Generative Adversarial Networks (GANs)

Oussema Dhaouadi

FERIENAKADEMIE 2018

1. Motivation
2. Principles of Information Theory & Machine Learning
3. Generative Adversarial Networks
4. Photographic Image Synthesis
5. MR to CT Synthesis

1. Motivation

What can generative models do?

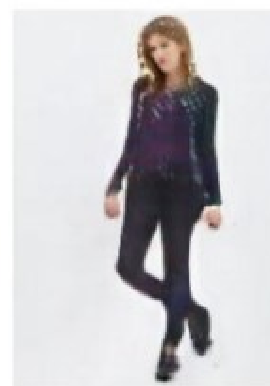
Quiz: Which ones are real ?

Progressive GAN
10/2017
1024x1024

1. Motivation

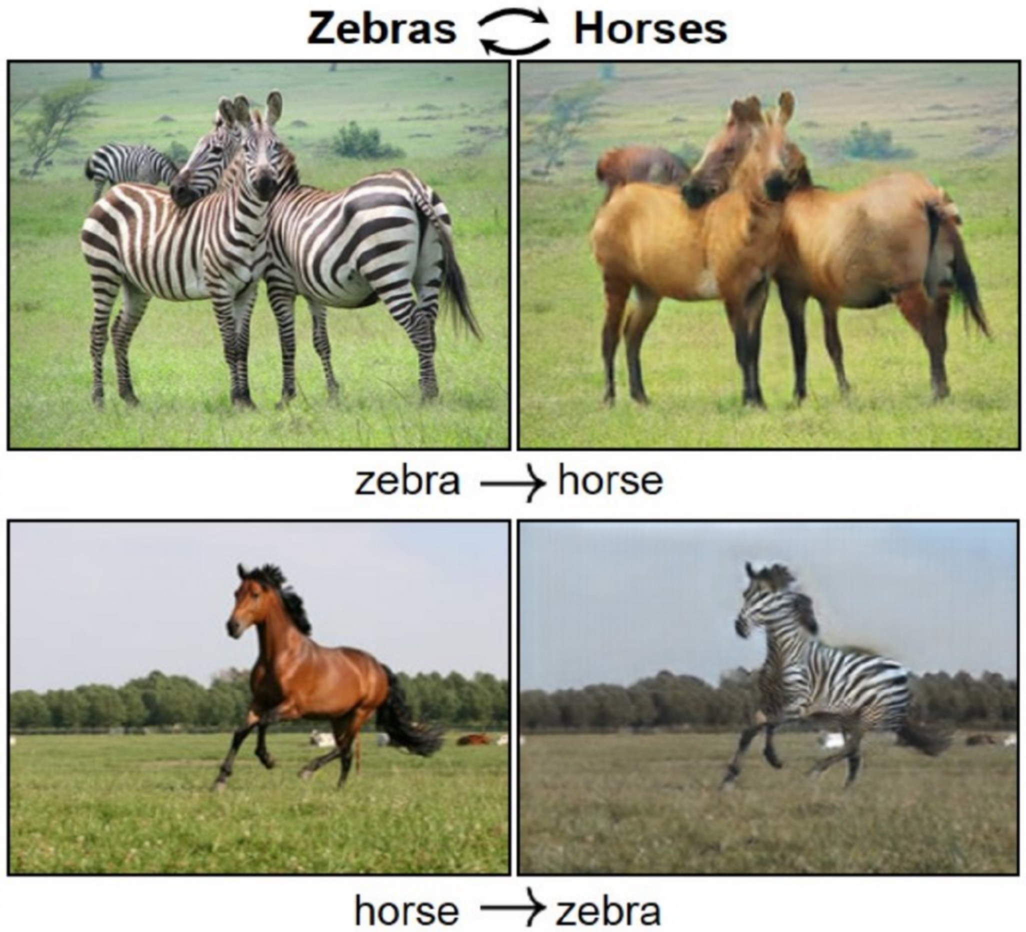
ose Guided Person Image Generation

Ground truth



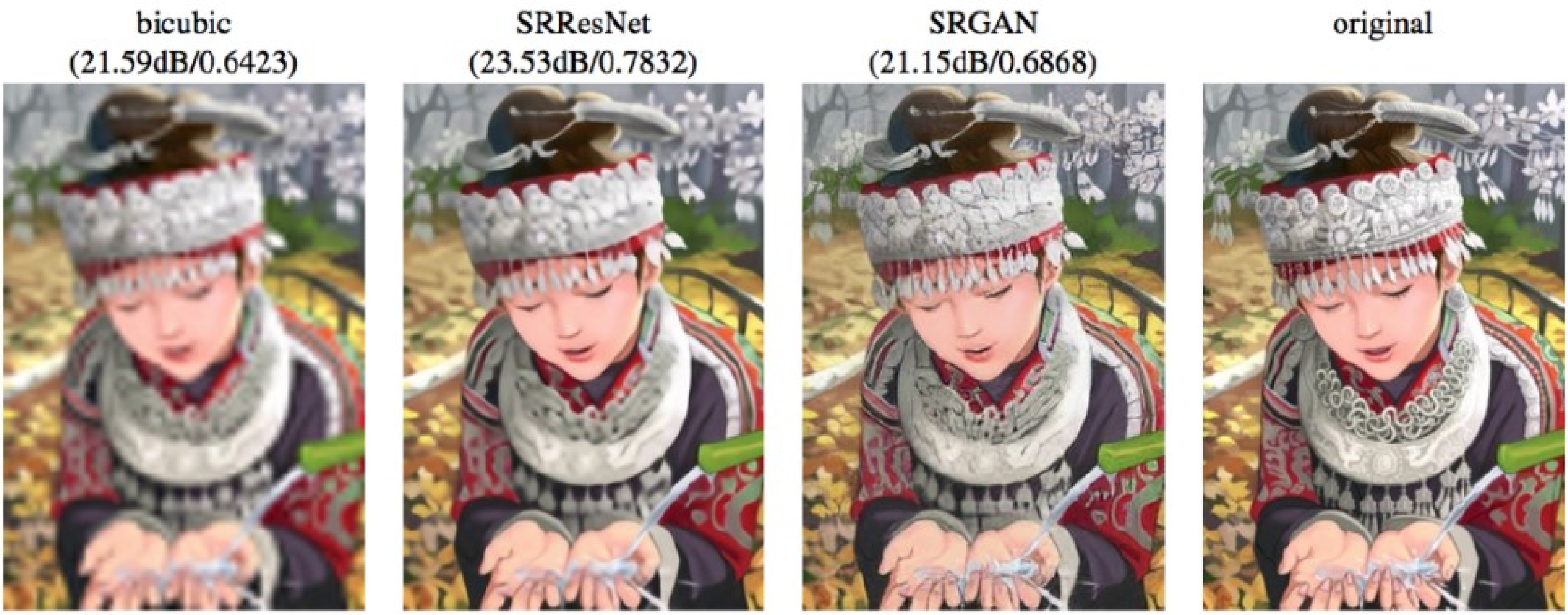
Generated

Cross-domain transfer: e.g. style transfer



CycleGAN

Low resolution to high resolution

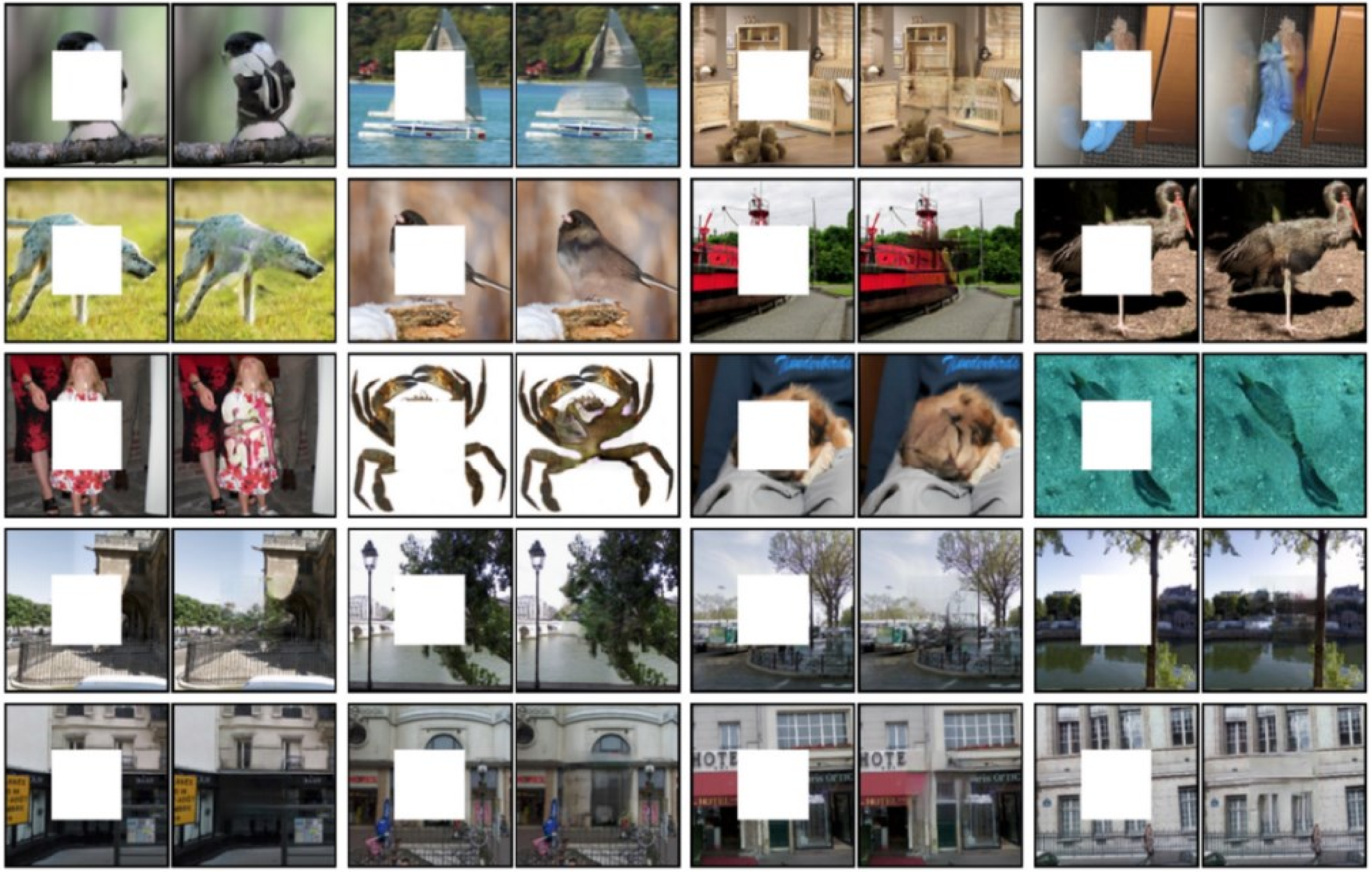


Super resolution
GAN (SRGAN)

Text to image

StackGAN

Image inpainting

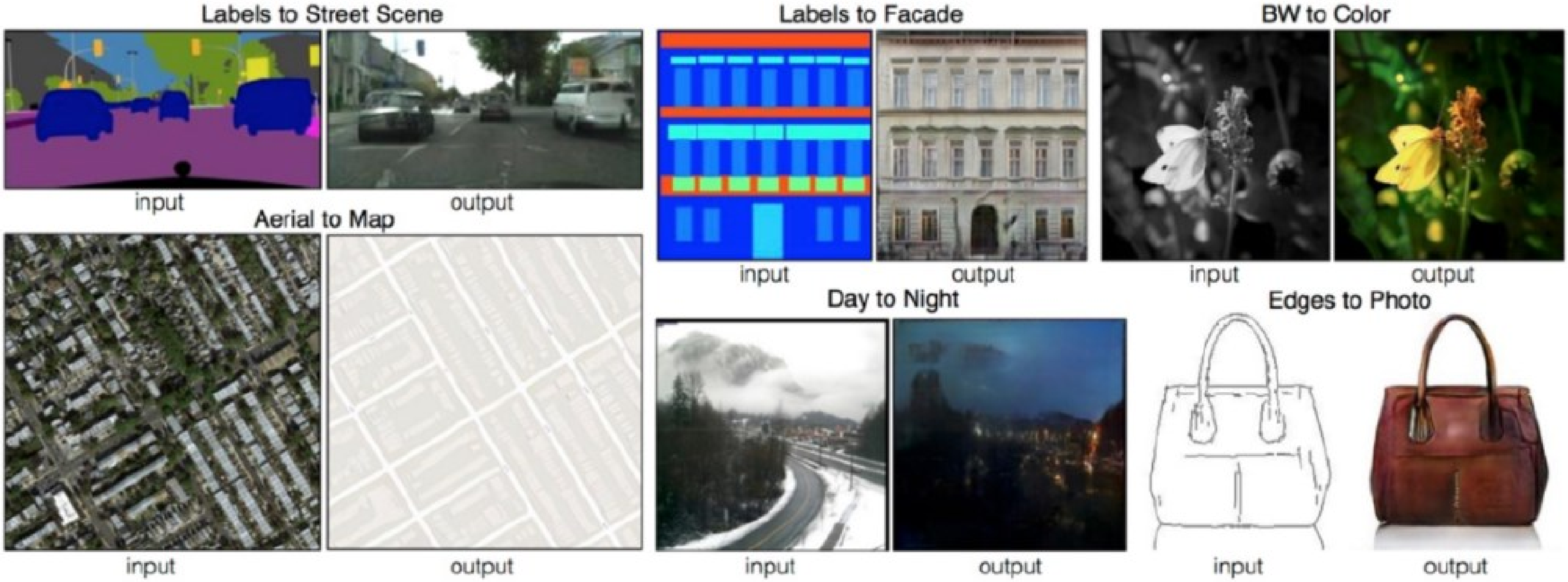


Maching style

(b) Handbag images (input) & **Generated** shoe images (output)

