

BEUTH HOCHSCHULE FÜR TECHNIK BERLIN
University of Applied Sciences

Learning from Images

Generative Models

Master DataScience
Winter term 2019/20

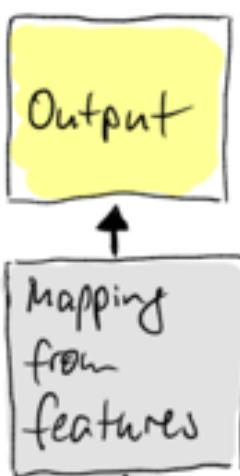
Prof. Dr. Kristian Hildebrand
khildebrand@beuth-hochschule.de

Learned

Engineered



Rule-based
System



Classic
Machine
Learning



Representation
learning

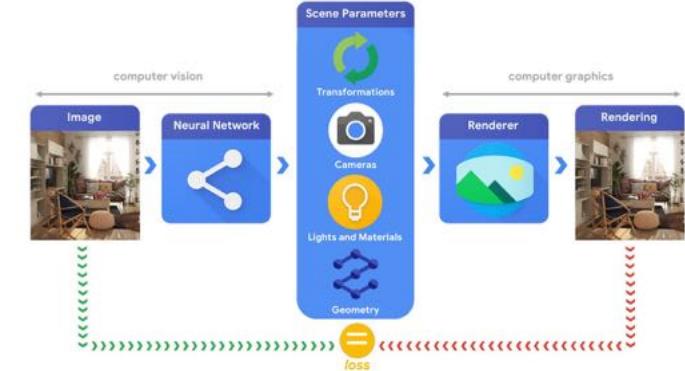


Deep
Learning

Why do we want to generate Content?

- Learning an understanding of the world

“What I cannot create, I do not understand.” Richard Feynman



<https://github.com/tensorflow/graphics>

- Support creative processes
 - Image Denoising, Super Resolution
 - 3D Model generation
- Data Augmentation
- Can benefit Reinforcement Learning exploration strategies

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

X is data, y is label

Goal: Learn a function to map $x \rightarrow y$

Examples: Classification, regression, Object detection, Semantics segmentation, image captioning, etc.

Unsupervised Learning

Data: x

Just data, no labels

Goal: Learn underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised Learning

Supervised Learning

Data: (x, y)

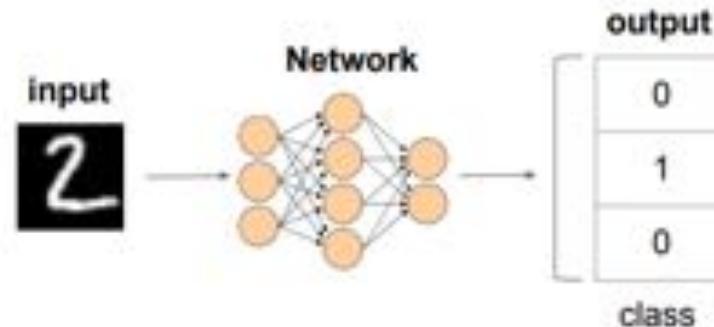
X is data, y is label

Goal:

Learn a function to map $x \rightarrow y$

Examples: Classification,
regression,
Object detection, Semantics
segmentation, image
captioning, etc.

Discriminative model: $P(Y | X)$



$$P(Y = [0,1,0] \mid X = [\text{pixel}_1, \text{pixel}_2, \dots, \text{pixel}_{784}])$$



DOG, DOG, CAT

Object Detection



A cat sitting on a suitcase on the floor

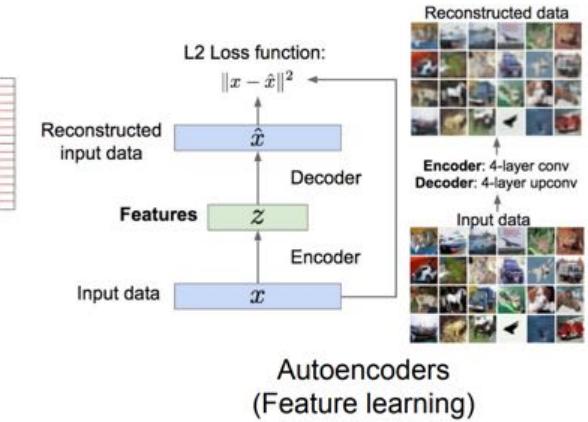
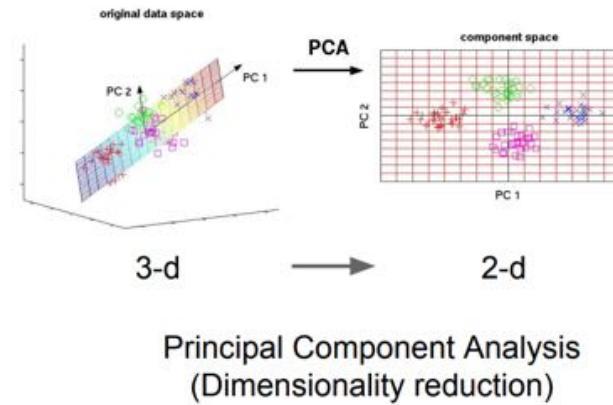
Image captioning

Supervised vs Unsupervised Learning

Unsupervised Learning

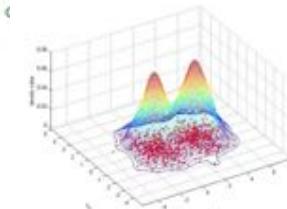
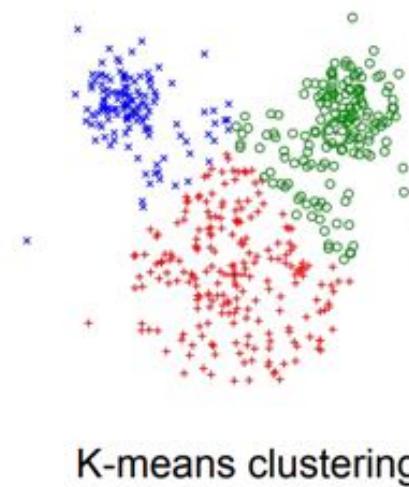
Data: x

Just data, no labels

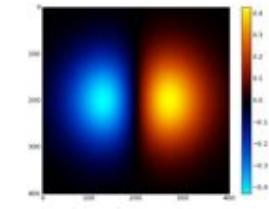


Goal: Learn underlying hidden structure of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



2-d density estimation



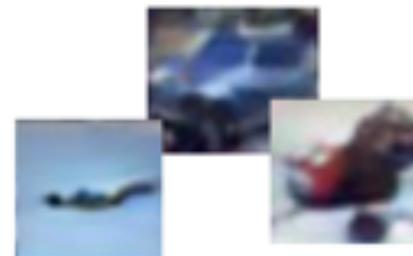
Holy Grail: Solve Unsupervised Learning
=> Understand the underlying structure of the visual world

