IMAG/e

TU/e Technische Universiteit Eindhoven University of Technology

Generative Models part II

Cian M. Scannell, PhD
Assistant Professor
Medical Image Analysis
c.m.scannell@tue.nl

Topics:

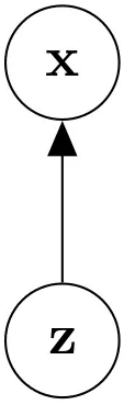
- Addressing VAE limitations
- Normalising flows
- GANs
- Conditional generation
- Image-to-image translation
- Limitations and extensions to GANs
- Fréchet inception distance

Learning objectives

The student can:

- Understand the benefits of normalising flows in the context of the limitations of VAEs
- Describe the origin of the name “normalising flows”
- Summarise the normalising flow algorithm
- Discuss the intuition behind generative adversarial networks (GANs)
- Formulate the GAN training process and loss function
- Classify applications of GANs
- Give examples of issues with GANs
- Motivate solutions to these issues

Recap VAEs



- A VAE is a latent variable model
  - Sample $\mathrm{z} \sim p(\mathrm{z})$
  - Generate $\mathrm{x} \sim p(\mathrm{x}|\mathrm{z})$

- A VAE is an *explicit* latent variable model
  - It tries to explicitly compute (approximate) the density $p(\mathrm{x})$
  - But we need to do a lot of work to get around the lack of an analytical solution and the curse of dimensionality

Limitation:

- Intractable likelihood

Normalising flows

Motivation:

- Can we design a latent variable model with a tractable likelihood?
- Ideally:
  - Easy to evaluate
  - Easy to sample from

  (like a Gaussian)

Normalising flows - main idea:

- Learn to map simple distributions to complex ones with an invertible mapping
- In particular, map Z to X with a deterministic, invertible function f_θ such that $X = f_\theta(Z)$ and $Z = f_\theta^{-1}(X)$

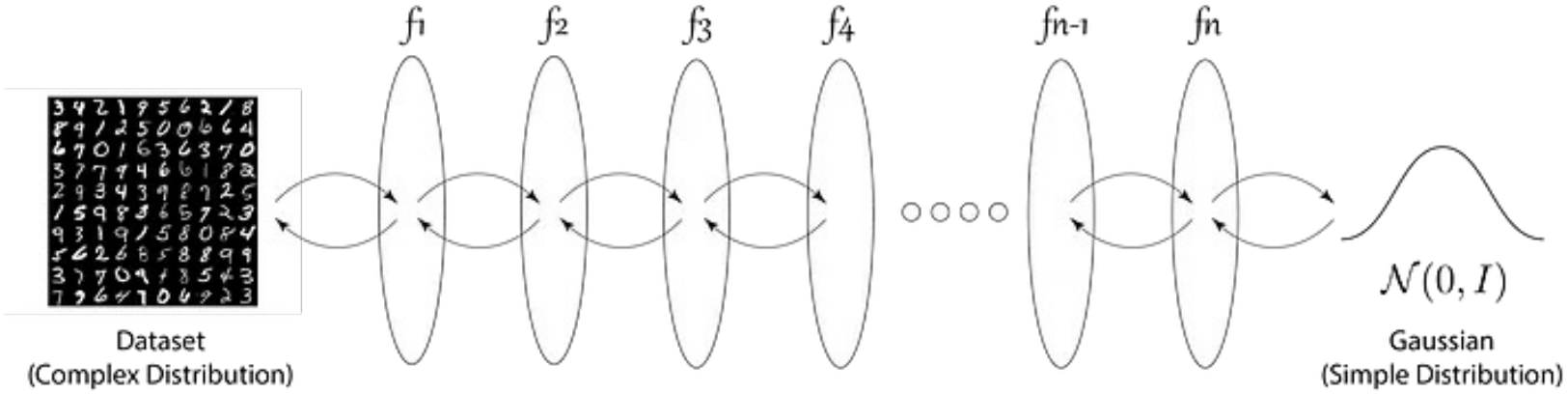
Normalising flows:

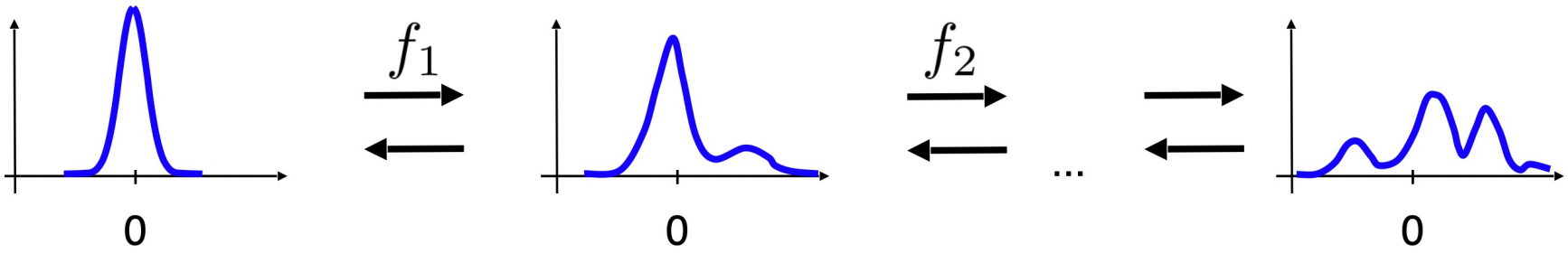
- If $z \sim p(z)$ and $f: Z \rightarrow X$
- Then $x \sim p(x) = p(z = f^{-1}(x)) \left| det(\frac{\partial f^{-1}(x)}{\partial x}) \right|$

Normalising

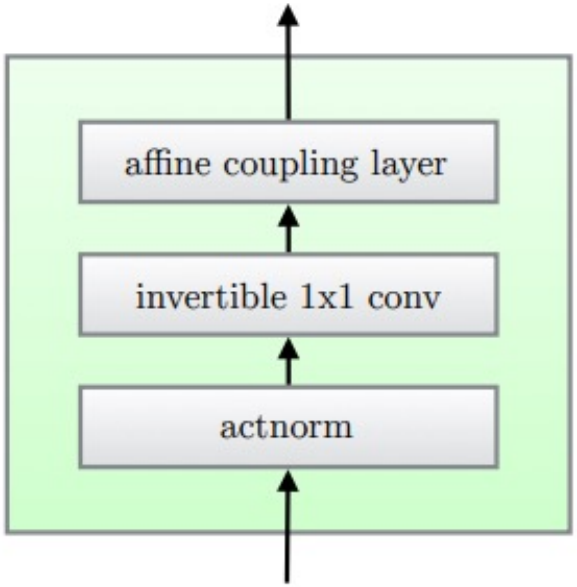
- To model complex high dimensional distributions, a sequence of these invertible transformations is applied.

Flows



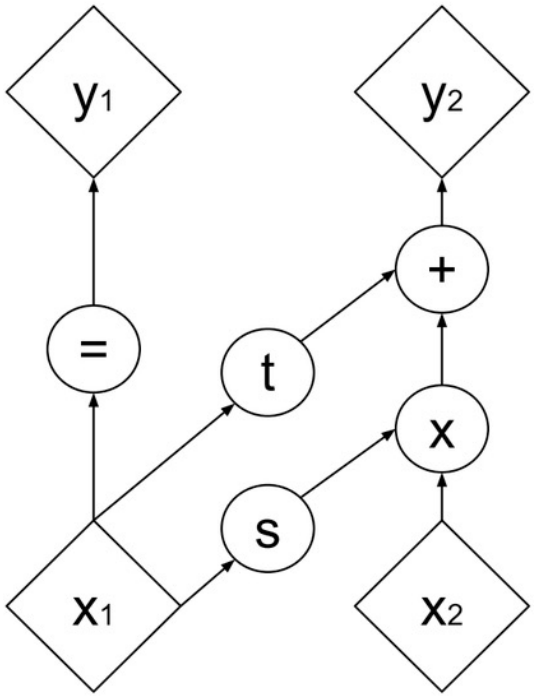

f_1
f_2
...
0
0
0
“latent” space
pixel space

Glow:



- **ActNorm**: Normalization layer similar to batch norm, except that the mean and standard deviation statistics are trainable parameters rather than estimated from the data (this is in particular useful here because the model sometimes has to be trained with very small batch sizes due to memory requirements)
- **Conv 1x1**: An invertible 1x1 convolution layer.
- **Affine Coupling Layer**: A coupling layer which splits the input data along channel dimensions, using the first half to estimate parameters of a transformation then applied to the second half.

Coupling layer:



(a) Forward propagation

Forward

$$x_a, x_b = \text{split}(x)$$
$$(\log \sigma, \mu) = \text{NN}(x_b)$$
$$y_a = \sigma \odot x_a + \mu$$
$$y = \text{concat}(y_a, x_b)$$

Generative adversarial networks (GANs)

Previously:

- We used the log-likelihood to describe the differences between generative samples and the training distribution during training.
- As well as the practical difficulties, it is unclear if better likelihood numbers correspond to better samples.

