GANs for cross-modality image synthesis

Anders Eklund, David Abramian

anders.eklund@liu.se

Department of Biomedical Engineering (IMT)
Department of Computer and Information Science (IDA)
Center for Medical Image Science and Visualization (CMIV)
Linköping University, Sweden

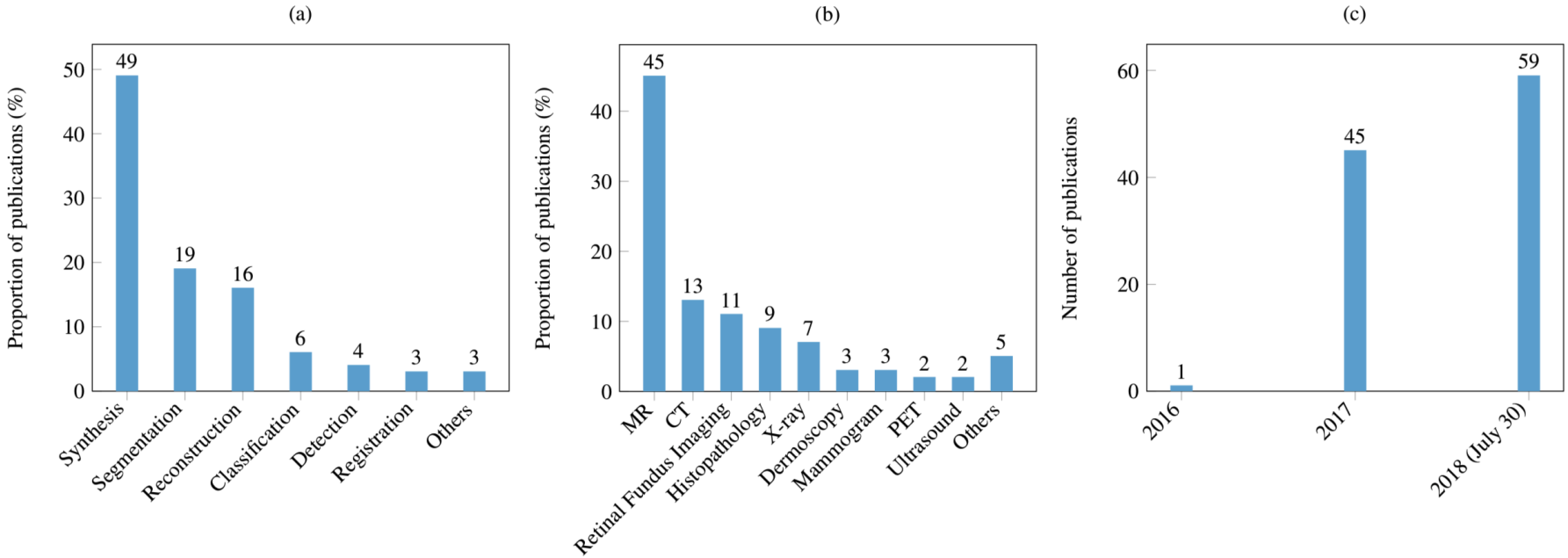
Conflicts of interest

- We have received free graphics cards from Nvidia

Outline

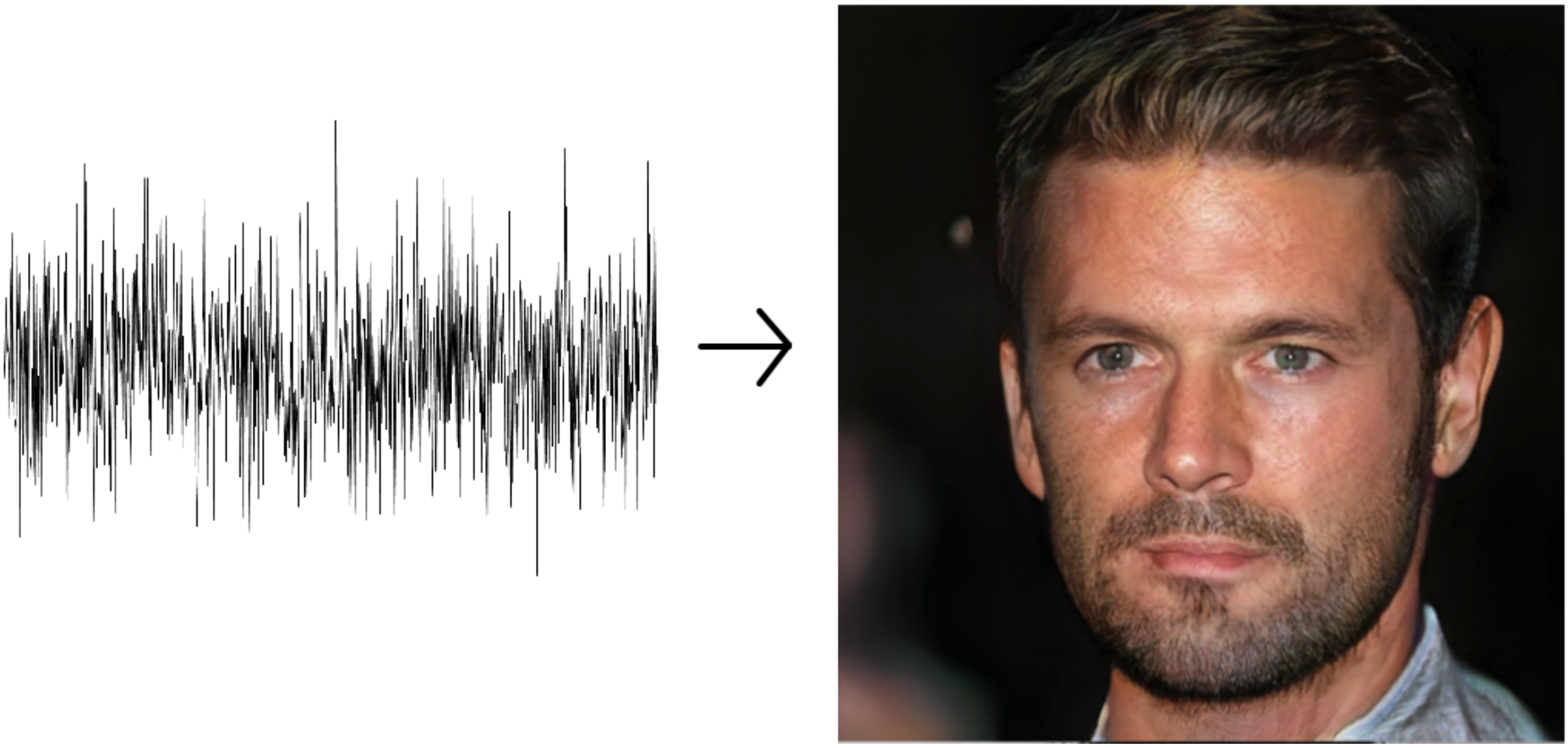
- Noise-to-image GANs vs image-to-image GANs
- Paired training, pix2pix
- Unpaired training, CycleGAN
- Applications
  - T1 to T2, CT to MR, MR to PET, ...
  - Distortion correction
  - Abnormality detection
- Challenges
  - From 2D to 3D GANs
  - Making it work for different scanners
  - Training on controls, using for diseased?

GAN OVERVIEW



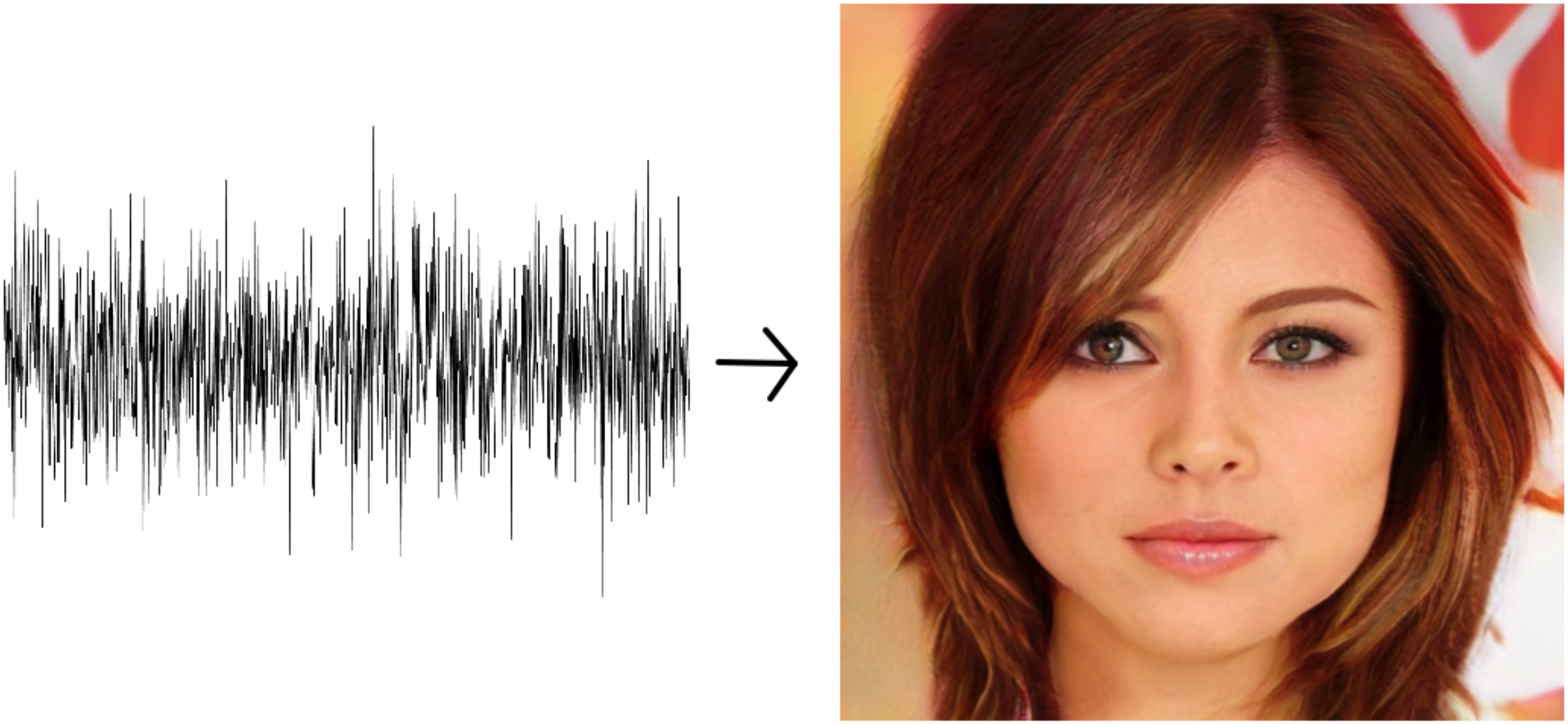
Yi, X., Walia, E., & Babyn, P. (2018). Generative adversarial network in medical imaging: A review. arXiv:1809.07294

