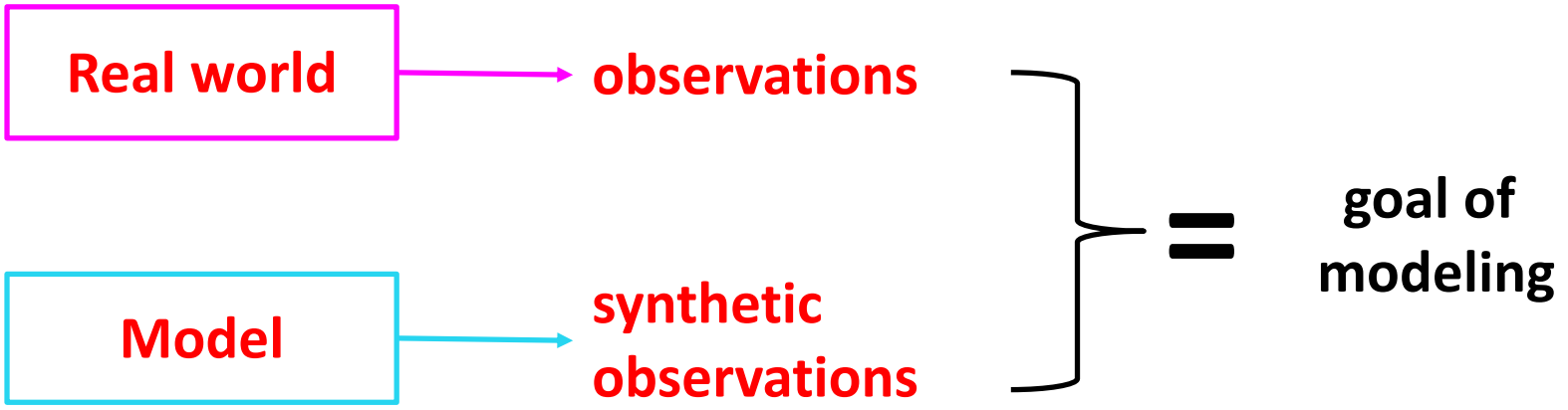
# Generative Adversarial Networks
# GANs

Vahid Tarokh

ECE 685D, Fall 2020

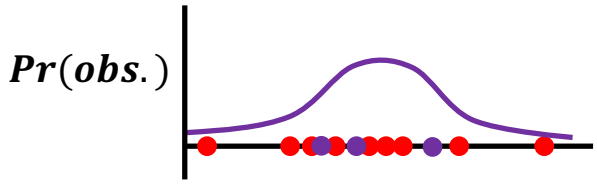
# Probabilistic Generative Models

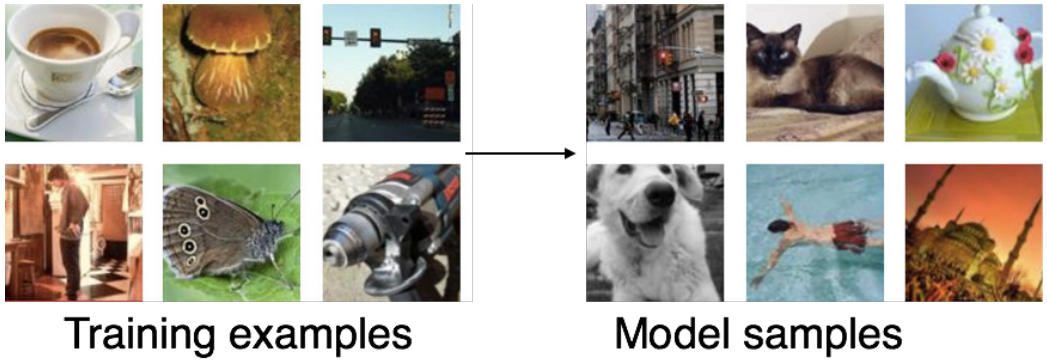



**Density Estimation**

$$Pr(observation) \sim Pr(synthetic\ obs.)$$

# Synthesizing Examples From Probabilistic Generative Model



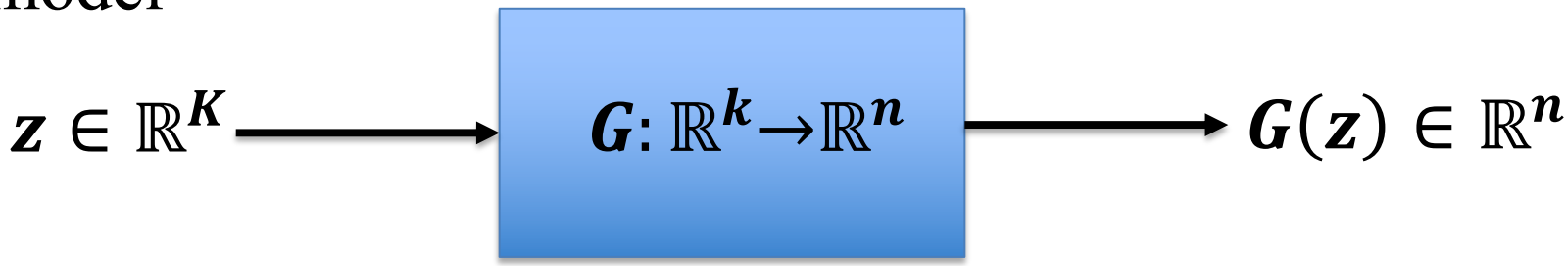

**Goodfellow NIPS tutorial and accompanying paper (arXiv:1701.00160v4 [cs.LG]) provided some figures**

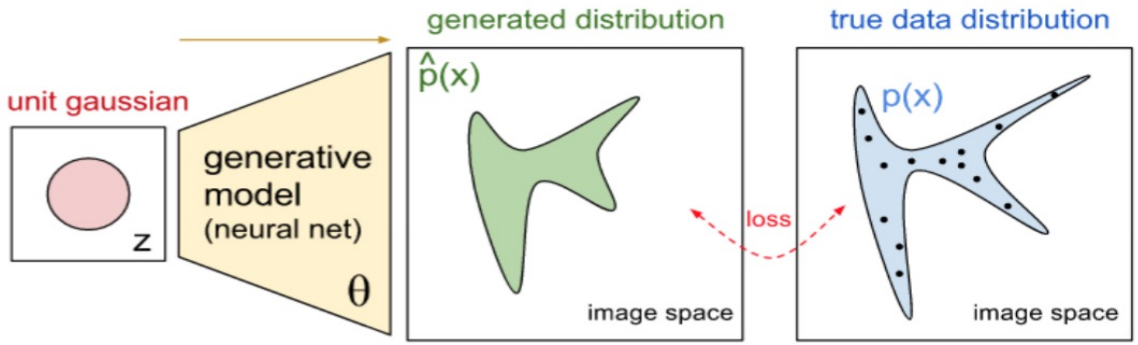
# Density function $Pr_{\text{model}}(x|\theta)$

- Explicit and analytical
  - Gaussian
  - Can sample directly from model
- Explicit and approximate
  - Boltzmann machine
  - VAE
  - Normalizing Flow
  - Autoregressive models
- Implicit
  - GAN
  - Can't estimate probability but can draw from distribution with given probability

# Implicit Generative Models

- Sampling a latent code (vector) $z$ from a fixed and simple distribution such Gaussian
- Passing the drawn code to a differentiable Generator model



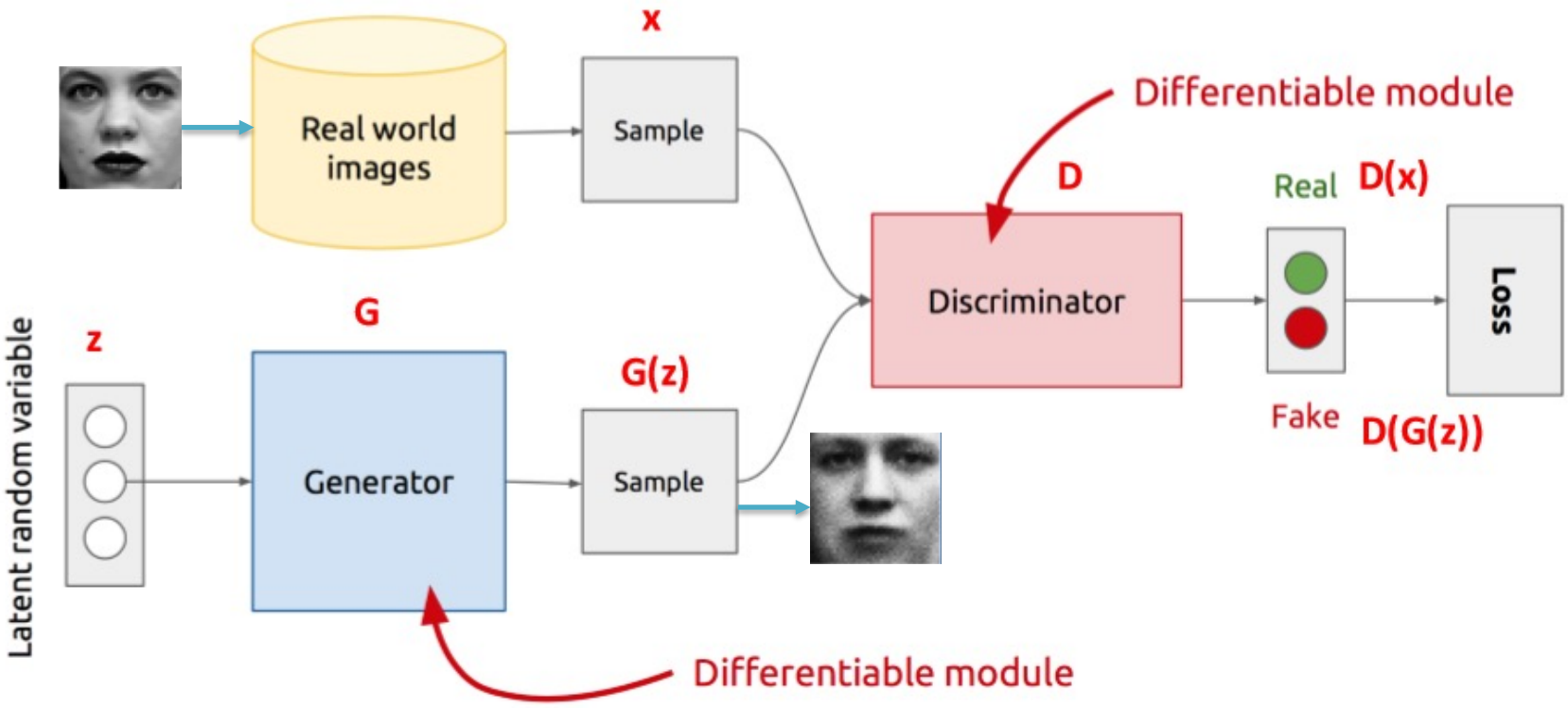

https://blog.openai.com/generative-models/

# Why GANs?

- VAE are known to generate blurry images
  - Not exact inference (Variational inference)
  - Maximizing a lower bound

- GANs revolutionized generative modeling by producing crisp, high-resolution images
  - Sampling (or generation) is straightforward
  - Training doesn't involve Maximum Likelihood estimation
  - Robust to overfitting since Generator never sees the training data
  - Empirically, GANs are good at capturing the modes of the distribution
  - Generate samples through a competition between two-players (a pair of generator and discriminator)
  - It is not known how well they're modeling the distribution
  - No clear way to tell if they are dropping important modes from the distribution

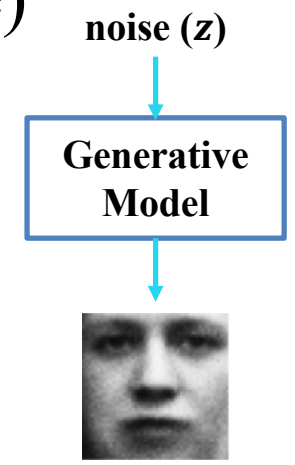
# General GAN's Architecture




- **Generator:** generate fake samples, tries to fool the Discriminator
- **Discriminator**: tries to distinguish between real and fake samples
- Training Generator and Discriminator against each other
- $z$ is some **random noise** (Gaussian/Uniform)

# Generative Model

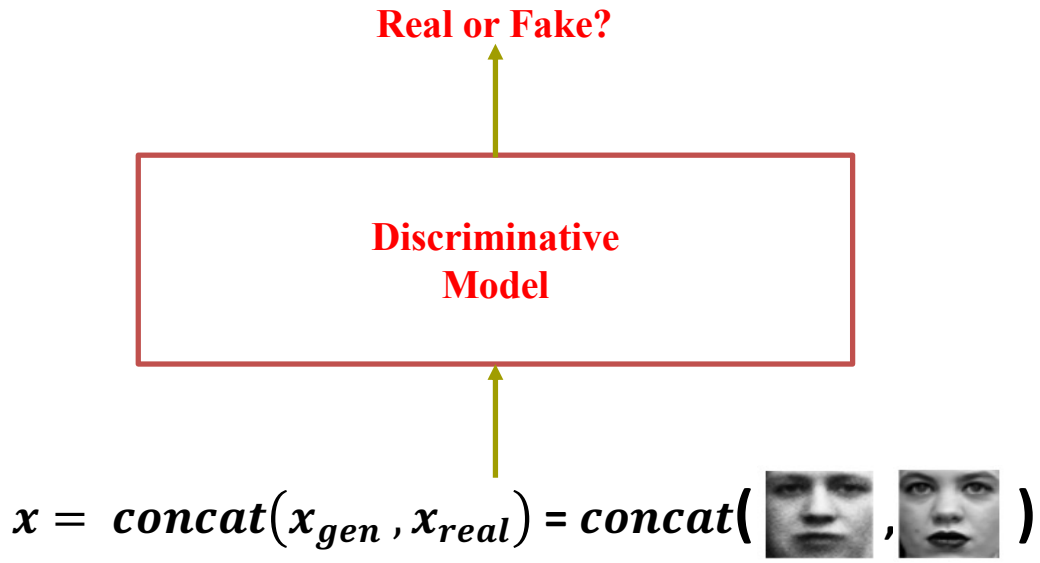
- How to make it generate different samples?
  - Input to model is noise (latent/hidden code)




