Deep learning for medical imaging

Olivier Colliot, PhD
Research Director at CNRS
Co-Head of the ARAMIS Lab – www.aramislab.fr
PRAIRIE – Paris Artificial Intelligence Research Institute

Maria Vakalopoulou, PhD
Assistant Professor at CentraleSupelec
Center for Visual Computing

Master 2 - MVA

Course website: http://www.aramislab.fr/teaching/DLMI-2019-2020/
Piazza (for registered students): https://piazza.com/centralesupelec/spring2020/mvadlmi/

Part 7 – Generative models

Part 7 – Generative models

7.1 Introduction

Introduction – Image generation

https://thispersondoesnotexist.com/

Introduction – Image generation

https://thispersondoesnotexist.com/

Introduction – Image generation

https://thispersondoesnotexist.com/

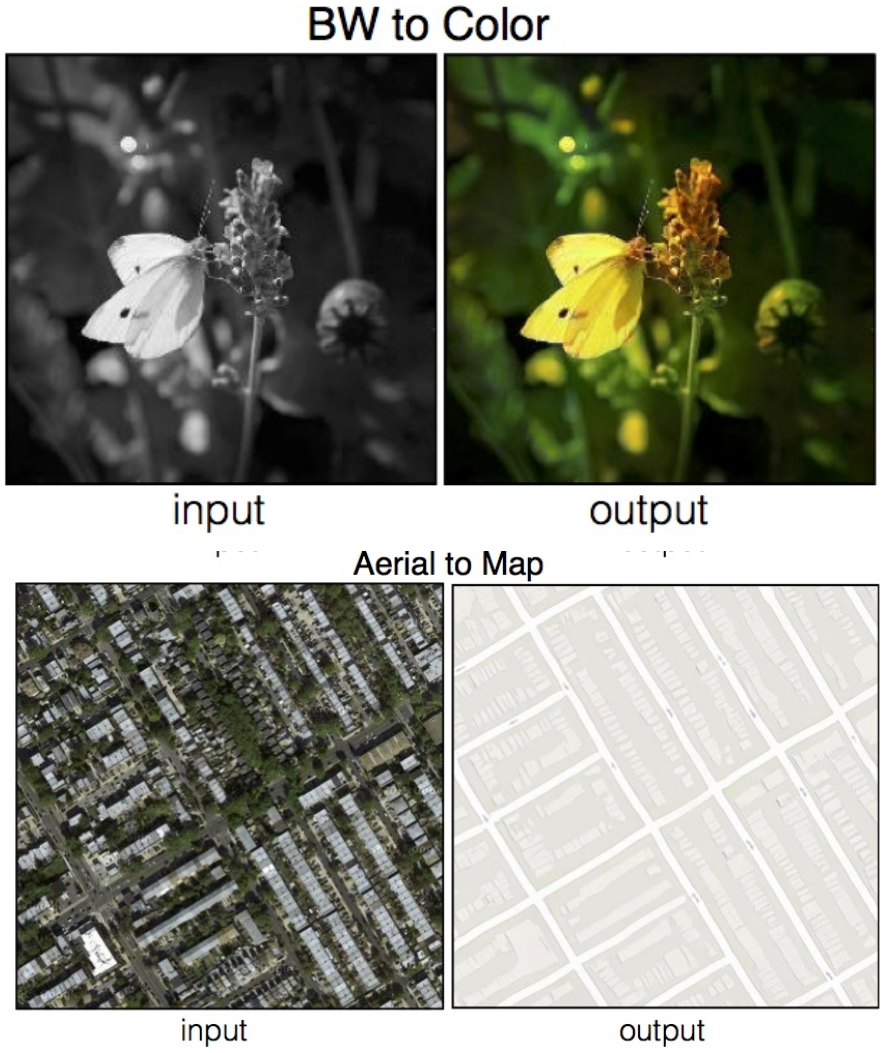
Introduction – Image generation

https://thispersondoesnotexist.com/

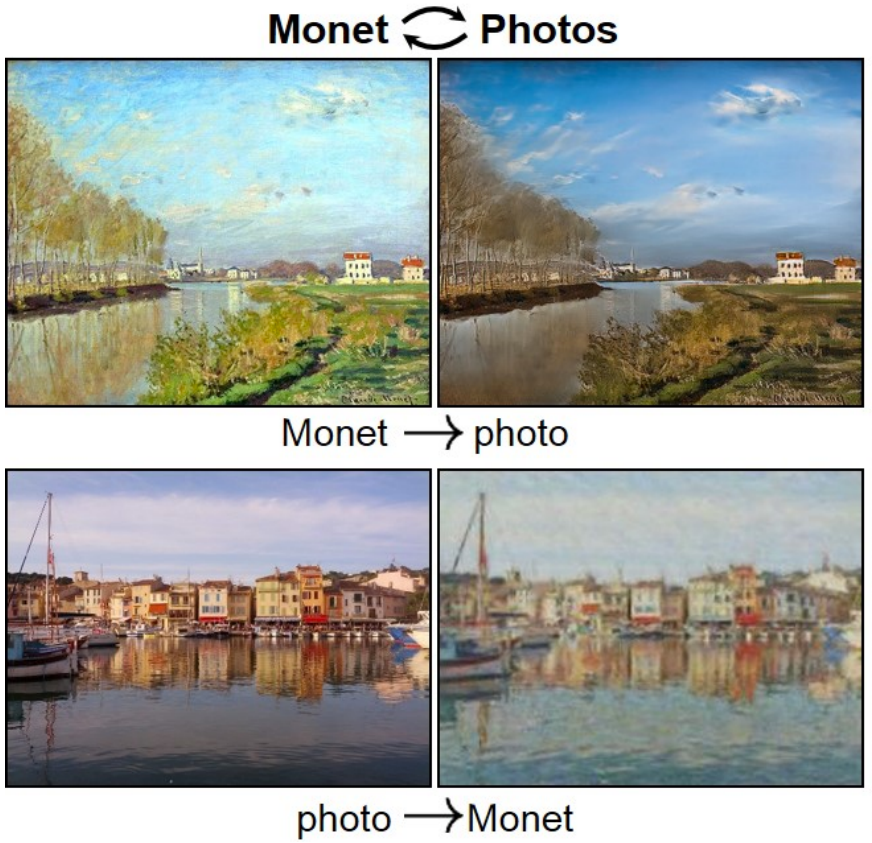
Introduction – Image generation

Karras et al, A Style-Based Generator Architecture for Generative Adversarial Networks, CVPR 2019

Introduction - Image translation



(Isola et al, Proc. CVPR 2017)



(Zhu et al, Proc. ICCV 2017)

Monet ⇄ Photos

Monet → photo

photo → Monet

Introduction

Supervised learning

x: data
y: label (class, continuous, other…)

Learn p(y|x)

Examples: classification, regression, detection, segmentation

Unsupervised learning

x: data

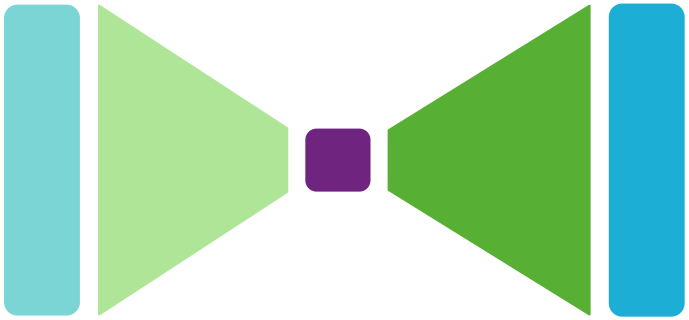
Learn p(x)

Examples: dimensionality reduction, generation of new samples

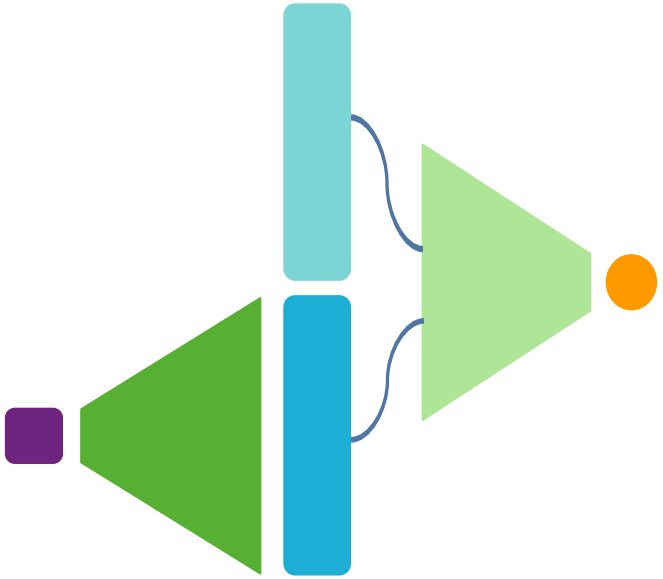
Inspired from: http://introtodeeplearning.com/

Course content

Recap on autoencoders

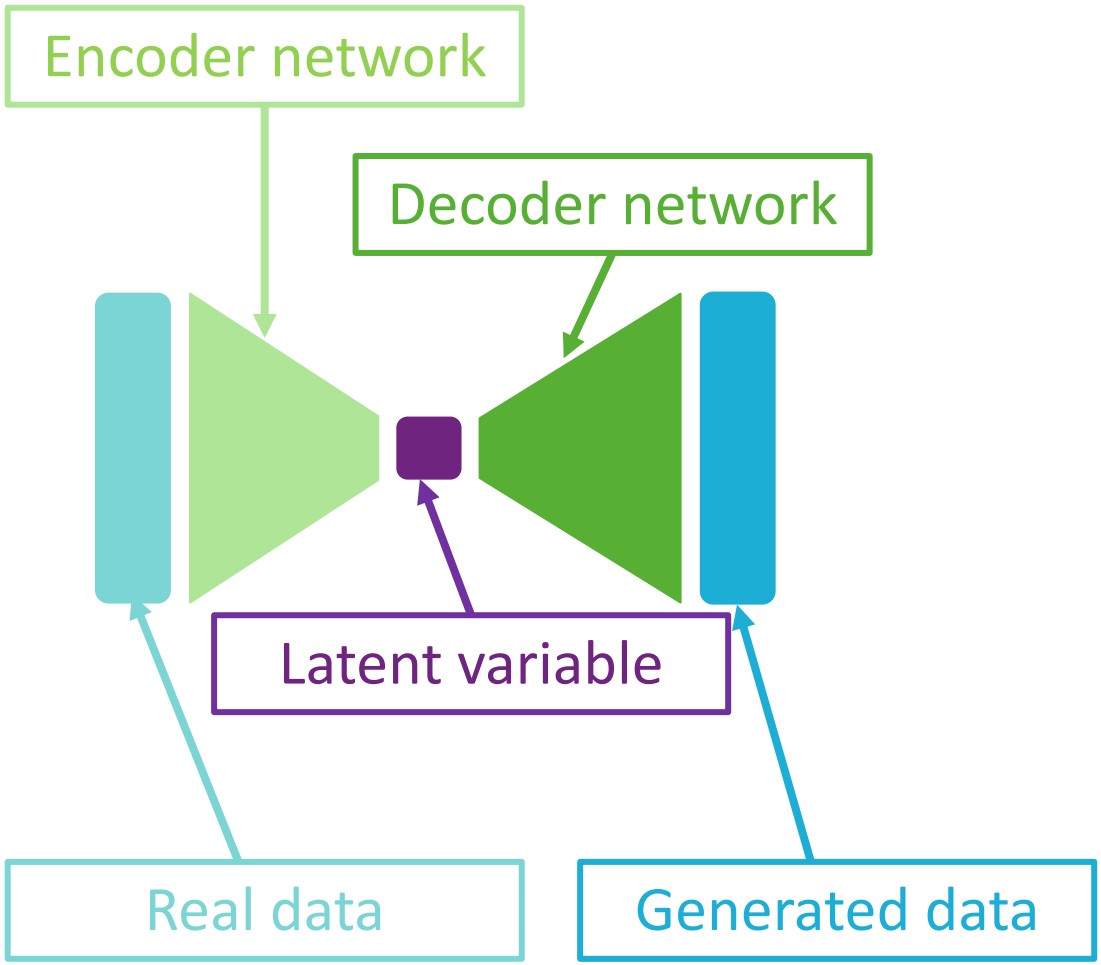


Generative adversarial networks (GANs)

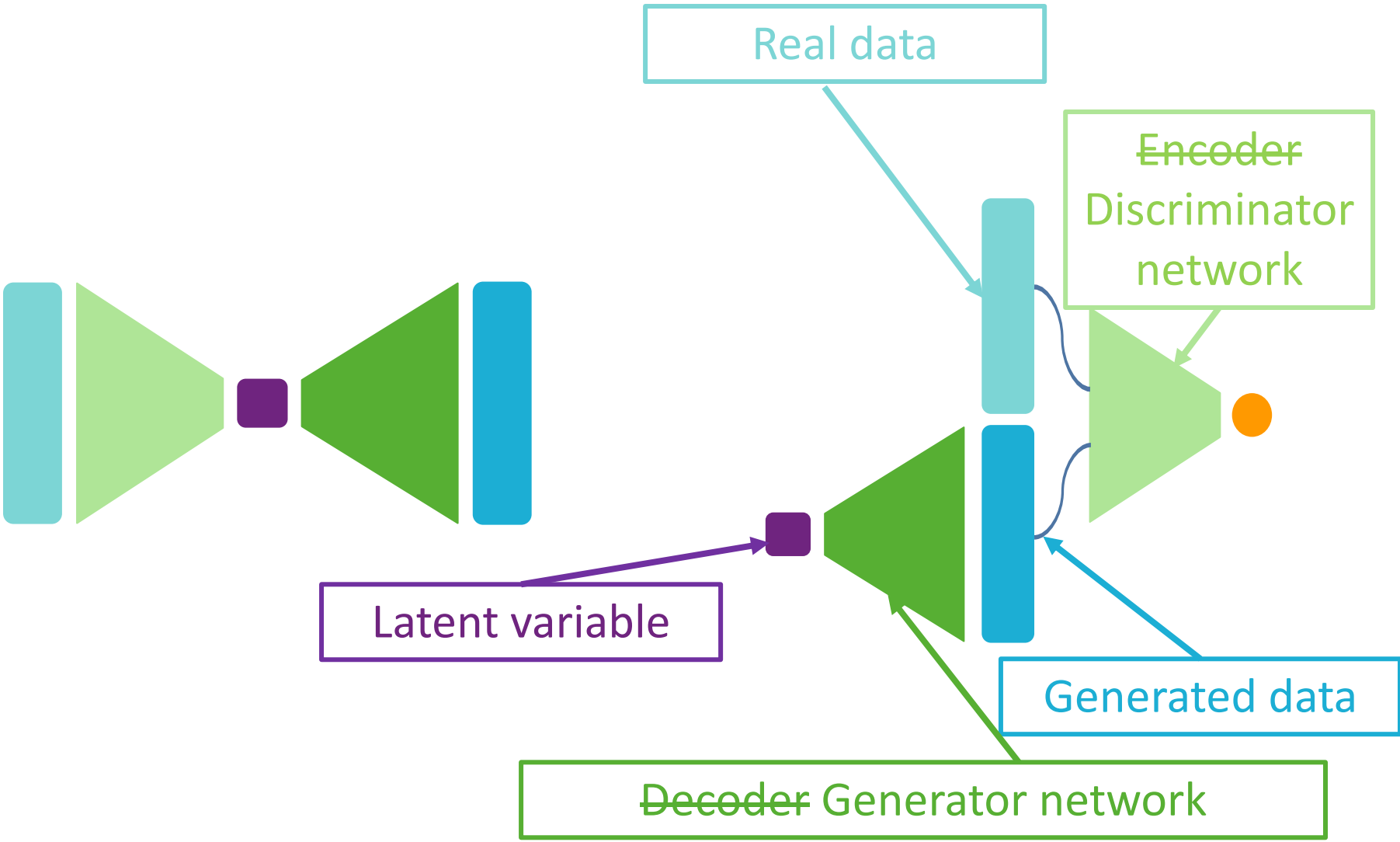


Inspired from: http://introtodeeplearning.com/

Course content



Course content



What is a latent variable?

Wikipedia definition

Latent variables (from Latin: present participle of lateo (“lie hidden”), as opposed to **observable variables**) are variables that are not directly observed but are rather **inferred** (through a mathematical model) **from other variables that are observed** (directly measured).

Source: https://en.wikipedia.org/wiki/Latent_variable

What is a latent variable?

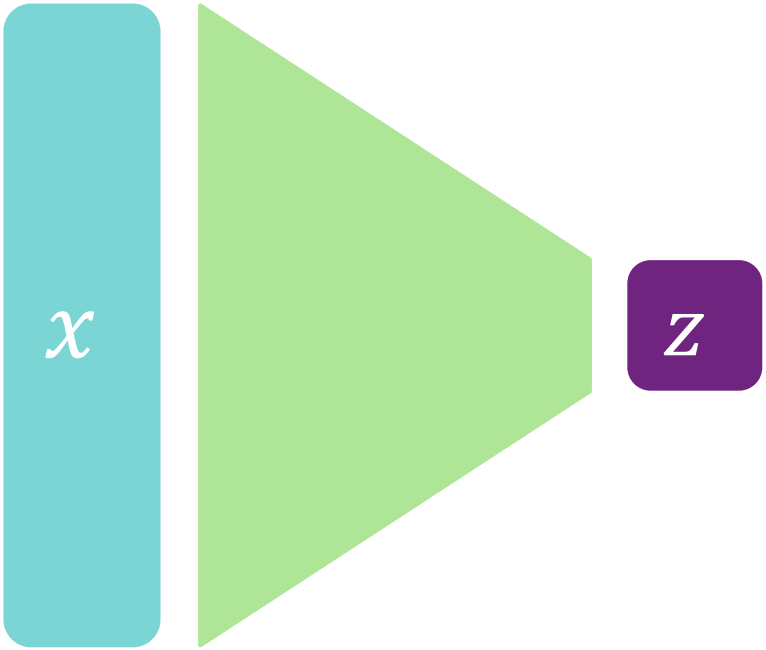
Can we learn the true explanatory factors (latent variables) from observed data?

Part 7 – Generative models

7.2 Recap on autoencoders

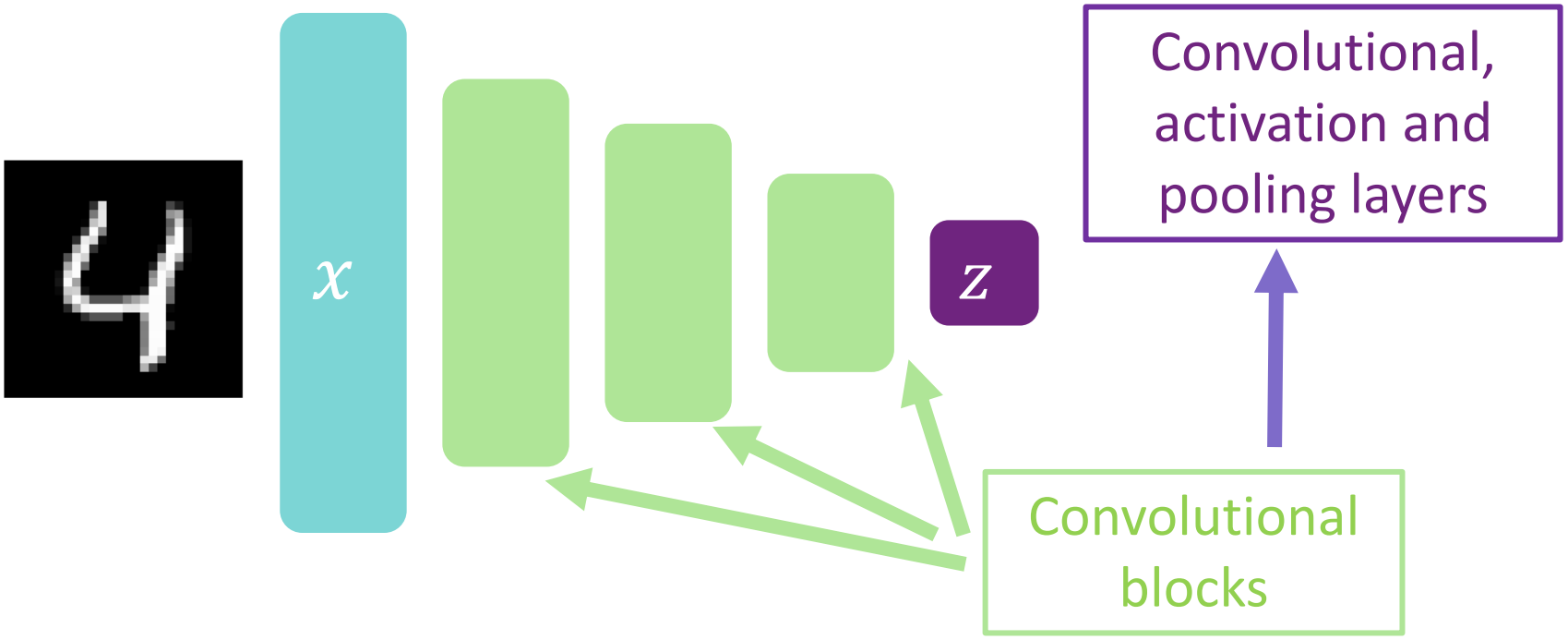
Autoencoders

- **Autoencoder:** technique for **learning a latent space representation**, denoted by z
- The latent space representation is of lower dimension
- The part that learns the mapping from x to z is called an **encoder**



