Generative Adversarial Networks (GANs)

EE599 Deep Learning

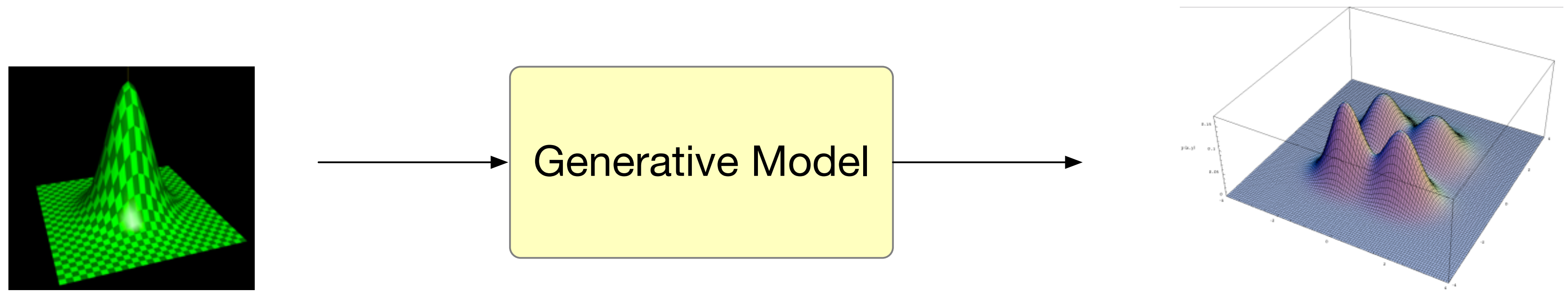
Keith M. Chugg
Spring 2020

USC University of Southern California

Outline for Slides

- Generative models
- GANs
  - Sample code
- Conditional GANs
- Style transfer with Cycle-GANs

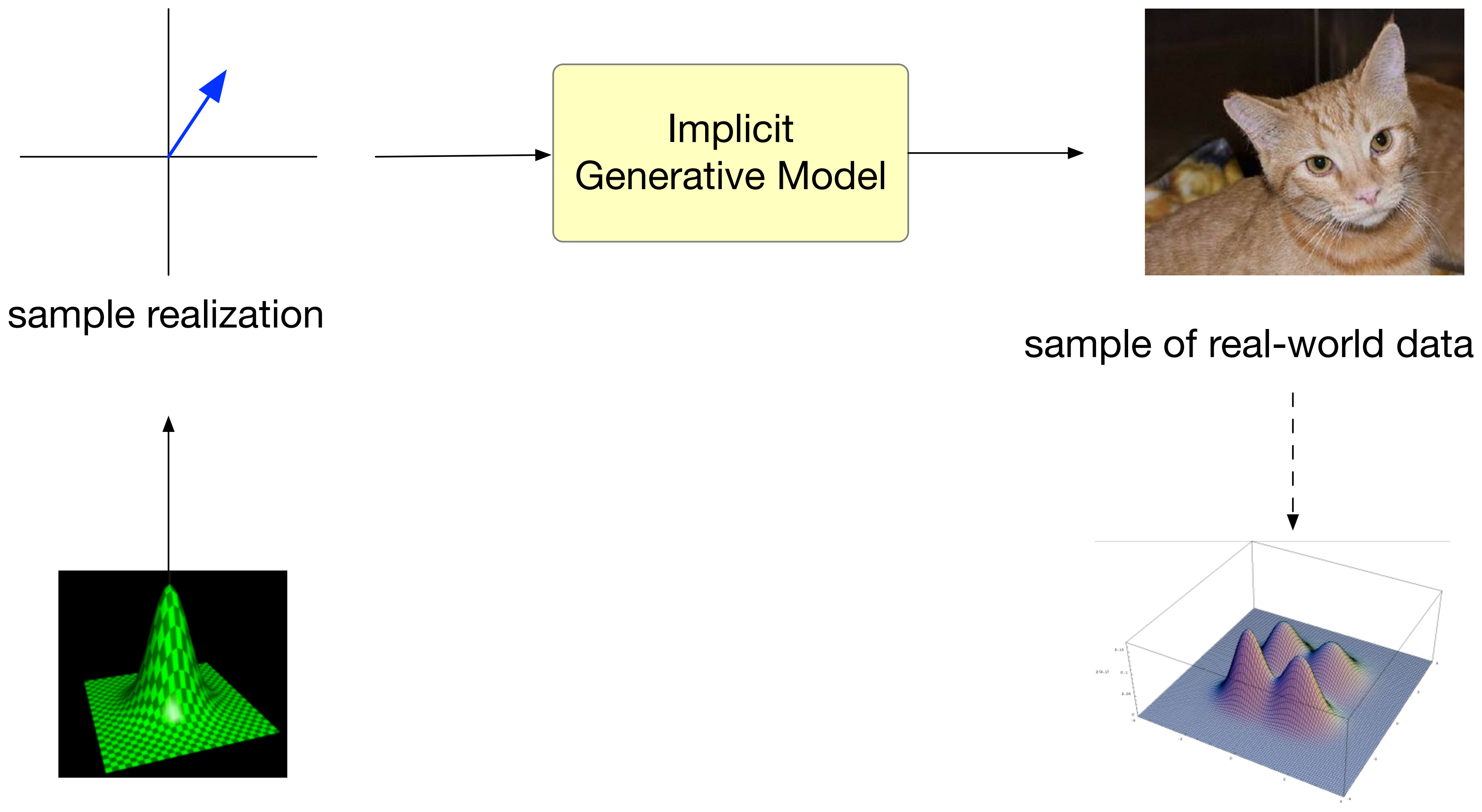
Generative Models



canonical distribution

distribution of real-world dataset

GANs are Implicit Generative Models

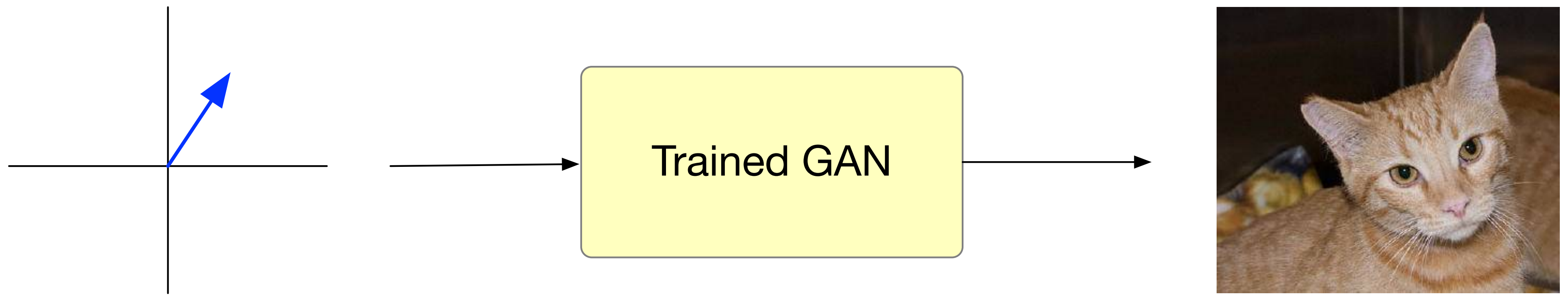


canonical distribution

distribution of real-world dataset

contributions from Brandon Franzke

Using a Trained GAN



sample realization from
canonical distribution

Put “noise” into the model and it generates real-world images as determined by the training set!

Very realistic high-resolution results have been obtained

We Touched on GAN Results Briefly Already

https://thispersondoesnotexist.com

not real people — output of a GAN driven by random noise!

How Can We Use GANs?

AI version of
“paint” (Nvidia Video)

Video games

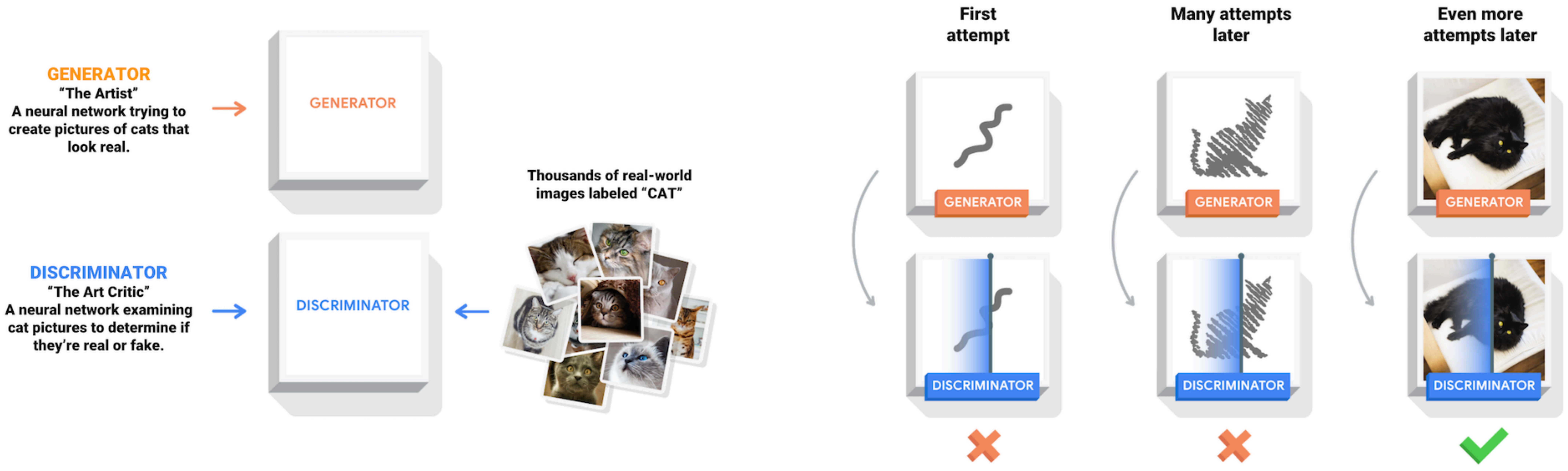
synthetic data to train
networks
(avoid the labeling problem)

style transfer

super-resolution
(enhance images)

