INTRO GENERATIVE ADVERSARIAL NETWORK (GAN)

– Matee Vadrukchid –

Part 1: Autoencoders

Part 1: Autoencoders

- Auto = self
- Encode = convert into a different form

Autoencoder = a system that teaches itself how to encode information

It is a model that teaches itself how to encode information

Part 1: Autoencoders

- An unsupervised learning technique that is used as a data representation

- The idea is to use CNN to act as data compression/data encoding by introducing a bottleneck layer

- We must have encoding layers and decoding layers

Part 1: Autoencoders

Loss function

Binary Cross Entropy/Log Loss

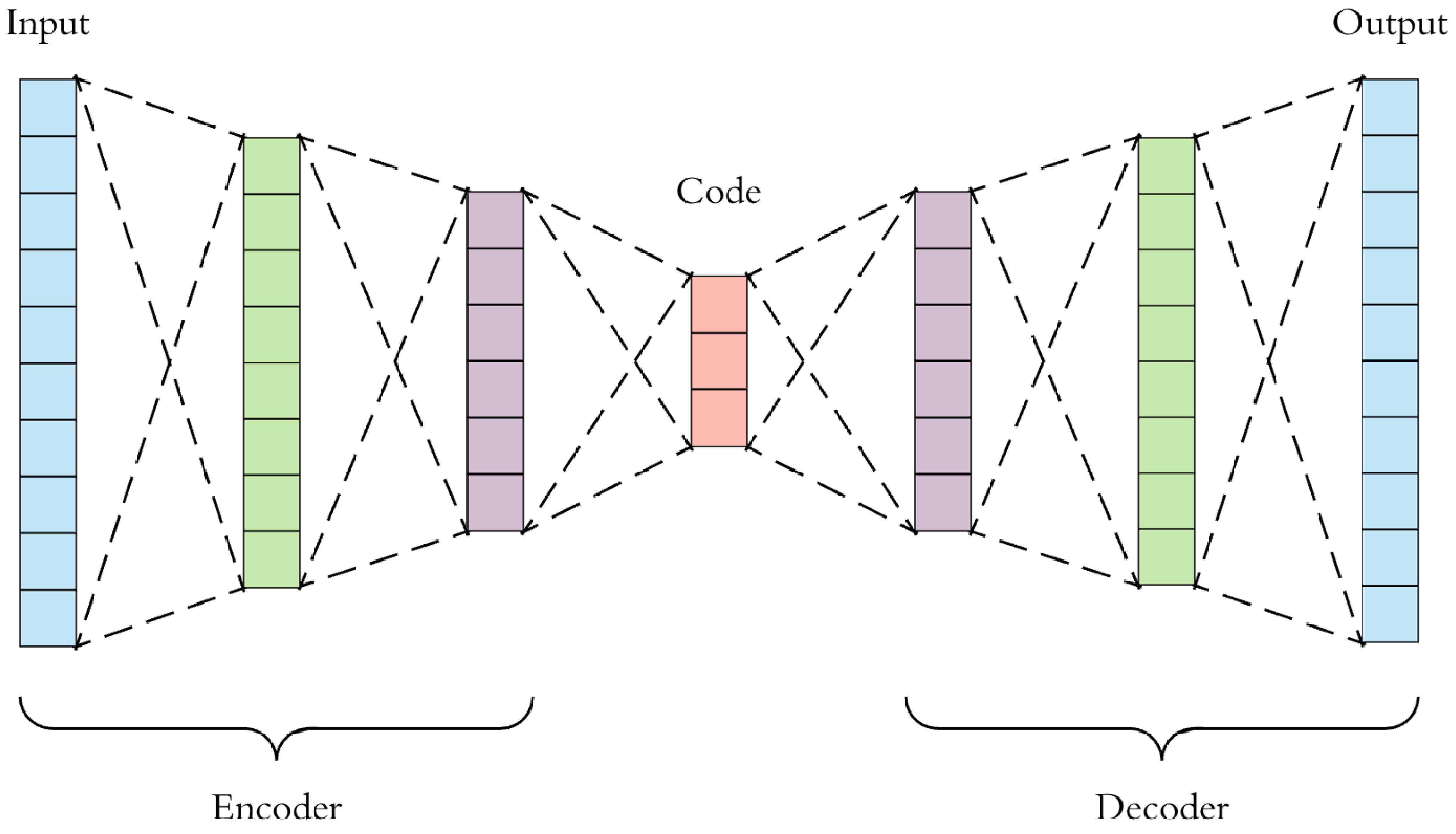
$$BCE = -\frac{1}{n}\sum_{j=1}^{n}\sum_{i=1}^{c}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