NOISE-TO-IMAGE GAN



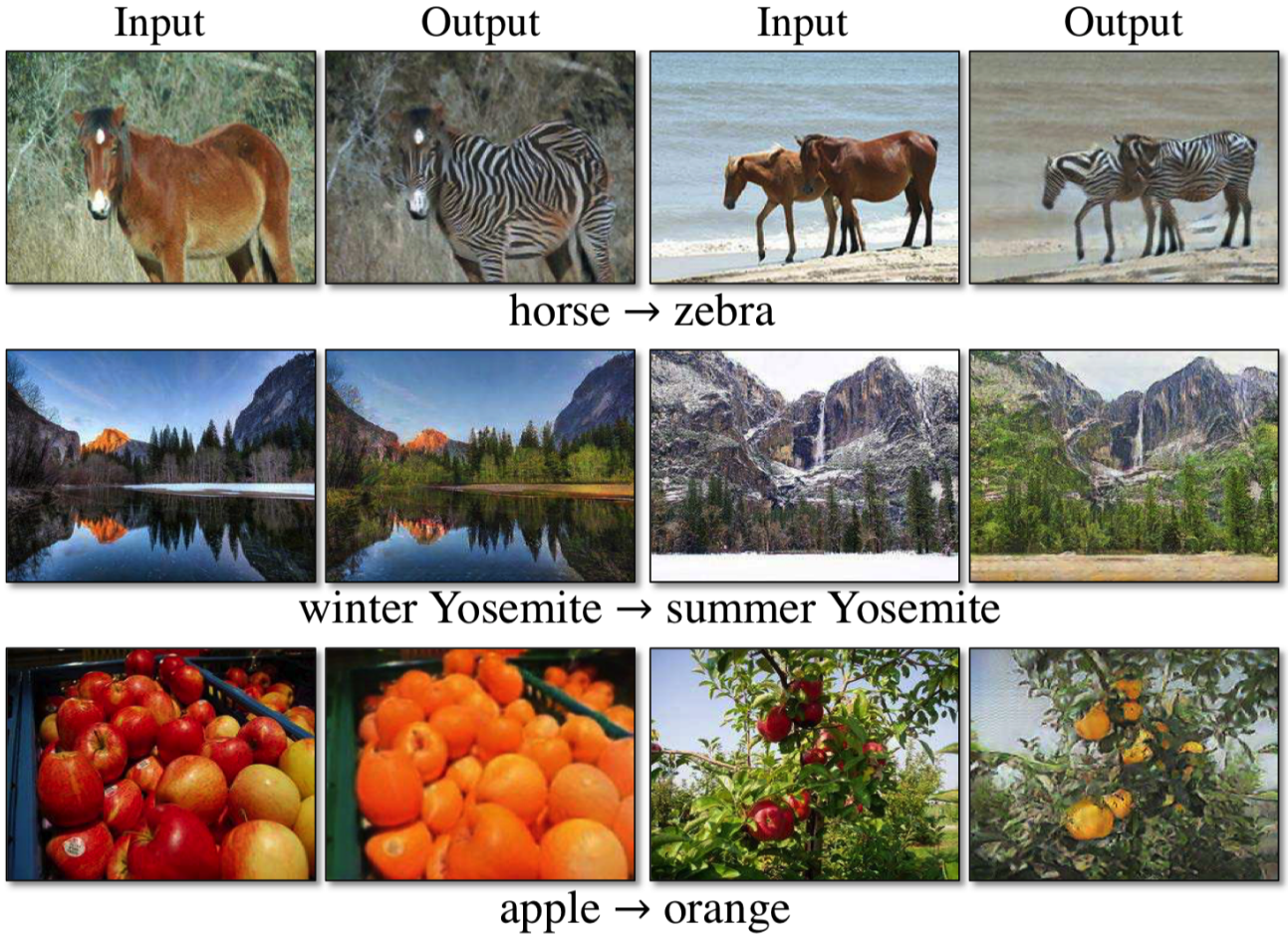
Karras, T., Aila, T., Laine, S., & Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation, ICLR, 2018

NOISE-TO-IMAGE GAN



Karras, T., Aila, T., Laine, S., & Lehtinen, J. Progressive growing of GANs for improved quality, stability, and variation, ICLR, 2018

IMAGE-TO-IMAGE GAN (CONDITIONAL GANS)



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV, 2017

IMAGE-TO-IMAGE GAN - IN SNAPCHAT

VRSCOUT NEWS

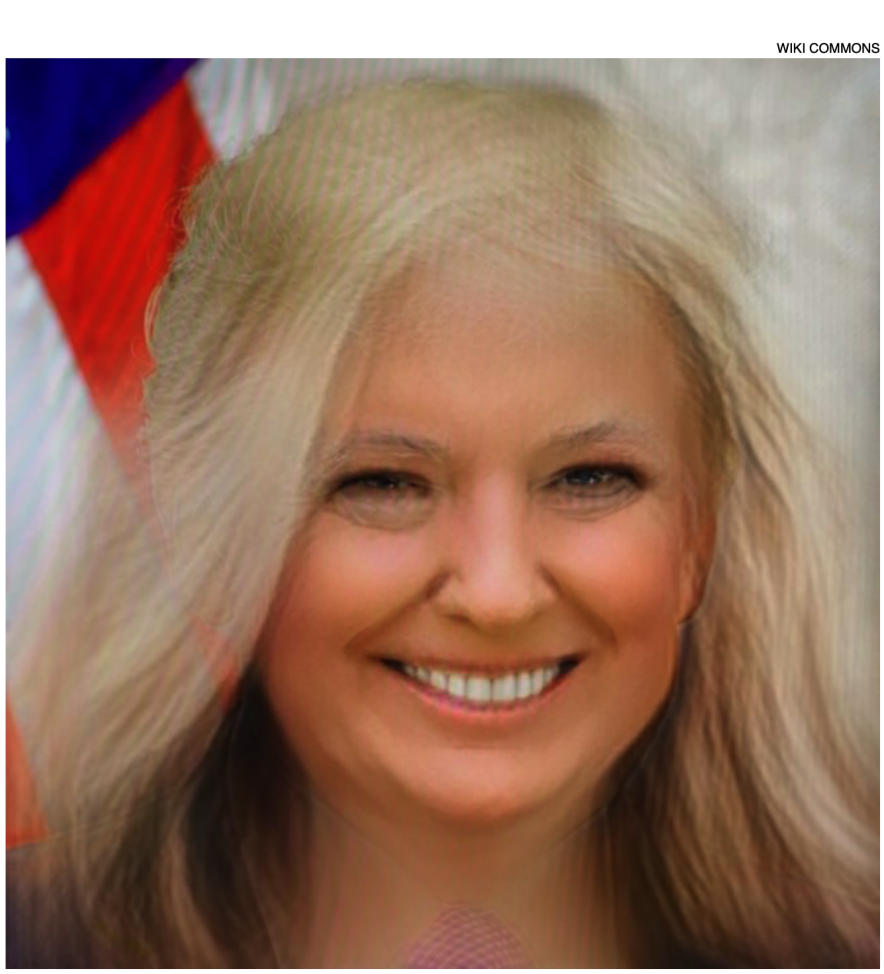
News

Snapchat's Gender Swap AR Face Filter Is A Big Hit

May 13, 2019 • by Bobby Carlton

https://vrscout.com/news/snapchat-gender-swap-ar-face-filter/

IMAGE-TO-IMAGE GAN - IN SNAPCHAT

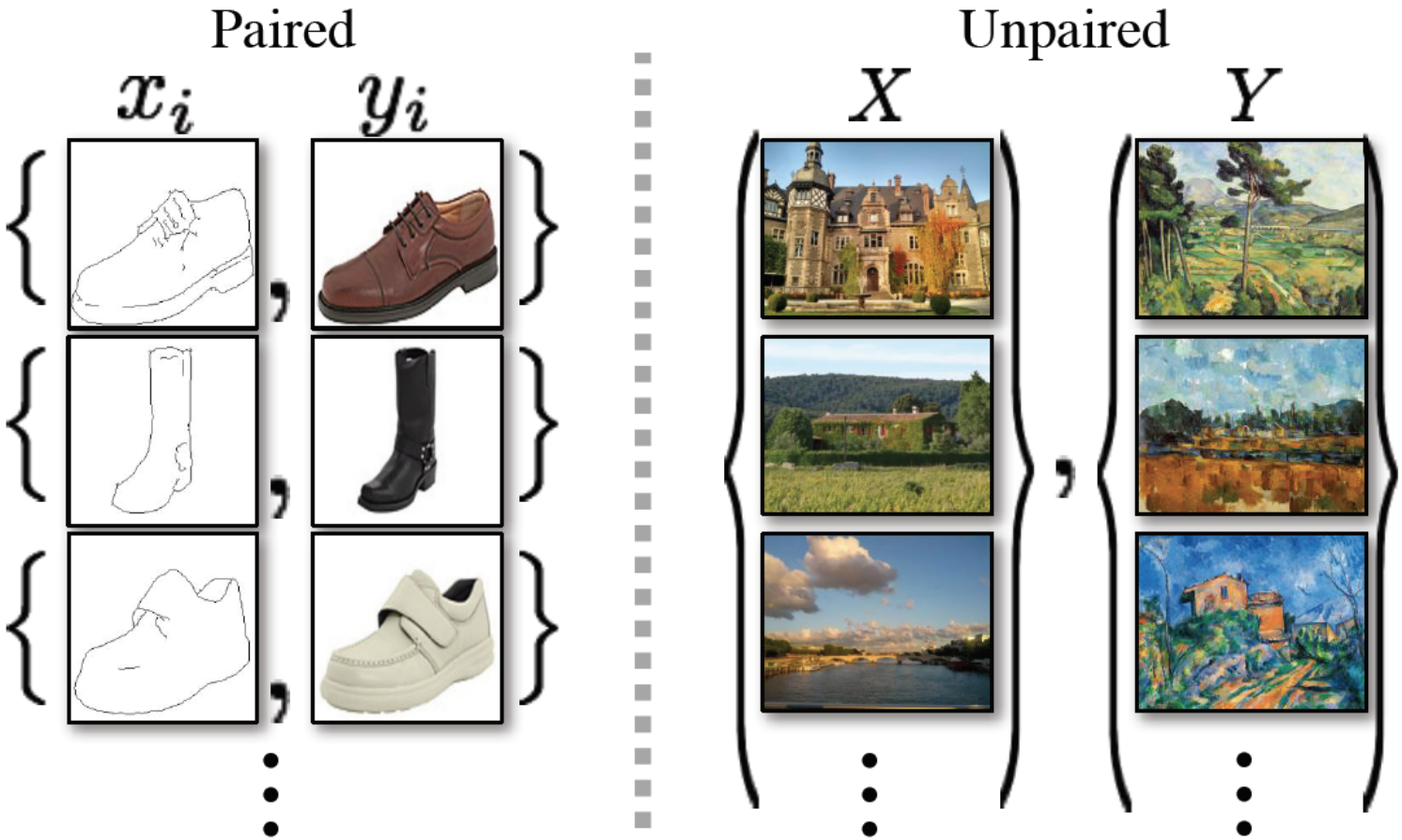


https://www.pocket-lint.com/apps/news/snapchat/148095-snapchat-gender-swap-how-to-do-it-and-check-out-these-crazy-celebrity-results

SUPERVISED VS UNSUPERVISED GANS

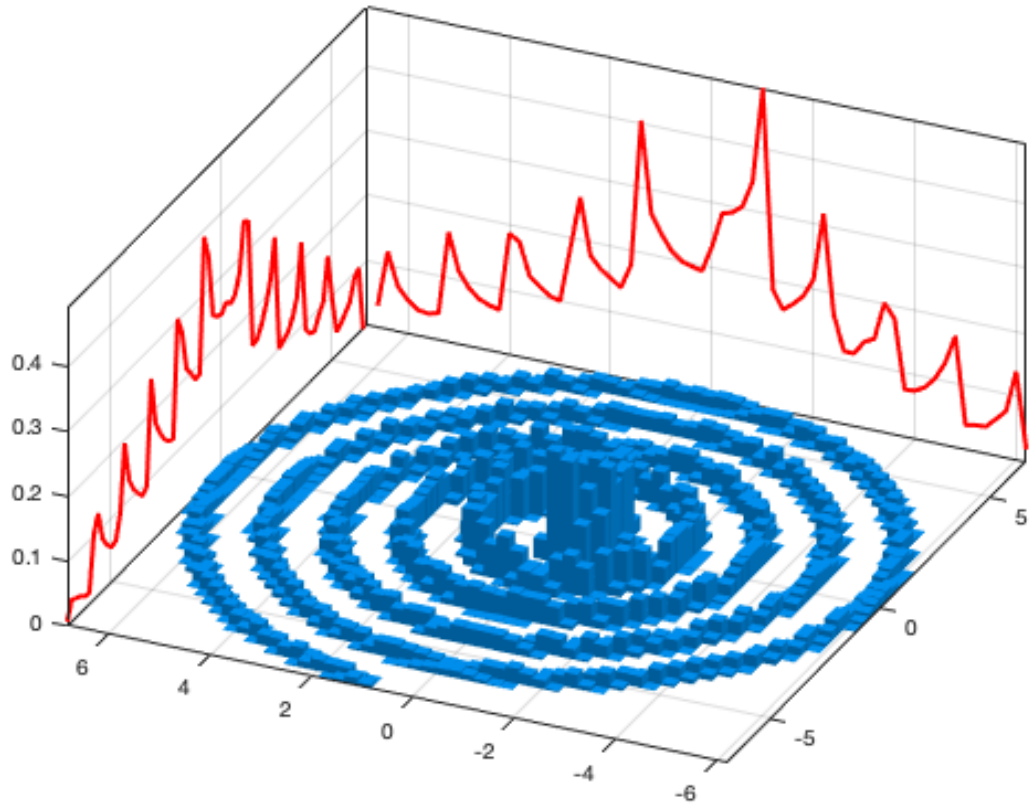
- We have images from two domains, X & Y
- We can train a GAN to convert from domain X to domain Y
  - If we have every image represented in both domains (x1; y1); (x2; y2); we can train a *supervised* GAN, which learns from the joint distribution of the data in both domains
  - If we have unmatched images in both domains (x1; ?); (?; y2); we can train an *unsupervised* GAN, which learns from the marginal distributions of the data in both domains.

