10707
Deep Learning: Spring 2021

Andrej Risteski

Machine Learning Department

Lecture 16:
Applications of GANs, normalizing flows

Conditional GANs (Mirza-Osindero '14)

What if we want to have a notion of "class" and have class-labeled data?

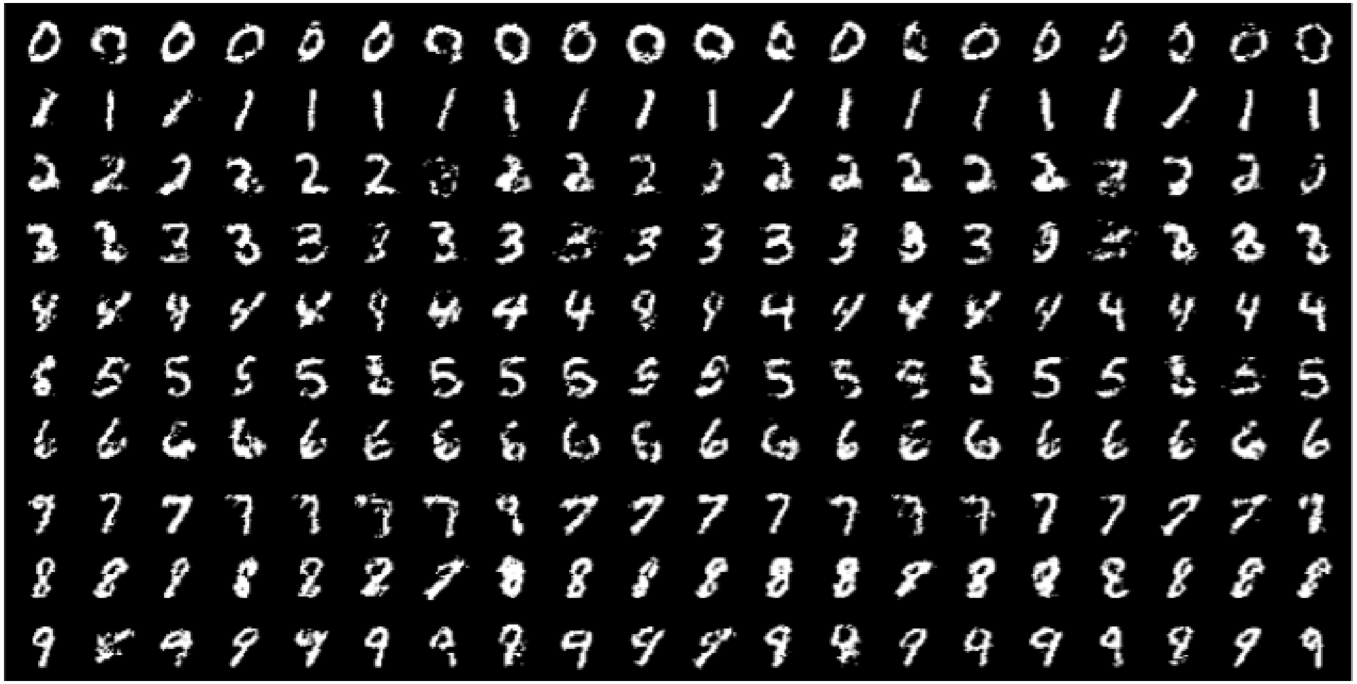
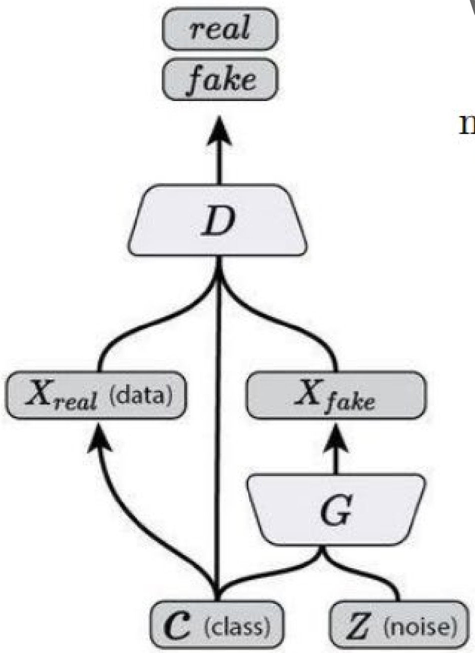


Figure 2: Generated MNIST digits, each row conditioned on one label

Figure from Mirza-Osindero '14

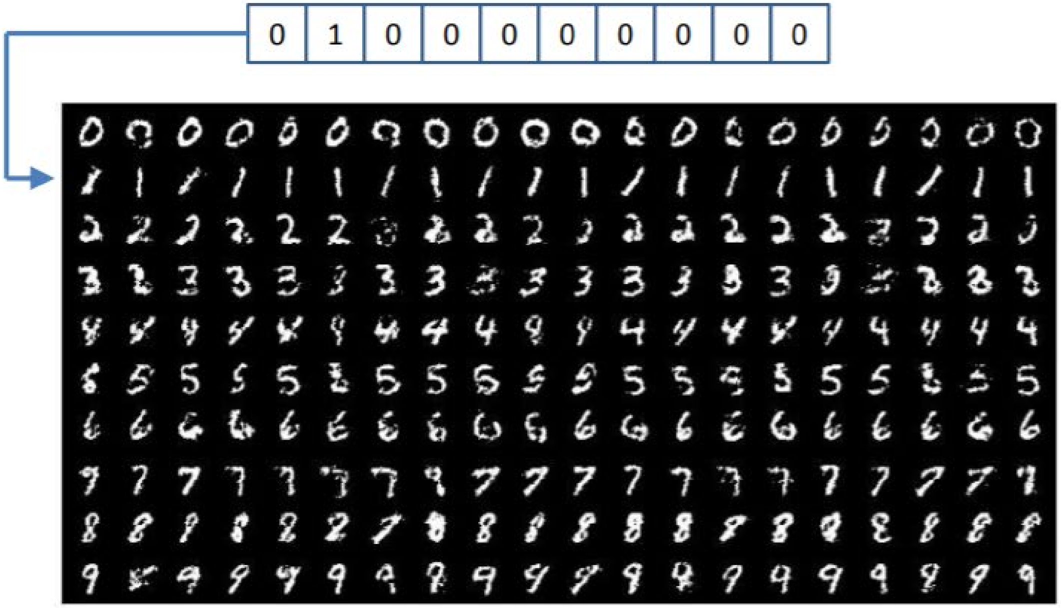
Conditional GANs (Mirza-Osindero ’14)



Discriminator and generator receive an image as well as a class label. The loss is then:

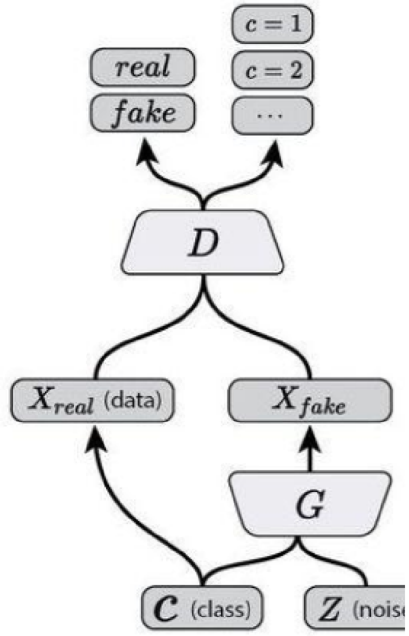
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_z(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]$$

0	1	0	0	0	0	0	0	0	0



Auxillary classifier GANs (Odena et al '16)

Idea: give discriminator **only** image, and ask it to both predict label, and tell real/fake.



Let us define the "distinguishing" loss and the class prediction loss:

$$L_S = E[\log P(S = real \mid X_{real})] + E[\log P(S = fake \mid X_{fake})]$$
$$L_C = E[\log P(C = c \mid X_{real})] + E[\log P(C = c \mid X_{fake})]$$

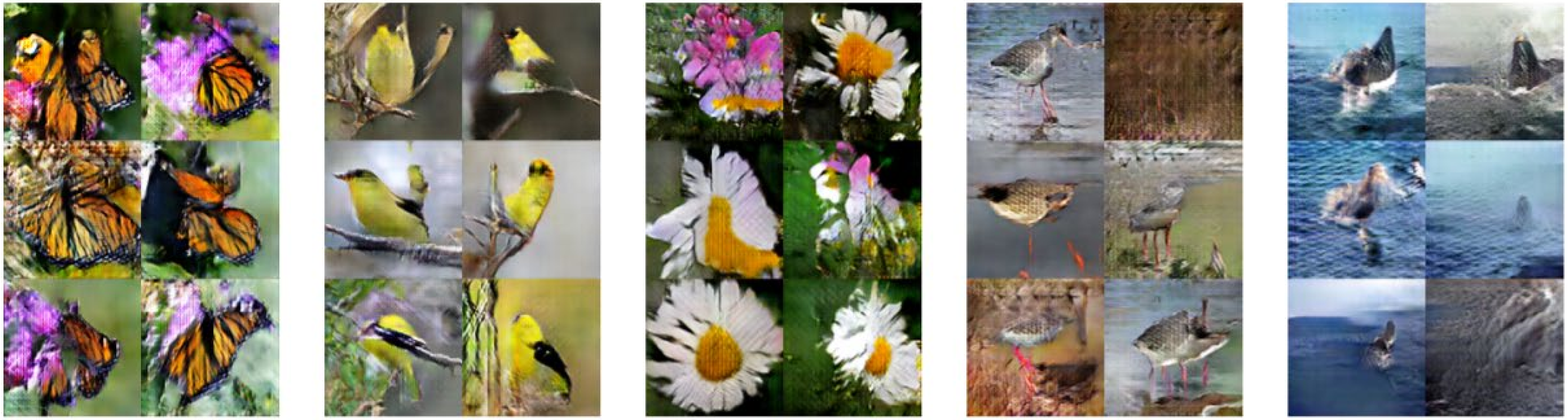
Discriminator is trained to maximize $L_s + L_c$

Generator is trained to maximize $L_c - L_s$

(i.e. discriminator tries to both distinguish and predict label; generator tries to fool discriminator and generate "classifiable" images)

Main benefit: class-independent latent embedding; stabler training

Auxillary classifier GANs (Odena et al ‘16)



monarch butterfly goldfinch daisy redshank grey whale

Figure 1. 128 × 128 resolution samples from 5 classes taken from an AC-GAN trained on the ImageNet dataset. Note that the classes shown have been selected to highlight the success of the model and are not representative. Samples from all ImageNet classes are linked later in the text.