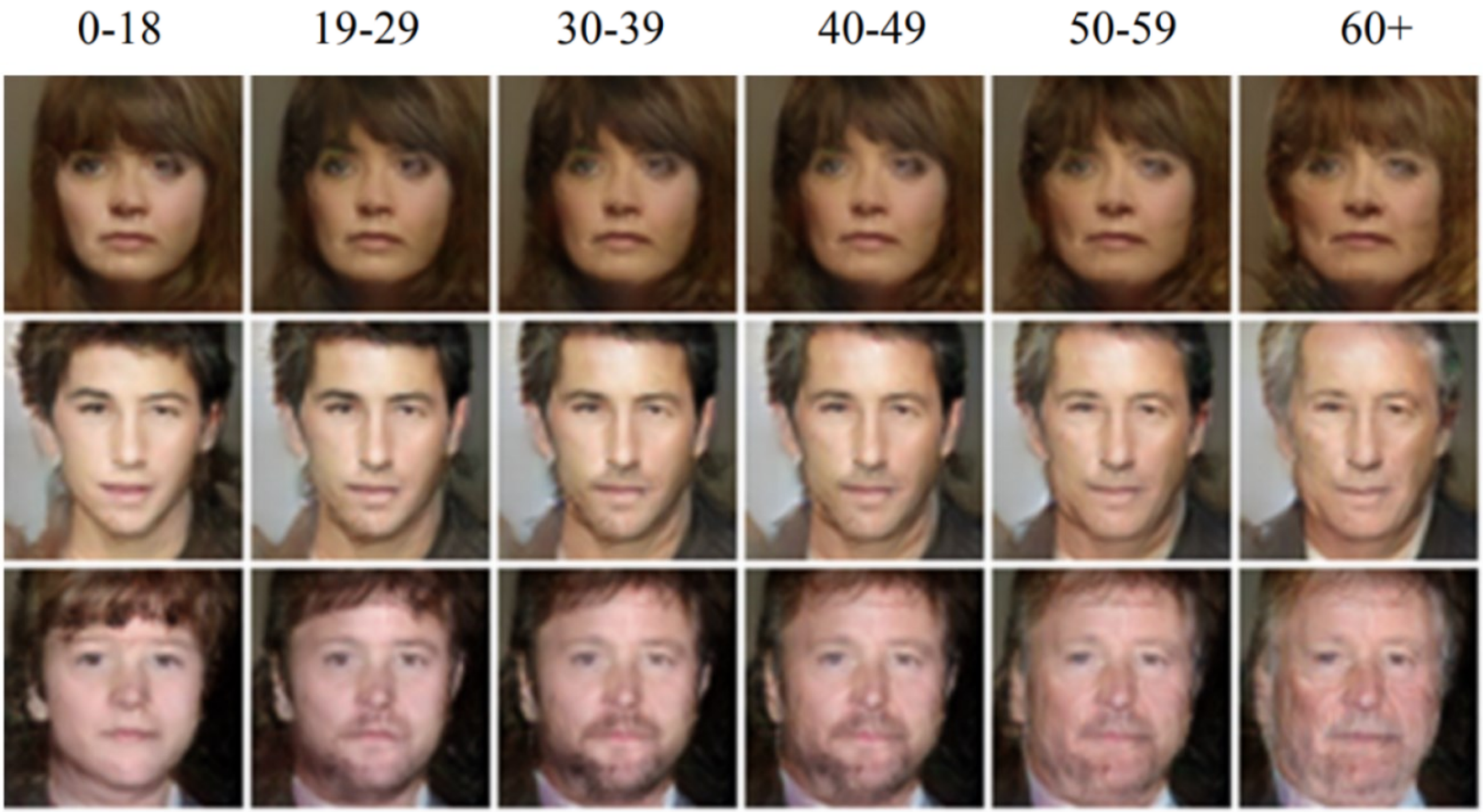
DiscoGAN

image-to-image translation



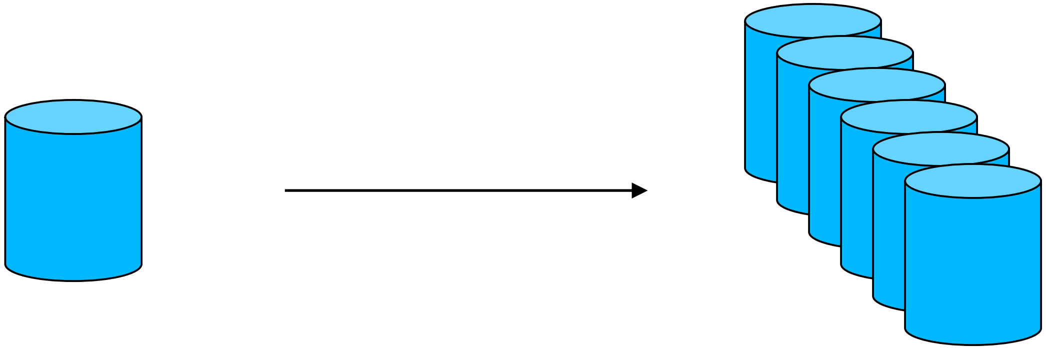
pix2pix

Face aging



Age-cGAN

Data augmentation



2. Principles of Information Theory & Machine Learning

Shannon information or the self-information content:

$$\mathbb{I}(X) := -\log p(X) \in [0, \infty)$$

measures the likeliness of an event

Entropy:

$$\mathbb{H}(X) := \mathbb{E}_{x \sim p_X}[\mathbb{I}(X)] = -\sum_{i=1}^{n} p_X(x_i) \log p_X(x_i) .$$

measure of the “uncertainty”

Binary cross entropy:

$$\mathbb{H}(X, Y) := \mathbb{E}_{x \sim p_X}[\mathbb{I}(Y)] = -\sum_{i=1}^{n} p_X(x_i) \log p_Y(y_i) .$$

measures the amount of certainty how similar two random variables are f-divergence:

$$\mathbb{D}_f(p_X \| p_Y) := \sum_{i=1}^{n} f\left(\frac{p_X}{p_Y}\right) dY$$

where X and Y are random variables and f is a convex function such that f(1) = 0.

Kullback–Leibler divergence (also called relative entropy):

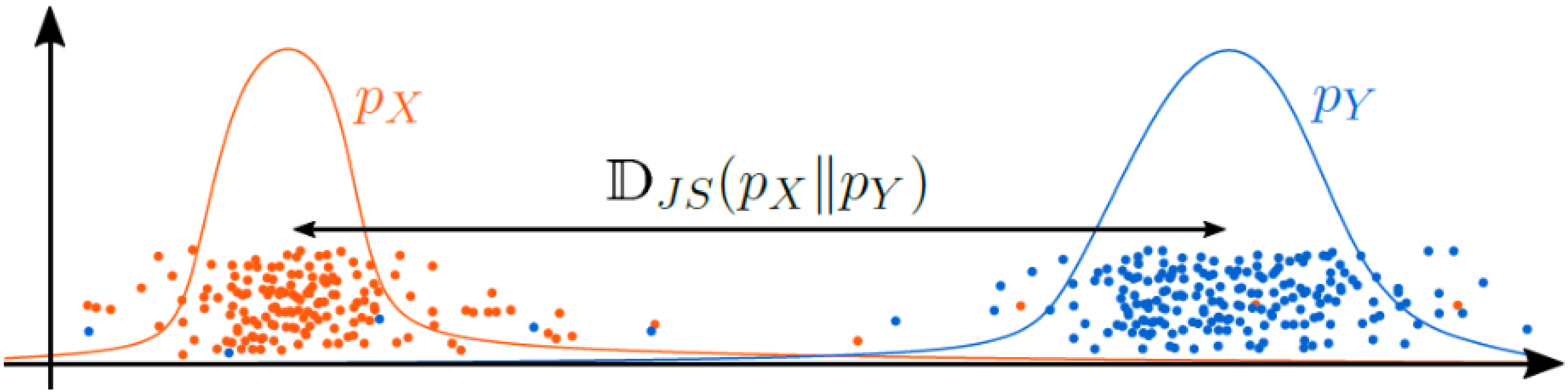
$$\mathbb{D}_{KL}(p_X \| p_Y) := \sum_i p_X(x_i) \cdot \log \frac{p_X(x_i)}{p_Y(y_i)}$$

Not symmetric

Jensen–Shannon divergence:

$$\mathbb{D}_{JS}(p_X \| p_Y) = \mathbb{D}_{JS}(p_Y \| p_X) = \frac{1}{2}\mathbb{D}_{KL}(p_X \| \frac{p_X + p_Y}{2}) + \frac{1}{2}\mathbb{D}_{KL}(p_Y \| \frac{p_X + p_Y}{2})$$

Smoothed version of KL-divergence



- A discriminative model D describes the discrete mapping function
$$x \mapsto \hat{y} \coloneqq \mathrm{D}(\mathrm{x}; \boldsymbol{\theta}_D) \sim p_{\boldsymbol{\theta}_D}$$
x : features $\hat{y}$: predictions y : labels
$\boldsymbol{\theta}_D$: parameters (biases & weights)

- Goal: To find a good representation for $p(y|x)$ without explicitly modeling the generative process, such that
$$p_{\boldsymbol{\theta}_D} \approx p(y|x)$$

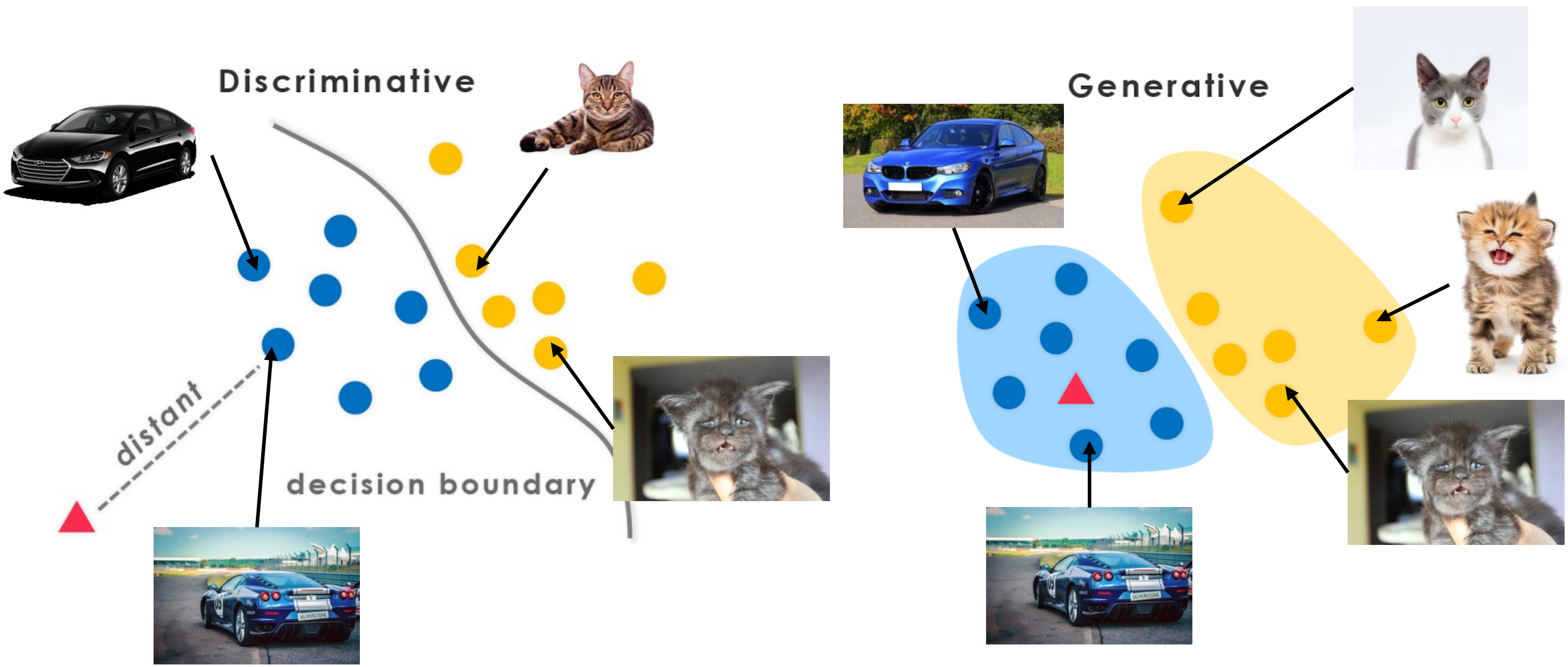
- Example techniques: K nearest neighbors, logistic regression, linear regression, etc.

