

Amity School of Engineering and Technology

B.Tech. Computer Science and Engineering

Semester VI

Generative Artificial Intelligence (Generative AI)

[AIML 303]

Faculty: Dr Rinki Gupta

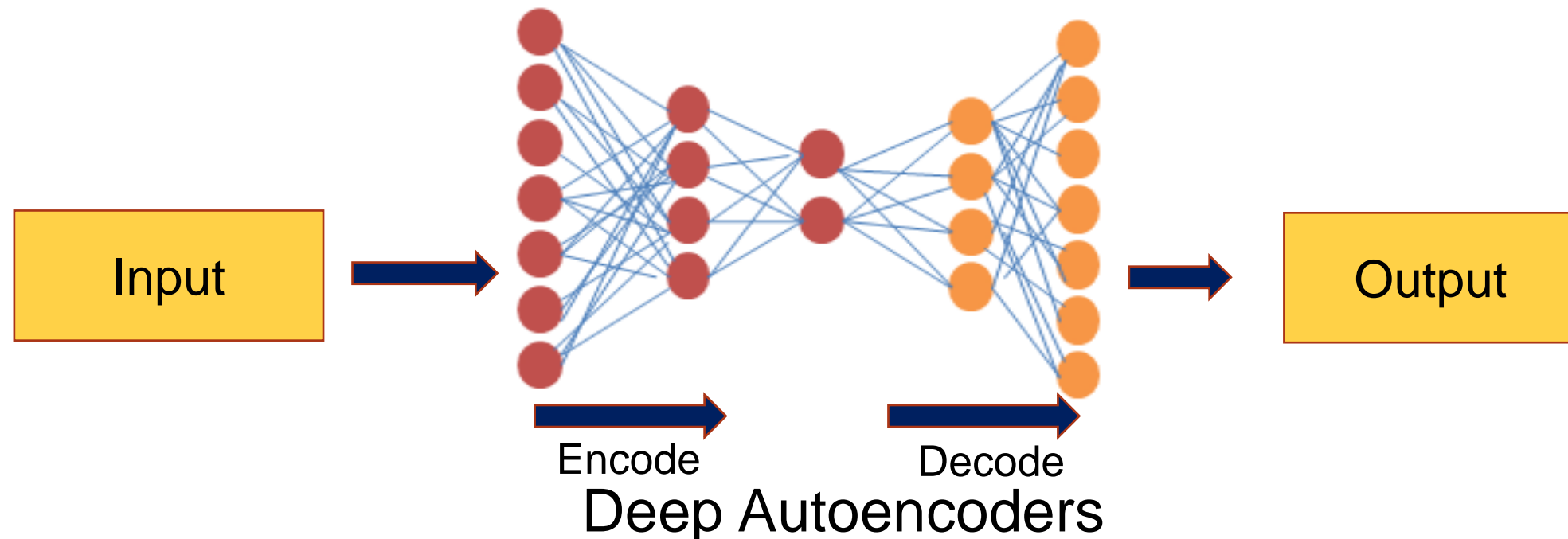
Module III (20%)

Generative Adversarial Networks

- Brief overview of Generative Models,
- Introduction to **Autoencoders** and Variational Autoencoders,
- Generative Adversarial Networks (GANs) for Realistic Data Synthesis,
- Conditional Generative Models for Controlled Data Generation,
- Dealing with common issues like mode collapse and instability,
- Use cases of GANs in image synthesis and data augmentation

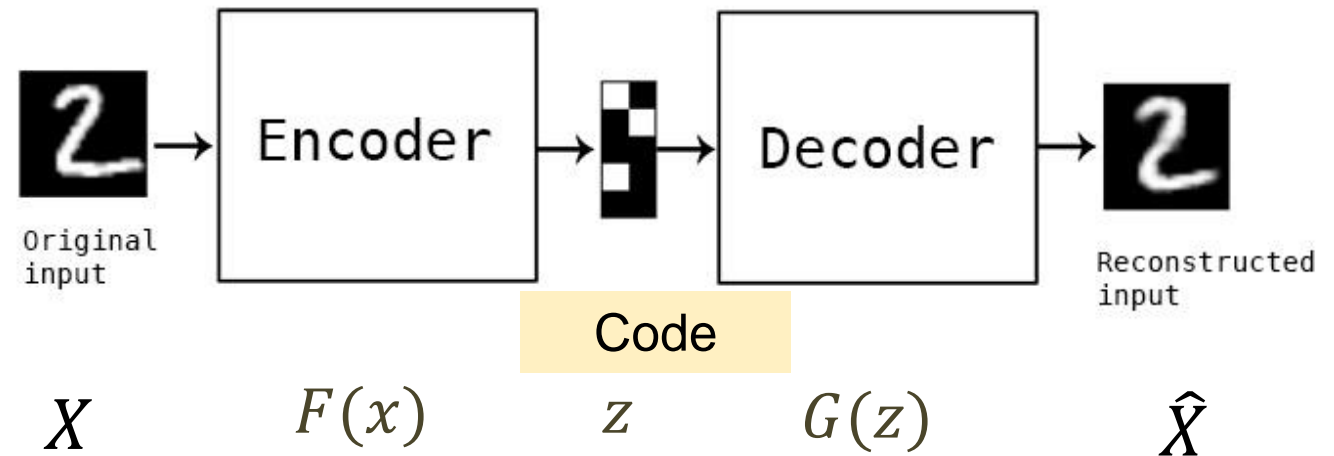
Autoencoders

- **Unsupervised** Feed-forward NN: Target = Input
- **Representation Learning**: NN trained to copy its input to its output



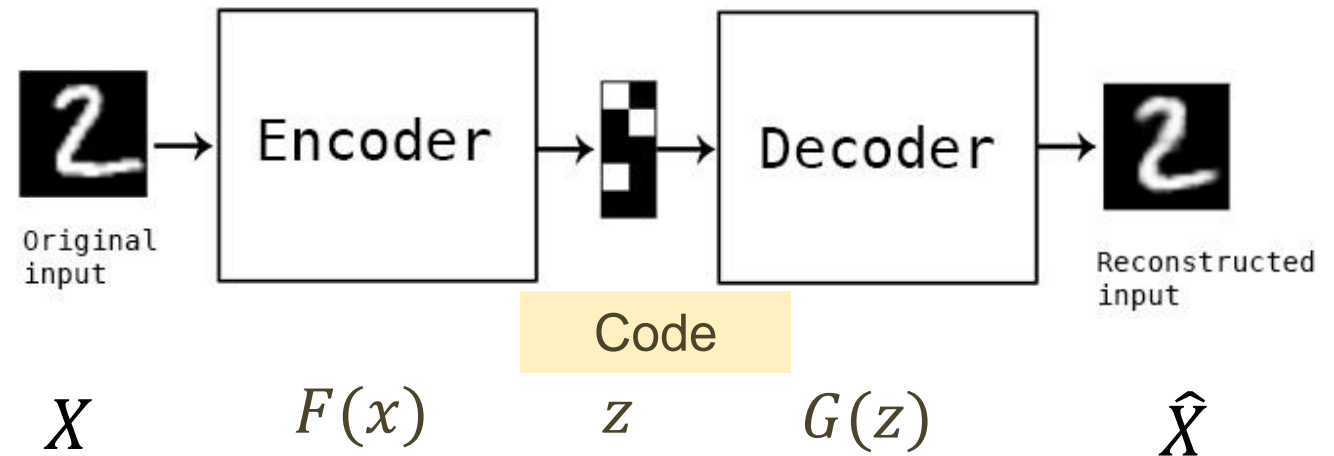
Autoencoders

- **Unsupervised** Feed-forward NN: Target = Input
- **Representation Learning**: NN trained to copy its input to its output
 - **Encoder**: Internally *compress* the input data into a **latent-space representation** (i.e., a single vector that compresses and quantifies the input)
 - **Decoder**: **Reconstructs** the input data from this latent representation (i.e., the output)