Figure from Odena et al ‘16

Superresolution from a single image (Ledig et al ‘17)

Task: estimate a high-resolution version of an image, given a low-resolution version of it.

Training data: high-resolution images, along with manually created low-res versions (e.g. by Gaussian filtering).

Superresolution from a single image (Ledig et al ‘17)

Outperforms approaches based on “pre-designed” features/losses.

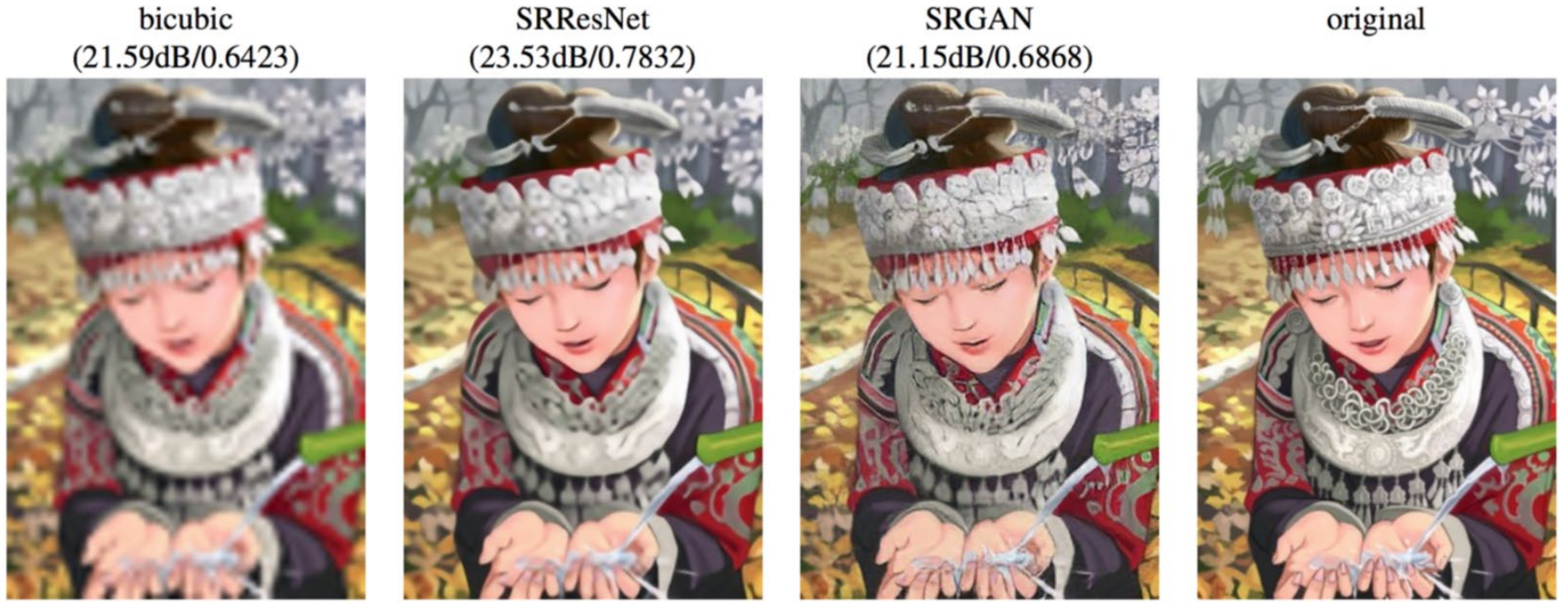


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

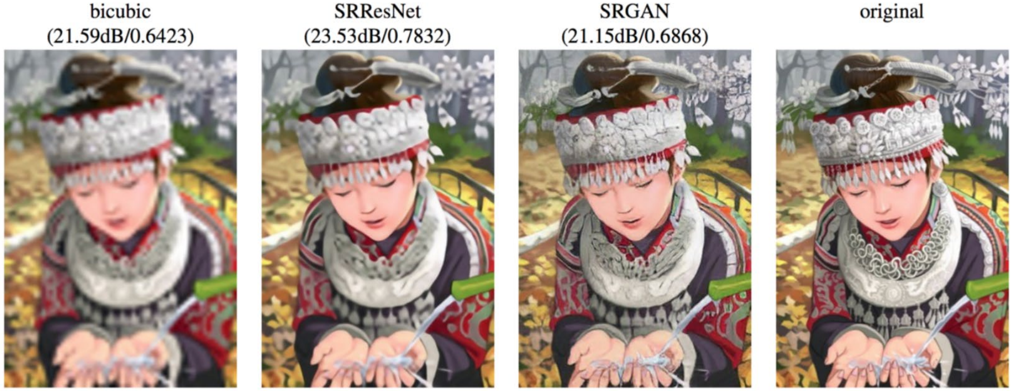
Superresolution from a single image (Ledig et al ‘17)

Combines l2 loss on VGG-19 features with adversarial loss:

$$l^{SR}_{VGG/i.j} = \frac{1}{W_{i,j}H_{i,j}} \sum_{x=1}^{W_{i,j}} \sum_{y=1}^{H_{i,j}} (\phi_{i,j}(I^{HR})_{x,y} - \phi_{i,j}(G_{\theta_G}(I^{LR}))_{x,y})^2$$

$\phi_{i,j}$ The feature map for the j-th convolution (after activation) before the i-th maxpooling layer.

$$\min_{\theta_G} \max_{\theta_D} \mathbb{E}_{I^{HR} \sim p_{\text{train}}(I^{HR})}[\log D_{\theta_D}(I^{HR})] + \mathbb{E}_{I^{LR} \sim p_G(I^{LR})}[\log(1 - D_{\theta_D}(G_{\theta_G}(I^{LR})))]$$



bicubic (21.59dB/0.6423)

SRResNet (23.53dB/0.7832)

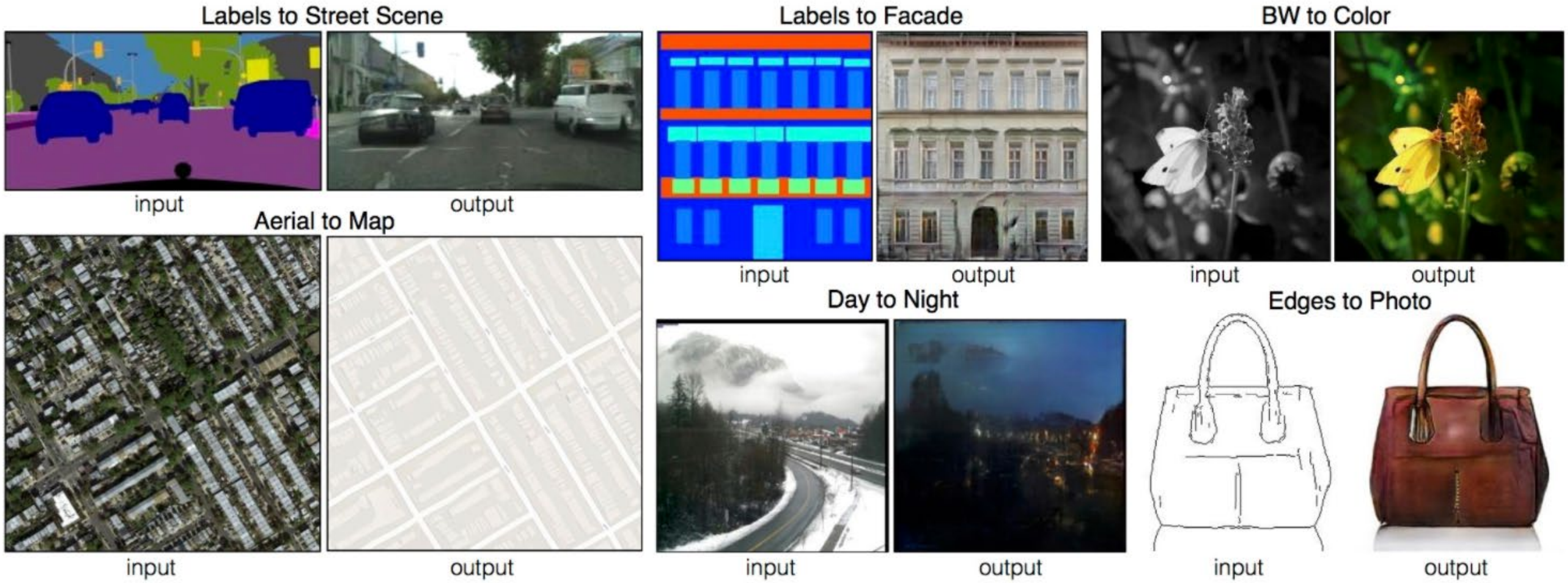
SRGAN (21.15dB/0.6868)

original

Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Image-to-image translation (Pix2Pix, Isola et al ‘17)

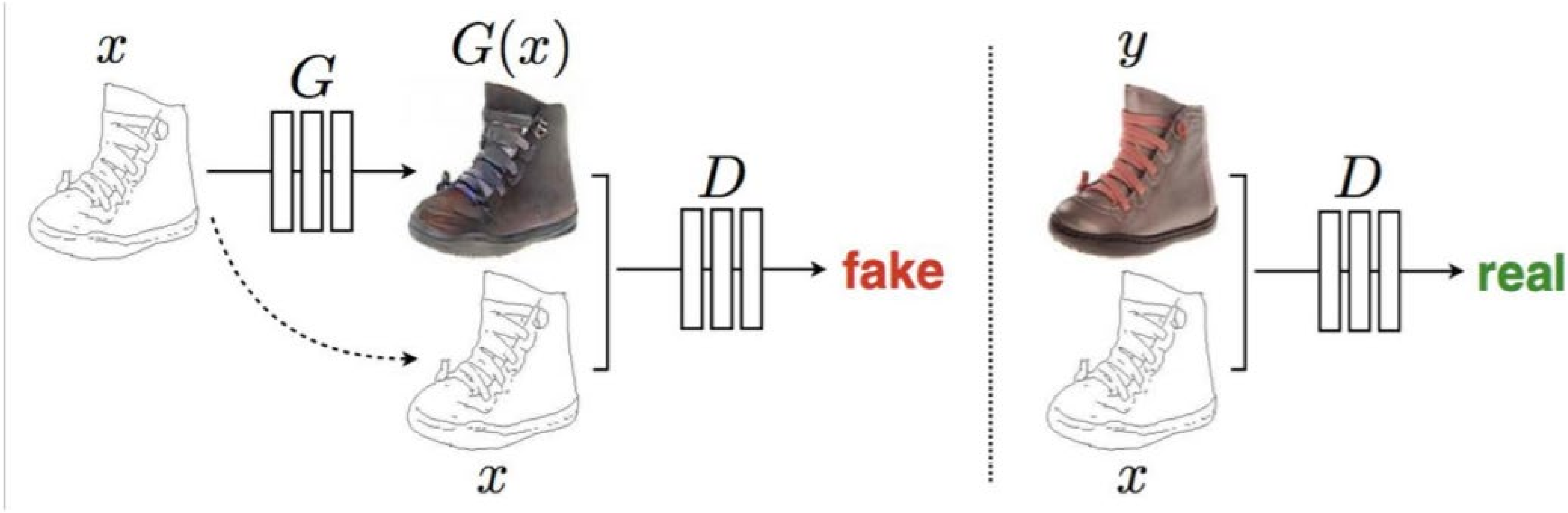
Goal: “translate” one “style” of image to another, given as training paired images of the styles.