Now:

- Is there another way?
- I.e. instead of using the log-likelihood, is there another way to calculated the differences between real and generate data?
- Can we learn it?

Generative adversarial networks (GANs)

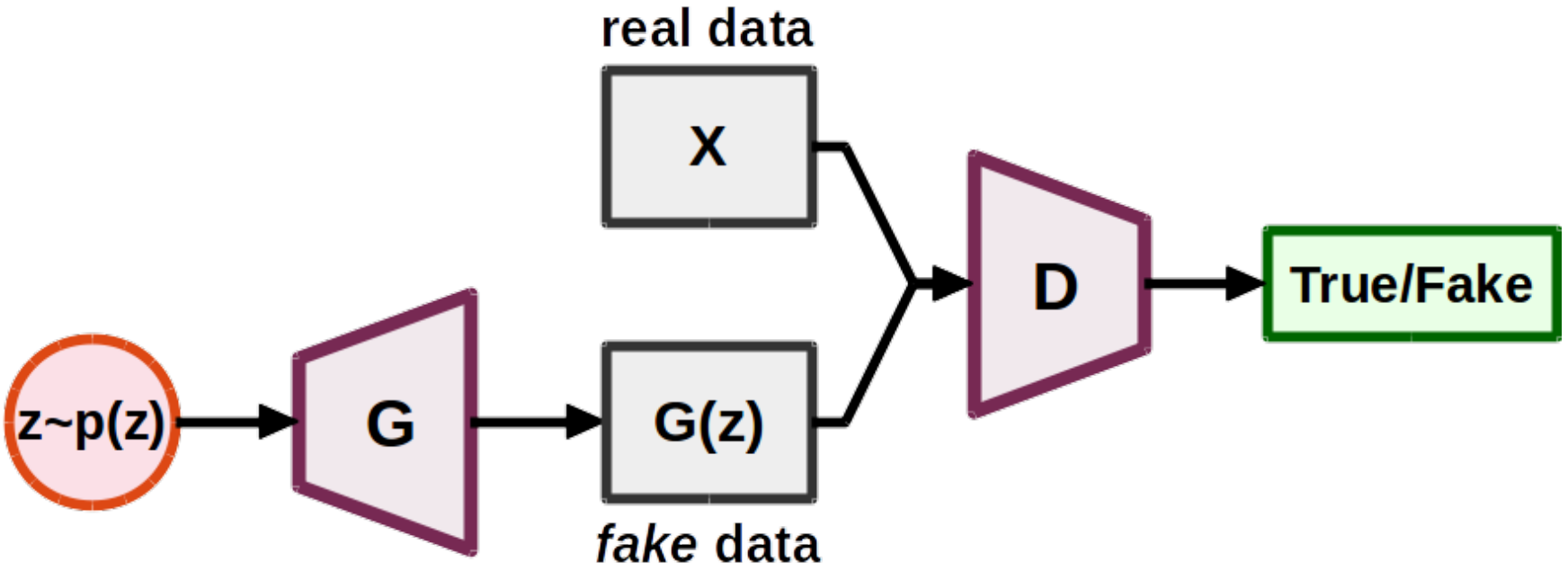
Main idea:

- Train a classifier to distinguish samples as either real or generated
- Train the generator to create realistic samples that fool the classifier

GAN intuition:

- Consider an art forger trying to create fake paintings of Leonardo da Vinci and a detective who is trying to identify if the painting is a real or a fake painting.
- Initially the forger may not be great at creating replicas of da Vinci's paintings and it might be easier for the detective to spot the fake paintings.
- With time, the forger learns the key components that the detective checks in the painting to classify it as a real painting or a fake painting. Similarly, as the detective sees more of the real and the fake paintings, the detective is able to learn the finer details required to differentiate between the two.
- Eventually the forger will learn the trick so well that the forger will start fooling the detective into believing that the fake paintings are the original paintings of da Vinci. Now we can say that the training for the forger is complete.

GAN:



Algorithm:

- Generator - $G_\beta: Z \rightarrow X$.
- Discriminator - $D_\alpha: X \rightarrow [0, 1]$.
- Discriminator tries to maximize $\log D_\alpha(x) + \log 1 - D_\alpha\big(G_\beta(z)\big)$ with respect to α.
- Generator tries to minimise it with respect to β.

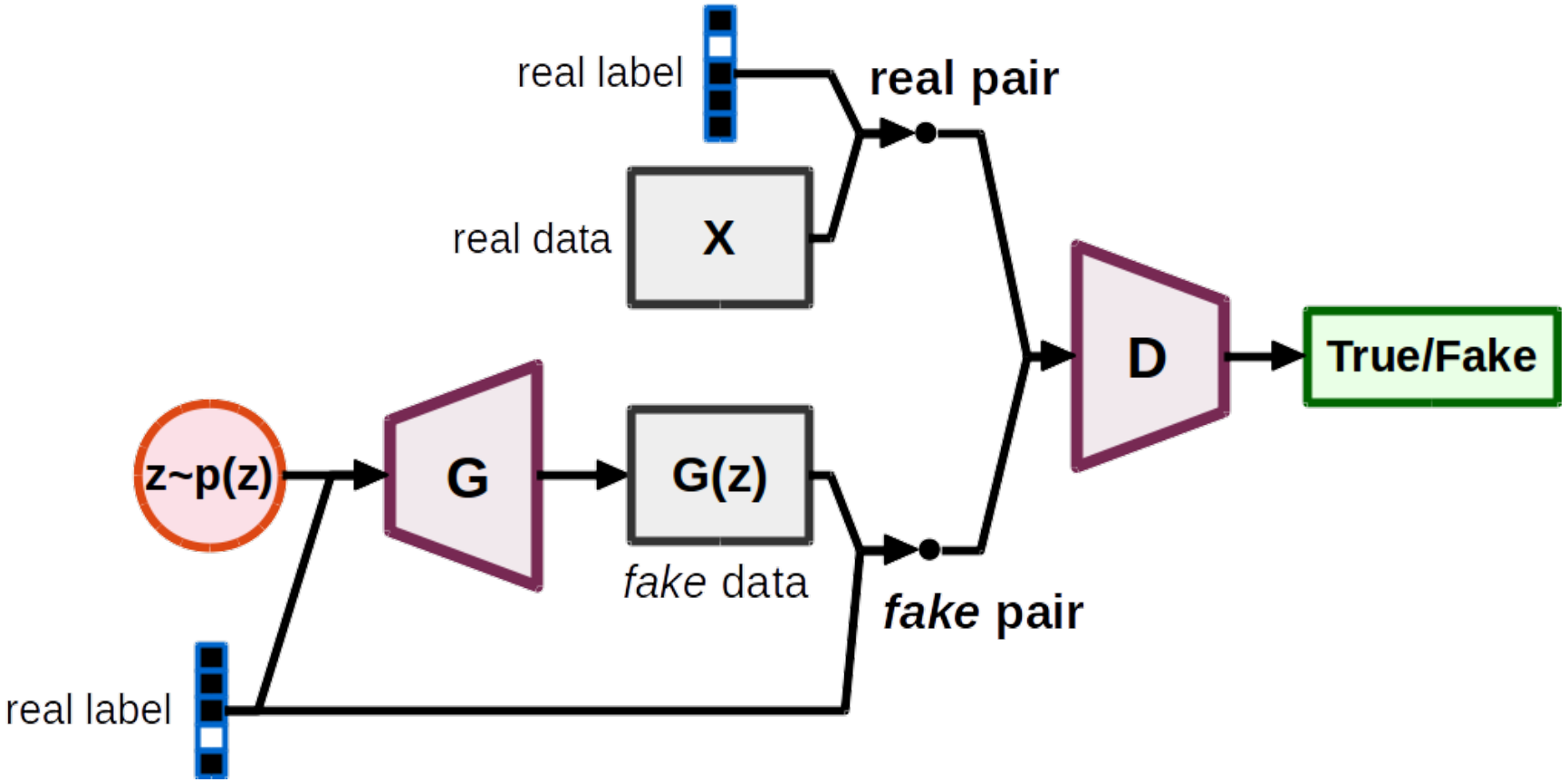
$$\min_\beta \max_\alpha \mathbb{E}_{x \sim p_{real}} [\log D_\alpha(x)] + \mathbb{E}_{z \sim p(z)} [\log 1 - D_\alpha\big(G_\beta(z)\big)]$$

- This is known as the adversarial loss as there are two parts trying to achieve two opposite goals.