PAIRED AND UNPAIRED DATA



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV, 2017

SUPERVISED VS UNSUPERVISED GANS



Supervised GAN: Learn the blue distribution
from paired examples from the blue distribution

Unsupervised GAN: Learn the blue distribution
from examples from the red distributions

PAIRED TRAINING - PIX2PIX

Image-to-Image Translation with Conditional Adversarial Networks

Phillip Isola Jun-Yan Zhu Tinghui Zhou Alexei A. Efros

Berkeley AI Research (BAIR) Laboratory, UC Berkeley

{isola,junyanz,tinghuiz,efros}@eecs.berkeley.edu

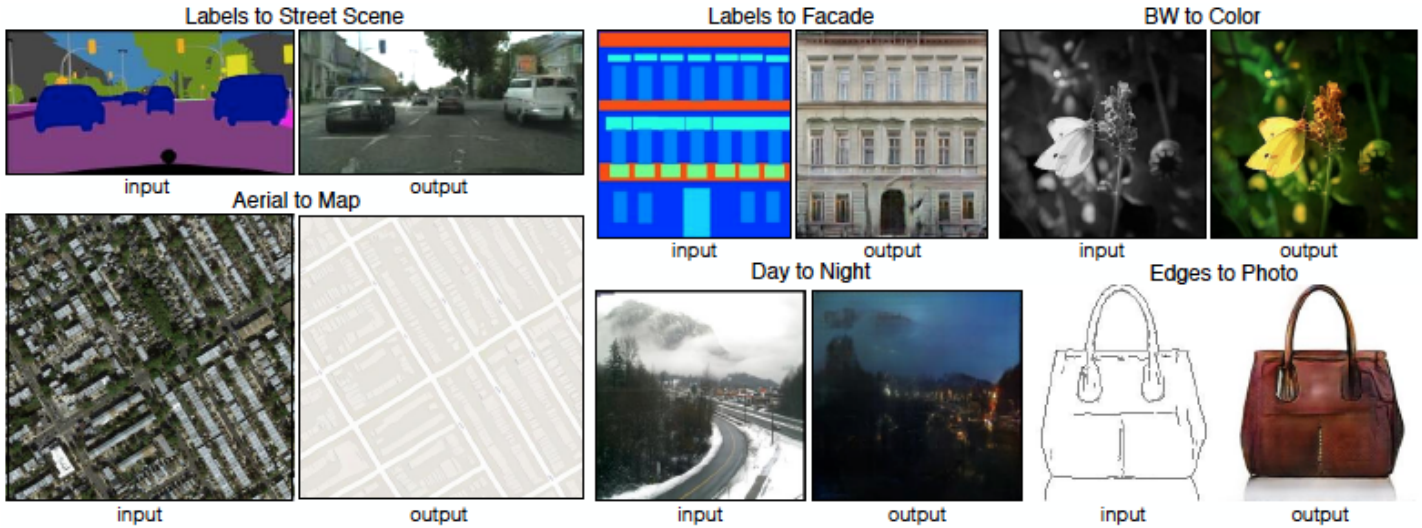
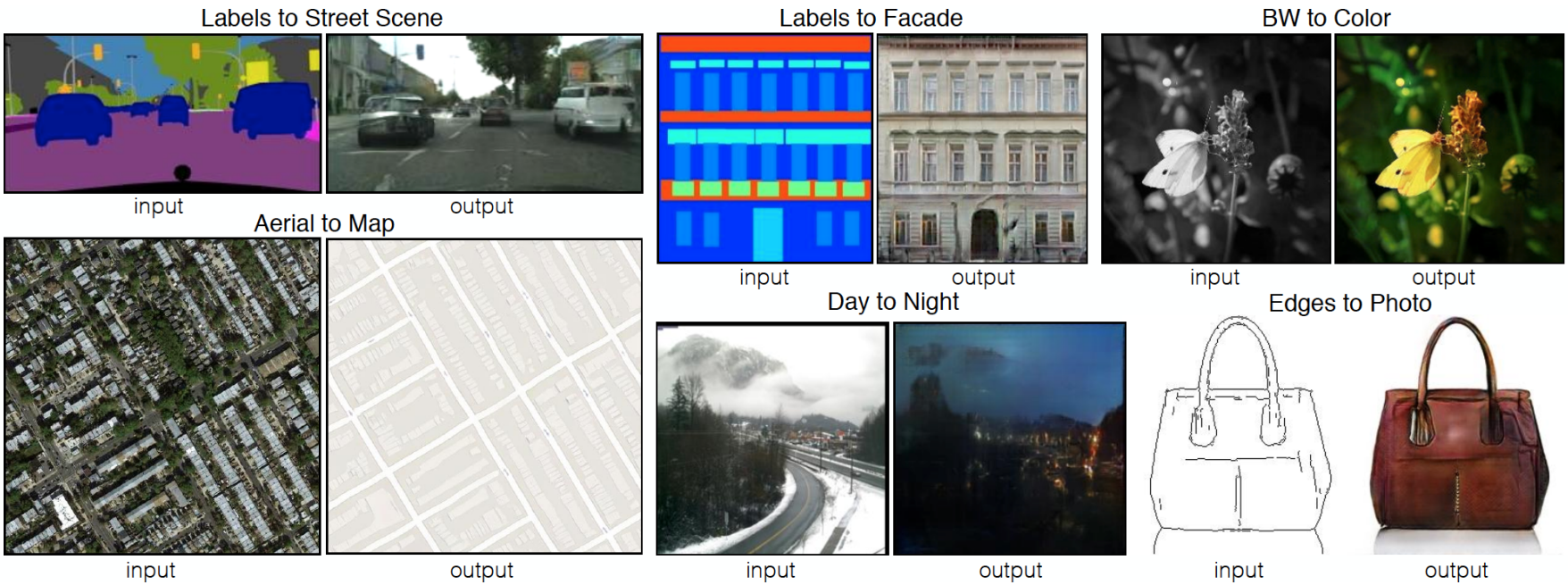


Figure 1: Many problems in image processing, graphics, and vision involve translating an input image into a corresponding output image. These problems are often treated with application-specific algorithms, even though the setting is always the same: map pixels to pixels. Conditional adversarial nets are a general-purpose solution that appears to work well on a wide variety of these problems. Here we show results of the method on several. In each case we use the same architecture and objective, and simply train on different data.

Isola et al., Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017

PAIRED TRAINING - PIX2PIX

- Given images in two domains, X and Y, the GAN learns to convert an image in domain X to domain Y
- Drawback: We need paired (and registered) training images

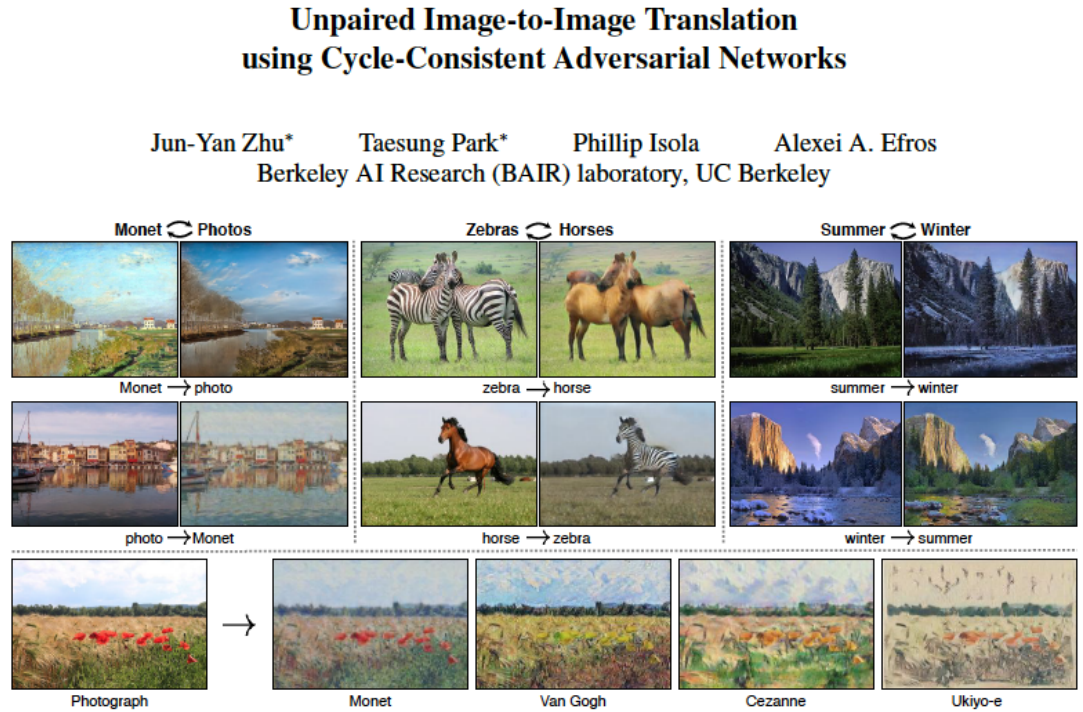


UNPAIRED TRAINING - CYCLEGAN

- Unsupervised image-to-image translation is more difficult
- Learning a joint distribution from marginal distributions is a vastly underdetermined problem, need regularization.
- One solution: CycleGAN. Learn transformations in both directions $X \leftrightarrow Y$ and impose a consistency constraint in forward-backward conversion
- $$\begin{cases} x_i \to \widehat{y}_i \to \widetilde{x}_i \approx x_i \\ y_i \to \widehat{x}_i \to \widetilde{y}_i \approx y_i \end{cases}$$

UNPAIRED TRAINING - CYCLEGAN

- Given images in two domains, X and Y, CycleGAN learns to convert an image in domain X to domain Y *and* to convert an image in domain Y to domain X



Zhu et al., Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV, 2017

Loss functions for training

- Mappings $G : X -> Y$ and $F : Y -> X$, discriminators D_X and D_Y
- Adversarial loss function (GAN), L_{GAN}

  $$L_{GAN}(G, D_Y, X, Y) = E\left[logD_Y(y)\right] + E\left[log(1 - D_Y(G(x))\right]$$

- Cycle consistency loss function (CycleGAN), L_{cyc}

  $$L_{cyc}(G, F) = E\left[||F(G(x)) - x||_1\right] + E\left[||G(F(y)) - y||_1\right]$$