MSE
Loss

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

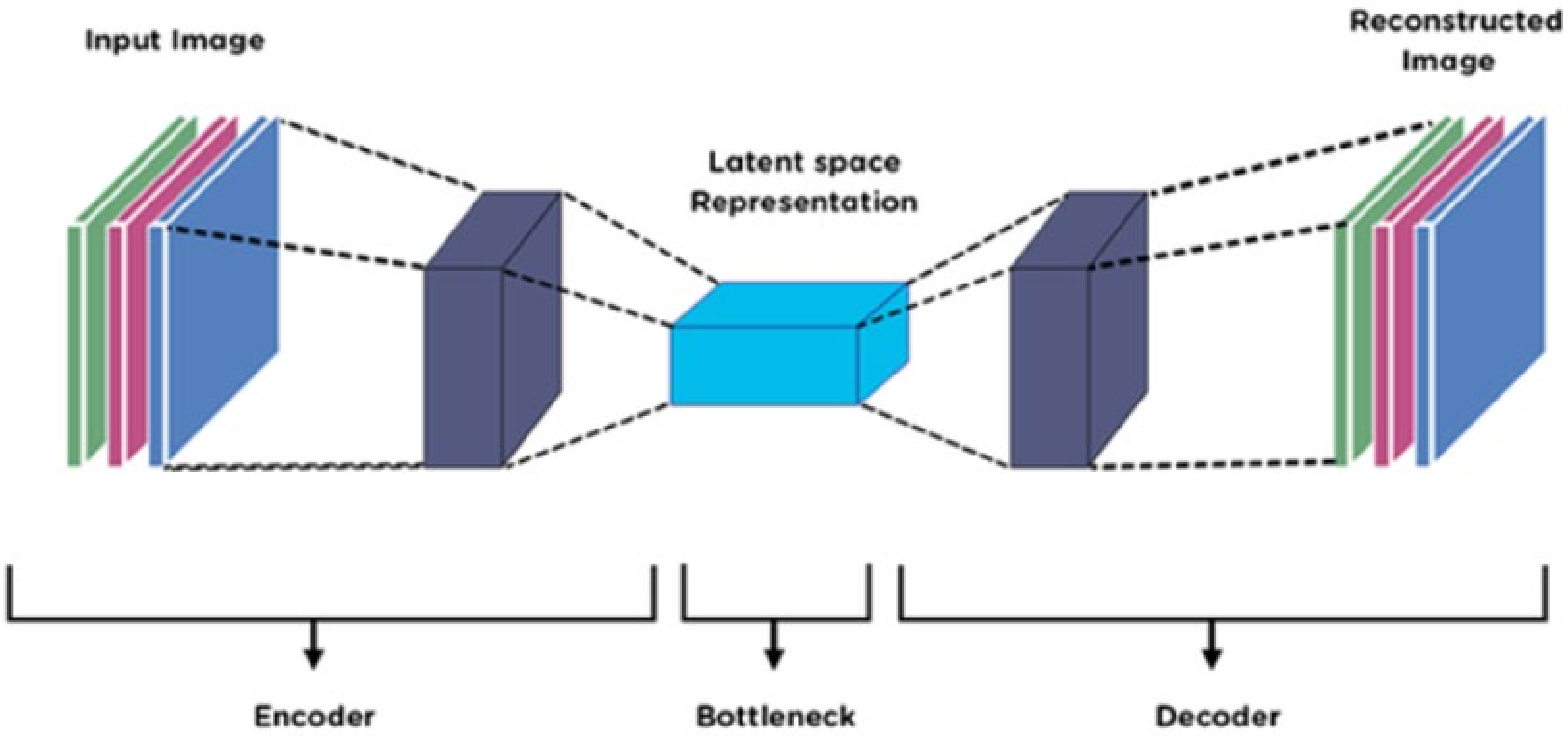
Part 1: Autoencoders

Example



Part 1: Autoencoders

CNN Autoencoder

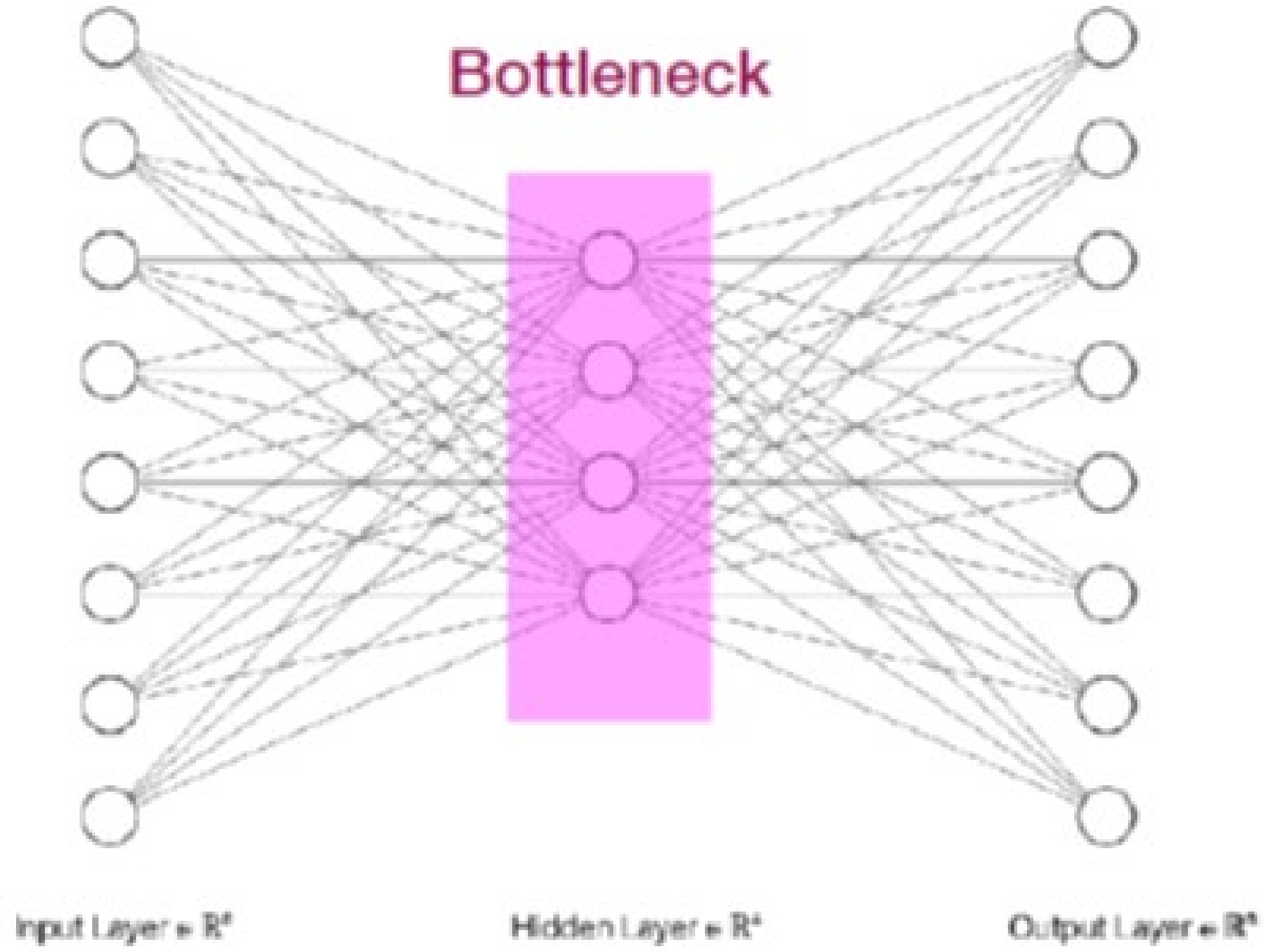


Autoencoder applications

- Denoising
- Fix Image Inpainting
- Information Retrieval
- Anomaly Detection

Part 1: Autoencoders

- Introduce a bottleneck layer to compress the data

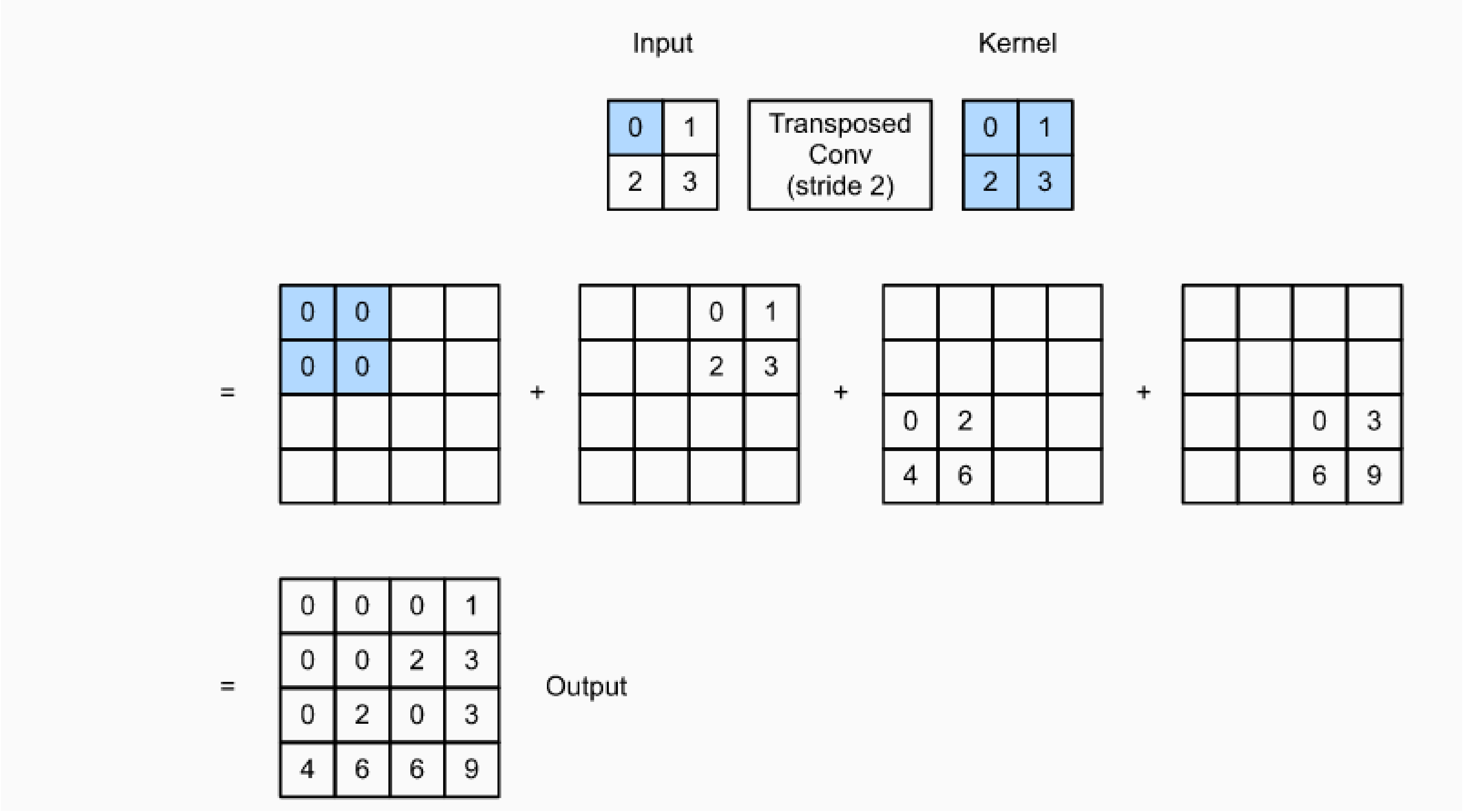


Terminology

- Convolution with stride >= 2 (downsampling)
- Transpose convolution (upsampling)

