

The **First (TvAI) Skyroom**
International **Virtual Congress** on
the practical Application of Artificial
Intelligence in **Medical Sciences**
Date & Time: 1-5 February, 2025 (09:00 Am - 12:00)



تاریخ و زمان برگزاری: ۱۳ تا ۱۷ بهمن ۱۴۰۵ (۰۹:۰۰ صبح - ۱۲:۰۰)

اولین کنگره بین المللی مجازی
کاربرد هوش مصنوعی
در علوم پزشکی



Medical Image Generation with Limited Data

Milad Abdollahzadeh
Singapore Institute of Technology
BetterData AI

Speaker



Milad Abdollahzadeh

Research Manager, Singapore Institute of Technology

Research Scientist, Betterdata AI

MILAD ABDOLLAHZADEH, TOUBA MALEKZADEH*, CHRISTOPHER T.H. TEO*,
KESHIGEYAN CHANDRASEGARAN*, GUIMENG LIU, and NGAI-MAN CHEUNG†,
Singapore University of Technology and Design (SUTD), Singapore



A Survey on Generative Modeling with Limited Data, Few Shots, and Zero Shot

MILAD ABDOLLAHZADEH, TOUBA MALEKZADEH*, CHRISTOPHER T.H. TEO*, KESHIGEYAN CHANDRASEGARAN*, GUIMENG LIU, and NGAI-MAN CHEUNG†, Singapore University of Technology and Design (SUTD), Singapore



A Survey on Generative Modeling with Limited Data, Few Shots, and Zero Shot

Milad Abdollahzadeh, Touba Malekzadeh*, Christopher T. H. Teo*, Keshigeyan Chandrasegaran*, Guimeng Liu, Ngai-Man Cheung†,

* Equal Contribution † Corresponding Author

Singapore University of Technology and Design

[arXiv](#) [Code](#) [Interactive Sankey](#)

Outline

- I. Background of Generative Models**
- II. Major Challenges of Training Generative Models with Limited Data**
- III. Approaches for Image Generation with Limited Data**

Outline

I. Background of Generative Models

II. Major Challenges of Training Generative Models with Limited Data

III. Approaches for Image Generation with Limited Data

Background of Generative Modeling

Definition of Domain and Generative Modeling

Unconditional vs Conditional Image Generation

Popular Image Generation Models

- **Variational AutoEncoder (VAE)**
- **Generative Adversarial Networks (GAN)**
- **Flow-based Models**
- **Diffusion Models (DM)**

Background of Generative Modeling

Definition of Domain and Generative Modeling

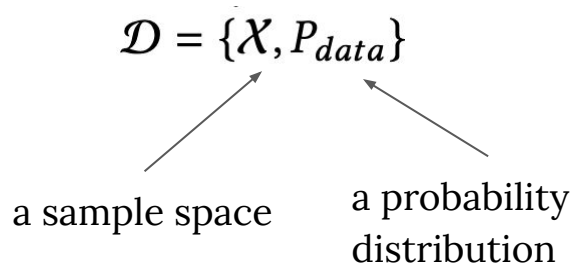
Unconditional vs Conditional Image Generation

Popular Image Generation Models

- Variational AutoEncoder (VAE)
- Generative Adversarial Networks (GAN)
- Flow-based Models
- Diffusion Models (DM)

Background: Definition of Domain

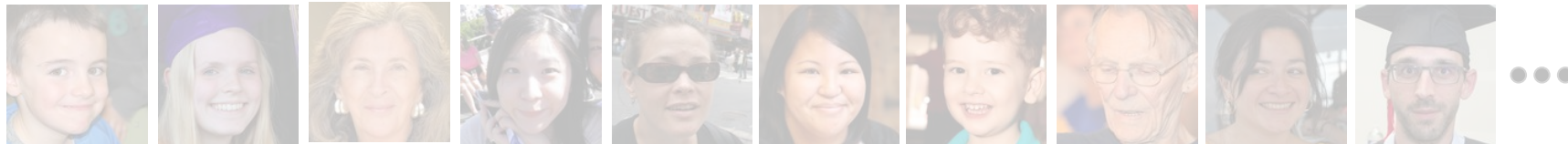
Domain. a domain consists of two components:



a sample from a domain

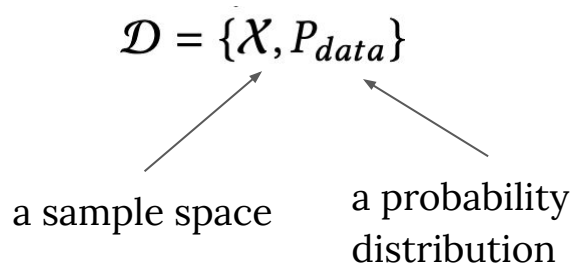
$$x \sim P_{data} \in \mathcal{X}$$

Example. Flickr-Faces-HQ (FFHQ) as the domain of image of human faces



Background: Definition of Domain

Domain. a domain consists of two components:



a sample from a domain

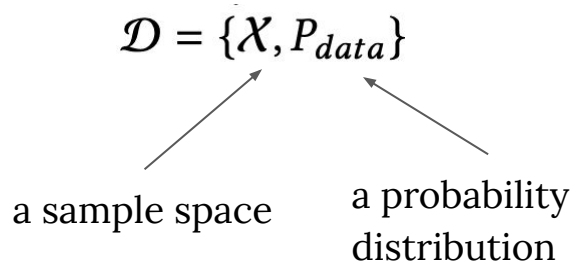
$$x \sim P_{data} \in \mathcal{X}$$

Example. Flickr-Faces-HQ (FFHQ) as the domain of image of human faces



Background: Definition of Domain

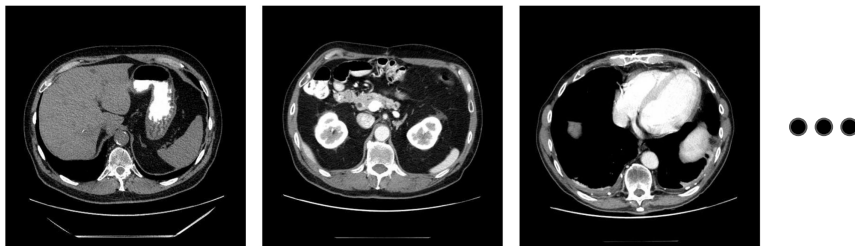
Domain. a domain consists of two components:



a sample from a domain

$$x \sim P_{data} \in \mathcal{X}$$

Example. SLIVER07 as the domain of 3D CT scans of livers



Background: Generative Modeling

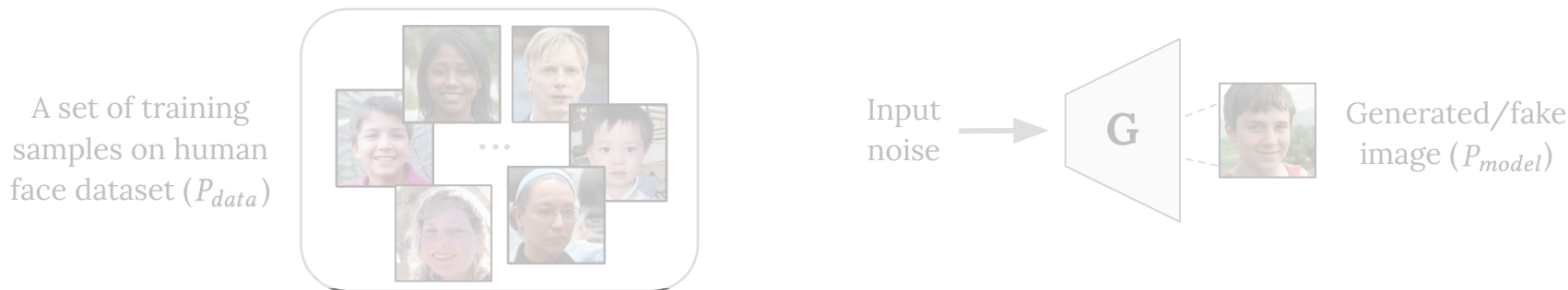
Generative Models.

Given a set of training samples from a domain $\mathcal{D} = \{\mathcal{X}, P_{data}\}$, generative modeling aims to learn to capture the distribution of these samples, i.e., P_{data} .

Result is a **generative model** G , encoding a probability distribution P_{model} .

Learning objective is to have P_{model} similar to P_{data} statistically.

After training, **G can generate samples** following P_{model} .



Background: Generative Modeling

Generative Models.

Given a set of training samples from a domain $\mathcal{D} = \{\mathcal{X}, P_{data}\}$, generative modeling aims to learn to capture the distribution of these samples, i.e., P_{data} .

Result is a **generative model** G , encoding a probability distribution P_{model} .

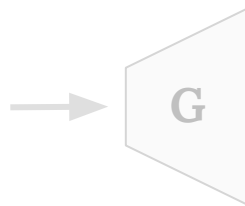
Learning objective is to have P_{model} similar to P_{data} statistically.

After training, G can generate samples following P_{model} .

A set of training samples on human face dataset (P_{data})



Input noise



Generated/fake image (P_{model})

Background: Generative Modeling

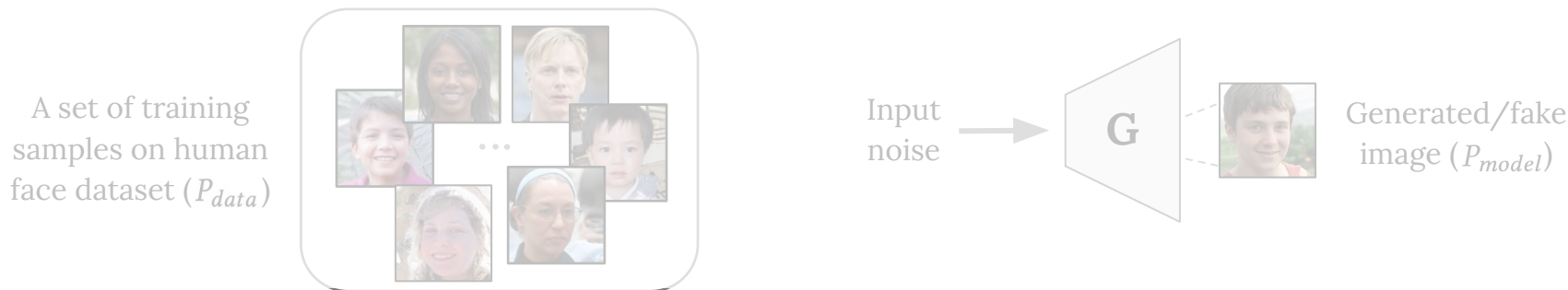
Generative Models.

Given a set of training samples from a domain $\mathcal{D} = \{\mathcal{X}, P_{data}\}$, generative modeling aims to learn to capture the distribution of these samples, i.e., P_{data} .

Result is a **generative model** G , encoding a probability distribution P_{model} .

Learning objective is to have P_{model} similar to P_{data} statistically.

After training, G can generate samples following P_{model} .



Background: Generative Modeling

Generative Models.

Given a set of training samples from a domain $\mathcal{D} = \{\mathcal{X}, P_{data}\}$, generative modeling aims to learn to capture the distribution of these samples, i.e., P_{data} .

Result is a **generative model** G , encoding a probability distribution P_{model} .

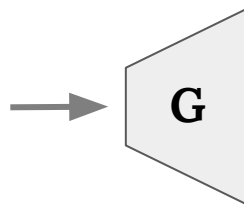
Learning objective is to have P_{model} similar to P_{data} statistically.

After training, **G can generate samples** following P_{model} .

A set of training samples on human face dataset (P_{data})



Input noise

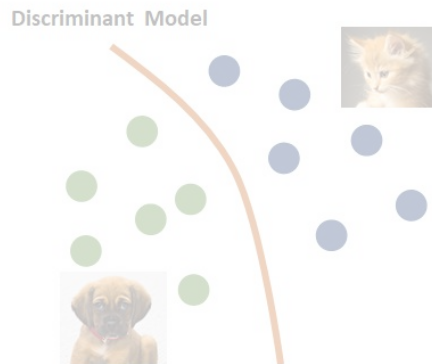


Generated/fake image (P_{model})

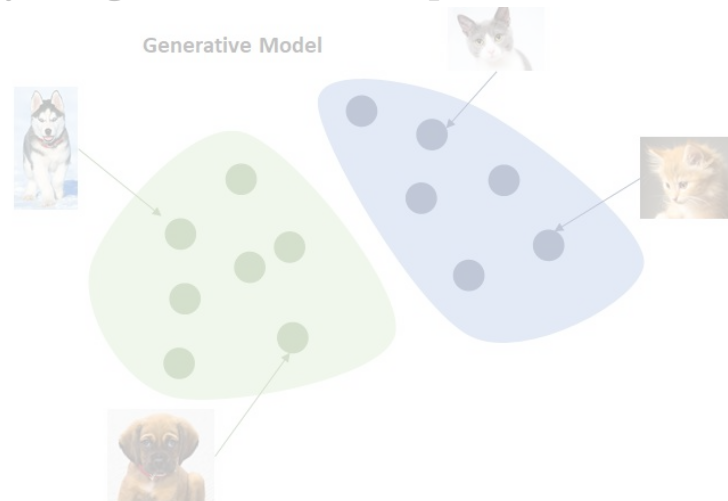
Background: Generative Modeling

Generative vs Discriminative Modeling

Discriminative Models learn the boundary in data space to be able to discriminate samples from different classes (e.g., ResNet classifier).



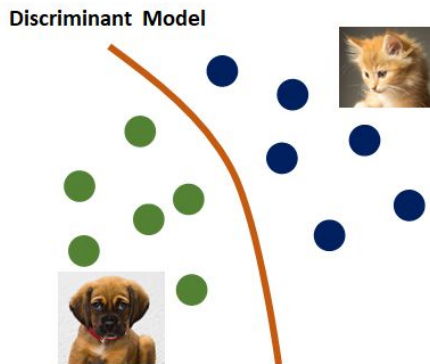
Generative Models learn the distribution of the data itself. Later, by sampling from learned distribution, they can **generate new samples**.



Background: Generative Modeling

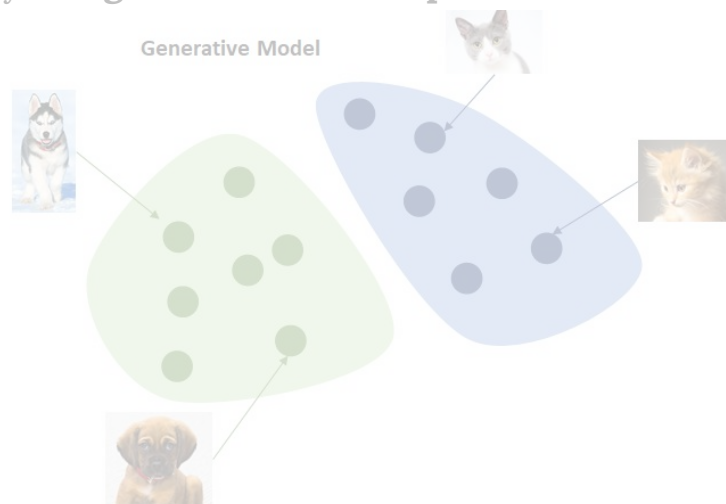
Generative vs Discriminative Modeling

Discriminative Models learn the boundary in data space to be able to discriminate samples from different classes (e.g., ResNet classifier).



Generative Models learn the distribution of the data itself.

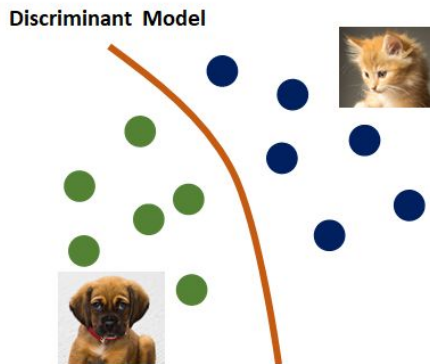
Later, by sampling from learned distribution, they can **generate new samples**.