- Full loss function for CycleGAN

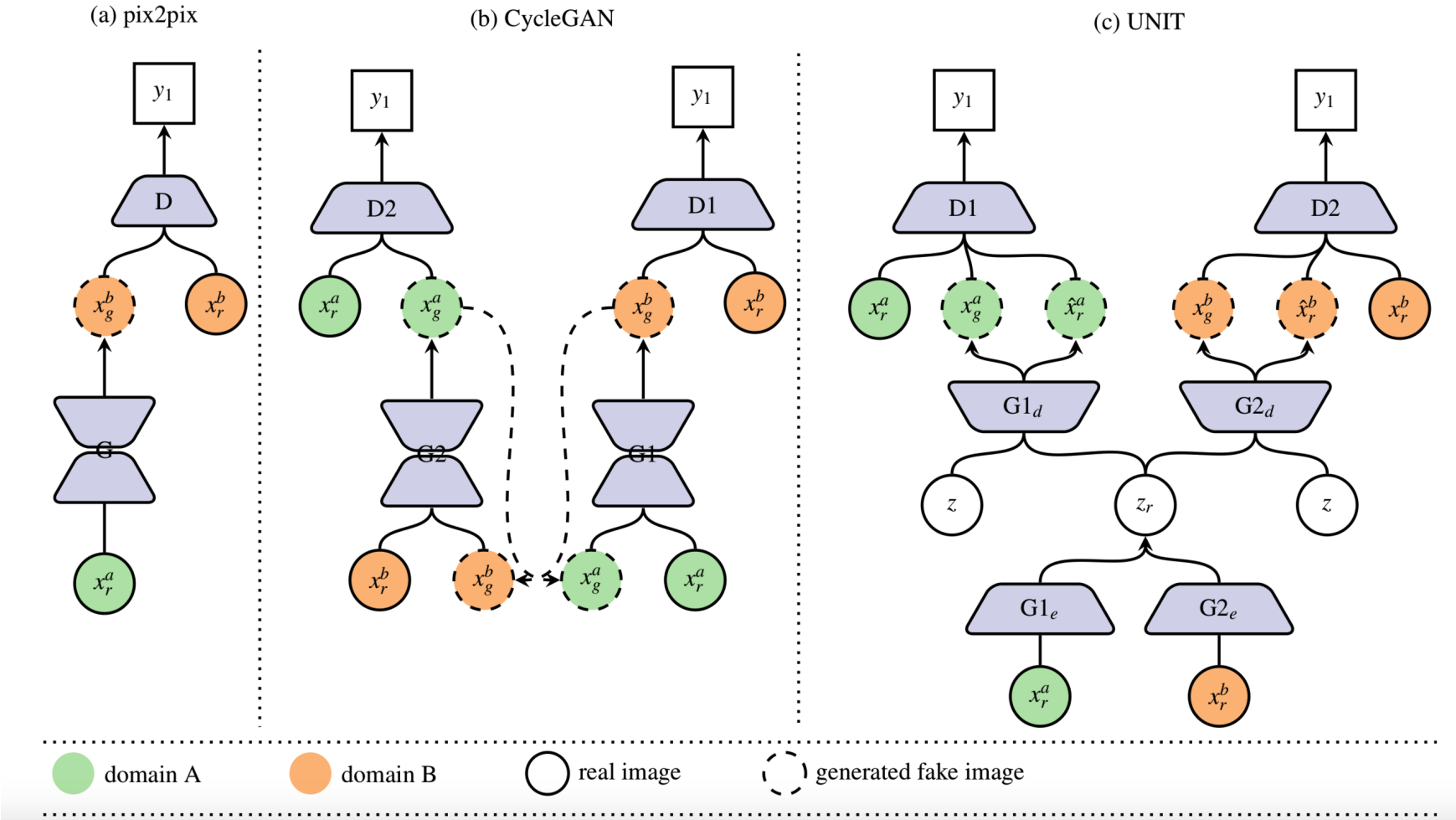
  $$L(G, F, D_X, D_Y) = L_{GAN}(G, D_Y, X, Y) + L_{GAN}(F, D_X, Y, X) + \lambda L_{cyc}(G, F)$$

Loss functions

Abbr.	**Losses**	**Requirement**
L1	$\mathcal{L}_{\text{adversarial}}$	–
L2	$\mathcal{L}_{\text{image}}$	Aligned training pair
L3	$\mathcal{L}_{\text{cycle}}$	–
L4	$\mathcal{L}_{\text{gradient}}$	Aligned training pair
L5	$\mathcal{L}_{\text{edge}}$	Aligned training pair
L6	$\mathcal{L}_{\text{sharp}}$	Aligned training pair
L7	$\mathcal{L}_{\text{shape}}, \mathcal{L}_{\text{seg}}$	Annotated pixel-wise label
L8	$\mathcal{L}_{\text{perceptual}}$	Aligned training pair
L9	$\mathcal{L}_{\text{structure}}$	Aligned training pair
L10	$\mathcal{L}_{\text{style-content}}$	Aligned training pair
L11	$\mathcal{L}_{\text{self-reg}}$	–
L12	$\mathcal{L}_{\text{steer}}$	Aligned training pair
L13	$\mathcal{L}_{\text{classify}}$	Aligned image-wise label
L14	$\mathcal{L}_{\text{frequency}}$	Aligned training pair
L15	$\mathcal{L}_{\text{KL}}$	–
L16	$\mathcal{L}_{\text{saliency}}$	Aligned training pair
L17	$\mathcal{L}_{\text{physical}}$	Physical model
L18	$\mathcal{L}_{\text{regulation}}$	–

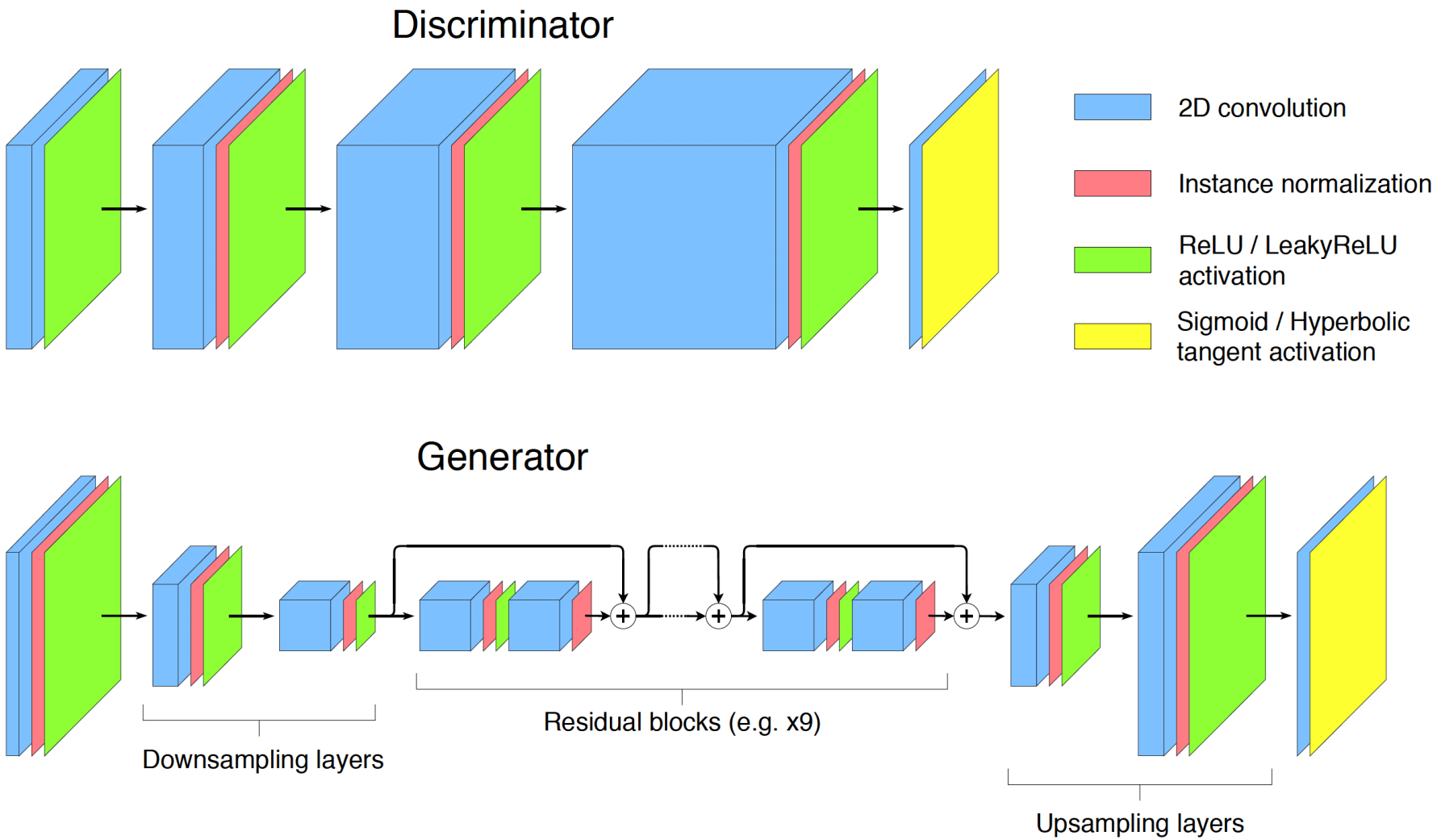
Yi, X., Walia, E., & Babyn, P. (2018). Generative adversarial network in medical imaging: A review. arXiv:1809.07294

PIX2PIX VS CYCLEGAN VS UNIT



Yi, X., Walia, E., & Babyn, P. (2018). Generative adversarial network in medical imaging: A review. arXiv:1809.07294

CycleGAN - detailed architecture



CycleGAN - generator

- What do the generator and the discriminator look like in Keras?

Generator architectures We adopt our architectures from Johnson et al. [23]. We use 6 residual blocks for 128×128 training images, and 9 residual blocks for 256×256 or higher-resolution training images. Below, we follow the naming convention used in the Johnson et al.'s Github repository.

Let `c7s1-k` denote a 7×7 Convolution-InstanceNorm-ReLU layer with k filters and stride 1. `dk` denotes a 3×3 Convolution-InstanceNorm-ReLU layer with k filters and stride 2. Reflection padding was used to reduce artifacts. `Rk` denotes a residual block that contains two 3×3 convolutional layers with the same number of filters on both layer. `uk` denotes a 3×3 fractional-strided-Convolution-InstanceNorm-ReLU layer with k filters and stride $\frac{1}{2}$.

The network with 6 residual blocks consists of:
`c7s1-64,d128,d256,R256,R256,R256,R256,R256,R256,u128,u64,c7s1-3`

The network with 9 residual blocks consists of:
`c7s1-64,d128,d256,R256,R256,R256,R256,R256,R256,R256,R256,u128 u64,c7s1-3`

```
def modelGenerator(self, name=None):
    # Specify input
    input_img = Input(shape=self.img_shape)
    # Layer 1
    x = ReflectionPadding2D((3, 3))(input_img)
    x = self.c7Ak(x, 32)
    # Layer 2
    x = self.dk(x, 64)
    # Layer 3
    x = self.dk(x, 128)

    # Layer 4-12: Residual layer
    for _ in range(4, 13):
        x = self.Rk(x)

    # Layer 13
    x = self.uk(x, 64)
    # Layer 14
    x = self.uk(x, 32)
    x = ReflectionPadding2D((3, 3))(x)
    x = Conv2D(self.channels, kernel_size=7, strides=1)(x)
    x = Activation('tanh')(x)  # They say they use Relu but really they do not
    return Model(inputs=input_img, outputs=x, name=name)
```

- https://github.com/simontomaskarlsson/CycleGAN-Keras

CycleGAN - help functions

```
def Rk(self, x0):
    k = int(x0.shape[-1])
    # first layer
    x = ReflectionPadding2D((1,1))(x0)
    x = Conv2D(filters=k, kernel_size=3, strides=1, padding='valid')(x)
    x = self.normalization(axis=3, center=True, epsilon=1e-5)(x, training=True)
    x = Activation('relu')(x)
    # second layer
    x = ReflectionPadding2D((1, 1))(x)
    x = Conv2D(filters=k, kernel_size=3, strides=1, padding='valid')(x)
    x = self.normalization(axis=3, center=True, epsilon=1e-5)(x, training=True)
    # merge
    x = add([x, x0])
    return x
def dk(self, x, k):
    x = Conv2D(filters=k, kernel_size=3, strides=2, padding='same')(x)
    x = self.normalization(axis=3, center=True, epsilon=1e-5)(x, training=True)
    x = Activation('relu')(x)
    return x
```