- Generative model as a neural network
  - Computes $\boldsymbol{x_{gen} = G(z|\theta)}$
  - Differentiable
  - Does not have to be invertible
  - $\boldsymbol{z}$ typically has lower dimension than $\boldsymbol{x_{gen}}$

# Discriminative Model

- Think of it as a critic
  - ➢A good critic can tell real from fake
- Discriminative model as a neural net
  - ➢differentiable
  - ➢computes $0 \leq \boldsymbol{D}(\boldsymbol{x}) \leq 1$, with value 1 if real, 0 if fake

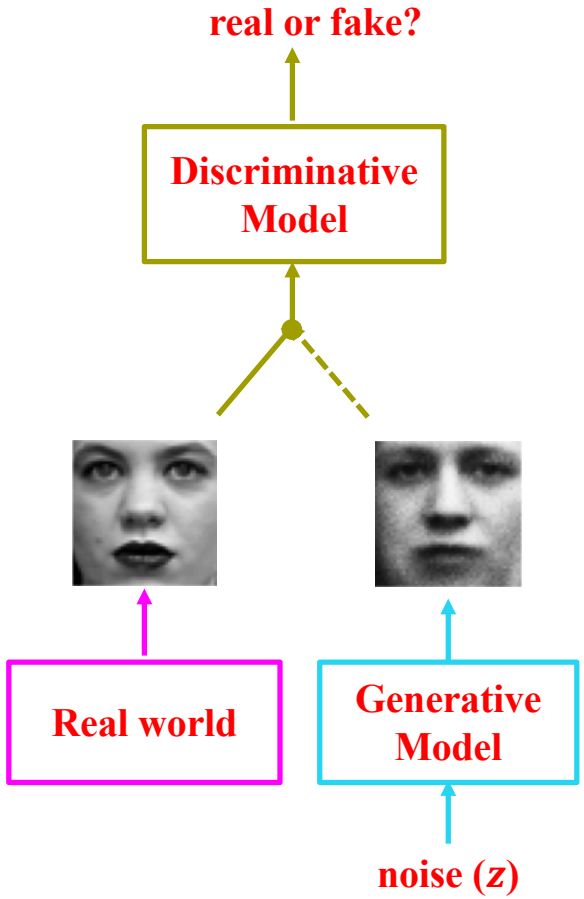


# Training Procedure: Basic Idea

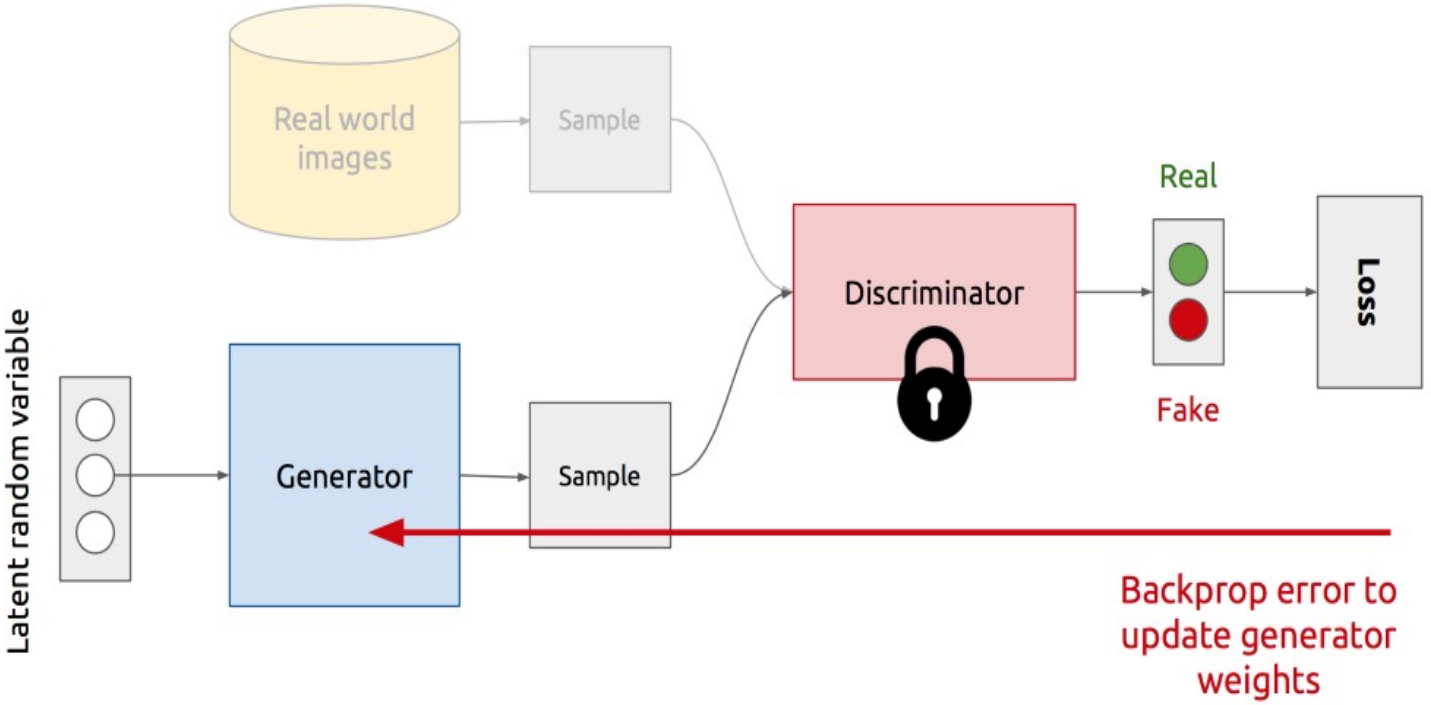
- $G$ tries to fool $D$
- $D$ tries not to be fooled by $G$

- Models are trained **simultaneously**
  - As $G$ gets better, $D$ has a more challenging task
  - As $D$ gets better, $G$ has a more challenging task

- Ultimately, we don't care about the $D$
  - Its role is to force $G$ to work harder

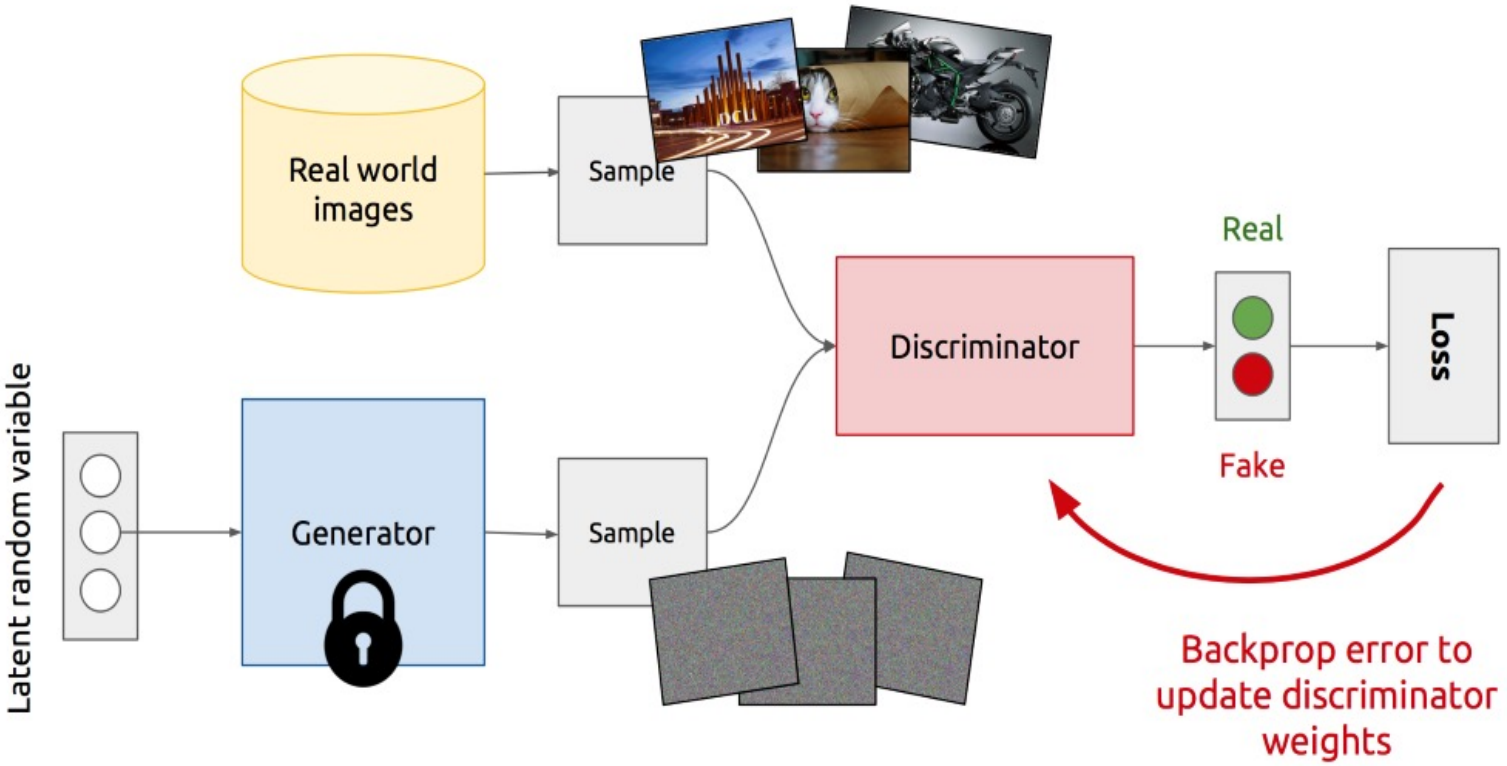


# Training Generator




https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

# Training Discriminator




https://www.slideshare.net/xavigiro/deep-learning-for-computer-vision-generative-models-and-adversarial-training-upc-2016

# Generative models as distance/divergence minimization

- **KL-divergence (asymmetric):**

$$KL(P_{model} \,||\, P_{real}) = \int P_{model}(x) log \frac{P_{model}(x)}{P_{real}(x)} dx$$

$$KL(P_{model} \,||\, P_{real}) = 0 \;\Rightarrow\; P_{model} = P_{real}$$

- **Jensen Shannon divergence (symmetric):**

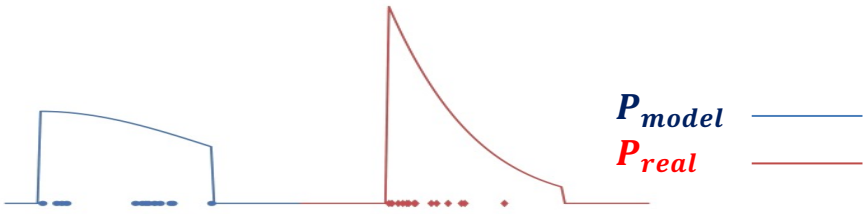
$$JSD(P_{model} \,||\, P_{real}) = \frac{1}{2} KL\left(P_{real} \,||\, \frac{P_{real}+P_{model}}{2}\right) + \frac{1}{2} KL\left(P_{model} \,||\, \frac{P_{real}+P_{model}}{2}\right)$$

$$JSD(P_{model} \,||\, P_{real}) = 0 \;\Rightarrow\; P_{model} = P_{real}$$

- **No signal is learned from KL or JSD divergence if non overlapping support between the data and the model.**

$$KL(P_{model} \,||\, P_{real}) = \infty \qquad JSD(P_{model} \,||\, P_{real}) = log2$$

**NOTE**: The generator is minimizing the *Jensen Shannon divergence* between the real and generated (model) distributions.






https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L9%20-%20UCLxDeepMind%20DL2020.pdf

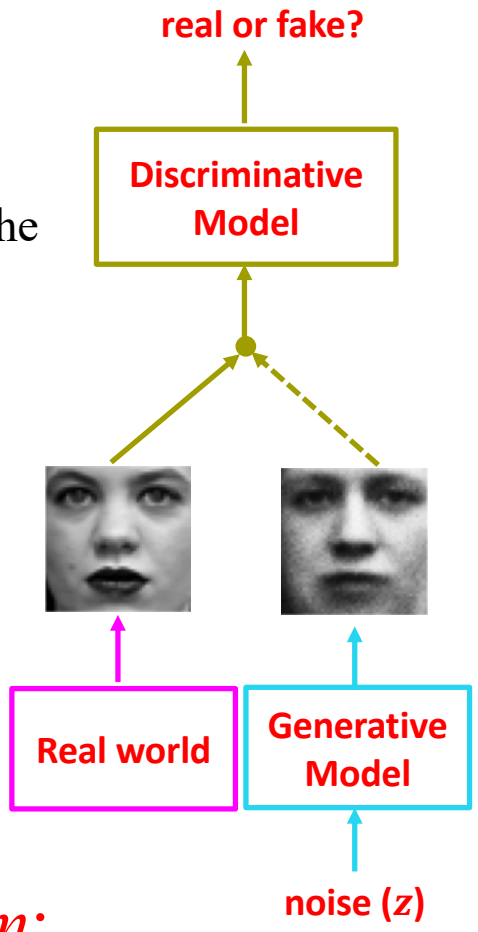
# Loss Functions in Vanilla GAN

- We discuss only the parametric case
- Loss function for $D$
  - Maximizing the likelihood such that the model says 'real' to the samples from the world and 'fake' to the generated samples

$$V(D,G) = \mathbb{E}_{x\sim \text{real}}\, log\, D_{\theta_d}(x) + \mathbb{E}_z \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)$$

log-probability that $D$ correctly predicts real data x are real

log-probability that $D$ correctly predicts generated data $G(z)$ are generated




  - The Discriminator is trying to **maximize** its reward.
  - The Generator is trying to **minimize** Discriminator's reward (or maximize its loss).