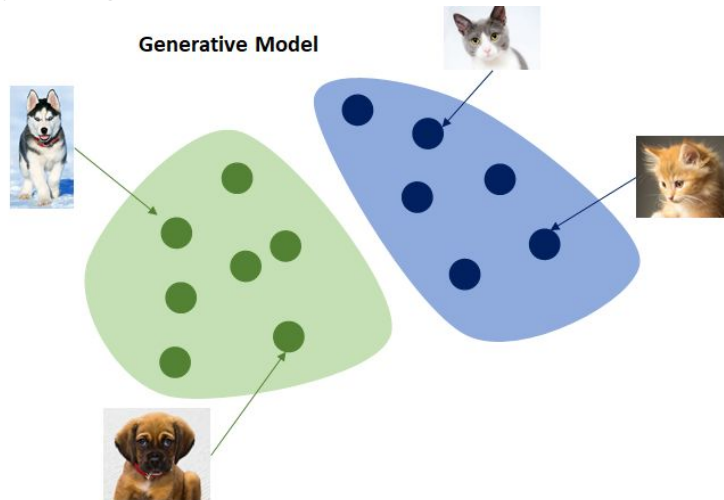
Background: Generative Modeling

Generative vs Discriminative Modeling

Discriminative Models learn the boundary in data space to be able to discriminate samples from different classes (e.g., ResNet classifier).



Generative Models learn the distribution of the data itself. Later, by sampling from learned distribution, they can **generate new samples**.



Background of Generative Modeling

Definition of Domain and Generative Modeling

Unconditional vs Conditional Image Generation

Popular Image Generation Models

- Variational AutoEncoder (VAE)
- Generative Adversarial Networks (GAN)
- Flow-based Models
- Diffusion Models (DM)

Background: Unconditional vs Conditional Image Generation

Image Generation: after learning generative model, typically:

- image generation starts with sampling a random noise z (also called latent code) as input.
- This input is passed into generative model G which transforms it to a new sample $G(z)$

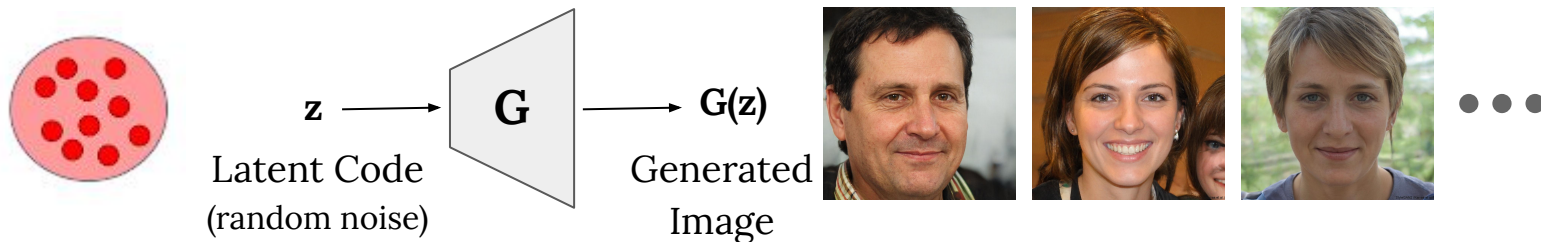


This is called **unconditional image generation** as there is no conditioning mechanism to restrict image generation.

Background: Unconditional vs Conditional Image Generation

Image Generation: after learning generative model, typically:

- image generation starts with sampling a random noise z (also called latent code) as input.
- This input is passed into generative model G which transforms it to a new sample $G(z)$



This is called **unconditional image generation** as there is no conditioning mechanism to restrict image generation.

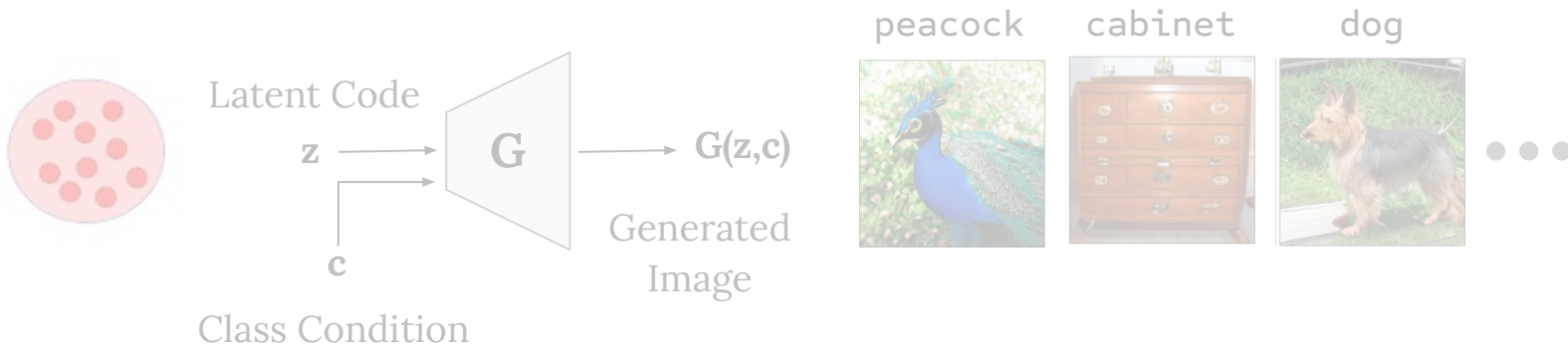
Background: Unconditional vs Conditional Image Generation

Image Generation: after learning generative model, typically:

- image generation starts with sampling a random noise z (also called latent code) as input.
- This input is passed into generative model G which transforms it to a new sample $G(z)$

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z,c)$

For example, **conditioning on class label**:



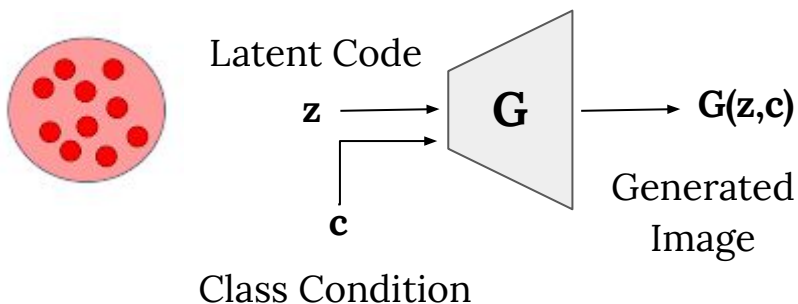
Background: Unconditional vs Conditional Image Generation

Image Generation: after learning generative model, typically:

- image generation starts with sampling a random noise z (also called latent code) as input.
- This input is passed into generative model G which transforms it to a new sample $G(z)$

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

For example, **conditioning on class label**:



peacock



cabinet



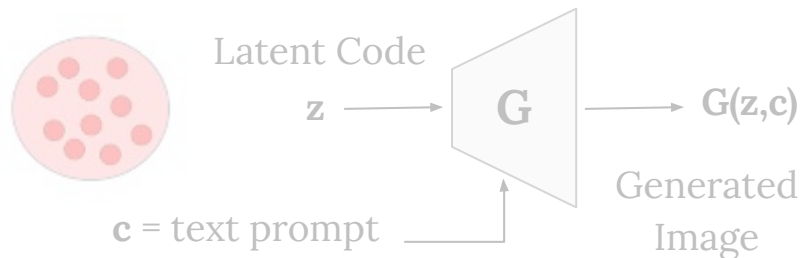
dog



Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

conditioning on text prompt



An art installation floats in the air, the installation is a cute shark made of candy, bright colors, light gray background, studio, contemporary art, minimalism, telephoto lens, large aperture photography,

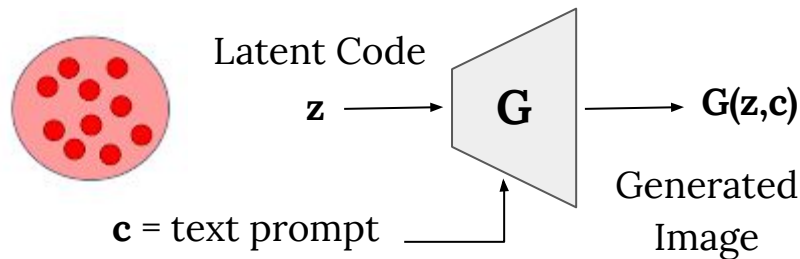


Generator: Midjourney

Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

conditioning on text prompt



An art installation floats in the air, the installation is a cute shark made of candy, bright colors, light gray background, studio, contemporary art, minimalism, telephoto lens, large aperture photography

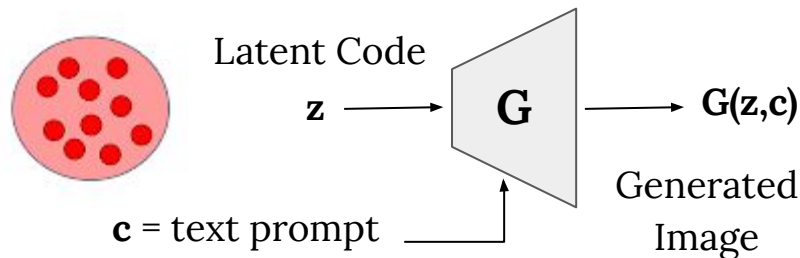


Generator: Midjourney

Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

conditioning on text prompt



Fundoscopy image of the
left retina with no
Diabetic retinopathy

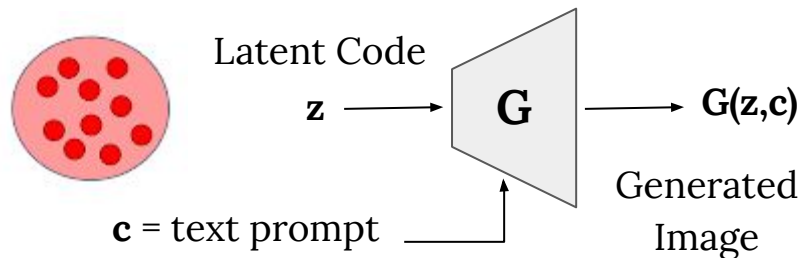


Generator:
[MediSyn] Text-Guided Diffusion Models
for Broad Medical 2D and 3D Image
Synthesis (Stanford Medicine)

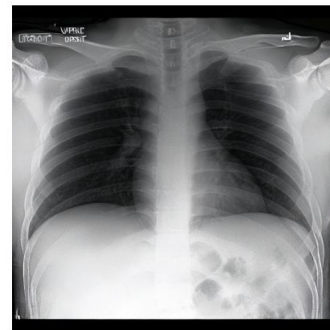
Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z,c)$

conditioning on text prompt



AP Frontal Chest X-ray
(CXR) of a male patient



Generator:

[MediSyn] Text-Guided Diffusion Models
for Broad Medical 2D and 3D Image
Synthesis (Stanford Medicine)

Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

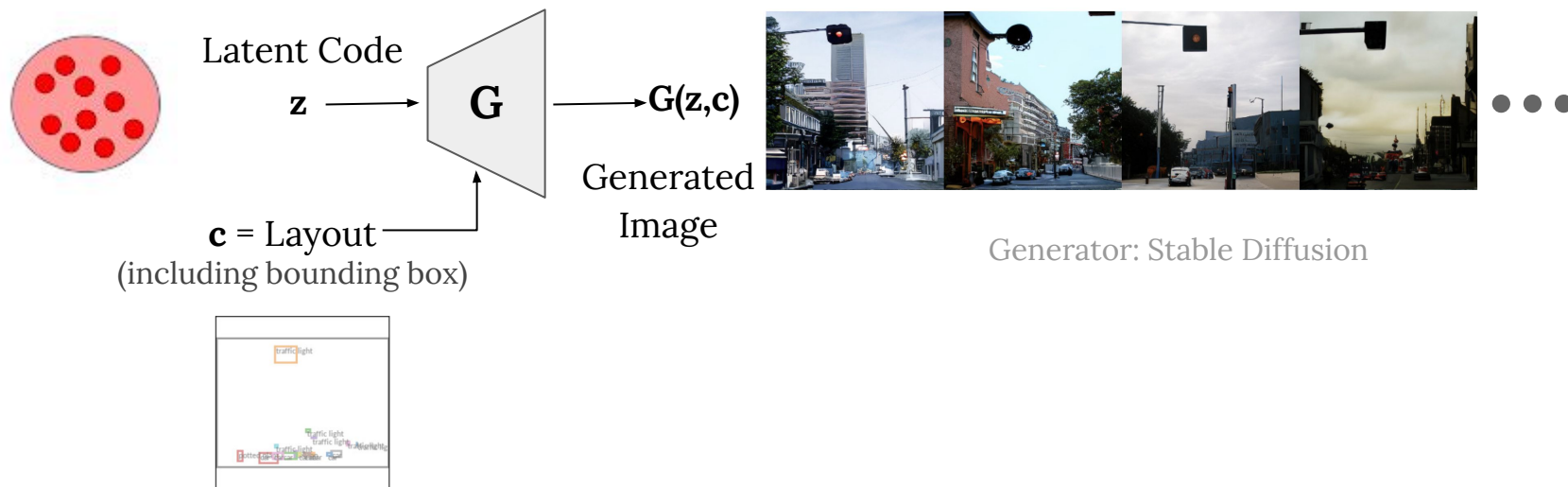
conditioning on layout



Background: Unconditional vs Conditional Image Generation

If **an additional condition c** (e.g., class label, ...) is used to steer sample generation towards c , the sample generation is called **conditional image generation**: $G(z, c)$

conditioning on layout



Background of Generative Modeling

Definition of Domain and Generative Modeling

Unconditional vs Conditional Image Generation

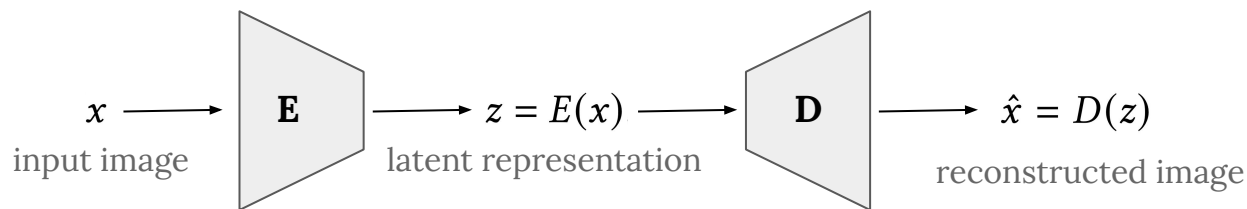
Popular Image Generation Models

- **Variational AutoEncoder (VAE)**
- **Generative Adversarial Networks (GAN)**
- **Diffusion Models (DM)**

Generative Models: Variational Auto-Encoder (VAE)

Auto-Encoder (AE) consist of **two networks**:

- **Encoder (E)** learn to map input image to a low-dimensional latent representation
- **Decoder (D)** aims to reconstruct the image from that latent representation



Learning objective:

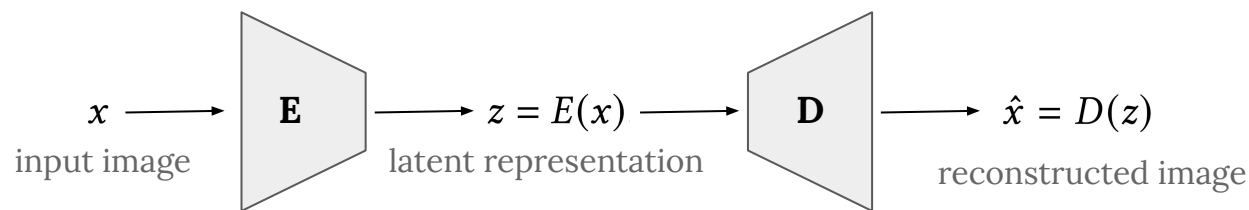
$$\mathcal{L}_{rec} = ||x - D(z)||_2$$

AEs focus on **dimensionality reduction**, and the **irregularity of their latent space** makes them improper for sample generation.

