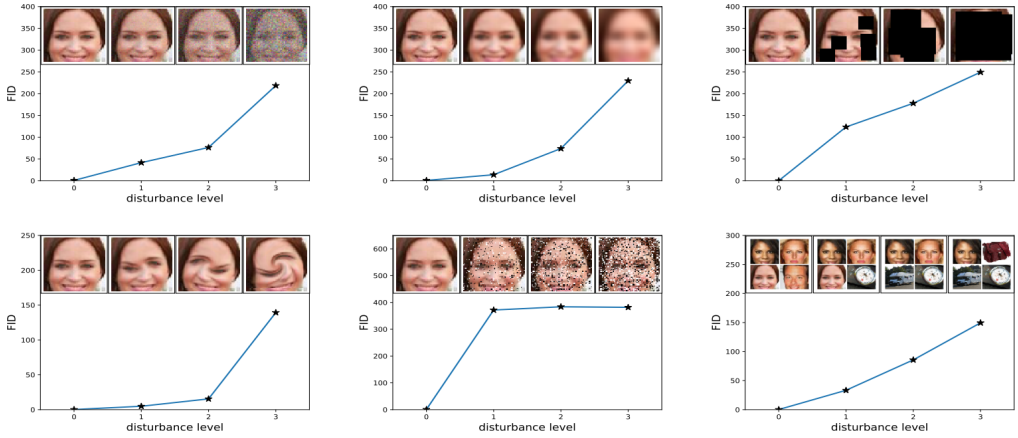
Fréchet Inception Distance

- Evaluating the performance of generative models without exact likelihood is hard
- The **Fréchet inception distance (FID)** is a metric used to assess the quality of images created by the generator of a generative adversarial network (GAN)
- The FID compares the distribution of generated images with the distribution of real images based on deeper features of a pre-trained **Inception v3 network**
- The FID metric is the Fréchet distance between two multidimensional Gaussian distributions: $\mathcal{N}(\boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)$, the distribution of the Inception v3 features of the images generated by a GAN and $\mathcal{N}(\boldsymbol{\mu}_d, \boldsymbol{\Sigma}_d)$, the distribution of the Inception v3 features computed over the training set:

$$FID = \|\boldsymbol{\mu}_m - \boldsymbol{\mu}_d\|_2^2 + \mathrm{Tr}(\boldsymbol{\Sigma}_m + \boldsymbol{\Sigma}_d - 2(\boldsymbol{\Sigma}_m \boldsymbol{\Sigma}_d)^{1/2})$$

Heusel, Ramsauer, Unterthiner, Nessler and Hochreiter: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NIPS, 2017.

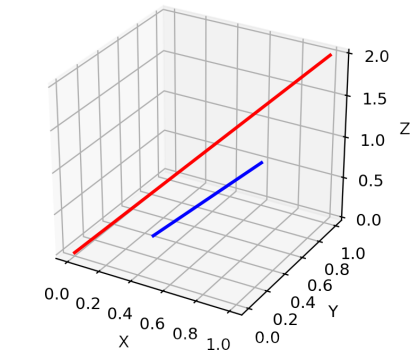
Fréchet Inception Distance

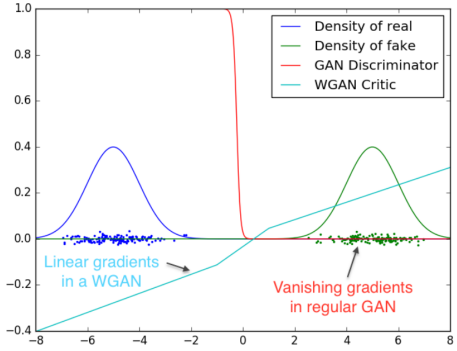


- The FID **measures image fidelity** but cannot measure or prevent mode collapse

Heusel, Ramsauer, Unterthiner, Nessler and Hochreiter: GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NIPS, 2017.