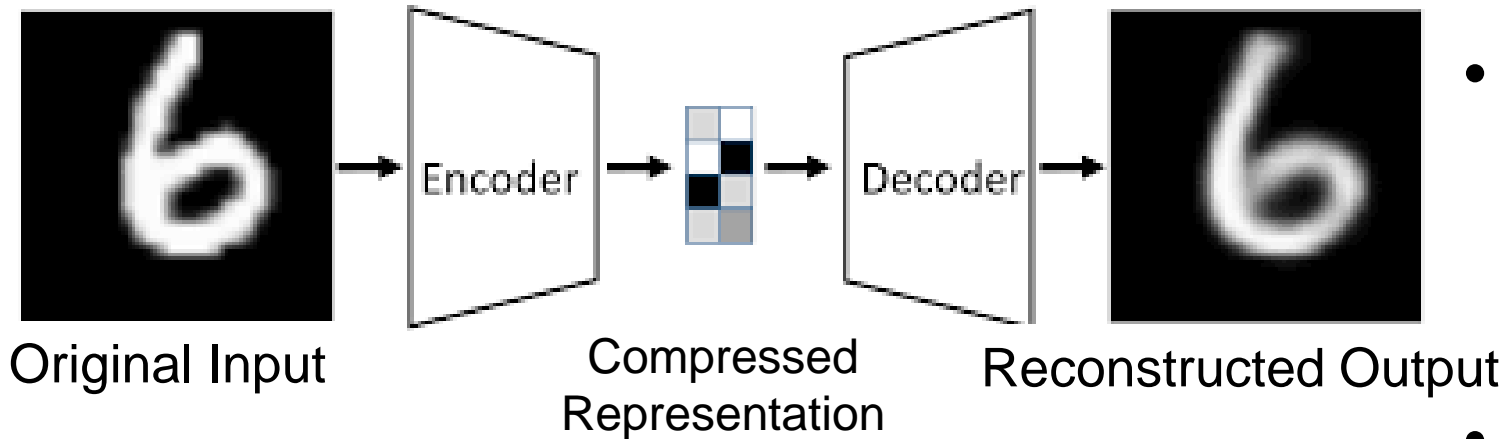


Autoencoders

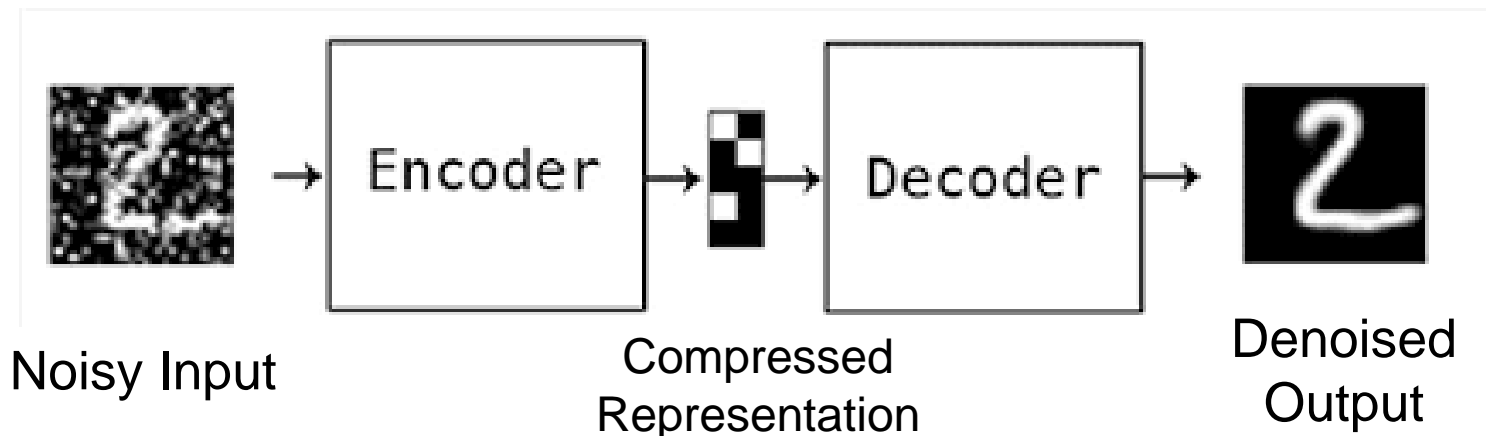
- Autoencoders are designed to be unable to learn to copy perfectly
- Autoencoder that learns $G(F(X))=X$ is not especially useful
- It should be **sensitive** enough to input for accurate representation
- But, it should be **insensitive** enough so that it does not memorize or overfit the training data