Challenges:

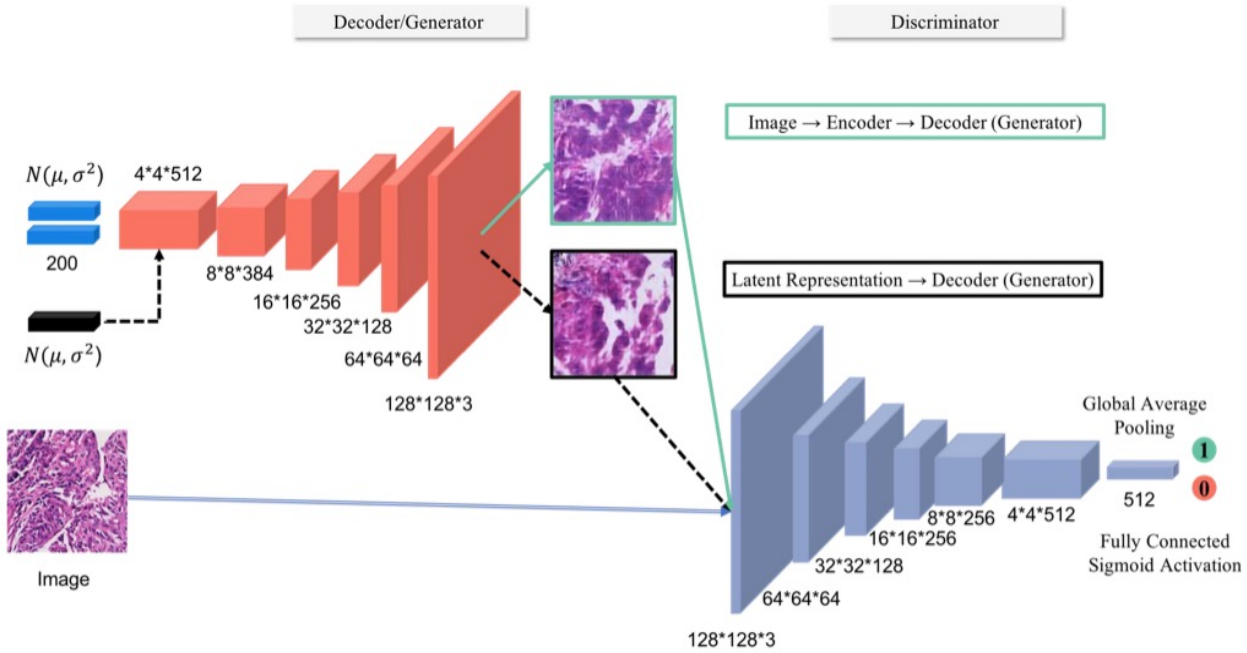
- Unstable optimization procedure
  - Generator and discriminator loss often oscillate without converging.
- Potential for mode collapse
  - Generator beats discriminator but only predicts same (or limited set of) image(s) every time
- Depends heavily on choice of hyperparameters
- Difficult to evaluate
  - When is an image realistic?

Conditional GAN (cGAN):

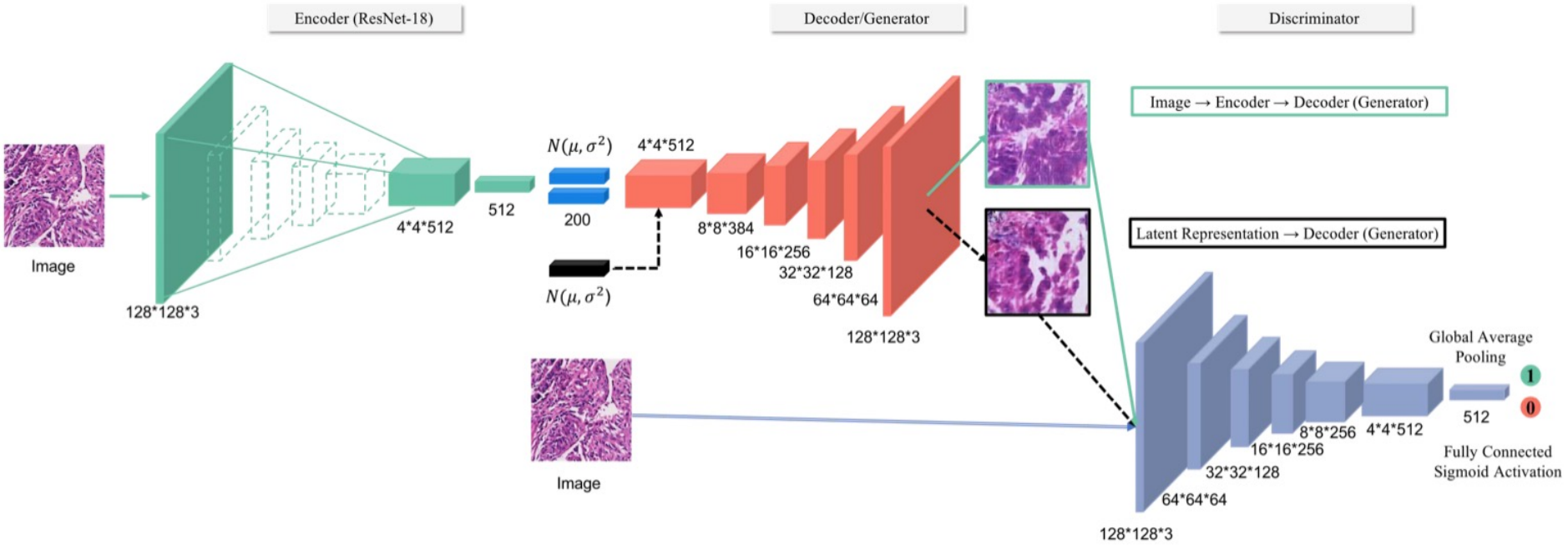


$D_\alpha(x|y)$ and $G_\beta(z|y)$ instead of $D_\alpha(x)$ and $G_\beta(z)$.

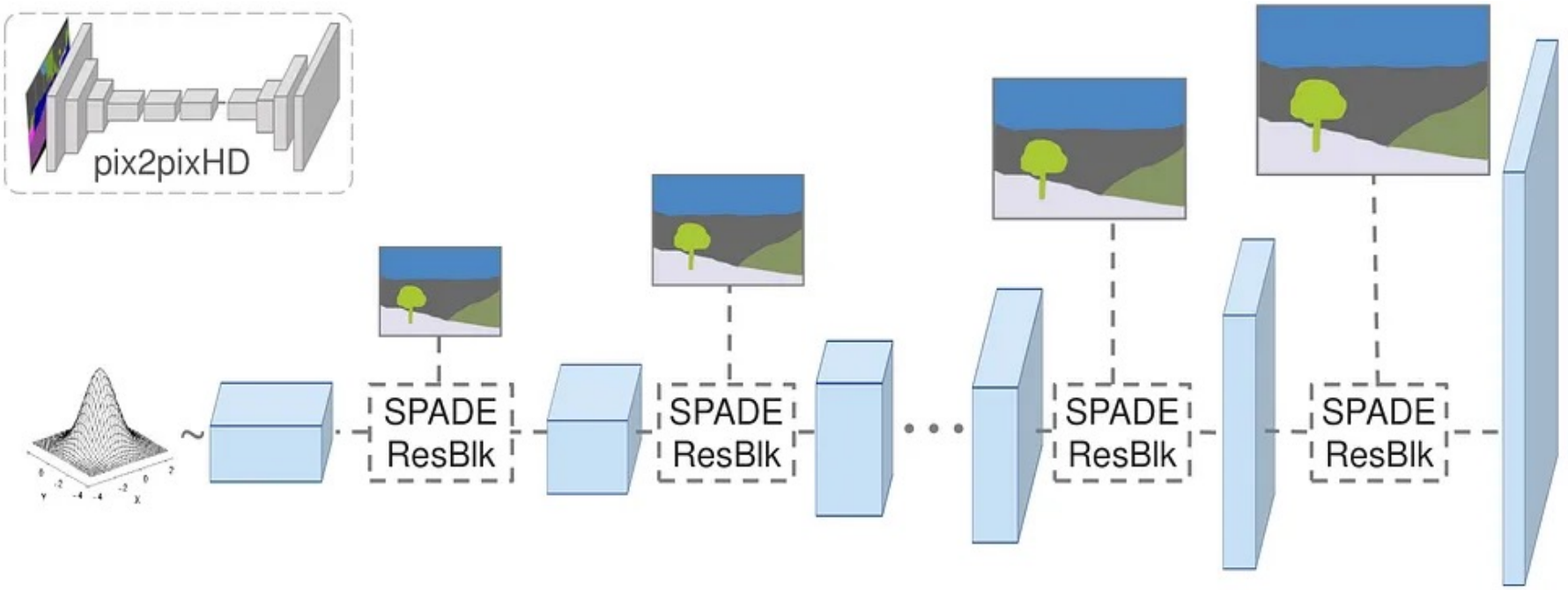
Hybrid models – VAE-GAN:



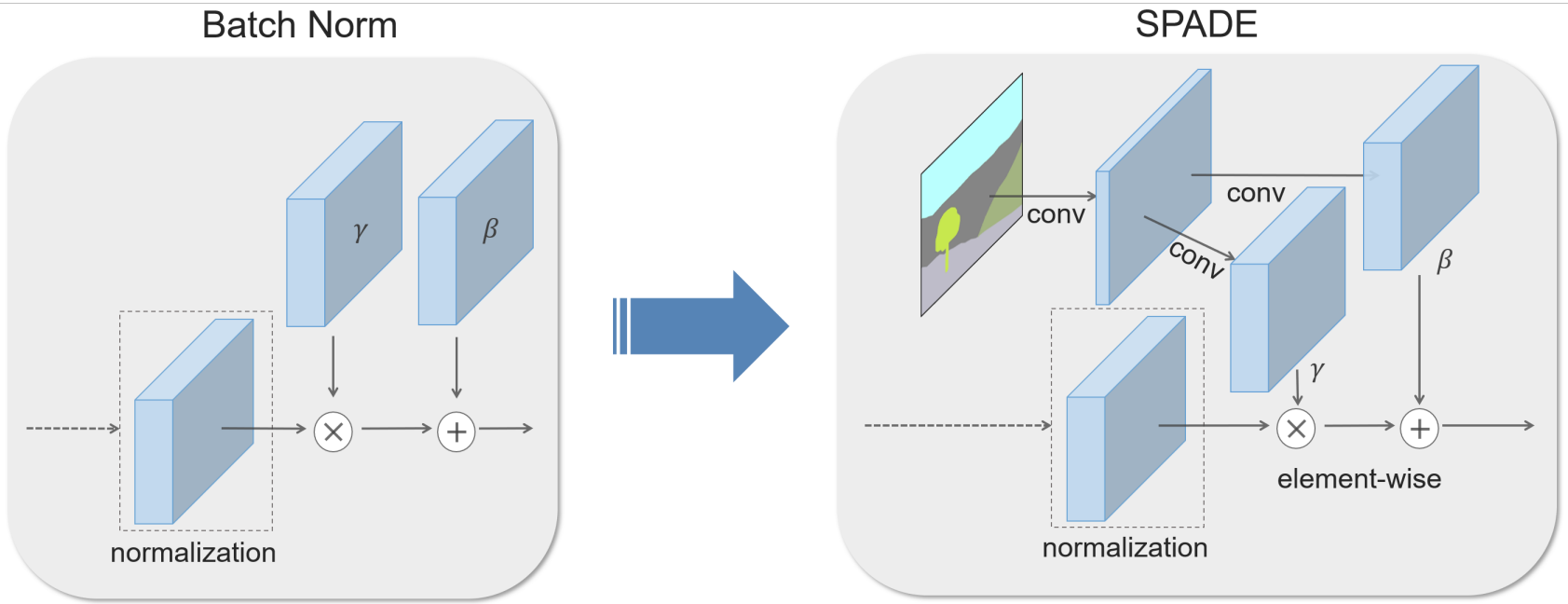
Hybrid models – VAE-GAN:



SPADE:


pix2pixHD
SPADE ResBlk
SPADE ResBlk
SPADE ResBlk
SPADE ResBlk

SPADE:


Batch Norm
SPADE
γ
β
normalization
conv
conv
conv
β
γ
element-wise
normalization

BiGAN (bidirectional GAN):

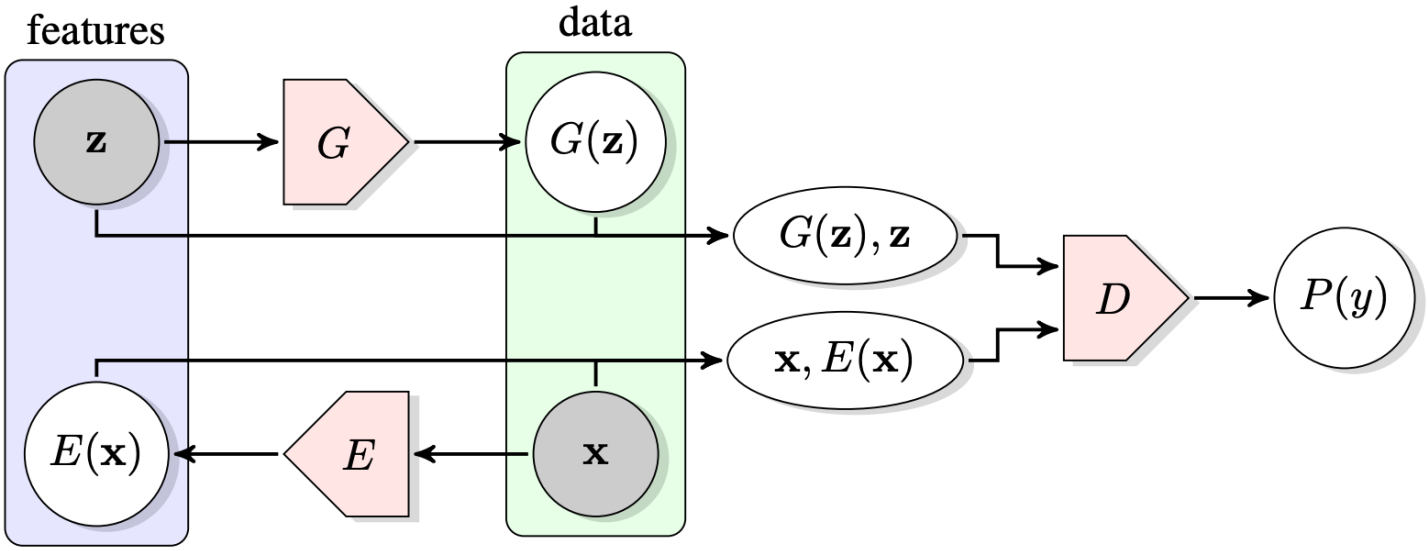
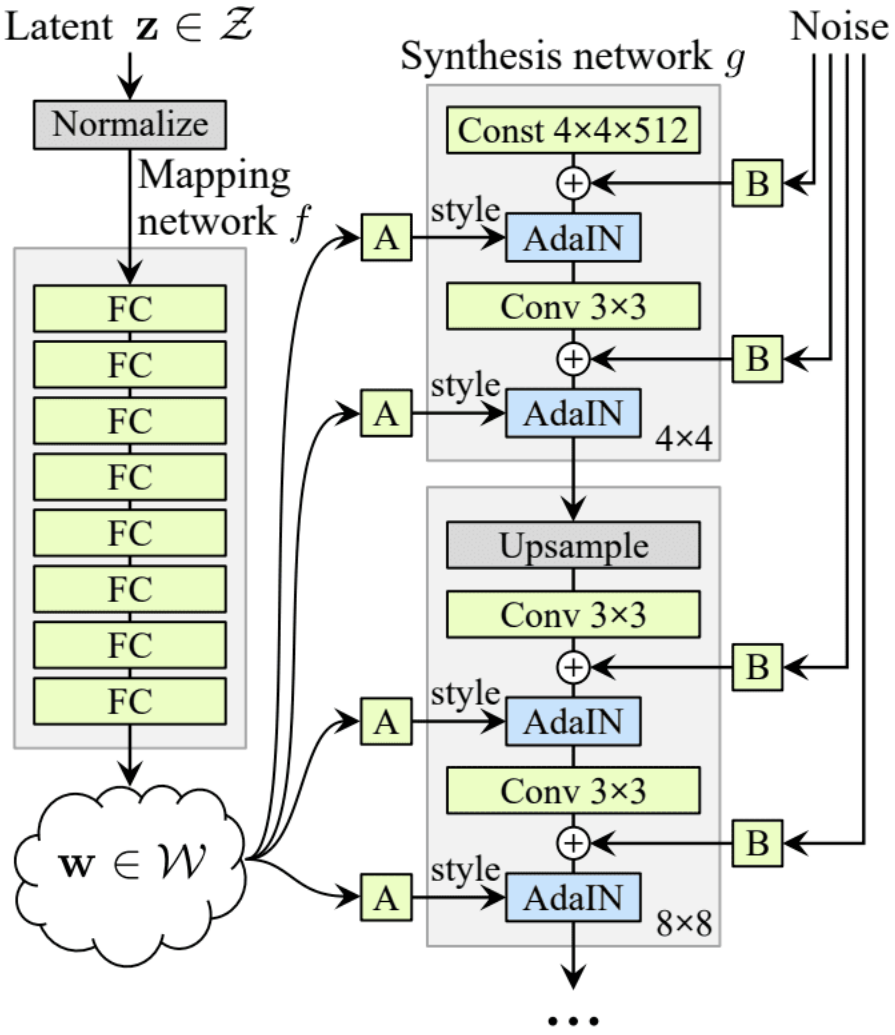


Figure 1: The structure of Bidirectional Generative Adversarial Networks (BiGAN).