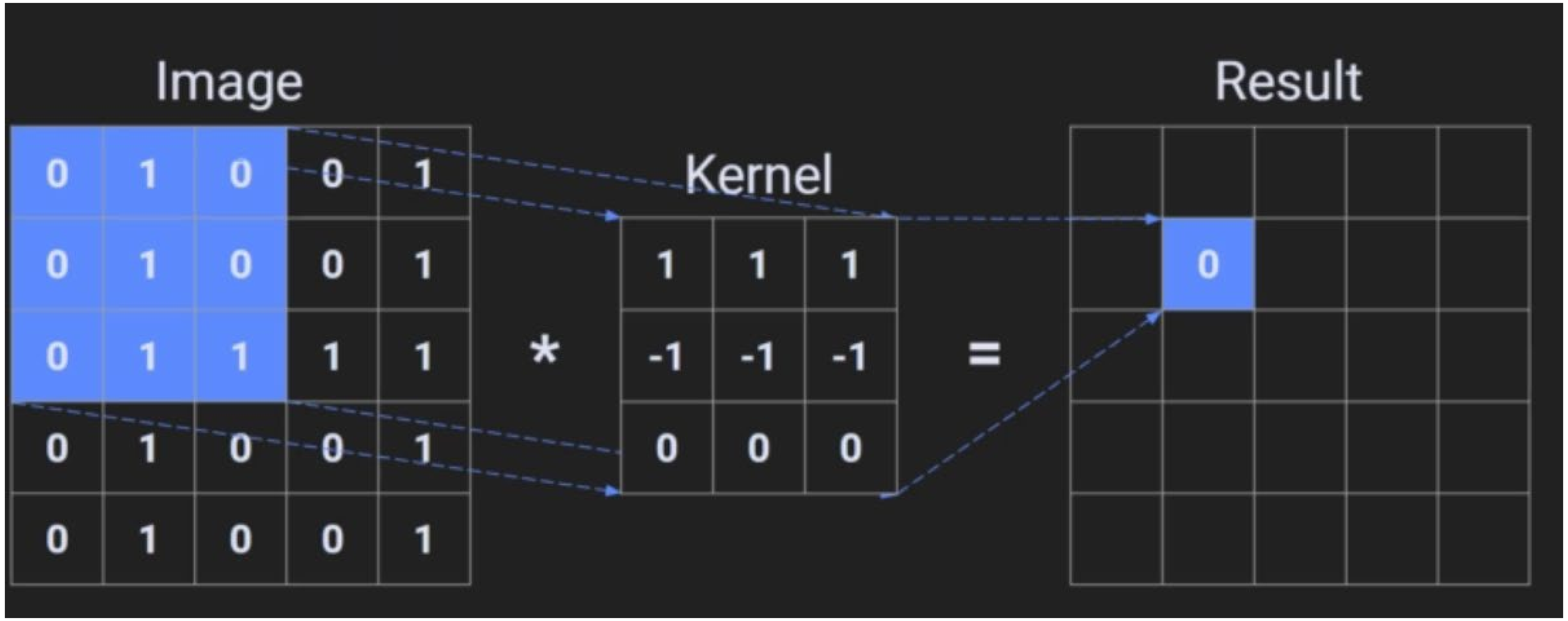
Part 1: Autoencoders

Convolution/Transposed Convolution



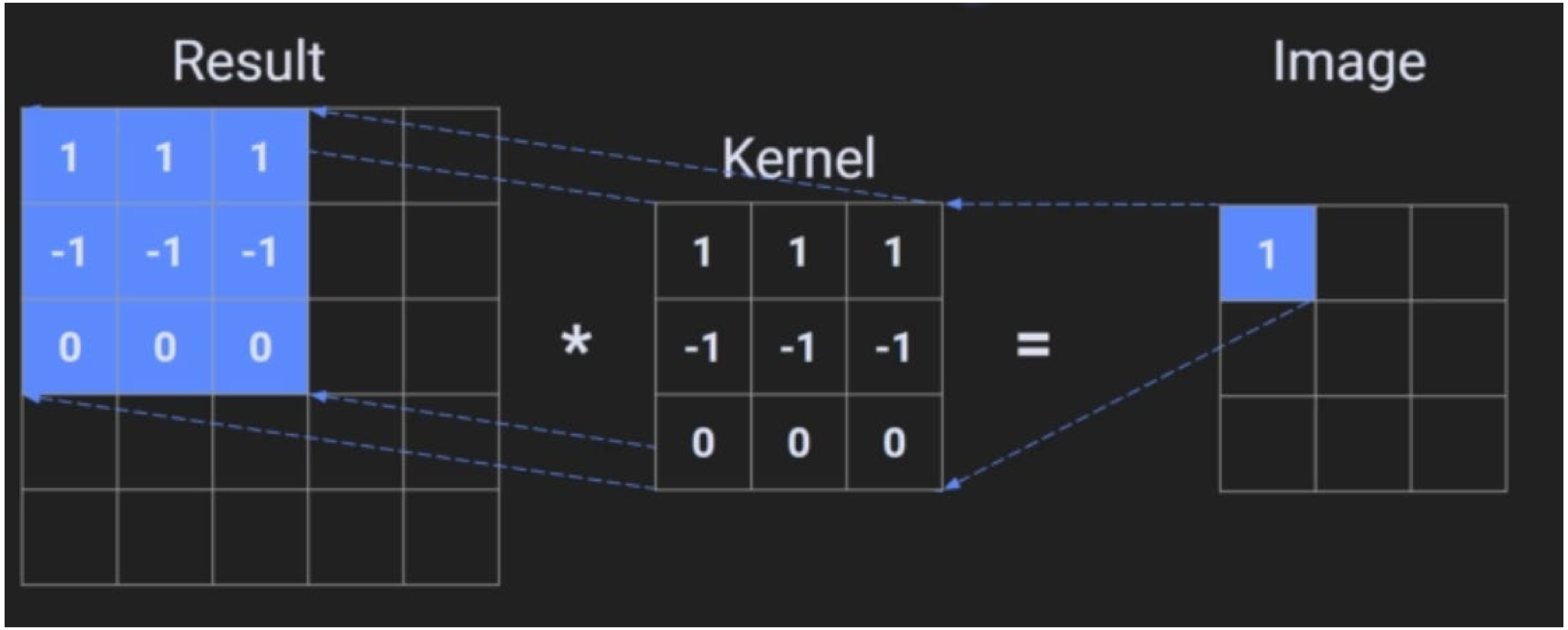
Part 1: Autoencoders

Convolution

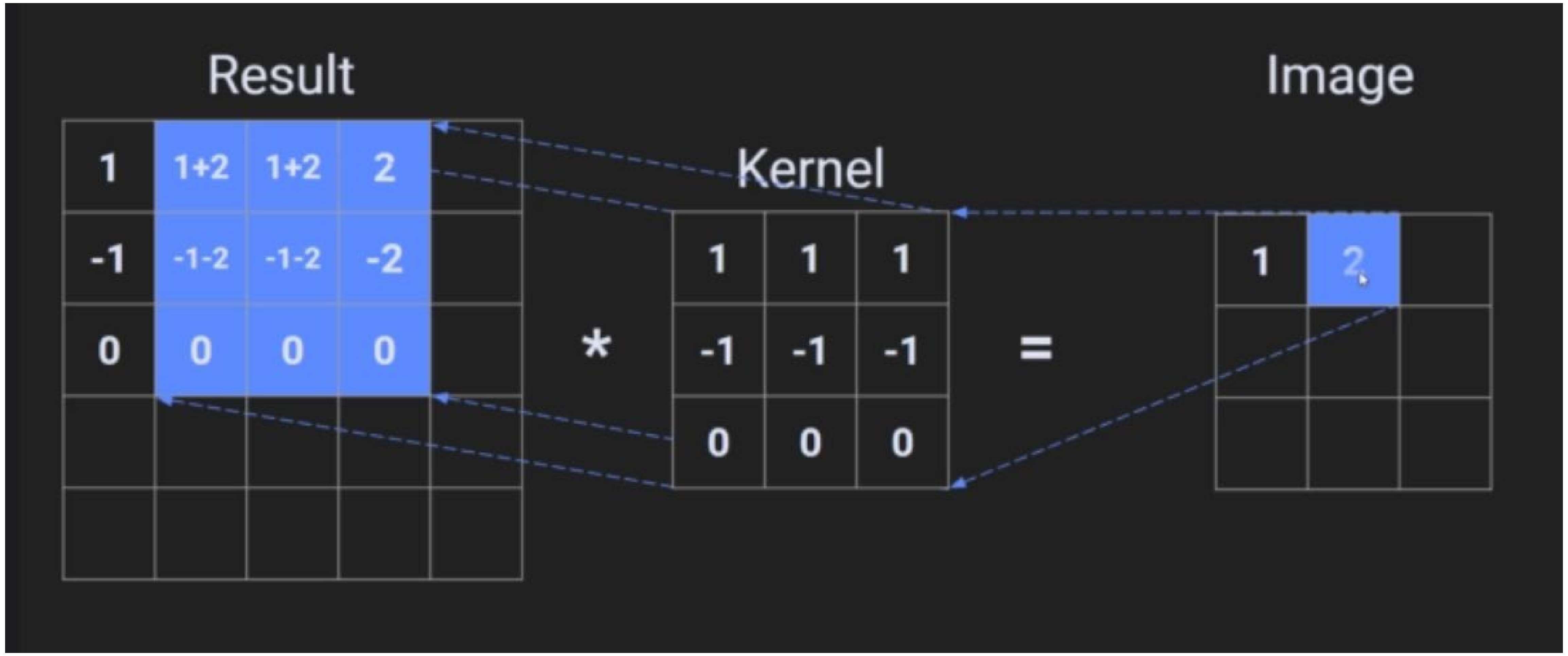


Part 1: Autoencoders

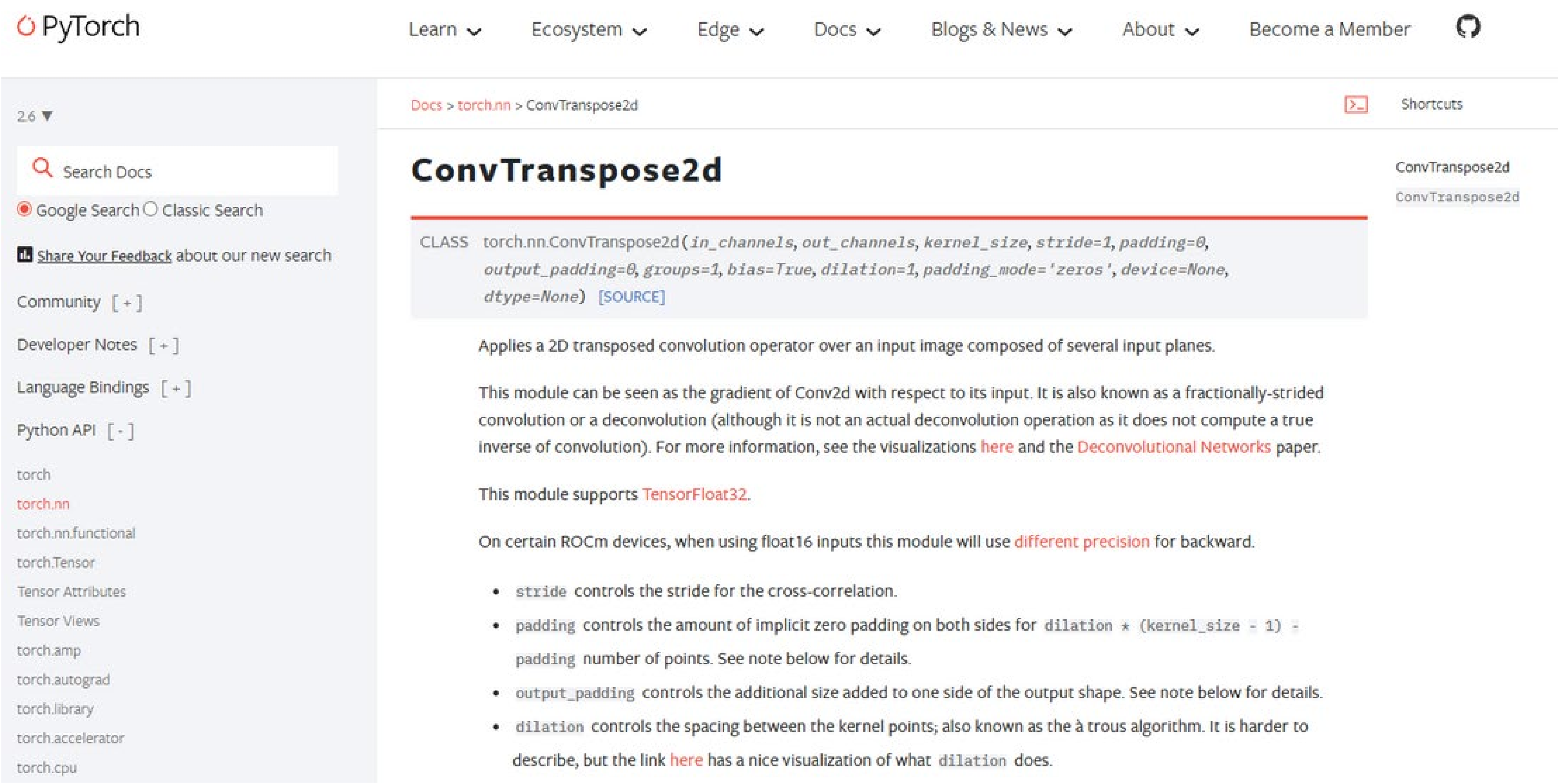
Transpose Convolution



Next Step



Part 1: Autoencoders



PyTorch Learn Ecosystem Edge Docs Blogs & News About Become a Member

2.6

Search Docs

Google Search Classic Search

Share Your Feedback about our new search

Community [+]

Developer Notes [+]

Language Bindings [+]

Python API [-]

torch

torch.nn

torch.nn.functional

torch.Tensor

Tensor Attributes

Tensor Views

torch.amp

torch.autograd

torch.library

torch.accelerator

torch.cpu

Docs > torch.nn > ConvTranspose2d

Shortcuts

ConvTranspose2d

ConvTranspose2d

ConvTranspose2d

CLASS torch.nn.ConvTranspose2d(*in_channels*, *out_channels*, *kernel_size*, *stride=1*, *padding=0*, *output_padding=0*, *groups=1*, *bias=True*, *dilation=1*, *padding_mode='zeros'*, *device=None*, *dtype=None*) [SOURCE]

Applies a 2D transposed convolution operator over an input image composed of several input planes.

This module can be seen as the gradient of Conv2d with respect to its input. It is also known as a fractionally-strided convolution or a deconvolution (although it is not an actual deconvolution operation as it does not compute a true inverse of convolution). For more information, see the visualizations here and the Deconvolutional Networks paper.

This module supports TensorFloat32.

On certain ROCm devices, when using float16 inputs this module will use different precision for backward.

- `stride` controls the stride for the cross-correlation.
- `padding` controls the amount of implicit zero padding on both sides for `dilation * (kernel_size - 1) - padding` number of points. See note below for details.
- `output_padding` controls the additional size added to one side of the output shape. See note below for details.
- `dilation` controls the spacing between the kernel points; also known as the à trous algorithm. It is harder to describe, but the link here has a nice visualization of what `dilation` does.

$$Hout=(Hin-1)\times stride[0]-2\times padding[0]+dilation[0]\times (kernel\_size[0]-1)+output\_padding[0]+1$$

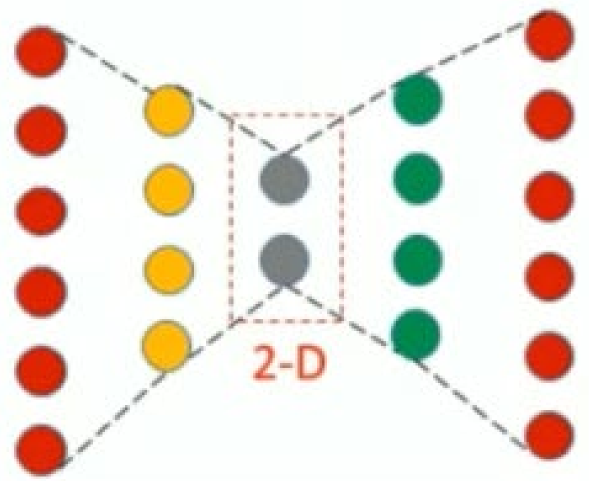
$$Wout=(Win-1)\times stride[1]-2\times padding[1]+dilation[1]\times (kernel\_size[1]-1)+output\_padding[1]+1$$

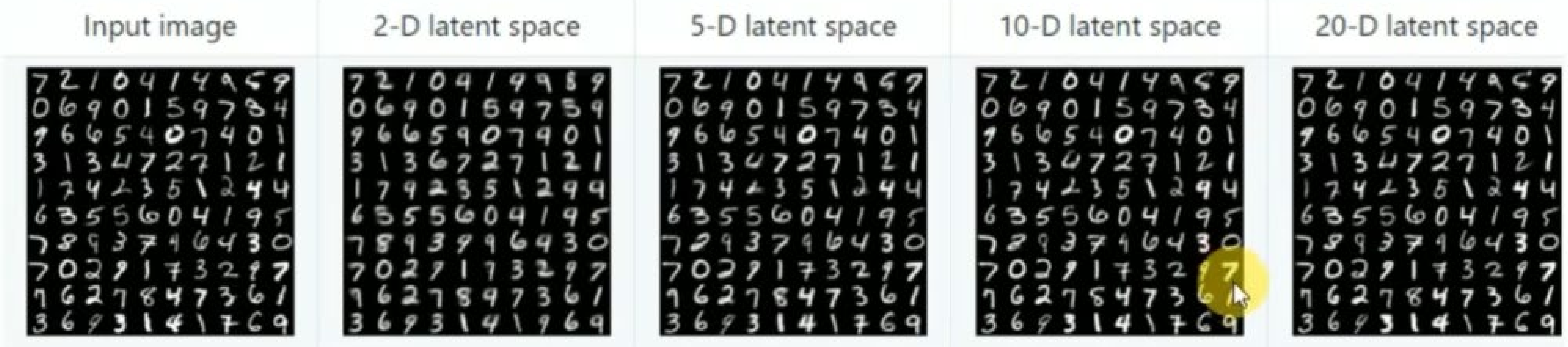
Link: https://pytorch.org/docs/stable/generated/torch.nn.ConvTranspose2d.html

Part 1: Autoencoders

The number of latent variables

- The number of latent variables (the number of output neurons/dimension) of the last encoder layer/the compressing layer (the dimensionality of the compressed representation) matters. The more, the better:





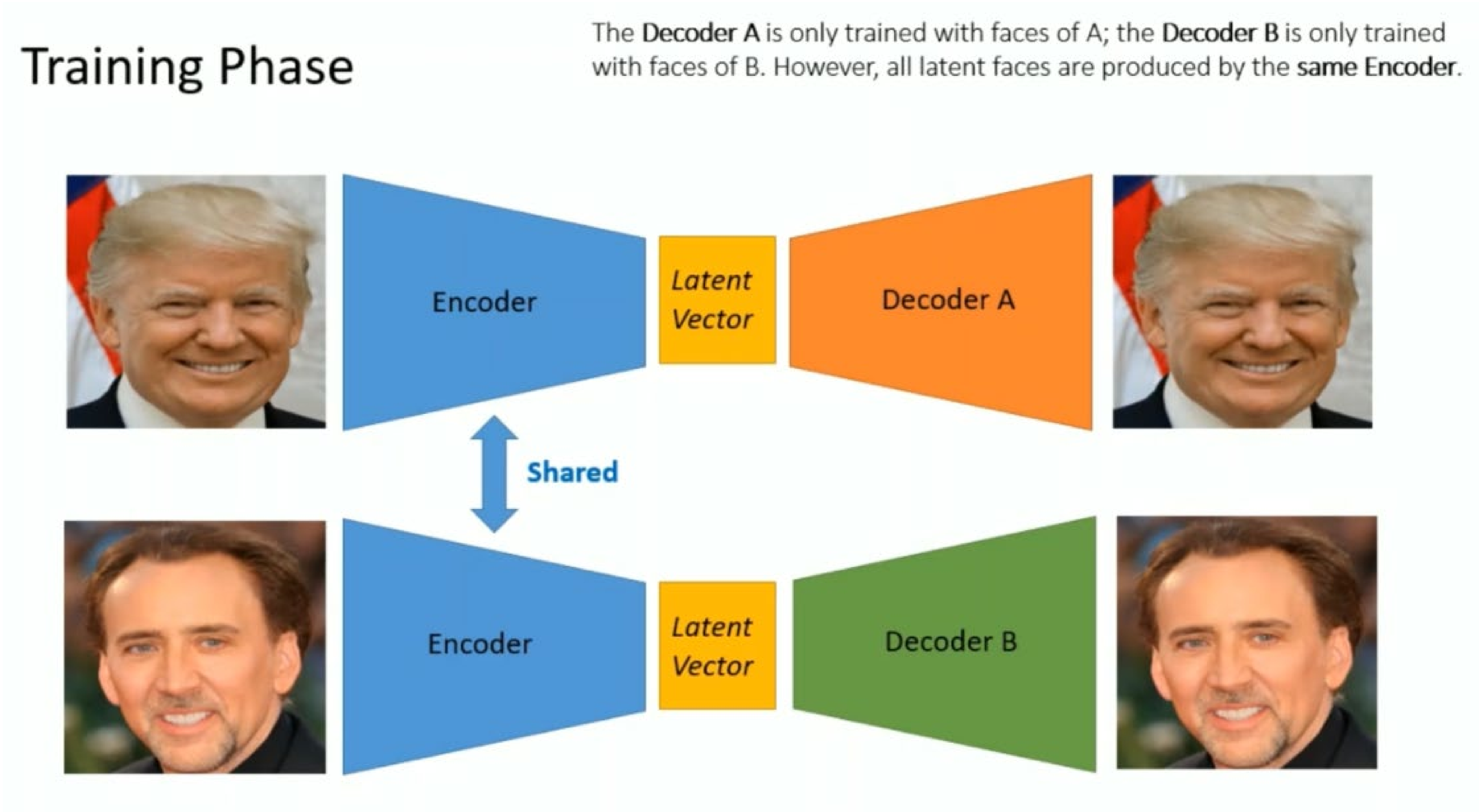
GitHub Implementation: https://github.com/dragen1860/pytorch-mnist-vae