Basic Idea of GANs

TensorFlow: Deep Convolutional GAN

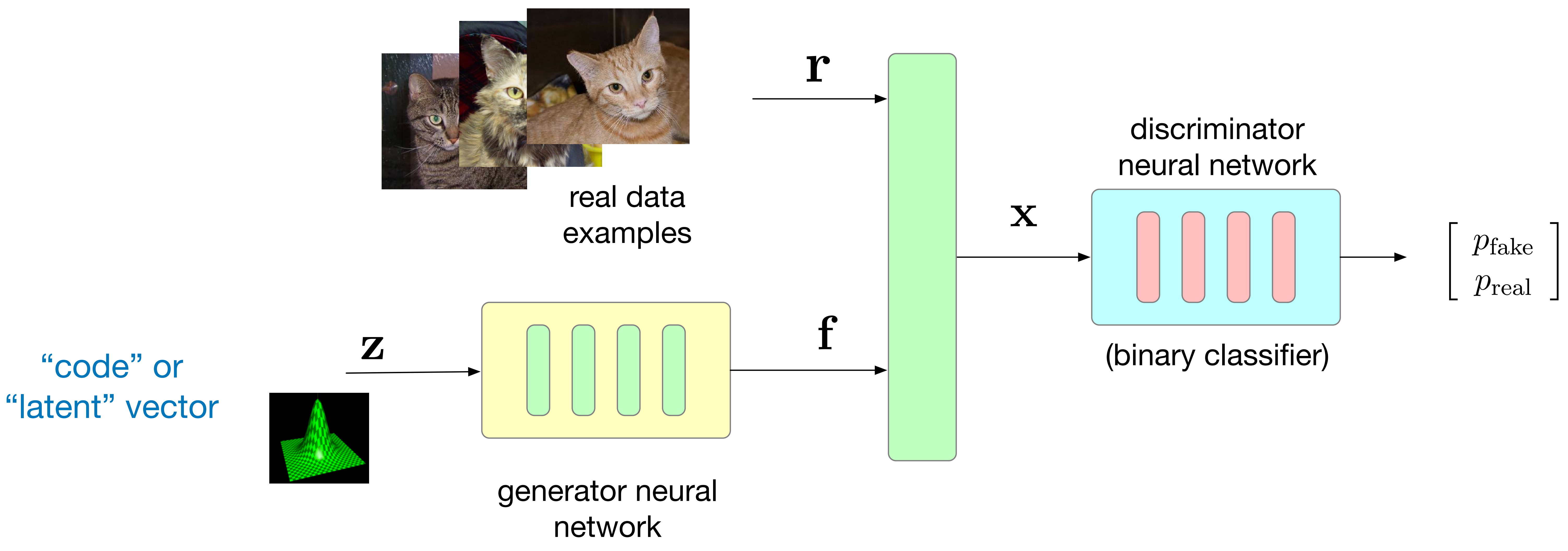


A **generator** network and a **discriminator** network are trained together

Generator: make fakes so well, the discriminator cannot tell them from real examples

Discriminator: classify fakes from real examples

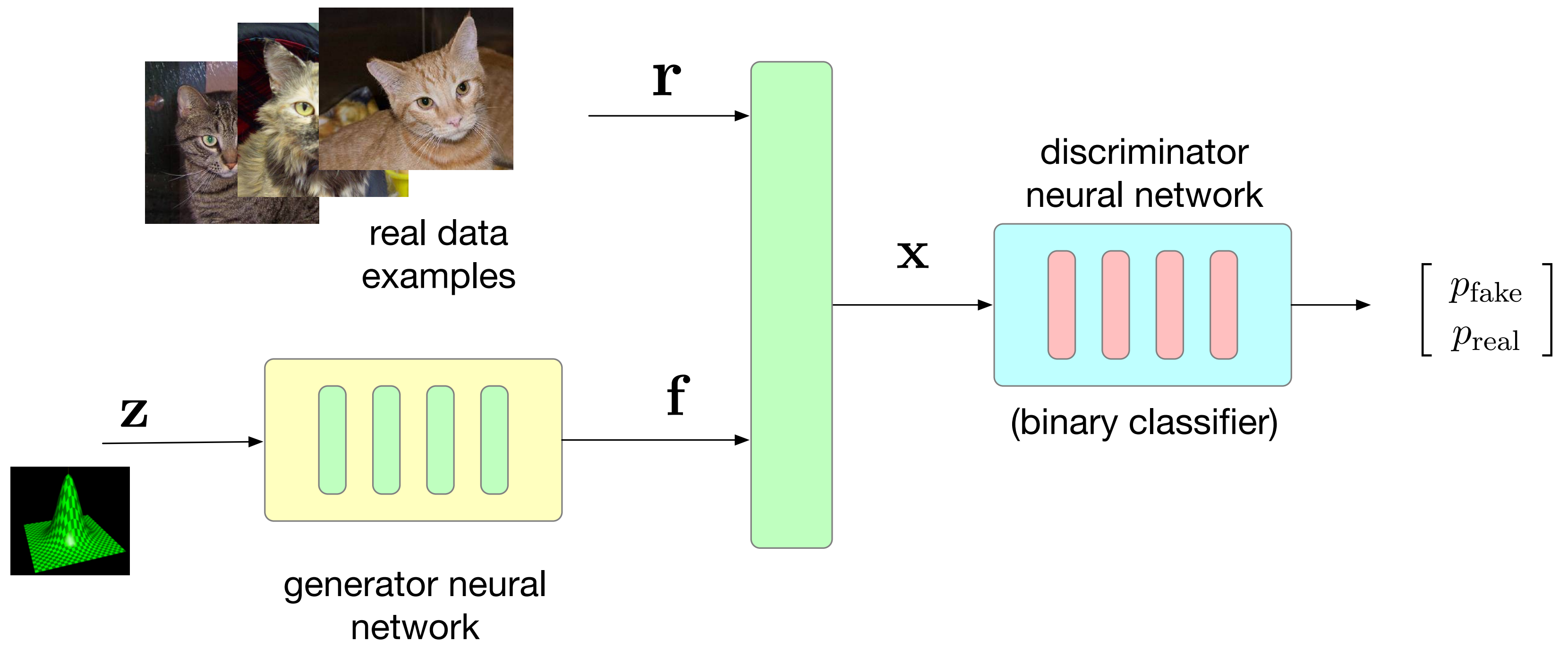
Training a GAN



Discriminator: input is mixture of real (**r**) and fake (**f**) input images, trained as a typical binary classifier (i.e., binary cross entropy loss)

Generator: must be trained with a cost function that encourages the generator to fool the discriminator and must backdrop through the discriminator

Training a GAN - Discriminator



labeling

fake <—> 0

real <—> 1

$$C_D = \mathrm{CE}\left(\begin{bmatrix} \ell_{\text{fake}} \\ \ell_{\text{real}} \end{bmatrix}, \begin{bmatrix} p_{\text{fake}} \\ p_{\text{real}} \end{bmatrix}\right)$$

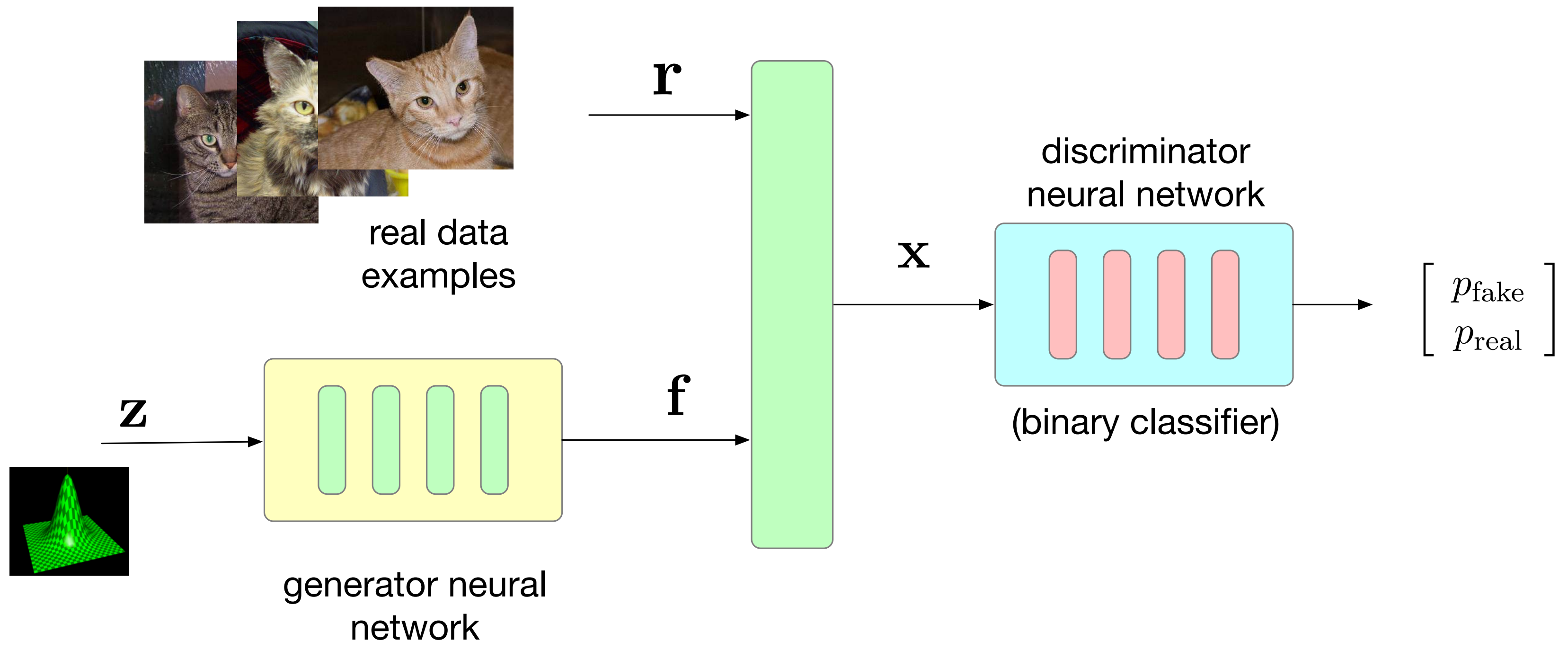
$$= -(\ell_{\text{real}} \log p_{\text{real}}) - (\ell_{\text{fake}} \log p_{\text{fake}})$$

$$= \begin{cases} -\log p_{\text{fake}} = -\log(1 - p_{\text{real}}) & \mathbf{x} \text{ is fake} \\ -\log p_{\text{real}} & \mathbf{x} \text{ is real} \end{cases}$$

standard binary cross-entropy loss for binary classification

showing both (hard) labels and both probabilities for clarity

Training a GAN - Generator



labeling

fake <—> 0

real <—> 1

$$C_G = \text{CE}\left(\begin{bmatrix} \ell_{\text{fake}} \\ \ell_{\text{real}} \end{bmatrix}, \begin{bmatrix} p_{\text{real}} \\ p_{\text{fake}} \end{bmatrix} \right)$$

$$= -(\ell_{\text{real}} \log p_{\text{fake}}) - (\ell_{\text{fake}} \log p_{\text{real}})$$

$$= \begin{cases} -\log p_{\text{real}} & \mathbf{x} \text{ is fake} \\ -\log p_{\text{fake}} = -\log(1 - p_{\text{real}}) & \mathbf{x} \text{ is real} \end{cases}$$