Wasserstein GAN

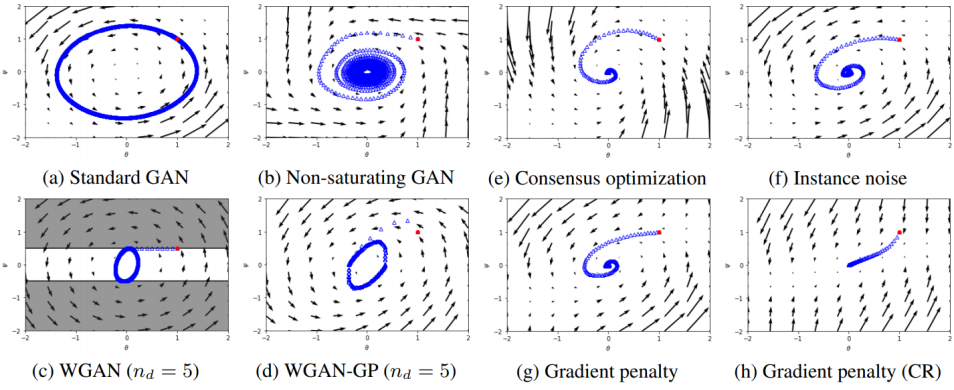




- Low dimensional manifolds in high dimension space often have little overlap
- The discriminator of a vanilla GAN saturates if there is no **overlapping support**
- **WGANs** uses the **Earth Mover's distance** which can handle such scenarios

Arjovsky, Chintala and Bottou: Wasserstein GAN. Arxiv, 2017.

Gradient Penalties and Convergence

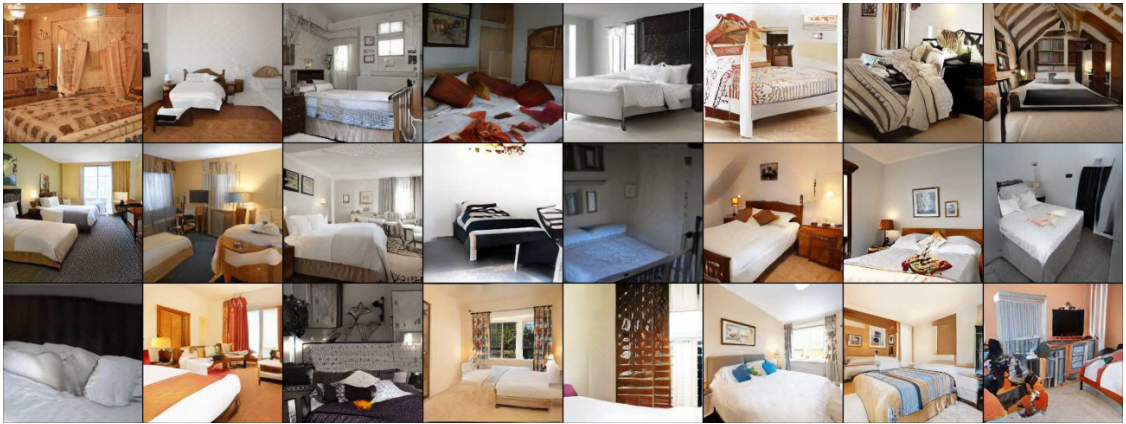


- Adding a **gradient penalty** wrt. the gradients of D **stabilizes GAN training:**

$$V(D,G) = \mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})}\left[\log D(\mathbf{x}) - \lambda\|\nabla_{\mathbf{x}} D(\mathbf{x})\|^2\right] + \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z})}\left[\log(1 - D(G(\mathbf{z})))\right]$$

Mescheder, Geiger and Nowozin: Which Training Methods for GANs do actually Converge? ICML, 2018.

Gradient Penalties and Convergence



- A DCGAN with **gradient penalties** leads to high-quality samples without hacks

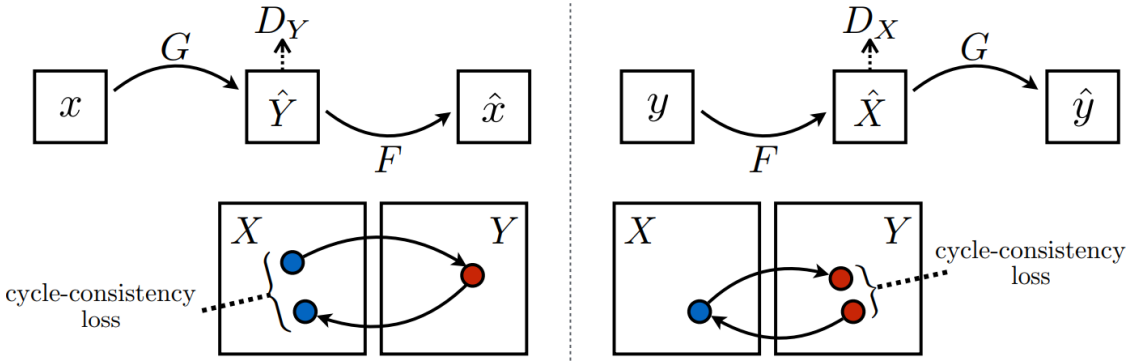
Mescheder, Geiger and Nowozin: Which Training Methods for GANs do actually Converge? ICML, 2018.

