Generative Adversarial Networks (GANs)

Naeemullah Khan
naeemullah.khan@kaust.edu.sa

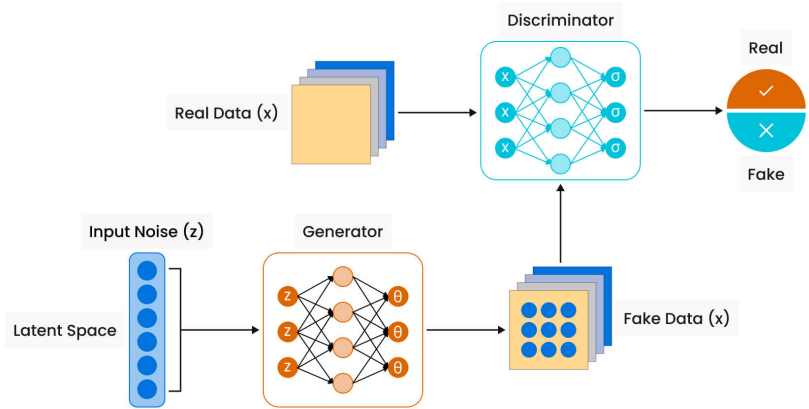


جامعة الملك عبدالله
للعلوم والتقنية
King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

August 9, 2026

Generative Adversarial Network (GAN)



[Source: https://www.solulab.com]

Table of Contents

1. Motivation
2. Objectives & Learning Outcomes
3. Introduction
4. Definition and Core Components
5. Training
6. Objective function
7. Results
8. Variants
    1. Deep Convolutional GAN (DCGAN)
    2. Wasserstein GANs (WGANs)
    3. Conditional GANs
    4. Pix2Pix GANs

Table of Contents (cont.)

5. Cycle GANs
6. Style GANs

9. Latent space Interpolation

10. More Results

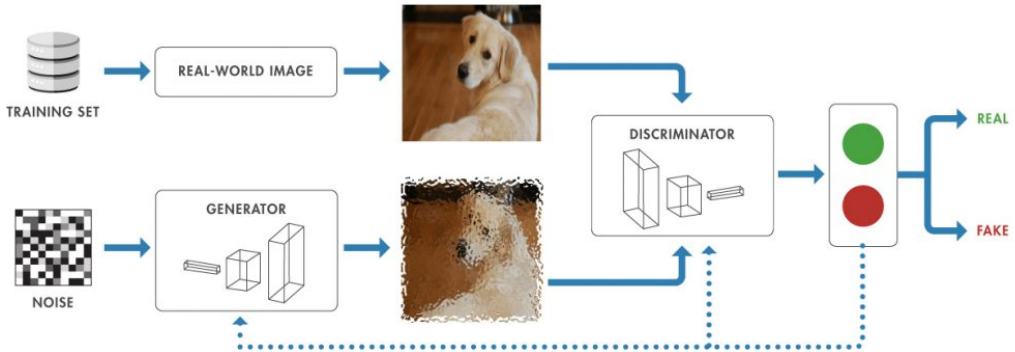
11. Problems

12. Summary

13. Limitations

14. References

GANs: **Motivation**



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. [Source: MathWorks Artificial Intelligence blog]

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)**: $p_\theta(x) = \int_z p_\theta(x|z)p_\theta(z)\,dz$
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?

GANs: Motivation (cont.)

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); they do not require explicit density estimation or Markov chains.
- They 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.)



["Portrait of Edmond de Belamy," Obvious collective, sold at Christie's, 25 October 2018]

GANs: **Objectives & Learning Outcomes**

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 Deep Convolutional GANs, Wasserstein GANs, Conditional GANs, Pix2Pix, CycleGANs, StyleGANs.
   - 4.2 Understand the role and selection of latent spaces in GANs.

⇒ Learning Outcomes

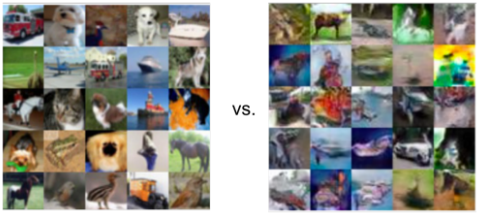
By the end of this session, you will be able to:

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., $P = Q$?)



[Source: Ermon & Grover, "Deep Generative Models," Stanford CS236, Lecture 9]

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**

Definition and Core Components

- Let $S_1 = \{x \sim p_{\text{data}}\}$ 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.

Definition and Core Components

- Let $S_1 = \{x \sim p_{\text{data}}\}$ 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?

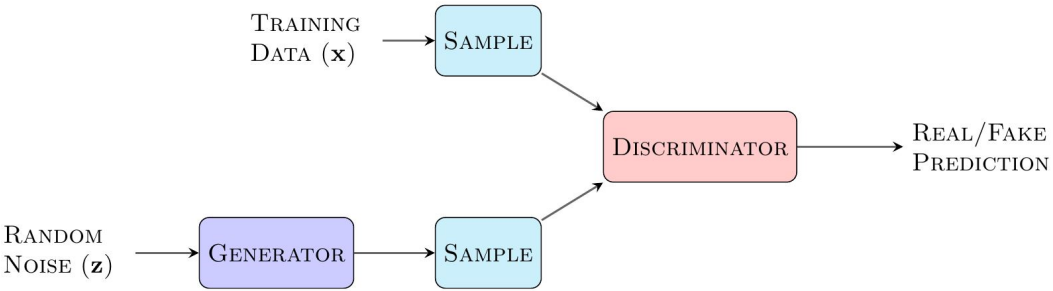
Definition and Core Components

- Let $S_1 = \{x \sim p_{\text{data}}\}$ 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 $S_1 = \{x \sim p_{\text{data}}\}$ 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

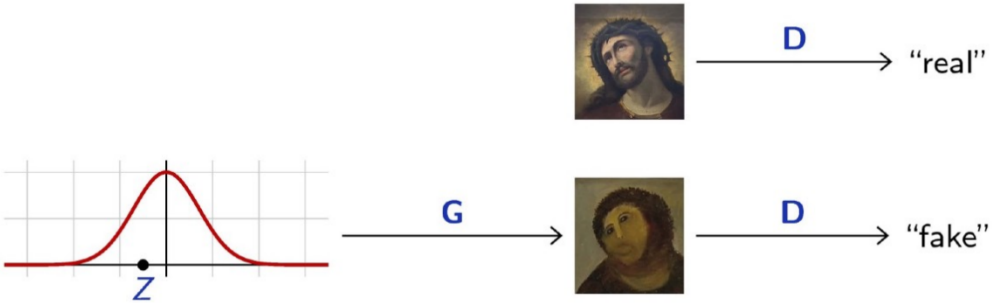


A GAN model diagram[1]

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. [Source: F. Fleuret, "Deep Learning" (EE-559), 11.1 Generative Adversarial Networks]

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

GANs: **Training**

Training

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

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

- 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).

