Deep Generative Models: Part I

Q. Bertrand[1, 3], M. Massias[1, 2], R. Emonet[3, 1]

[1]Inria Lyon

[2]ENS Lyon

[3]Laboratoire Hubert Curien - Saint-Étienne

April 11, 2025

A Few Words About Me

Academic

- 2018 - 2021 Ph. D. at Inria Saclay: Large-Scale Optimization for Neuroimaging
- 2021 - 2024 Postdoc at Mila and Université de Montréal
- 2024 - now Reasearcher at Inria Lyon, based in Laboratoire Hubert Curien

A Few Words About Me

Academic

- 2018 - 2021 Ph. D. at Inria Saclay: Large-Scale Optimization for Neuroimaging
- 2021 - 2024 Postdoc at Mila and Université de Montréal
- 2024 - now Reasearcher at Inria Lyon, based in Laboratoire Hubert Curien

Research interests:

- Numerical optimization
- Generative optimization
  - Retraining of generative models on their own data
- Multi-agent reinforcement learning
  - Emergence of cooperation in multi-agent settings

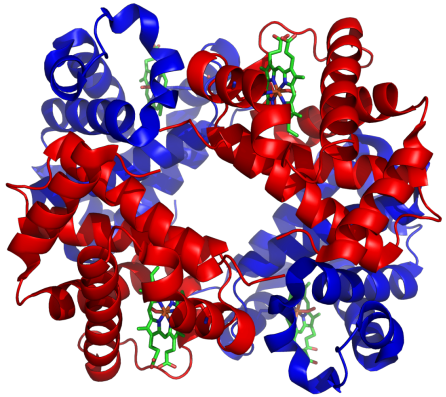
Outline

What are Generative Models?

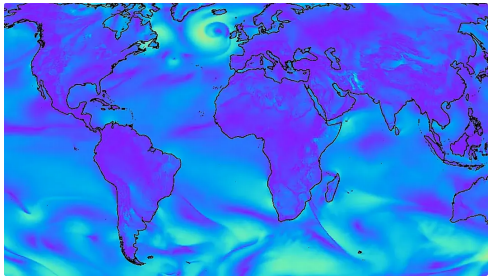
Generative Models are Powerful

Text-to-Image

Text-to-Text



Protein Design

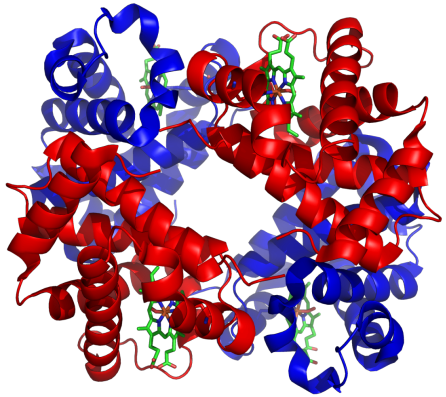


Weather Forecast

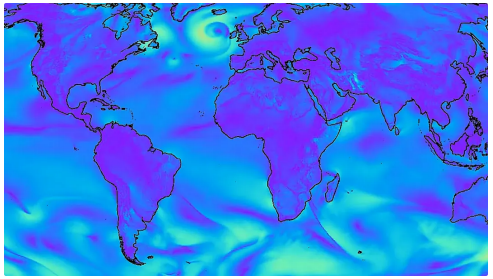
Generative Models are Powerful

Text-to-Image

Text-to-Text



Protein Design



Weather Forecast

Academic References:

- Stability AI (2023)
- Watson et al. (2023)
- Lam et al. (2023)
- Xu et al. (2025)

What are Generative Models 1/3

Data: $x^{(1)}, \ldots, x^{(n)}$

Goal: new synthetic samples $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(n)}$

Generative Models 101:

- **Setting**: Access to samples $\left(x^{(1)}, \ldots, x^{(n)}\right) \rightarrow$ unlabeled data

What are Generative Models 1/3

Data: $x^{(1)}, ..., x^{(n)}$

Goal: new synthetic samples $\tilde{x}^{(1)}, ..., \tilde{x}^{(n)}$

Generative Models 101:

- **Setting**: Access to samples $\left(x^{(1)}, ..., x^{(n)}\right) \rightarrow$ unlabeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p$

What are Generative Models 1/3

Data: $x^{(1)}, \ldots, x^{(n)}$

Goal: new synthetic samples $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(n)}$

Generative Models 101:

- **Setting**: Access to samples $\left(x^{(1)}, \ldots, x^{(n)}\right) \rightarrow$ unlabeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p$
  - *e.g.*, a set of images

What are Generative Models 1/3

Data: $x^{(1)}, ..., x^{(n)}$

Goal: new synthetic samples $\tilde{x}^{(1)}, ..., \tilde{x}^{(n)}$

Generative Models 101:

- **Setting**: Access to samples $\left(x^{(1)}, ..., x^{(n)}\right) \rightarrow$ unlabeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p$
  - *e.g.*, a set of images
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p$

