Generative Adversarial Networks (GANs)

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

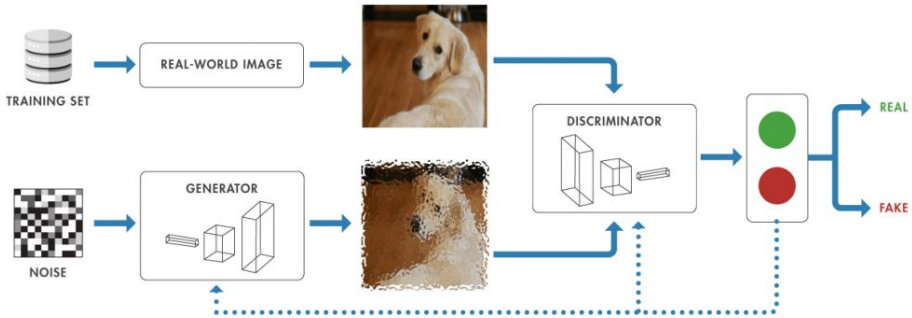
جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

June 23, 2025

Table of Contents

Table of Contents (cont.)



Architecture of a Generative Adversarial Network (GAN). The generator creates fake data, while the discriminator distinguishes between real and fake data. The two networks are trained in opposition to each other, leading to improved performance over time.

GANs: Motivation

So far, we have studied generative models based on maximizing likelihood or its approximations:

1. **Autoregressive Models**: $p_\theta(x) = \prod_{i=1}^{N} p_\theta(x_i|x_{<i})$
2. **Variational Autoencoders (VAEs)**: $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)$
3. **Normalizing Flow Models**: $p_X(x;\theta) = p_Z(f_\theta^{-1}(x)) \left|\det\left(\frac{\partial f_\theta^{-1}(x)}{\partial x}\right)\right|$

Problem with Traditional Generative Models:

- They often require complex approximations or sampling methods.
- They can produce blurry or unrealistic samples.
- They depend on explicit density estimation, which can be computationally expensive.

What if we give up on explicitly modeling density, and just want ability to sample?

Generative Adversarial Networks (GANs) provide a solution to this problem by introducing a novel approach to generative modeling:

- (introduced by Ian Goodfellow et al., 2014) do not require explicit density estimation or Markov chains.
- consist of two neural networks—a **generator** and a **discriminator**—in a 2-player (zero-sum) game.
- The adversarial setup enables the generator to learn complex data distributions and produce high-fidelity, realistic samples.

GAN Progress

- GANs are the most prominent example of Implicit Models.

GAN Progress (cont.)

https://www.youtube.com/watch?v=sW6D34mckkk

[BigGAN, Brock, Donahue, Simonyan, 2018]

GAN Progress (cont.)

$$\min_G \max_D \mathbb{E}_x[\log(D(x))] + \mathbb{E}_z[\log(1 - D(G(z)))]$$

10/25/2018

Sold at Christies for $432,500

GANs: Objectives

1. **Understand the fundamental concepts and motivations behind GANs.**
   - 1.1 Learn why GANs were introduced and what problems they aim to solve in generative modeling.
2. **Explore the mathematical formulation and training process of GANs.**
   - 2.1 Study the minimax objective and the roles of the generator and discriminator.
   - 2.2 Analyze how GANs compare distributions via samples.
3. **Examine objective functions and training challenges.**
   - 3.1 Review standard and alternative objective functions.
   - 3.2 Identify common problems in GAN training, such as mode collapse and instability.
4. **Survey popular GAN variants and latent space concepts.**
   - 4.1 Wasserstein GANs, Conditional GANs, CycleGANs, StyleGANs.
   - 4.2 Understand the role and selection of latent spaces in GANs.

⇒ Learning Outcomes

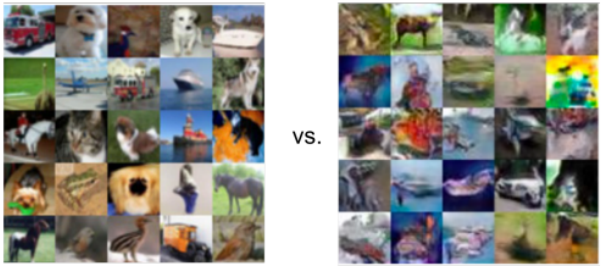
By the end of this session, you will be able to:

1. Understand how GANs work at a high level.
2. Describe the roles of the Generator and Discriminator.
3. Explain how GANs are trained.
4. Recognize major variants of GANs.
5. Understand challenges and limitations of GANs.

GANs: **Introduction**

Comparing Distributions via Samples

Given a finite set of samples from two distributions $S_1 = \{x \sim P\}$ and $S_2 = \{x \sim Q\}$, how can we tell if these samples are from the same distribution? (i.e., $P = Q$?)



GANs aim to replicate data distributions, but we rarely have access to the true probability distribution function. Instead, we compare samples from the real data distribution with those generated by the model.

Comparing Distributions via Samples (cont.)

- Let's consider a test statistic T for this purpose.
- Test statistic T compares S_1 and S_2 e.g., difference in means, variances of the two sets of samples.
- **Key observation**: Test statistic is likelihood-free since it does not involve the densities P or Q (only samples)

GANs: **Definition and Core Components**

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.

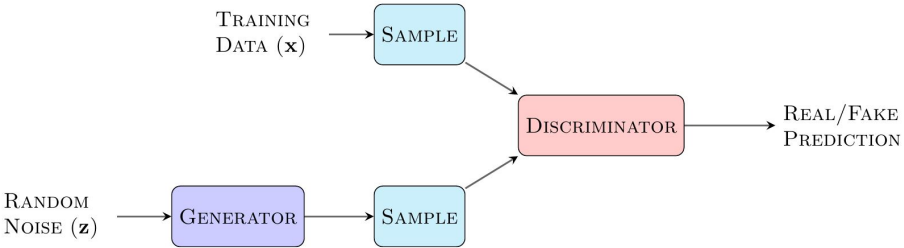
- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?
- **Another Idea**: Train another neural network to discriminate between the two samples, i.e., the discriminator.

Definition and Core Components

- Let $S_1 = D = \{x \sim p_{\text{data}}\}$ and $S_2 = \{x \sim p_\theta\}$.
- **Idea**: Train the generative model to minimize a two-sample test objective between S_1 and S_2, i.e., the generator.
- **Question**: How do we obtain a two-sample test objective?
- **Another Idea**: Train another neural network to discriminate between the two samples, i.e., the discriminator.
- And Voila! That's how we arrive at a GAN. The remaining step is to define the training method.