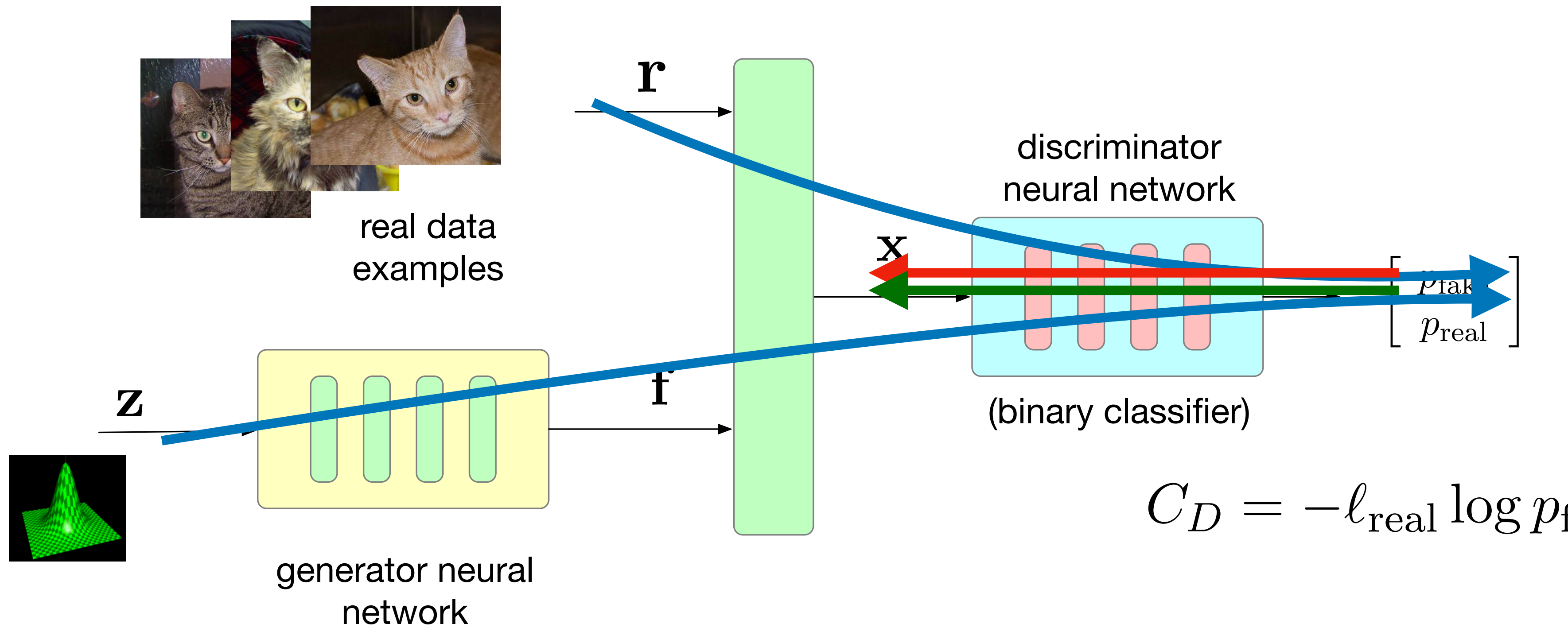
trains the generator to fool the discriminator — i.e., if standard BCE ~minimizes classification error, this ~minimizes classification accuracy

when training the generator, we only give fake examples, so the loss is simply:

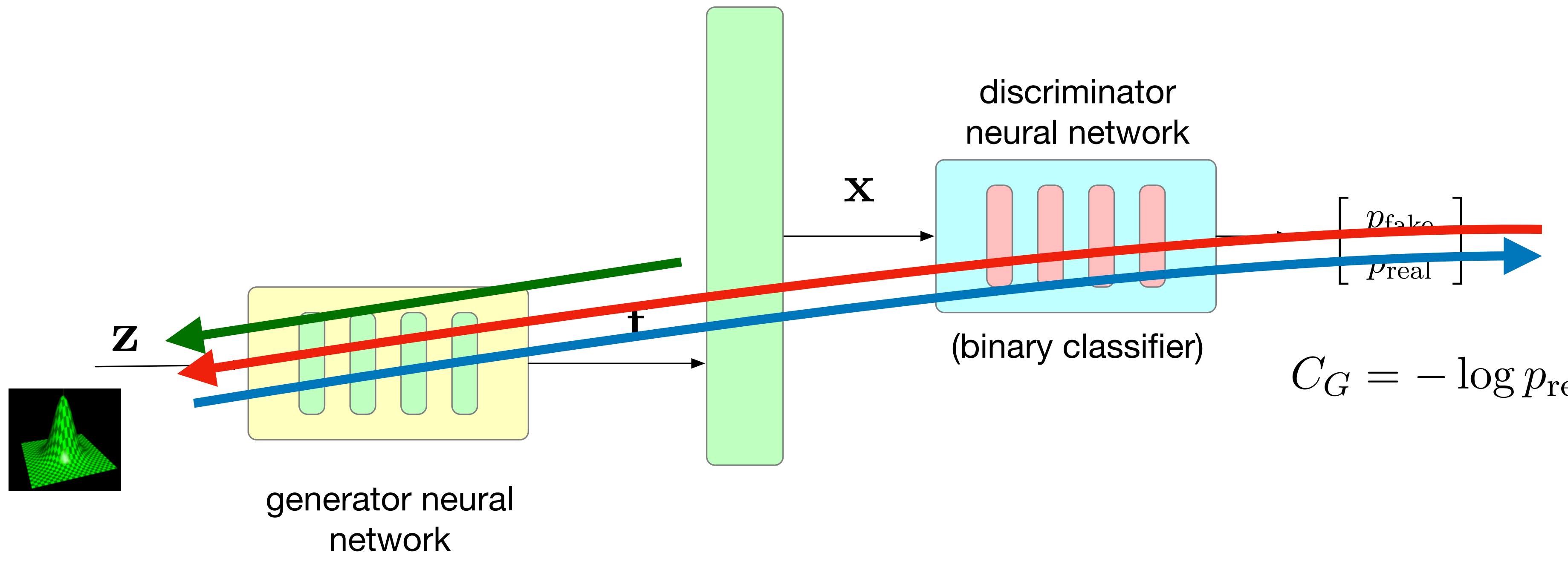
$$\boxed{C_G = -\log p_{\text{real}}}$$

this can be accomplished by labeling the fakes as reals and using standard BCE!

Training a GAN - All Together



$$C_D = -\ell_{\text{real}} \log p_{\text{fake}} - (1 - \ell_{\text{real}}) \log(1 - p_{\text{real}})$$



$$C_G = -\log p_{\text{real}}$$

Discriminator Training

generate fakes and use reals, backprop only through the discriminator and update the discriminator parameters

forward computation

back-prop. gradient computation

parameter updates

Generator Training

generate fakes, label as real, backprop through the discriminator and update the generator weights

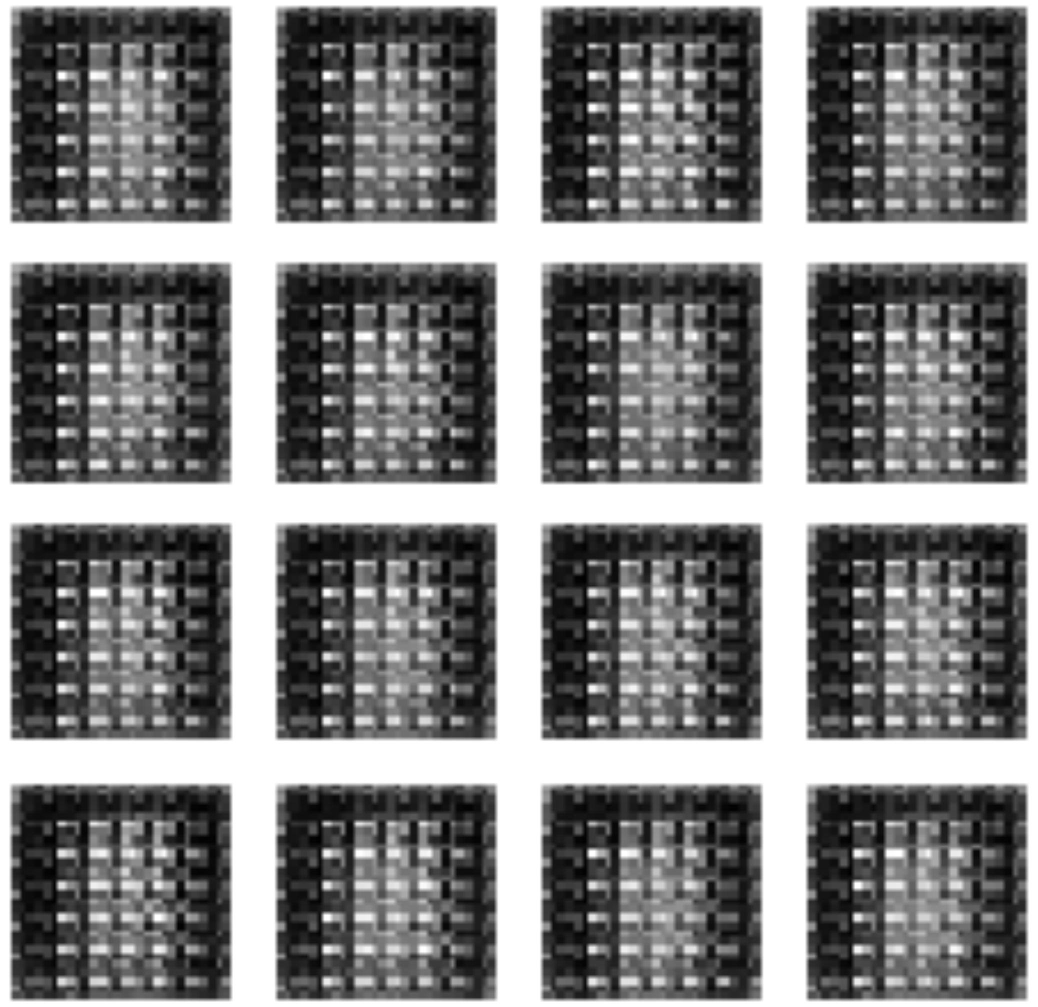
GAN — Code Examples

TensorFlow tutorial with tf.keras model definition

tf.keras example by overloading model.fit

blog post: using standard keras!