Generative Models

- Given training data, generate new samples from same distribution



Training data $\sim p_{\text{data}}(x)$



Generated samples $\sim p_{\text{model}}(x)$

Want to learn $p_{\text{model}}(x)$ similar to $p_{\text{data}}(x)$

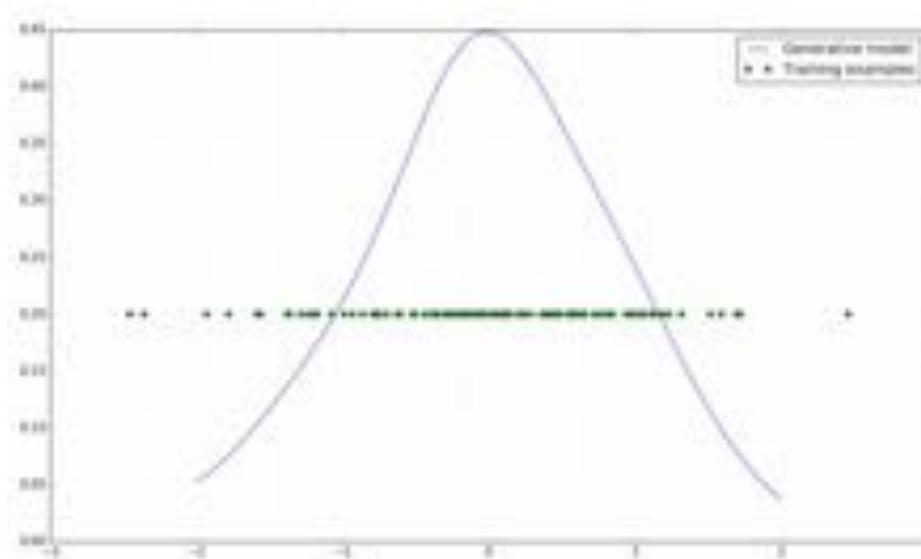
- Addresses density estimation, a core problem in unsupervised learning
- Several flavors:**
 - Explicit density estimation: explicitly define and solve for $p_{\text{model}}(x)$
 - Implicit density estimation: learn model that can sample from $p_{\text{model}}(x)$ w/o explicitly defining it

What is a generative model?

We want our model with parameters θ to output samples distributed P_{model} , matching the distribution of our training data P_{data} or $P(X)$.

Example: $y = f(x)$, where y is scalar, make P_{model} similar to P_{data} by training the parameters θ to maximize their similarity

If we have 1-D data that basically following a gaussian distribution (green dots), then a perfectly trained model (P_{model}) need to learn the μ and σ of the data distribution (P_{data})



What is a generative model?

- **Key Idea:** our model cares about what distribution generated the input data points, and we want to mimic it with our probabilistic model. **Our learned model should be able to make up new samples from the distribution, not just copy and paste existing samples!**

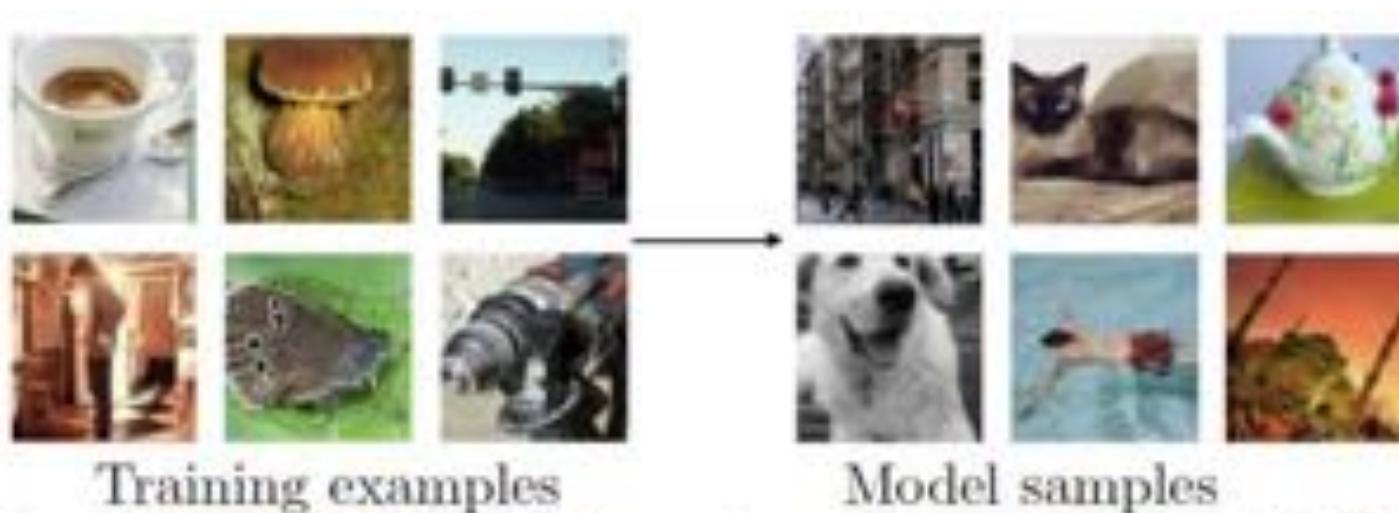
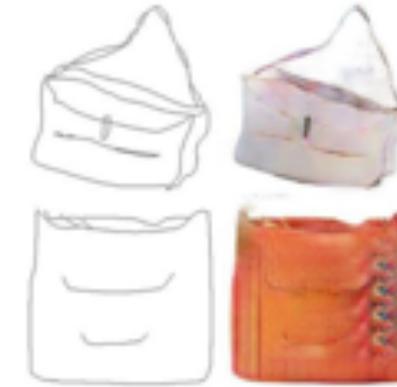


Figure from [NIPS 2016 Tutorial: Generative Adversarial Networks \(I. Goodfellow\)](#)

Why Generative Models?

- Realistic samples for artwork, super-resolution, colorization, etc.



(1) Alec Radford et al. 2016; (2) David Berthelot et al. 2017; Phillip Isola et al. 2017.