GANs: Definition and Core Components

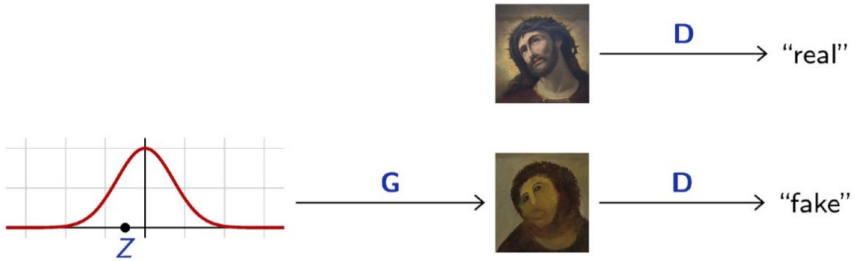


A GAN model diagram[1]

GANs: Definition and Core Components (cont.)

- Generative Adversarial Networks were introduced by Ian Goodfellow et al. (2014).
- The idea behind GANs is to train two networks jointly:
  - A **discriminator (D)** to classify samples as "real" or "fake".
  - A **generator (G)** to map a simple fixed distribution to samples that fool **D**.
- The approach is **adversarial** since the two networks have opposing objectives.

GANs: Definition and Core Components (cont.)



The generator transforms a simple distribution into samples resembling real data. The discriminator distinguishes between real and fake samples.

[1] https://github.com/JamesAllingham/LaTeX-TikZ-Diagrams

GANs: **Training**

- Both Discriminator and Generator are trained jointly in a min-max game.
- Minimax objective function:

$$\min_{\theta_G} \max_{\theta_D} = E_{x \sim p_{data}} log(\underbrace{D_{\theta_D}(x)}_{\substack{\text{Discriminator} \\ \text{output for} \\ \text{real data } x}}) + E_{x \sim p(z)} log(1 - \underbrace{D_{\theta_D}(G_{\theta_G}(z))}_{\substack{\text{Discriminator output} \\ \text{for generated fake} \\ \text{data } G(z)}})$$

- Discriminator θ_D wants to maximise objective such that $D(x)$ is close to 1 (real) and $D(G(z))$ is close to 0 (fake).
- Generator θ_G wants to minimise objective such that $D(G(z))$ is close to 1 (discriminator is fooled into thinking generated $G(z)$ is real).

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.

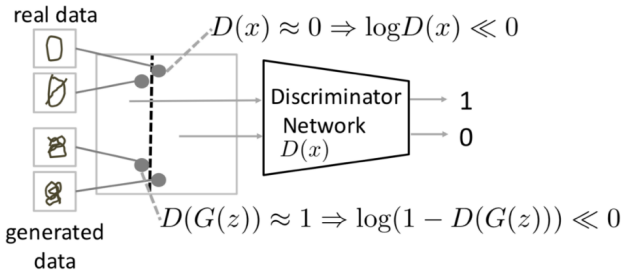
GANs: **Objective function**

Understanding the Objective function

The goal of GANs is to find a **Nash equilibrium** between the generator and discriminator. The generator tries to minimize the probability of the discriminator correctly classifying fake data.

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

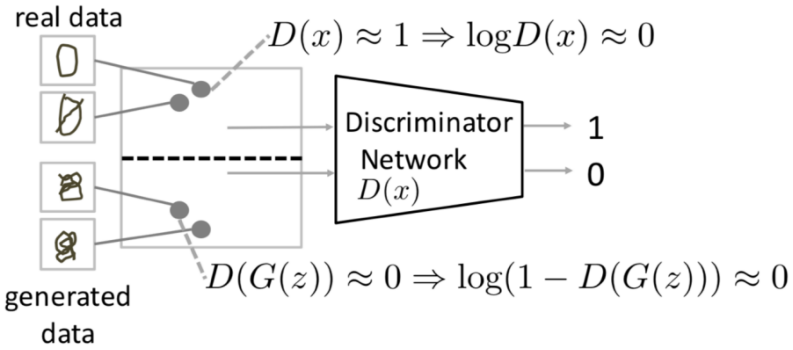
$D(x)$ should be 1 $\quad$ $D(G(z))$should be 0



Understanding the Objective function (cont.)

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

$D(x)$ should be 1 $D(G(z))$should be 0

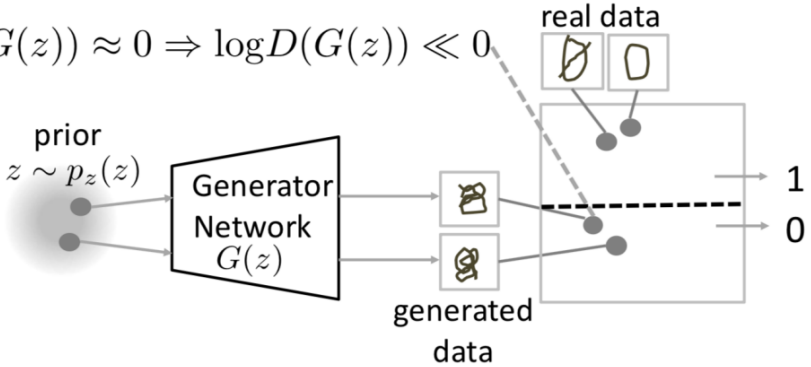


Understanding the Objective function (cont.)

$$\max_G(\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))])$$

$D(G(z))$ should be 1

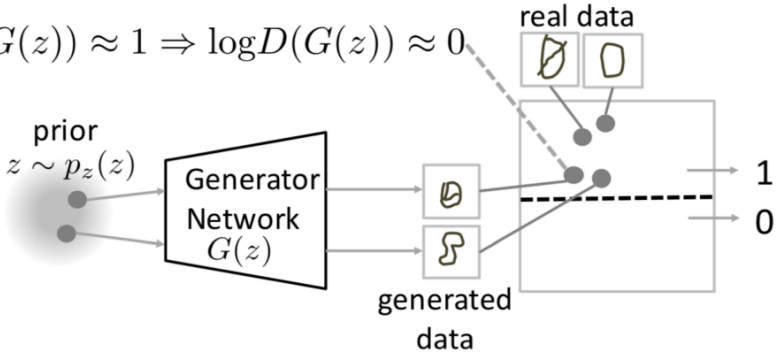
$$D(G(z)) \approx 0 \Rightarrow \log D(G(z)) \ll 0$$



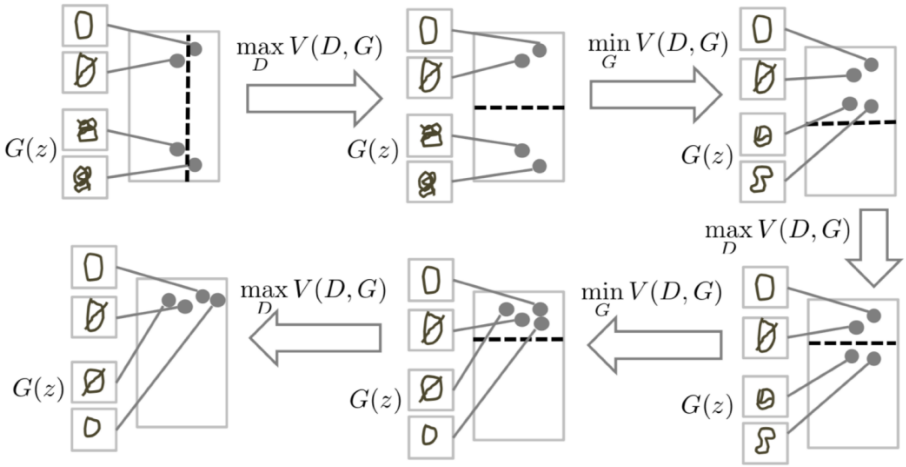
$$\max_G(\mathbb{E}_{z\sim p_z(z)}[\log(D(G(z))])$$

$D(G(z))$ should be 1

$D(G(z)) \approx 1 \Rightarrow \log D(G(z)) \approx 0$



Understanding the Objective function (cont.)