Example results on several image-to-image translation problems. In each case we use the same architecture and objective, simply training on different data.

Image-to-image translation (Pix2Pix, Isola et al ‘17)

Loss: generator tries to “translate”; discriminator tries to distinguish whether it’s a “real” translation or “fake” translation.



$$\arg\min_G \max_D \ \mathbb{E}_{\mathbf{x},\mathbf{y}}[\ \log D(\mathbf{x}, G(\mathbf{x})) + \log(1 - D(\mathbf{x},\mathbf{y}))\ ]$$

Image-to-image translation (Pix2Pix, Isola et al '17)

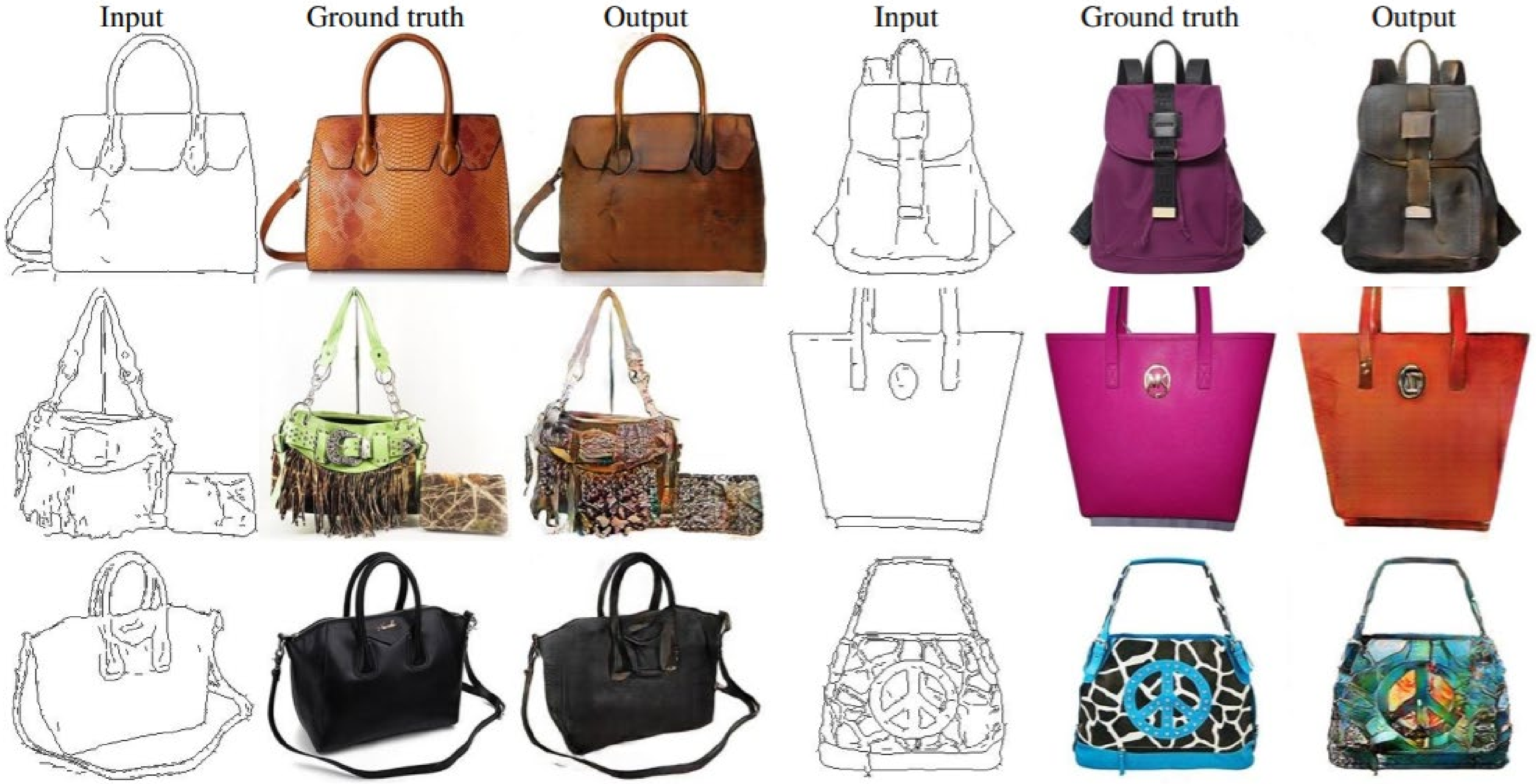
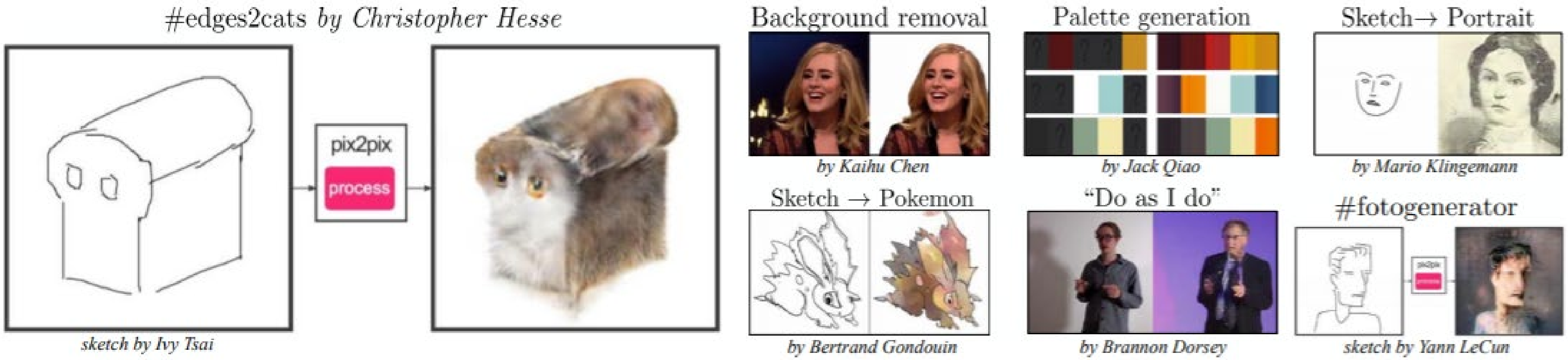


Figure 16: Example results of our method on automatically detected edges→handbags, compared to ground truth.

Image-to-image translation (Pix2Pix, Isola et al ‘17)



#edges2cats *by Christopher Hesse*

sketch by Ivy Tsai

Background removal

by Kaihu Chen

Palette generation

by Jack Qiao

Sketch→ Portrait

by Mario Klingemann

Sketch → Pokemon

by Bertrand Gondouin

“Do as I do”

by Brannon Dorsey

#fotogenerator

sketch by Yann LeCun

Figure 11: Example applications developed by online community based on our `pix2pix` codebase: *#edges2cats* [3] by Christopher Hesse, *Background removal* [6] by Kaihu Chen, *Palette generation* [5] by Jack Qiao, *Sketch → Portrait* [7] by Mario Klingemann, *Sketch→ Pokemon* [1] by Bertrand Gondouin, *“Do As I Do” pose transfer* [2] by Brannon Dorsey, and *#fotogenerator* by Bosman et al. [4].

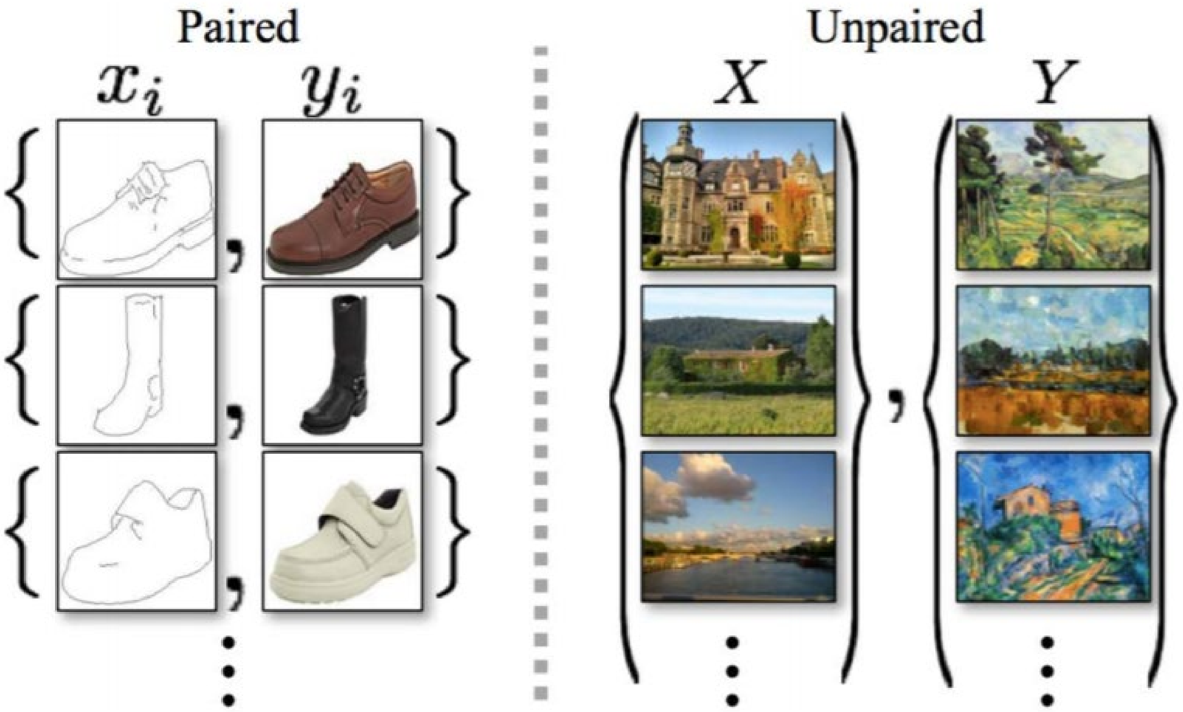
Unpaired image-to-image translation (CycleGAN, Zhu et al ‘17)

Major problem with prior approach: we need **paired** samples.