Training (cont.)

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.

[Goodfellow et al., 2014]

Training (cont.)

The Adversarial Process:

1. Discriminator is trained to distinguish real from fake.
2. Generator is trained to fool the discriminator.
3. Repeat: Alternate steps until convergence.

Training Tips:

- Use batch normalization.
- Use logits with BCE-with-logits for numerical stability (WGAN critics use a linear, no-sigmoid output).
- Use Leaky ReLU in discriminator.

Practical Issues:

- Discriminator becomes too strong.
- Gradient vanishing for generator.
- Careful balance of training speeds is necessary.

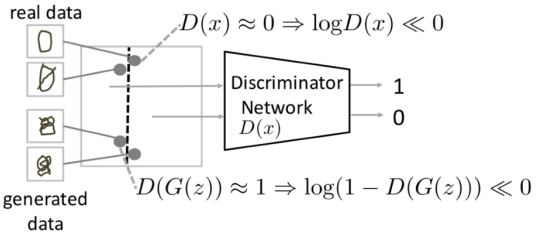
GANs: **Objective function**

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

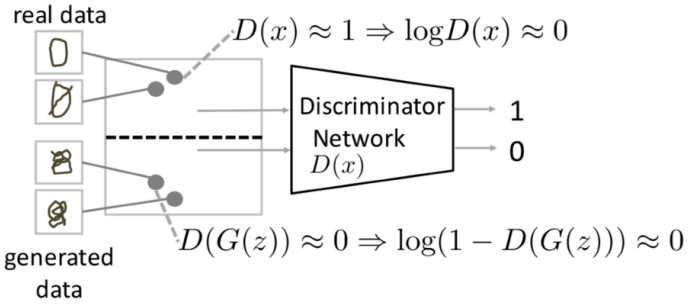


[Figure source: Mark Chang, "Generative Adversarial Networks" (slides)]

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



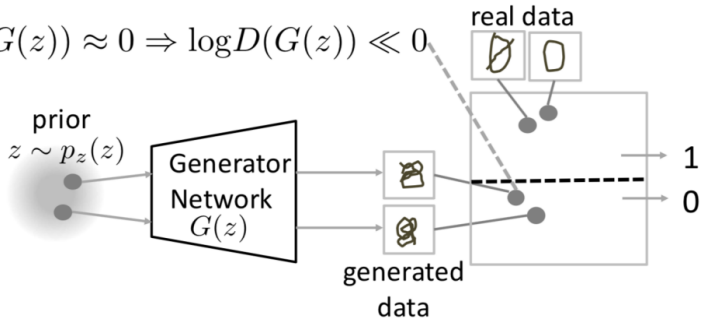
[Figure source: Mark Chang, "Generative Adversarial Networks" (slides)]

Understanding the Objective function (cont.)

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

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

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



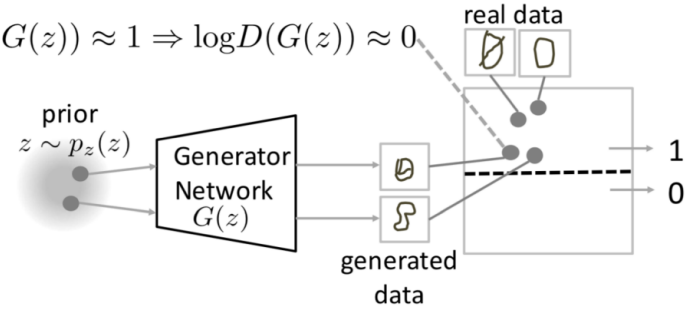
[Figure source: Mark Chang, "Generative Adversarial Networks" (slides)]

Understanding the Objective function (cont.)

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

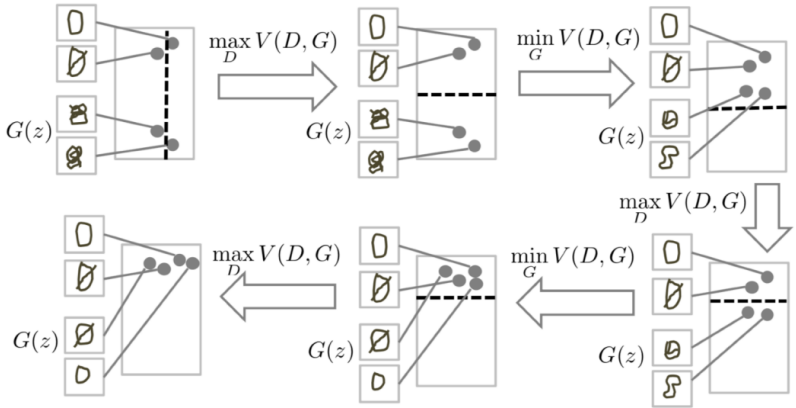
$D(G(z))$ should be 1

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



[Figure source: Mark Chang, "Generative Adversarial Networks" (slides)]

Understanding the Objective function (cont.)



[Figure source: Mark Chang, "Generative Adversarial Networks" (slides)]

GANs - Interactive Demo

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

GANs: **Results**

Results

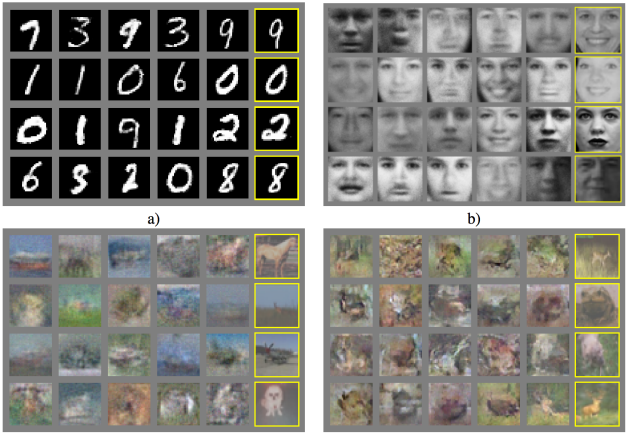
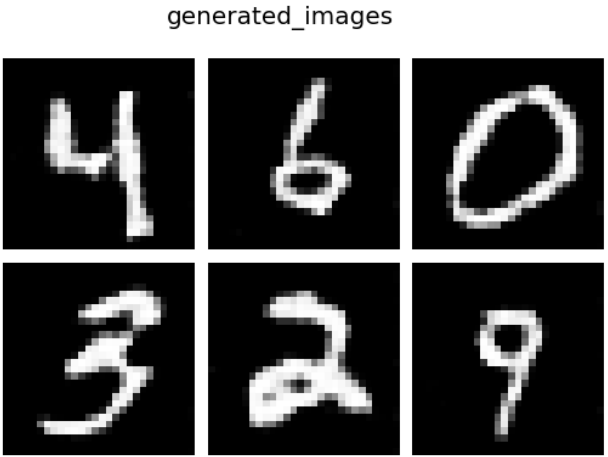


Figure from Goodfellow et al 2014, showing GAN results on various datasets.