Motivation: enforce structure on latent representations

StyleGAN:

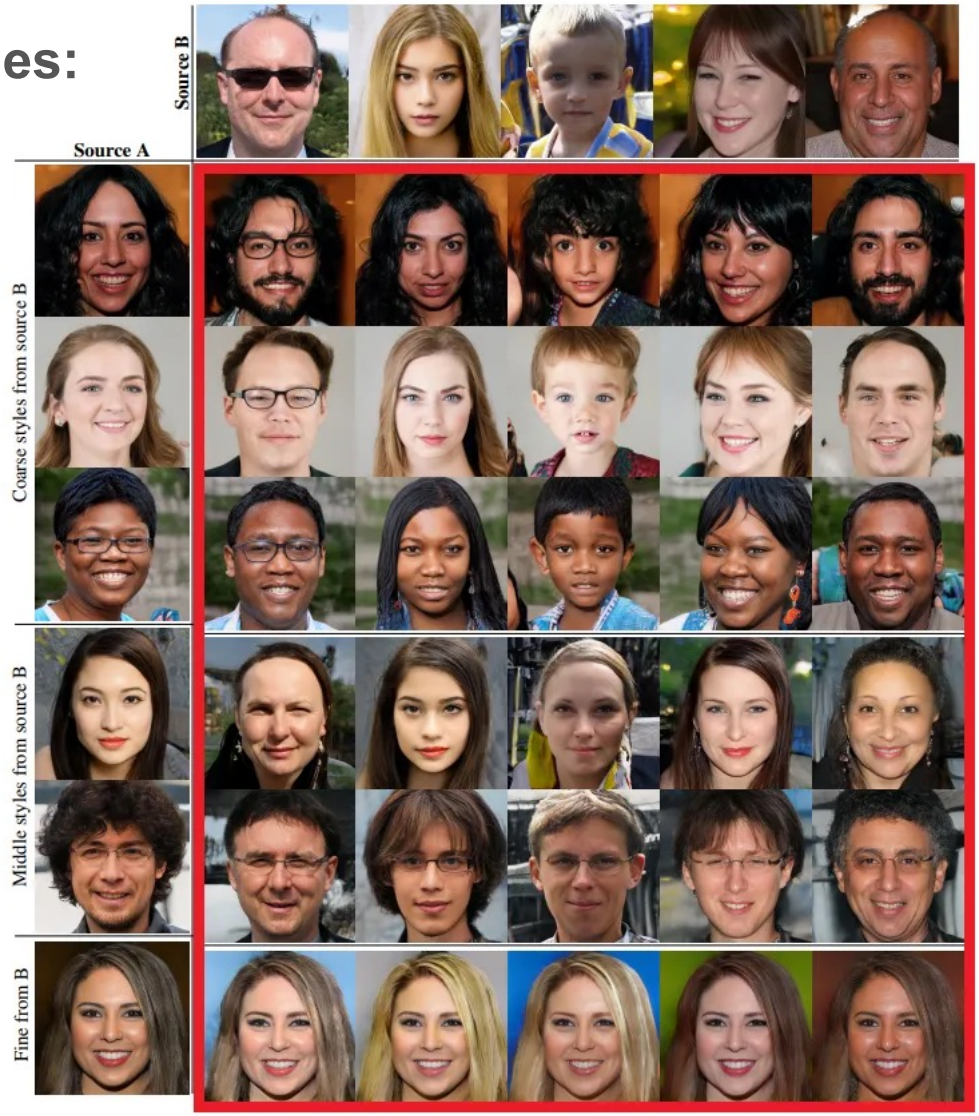

Latent $\mathbf{z} \in \mathcal{Z}$
Normalize
Mapping network f
FC
FC
FC
FC
FC
FC
FC
FC
$\mathbf{w} \in \mathcal{W}$
Synthesis network g
Noise
Const 4×4×512
B
A
style
AdaIN
Conv 3×3
B
A
style
AdaIN
4×4
Upsample
Conv 3×3
B
A
style
AdaIN
Conv 3×3
B
A
style
AdaIN
8×8
...

Adaptive Instance Normalization:

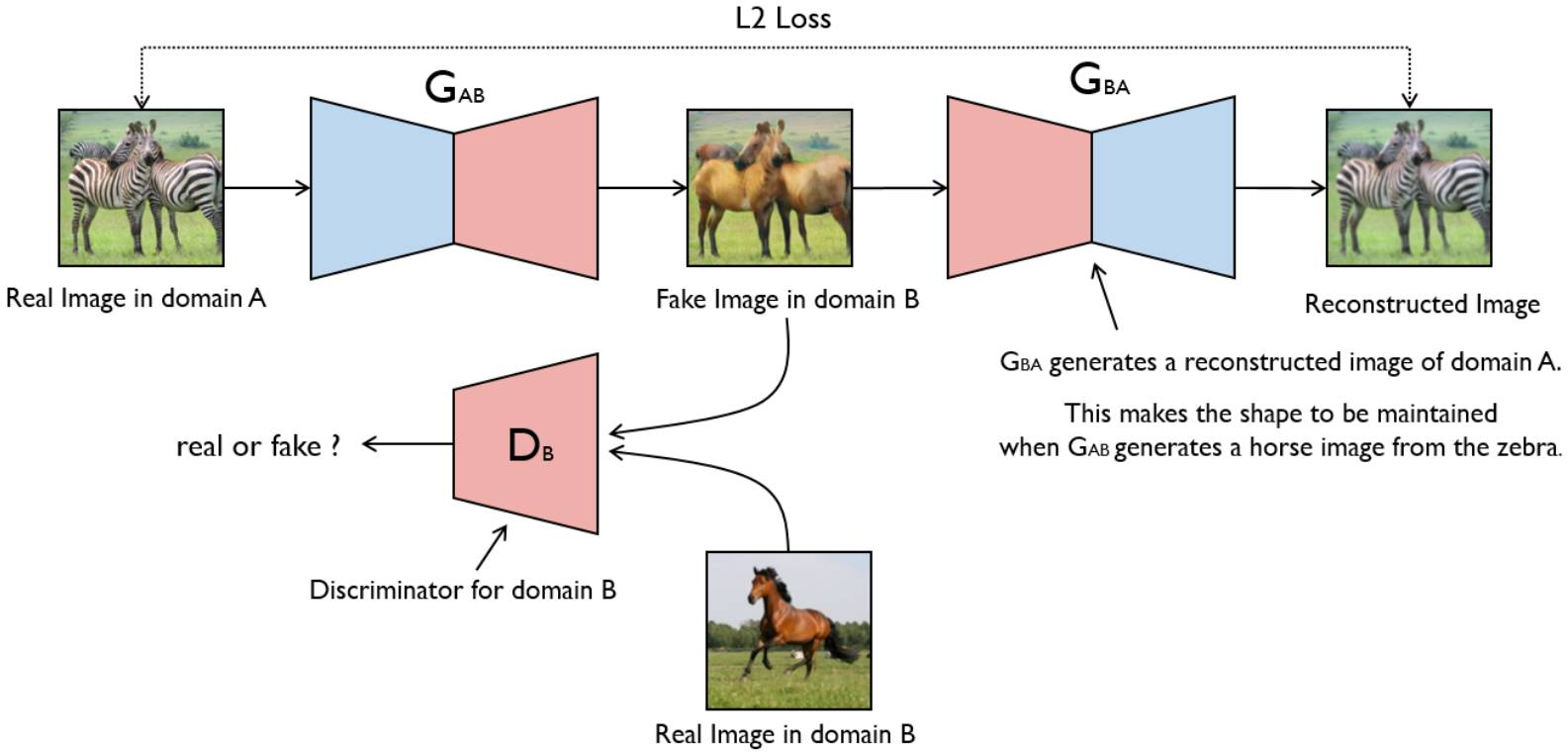
$$AdaIn(x, y) = \sigma(y)\left(\frac{x-\mu(x)}{\sigma(x)}\right) + \mu(y)$$

Shifts and rescales the layers of the content image to match those of the style image

StyleGAN examples:

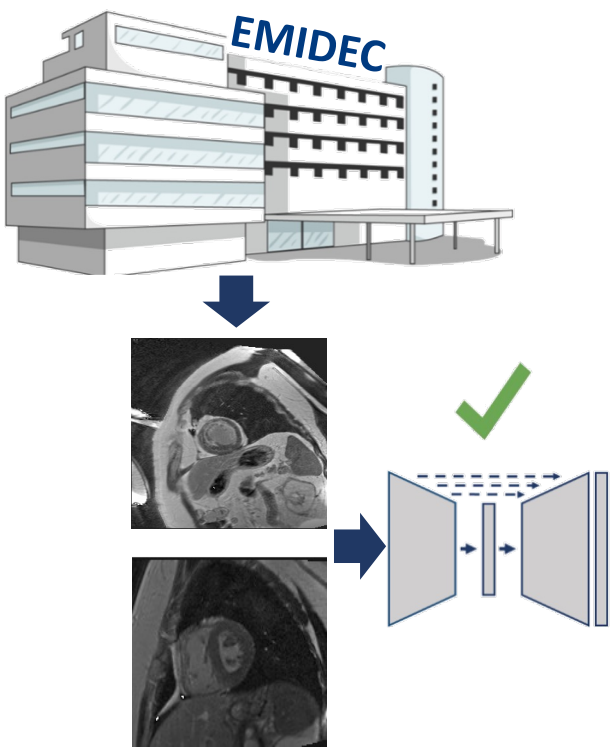


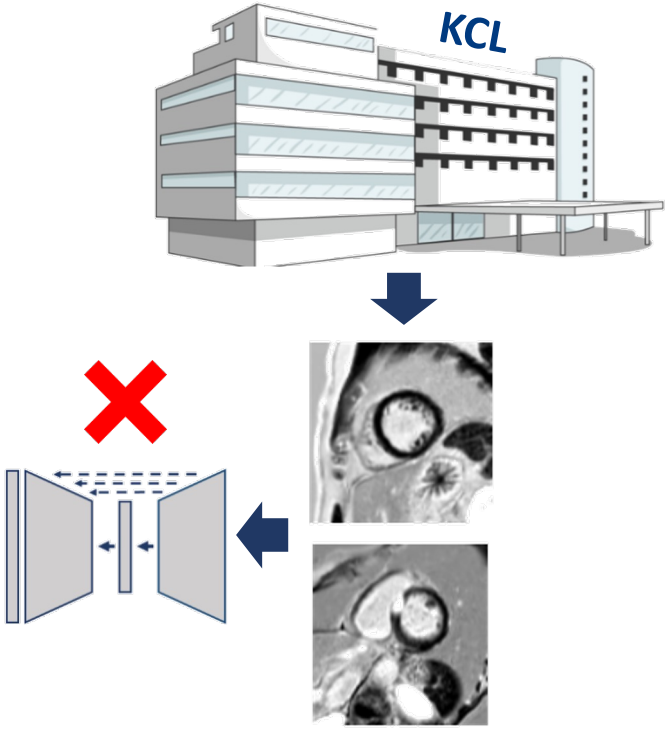
CycleGAN:


L2 Loss
G_{AB}
G_{BA}
Real Image in domain A
Fake Image in domain B
Reconstructed Image
G_{BA} generates a reconstructed image of domain A.
This makes the shape to be maintained
when G_{AB} generates a horse image from the zebra.
real or fake ?
D_B
Discriminator for domain B
Real Image in domain B

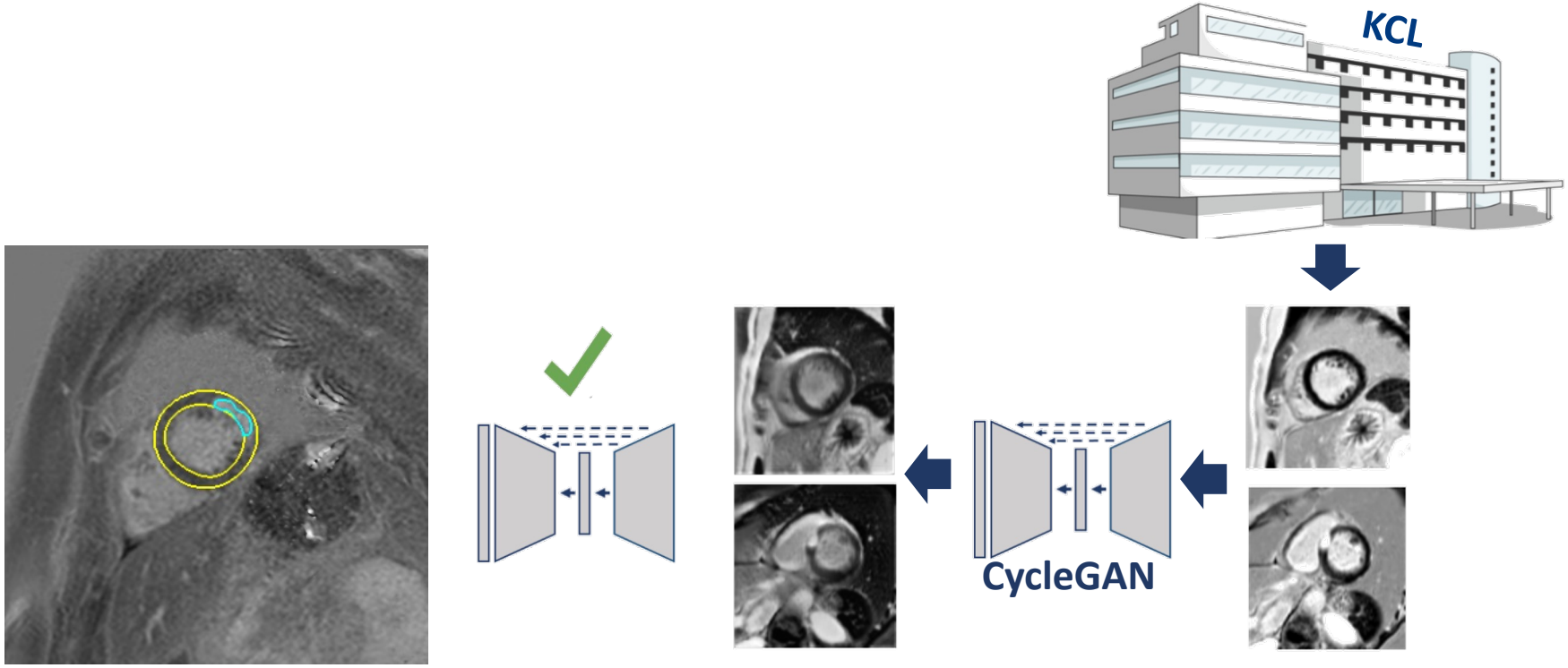
CycleGAN examples:

CycleGAN examples:

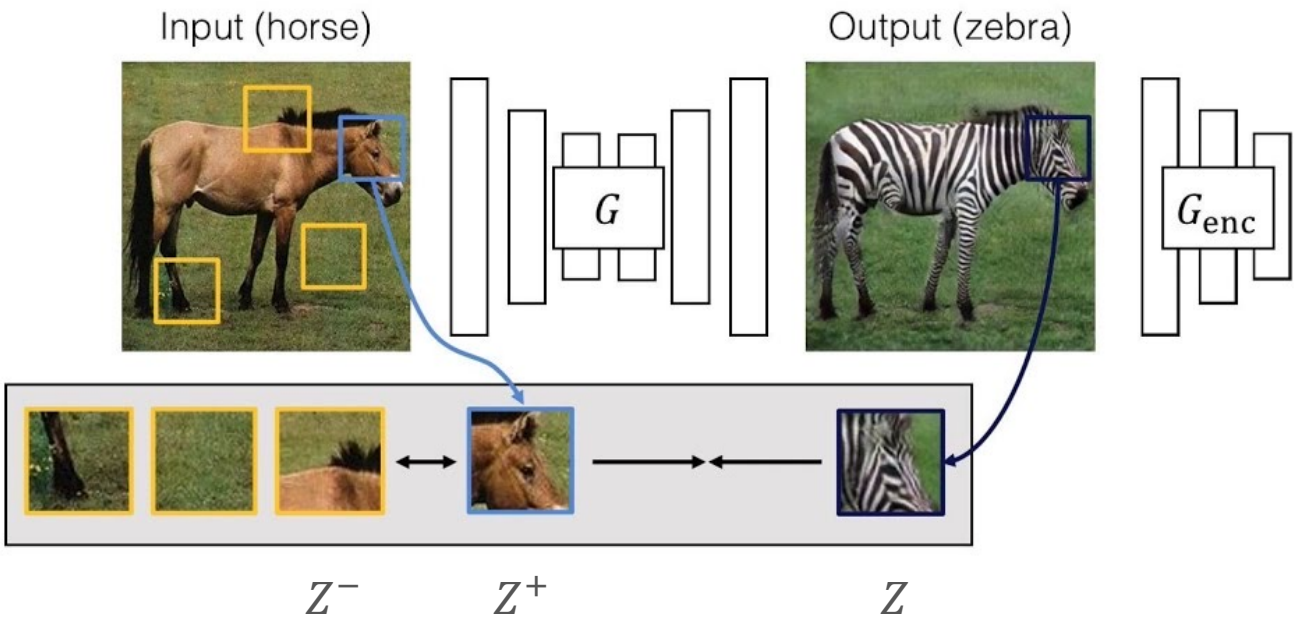




CycleGAN examples:

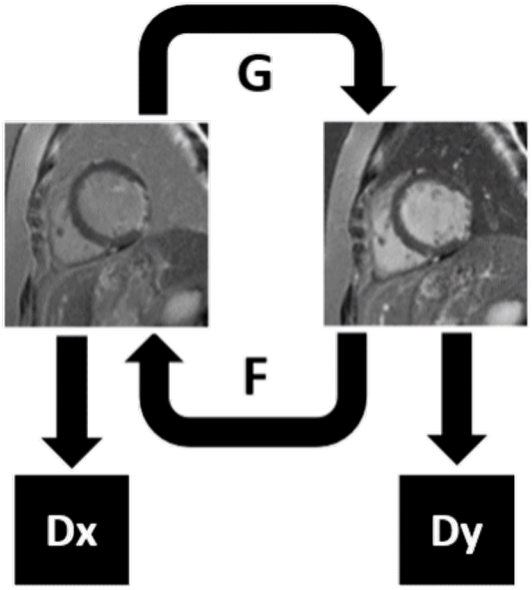


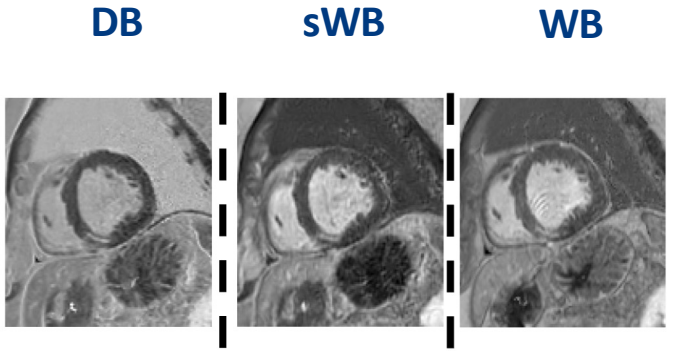
CUT:



$$\mathcal{L}_{con}(Z, Z^+, Z^-) = -\log\left[\frac{\exp(Z \cdot Z^+/\tau)}{\exp(Z \cdot Z^+/\tau) + \sum_{n=1}^{N} \exp(Z \cdot Z_n^-/\tau)}\right]$$