- Known as a ***minimax*** *optimization problem*:

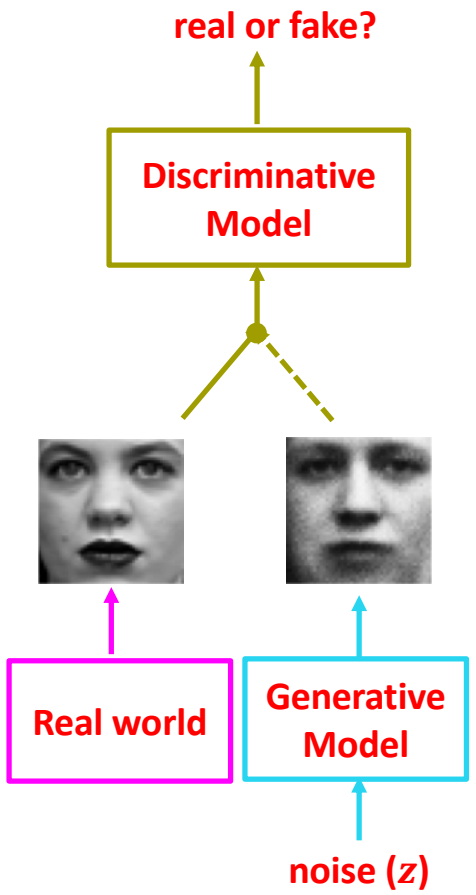
$$min_{\theta_g}\, max_{\theta_d} V(D,G) = \mathbb{E}_{x\sim real}\, log\, D_{\theta_d}(x) + \mathbb{E}_z \log\left(1 - D_{\theta_d}(G_{\theta_g}(z))\right)$$

$\boldsymbol{\theta_d}$: parameters of Discriminator $\boldsymbol{\theta_g}$: parameters of Generator

# Training Procedure

- Train both models simultaneously via stochastic gradient descent using mini-batches consisting of
  - ➢ Some generated samples
  - ➢ Some real-world samples

- $D$ can be trained without altering $G$, and vice versa
  - ➢ May want multiple training epochs of just $D$ so it can stay ahead
  - ➢ May want multiple training epochs of just $G$ because it has a harder task



# Optimization by Alternative Gradient Descent Algorithm

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

**for** number of training iterations **do**

**for** $k$ steps **do**

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

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

**end for**

- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

**Discriminator updates**

**$k = 1$ may result more stability, others use $k > 1$, no best rule.**

**Generator updates**

# Vanishing Gradient

- Saturation problem in $\mathbb{E}_z \log\left(1 - D_{\theta_d}(G_{\theta_g}(z))\right)$
- If the generated sample is bad, the discriminator's prediction **is close to 0**, and the generator's cost is flat

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

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \boxed{\mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D(G(z))\right)\right]}$$

$$\nabla_{\theta_G} V(D,G) = \nabla_{\theta_G} \mathbb{E}_{z\sim q(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

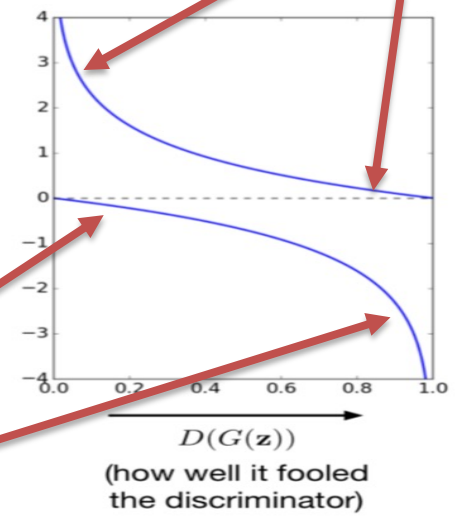
- $\nabla_a \log\left(1-\sigma(a)\right) = \frac{-\nabla_a \sigma(a)}{1-\sigma(a)} = \frac{-\sigma(a)\left(1-\sigma(a)\right)}{1-\sigma(a)} = -\sigma(a) = -D\left(G(z)\right)$
- Gradient goes to 0 if $D$ is confident, i.e. $D\left(G(z)\right) \to 0$
- Minimize $\boxed{-\mathbb{E}_{z\sim q(z)}\left[\log D\left(G(z)\right)\right]}$ for **Generator** instead (keep Discriminator as it is)

➢ Modified generator cost:

$$\mathbb{E}_z - \log\left(D_{\theta_d}(G_{\theta_g}(z))\right)$$

➢ Original minimax cost:

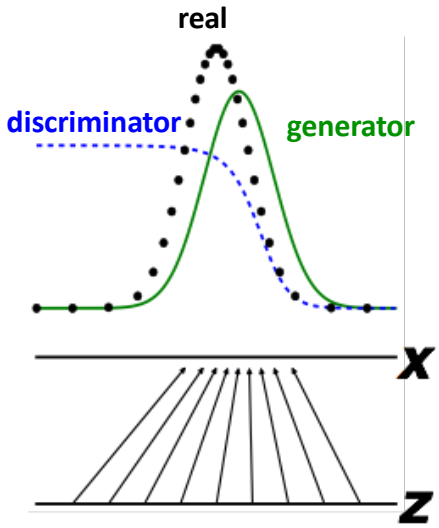
$$\mathbb{E}_z \log\left(1 - D_{\theta_d}(G_{\theta_g}(z))\right)$$



# Global Optimality of $Pr(x|real) = Pr(x|synthesized)$ (Nash Equilibrium)

Recall the minimax loss:

$$V(D,G) = \mathbb{E}_{x\sim real}\, log\, D_{\theta_d}(x) + \mathbb{E}_z \log\left(1 - D(G_{\theta_g}(z))\right)$$




**Proposition.** For $G$ fixed, the optimal discriminator $D_G^*$ is given by

$$D_G^*(x) = \frac{Pr(x|\text{real})}{Pr(x|\text{real}) + Pr(x|\text{synthesized})}$$

**Theorem.** The global minimum of the training criterion, $max_D\, V(D,G)$ is achieved if and only if $Pr(x|real) = Pr(x|synthesized)$.

**NOTE:**

$max_D = max_{\theta_d}$ : maximizing over the parameters of discriminator

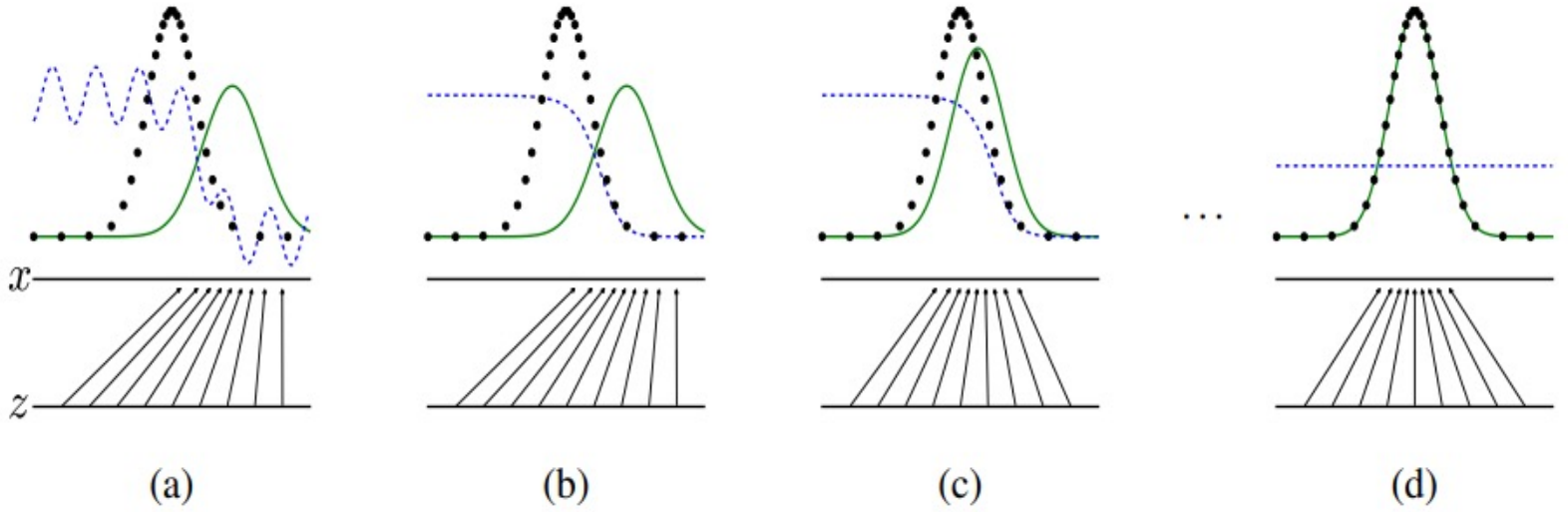
- **Jointly training two networks is difficult.**
- **It can be unstable.**
- **Choosing objectives with better loss landscapes helps training.**



Generative Adversarial networks, Goodfellow et al., 2014

# Towards Nash Equilibrium

**Solid green: generator**
**Dashed blue: discriminator**
**Dotted black: real samples**



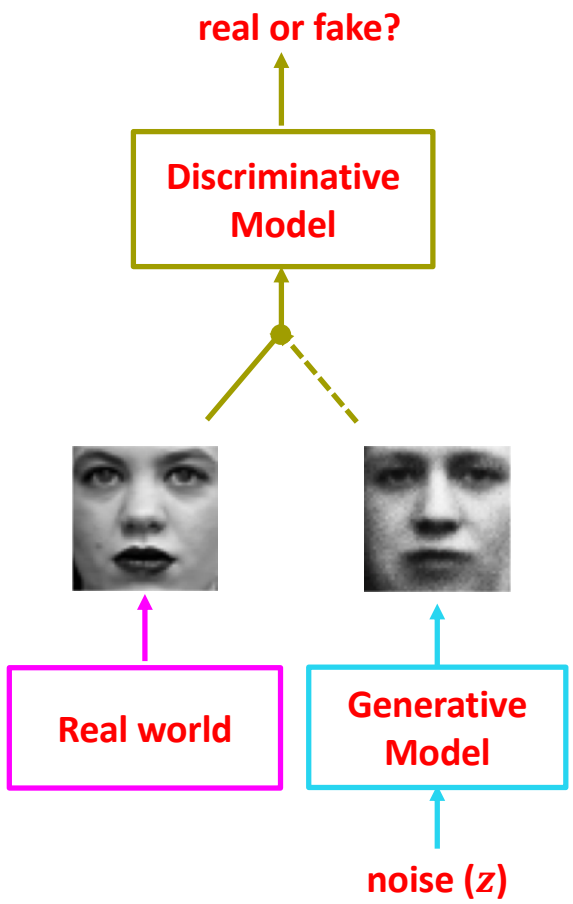

a) Near convergence. $Pr(x|synthesized)$ is similar to $Pr(x|real)$ and $D$ is a partially accurate classifier.
b) $D$ is trained to discriminate samples from data
c) After update to $G$, gradient of $D$ has guided $G(z)$ to flow to regions that are more likely to be classified as data
d) Reaching an equilibrium point at which both distribution cannot improve because as (with enough capacity of $D$ and $G$):

$$\boldsymbol{Pr(x|synthesized) = Pr(x|real)}$$

Goodfellow et al., 2014

# Three Intuitive Reasons that GANs Work

- *G* has a "forced" learning task
  - ➢ It knows when it does good (i.e., fools D) but it is not given a supervised signal
  - ➢ Backprop through *D* provides *G* with a supervised signal; the better *D* is, the better this signal will be
- Can't describe optimum via a single loss
  - ➢ Will there be an equilibrium?
- *D* is seldom fooled
  - ➢ But *G* still learns because it gets a gradient telling it how to change in order to do better the next round



# Optimal and Non-Optimal Discriminator

- **If the discriminator (D) is optimal:**
  - The generator is minimizing the *Jensen Shannon divergence* between the real and generated (model) distributions.

$$\boldsymbol{min_{\theta_g}\, max_{\theta_d} V(D,G) = \mathbb{E}_{x\sim \text{real}}\, log\, D_{\theta_d}(x) + \mathbb{E}_z \log\left(1 - D_{\theta_d}\left(G_{\theta_g}(z)\right)\right)}$$

- **However, D is not optimal in practice**
  - We have access to the limited computational resources.
  - Optimization problems are non-convex.
  - W do not see the real data distribution (only samples)
  - May not learn anything if there is no overlapping support between the data and the model.
    - The theory assumes that $\boldsymbol{P_{model}(x)\ and\ P_{real}(x)}$ are non-zero everywhere.
    - No true if we have data lying on a manifold.
  - The theory assumes that the optimal discriminator is unique. In practice other discriminators can do nearly as good a job: i.e. the discriminator can overfit the data.