Generative Models: Variational Auto-Encoder (VAE)

Auto-Encoder (AE) consist of **two networks**:

- **Encoder (E)** learn to map input image to a low-dimensional latent representation
- **Decoder (D)** aims to reconstruct the image from that latent representation



Learning objective:

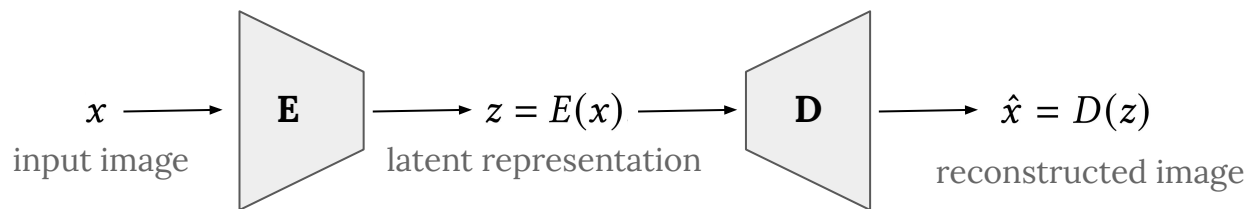
$$\mathcal{L}_{rec} = ||x - D(z)||_2$$

AEs focus on **dimensionality reduction**, and the **irregularity of their latent space** makes them improper for sample generation.

Generative Models: Variational Auto-Encoder (VAE)

Auto-Encoder (AE) consist of **two networks**:

- **Encoder (E)** learn to map input image to a low-dimensional latent representation
- **Decoder (D)** aims to reconstruct the image from that latent representation



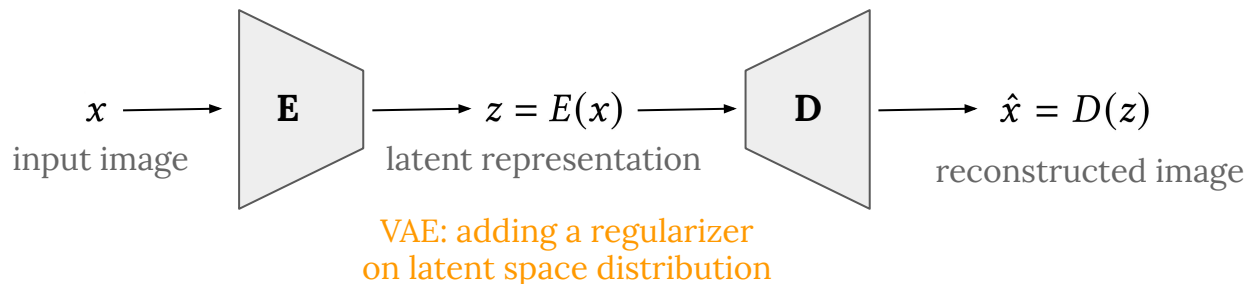
Learning objective:

$$\mathcal{L}_{rec} = ||x - D(z)||_2$$

AEs focus on **dimensionality reduction**, and the **irregularity of their latent space** makes them improper for sample generation.

Generative Models: **Variational Auto-Encoder (VAE)**

Variational Auto-Encoder (VAE) aims to address this irregularity by enforcing **E** to return a normal distribution over latent space.



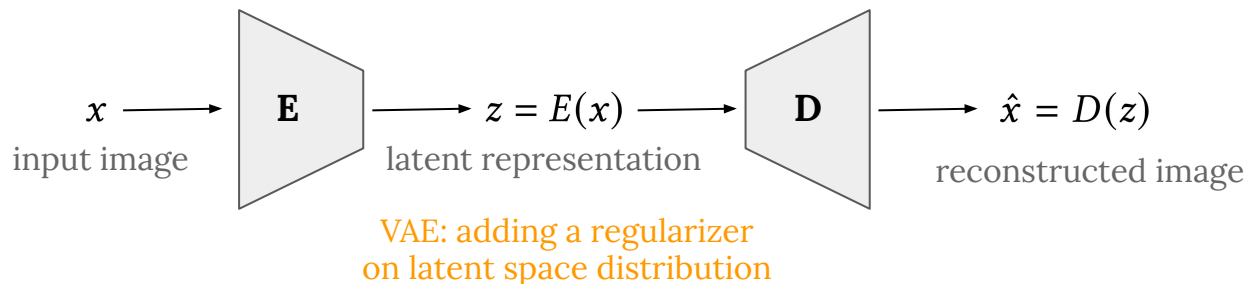
Learning objective:

$$\mathcal{L} = ||x - D(z)||_2 + KL(\mathcal{N}(\mu, \sigma^2), \mathcal{N}(0, I))$$

Vector-Quantized VAE (VQ-VAE) adds tokenization which quantizes the embedding into visual tokens for mitigating the challenge of direct maximization of likelihood in image space.

Generative Models: **Variational Auto-Encoder (VAE)**

Variational Auto-Encoder (VAE) aims to address this irregularity by enforcing **E** to return a normal distribution over latent space.



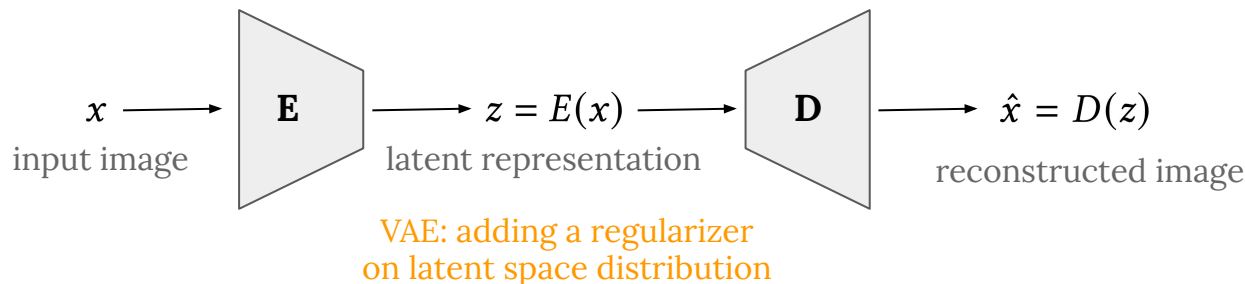
Learning objective:

$$\mathcal{L} = \|x - D(z)\|_2 + KL(\mathcal{N}(\mu, \sigma^2), \mathcal{N}(0, I))$$

Vector-Quantized VAE (VQ-VAE) adds tokenization which quantizes the embedding into visual tokens for mitigating the challenge of direct maximization of likelihood in image space.

Generative Models: **Variational Auto-Encoder (VAE)**

Variational Auto-Encoder (VAE) aims to address this irregularity by enforcing **E** to return a normal distribution over latent space.



Learning objective:

$$\mathcal{L} = \|x - D(z)\|_2 + KL(\mathcal{N}(\mu, \sigma^2), \mathcal{N}(0, I))$$

Vector-Quantized VAE (VQ-VAE) adds tokenization which quantizes the embedding into visual tokens for mitigating the challenge of direct maximization of likelihood in image space.

Generative Models: Generative Adversarial Networks (GANs)

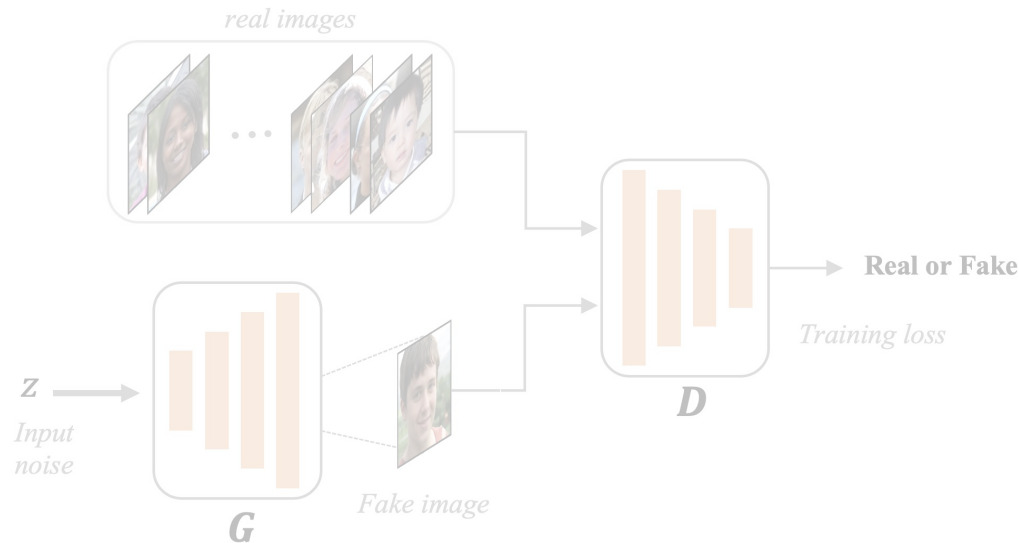
GANs consist of **two networks**:

- **Generator (G)** learn to generate images from input noise (latent code)
- **Discriminator (D)** learn to distinguish between **real** and **fake** images

These two network compete in a **min-max game**:

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

Finally **D is discarded** and **G is used for image generation**.



Generative Models: Generative Adversarial Networks (GANs)

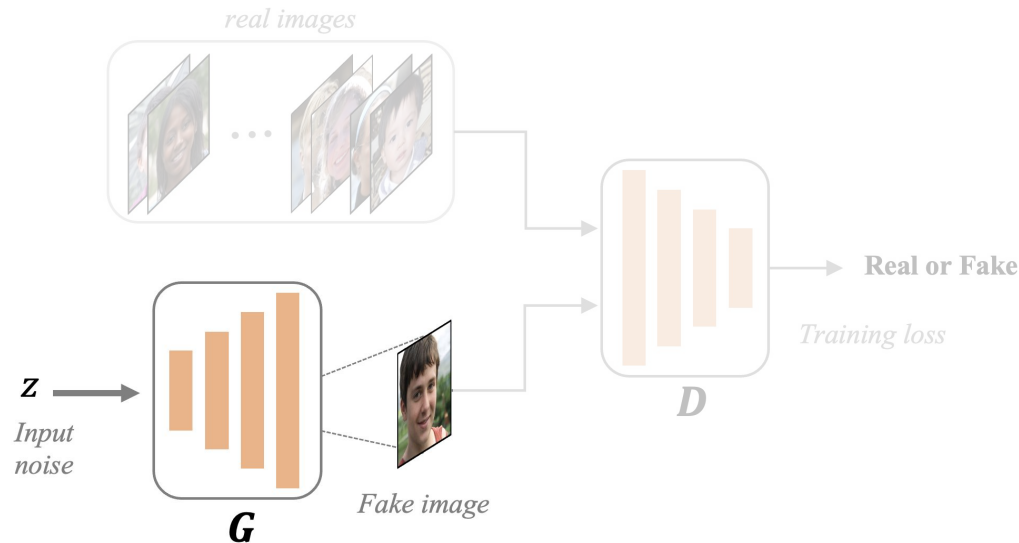
GANs consist of **two networks**:

- **Generator (G)** learn to generate images from input noise (latent code)
- **Discriminator (D)** learn to distinguish between **real** and **fake** images

These two network compete in a **min-max game**:

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

Finally **D is discarded** and **G is used for image generation**.



Generative Models: Generative Adversarial Networks (GANs)

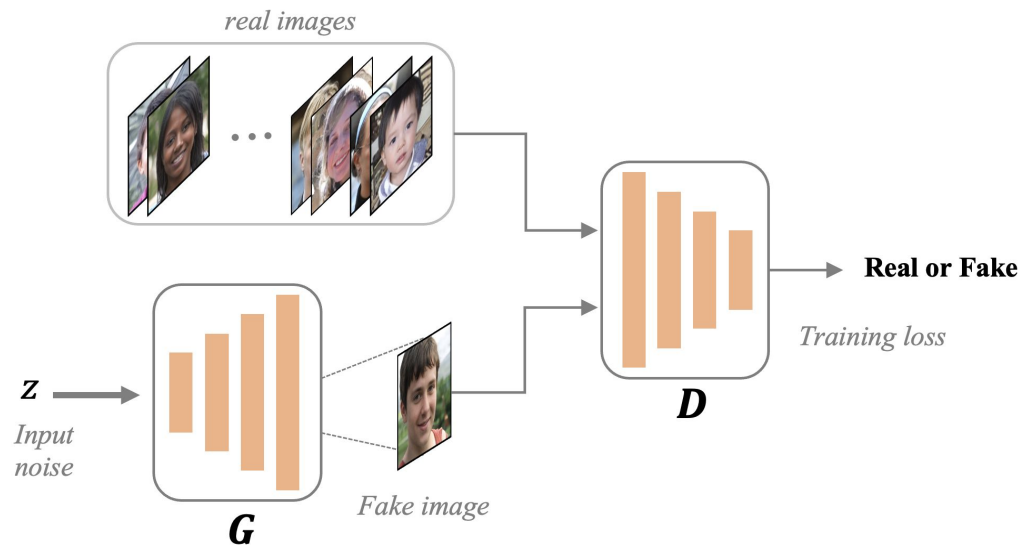
GANs consist of **two networks**:

- **Generator (G)** learn to generate images from input noise (latent code)
- **Discriminator (D)** learn to distinguish between **real** and **fake** images

These two network compete in a **min-max game**:

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

Finally **D is discarded** and **G is used for image generation**.



Generative Models: Generative Adversarial Networks (GANs)

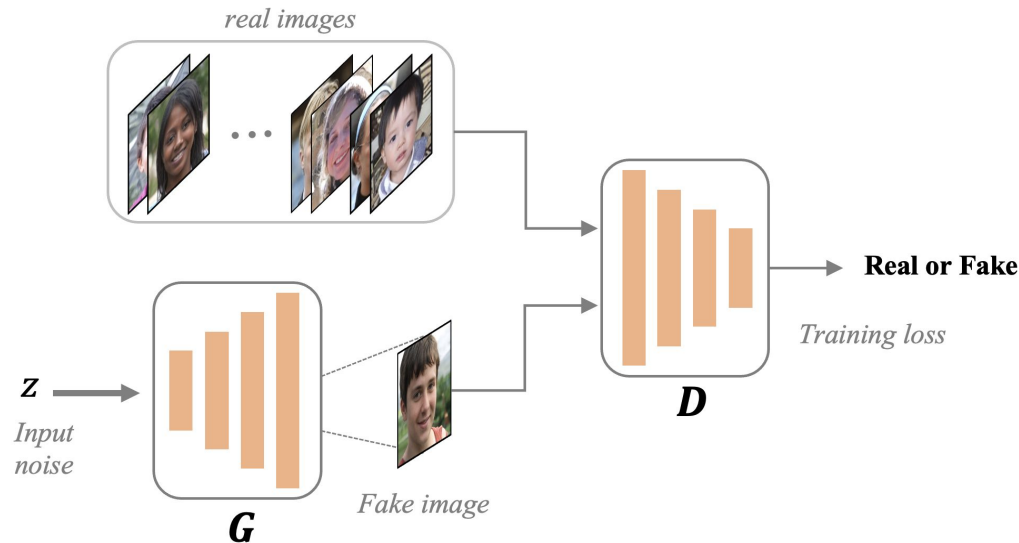
GANs consist of **two networks**:

- **Generator (G)** learn to generate images from input noise (latent code)
- **Discriminator (D)** learn to distinguish between **real** and **fake** images

These two network compete in a **min-max game**:

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

Finally **D is discarded** and **G is used for image generation**.