Part 1: Autoencoders

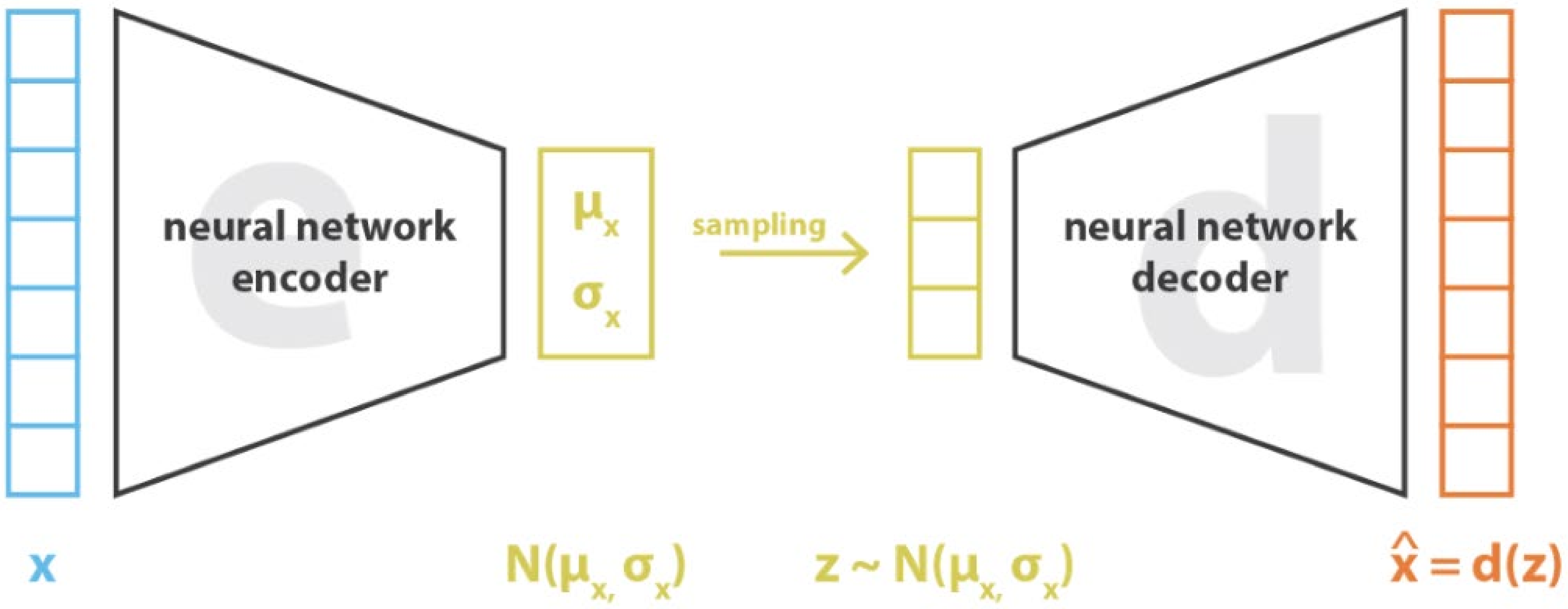
Deep Fake with AutoEncoder

Donald Trump → Mr. Bean *Home Alone → Home Stallone*

Part 1: Autoencoders



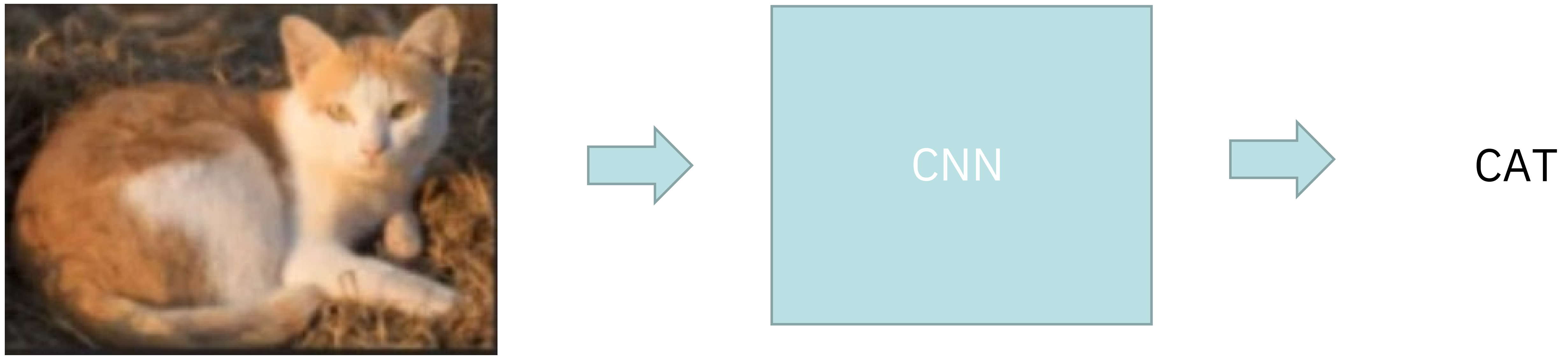
Part 1: Autoencoders



Part 2: Intro GAN

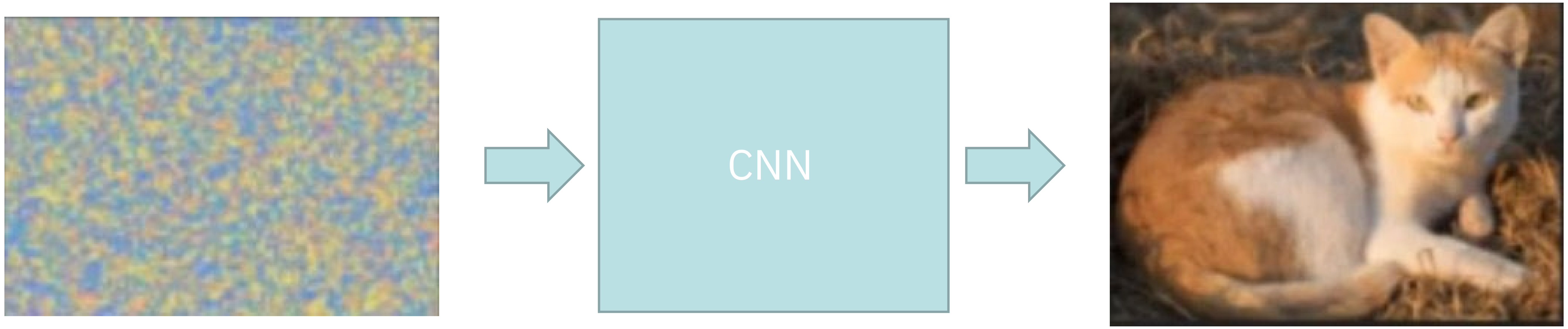
Discriminative models

- CNN classify images



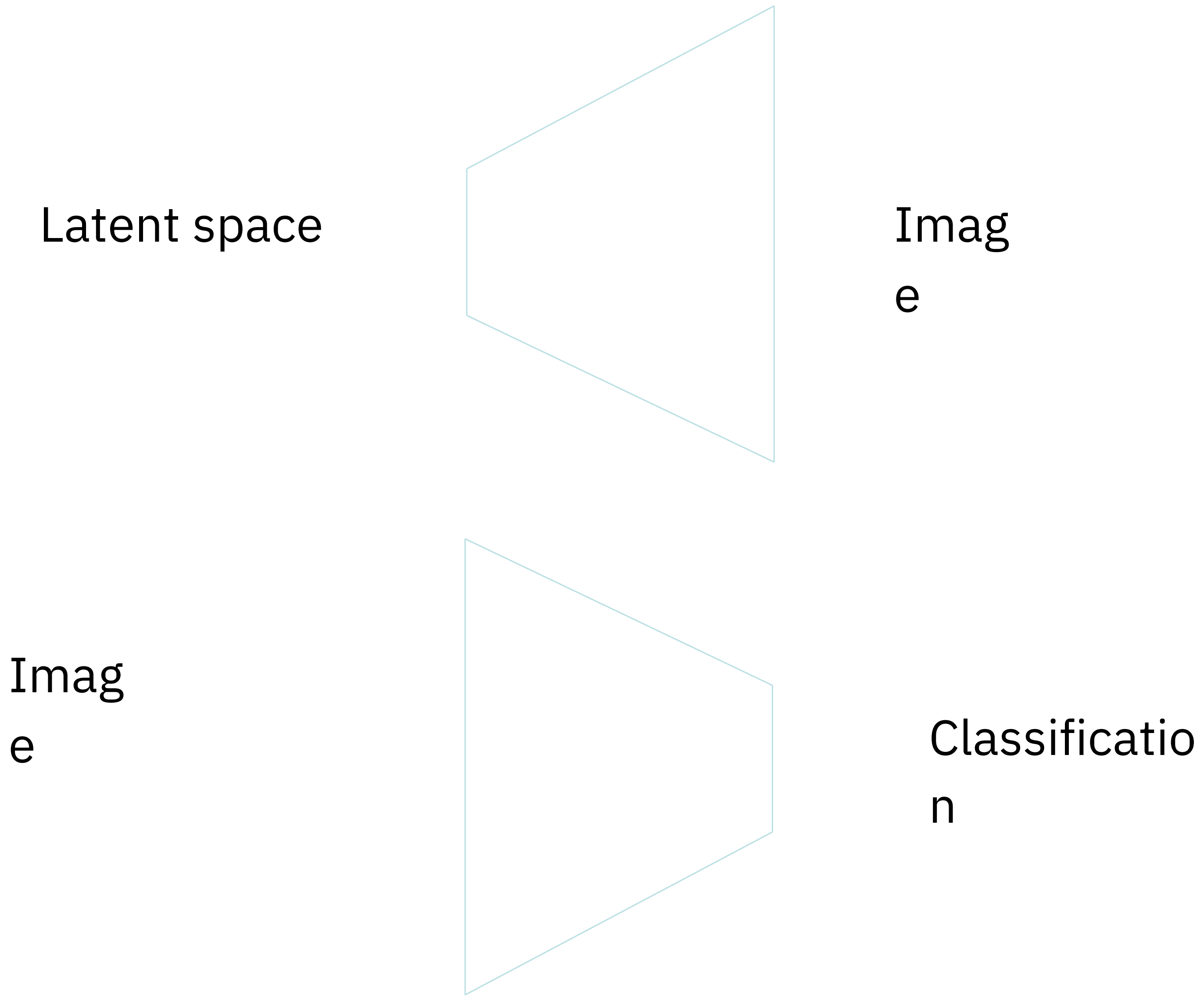
Generative models

- Create new samples

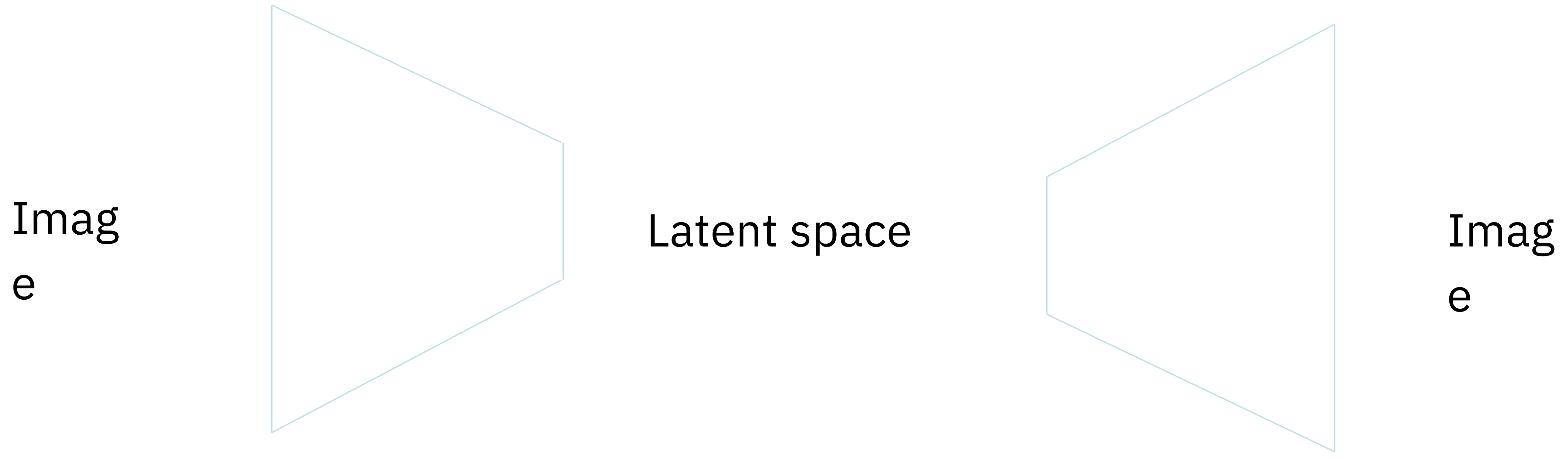


Part 2: Intro GAN

More details



AutoEncoder



GAN

- It is introduced by Ian Goodfellow et al. in 2014
- It is the NN model that generates data from an existing distribution of samples thru loop approach