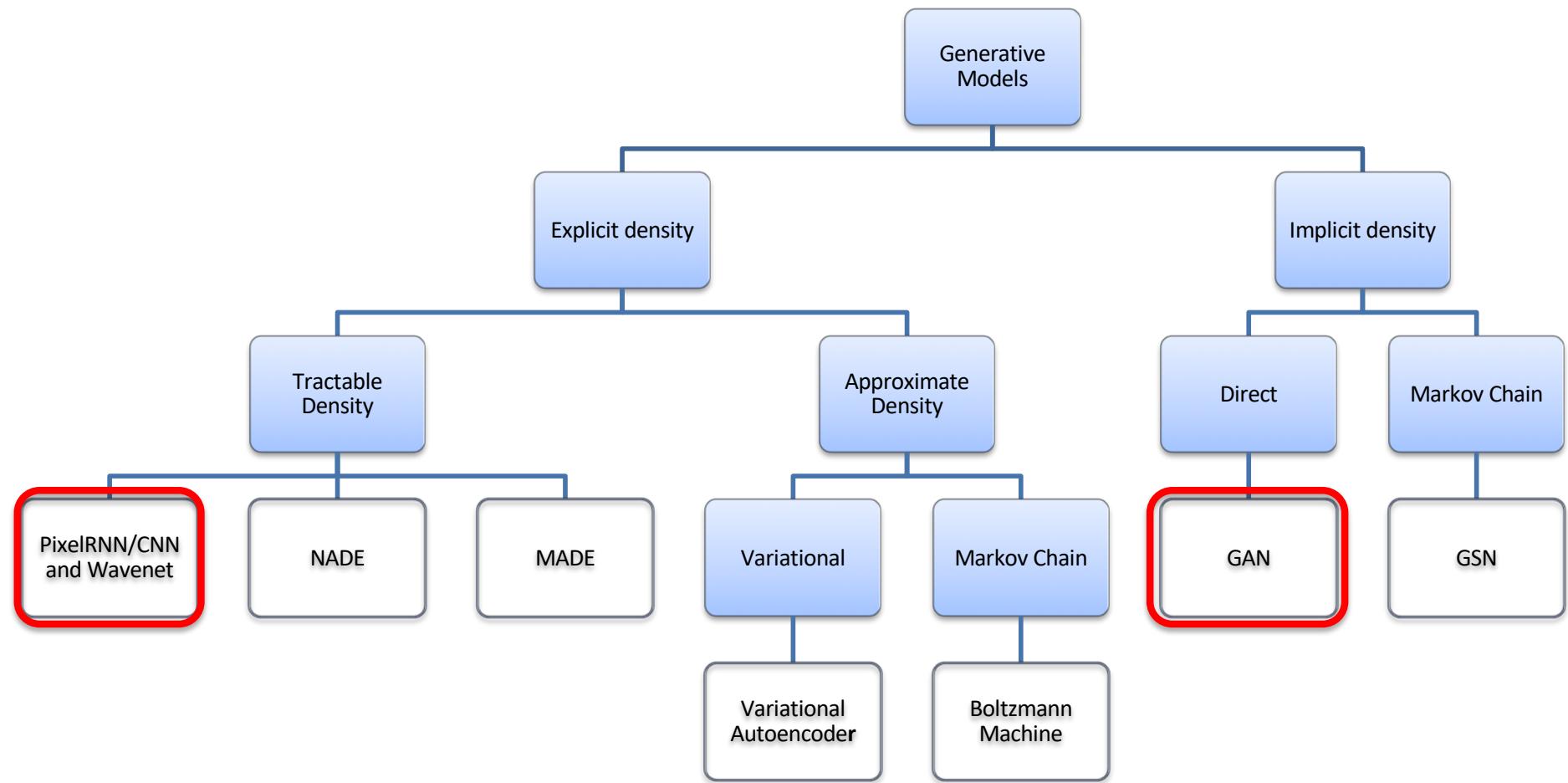
- Generative models of time-series data can be used for simulation and planning (reinforcement learning applications)
- Training generative models can also enable inference of latent representations that can be useful as general features

Why Generative Models?

- Be able to generate realistic synthetic samples
 - Possibly perform data augmentation
 - Simulate possible futures for learning algorithms
- Fill the blanks in the data
- Manipulate real samples with the assistance of the generative model
 - edit pictures with guidance
- GAN quality over the last years



Taxonomy of generative models



Copyright and adapted from Ian Goodfellow, Tutorial on GAN, 20017

PixelRNN / PixelCNN

**see jupyter notebook
example implementation**

Factorizing the joint distribution

- Model explicitly the joint probability distribution of data stream \mathbf{x} as a product of element-wise conditional distribution for each element x_i in the stream.
 - Example: An image \mathbf{x} of size (n,n) is decomposed row by row and pixel by pixel within every row
 - Apply **probability chain rule**: x_i is the i -th pixel in the image.
 - Then maximize the likelihood of training data

$$p(x) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

Likelihood of image x → $p(x)$

Probability of i 'th pixel value given all previous pixel values

To model highly nonlinear and long-range correlations between pixels and their conditional distributions, we will need a powerful non-linear sequential model.

Question: Which model can we use?

Answer: Recurrent Neural Network.

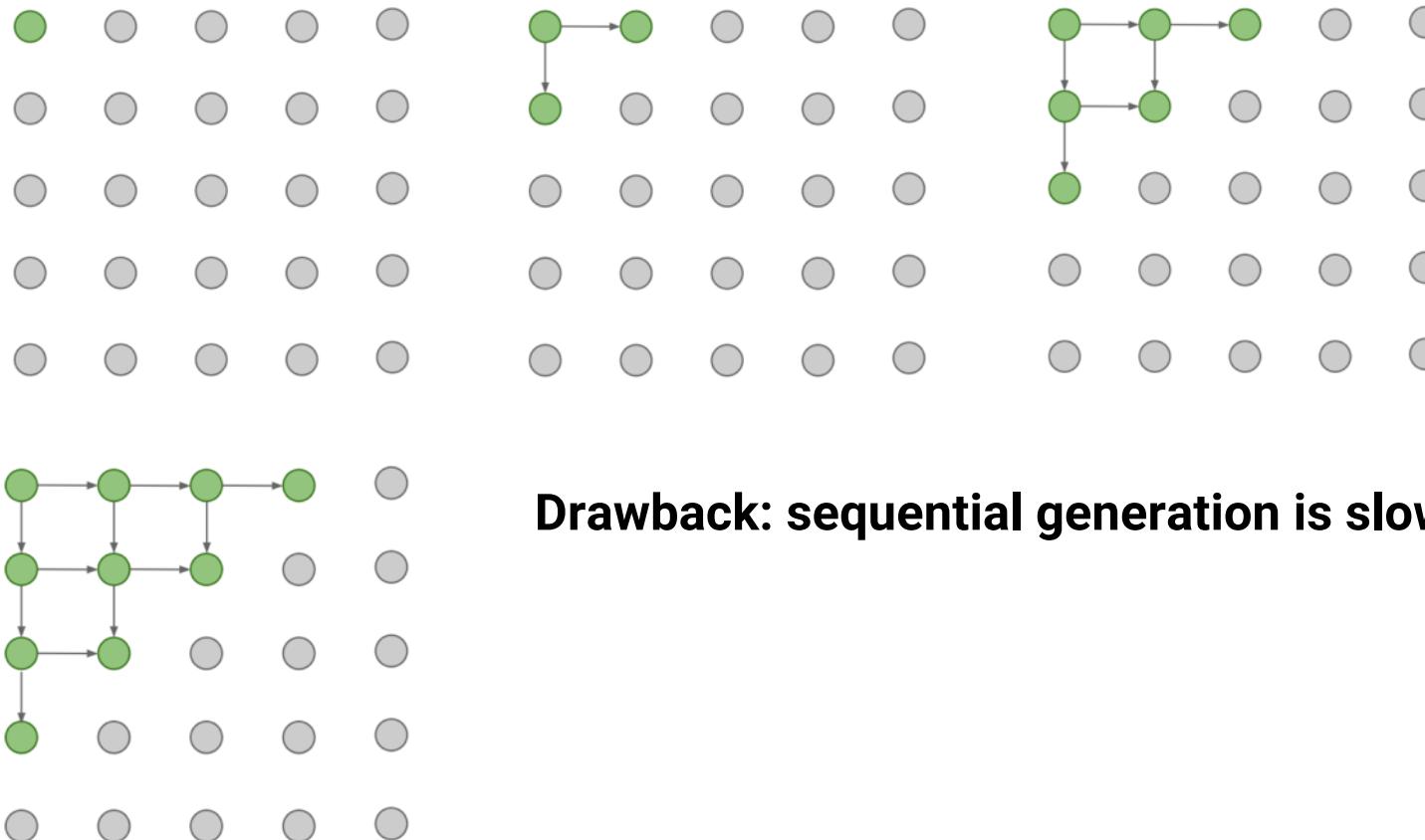
Char-RNN

Training data: Linux Kernel C Code

```
/*
 * Increment the size file of the new incorrect UI_FILTER group information
 * of the size generatively.
 */
static int indicate_policy(void)
{
    int error;
    if (fd == MARN_EPT) {
        /*
         * The kernel blank will coeld it to userspace.
         */
        if (ss->segment < mem_total)
            unblock_graph_and_set_blocked();
        else
            ret = 1;
        goto bail;
    }
    segaddr = in_SB(in.addr);
    selector = seg / 16;
    setup_works = true;
    for (i = 0; i < blocks; i++) {
        seq = buf[i++];
        bpf = bd->bd.next + i * search;
        if (fd) {
            current = blocked;
        }
    }
    rw->name = "Getjbbregs";
    bprm_self_clearl(&iv->version);
    regs->new = blocks[(BPF_STATS << info->historidac)] | PFMR_CLOBATHINC_SECONDS << 12;
    return segtable;
}
```