[1] https://www.slideshare.net/ckmarkohchang/generative-adversarial-networks

GANs - Interactive Demo

https://poloclub.github.io/ganlab/

GANs: **Results**

Results

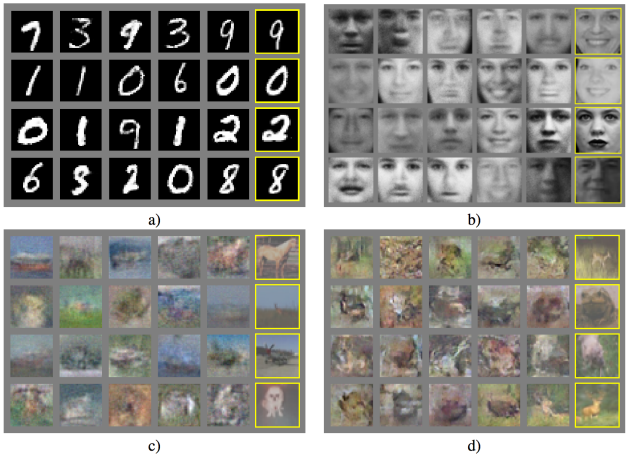
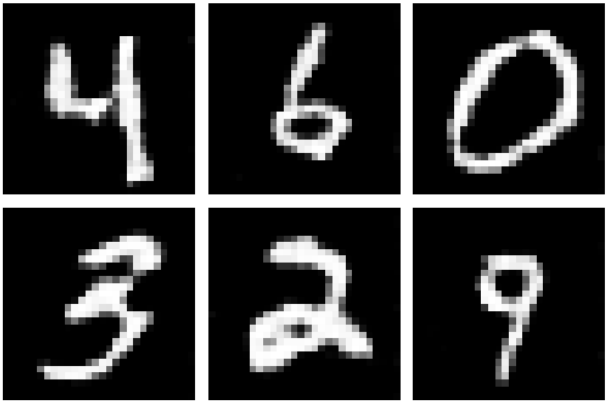


Figure from Goodfellow et al 2014, showing GAN results on various datasets.

Results (cont.)

generated_images

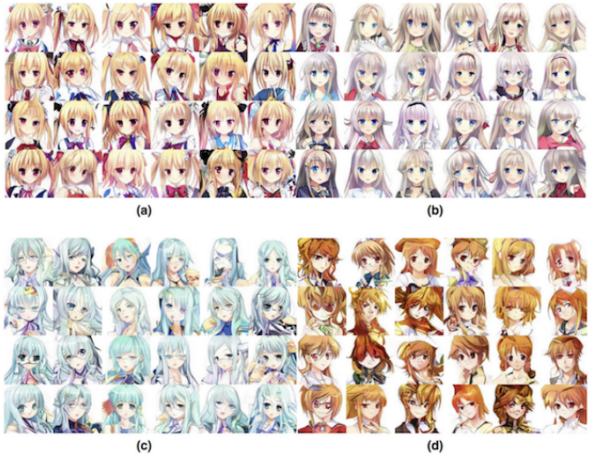


GAN generated samples for MNIST digits dataset

Results (cont.)

GAN generated samples for bedroom images

Results (cont.)



GAN generated samples for anime character faces

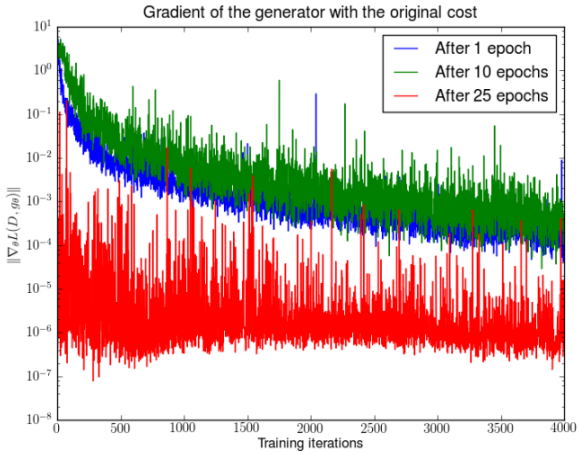
GANs: **Problems**

Vanishing Gradients

- When the discriminator is perfect, we are guaranteed with $D(x) = 1, \forall x \in p_{data}$ and $D(x) = 0, \forall x \in p_G$.
- The loss function drops to zero, resulting in no gradient for updates.
- **Solution**: Perform gradient ascent on the generator, i.e., use a different objective:

$$\max_{\theta_G} E_{x \sim p(z)} log(D_{\theta_D}(G_{\theta_G}(z)))$$

Problems with GANs (cont.)



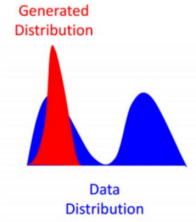
Vanishing gradient in GANs (Image source: Arjovsky and Bottou, 2017)

Difficulty in achieving Nash equilibrium

- GANs involve training two models simultaneously to reach a Nash equilibrium in a two-player non-cooperative game. However, since each model updates its cost independently, convergence is not guaranteed.
- For practical tips on training GANs, see "How to Train a GAN? Tips and tricks to make GANs work" by Soumith Chintala: https://github.com/soumith/ganhacks

Mode collapse

- The generator aims to fool the discriminator D into classifying its outputs as real. If the generator G finds a single output that consistently fools D, it may repeatedly produce that output, leading to mode collapse.
- Solutions to mode collapse are mostly empirical, including alternative architectures, modified GAN losses, and additional regularization terms.



Problems with GANs (cont.)

GAN mode collapse on MNIST digits dataset

Problems with GANs (cont.)

GANs: **Variants**

GAN Variants

- Since the introduction of GANs, extensive research has led to many variants.
- A comprehensive list of GAN variants can be found here.

GAN Variants (cont.)