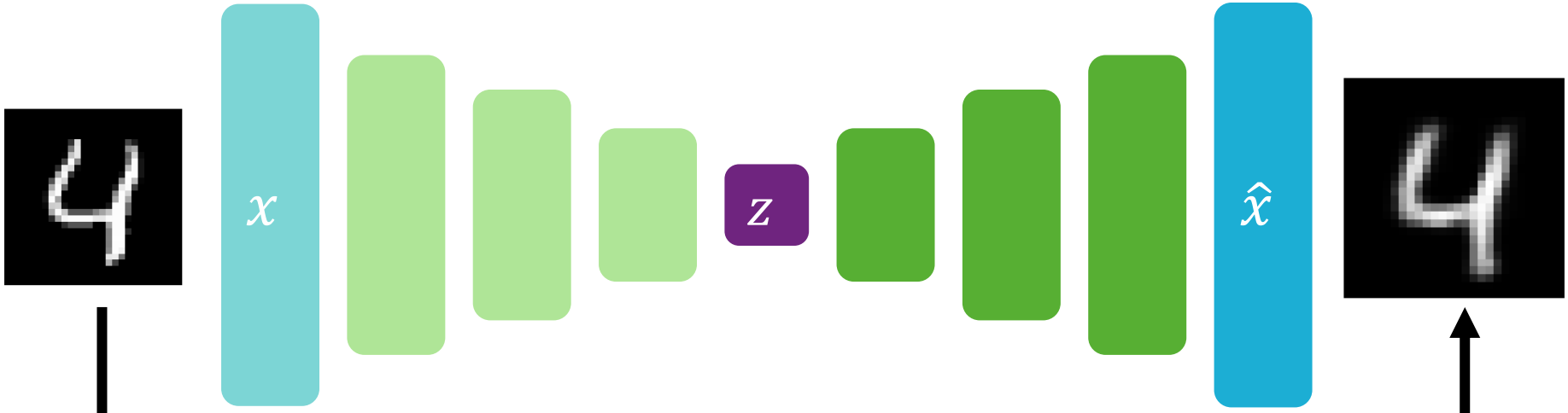
Autoencoders

- When dealing with images, the autoencoder is **most often a CNN**
- The encoder part is then made of contracting **convolutional blocks**



Autoencoders

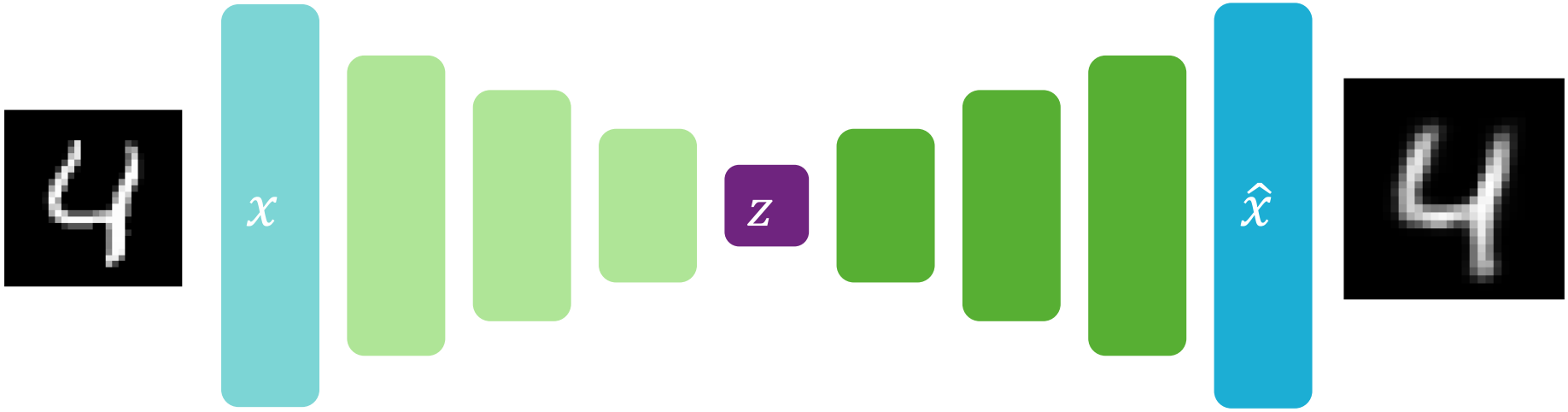
- To learn the latent space, the autoencoder is trained to **reconstruct the original data**
- The part that reconstructs the data from the latent representation is called **the decoder**
- The loss can for instance be the l2 norm



$$\mathcal{L}(x, \hat{x}) = \|x - \hat{x}\|^2$$

Autoencoders

- Again, when dealing with images, we often use a **CNN architecture**
- What do we need for the decoder?
  - **Upsampling** (remember from segmentation course?)
    - Unpooling
    - Max Unpooling
    - Transpose convolutions



Autoencoder

Example for varying dimensionality of the latent space

2D latent space

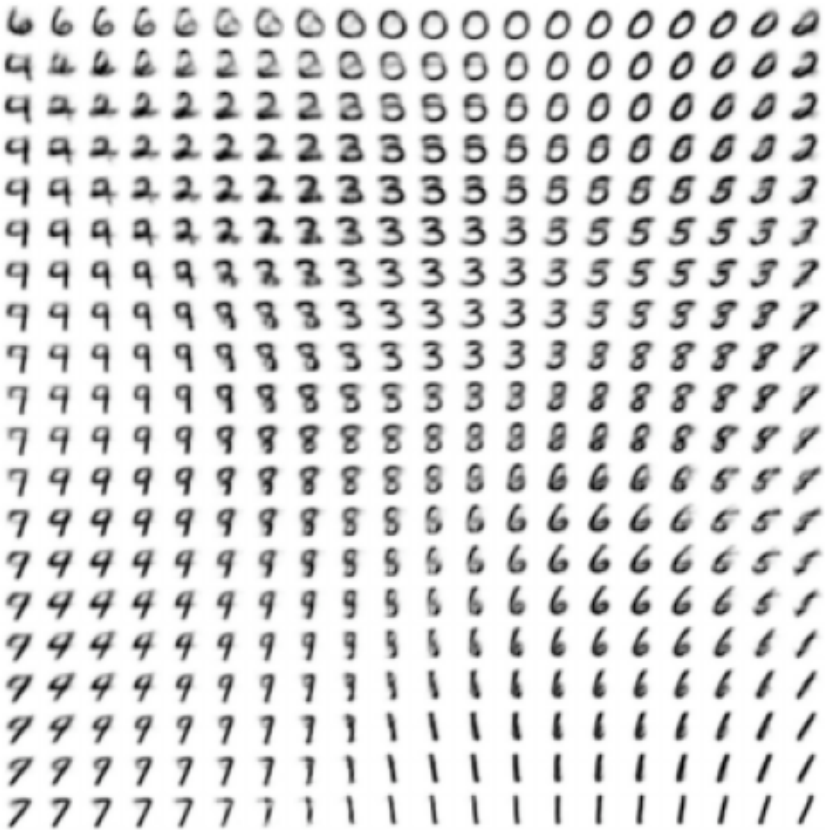
5D latent space

Ground Truth

Source: http://introtodeeplearning.com/

Autoencoder

Example of 2D latent space learned by an autoencoder



This is is in fact produced by a variational autoencoder

Source: Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR). 689, 700

Autoencoder

Example of 2D latent space learned by an autoencoder

This is is in fact produced by a variational autoencoder

Source: Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In Proceedings of the International Conference on Learning Representations (ICLR). 689, 700

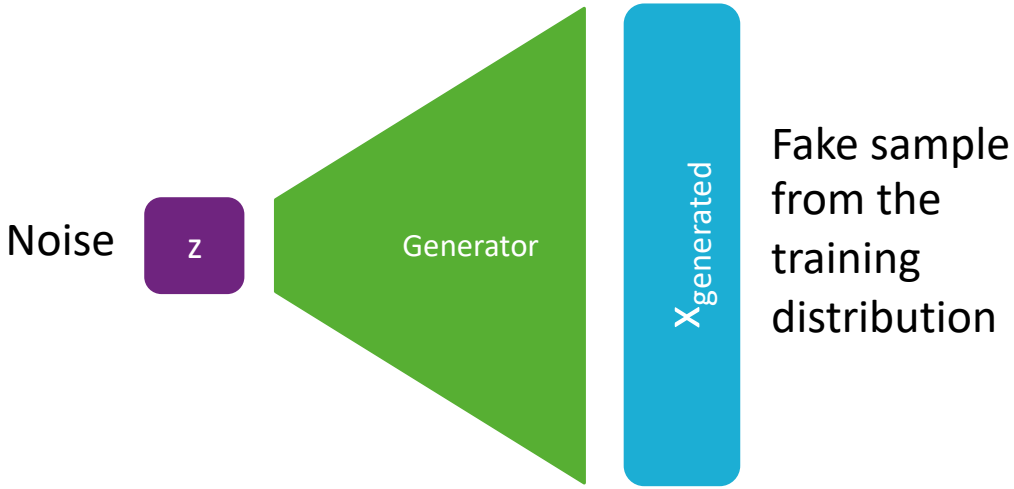
Part 7 – Generative models

7.3 GANs

GANs

What if we just want to generate samples?

- without modeling the distribution or using the learned latent space
- **Idea:** sample from noise and learn a transformation to the training distribution



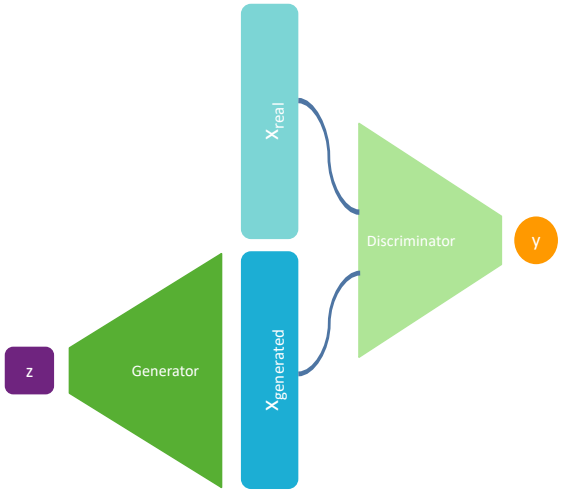
Inspired from: http://introtodeeplearning.com/

GANs

Objective function

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

This is a two-player minimax game.



Reference: Goodfellow et al, Generative Adversarial Nets, NeurIPS, 2014

GANs

Training algorithm

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Source: Goodfellow et al, Generative Adversarial Nets, NeurIPS, 2014

Part 7 – Generative models

7.4 Medical image translation with cGANs

Image translation

- **Motivation**
  - Image enhancement
    - Low dose to full dose
    - Super resolution
  - Cross-modality synthesis
    - MRI to CT
    - MRI to PET

- **Setting**
  - Learn a mapping from c (image) to x (also image)
  - Training data can be
    - paired, i.e. we have a set of couples (ci,xi) where i is the index of a given patient
    - or not, i.e. we have set of input data ci and a set of output data xj but not in the same patient

cGAN

Conditional GAN (cGAN)

- Learns a mapping from input c to output x
- Leans a conditional distribution p(x|c) instead of a marginal distribution p(x)

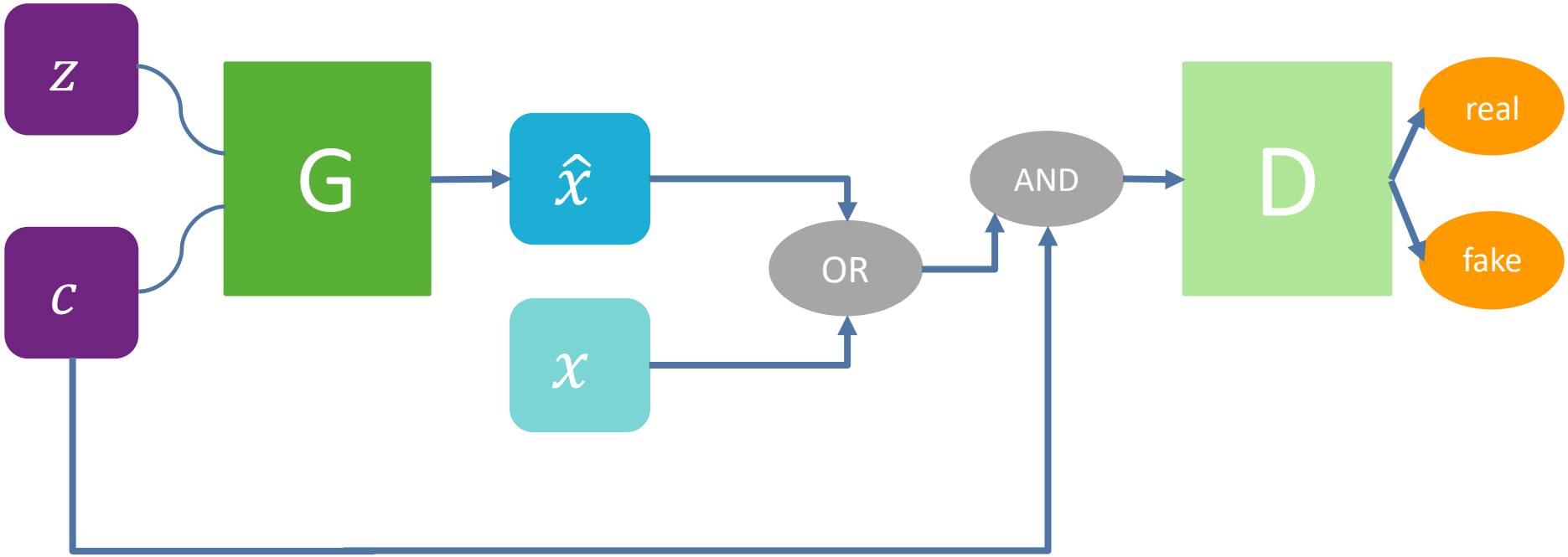
$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[log(D(x|c))] + \mathbb{E}_{z\sim p_z(z)}[1 - log(D(G(z|c)))]$$

- c can be of different types
  - In the original cGAN paper (Mizra and Osindero, 2014), c is a class
  - It will be an image in image translation
- Requires paired data

Reference: Mirza and Osindero, Conditional Generative Adversarial Nets, ArXiv, 2014

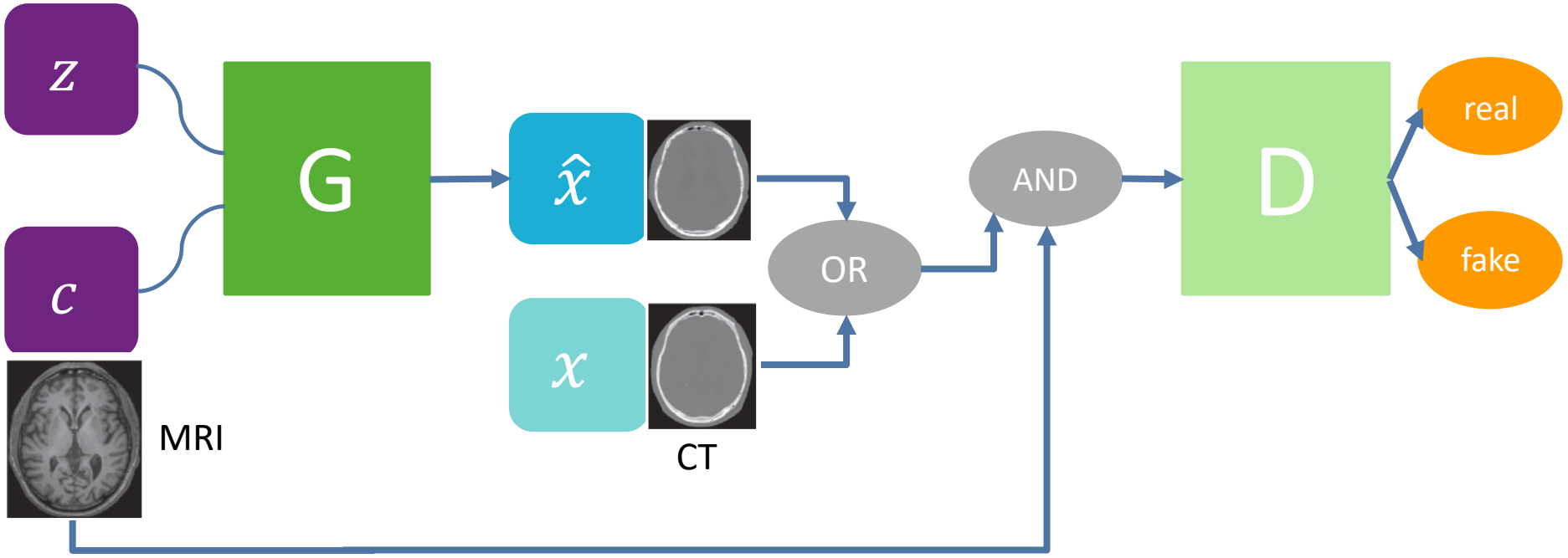
cGAN

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



cGAN

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



cGAN – adding a generator loss

- **It is often beneficial to add a loss to the generator, directly comparing generated and real data**
  - Same role as reconstruction loss in autoencoders
  - L2 loss (Pathak et al, 2016)
  - L1 loss (Isola et al, 2017 – pix2pix method)
    - Avoids blurring of the generated samples

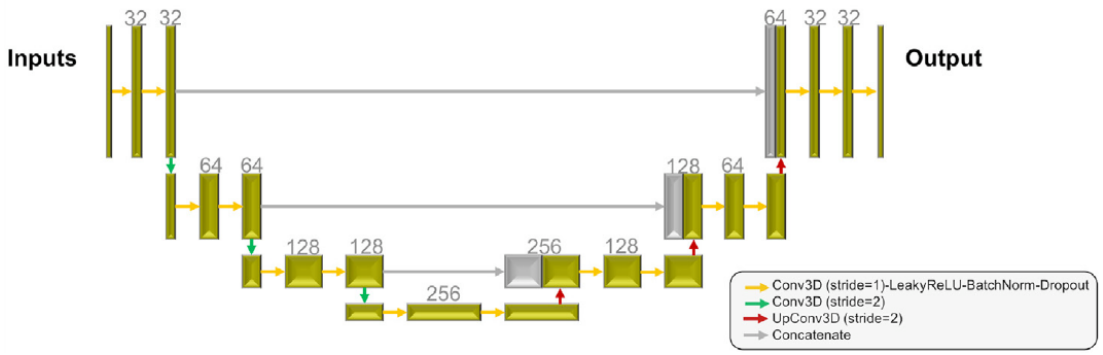
$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x,z)\|_1].$$

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G,D) + \lambda \mathcal{L}_{L1}(G)$$

References:
- D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In CVPR, 2016
- Isola, P., Zhu, J.-Y., Zhou, T., Efros, A . A . Image-to-image translation with con- ditional adversarial networks. CVPR 2017

Architectures

- **Typical architectures for the generator**



U-net

e.g. in (Isola et al, CVPR 2017; Wei et al, Medical Image Analysis, 2019; Wang et al, NeuroImage, 2018…)

Better than autoencoder for reconstructing details

Fully convolutional network (no pooling layers)

e.g. in (Wolterink et al, IEEE TMI, 2017; Wolterink et al, SASHIMI 2017; Nie et al, 2017…)

Architectures

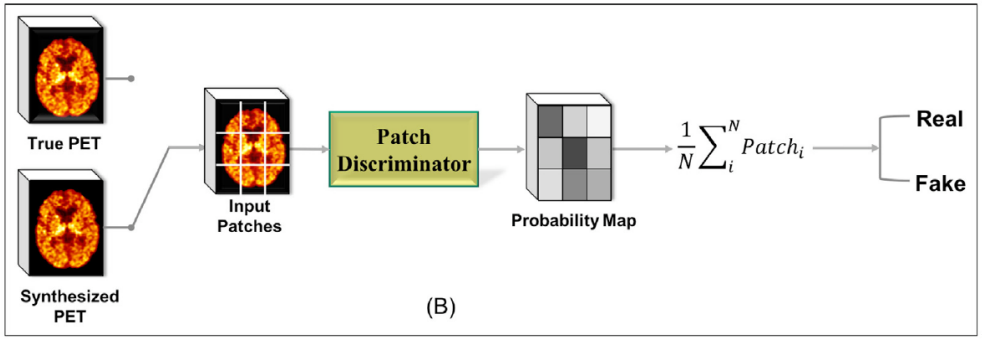
- **Typical architectures for the discriminator**

Classical CNN (full image)

e.g. in (Nie et al, MICCAI 2017;

Patch-based CNN

e.g. in (Wolterink et al, SASHIMI 2017; Wei et al, Medical Image Analysis, 2019…)



The CNN can better focus on high frequency information that may distinguish real from synthesized images.

Is it worth being adversarial?

- **Example 1: MRI-to-CT**
  - Comparison of FCN and GAN
  - Here the FCN architecture is the same as that of the GAN generator

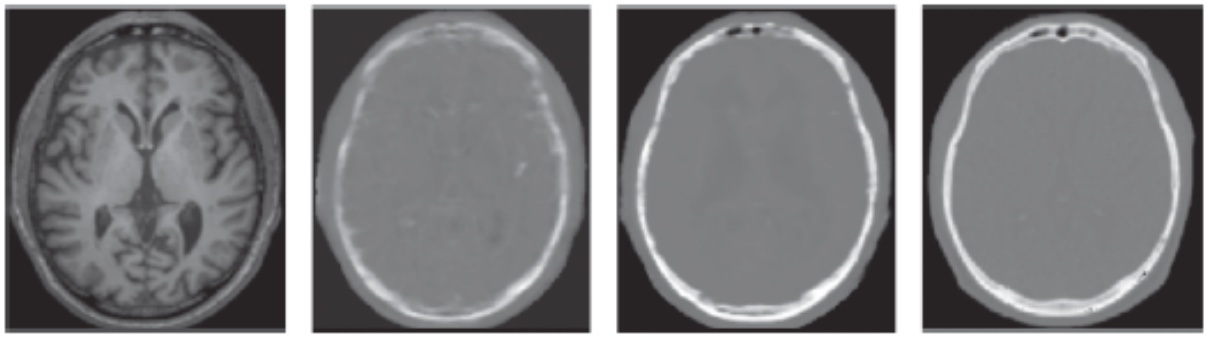


Fig. 3. Visual comparison for impact of adversarial training. FCN means the case without adversarial training, and GAN means the case with adversarial training.