Gradient Penalties and Convergence

- A DCGAN with **gradient penalties** leads to high-quality samples without hacks

Mescheder, Geiger and Nowozin: Which Training Methods for GANs do actually Converge? ICML, 2018.

CycleGAN

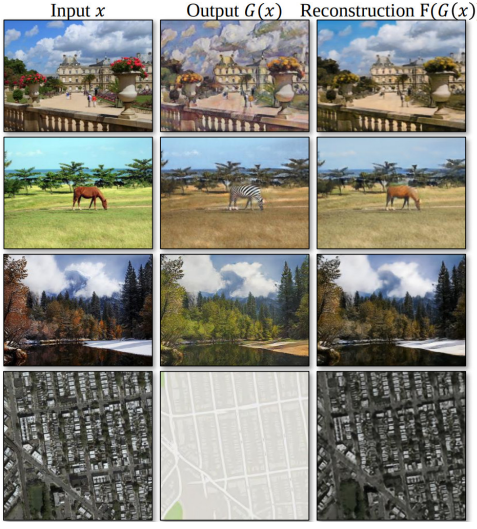


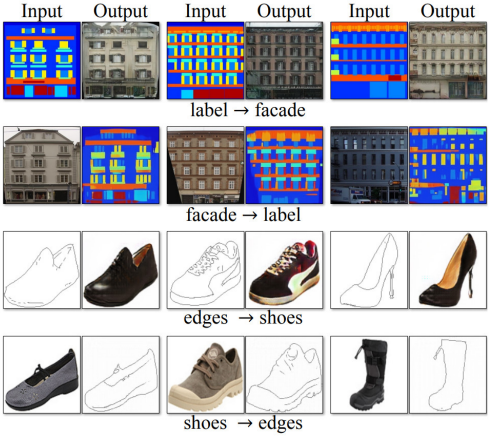
CycleGAN Image-to-Image Translation:

- Learn forward and backward mapping btw. two domains (domains = latents)
- Use cycle-consistency and adversarial losses to constrain this mapping

Zhu, Park, Isola and Efros: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. ICCV, 2017.