pytorch example from U Toronto CSC321



All of these generate fake examples of MNIST

Aside: Conv2DTranspose layer

this is an “up-sampling” layer where the size of the output image is larger than the size of the input image

left-to-right, top-to-bottom rastor scan

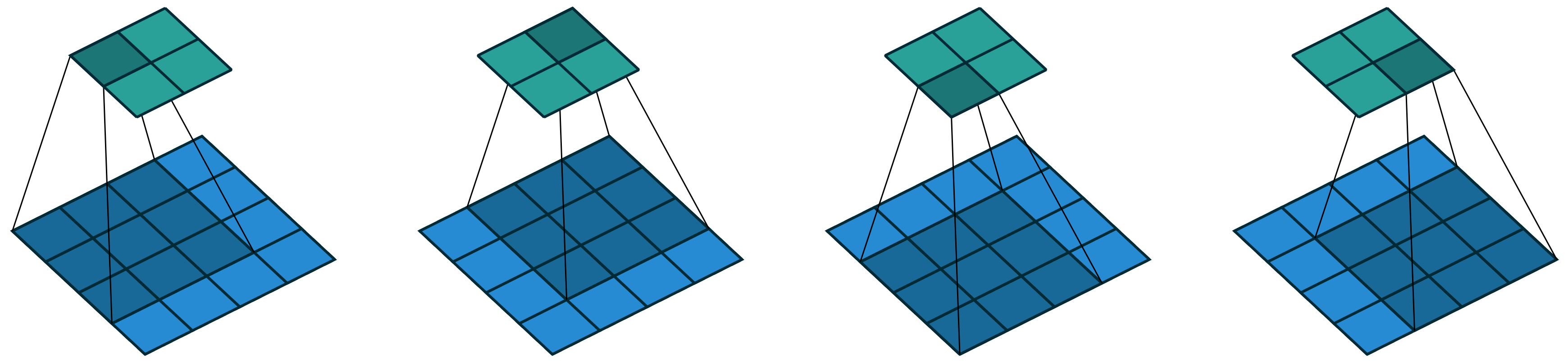


Figure 2.1: (No padding, unit strides) Convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$).

$$\begin{pmatrix}
w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 & 0 \\
0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2} & 0 \\
0 & 0 & 0 & 0 & 0 & w_{0,0} & w_{0,1} & w_{0,2} & 0 & w_{1,0} & w_{1,1} & w_{1,2} & 0 & w_{2,0} & w_{2,1} & w_{2,2}
\end{pmatrix}$$

16 pixels —> 4

matrix mapping (16 —> 4)

If we use the transpose of the convolution matrix, it will map 4 pixels to 16 pixel

this is the conv2DTranspose operation

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

Aside: Conv2DTranspose layer

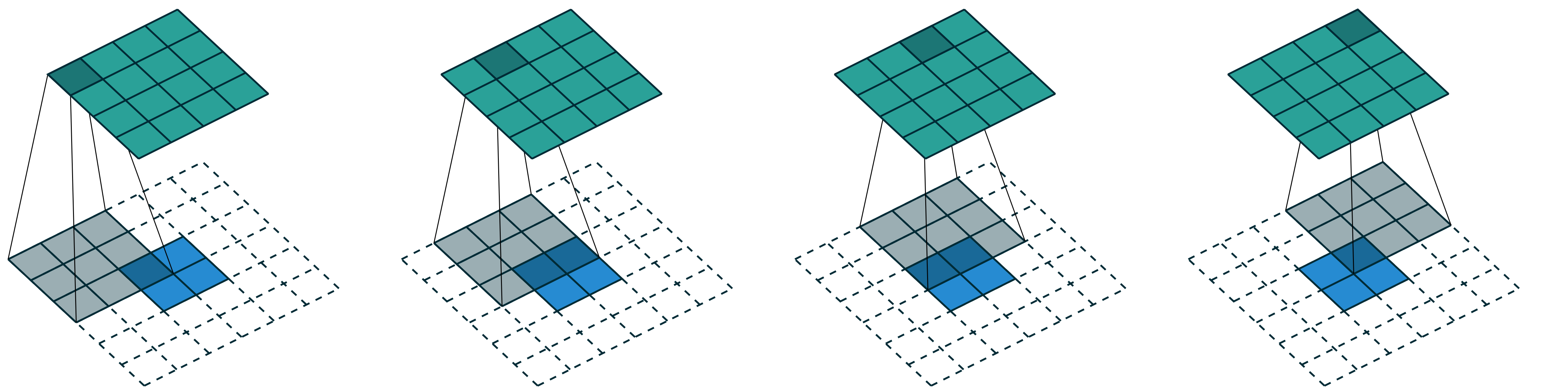


Figure 4.1: The transpose of convolving a 3×3 kernel over a 4×4 input using unit strides (i.e., $i = 4$, $k = 3$, $s = 1$ and $p = 0$). It is equivalent to convolving a 3×3 kernel over a 2×2 input padded with a 2×2 border of zeros using unit strides (i.e., $i' = 2$, $k' = k$, $s' = 1$ and $p' = 2$).

Example of 2D transpose Convolution corresponding to standard convolution on previous slide

Note: carefully inspecting the CNN backprop equations shows that the "transpose" filter is used in the reverse direction

aka: deconvolution and fractionally-stride convolution

Dumoulin, Vincent, and Francesco Visin. "A guide to convolution arithmetic for deep learning." *arXiv preprint arXiv:1603.07285* (2016).

GAN — Key References

Goodfellow, Ian J., et al. "Generative adversarial networks." Proc. 27th Int. Conf. Neural Information Processing Systems. 2014.

Original GAN reference

Goodfellow, Ian J., et al. "Generative adversarial networks." Proc. 27th Int. Conf. Neural Information Processing Systems. 2014.

NuerIPS Tutorial

Karras, Tero, et al. "Progressive growing of GANs for improved quality, stability, and variation." arXiv preprint arXiv:1710.10196 (2017).

First excellent high-res deep-fake human faces

I also used the notes from here: **https://www.cs.toronto.edu/~rgrosse/courses/csc321_2018/**

GAN — Challenges

Training GANs can be “finicky” since the generator and discriminator need to get better together

If one gets “ahead” of the other, the training process will degenerate into one “winning” — e.g., mode collapse

highly cited paper on improved techniques for training GANs

Salimans, Tim, et al. "Improved techniques for training gans." Advances in neural information processing systems. 2016.

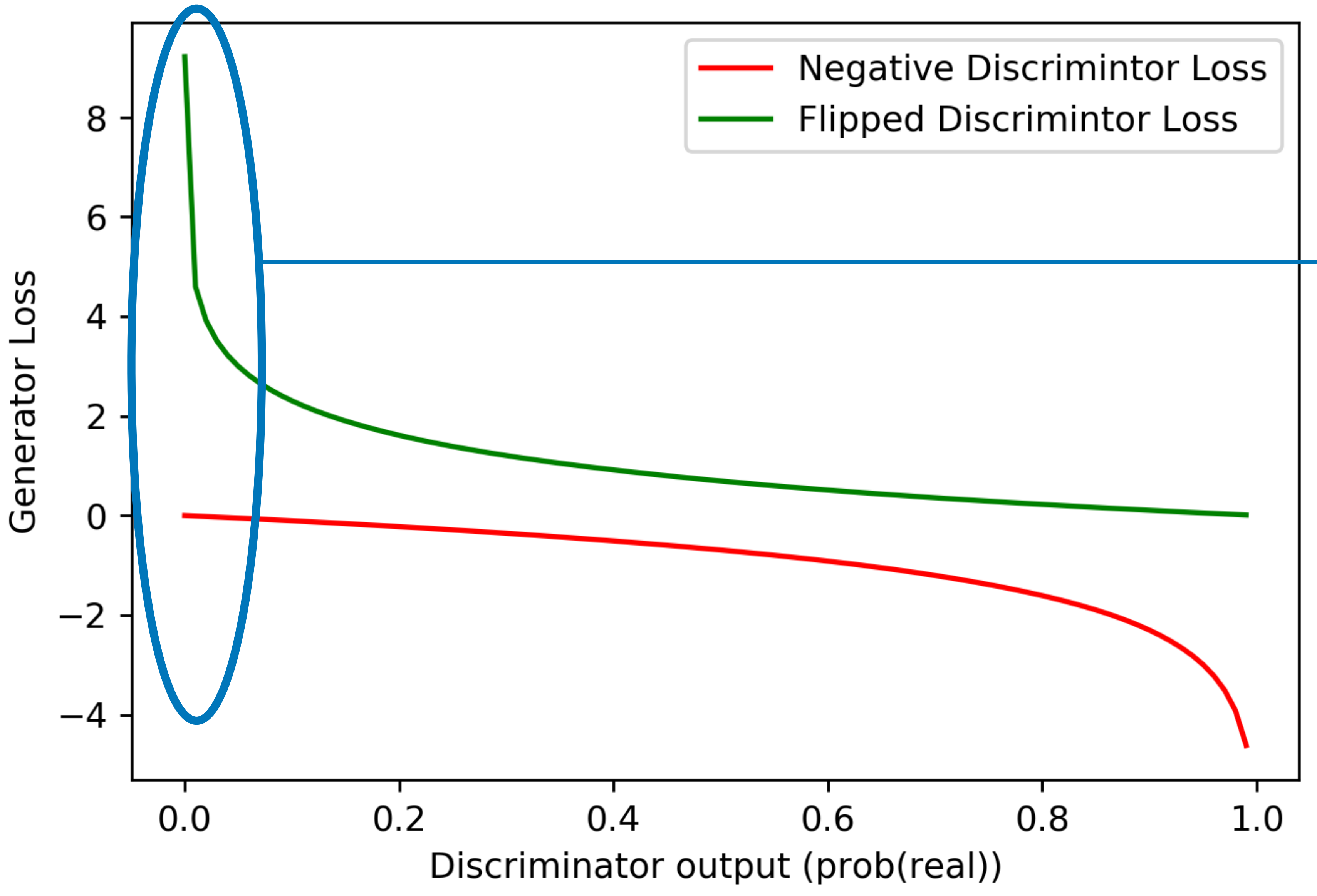
Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein GAN." arXiv preprint arXiv:1701.07875 (2017).