What are Generative Models 1/3

Data: $x^{(1)}, \ldots, x^{(n)}$

Goal: new synthetic samples $\tilde{x}^{(1)}, \ldots, \tilde{x}^{(n)}$

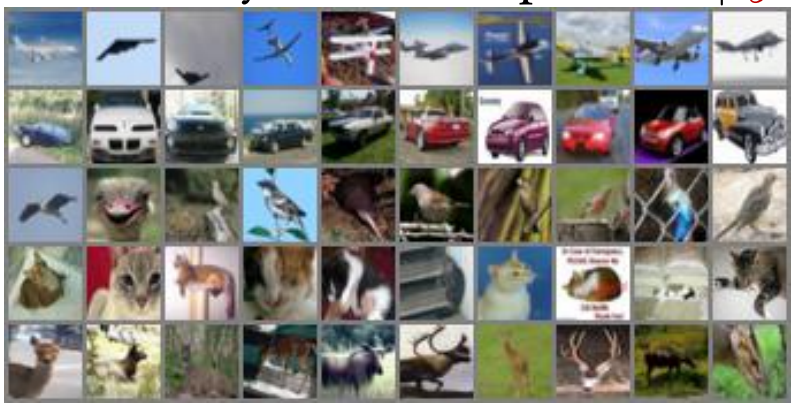
Generative Models 101:

- **Setting**: Access to samples $\left(x^{(1)}, \ldots, x^{(n)}\right) \rightarrow$ unlabeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p$
  - *e.g.*, a set of images
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p$
  - *e.g.*, generate new images

What are Generative Models 2/3

Data: $\left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(n)}, y^{(n)}\right)$

Goal: new synthetic samples $\tilde{x}^{(i)} \mid y^{(i)}$



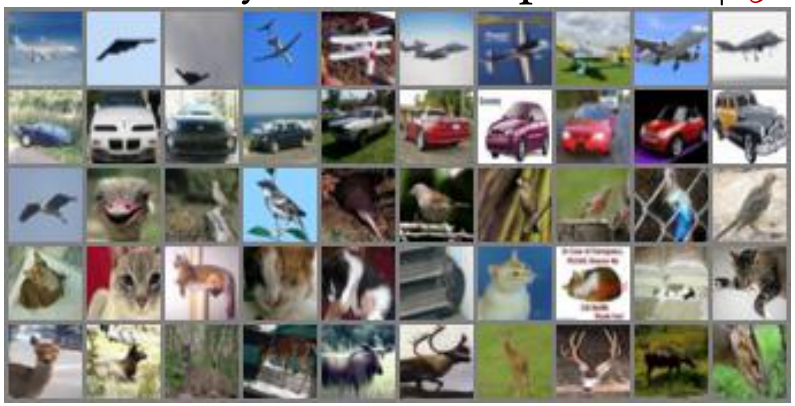
Generative Models 201: Class Conditional Generative Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), ..., \left(x^{(n)}, y^{(n)}\right)) \rightarrow$ labeled data

What are Generative Models 2/3

Data: $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$

Goal: new synthetic samples $\tilde{x}^{(i)} \mid y^{(i)}$



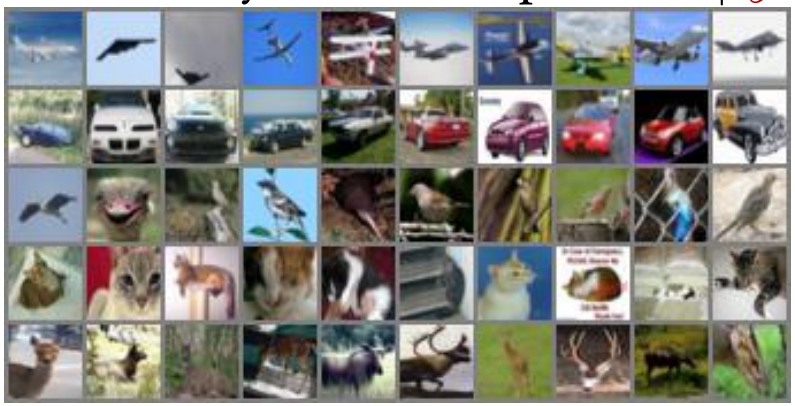
Generative Models 201: Class Conditional Generative Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)) \rightarrow$ labeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$

What are Generative Models 2/3

Data: $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$

Goal: new synthetic samples $\tilde{x}^{(i)} \mid y^{(i)}$



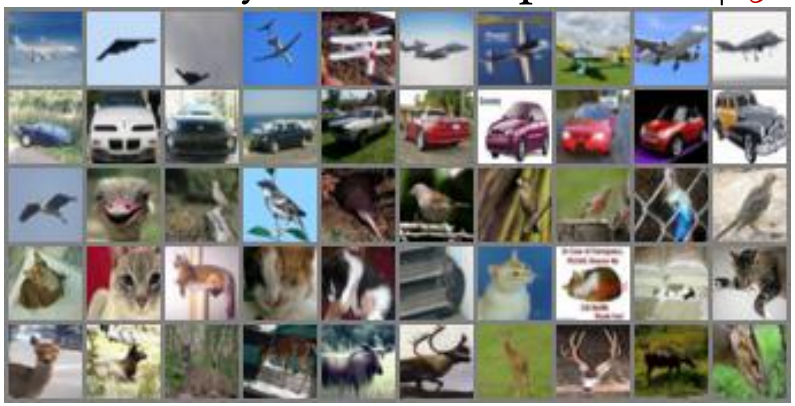
Generative Models 201: Class Conditional Generative Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)) \rightarrow$ labeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with class $y^{(i)}$