Why Autoencoders? Eg Denoising Images

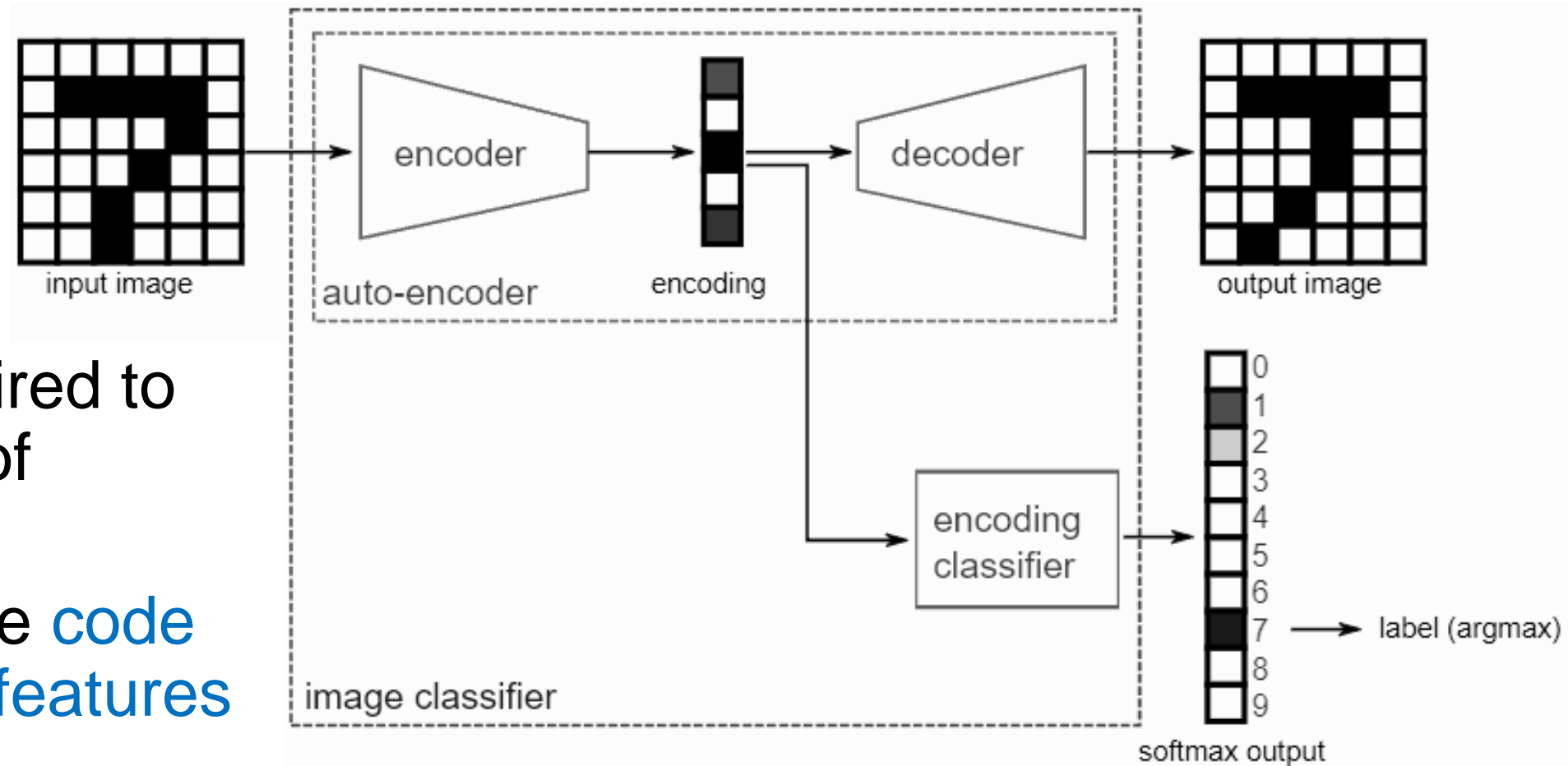


- Eg: Autoencoder is trained to reconstruct pattern of Handwritten digits



- Now, given a noisy image as input, Autoencoder can only reconstruct the Handwritten digit, but not the noise => output is denoised

Autoencoder as Feature Extractor for Classification



- Decoder is required to enable **training** of Autoencoder
- Once trained, the **code** can be used as **features** to a Classifier

Autoencoders are typically used for

- **Denoising** (ex., removing noise and pre-processing images to improve OCR accuracy)- Denoising Autoencoder (DAE)
- **Dimensionality reduction** (like PCA but more powerful/intelligent)
 - Eg: $28 \times 28 = 784$ pixel MNIST image to vector of 32 values!
- **Anomaly/outlier detection** (eg. detecting mislabeled data points in a dataset or detecting when an input data point that falls well outside the typical data distribution)

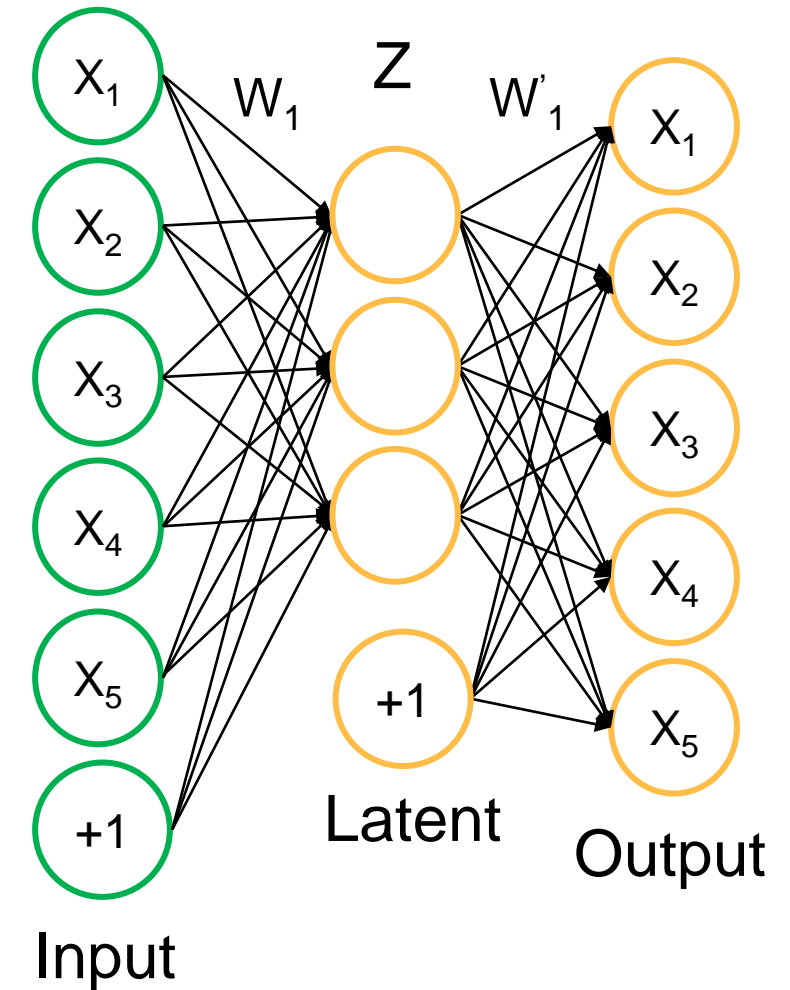
Autoencoders

- Unsupervised Learning.
- Representation Learning.
- Impose a bottleneck in the network.
- The bottleneck forces a compressed knowledge representation of the input.

Assumption:

High degree of correlation or structure exist in the data.

This is also called as **Vanilla Autoencoder** →



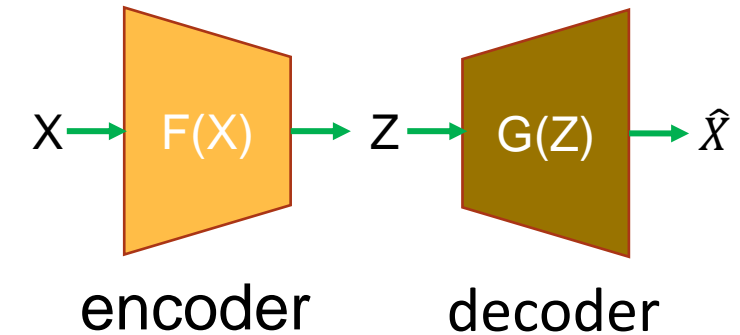
Mathematical Formulation of Autoencoder

- Given data X (no labels) we would like to learn the functions F (encoder) and G (decoder) where:

$$F(X) = S(W.X + b) = Z$$

$$G(Z) = S(W'.Z + b') = \hat{X}$$

$$h(X) = G(F(X)) = \hat{X}$$

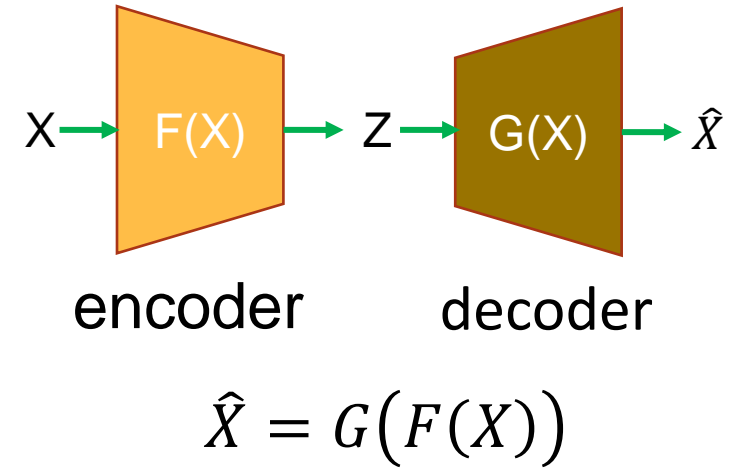


(Z is some latent representation or code and S is non-linearity such as the sigmoid.)