- A generative model G describes the mapping function
$$y \mapsto \hat{x} \coloneqq \mathrm{G}(\mathrm{y}; \boldsymbol{\theta}_G) \sim p_{\boldsymbol{\theta}_G}$$
x : features $\hat{x}$: outputs
y : latent variable $\boldsymbol{\theta}_G$: parameters

- Goal: To find a probabilistic model that explicitly models the distribution of the features, such that
$$p_{\boldsymbol{\theta}_G} \approx p(x) = \int p(x|y)\, p(y)\, dy$$

- Example techniques: Hidden Markov models, Mixture models, etc.

2.2. Generative vs. Discriminative Models



The objective of a generative model is to find the optimal $\boldsymbol{\theta}_G$ such that:

$$p_{\boldsymbol{\theta}_G} \approx p(x) = \int p(x|y)\, p(y)\, dy$$

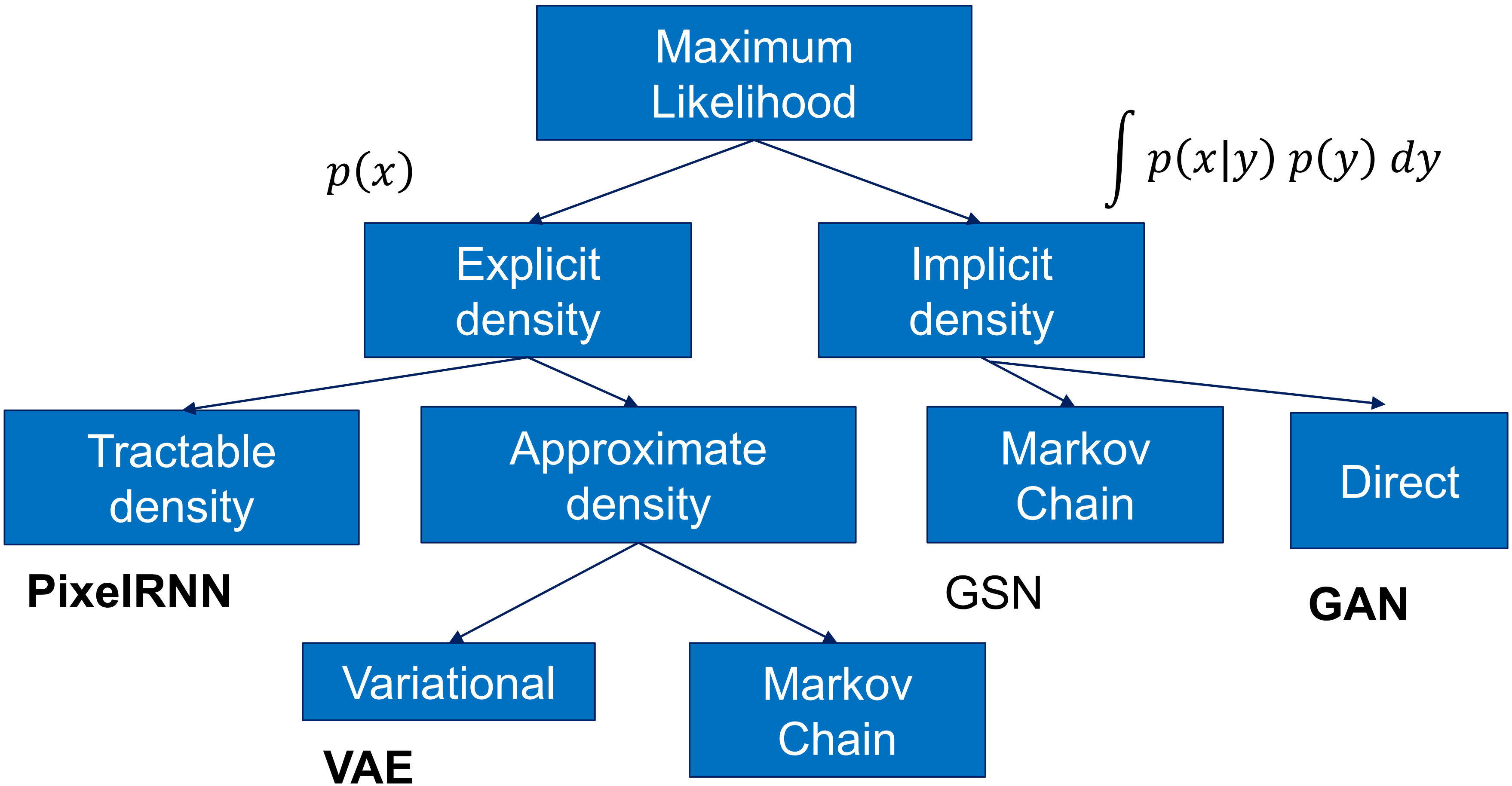
How to estimate $p(x)$?

- In a high dimensional space, estimating p(x) is not easy!
- Neural networks are the best models that can estimate high dimensional distributions by providing a high number of parameters and thus represent complex transformations.

But how to update $\boldsymbol{\theta}_G$ in order to represent $\mathbf{p}(\boldsymbol{x})$?

$$\begin{aligned}\boldsymbol{\theta}_G^* &= argmax_{\boldsymbol{\theta}_G}\, p(x) \\ &= argmax_{\boldsymbol{\theta}_G} \log \mathbf{p}(\boldsymbol{x})\end{aligned}$$

log Maximum Likelihood


Maximum Likelihood
$p(x)$
$\int p(x|y)\, p(y)\, dy$
Explicit density
Implicit density
Tractable density
Approximate density
Markov Chain
Direct
PixelRNN
GSN
GAN
Variational
Markov Chain
VAE

2.4. Pixel RNN

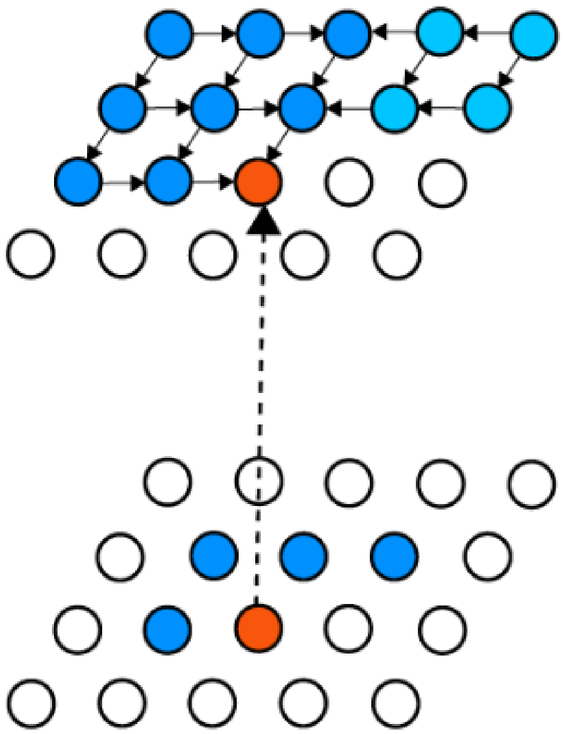
- Describe the Likelihood of an image as the joint distribution of all pixels:

$$p(x) = \prod_{i=1}^{n} p(x_i|x_1, \ldots, x_{i-1}) = \prod_{i=1}^{n} p(x_i|x_{<i})$$

- A sequence problem wherein the next pixel value is determined by all the previously generated pixel values.
- Use LSTM to describe the recurrence ➔ BiLSTM

- Drawbacks: sequential generation
slow to train

- Alternative: Use CNN to reduce the computational cost => PixelCNN and PixelCNN++



Diagonal BiLSTM

2.5. Variational Autoencoders VAEs

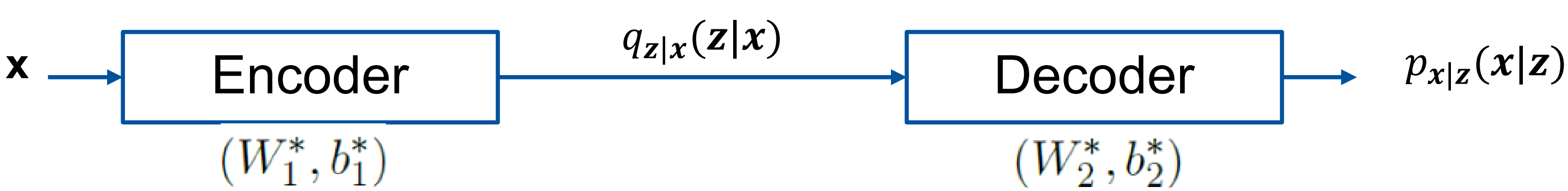
- Describe the Likelihood of an image using a latent vector **z**:

$$p_{\mathbf{x}}(\mathbf{x}) = \int p_{\mathbf{z}}(\mathbf{z}) \cdot p_{\mathbf{x}|\mathbf{z}}(\mathbf{x}|\mathbf{z})dz \;\; = \int p_{\mathbf{x}}(\mathbf{x}) \cdot p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})dz$$

Intractable!

$$log\, p_{\mathbf{x}}(\mathbf{x}) = \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})}\left[log\, p_{\mathbf{x}}(\mathbf{x})\right]$$
$$\overset{A.1}{=} \underbrace{\mathbb{E}_{\mathbf{z}}\left[log\, p(\mathbf{x}|\mathbf{z})\right] - \mathbb{D}_{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))}_{\mathcal{L}_{VAE}(\mathbf{x},\theta,\phi)} + \underbrace{\mathbb{D}_{KL}(q(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}|\mathbf{x}))}_{\geq 0}$$

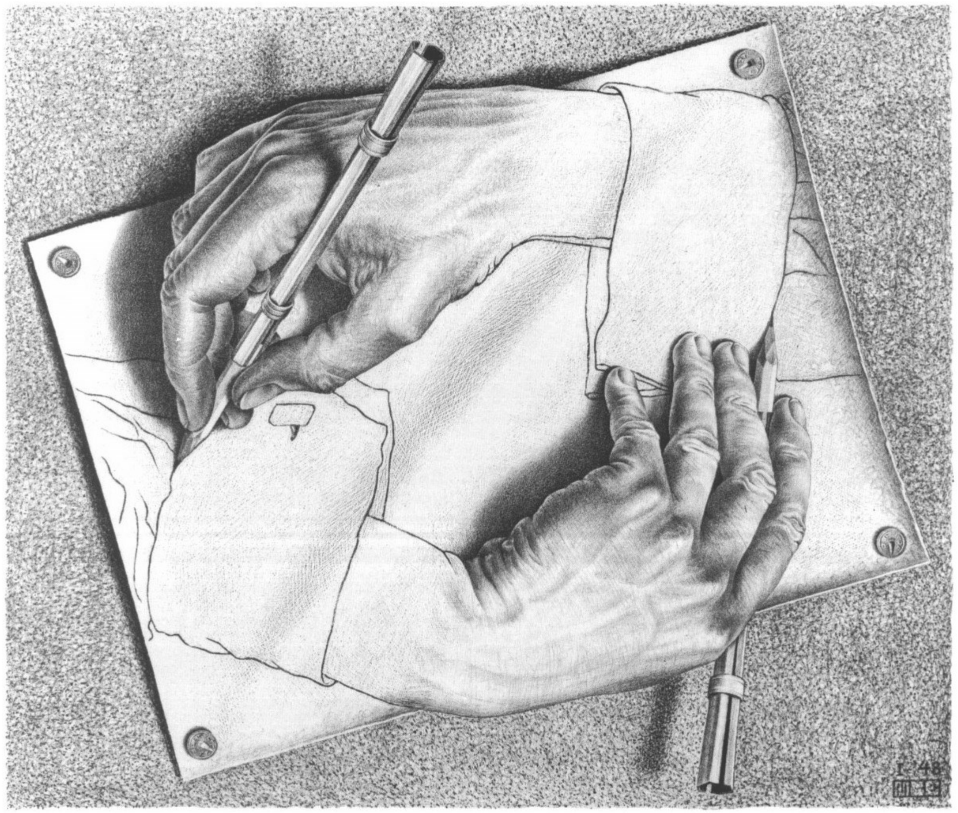
$p_{\boldsymbol{z}}(\boldsymbol{z})$ is known, e.g. Gaussian



$$(W_1^*, b_1^*), (W_2^*, b_2^*) = \arg \max_{(W_1,b_1),(W_2,b_2)} \sum_{i=1}^{N} \mathcal{L}_{VAE}(x_i, \theta, \phi) \text{ with } \mathcal{L}_{VAE}(x_i, \theta, \phi) \leq log\, p_{\mathbf{x}}(\mathbf{x})$$