Reference:
- Nie et al, Medical Image Synthesis with Context-Aware Generative Adversarial Networks, MICCAI 2017

Is it worth being adversarial?

- **Example 1: MRI-to-CT**
  - Comparison of GAN and different approaches (atlas-based, sparse representation, structured random forest…)

Table 1. Performances on the brain dataset.

Method	MAE		PSNR	
	Mean(std.)	Med.	Mean(std.)	Med.
Atlas	171.5(35.7)	170.2	20.8(1.6)	20.6
SR	159.8(37.4)	161.1	21.3(1.7)	21.2
SRF+	99.9(14.2)	97.6	26.3(1.4)	26.3
Proposed	**92.5(13.9)**	**92.1**	**27.6(1.3)**	**27.6**

Table 2. Performances on the pelvic dataset.

Method	MAE		PSNR	
	Mean(std.)	Med.	Mean(std.)	Med.
Atlas	66.1(6.9)	66.7	29.0(2.1)	29.6
SR	52.1(9.8)	52.3	30.3(2.6)	31.1
SRF+	48.1(4.6)	48.3	32.1(0.9)	31.8
Proposed	**39.0(4.6)**	**39.1**	**34.1(1.0)**	**34.1**

Reference:
- Nie et al, Medical Image Synthesis with Context-Aware Generative Adversarial Networks, MICCAI 2017

Is it worth being adversarial?

- **Example 2: MRI-to-PET**
  - Comparison of 3 GANs to a U-net and a FCN

Table 1
Image quality metrics obtained with our method and the other methods. MSE: mean square error; PSNR: peak signal-to-noise ratio. Results are displayed as mean (standard deviation).

	MSE	PSNR
2-Layer DNN	0.0136 (0.0048)*	27.767 (1.214)*
3D U-Net	0.0107 (0.0041)*	29.297 (0.986)
3D U-Net+L1W	0.0113 (0.0043)*	28.606 (1.007)*
Sketcher	0.0094 (0.0038)	29.475 (0.981)
Sketcher+L1W	0.0103 (0.0042)	29.077 (0.995)*
Refiner (Proposed)	**0.0083 (0.0037)**	**30.044 (1.095)**

GANs

Reference:

- Wei et al, Predicting PET-derived demyelination from multimodal MRI using sketcher-refiner adversarial training for multiple sclerosis, Medical Image Analysis, 2019

Is it worth being adversarial?

- **Example 2: MRI-to-PET**
  - Comparison of 3 GANs to a U-net and a FCN

Table 2
Comparison of myelin content prediction discrepancy (defined as mean absolute difference between the ground truth and the predicted PET) in three defined ROIs between our method and other methods. WM in HC: white matter in healthy controls; NAWM: normal appearing white matter in patients. Results are displayed as mean (standard deviation).

	WM in HC	NAWM	MS Lesions
2-Layer DNN	0.059 (0.040)	0.041 (0.036)	0.131 (0.051)*
3D U-Net	0.053 (0.034)	0.039 (0.033)	0.035 (0.027)
3D U-Net+L1W	0.054 (0.034)	0.038 (0.031)	0.032 (0.029)
Sketcher	0.053 (0.041)	0.034 (0.022)	0.030 (0.017)
Sketcher+L1W	0.052 (0.037)	0.035 (0.027)	0.027 (0.022)
Refiner (Proposed)	**0.048 (0.026)**	**0.029 (0.021)**	**0.022 (0.015)**

GANs

* indicates our method is significantly better with $p < .05$ by two-sided T-test

Reference:

- Wei et al, Predicting PET-derived demyelination from multimodal MRI using sketcher-refiner adversarial training for multiple sclerosis, Medical Image Analysis, 2019

Is it worth being adversarial?

- **There are plenty of examples showing improved performance using GANs**

- **However:**
  - Most of these papers are presenting a GAN method (maybe be biased in favour of GANs)
  - Remember that GANs are more difficult to train
  - It is always worth comparing with a non-adversarial approach (e.g. U-Net or FCN)

Part 7 – Generative models

7.5 Validation

Validation

- **Paired data: comparison of y and $\hat{y}$**

$\hat{y}$ y

  - Visualization of absolute differences
    - Can identify slight misalignments

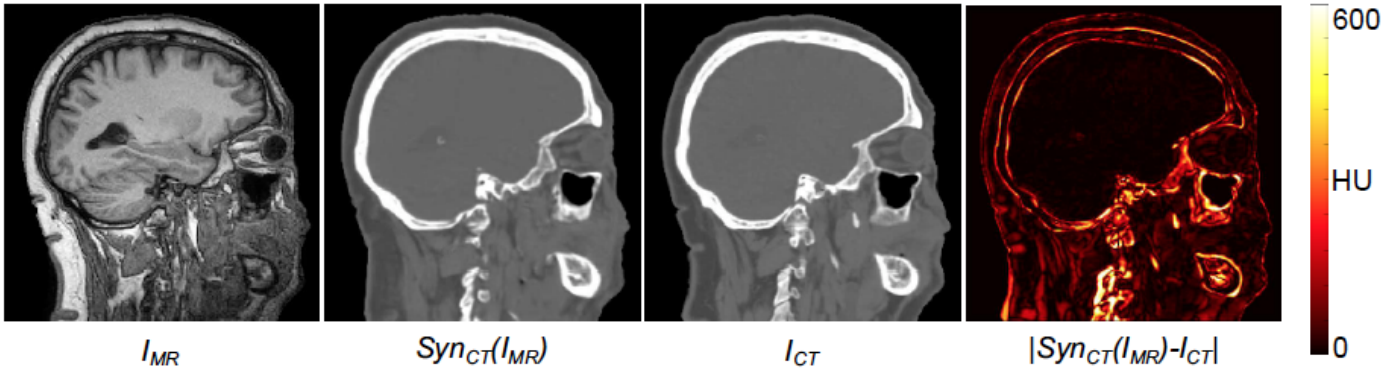


Fig. 4: *From left to right* Input MR image, synthesized CT image, reference real CT image, and absolute error between real and synthesized CT image.

Image source: Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Validation

- **Paired data: comparison of y and $\hat{y}$**

$\hat{y}$ y

  - MSE (Mean Squared Error)

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N} \| (\boldsymbol{I}_{\mathbf{p}}^{i}, \hat{\boldsymbol{I}}_{\mathbf{p}}^{i}) \|_2$$

  - PSNR (Peak Signal to Noise Ratio)
    - Similar to MSE, but normalized

$$PSNR = 20 \cdot \log_{10}(MAX_{I_p}) - 10 \cdot \log_{10}(MSE)$$

where MAX_{I_p} is the maximum voxel value of the image.

  - SSIM (Structural similarity)
    - Better reflects visual perception

Validation

- **Paired data: comparison of y and $\hat{y}$**

  - SSIM (Structural similarity)
    - Better reflects visual perception
    - Compares luminance, contrast and structure

Reference: Wang and Bovik, IEEE Trans Image Processing, 2004

Luminance $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i$ $\quad l(x,y) = \frac{2\mu_x\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1}$

Contrast $\sigma_x = (\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)^2)^{\frac{1}{2}}$ $\quad c(x,y) = \frac{2\sigma_x\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2}$

Structure $\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)$ $\quad s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x\sigma_y + C_3}$

C1, C2 and C3 are factors that stabilize the division when the denominator is small

$$SSIM(x,y) = l(x,y) * c(x,y) * s(x,y)$$

Can be computed locally (in a surrounding window) or globally

Validation

- **Paired data: comparison of y and $\hat{y}$**
  - SSIM (Structural similarity)
    - Better reflects visual perception
    - Compares luminance, contrast and structure

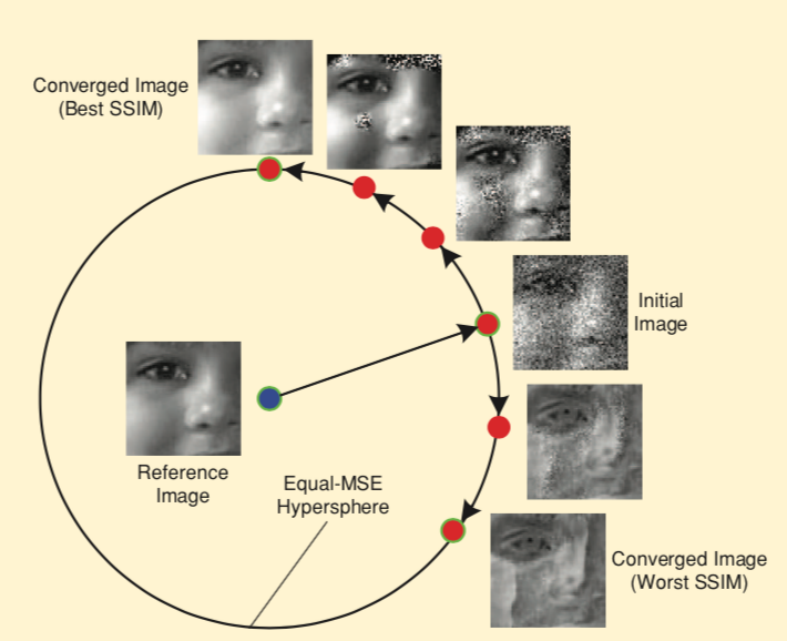


Image source: Wang et al, IEEE Signal Processing Magazine, 2009

Different images with equal MSE but different SSIM

Validation

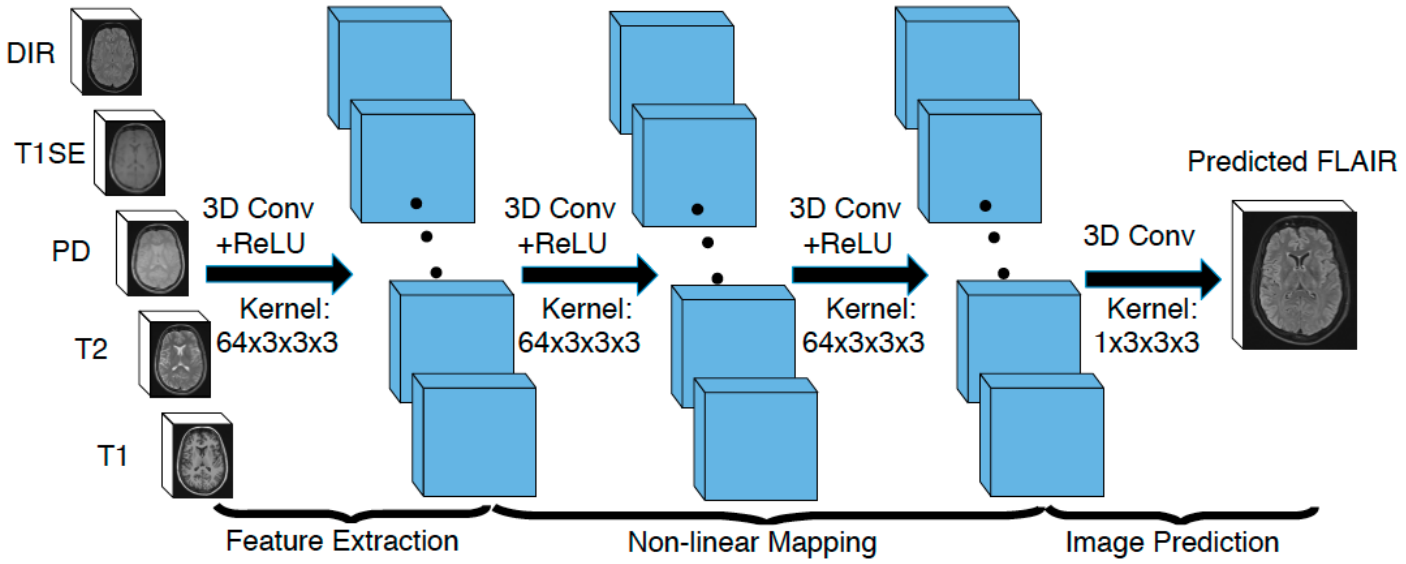
- **Validation through the use of the synthetic images**
  - Image similarity is not sufficient
  - What matters most is that the synthetic images have the same clinical value as the original image
  - Depends on the application
    - Example: Synthesis of CT from MRI
      - Application 1: Attenuation correction of PET images
        - » Recent introduction of PET/MR scanners (standard=PET/CT scanner)
        - » Attenuation correction of PET images requires CT image
      - Application 2: Radiotherapy planning from MR images only
        - » Requires CT information

Validation

- **Validation through the use of the synthetic images**
  - Is the relevant information still present in the synthetic data?
  - For instance, is the detectability of lesions (e.g. tumours) by a clinician as good with a synthetic and a true image?
  - Ideally, one should perform a clinical trial
    - Presenting randomly synthetic and real images to clinicians
    - Compare the rates of detection
  - This is often difficult to do at the stage of academic research
  - An intermediary solution is to assess the caracteristics of the synthetic images

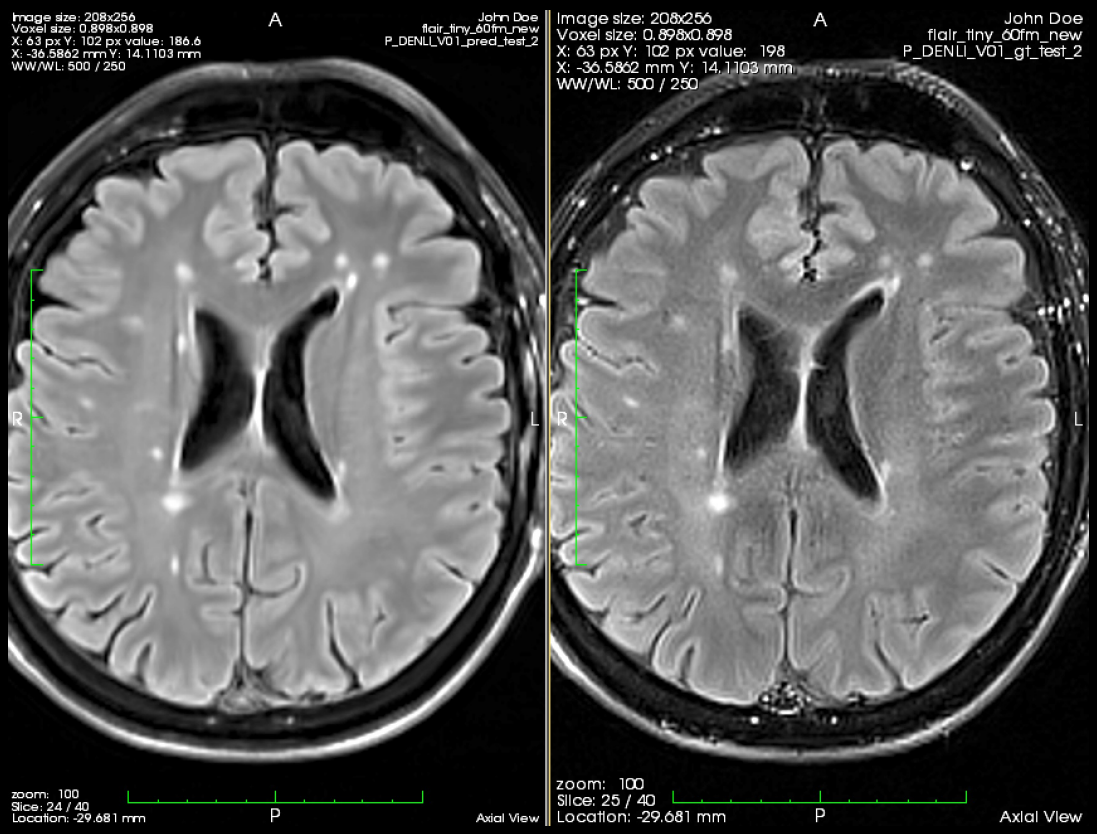
Validation

- **Validation through the use of the synthetic images**
  - Is the relevant information still present in the synthetic data?
  - Example: Synthesis of FLAIR MRI from other MRI sequences
    - (not using GANs but it does not matter here!)



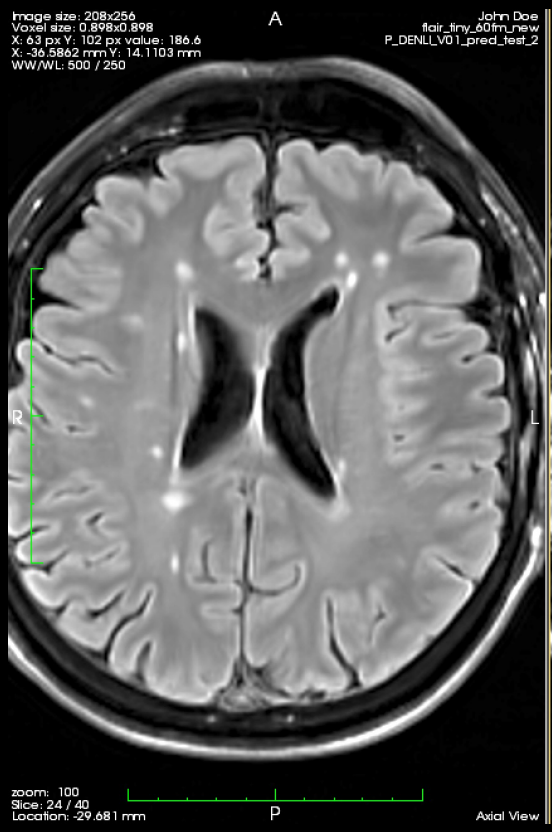
Wen et al, *J Med Imaging*, 2019

True/synthetic FLAIR

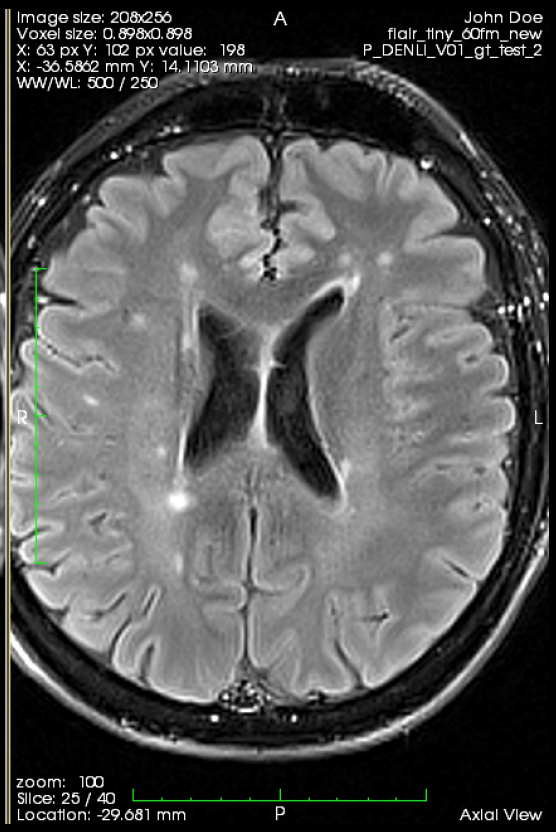


Synthetic FLAIR

True FLAIR



Synthetic FLAIR



True FLAIR

In multiple sclerosis, FLAIR is used for assessing hyperintense white matter lesions

- Visually or
- Using segmentation

Validation

- **Validation through the use of the synthetic images**
  - Is the relevant information still present in the synthetic data?
  - Lesion contrast
    - Evaluate the MS lesion contrast with the normal appearing white matter tissue

$$\text{Ratio 1} = \frac{1}{N}\sum_{i=1}^{N}\frac{I_i(\text{Lesions})}{I_i(\text{NAWM})},$$

Table 3 Evaluation of MS lesion contrast (SD).