Results (cont.)



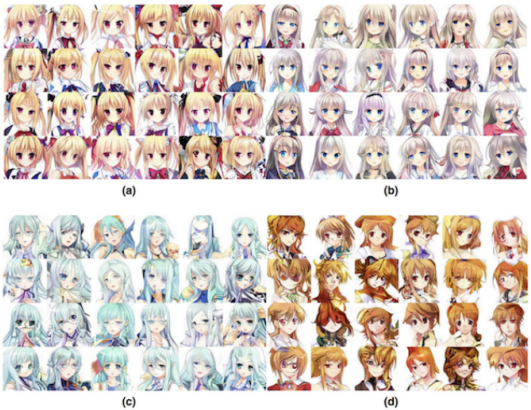
GAN generated samples for MNIST digits dataset

Results (cont.)



GAN generated samples for bedroom images. [Radford, Metz & Chintala, ICLR 2016, Fig. 3]

Results (cont.)



GAN generated samples for anime character faces. [Jin et al., "Towards the Automatic Anime Characters Creation with GANs," 2017, Fig. 9]

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.

GAN Variants (cont.)

- **Conditional GAN (cGAN):**
  - Conditions both the generator and discriminator on additional information (e.g., class labels).
  - Useful for targeted generation.
- **Pix2Pix:**
  - cGAN framework for supervised, paired image-to-image translation.
  - U-Net generator with a PatchGAN discriminator.
- **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.

GAN Variants (cont.)

- Allows fine-grained control over image features.

GAN Variant:

Deep Convolutional GAN (DCGAN)

UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS

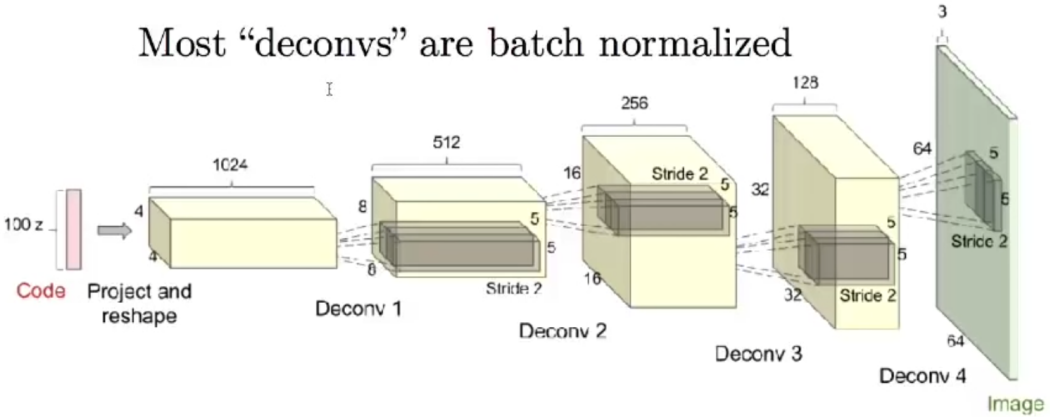
Alec Radford & Luke Metz
indico Research
Boston, MA
{alec,luke}@indico.io

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]

DCGAN (cont.)

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 (no pooling layers).
  - 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.

DCGAN (cont.)

- 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

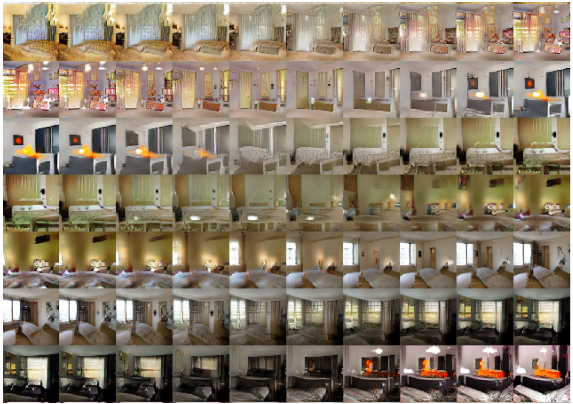
DCGAN: Results (cont.)



[Radford et al 2016]

DCGAN: Results (cont.)

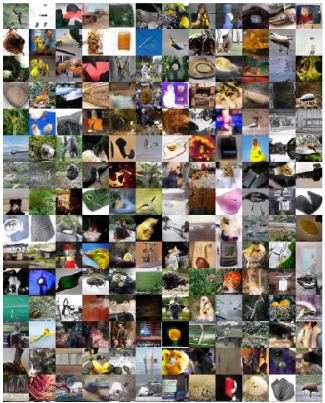
Smooth interpolations in high dimensions



[Radford et al 2016]

DCGAN: Results (cont.)

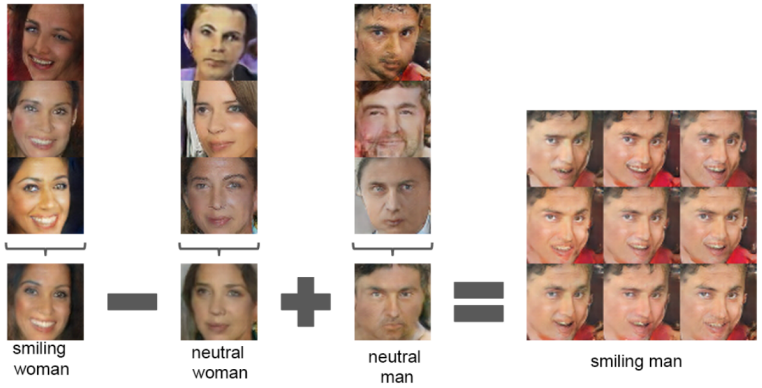
Imagenet samples (32x32)



[Radford et al 2016]

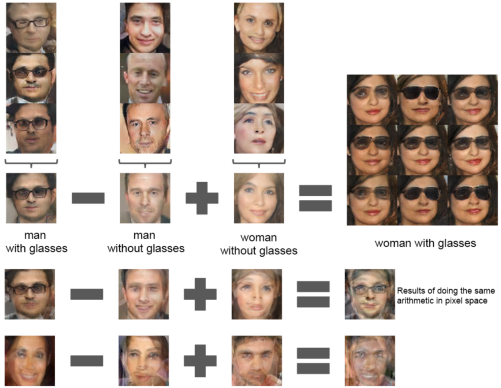
DCGAN: Results (cont.)