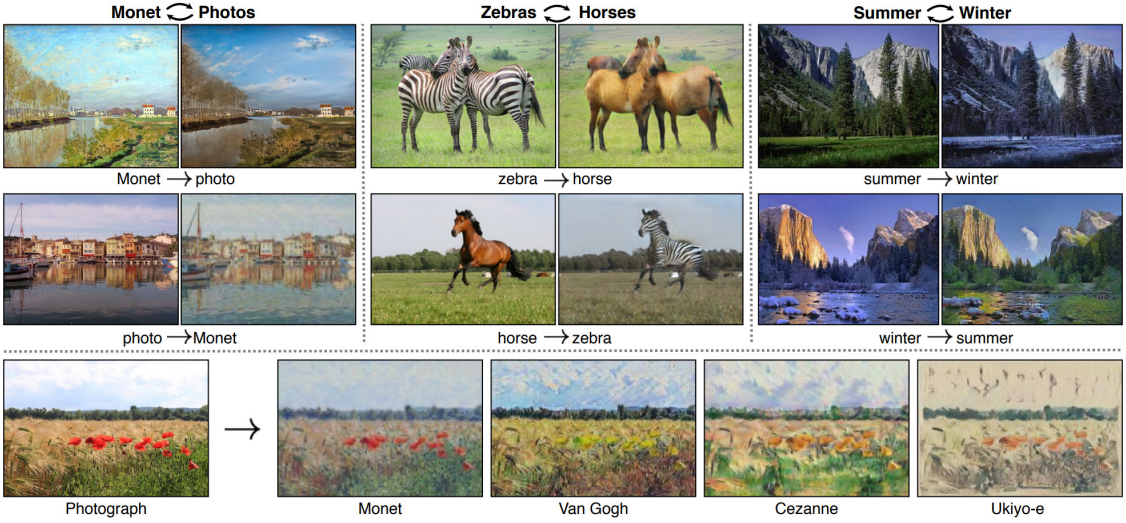
CycleGAN





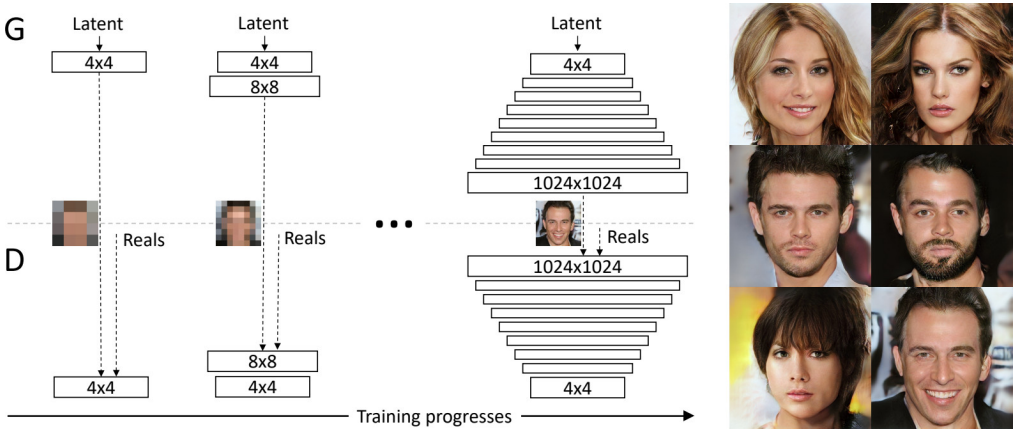
Zhu, Park, Isola and Efros: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. ICCV, 2017.

CycleGAN



Zhu, Park, Isola and Efros: Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks. ICCV, 2017.

Progressive Growing of GANs



- **Grow** the generator and discriminator **resolution** by adding layers during training

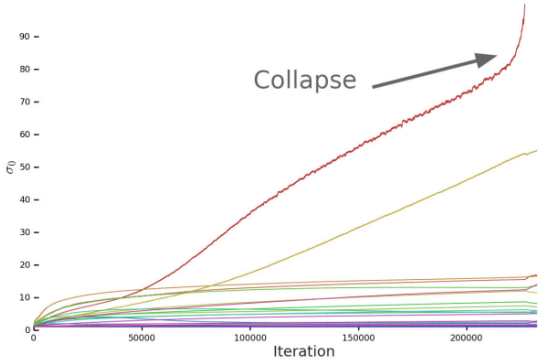
Karras, Aila, Laine and Lehtinen: Progressive Growing of GANs for Improved Quality, Stability, and Variation. ICLR, 2018.

BigGAN

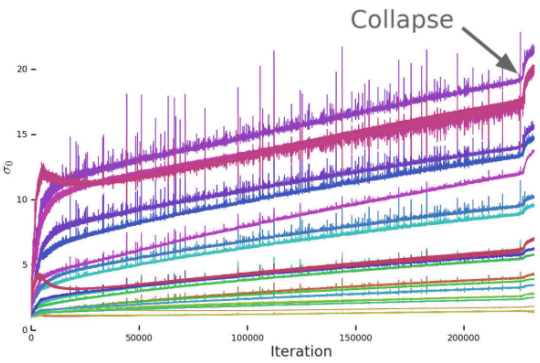
- Scale class-conditional GANs to **ImageNet** (512^2) without progressive growing
- Key: more parameters, larger minibatches, orthogonal regularization of G
- Explore variants of spectral normalization and gradient penalties for D
- Analyze **trade-off** between stability (regularization) and performance (FID)

Brock, Donahue and Simonyan: Large Scale GAN Training for High Fidelity Natural Image Synthesis. ICLR, 2019.

BigGAN



(a) **G**

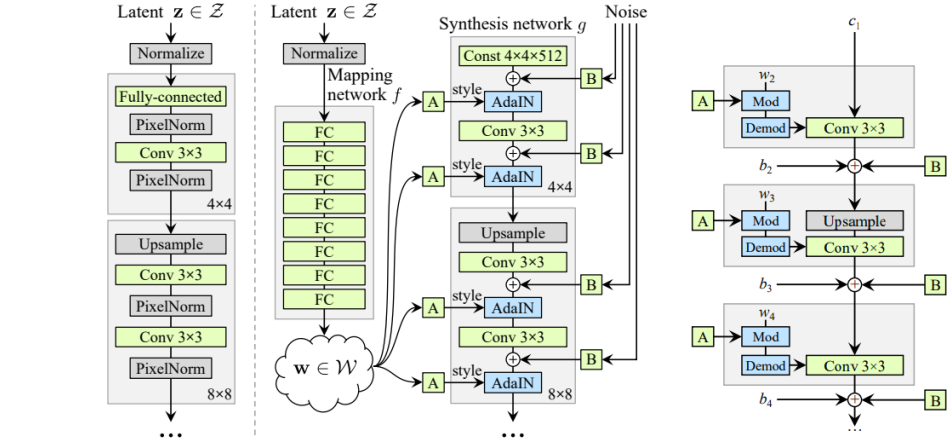


(b) **D**

- Monitor **singular values** of weight matrices of generator and discriminator
- Found **early stopping** leads to better FID scores compared to regularizing D

Brock, Donahue and Simonyan: Large Scale GAN Training for High Fidelity Natural Image Synthesis. ICLR, 2019.