(How do you “translate” a photograph to the style of Van Gogh?

No “paired” up photos…)

Can we solve “unpaired” version of problem?

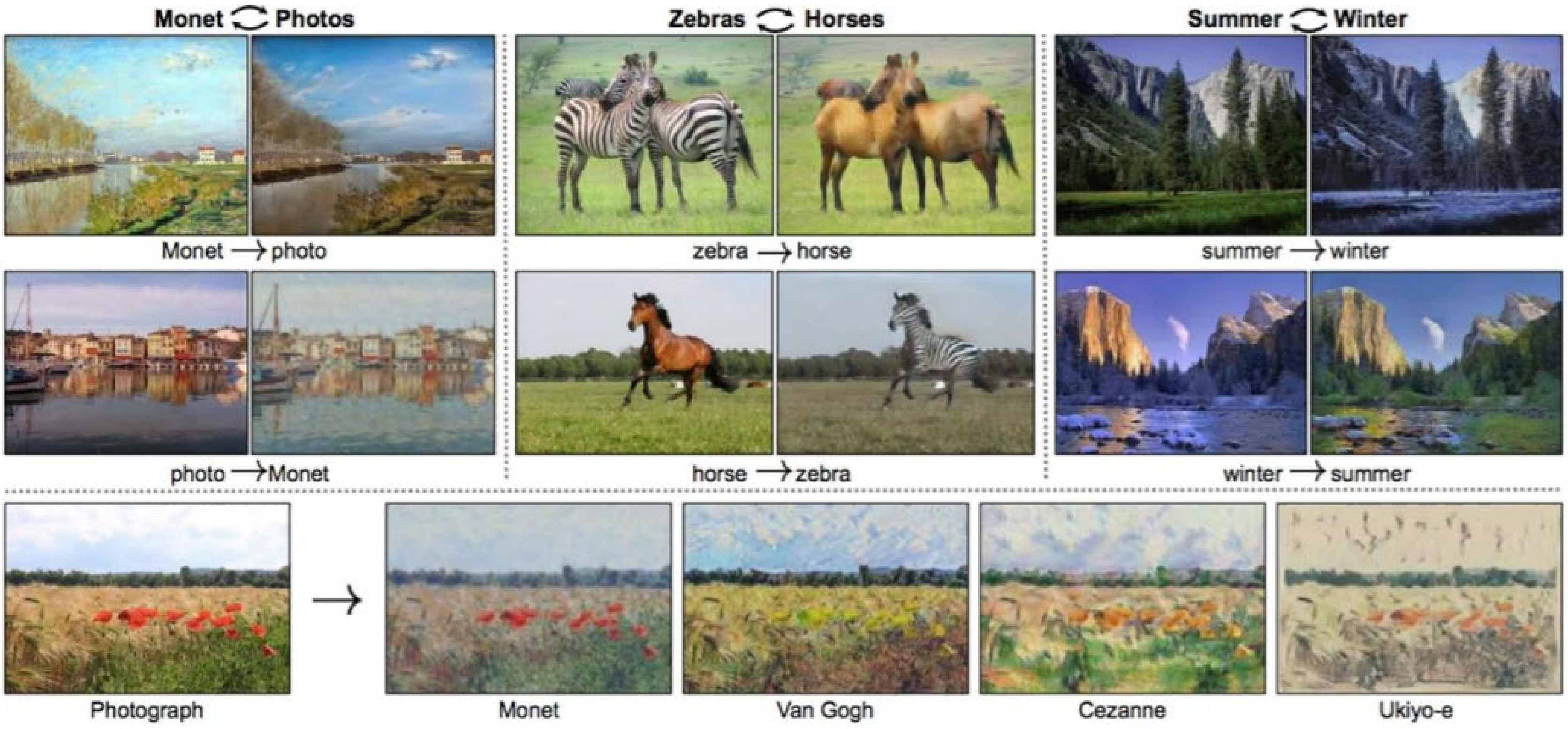


Unpaired image-to-image translation (CycleGAN, Zhu et al '17)

Major problem with prior approach: we need **paired** samples.

(How do you "translate" a photograph to the style of Van Gogh?

No "paired" up photos...)



Unpaired image-to-image translation (CycleGAN, Zhu et al '17)

Main idea:

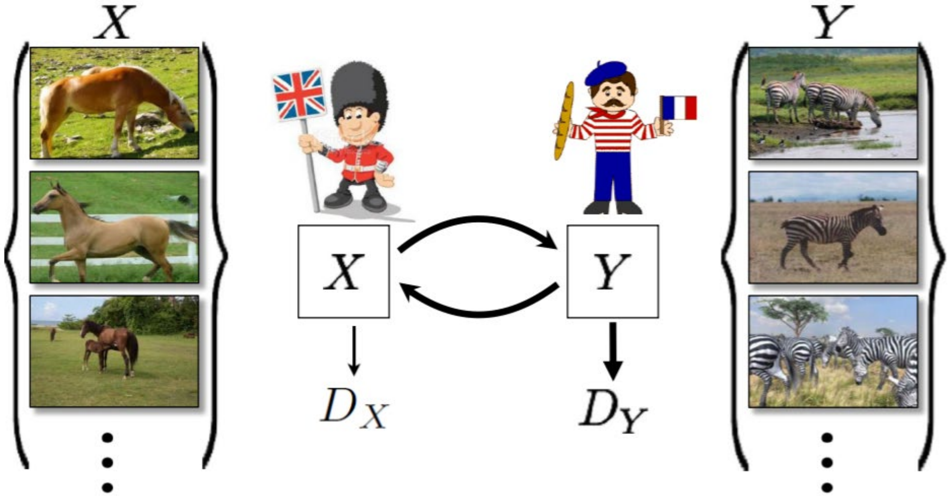
Train *two* generators F, G, s.t.

F translates from domain X to domain Y,

G translates from domain Y to domain X.

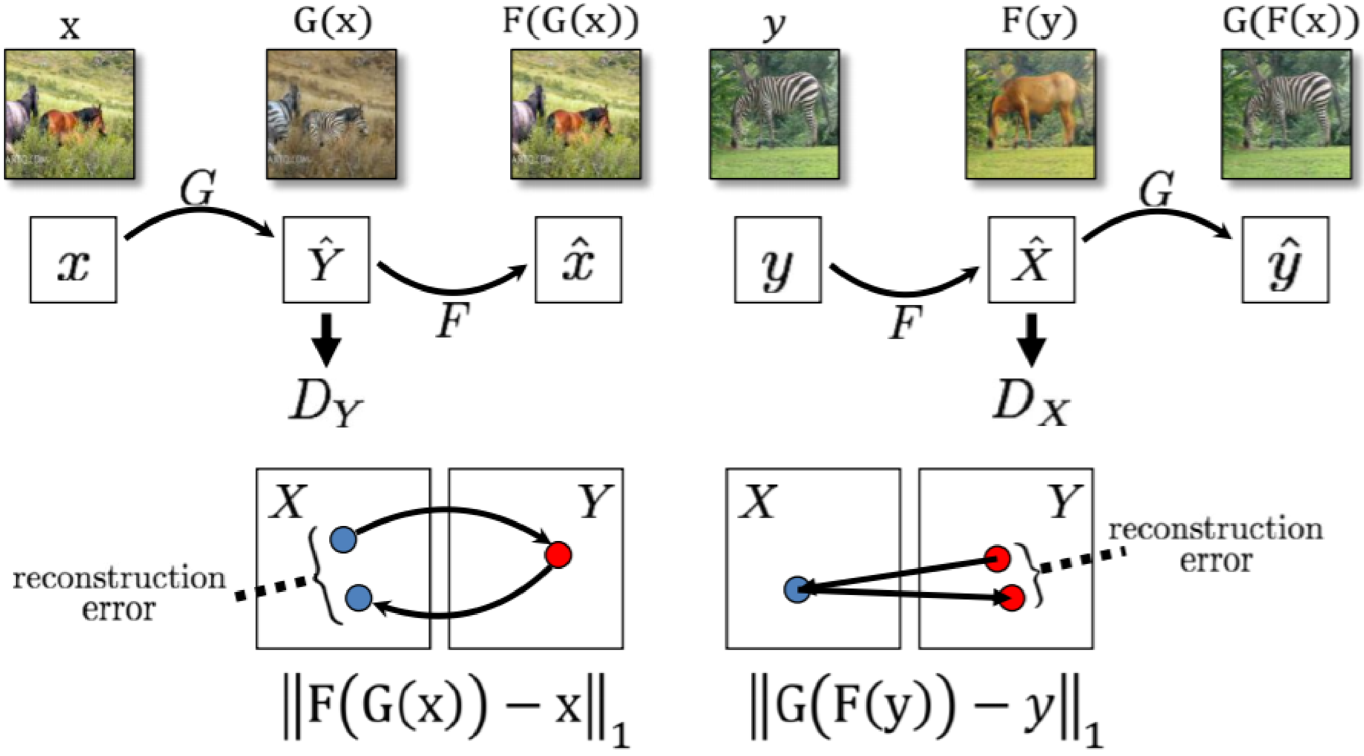
Discriminators D_X, D_Y trying to recognize domains X, Y.