What are Generative Models 2/3

Data: $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$

Goal: new synthetic samples $\tilde{x}^{(i)} \mid y^{(i)}$



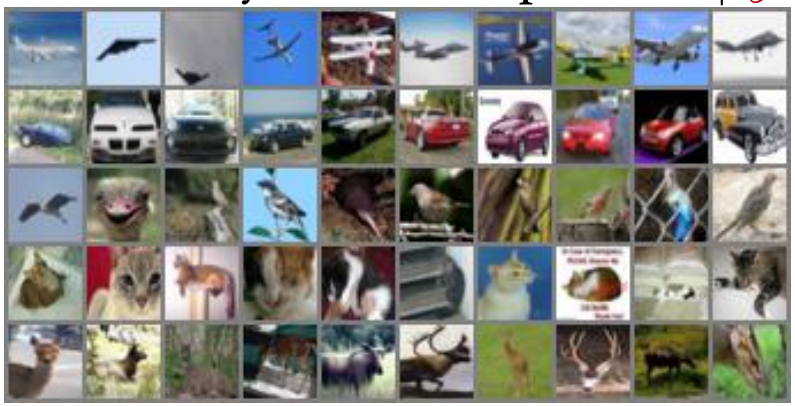
Generative Models 201: Class Conditional Generative Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)) \rightarrow$ labeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with class $y^{(i)}$
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$

What are Generative Models 2/3

Data: $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)$

Goal: new synthetic samples $\tilde{x}^{(i)} \mid y^{(i)}$



Generative Models 201: Class Conditional Generative Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right)) \rightarrow$ labeled data
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with class $y^{(i)}$
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p\left(\cdot \mid y^{(i)}\right)$
  - *e.g.*, generate specifically new images of planes

What are Generative Models 3/3

An avocado chair

A photo of Einstein

A house made of sushi

Generative Models 301: Text-to-Image Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \rightarrow$ pairs (image, caption)

What are Generative Models 3/3

An avocado chair

A photo of Einstein

A house made of sushi

Generative Models 301: Text-to-Image Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \rightarrow$ pairs (image, caption)
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$

What are Generative Models 3/3

An avocado chair

A photo of Einstein

A house made of sushi

Generative Models 301: Text-to-Image Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \rightarrow$ pairs (image, caption)
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with caption $y^{(i)}$

What are Generative Models 3/3

An avocado chair

A photo of Einstein

A house made of sushi

Generative Models 301: Text-to-Image Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \rightarrow$ pairs (image, caption)
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with caption $y^{(i)}$
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$

What are Generative Models 3/3

An avocado chair

A photo of Einstein

A house made of sushi

Generative Models 301: Text-to-Image Models

- **Setting**: Access to samples $\left(x^{(1)}, y^{(1)}\right), \ldots, \left(x^{(n)}, y^{(n)}\right) \rightarrow$ pairs (image, caption)
  - Draw from a probability distribution p, $x^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$
  - *e.g.*, a set of images $x^{(i)}$ with caption $y^{(i)}$
- **Goal**: Create new synthetic samples $\tilde{x}^{(i)} \sim p\left(\cdot \mid \text{caption } y^{(i)}\right)$
- **Trick**: Use a "meaningful" text/caption representation

A Few Comments

A Few Comments

- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence

A Few Comments

- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence
- **Evaluation?**

A Few Comments

- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence
- **Evaluation?**
  - Quality of the generated images?

A Few Comments

- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence
- **Evaluation?**
  - Quality of the generated images?
  - No cross-validation

A Few Comments

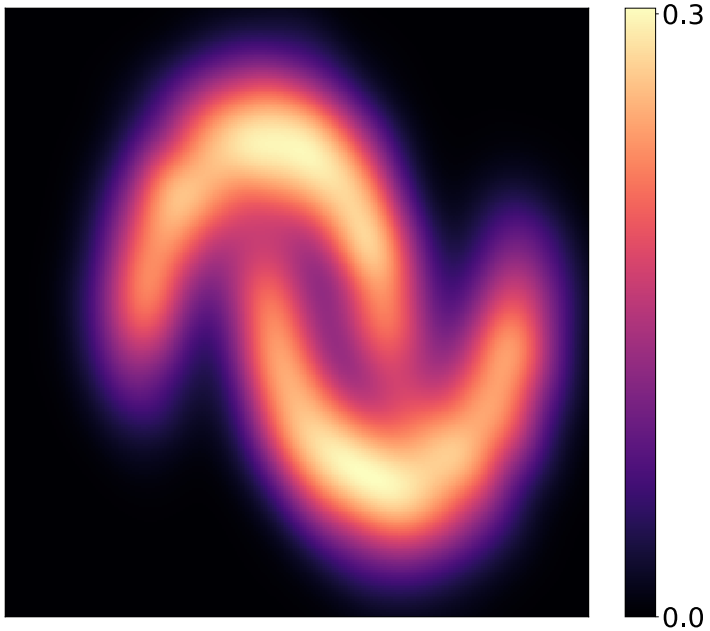
- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence
- **Evaluation?**
  - Quality of the generated images?
  - No cross-validation
  - **Very challenging** for text-to-image and text-to-text models