# Problems with (Vanilla) GANs

- Estimating probability distribution is Implicit
  - Not straightforward to compute
  - Vanilla GANs may only be good for Sampling/Generation

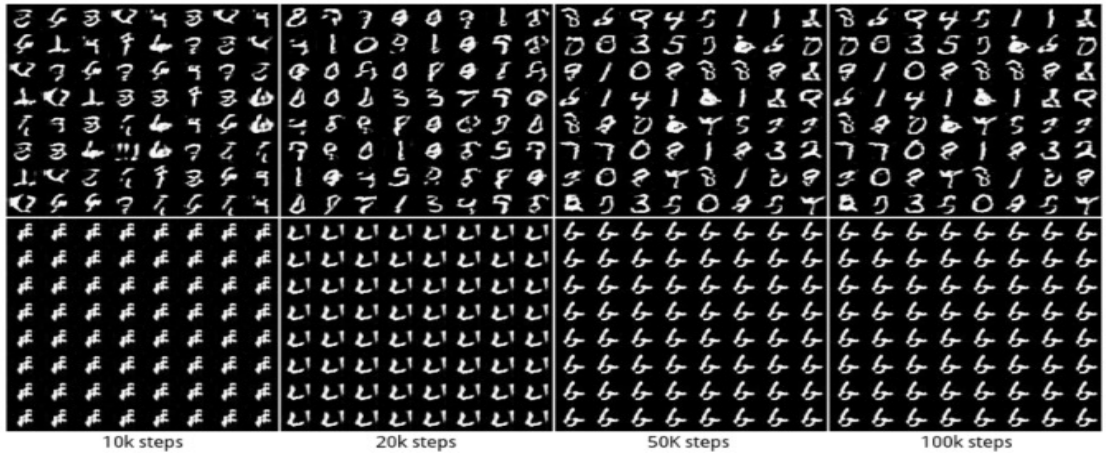
- Training is Hard (Instable)
  - Non-Convergence
  - Mode-Collapse
  - Vanishing Gradient

# Problems with (Vanilla) GANs

- Non-Convergence Issue
  - ❑ Training GANs involves minimax optimization
    - ➢ SGD was not designed to find the Nash equilibrium of a game
    - ➢ May not converge to the Nash equilibrium at all
- Mode-Collapse
  - ❑ Only a few modes of data are generated

No mode Collapse

Mode Collapse




- ❑ Class leakage from a partially trained BigGAN, showing a cross between a tennis ball and perhaps a dog

https://arxiv.org/pdf/1809.11096.pdf

# Mode-Collapse (more example)

Reed, S., et al. ***Generating interpretable images with controllable structure.*** Technical report, 2016. 2, 2016.

# Some Solutions to Partially Prevent Mode-Collapse

- Mini-Batch GANs
  - ➢ Extract features that capture diversity in the mini-batch
  - ➢ Feed those features to the discriminator along with the image

- Supervision with labels
  - ➢ Similar to conditional GANs

# Some Solutions to Partially Prevent Mode-Collapse

- Changing the loss function
  - ➢ Some choices have better behavior (more stable) during training
  - ➢ Some choices will modify the latent space
    - WGAN (Wasserstein-GAN)
    - f-GAN
    - MMD-GAN
    - Fisher-GAN
- Constraining the discriminator model during the training process

https://arxiv.org/pdf/1611.02163.pdf
https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2015/slides/lec13.GAN.pdf

# WGAN (Wasserstein GAN)

- If data comes from a low-dimensional manifold of a high dimensional space
  - ➢ Negligible intersection between the model's manifold
    - KL divergence is undefined or infinite

- The loss function and gradients may not be continuous and well behaved

- The Earth Mover's Distance is well defined:
  - ➢ Minimum transportation cost for making one pile of dirt (pdf/pmf) look like the other

# WGAN (Wasserstein GAN)

- **The Earth-Mover (EM) distance or Wasserstein-1**

$$inf_{\gamma \in \Pi(\mathrm{P_{real}}, \mathrm{P}_{model})} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|_1$$

- $\Pi(P_{real}, P_{model})$ denotes the set of all joint distributions $\gamma$(x, y) whose marginals are respectively $P_{real}$ and $P_{model}$.

- Intuitively, $\gamma$(x, y) indicates how much "mass" must be transported from $x$ to $y$ in order to transform the distributions$P_{real}$into the distribution $P_{model}$. https://yoo2080.wordpress.com/2015/04/09/introduction-to-wasserstein-metric-earth-movers-distance/

- **Using Kantorovich-Rubinstein duality**

$$sup_{||f||_L \leq 1} \mathbb{E}_{x\sim P_{real}} f(x) - \mathbb{E}_{x\sim P_{model}} f(x)$$

- The EM distance then is the "cost" of the optimal transport plan.

  - *Sup* is taken over w.r.t. 1-Lipschitz continuous
  - $f$ is a discriminator belonging to the family of 1-Lipschitz

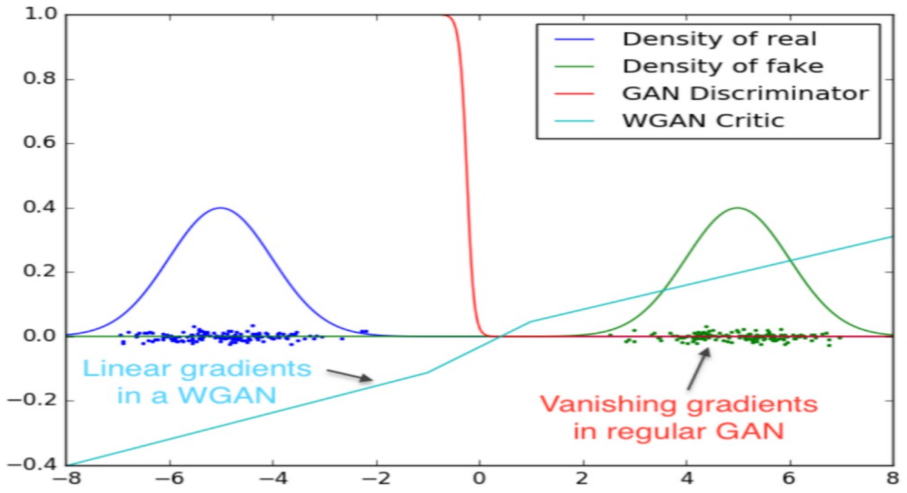
Arjovsky et al., "WGAN", 2017.

# WGAN (Wasserstein GAN)

- **Training loss:**

$$\min_{G} \sup_{\|D\|_L \le 1} \mathbb{E}_{x \sim P_{real}} D(x) - \mathbb{E}_z D(G(z))$$

- The Discriminator is trained for multiple steps before the Generator is updated
- To ensure $D(x)$ has the Lipschitz continuity (i.e., $||D||_L \le 1$) ***gradient-clipping*** is used in the discriminator.
- It is said that WGAN resolves many training issues such as mode collapse.




https://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Fall.2015/slides/lec13.GAN.pdf

# Enforcing the Lipschitz Constraint

1. **WGAN + Gradient-clipping**
2. **WGAN + Gradient Norm Penalty Regularizer**

$$\mathbb{E}_{\tilde{x}\sim P_{real}} D(\tilde{x}) - \mathbb{E}_{x\sim P_{model}} D(x) + \lambda \mathbb{E}_{P_{\hat{X}}} \left( \left\| \nabla_{\hat{X}} \mathrm{D}(\hat{X}) \right\|_2 - 1 \right)^2$$

- Gradient Norm Penalty Regularizer forces the Lipschitz Constraint.
- $P_{\hat{X}}$ samples uniformly along straight lines between pairs of points sampled from the data distribution $P_{real}$ and the generator distribution $P_{model}$.
- This is motivated by the fact that the optimal critic (Discriminator) contains straight lines with gradient norm 1 connecting coupled points from $P_{\hat{X}}$ and $P_{model}$.
- Enforcing the unit gradient norm constraint everywhere is intractable.
- Enforcing only along these straight lines is sufficient and experimentally has good performance.

https://arxiv.org/pdf/1704.00028.pdf

# Enforcing the Lipschitz Constraint

## 3. WGAN + Spectral Normalization

- The weights of the generator are normalized using spectral normalization.
- Zhang et al. (2018) (SAGAN) ) found that employing Spectral Normalization in Generator improves stability, allowing for fewer Discriminator steps per iteration.
- Controlling the Lipschitz constant of the Discriminator function D (with $L$ layers) by constraining the spectral norm of each layer $g: h_{in} \to h_{out}$
- Define the Lipschitz norm of *layer g* by $\|g\|_{Lip} = \sup_h \sigma(\nabla_h(g))$, where $\sigma(A)$ denotes the spectral norm of the matrix $A$.
- Using the facts that $\|g_1 \circ g_2\|_{Lip} \leq \|g_1\|_{Lip} \, \|g_2\|_{Lip}$, and for many activation functions (e.g., Relu), the Lipschitz constant is 1, we have:

$$\|G\|_{Lip} \leq \prod_{l=0}^{L} \sigma(W^l),$$

- $W^l$ denotes the weights of the $l^{th}$ layer
- By normalizing the weights of each layer, $\overline{w_{SN}^l} = \frac{W}{\sigma(W^l)}$, we guarantee that $\|G\|_{Lip} \leq 1$.

# Enforcing the Lipschitz Constraint

## 3. WGAN + Spectral Normalization

- Very computationally cheap even in comparison to the calculation of the forward and backward propagations on neural networks

**Algorithm 1** SGD with spectral normalization

- Initialize $\tilde{\boldsymbol{u}}_l \in \mathcal{R}^{d_l}$ for $l = 1, \ldots, L$ with a random vector (sampled from isotropic distribution).
- For each update and each layer $l$:
  1. Apply power iteration method to a unnormalized weight $W^l$:

$$\tilde{\boldsymbol{v}}_l \leftarrow (W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l / \|(W^l)^{\mathrm{T}} \tilde{\boldsymbol{u}}_l\|_2$$
$$\tilde{\boldsymbol{u}}_l \leftarrow W^l \tilde{\boldsymbol{v}}_l / \|W^l \tilde{\boldsymbol{v}}_l\|_2$$

  2. Calculate $\bar{W}_{\mathrm{SN}}$ with the spectral norm:

$$\bar{W}^l_{\mathrm{SN}}(W^l) = W^l / \sigma(W^l), \text{ where } \sigma(W^l) = \tilde{\boldsymbol{u}}_l^{\mathrm{T}} W^l \tilde{\boldsymbol{v}}_l$$

  3. Update $W^l$ with SGD on mini-batch dataset $\mathcal{D}_M$ with a learning rate $\alpha$:

$$W^l \leftarrow W^l - \alpha \nabla_{W^l} \ell(\bar{W}^l_{\mathrm{SN}}(W^l), \mathcal{D}_M)$$

https://arxiv.org/pdf/1802.05957.pdf

# Evaluation of Generated Samples

- **Inception Score (IS)**
  - Intuitively, the conditional label distribution of samples containing meaningful objects should have **low entropy**, and the variability of the samples should be high:

$$IS = e^{KL(P(y|x)||P(y))}$$

    - Applying trained Inception-V3 model with ImageNet to every generated image to get $P(y|x)$
    - **The larger the IS, the better generated image**
    - **Advantage**: Well correlated with scores from human annotators
    - **Disadvantage**: Insensitivity to the prior distribution over labels and not being a proper distance

https://openreview.net/pdf/6e08ef84b70c2be77e96f8e15ebfb24506482a19.pdf

# Evaluation of Generated Samples

- **Fréchet Inception Distance (FID)**
  - Comparing the statistics of generated samples to real samples
  - Samples from $P_{real}$ and $P_{model}$ are first embedded into a feature space (the 2048-dimensional activations of the Inception-v3 pool3 layer).
  - By assuming the multivariate Gaussian distribution on the embedded data, the mean and covariance are estimated:

$$X_{real}^{em} \sim \mathcal{N}(\mu_1, \Sigma_1) \qquad X_{model}^{em} \sim \mathcal{N}(\mu_2, \Sigma_2)$$

  - The Fréchet distance between the two Gaussians is given by:

$$FID = \|\mu_1 - \mu_2\|_2^2 + Tr\left(\Sigma_1 + \Sigma_2 - 2(\Sigma_1\Sigma_2)^{\frac{1}{2}}\right)$$

  - **The lower FID, the more similar real and 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
- 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

# Different GANs

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods.

- Most modern architectures are based on the Deep Convolutional GAN (DCGAN), where the generator and discriminator are both conv nets.

# GAN Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

# GAN Samples

Bedrooms:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

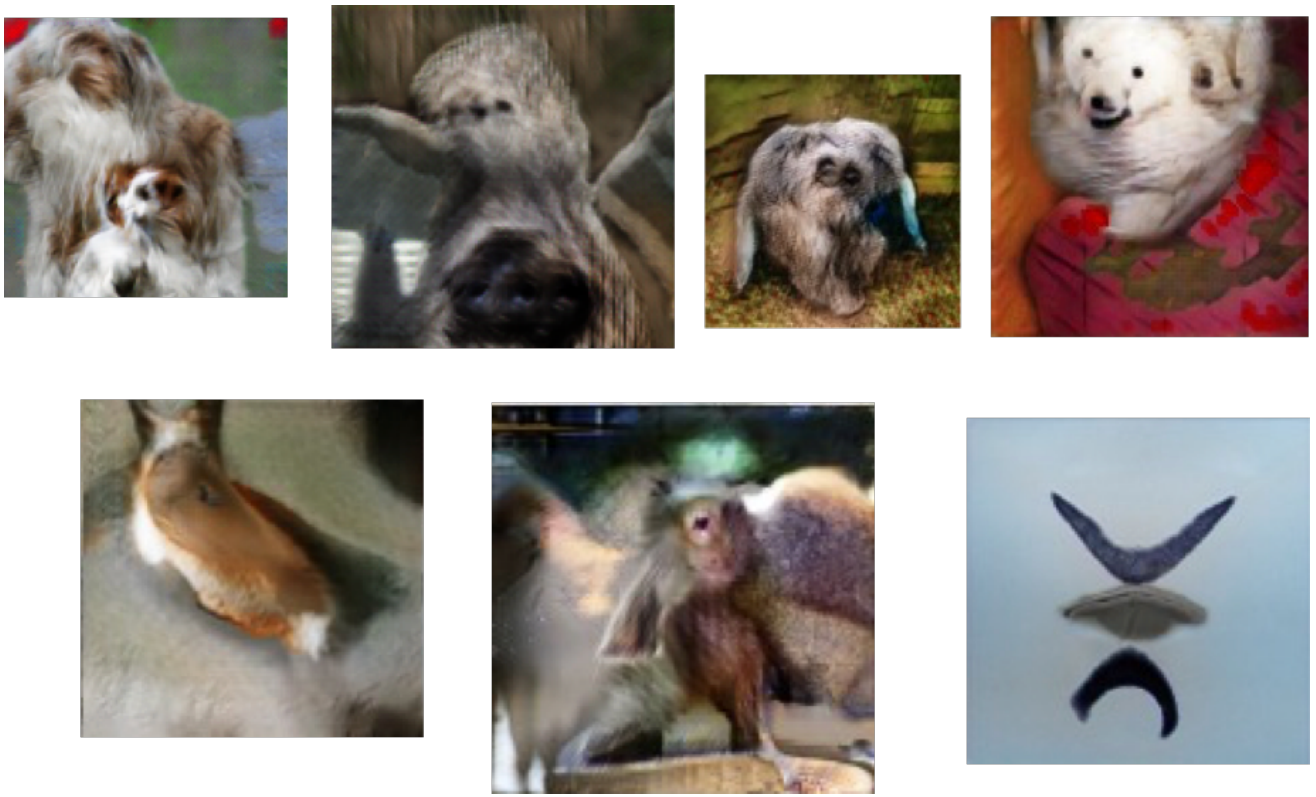
# GAN Samples

Objects:


Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

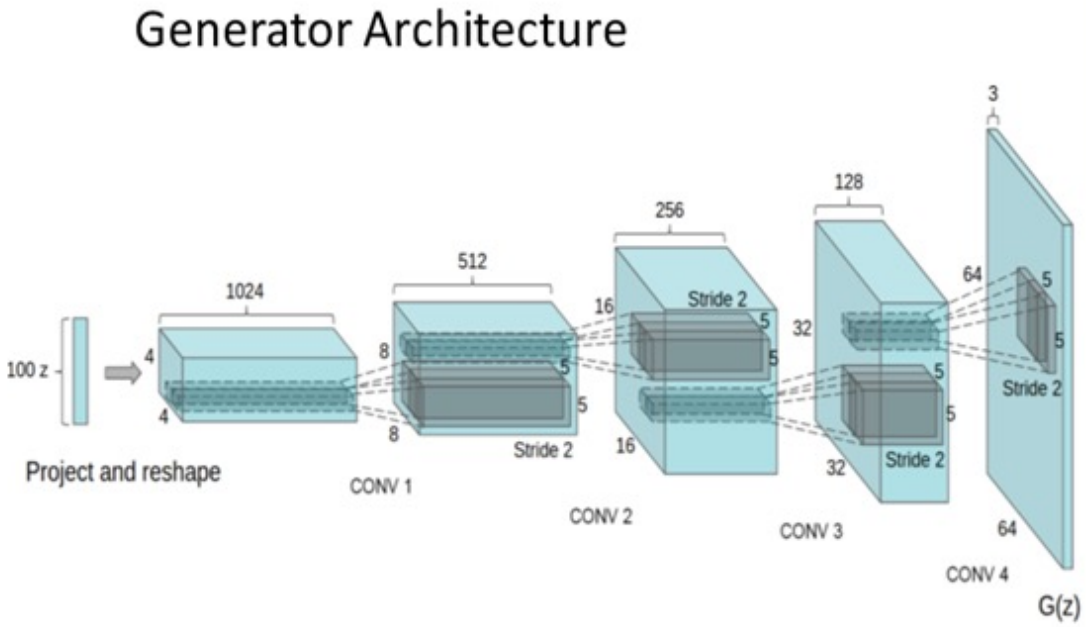
# GAN Samples

Cherry Picked:



Goodfellow (2017)

# Deep convolutional GANs (DCGAN) (Radford et al., 2015)



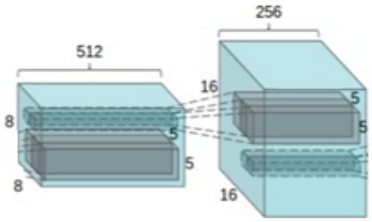

**Key ideas:**

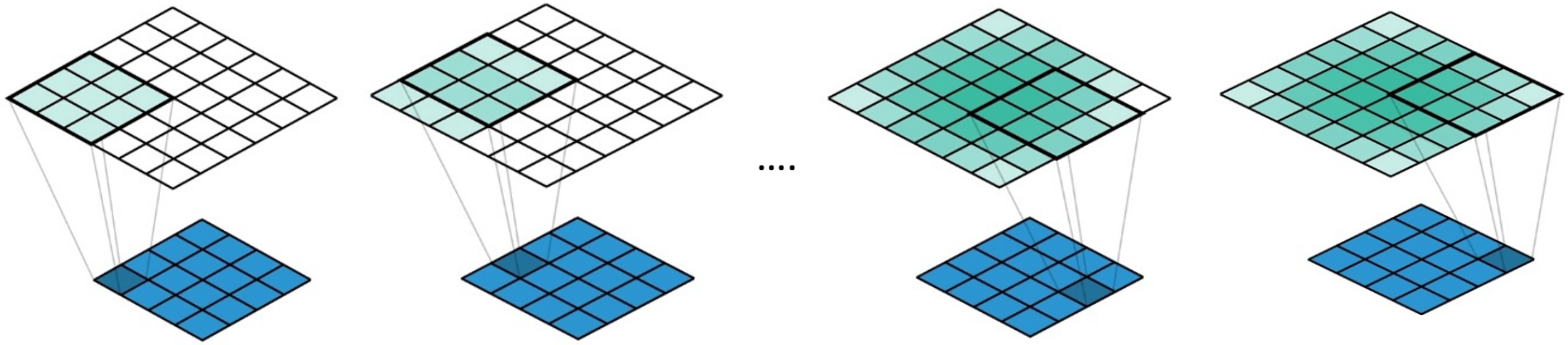
- Replace FC hidden layers with Convolutions
  - **Generator:** Fractional-Strided convolutions
- Use Batch Normalization after each layer
- **Inside Generator**
  - Use ReLU for hidden layers
  - Use Tanh for the output layer

- Batch normalization is important here, apparently.
- **Fractional-Strided convolutions** is one type of *Convolutional Transpose* operation (see next slide).

# Deconvolutional GANs (DCGAN)

- Convolutional transpose operation
- It is like upsampling



Conv2D Transpose with $3x3$ kernel (not seen explicitly) applied to a $4x4$ input to give a $6x6$ output.

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | | | | | | | | | | | | | |
| 2 | | | Input | | | | | | | Kernel | | | | | Output | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | 1 | 2 | 3 | 3 | 2 | 1 | |
| 5 | | | 1 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | | | 2 | 4 | 6 | 6 | 4 | 2 | |
| 6 | | | 1 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | | | 3 | 6 | 9 | 9 | 6 | 3 | |
| 7 | | | 1 | 1 | 1 | 1 | | | | 1 | 1 | 1 | | | | 3 | 6 | 9 | 9 | 6 | 3 | |
| 8 | | | 1 | 1 | 1 | 1 | | | | | | | | | | 2 | 4 | 6 | 6 | 4 | 2 | |
| 9 | | | | | | | | | | | | | | | | 1 | 2 | 3 | 3 | 2 | 1 | |
| 10 | | | | | | | | | | | | | | | | | | | | | | |

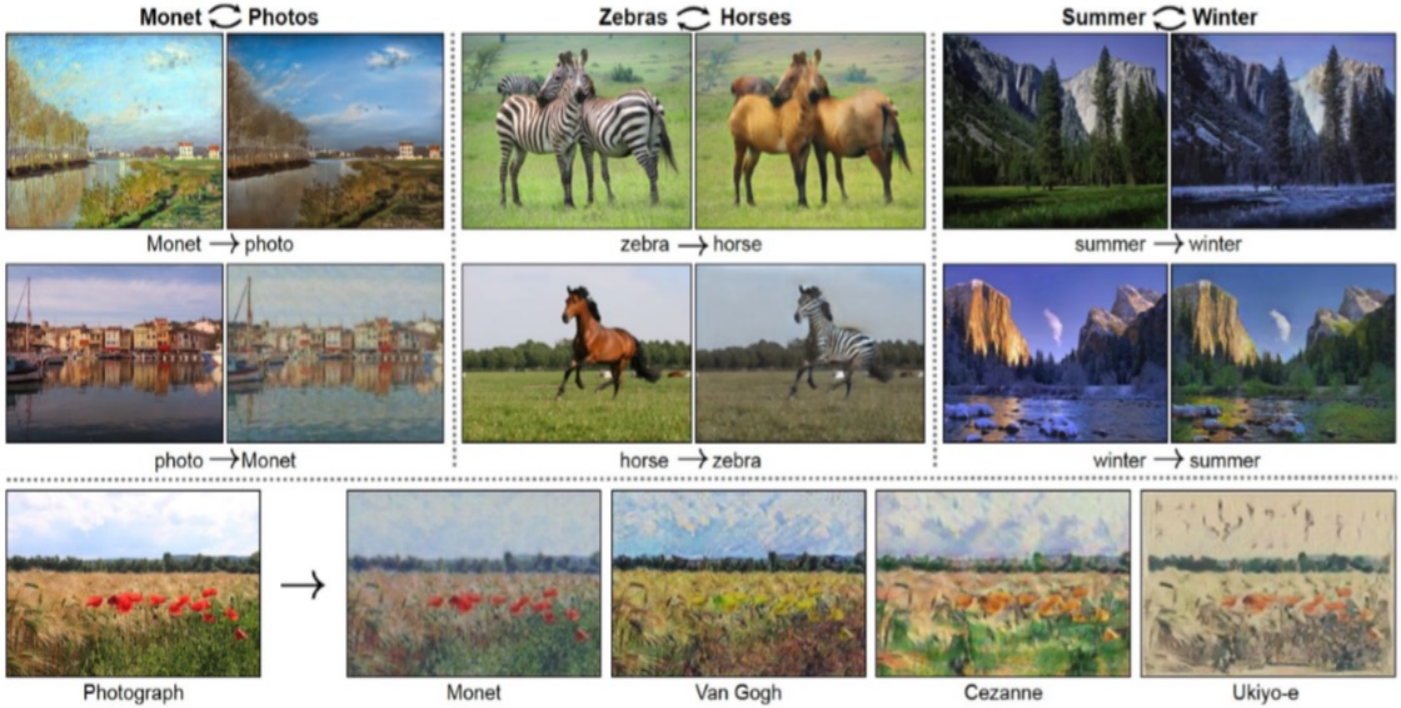
Conv2D Transpose with $3x3$ kernel (seen explicitly) applied to a $4x4$ input to give a $6x6$ output.



https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8

# CycleGAN

- Style transfer problem: change the style of an image while preserving the content



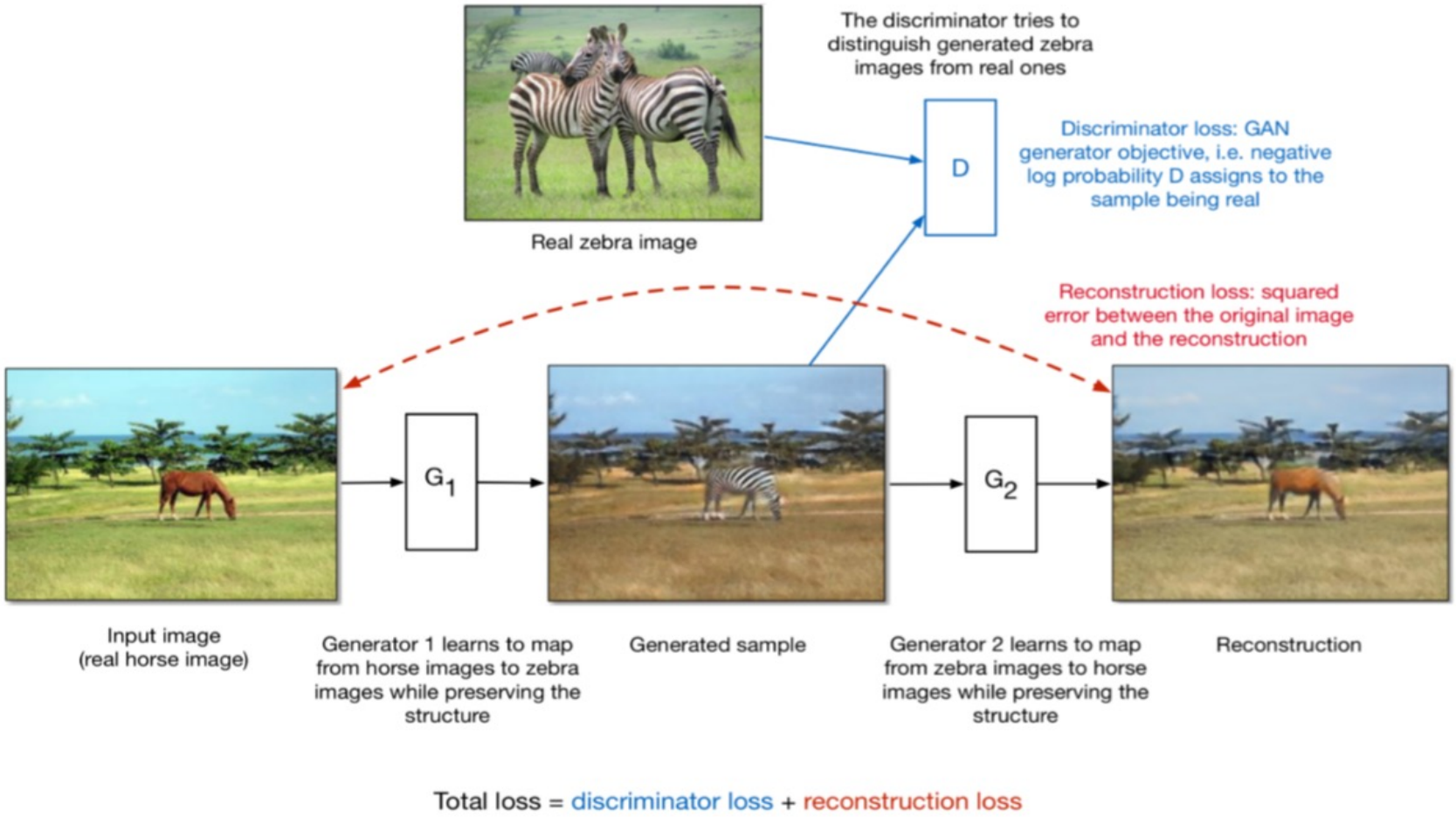

Data: Two unrelated collections of images, one for each style

https://junyanz.github.io/CycleGAN/

# CycleGAN

- This is not a supervised learning problem as we do not have paired data (same content in both styles).
  - ➢ This is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
- Train two different generator nets to go from style 1 to style 2, and vice versa.
  - ➢ Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - ➢ Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image.