StyleGAN / StyleGAN2



https://youtu.be/c-NJtV9Jvp0

StyleGAN / StyleGAN2

- Complex **stochastic variation** with different realizations of input noise

StyleGAN / StyleGAN2

http://thispersondoesnotexist.com/

Karras, Laine, Aittala, Hellsten, Lehtinen and Aila: Analyzing and Improving the Image Quality of StyleGAN. CVPR, 2020.

StyleGAN / StyleGAN2

http://thispersondoesnotexist.com/

Karras, Laine, Aittala, Hellsten, Lehtinen and Aila: Analyzing and Improving the Image Quality of StyleGAN. CVPR, 2020.

StyleGAN / StyleGAN2

http://thispersondoesnotexist.com/

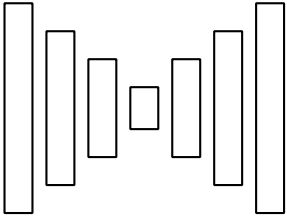
Karras, Laine, Aittala, Hellsten, Lehtinen and Aila: Analyzing and Improving the Image Quality of StyleGAN. CVPR, 2020.

12.3
Research at AVG

Intelligent systems interact with a **3D world**

3D Reconstruction

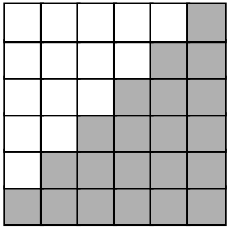
Input Images

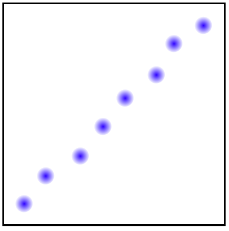


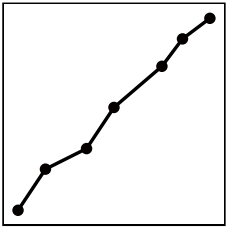
Neural Network

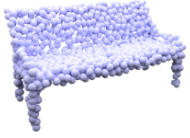
3D Reconstruction

3D Representations









Voxels

[Maturana et al., IROS 2015]

Points

[Fan et al., CVPR 2017]

Meshes

[Groueix et al., CVPR 2018]

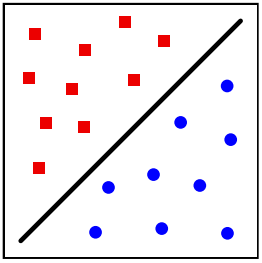
Occupancy Networks

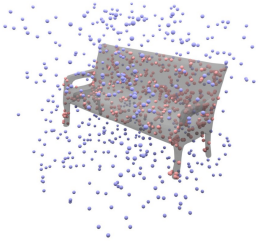
Key Idea:

- Do not represent 3D shape explicitly
- Instead, consider surface **implicitly** as **decision boundary** of a non-linear classifier:

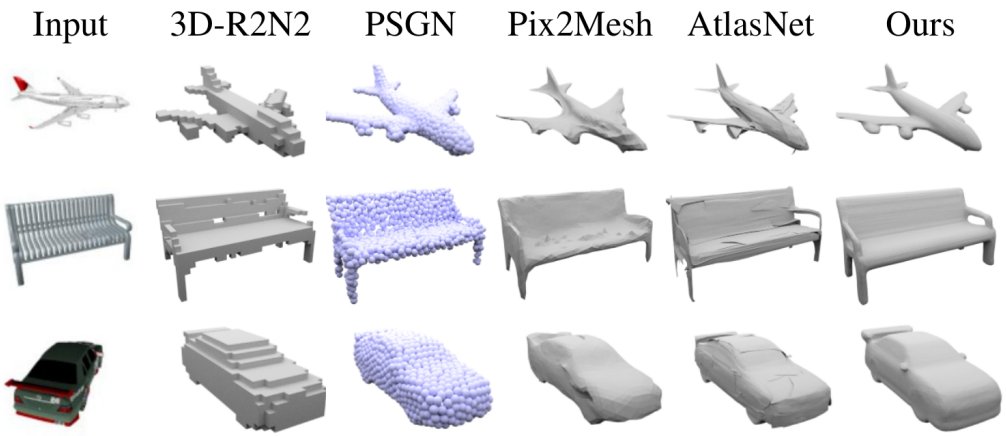
$$f_\theta : \mathbb{R}^3 \times \mathcal{X} \rightarrow [0, 1]$$