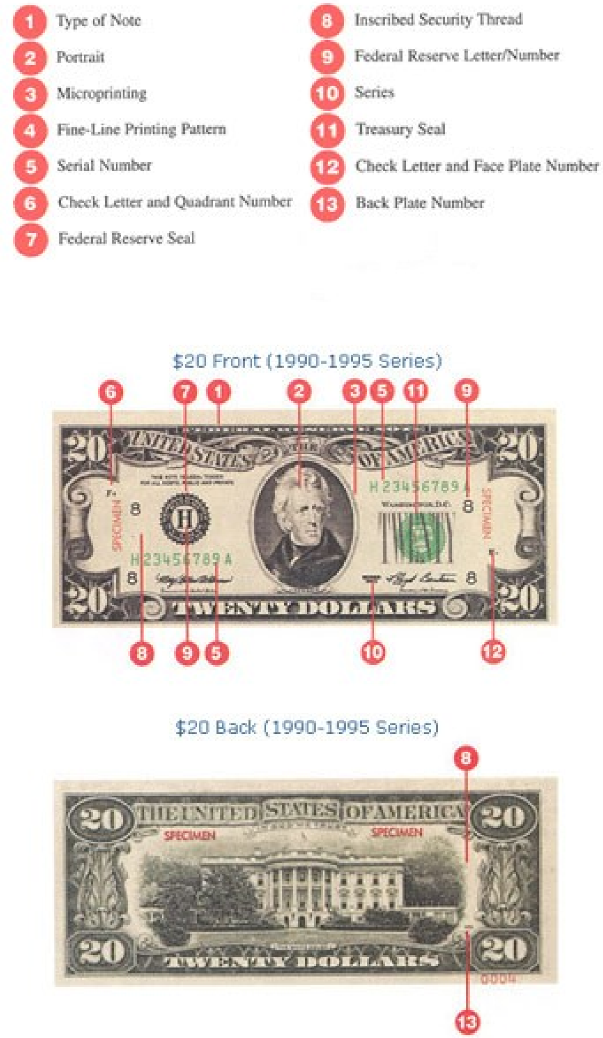
Low quality, since VAE maximizes the so-called Evidence Lower Bound (ELBO) $\mathcal{L}_{VAE}(x_i, \theta, \phi)$

3. Generative Adversarial Networks



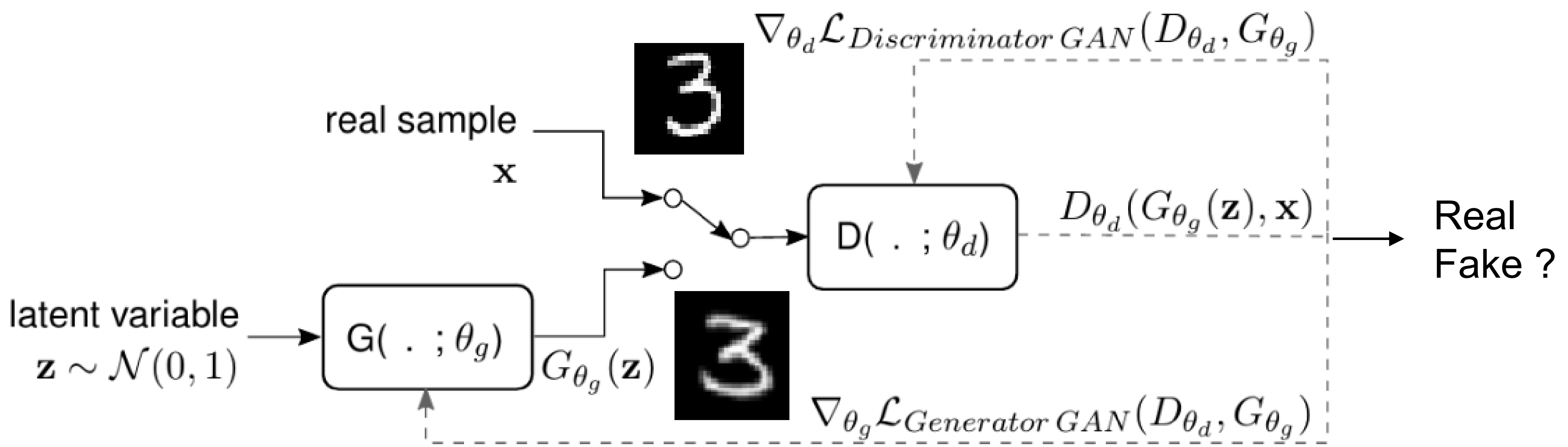
3.0. Idea

Optimization problem : *zerosum/minimax game*



[Wikiwand]

3.1. Network Structure



z is a latent variable (e.g. random varaible), x is a sample from the dataset to learn

General objective: $G_\theta(\mathbf{z}) \sim p_\mathrm{g} \approx p_\mathrm{x}$

$$\mathbf{z} \sim p_\mathbf{z}(\mathbf{z}) \mapsto G(\mathbf{z}; \theta_g) \sim p_g$$

$$(\mathbf{x} \sim p_{data}(\mathbf{x}), G(\mathbf{z}; \theta_g) \sim p_g) \mapsto D(\mathbf{x}, G(\mathbf{z}; \theta_g); \theta_d) \sim p_d \in \{0, 1\}$$

How to learn sampling from complex and high-dimentional distribution ?

Game-theory approach: learn to generate from training distribution through 2-player game

https://www.youtube.com/watch?v=XOxxPcy5Gr4

3.1. Network Structure

- Training CelebA & interpolating over **z**

https://www.youtube.com/watch?v=XOxxPcy5Gr4

3.2. Optimization Problem

Optimization problem : *zerosum/minimax game*

$$\theta_g^*, \theta_d^* = \arg\min_{\theta_g} \arg\max_{\theta_d} \mathcal{L}_{GAN}(D_{\theta_d}, G_{\theta_g}) \tag{2.29}$$

$$= \arg\min_{\theta_g} \arg\max_{\theta_d} \mathbb{E}_{\mathbf{x}\sim p_{data}} \left[log\left(D(\mathbf{x};\theta_d)\right)\right] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \left[log\left(1 - D(G(\mathbf{z};\theta_g);\theta_d)\right)\right] \tag{2.30}$$

$$= \arg\min_{\theta_g} \arg\max_{\theta_d} \mathbb{E}_{\mathbf{x}\sim p_{data}} \left[log\left(D(\mathbf{x};\theta_d)\right)\right] + \mathbb{E}_{\mathbf{x}\sim p_g} \left[log\left(1 - D(\mathbf{x};\theta_d)\right)\right] \tag{2.31}$$

$$\theta_g^* = \arg\min_{\theta_g} \mathcal{L}_G(D_{\theta_d}, G_{\theta_g}) \tag{2.32}$$

gradient descent on generator

$$= \arg\min_{\theta_g} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \left[log\left(1 - D_{\theta_d}(G(\mathbf{z};\theta_g))\right)\right] \tag{2.33}$$

Minimize likelihood of discriminator being correct

$$\theta_d^* = \arg\min_{\theta_d} \mathcal{L}_D(D_{\theta_d}, G_{\theta_g}) \tag{2.34}$$

gradient ascent on discriminator

$$= \arg\max_{\theta_d} \left[\mathbb{E}_{\mathbf{x}\sim p_{data}(\mathbf{x})} \left[log\, D(\mathbf{x};\theta_d)\right] + \mathbb{E}_{\mathbf{z}\sim p_z(\mathbf{x})} \left[log\left(1 - D(G_{\theta_g}(\mathbf{z});\theta_d)\right)\right]\right] \tag{2.35}$$

Maximize likelihood of discriminator being correct

Problem: In practice, optimizing the generator objective does not work well!

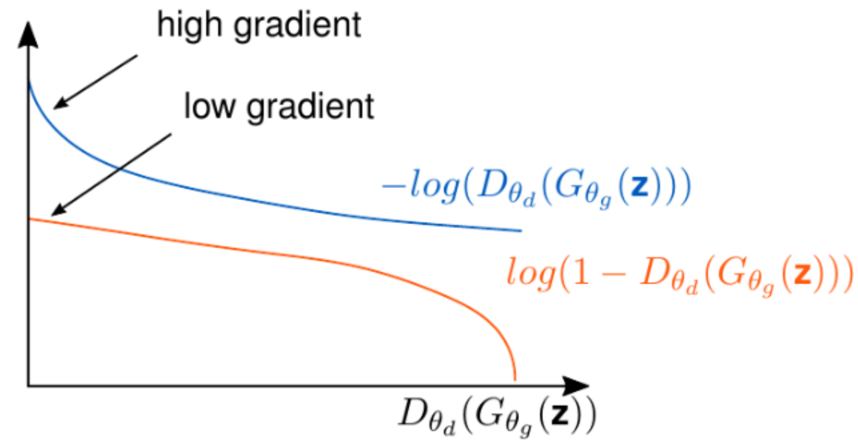


Figure 2.13.: Generator loss gradient

Non-saturating heuristic game:

$$\theta_g^* = \arg\min_{\theta_g} \mathcal{L}_G(D_{\theta_d}, G_{\theta_g})$$
$$= \arg\max_{\theta_g} \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}} \left[log \left(D_{\theta_d}(G(\mathbf{z};\theta_g)) \right) \right]$$

gradient ascent on generator

Maximize likelihood of discriminator being wrong

3.2. Optimization Problem

Problem: GANs may be very instable since they are sensible to hyperparameters such as the learning rate of the optimizer

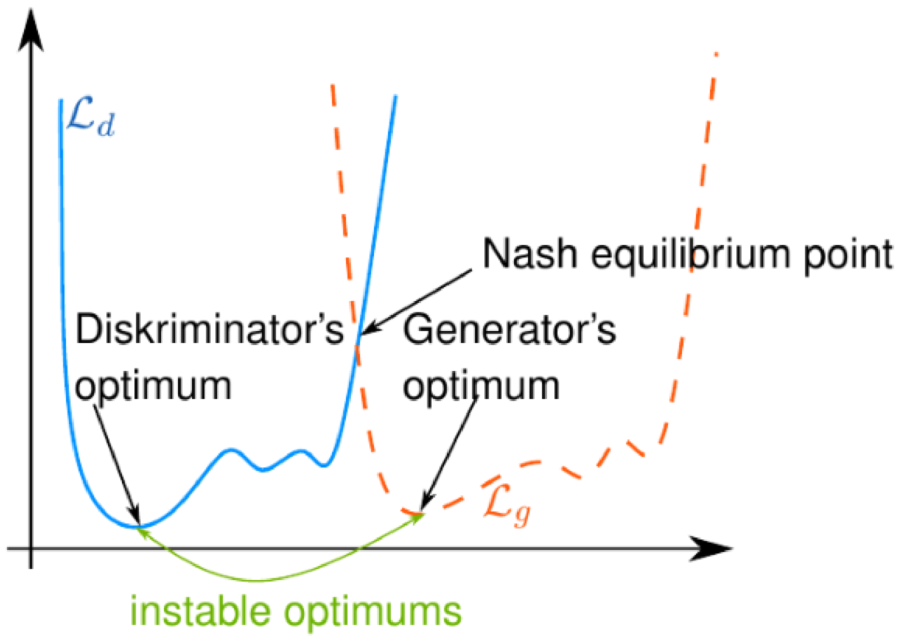


Figure 2.16.: Generator's and Discriminator's loss functions and general instability

Minimizing the overall loss function ⇔ Minimizing the JS(Jenson-Shannon)-Divergence:

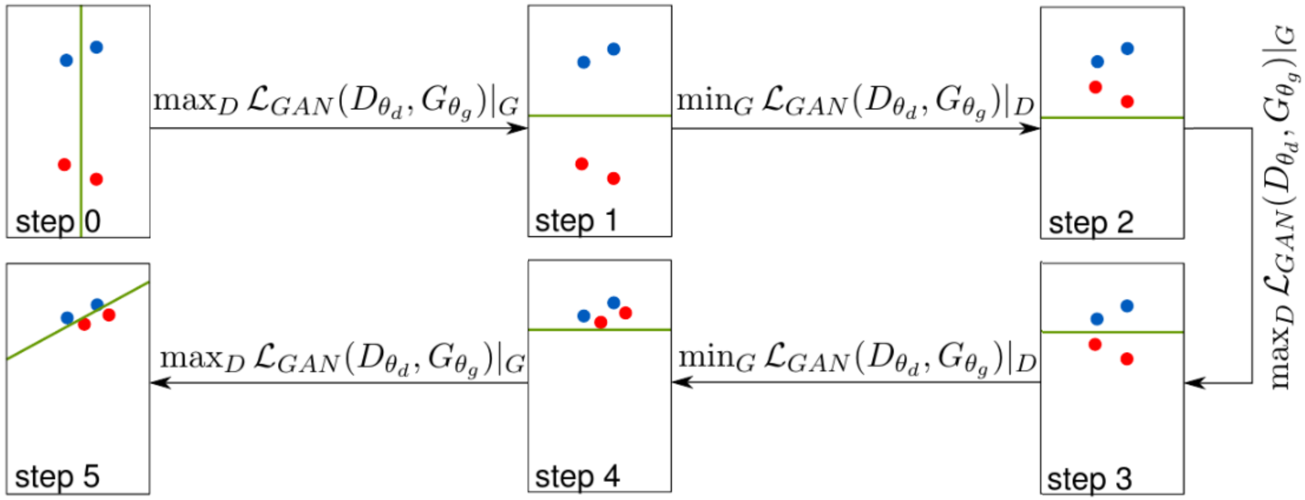
$$\begin{aligned}
\mathcal{L}_{GAN} =& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log D_{\theta_d}(\mathbf{x})\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log(1 - D_{\theta_d}(\mathbf{x}))\right] \\
\overset{2.38}{=}& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log(1 - \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})})\right] \\
=& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log \frac{p_g(\mathbf{x})}{p_{data}(\mathbf{x})+p_g(\mathbf{x})}\right] \\
=& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log \frac{p_{data}(\mathbf{x})}{2*\frac{1}{2}(p_{data}(\mathbf{x})+p_g(\mathbf{x}))}\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log \frac{p_g(\mathbf{x})}{2*\frac{1}{2}(p_{data}(\mathbf{x})+p_g(\mathbf{x}))}\right] \\
=& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log \frac{p_{data}(\mathbf{x})}{\frac{1}{2}(p_{data}(\mathbf{x})+p_g(\mathbf{x}))}\right] - log2 + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log \frac{p_g(\mathbf{x})}{\frac{1}{2}(p_{data}(\mathbf{x})+p_g(\mathbf{x}))}\right] - log2 \\
=& \mathbb{E}_{\mathbf{x}\sim p_{data}}\left[log \frac{p_{data}(\mathbf{x})}{p_{average}(\mathbf{x})}\right] + \mathbb{E}_{\mathbf{x}\sim p_g}\left[log \frac{p_g(\mathbf{x})}{p_{average}(\mathbf{x})}\right] - 2log2 \\
\overset{2.14}{=}& 2\,\mathbb{D}_{JS}(p_{data}||p_g) - 2log2
\end{aligned}$$

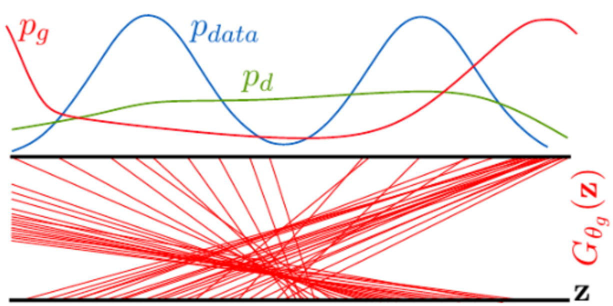
Global optimum (Nash equilibrium) is reached for:

$$D_G^*(\mathbf{x}) \overset{A.2}{=} \frac{p_{data}(\mathbf{x})}{p_{data}(\mathbf{x}) + p_g(\mathbf{x})}$$ for G fixed

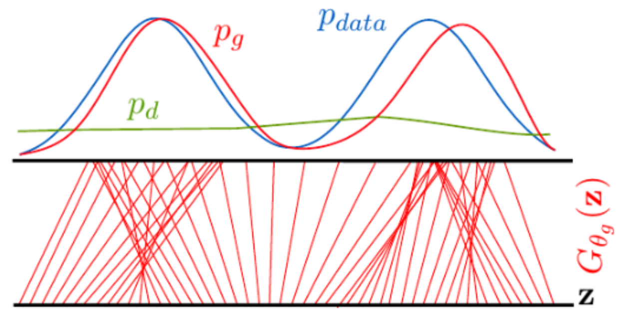
$$p_g \rightarrow p_{data}$$ if G and D have enough capacity

3.3. Training





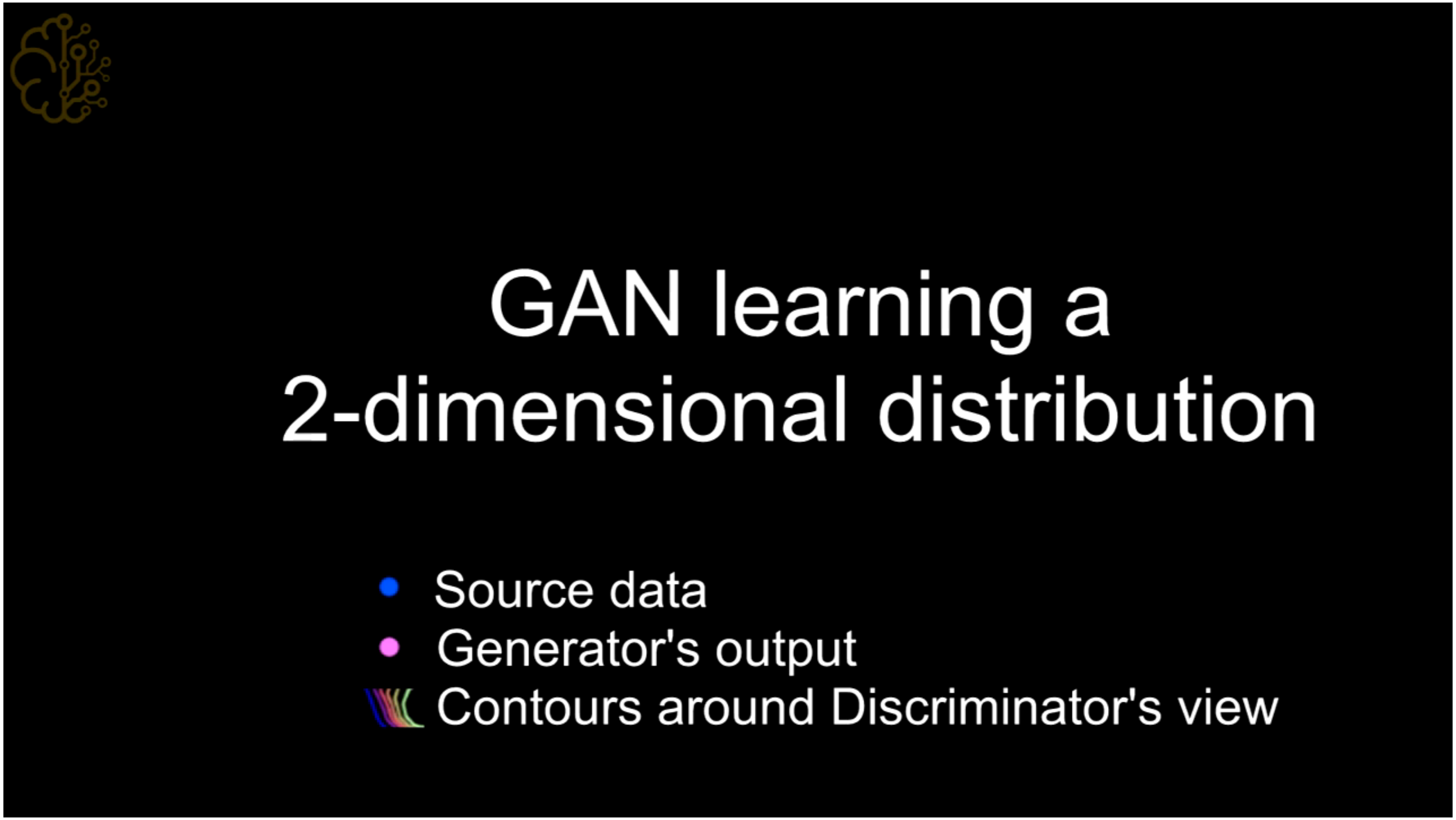
(b) Mapping function of Generator in step 1: The objective of the generator is to find the mapping function $\mathbf{z} \mapsto G_{\theta_g}(\mathbf{z})$ such that $p_g \approx p_d$. The objective of the discriminator is to distingush between the ground truth data and the generated samples, i.e. to find a distribution p_d that is non-uniform.



(c) Mapping function of Generator in step 5: The generator outputs almost similar samples to the ground truth data. The discriminator cannot distinguish between the two distributions anymore und its PDF becomes nearly uniform.

Adveraial Nets.html

GAN learning a 2D distribution:



https://www.youtube.com/watch?v=a1fjBkwRDY8

3.3. Deep Convolutional GAN (DCGAN)

The original paper (GAN) uses Fully connected layer to describe the generator and the discriminator

Drawbacks:

- ➔ very slow
- ➔ instable to train

Alternative:

Use convolutional layer to learn and evaluate only relative features (e.g. Deep Convolutional GAN (DCGAN) and all recent GANs) instead of using fully connected hidden layers

+ Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator)
+ Use batchnorm in both the generator and the discriminator
+ Use ReLU activation in generator for all layers except for the output, which uses Tanh
+ Use LeakyReLU activation in the discriminator for all layers

- ➔ faster
- ➔ much more stable to train

3.3. Deep Convolutional GAN (DCGAN)

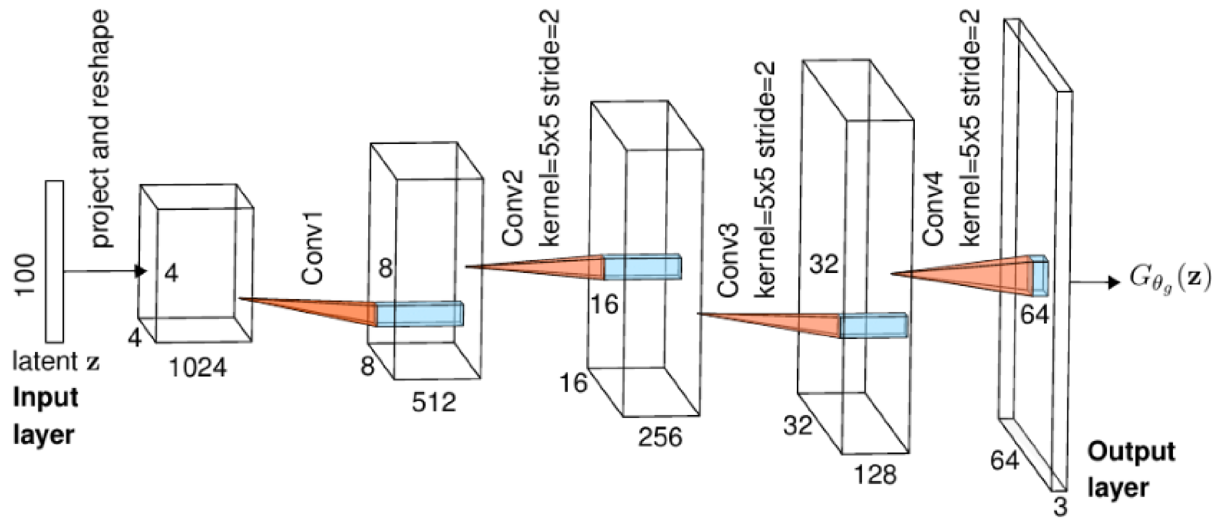


Figure 2.14.: Architecture of the generator in DCGAN

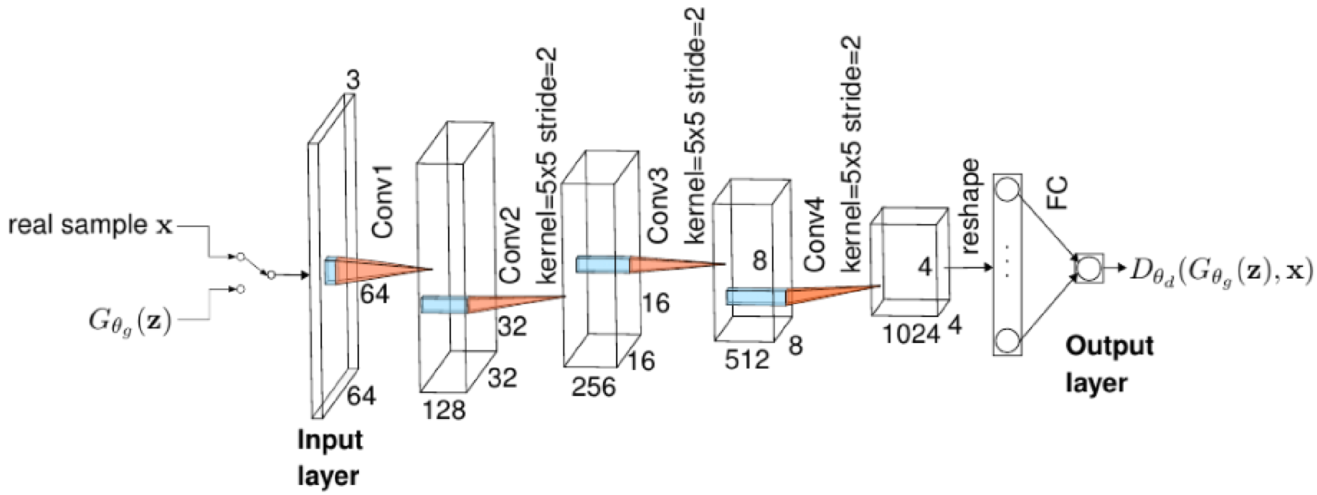


Figure 2.15.: Architecture of the discriminator in DCGAN

Pros:

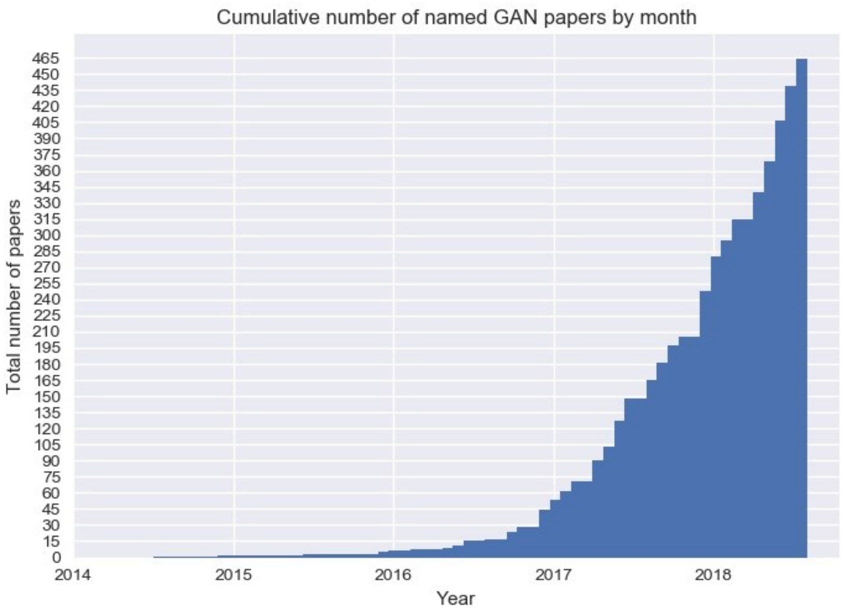
- Beautiful, state-of-the-art samples!

