Administrative

Grading (course 1)

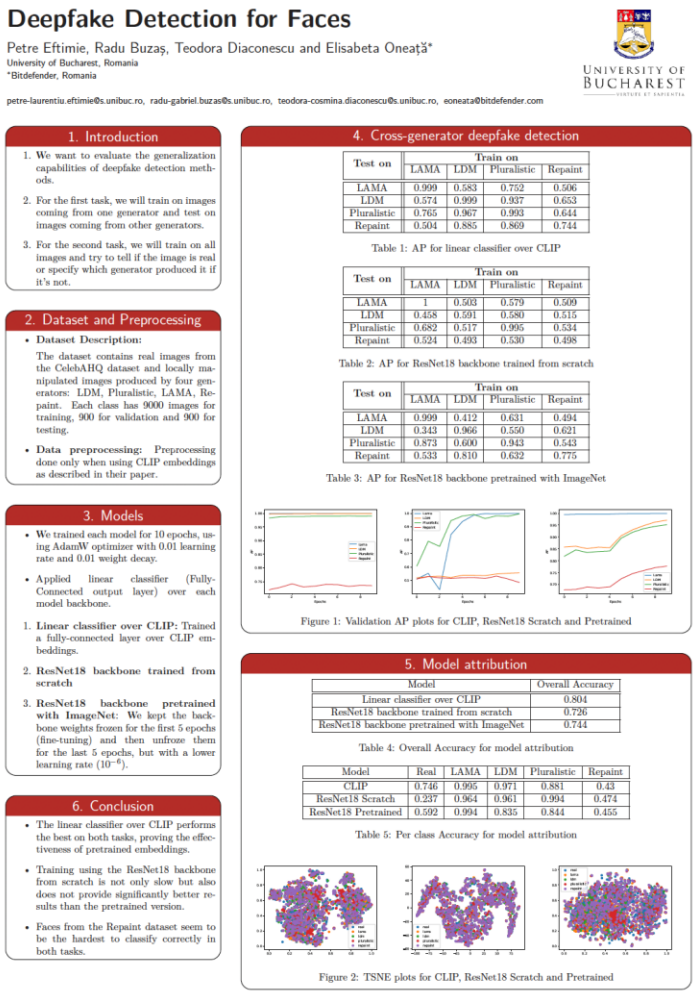
		Requirements
Assignment 1	2 points	**50% (2.5 points)**
Assignment 2	2 points	
[**BONUS**] Good answers / questions during lectures and labs	max 1 points	
[BONUS] EEML submission	max 2 points	
Project	5 points	**50% (2.5 points)**
[**BONUS**] Good interventions / questions during the poster session	max 1 point	

Project - Poster Session

1. Introduction
2. Dataset Description
3. **Method (focus on this)**
   + NN Architecture
   + Algorithm steps
1. **Experiments (focus on this)**
   + Performance
   + Qualitative Results
1. Conclusions
   + Encountered problems
   + Future work

In-person session
- A2, portrait

Ex.: Latex Templates(NLP)
Various Latex Templates



Deepfake Detection for Faces

Petre Eftimie, Radu Buzaș, Teodora Diaconescu and Elisabeta Oneață*
University of Bucharest, Romania
*Bitdefender, Romania
petre-laurentiu.eftimie@s.unibuc.ro, radu-gabriel.buzas@s.unibuc.ro, teodora-cosmina.diaconescu@s.unibuc.ro, eoneata@bitdefender.com

UNIVERSITY OF BUCHAREST

1. Introduction

1. We want to evaluate the generalization capabilities of deepfake detection methods.
2. For the first task, we will train on images coming from one generator and test on images coming from other generators.
3. For the second task, we will train on all images and try to tell if the image is real or specify which generator produced it if it's not.

2. Dataset and Preprocessing

- **Dataset Description:**
  The dataset contains real images from the CelebAHQ dataset and locally manipulated images produced by four generators: LDM, Pluralistic, LAMA, Repaint. Each class has 9000 images for training, 900 for validation and 900 for testing.
- **Data preprocessing:** Preprocessing done only when using CLIP embeddings as described in their paper.

3. Models

- We trained each model for 10 epochs, using AdamW optimizer with 0.01 learning rate and 0.01 weight decay.
- Applied linear classifier (Fully-Connected output layer) over each model backbone.

1. **Linear classifier over CLIP:** Trained a fully-connected layer over CLIP embeddings.
2. **ResNet18 backbone trained from scratch**
3. **ResNet18 backbone pretrained with ImageNet:** We kept the backbone weights frozen for the first 5 epochs (fine-tuning) and then unfroze them for the last 5 epochs, but with a lower learning rate (10^{-6}).

6. Conclusion

- The linear classifier over CLIP performs the best on both tasks, proving the effectiveness of pretrained embeddings.
- Training using the ResNet18 backbone from scratch is not only slow but also does not provide significantly better results than the pretrained version.
- Faces from the Repaint dataset seem to be the hardest to classify correctly in both tasks.

4. Cross-generator deepfake detection

Test on	Train on LAMA	LDM	Pluralistic	Repaint
LAMA	0.999	0.583	0.752	0.506
LDM	0.574	0.999	0.937	0.653
Pluralistic	0.765	0.967	0.993	0.644
Repaint	0.504	0.885	0.869	0.744

Table 1: AP for linear classifier over CLIP

Test on	Train on LAMA	LDM	Pluralistic	Repaint
LAMA	1	0.503	0.579	0.509
LDM	0.458	0.591	0.580	0.515
Pluralistic	0.682	0.517	0.995	0.534
Repaint	0.524	0.493	0.530	0.498

Table 2: AP for ResNet18 backbone trained from scratch

Test on	Train on LAMA	LDM	Pluralistic	Repaint
LAMA	0.999	0.412	0.631	0.494
LDM	0.343	0.966	0.550	0.621
Pluralistic	0.873	0.600	0.943	0.543
Repaint	0.533	0.810	0.632	0.775

Table 3: AP for ResNet18 backbone pretrained with ImageNet

Figure 1: Validation AP plots for CLIP, ResNet18 Scratch and Pretrained

5. Model attribution

Model	Overall Accuracy
Linear classifier over CLIP	0.804
ResNet18 backbone trained from scratch	0.726
ResNet18 backbone pretrained with ImageNet	0.744

Table 4: Overall Accuracy for model attribution

Model	Real	LAMA	LDM	Pluralistic	Repaint
CLIP	0.746	0.995	0.971	0.881	0.43
ResNet18 Scratch	0.237	0.964	0.961	0.994	0.474
ResNet18 Pretrained	0.592	0.994	0.835	0.844	0.455

Table 5: Per class Accuracy for model attribution

Figure 2: TSNE plots for CLIP, ResNet18 Scratch and Pretrained

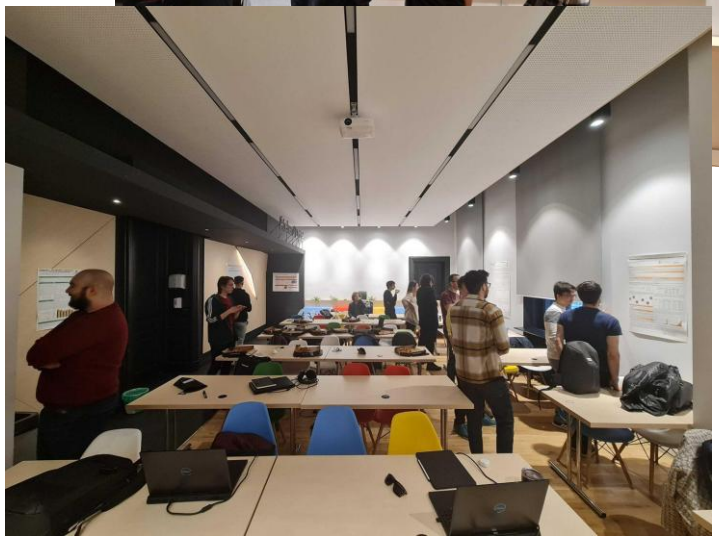
Project - Poster Session

Evaluation based on:

- The model is training, the dataset is preprocessed, pytorch and conda worked! (yuhu)
- Ablation study (vary the components of the base model)
- Answers at the presentation moment (each team member should know all the content of the project)

All team members will have the **same grade** for the project.

Follow your mentor's instructions!



BRAIT competition @ Romanian AI Days

Best Romanian AI Thesis Competition

https://days.airomania.eu/brait2025

🗓️ Competition Calendar

- 🚀 Submissions Open 30 April 2025
- ⏳ **Submission Deadline 30 June 2025**
- 🎯 Decisions Announced 10 September 2025

📢 Competition Submission Guidelines

📌 See Submission Guidelines page.

📄 What to Submit?

✅ Short Paper (4 pages) summarizing your thesis contributions

✅ Bachelor's / Master's / PhD Thesis

Deep Learning

9. Generative Models: VAE, GAN, Diffusion Models

Lecture Overview

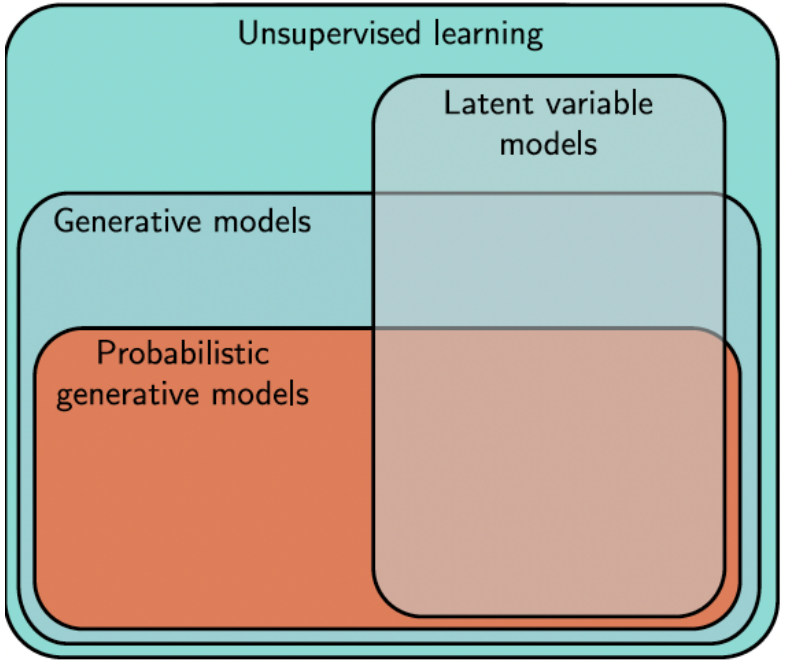
- **Preliminaries**
  - Unsupervised Learning
  - Latent Variable Models
  - What makes a good generative model?
- **Generative Models**
  - Variational Autoencoder
  - Generative Adversarial Networks
  - Diffusion Models

Unsupervised models

Understanding Deep Learning, Simon J.D. Prince (2023)

Unsupervised models

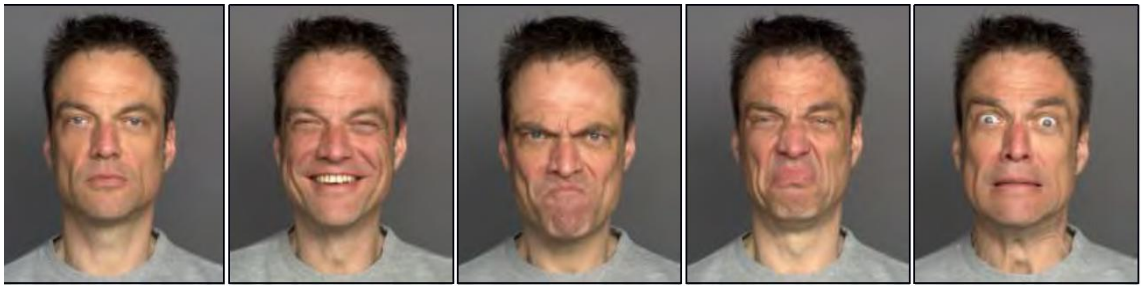
- Are learned from a set of observed data in the absence of labels
- They may have diverse goals:
  - **Generate** new samples
  - Manipulate, denoise, interpolate between, compress samples
  - Reveal the internal structure of the dataset



Understanding Deep Learning, Simon J.D. Prince (2023)

Latent Variable Models

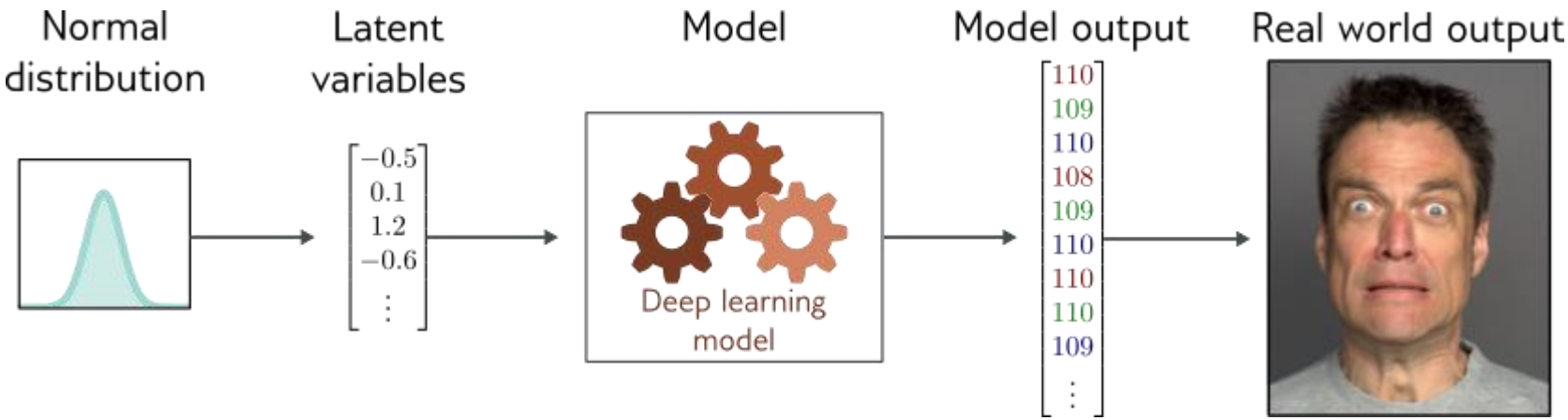
- Real-world data often lies on a **lower-dimensional manifold** than raw input space suggests.
- Latent variables represent the **underlying structure generating** the data (e.g., sentence structure, physical image processes).