Requirement: $F\left(G(x)\right) \approx x, G\left(F(X)\right) \approx x$



Unpaired image-to-image translation (CycleGAN, Zhu et al '17)

The CycleGAN loss:



Unpaired image-to-image translation (CycleGAN, Zhu et al ‘17)

The CycleGAN loss:

$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) = {} & \mathbb{E}_{y \sim p_{\text{data}}(y)}[\log D_Y(y)] \\ & + \mathbb{E}_{x \sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) = {} & \mathbb{E}_{x \sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ & + \mathbb{E}_{y \sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$

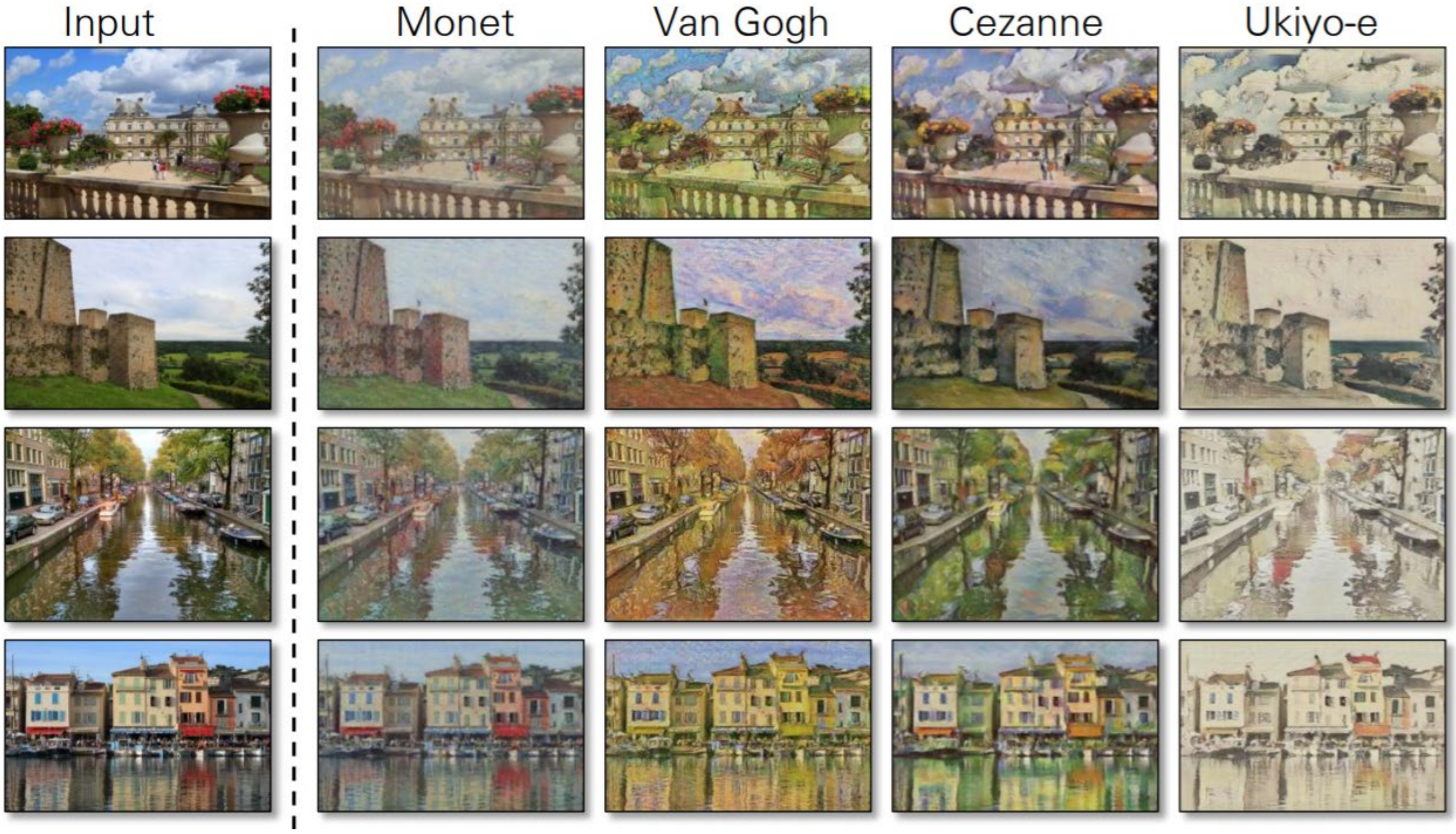
Putting them together, we get:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) = {} & \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ & + \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ & + \lambda \mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

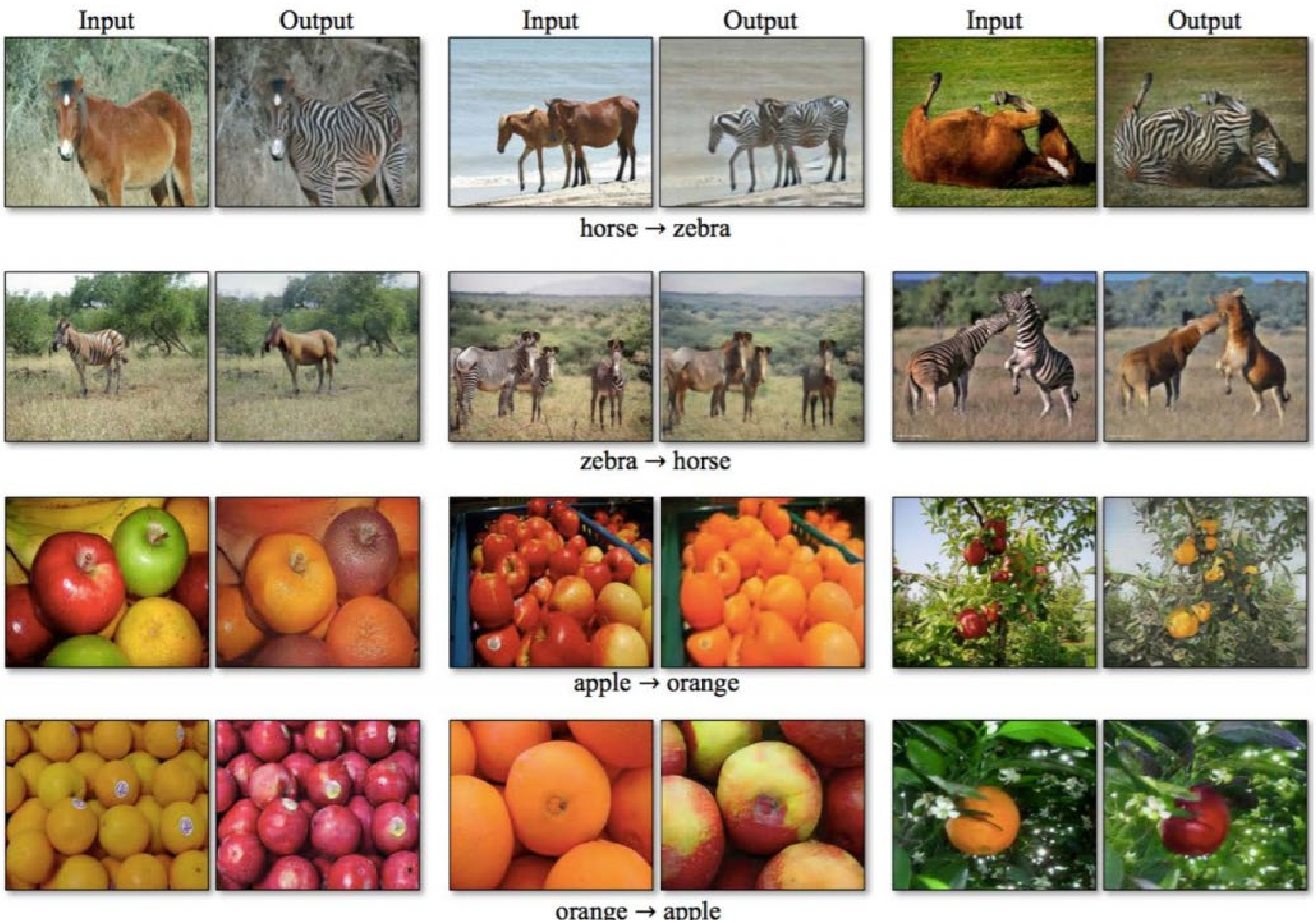
We are trying to optimize:

$$G^*, F^* = \arg \min_{G, F} \max_{D_x, D_Y} \mathcal{L}(G, F, D_X, D_Y).$$

Unpaired image-to-image translation (CycleGAN, Zhu et al '17)