Generative Models: Generative Adversarial Networks (GANs)

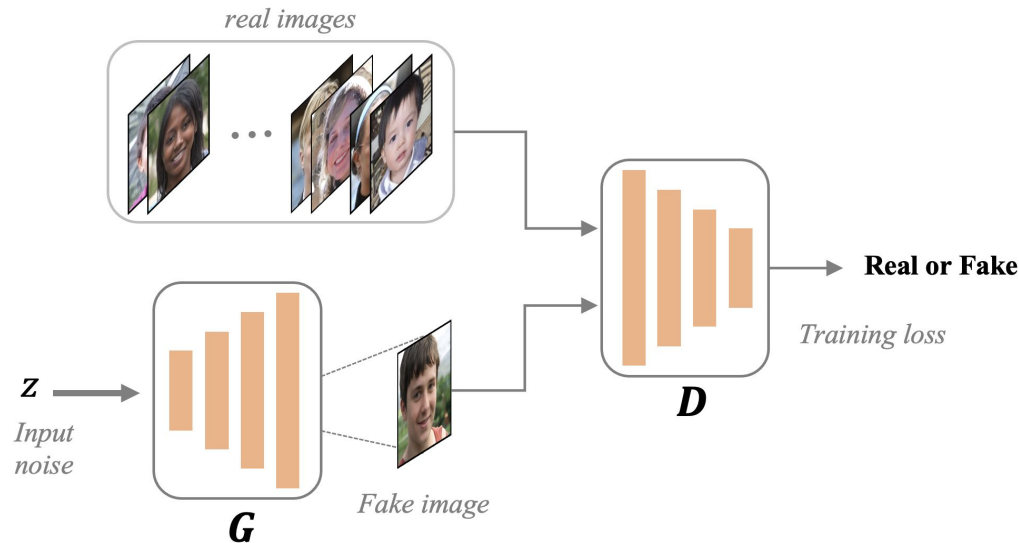
GANs consist of **two networks**:

- **Generator (G)** learn to generate images from input noise (latent code)
- **Discriminator (D)** learn to distinguish between **real** and **fake** images

These two network compete in a **min-max game**:

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

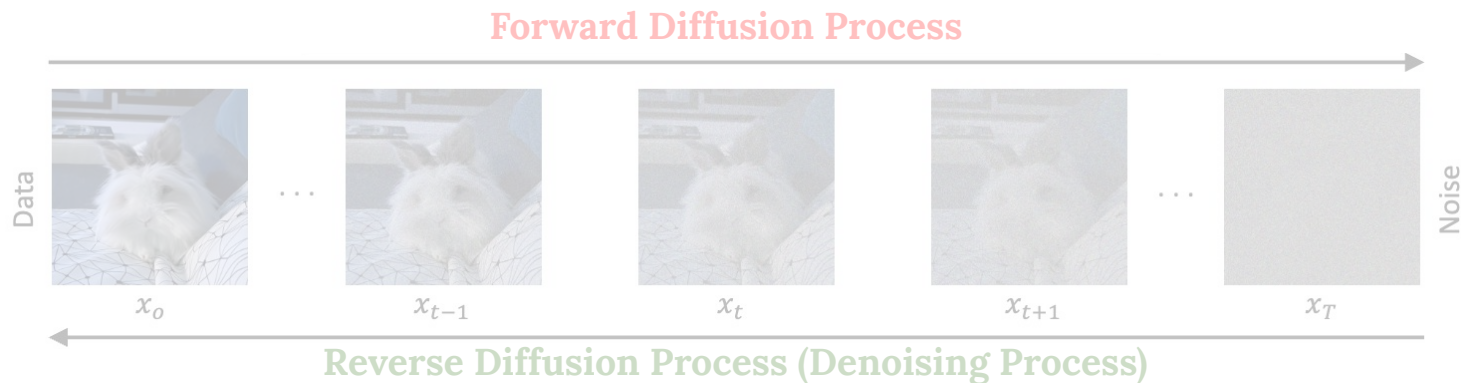
Finally **D is discarded** and **G is used for image generation**.



Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

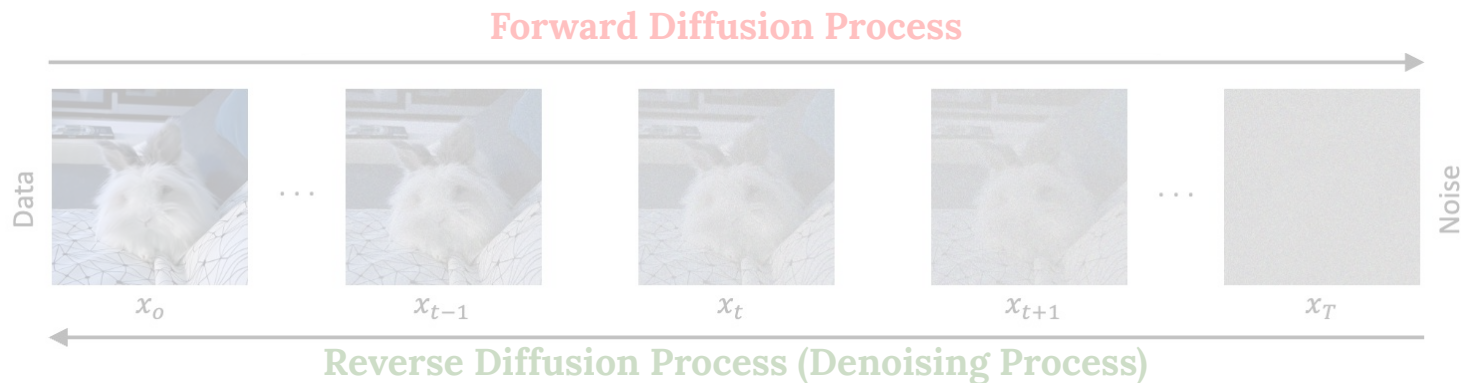
- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.
- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.



Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

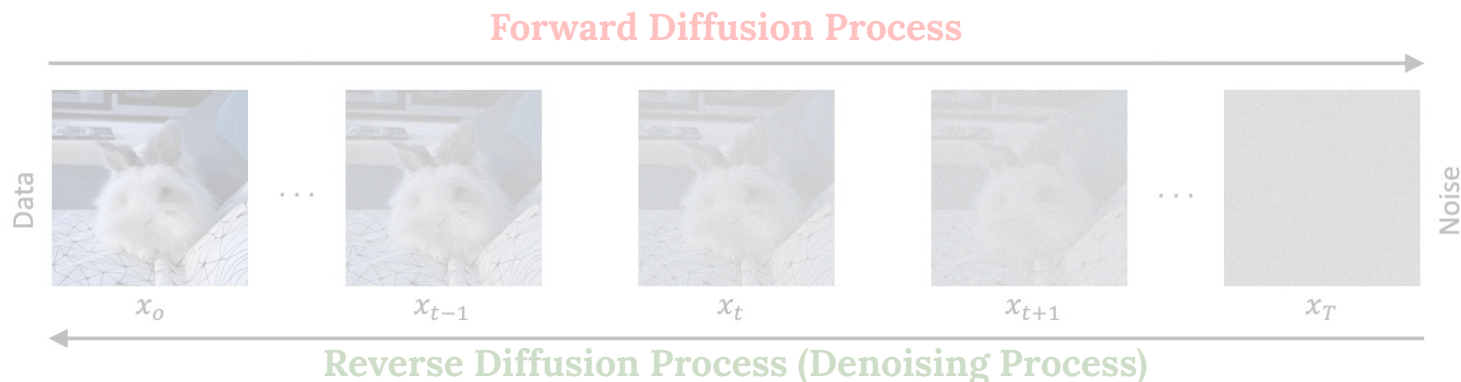
- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.
- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.



Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

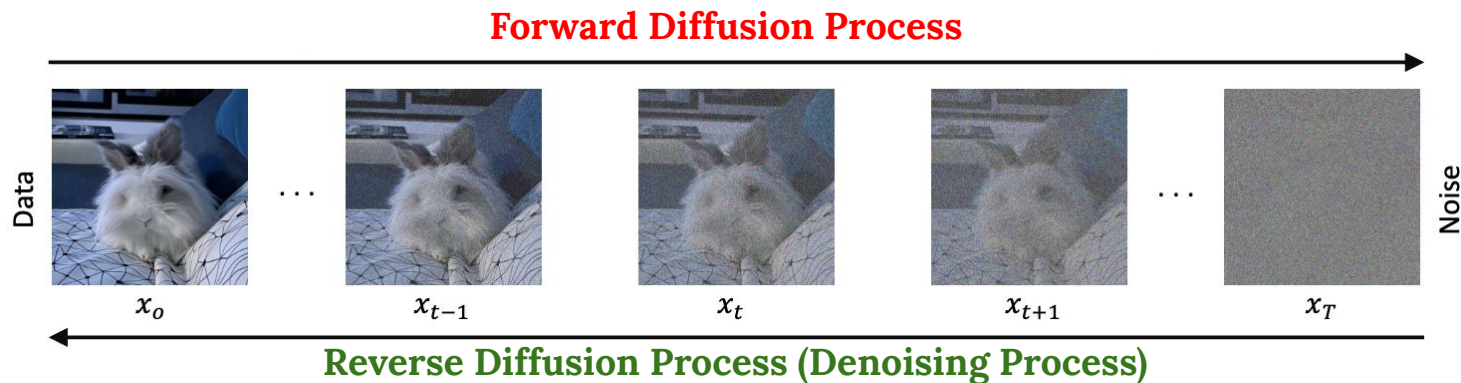
- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.
- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.



Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.
- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.



Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.

Given an uncorrupted training sample $x_0 \sim p(x_0)$

the noisy versions of this sample $x_1, x_2 \dots, x_T$ are obtained with following markovian process:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot \mathbf{I}), \forall t \in \{1, \dots, T\}$$

Important property of this formulation is the possibility to directly sample at time step t :

$$p(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\hat{\beta}_t} \cdot x_0, (1 - \hat{\beta}_t) \cdot \mathbf{I}), \quad \hat{\beta}_t = \prod_{i=1}^t \alpha_i \text{ and } \alpha_t = 1 - \beta_t$$

This is usually done by reparameterization trick: $x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} \cdot z_t$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.

Given an uncorrupted training sample $x_0 \sim p(x_0)$

the noisy versions of this sample $x_1, x_2, \dots, x_T$ are obtained with following markovian process:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot \mathbf{I}), \forall t \in \{1, \dots, T\}$$

Important property of this formulation is the possibility to directly sample at time step t :

$$p(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\hat{\beta}_t} \cdot x_0, (1 - \hat{\beta}_t) \cdot \mathbf{I}), \quad \hat{\beta}_t = \prod_{i=1}^t \alpha_i \text{ and } \alpha_t = 1 - \beta_t$$

This is usually done by reparameterization trick: $x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} \cdot z_t$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Forward Diffusion Process** consists of multiple steps in which low-level noise is added to each input image, where the scale of the noise varies at each step. The training data is progressively destroyed until it results in pure Gaussian noise.

Given an uncorrupted training sample $x_0 \sim p(x_0)$

the noisy versions of this sample $x_1, x_2 \dots, x_T$ are obtained with following markovian process:

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t} \cdot x_{t-1}, \beta_t \cdot \mathbf{I}), \forall t \in \{1, \dots, T\}$$

Important property of this formulation is the possibility to directly sample at time step t :

$$p(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\hat{\beta}_t} \cdot x_0, (1 - \hat{\beta}_t) \cdot \mathbf{I}), \quad \hat{\beta}_t = \prod_{i=1}^t \alpha_i \text{ and } \alpha_t = 1 - \beta_t$$

This is usually done by reparameterization trick: $x_t = \sqrt{\hat{\beta}_t} \cdot x_0 + \sqrt{(1 - \hat{\beta}_t)} \cdot z_t$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

We **train a neural network** (usually UNet) to approximate these steps

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

This network **predicts the mean and variance of the noise at time step t** based on the input and is trained to maximize the posterior distribution (a variational lower bound of it) of the real samples:

$$\mathcal{L}_{vlb} = -\log p_{\theta}(x_0|x_1) + KL(p(x_T|x_0)||\pi(x_T)) + \sum_{t>1} KL(p(x_{t-1}|x_t, x_0)||p_{\theta}(x_{t-1}|x_t))$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

We **train a neural network** (usually UNet) to approximate these steps

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

This network **predicts the mean and variance of the noise at time step t** based on the input and is trained to maximize the posterior distribution (a variational lower bound of it) of the real samples:

$$\mathcal{L}_{vlb} = -\log p_{\theta}(x_0|x_1) + KL(p(x_T|x_0) \parallel \pi(x_T)) + \sum_{t>1} KL(p(x_{t-1}|x_t, x_0) \parallel p_{\theta}(x_{t-1}|x_t))$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

We **train a neural network** (usually UNet) to approximate these steps

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

This network **predicts the mean and variance of the noise at time step t** based on the input and is trained to maximize the posterior distribution (a variational lower bound of it) of the real samples:

$$\mathcal{L}_{vlb} = -\log p_{\theta}(x_0|x_1) + KL(p(x_T|x_0) \parallel \pi(x_T)) + \sum_{t>1} KL(p(x_{t-1}|x_t, x_0) \parallel p_{\theta}(x_{t-1}|x_t))$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

We **train a neural network** (usually UNet) to approximate these steps

$$p_{\theta}(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_{\theta}(x_t, t), \Sigma_{\theta}(x_t, t))$$

This network **predicts the mean and variance of the noise at time step t** based on the input and is trained to maximize the posterior distribution (a variational lower bound of it) of the real samples:

$$\mathcal{L}_{vlb} = -\log p_{\theta}(x_0|x_1) + KL(p(x_T|x_0) \parallel \pi(x_T)) + \sum_{t>1} KL(p(x_{t-1}|x_t, x_0) \parallel p_{\theta}(x_{t-1}|x_t))$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

A **new simplified framework**, fixes the variance and rewrite the mean as a function of noise:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \cdot \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} \cdot z_\theta(x_t, t) \right)$$

Then, the **network is trained to predict noise itself** instead of mean and variance

$$\mathcal{L}_{simple} = \mathbb{E}_{t \sim [1, T]} \mathbb{E}_{x_0 \sim p(x_0)} \mathbb{E}_{z_t \sim \mathcal{N}(0, \mathbf{I})} \|z_t - z_\theta(x_t, t)\|^2$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

A **new simplified framework**, fixes the variance and rewrite the mean as a function of noise:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \cdot \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} \cdot z_\theta(x_t, t) \right)$$

Then, the **network is trained to predict noise itself** instead of mean and variance

$$\mathcal{L}_{simple} = \mathbb{E}_{t \sim [1, T]} \mathbb{E}_{x_0 \sim p(x_0)} \mathbb{E}_{z_t \sim \mathcal{N}(0, \mathbf{I})} \|z_t - z_\theta(x_t, t)\|^2$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

A **new simplified framework**, fixes the variance and rewrite the mean as a function of noise:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \cdot \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} \cdot z_\theta(x_t, t) \right)$$

Then, the **network is trained to predict noise itself** instead of mean and variance

$$\mathcal{L}_{simple} = \mathbb{E}_{t \sim [1, T]} \mathbb{E}_{x_0 \sim p(x_0)} \mathbb{E}_{z_t \sim \mathcal{N}(0, \mathbf{I})} \|z_t - z_\theta(x_t, t)\|^2$$

Generative Models: Diffusion Models (DMs)

Diffusion Models (DMs) leverage the **concepts of the diffusion process** from stochastic calculus and consists of two main steps:

- **Reverse Diffusion Process (Denoising Process)** has same iterative procedure, but backwards: the noise is sequentially removed, and hence, the original image is recreated.

Generate new samples from $p(x_0)$ starting from a random noise $x_T \sim \mathcal{N}(0, \mathbf{I})$ and following reverse steps:

$$p(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu(x_t, t), \Sigma(x_t, t)).$$

A **new simplified framework**, fixes the variance and rewrite the mean as a function of noise:

$$\mu_\theta = \frac{1}{\sqrt{\alpha_t}} \cdot \left(x_t - \frac{1 - \alpha_t}{\sqrt{1 - \hat{\beta}_t}} \cdot z_\theta(x_t, t) \right)$$

Popularity of DMs start from HERE!!