- The human face contains roughly 42 muscles.
- It is possible to describe most of the variation in images of the same person in the same lighting with just 42 numbers.

Understanding Deep Learning, Simon J.D. Prince (2023)

Latent Variable Models



- Deep learning models learn the **mapping between latent space and data space**.
- Latent variables are typically drawn from a **simple prior distribution** (e.g., Gaussian).
- New data can be generated by **sampling from the latent space** and decoding it.
- Latent **interpolation** enables smooth transitions between real data points by blending their latent representations.

Understanding Deep Learning, Simon J.D. Prince (2023)

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.
- **Coverage:** Samples should represent the entire training distribution.

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.
- **Coverage:** Samples should represent the entire training distribution.
- **Well-behaved latent space**: Every latent variable z corresponds to a plausible data example x. Smooth changes in z correspond to smooth changes in x.

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.
- **Coverage:** Samples should represent the entire training distribution.
- **Well-behaved latent space**: Every latent variable z corresponds to a plausible data example x. Smooth changes in z correspond to smooth changes in x.
- **Disentangled latent space:** Manipulating each dimension of z should correspond to changing an interpretable property of the data. For example, in a model of language, it might change the topic, tense, or verbosity.

What makes a good generative model?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism of modern hardware.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.
- **Coverage:** Samples should represent the entire training distribution.
- **Well-behaved latent space**: Every latent variable z corresponds to a plausible data example x. Smooth changes in z correspond to smooth changes in x.
- **Disentangled latent space:** Manipulating each dimension of z should correspond to changing an interpretable property of the data. For example, in a model of language, it might change the topic, tense, or verbosity.
- **Efficient likelihood computation:** If the model is probabilistic, we would like to be able to calculate the probability of new examples efficiently and accurately.

Understanding Deep Learning, Simon J.D. Prince (2023)

Lecture Overview

- **Preliminaries**
  - Unsupervised Learning
  - Latent Variable Models
  - What makes a good generative model?
- **Generative Models**
  - Variational Autoencoder
  - Generative Adversarial Networks
  - Diffusion Models

VAE: Autoencoder

VAE

GAN

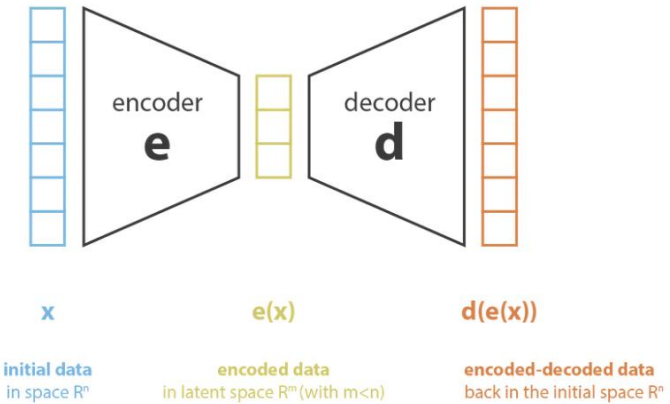
Diffusion Models

How:

- **Learn lower-dimensional feature representation** from **unlabelled training data**

Components:

- Encoder: z= encoder(x)
- Decoder: decoder(z) = decoder(encoder(x))



https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

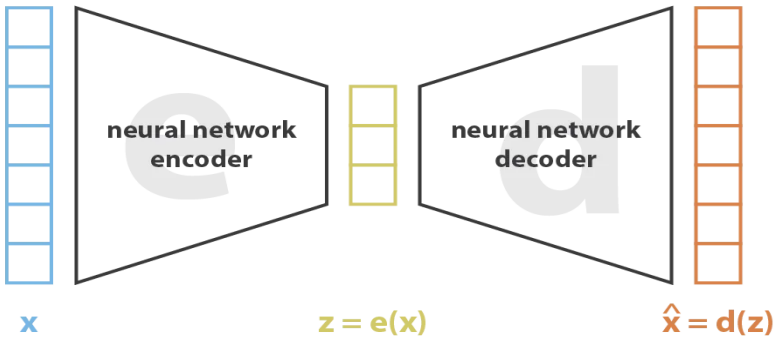
Autoencoder

VAE

GAN

Diffusion Models

Q: How to learn this new representation?



$$\text{loss} = \| x - \hat{x} \|^2 = \| x - d(z) \|^2 = \| x - d(e(x)) \|^2$$

Illustration of an autoencoder with its loss function.

https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73

Autoencoder

VAE

GAN

Diffusion Models

Q: How can we use the low-dimensional embedding?

- What part of the AE should we use?

Autoencoder

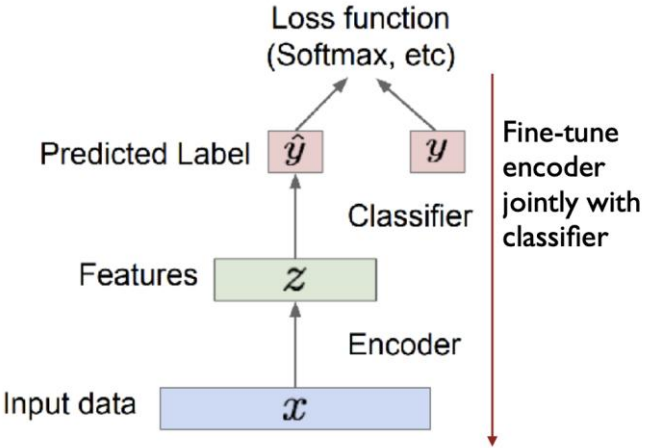
VAE

GAN

Diffusion Models

Q: **How can we use the low-dimensional embedding?**

- What part of the AE should we use?
- Use the pretrained code z as a **feature extractor**
- Input to a classifier with a smaller dataset



Autoencoder Examples

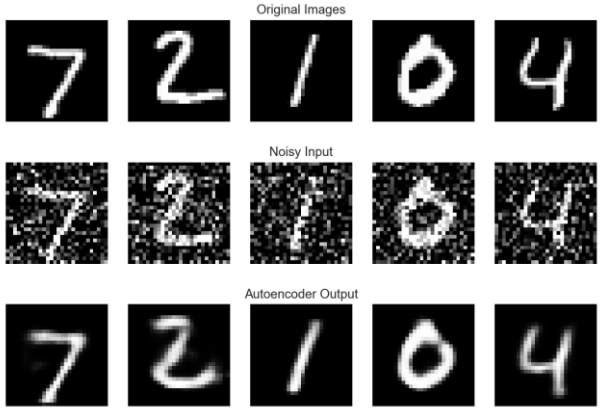
VAE

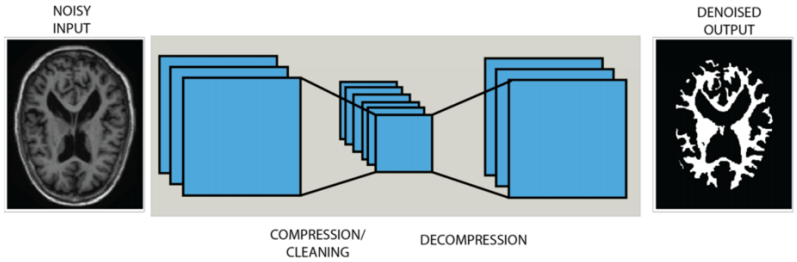
GAN

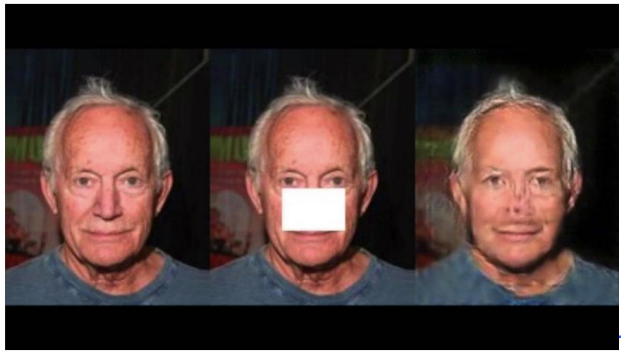
Diffusion Models

Example use cases:

- Denoising
- Inpainting
- SuperResolution
- Anomaly detection







https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798
https://faculty.cc.gatech.edu/~hays/7476/projects/Avery_Wenchen/

Autoencoder

VAE

GAN

Diffusion Models

Q: **Can we use autoencoders to generate NEW samples**?

- What part should we use here?

Q: **How can we interpolate between 2 inputs**?

Q: **What problems can appear**?

Autoencoder

VAE

GAN

Diffusion Models

Q: **Can we use autoencoders to generate NEW samples?**
- What part should we use here?
- z - random initialization
  or
- z - small variation

Q: **How can we interpolate between 2 inputs?**
- In the hidden space.

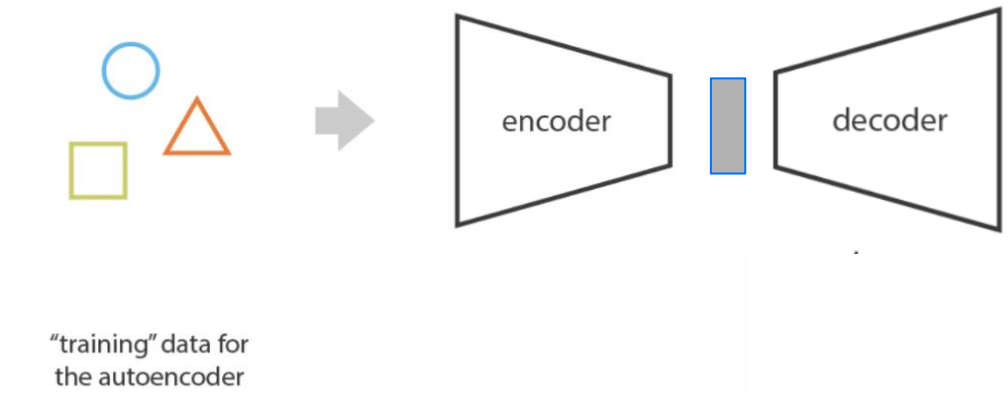
Q: **What problems can appear?**

Variational AutoEncoder (Latent Space Intuition)

VAE

GAN

Diffusion Models



https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder (Latent Space Intuition)

VAE

GAN

Diffusion Models

Difference between a “regular” and an “irregular” latent space.

https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder (Latent Space Intuition)

VAE

GAN

Diffusion Models

Difference between a “regular” and an “irregular” latent space.

- Desirable properties of the hidden space
  - **Continuity**: points close in hidden space -> similar content after decoding
  - **Completeness**: a point sampled from latent space -> “meaningful” content after decoding

https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder

VAE

GAN

Diffusion Models

VAE is an autoencoder whose encodings distribution is regularised during the training

Similar to AutoEncoder

- Starts from the simple autoencoder architecture

Different from AutoEncoder

- Explore variations on data you already have
- Map the input into a distribution (instead of a fixed vector z)
  - **mean** and **standard deviation** for a multivariate Gaussian
  - z ~ sample from this distribution
  - the basic autoencoder generates z directly

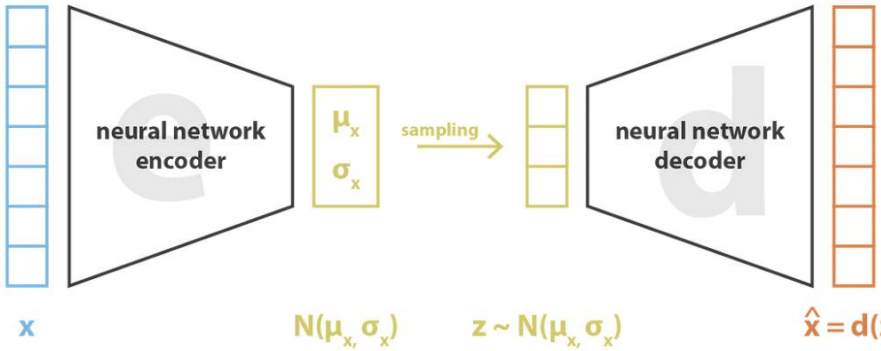
[Auto-Encoding Variational Bayes]

Variational AutoEncoder

VAE

GAN

Diffusion Models



$$\text{loss} = ||x - \hat{x}||^2 + KL[N(\mu_x, \sigma_x), N(0, I)] = ||x - d(z)||^2 + KL[N(\mu_x, \sigma_x), N(0, I)]$$

In variational autoencoders, the loss function is composed of a reconstruction term (that makes the encoding-decoding scheme efficient) and a regularisation term (that makes the latent space regular).