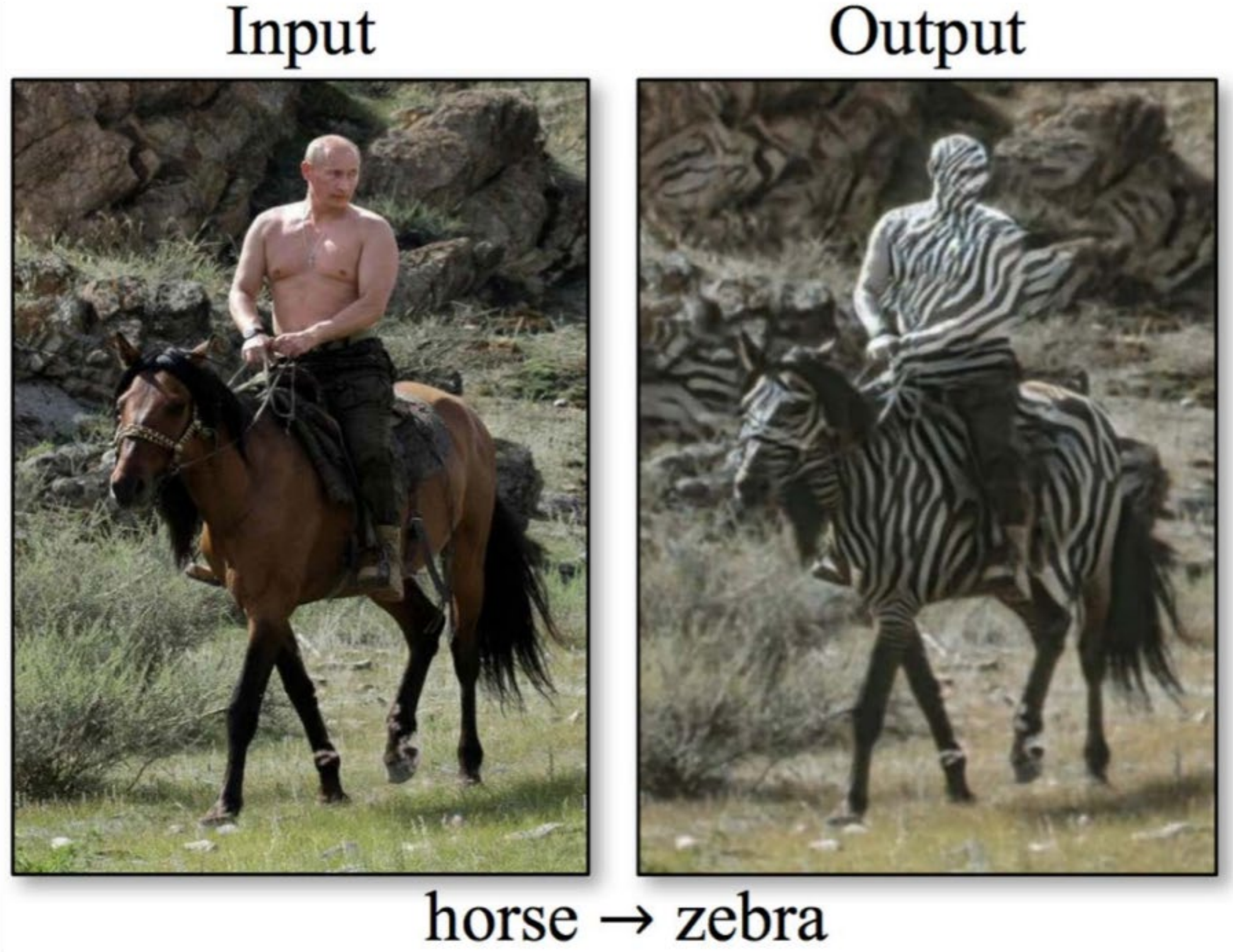


Unpaired image-to-image translation (CycleGAN, Zhu et al '17)



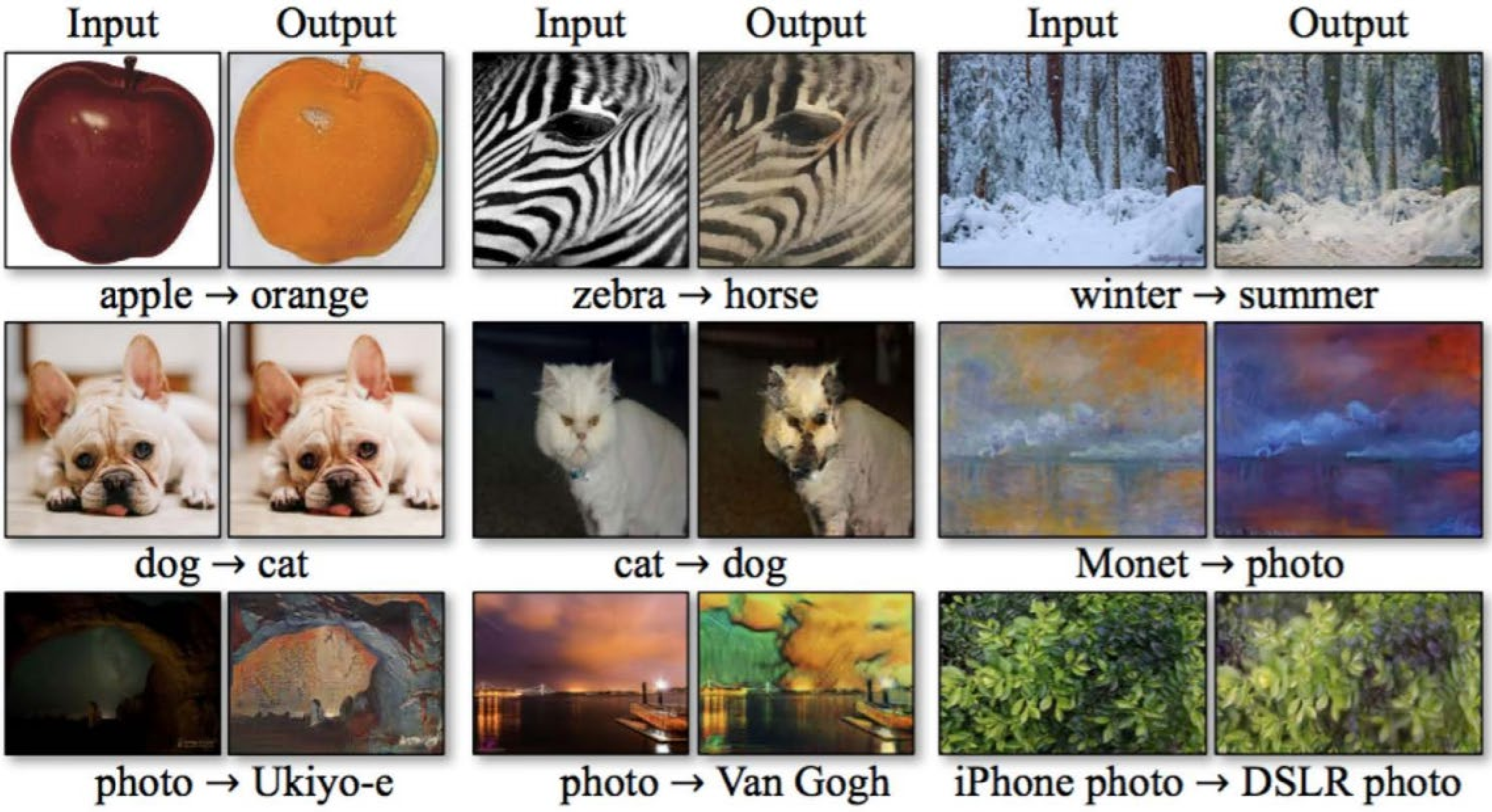
Unpaired image-to-image translation (CycleGAN, Zhu et al ‘17)

Can definitely fail…



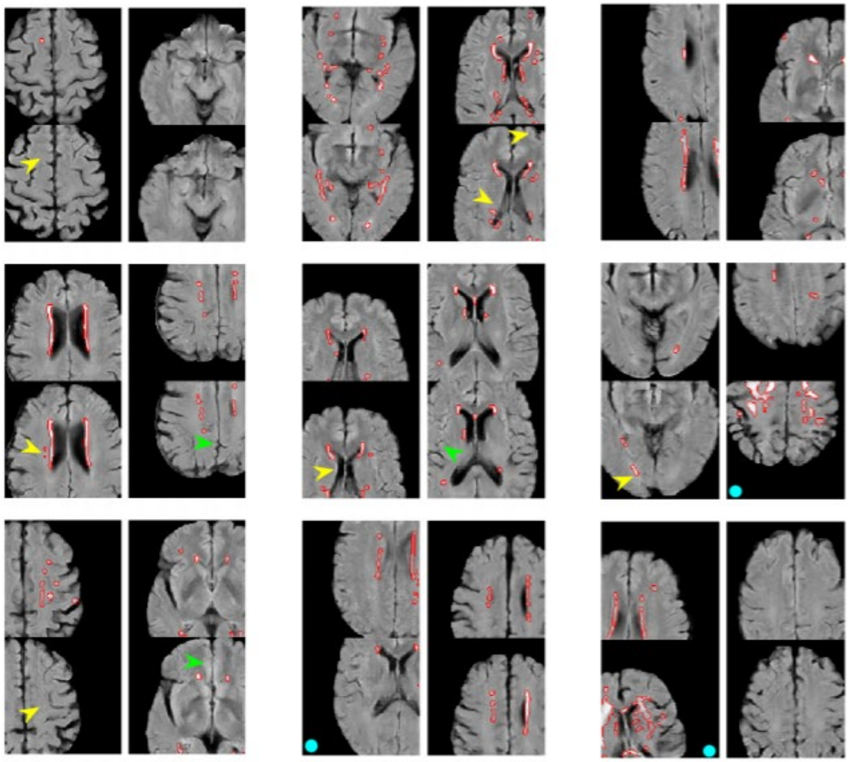
Unpaired image-to-image translation (CycleGAN, Zhu et al ‘17)

Can definitely fail…



Applications to other domains too

Brain lesion segmentation (Bowles et al 2018)



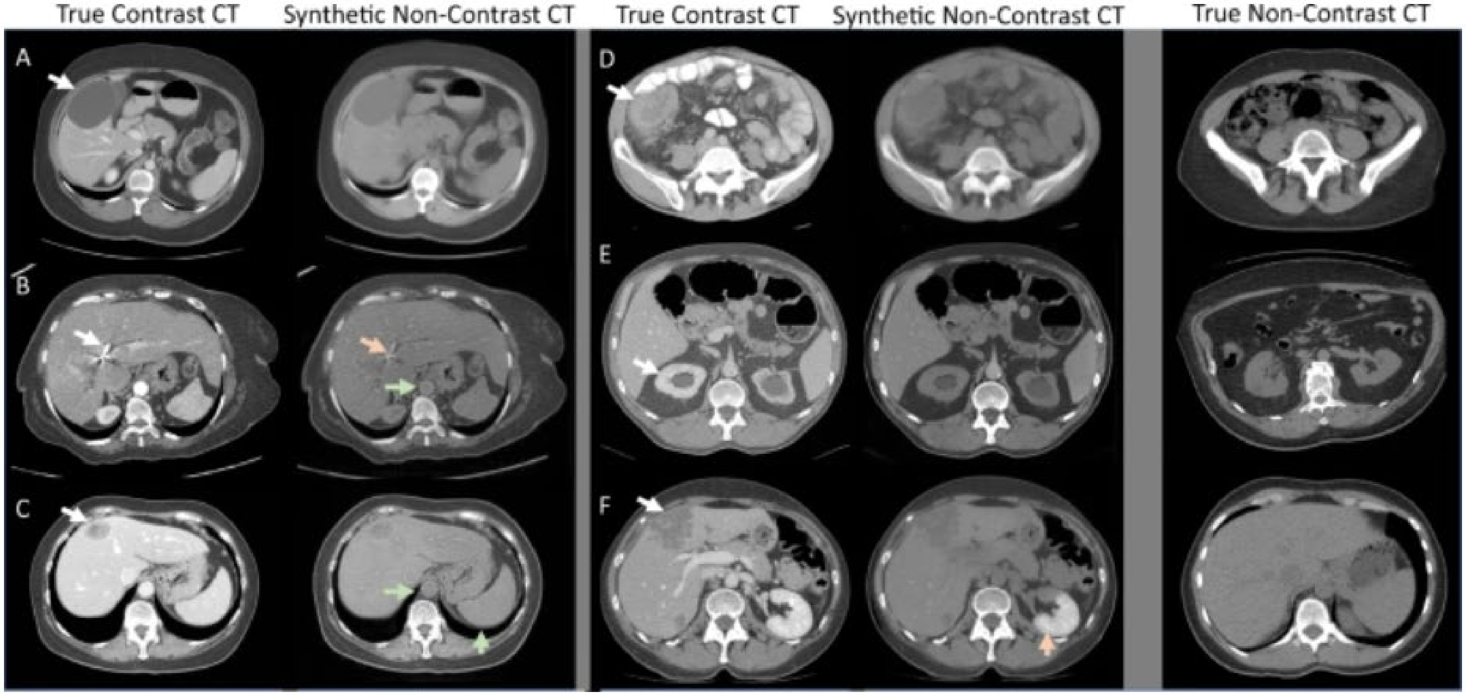
(a) 5 training images (b) 25 training images (c) 50 training images