CycleGAN - discriminator

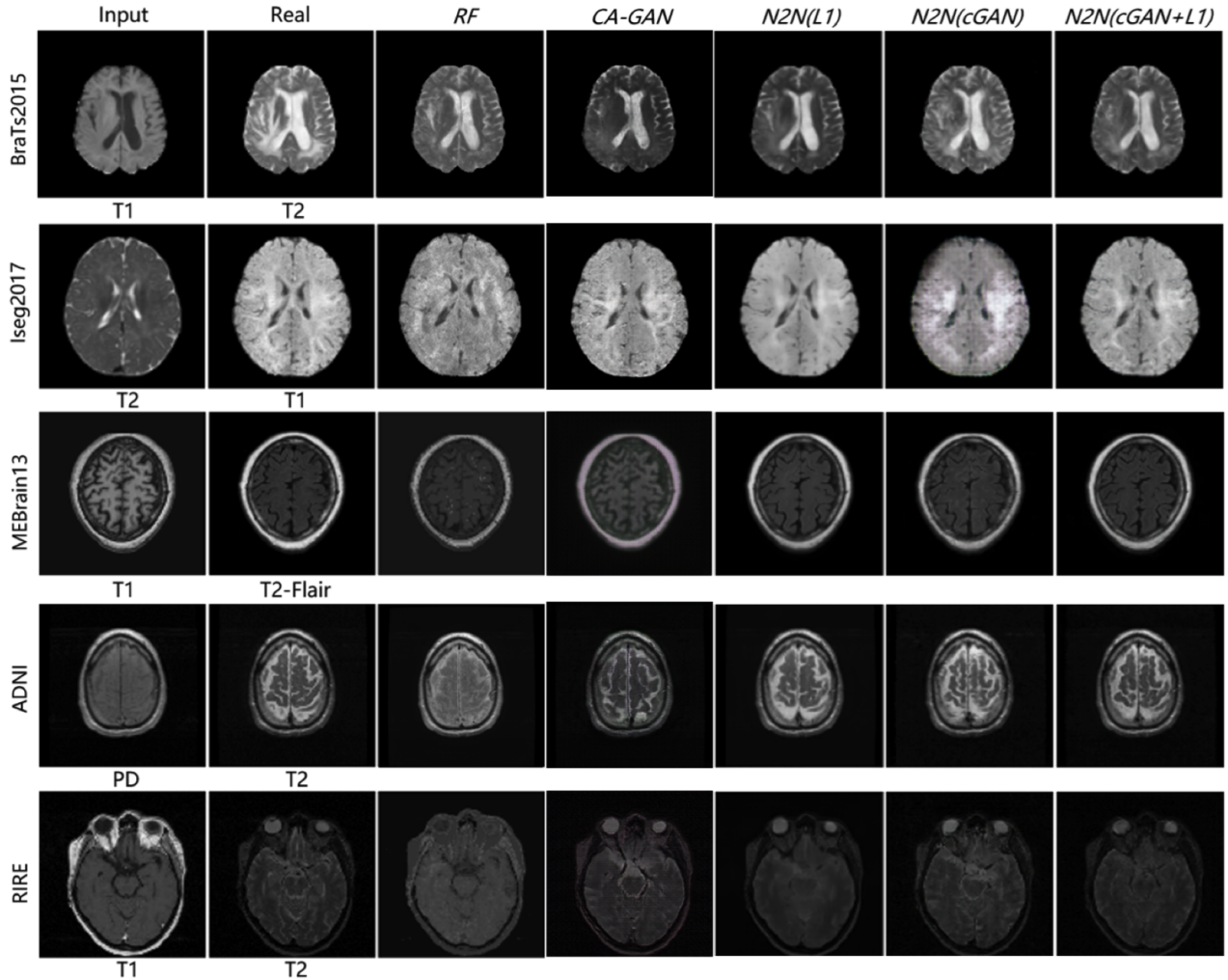
- The discriminator has fewer layers

Discriminator architectures For discriminator networks, we use 70×70 PatchGAN [22]. Let `Ck` denote a 4×4 Convolution-InstanceNorm-LeakyReLU layer with k filters and stride 2. After the last layer, we apply a convolution to produce a 1-dimensional output. We do not use InstanceNorm for the first `C64` layer. We use leaky ReLUs with a slope of 0.2. The discriminator architecture is:

- `C64-C128-C256-C512`

```
def modelDiscriminator(self, name=None):
    # Specify input
    input_img = Input(shape=self.img_shape)
    # Layer 1 (#Instance normalization is not used for this layer)
    x = self.ck(input_img, 64, False)
    # Layer 2
    x = self.ck(x, 128, True)
    # Layer 3
    x = self.ck(x, 256, True)
    # Layer 4
    x = self.ck(x, 512, True)
    # Output layer
    if self.use_patchgan:
        x = Conv2D(filters=1, kernel_size=4, strides=1, padding='same')(x)
    else:
        x = Flatten()(x)
        x = Dense(1)(x)
    x = Activation('sigmoid')(x)
    return Model(inputs=input_img, outputs=x, name=name)
```

APPLICATIONS - MR T1 TO MR T2



Yang et al. (2018). MRI Cross-Modality NeuroImage-to-NeuroImage Translation, arXiv:1801.06940.

APPLICATIONS - MR TO CT

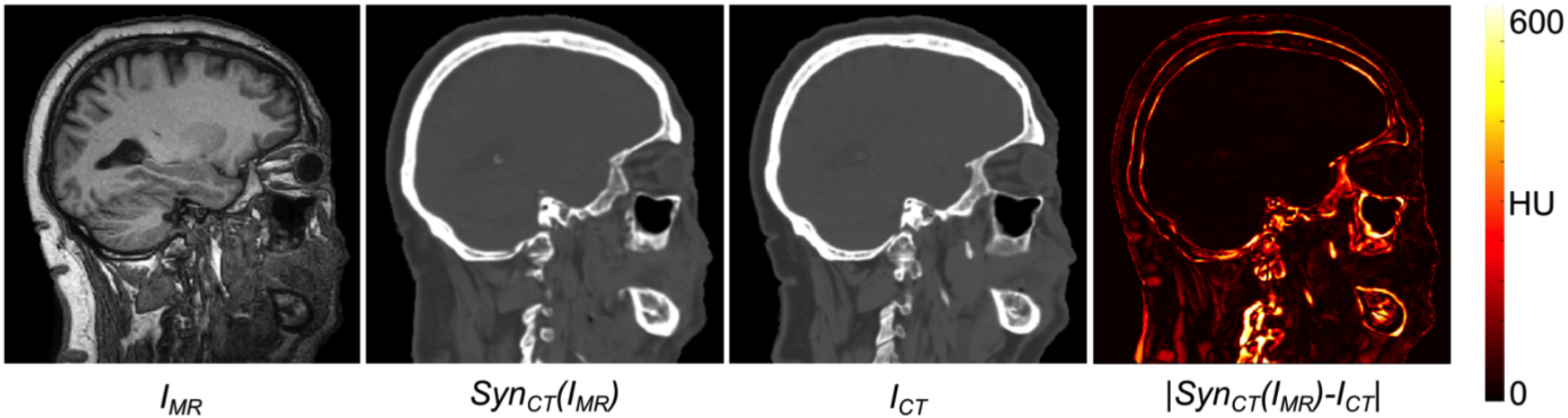


Fig. 4: *From left to right* Input MR image, synthesized CT image, reference real CT image, and absolute error between real and synthesized CT image.

Wolterink et al., Deep MR to CT synthesis using unpaired data. International Workshop on Simulation and Synthesis in Medical Imaging, 2017

APPLICATIONS - MR TO CT

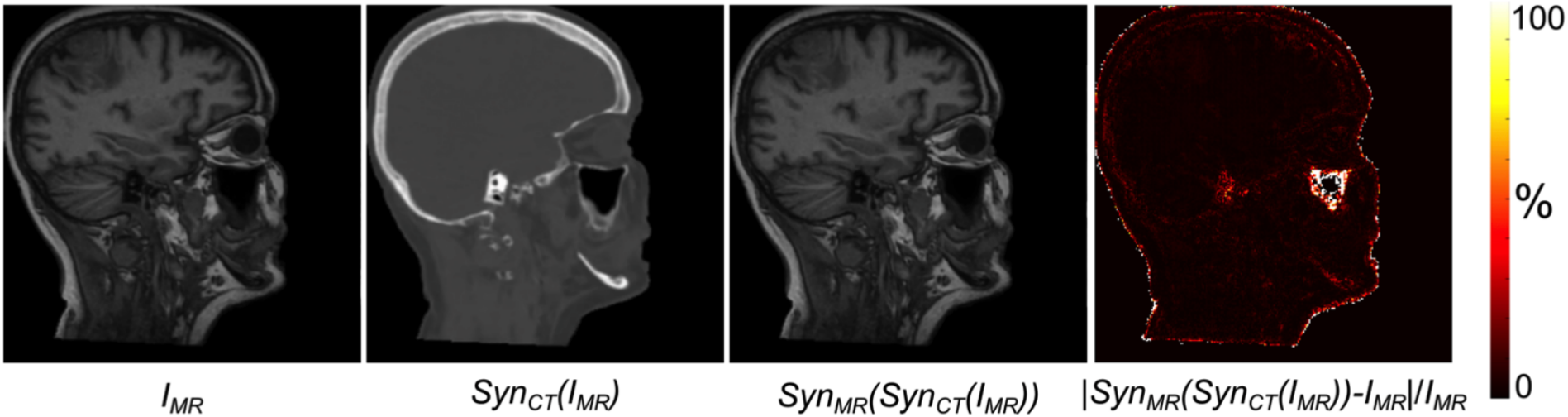


Fig. 6: *From left to right* Input MR image, synthesized CT image, reconstructed MR image, and relative error between the input and reconstructed MR image.

Wolterink et al., Deep MR to CT synthesis using unpaired data. International Workshop on Simulation and Synthesis in Medical Imaging, 2017

APPLICATIONS - MRI TO PET

ADNI: Most subjects have MRI and PET data,
for some subjects the PET data is missing

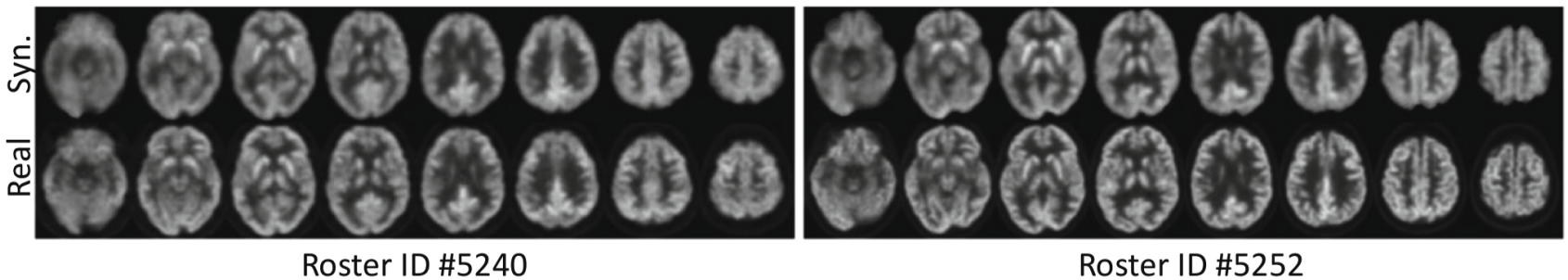


Fig. 4. Illustration of synthetic (Syn.) PET generated by our method for two typical subjects (Roster IDs: 5240, 5252), as well as their corresponding real images.

Pan et al., Synthesizing missing PET from MRI with cycle-consistent generative adversarial networks for Alzheimer's disease diagnosis. In International Conference on Medical Image Computing and Computer-Assisted Intervention, 2018

APPLICATIONS - MRI TO PET

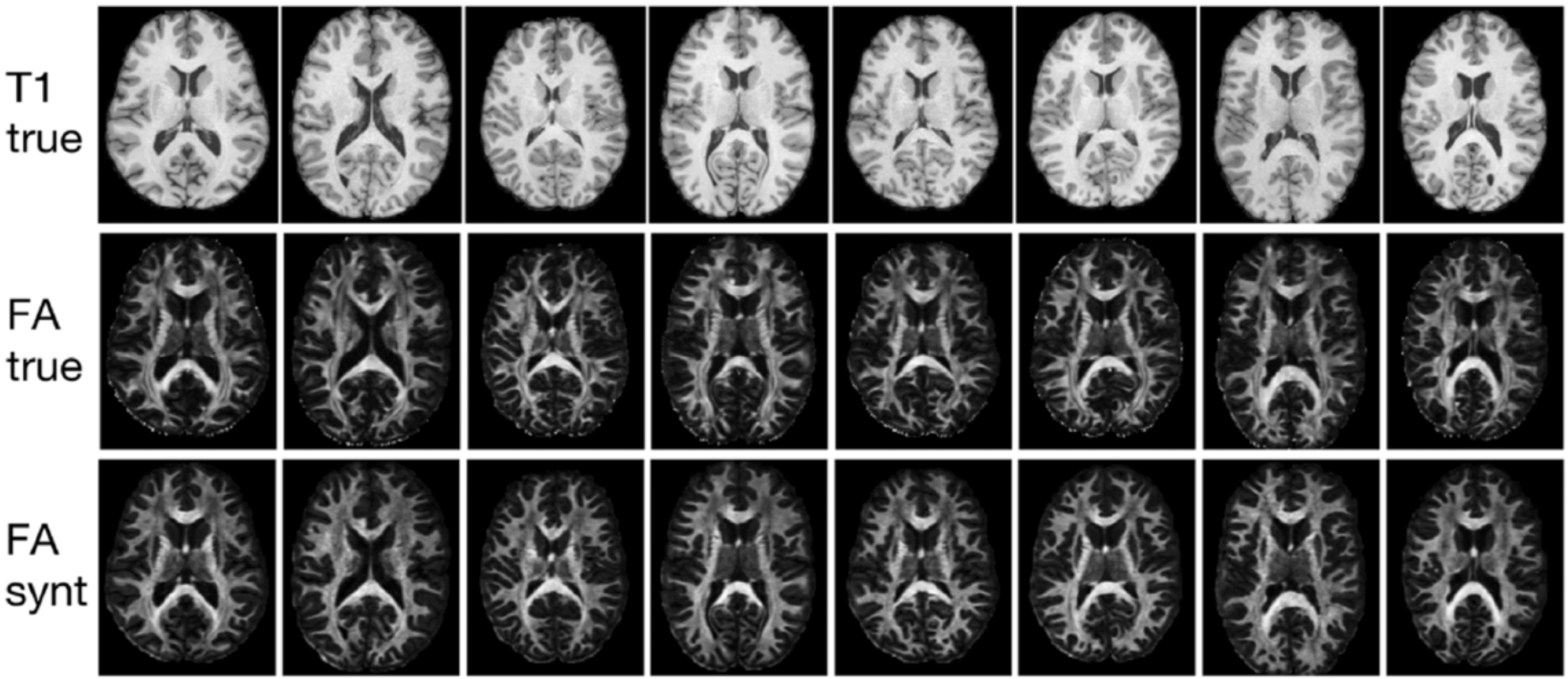
Synthesized PET data, for subjects with no PET data, boosts classification performance