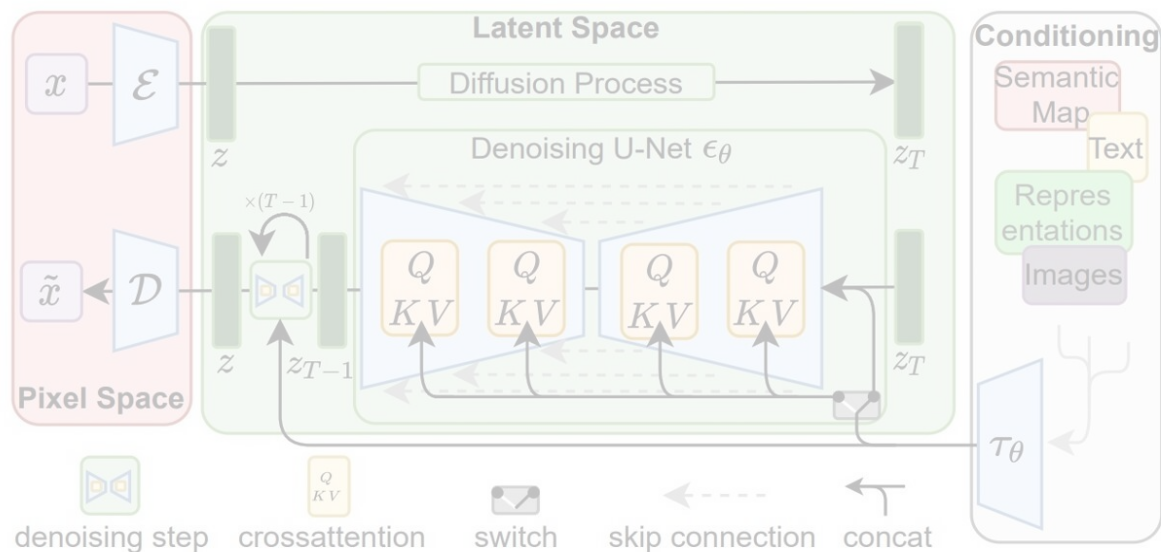
Then, the **network is trained to predict noise itself** instead of mean and variance

$$\mathcal{L}_{simple} = \mathbb{E}_{t \sim [1, T]} \mathbb{E}_{x_0 \sim p(x_0)} \mathbb{E}_{z_t \sim \mathcal{N}(0, \mathbf{I})} \|z_t - z_\theta(x_t, t)\|^2$$

Major Advances in DMs

Latent Diffusion Model (LDM), and **Stable Diffusion Model** (SDM)

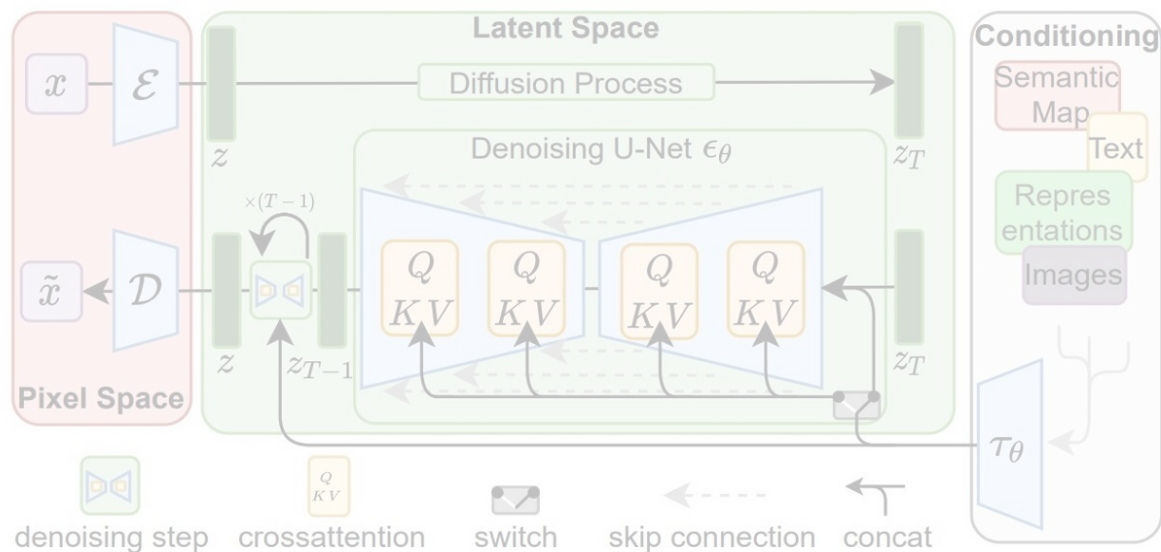
trains the denoiser network (reverse diffusion) in the latent space instead of image space
uses cross-attention mechanism for conditioning



Major Advances in DMs

Latent Diffusion Model (LDM), and **Stable Diffusion Model** (SDM)

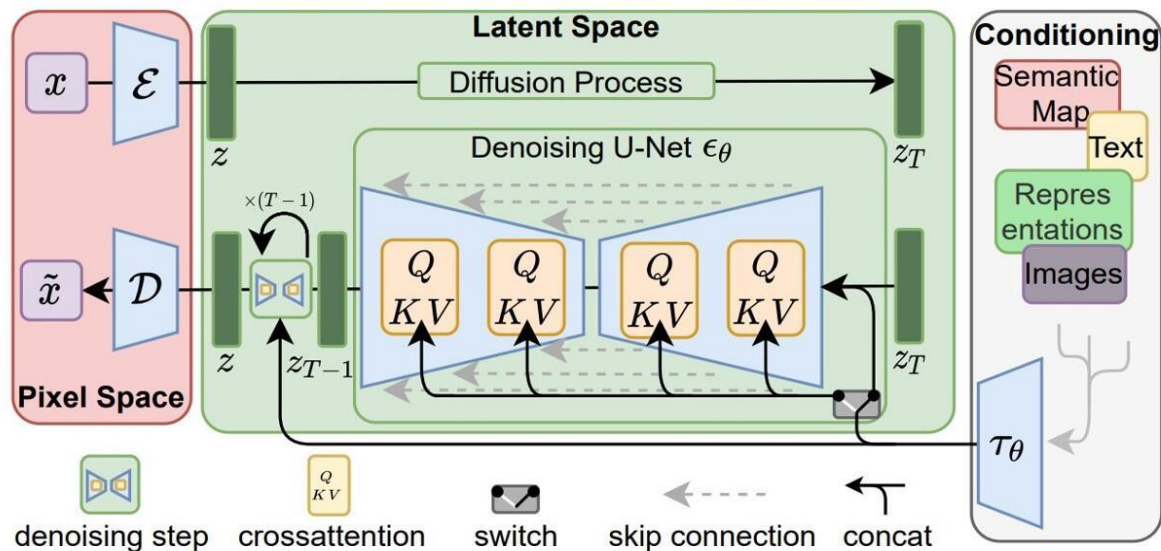
trains the denoiser network (reverse diffusion) in the latent space instead of image space
uses cross-attention mechanism for conditioning



Major Advances in DMs

Latent Diffusion Model (LDM), and **Stable Diffusion Model** (SDM)

trains the denoiser network (reverse diffusion) in the latent space instead of image space
uses cross-attention mechanism for conditioning



Outline

I. Background of Generative Models

II. Major Challenges of Training Generative Models with Limited Data

III. Approaches for Image Generation with Limited Data

Major Challenges

Challenges for Training Generative Models under Data Constraint

- **Overfitting**
- **Frequency Bias**

Major Challenges: **Training**

Overfitting to Training Data

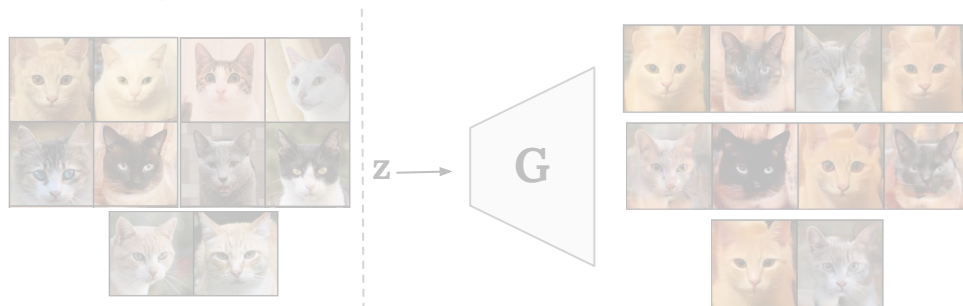
Overfitting. a common issue in machine learning when powerful models start to memorize the training data instead of learning the generalizable semantics

Mode Collapse. Under data constraints generative models are more prone to mode collapse



Example of mode collapse on MNIST handwritten digits

Replicate Training Data. Under extreme cases, the generator just learns to replicate the training data



10-shot training data

Generated images are extremely similar to training data

Major Challenges: **Training**

Overfitting to Training Data

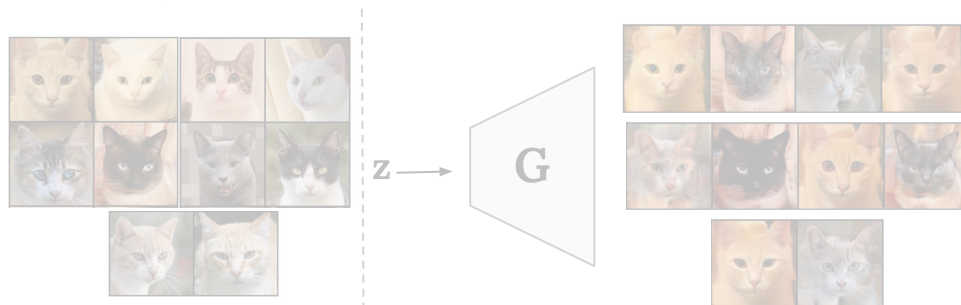
Overfitting. a common issue in machine learning when powerful models start to memorize the training data instead of learning the generalizable semantics

Mode Collapse. Under data constraints generative models are more prone to mode collapse



Example of mode collapse on MNIST handwritten digits

Replicate Training Data. Under extreme cases, the generator just learns to replicate the training data



10-shot training data

Generated images are extremely similar to training data

Major Challenges: **Training**

Overfitting to Training Data

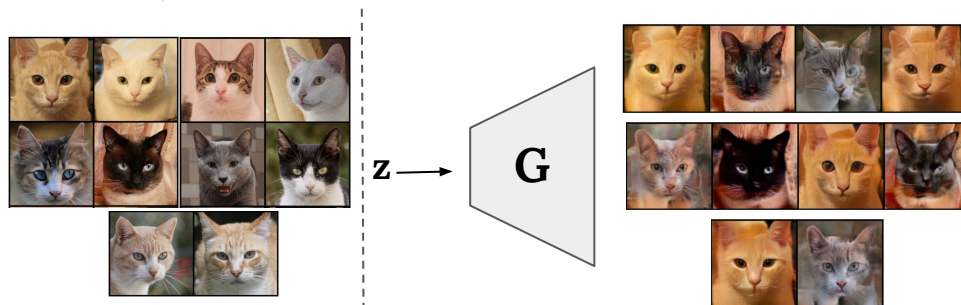
Overfitting. a common issue in machine learning when powerful models start to memorize the training data instead of learning the generalizable semantics

Mode Collapse. Under data constraints generative models are more prone to mode collapse



Example of mode collapse on MNIST handwritten digits

Replicate Training Data. Under extreme cases, the generator just learns to replicate the training data



Generated images are extremely similar to training data

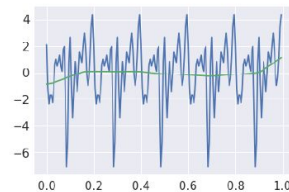
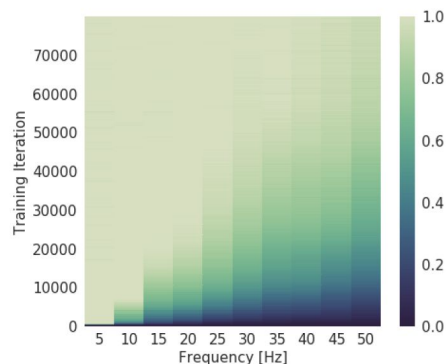
Major Challenges: **Training**

Frequency Bias: Neural networks learn lower frequencies first

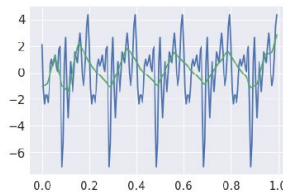
On the Spectral Bias of Neural Networks

Nasim Rahaman^{*1,2} Aristide Baratin^{*1} Devansh Arpit¹ Felix Draxler² Min Lin¹ Fred A. Hamprecht²
Yoshua Bengio¹ Aaron Courville¹

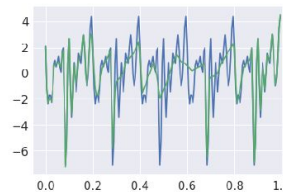
$$\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$$



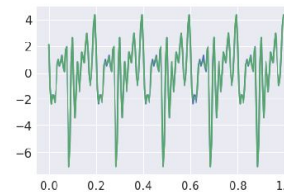
(a) Iteration 100



(b) Iteration 1000



(c) Iteration 10000



(d) Iteration 80000

Figure 2. The learnt function (green) overlaid on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, \dots, 45, 50)$, equal amplitudes and randomly sampled phases.

Major Challenges: **Training**

Frequency Bias: generative models **prioritize fitting low-frequency** components while disregarding the high-frequency components

Spatial Frequency Bias in Convolutional Generative Adversarial Networks

Mahyar Khayatkhoei, Ahmed Elgammal

Department of Computer Science, Rutgers University
New Brunswick, New Jersey
{m.khayatkhoei, elgammal}@cs.rutgers.edu

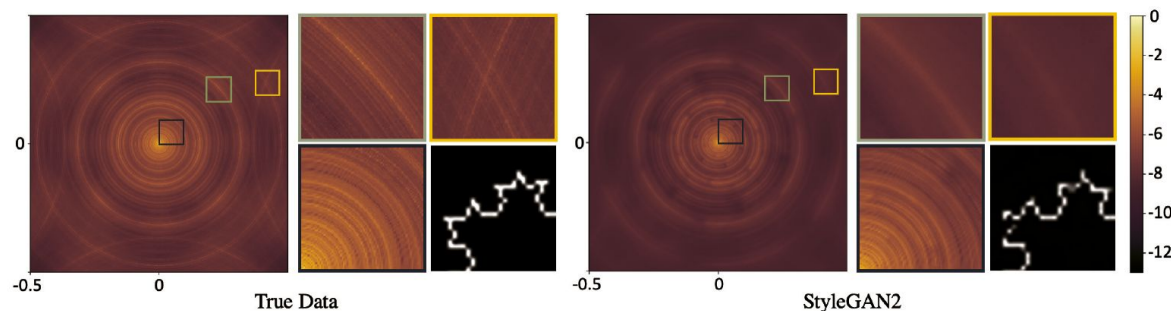


Figure 1: Average power spectrum of a large-scale GAN trained on a fractal-based dataset clearly reveals how the low frequencies (closer to center) are matched much more accurately than the high frequencies (closer to corners). (Left) Average power spectrum of randomly rotated Koch snowflakes of level 5 and size 1024×1024 . (Right) Average power spectrum of StyleGAN2 trained on the latter. A representative patch from the perimeter of true and generated fractals are also displayed.

Major Challenges: **Training**

Frequency Bias: generative models **prioritize fitting low-frequency** components while disregarding the high-frequency components

This bias **worsens** when data for training is limited.

Exclusion of these high-frequency components which encode intricate image details significantly impacts the quality of generated samples



Generated images by FastGAN showing loss of high-frequency details.