A Few Comments

- **Text-to-Text Models**
  - Conditional text generation w.r.t. the previous text sequence
- **Evaluation?**
  - Quality of the generated images?
  - No cross-validation
  - **Very challenging** for text-to-image and text-to-text models
  - Often use other models to assess the quality e.g., GPT

“Implicit” Generative Models: Problem Setting 1/4

Goal: estimate the unknown complex density p_{data}



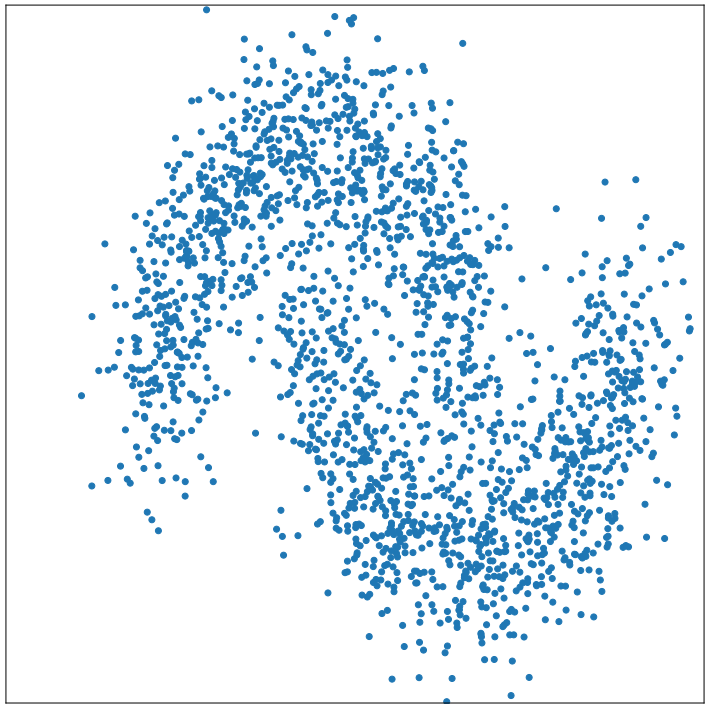
Unknown density p_{data}

“Implicit” Generative Models: Problem Setting 2/4

- **Unknown complex** data distribution p_{data}
- **Data**: what do you have access to?

“Implicit” Generative Models: Problem Setting 2/4

- **Unknown complex** data distribution p_{data}
- **Data**: what do you have access to?

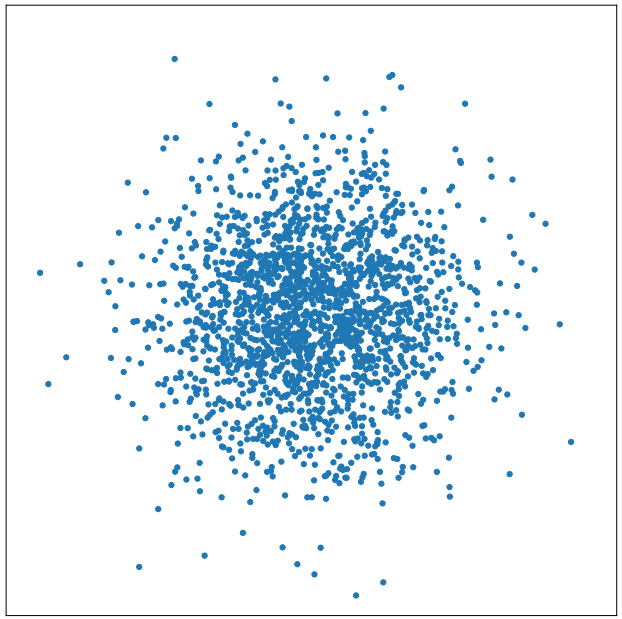


Data points $x^{(i)} \in \mathbb{R}^2$, $x^{(i)} \sim p_{\text{data}}$

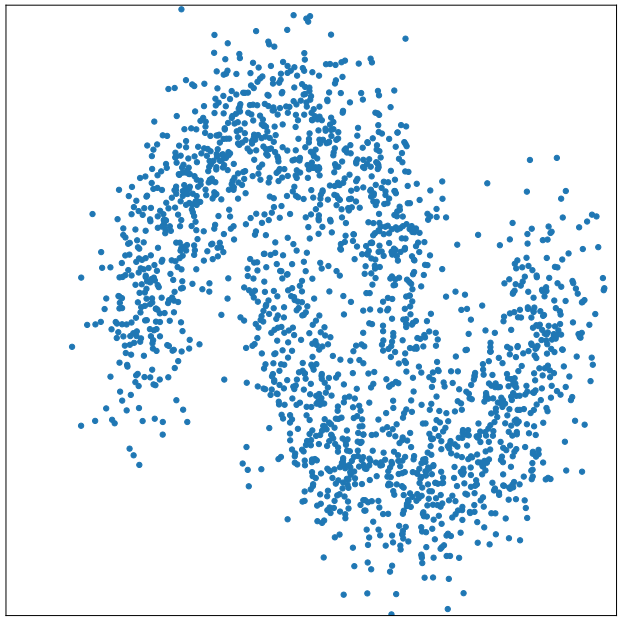
"Implicit" Generative Models: Problem Setting 3/4