Note On Generator Loss

You will often see the idea that the generator should use:

$$C_G = -C_D = \ell_{\text{real}} \log p_{\text{real}} + (1 - \ell_{\text{real}}) \log(1 - p_{\text{real}}) \equiv \log(1 - p_{\text{real}})$$

This comes from the original 2014 paper where GANs are formulated as a minimax optimization, then they suggest the "flipped" loss I showed



Using the C_g = - C_d yields a low derivative when p_real is low — i.e., when the generator is doing a poor job of filing the discriminator

$$\frac{d}{d\epsilon} \log(1-\epsilon) = \frac{-1}{1-\epsilon} \quad \text{vs} \quad \frac{d}{d\epsilon} - \log(\epsilon) = \frac{-1}{\epsilon}$$

for $\epsilon \ll 1$

Outline for Slides

- Generative models
- GANs
  - Sample code
- Conditional GANs
- Style transfer with Cycle-GANs

Conditional GANs

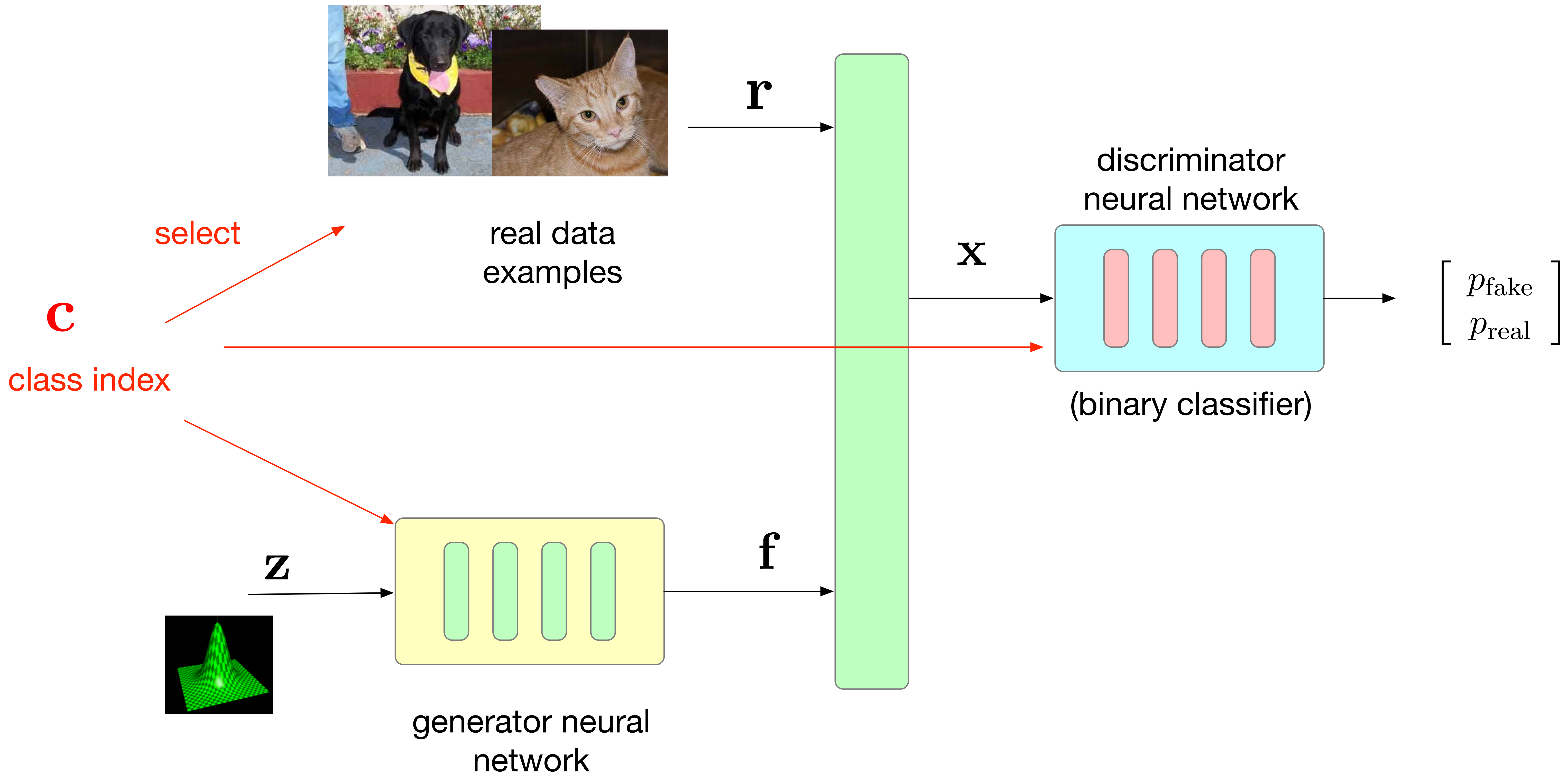
Standard GANs can generate arbitrary fake examples of real data — e.g., MNIST digits

but what if we want to generate an example from a specific class — e.g., generate a fake MNIST “6”

Conditional GANs addresses this desire

Mirza, Mehdi, and Simon Osindero. "Conditional generative adversarial nets." arXiv preprint arXiv:1411.1784 (2014).

Conditional GANs

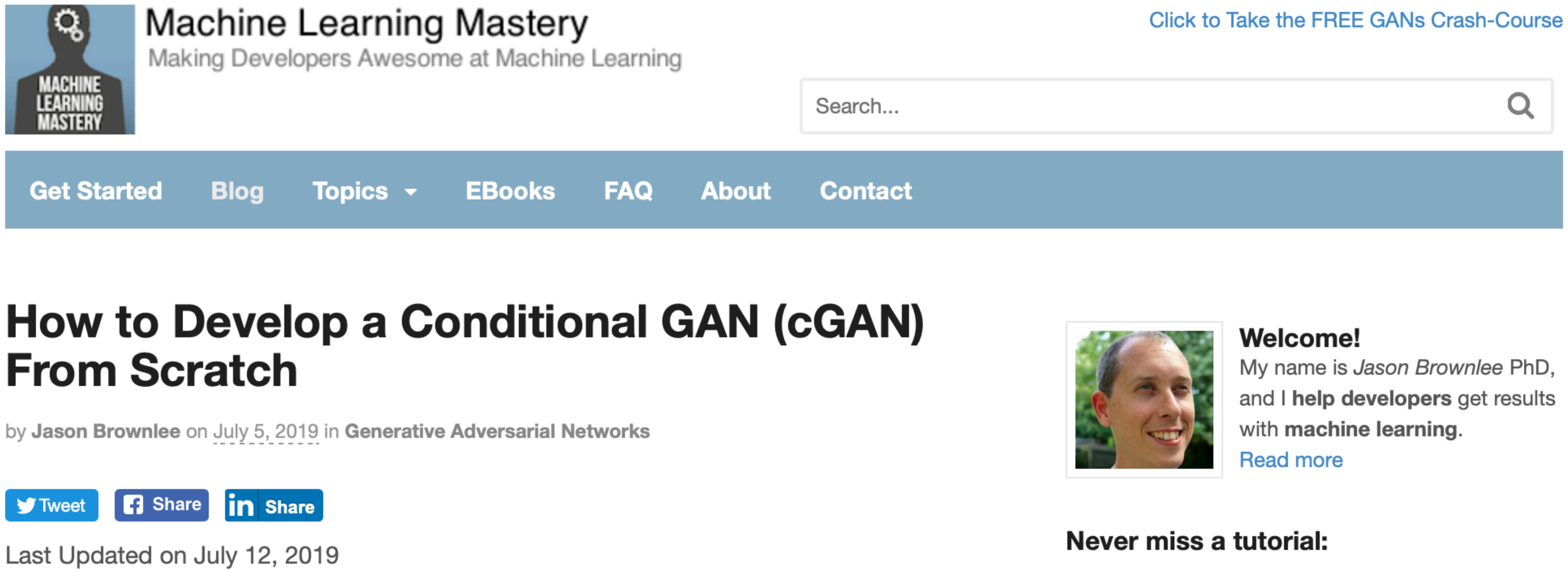


the class index is passed as input to both the generator and the discriminator — all else is the same

Conditional GANs - Example

nice example using keras…

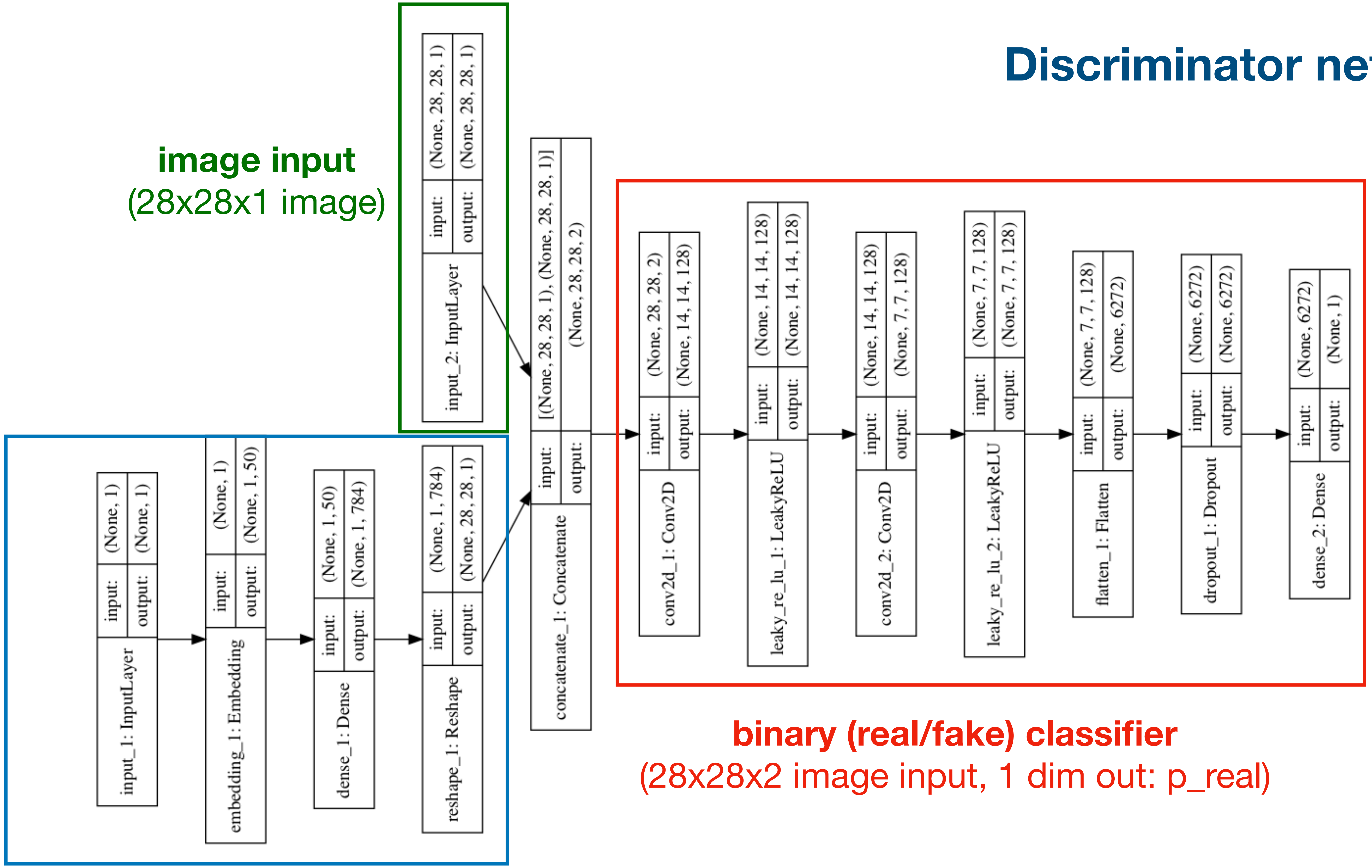
[blog post: using standard keras!]