May 25, 2018, 08:15am EDT

The Best Tech Innovations Of The Last Three Years

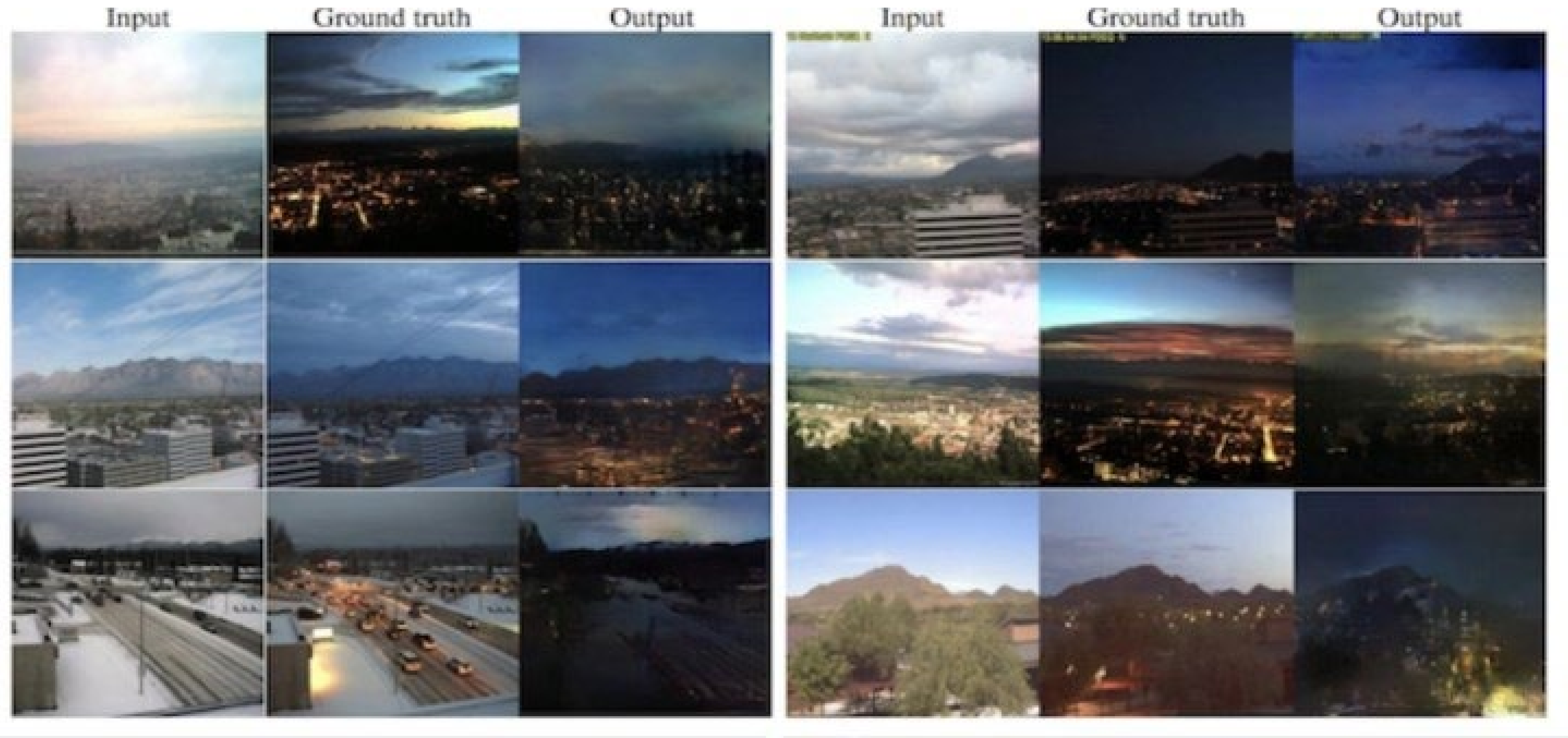
Forbes Technology Council COUNCIL POST | Paid Program
Innovation

POST WRITTEN BY
Forbes Technology Council

Successful CIOs, CTOs & executives from Forbes Technology Council offer firsthand insights on tech & business.

Image-to-Image translation

- Night timefrom Day time image



Super Resolution

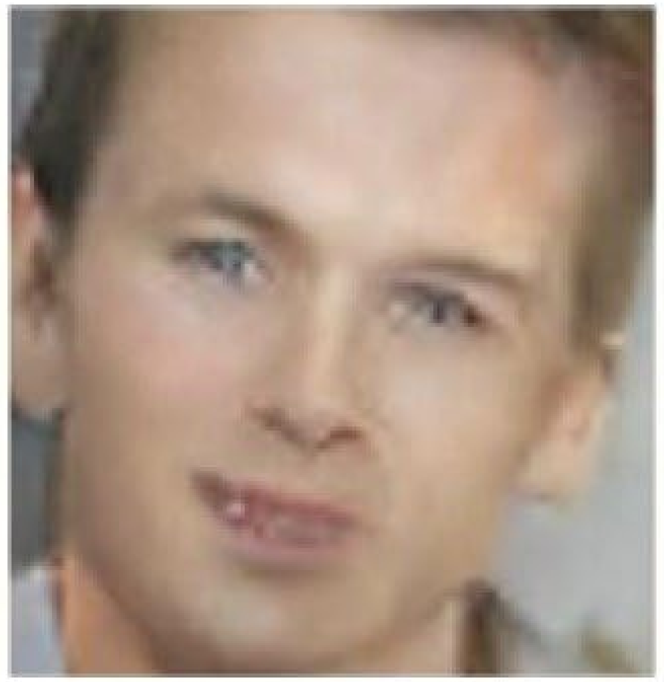
Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

Part 2: Intro GAN

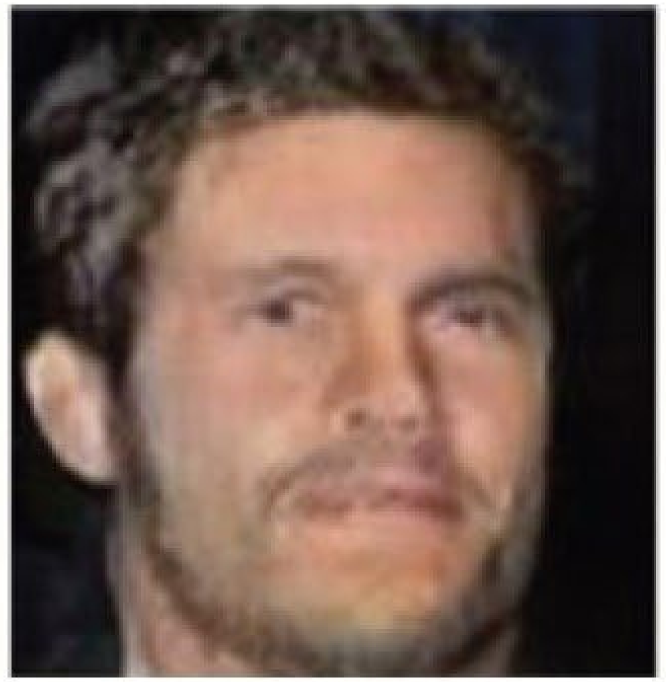
ARCANE GAN

Generation

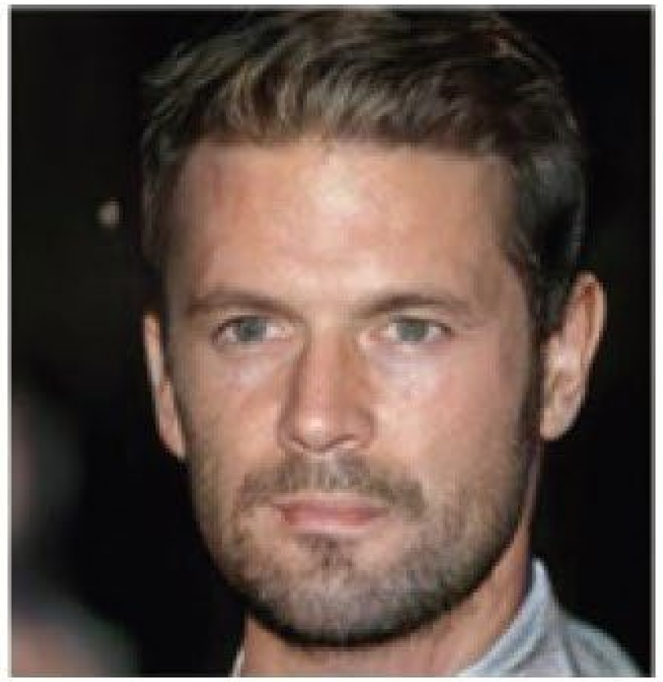
2014



2015



2016



2017

Issues with GANs

- Sensitive to model architecture and features
- Training time
- Evaluating GANs is mostly quantitative
- Difficult to bring research lab to real-world applications
- Mode collapse: Generator only output good knows results (not enough varieties)