3D Location | Condition (eg, Image) | Occupancy Probability



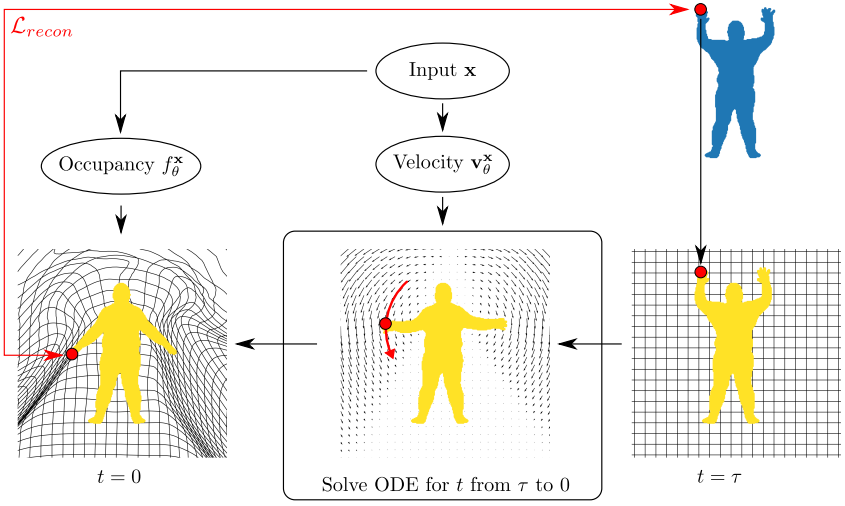


Occupancy Networks



Mescheder, Oechsle, Niemeyer, Nowozin and Geiger: Occupancy Networks: Learning 3D Reconstruction in Function Space. CVPR, 2019.