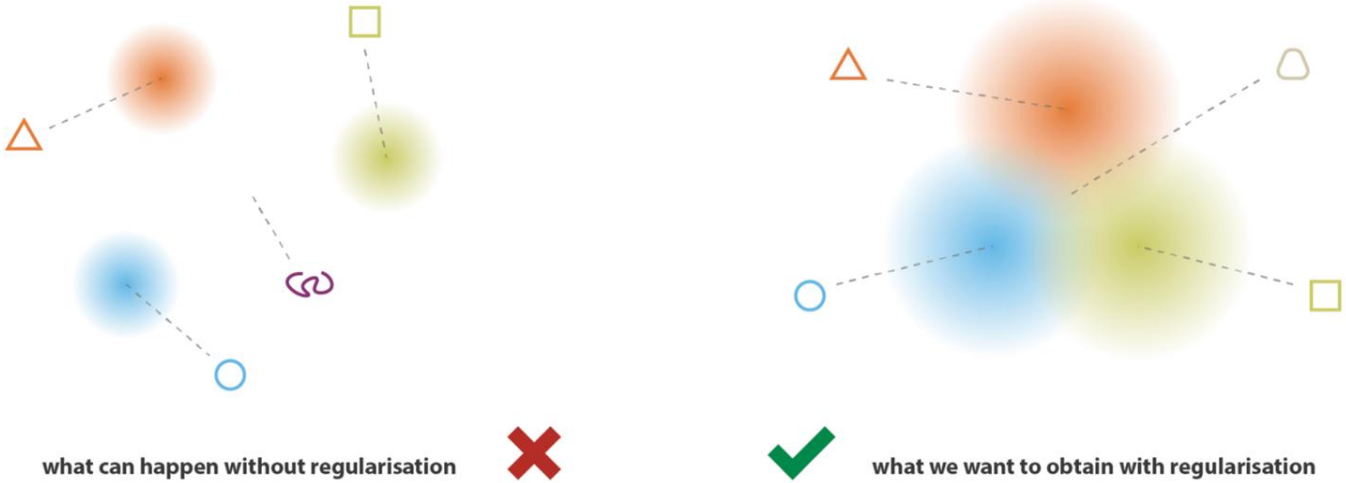
https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder

VAE

GAN

Diffusion Models



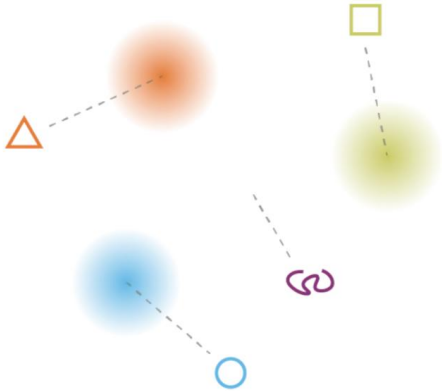
https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder

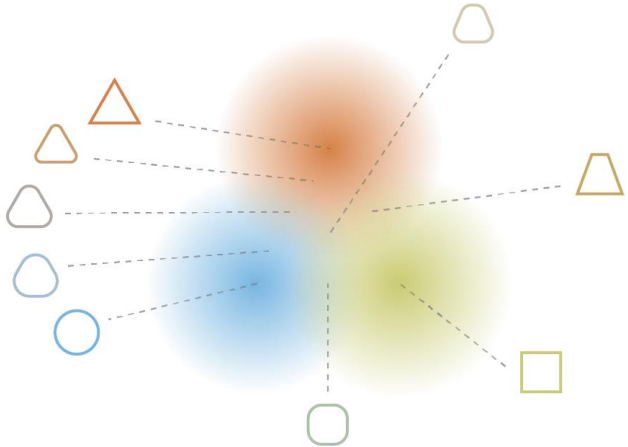
VAE

GAN

Diffusion Models



what can happen without regularisation



Regularisation tends to create a “gradient” over the information encoded in the latent space.

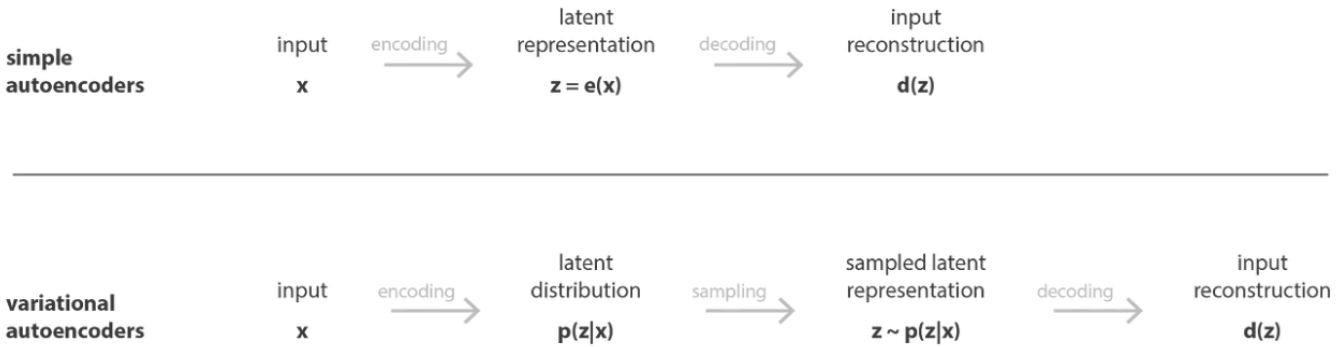
https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

Variational AutoEncoder

VAE

GAN

Diffusion Models



Difference between autoencoder (deterministic) and variational autoencoder (probabilistic).

https://medium.com/data-science/understanding-variational-autoencoders-vaes-f70510919f73

VAE - Faces in the Wild

VAE

GAN

Diffusion Models

https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Generative Adversarial Networks

VAE

GAN

Diffusion
Models

Generative Adversarial Networks

VAE

GAN

Diffusion Models

- **Unsupervised model** that aims generate new samples that are **indistinguishable** from a set of training examples.
- GANs are mechanisms to create new samples; **they do not build a probability distribution** over the modeled data

- Two networks:
  - **Generator** - Learns to generate plausible data
  - **Discriminator** - Learns to distinguish the real data from the generated ones

- During training, they both become better step by step, by getting penalized when they are wrong

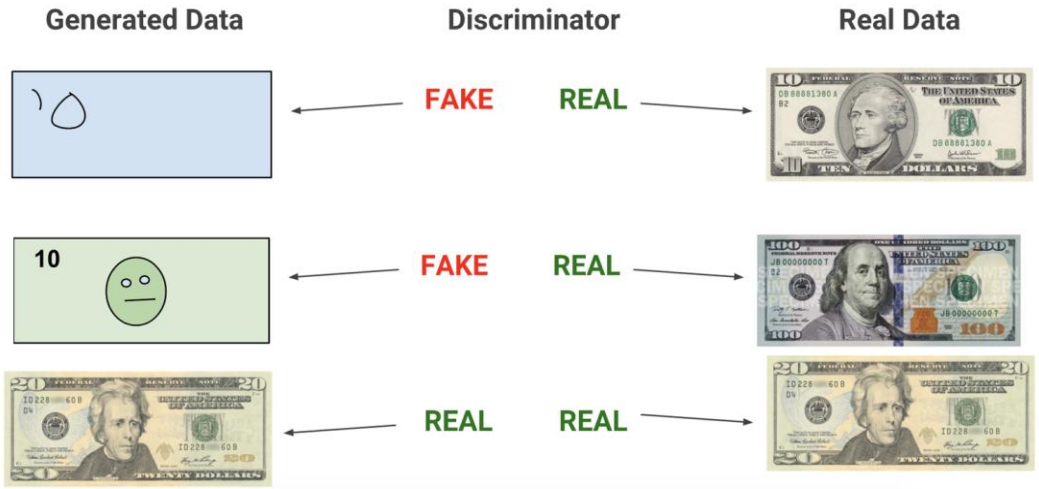
Generative Adversarial Networks: Toy example

VAE

GAN

Diffusion Models

- At the beginning, the generator is a very weak opponent



https://developers.google.com/machine-learning/gan

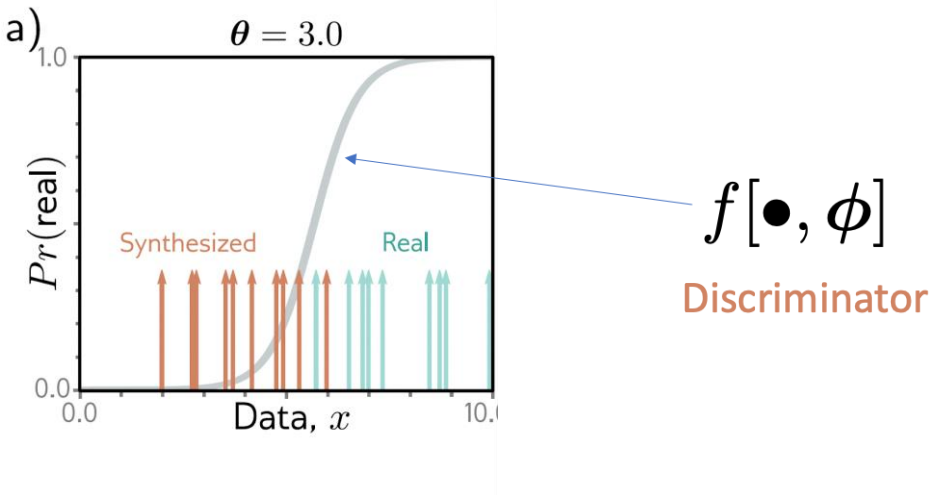
Generative Adversarial Networks: Toy example

$$x_j^* = \mathrm{g}[z_j, \theta] = z_j + \theta$$

VAE

GAN

Diffusion Models



$f[\bullet, \phi]$

Discriminator

Understanding Deep Learning, Simon J.D. Prince (2023)

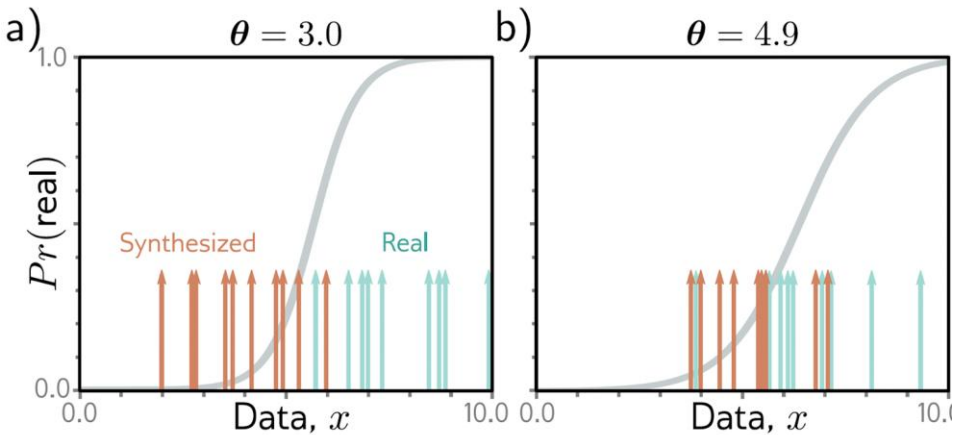
Generative Adversarial Networks: Toy example

$$x_j^* = \mathrm{g}[z_j, \theta] = z_j + \theta$$

VAE

GAN

Diffusion Models



Understanding Deep Learning, Simon J.D. Prince (2023)

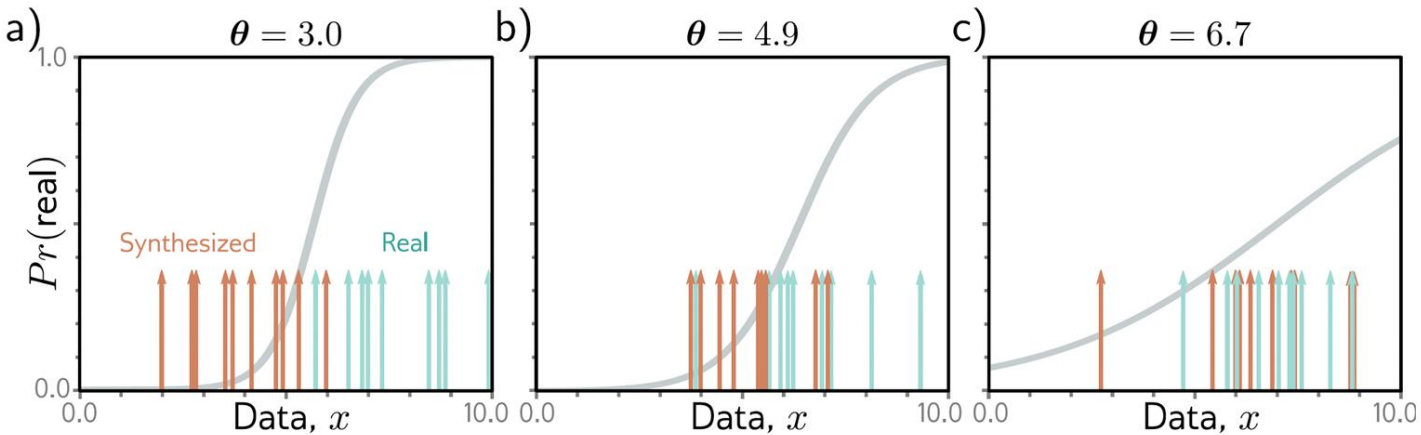
Generative Adversarial Networks: Toy example

$$x_j^* = \mathrm{g}[z_j, \theta] = z_j + \theta$$

VAE

GAN

Diffusion Models



Understanding Deep Learning, Simon J.D. Prince (2023)

GAN cost function

VAE

GAN

Diffusion Models

Discriminator uses standard cross entropy loss:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_i -(1-y_i) \log\Big[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\Big] - y_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\Big] \right]$$