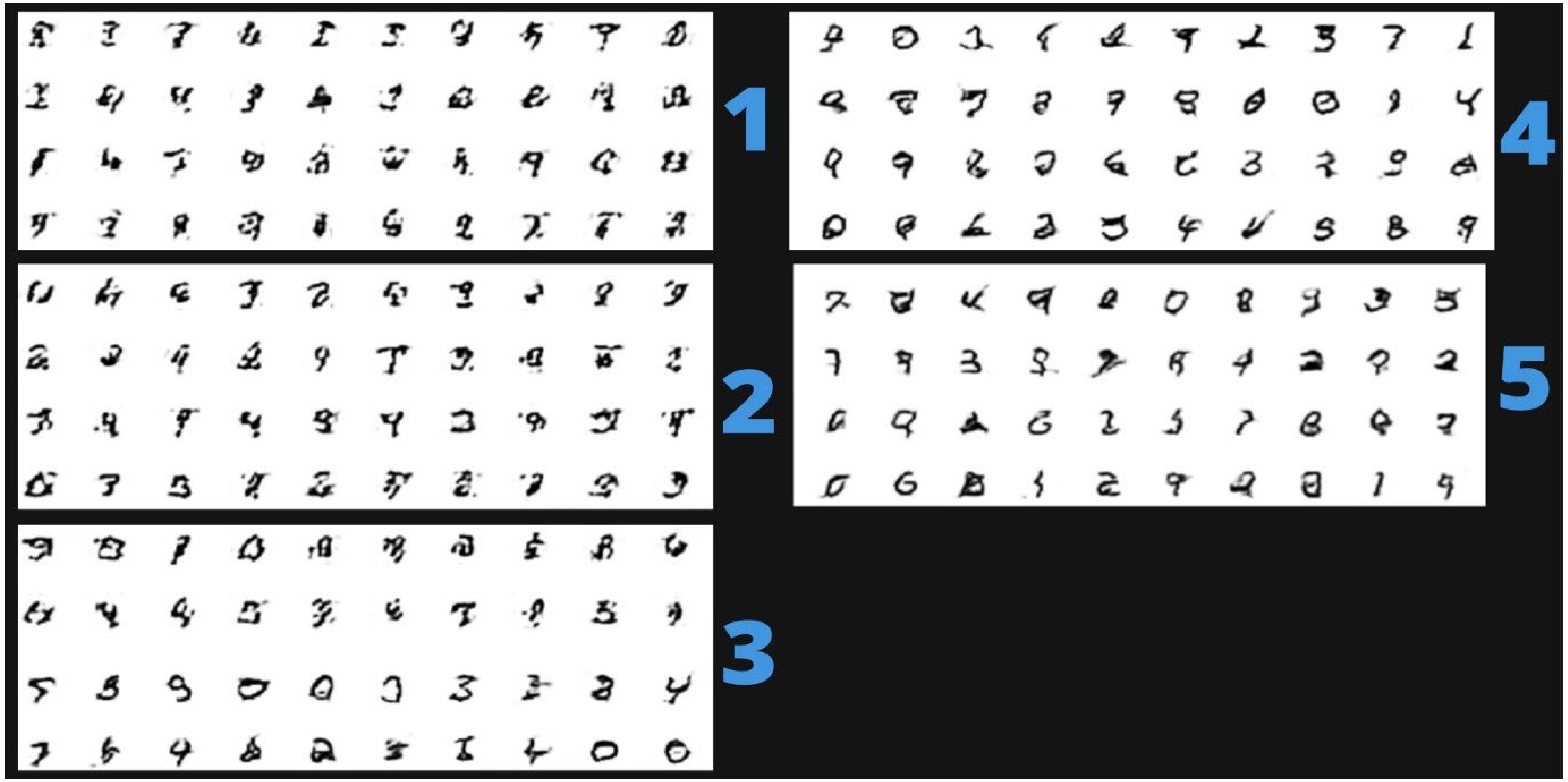
Part 2: Intro GAN

Example

Training Process

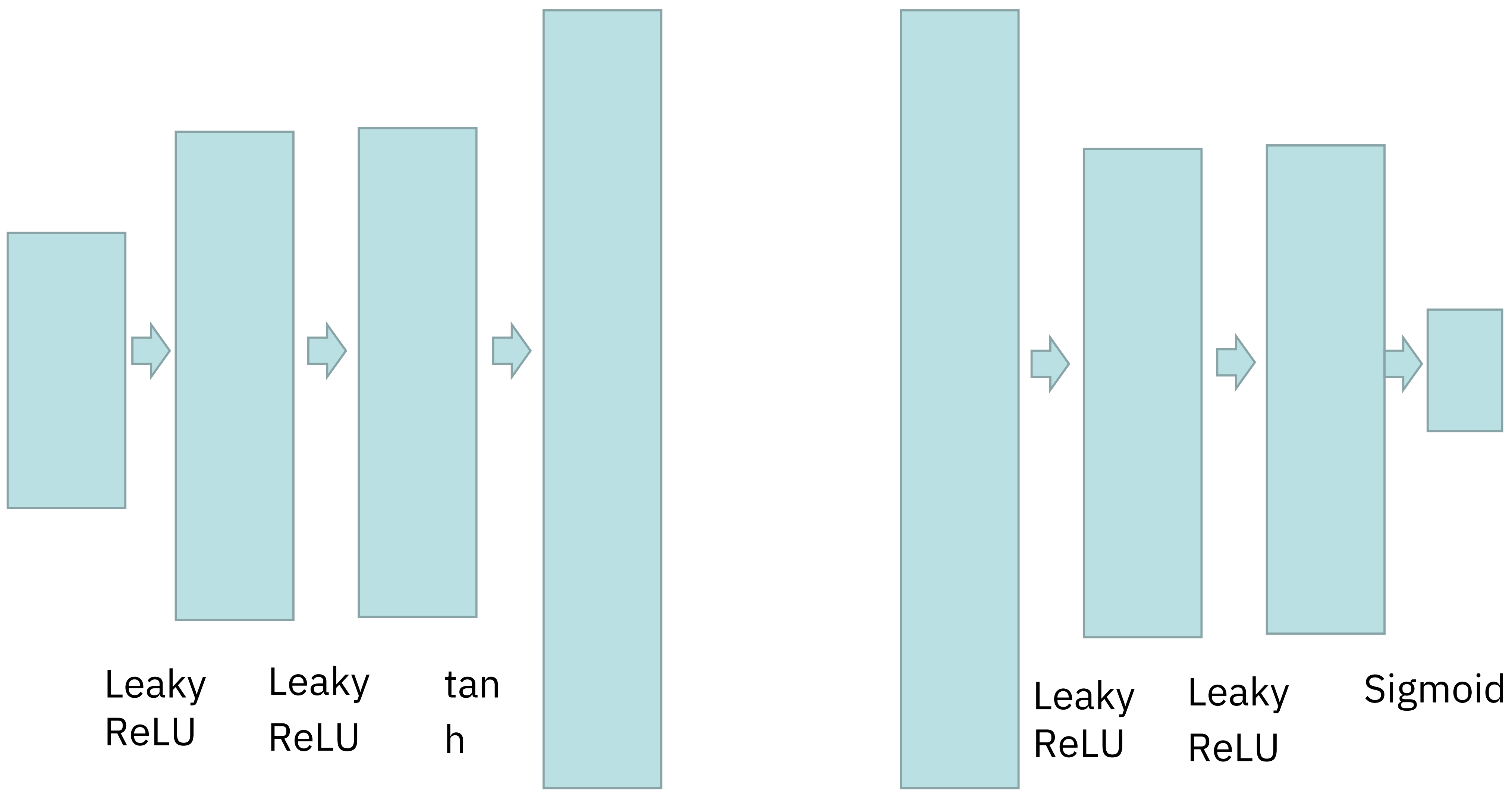
- Randomly generate a noisy vector
- Input this into generator network to create samples
- Take some sample and mix with some real data
- Train discriminator to classify mixed dataset
- Discriminator predict 0 (real) or 1 (fake)

Generated Sample Data



Part 2: Intro GAN

FCN GAN



Part 2: Intro GAN

Loss Function

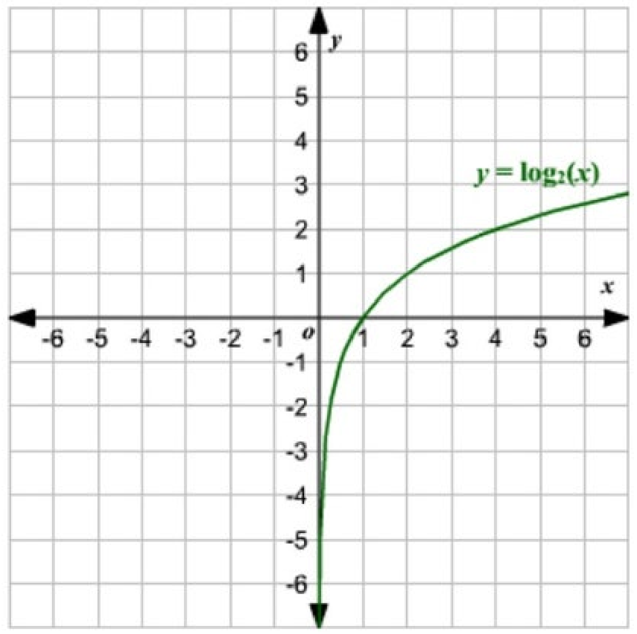
•Discriminator loss:

we provide ground truth whether the data comes from real or generate images

•Generator loss:

we use discriminator feedback on fake images to train

generator $J^D = E_{x \sim p_r} \log[D(x)] + E_{z \sim p_g} \log[1 - D(G(z))]$

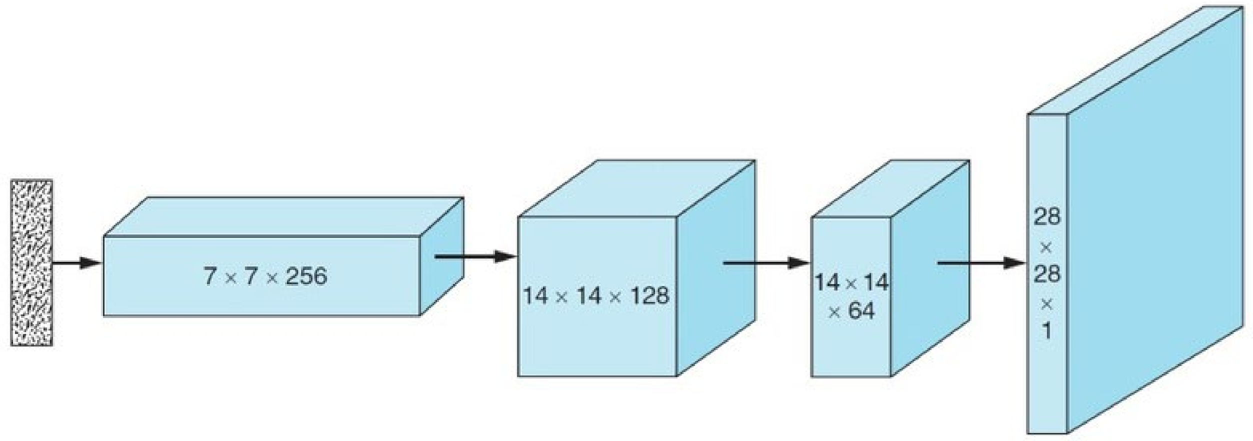


Part 2: Intro GAN

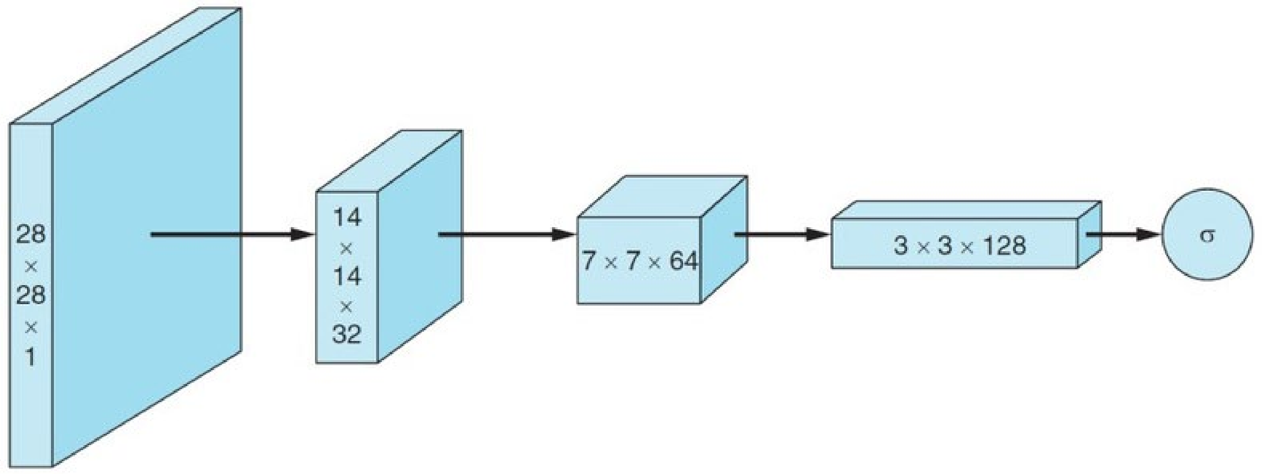
Results on MNIST

Part 2: Intro GAN

DCGAN



Generator



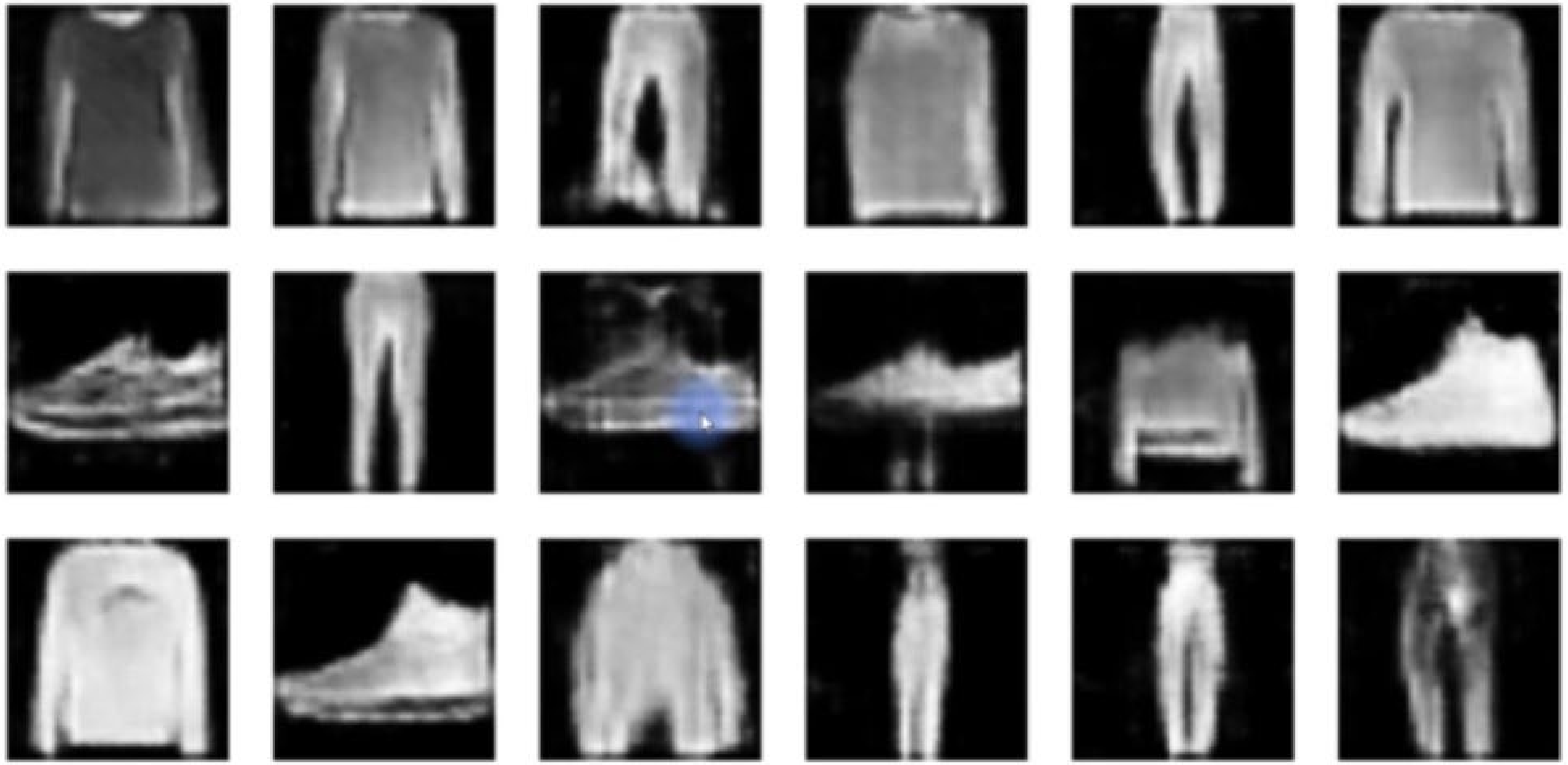
Discriminato
r

Part 2: Intro GAN

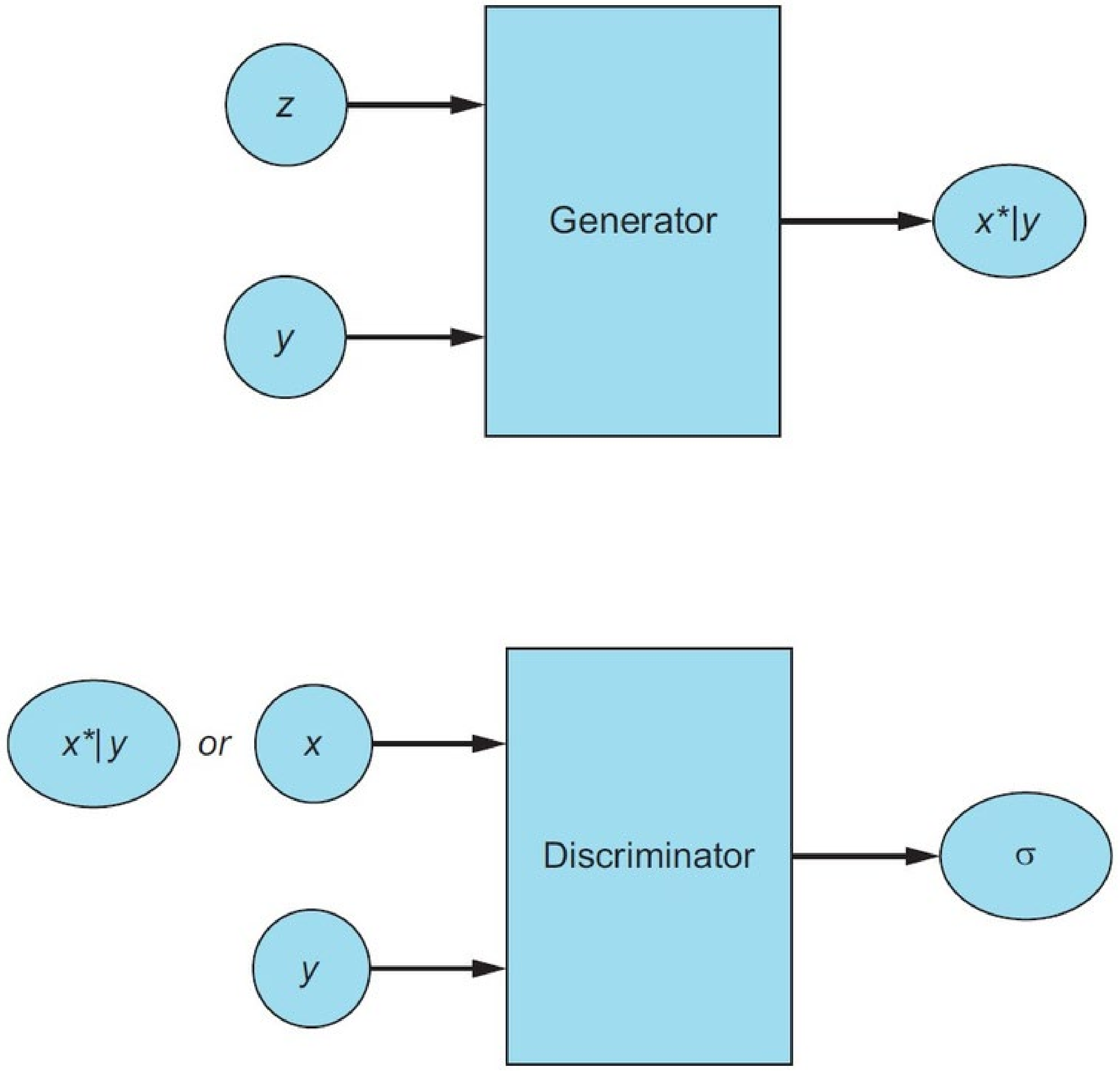
DCGAN on MNIST

Part 2: Intro GAN

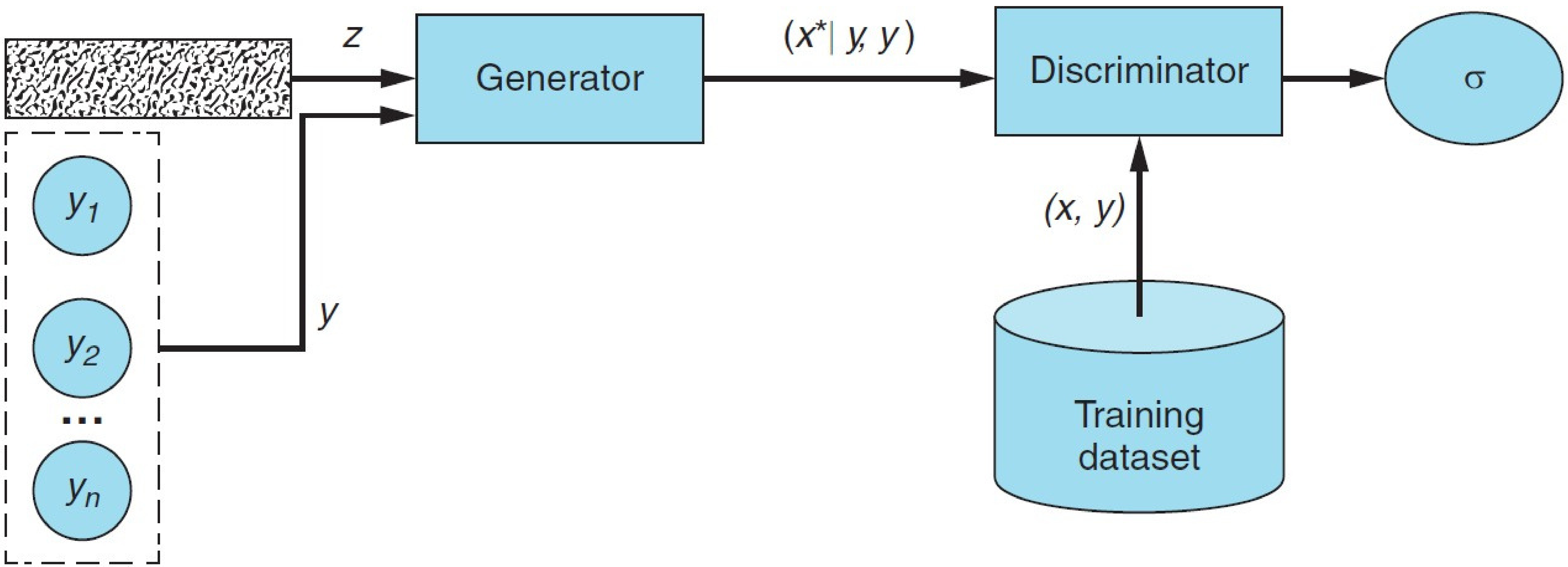
DCGAN on FMNIST



Conditional GAN

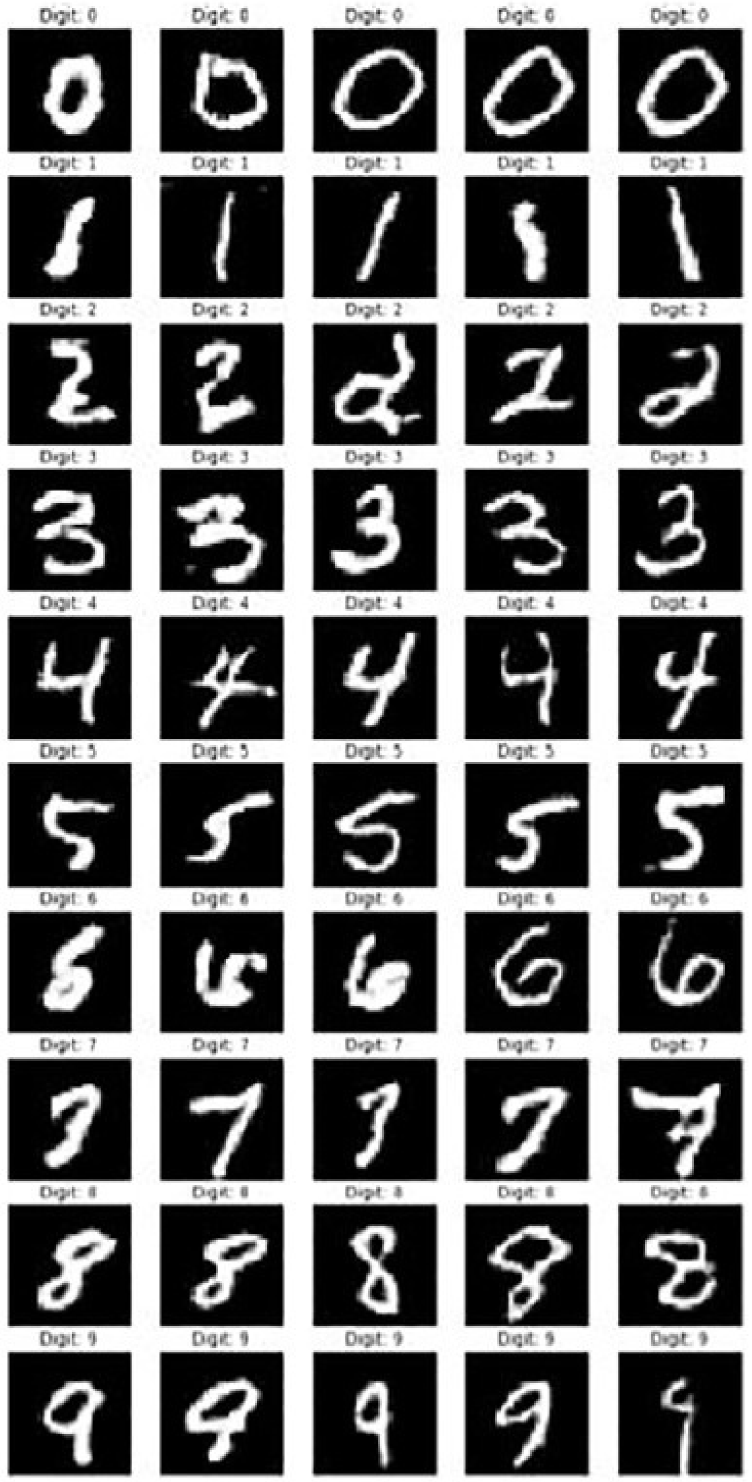


CGAN Architecture