Occupancy Flow

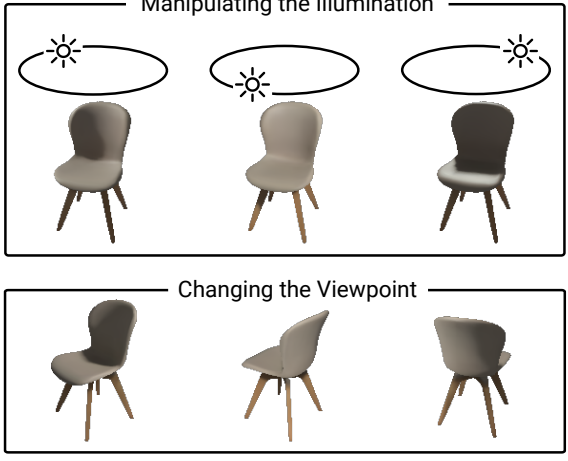


Conditional Surface Light Fields

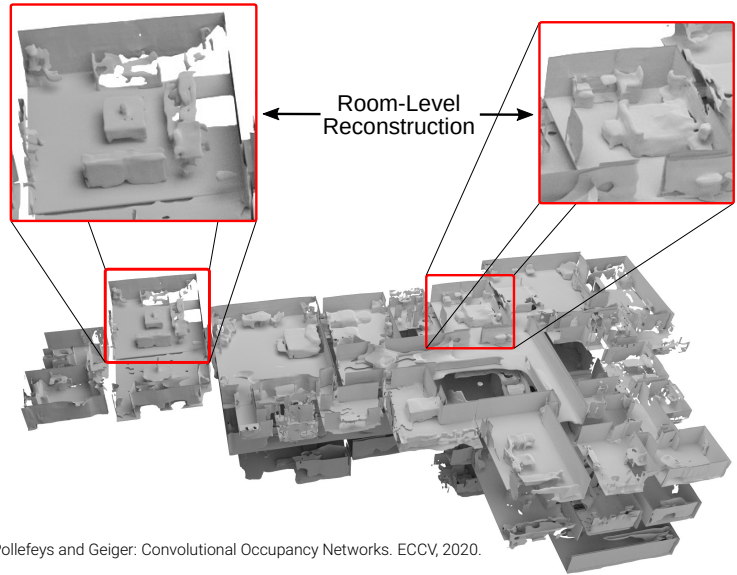
Input Image

3D Geometry

Neural Network

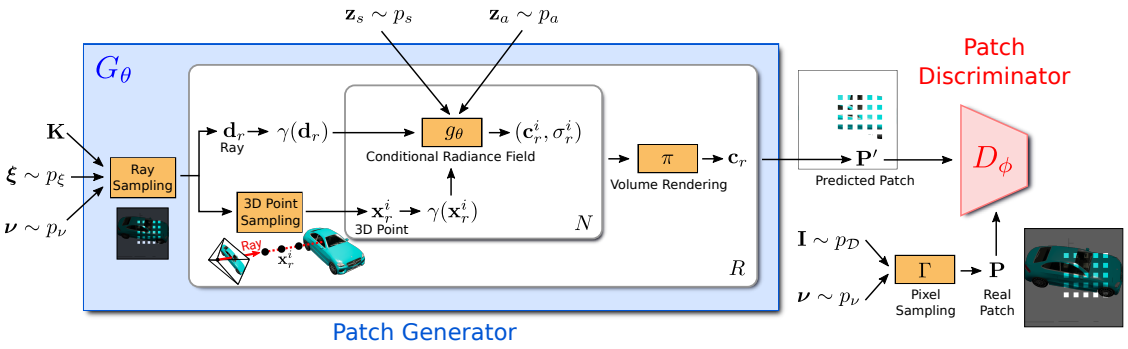


Convolutional Occupancy Networks

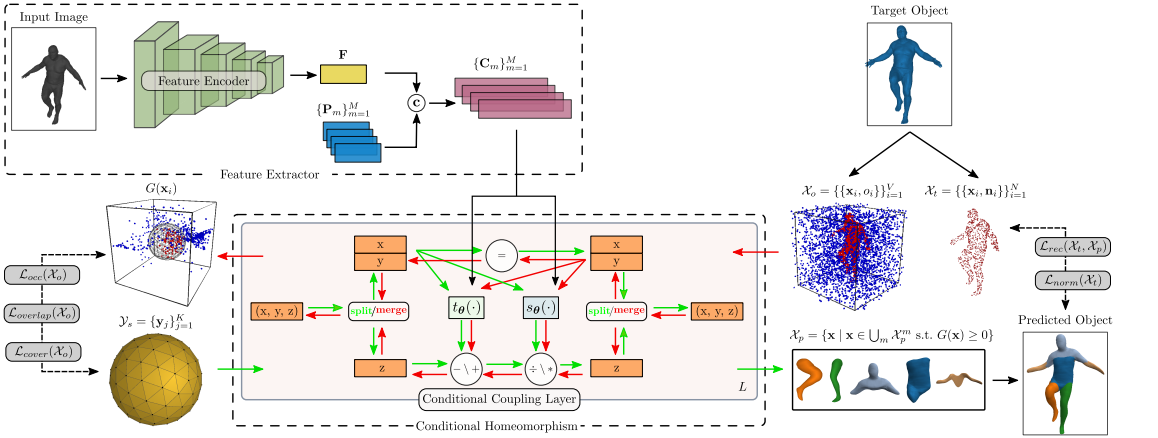


Peng, Niemeyer, Mescheder, Pollefeys and Geiger: Convolutional Occupancy Networks. ECCV, 2020.