\hat{X} is the X 's reconstruction

Training the Autoencoder

- Trained the same way as ANNs via **backpropagation**
- Learning process is described simply as **minimizing a loss** function, $L(X, G(F(X)))$
- Loss function: mean squared error (mse) or binary crossentropy
- Using **Backpropagation**, we can simply train the model as any other FC NN with:
 - Sensitive Enough to input for accurate reconstruction.
 - Insensitive enough that it does not memorize or overfit the training data.



Optimize parameters to
Minimize Loss

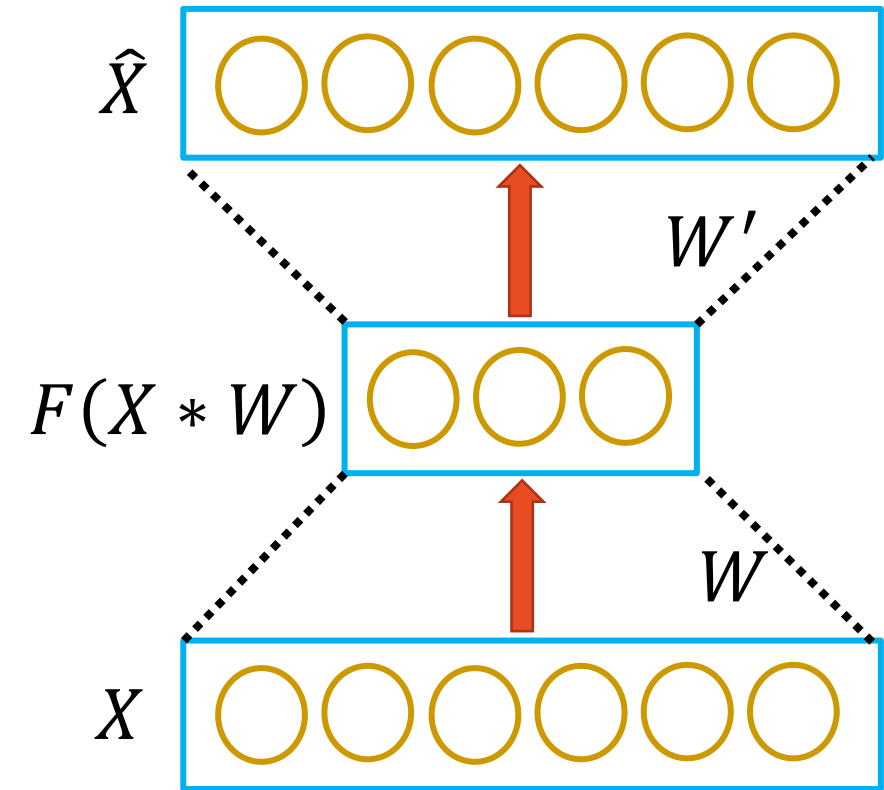
$$L(X, \hat{X}) = \|X - \hat{X}\|^2$$

Types of Autoencoders

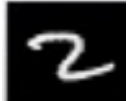

- Undercomplete Autoencoder
- Overcomplete Autoencoder
- Sparse Autoencoder
- Deep Autoencoder
- Convolutional Autoencoder
- Denoising Autoencoder

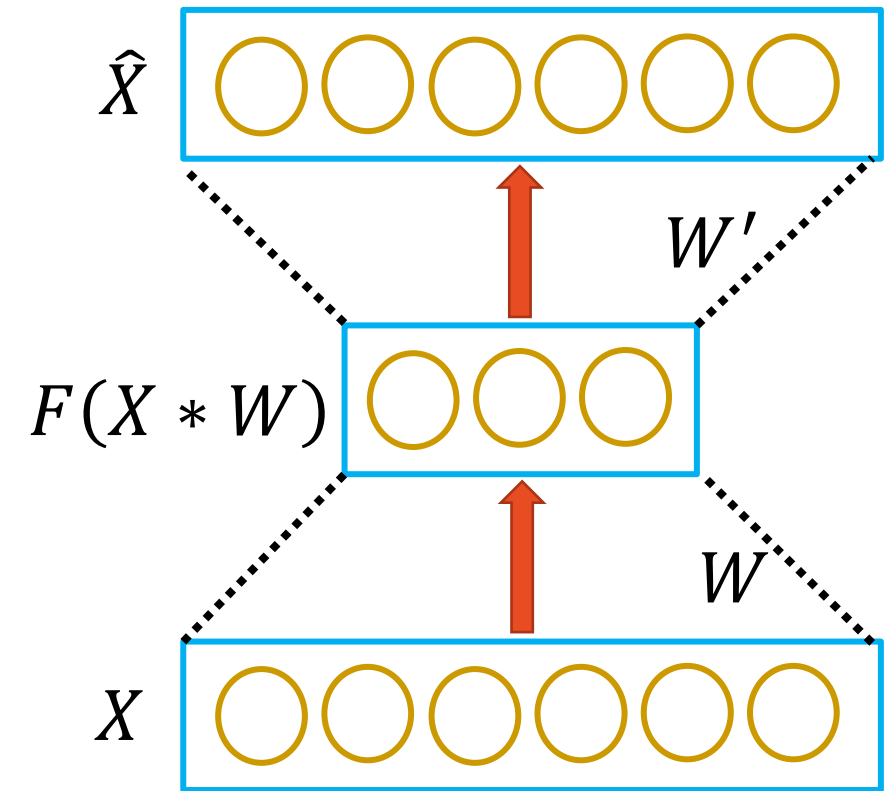
Undercomplete Autoencoder

- Code dimension < Input dimension
- Hidden layer is **Undercomplete** if smaller than the input layer.
 - Compresses the input.
 - Compresses well only for the training distribution.
- “**Bottleneck**” is imposed
 - Map the input to a lower dimension feature map
 - then network does not learn Identity function
 - It is assumed that there is high degree of correlation in the data (data concentrates around a low-dimensional manifold)