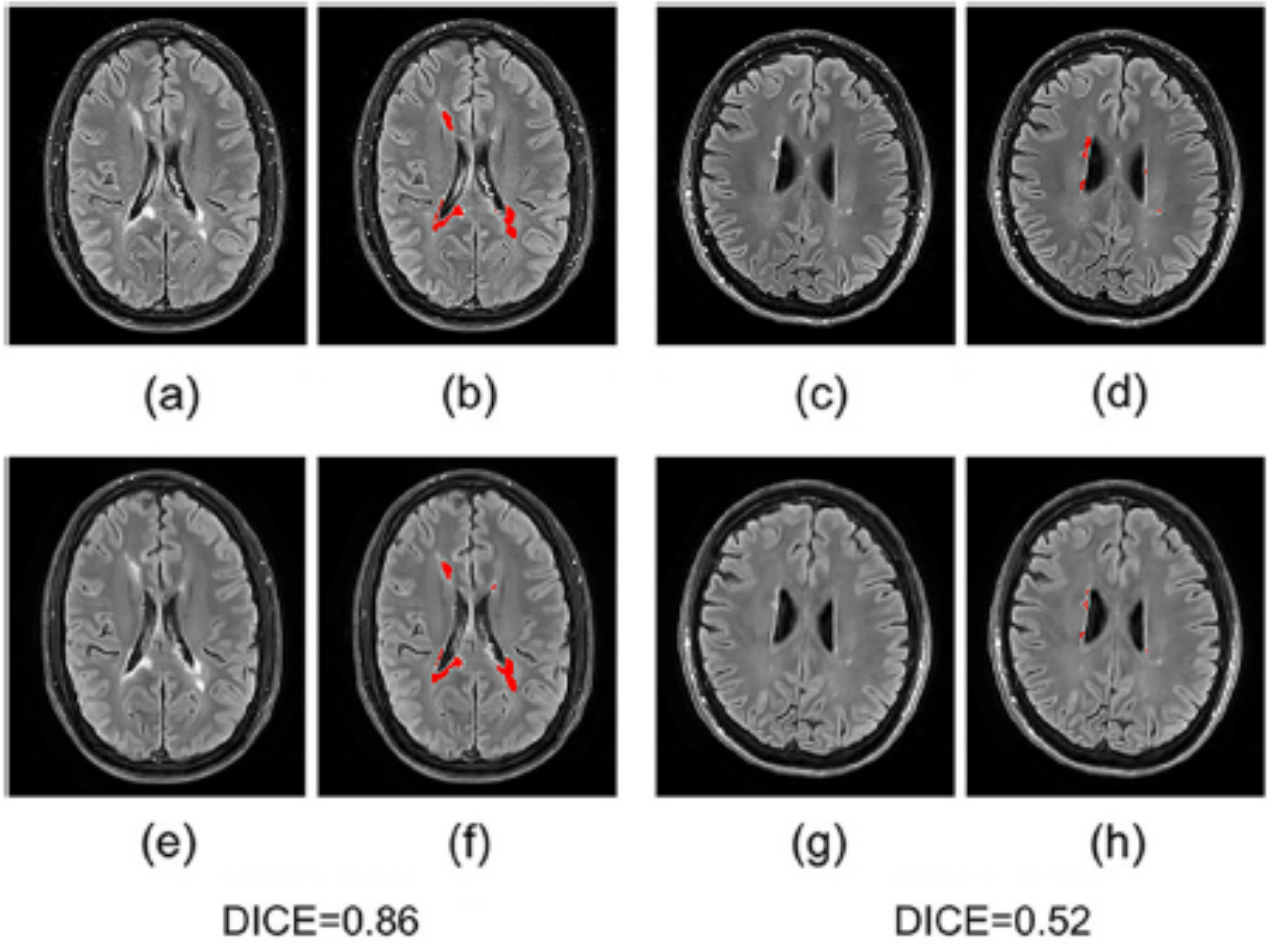
	Random forest 60	Modality propagation	Multilayer perceptron	U-Net	Our method	Ground truth
Ratio 1	1.33 (0.07)	1.31 (0.06)	1.39 (0.11)	1.34 (0.09)	**1.47 (0.13)**	*1.66 (0.12)*

Contrast is not as good as in the true FLAIR but still better than that of competing methods

Validation

- **Validation through the use of the synthetic images**
  - Automatic lesion segmentation
    - Compare segmentation from real and synthetic images

Wen et al, *J Med Imaging*, 2019



True FLAIR

Synthetic FLAIR

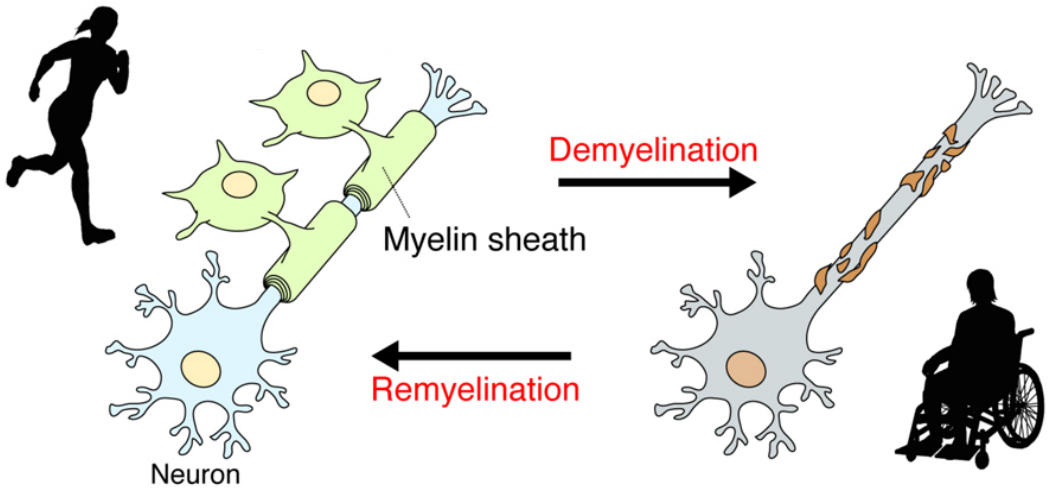
Average Dice index: 0.72

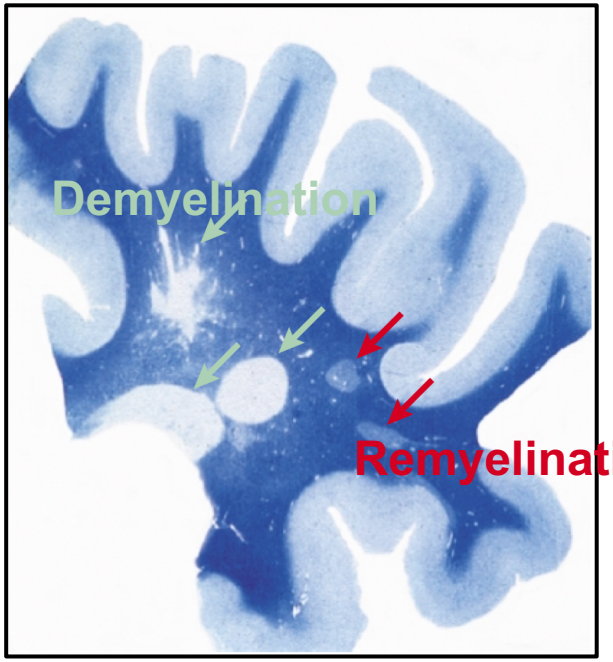
Part 7 – Generative models

7.6 Example: MRI to PET

Synthesizing PET from MRI

- Tracking demyelination is important for management of multiple sclerosis

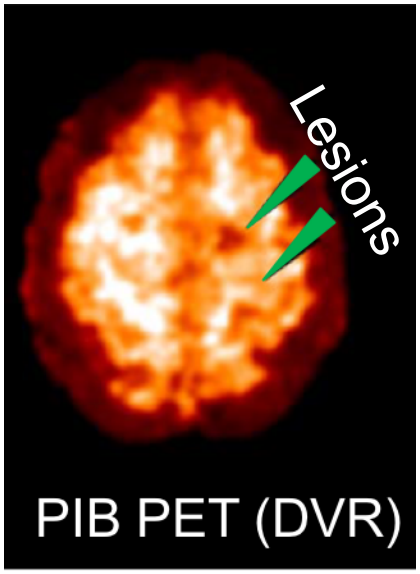




Robin J. M. Franklin: Why does remyelination fail in multiple sclerosis? Nature Reviews Neuroscience 2002
Image source: Medical Press: https://medicalxpress.com/news/2015-09-elucidation-molecular-mechanisms-involved-remyelination.html

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

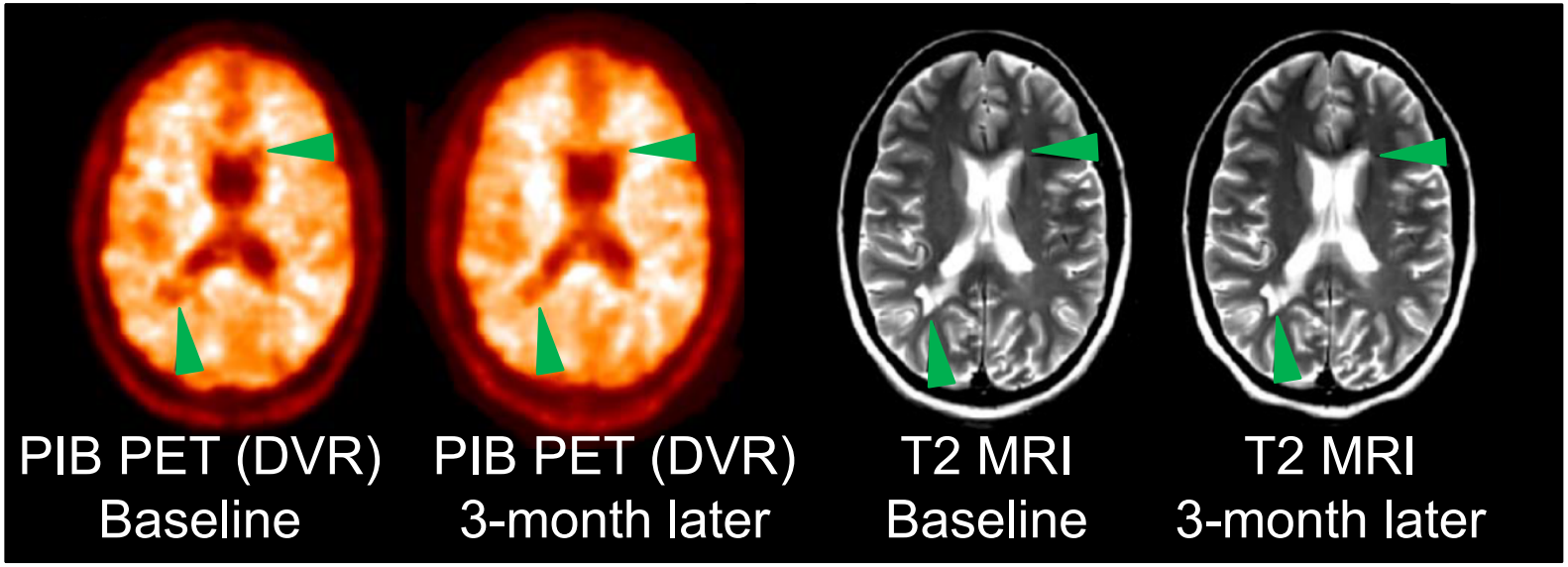
Synthesizing PET from MRI



- [^{11}C] PIB PET can be used to visualize and measure myelin loss and repair in MS lesions.
- **BUT:** It is expensive and invasive due to the injection of the radioactive tracer.

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

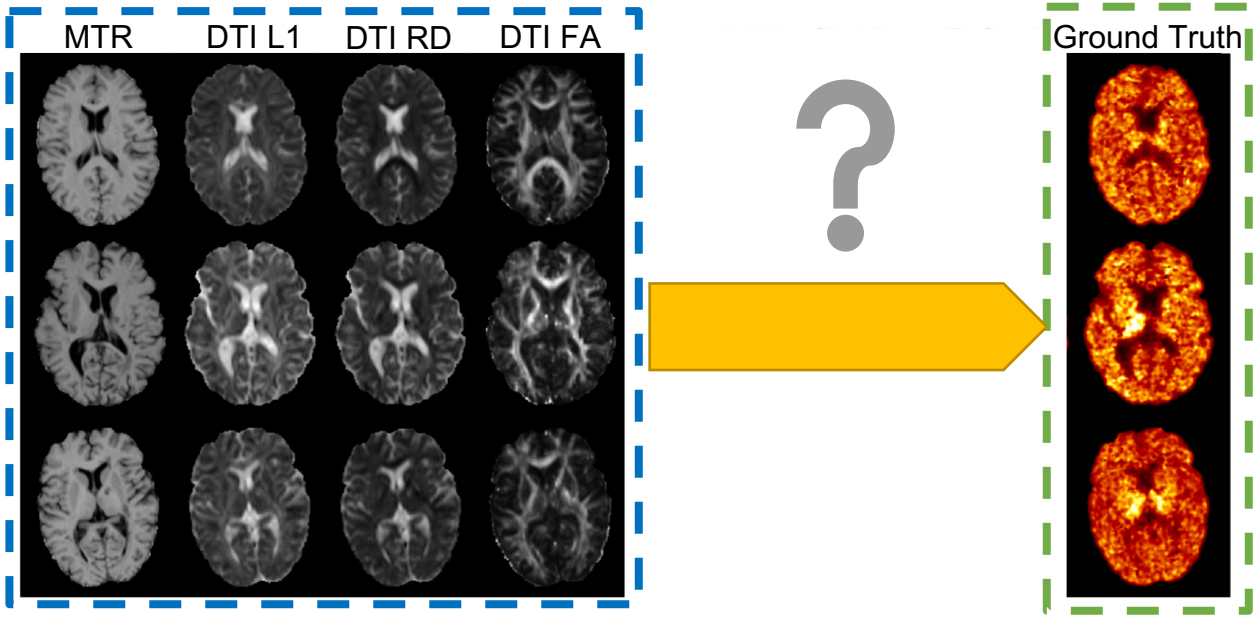
Synthesizing PET from MRI



- MR imaging is a widely available and non invasive technique.
- **BUT:** Existing MRI sequences do not provide, to date, a reliable, specific, or direct marker of demyelination and remyelination.

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

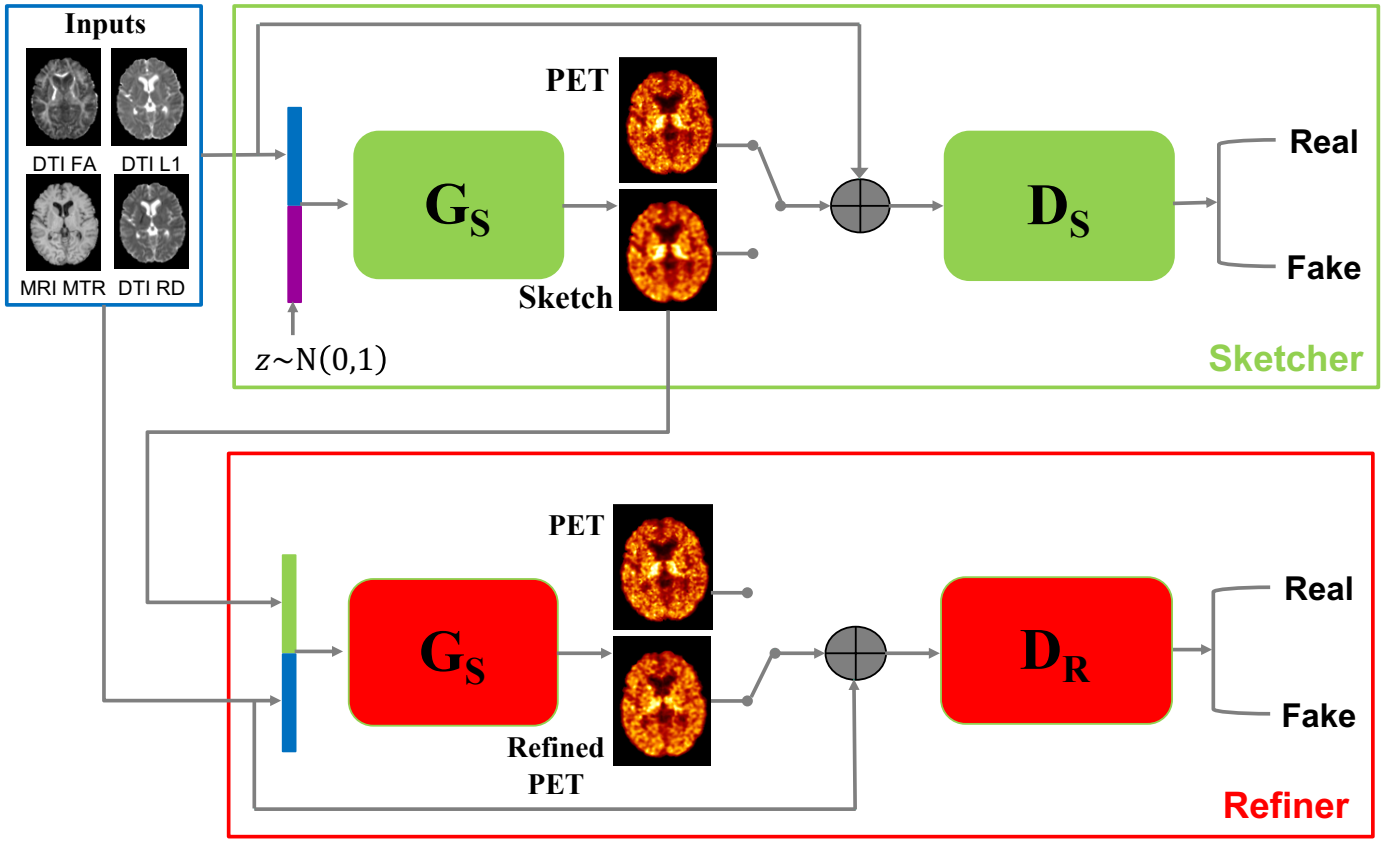


Predict *PIB PET derived myelin content* from *multimodal MRI (MTR and DTI which are sensitive to myelin)*

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

- **Proposed approach: Sketcher-Refiner Generative Adversarial Networks (GANs)**



Wen et al, *MICCAI*, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

Loss function (Sketcher):

$$G_{\mathrm{S}}^{*} = \arg \min_{G_{\mathrm{S}}} \max_{D_{\mathrm{S}}} \mathcal{L}(D_{\mathrm{S}}, G_{\mathrm{S}}) + \lambda_{\mathrm{S}} \mathcal{L}_{L1}(G_{\mathrm{S}})$$

Cross-entropy

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

Loss function (Sketcher):

$$G_{\mathrm{S}}^{*} = \arg \min_{G_{\mathrm{S}}} \max_{D_{\mathrm{S}}} \mathcal{L}(D_{\mathrm{S}}, G_{\mathrm{S}}) + \lambda_{\mathrm{S}} \mathcal{L}_{L1}(G_{\mathrm{S}})$$

L1 loss to encourage less blurring

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

Loss function (Refiner):

$$G_{\mathrm{R}}^{*} = \arg\min_{G_{\mathrm{R}}} \max_{D_{\mathrm{R}}} \mathcal{L}(D_{\mathrm{R}}, G_{\mathrm{R}}) + \lambda_{\mathrm{R}} \mathcal{L}_{L1}(G_{\mathrm{R}})$$

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

Loss function (Refiner):

$$G_{\mathrm{R}}^{*} = \arg\min_{G_{\mathrm{R}}} \max_{D_{\mathrm{R}}} \mathcal{L}(D_{\mathrm{R}}, G_{\mathrm{R}}) + \lambda_{\mathrm{R}} \mathcal{L}_{L1}(G_{\mathrm{R}})$$

Weighted L1 loss to force the refiner on MS lesions where demyelination happens

$$\mathcal{L}_{L1}(G_R) = \frac{1}{N \times M} \sum_{i=1}^{N} \left(\frac{1}{N_{Les}} \sum_{j \in R_{Les}} \left| I_P^{i,j} - \hat{I}_P^{i,j} \right| + \frac{1}{N_{NAWM}} \sum_{j \in R_{NAWM}} \left| I_P^{i,j} - \hat{I}_P^{i,j} \right| + \frac{1}{N_{other}} \sum_{j \in R_{other}} \left| I_P^{i,j} - \hat{I}_P^{i,j} \right| + \right)$$

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

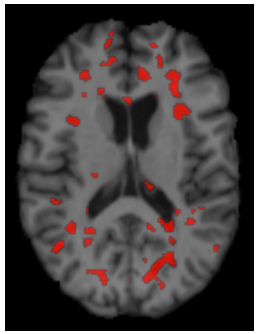
Synthesizing PET from MRI

Loss function (Refiner):

$$G_{\mathrm{R}}^{*} = \arg\min_{G_{\mathrm{R}}} \max_{D_{\mathrm{R}}} \mathcal{L}(D_{\mathrm{R}}, G_{\mathrm{R}}) + \lambda_{\mathrm{R}} \mathcal{L}_{L1}(G_{\mathrm{R}})$$

Weighted L1 loss to force the refiner on MS lesions where demyelination happens

$$\mathcal{L}_{L1}(G_R) = \frac{1}{N \times M} \sum_{i=1}^{N} \left(\frac{1}{N_{Les}} \sum_{j \in R_{Les}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{NAWM}} \sum_{j \in R_{NAWM}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{other}} \sum_{j \in R_{other}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \right)$$



Wen et al, *MICCAI*, 2018;
Medical Image Analysis, 2019

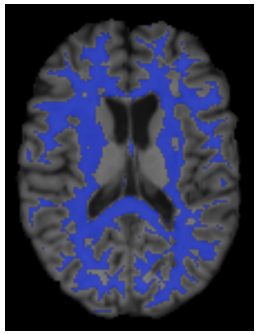
Synthesizing PET from MRI

Loss function (Refiner):

$$G_{\mathrm{R}}^{*} = \arg\min_{G_{\mathrm{R}}} \max_{D_{\mathrm{R}}} \mathcal{L}(D_{\mathrm{R}}, G_{\mathrm{R}}) + \lambda_{\mathrm{R}} \mathcal{L}_{L1}(G_{\mathrm{R}})$$

Weighted L1 loss to force the refiner on MS lesions where demyelination happens

$$\mathcal{L}_{L1}(G_R) = \frac{1}{N \times M} \sum_{i=1}^{N} \left(\frac{1}{N_{Les}} \sum_{j \in R_{Les}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{NAWM}} \sum_{j \in R_{NAWM}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{other}} \sum_{j \in R_{other}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \right)$$



Wen et al, *MICCAI*, 2018;
Medical Image Analysis, 2019

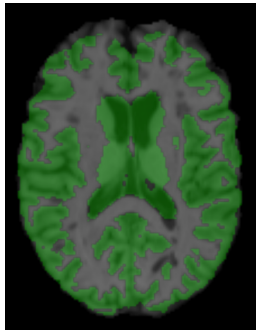
Synthesizing PET from MRI

Loss function (Refiner):

$$G_{\mathrm{R}}^{*} = \arg\min_{G_{\mathrm{R}}} \max_{D_{\mathrm{R}}} \mathcal{L}(D_{\mathrm{R}}, G_{\mathrm{R}}) + \lambda_{\mathrm{R}} \mathcal{L}_{L1}(G_{\mathrm{R}})$$

Weighted L1 loss to force the refiner on MS lesions where demyelination happens

$$\mathcal{L}_{L1}(G_R) = \frac{1}{N \times M} \sum_{i=1}^{N} \left(\frac{1}{N_{Les}} \sum_{j \in R_{Les}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{NAWM}} \sum_{j \in R_{NAWM}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \frac{1}{N_{other}} \sum_{j \in R_{other}} |I_P^{i,j} - \hat{I}_P^{i,j}| + \right)$$