Generative Radiance Fields

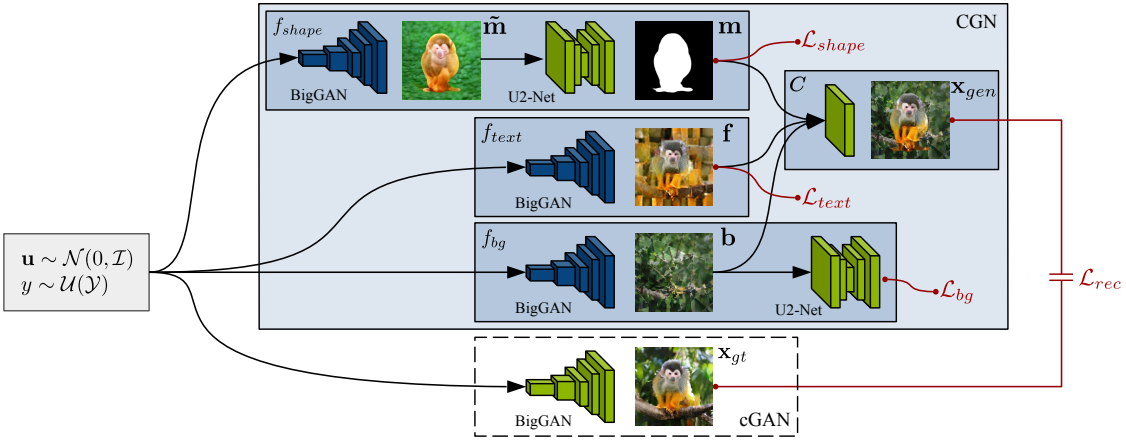


Neural Parts

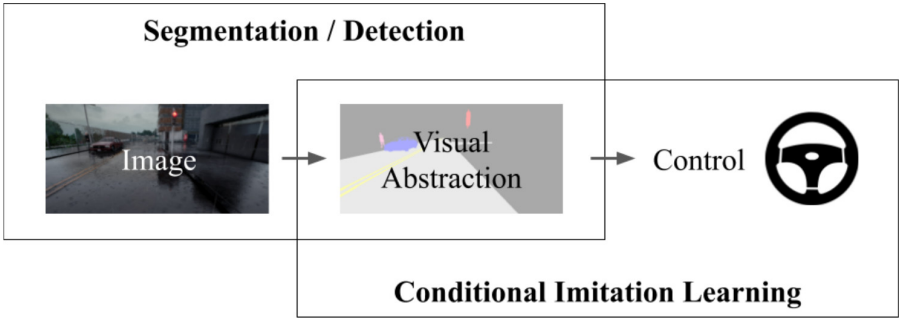


Paschalidou, Katharopoulos, Geiger and Fidler: Neural Parts: Learning Expressive 3D Shape Abstractions with Invertible Neural Networks, 2021.

Counterfactual Generative Networks



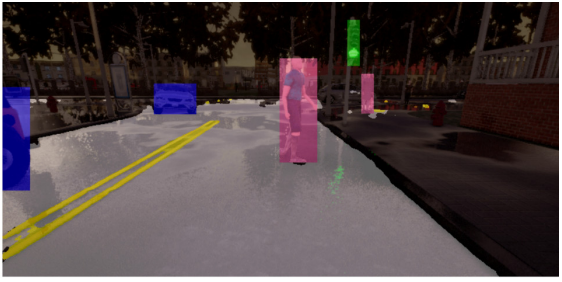
Label Efficient Visual Abstractions



Behl, Chitta, Prakash, Ohn-Bar and Geiger: Label Efficient Visual Abstractions for Autonomous Driving. IROS, 2020.

Label Efficient Visual Abstractions

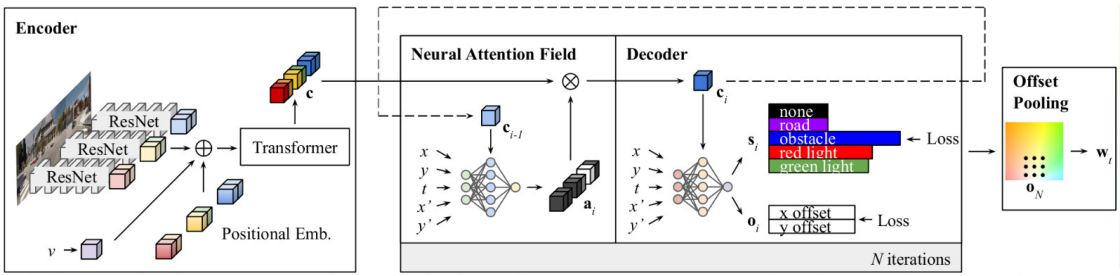
Trained with 6400 finely annotated images and 14 classes
Annotation time ≈ 7500 hours, policy success rate = 50%



Trained with 1600 coarsely annotated images and 6 classes
Annotation time ≈ 50 hours, policy success rate = 58%

Behl, Chitta, Prakash, Ohn-Bar and Geiger: Label Efficient Visual Abstractions for Autonomous Driving. IROS, 2020.

Neural Attention Fields for End-to-End Autonomous Driving



Chitta, Prakash and Geiger: Neural Attention Fields for End-to-End Autonomous Driving. 2021.

Visit our Research Blog

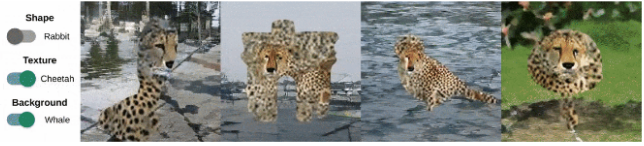
Autonomous Vision Blog Posts Categories Tags Website Subscribe

Andreas Geiger

Tübingen - Germany
Website
Facebook
Email

Recent Posts

Counterfactual Generative Networks



A generative model structured into independent causal mechanisms produces images for training invariant classifiers.

http://autonomousvision.github.io

Thank you for listening!

Good luck for the exam!