Conditional GANs - Example, Discriminator

Discriminator network

image input
(28x28x1 image)

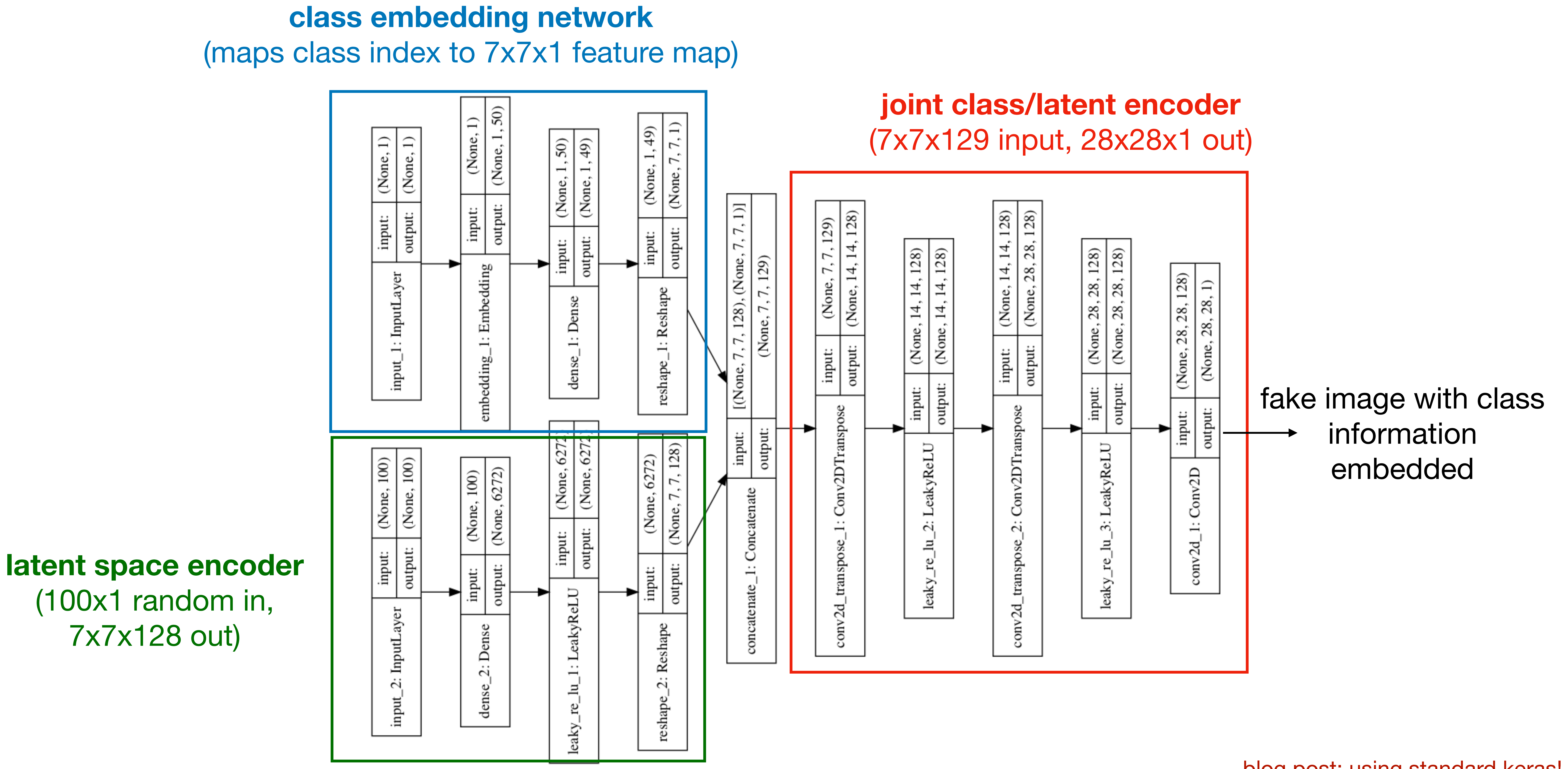


binary (real/fake) classifier
(28x28x2 image input, 1 dim out: p_real)

class embedding network
(maps class index to 28x28x1 image)

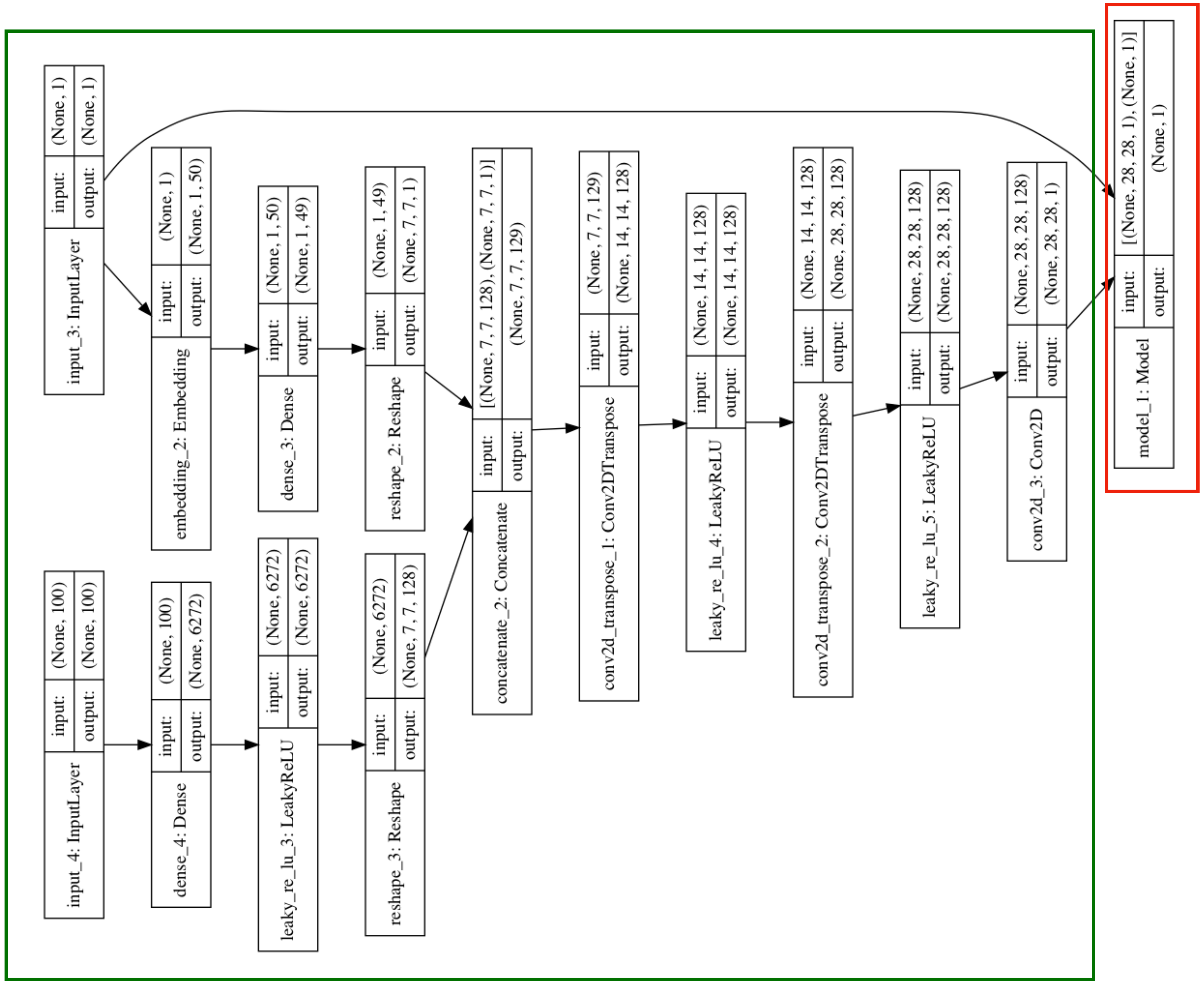
blog post: using standard keras!

Conditional GANs - Example, Generator



blog post: using standard keras!

Conditional GANs - Example, Generator



discriminator

generator

blog post: using standard keras!

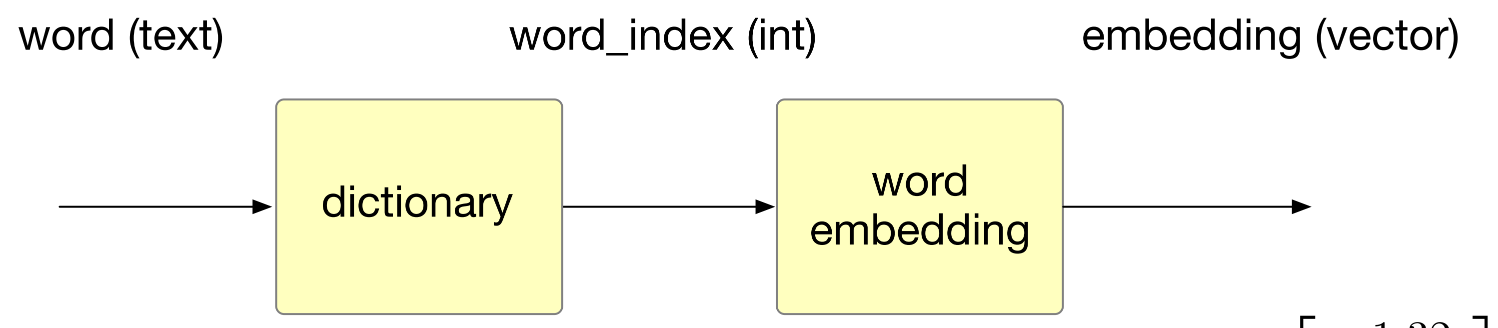
Conditional GANs - Example, Entire GAN

example fake fashion-MNIST generated by class

blog post: using standard keras!