Understanding Deep Learning, Simon J.D. Prince (2023)

GAN cost function

VAE

GAN

Diffusion Models

Discriminator uses standard cross entropy loss:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_i -(1-y_i) \log\Big[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\Big] - y_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\Big] \right]$$

Discriminator: generated samples, y = 0, real examples, y = 1:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_j -\log\Big[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}^*_j, \boldsymbol{\phi}]]\Big] - \sum_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\Big] \right]$$

Understanding Deep Learning, Simon J.D. Prince (2023)

GAN cost function

VAE

GAN

Diffusion Models

Discriminator uses standard cross entropy loss:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_i -(1-y_i)\log\Big[1-\operatorname{sig}[\mathrm{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big] - y_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big] \right]$$

Discriminator: generated samples, y = 0, real examples, y = 1:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_j -\log\Big[1-\operatorname{sig}[\mathrm{f}[\mathbf{x}^*_j,\boldsymbol{\phi}]]\Big] - \sum_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big] \right]$$

Generator loss: make generated samples more likely under discriminator (i.e. make discriminator loss larger)

$$\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_j -\log\Big[1-\operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j,\boldsymbol{\theta}],\boldsymbol{\phi}]]\Big] - \sum_i \log\Big[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i,\boldsymbol{\phi}]]\Big] \right] \right]$$

Understanding Deep Learning, Simon J.D. Prince (2023)

GAN cost function

VAE

GAN

Diffusion Models

$$\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right] \right]$$

Can divide into two parts:

$$L[\boldsymbol{\phi}] = \sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right]$$

$$L[\boldsymbol{\theta}] = \sum_j \log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{g}[\mathbf{z}_j, \boldsymbol{\theta}], \boldsymbol{\phi}]]\right]$$

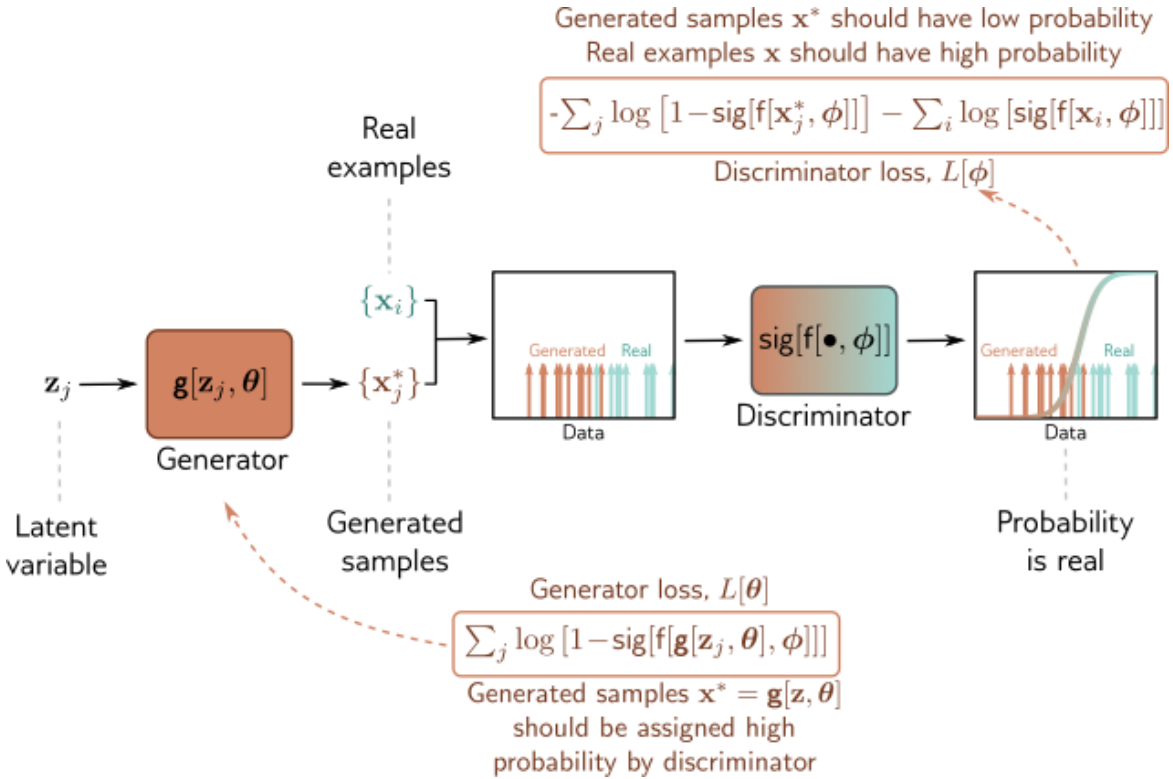
https://developers.google.com/machine-learning/gan

GAN loss function

VAE

GAN

Diffusion Models



Understanding Deep Learning, Simon J.D. Prince (2023)

GAN - Training

VAE

GAN

Diffusion Models

Loop until convergence (very unstable state):

- Train only the discriminator for several epochs
- Train the generator for several epochs (D is freezed here)
  - G never sees training data, only feedback from loss!

Problems

- Train two different kinds of networks, with moving target
- GAN convergence is hard to identify.

Understanding Deep Learning, Simon J.D. Prince (2023)

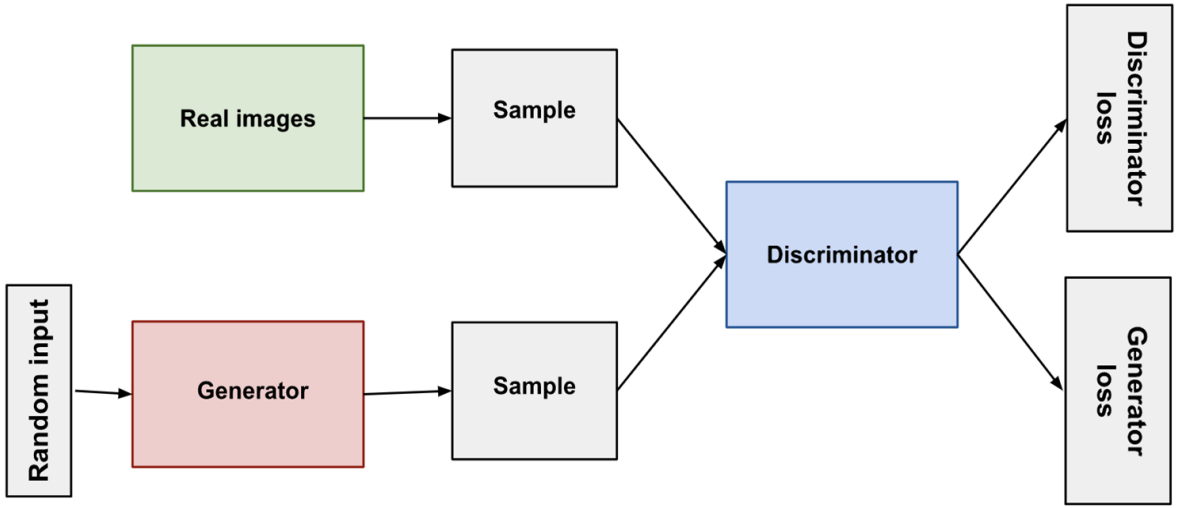
GAN - Architecture

VAE

GAN

Diffusion Models

Let the game begin and penalize both opponents when they are wrong.



https://developers.google.com/machine-learning/gan

GAN - Discriminator

VAE

GAN

Diffusion Models

- **Real data**: label 1 + **Fake data** from the generator: label 0

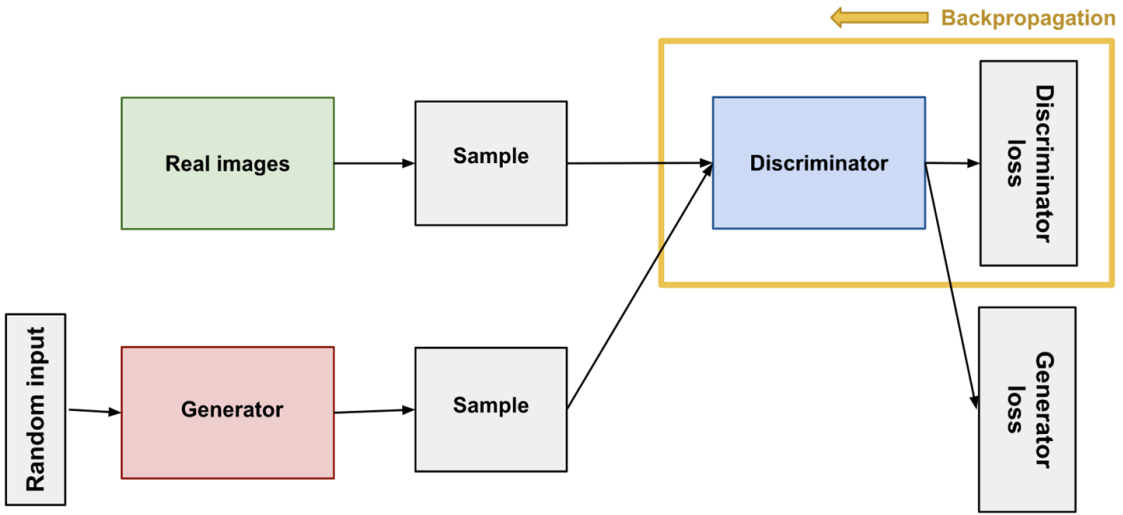


Figure 1: Backpropagation in discriminator training.

https://developers.google.com/machine-learning/gan

GAN - Generator

VAE

GAN

Diffusion Models

Train the Generator by keeping the **Discriminator Freezed**

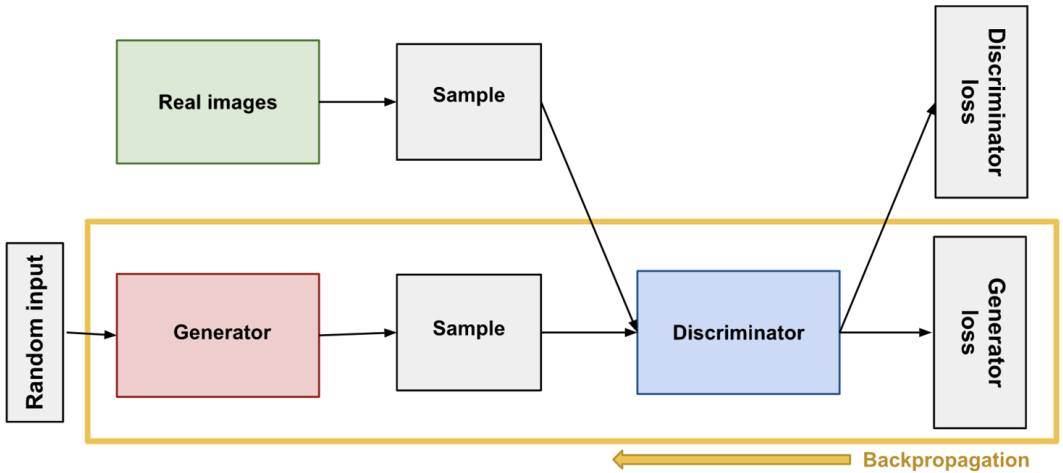


Figure 1: Backpropagation in generator training.

https://developers.google.com/machine-learning/gan

GAN - Common Problems

VAE

GAN

Diffusion Models

Vanishing Gradients

- When the discriminator gets too good, generator training can fail due to vanishing gradient
- An optimal discriminator doesn't provide enough information for the generator to make progress.

Mode Collapse

- Ideally, we want GAN to produce a wide variety of outputs.
- If a generator produces an especially plausible output, the generator may learn to produce ***only*** that output.

Understanding Deep Learning, Simon J.D. Prince (2023)

Wasserstein GAN - loss function

VAE

GAN

Diffusion Models

- Original function:

$$\hat{\boldsymbol{\phi}} = \underset{\boldsymbol{\phi}}{\operatorname{argmin}} \left[\sum_j -\log\left[1 - \operatorname{sig}[\mathrm{f}[\mathbf{x}_j^*, \boldsymbol{\phi}]]\right] - \sum_i \log\left[\operatorname{sig}[\mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]]\right] \right]$$

- Wasserstein GAN:

$$L[\boldsymbol{\phi}] = \sum_j \mathrm{f}[\mathbf{x}_j^*, \boldsymbol{\phi}] - \sum_i \mathrm{f}[\mathbf{x}_i, \boldsymbol{\phi}]$$

subject to:

$$\left| \frac{\partial \mathrm{f}[\mathbf{x}, \boldsymbol{\phi}]}{\partial \mathbf{x}} \right| < 1$$

Understanding Deep Learning, Simon J.D. Prince (2023)

Conditional GANs

VAE

GAN

Diffusion Models

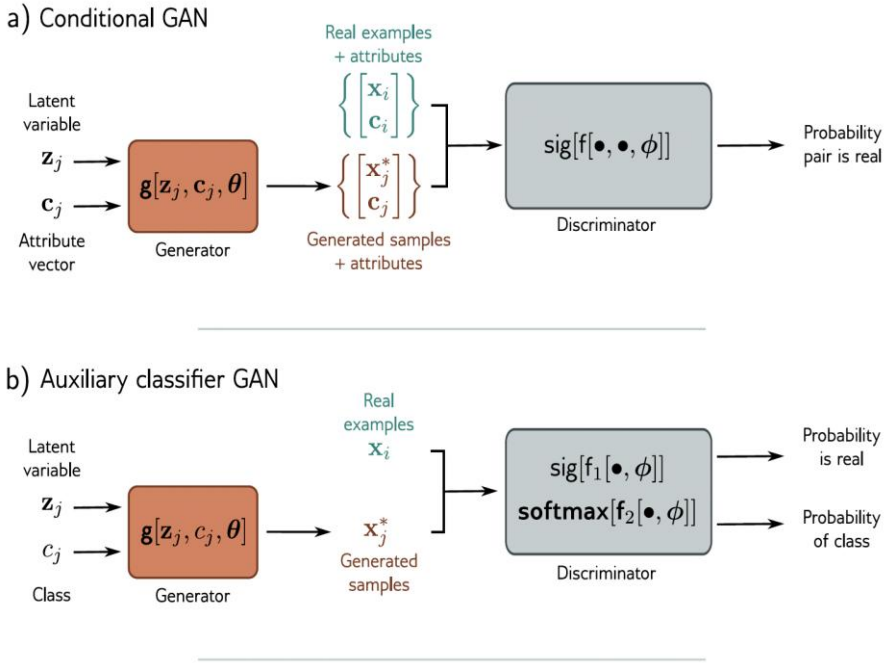
a) The generator(g) receives an attribute vector c; the discriminator receives either a real example or a generated sample, and it also receives the attribute vector;

=> this encourages the samples both to be **realistic** and **compatible with the attribute.**

b) The generator of the auxiliary classifier GAN (ACGAN) takes a discrete attribute variable. The discriminator must both

(i) determine if its input is real or synthetic and

(ii) identify the class correctly.