Fig. 3: Synthetic images (top of pair) with their nearest neighbours in the training set (bottom of pair) from GANs trained on patches from 5, 25 and 50 real MR images. Some local signs of successful augmentation are indicated using green (same lesions, different anatomy) and yellow (same anatomy, different lesions) arrows, and novel images (new anatomy and lesions) are shown with blue dots.

Applications to other domains too

CT scan segmentation (Sanfort et al, Nature 2019)

Figure 1



Examples of true IV contrast CT scans (left column) and synthetic non-contrast CT scans generated by a CycleGAN. The rightmost column shows unrelated example non-contrast images. Overall the synthetic non-contrast images appear convincing - even when significant abnormalities are present in the contrast CT scans.

The pros and cons of GANs

Pros

Photorealism: photorealistic images, even w/ **relatively small models.**

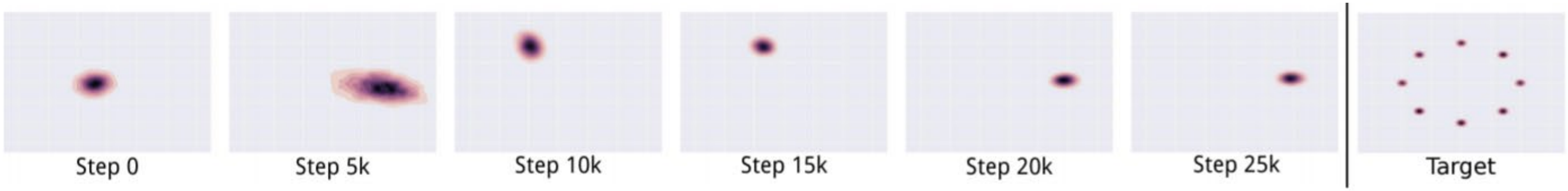
Efficient sampling: easy to draw samples from model (unlike e.g. energy models).

Cons

Unstable training: min-max problem – typically optimization much less stable than pure minimization.

Mode collapse: training only recovers some of the "modes" of the underlying distribution.

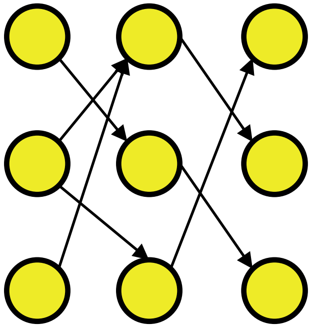
Evaluation: no likelihood, so hard to evaluate fit.



Middle ground:
"Invertible GANs/Normalizing Flows"

Can we "marry" likelihood models w/ GANs?

Suppose generator g: $\mathbb{R}^d \to \mathbb{R}^d$ were **invertible**.



Recall from the prev. lecture: if we denote by $\phi(z)$ the density of z under the standard Gaussian, by the change of variables formula:

$$P_g(x) = \phi(g^{-1}(x))\left|\det\left(J_x\left(g^{-1}(x)\right)\right)\right|$$

Hence, we can write down the likelihood in terms of the parameters of g^{-1} under this model!

Middle ground: “Invertible GANs/Normalizing Flows”

$$P_g(x) = \phi(g^{-1}(x))|\det\left(J_x\left(g^{-1}(x)\right)\right)|$$

Hence, denoting $g^{-1} = f_\theta$, for some family of parametric functions $\{f_\theta, \theta \in \Theta\}$, the max-likelihood estimator solves

$$\max_\theta \sum_{i=1}^{N} \log \phi(f_\theta(x_i)) + \log|\det\left(J_x\left(f_\theta(x_i)\right)\right)|$$

If we can evaluate and differentiate the above objective efficiently, we can do gradient-based likelihood fitting.

Choosing invertible transforms

Note that since the change-of-variables formula composes, so if $f_\theta = f_1 \circ f_2 \circ \cdots f_L$, we have

Value of k-th layer

$$\log p_\theta(x) = \sum_{i=1}^{N} \log \phi(f_\theta(x_i)) + \sum_{k=1}^{L} \log|\det(J_x(f_k\,(h_k(x_i)))|$$

So, if we can design a “simple” family of invertible transforms, we can just keep composing it.

Try 1: *General linear maps.*

Poor representational power: composition of linear maps is linear. If x = Az, and z is sampled from a Gaussian – x is Gaussian too.

Inefficient: Evaluating determinant of a d x d matrix takes O(d^3) time – infeasible.

Choosing invertible transforms

Try 2: *Elementwise (possibly non-linear) maps.*

Suppose that $f_\theta(x) = \left(f_\theta(x_1), f_\theta(x_2), \dots, f_\theta(x_d)\right)$

Efficient evaluation: Determinant is diagonal (since $\frac{\partial f_\theta(x_i)}{\partial x_j} = 0$, for $i \neq j$), so $\det(J_x\left(f_\theta(x)\right) = \Pi_i \frac{\partial f_\theta(x_i)}{\partial x_i}$.

Poor representational power: Transforms don't "combine" coordinates.

But, even if a matrix is triangular, Jacobian is just the product of the diagonals!!

Choosing invertible transforms

Try 3: *NICE/RealNVP (Non-linear Independent Component Estimation)*

Divide the coordinates of x into two sub-vectors with half the coords: $x_{1:\frac{d}{2}}, x_{\frac{d}{2}+1,d}$

Divide the coordinates of $z \coloneqq f_\theta(x)$ into two sub-vectors, $z_{1:\frac{d}{2}}, z_{\frac{d}{2}+1,d}$ and set:

$$z_{1:\frac{d}{2}} = x_{1:\frac{d}{2}}$$

$$z_{d/2+1,d} = x_{d/2+1,d} \odot \exp\left(s_\theta\left(x_{1:d/2}\right)\right) + t_\theta\left(x_{1:d/2}\right)$$

When is this invertible, and is the Jacobian efficiently calculated?

Choosing invertible transforms

Try 3: *NICE/RealNVP (Non-linear Independent Component Estimation)*

$$z_{1:\frac{d}{2}} = x_{1:\frac{d}{2}}$$

$$z_{\frac{d}{2}+1,d} = x_{\frac{d}{2}+1,d} \odot \exp\left(s_\theta\left(x_{1:d/2}\right)\right) + t_\theta\left(x_{1:d/2}\right)$$

$$J_x\left(f_\theta(x)\right) = \begin{bmatrix} I & 0 \\ \frac{\partial \mathbf{z}_{d/2:d}}{\partial \mathbf{x}_{1:d/2}} & \operatorname{diag}\left(\exp(\boldsymbol{s}_{\boldsymbol{\theta}}\left(\boldsymbol{x}_{\mathbf{1}:\boldsymbol{d}/\mathbf{2}}\right)\right) \end{bmatrix}$$

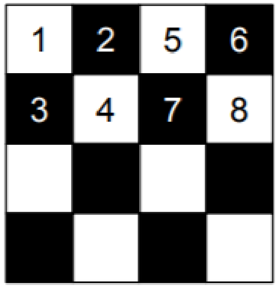
The determinant of a triangular matrix is the product of the diagonals!

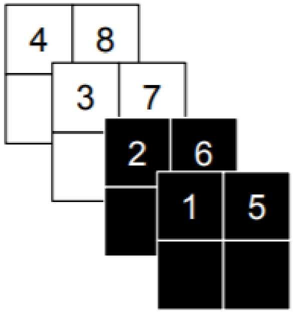
Hence, det $J_x\left(f_\theta(x)\right) = \Pi_i \exp\left(s_\theta\left(x_{1:d/2}\right)\right)_i$

If t_θ, s_θ is say, a neural net, easy to evaluate and take derivatives.

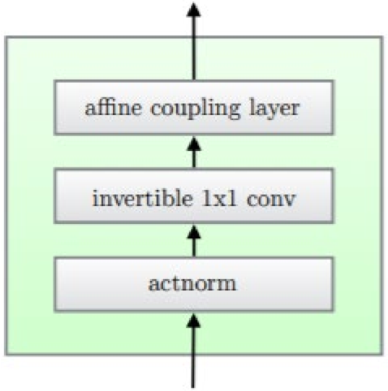
How to choose partitions?

NICE (Dinh et al '14), RealNVP (Dinh et al '16): The choice of partitions is fixed, checkerboard of channel-wise.