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% ($\pm$0.7%)	4800
3 Layer K-means Learned RF	82.0%	70.7% ($\pm$0.7%)	3200
View Invariant K-means	81.9%	72.6% ($\pm$0.7%)	6400
Exemplar CNN	84.3%	77.4% ($\pm$0.2%)	1024
DCGAN (ours) + L2-SVM	82.8%	73.8% ($\pm$0.4%)	512

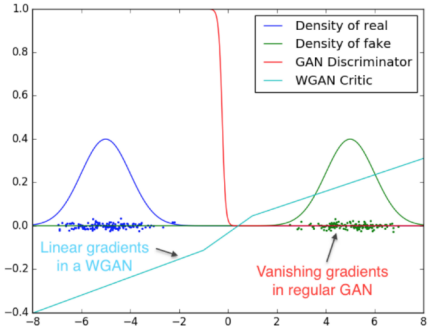
[Radford et al 2016]

GAN Variant:

Wasserstein GANs (WGANs)

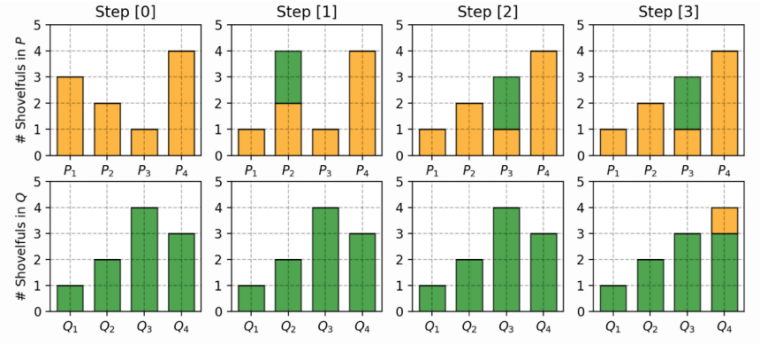
Wasserstein GAN

- Wasserstein GAN uses the Wasserstein distance instead of the cross-entropy loss.
- The Wasserstein distance has a smoother gradient everywhere.



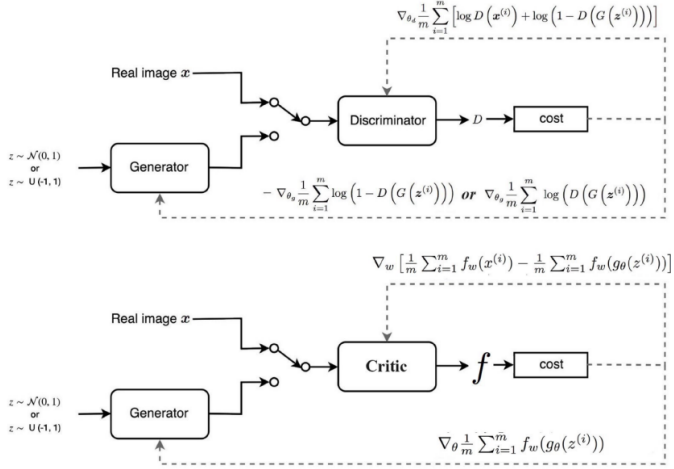
[Arjovsky, Chintala & Bottou, "Wasserstein GAN," 2017, Fig. 2]

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



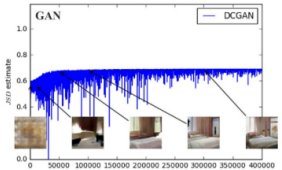
Step by Step plan of moving dirt between piles P and Q to make them match. [Source: L. Weng, "From GAN to WGAN," Fig. 7, https://lilianweng.github.io/posts/2017-08-20-gan/]

WGAN

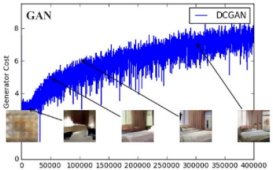


[Source: J. Hui, "GAN: Wasserstein GAN and WGAN-GP," Medium, 2018]

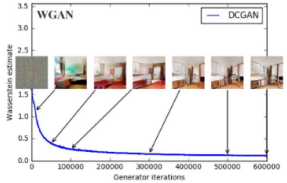
WGAN (cont.)



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



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



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

[Panels from Arjovsky, Chintala & Bottou, "Wasserstein GAN," 2017, Figs. 3, 4 and 8]

WGAN (cont.)



WGAN generation results on bedroom images. [Gulrajani et al 2017, Fig. 4]

WGAN-GP: Gradient Penalty for Lipschitzness

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

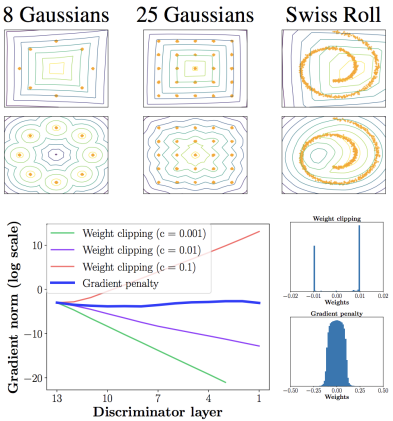
Generative Adversarial Networks (GANs) are powerful generative models, but suffer from training instability. The recently proposed Wasserstein GAN (WGAN) makes progress toward stable training of GANs, but sometimes can still generate only poor samples or fail to converge. We find that these problems 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.)

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

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



[Gulrajani et al 2017, Fig. 1 and Algorithm 1]

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]

WGAN-GP: BatchNorm

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]

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

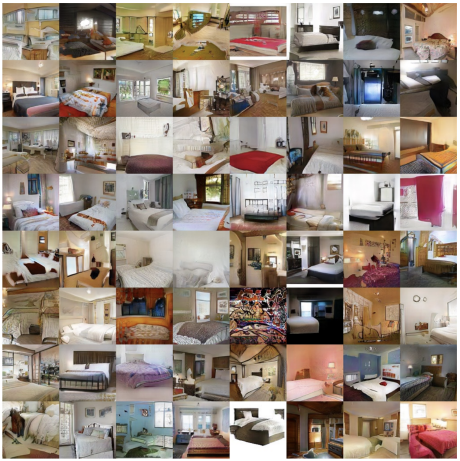
[Gulrajani et al 2017]

WGAN-GP: Robustness to architectures (cont.)