- **Unknown complex** data distribution p_{data}
- **Goal**: transform a **simple** data distribution p_0 in the **complex** p_{data}

Points $x_0 \sim p_0$ drawn from a **simple** data distribution $\mathcal{N}(0, 1)$



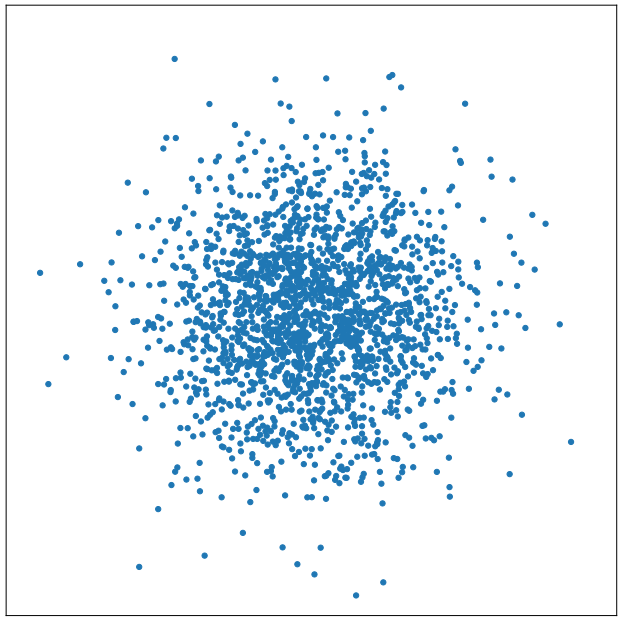
Data points $x^{(i)} \in \mathbb{R}^2$ from a **complex data distribution** $x^{(i)} \sim p_{\text{data}}$



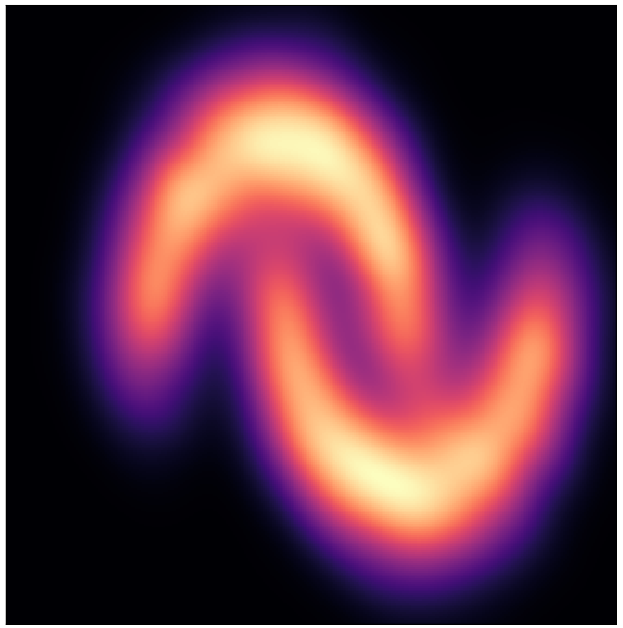
"Implicit" Generative Models: Problem Setting 4/4

- **Unknown complex** data distribution p_{data}
- **Goal**: transform a **simple** data distribution p_0 in the **complex** p_{data}

Points $x_0 \sim p_0$ drawn from a **simple** data distribution $\mathcal{N}(0, 1)$