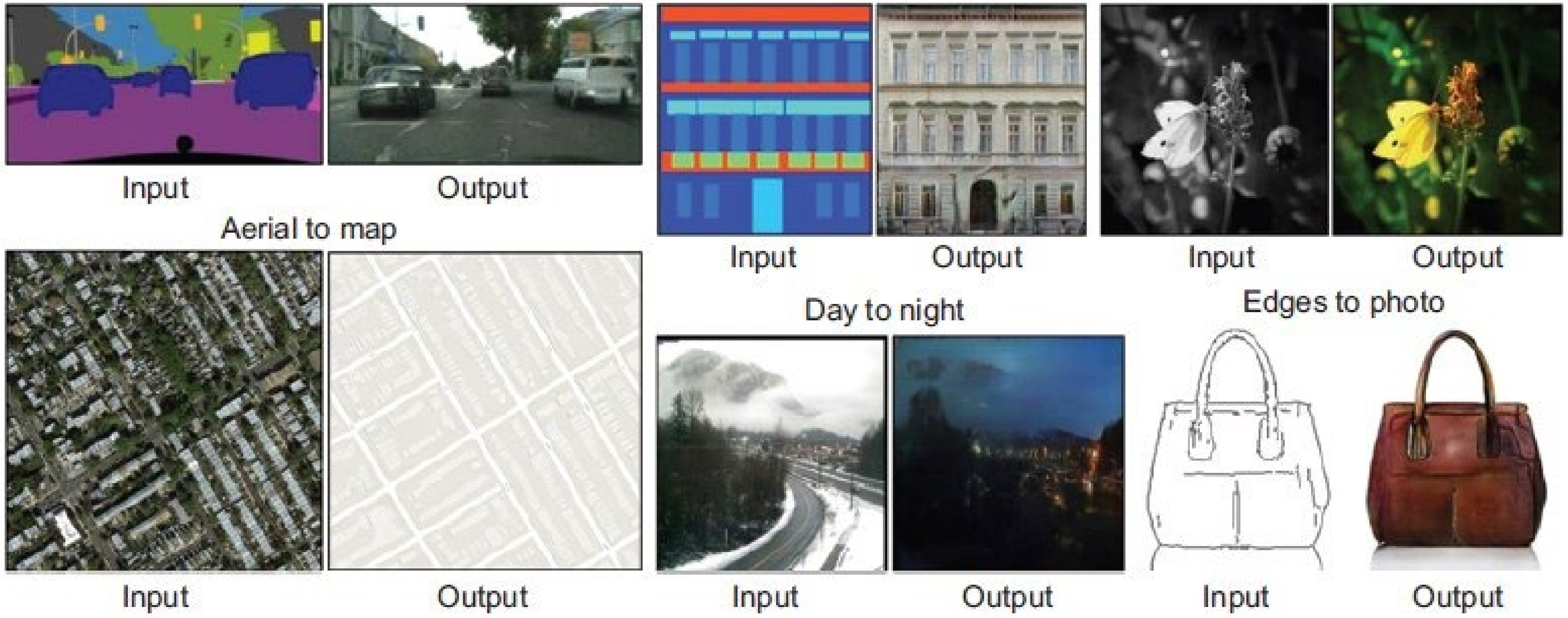


Part 2: Intro GAN

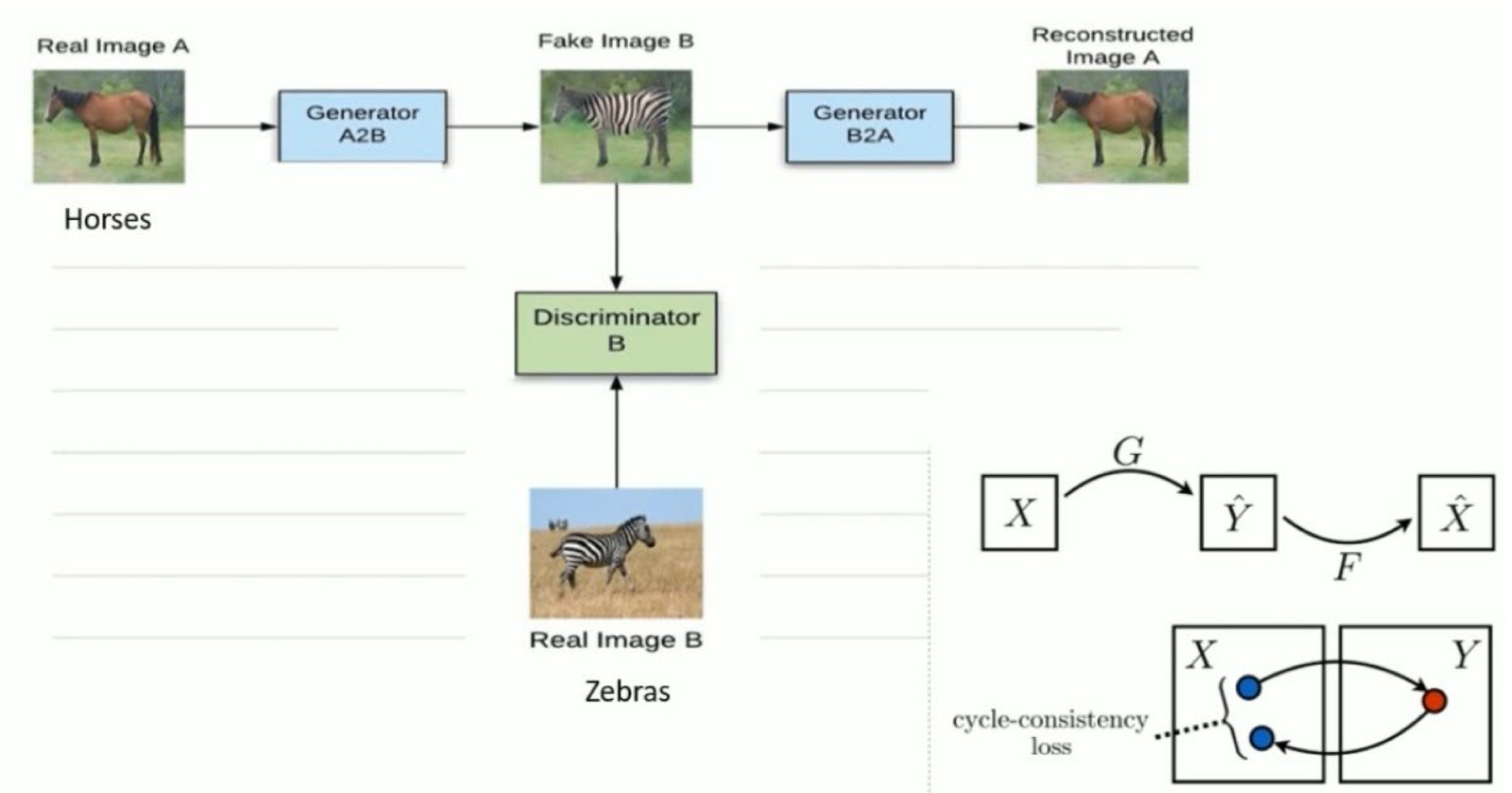
CGAN Output



CycleGAN Applications

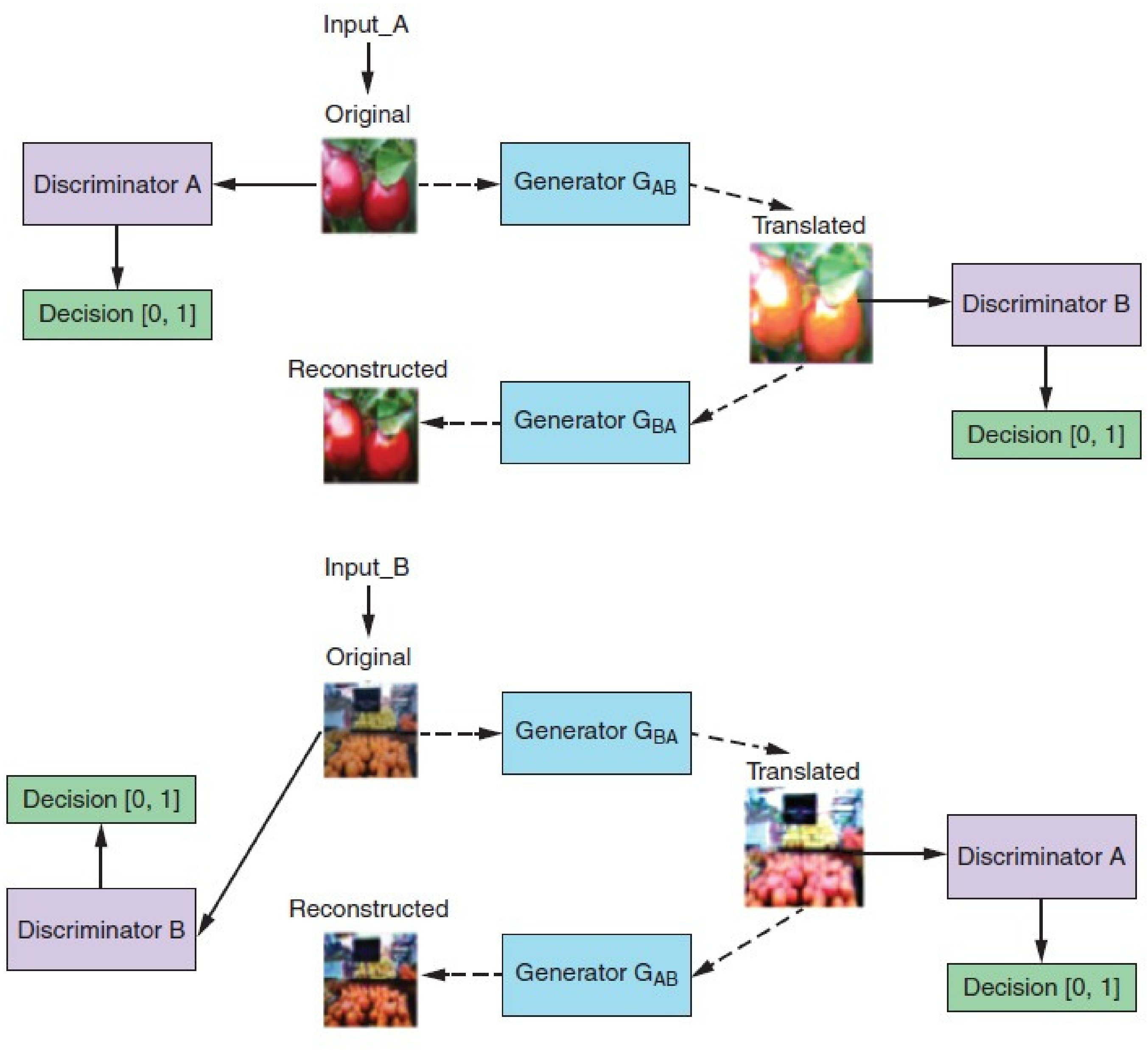


Part 2: Intro GAN



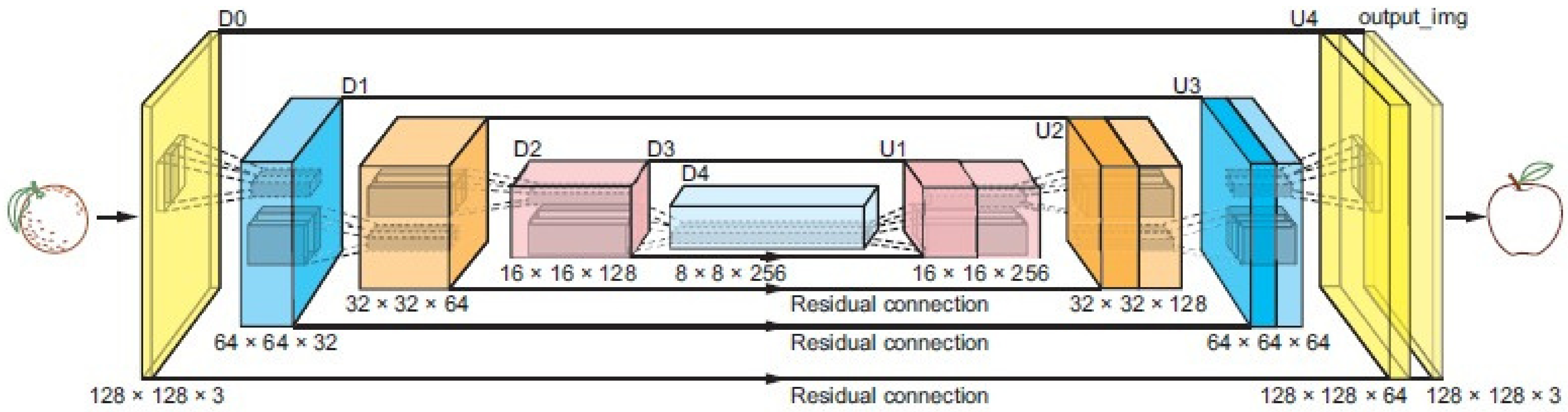
Part 2: Intro GAN

CycleGAN

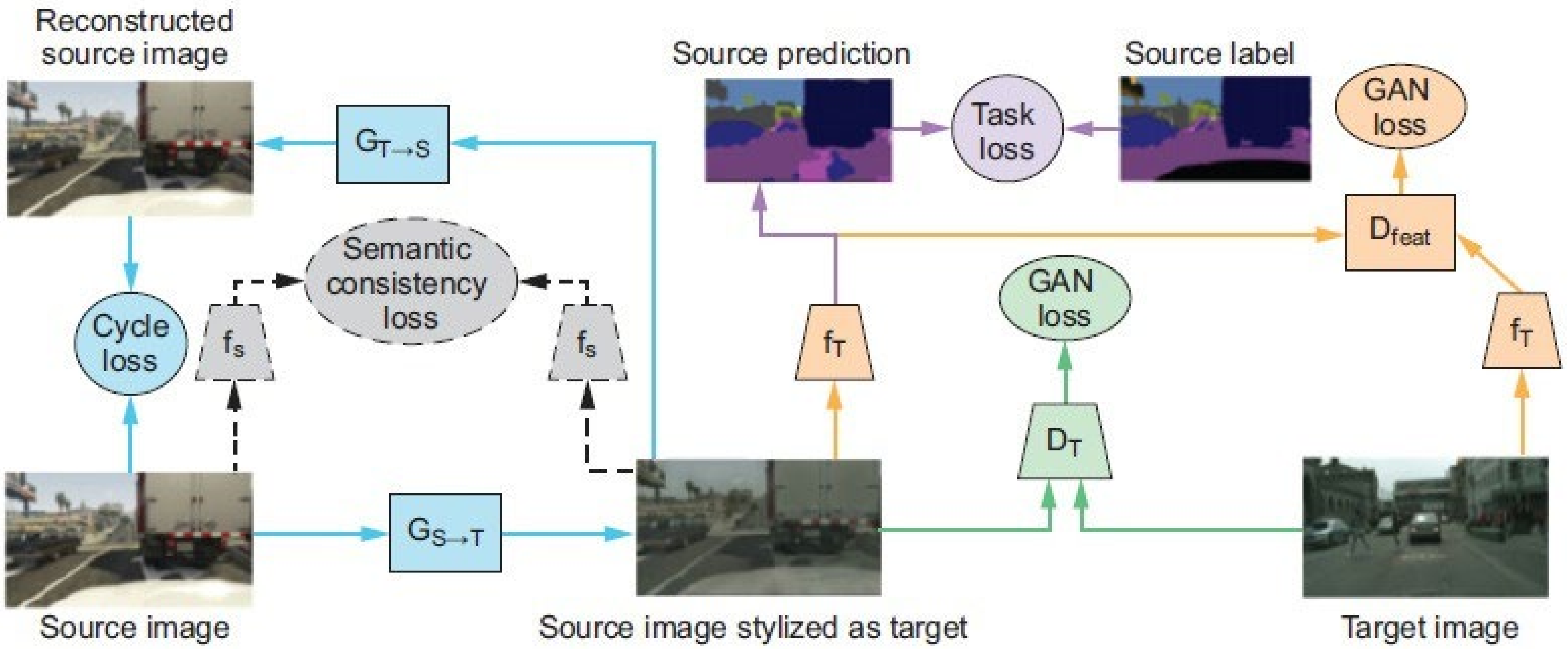


Part 2: Intro GAN

CycleGAN Architecture



Output


Reconstructed source image
Source prediction
Source label
Task loss
GAN loss
$G_{T\to S}$
D_{feat}
Semantic consistency loss
Cycle loss
f_S
f_S
f_T
GAN loss
f_T
D_T
$G_{S\to T}$
Source image
Source image stylized as target
Target image