Cons:

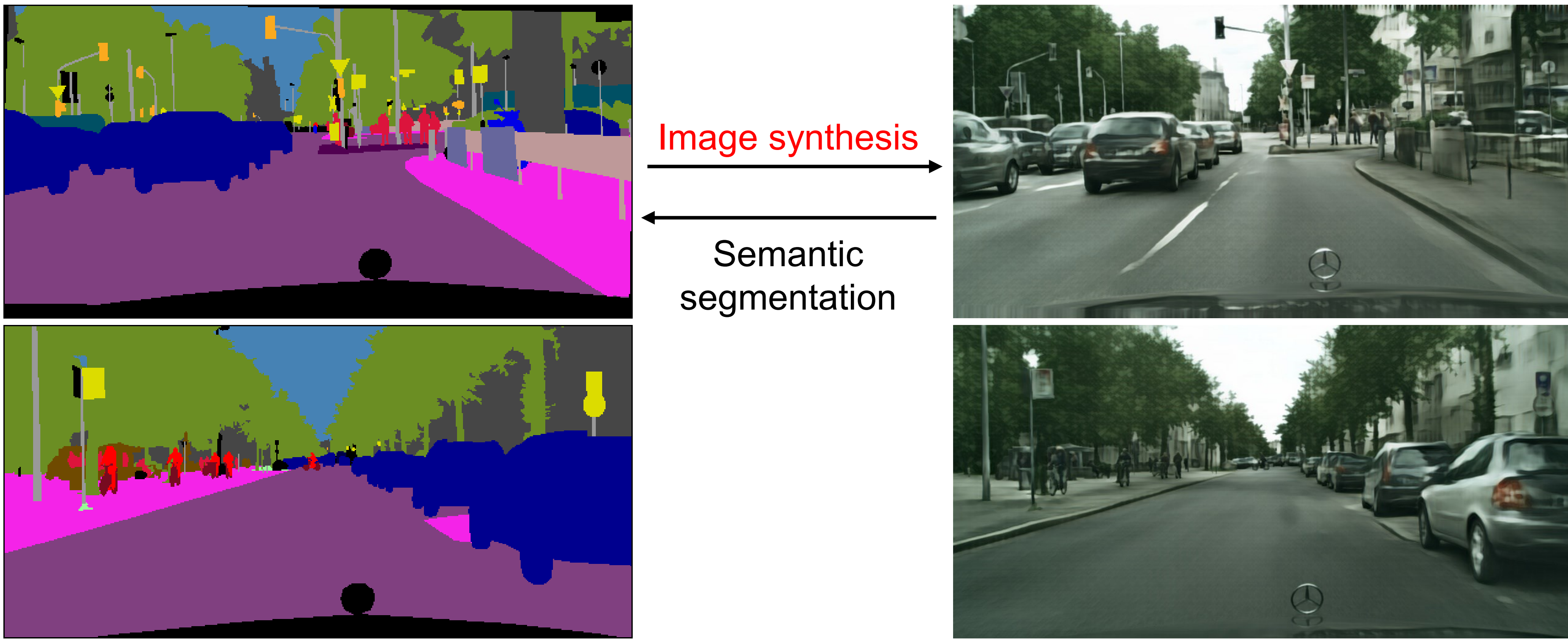
- Trickier / more unstable to train
- Hard to generate discrete data, like text

Improvement methods and active areas of research:

- Better loss functions to improve stability
(Wasserstein GAN)
- Novel architecture of the discriminator and/or generator
(e.g. Capsule GAN)
- Changing in the global structure of the GAN
(e.g. Muti-Generator GAN)



4. Photographic Image Synthesis



Input semantic layouts

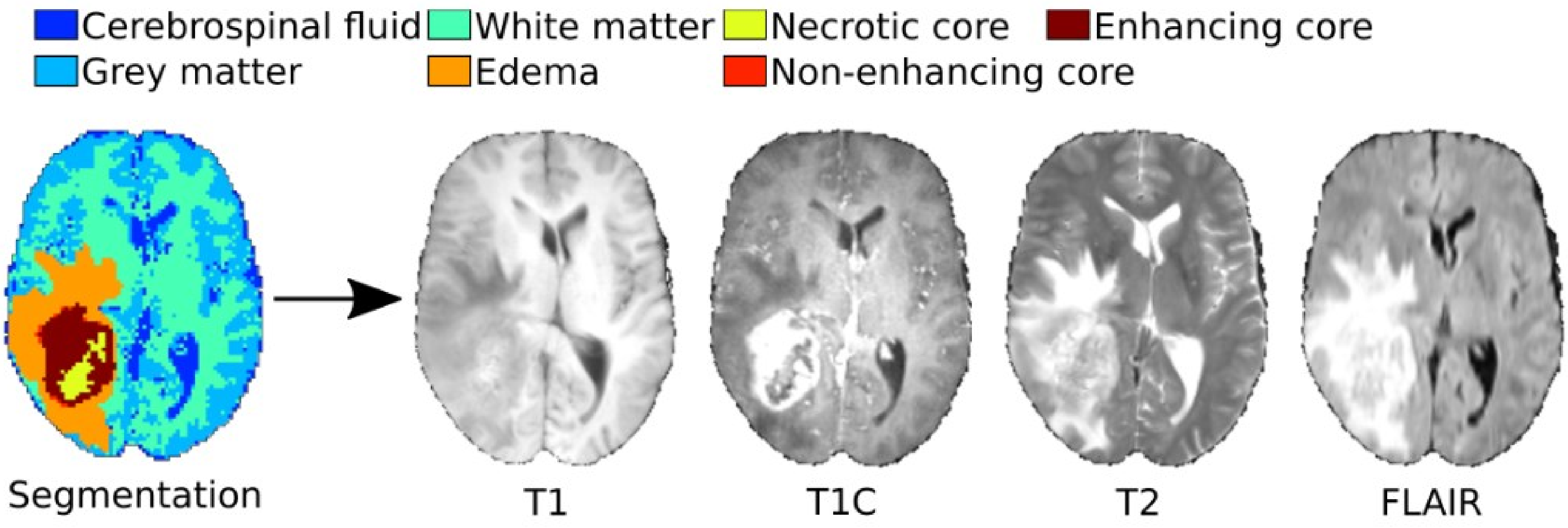
Synthesized images

4.1. Motivation

Computer graphics:

- Alternative route to photorealism
- Capture photographic apperance
- Fast image synthesis

4.1. Motivation



Medicine:

- Medical imaging: semantic labels ➔ MRI / CT /
  MRI / CT ➔ photographic image
- Data augmentation ??

4.2. Photographic Image Synthesis with Cascaded Refinement Networks (CRN)

- Cascaded Refinement Network (CRN)
- Perceptual loss
- Diversity (synthesis of a set of images)

Important characteristic for synthesizing photorealistic images:
- Global coordination (e.g. symmetry)
- High resolution (depending on the application)
- Memory/ high model capacity (generatlization)

CRN

A single refinement module in a CRN

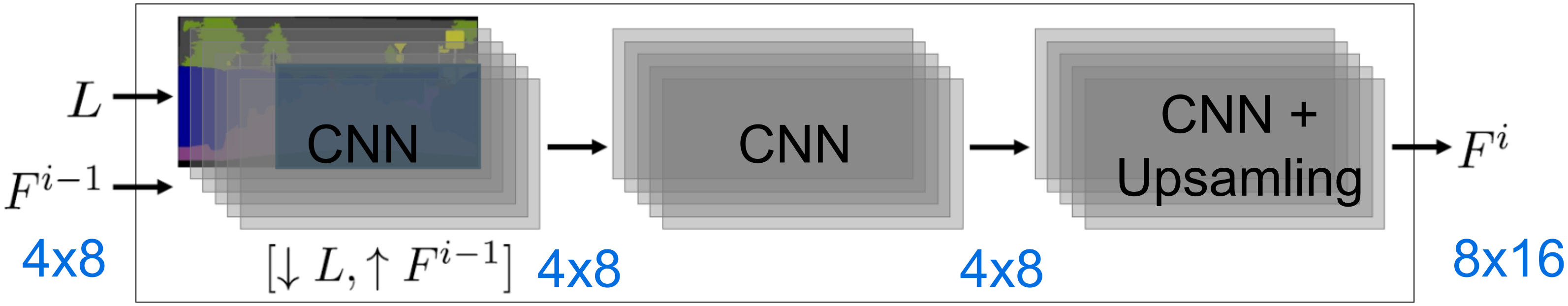


Figure 3. A single refinement module.

4.2. Photographic Image Synthesis with Cascaded Refinement Networks (CRN)

Perceptual Loss

$$\mathcal{L}_{I,L}(\theta) = \sum_{l} \lambda_l \|\Phi_l(I) - \Phi_l(g(L;\theta))\|_1 .$$

Match activation in a pretrained visual perception network VGG

Φ_l Activations of the layer l in the VGG netw Φk

I Ground truth image

g The mapping function performed by the CRN

λ_l hyperparameters in order to balance the contribution of each layer l to the loss

4.2. Photographic Image Synthesis with Cascaded Refinement Networks (CRN)

Results

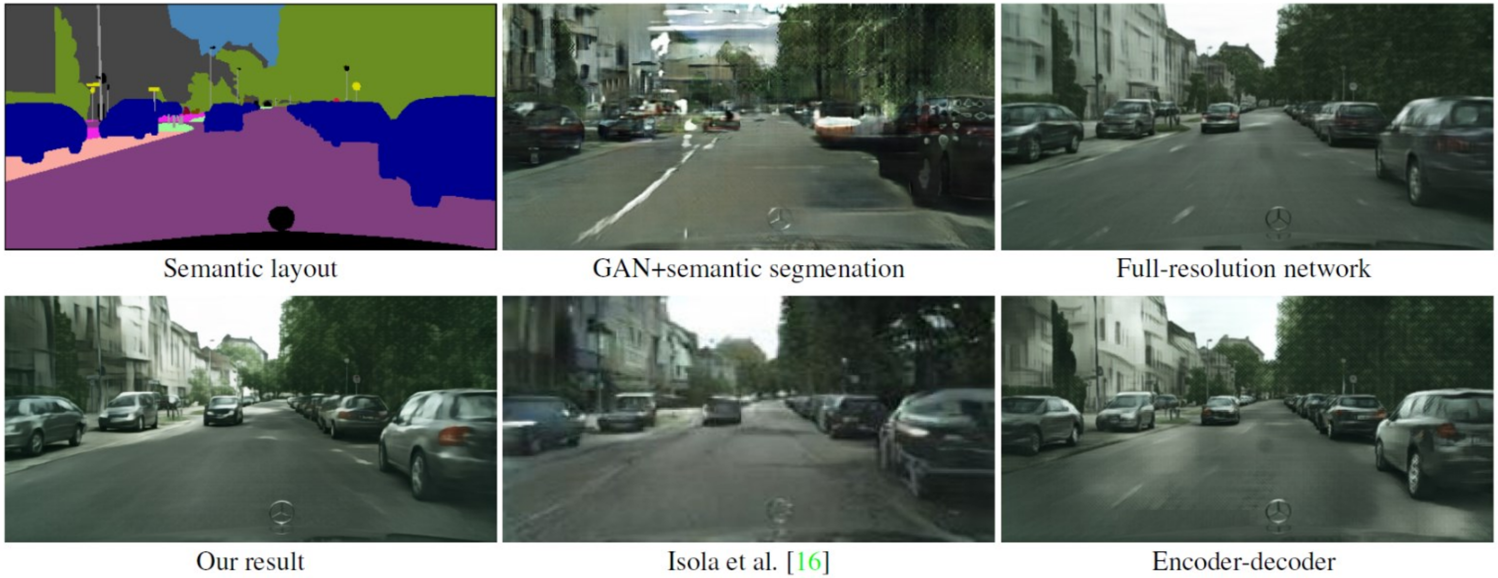


Figure 5. Qualitative comparison on the Cityscapes dataset.