Table 1. Performance of seven different methods in both tasks of AD classification (AD vs. HC classification) and MCI conversion prediction (pMCI vs. sMCI classification).

Method	AD vs. HC classification						pMCI vs. sMCI classification					
	ACC (%)	SEN (%)	SPE (%)	F1S (%)	MCC (%)	AUC (%)	ACC (%)	SEN (%)	SPE (%)	F1S (%)	MCC (%)	AUC (%)
ROI	79.17	78.62	79.60	76.92	58.00	86.73	66.06	47.37	69.04	27.69	11.98	63.77
VGD	80.50	77.35	83.00	77.84	60.44	87.62	64.26	36.84	68.62	22.05	04.02	59.29
LLEP	82.22	77.36	86.07	79.35	63.83	88.11	68.59	39.47	73.22	25.64	09.67	63.63
LDSIL	90.56	87.42	93.03	89.10	80.82	95.74	70.04	36.84	75.31	25.23	09.49	64.48
LDMIL	91.09	88.05	93.50	89.74	81.91	95.86	76.90	42.11	82.43	33.33	20.74	**77.64**
LM^3IL-C	87.50	84.85	89.36	84.85	74.21	93.08	76.92	44.44	81.16	30.77	19.81	68.59
LM^3IL	**92.50**	**89.94**	**94.53**	**91.37**	**84.78**	**95.89**	**79.06**	**55.26**	**82.85**	**40.86**	**30.13**	75.84

Pan et al., Synthesizing missing PET from MRI with cycle-consistent generative adversarial networks for Alzheimer's disease diagnosis. In International Conference on Medical Image Computing and Computer-Assisted Intervention, 2018

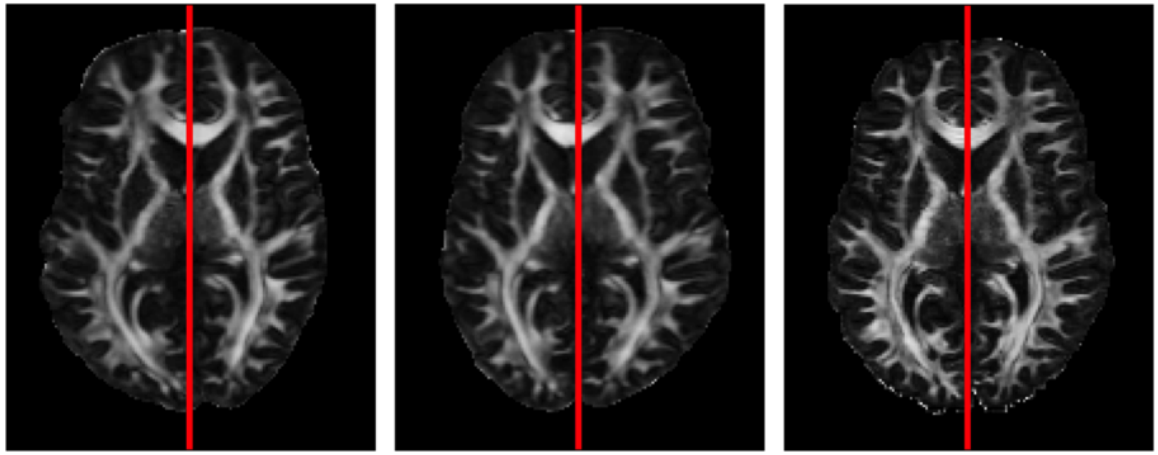
APPLICATIONS - T1 TO FA



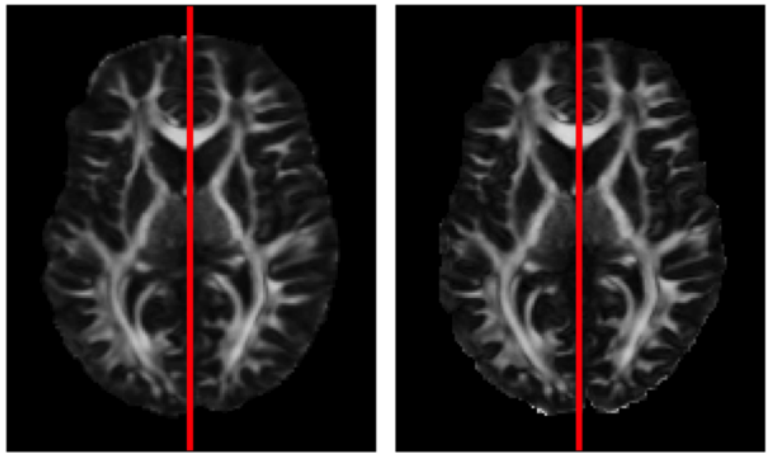
HCP data, 1000 subjects for training, 65 subjects for testing

Gu et al., Generating diffusion MRI scalar maps from T1 weighted images using generative adversarial networks, SCIA, 2019

APPLICATIONS - DISTORTION CORRECTION



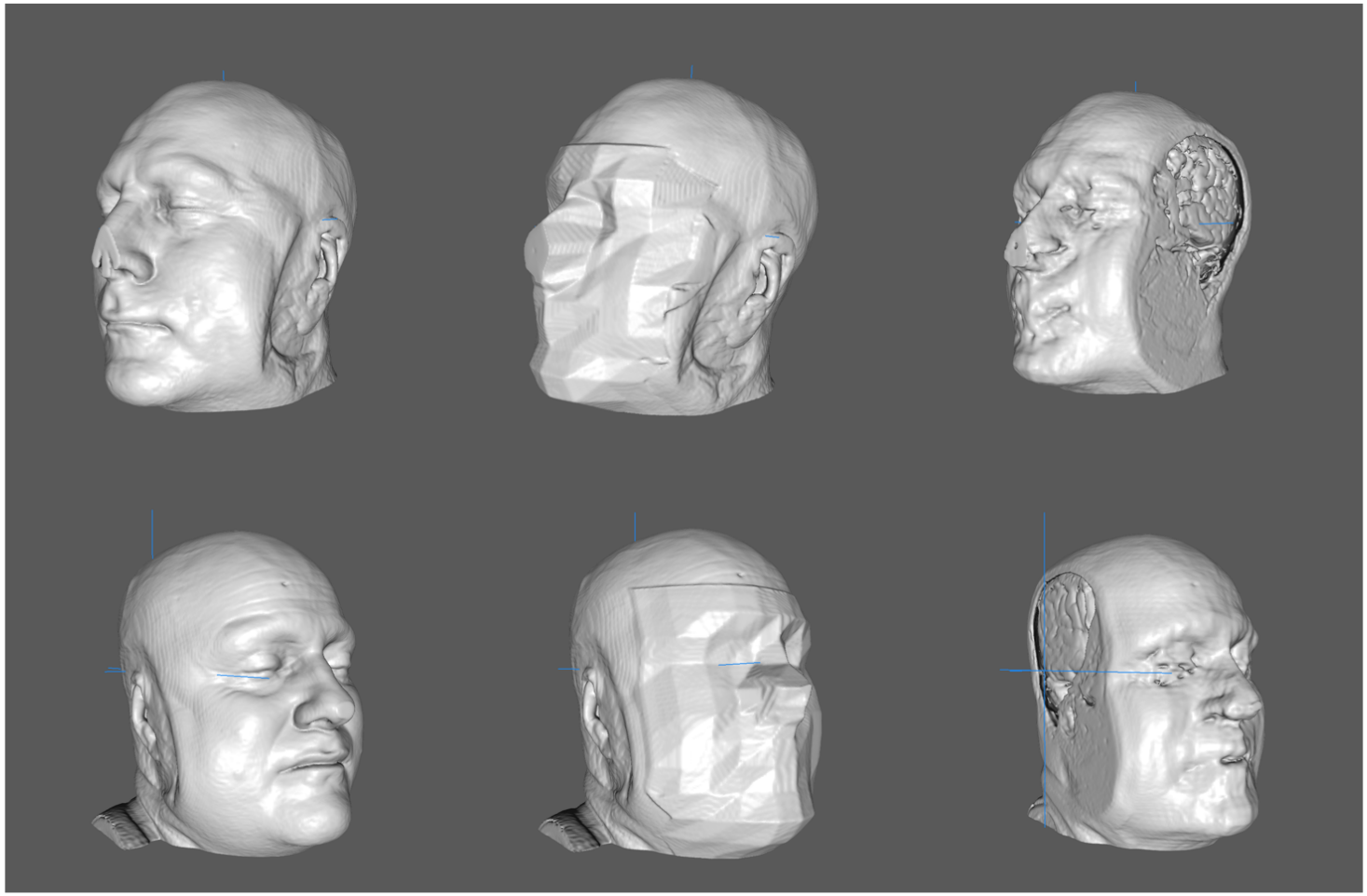
(a) FA LR. (b) FA RL. (c) FA synthetic.



(d) FA LR registered. (e) FA topup.