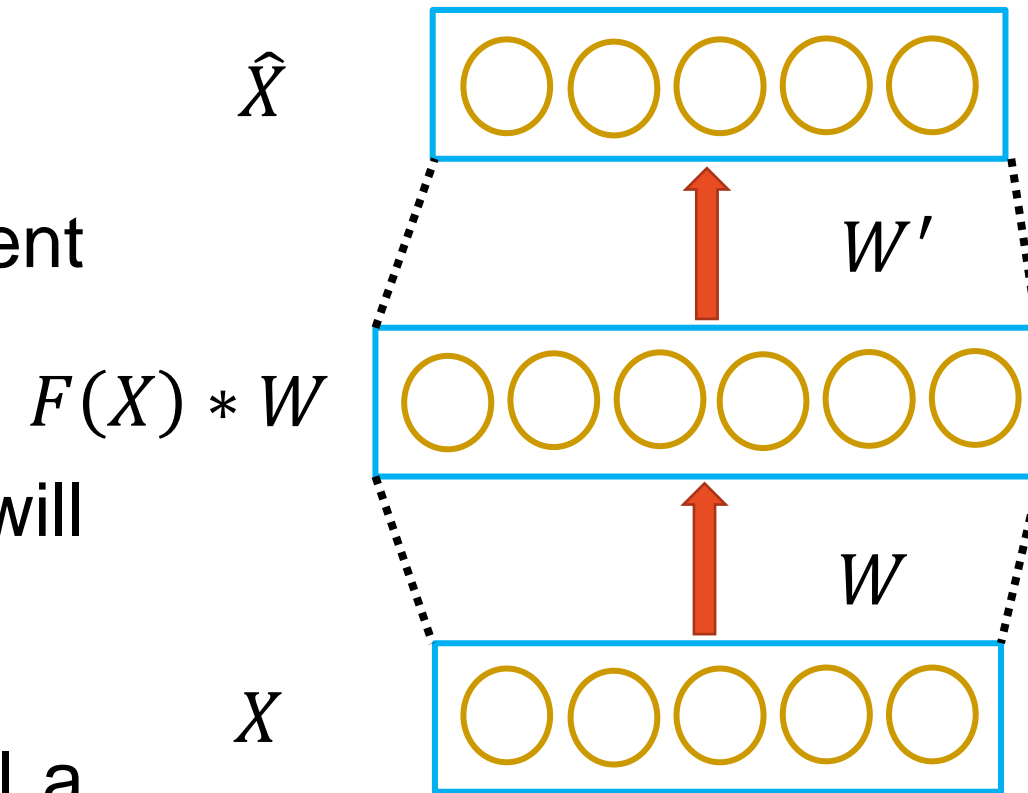
Undercomplete Autoencoder

- Hidden layer is **Undercomplete** if smaller than the input layer.
 - Compresses the input.
 - Compresses well only for the training distribution.
- Goal of the Autoencoder is to capture the most important features present in the data.
 - Hidden nodes will be good features for the training distribution. 
 - Hidden nodes will be bad for other types on input. 



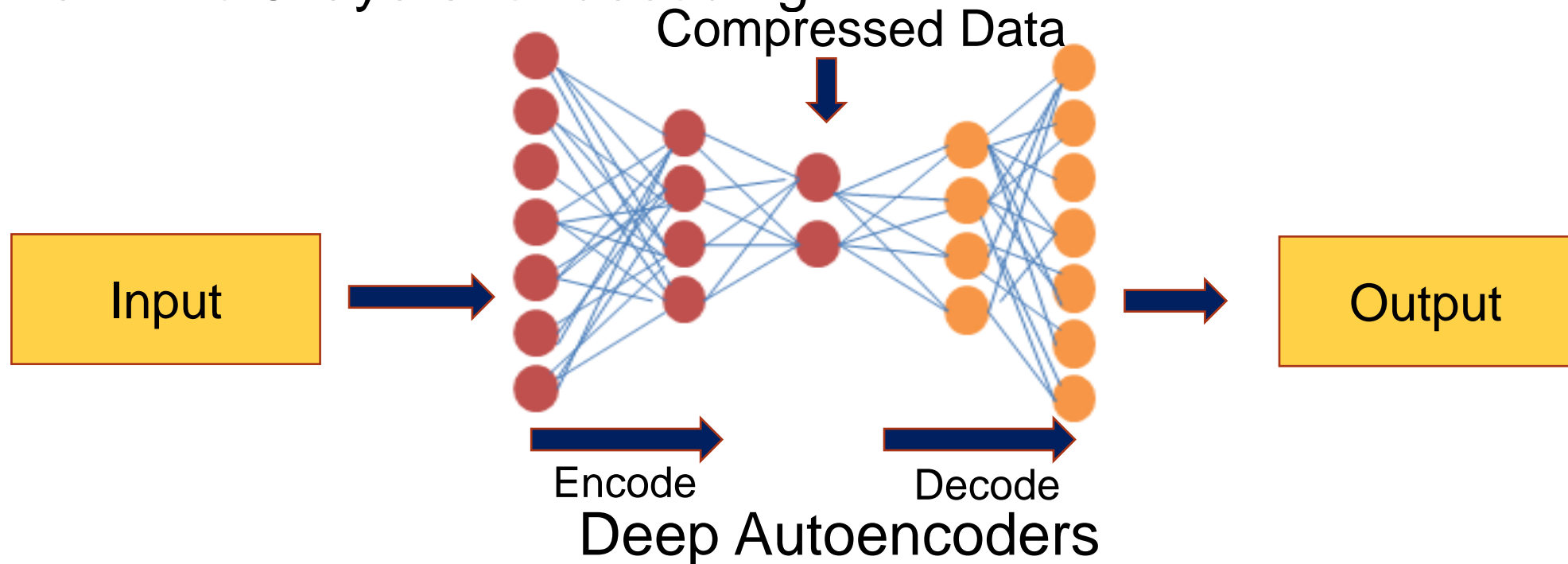
Overcomplete Autoencoder

- Hidden layer is **Overcomplete** if greater than the input layer.
 - No compression in the hidden layer.
 - Each hidden unit could copy a different input component.
- No guarantee that the hidden units will extract meaningful structure.
- A higher dimension code helps model a more complex distribution.

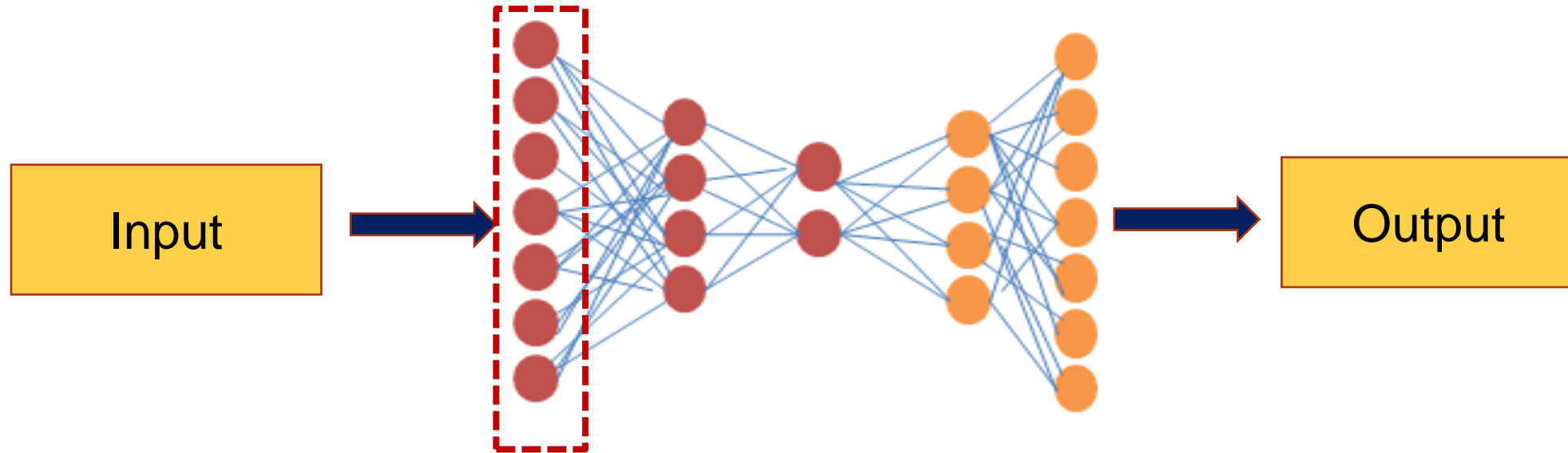


Deep Autoencoder

- Deep Autoencoders consist of two identical deep neural networks
- One network for encoding and another for decoding.
- Typically, deep autoencoders have 4 to 5 layers for encoding and the next 4 to 5 layers for decoding.