Figure 6. Qualitative comparison on the NYU dataset.

An attempt to train a image synthesis system based on **GANs** was not successful

4. MR to CT Synthesis

4.1. Motivation

Similarities to computer vision:

- Object detection ➔ organ detection
- Object segmentation ➔ organ segmentation
- Object tracking ➔ orgran tracking

Challenges:

1. Images are often 3D or 4D ➔ dimenstionality reduction
2. Number of images for training is often limited
3. Training data is expensive (annotation of data by hand: manpower, cost, time)
4. Training data is sometimes imperfect (e.g. diseases such as Alzheimer‘s require confrmation through pathology: difficult and costly to obtain)

1. Learning the right features
2. Detecting when it goes wrong
3. Going beyond human-level performance

4.2. CT vs MR

	CT	MR
Basic principles of scanning	X-rays - slices	Magnetic field + radio waves Idenfigy hydrogen atoms
Harmfull radiation	Yes for long exposure	No
Type of tissues scanned	Tumors Lungs Brain	Ligaments Heart Liver Blood vessels

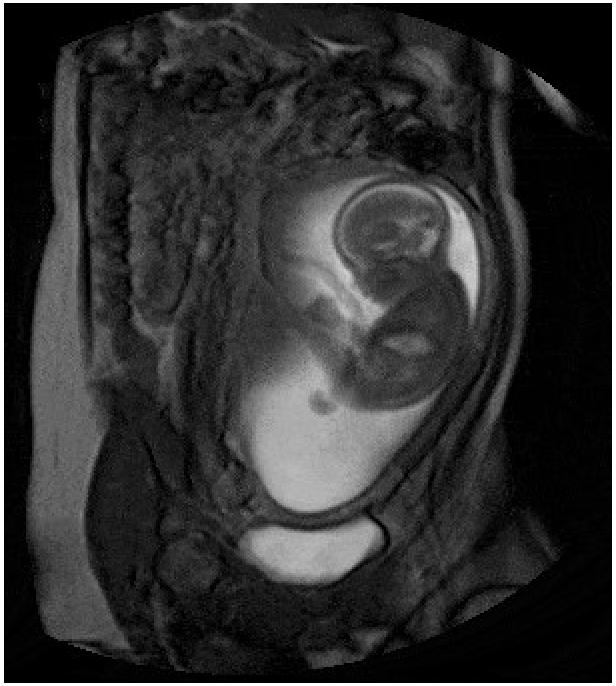
4.2. CT vs MR

	CT	MR
Noise	No noise	noisy
Time	Seconds ➔ Minutes	Minutes ➔ >Hours
Metallic implants	No impact	High impact
Cost	cheap	expensive

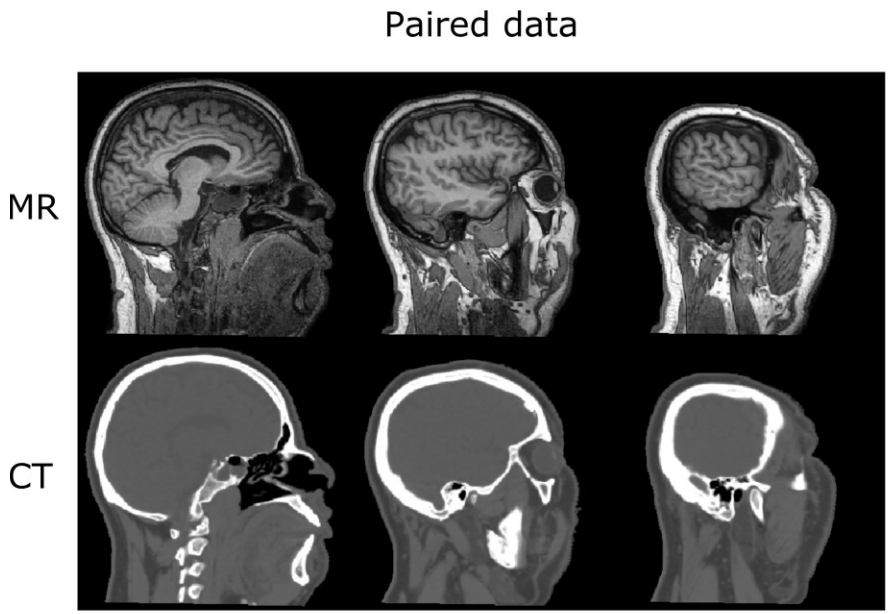
4.2. CT vs MR

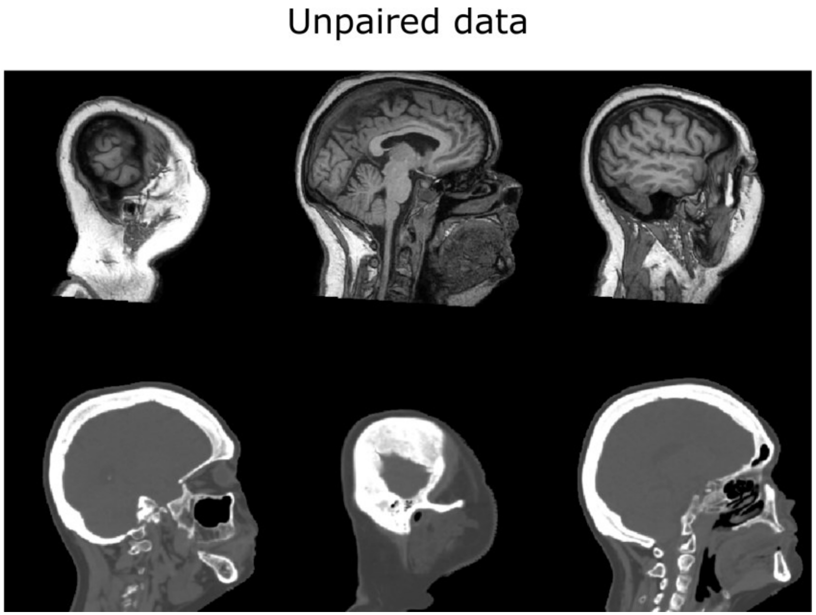
Challenges:

- 2D slices to 3D transformation:
  MR is problematic for moving objects
- CT can capture structures that MR is not able to.
  CT uses x-rays, which may harm the fetal
  ➔ **CT to MR**



Data

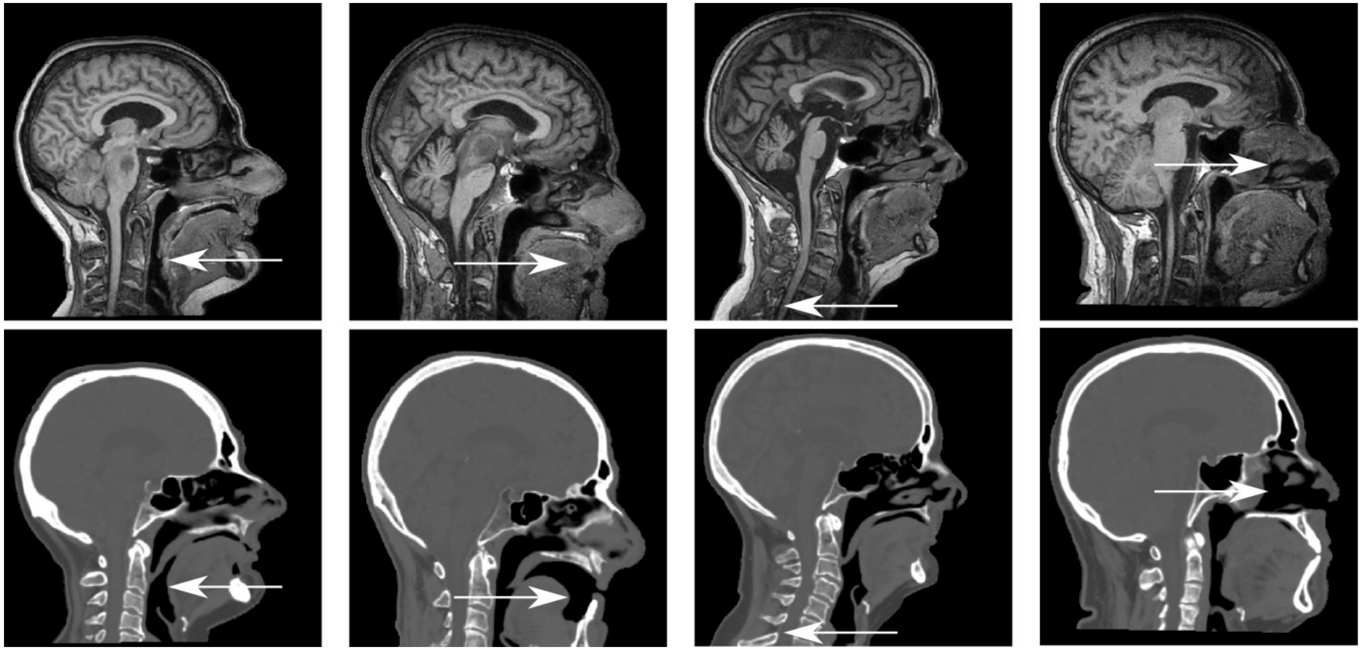




- same patient
- same anatomical location

- different patient
- different anatomical location in the brain

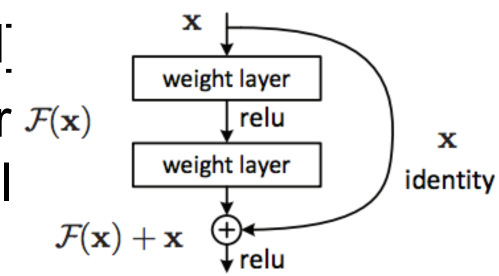
Local misalignment



- The skull is generally wellaligned
- Misalignments in the thorat, mouth, vertbrae and nasal cavities

Architecture

- CycleGAN (Zhu et al.)
- Consits of:
    - forward cycle (3 separate CNNs):
        - Syn_{CT}: I_{MR} ➔ Syn_{CT} (I_{MR})
        - Syn_{MR}: I_{CT} ➔ Syn_{MR}(I_{CT})
        - Dis_{CT}: [Syn_{CT} (I_{MR}), I_{CT}] ➔ [synthesized, real]
    - backward cycle (to improve training stability):
        - Syn_{MR}: I_{CT} ➔ Syn_{MR} (I_{CT})
        - Syn_{CT}: I_{MR} ➔ Syn_{CT}(I_{MR})
        - Dis_{MR}: [Syn_{MR} (I_{CT}), I_{MR}] ➔ [synthesized, real]
- Syn_{CT} and Syn_{MR} are identical: DeConvolutional Networ
  2D ConvLayers, strides=2x2, 9 *ResBlocks*, Upsaml
  Input: 256x256 image, output: 256x256 image
- Dis_{CT} and Dis_{MR} are identical: Convolutional Network
  2D ConvLayers
  input: overlapping 70x70 image patches, output: scalar (0 or 1)