# CycleGAN



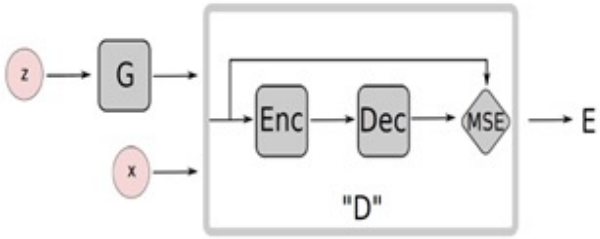

For more details see https://junyanz.github.io/CycleGAN/

# Energy-Based GAN

- Modified game plans
  - **Generator** will try to generate samples with low values
  - **Discriminator** will try to assign high scores to fake values

$$D(x) = ||Dec(Enc(x)) - x||_{MSE}$$

- Use AutoEncoder inside the Discriminator
- Use Mean-Squared Reconstruction error as $D(x)$
  - High Reconstruction Error for Fake samples
  - Low Reconstruction Error for Real samples




Zhao, Junbo, Michael Mathieu, and Yann LeCun. **"Energy-based generative adversarial network."** arXiv preprint arXiv:1609.03126 (2016)

- Views the discriminator as an energy function that attributes
  - Low energies to the regions near the data manifold
  - Higher energies to other regions
- Stable behavior than regular GANs during training
- Discriminator cost function composes of two goals:
  - A good autoencoder: The reconstruction cost $\boldsymbol{D(x)}$ for real images to be low
  - A good critic: Penalizing the discriminator if the reconstruction error for generated images drops below some value $\boldsymbol{m>0}$.
  - Function $\boldsymbol{D}$: $\boldsymbol{u \rightarrow D(u)}$ is a real valued function not a binary one (the original GAN)

$$\boldsymbol{L_D(x,z) = ||Dec(Enc(x) - x)||_2^2 + max\left(0, m - D\left(G(z)\right)\right)}$$

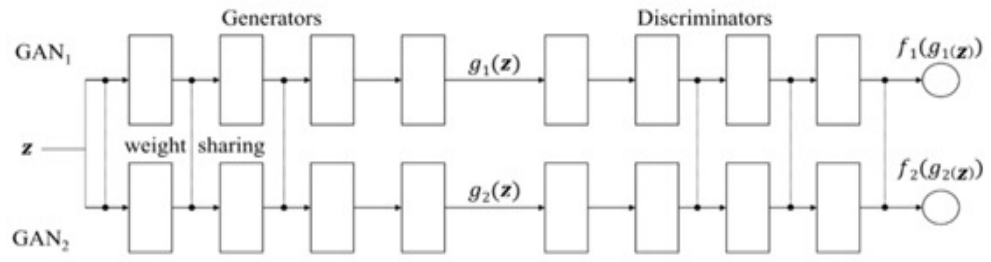
$$\boldsymbol{L_G(z) = D\left(G(z)\right)}$$

https://medium.com/@jonathan_hui/gan-energy-based-gan-ebgan-boundary-equilibrium-gan-began-4662cceb7824

# Coupled GAN

- Learning a joint distribution of multi-domain images
- Using GANs to learn the joint distribution with samples drawn from the marginal distributions
- Applications in domain adaptation and image translation

• Architecture




Weight-sharing constraints the network to learn a *joint distribution* without corresponding supervision.

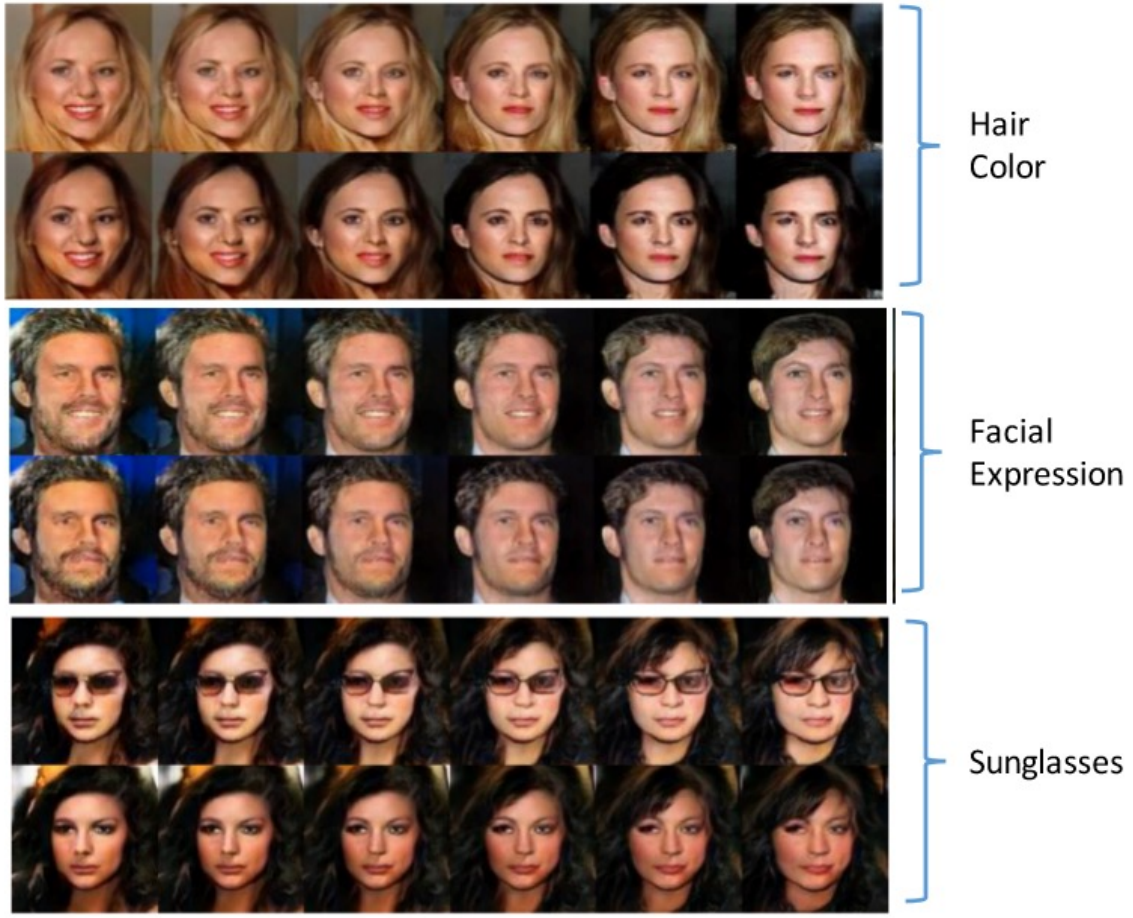
Liu, Ming-Yu, and Oncel Tuzel. "Coupled generative adversarial networks". NIPS (2016).

- Two teams and each team has two players
- The generative models form a team and work together for synthesizing a pair of images in two different domains for confusing the discriminative models
- The discriminative models try to differentiate images drawn from the training data distribution in the respective domains from those drawn from the respective generative models.
- Learning requires training samples drawn from the marginal distributions, $P_{x1}$ and $P_{x2}$ . It does not rely on samples drawn from the joint distribution
- Learn a joint distribution of images in the two domains

# Coupled GAN

## Coupled GANs

- Some examples of generating facial images across different feature domains.
- Corresponding images in a column are generate from the same latent code $z$

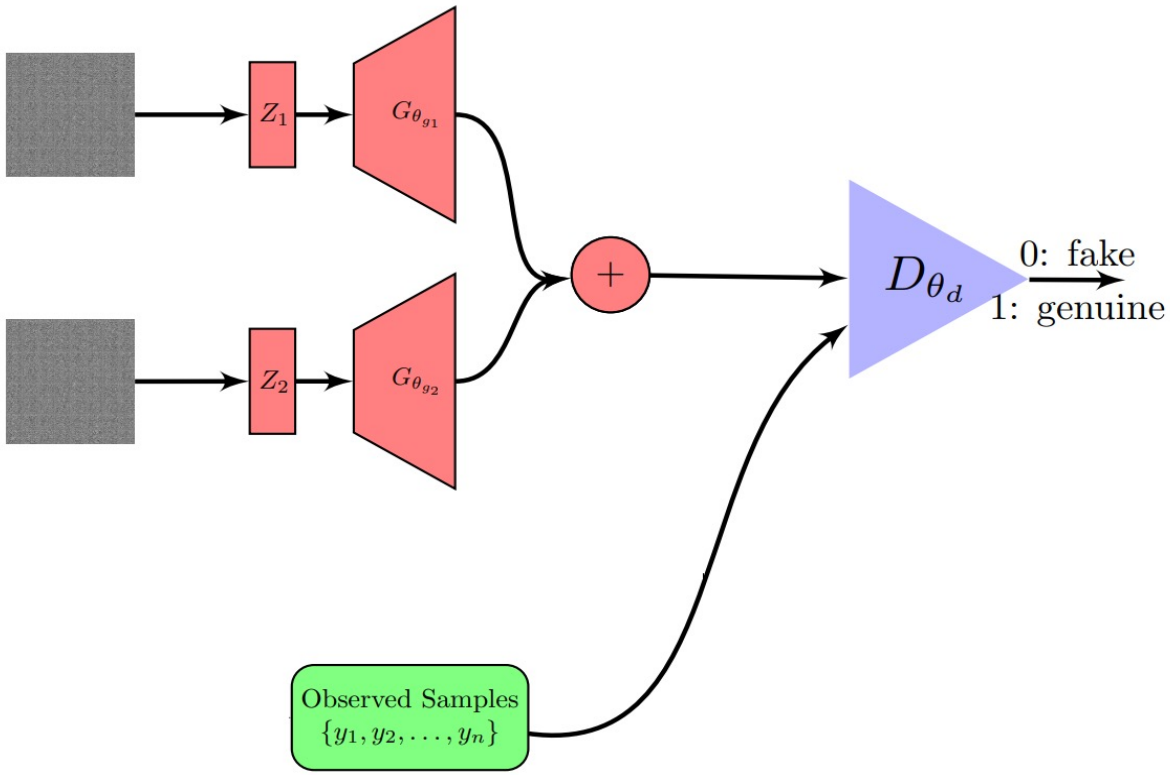



Liu, Ming-Yu, and Oncel Tuzel. "**Coupled generative adversarial networks**". NIPS (2016).