PixelRNN [van der Oord et al. 2016]

- Generate image pixels starting from corner
- Dependency on previous pixels modeled using an RNN (LSTM)



PixelCNN [van der Oord et al. 2016]

- Still generate image pixels starting from corner
- Dependency on previous pixels now modeled using a CNN over context region
- Training is faster than PixelRNN (can parallelize convolutions since context region values known from training images)
- Generation must still proceed sequentially => still slow

Softmax loss at each pixel

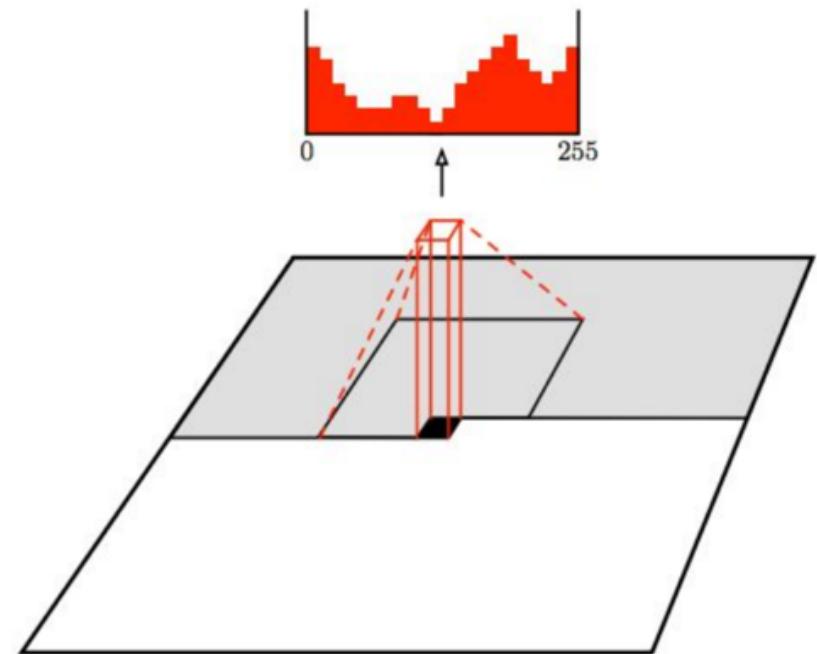
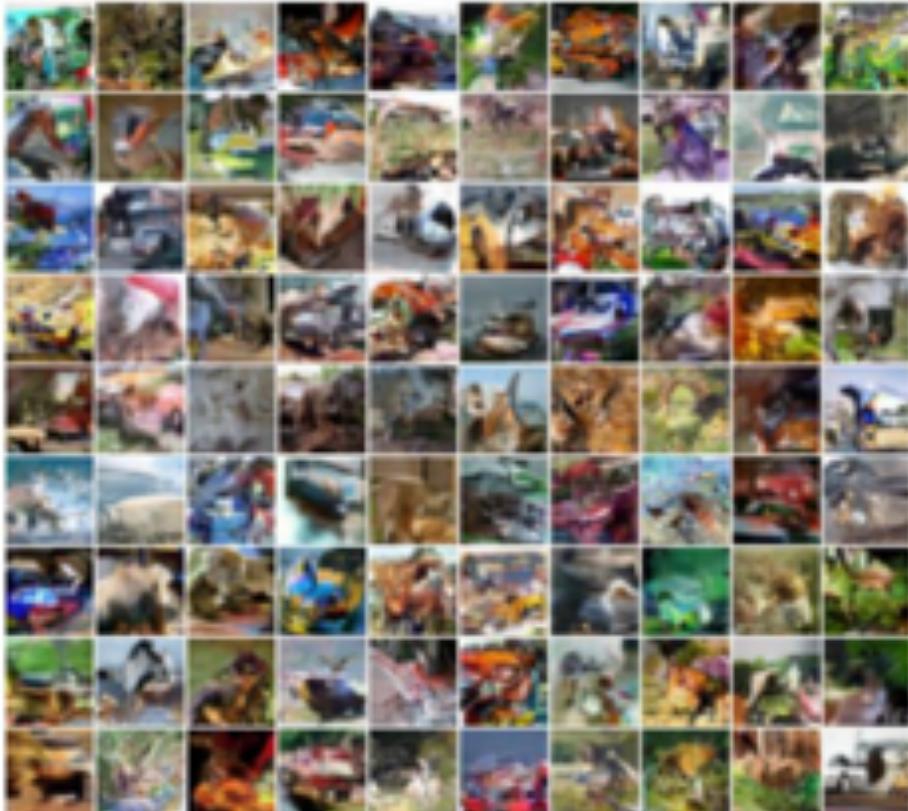
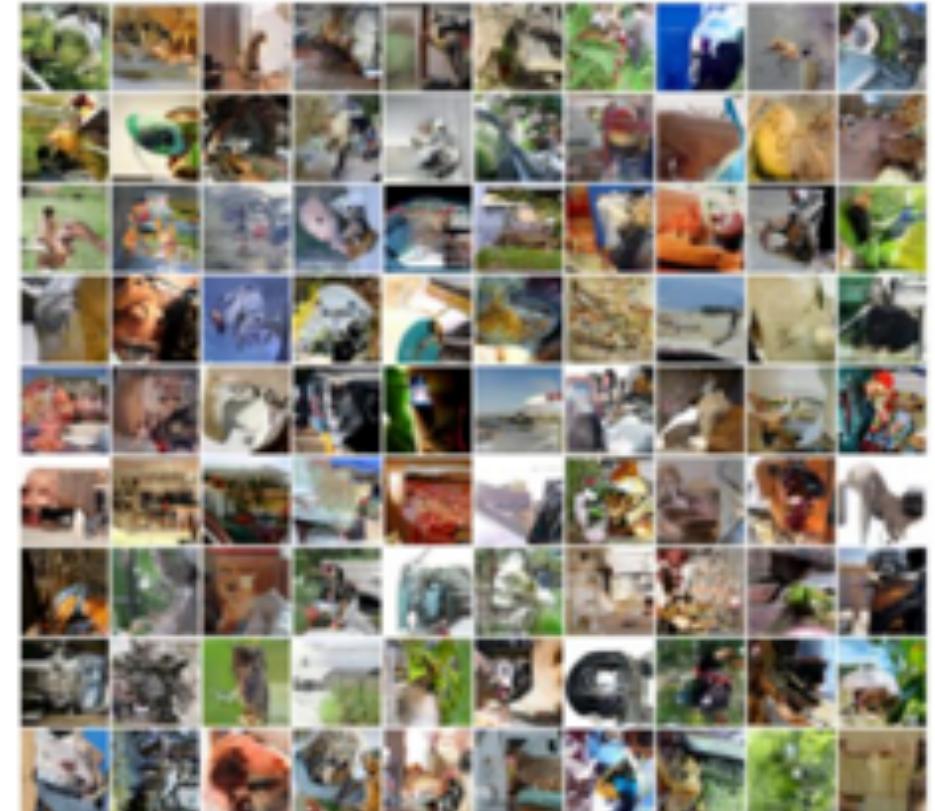


Figure copyright van der Oord et al., 2016.