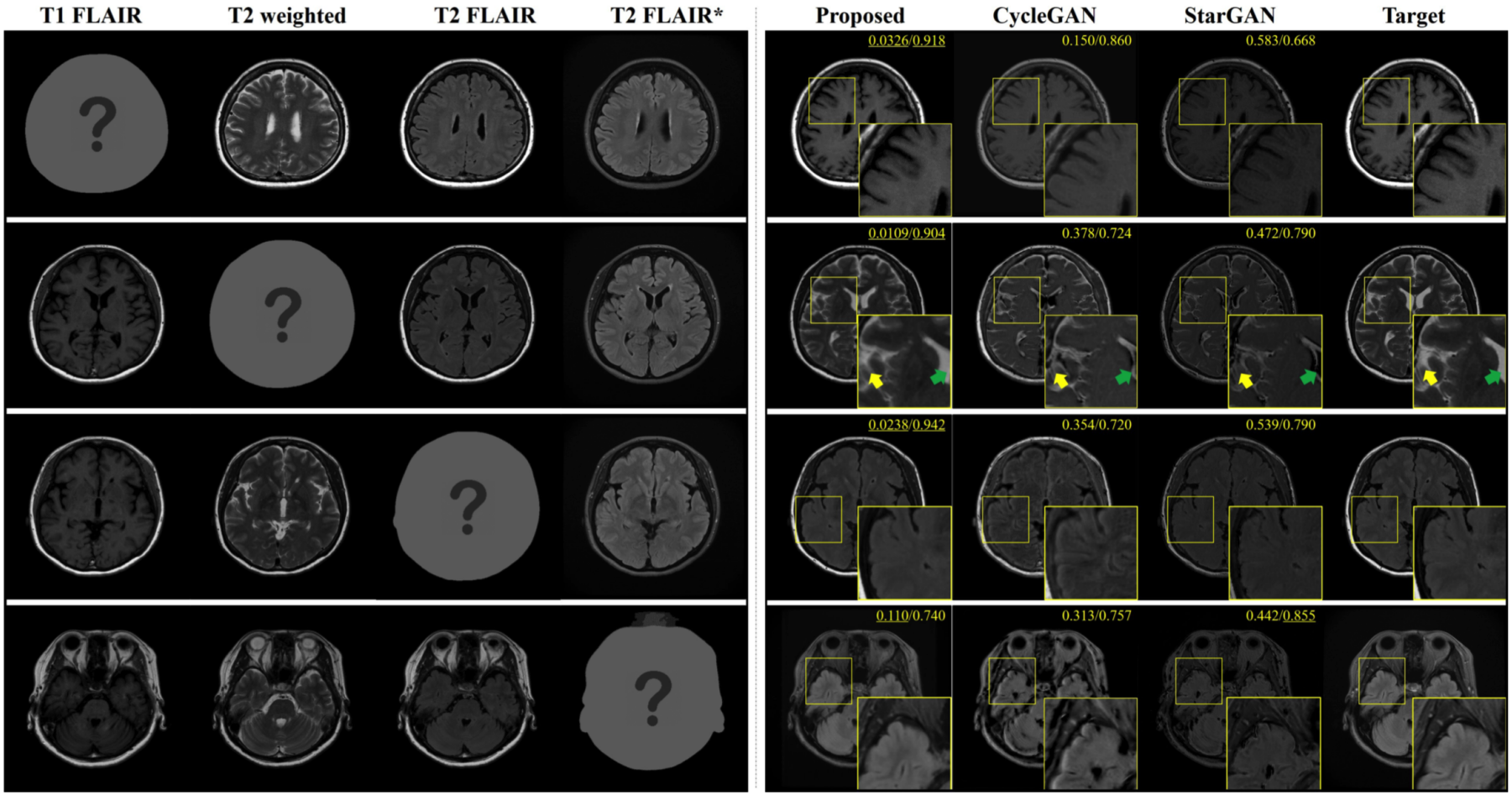
Gu et al., Generating diffusion MRI scalar maps from T1 weighted images using generative adversarial networks, SCIA, 2019

ARE GANS TOO POWERFUL?



Abramian & Eklund, Refacing: reconstructing anonymized facial features using GANs, ISBI, 2019

More advanced - collagan



Lee, D., Kim, J., Moon, W. J., & Ye, J. C. (2019). CollaGAN: Collaborative GAN for Missing Image Data Imputation. arXiv:1901.09764

More advanced - collagan

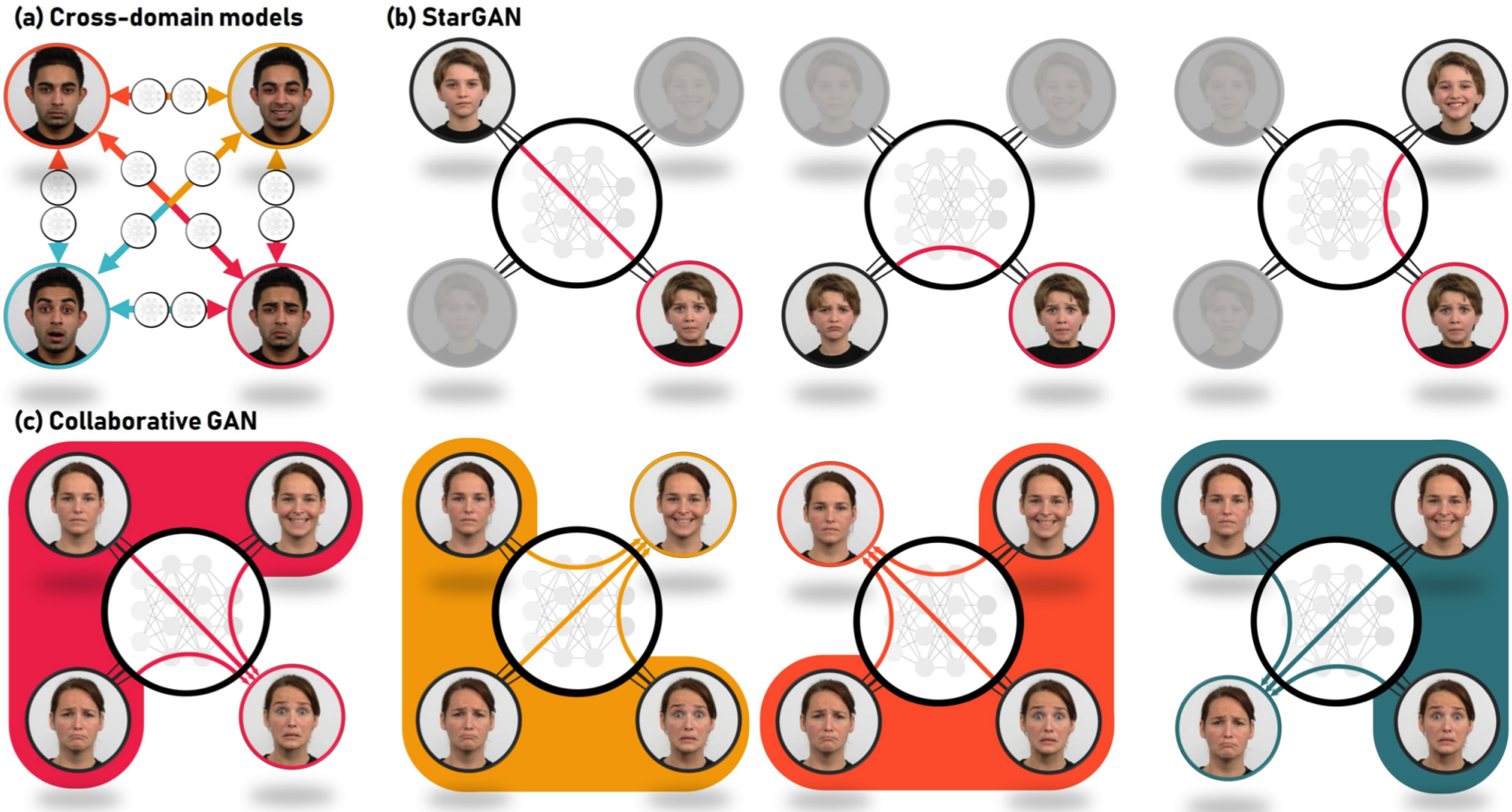


Figure 1: Image translation tasks using (a) cross-domain models, (b) StarGAN, and (c) the proposed collaborative GAN (CollaGAN). Cross-domain model needs large number of generators to handle multi-class data. StarGAN and CollaGAN use a single generator with one input and multiple inputs, respectively, to synthesize the target domain image.

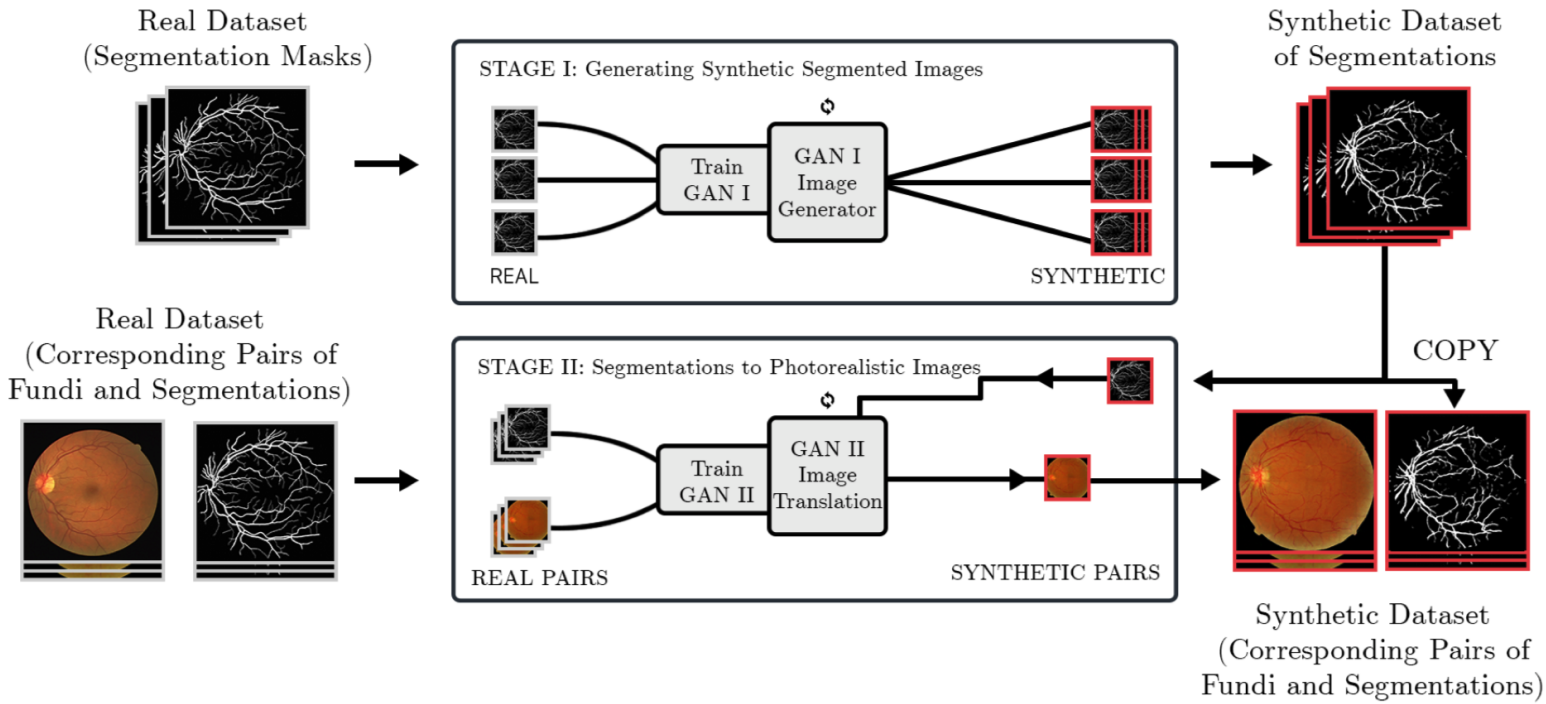
More advanced - combining GANs

- Difficult to synthesize realistic data from scratch (image-to-image GANs are normally easier / more stable to train compared to noise-to-image GANs)
- Use noise-to-image GAN to synthesize object of interest (as a (small) binary image / volume)
- Use image-to-image GAN to transform binary image / volume into a realistic dataset