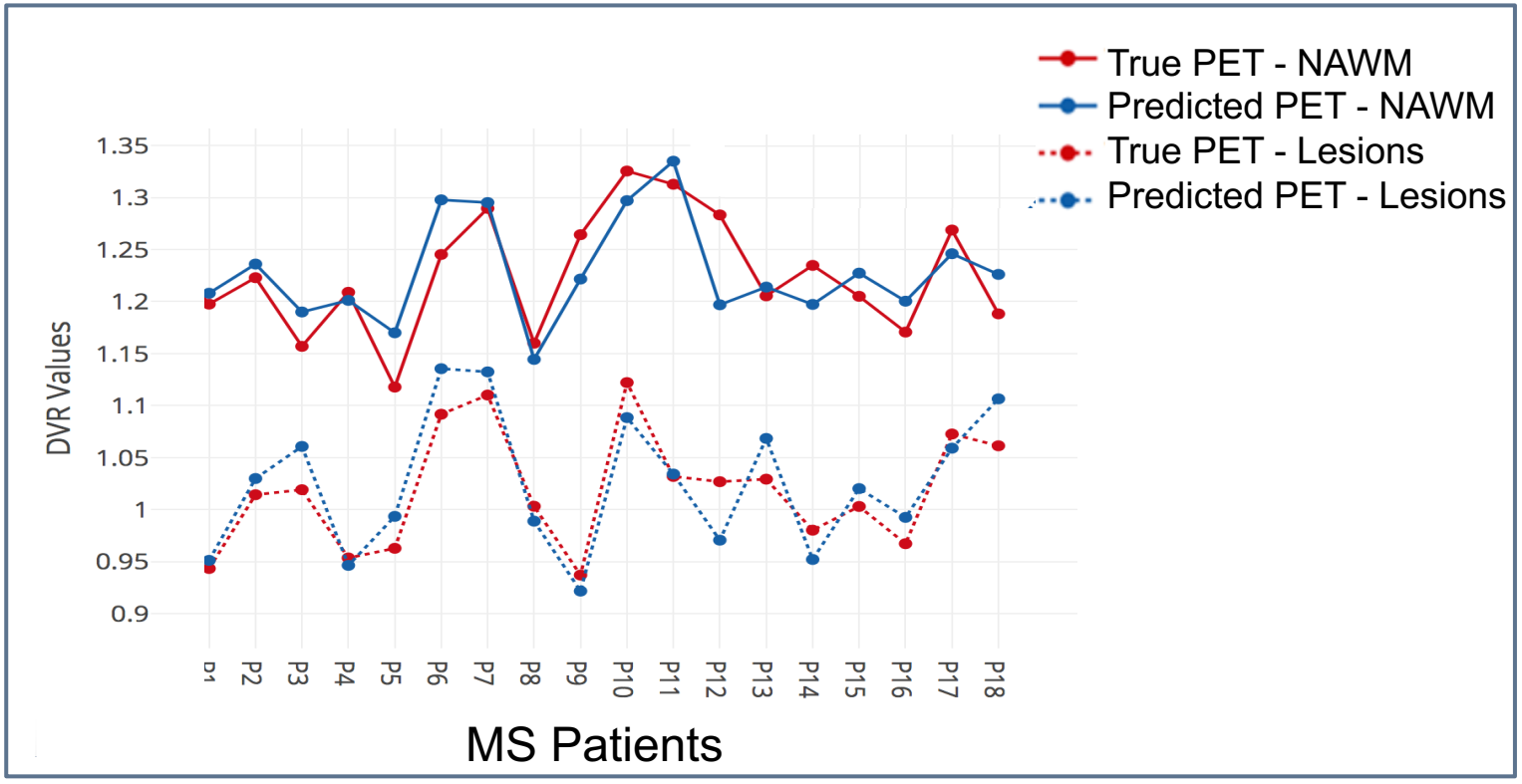
Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

Table 1
Image quality metrics obtained with our method and the other methods. MSE: mean square error; PSNR: peak signal-to-noise ratio. Results are displayed as mean (standard deviation).

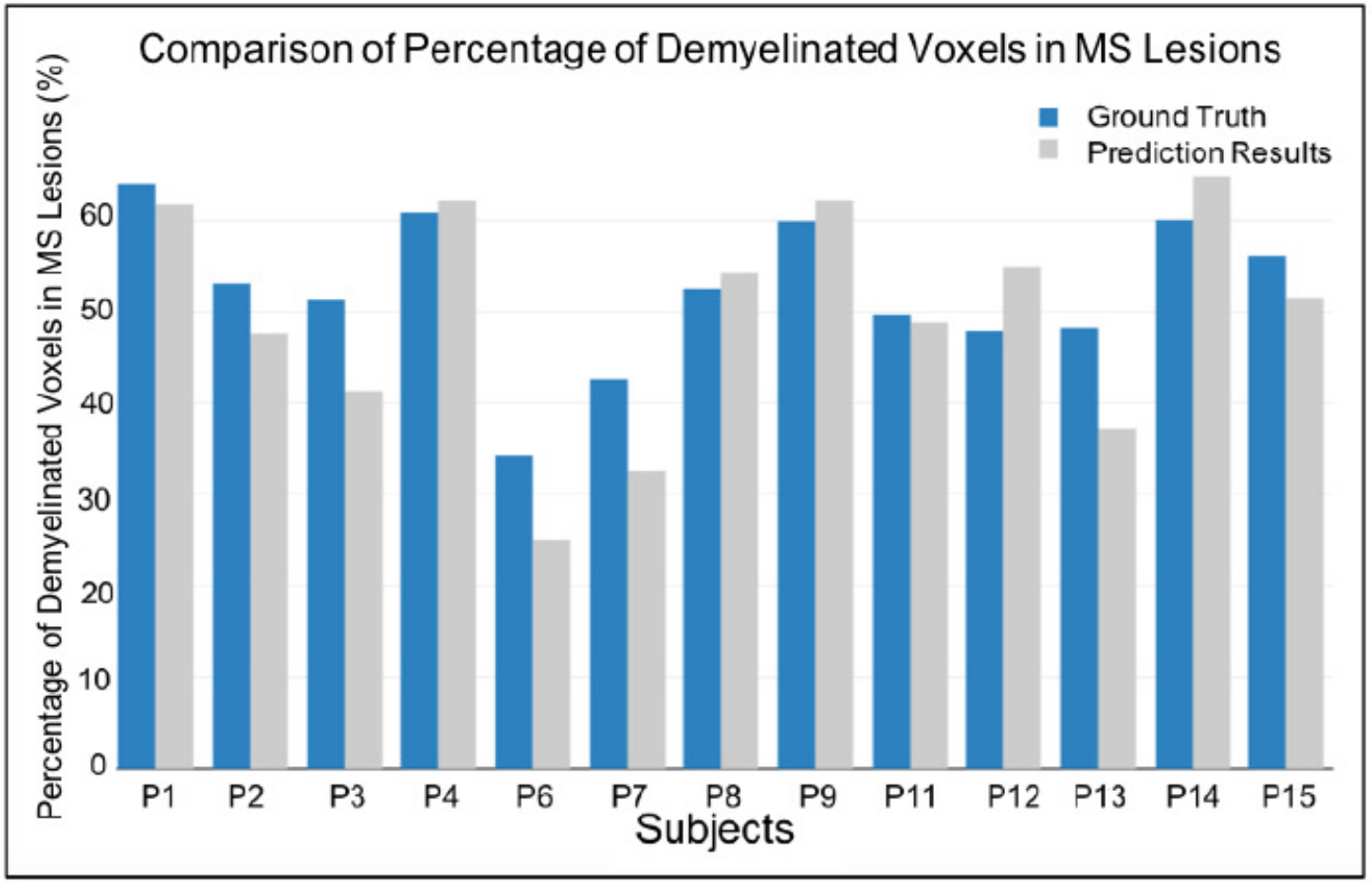
	MSE	PSNR
2-Layer DNN	0.0136 (0.0048)*	27.767 (1.214)*
3D U-Net	0.0107 (0.0041)*	29.297 (0.986)
3D U-Net+L1W	0.0113 (0.0043)*	28.606 (1.007)*
Sketcher	0.0094 (0.0038)	29.475 (0.981)
Sketcher+L1W	0.0103 (0.0042)	29.077 (0.995)*
Refiner (Proposed)	**0.0083 (0.0037)**	**30.044 (1.095)**

Synthesizing PET from MRI



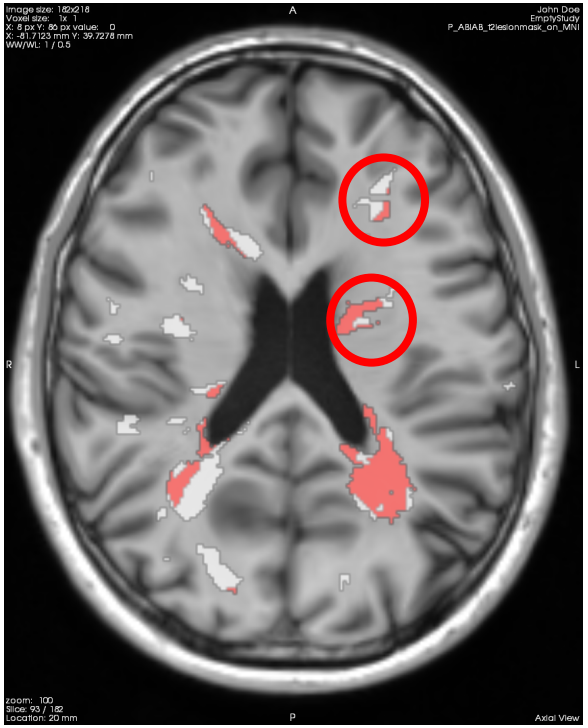
Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI

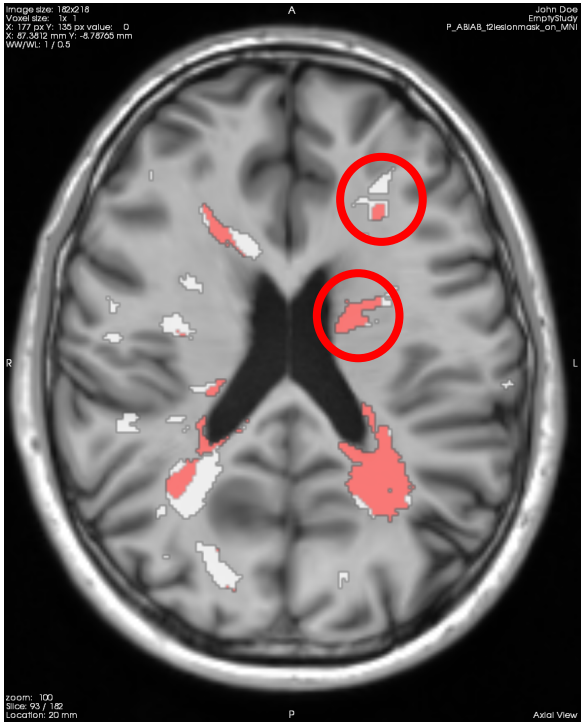


Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Synthesizing PET from MRI



Demyelination map from True PET



Demyelination map from Predicted PET

Average DICE between two demyelination maps (rose): 0.83

Wen et al, ***MICCAI***, 2018;
Medical Image Analysis, 2019

Part 7 – Generative models

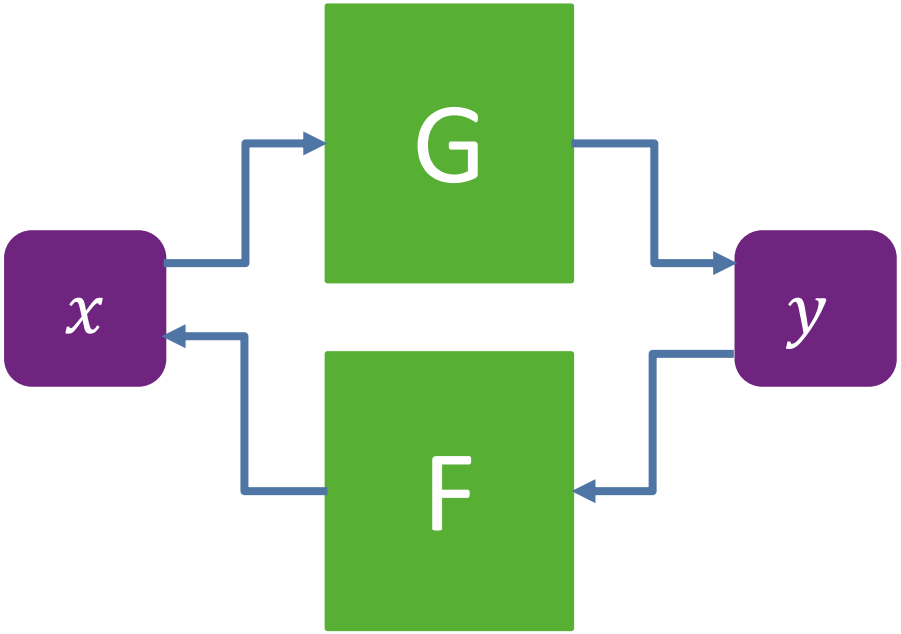
7.7 Medical image translation with CycleGANs

Image translation

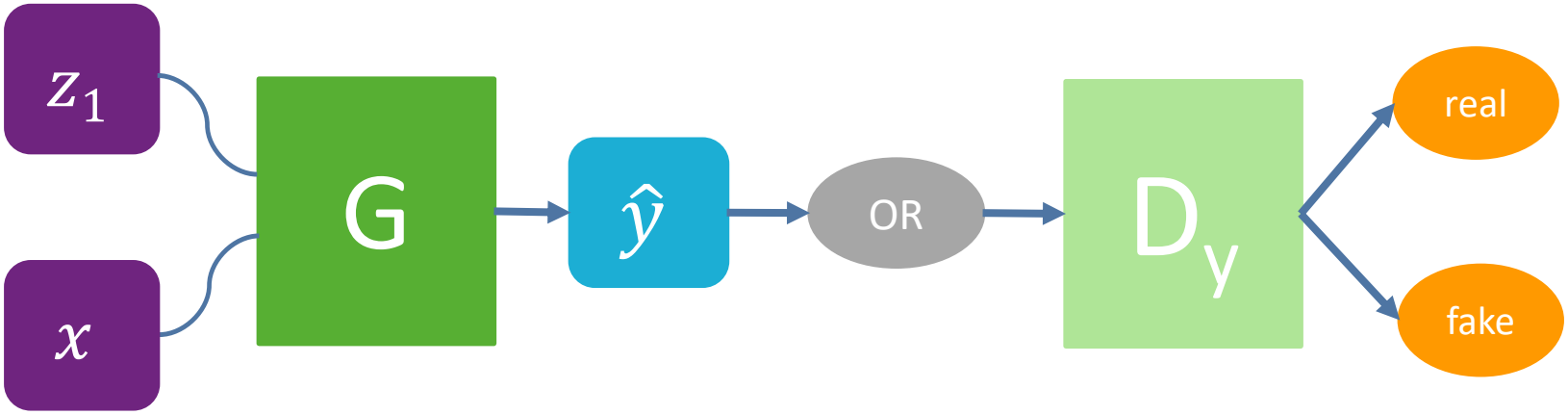
- **What if I don’t have paired data?**
  - Or if only part of may data is paired
  - i.e. we have set of input data xi and a set of output data yj but not in the same patients
  - Remember the paired case: we have a set of couples (xi,yi) where i is the index of a given patient

CycleGAN

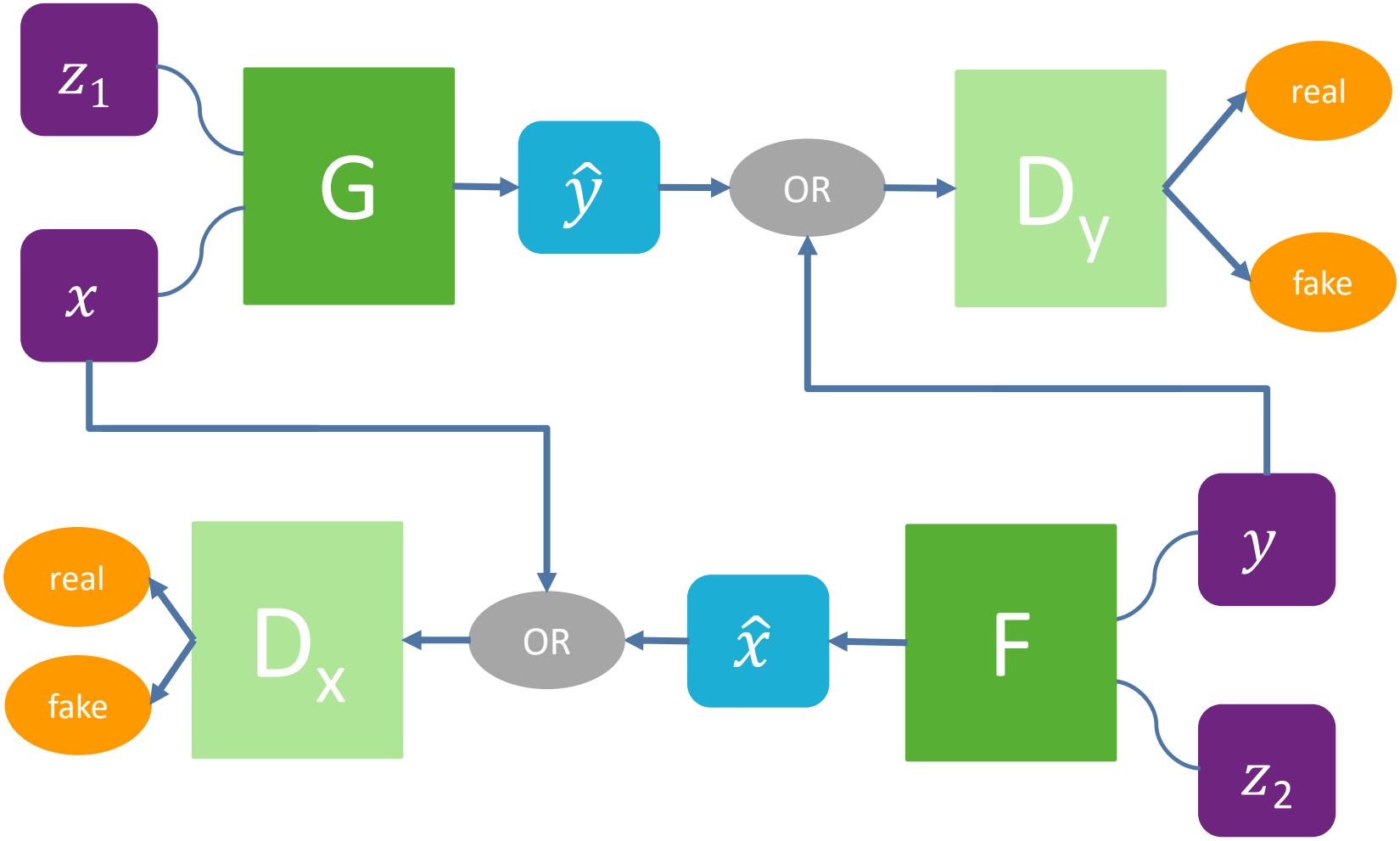
- **Two GANs**
  - G transforms x into y
  - F transforms y into x