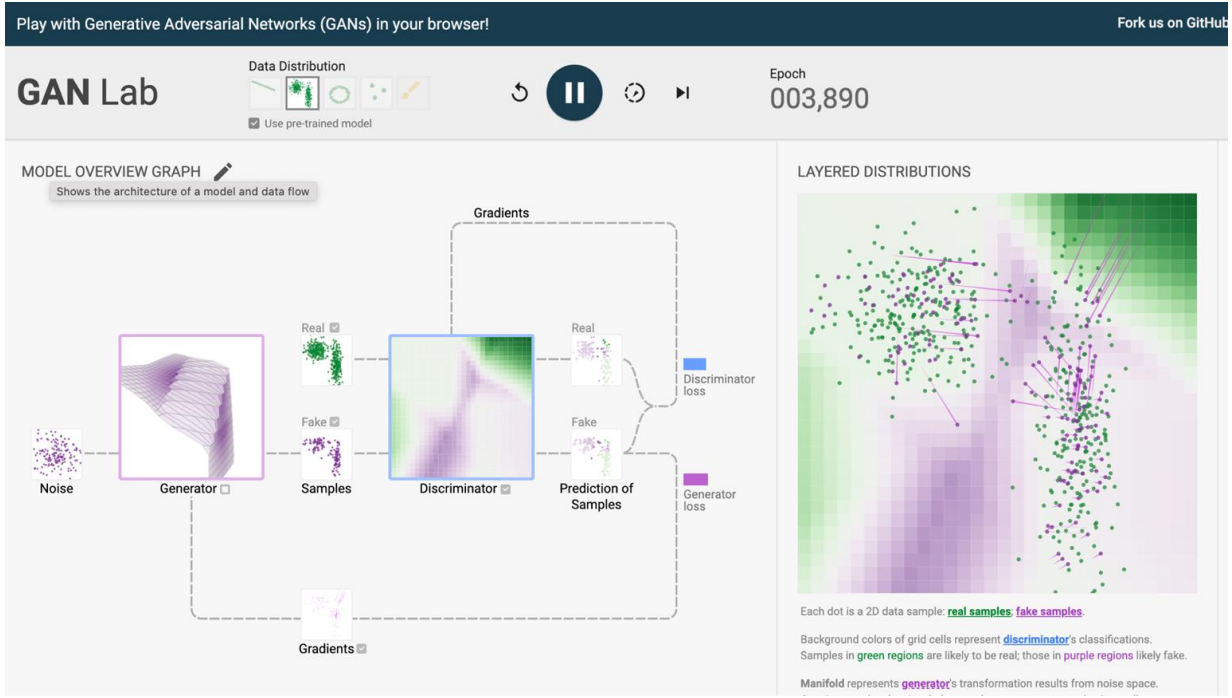
Understanding Deep Learning, Simon J.D. Prince (2023)

GAN - Training Toy Examples

VAE

GAN

Diffusion Models



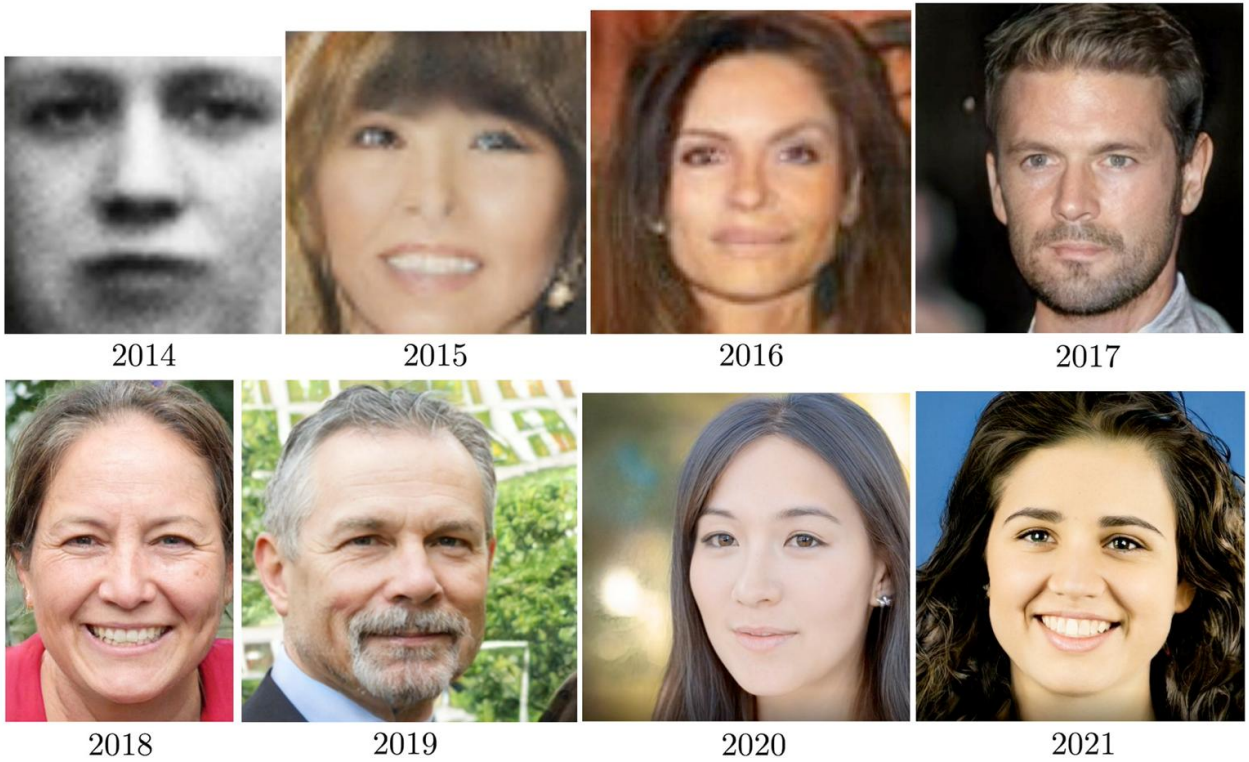
https://poloclub.github.io/ganlab/

GAN - Evolution Examples

VAE

GAN

Diffusion Models



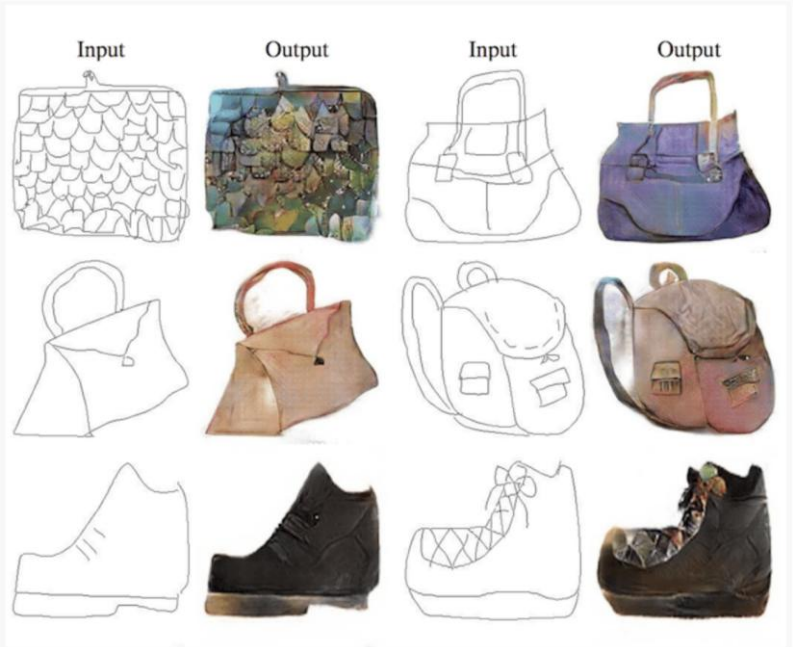
Generative Language Models and Automated Influence Operations: Emerging Threats and Potential Mitigations January 2023

GAN - Use Cases

VAE

GAN

Diffusion Models



Example of Sketches to Color Photographs With pix2pix.Taken from Image-to-Image Translation with Conditional Adversarial Networks, 2016.

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

GAN - Use Cases

VAE

GAN

Diffusion Models

Example of Semantic Image and GAN-Generated Cityscape Photograph.Taken from High-Resolution Image Synthesis and Semantic Manipulation with Conditional GANs, 2017.

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

GAN - Use Cases

VAE

GAN

Diffusion Models

Example of GAN-Generated Photographs of Human PosesTaken from Pose Guided Person Image Generation, 2017.

https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/

GAN - Use Cases

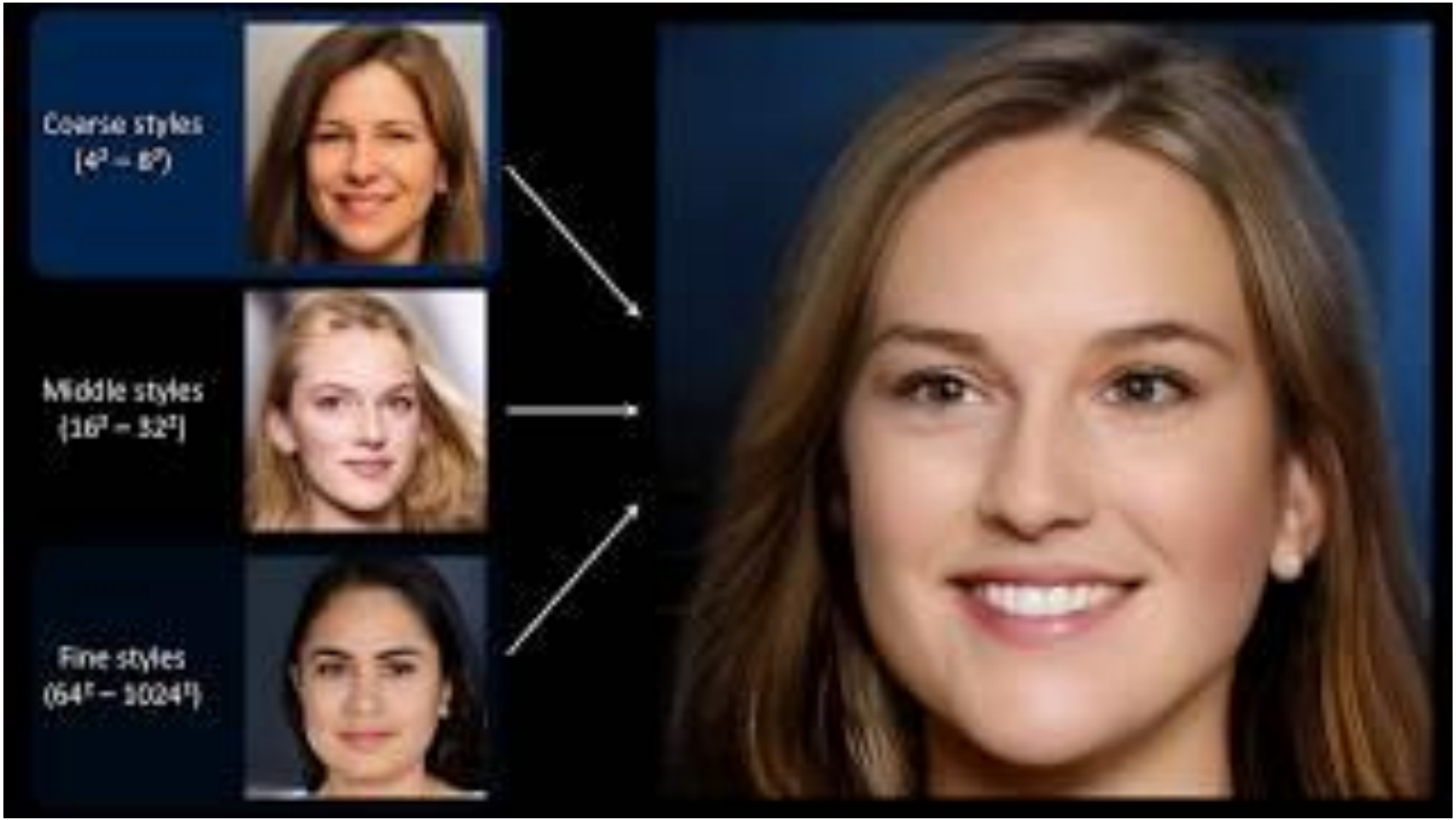
VAE

GAN

Diffusion Models

Generating different poses of a person.