More advanced - combining GANs

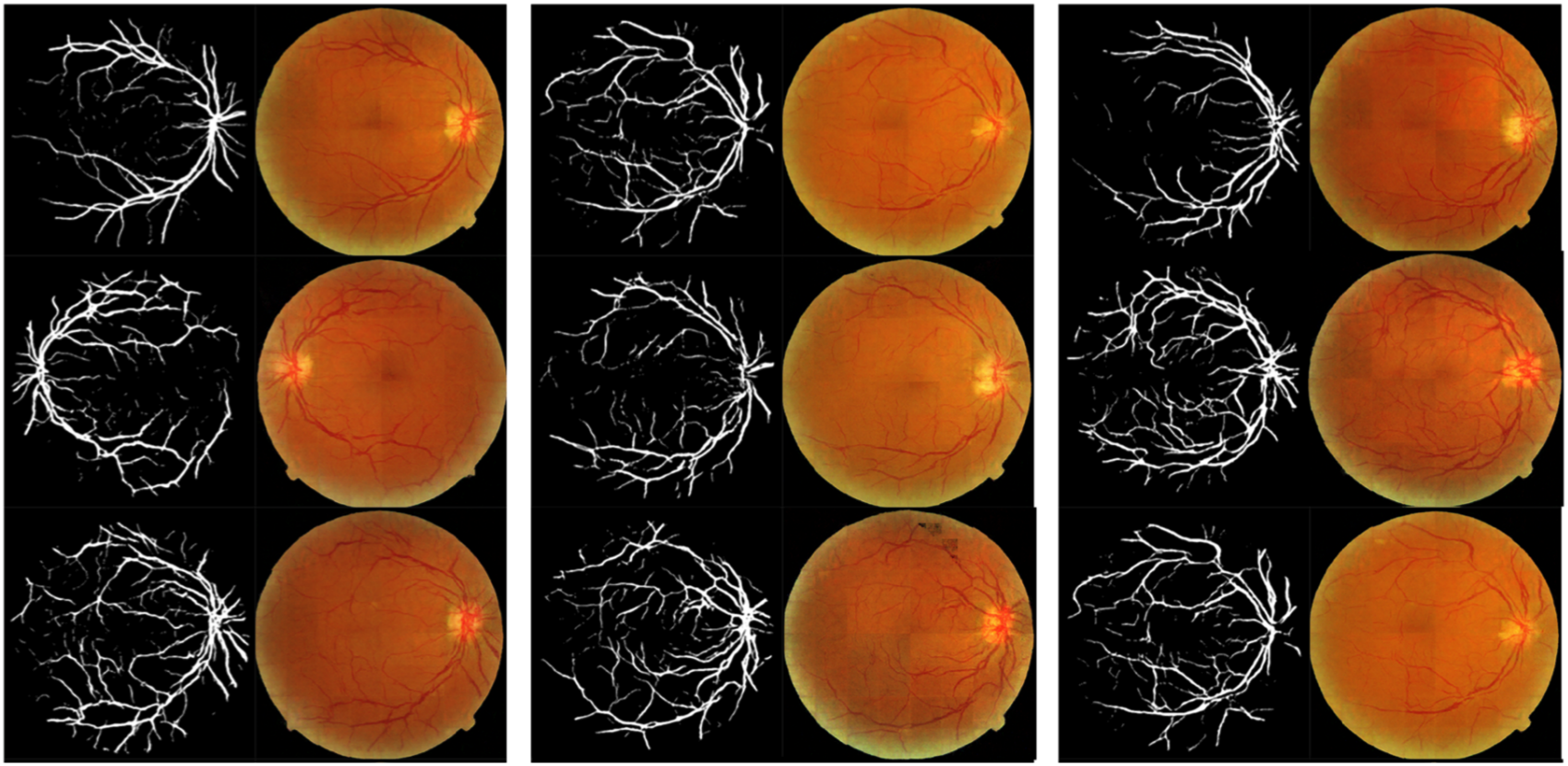
1. Stage-I GAN: Produce segmentation masks that represent the variable geometries of the dataset.
2. Stage-II GAN: Translate the masks produced in Stage-I to photorealistic images.

We illustrate the process with retinal fundi images.



Guibas, J. T., Virdi, T. S., & Li, P. S. Synthetic medical images from dual generative adversarial networks. arXiv:1709.01872.

More advanced - combining GANs



Guibas, J. T., Virdi, T. S., & Li, P. S. Synthetic medical images from dual generative adversarial networks. arXiv:1709.01872.

CHALLENGES - 2D TO 3D

- Almost all GAN papers use 2D GANs for image to image translation
- In neuroimaging we have 3D and 4D data
- Can apply a 2D GAN slice by slice,
  but will not give a perfect 3D reconstruction
- Code for 2D GANs can easily be changed to 3D,
  but 3D GANs are much more memory demanding
- Basically two solutions exist for true 3D GANs
  - Work on subvolumes (e.g. 32 x 32 x 32 or 64 x 64 x 64)
  - Use GPU hardware with more memory (expensive),
    but once the GAN is trained it can be applied using a CPU

2D to 3D - Keras

- Conv2D -> Conv3D
- Conv2DTranspose -> Conv3DTranspose
- Upsampling2D -> Upsampling3D

- Why is 3D difficult?

2D to 3D - memory consumption

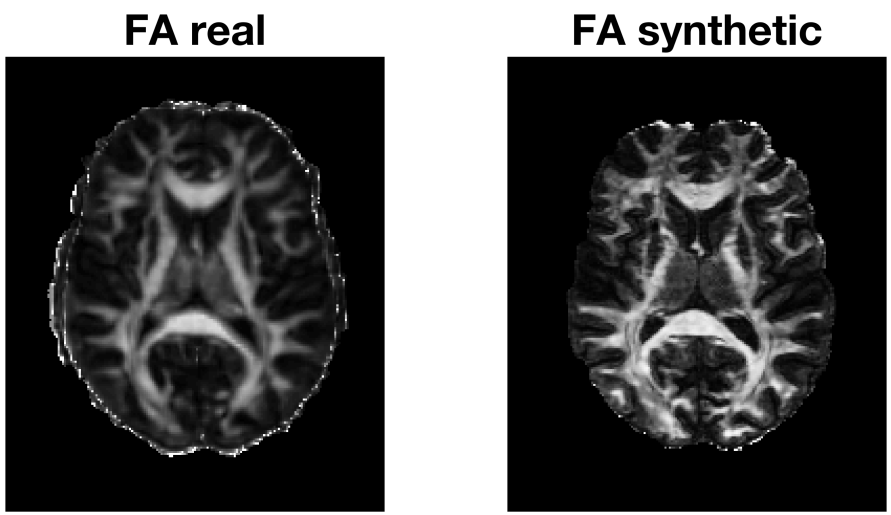
- 256 filter responses for 256 x 256 image = 64 MB as floats
- 256 filter responses for 256 x 256 x 256 volume = 16 GB as floats
- Nvidia Titan X Pascal 12 GB,
  no gaming graphics card with more than 12 GB ?
- Nvidia DGX station, 128 GB graphics memory
  (expensive)

CHALLENGES - 2D TO 3D

- Noise-to-image GANs have 1 generator and 1 discriminator
- CycleGAN has 2 generators and 2 discriminators
- UNIT has 4 generators and 2 discriminators
- Image-to-image GANs are more memory demanding than noise-to-image GANs, therefore more difficult to run in 3D

CHALLENGES - MAKING IT WORK FOR ALL SCANNERS

- A network trained on data from scanner A will not perform as well on data from scanner B (general problem in deep learning)
- Train a GAN to convert from T1 to FA on data from MR scanner A, will probably fail for data from MR scanner B
- Example, trained on 1000 subjects in HCP, tested on "Unilateral Glaucoma" dataset in OpenNeuro

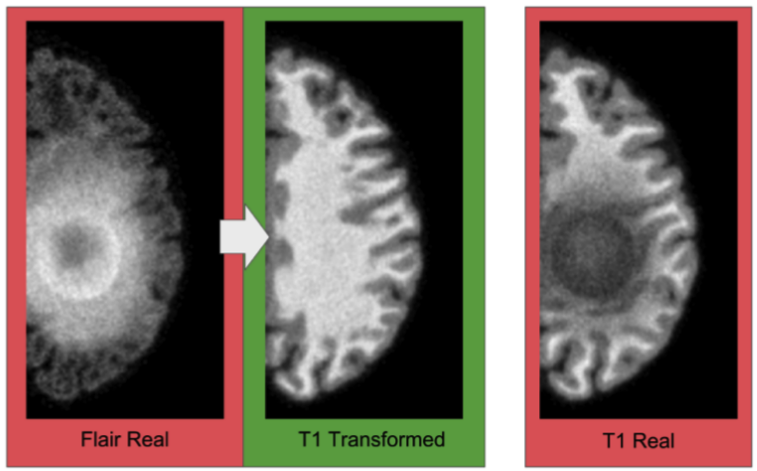


CHALLENGES - MAKING IT WORK FOR ALL SCANNERS

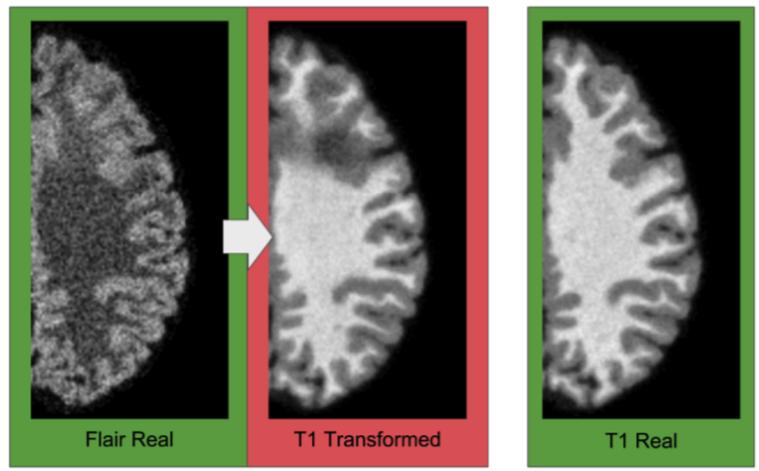
- This problem is difficult to solve, possible solutions are
  - Do initial training with many different datasets (e.g. from openneuro.org), fine tune the training using data from a specific scanner
  - Train a GAN to translate from scanner A to scanner B (requires many images from scanner A and B)
  - Quantitative MRI, instead of collecting T1-weighted images, estimate T1 relaxation rate, T2 relaxation rate, proton density in each voxel, should give the same results on all MR scanners

CHALLENGES - TRAINING ON CONTROLS

- If a GAN is trained on healthy controls, can it be used for diseased subjects?



(a) A translation removing tumors



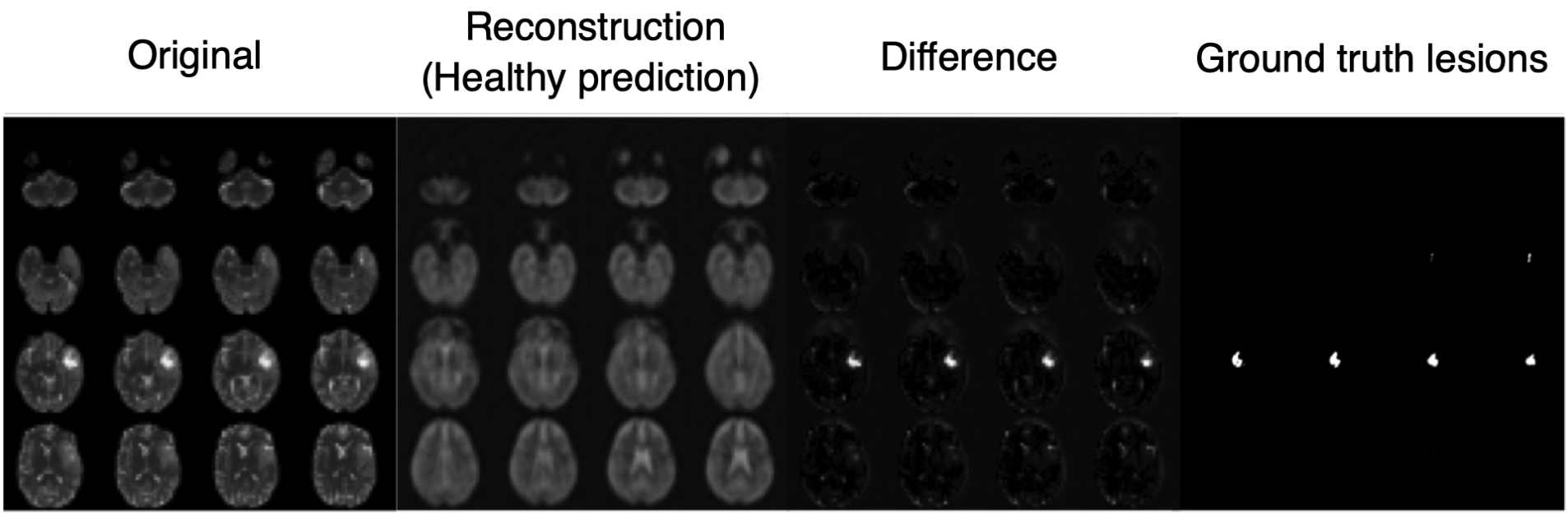
(b) A translation adding tumors

- Cohen, J. P., Luck, M., & Honari, S., Distribution matching losses can hallucinate features in medical image translation. In International Conference on Medical Image Computing and Computer-Assisted Intervention, 2018

CHALLENGES - TRAINING ON CONTROLS

- This can be seen as a problem or an opportunity
- Diseased True T2 -> Healthy Fake T1
- Compare with diseased true T1, difference is abnormality?
- Registration errors...
- Diseased True T1 -> Healthy Fake T2 -> Healthy Fake T1 (registration avoided)

Abnormality detection



Chen, X., & Konukoglu, E., Unsupervised Detection of Lesions in Brain MRI using constrained adversarial auto-encoders, arXiv:1806.04972.

Questions

- Questions?
- David Abramian has prepared a jupyter notebook that you can play with (available on course homepage)