[Gulrajani et al 2017]

WGAN-GP: 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]	$8.59 \pm .12$

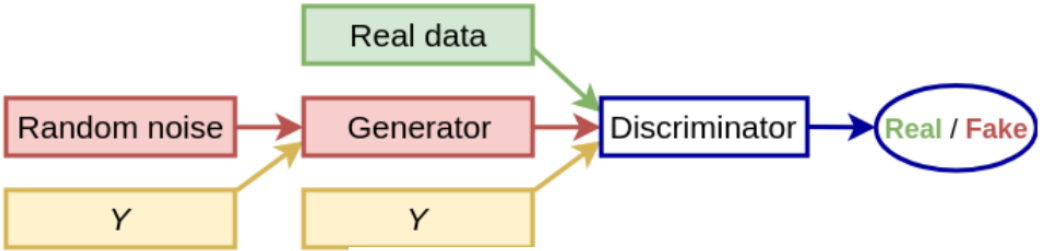
[Gulrajani et al 2017]

GAN Variant:

Conditional GANs

Conditional GAN

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



[Source: Busayor, "The Story of DCGANs, WGAN and CGANs," Analytics Vidhya, 2021]

Conditional GAN (cont.)

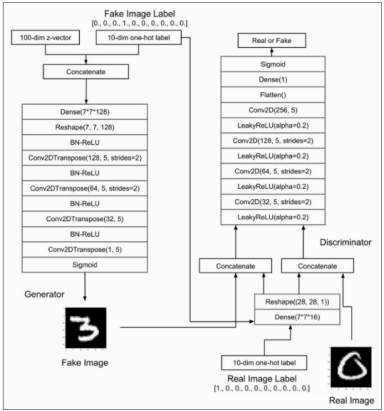
- Discriminator Loss:

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

- Generator Loss:

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

Conditional GAN (cont.)



A conditional GAN architecture for the MNIST digits dataset. [Source: Busayor, "The Story of DCGANs, WGAN and CGANs," Analytics Vidhya, 2021]

GAN Variant:

Pix2Pix GANs

Motivation:

- Many computer vision tasks require translating one type of image to another:
  - Sketch $\rightarrow$ Photo
  - Day $\rightarrow$ Night
  - Black & white $\rightarrow$ Color
  - Aerial photo $\rightarrow$ Map
- Traditional methods rely on paired datasets (input-output examples), manual labeling, or hand-crafted heuristics.
- Generative Adversarial Networks (GANs) have shown great success in generating realistic images.
- Conditional GANs (cGANs) extend GANs to perform image-to-image translation by conditioning on input images.

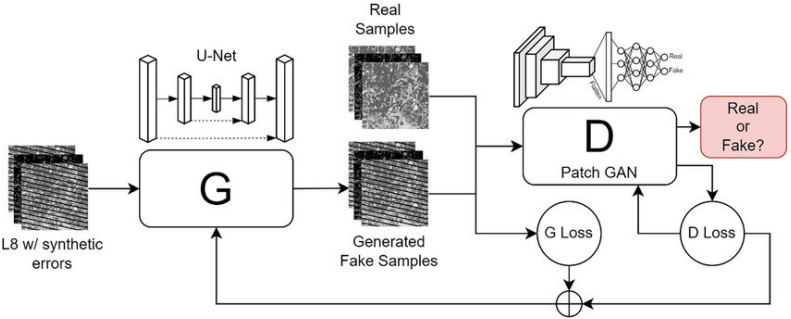
What is Pix2Pix?

- Pix2Pix is a Conditional GAN (cGAN) framework for supervised image-to-image translation.
- Introduced in the paper: *Image-to-Image Translation with Conditional Adversarial Networks* by Isola et al., 2017.

Pix2Pix GANs (cont.)

Architecture:

- **Generator (G):** U-Net (encoder-decoder with skip connections) that transforms the input image (e.g., sketch) into the target image (e.g., photo).
- **Discriminator (D):** PatchGAN classifier that determines whether a patch of the image is real or fake, enforcing local realism.



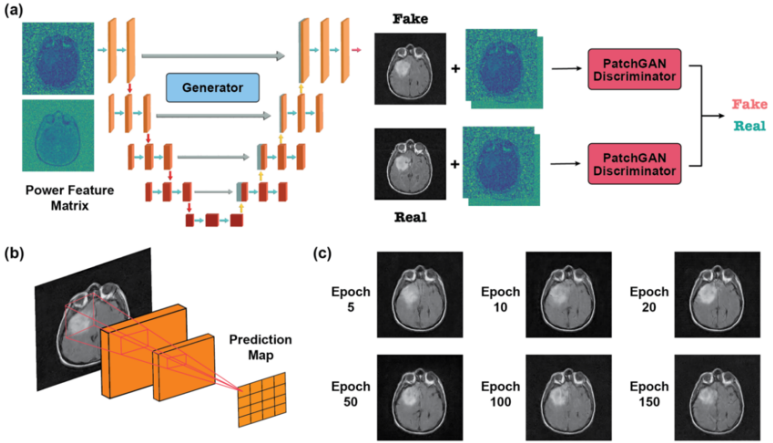
Pix2Pix GANs (cont.)

[Source: https://www.researchgate.net/publication/383558465]

How It Works:

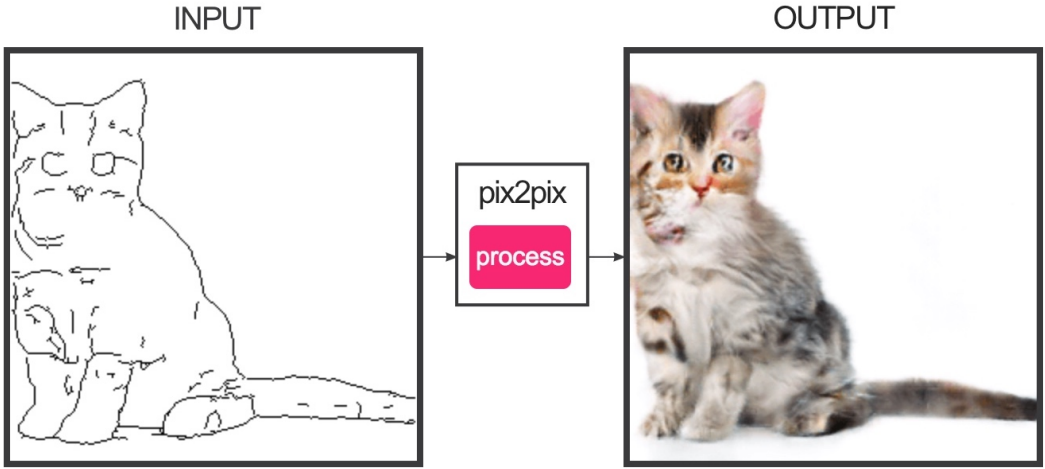
- The generator learns to map an input image to a target image.
- The discriminator distinguishes between:
  - Real pairs: (input, real output)
  - Fake pairs: (input, generated output)
- **Loss Function:**
  - **Adversarial Loss:** Encourages the output to look realistic.
  - **L1 Loss:** Encourages the output to be close to the target image (pixel-wise similarity).
- **Total Loss:** Adversarial Loss $+ \lambda \times$ L1 Loss
  (λ is a weight to balance realism and similarity)

Pix2Pix GANs (cont.)



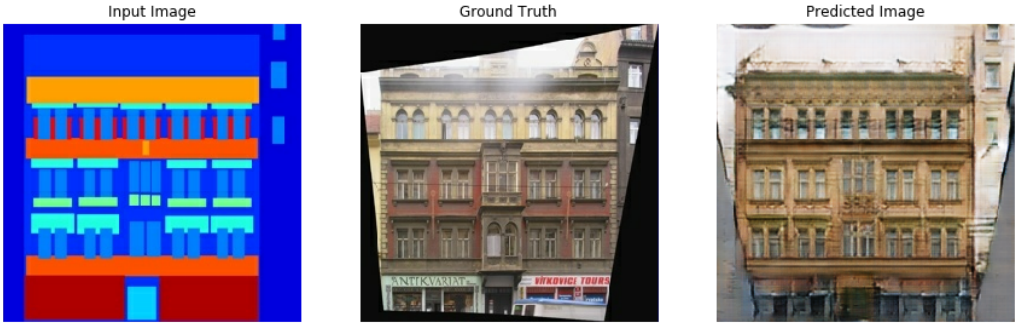
[Source: https://www.researchgate.net/publication/370213457]

Pix2Pix GANs: Results



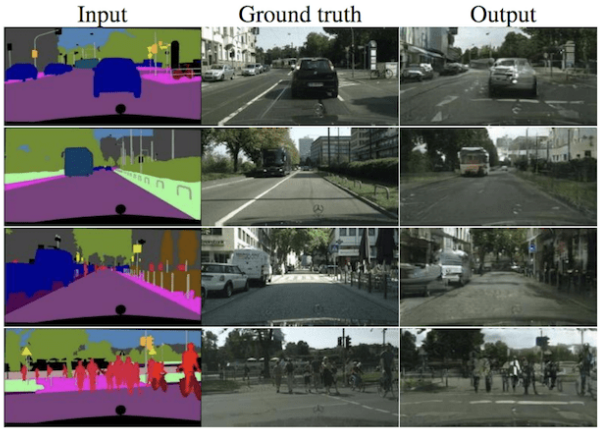
[Isola, Zhu, Zhou & Efros, pix2pix project page, https://phillipi.github.io/pix2pix/]

Pix2Pix GANs: Results (cont.)



[Source: TensorFlow pix2pix tutorial, https://www.tensorflow.org/tutorials/generative/pix2pix]

Pix2Pix GANs: Results (cont.)



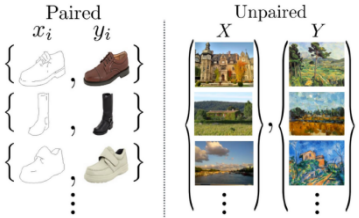
[Source: J. Brownlee, MachineLearningMastery.com, https://machinelearningmastery.com]

GAN Variant:

Cycle GANs

Cycle GANs

- Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, Zhu, Park, Isola, Efros, ICCV 2017



Cycle Consistency

- **Cycle Consistency Loss:** Ensures that translating an image to the target domain and back yields the original image.
- For $x \in X$: $x \rightarrow G(x) = \hat{y} \rightarrow F(\hat{y}) \approx x$
- **Mathematically:**

$$\mathcal{L}_{cyc}(G, F) = \mathbb{E}_{x \sim p_{data}(x)} \left[\| F(G(x)) - x \|_1 \right] + \mathbb{E}_{y \sim p_{data}(y)} \left[\| G(F(y)) - y \|_1 \right]$$

Cycle GANs (cont.)

Total Loss

- **Adversarial Loss:** Makes generated images indistinguishable from real images.
- **Cycle Consistency Loss:** Enforces reversibility of translation.
- **(Optional) Identity Loss:** Preserves color or structure.
- **Total Loss:**

$$\mathcal{L}_{total} = \mathcal{L}_{GAN}(G, D_Y, X, Y) + \mathcal{L}_{GAN}(F, D_X, Y, X) + \lambda \mathcal{L}_{cyc}(G, F) + \gamma \mathcal{L}_{identity}$$

Cycle GANs (cont.)

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



[Source: J. Brownlee, "How to Develop a CycleGAN for Image-to-Image Translation with Keras," MachineLearningMastery.com]

Cycle GANs (cont.)

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



[Source: J. Brownlee, "How to Develop a CycleGAN for Image-to-Image Translation with Keras," MachineLearningMastery.com]

Cycle GANs (cont.)

$G_{zebra \rightarrow horse}$



[Source: J. Brownlee, "How to Develop a CycleGAN for Image-to-Image Translation with Keras," MachineLearningMastery.com]

Cycle GANs (cont.)

$G_{horse \rightarrow zebra}$



[Source: J. Brownlee, "How to Develop a CycleGAN for Image-to-Image Translation with Keras," MachineLearningMastery.com]

Cycle GANs (cont.)