Outline

I. Background of Generative Models

II. Major Challenges of Training Generative Models with Limited Data

III. Approaches for Image Generation with Limited Data

Approaches

Approaches for Image Generation with Limited Data (Training Generative Models with Limited Data):

- 1. Transfer Learning**
- 2. Data Augmentation**
- 3. Network Architecture**

Approaches

Approaches for Image Generation with Limited Data (Training Generative Models with Limited Data):

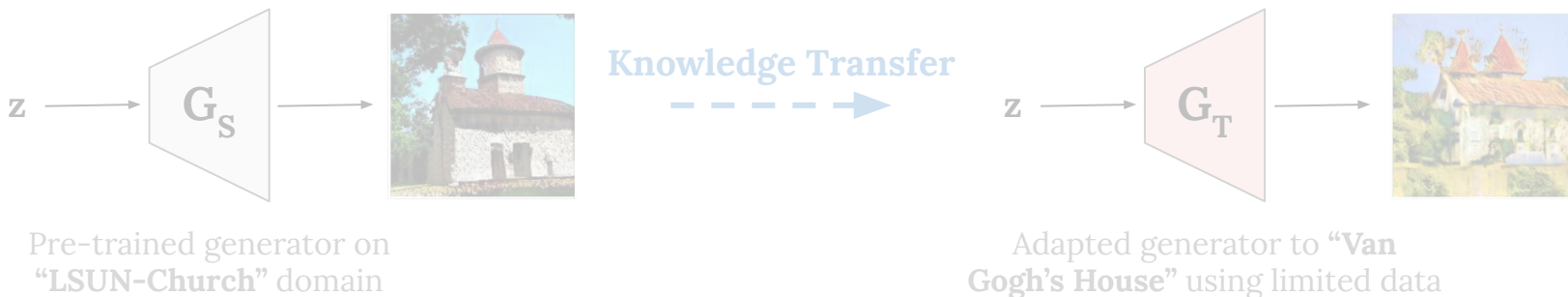
- 1. Transfer Learning**
2. Data Augmentation
3. Network Architecture

Approaches: **Transfer Learning**

Transfer Learning in Generative Modeling:

Transfer the knowledge of a pre-trained generator (on a large and diverse dataset) to a target domain with limited data

- ➡ Initialize the generator G_T with weights of pre-trained generator G_S
- ➡ Fine-tune G_T using limited data from target domain
- ➡ General knowledge (domain-agnostic) from G_S is useful for target domain, and the domain-specific knowledge needs to be acquired using limited data from target domain

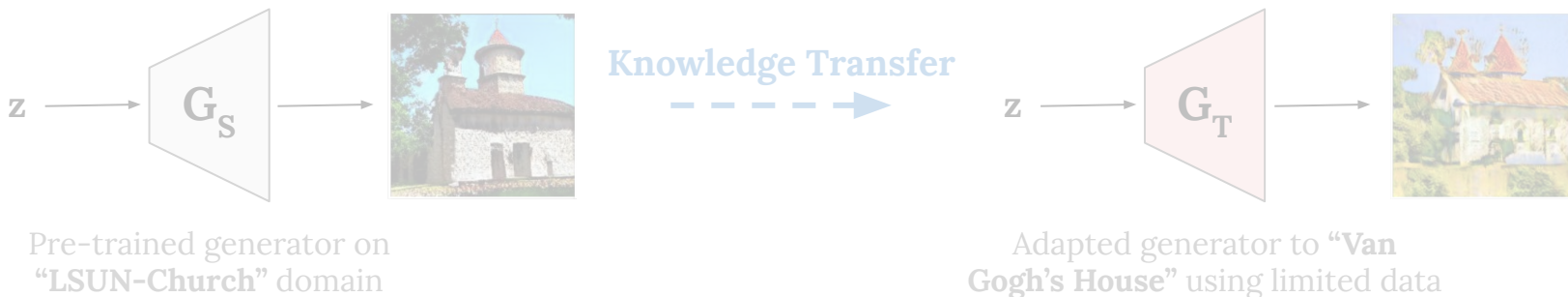


Approaches: **Transfer Learning**

Transfer Learning in Generative Modeling:

Transfer the knowledge of a pre-trained generator (on a large and diverse dataset) to a target domain with limited data

- ➡ Initialize the generator G_T with weights of pre-trained generator G_S
- ➡ Fine-tune G_T using limited data from target domain
- ➡ General knowledge (domain-agnostic) from G_S is useful for target domain, and the domain-specific knowledge needs to be acquired using limited data from target domain

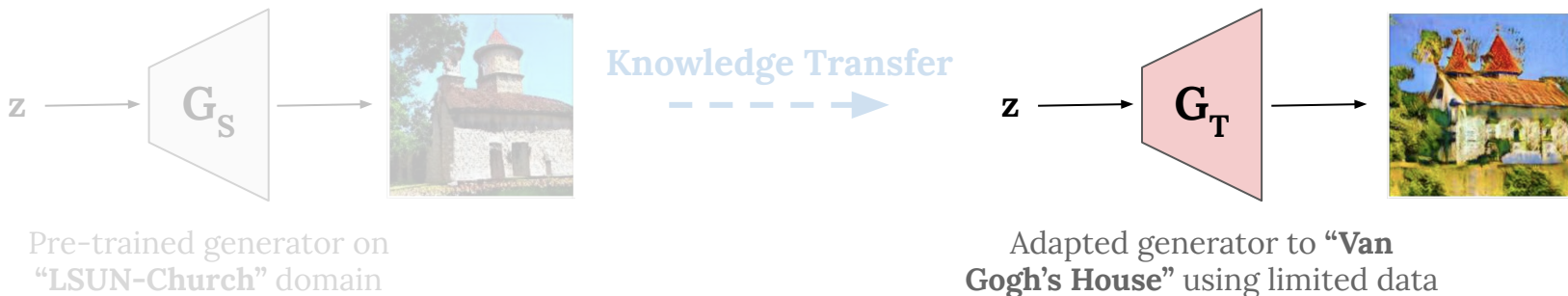


Approaches: **Transfer Learning**

Transfer Learning in Generative Modeling:

Transfer the knowledge of a pre-trained generator (on a large and diverse dataset) to a target domain with limited data

- ➡ Initialize the generator G_T with weights of pre-trained generator G_S
- ➡ Fine-tune G_T using limited data from target domain
- ➡ General knowledge (domain-agnostic) from G_S is useful for target domain, and the domain-specific knowledge needs to be acquired using limited data from target domain

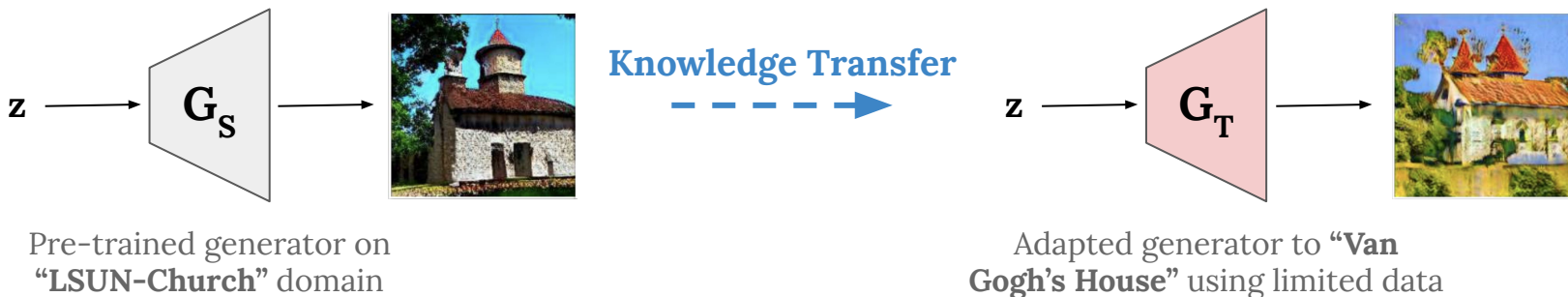


Approaches: **Transfer Learning**

Transfer Learning in Generative Modeling:

Transfer the knowledge of a pre-trained generator (on a large and diverse dataset) to a target domain with limited data

- ➡ Initialize the generator G_T with weights of pre-trained generator G_S
- ➡ Fine-tune G_T using limited data from target domain
- ➡ General knowledge (domain-agnostic) from G_S is useful for target domain, and the domain-specific knowledge needs to be acquired using limited data from target domain



Approaches: Transfer Learning

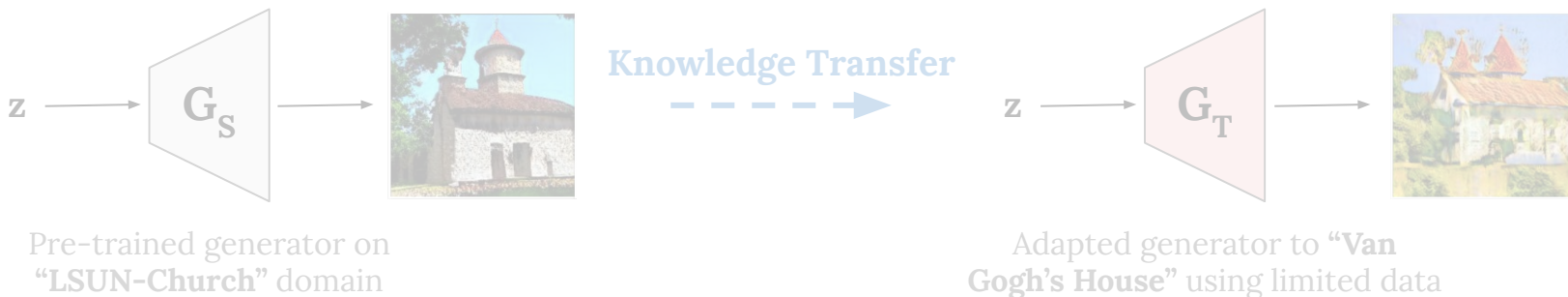
Transfer Learning in Generative Modeling:

Transfer the knowledge of a pre-trained generator (on a large and diverse dataset) to a target domain with limited data

- ➡ Initialize the generator
- ➡ Fine-tune G_T using
- ➡ General knowledge from source domain and the target domain-specific knowledge

Similar approach can be applied to Medical Image Generation conditioned on having a powerful source model trained on medical data!

main, and the target domain



Approaches: **Transfer Learning**

Major Limitation

Because of data constraints on target domain, the **general knowledge can be degraded** during fine-tuning generator on target data (acquiring the domain-specific knowledge)

Example. Adapting a generator trained on human faces (FFHQ) to painting of human faces (e.g., Fernand Léger):

- **Shared (general) knowledge:** face structure, diversity in pose, hairstyle, ...
- **Domain-specific knowledge:** the style of the painting with Fernand Léger

Result. Conventional transfer learning results in missing general knowledge: losing diversity and only containing the target style



Training
examples

Generated images by
adapting a pre-trained
StyleGAN on FFHQ to
Fernand Léger Painting

Approaches: **Transfer Learning**

Major Limitation

Because of data constraints on target domain, the **general knowledge can be degraded** during fine-tuning generator on target data (acquiring the domain-specific knowledge)

Example. Adapting a generator trained on human faces (FFHQ) to painting of human faces (e.g., Fernand Léger):

- **Shared (general) knowledge:** face structure, diversity in pose, hairstyle, ...
- **Domain-specific knowledge:** the style of the painting with Fernand Léger

Result. Conventional transfer learning results in missing general knowledge: losing diversity and only containing the target style



Training
examples

Generated images by
adapting a pre-trained
StyleGAN on FFHQ to
Fernand Léger Painting

Approaches: Transfer Learning

Major Limitation

Because of data constraints on target domain, the **general knowledge can be degraded** during fine-tuning generator on target data (acquiring the domain-specific knowledge)

Example. Adapting a generator trained on human faces (FFHQ) to painting of human faces (e.g., Fernand Léger):

- **Shared (general) knowledge:** face structure, diversity in pose, hairstyle, ...
- **Domain-specific knowledge:** the style of the painting with Fernand Léger

Result. Conventional transfer learning results in missing general knowledge: losing diversity and only containing the target style



Training
examples

Generated images by
adapting a pre-trained
StyleGAN on FFHQ to
Fernand Léger Painting

Approaches: Transfer Learning

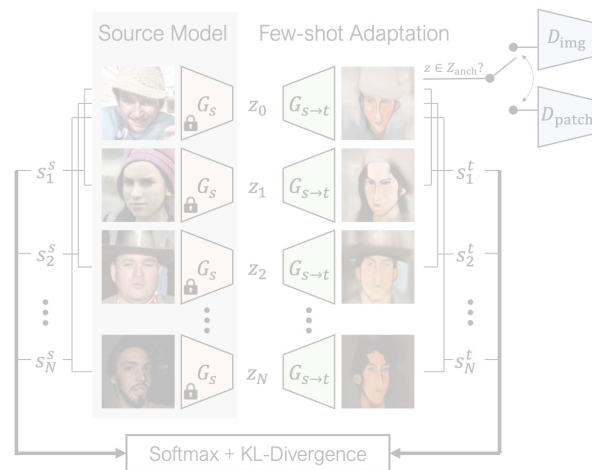
Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)

Key Observation. Overfitting in transfer learning, leads to the **loss of correspondence** between images generated by source and target domain



Objective. Add a regularizer to **preserve the correspondence between generated images before and after adapting the generator to target domain**



Approaches: Transfer Learning

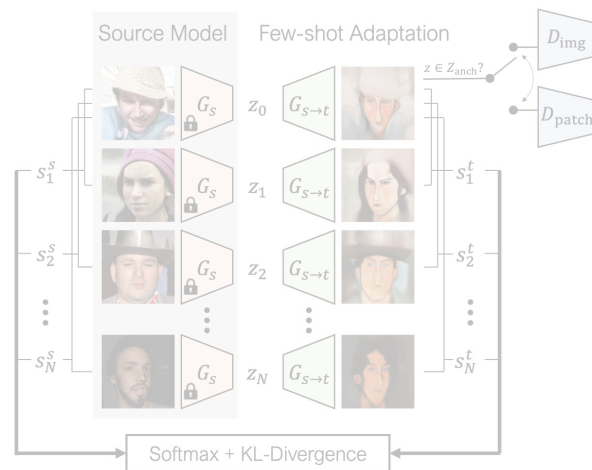
Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)

Key Observation. Overfitting in transfer learning, leads to the **loss of correspondence** between images generated by source and target domain

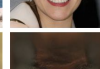


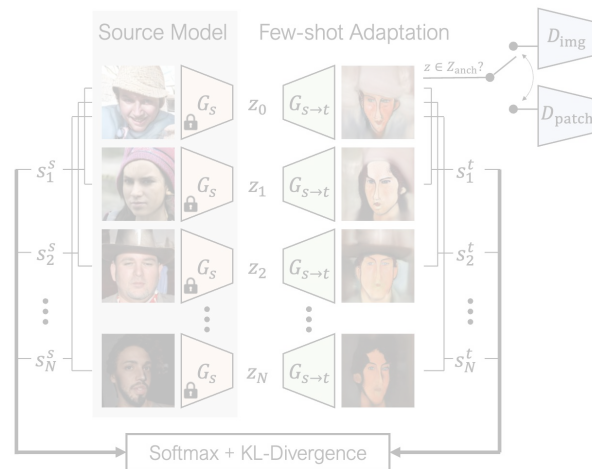
Objective. Add a regularizer to **preserve the correspondence between generated images before and after adapting the generator to target domain**



Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Key Observation. Overfitting in transfer learning, leads to the **loss of correspondence** between images generated by source and target domain

	z_0	z_1	z_2	z_3
Source G_s				
Overfit $G_{s \rightarrow t}$				
Our $G_{s \rightarrow t}$				



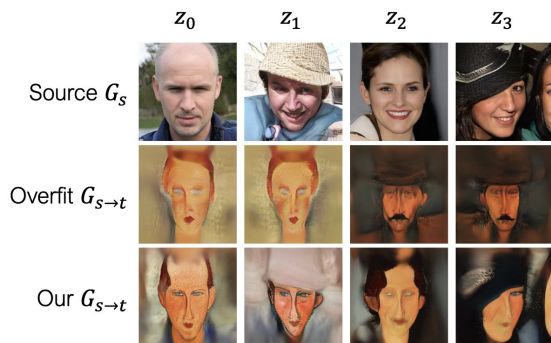
Ojha, Utkarsh, et al. "Few-shot image generation via cross-domain correspondence." CVPR'21.