[Human Synthesis and Scene Compositing]



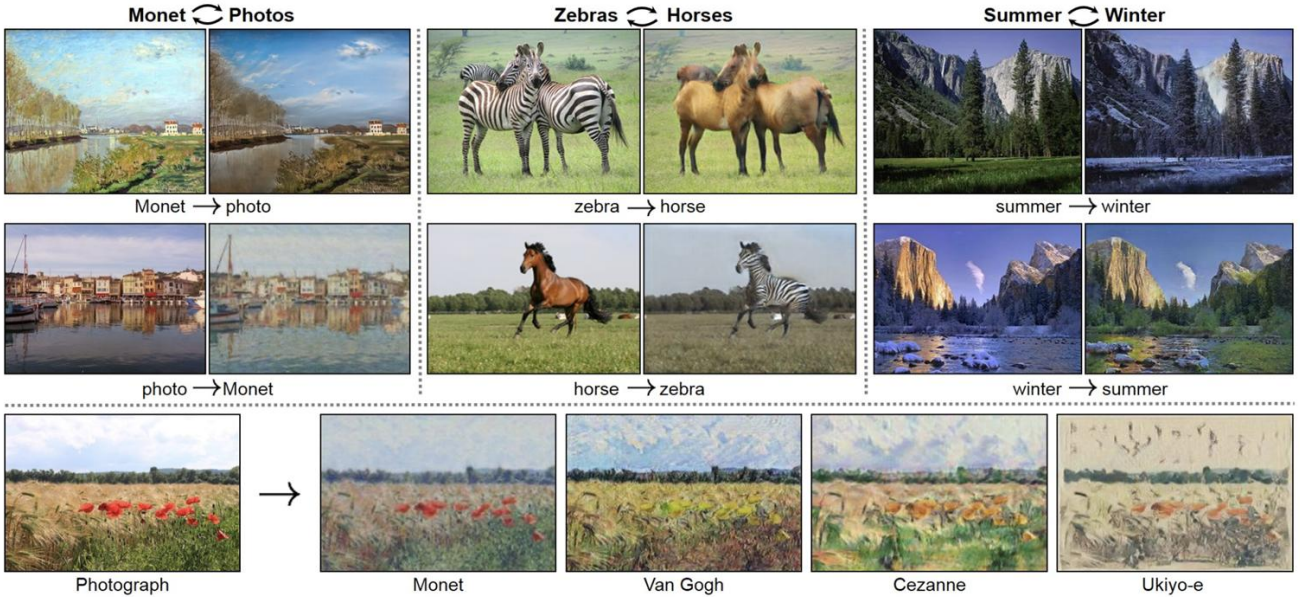
[A Style-Based Generator Architecture for Generative Adversarial Networks (2018)]

CycleGAN

VAE

GAN

Diffusion Models



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

CycleGAN

VAE

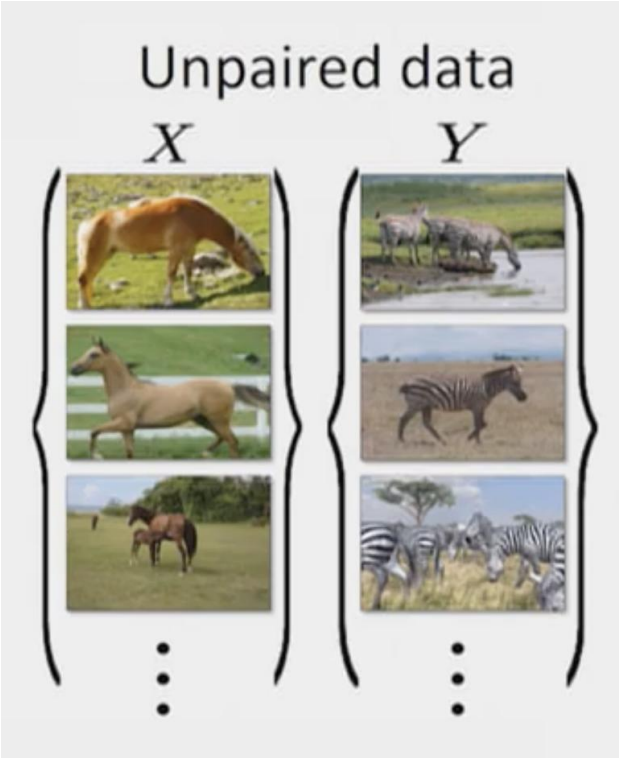
GAN

Diffusion Models

Unpaired Image-to-Image Translation

using

Cycle-Consistent Adversarial Networks



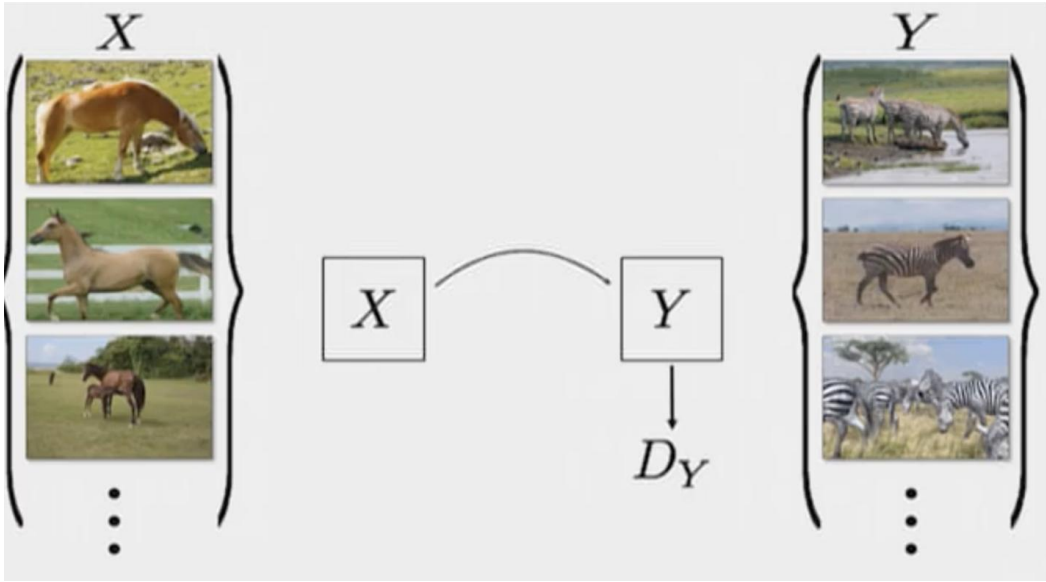
CycleGAN - Intuition

VAE

GAN

Diffusion Models

Q: What will the Generator learn?



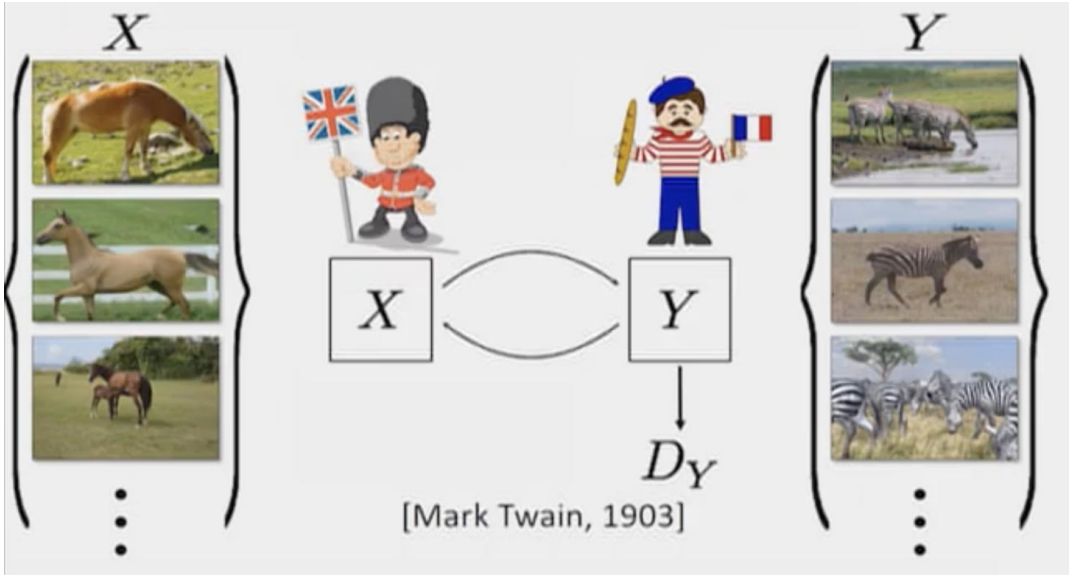
CycleGAN - Intuition

VAE

GAN

Diffusion Models

Cycle Consistency Constraint

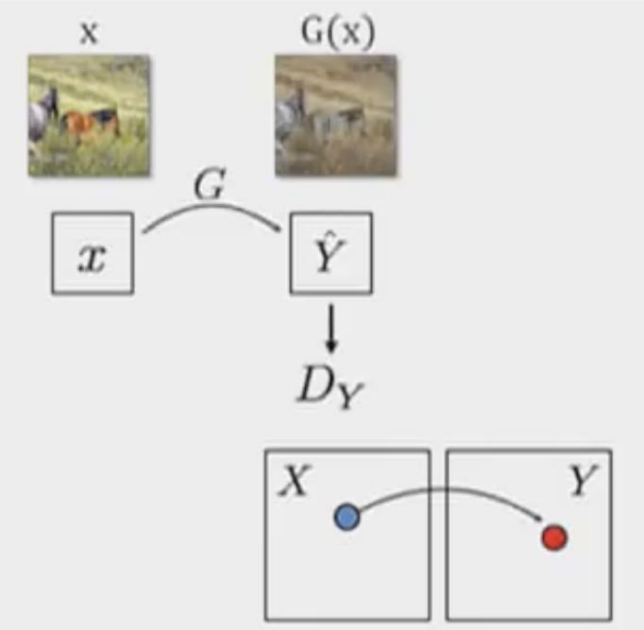


CycleGAN - Architecture

VAE

GAN

Diffusion Models



Go to Y space:

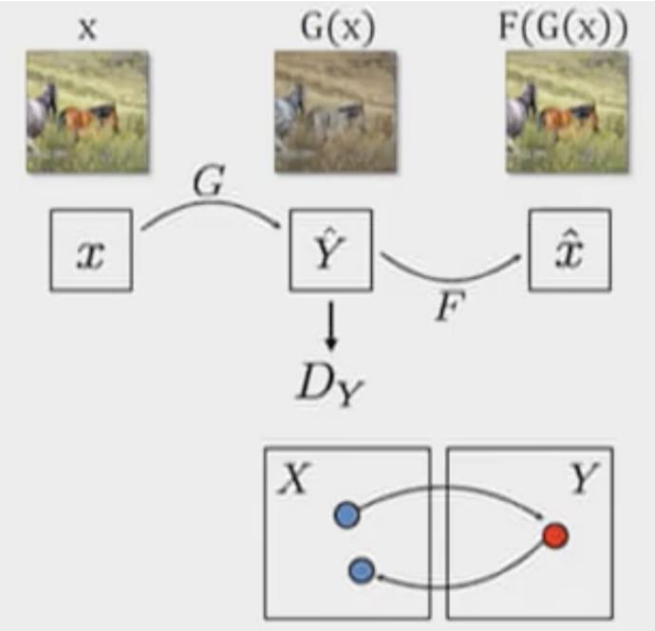
- G: X -> Y
- D: Y -> [0, 1]

CycleGAN - Architecture

VAE

GAN

Diffusion Models



Go to Y space:

- G: X -> Y
- D: Y -> [0, 1]

Go back to X space:

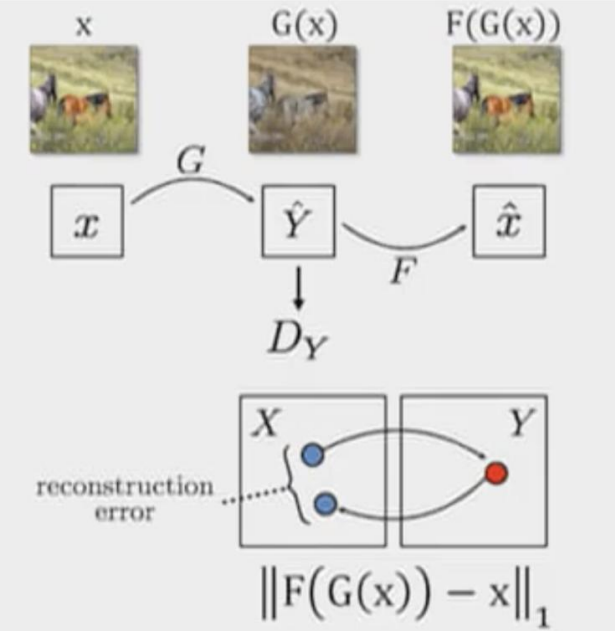
- F : Y -> X

CycleGAN - Architecture

VAE

GAN

Diffusion Models



Go to Y space:

- G: X -> Y
- D: Y -> [0, 1]

Go back to X space:

- F : Y -> X

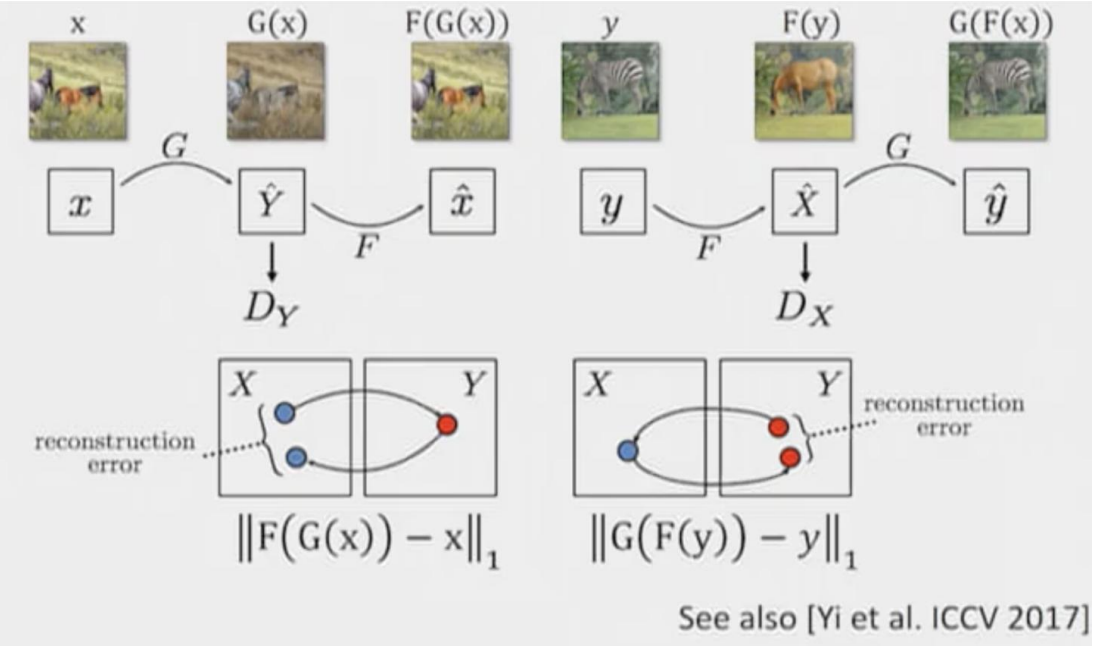
Goal: new X is close to initial X

CycleGAN - Architecture

VAE

GAN

Diffusion Models



CycleGAN

VAE

GAN

Diffusion Models

CycleGAN - Day to Night

VAE

GAN

Diffusion Models

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

Diffusion Models

VAE

GAN

Diffusion Models

- **Diffusion Models: Theory**
  - Diffusion Models Overview
  - Forward Diffusion Process
  - Reverse Denoising Process
  - Training Diffusion Models: Simple Loss
  - Comparison to GANs
  - Latent Diffusion Models

- **Diffusion Models: Examples**
  - Stable Diffusion
  - Imagen
  - Dalle-2
  - Dalle-3
  - Sora

Diffusion Models Overview

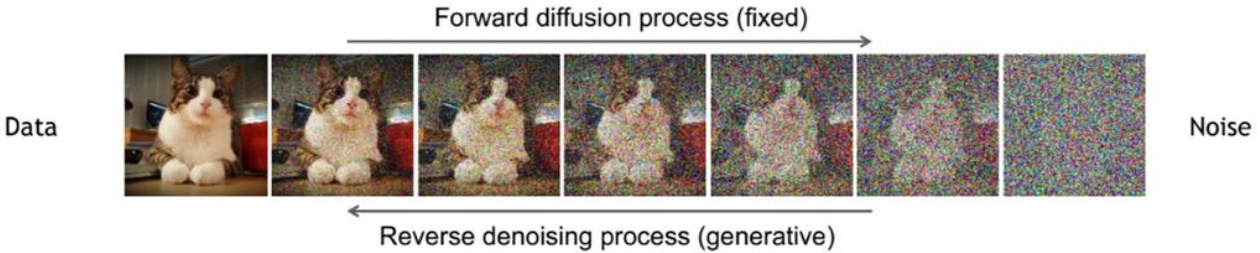
VAE

GAN

Diffusion Models

Denoising Diffusion Models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Source Slide:**CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

Forward Diffusion Process

VAE

GAN

Diffusion

Models

- The original image (x_0) is slowly corrupted iteratively by adding Gaussian noise.
- This process is done for some T time steps
- Image at timestep t is created by: $x_{t-1} + \epsilon_{t-1} x \rightarrow x_t$
- No model is involved at this stage.
- At the end, x_T is pure noise.

Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

https://learnopencv.com/image-generation-using-diffusion-models/

Forward Diffusion Process

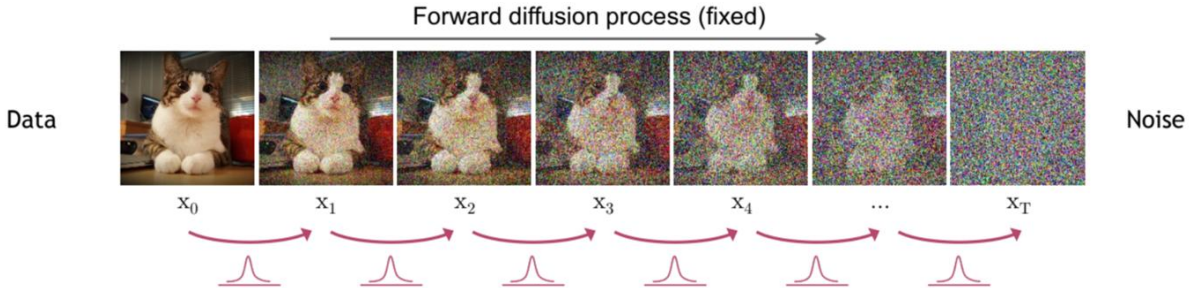
VAE

GAN

Diffusion

Models

- The original image (x_0) is slowly corrupted iteratively by adding Gaussian noise.
- This process is done for some T time steps
- Image at timestep t is created by: $x_{t-1} + \epsilon_{t-1} x \rightarrow x_t$
- No model is involved at this stage.
- At the end, x_T is pure noise.



Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

https://learnopencv.com/image-generation-using-diffusion-models/

Forward Diffusion Process

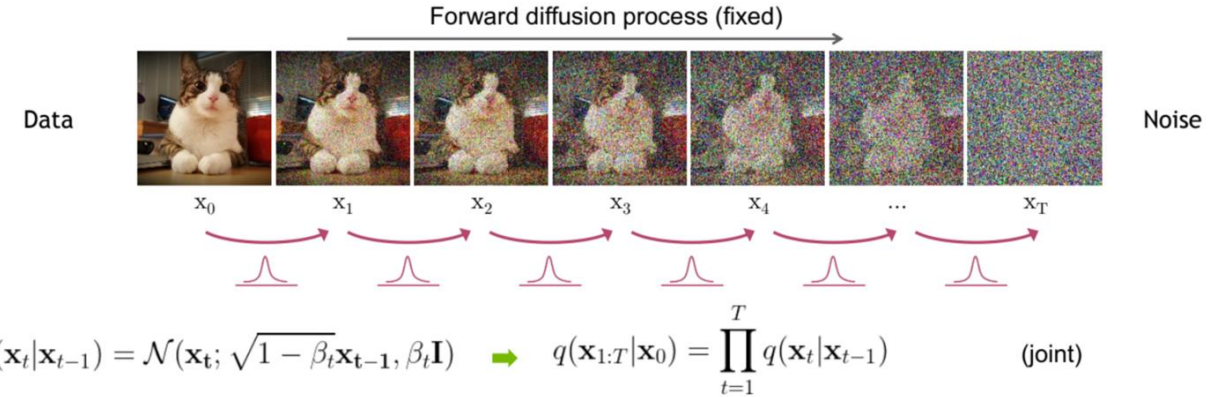
VAE

GAN

Diffusion

Models

- The original image (x_0) is slowly corrupted iteratively by adding Gaussian noise.
- This process is done for some T time steps,
- Image at timestep t is created by: $x_{t-1} + \epsilon_{t-1} x \rightarrow x_t$
- No model is involved at this stage.
- At the end, x_T is pure noise.



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \quad \Rightarrow \quad q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

https://learnopencv.com/image-generation-using-diffusion-models/

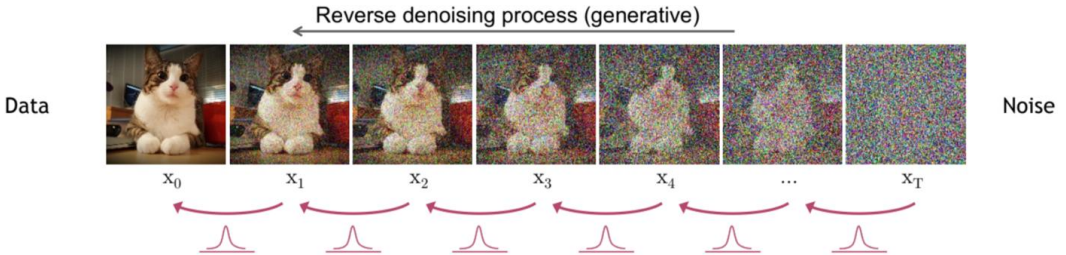
Reverse Denoising Process

VAE

GAN

Diffusion Models

- We undo the forward process by removing the noise added in the forward process using a **neural network model.**
- Given a timestep t and the noisy image x_t predict the noise added to the image at step t-1.



Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

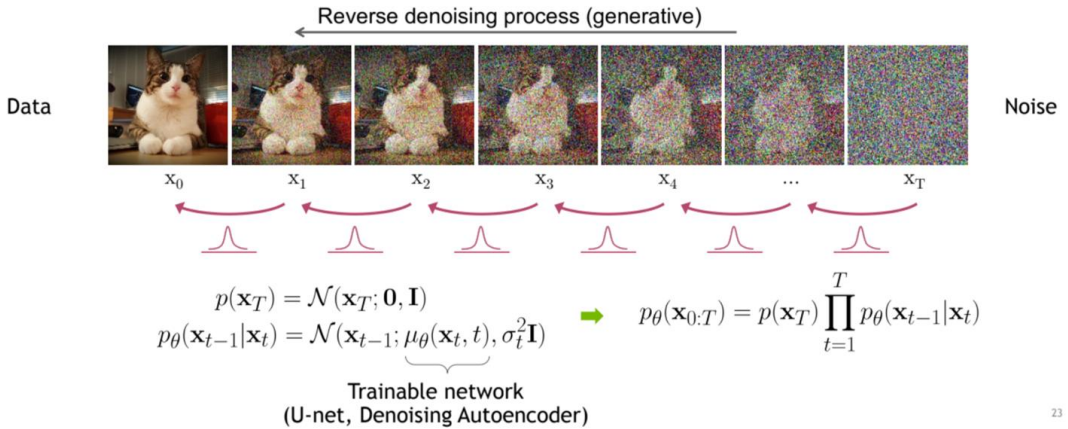
Reverse Denoising Process

VAE

GAN

Diffusion Models

- We undo the forward process by removing the noise added in the forward process using a **neural network model.**
- Given a timestep t and the noisy image x_t predict the noise added to the image at step t-1.



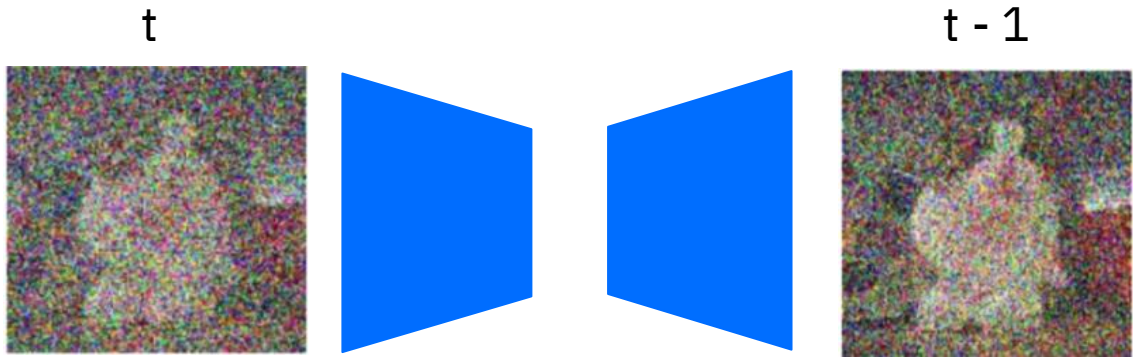
Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

Training Diffusion Models: Simple Loss

VAE

GAN

Diffusion Models



- In practice, the final loss function we use to train Denoising Diffusion Probabilistic Models, is just a **Mean Squared Error** between the **noise added** in the forward process and the **noise predicted** by the model. $$L_{\text{simple}} = E_{t,x_0,\epsilon}\left[||\epsilon - \epsilon_\theta(x_t, t)||^2\right]$$

Partial Source Slide: **CVPR 2022 Tutorial:** Denoising Diffusion-based Generative Modeling:Foundations and Applications

Denoising Diffusion Probabilistic Models

Comparison to VAEs

VAE

GAN

Diffusion Models

- Both have an **inference model** (forward process), transforming data to latent representation. For Diffusion Models this process is **fixed**.
- Both have a **generative model** (reverse process) that samples some **latent variable** and transforms it with a neural network. In Diffusion Models this is done **iteratively**.
- In Diffusion Models latent dimensionality must match the data. In VAEs we can **reduce dimensionality.**

https://angusturner.github.io/generative_models/2021/06/29/diffusion-probabilistic-models-I.html

Comparison to GANs

VAE

GAN

Diffusion Models

- Due to the iterative nature of the diffusion process, **the training and generation process is generally more stable** than GANs.
- In Diffusion Models latent dimensionality must match the data. In GANs, the latent tensor can be of **different sizes**.
- Diffusion models only require **one model**.