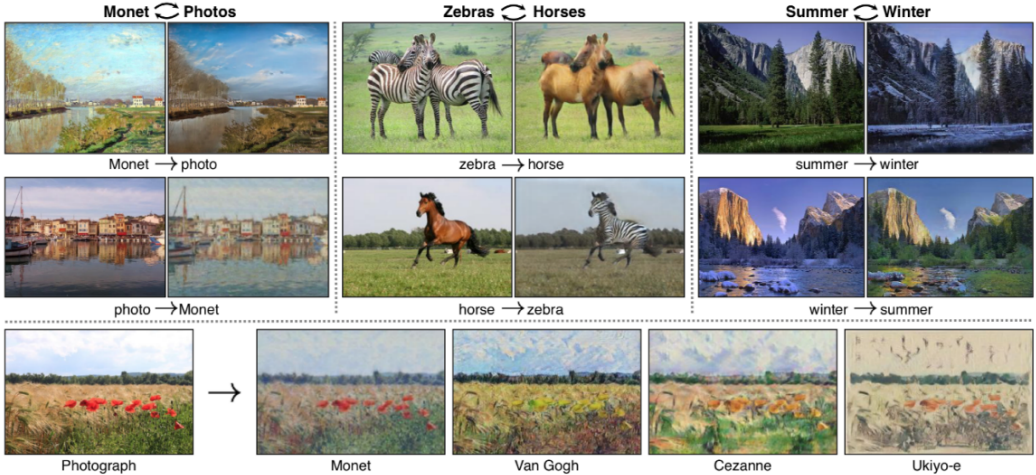
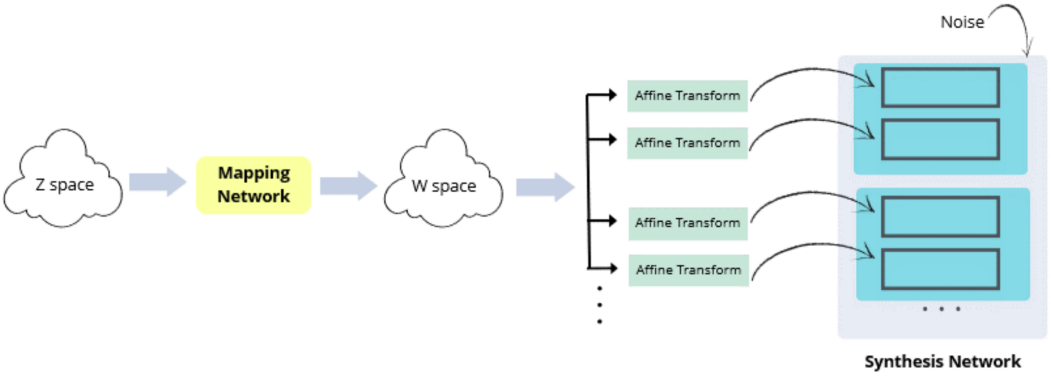


Image to Image translation with Cycle GANs. [Zhu, Park, Isola & Efros, ICCV 2017, Fig. 1]

GAN Variant:

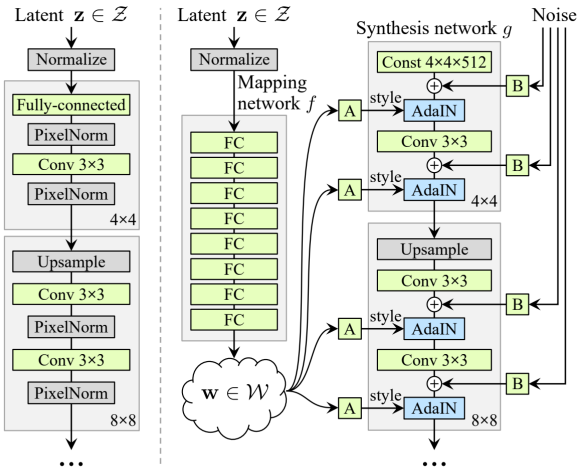
Style GANs

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



[1] https://www.cs.unc.edu/~ronisen/teaching/fall_2022/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**nstance **N**ormalization (very effective in controlling styles)

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

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

Which Latent space to choose for embedding and editing?

- **Z** : 512 dimensional latent space (not good)
- **W** : 512 dimensional latent space (better but not perfect)
- **W+** : 18x512 dimensional latent space (a different 512-d w vector for each of the 18 generator layer inputs)
- **W** is better for editing.
- **W+** is better for reconstruction or embedding of real images.

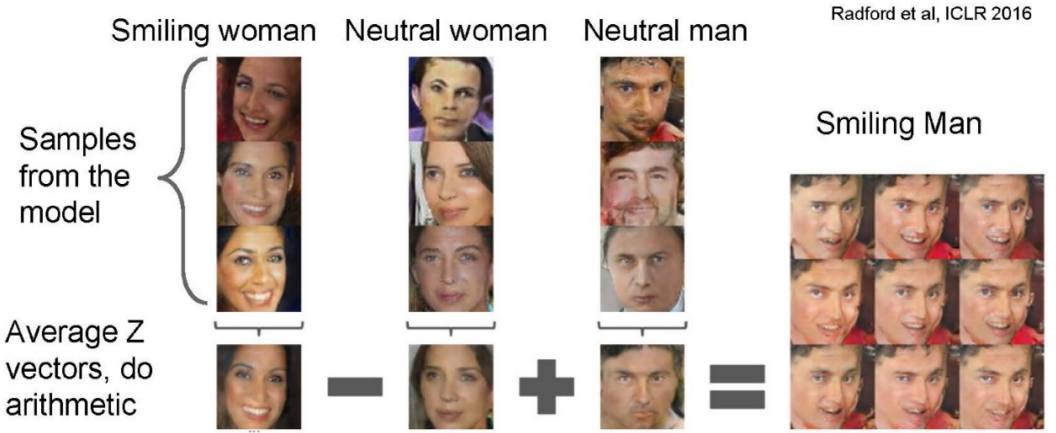
Style GANs - Results



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

GANs: **Latent space Interpolation**

Latent space Interpolation



Latent space Interpolation (cont.)

Latent space interpolation with GANs

Latent space Interpolation (cont.)

Latent space interpolation with StyleGAN
(Demo by Xander Steenbrugge)

https://colab.research.google.com/drive/1mH70YxGNlnEaSOn0J8Lsgkl-QOvsIb3M

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.