PixelRNN / PixelCNN generation samples



32x32 CIFAR-10



32x32 ImageNet

Figures copyright Aaron van der Oord et al., 2016.

Generative Adversarial Networks (GAN)

Learn the structure of the world

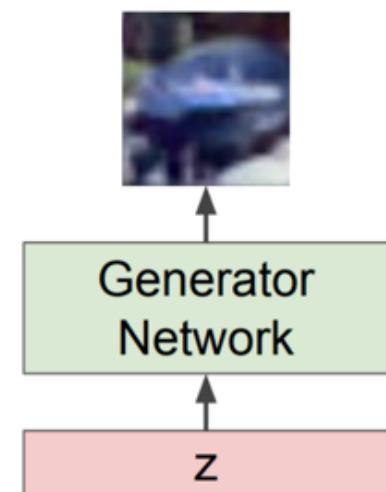
<https://www.youtube.com/embed/kSLJriaOumA?start=335>

Generative Adversarial Networks

- **Problem:** Want to sample from complex, high-dimensional training distribution. No direct way to do this!
- **Solution:** Sample from a simple distribution, e.g. random noise. Learn transformation to training distribution.
- **Question:** What can we use as transformation?
- **Answer:** A neural network!

Output: Sample from training distribution

Input: Random noise



Training GANs: Two-player game (Counterfeit example)

[Goodfellow et al., "Generative Adversarial Nets", NIPS 2014]

Generator

Counterfeiter

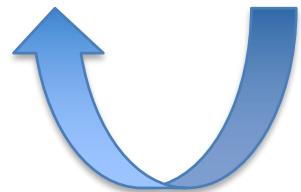
Discriminator

FBI



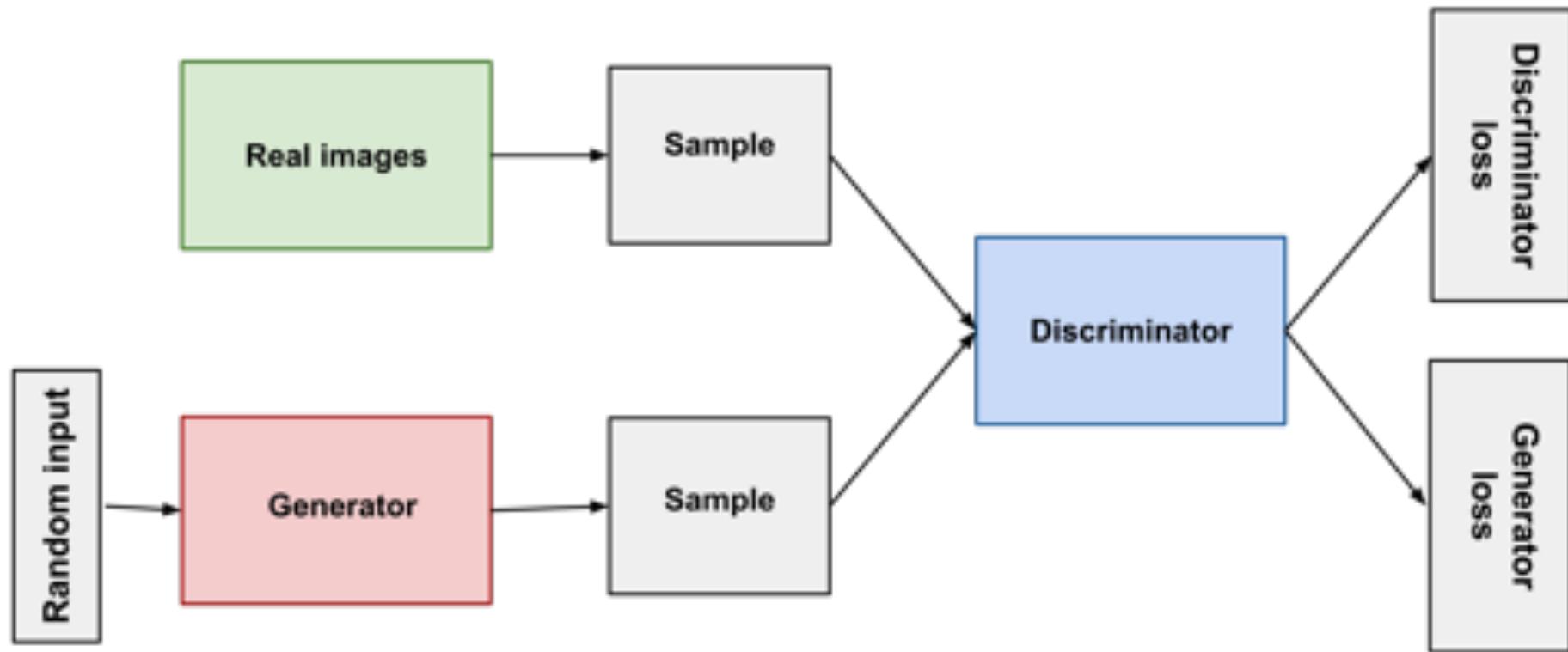
Fake

Real



Training GANs: Two-player game

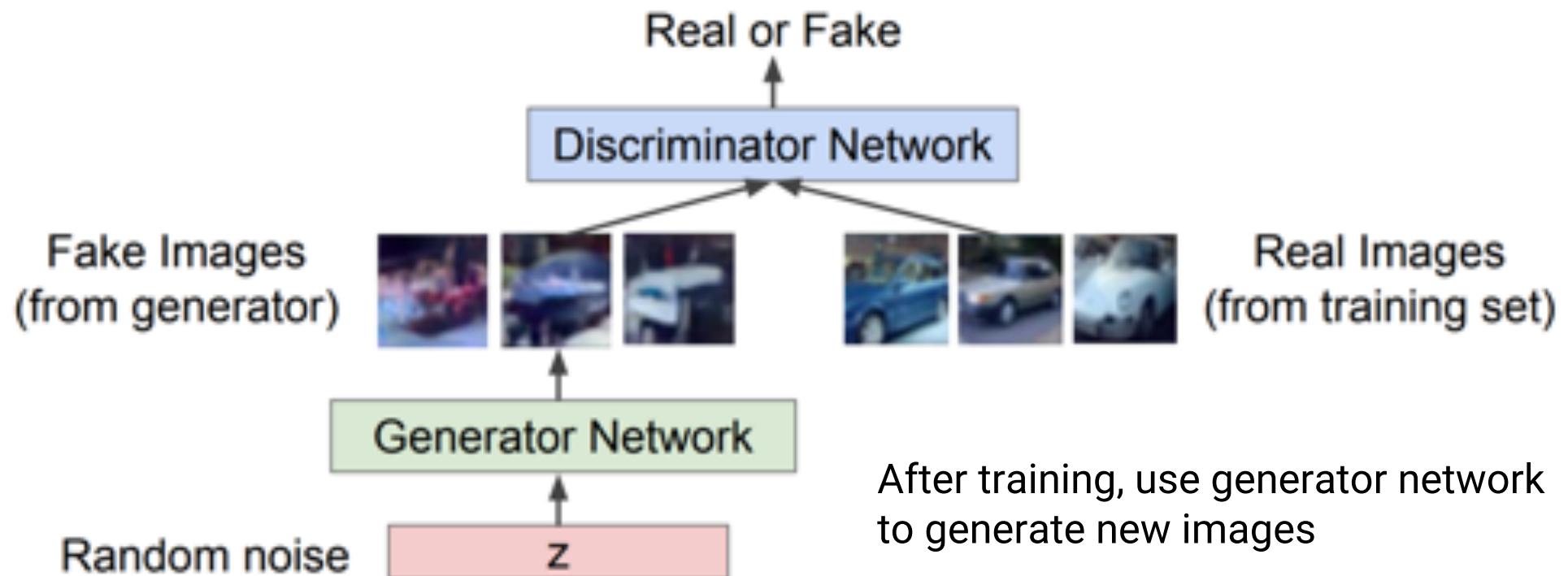
[Goodfellow et al., “Generative Adversarial Nets”, NIPS 2014]