CycleGAN

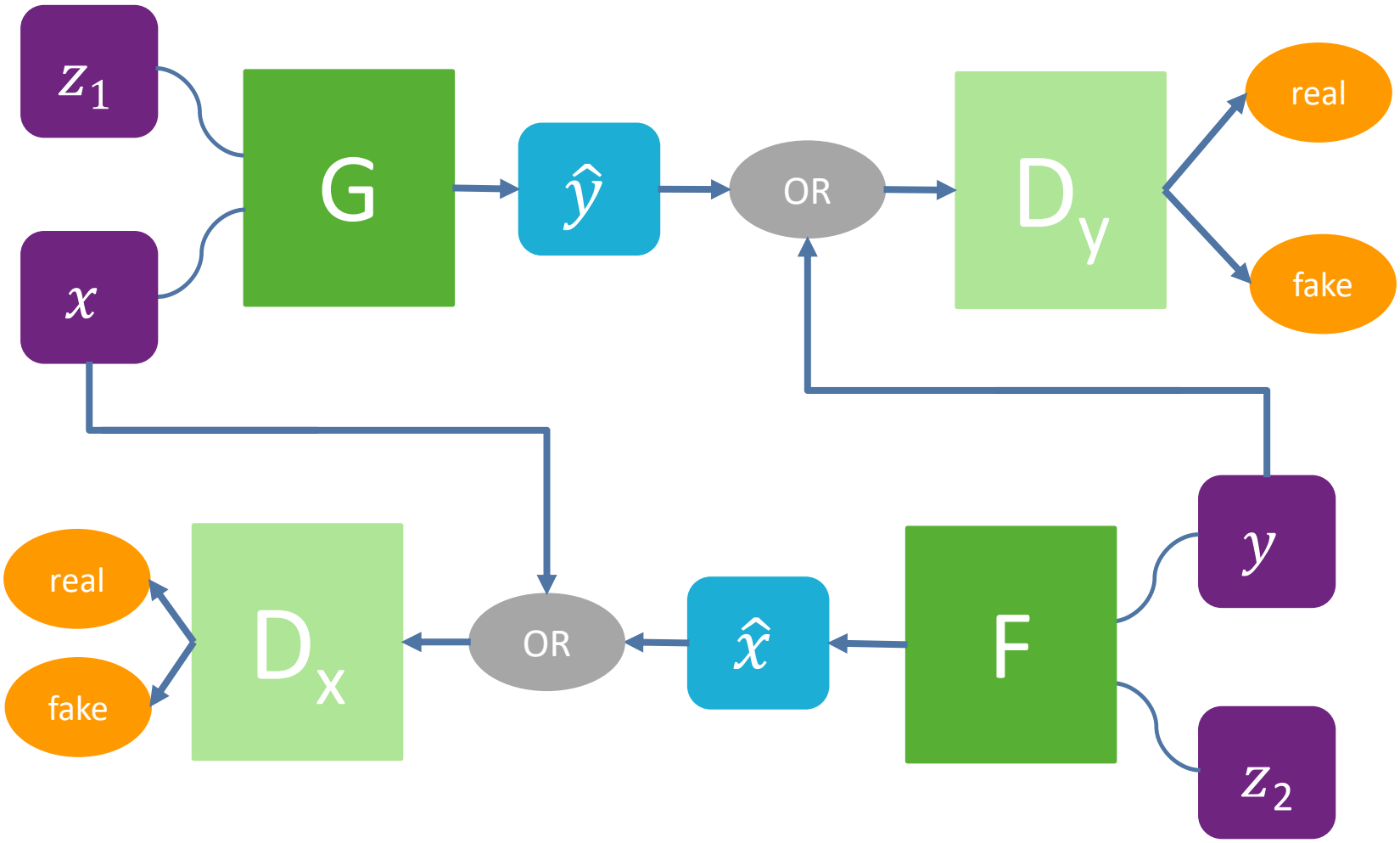


CycleGAN



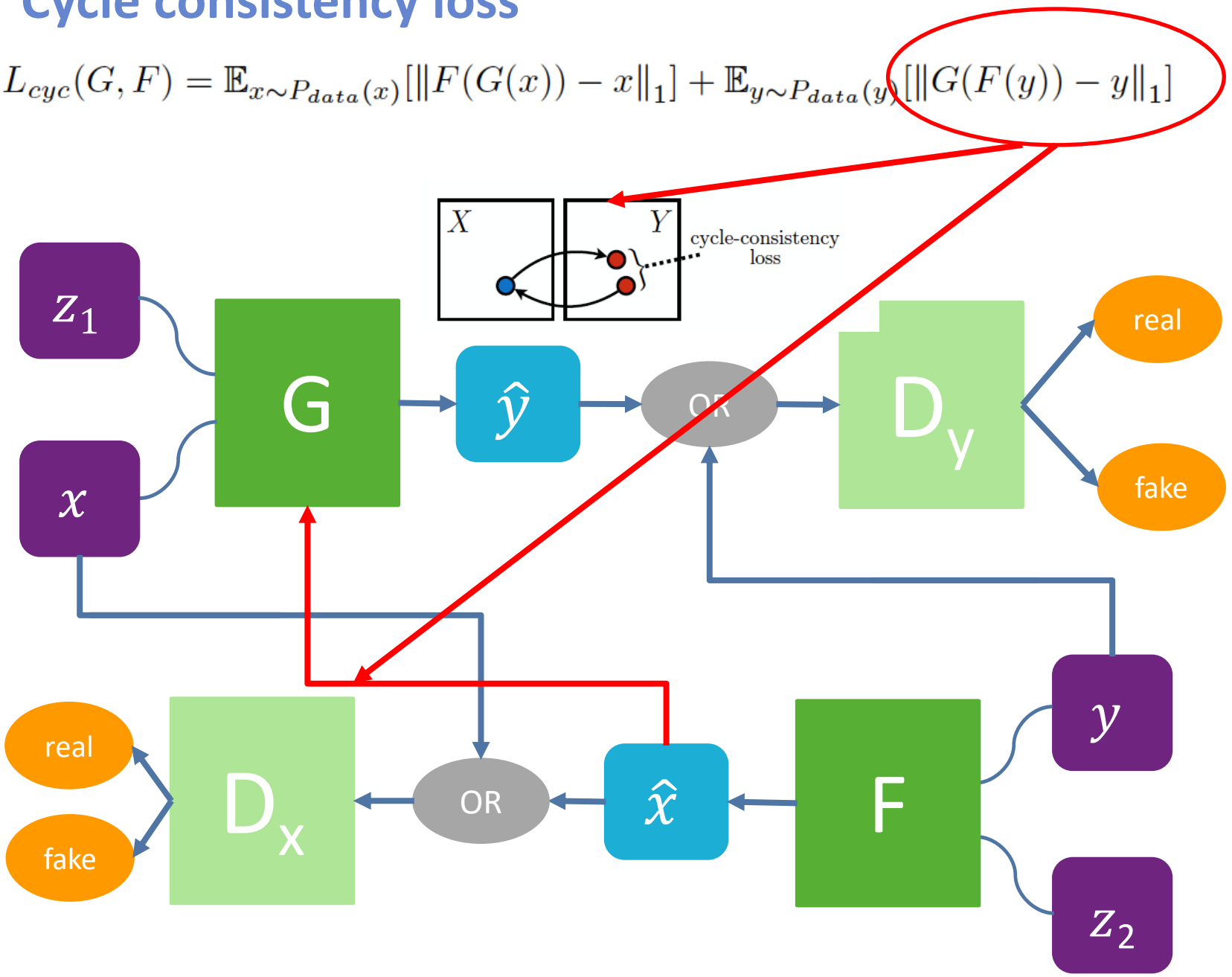
Cycle consistency loss

$$L_{cyc}(G,F) = \mathbb{E}_{x\sim P_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim P_{data}(y)}[\|G(F(y)) - y\|_1]$$



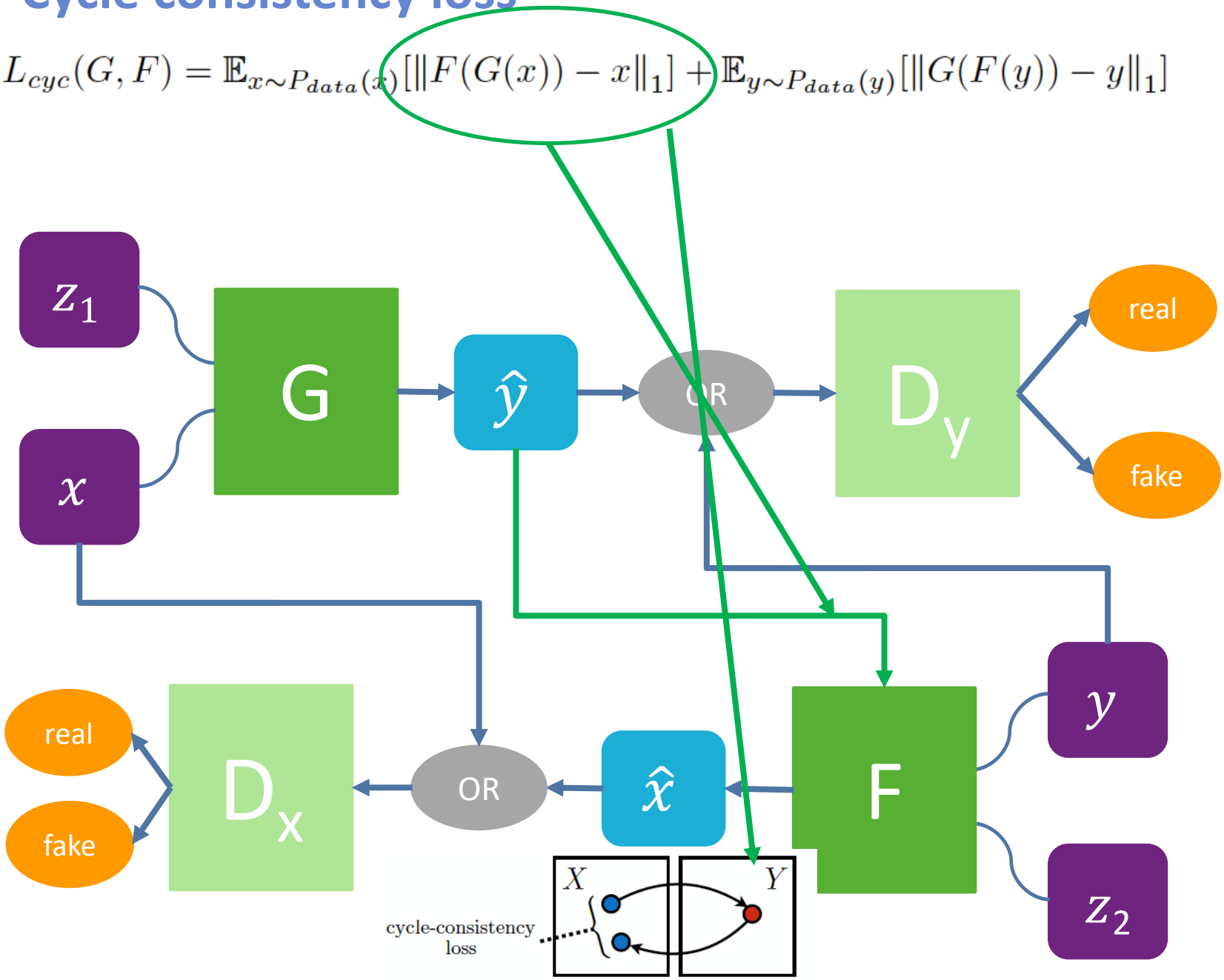
Cycle consistency loss

$$L_{cyc}(G,F) = \mathbb{E}_{x\sim P_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y\sim P_{data}(y)}[\|G(F(y)) - y\|_1]$$



Cycle consistency loss

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim P_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim P_{data}(y)}[\|G(F(y)) - y\|_1]$$

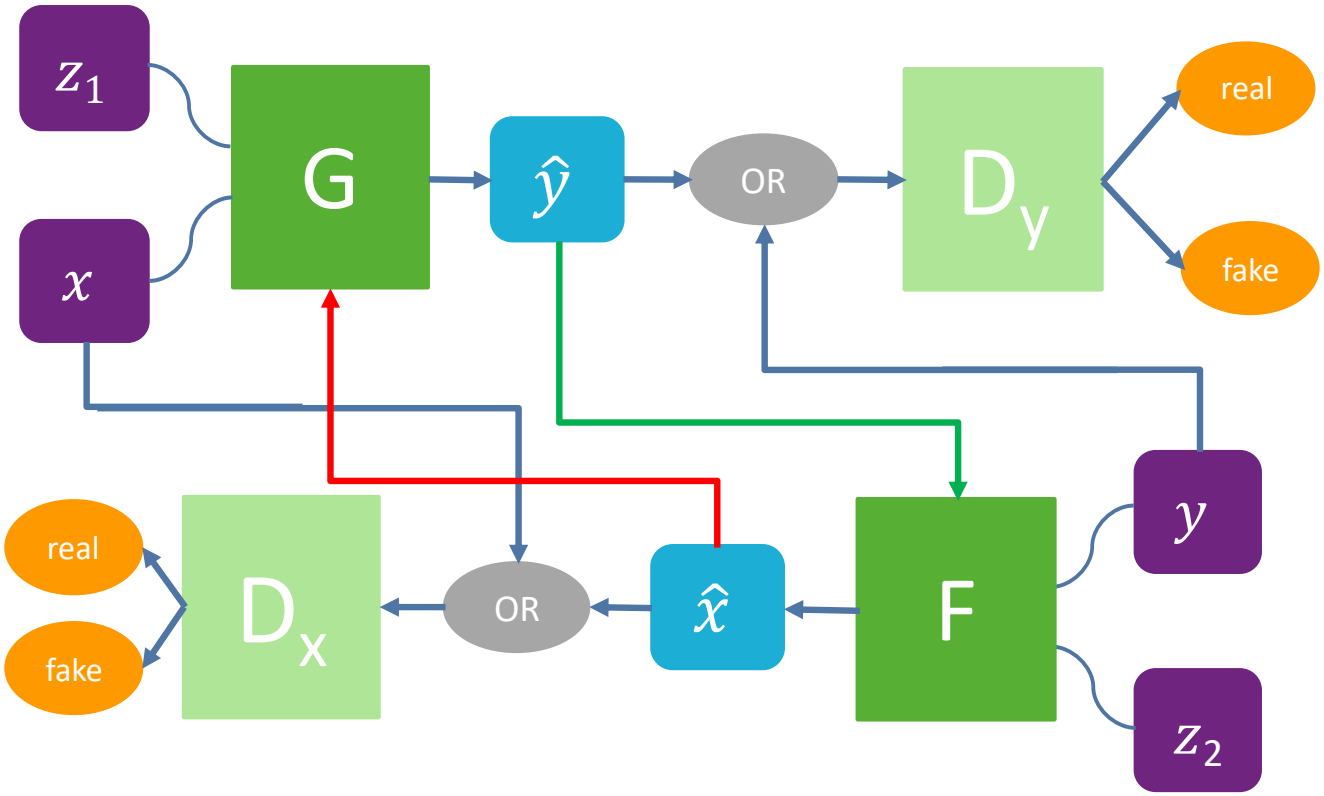


Cycle GAN

$$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F) \quad (3)$$

with

$$L_{cyc}(G, F) = \mathbb{E}_{x \sim P_{data}(x)}[\|F(G(x)) - x\|_1] + \mathbb{E}_{y \sim P_{data}(y)}[\|G(F(y)) - y\|_1] \quad (4)$$



Reference: Zhu et al. Unpaired image-to-image translation using cycle-consistent adversarial networks. ICCV 2017

Application: MRI to CT

Deep MR to CT Synthesis using Unpaired Data

Jelmer M. Wolterink[1]✉, Anna M. Dinkla[2], Mark H.F. Savenije[2],
Peter R. Seevinck[1], Cornelis A.T. van den Berg[2], Ivana Išgum[1]

[1] Image Sciences Institute, University Medical Center Utrecht, The Netherlands
j.m.wolterink@umcutrecht.nl
[2] Department of Radiotherapy, University Medical Center Utrecht, The Netherlands

Application: MRI to CT

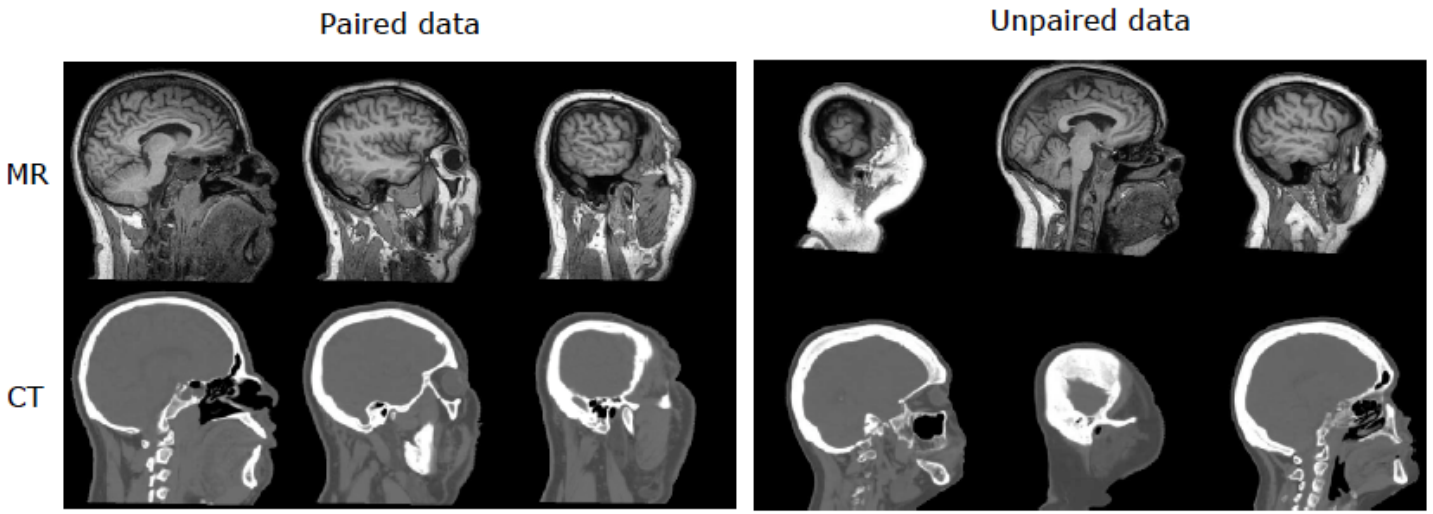


Fig. 1: *Left* When training with paired data, MR and CT slices that are simultaneously provided to the network correspond to the same patient at the same anatomical location. *Right* When training with unpaired data, MR and CT slices that are simultaneously provided to the network belong to different patients at different locations in the brain.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

2 Data

This study included brain MR and CT images of 24 patients that were scanned for radiotherapy treatment planning of brain tumors. MR and CT images were acquired on the same day in radiation treatment position using a thermoplastic mask for immobilization. Patients with heavy dental artefacts on CT and/or MR were excluded. T1 3D MR (repetition time 6.98 ms, echo time 3.14 ms, flip angle 8°) images were obtained with dual flex coils on a Philips Ingenia 1.5T MR scanner (Philips Healthcare, Best, The Netherlands). CT images were acquired helically on a Philips Brilliance Big Bore CT scanner (Philips Healthcare, Best, The Netherlands) with 120 kVp and 450 mAs. To allow voxel-wise comparison of synthetic and reference CT images, MR and CT images of the same patient were aligned using rigid registration based on mutual information following a clinical procedure. This registration did not correct for local misalignment (Fig. 2). CT images had a resolution of $1.00 \times 0.90 \times 0.90$ mm^3 and were resampled to the same voxel size as the MR, namely $1.00 \times 0.87 \times 0.87$ mm^3. Each volume had $183 \times 288 \times 288$ voxels. A head region mask excluding surrounding air was obtained in the CT image and propagated to the MR image.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

Fig. 2: Examples showing local misalignment between MR and CT images after rigid registration using mutual information. Although the skull is generally well-aligned, misalignments may occur in the throat, mouth, vertebrae, and nasal cavities.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

The PyTorch implementation provided by the authors of [12] was used in all experiments[3]. This implementation performs voxel regression and image classification in 2D images. Here, experiments were performed using 2D sagittal image slices (Fig. 1). We provide a brief description of the synthesis and discriminator CNNs. Further implementation details are provided in [12].

The 24 data sets were separated into a training set containing MR and CT volumes of 18 patients and a separate test set containing MR and corresponding reference CT volumes of 6 patients.

Each MR or CT volume contained 183 sagittal 2D image slices. These were resampled to 256×256 pixel images with 256 grayscale values uniformly distributed in $[-600, 1400]$ HU for CT and $[0, 3500]$ for MR. This put image values in the same range as in [12], so that the default value of $\lambda = 10$ was used to weigh cycle consistency loss. To augment the number of training samples, each image was padded to 286×286 pixels and sub-images of 256×256 pixels were randomly cropped during training. The model was trained using Adam [7] for 100 epochs with a fixed learning rate of 0.0002, and 100 epochs in which the learning rate was linearly reduced to zero. Model training took 52 hours on a single NVIDIA Titan X GPU. MR to CT synthesis with a trained model took around 10 seconds.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

The PyTorch implementation provided by the authors of [12] was used in all experiments[3]. This implementation performs voxel regression and image classification in 2D images. Here, experiments were performed using 2D sagittal image slices (Fig. 1). We provide a brief description of the synthesis and discriminator CNNs. Further implementation details are provided in [12].

The 24 data sets were separated into a training set containing MR and CT volumes of 18 patients and a separate test set containing MR and corresponding reference CT volumes of 6 patients.

Each MR or CT volume contained 183 sagittal 2D image slices. These were resampled to 256×256 pixel images with 256 grayscale values uniformly distributed in $[-600, 1400]$ HU for CT and $[0, 3500]$ for MR. This put image values in the same range as in [12], so that the default value of $\lambda = 10$ was used to weigh cycle consistency loss. To augment the number of training samples, each image was padded to 286×286 pixels and sub-images of 256×256 pixels were randomly cropped during training. The model was trained using Adam [7] for 100 epochs with a fixed learning rate of 0.0002, and 100 epochs in which the learning rate was linearly reduced to zero. Model training took 52 hours on a single NVIDIA Titan X GPU. MR to CT synthesis with a trained model took around 10 seconds.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

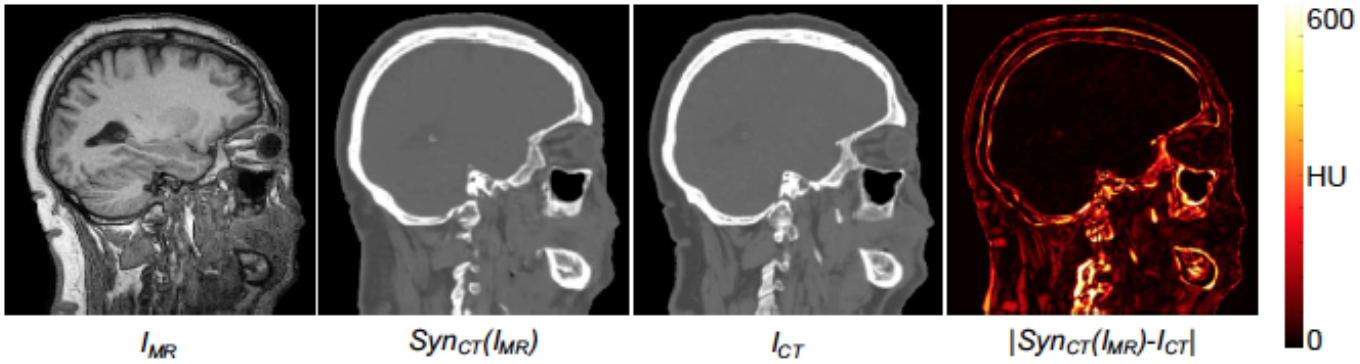


Fig. 4: *From left to right* Input MR image, synthesized CT image, reference real CT image, and absolute error between real and synthesized CT image.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

Table 1: Mean absolute error (MAE) values in HU and peak-signal-to-noise ratio (PSNR) between synthesized and real CT images when training with paired or unpaired data.