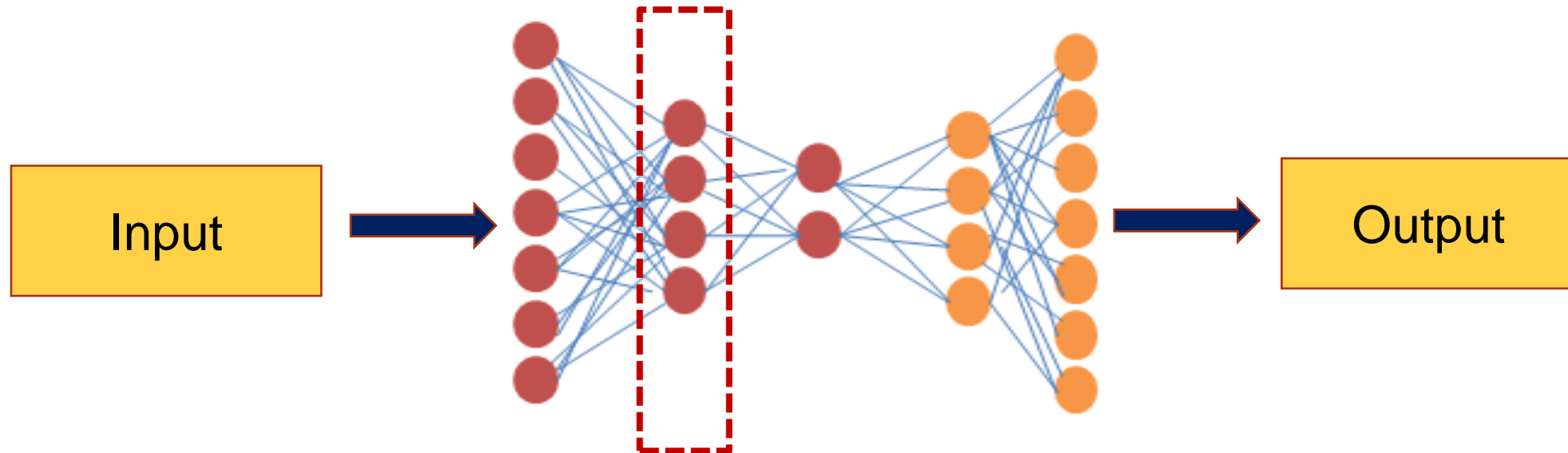


Deep Autoencoder



The first layer of the Deep Autoencoders learns first-order features in the raw input such as edges in the image.

Deep Autoencoder



The second layer of the Deep Autoencoders learns second-order features corresponding to patterns in the appearance of first-order features.

Deep Autoencoder

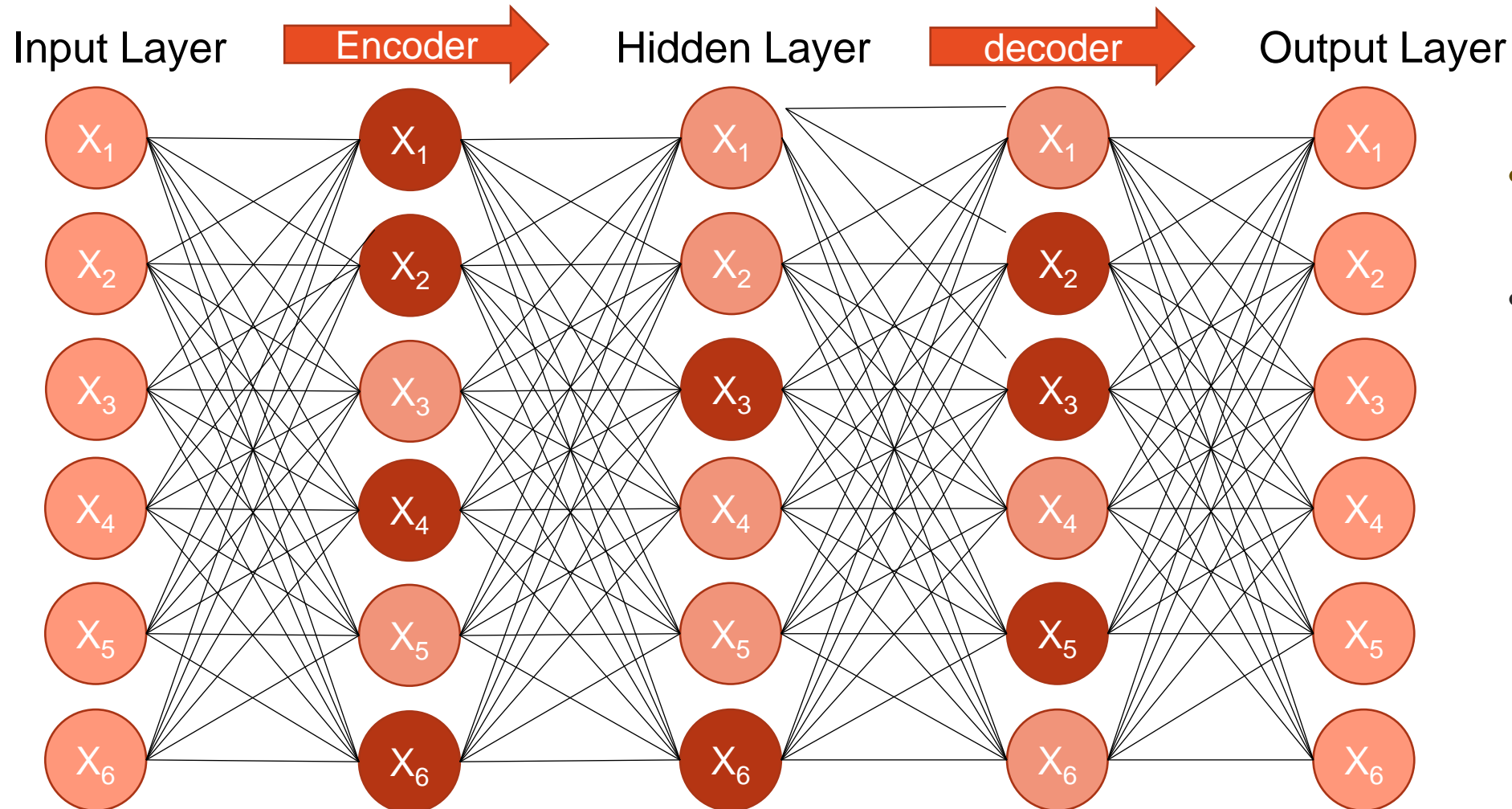
Advantages:

- Final encoding layer is compact and fast.