Aside: Embedding Layer

We will discuss further during NLP lecture



"house"

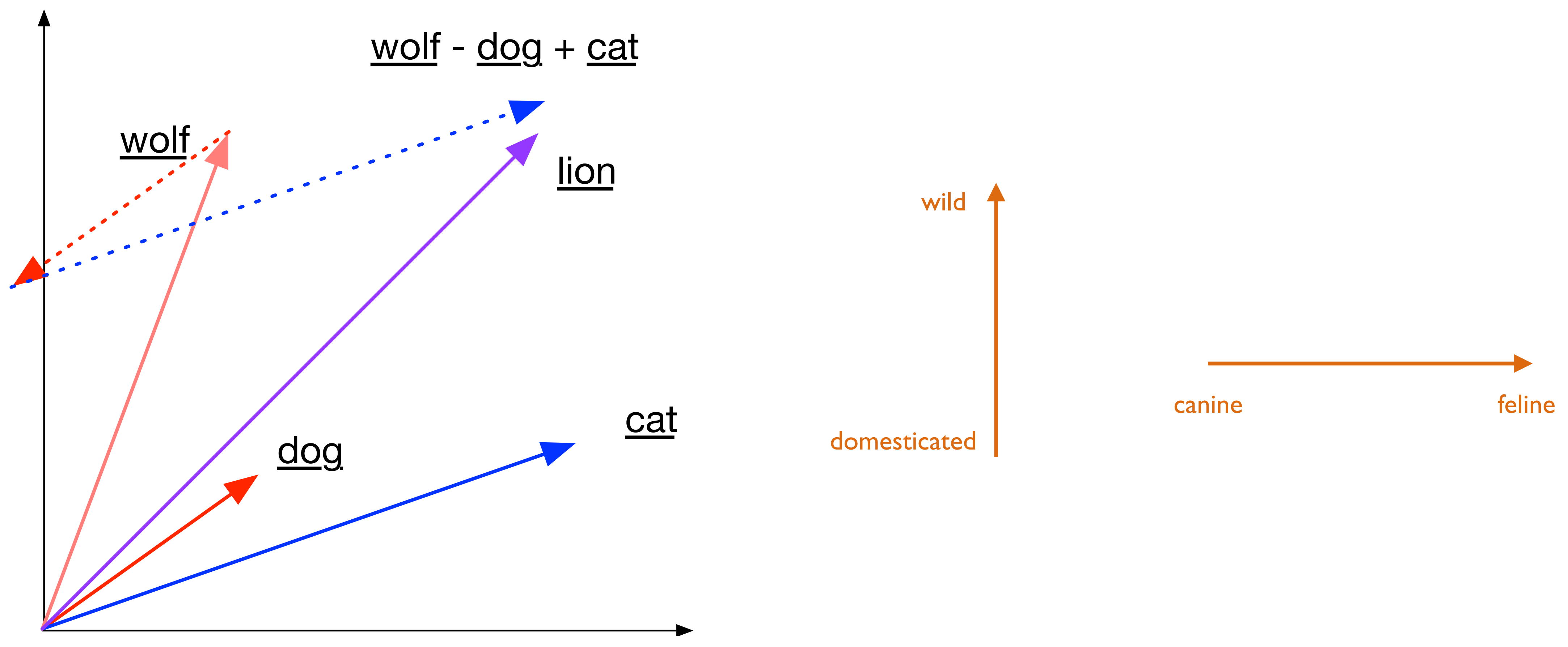
92

$$\mathbf{v}_{92} = \begin{bmatrix} -1.32 \\ +2.13 \\ 4.28 \\ \vdots \\ -3.12 \end{bmatrix}$$

text from a vocabulary of size V (e.g., V=10,000)

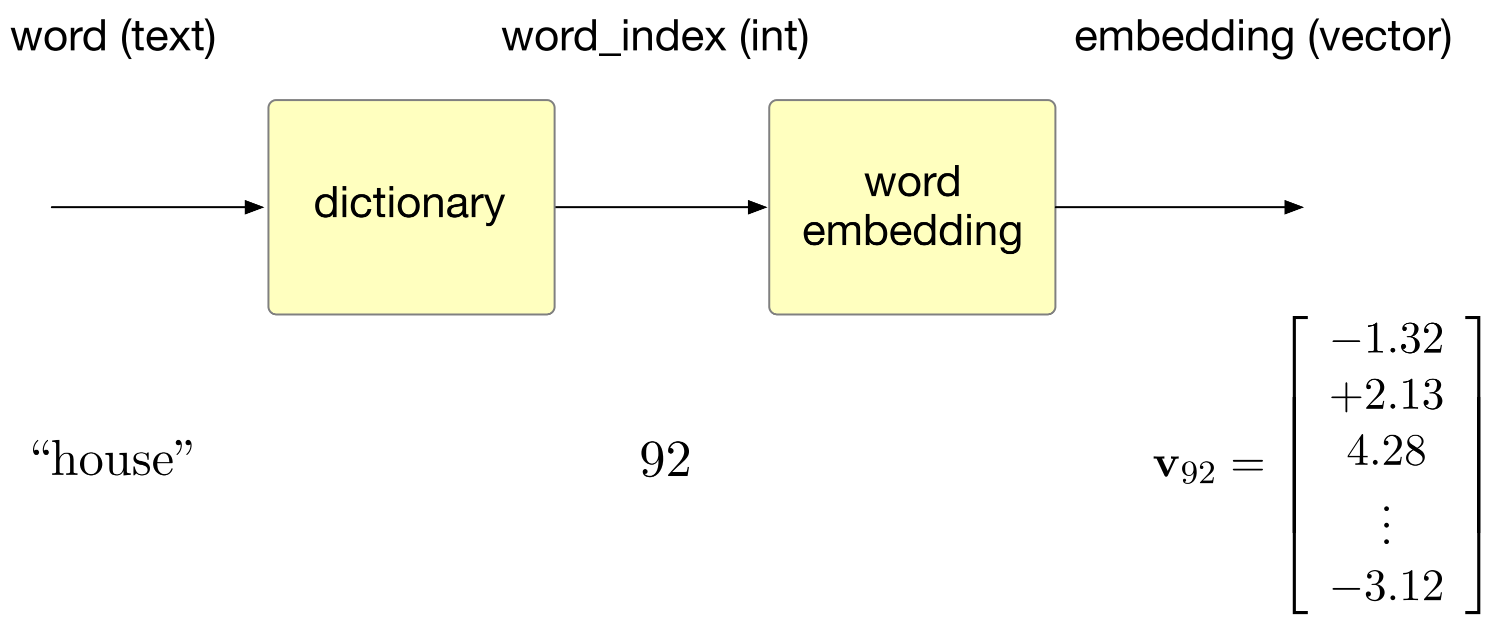
int between 1 and V (one-to-one)

word embedding (vector) dim = D (e.g., 300)



PCA used to reduce from D to 2 dimensions

Aside: Embedding Layer



Can use in other contexts as well — e.g., the class index embedding

Embedding Layer: basically a one-hot-encoder followed by Dense layer

```
tf.keras.layers.Embedding(
    input_dim, output_dim, embeddings_initializer='uniform',
    embeddings_regularizer=None, activity_regularizer=None,
    embeddings_constraint=None, mask_zero=False, input_length=None, **kwargs
)
```

embedding layer must be first layer in network in tf.keras

input_dim: vocabulary size (word embedding) or number of classes in GAN example

output_dim: dimension of vector space for the encoding (number of dense nodes)

input_length: for embedding sequences (e.g., sentences)

Outline for Slides

- Generative models
- GANs
  - Sample code
- Conditional GANs
- Style transfer with Cycle-GANs

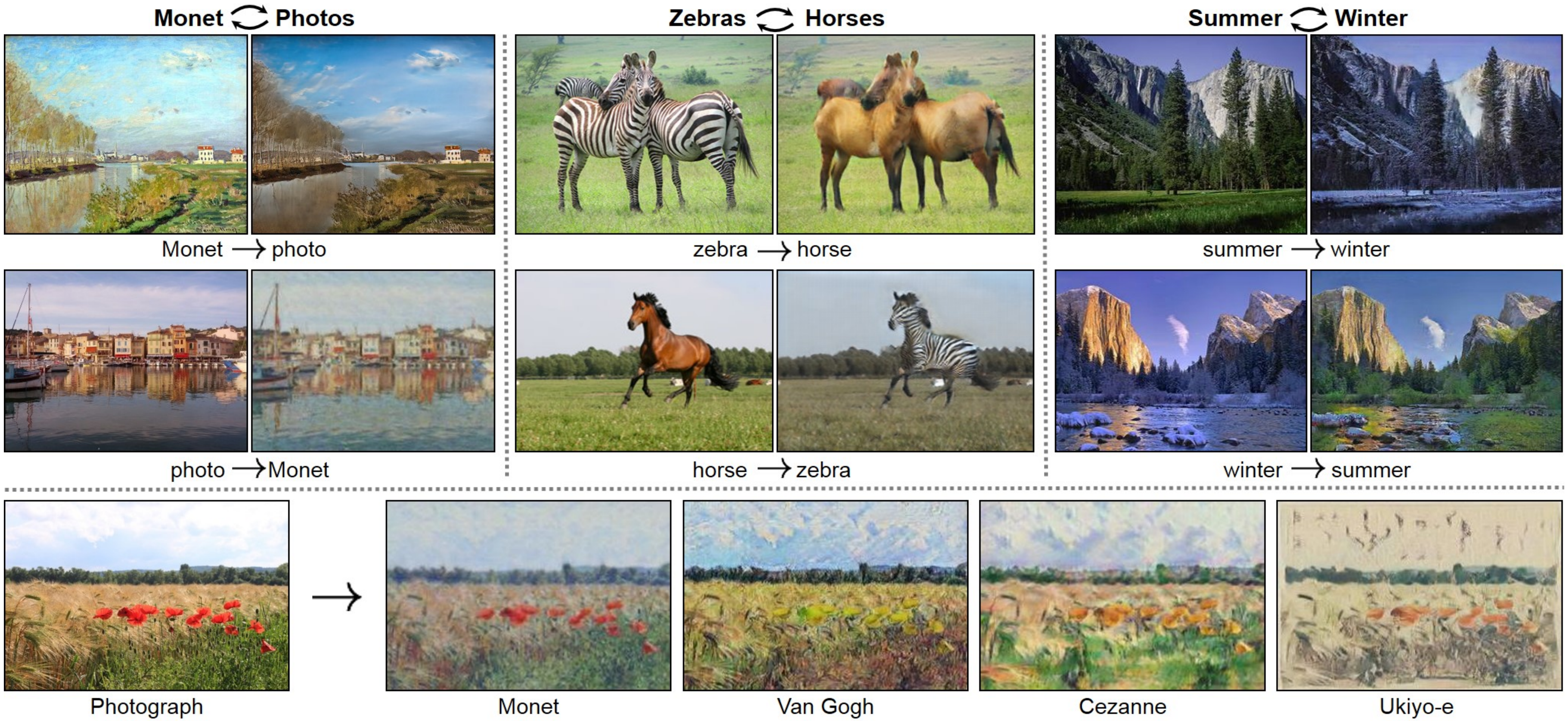
Style Transfer Using Cycle GANs

style transfer is taking the “style” of one image and mapping it to a target image