We will discuss the following variants:

- **Deep Convolutional GAN (DCGAN):**
  - Uses convolutional and transposed convolutional layers.
  - Employs batch normalization and ReLU/LeakyReLU activations.
  - Improves training stability and image quality.
- **Wasserstein GAN (WGAN):**
  - Introduces Wasserstein distance as the objective.
  - Replaces sigmoid with linear output.
  - Uses weight clipping or gradient penalty.
  - **Variants:**
    - **WGAN-GP:** Uses gradient penalty instead of weight clipping for better stability.
    - **WGAN-LP:** Uses Lipschitz penalty as an alternative to gradient penalty.

GAN Variants (cont.)

- **Conditional GAN (cGAN):**
  - Conditions both the generator and discriminator on additional information (e.g., class labels).
  - Useful for targeted generation.
- **CycleGAN:**
  - Enables unpaired image-to-image translation.
  - Uses cycle consistency loss to ensure invertibility.
- **StyleGAN:**
  - Developed by NVIDIA for high-fidelity face generation.
  - Introduces style vectors at each generator layer.
  - Allows fine-grained control over image features.

GAN Variant:

Deep Convolutional GAN (DCGAN)

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

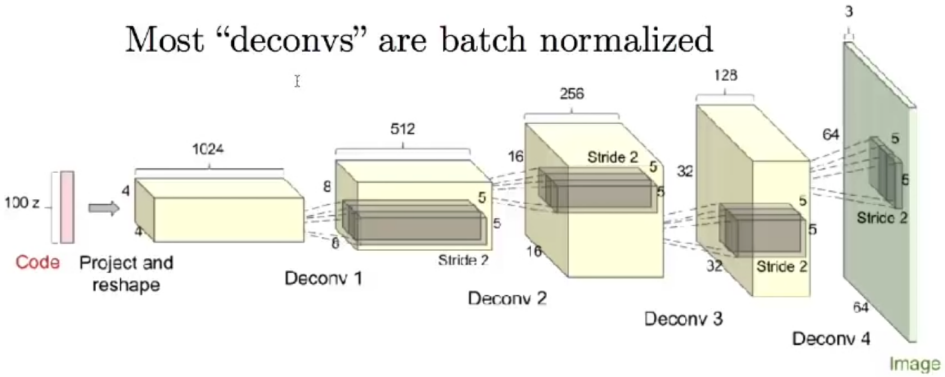
Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

Soumith Chintala
Facebook AI Research
New York, NY
soumith@fb.com

ABSTRACT

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this work we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator. Additionally, we use the learned features for novel tasks - demonstrating their applicability as general image representations.

DCGAN (cont.)



[Radford et al 2016]

Architecture Design Principles:

- Utilizes convolutional and transposed convolutional layers instead of fully connected layers.
- Key architectural guidelines:
  - Standard supervised CNNs are not directly usable for GANs.
  - Remove max-pooling and mean-pooling layers.
  - Generator upsamples using transposed convolutions.
  - Discriminator downsamples with strided convolutions and average pooling.
  - Non-linearity: ReLU for generator (except output), LeakyReLU (slope 0.2) for discriminator.
  - Output non-linearity: `tanh` for generator, `sigmoid` for discriminator.
  - Batch normalization is used to prevent mode collapse, but not applied at the output of G or input of D.

- **Optimization**: Adam optimizer with learning rate 2×10^{-4}, momentum 0.5, batch size 128.
- Achieves stable training and generates high-quality images.

DCGAN: Results

Good samples on datasets with 3M images (Faces, Bedrooms) for the first time

DCGAN: Results (cont.)

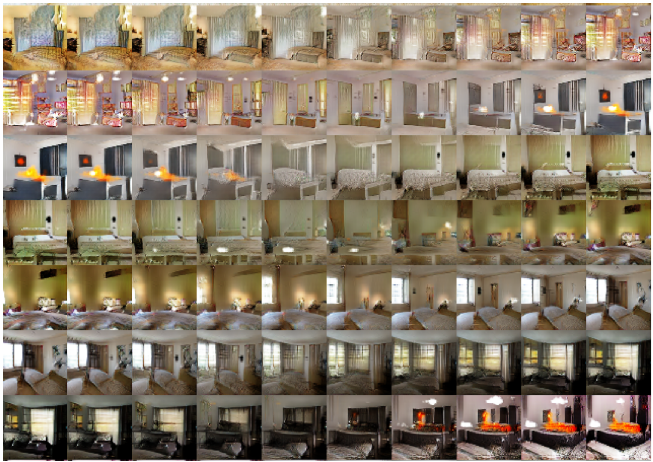
Reference: Radford, Metz, and Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks", ICLR 2016

DCGAN: Results (cont.)

[Radford et al 2016]

DCGAN: Results (cont.)

Smooth interpolations in high dimensions



[Radford et al 2016]

DCGAN: Results (cont.)

Imagenet samples (32x32)

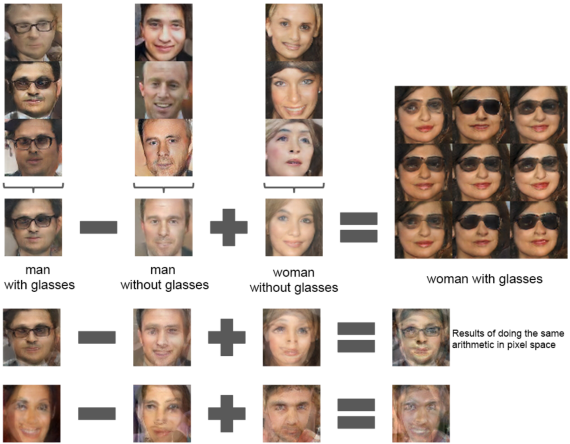
[Radford et al 2016]

DCGAN: Results (cont.)

Vector Arithmetic

[Radford et al 2016]

DCGAN: Results (cont.)



[Radford et al 2016]

DCGAN: Results (cont.)

[Radford et al 2016]

DCGAN: Results (cont.)

Representation Learning

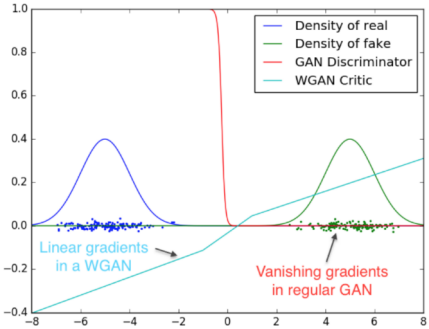
Model	Accuracy	Accuracy (400 per class)	max # of features units
1 Layer K-means	80.6%	63.7% (±0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% (±0.7%)	3200
View Invariant K-means	81.9%	72.6% (±0.7%)	6400
Exemplar CNN	84.3%	77.4% (±0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% (±0.4%)	512

[Radford et al 2016]