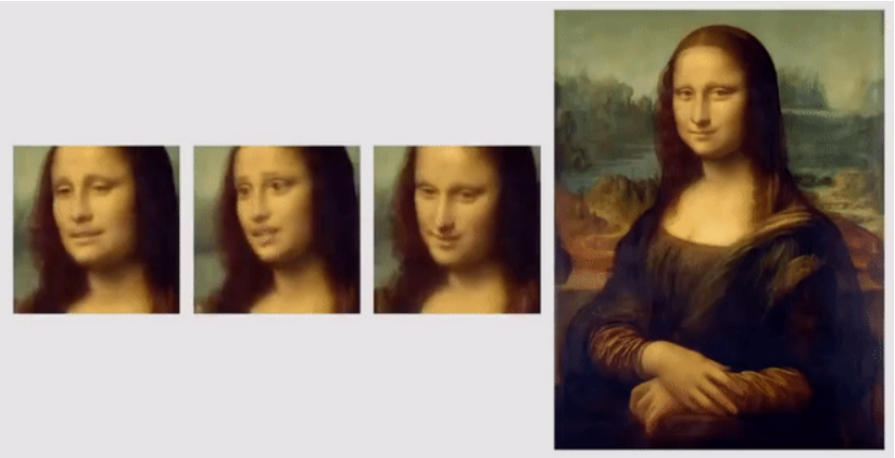


this white and yellow flower have thin white petals and a round yellow stamen



Text to Image Synthesis with GANs. [Reed et al., "Generative Adversarial Text to Image Synthesis," ICML 2016, Fig. 1]

Some more GAN results (cont.)



Living Portraits with GANs. [Zakharov et al., "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models," ICCV 2019]

Some more GAN results (cont.)

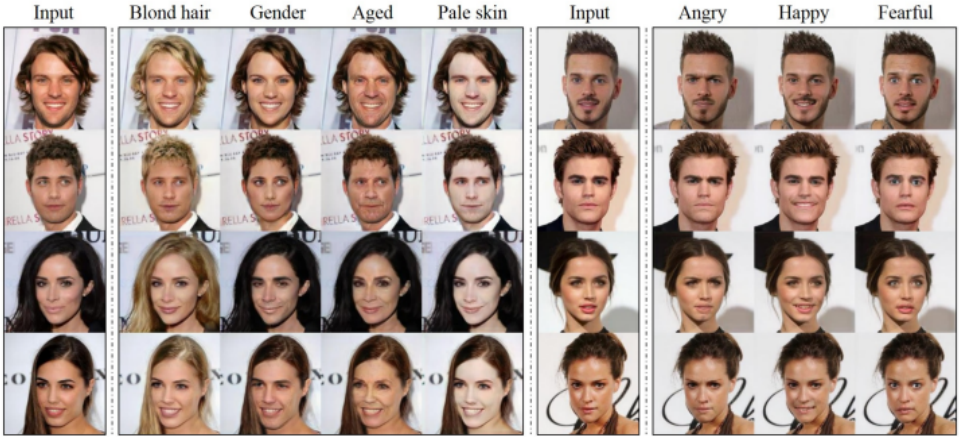


Image to Image translation in multiple domains with StarGAN (Choi et al., CVPR 2018)

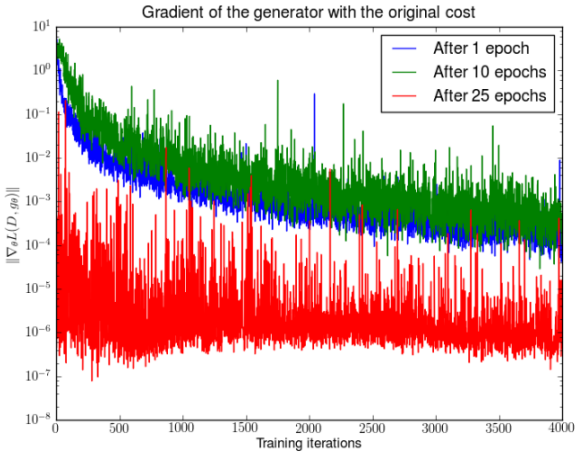
GANs: **Problems**

Vanishing Gradients

- When the discriminator is perfect, we are guaranteed with $D(x) = 1$ for all real samples x and $D(x) = 0$ for all generated samples.
- 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:

$$\max_{\theta_G} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_D}(G_{\theta_G}(z)))$$

Problems with GANs (cont.)



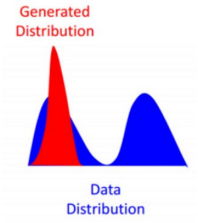
Vanishing gradient in GANs (Image source: Arjovsky and Bottou, 2017)

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.



[Source: https://www.cnblogs.com/recoverableTi/p/13452479.html]

Problems with GANs (cont.)



GAN mode collapse on MNIST digits dataset. [Metz et al., "Unrolled Generative Adversarial Networks," ICLR 2017, Fig. 3]

GANs: **Summary**

Summary

- Instead of explicitly learning the 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 et al., "Generative Models," Stanford CS231n
- Francois Fleuret, "Deep Learning," EE559
- Murtaza Taj, "Deep Learning," CS437
- Stefano Ermon and Aditya Grover, "Deep Generative Models," Stanford CS236
- Mark Chang, "Generative Adversarial Networks" (slides)
- Lilian Weng, "From GAN to WGAN" (lilianweng.github.io)
- Jonathan Hui, "GAN: Wasserstein GAN and WGAN-GP," Medium, 2018 (jonathan-hui.medium.com)
- Jason Brownlee, "How to Develop a CycleGAN for Image-to-Image Translation with Keras" (machinelearningmastery.com)
- Busayor, "The Story of DCGANs, WGAN and CGANs," Analytics Vidhya, 2021 (medium.com)

References (Papers)

- Goodfellow et al., "Generative Adversarial Networks," 2014. arXiv:1406.2661
- Mirza and Osindero, "Conditional Generative Adversarial Nets," 2014. arXiv:1411.1784
- Radford, Metz, and Chintala, "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks," ICLR 2016. arXiv:1511.06434
- Arjovsky et al., "Wasserstein GAN," 2017. arXiv:1701.07875
- Gulrajani et al., "Improved Training of Wasserstein GANs," NIPS 2017. arXiv:1704.00028
- Isola et al., "Image-to-Image Translation with Conditional Adversarial Networks," CVPR 2017. arXiv:1611.07004
- Zhu et al., "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks," 2017. arXiv:1703.10593
- Reed et al., "Generative Adversarial Text to Image Synthesis," ICML 2016. arXiv:1605.05396

References (Papers) (cont.)

- Metz, Poole, Pfau, and Sohl-Dickstein, "Unrolled Generative Adversarial Networks," ICLR 2017. arXiv:1611.02163
- Jin et al., "Towards the Automatic Anime Characters Creation with Generative Adversarial Networks," 2017. arXiv:1708.05509
- Karras et al., "A Style-Based Generator Architecture for Generative Adversarial Networks," CVPR 2019. arXiv:1812.04948
- Zakharov et al., "Few-Shot Adversarial Learning of Realistic Neural Talking Head Models," ICCV 2019. arXiv:1905.08233
- Creswell et al., "Generative Adversarial Networks: An Overview," IEEE Signal Processing Magazine, 2018. IEEE Xplore