	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 ± SD	73.7 ± 2.3	89.4 ± 6.8	32.3 ± 0.7	30.6 ± 0.9

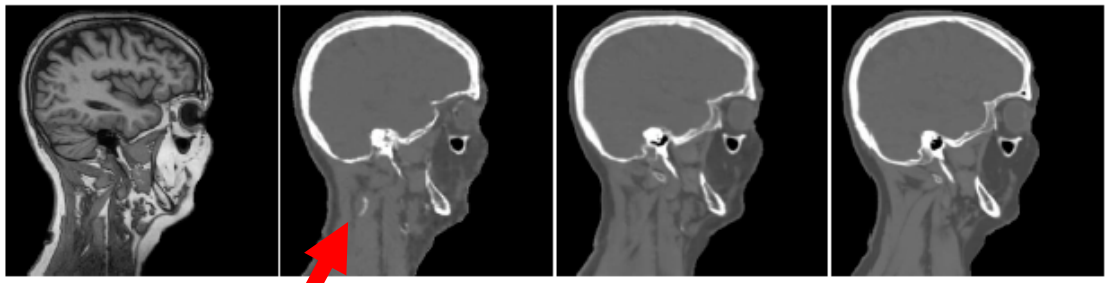


Fig. 5: *From left to right* Input MR image, synthesized CT image with paired training, synthesized CT image with unpaired training, reference real CT image.

this model than for images obtained using the unpaired model. Fig. 5 shows a visual comparison of results obtained with unpaired and paired training data. The image obtained with paired training data is more blurry and contains a high-intensity artifact in the neck.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

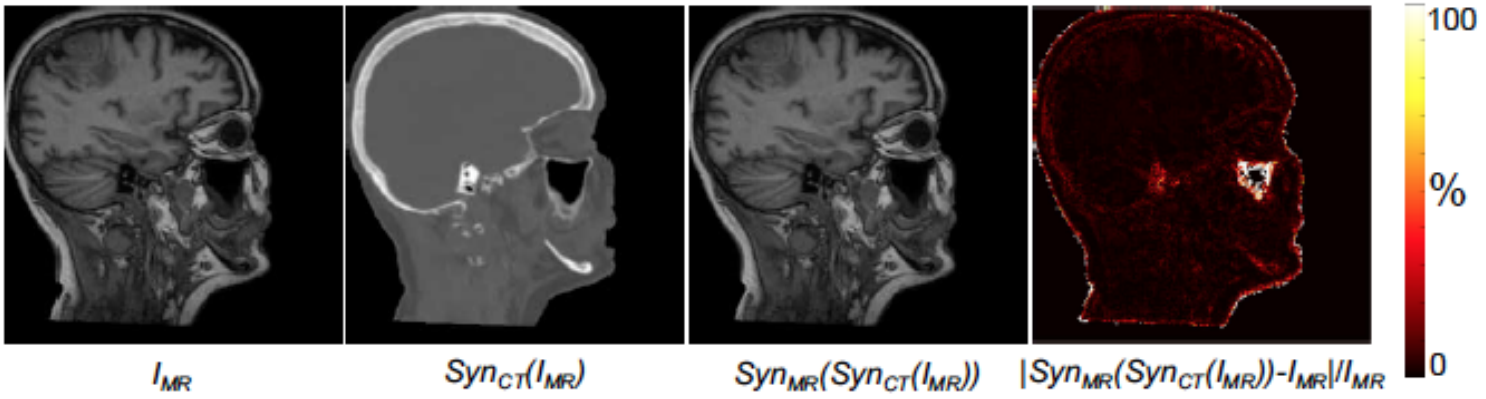


Fig. 6: *From left to right* Input MR image, synthesized CT image, reconstructed MR image, and relative error between the input and reconstructed MR image.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

Yi et al. showed that a model using cycle consistency for unpaired data can in some cases outperform a GAN-model on paired data [11]. Similarly, we found that in our test data sets, the model trained using unpaired data outperformed the model trained using paired data. Qualitative analysis showed that CT images obtained by the model trained with unpaired data looked more realistic, contained less artifacts and contained less blurring than those obtained by the model trained with paired data. This was reflected in the quantitative analysis. This could be due to misalignment between MR and CT images (Fig. 2), which is ignored when training with unpaired data.

The results indicate that image synthesis CNNs can be trained using unaligned data. This could have implications for MR-only radiotherapy treatment planning, but also for clinical applications where patients typically receive only one scan of a single anatomical region. In such scenarios, paired data is scarce, but there are many single acquisitions of different modalities. Possible applications are synthesis between MR images acquired at different field strengths [1], or between CT images acquired at different dose levels [10].

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Application: MRI to CT

The results in this study were obtained using a model that was trained with MR and CT images of the same patients. These images were were rigidly registered to allow a voxel-wise comparison between synthesized CT and reference CT images. We do not expect this registration step to influence training, as training images were provided in a randomized unpaired way, making it unlikely that both an MR image and its registered corresponding CT image were simultaneously shown to the GAN. In addition, images were randomly cropped, which partially cancels the effects of rigid registration. Nevertheless, using images of the same patients in the MR set and the CT set may affect training. The synthesis networks could receive stronger feedback from the discriminator, which would occasionally see the corresponding reference image. In future work, we will extend the training set to investigate if we can similarly train the model with MR and CT images of *disjoint* patient sets.

Wolterink et al, Deep MR to CT Synthesis using Unpaired Data, SASHIMI 2017

Part 7 – Generative models

7.8 Difficulties and extensions

Difficulties with GANs

Vanishing gradients

- When the discriminator easily succeeds (in particular during the first steps of training), its gradient is close to zero
- Thus, no useful feedback to discriminator

Tentative solutions

- Make generator maximize $\log(D(G(z)))$ instead of minimizing $\log(1 - D(G(z)))$ (From original paper (Goodfellow et al, 2014))
- Wasserstein GANs (see after)

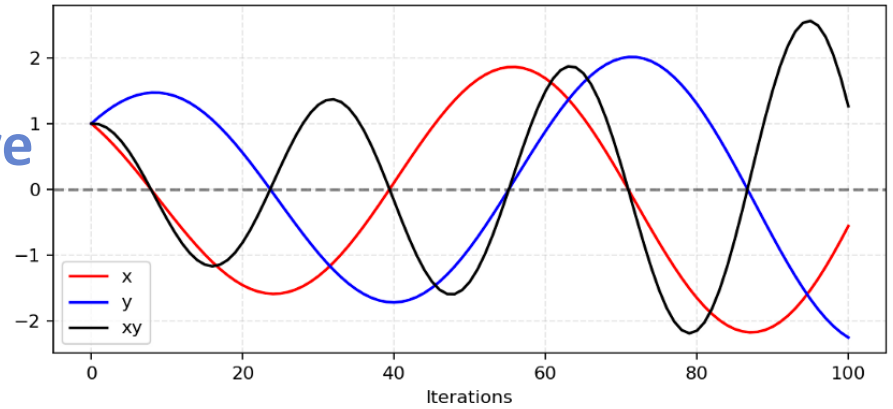
Difficulties with GANs

Convergence problems - Nash equilibrium may be hard to achieve

Toy example

$$\min_G \max_D V(D,G) = xy$$

Player G: Controls x to minimize $f_G(x) = xy$
Player D: Controls y to minimize $f_D(y) = -xy$

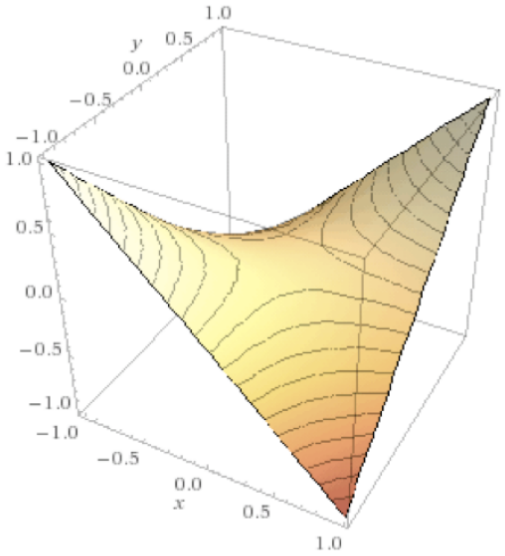


Values of xy, x and y for increasing iterations

Optimization

Gradients: $\frac{\partial f_G}{\partial x} = y$ and $\frac{\partial f_D}{\partial y} = -x$

Update x with $x - \eta y$ and y with $y + \eta x$



Visualization of V(x,y). Optimum is the saddle point at x=y=0

Images sources: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html
Goodfellow, NIPS Tutorial on GANs, 2016 - https://arxiv.org/pdf/1701.00160.pdf

Difficulties with GANs

Mode collapse

- Generator always produces a small set of outputs (or even always the same output)
  - Generator maps different values of z to the same output point

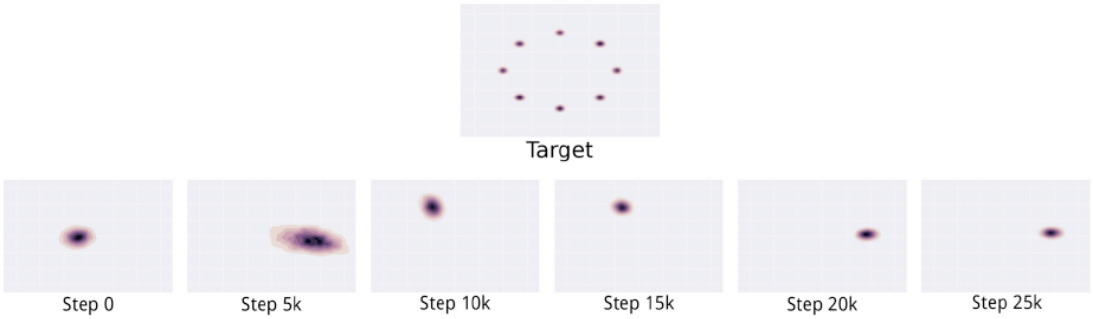


Figure 22: An illustration of the mode collapse problem on a two-dimensional toy dataset. In the top row, we see the target distribution p_{data} that the model should learn. It is a mixture of Gaussians in a two-dimensional space. In the lower row, we see a series of different distributions learned over time as the GAN is trained. Rather than converging to a distribution containing all of the modes in the training set, the generator only ever produces a single mode at a time, cycling between different modes as the discriminator learns to reject each one. Images from Metz *et al.* (2016).

Images sources: Goodfellow, NIPS Tutorial on GANs, 2016 - https://arxiv.org/pdf/1701.00160.pdf ; Metz et al, 2016

Difficulties with GANs

Mode collapse

- Generator always produces a small set of outputs (or even always the same output)
  - Generator maps different values of z to the same output point

Mode collapse

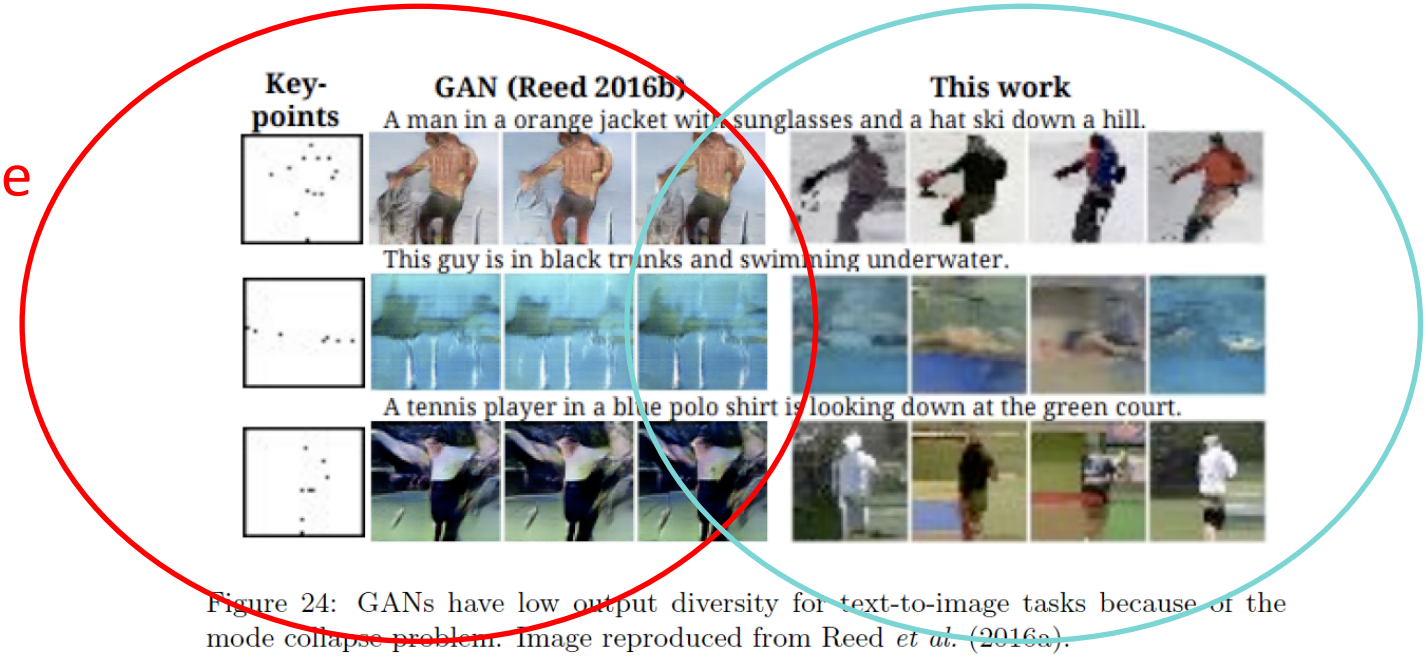


Figure 24: GANs have low output diversity for text-to-image tasks because of the mode collapse problem. Image reproduced from Reed *et al.* (2016a).

Better behavior

Images sources: Goodfellow, NIPS Tutorial on GANs, 2016 - https://arxiv.org/pdf/1701.00160.pdf ; Reed et al, 2016

Difficulties with GANs

Mode collapse

Tentative solutions

- Tune the learning rate
- Minibatch discrimination
  - Discriminator looks for similarity between images of a minibatch. Will penalize mode collapsing
- Wasserstein GANs
  - Allows to train the discriminator to optimality without vanishing gradients problems

Difficulties with GANs

Other tentative solutions to convergence problems

- One-sided label smoothing
  - Change the label of the real images from 1.0 to something smaller (e.g. 0.9)
- Feature matching
  - The discriminator inspects the features of the generated samples on top of doing discrimination
  - Supposedly less efficient than minibatch discrimination
- Virtual batch normalization
  - Each data sample is normalized based on a fixed batch ("reference batch") of data rather than within its minibatch. The reference batch is chosen once at the beginning and stays the same through the training.
- Adding noise to the samples
  - Increase the overlap between generated and real samples

Difficulties with GANs

Training with labels

- Using labels results in a massive improvement of subjective quality of generated samples
  - GAN conditioned on class labels (Denton et al, 2015)
  - Discriminator trained to recognize classes (Salimans et al, 2016)

Wasserstein GANs

- **Original GANs basically optimize Jensen-Shannon divergence**
  - Already convinced?

Wasserstein GANs

- **Original GANs basically optimize Jensen-Shannon divergence**

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

$$L(G,D) = \int_x \left(p_r(x)\log(D(x)) + p_g(x)\log(1-D(x)) \right) dx$$

Wasserstein GANs

- **Original GANs basically optimize Jensen-Shannon divergence**

Let D* the optimal discriminator

$$
\begin{aligned}
D_{JS}(p_r \| p_g) &= \frac{1}{2} D_{KL}(p_r || \frac{p_r + p_g}{2}) + \frac{1}{2} D_{KL}(p_g || \frac{p_r + p_g}{2}) \\
&= \frac{1}{2}\left(\log 2 + \int_x p_r(x) \log \frac{p_r(x)}{p_r + p_g(x)} dx \right) + \\
&\quad \frac{1}{2}\left(\log 2 + \int_x p_g(x) \log \frac{p_g(x)}{p_r + p_g(x)} dx \right) \\
&= \frac{1}{2}\left(\log 4 + L(G, D^*) \right)
\end{aligned}
$$

Thus,

$$L(G, D^*) = 2 D_{JS}(p_r \| p_g) - 2 \log 2$$

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
  - This often happens
  - Possible trick: add noise to make the distributions overlap

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs:** use Wasterstein-1 (Earth's mover) distance
  - Already know what this is?

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs:** use Wasterstein-1 (Earth's mover) distance

$$W(p_r, p_g) = \inf_{\gamma \in \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

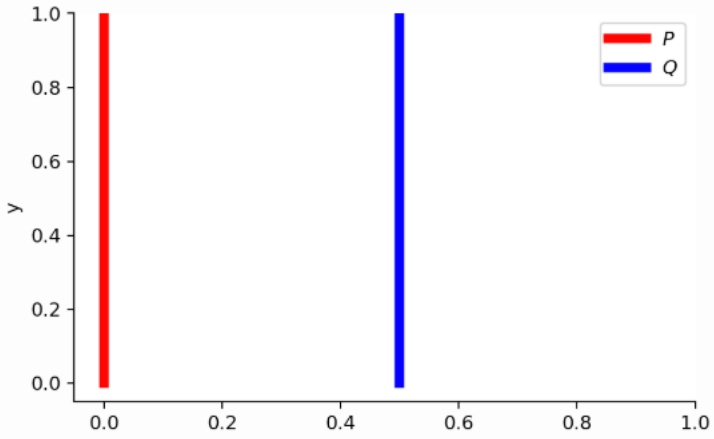
Transport Plan

All possible joint probability distributions

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs**: use Wasterstein-1 (Earth's mover) distance
  - Wassersein vs JS on a simple example

$$\forall (x, y) \in P, x = 0 \text{ and } y \sim U(0, 1)$$
$$\forall (x, y) \in Q, x = \theta, 0 \leq \theta \leq 1 \text{ and } y \sim U(0, 1)$$



When $\theta \neq 0$:

$$D_{KL}(P\|Q) = \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{KL}(Q\|P) = \sum_{x=\theta, y\sim U(0,1)} 1 \cdot \log \frac{1}{0} = +\infty$$

$$D_{JS}(P, Q) = \frac{1}{2}\left(\sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2} + \sum_{x=0, y\sim U(0,1)} 1 \cdot \log \frac{1}{1/2}\right) = \log 2$$

$$W(P, Q) = |\theta|$$

But when $\theta = 0$, two distributions are fully overlapped:

$$D_{KL}(P\|Q) = D_{KL}(Q\|P) = D_{JS}(P, Q) = 0$$
$$W(P, Q) = 0 = |\theta|$$

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs:** use Wasterstein-1 (Earth's mover) distance
  - Computation is intractable in practice

$$W(p_r, p_g) = \inf_{\gamma \sim \Pi(p_r, p_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x - y\|]$$

Transport Plan

All possible joint probability distributions

Wasserstein GANs

- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs:** use Wasterstein-1 (Earth's mover) distance
  - Computation is intractable in practice
  - Use **Kantorovich-Rubinstein duality**

$$W(p_r, p_g) = \frac{1}{K} \sup_{\|f\|_L \leq K} \mathbb{E}_{x \sim p_r}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

  - Wasserstein GAN

$$L(p_r, p_g) = W(p_r, p_g) = \max_{w \in W} \mathbb{E}_{x \sim p_r}[f_w(x)] - \mathbb{E}_{z \sim p_r(z)}[f_w(g_\theta(z))]$$

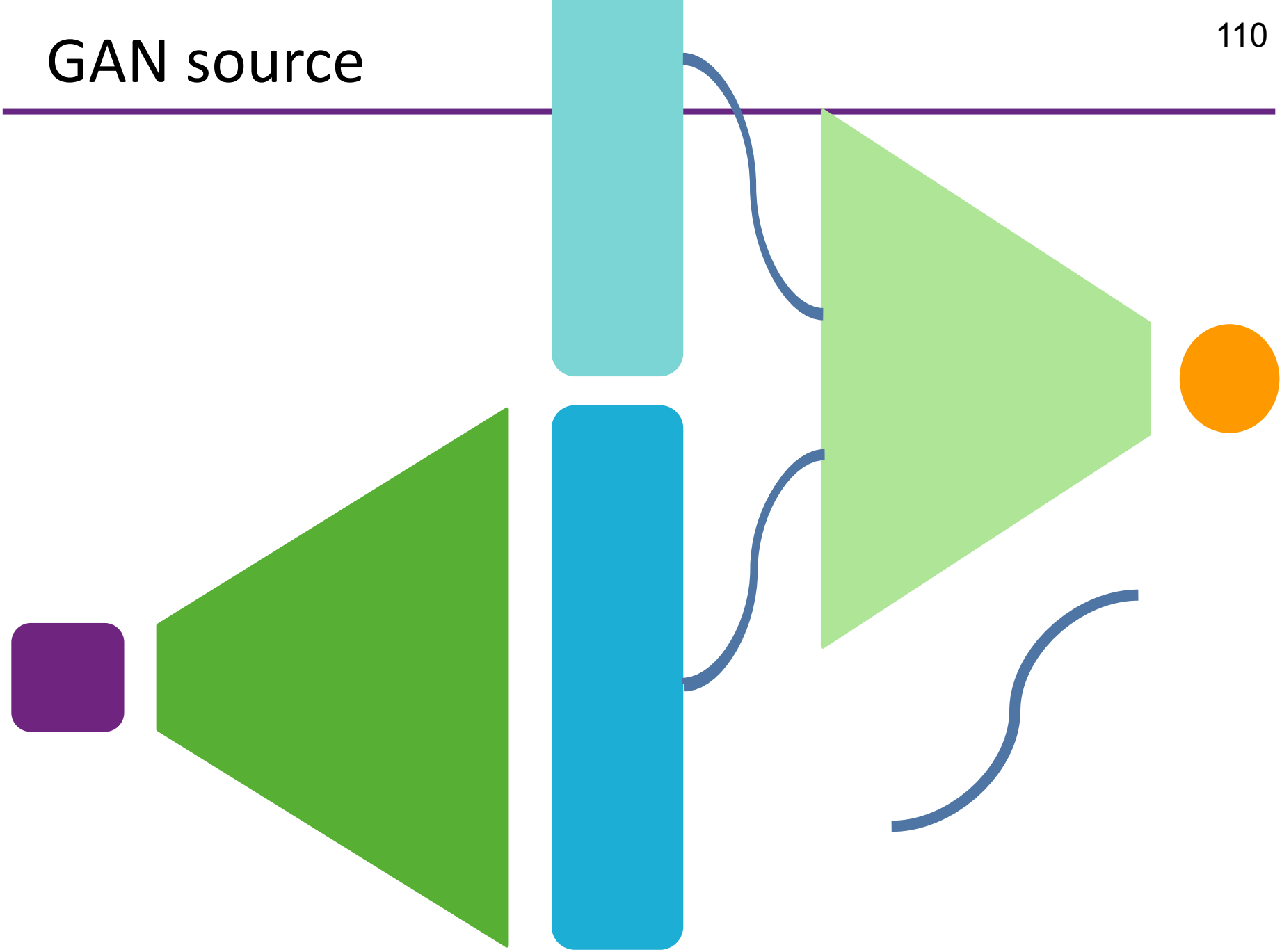
  - The discriminator is trained to learn the function f_W (parametrized by some weights w)