Glow (Kingma et al '18): add trained linear transforms between affine coupling layers – i.e. a generalization of a "learned" permutation



Some samples

NICE/RealNVP (Non-linear Independent Component Estimation)

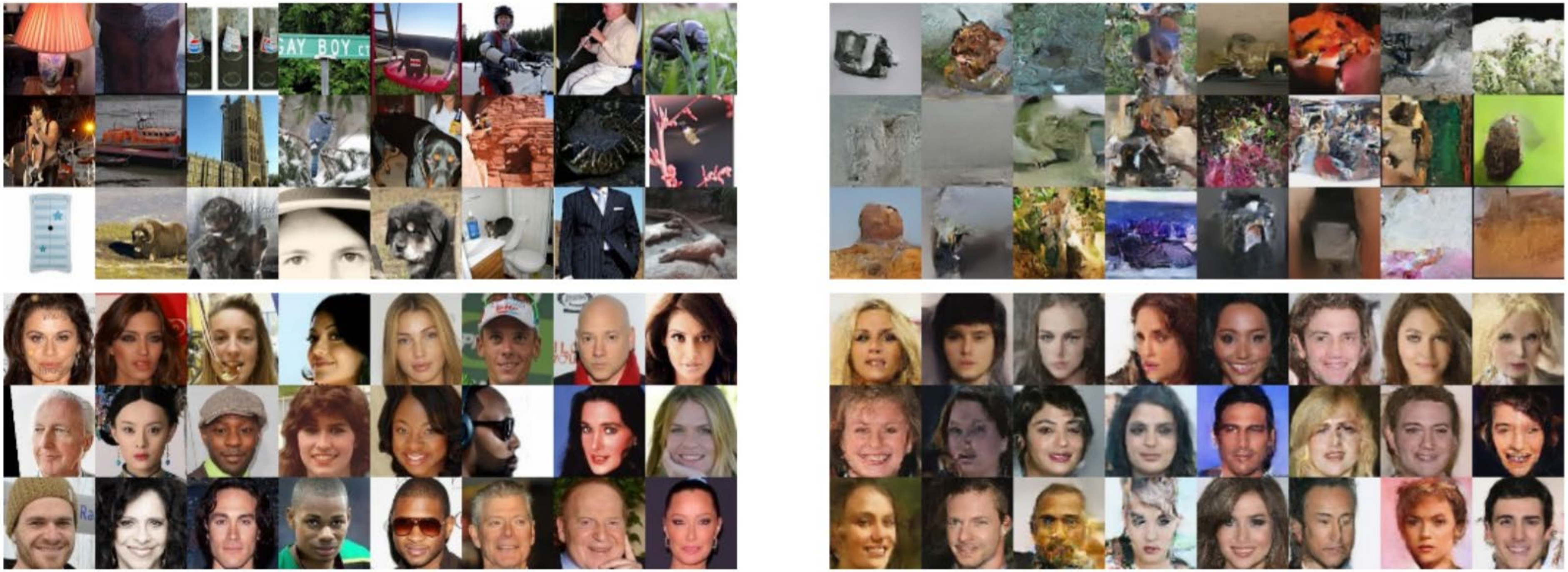


Figure from “*Density estimation using Real NVP*” by *Dinh et al ‘16*

Some samples

Glow

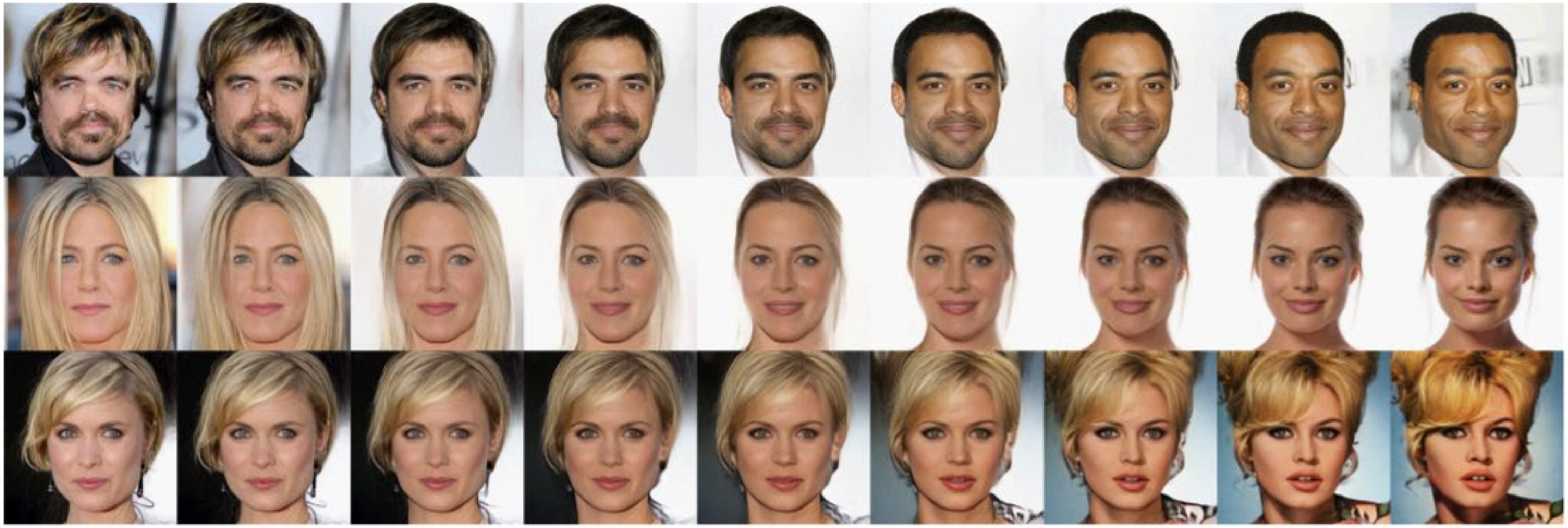
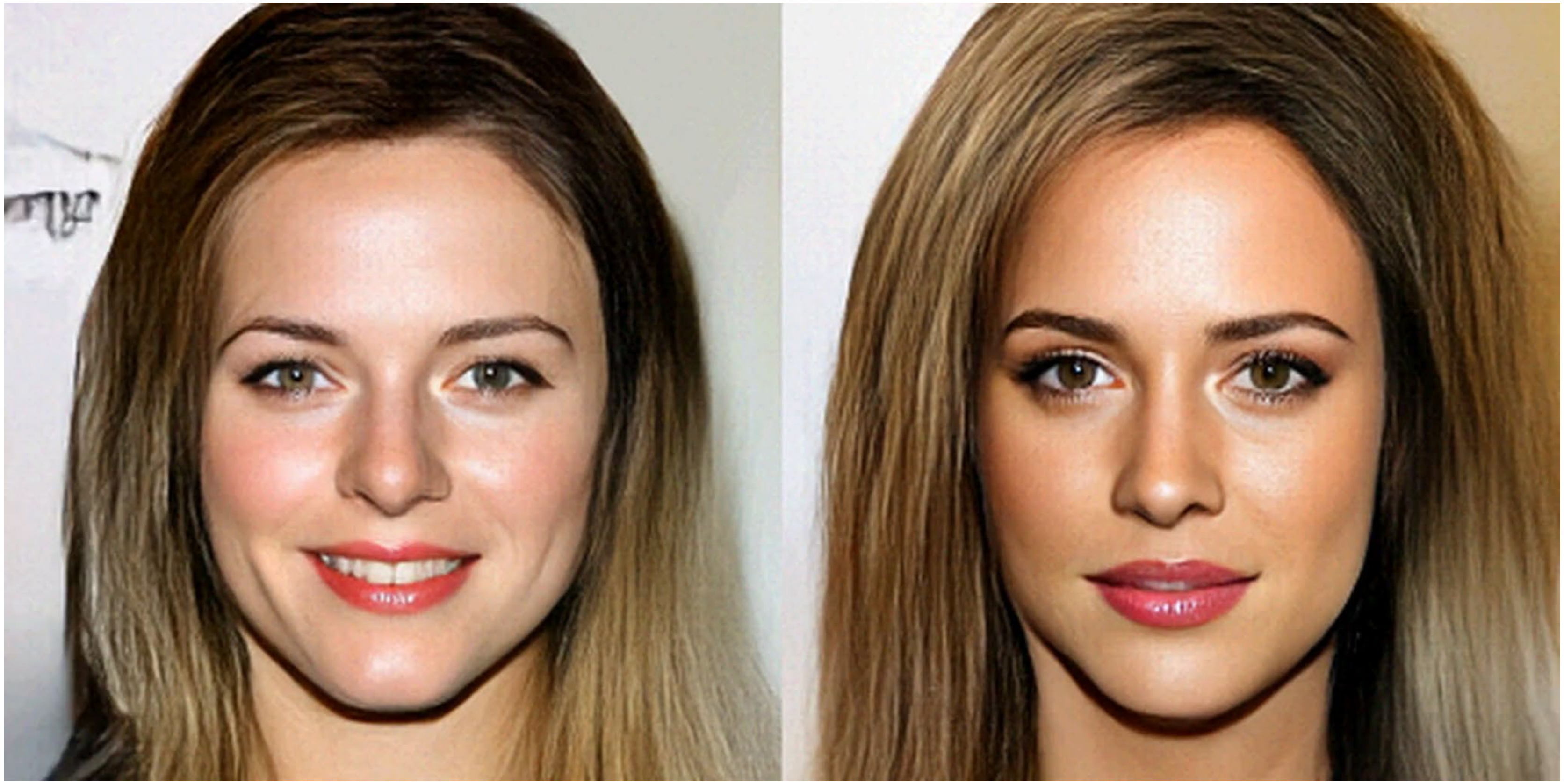


Figure 5: Linear interpolation in latent space between real images

Figure *from (Kingma et al ‘18)*

Some samples



https://openai.com/blog/glow/

The pros and cons of flow models

Pros

Photorealism: photorealistic images.

Efficient sampling: easy to draw samples from model.

Stabl(er) training: likelihood objective, pure minimization.

Evaluation: easier, comes with likelihood.

Cons

Extremely large: in practice, good models need to be *extremely* large (Glow: 40 GPUs for ~week)

Model depth: training gets harder as models are typically very deep. (Glow: ~1200 layers in total)

Ill-conditioned: Jacobian is close-to-singular, so objective blows up.

Some representational reasons why

(Koehler-Mehta-Risteski ‘20), half-baked conclusions:

Universal approximation with ill-conditioned affine couplings: any sufficiently nice distribution can be approximated arbitrarily closely with an (*ill-conditioned*) affine coupling network.

Learned permutations don’t help that much: any linear transform can be simulated via 8 affine coupling layers with any fixed partition.

Inevitable increase in depth: there is a distribution representable as a **small and shallow** generator, s.t. a ***much deeper*** invertible model is necessary to approximate it (*architecture-independent*).