Losses

On Dis_{CT}

$$\mathcal{L}_{CT} = (1 - Dis_{CT}(I_{CT}))^2 + Dis_{CT}(Syn_{CT}(I_{MR}))^2$$

On Dis_{MR}

$$\mathcal{L}_{MR} = (1 - Dis_{MR}(I_{MR}))^2 + Dis_{MR}(Syn_{MR}(I_{CT}))^2$$

On Syn_{CT} and on Syn_{MR}

Forward cycle:

$$L_{SynCT} = -Dis_{CT}(Syn_{CT}(I_{MR}))^2$$

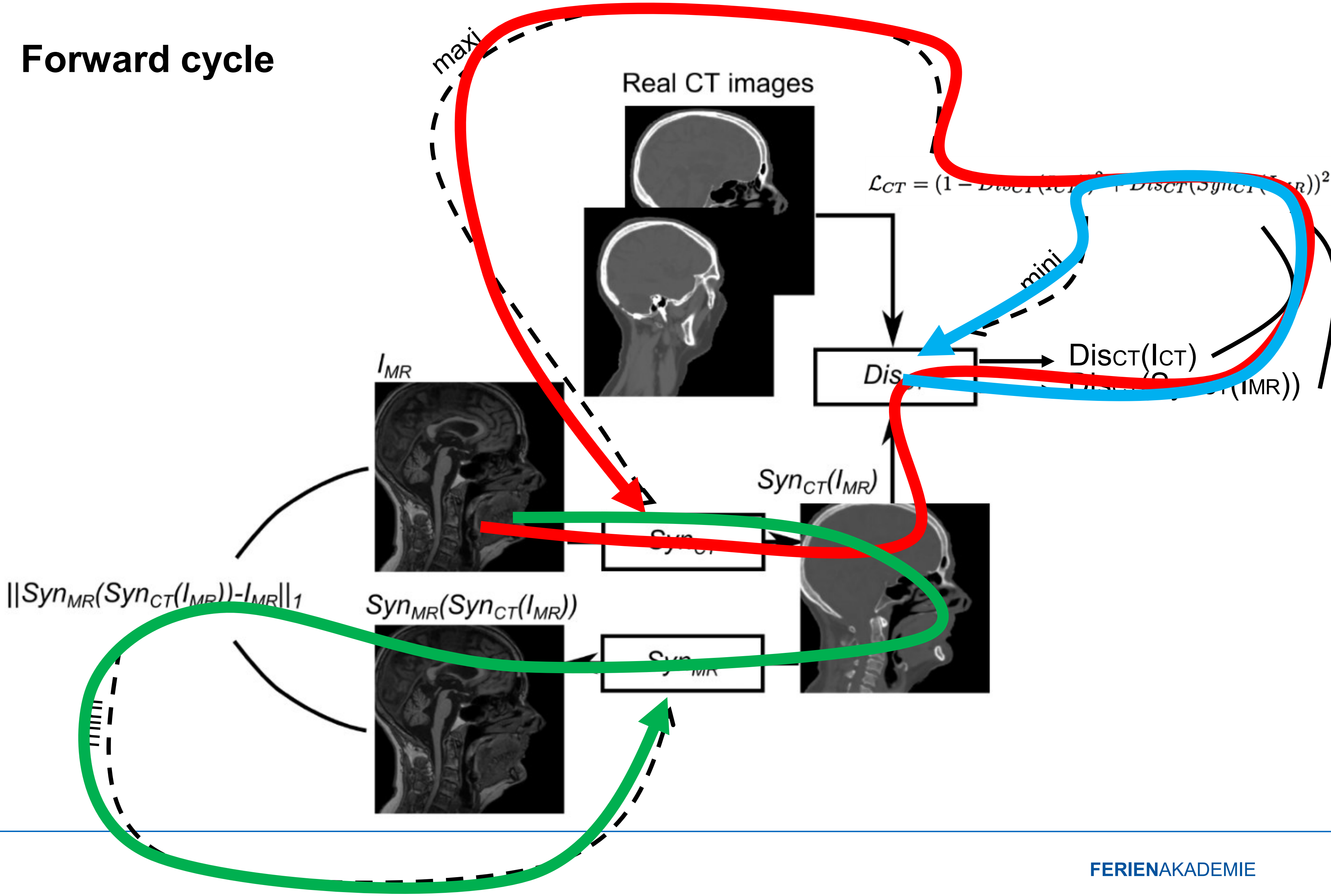
$$L_{SynMR} = -Dis_{MR}(Syn_{MR}(I_{CT}))^2$$

Backward cycle:

$$L_{SynCT} = -\lambda \, ||Syn_{CT}(Syn_{MR}(I_{CT})) - I_{CT}||_1$$

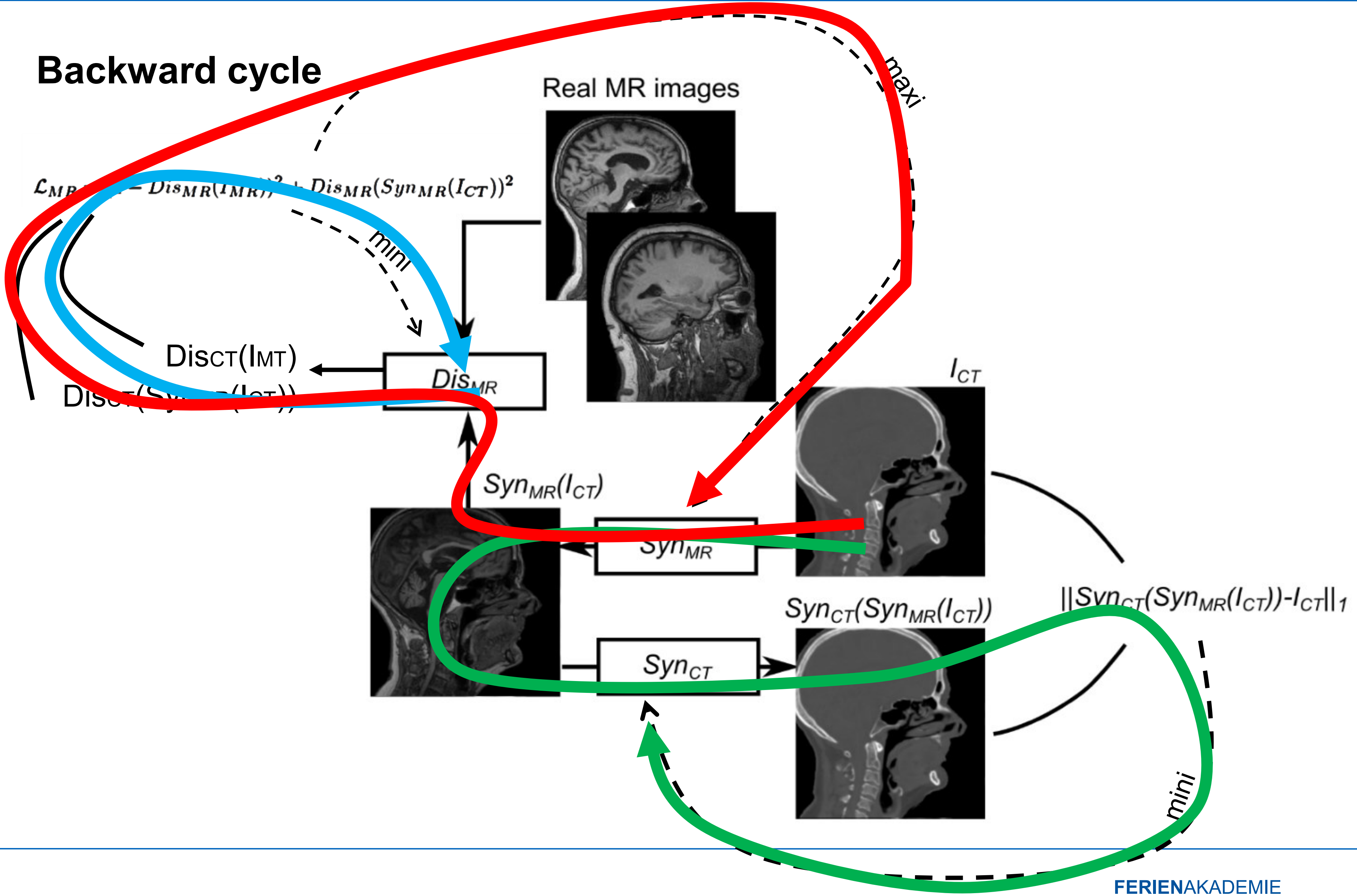
$$L_{SynMR} = -\lambda \, ||Syn_{MR}(Syn_{CT}(I_{MR})) - I_{MR}||_1$$

Forward cycle



4.4. Deep MR to CT Synthesis using Unpaired Data

Backward cycle



Evaluation

The mean absolute error

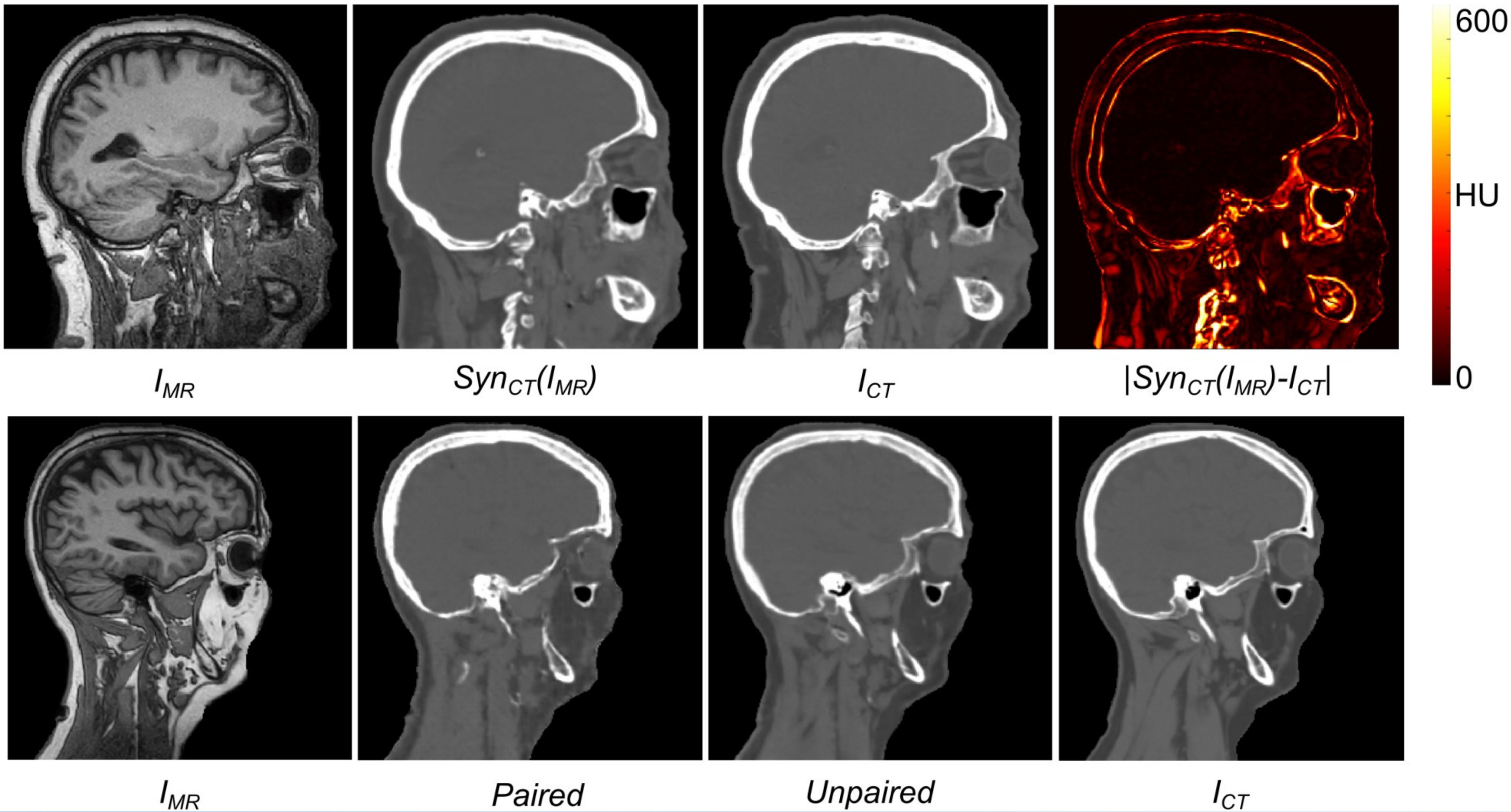
$$MAE = \frac{1}{N}\sum_{i=1}^{N} |I_{CT}(i) - Syn_{CT}(I_{MR}(i))|,$$

The peak-signal-to-noise-ratio

$$PSNR = 20\log_{10}\frac{4095}{MSE},$$

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(I_{CT}(i) - Syn_{CT}(I_{MR}(i))^2$$

Results


I_{MR}
$Syn_{CT}(I_{MR})$
I_{CT}
$|Syn_{CT}(I_{MR})-I_{CT}|$
600
HU
0
I_{MR}
Paired
Unpaired
I_{CT}

Results

	MAE		PSNR	
	Unpaired	Paired	Unpaired	Paired
Patient 1	70.3	86.2	31.1	29.3
Patient 2	76.2	98.8	32.1	30.1
Patient 3	75.5	96.9	32.9	30.1
Patient 4	75.2	86.0	32.9	31.7
Patient 5	72.0	81.7	32.3	31.2
Patient 6	73.0	87.0	32.5	30.9
Average $\pm$ SD	73.7 ± 2.3	89.4 ± 6.8	32.3 ± 0.7	30.6 ± 0.9

Notes

Tricks that help the network learning a generalization :

- Unpaired data (because the network was trained with random unpaired data).
- Images fed into the discriminator are randomly cropped : cancels the effects of rigid registration.

Limitation:

- using images of the same patients in the MR set and the CT set may affect training.

Ref.

[1] Generative Adversarial Networks - Ian Goodfellow – Jun 2014 - arXiv:1406.2661

[2] Photographic Image Synthesis with Cascaded Refinement Networks - Qifeng Chen et al. - Jul 2017 - arXiv:1707.09405

[3] Deep MR to CT Synthesis using Unpaired Data - Jelmer M. Wolterink and Anna M. Dinkla and Mark H. F. Savenije and Peter R. Seevinck and Cornelis A. T. van den Berg and Ivana Isgum - Aug 2017 – arXiv:1708.01155

[4] Extended Modality Propagation: Image Synthesis of Pathological Cases. Cordier N, Delingette H, Le M, Ayache N. – Jul 2016 - IEEE Trans Med Imaging

[5] Lecture 13 | Generative Models - Stanford University School of Engineering

[6] Novel approach for generative modelling using capsule generative adversarial networks – Oussema Dhaouadi – BSc. Thesis – LDV & IN6 TUM