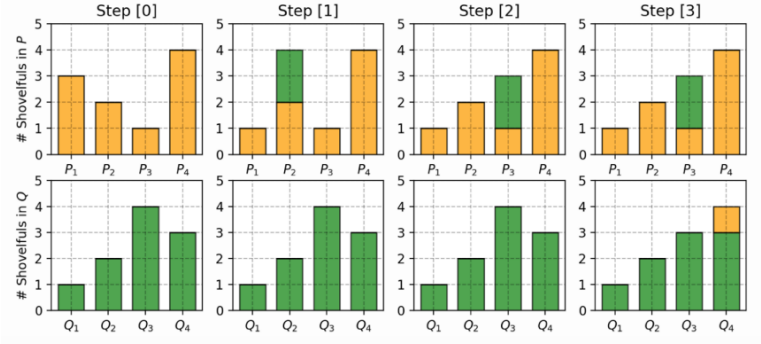
GAN Variant:

Wasserstein GANs (WGANs)

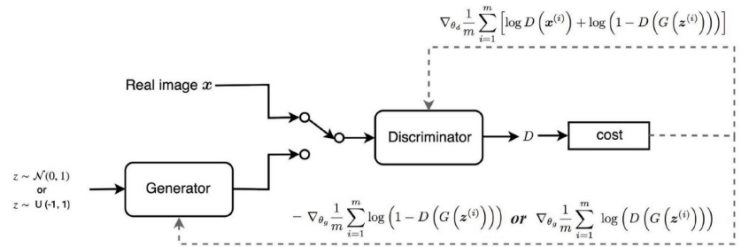
- Wasserstein GAN uses wasserstein distance instead of crossentropy loss.
- Wasserstein distance that has a smoother gradient everywhere.

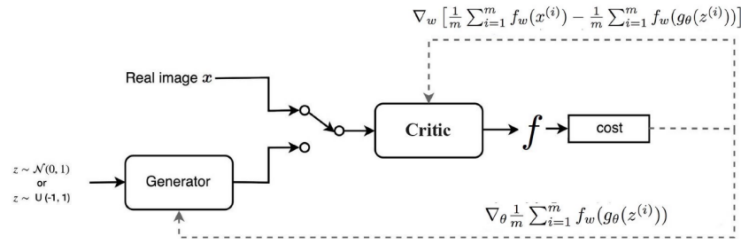


- Intuitively, it is the shovels of earth moved to make two distributions look alike.

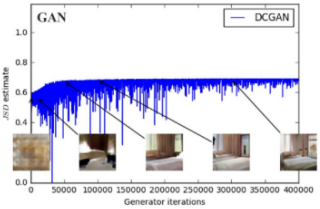


Step by Step plan of moving dirt between piles P and Q to make them match

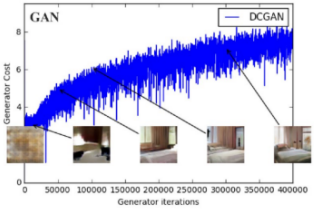

$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$
Real image x
Discriminator
D
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$-\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$ **or** $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right)$


$\nabla_w \left[\frac{1}{m} \sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)})) \right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0, 1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

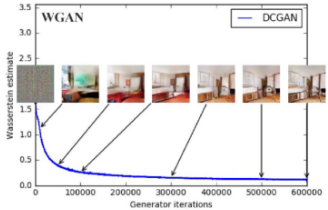
WGAN (cont.)



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



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



$$\frac{1}{m}\sum_{i=1}^{m} f_w\left(\boldsymbol{x}^{(i)}\right)-\frac{1}{m}\sum_{i=1}^{m} f_w\left(g_\theta\left(\boldsymbol{z}^{(i)}\right)\right)$$

WGAN (cont.)

WGAN generation results on bedroom images

Improved Training of Wasserstein GANs

Ishaan Gulrajani[1], **Faruk Ahmed**[1], **Martin Arjovsky**[2], **Vincent Dumoulin**[1], **Aaron Courville**[1,3]
[1] Montreal Institute for Learning Algorithms
[2] Courant Institute of Mathematical Sciences
[3] CIFAR Fellow
igul222@gmail.com
{faruk.ahmed,vincent.dumoulin,aaron.courville}@umontreal.ca
ma4371@nyu.edu

Abstract

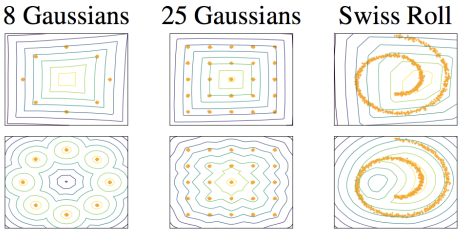
Generative Adversarial Networks (GANs) are powerful generative models, but suffer from training instability. The recently proposed Wasserstein GAN (WGAN) makes progress toward stable training of GANs, but sometimes can still generate only poor samples or fail to converge. We find that these problems are often due to the use of weight clipping in WGAN to enforce a Lipschitz constraint on the critic, which can lead to undesired behavior. We propose an alternative to clipping weights: penalize the norm of gradient of the critic with respect to its input. Our proposed method performs better than standard WGAN and enables stable training of a wide variety of GAN architectures with almost no hyperparameter tuning, including 101-layer ResNets and language models with continuous generators. We also achieve high quality generations on CIFAR-10 and LSUN bedrooms. †

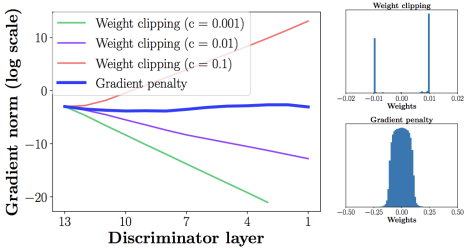
[Gulrajani et al 2017]

WGAN-GP: Gradient Penalty for Lipschitzness (cont.)

$$\max_D \underbrace{\mathbb{E}_{x\sim P_r}\left[D(x)\right] - \mathbb{E}_{\tilde{x}\sim P_g}\left[D(\tilde{x})\right]}_{\text{Wasserstein critic objective}} + \lambda \underbrace{\mathbb{E}_{\hat{x}\sim P_{\hat{x}}}\left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1\right)^2\right]}_{\text{Gradient Penalty for Lipschitzness}}$$

$$\hat{x} \leftarrow \epsilon x + (1-\epsilon)\tilde{x}$$





WGAN-GP: Pseudocode

Algorithm 1 WGAN with gradient penalty. We use default values of $\lambda = 10$, $n_{\text{critic}} = 5$, $\alpha = 0.0001$, $\beta_1 = 0$, $\beta_2 = 0.9$.