Training GANs: Two-player game

[Goodfellow et al., "Generative Adversarial Nets", NIPS 2014]

- **Generator network:** try to fool the discriminator by generating real-looking images
- **Discriminator network:** try to distinguish between real and fake images



Training GANs: Two-player game

[Goodfellow et al., "Generative Adversarial Nets", NIPS 2014]

- **Generator network:** try to fool the discriminator by generating real-looking images
- **Discriminator network:** try to distinguish between real and fake images

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data } x} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data } G(z)}) \right]$$

- Discriminator D_{θ_d} wants to maximize objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake)
- Generator G_{θ_g} wants to minimize objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real)

Training GANs: Two-player game



D(x) tries to be near 1

Differentiable function D

X sampled from data

D tries to make $D(G(z))$ near 0,
G tries to make $D(G(z))$ near 1

D

X sampled from model

Differentiable function G

Input noise z



$$J^{(D)} = -\frac{1}{2} \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}} \log D(\mathbf{x}) - \frac{1}{2} \mathbb{E}_{\mathbf{z}} \log (1 - D(G(\mathbf{z})))$$

$$, J^{(G)} = -\frac{1}{2} \mathbb{E}_{\mathbf{z}} \log D(G(\mathbf{z}))$$

GAN Training algorithm

Lets code a GAN

for number of training iterations **do**

for k steps **do**

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Sample minibatch of m examples $\{\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\mathbf{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^m \left[\log D_{\theta_d}(\mathbf{x}^{(i)}) + \log(1 - D_{\theta_d}(G_{\theta_g}(\mathbf{z}^{(i)}))) \right]$$

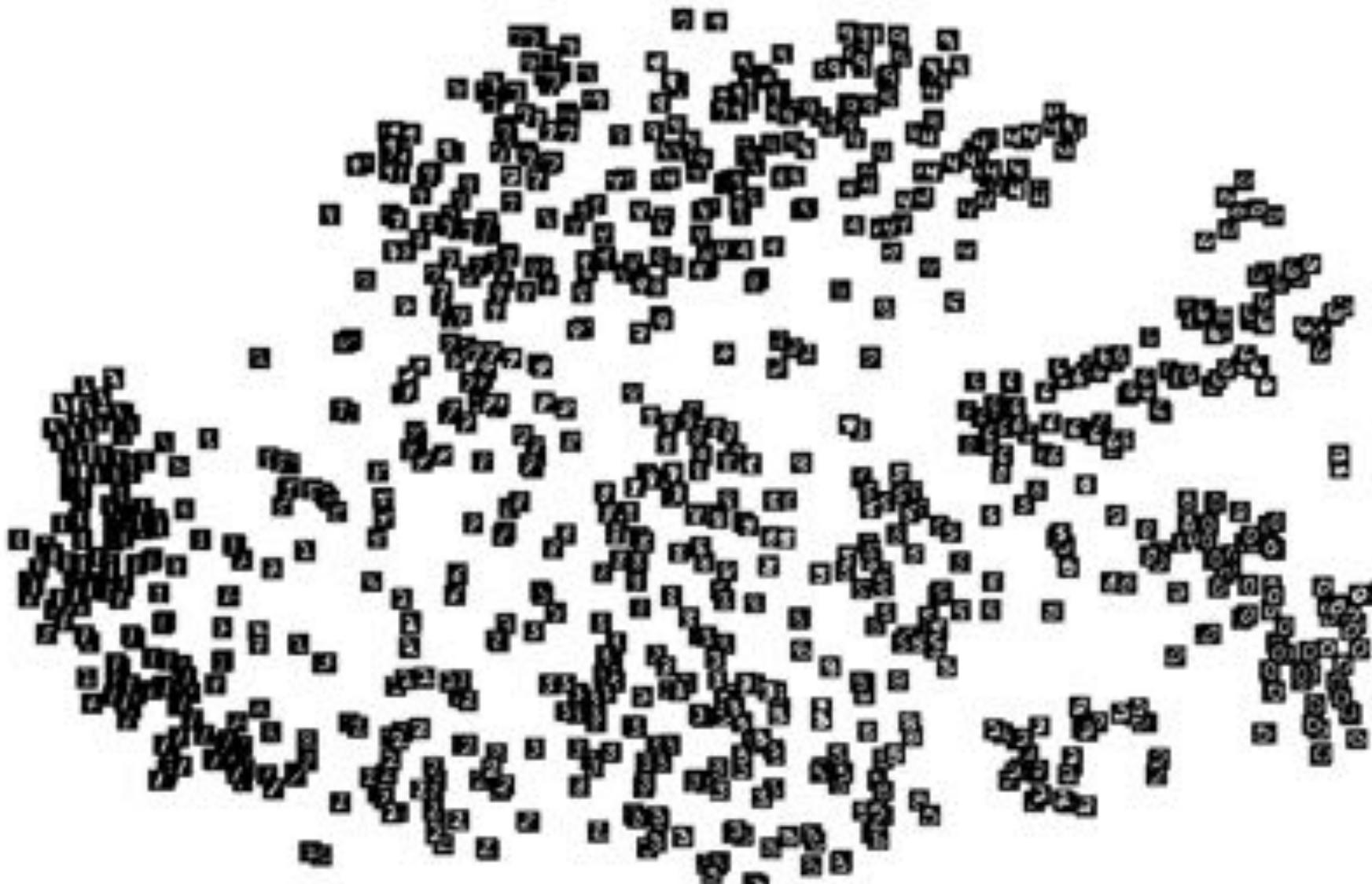
end for

- Sample minibatch of m noise samples $\{\mathbf{z}^{(1)}, \dots, \mathbf{z}^{(m)}\}$ from noise prior $p_g(\mathbf{z})$.
- Update the generator by ascending its stochastic gradient (improved objective):