applying art styles, cartoon style, anime for images

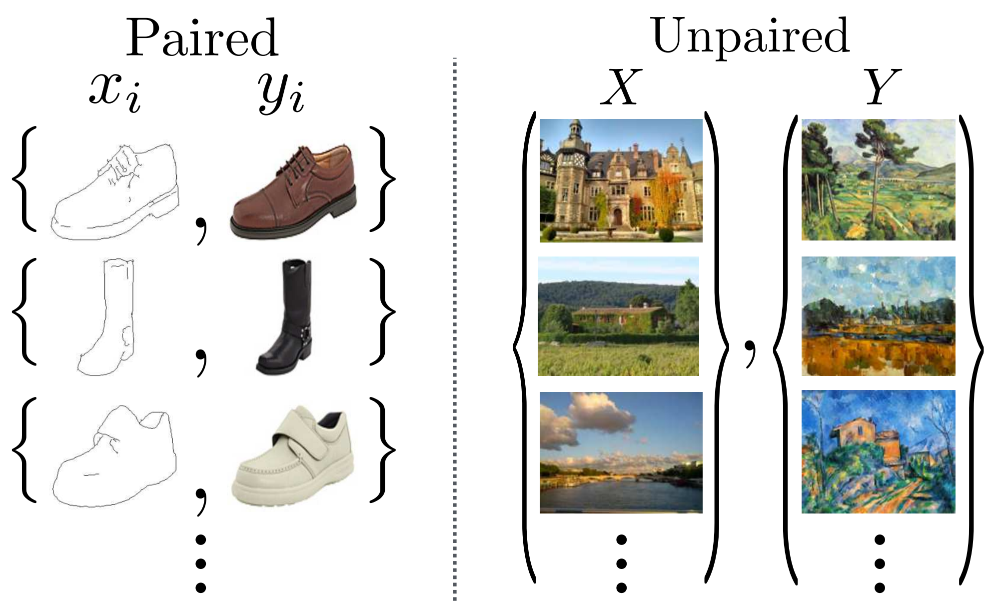
change the voice of a speaker in audio

Style Transfer Using Cycle GANs



Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.

Style Transfer Using Cycle GANs



Paired Data: dataset comprises pairs (original, stylized)

- e.g., set of pictures and cartoon drawings of same people

Unpair Data: dataset comprises separate originals and type examples

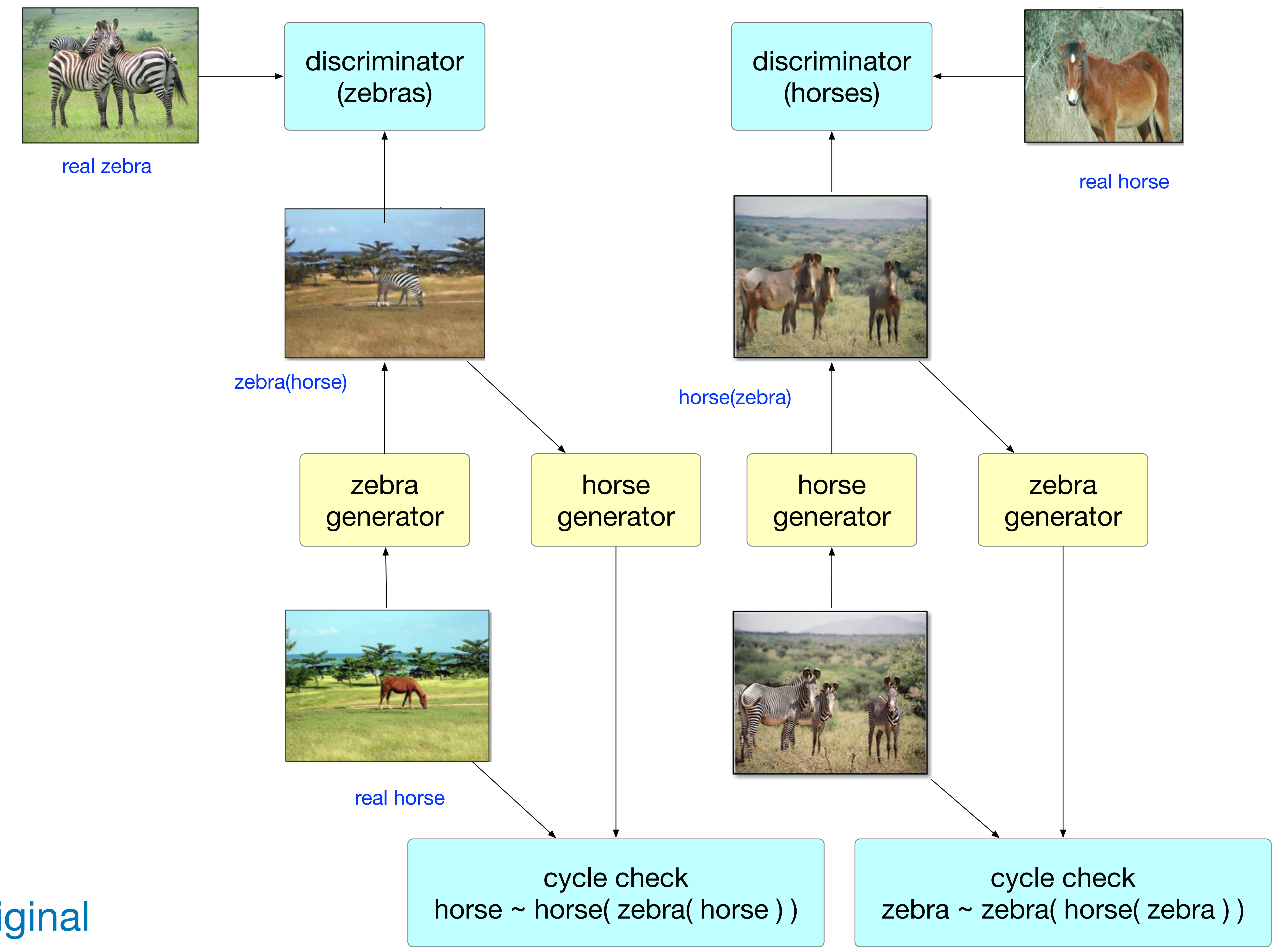
e.g., photos of landscapes and Van Gogh paintings

unpaired data is clearly easier to obtain, so a solution for style transfer for unpaired data is preferred

Cycle GANs perform style transfer on unpaired datasets

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.

Style Transfer Using Cycle GANs



checks:

1. Fake zebra looks like real zebra
2. Fake horse looks like real horse
3. reconstructions are similar to original

Style Transfer Using Cycle GANs

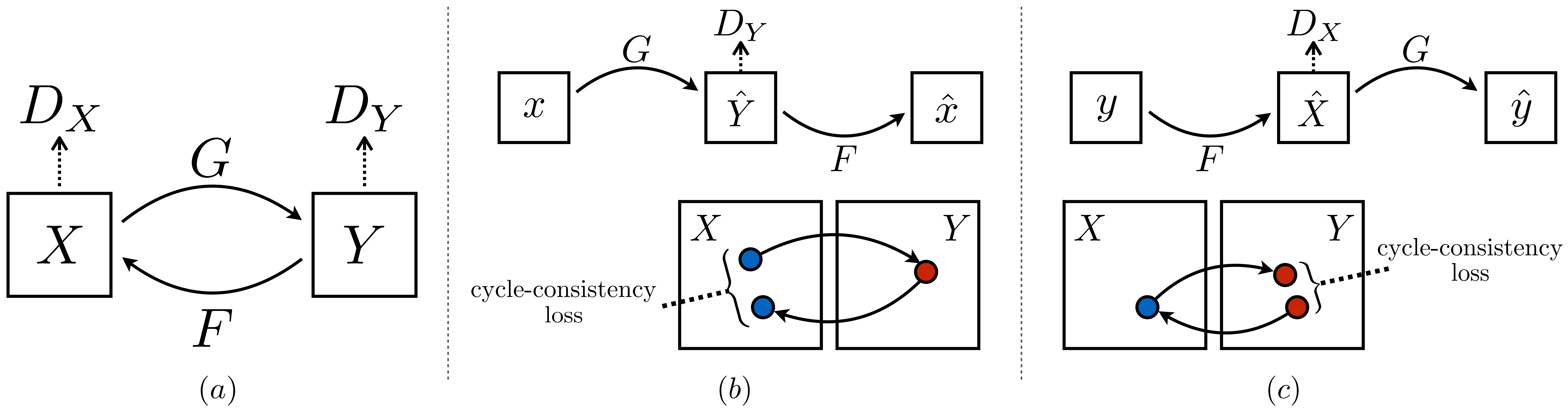


Figure 3: (a) Our model contains two mapping functions $G : X \to Y$ and $F : Y \to X$, and associated adversarial discriminators D_Y and D_X. D_Y encourages G to translate X into outputs indistinguishable from domain Y, and vice versa for D_X, F, and X. To further regularize the mappings, we introduce two "cycle consistency losses" that capture the intuition that if we translate from one domain to the other and back again we should arrive where we started: (b) forward cycle-consistency loss: $x \to G(x) \to F(G(x)) \approx x$, and (c) backward cycle-consistency loss: $y \to F(y) \to G(F(y)) \approx y$

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

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

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

Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.

Style Transfer Using Cycle GANs

https://github.com/junyanz/CycleGAN

example where CycleGAN fails

EE599 Projects Using GANs

Team 1: Mouli Aphale, Shilpa Thomas, Swetha Ann Thomas: Audio Style Transfer

Team 3: Chengxuan Cai, Zixuan Zhang, License PlateImage Enhancement with GANs

Team 4: Mutian Zhu, Dake Chen, Producing a classical piano music with GANs

Team 11: Vineeth Ellore, Ashwin T Ravi, Karkala Shashank Hegde: Emotion transfer on images and spectrogram of speech

Team 13: Ashwin Shetty, Pruthvi Gollahalli Niranjana: Creating Cartoon (artistic) Styled Images using GANs

Team 15: Haojing Hu, Zheng Wen: Object transfiguration with attention-aided GANs

Team 21: Yang Tao, Lingkai Kong: Image Style Transfer

Team 27: Jiahui Zhang, Zhuoran Liu: Single Image Super Resolution Generation via GANs

Team 35: Fan Yang, Shuna Ye, Yelei Zhang: 3D Printing Designer from 2D Images