Require: The gradient penalty coefficient λ, the number of critic iterations per generator iteration n_{critic}, the batch size m, Adam hyperparameters α, β_1, β_2.
Require: initial critic parameters w_0, initial generator parameters θ_0.

while θ has not converged **do**
  for $t = 1, ..., n_{\text{critic}}$ **do**
    for $i = 1, ..., m$ **do**
      Sample real data $\boldsymbol{x} \sim \mathbb{P}_r$, latent variable $\boldsymbol{z} \sim p(\boldsymbol{z})$, a random number $\epsilon \sim U[0, 1]$.
      $\tilde{\boldsymbol{x}} \leftarrow G_\theta(\boldsymbol{z})$
      $\hat{\boldsymbol{x}} \leftarrow \epsilon \boldsymbol{x} + (1 - \epsilon)\tilde{\boldsymbol{x}}$
      $L^{(i)} \leftarrow D_w(\tilde{\boldsymbol{x}}) - D_w(\boldsymbol{x}) + \lambda(\|\nabla_{\hat{\boldsymbol{x}}} D_w(\hat{\boldsymbol{x}})\|_2 - 1)^2$
    end for
    $w \leftarrow \text{Adam}(\nabla_w \frac{1}{m} \sum_{i=1}^{m} L^{(i)}, w, \alpha, \beta_1, \beta_2)$
  end for
  Sample a batch of latent variables $\{\boldsymbol{z}^{(i)}\}_{i=1}^{m} \sim p(\boldsymbol{z})$.
  $\theta \leftarrow \text{Adam}(\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} -D_w(G_\theta(\boldsymbol{z})), \theta, \alpha, \beta_1, \beta_2)$
end while

[Gulrajani et al 2017]

No critic batch normalization Most prior GAN implementations [22, 23, 2] use batch normalization in both the generator and the discriminator to help stabilize training, but batch normalization changes the form of the discriminator's problem from mapping a single input to a single output to mapping from an entire batch of inputs to a batch of outputs [23]. Our penalized training objective is no longer valid in this setting, since we penalize the norm of the critic's gradient with respect to each input independently, and not the entire batch. To resolve this, we simply omit batch normalization in the critic in our models, finding that they perform well without it. Our method works with normalization schemes which don't introduce correlations between examples. In particular, we recommend layer normalization [3] as a drop-in replacement for batch normalization.

[Gulrajani et al 2017]

WGAN-GP: Robustness to architectures

Nonlinearity (G)	[ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]
Nonlinearity (D)	[ReLU, LeakyReLU, $\frac{\text{softplus}(2x+2)}{2} - 1$, tanh]
Depth (G)	[4, 8, 12, 20]
Depth (D)	[4, 8, 12, 20]
Batch norm (G)	[True, False]
Batch norm (D; layer norm for WGAN-GP)	[True, False]
Base filter count (G)	[32, 64, 128]
Base filter count (D)	[32, 64, 128]

WGAN-GP: Robustness to architectures (cont.)

Min. score	Only GAN	Only WGAN-GP	Both succeeded	Both failed
1.0	0	8	192	0
3.0	1	88	110	1
5.0	0	147	42	11
7.0	1	104	5	90
9.0	0	0	0	200

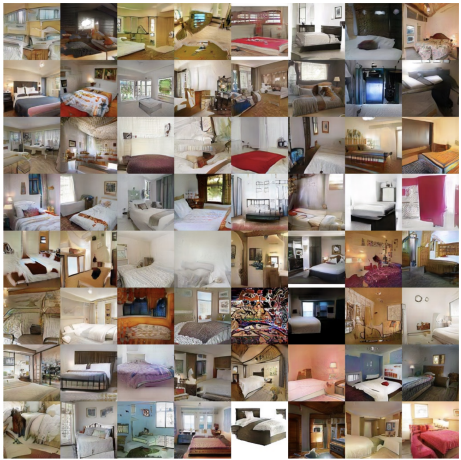
[Gulrajani et al 2017]

WGAN-GP: Robustness to architectures (cont.)

[Gulrajani et al 2017]

WGAN-GP: Robustness to architectures (cont.)

WGAN-GP: High quality samples



[Gulrajani et al 2017]

WGAN-GP: High quality samples (cont.)

Table 3: Inception scores on CIFAR-10. Our unsupervised model achieves state-of-the-art performance, and our conditional model outperforms all others except SGAN.

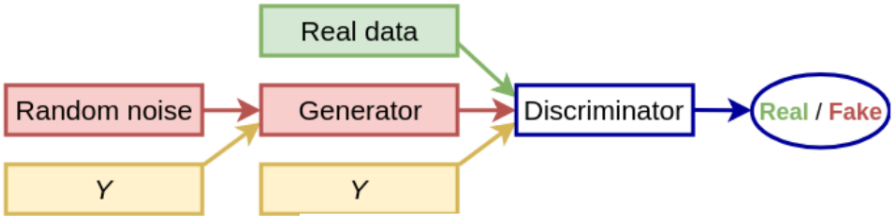
Unsupervised		Supervised	
Method	Score	Method	Score
ALI [8] (in [27])	$5.34 \pm .05$	SteinGAN [26]	6.35
BEGAN [4]	5.62	DCGAN (with labels, in [26])	6.58
DCGAN [22] (in [11])	$6.16 \pm .07$	Improved GAN [23]	$8.09 \pm .07$
Improved GAN (-L+HA) [23]	$6.86 \pm .06$	AC-GAN [20]	$8.25 \pm .07$
EGAN-Ent-VI [7]	$7.07 \pm .10$	SGAN-no-joint [11]	$8.37 \pm .08$
DFM [27]	$7.72 \pm .13$	WGAN-GP ResNet (ours)	$8.42 \pm .10$
WGAN-GP ResNet (ours)	$7.86 \pm .07$	**SGAN** [11]	$8.59 \pm .12$

[Gulrajani et al 2017]

GAN Variant:

Conditional GANs

- In addition to random noise, a condition is added to the generator input.
- The discriminator also receives this condition.



Conditional GAN (cont.)

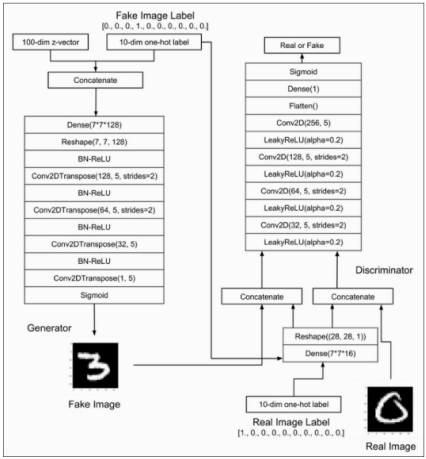
- Discriminator Loss:

$$\mathcal{L}^{(D)}\left(\theta^{(G)}, \theta^{(D)}\right) = -\mathbb{E}_{x\sim p_{data}}\left[\log D(x|y)\right] - \mathbb{E}_{z\sim p_z(z),\ y\sim p_{data}(y)}\left[\log(1 - D(G(z|y)))\right]$$

- Generator Loss:

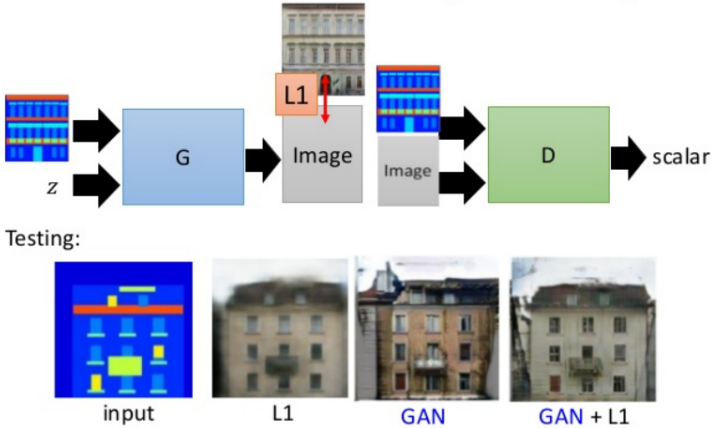
$$\mathcal{L}^{(G)}\left(\theta^{(G)}, \theta^{(D)}\right) = -\mathbb{E}_z\left[\log D(G(z|y))\right]$$

Conditional GAN (cont.)