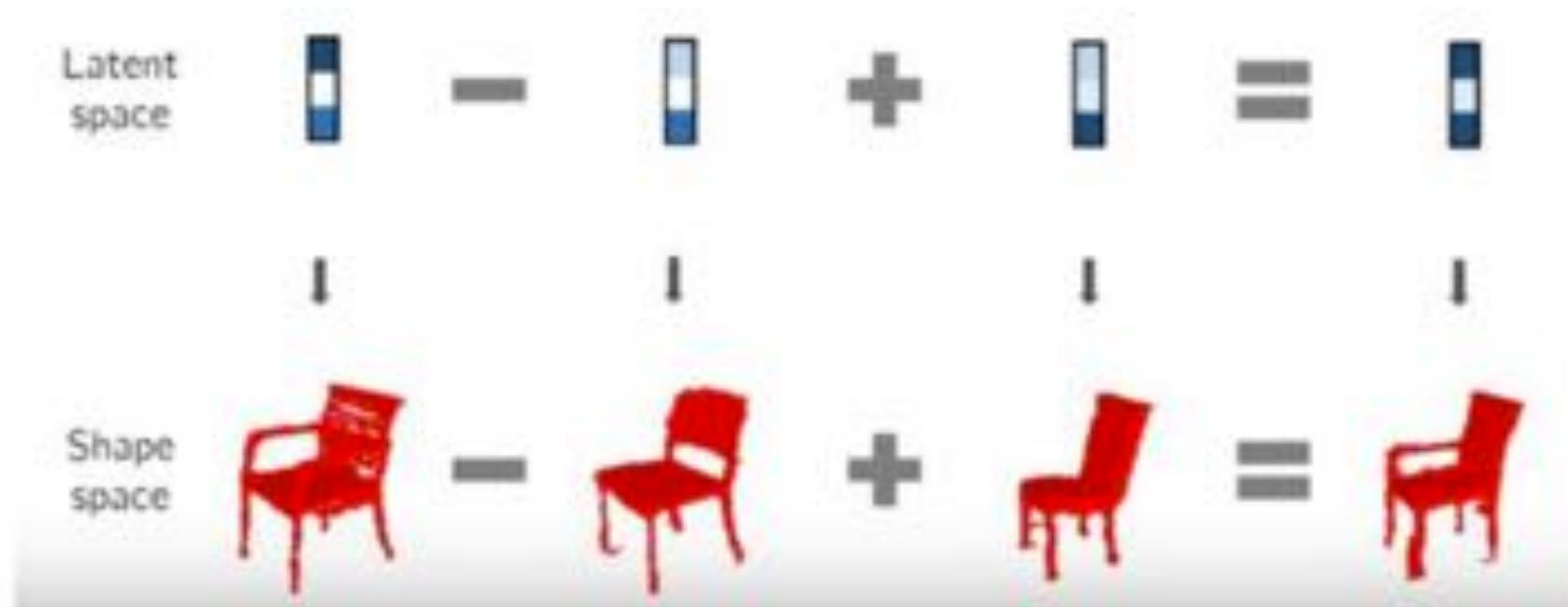
$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^m \log(D_{\theta_d}(G_{\theta_g}(\mathbf{z}^{(i)})))$$

end for

GAN Training algorithm



GAN Training algorithm



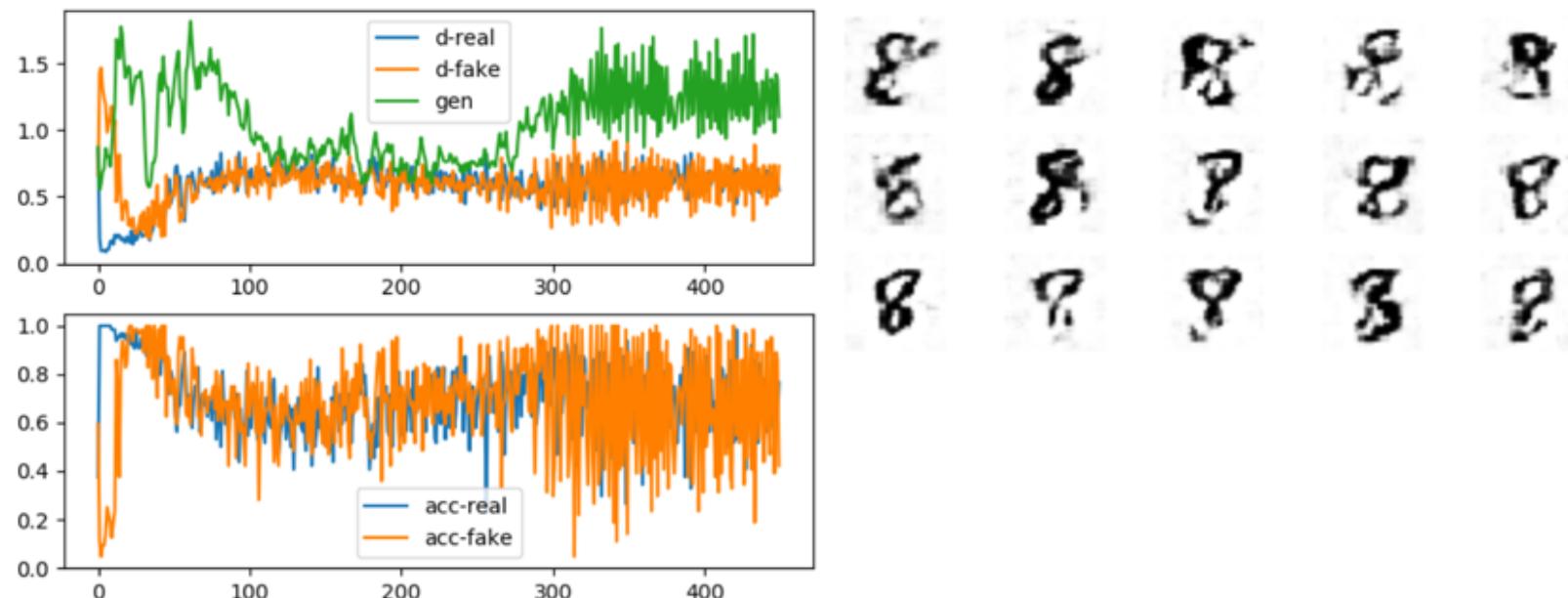
<https://medium.com/intel-student-ambassadors/reducing-mode-collapse-in-gans-using-guided-latent-spaces-36f52a08a668>

GAN Mode Collapse / Convergence Failure

1. GANs are difficult to train

2. Stable GAN

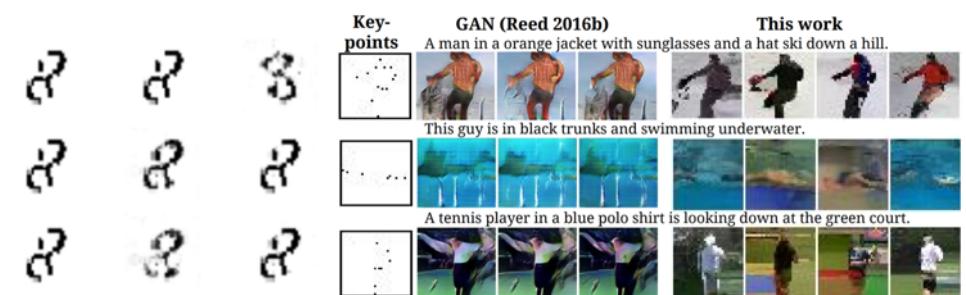
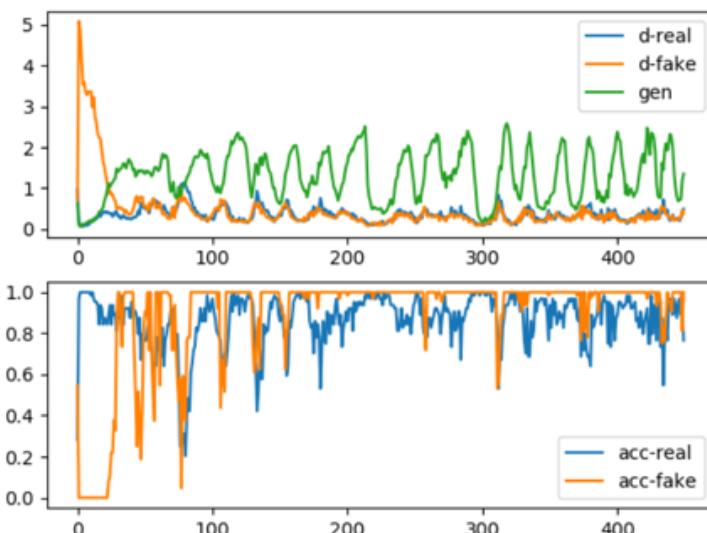
- Discriminator loss 0.5 – 0.8 (acc ~50% for fake images, ~70-80% real images)
- Generator loss higher 1.0 - 2.0 or even higher.