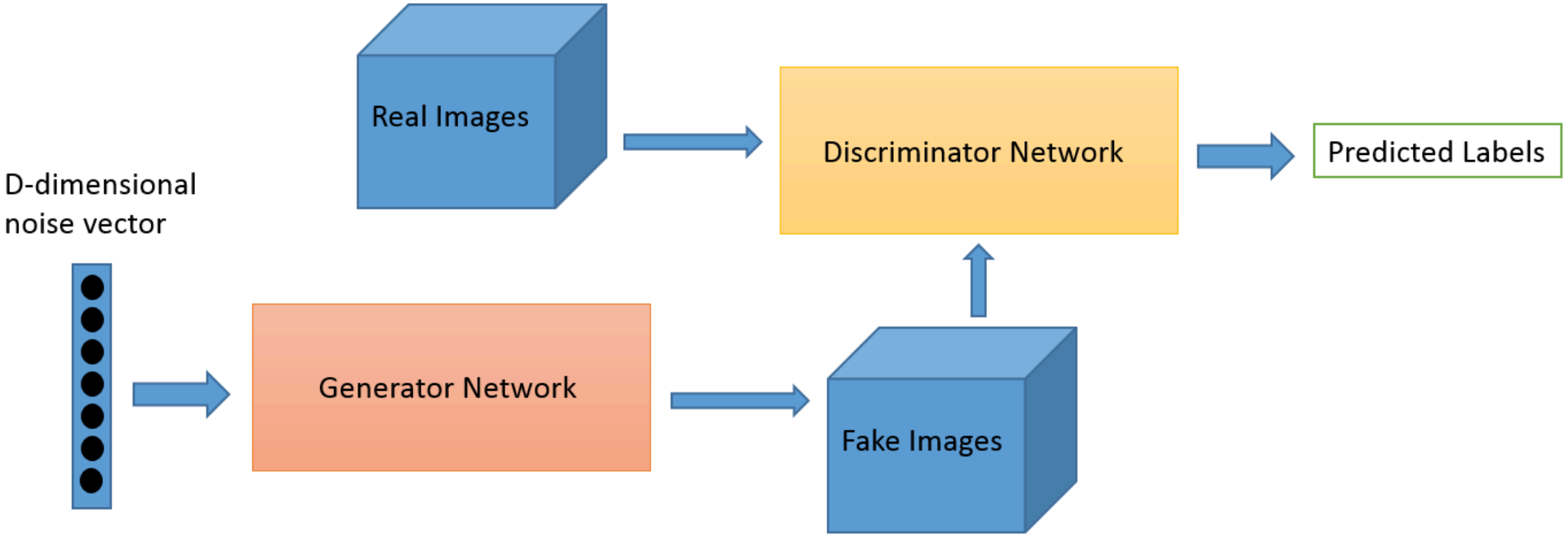


Complex data distribution p_{data}



Generative Adversarial Networks

Generative Adversarial Networks: General Idea



https://datascience.eu/machine-learning/a-beginners-guide-to-generative-adversarial-networks-gans/

Generative Adversarial Networks: Illustration

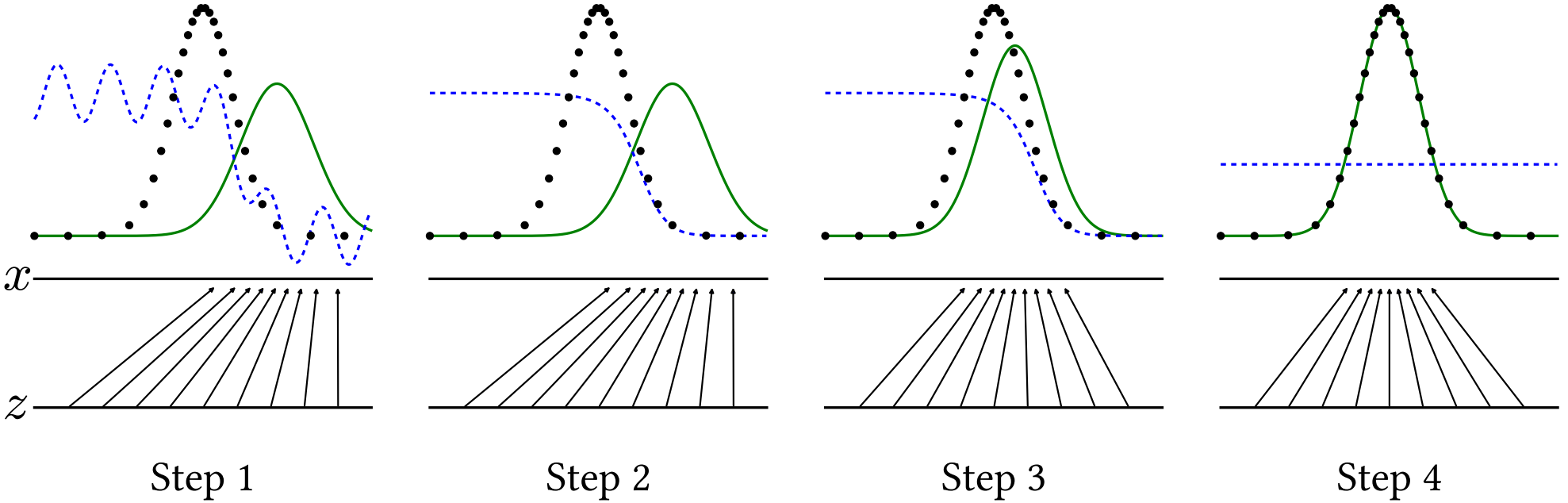


Image from Goodfellow et al. (2014)

How to Derive the GAN Loss?