A conditional GAN architecture for the MNIST digits dataset.

Condional GAN - Image to Image



Using L1 loss in addition in GAN loss can help in image to image translation

Condional GAN - Image to Image (cont.)

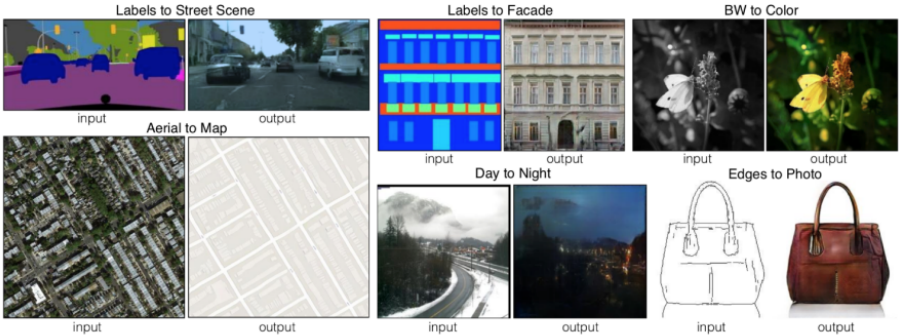
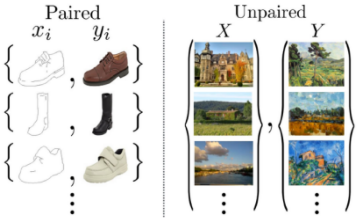


Image to Image translation with condional GANs

GAN Variant:

Cycle GANs

- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Efros, ICC7 2017



Cycle GANs (cont.)

$$C_{horse \to zebra} = horse \to G_{horse \to zebra} \to \hat{zebra} \to [D_{zebra}, G_{zebra \to horse}] \to \hat{horse}$$

Cycle GANs (cont.)

$$C_{zebra \to horse} = zebra \to G_{zebra \to horse} \to \hat{horse} \to [D_{horse}, G_{horse \to zebra}] \to \hat{zebra}$$

Cycle GANs (cont.)

Cycle GANs (cont.)

$G_{zebra \rightarrow horse}$

Cycle GANs (cont.)

$G_{horse \rightarrow zebra}$

Cycle GANs (cont.)

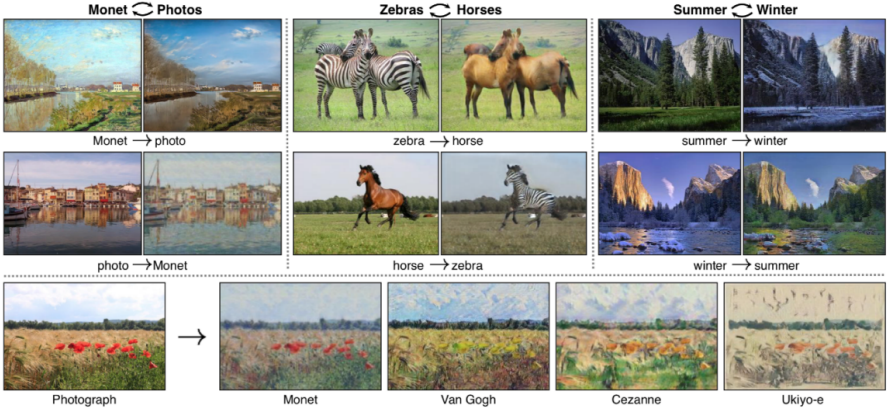
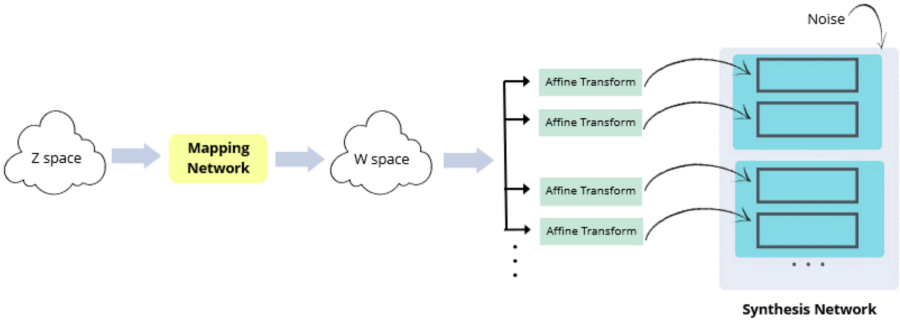


Image to Image translation with Cycle GANs

GAN Variant:

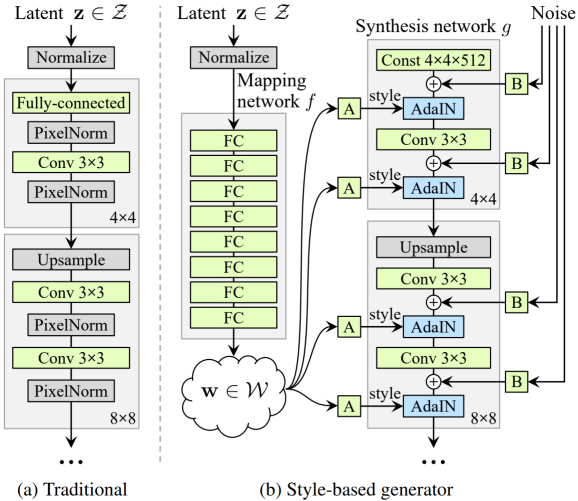
Style GANs

- Goal: Better disentanglement of features in latent space (**W** space)



[1] https://www.cs.unc.edu/ ronisen/teaching/fall2022/pdf/lectures/lecture6_gan2.pdf

Style GANs



(a) Traditional (b) Style-based generator

[1] https://arxiv.org/pdf/1812.04948

- **A** = learned affine transformation block for AdaIN (predicts y)
- **Ada**ptive **I**stance **N**ormalization (very effective in controlling styles)

$$\text{AdaIN}(x_i, y) = y_{s,i}\frac{x_i - u(x_i}{\sigma(x_i)} + y_{b,i}$$

- **B** = learned per-channel scaling factor for noise input.