CUT example:



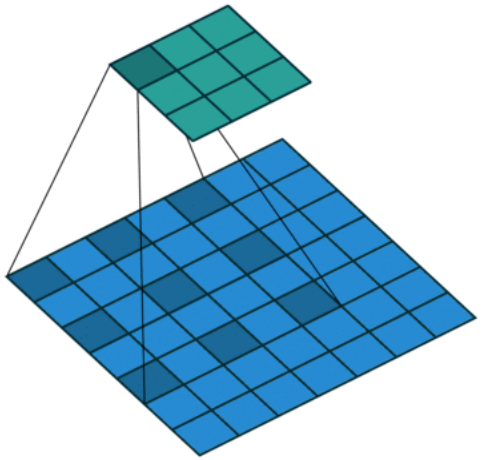


Tricks:

- Maximise $\log D_\alpha\big(G_\beta(z)\big)$ instead of minimising $\log 1 - D_\alpha\big(G_\beta(z)\big)$

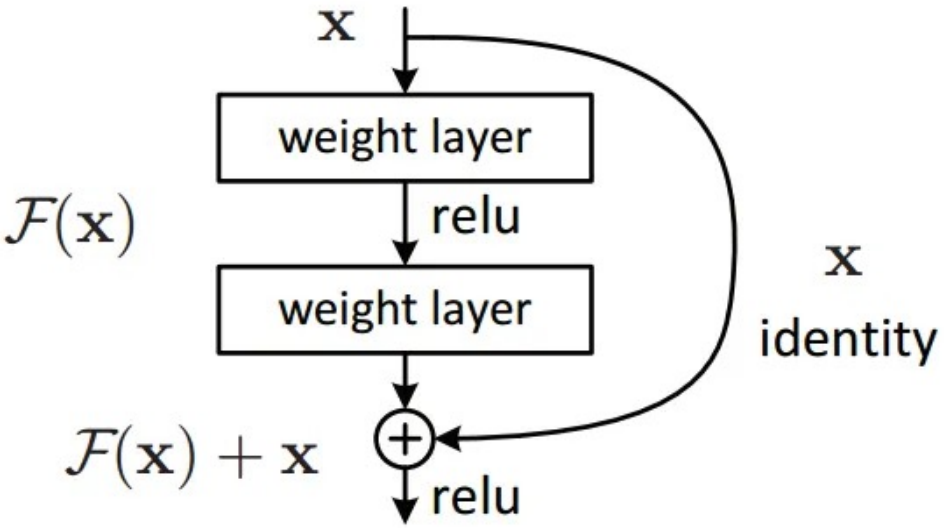
Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions



Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers

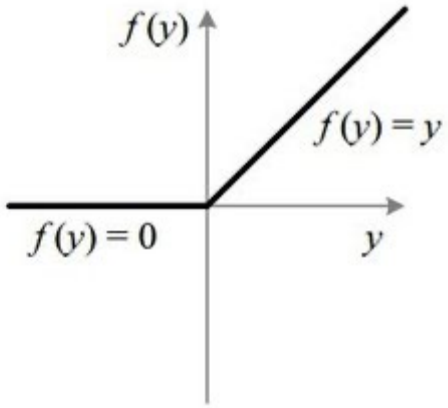


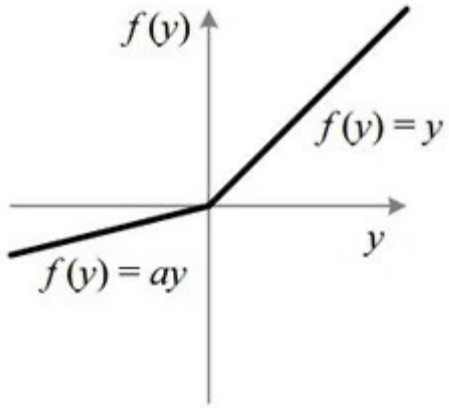
Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers
- Batch norm

Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers
- Batch norm
- Leaky ReLU





Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers
- Batch norm
- Leaky ReLU
- Normalise outputs to [-1,1] & tanh activate in output layer

Tricks:

- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers
- Batch norm
- Leaky ReLU
- Normalise outputs to [-1,1] & tanh activate in output layer
- Hybrid models – VAE-GAN

Tricks:

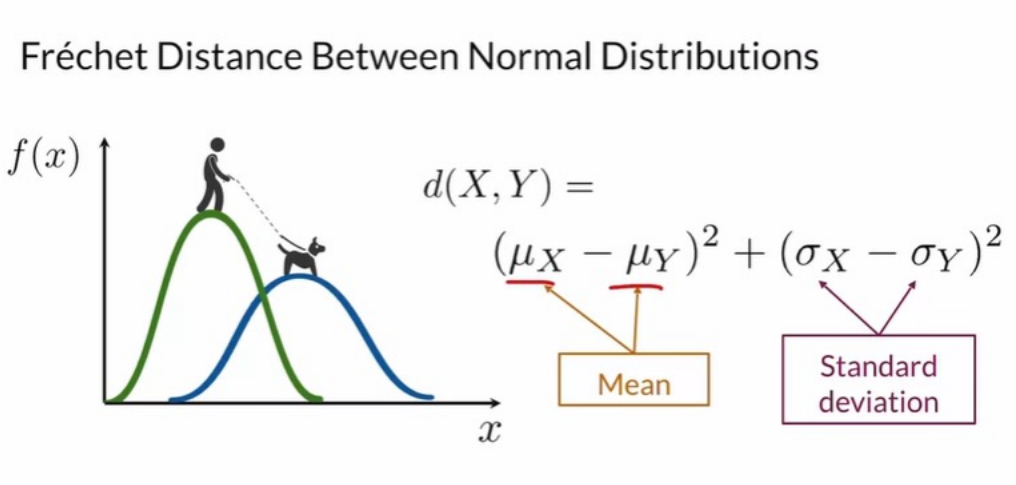
- Maximise $\log D_\alpha\left(G_\beta(z)\right)$ instead of minimising $\log 1 - D_\alpha\left(G_\beta(z)\right)$
- Strided convolutions
- Residual layers
- Batch norm
- Leaky ReLU
- Normalise outputs to [-1,1] & tanh activate in output layer
- Hybrid models – VAE-GAN
- Use labels - condition

Evaluation

Evaluating synthetic data is very hard and still requires a lot of research!!

(especially for medical domains)

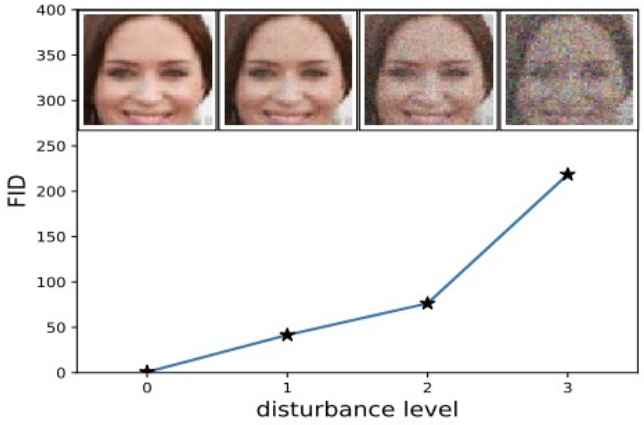
Fréchet inception distance (FID):

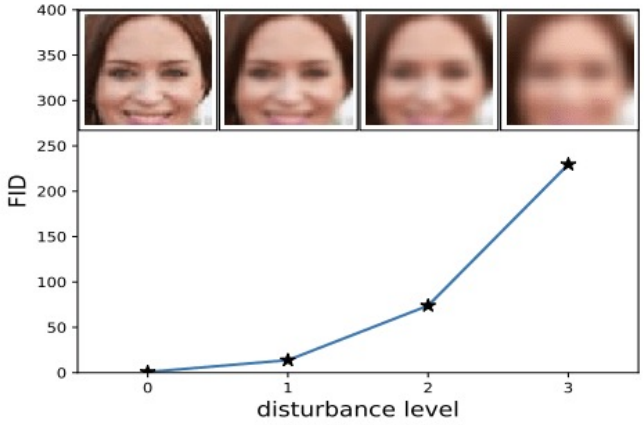


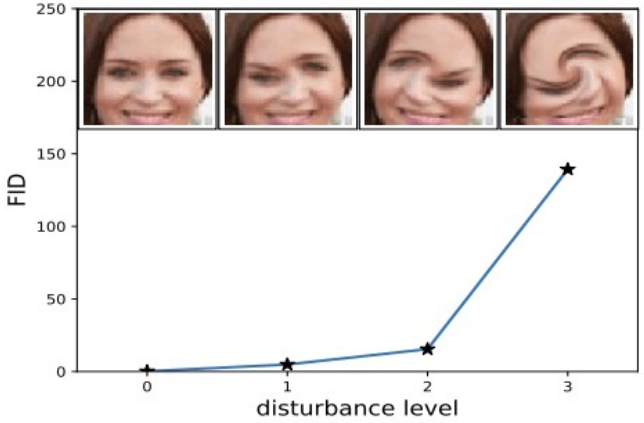
We use the activations from the pre-trained Inception V3 model to summarise each image and compute the distance between these features