# Demixing GAN

- Learning separate sources from mixtures using GAN

$$Y_i = X_i + N_i = 1,2, \dots, n$$

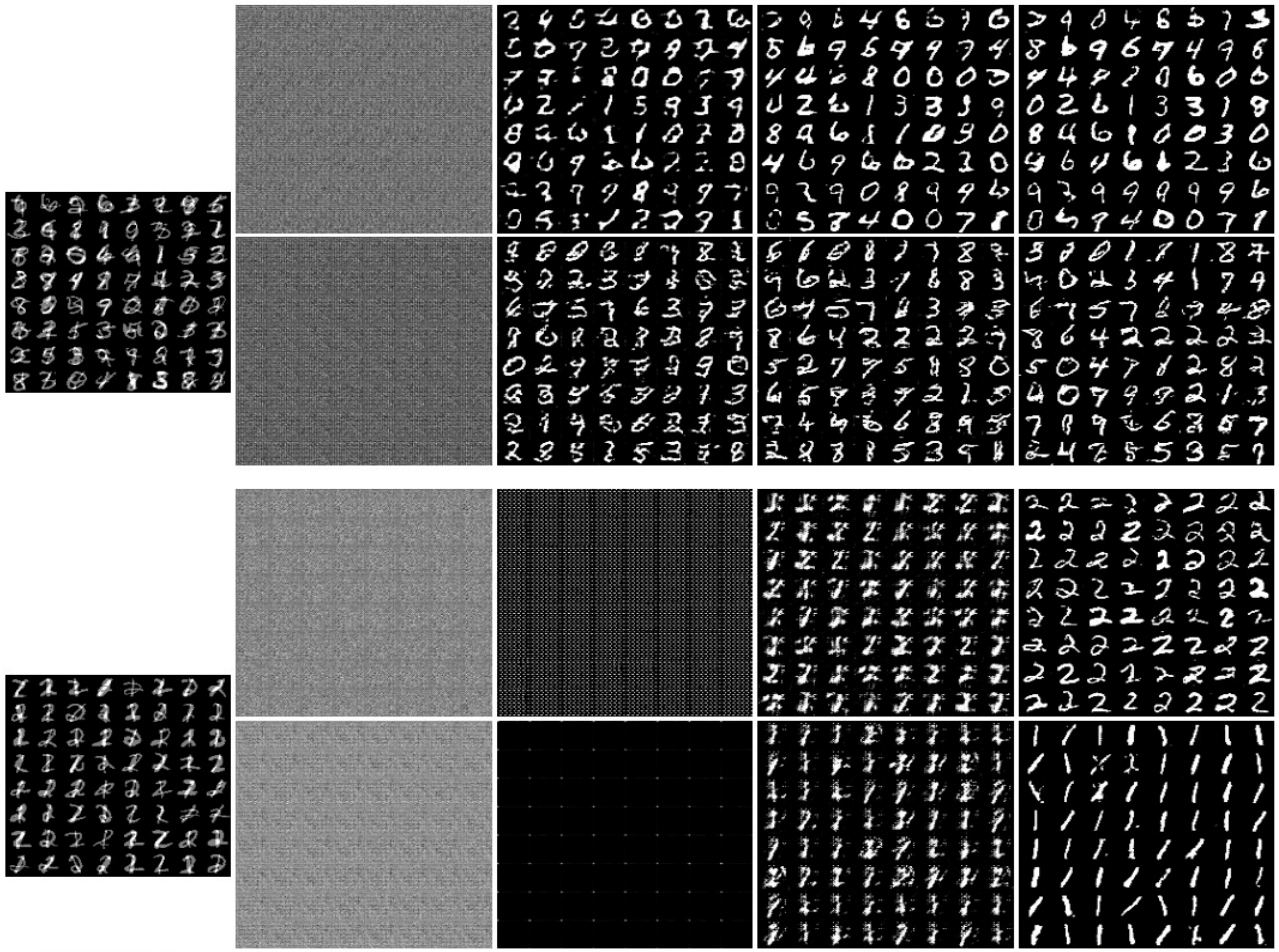





M. Soltani et al., "Learning Generative Models of Structured Signals from Their Superposition Using GANs", 2018.

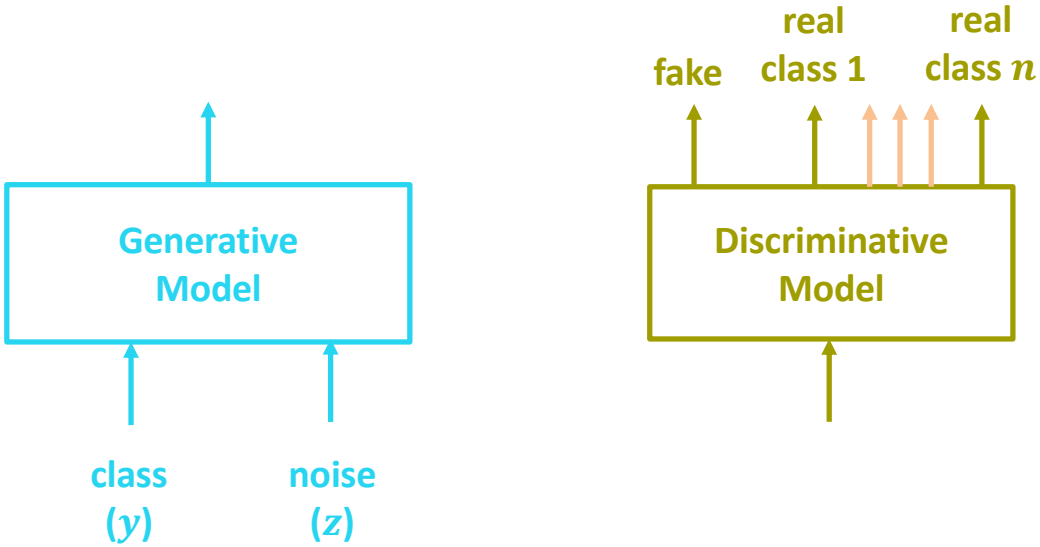
# Demixing GAN

- Superposing two MNIST digits
- Separating them using demixing GAN



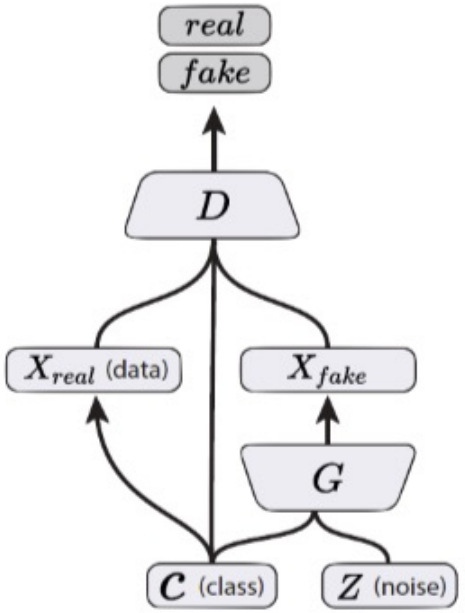
# Using Labels Can Improve Generated Samples

Denton et al. (2015)
Salimans et al. (2016)



# Conditional GANs

- Providing side information for generator
- Simple modification to the original GAN framework that conditions the model on additional information for better multi-modal learning
- It gives you more control on what you can generate
- Lends to many practical applications of GANs when we have explicit supervision available



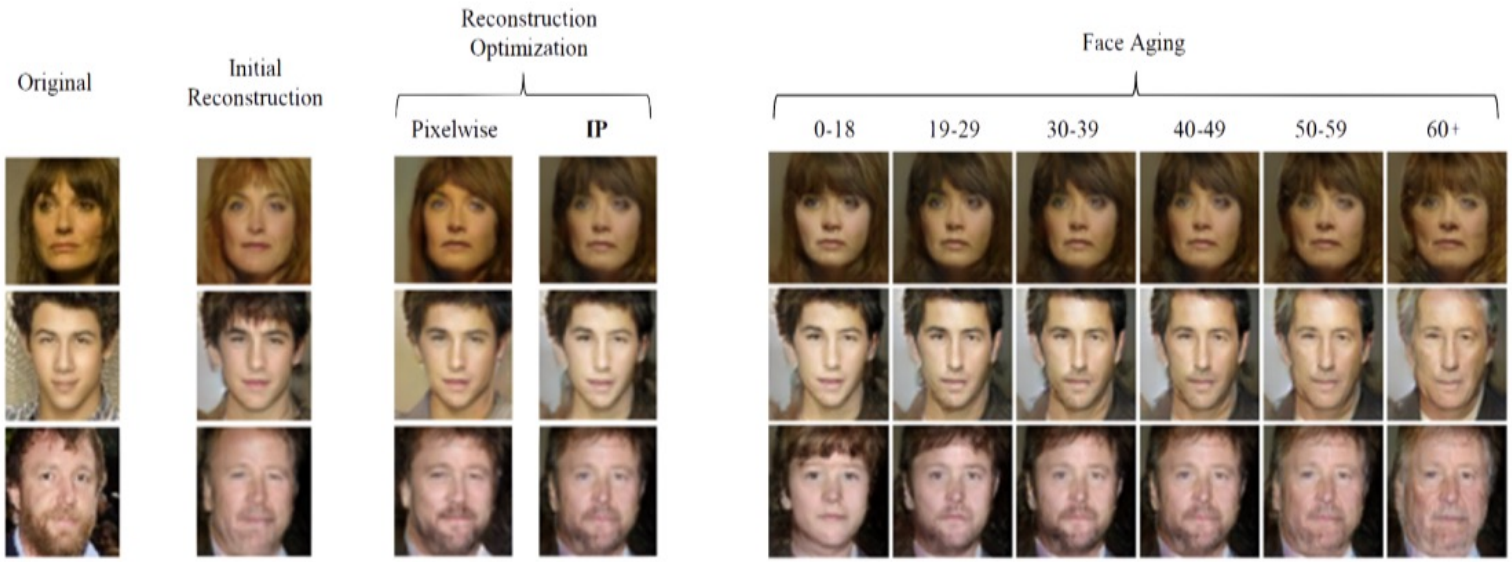
# Conditional GANs

MNIST digits generated conditioned on their class label.


Mirza, Mehdi, and Simon Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784 (2014).

# Applications of Conditional GANs

- Face Aging




Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.

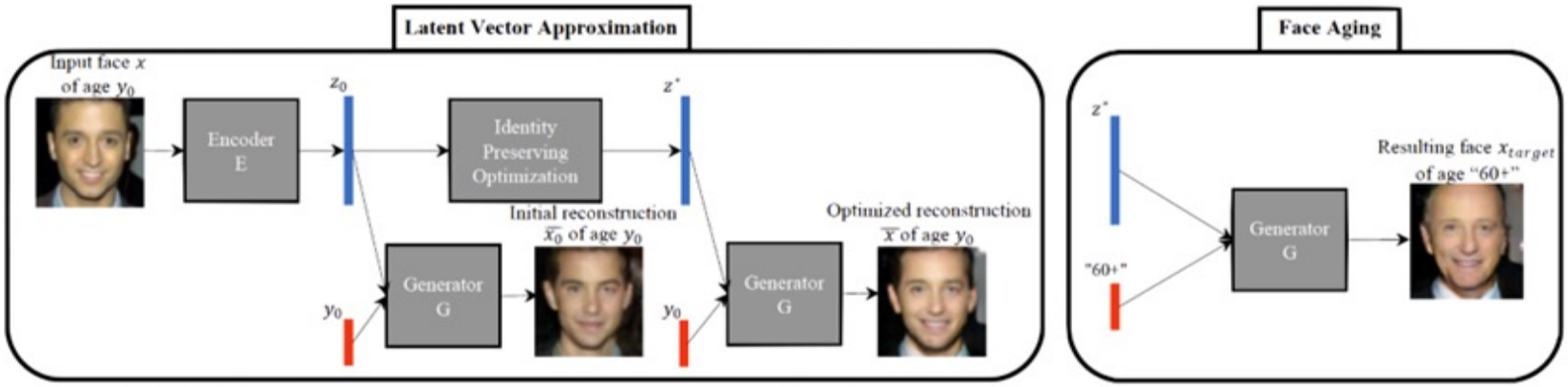
IP: Identity Preserving (please see the above paper for more details)

# Applications of Conditional GANs

- Face Aging
  - Using an auxiliary network to get a better approximation of the latent code ($z^*$) for an input image.
    - Given an input face image $x$ of age $y_0$, find an optimal latent vector $z^*$ which allows to generate a reconstructed face $\bar{x} = G(z^*, y_0)$ as close as possible to the initial one
    - Identity Preserving Optimization ($FR$ is neural network called Face Recognition (recognizes a person's identity in a given image))

$$z^*_{IP} = \underset{z}{\operatorname{argmin}} ||FR(x) - FR(\bar{x})||_{L_2}$$

  - Latent code is then conditioned on a discrete (one-hot) embedding of age categories




Antipov, G., Baccouche, M., & Dugelay, J. L. (2017). "**Face Aging With Conditional Generative Adversarial Networks**". arXiv preprint arXiv:1702.01983.

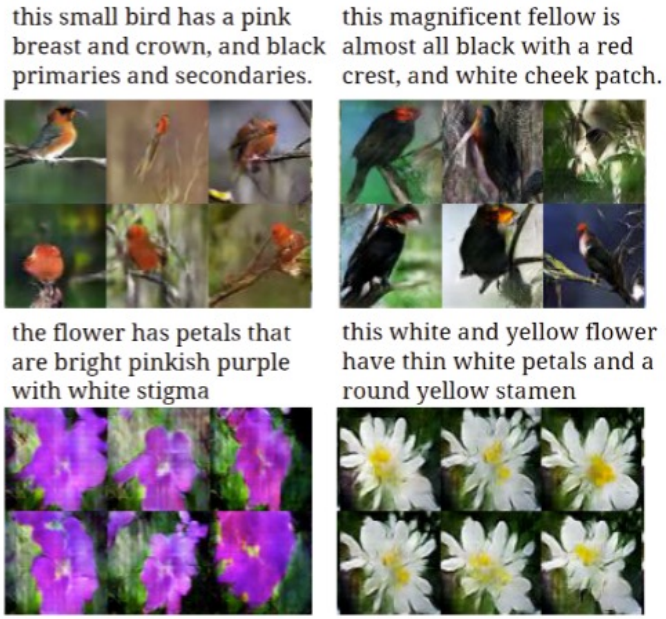
# Applications of Conditional GANs

- Text-to-Image Synthesis

Motivation

Given a text description, generate images closely associated.

Uses a conditional GAN with the generator and discriminator being condition on “dense” text embedding.

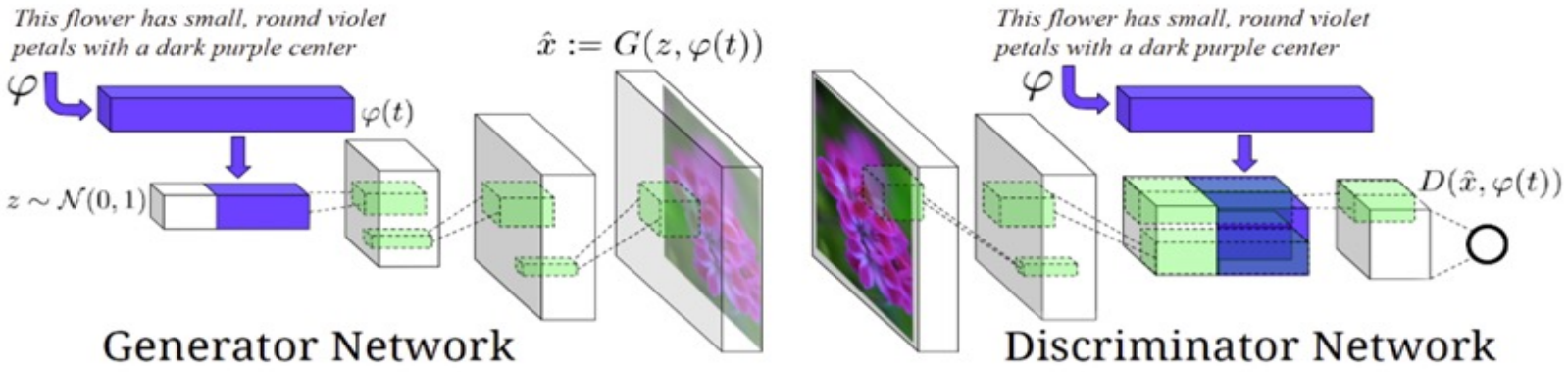



Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. “**Generative adversarial text to image synthesis**”. ICML (2016).