https://learnopencv.com/image-generation-using-diffusion-models/

Diffusion Models

VAE

GAN

Diffusion Models

- **Diffusion Models: Theory**
  - Diffusion Models Overview
  - Forward Diffusion Process
  - Reverse Denoising Process
  - Training Diffusion Models: Simple Loss
  - Comparison to GANs
  - Latent Diffusion Models

- **Diffusion Models: Examples**
  - Stable Diffusion
  - Imagen
  - Dalle-2
  - Dalle-3
  - Sora

Latent Diffusion Models

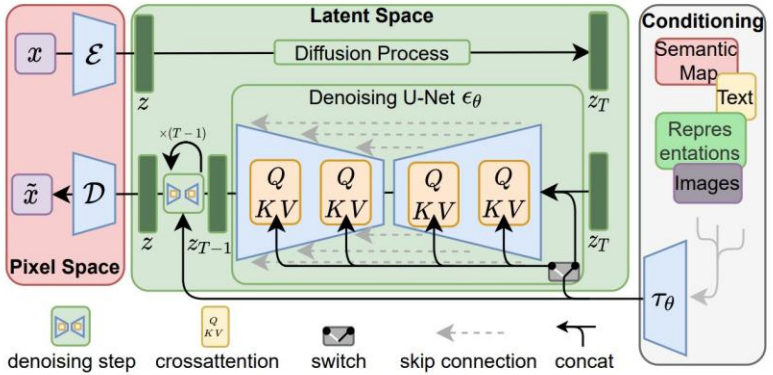
VAE

GAN

Diffusion Models

Key features:

- Compresses input images using a pretrained *autoencoder* to a lower-dimensional ***latent space***
- Trains diffusion in latent instead of pixel space
- Adds noise gradually to latent representations during training; the model learns to reverse this noising process step by step
- Conditions the model on external information (e.g., text, class labels, layouts) during denoising, enabling controlled and guided generation.
- Generates by sampling noise in latent space, denoising it using the learned model and the conditioning input.
- Decodes the final latent representation back to pixel space



High-Resolution Image Synthesis with Latent Diffusion Models

Diffusion Models: Stable Diffusion

VAE

GAN

Diffusion Models

A cat wearing a spiderman suit.

Students during deep learning course at the University of Bucharest

! Check the newer versions: SDXL1, SD3, etc

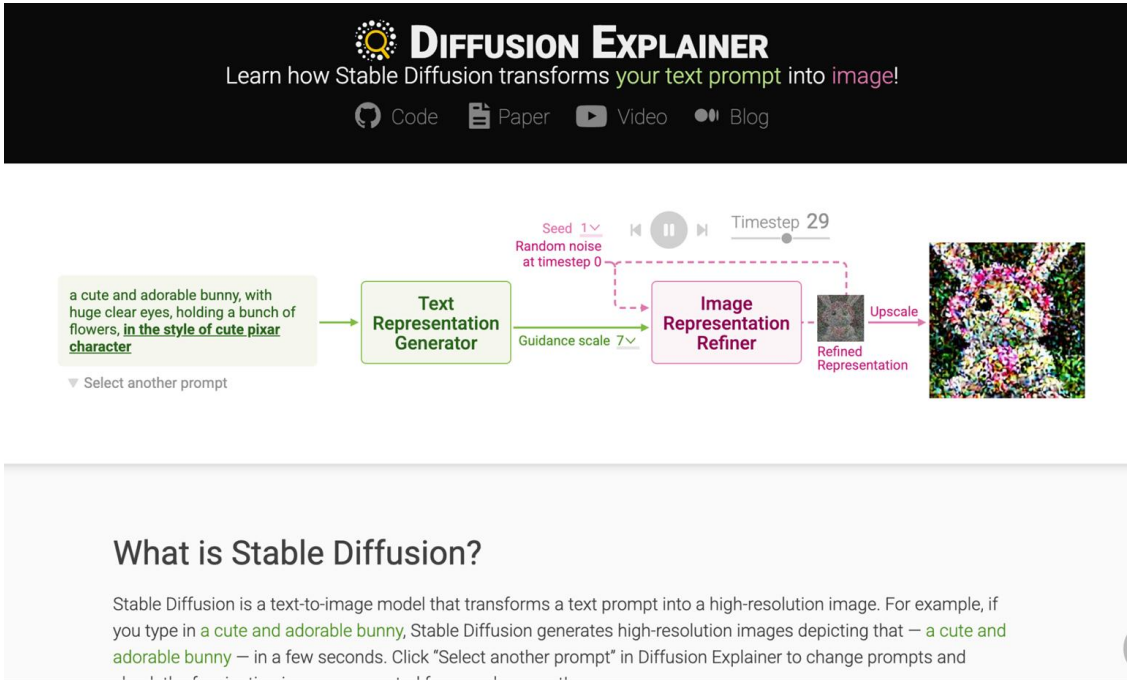
https://stablediffusionweb.com/

Diffusion Explainer

VAE

GAN

Diffusion Models



https://poloclub.github.io/diffusion-explainer/

Diffusion Models: Imagen

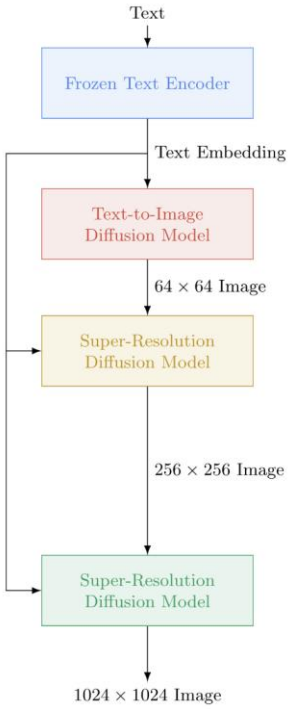
VAE

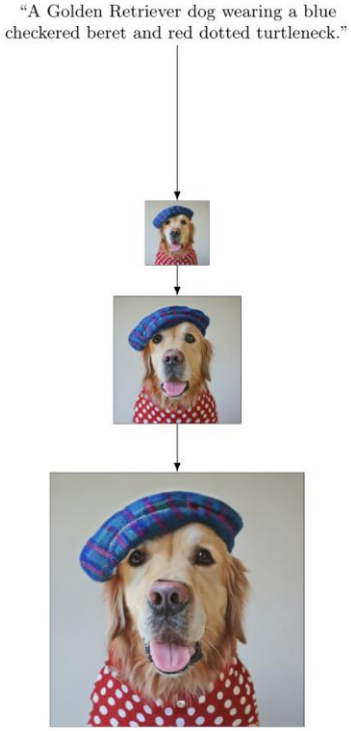
GAN

Diffusion Models

Key features:

- Text-to-image diffusion model: Generates high-fidelity images from natural language prompts using diffusion in pixel space.
- Comprises:
  - a frozen T5-XXL [52] encoder to map input text into a sequence of embeddings
  - and a 64×64 image diffusion model, followed by two super-resolution diffusion models for generating 256×256 and 1024×1024 images.
- .All diffusion models are conditioned on the text embedding sequence





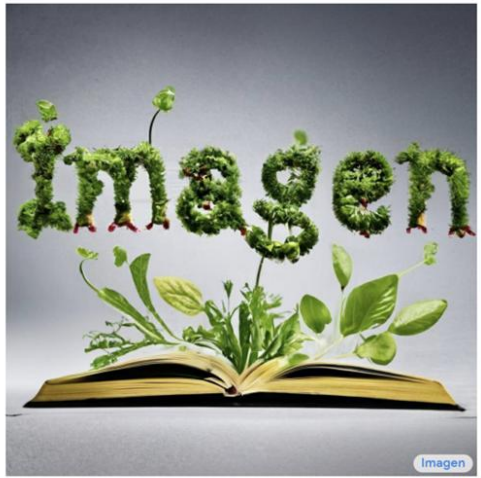
[Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding]

Diffusion Models: Imagen

VAE

GAN

Diffusion Models



Sprouts in the shape of text 'Imagen' coming out of a fairytale book.

A photo of a Shiba Inu dog with a backpack riding a bike. It is wearing sunglasses and a beach hat.

A high contrast portrait of a very happy fuzzy panda dressed as a chef in a high end kitchen making dough. There is a painting of flowers on the wall behind him.

[Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding]

Diffusion Models: Imagen3

VAE

GAN

Diffusion Models

Key features:

- **Latent diffusion model:** Faster and more efficient than pixel-space diffusion (used in original Imagen).
- High-fidelity generation: Produces detailed, 1024×1024 images with rich textures and lighting.
- Strong prompt alignment: Captures complex scenes with improved understanding of spatial and semantic relationships.
- Fast variant available: *Imagen 3 Fast* generates 4 images in under 4 seconds with quality intact.

Imagen3

Diffusion Models: Dalle-2

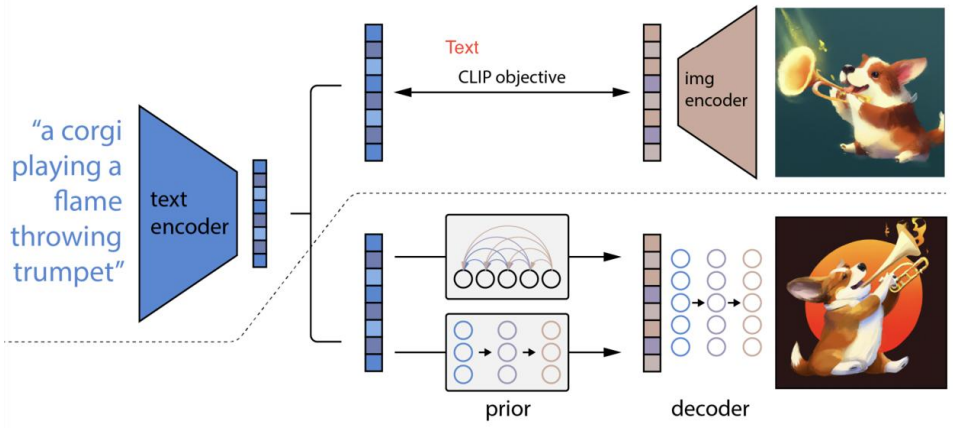
VAE

GAN

Diffusion Models

Key features

- Use CLIP for learning a joint representation space for text and images.
- CLIP text embedding is first fed to an autoregressive or diffusion prior to produce an image embedding,
- This embedding is used to condition a diffusion decoder which
- produces a final image.
- the CLIP model is frozen during training of the prior and decoder.



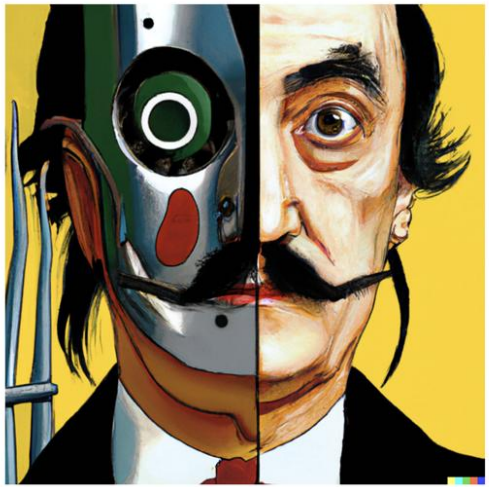
[Hierarchical Text-Conditional Image Generation with CLIP Latents](https://openai.com/product/dall-e-2)
https://openai.com/product/dall-e-2

Diffusion Models: Dalle-2

VAE

GAN

Diffusion Models



vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

[Hierarchical Text-Conditional Image Generation with CLIP Latents](https://openai.com/product/dall-e-2)
https://openai.com/product/dall-e-2

Diffusion Models: Dalle-3

Images from: https://openai.com/dall-e-3

Video Diffusion Models: Sora

Key features:

- Starts with random noise and gradually refines it into a coherent video using a denoising process guided by the prompt.
- It predicts *both spatial and temporal patterns* — not just "what" is in the video, but also *how* it moves over time.
- It divides the video into patches across both space and time and models dependencies between them.
- Can be guided by text prompts, but also potentially images or video snippets, making it *multimodal.*

Videos from: https://openai.com/sora

Thank you!

Next: Project presentations.

References

- https://sites.google.com/view/berkeley-cs294-158-sp19/home
- https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368
- https://m2dsupsdlclass.github.io/lectures-labs/slides/10_unsupervised_generative_models
- https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html
- https://courses.cs.ut.ee/LTAT.02.001/2017_fall/uploads/Main/Lecture%2011%20-%20NN%202017-18.pdf
- http://www.cs.cmu.edu/~bhiksha/courses/deeplearning/Spring.2018/www/slides/lec16.vae.pdf
- https://developers.google.com/machine-learning/gan
- https://storage.googleapis.com/deepmind-media/UCLxDeepMind_2020/L10%20-%20UCLxDeepMind%20DL2020.pdf

References

- Auto-Encoding Variational Bayes, Diederik P. Kingma , Max Welling (2013)
- Generative Adversarial Nets, Ian J. Goodfellow, Jean Pouget-Abadie,, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair , Aaron Courville, Yoshua Bengio (2014)
- RePaint: Inpainting using Denoising Diffusion Probabilistic Models, Andreas Lugmayr Martin Danelljan Andres Romero Fisher Yu Radu Timofte Luc Van Gool (2022)
- Deep Unsupervised Learning using Nonequilibrium Thermodynamics Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, Stanford University, Surya Ganguli (2015)
- Denoising Diffusion Probabilistic Models, Jonathan Ho, Ajay Jain, Pieter Abbeel (2015)
- High-Resolution Image Synthesis with Latent Diffusion Models, Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Bjorn Ommer (2022)
- Understanding Deep Learning, Simon J.D. Prince (2023)