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:
    - $\mathbb{P}(\text{head}|D) = D$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:
    - $\mathbb{P}(\text{head}|D) = D$, $\mathbb{P}(\text{tail}|D) = 1 - D$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:
    - $\mathbb{P}(\text{head}|D) = D$, $\mathbb{P}(\text{tail}|D) = 1 - D$
    - Likelihood $\mathbb{P}(x|D) = D^x(1 - D)^{1-x}$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:
    - $\mathbb{P}(\text{head}|D) = D$, $\mathbb{P}(\text{tail}|D) = 1 - D$
    - Likelihood $\mathbb{P}(x|D) = D^x(1 - D)^{1-x}$
    - Log-Likelihood $\log \mathbb{P}(x|D) = x \cdot \log D + (1 - x) \cdot \log(1 - D)$

Generative Adversarial Networks: Derive the Loss 1/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Key Idea:

- Maximum log-likelihood
  - For a biased coin:
    - $\mathbb{P}(\text{head}|D) = D$, $\mathbb{P}(\text{tail}|D) = 1 - D$
    - Likelihood $\mathbb{P}(x|D) = D^x(1 - D)^{1-x}$
    - Log-Likelihood $\log \mathbb{P}(x|D) = x \cdot \log D + (1 - x) \cdot \log(1 - D)$

  Maximum log-likelihood: for a given sequence of observations x_i

$$\max_D \sum_i x_i \cdot \log D + \sum_i (1 - x_i) \cdot \log(1 - D)$$

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

$$\mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

$$\mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

$$\mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_{\text{noise}}}(\log(1 - D(G(x_{\text{noise}}))))$$

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

$$\mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_{\text{noise}}}(\log(1 - D(G(x_{\text{noise}}))))$$

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_{\text{noise}}}(\log(1 - D(G(x_{\text{noise}}))))$$

Generative Adversarial Networks: Derive the Loss 2/2

We want to train:

- 2 networks
  - A generator $G : \mathbb{R}^d \mapsto \mathbb{R}^d$
  - A discriminator $D : \mathbb{R}^d \mapsto [0, 1]$
    - Tell how much an image is fake or not

$$\mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_{\text{noise}}}(\log(1 - D(G(x_{\text{noise}}))))$$

$$\min_G \max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_{\text{noise}}}(\log(1 - D(G(x_{\text{noise}}))))$$

What problem can you foresee?

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

- Require specific algorithms

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

- Require specific algorithms
- Tedious to tune

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

- Require specific algorithms
- Tedious to tune

Comment:

- Closed-form formula for the max:

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

- Require specific algorithms
- Tedious to tune

Comment:

- Closed-form formula for the max:

$$\arg\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}}))) = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

Generative Adversarial Networks: Comment on the Loss

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) := \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{noise}} \sim p_G}(\log(1 - D(G(x_{\text{noise}}))))$$

Min-max problem: **very challenging**

- Require specific algorithms
- Tedious to tune

Comment:

- Closed-form formula for the max:

$$\arg\max_D \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}}))) = \frac{p_{\text{data}}}{p_{\text{data}} + p_G}$$

Hence

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\substack{\text{compare} \\ \text{distribution}}}(p_{\text{data}} \parallel p_G)$$

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

- replace JS, “Jensen Shanon” divergence

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

- replace JS, “Jensen Shanon” divergence
- With something else that compares distributions:

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

- replace JS, “Jensen Shanon” divergence
- With something else that compares distributions:
  - *e.g.*, Wasserstein Distance

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

- replace JS, "Jensen Shanon" divergence
- With something else that compares distributions:
  - *e.g.*, Wasserstein Distance
  - *e.g.*, Maximum Mean Discrepency (MMD)

$$\min_G \max_D \mathcal{L}_{\text{WGAN}}(G, D) = \min_G \underbrace{\text{Wasserstein}}_{\text{compares distribution}} (p_{\text{data}}, p_G)$$

Wasserstein Generative Adversarial Networks (WGANs): Idea

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(G, D) = \min_G \underbrace{\text{JS}}_{\text{compares distribution}} (p_{\text{data}} \parallel p_G)$$

WGANs Idea:

- replace JS, "Jensen Shanon" divergence
- With something else that compares distributions:
  - *e.g.*, Wasserstein Distance
  - *e.g.*, Maximum Mean Discrepency (MMD)

$$\min_G \max_D \mathcal{L}_{\text{WGAN}}(G, D) = \min_G \underbrace{\text{Wasserstein}}_{\text{compares distribution}} (p_{\text{data}}, p_G)$$

Rewrite the Wasserstein distance, and after computations

$$\min_G \max_F \mathcal{L}_{\text{WGAN}}(G, F) = \min_G \max_{F,\ \text{Lip}(F) \leq 1} \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

GANs vs WGANs

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

GANs vs WGANs

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

GANs vs WGANs

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

- **Additional "Lipschitz" constraint** $\text{Lip}(F) \leq 1$

GANs vs WGANs

$$\mathcal{L}_{\text{GAN}}(G, D) = \mathbb{E}_{x \sim p_{\text{data}}}(\log(D(x))) + \mathbb{E}_{x_{\text{fake}} \sim p_G}(\log(1 - D(x_{\text{fake}})))$$

$$\mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

- **Additional "Lipschitz" constraint** $\text{Lip}(F) \leq 1$

How to enforce the constraint?

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 1/2

Any Idea?

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 1/2

Any Idea?

- Weight clipping
- Original WGANs paper (Arjovsky, Chintala, and Bottou 2017)

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 1/2

Any Idea?