Disadvantages:

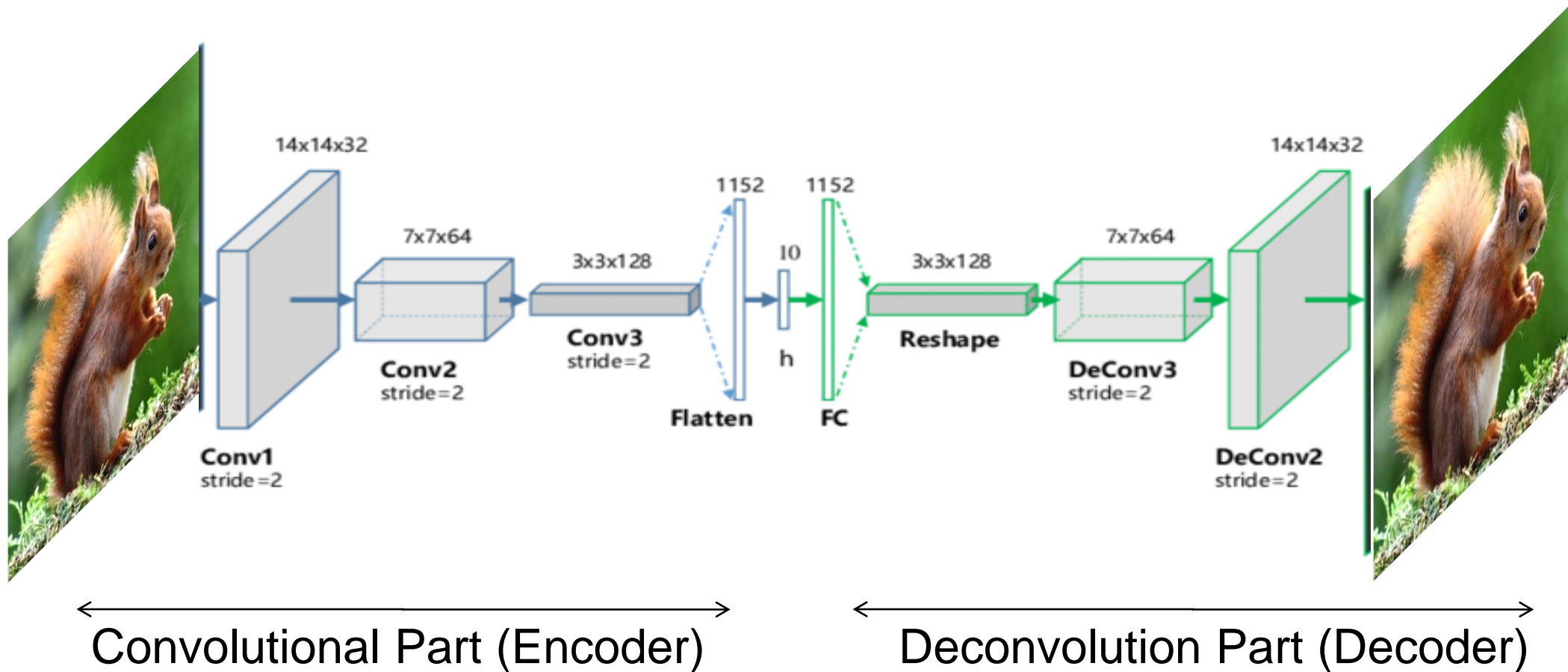
- Chances of overfitting to occur since there's more parameters than input data.
- Training the data maybe a nuance since at the stage of the decoder's backpropagation, the learning rate should be lowered or made slower depending on whether binary or continuous data is being handled.

Sparse Autoencoder



- Fewer nodes activate
- Limit network's capacity to memorize the input data without limiting the network's capability to extract features

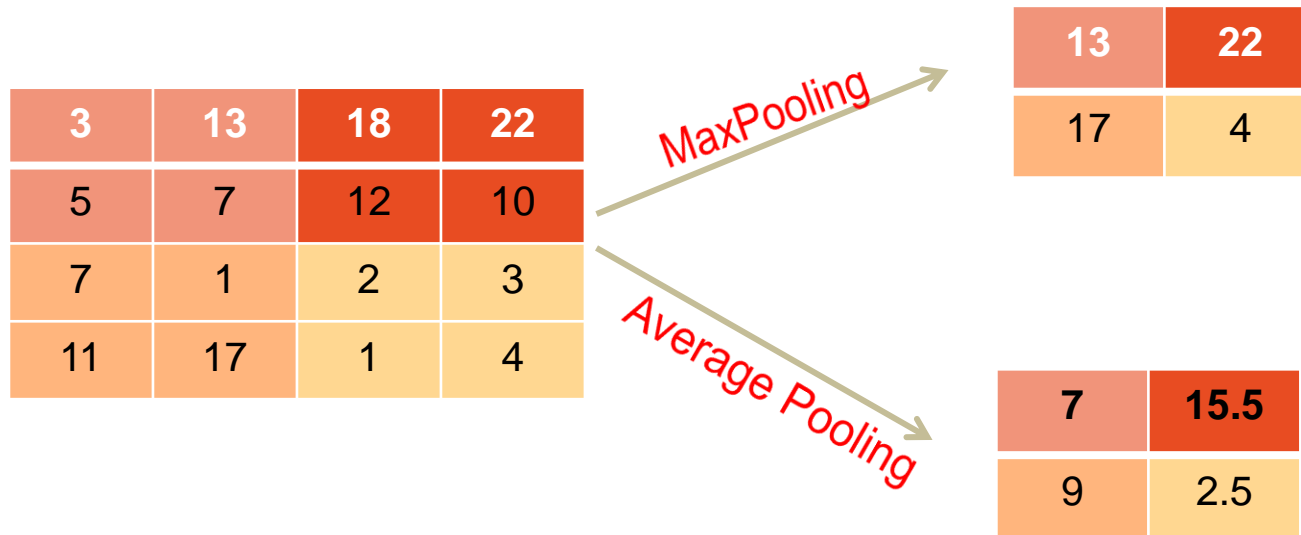
Convolutional Autoencoder



Convolution Autoencoder

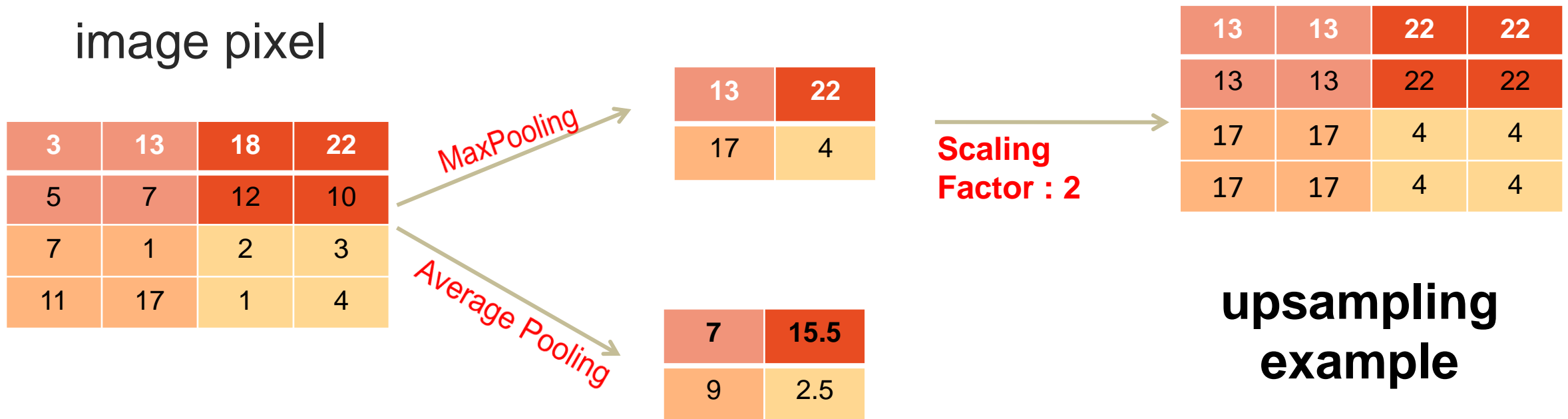
Convolutional Autoencoder

- Convolutional neural networks provide a better feature extraction.
- The tricky part of CAEs is at the decoder side of the model.