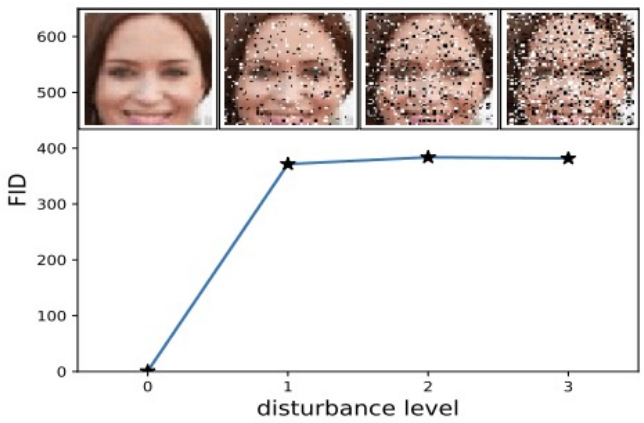
$$FID = \|\mu_r - \mu_g\|^2 + Tr(\Sigma_r + \Sigma_g - 2(\Sigma_r \Sigma_g)^{1/2})$$

FID example:


FID
disturbance level

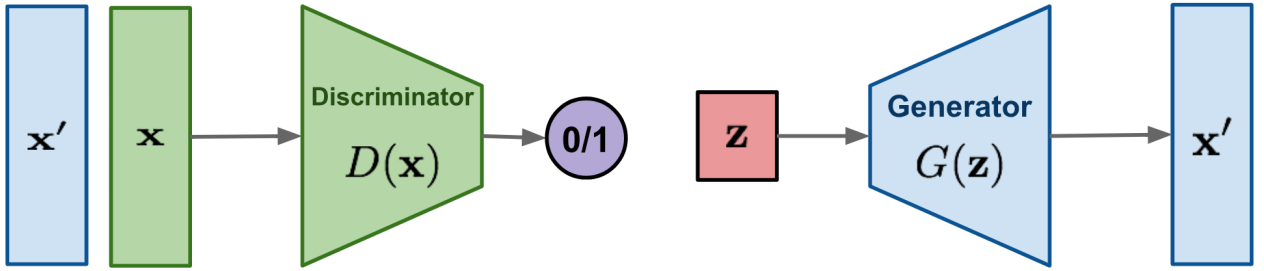

FID
disturbance level


FID
disturbance level

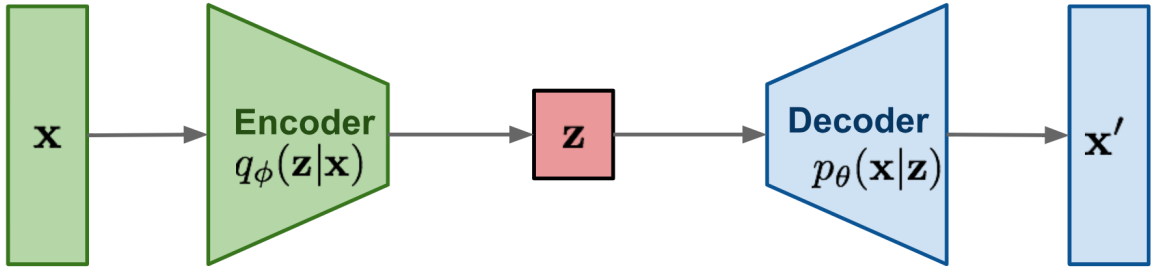

FID
disturbance level

Summary:

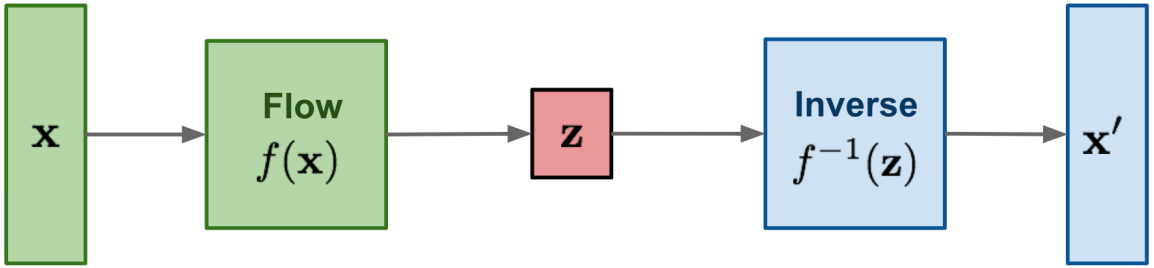
GAN: minimax the classification error loss.



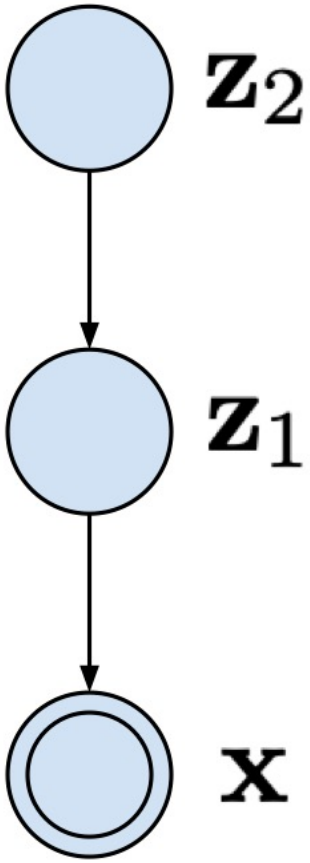
VAE: maximize ELBO.



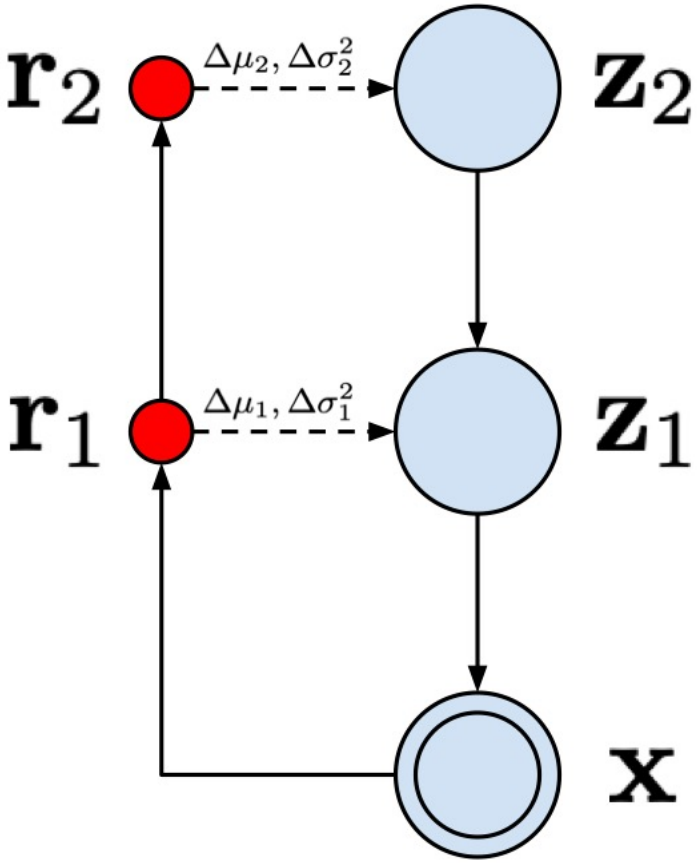
Flow-based generative models: minimize the negative log-likelihood



Hierarchical VAE:



Top-down hierarchical VAE with bottom-up path:



Diffusion models:

A diffusion model is a hierarchical VAE with the bottom-up path defined by a diffusion process and the top-down path parametrised by neural networks (reversed diffusion).

Discussion points

1. Normalising flows are attractive as you can explicitly evaluated the likelihood function and are good for learning distributions but are yet to be shown to generate realistic samples of high-dimensional data (like images).

Discussion points

2. GANs can generate highly realistic images but have an unstable training process that benefits from being regularised or constrained.

Learning objectives

The student can:

- Understand the benefits of normalising flows in the context of the limitations of VAEs
- Describe the origin of the name “normalising flows”
- Summarise the normalising flow algorithm
- Discuss the intuition behind generative adversarial networks (GANs)
- Formulate the GAN training process and loss function
- Classify applications of GANs
- Give examples of issues with GANs
- Motivate solutions to these issues

IMAG/e

TU/e Technische Universiteit Eindhoven University of Technology

Questions?