# Applications of Conditional GANs

- Text-to-Image Synthesis

Text-conditional convolutional GAN architecture




Positive Example:
Real Image, Right Text

Negative Examples:
Real Image, Wrong Text
Fake Image, Right Text

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., & Lee, H. "Generative adversarial text to image synthesis". ICML (2016).

- Text encoding $\phi(t)$ is used by both generator and discriminator.
- It is projected to a lower-dimensions and depth concatenated with image feature maps for further stages of convolutional processing.

# InfoGAN

- InfoGAN tries to provide more control over the generated samples (similar to the conditional GAN) by creating a ***disentangled representation***.
- Splitting the Generator input into two parts:
  - The traditional noise vector
  - A new "latent code" vector
- InfoGAN creates a meaningful code by maximizing the Mutual Information (MI) between the code and the generator output:

$$min_{\theta_g}\, max_{\theta_d} V\left(D_{\theta_d}, G_{\theta_g}\right) - \lambda I(c, G_{\theta_g}(z, c))$$

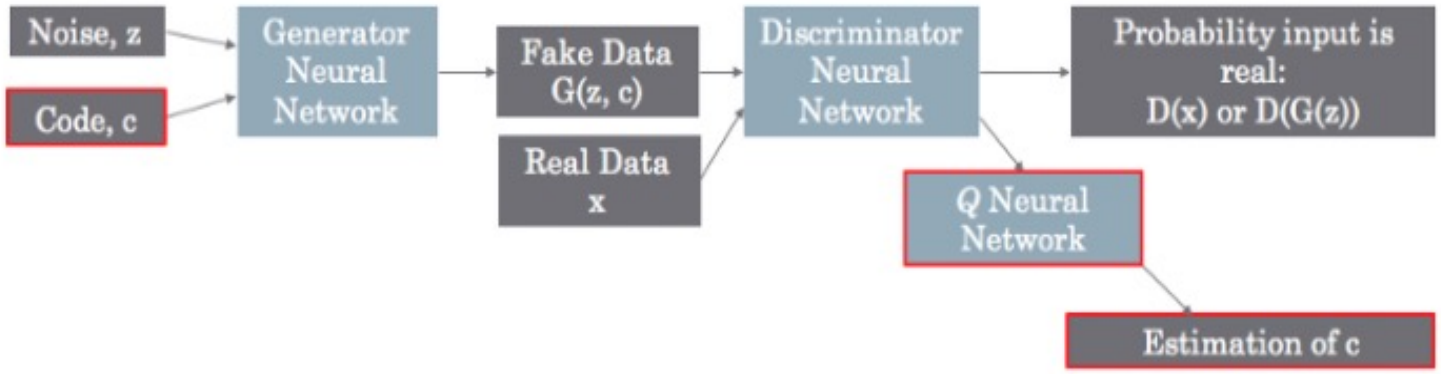
- $V\left(D_{\theta_d}, G_{\theta_g}\right)$ can be any min-max loss in (Jensen-Shannon, Wasserstein,…) .
  - Lambda is is typically just set to one.
- A variational (lower bound) approximation is used for the MI to make the computation tractable (requires access to the posterior $P(c|x)$):

$$\begin{aligned} L_I(G,Q) &= E_{c\sim P(c),x\sim G(z,c)}[\log Q(c|x)] + H(c) \\ &= E_{x\sim G(z,c)}[\mathbb{E}_{c'\sim P(c|x)}[\log Q(c'|x)]] + H(c) \\ &\leq I(c; G(z,c)) \end{aligned}$$

  - $Q(c|x)$ is an auxiliary distribution to approximate $P(c|x)$:

https://arxiv.org/pdf/1606.03657.pdf

# InfoGAN Architecture

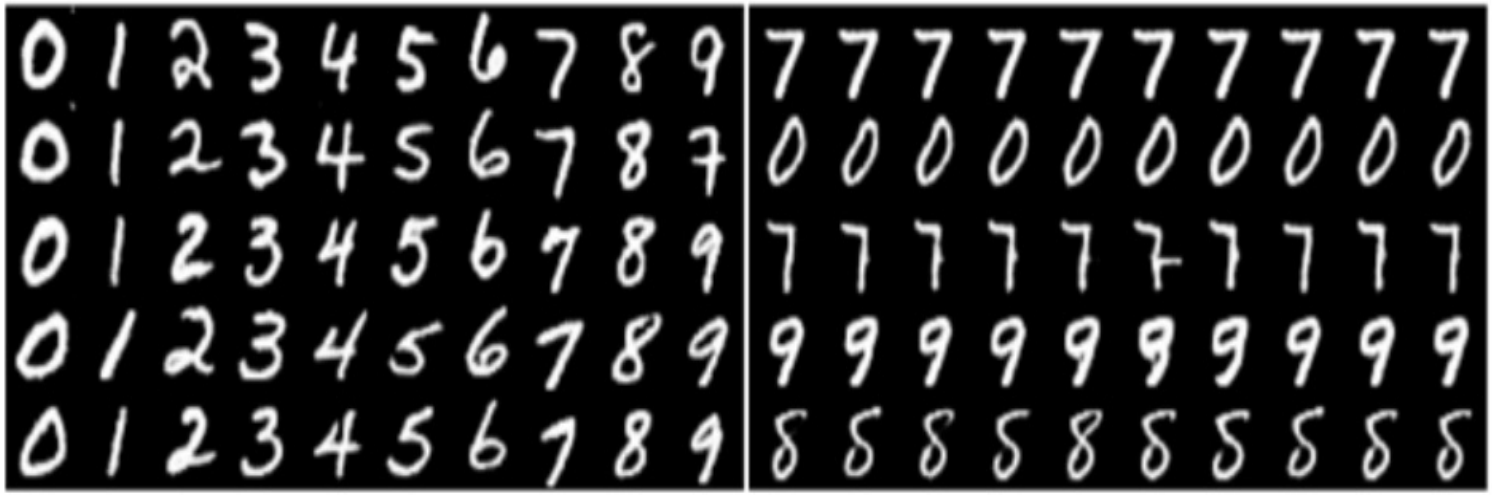



- Parametrizing the auxiliary distribution $Q$ as a neural network
- $Q$ and $D$ share all convolutional layers and there is one final fully connected layer to output parameters for the conditional distribution $Q(c|x)$
- Adding a negligible computation cost to GAN

https://towardsdatascience.com/infogan-generative-adversarial-networks-part-iii-380c0c6712cd

# InfoGAN Generated Samples

- The images below show a process where a particular noise vector is held constant (each row), but the latent code is changed (each column).
- In InfoGAN, the discrete code consistently change the digit. The regular GAN has essentially no meaningful or consistent change.



InfoGAN Regular GAN

https://towardsdatascience.com/infogan-generative-adversarial-networks-part-iii-380c0c6712cd

# BigGAN

- GANs typically generate small images.
- The training process is instable (needs careful hyperparameters fine-tuning).
- **The BigGAN combines a suite of recent best practices in training class-conditional images:**
  - More model parameters
    - 2 to 4 times as many parameters compared to prior art
  - Larger Batch Sizes
    - 8 times the batch size compared to prior art
  - Architectural changes
    - Improving scalability with general architectural changes, and modifying a regularization scheme

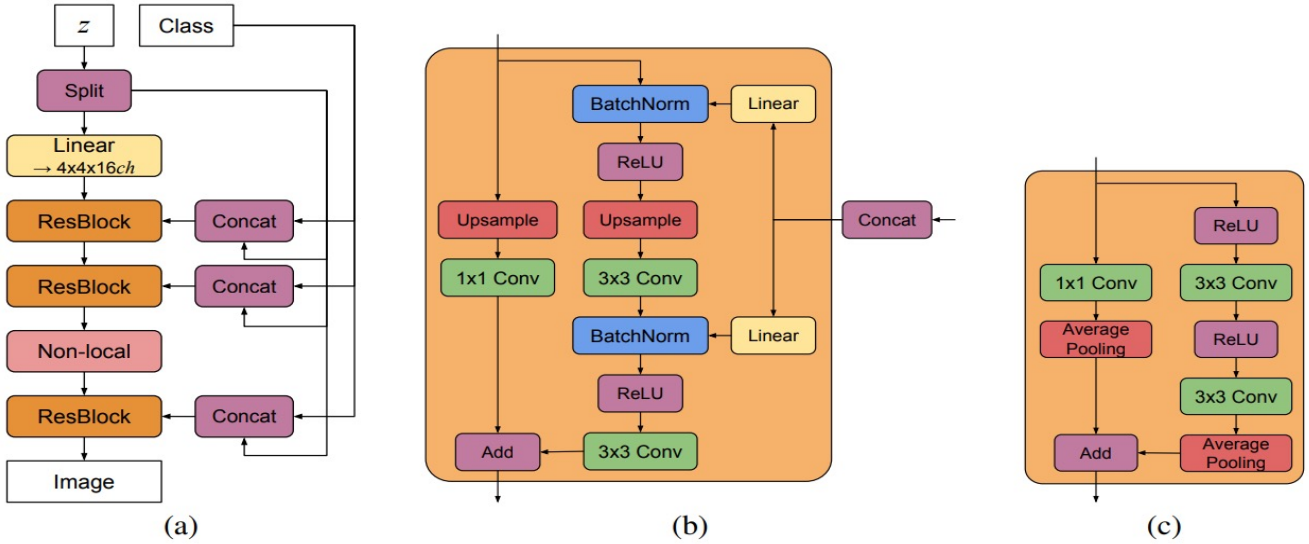
https://arxiv.org/pdf/1809.11096.pdf

# BigGAN Samples

Samples generated by BigGAN model at 256×256 resolution.

Samples generated by BigGAN model at 512×512 resolution.

# BigGAN—Architecture (128× 128)




(a) A typical architectural layout for BigGAN's **G**; details are in the following tables.
(b) A Residual Block (*ResBlock up*) in BigGAN's **G**.
(c) A Residual Block (*ResBlock down*) in BigGAN's **D**.

| $z \in \mathbb{R}^{120} \sim \mathcal{N}(0, I)$<br>Embed$(y) \in \mathbb{R}^{128}$ |
|---|
| Linear $(20 + 128) \to 4 \times 4 \times 16ch$ |
| ResBlock up $16ch \to 16ch$ |
| ResBlock up $16ch \to 8ch$ |
| ResBlock up $8ch \to 4ch$ |
| ResBlock up $4ch \to 2ch$ |
| Non-Local Block $(64 \times 64)$ |
| ResBlock up $2ch \to ch$ |
| BN, ReLU, $3 \times 3$ Conv $ch \to 3$ |
| Tanh |

(a) Generator

| RGB image $x \in \mathbb{R}^{128 \times 128 \times 3}$ |
|---|
| ResBlock down $ch \to 2ch$ |
| Non-Local Block $(64 \times 64)$ |
| ResBlock down $2ch \to 4ch$ |
| ResBlock down $4ch \to 8ch$ |
| ResBlock down $8ch \to 16ch$ |
| ResBlock down $16ch \to 16ch$ |
| ResBlock $16ch \to 16ch$ |
| ReLU, Global sum pooling |
| Embed$(y) \cdot \boldsymbol{h}$ + (linear $\to 1$) |

(b) Discriminator

- BigGAN architecture for 128 × 128 images.
- ch represents the channel width multiplier (e.g., ch=64).
- "split" splits the vector z along its channel dimension into chunks of equal size (20-D).
- Each chunk is concatenated to the shared class embedding and passed to a corresponding residual block as a conditioning vector.

https://arxiv.org/pdf/1809.11096.pdf

# BigGAN—Innovations

**1. Using Hing-Loss (similar to SVM loss):**

$$V(D,G) = -\mathbb{E}_{\boldsymbol{x}\sim\text{real}}\max(0, 1 - D_{\theta_d}(\boldsymbol{x})) - \mathbb{E}_{\boldsymbol{z}}\max(0, 1 - D_{\theta_d}(G_{\theta_g}(z)))$$

**2. Class conditional information**

- Class-conditional batch normalization (introduced by Dumoulin, et al. 2016)
  - Normalizing activations based on the statistics from images of a given class
  - Transforming a layer's activations into a normalized activation specific to the class of images
- Providing *class-conditional batch normalization* to the generator network
  - Instead of using one class embedding per label, a shared class-conditional code is used in order to reduce the number of weights.

**3. Spectral normalization for the weights of Generator function G!**

https://machinelearningmastery.com/a-gentle-introduction-to-the-biggan/

# BigGAN—Innovations

4. **Update the Discriminator more than the Generator**
   - Updating the Discriminator twice before updating the Generator in each training iteration
5. **Orthogonal Weight Initialization**
   - Initialized weights by random orthogonal matrix.
6. **Larger Batch Size**
   - e.g., batch sizes of 256, 512, 1024, and 2,048 images
7. **Larger number of parameters**
   - Doubling the number of channels or feature maps (filters) in each layer
8. **Skip-z Connections**
   - Directly connecting the input latent vector $z$ to specific layers deep in Generator

# BigGAN—Innovations

## 9. Truncation Trick

- during training samples the generator's latent code $z$ from standard distribution
- During inference samples the generator's latent code $z$ using a truncated standard distribution
  - Resampling if the values of z is above a threshold until it falls below the threshold
- The smaller the threshold, the better quality
- The larger the threshold, the more variety in sampled images

## 10. Orthogonal Weight Regularization

- Some of the deeper models creates saturated artifacts when using the truncation trick.
- Instead, using *Orthogonal Regularization* (Brock et al., 2017)

$$R_\beta(W) = \|W^T W - \boldsymbol{I}\|_F^2$$

- Using more complicated *Orthogonal Regularization:*

$$R_\beta(W) = \|W^T W \odot (\mathbf{1} - \boldsymbol{I})\|_F^2$$

  - $W$ denotes the weight matrices.
  - $\mathbf{1}$ denotes a matrix with all ones.
  - $\odot$ denotes an element-wise multiplication.

# 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
- 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