- Weight clipping
- Original WGANs paper (Arjovsky, Chintala, and Bottou 2017)

while θ has not converged **do**
  for $t = 0, \ldots, n_{\text{critic}}$ **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch from the real data.
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$
    $w \leftarrow \mathrm{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples.
  $g_\theta \leftarrow -\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$
  $\theta \leftarrow \theta - \alpha \cdot \mathrm{RMSProp}(\theta, g_\theta)$

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 2/2

$$\mathcal{L}_{\mathrm{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\mathrm{data}}} F(x) - \mathbb{E}_{x_{\mathrm{fake}} \sim p_G} F(x_{\mathrm{fake}})$$

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 2/2

$$\mathcal{L}_{\mathrm{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\mathrm{data}}} F(x) - \mathbb{E}_{x_{\mathrm{fake}} \sim p_G} F(x_{\mathrm{fake}})$$

- **Additional "Lipschitz" constraint** $\mathrm{Lip}(F) \leq 1$

Idea to enforce the constraint?

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 2/2

$$\mathcal{L}_{\mathrm{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\mathrm{data}}} F(x) - \mathbb{E}_{x_{\mathrm{fake}} \sim p_G} F(x_{\mathrm{fake}})$$

- **Additional "Lipschitz" constraint** $\mathrm{Lip}(F) \leq 1$

Idea to enforce the constraint?

- Gradient Penalty!
  - $\mathrm{Lip}(F) \leq 1$ is equivalent to
  - $\| \nabla_x F \| \sim 1$

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 2/2

$$\mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

- **Additional "Lipschitz" constraint** $\mathrm{Lip}(F) \leq 1$

Idea to enforce the constraint?

- Gradient Penalty!
  - $\mathrm{Lip}(F) \leq 1$ is equivalent to
  - $\| \nabla_x F \| \sim 1$

$$\mathcal{L}_{\text{WGAN-gp}}(G, F) = \underbrace{\mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})}_{\mathcal{L}_{\text{WGAN}}(G,F)} + (\| \nabla_x F \| - 1)^2$$

WGANs: How to Enforce the Constraint? $\mathrm{Lip}(F) \leq 1$ 2/2

$$\mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

- **Additional "Lipschitz" constraint** $\mathrm{Lip}(F) \leq 1$

Idea to enforce the constraint?

- Gradient Penalty!
  - $\mathrm{Lip}(F) \leq 1$ is equivalent to
  - $\| \nabla_x F \| \sim 1$

$$\mathcal{L}_{\text{WGAN-gp}}(G, F) = \underbrace{\mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})}_{\mathcal{L}_{\text{WGAN}}(G,F)} + (\| \nabla_x F \| - 1)^2$$

$$\mathcal{L}_{\text{WGAN-gp}}(G, F) = \underbrace{\mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})}_{\mathcal{L}_{\text{WGAN}}(G,F)} + \underbrace{\mathbb{E}_{x \sim p_{\text{data}} + p_G} (\| \nabla_x F(x) \| - 1)^2}_{\text{gradient penalty}}$$

LAB 1: WGANs

Pytorch recalls?

Pytorch 101 Before the LAB

```
def train_one_epoch(training_loader):
    for i, data in enumerate(training_loader):
        # Every data instance is an input + label pair
        inputs, labels = data

        # Zero your gradients for every batch!
        optimizer.zero_grad()

        # Make predictions for this batch
        outputs = model(inputs)

        # Compute the loss and its gradients
        loss = loss_fn(outputs, labels)
        loss.backward()

        # Adjust learning weights
        optimizer.step()
```

WGAN and WGAN-gp Losses

$$\min_G \max_F \mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

WGAN and WGAN-gp Losses

$$\min_G \max_F \mathcal{L}_{\text{WGAN}}(G, F) = \mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})$$

$$\mathcal{L}_{\text{WGAN-gp}}(G, F) = \underbrace{\mathbb{E}_{x \sim p_{\text{data}}} F(x) - \mathbb{E}_{x_{\text{fake}} \sim p_G} F(x_{\text{fake}})}_{\mathcal{L}_{\text{WGAN}}(G,F)} + \underbrace{\mathbb{E}_{x \sim p_{\text{data}} + p_G} (\| \nabla_x F(x) \| - 1)^2}_{\text{gradient penalty}}$$

References

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. 2017. “Wasserstein Generative Adversarial Networks”. In *International Conference on Machine Learning*, 214–23.

Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. “Generative Adversarial Networks”. *Neurips*.

Lam, Remi, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, et al. 2023. “Learning Skillful Medium-Range Global Weather Forecasting”. *Science* 382 (6677): 1416–21.

Stability AI. 2023.

Watson, Joseph L, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, et al. 2023. “De Novo Design of Protein Structure and Function with Rfdiffusion”. *Nature*, 1–3.

References

Xu, Minkai, Tomas Geffner, Karsten Kreis, Weili Nie, Yilun Xu, Jure Leskovec, Stefano Ermon, and Arash Vahdat. 2025. “Energy-Based Diffusion Language Models for Text Generation”. *ICLR*.