Approaches: Transfer Learning

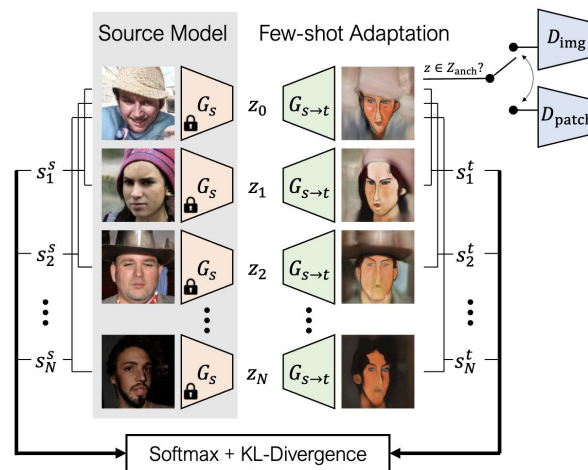
Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)

Key Observation. Overfitting in transfer learning, leads to the **loss of correspondence** between images generated by source and target domain



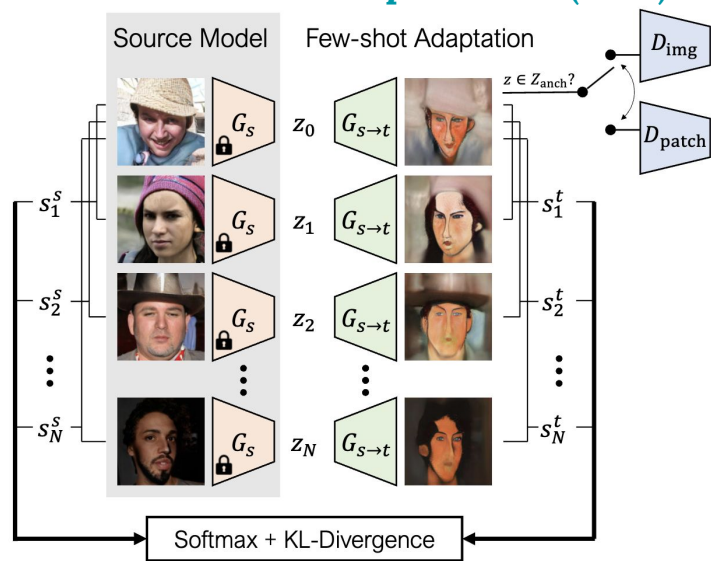
Objective. Add a regularizer to **preserve the correspondence between generated images before and after adapting the generator to target domain**



Approaches: Transfer Learning

Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)



Approach:

- Sample $N+1$ noise vectors
- Compute **pairwise similarity scores** s_i between images generated by these noise vectors
- Construct **N-way probability distribution** by applying softmax on similarity scores

$$y_i^{s,l} = \text{Softmax}(\{\text{sim}(G_s^l(z_i), G_s^l(z_j))\}_{\forall i \neq j})$$

$$y_i^{s \rightarrow t,l} = \text{Softmax}(\{\text{sim}(G_{s \rightarrow t}^l(z_i), G_{s \rightarrow t}^l(z_j))\}_{\forall i \neq j})$$

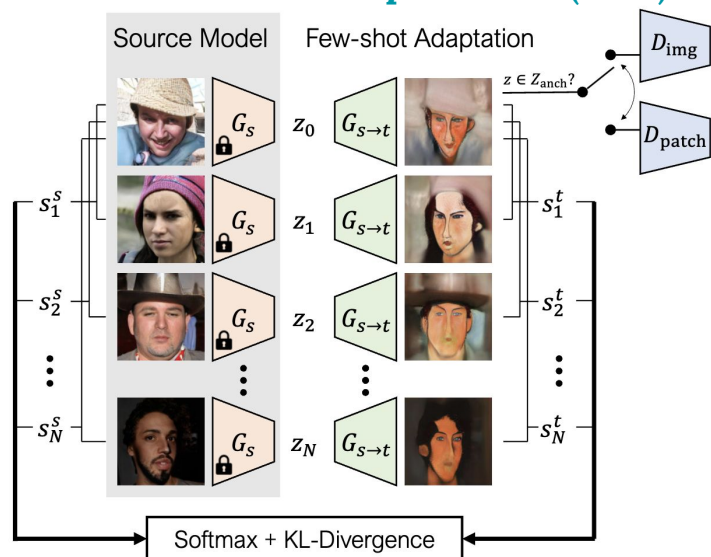
- Enforce adapted model to have similar distributions to the source

$$\mathcal{L}_{\text{dist}}(G_{s \rightarrow t}, G_s) = \mathbb{E}_{\{z_i \sim p_z(z)\}} \sum_{l,i} D_{KL}(y_i^{s \rightarrow t,l} || y_i^{s,l})$$

Approaches: Transfer Learning

Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)



Approach:

- Sample $N+1$ noise vectors
- Compute **pairwise similarity scores** s_i between images generated by these noise vectors
- Construct **N-way probability distribution** by applying softmax on similarity scores

$$y_i^{s,l} = \text{Softmax}(\{\text{sim}(G_s^l(z_i), G_s^l(z_j))\}_{\forall i \neq j})$$

$$y_i^{s \rightarrow t,l} = \text{Softmax}(\{\text{sim}(G_{s \rightarrow t}^l(z_i), G_{s \rightarrow t}^l(z_j))\}_{\forall i \neq j})$$

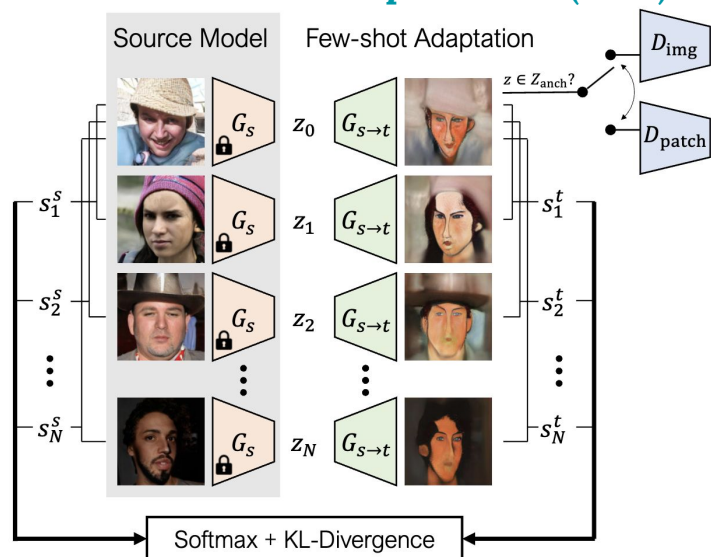
- Enforce adapted model to have similar distributions to the source

$$\mathcal{L}_{\text{dist}}(G_{s \rightarrow t}, G_s) = \mathbb{E}_{\{z_i \sim p_z(z)\}} \sum_{l,i} D_{KL}(y_i^{s \rightarrow t,l} || y_i^{s,l})$$

Approaches: Transfer Learning

Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)



Approach:

- Sample $N+1$ noise vectors
- Compute **pairwise similarity scores** s_i between images generated by these noise vectors
- Construct **N-way probability distribution** by applying softmax on similarity scores

$$y_i^{s,l} = \text{Softmax}(\{\text{sim}(G_s^l(z_i), G_s^l(z_j))\}_{\forall i \neq j})$$

$$y_i^{s \rightarrow t,l} = \text{Softmax}(\{\text{sim}(G_{s \rightarrow t}^l(z_i), G_{s \rightarrow t}^l(z_j))\}_{\forall i \neq j})$$

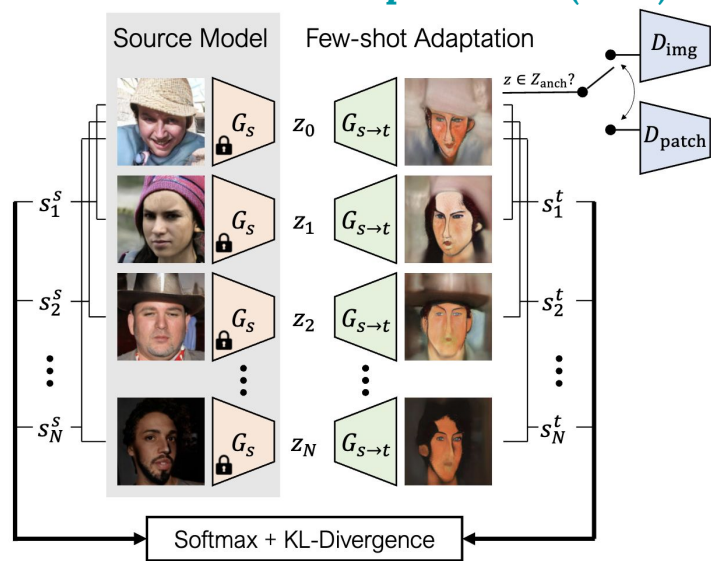
- Enforce adapted model to have similar distributions to the source

$$\mathcal{L}_{\text{dist}}(G_{s \rightarrow t}, G_s) = \mathbb{E}_{\{z_i \sim p_z(z)\}} \sum_{l,i} D_{KL}(y_i^{s \rightarrow t,l} || y_i^{s,l})$$

Approaches: Transfer Learning

Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)



Approach:

- Sample $N+1$ noise vectors
- Compute **pairwise similarity scores** s_i between images generated by these noise vectors
- Construct **N-way probability distribution** by applying softmax on similarity scores

$$y_i^{s,l} = \text{Softmax}(\{\text{sim}(G_s^l(z_i), G_s^l(z_j))\}_{\forall i \neq j})$$

$$y_i^{s \rightarrow t,l} = \text{Softmax}(\{\text{sim}(G_{s \rightarrow t}^l(z_i), G_{s \rightarrow t}^l(z_j))\}_{\forall i \neq j})$$

- Enforce adapted model to have similar distributions to the source

$$\mathcal{L}_{\text{dist}}(G_{s \rightarrow t}, G_s) = \mathbb{E}_{\{z_i \sim p_z(z)\}} \sum_{l,i} D_{KL}(y_i^{s \rightarrow t,l} || y_i^{s,l})$$

Approaches: Transfer Learning

Regularizer-based Fine-Tuning: These approaches add a regularizer term to the main objective to preserve the knowledge in the pre-trained generator

Cross-Domain Correspondence (CDC)

Qualitative Results.

10-shot adaptation
(FFHQ $\Rightarrow$ Sketches)



Approaches

Approaches for Image Generation with Limited Data (Training Generative Models with Limited Data):

1. **Transfer Learning**
2. **Data Augmentation**
3. **Network Architecture**

Approaches: Data Augmentation

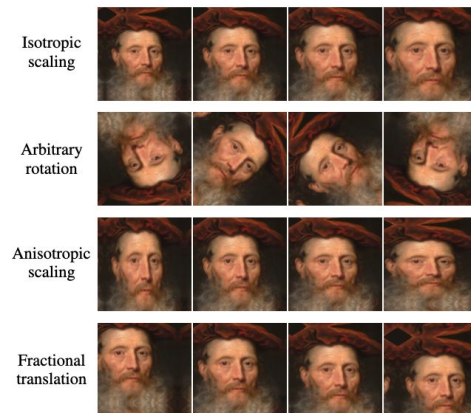
Data Augmentation in Generative Modeling

- Data augmentation aims to **increase the quantity and diversity of the training data** by applying some transformations in real data, e.g., adding noise, rotating images, ...
- Increased quantity can **prevent overfitting** of the generative model

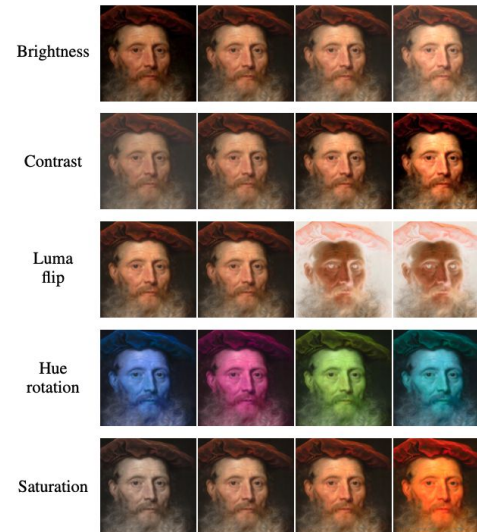
Pixel blitting



General geometric transformations



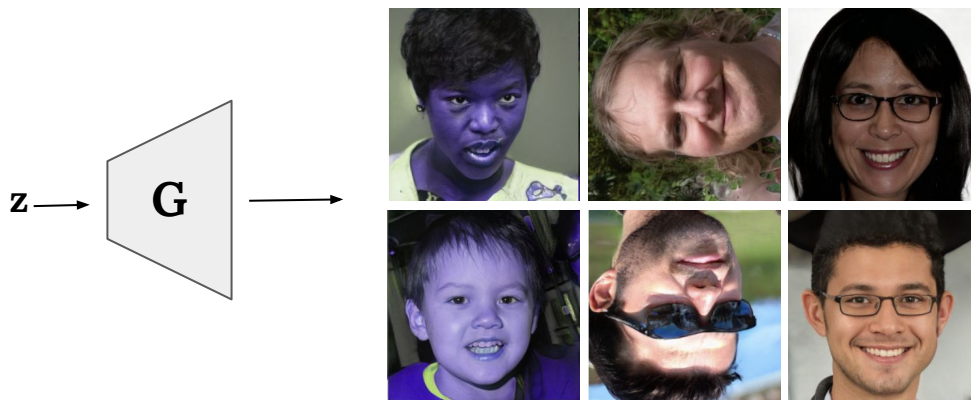
Color transformations



Approaches: Data Augmentation

Major Limitation of Classical Data Augmentation in Generative Learning

Generator **learns the augmented data distribution** instead of the real distribution and generate image with same transformations



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

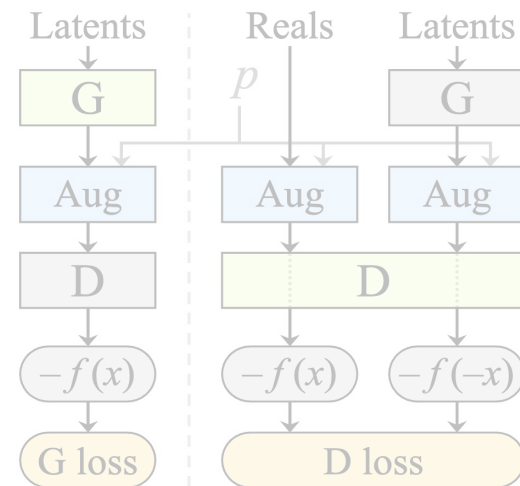
Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Approach. ADA includes following components:

- Augmentation is applied to **both real and fake images** (in training both D and G)
- The augmentation is applied **with a probability $p < 1$** to enable the occurrence of the real distribution
- The **strength of the augmentation (p)** is adjusted based on the **degree of overfitting**
- Two heuristics are proposed to monitor the overfitting

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

$r=0$ means no overfitting, and $r=1$ indicates complete overfitting



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

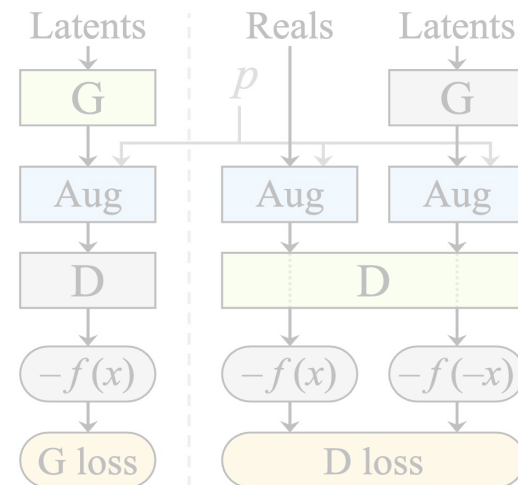
Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Approach. ADA includes following components:

- Augmentation is applied to **both real and fake images** (in training both D and G)
- The augmentation is applied **with a probability $p < 1$** to enable the occurrence of the real distribution
- The **strength of the augmentation (p)** is adjusted based on the **degree of overfitting**
- Two heuristics are proposed to monitor the overfitting

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

$r=0$ means no overfitting, and $r=1$ indicates complete overfitting



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

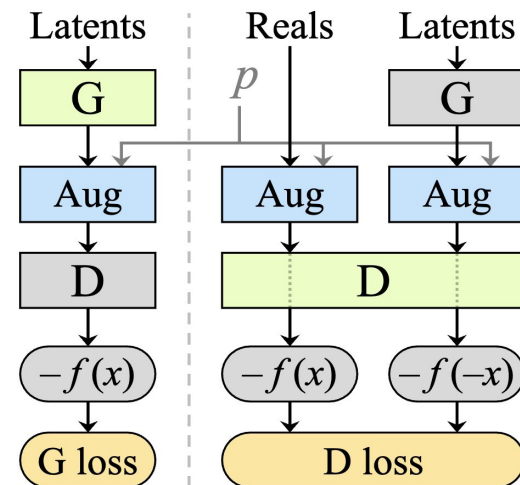
Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Approach. ADA includes following components:

- Augmentation is applied to **both real and fake images** (in training both D and G)
- The augmentation is applied **with a probability $p < 1$** to enable the occurrence of the real distribution
- The **strength of the augmentation (p)** is adjusted based on the **degree of overfitting**
- Two heuristics are proposed to monitor the overfitting

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

$r=0$ means no overfitting, and $r=1$ indicates complete overfitting



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

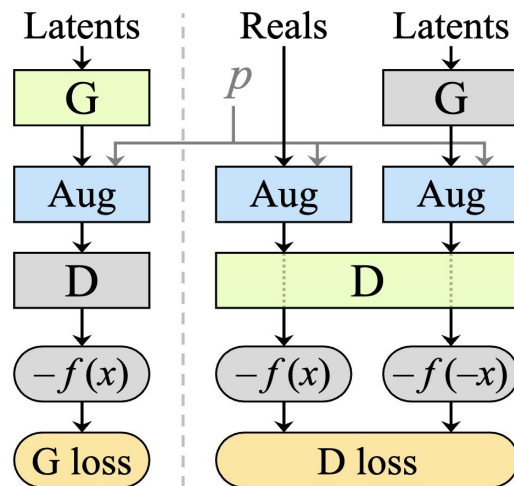
Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Approach. ADA includes following components:

- Augmentation is applied to **both real and fake images** (in training both D and G)
- The augmentation is applied **with a probability $p < 1$** to enable the occurrence of the real distribution
- The **strength of the augmentation (p)** is adjusted based on the **degree of overfitting**
- Two heuristics are proposed to monitor the overfitting

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

$r=0$ means no overfitting, and $r=1$ indicates complete overfitting



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

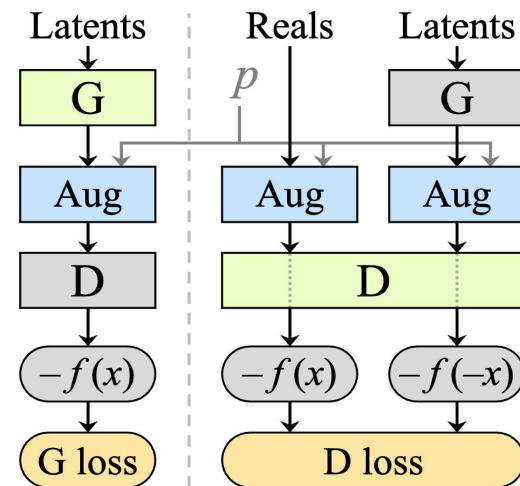
Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Approach. ADA includes following components:

- Augmentation is applied to **both real and fake images** (in training both D and G)
- The augmentation is applied **with a probability $p < 1$** to enable the occurrence of the real distribution
- The **strength of the augmentation (p)** is adjusted based on the **degree of overfitting**
- Two heuristics are proposed to monitor the overfitting

$$r_v = \frac{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{validation}}]}{\mathbb{E}[D_{\text{train}}] - \mathbb{E}[D_{\text{generated}}]} \quad r_t = \mathbb{E}[\text{sign}(D_{\text{train}})]$$

$r=0$ means no overfitting, and $r=1$ indicates complete overfitting



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Results

MetFaces dataset (1336 images) of art paintings



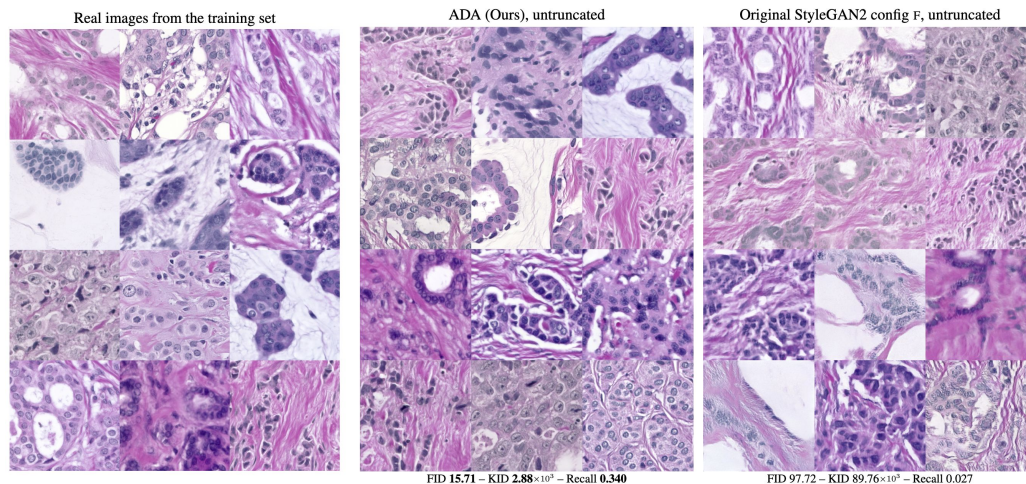
Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

Adaptive Data Augmentation (ADA): apply augmentation to the both real and fake images with an adjustable probability based on the training dynamics

Results

BreCaHAD dataset (1944 images) for breast cancer annotation



Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

Adaptive Data Augmentation (ADA)

Evaluating the Performance of StyleGAN2-ADA on Medical Images

McKell Woodland^{1,2}, John Wood¹, Brian M. Anderson^{1,4}, Suprateek Kundu¹
Ethan Lin¹, Eugene Koay¹, Bruno Odisio¹, Caroline Chung¹, Hyunseon
Christine Kang¹, Aradhana M. Venkatesan¹, Sireesha Yedururi¹, Brian De¹,
Yuan-Mao Lin¹, Ankit B. Patel^{2,3}, and Kristy K. Brock¹

¹ The University of Texas MD Anderson Cancer Center, Houston TX 77030, USA
MEWoodland@mdanderson.org

² Rice University, Houston TX 77005, USA

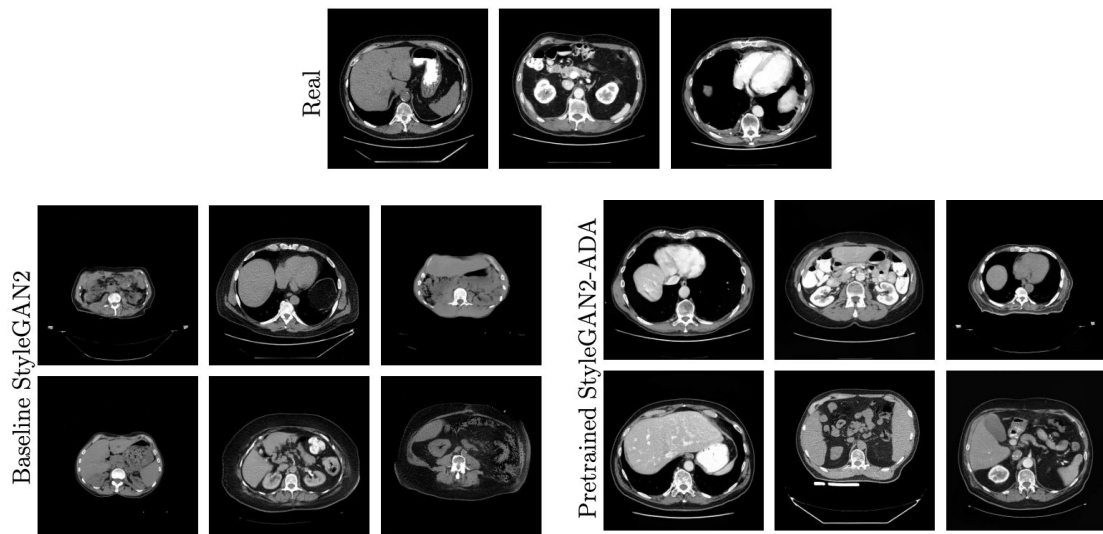
³ Baylor College of Medicine, Houston TX 77030, USA

⁴ University of California San Diego, La Jolla CA 92093, USA

Approaches: Data Augmentation

Image-Level Augmentation: Apply data transformation on image space.

Adaptive Data Augmentation (ADA)



Start from a pre-trained StyleGAN2-ADA on FFHQ and fine-tune on the SLIVER07 dataset

Approaches

Approaches for Image Generation with Limited Data (Training Generative Models with Limited Data):

1. Transfer Learning
2. Data Augmentation
3. **Network Architecture**

Comprehensive Review: **Network Architecture**

Network Architecture

Design specific architectures for the generators to improve their training performance under data constraints. Like designing shallow/sparse architectures to **prevent over-parameterization**.

Primary Challenge/Limitation

- When aiming to design a new architecture, the process of discovering optimal hyperparameters can be laborious
- Designing new architecture prevents leveraging the powerful pre-trained generators

Comprehensive Review: **Network Architecture**

Network Architecture

Design specific architectures for the generators to improve their training performance under data constraints. Like designing shallow/sparse architectures to **prevent over-parameterization**.

Primary Challenge/Limitation

- When aiming to design a new architecture, the process of discovering optimal hyperparameters can be laborious
- Designing new architecture prevents leveraging the powerful pre-trained generators

Comprehensive Review: **Network Architecture**

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN has **three major design** choices:

- i) Using a **compact size network** for both G and D in GAN
- ii) introducing **Skip-layer excitation** for G for better gradient flow
- iii) adding **self-supervised** task for D



Results of training FastGAN **from scratch** on **1024² resolution** using **single RTX 2080-Ti GPU** with only 1000 images. Left: **20 hours** on Nature photos; Right: **10 hours** on FFHQ.

Comprehensive Review: **Network Architecture**

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN has **three major design** choices:

- i) Using a **compact size network** for both G and D in GAN
- ii) introducing **Skip-layer excitation** for G for better gradient flow
- iii) adding **self-supervised** task for D



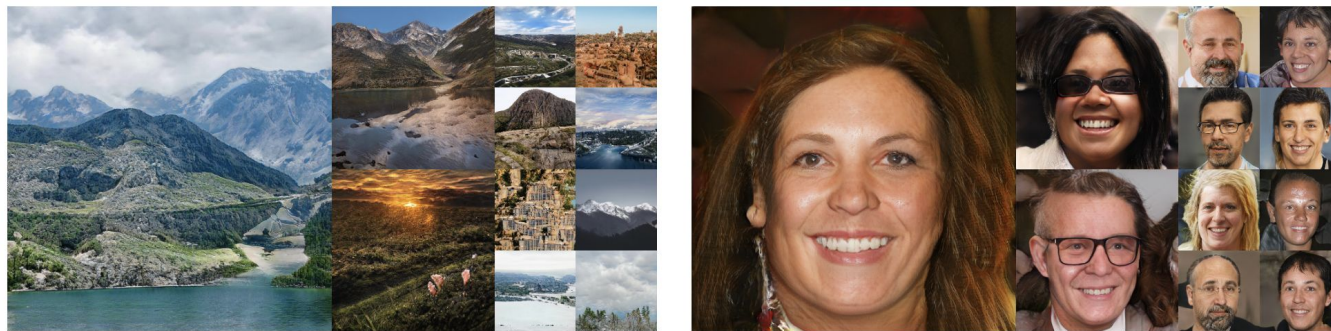
Results of training FastGAN **from scratch** on **1024² resolution** using **single RTX 2080-Ti GPU** with only 1000 images. Left: **20 hours** on Nature photos; Right: **10 hours** on FFHQ.

Comprehensive Review: Network Architecture

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN has **three major design** choices:

- i) Using a **compact size network** for both G and D in GAN
- ii) introducing **Skip-layer excitation** for G for better gradient flow
- iii) adding **self-supervised** task for D



Results of training FastGAN **from scratch** on 1024^2 resolution using **single RTX 2080-Ti GPU** with only 1000 images. Left: **20 hours** on Nature photos; Right: **10 hours** on FFHQ.

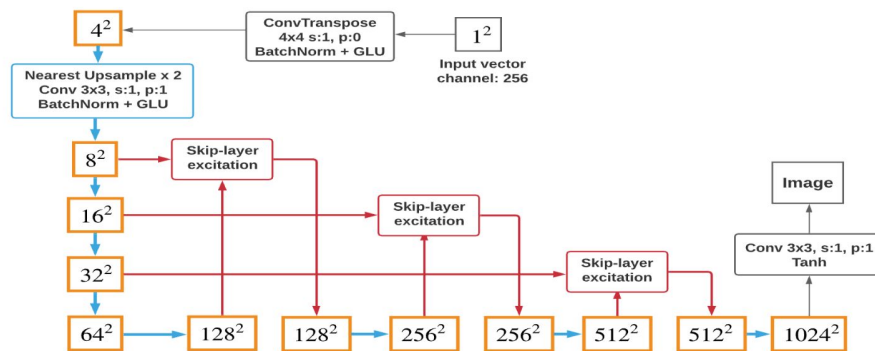
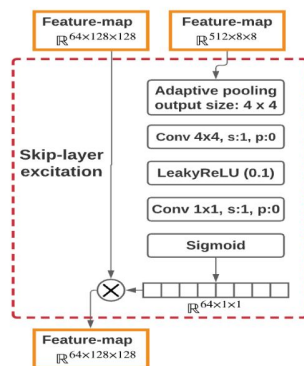
Comprehensive Review: Network Architecture

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN

Design of Generator (G):

- Use a **single Conv-layer** for each resolution
- Use **skip-layer excitation** including skip connection and connection between different resolutions for **better gradient flow during training G**



The forward flow of our Generator

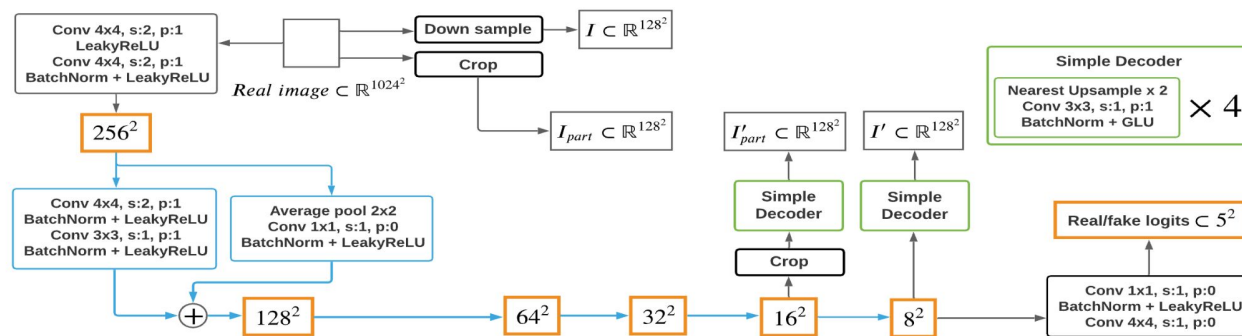
Comprehensive Review: Network Architecture

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN

Design of Discriminator (D):

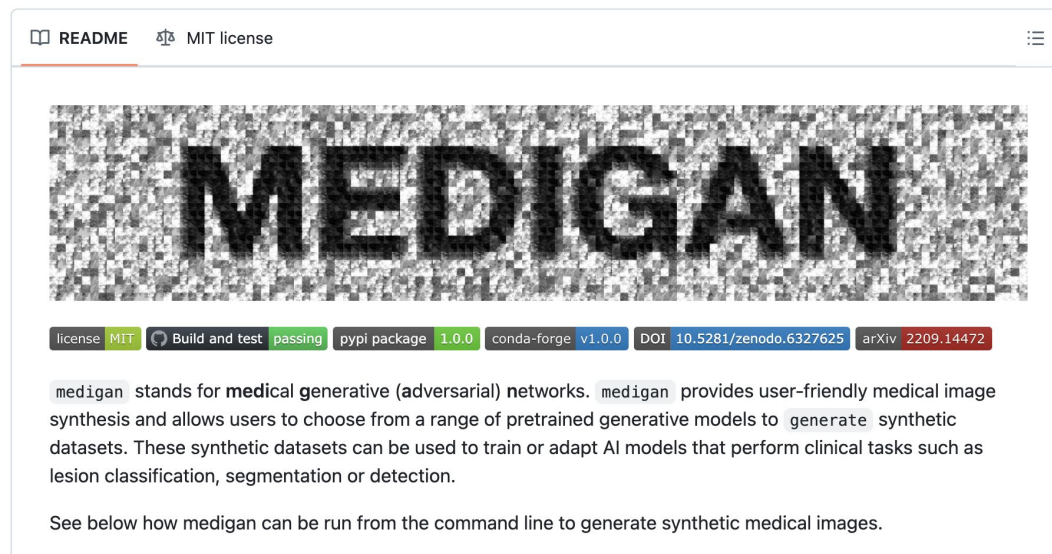
- Use a **compact design** for D
- Add reconstruction loss in two different resolution (additional task as auto-encoder) to improve learning of D using **additional supervisory signal**



Comprehensive Review: Network Architecture

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

FastGAN



<https://github.com/RichardObi/medigan>

Feature Enhancement: Design additional modules to enhance/retain the features of the generator

Output type	Modality	Model type	Output size	Base dataset	Output examples	model_id
Polyp with Mask	endoscopy	fastgan	256x256	HyperKvasir		00010_FASTGAN_POLYP_PATCHES_W_MASKS