Wasserstein GANs

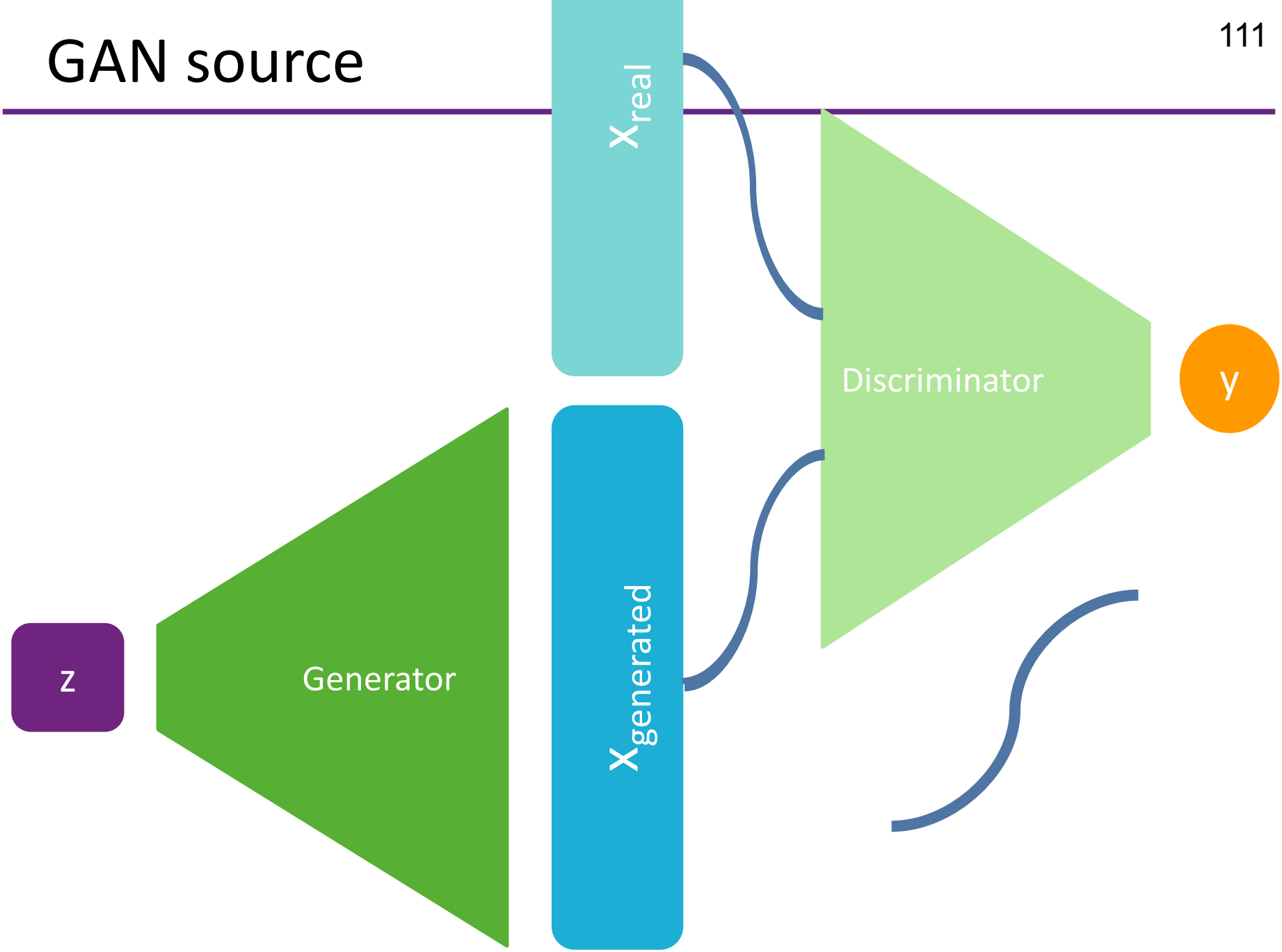
- Original GANs basically optimize **Jensen-Shannon(JS) divergence**
- Problem with JS: saturates when distributions are disjoint
- **Wasserstein GANs:** use Wasterstein-1 (Earth's mover) distance
  - Computation is intractable in practice
  - Use **Kantorovich-Rubinstein duality**
  - The discriminator is trained to learn the function f_w (parametrized by some weights w)
  - Problem: f_w needs to be **K-Lipschitz continuous**. $\|f\|_L \leq K$
  - (Unperfect) solutions:
    - Weight clipping
    - Gradient penalty

In vectorial format

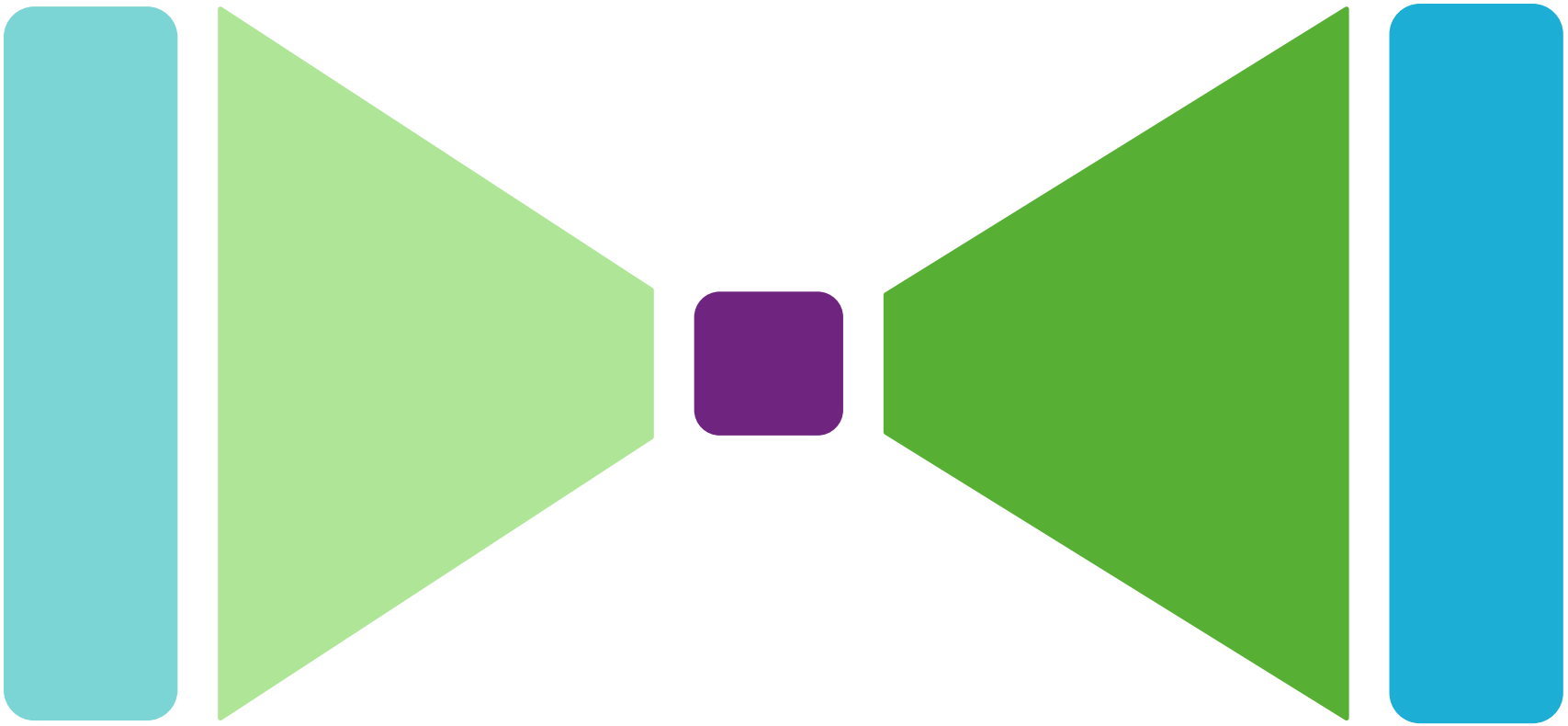
GAN source



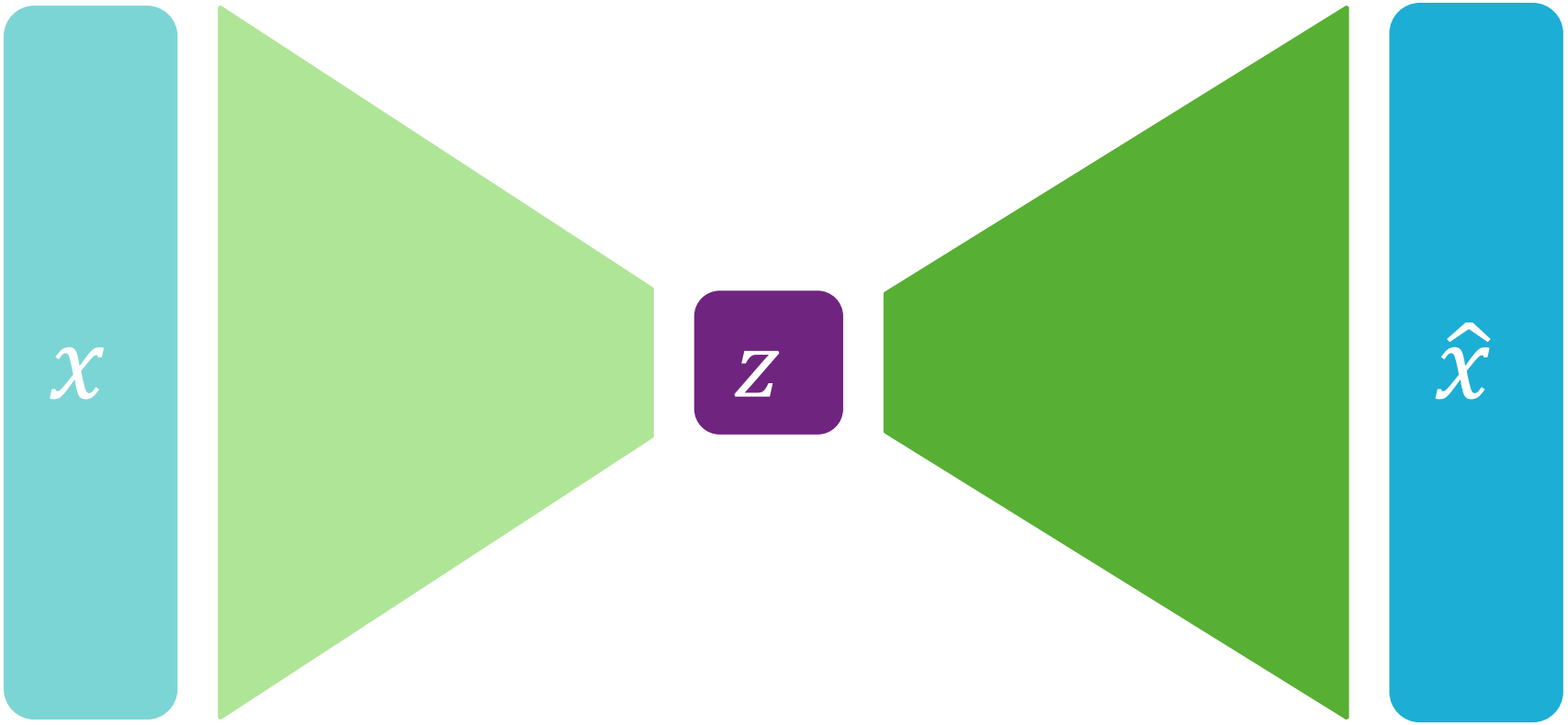
GAN source



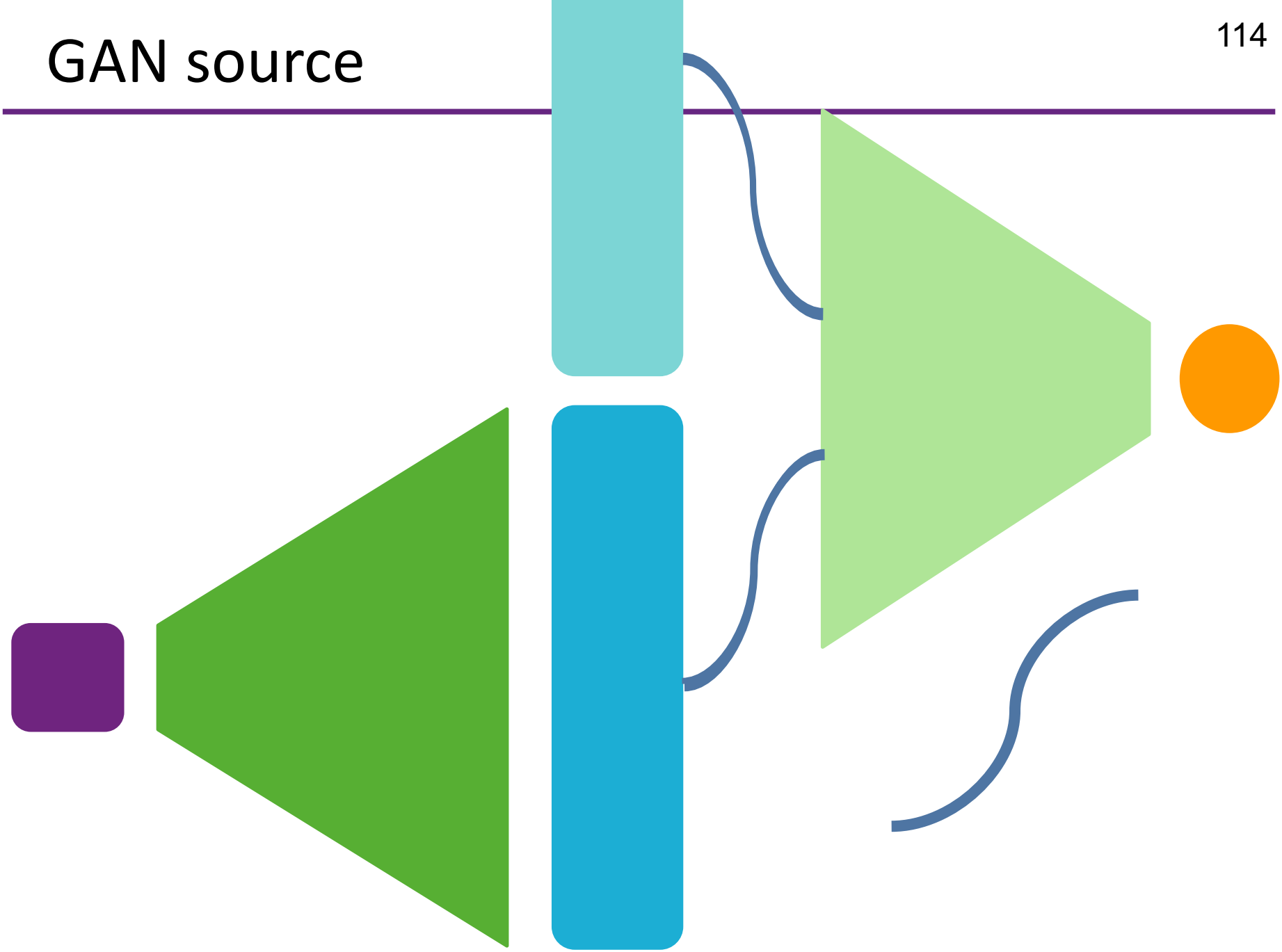
Autoencoder source



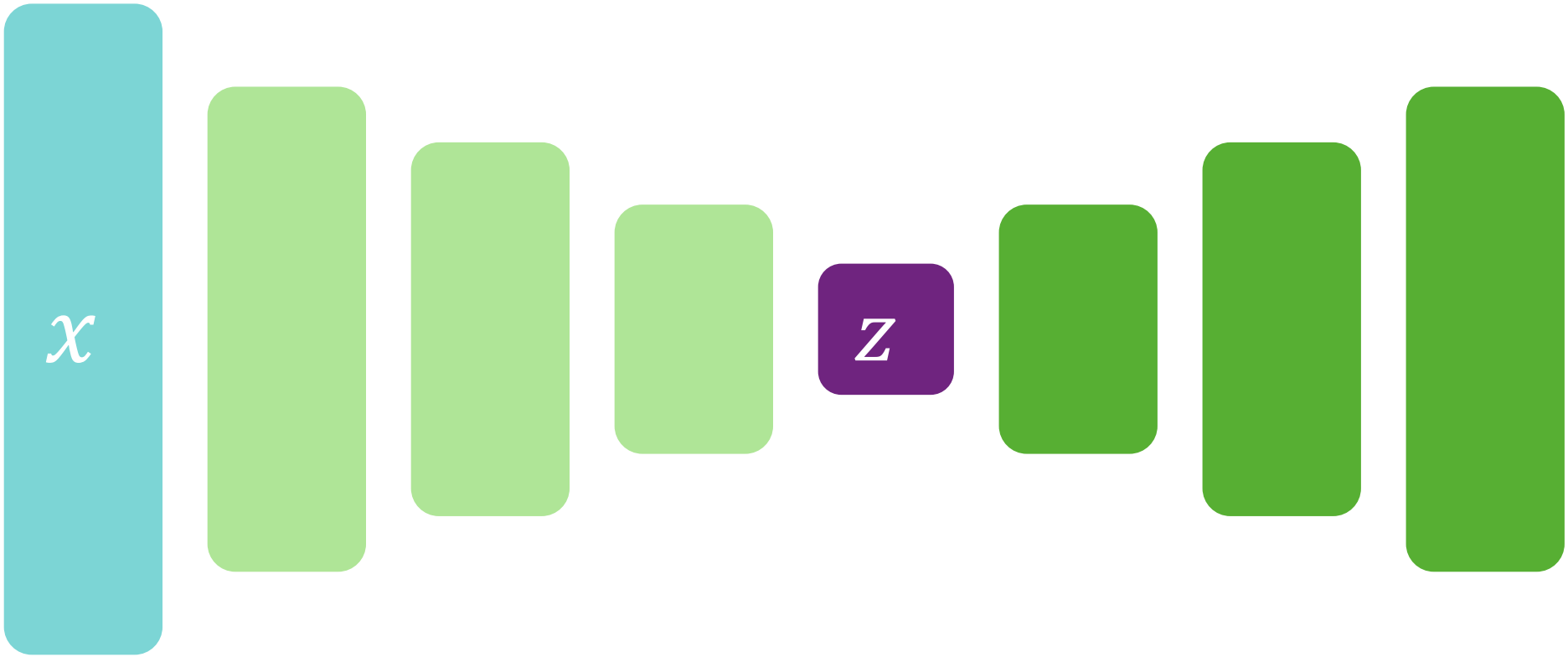
Autoencoder source with notations



GAN source

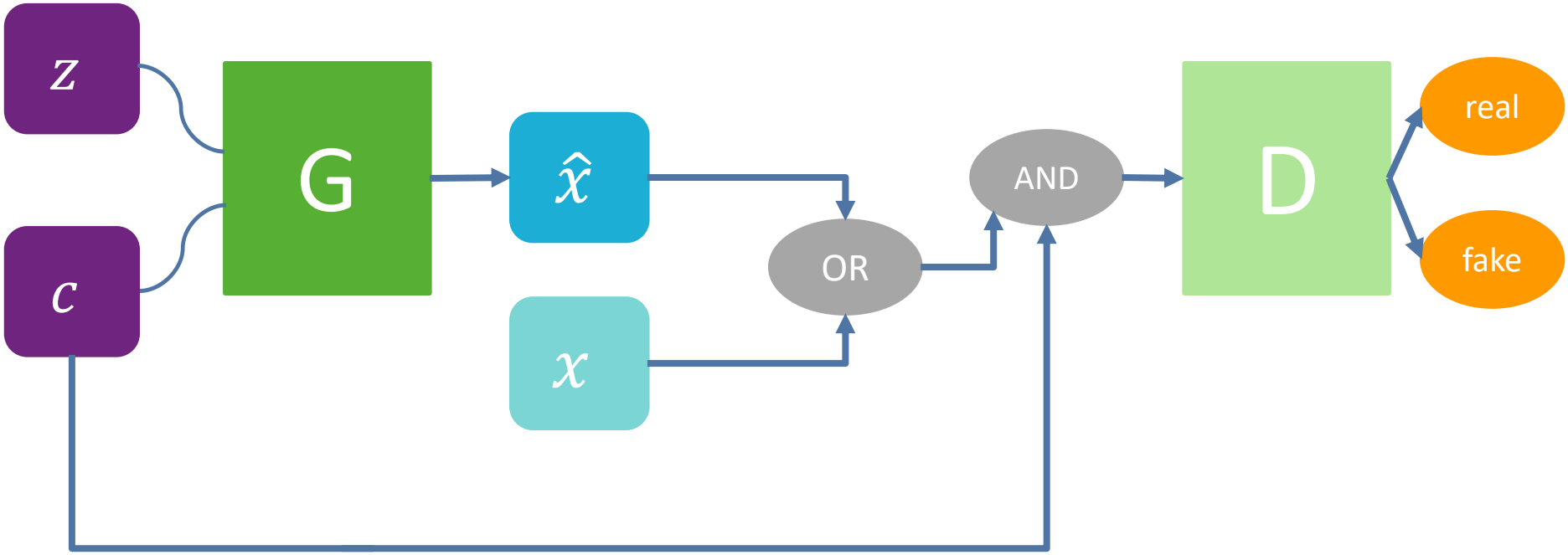


Autoencoders source



Adapted fome http://introtodeeplearning.com/

cGAN source



CycleGAN source