GAN Mode Collapse / Convergence Failure

3. Mode Collapse

- Generator model **only capable of generating one or a small subset of different outcomes** – modes
- Generated **images show low diversity**, with identical image or same small subset of identical images
- **Oscillating loss in generator** (jumps from generating one mode to another model that has different loss)
- Solution (sometimes) **increase latent space**

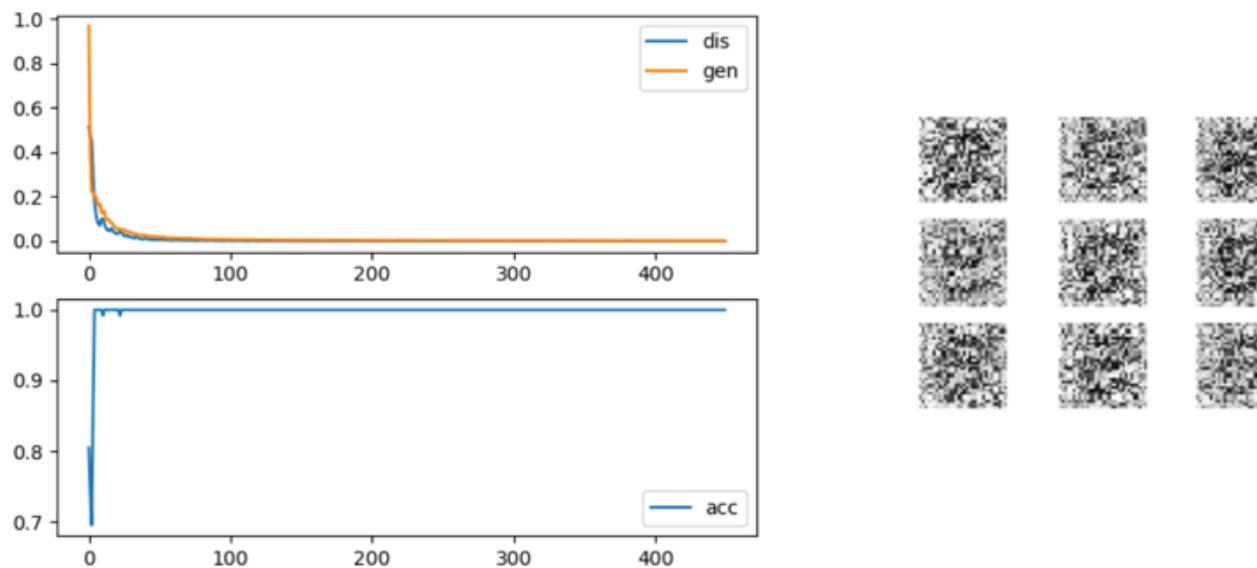


„Mode collapse, also known as the scenario, is a problem that occurs when the generator learns to map several different input z values to the same output point.“ - Goodfellow

GAN Mode Collapse / Convergence Failure

4. Convergence Failure

- NN fails to converge - not finding an (Nash) equilibrium between the discriminator and the generator
- Loss for the discriminator goes to zero, sometimes loss of generator rises
- Generator outputting garbage images



GAN Examples

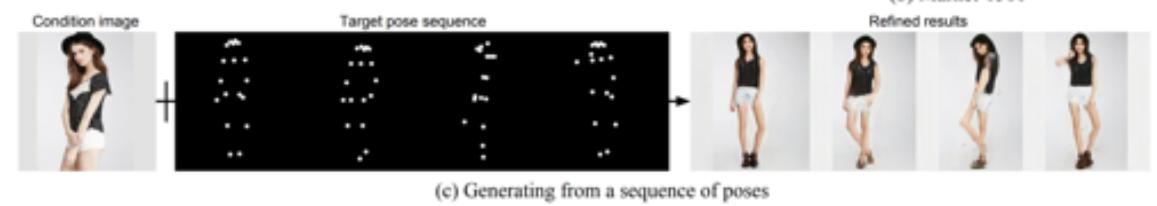
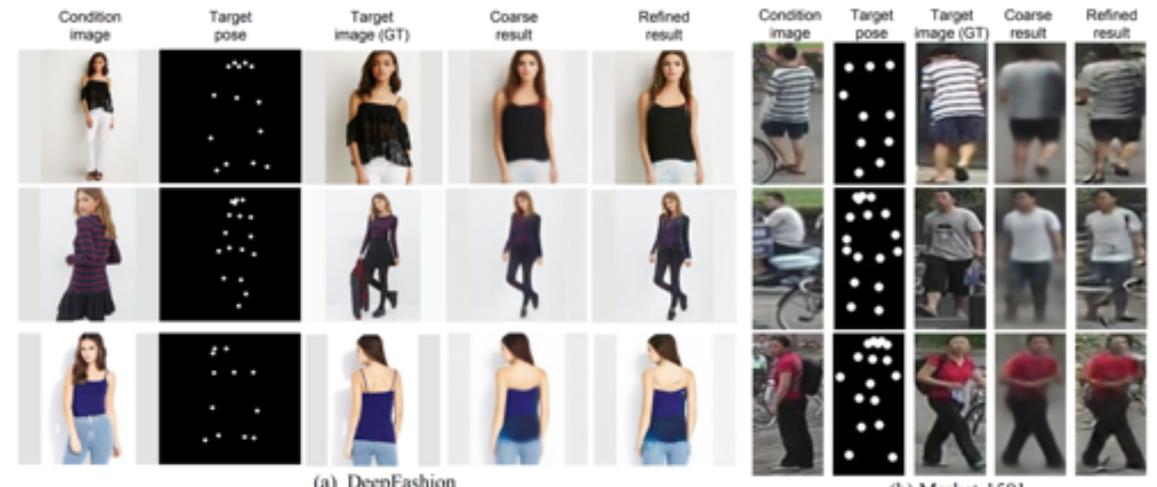


Figure 7: Generated samples

“The GAN Zoo”

- 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
- AFGAN - Amortised MAP Inference for Image Super-resolution
- AL-CDGAN - 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 Categorical GANs
- b-GAN - b-GAN: Unified Framework of Generative Adversarial Networks
- Bayesian GAN - Deep and Hierarchical Implicit Models
- 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
- DITN - Unsupervised Cross-Domain Image Generation
- DCCGAN - 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
- InGAN - Invertible Conditional GANs for Image editing
- ID-CDGAN - Image De-training Using a Conditional Generative Adversarial Network
- Improved-GAN - Improved Techniques for Training GANs
- InfoGAN - InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets
- LADAN - 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>