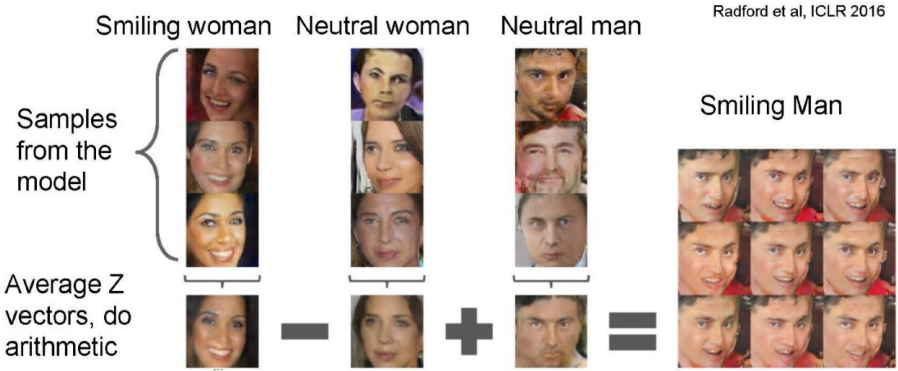
- **Z** : 512 dimensional latent space (not good)
- **W** : 512 dimensional latent space (better but not perfect)
- **W+** : 18x512 dimensional latent space (after affine transformation **A** has been applied)
- **W** is better for editing.
- **W+** is better for reconstruction or embedding of real images.

Style GANs - Results

[1] https://arxiv.org/pdf/1812.04948

GANs: **Latent space Interpolation**

Latent space Interpolation



Latent space interploation with GANs

Latent space Interpolation (cont.)

Latent space interpolation with StyleGAN
(Demo by Xander Steenburgge)

https://colab.research.google.com/drive/1mH70YxGNlnEaSOn0J8LsgkI-QOvsIb3MscrollTo=uEhxBvAR-7y3

GANs: **More Results**

Some more GAN results

Image Inpainting. https://www.nvidia.com/en-us/research/ai-demos/

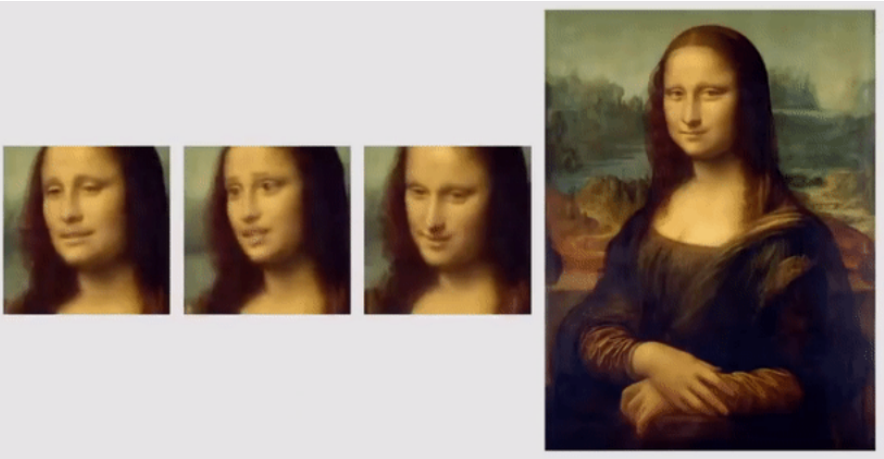
Some more GAN results (cont.)

this small bird has a pink breast and crown, and black primaries and secondaries.

this white and yellow flower have thin white petals and a round yellow stamen

Text to Image Synthesis with GANs

Some more GAN results (cont.)



Living Portraits with GANs

Some more GAN results (cont.)

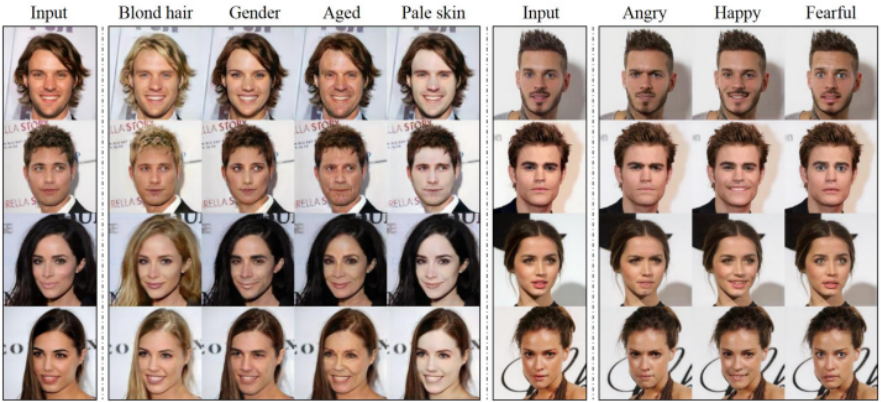


Image to Image translation in multiple domains with StyleGAN (Choi et al.)

GANs: **Summary**

Summary

- Instead of explicitly learning data probability distribution. Learn it implicitly.
- Train two networks jointly. Generator generates fake samples and aims to fool discriminator. Discriminator aims to discriminate between real and fake samples.
- GANs are notoriously difficult to train. Lots of tips and tricks available.
- Huge amount of research done on GANs and plenty of variations with interesting results.

GANs: **Limitations**

Limitations

- Hard to train
- Sensitive to hyperparameters
- No clear evaluation metric
- Mode collapse is common
- Often unpredictable behavior

GANs: **References**

References

Reference Slides

- Fei-Fei Li "Generative Deep Learning" CS231
- Francois Fleuret "Deep Learning" EE559
- Murtaza Taj "Deep Learning" CS437

References (Papers)

- Goodfellow et al., "Generative Adversarial Networks," 2014. arXiv:1406.2661
- Arjovsky et al., "Wasserstein GAN," 2017. arXiv:1701.07875
- Zhu et al., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," 2017. arXiv:1703.10593
- Karras et al., "A Style-Based Generator Architecture for Generative Adversarial Networks," CVPR 2019. arXiv:1812.04948
- Creswell et al., "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, 2018. IEEE Xplore

GANs: Definition and Core Components:

- **Definition**: Generative Adversarial Networks (GANs) consist of two neural networks trained in opposition to each other.
- **Core Components:**
  - **Generator (G):** Takes random noise (latent vector) and produces fake data samples.
  - **Discriminator (D):** Receives real or fake data and predicts whether the input is real.
- **Mathematical Formulation:**

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

Where:

- x: Real data sample
- z: Noise vector sampled from prior p_z
- $G(z)$: Generated sample

Credits

Dr. Prashant Aparajeya

Computer Vision Scientist — Director(AISimply Ltd)

p.aparajeya@aisimply.uk

This project benefited from external collaboration, and we acknowledge their contribution with gratitude.