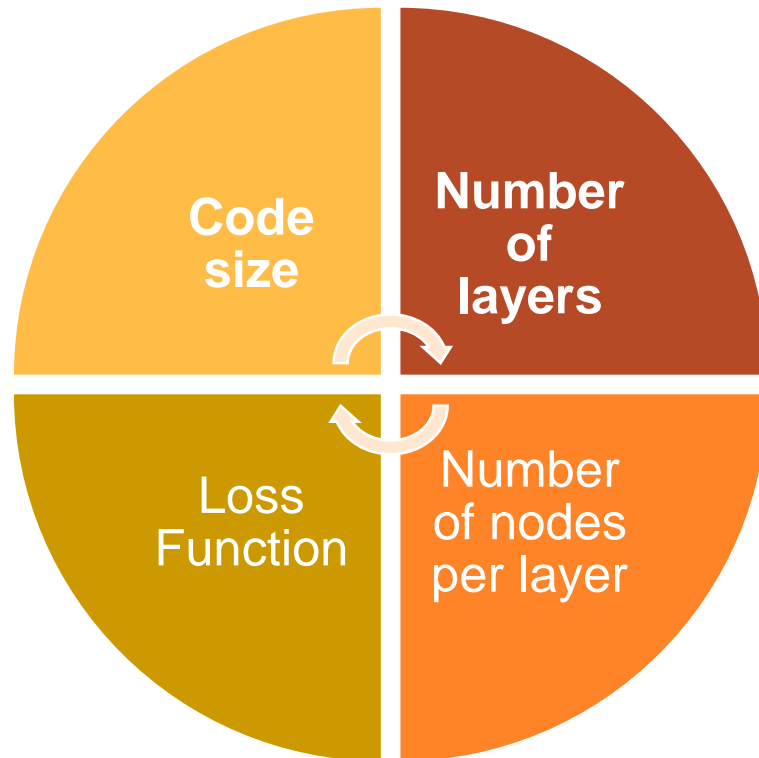


Convolutional Autoencoder

- Convolutional neural networks provide a better feature extraction.
- The tricky part of CAEs is at the decoder side of the model.
- During **decoding**, up-sampling required : populating the same image pixel



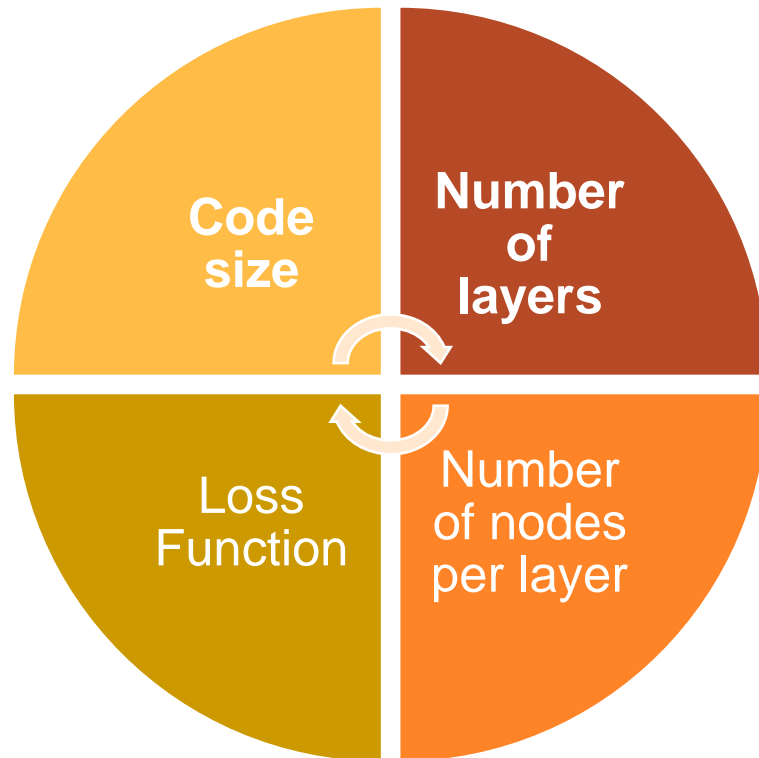
Hyperparameters of Autoencoders



Code size or Latent size

- Smaller size results in more compression.

Hyperparameters of Autoencoders



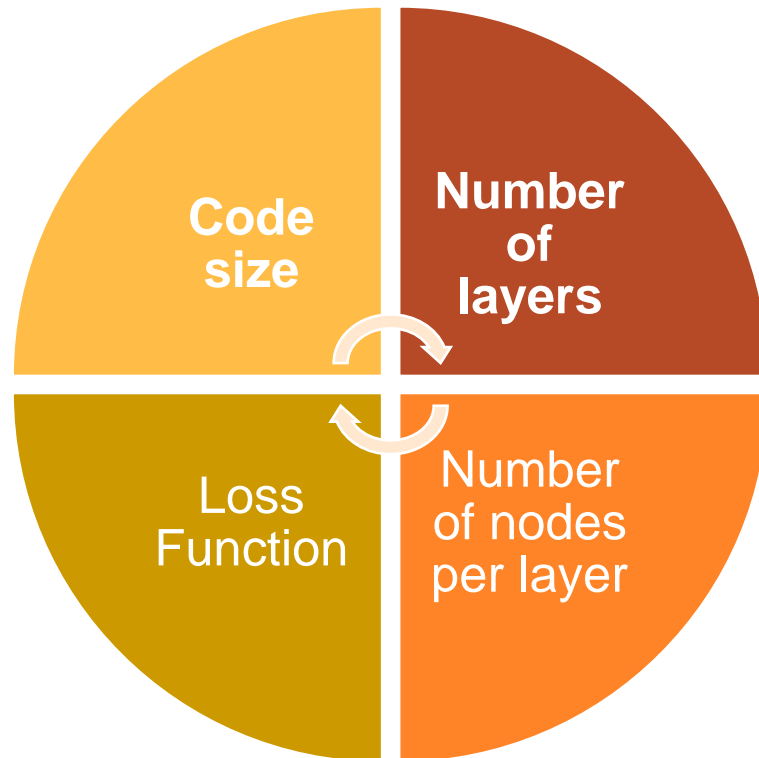
Code size or Latent size

- Smaller size results in more compression.

Number of layers

- The autoencoder consists of as many layers as we want.

Hyperparameters of Autoencoders



Code size or Latent size

- Smaller size results in more compression.

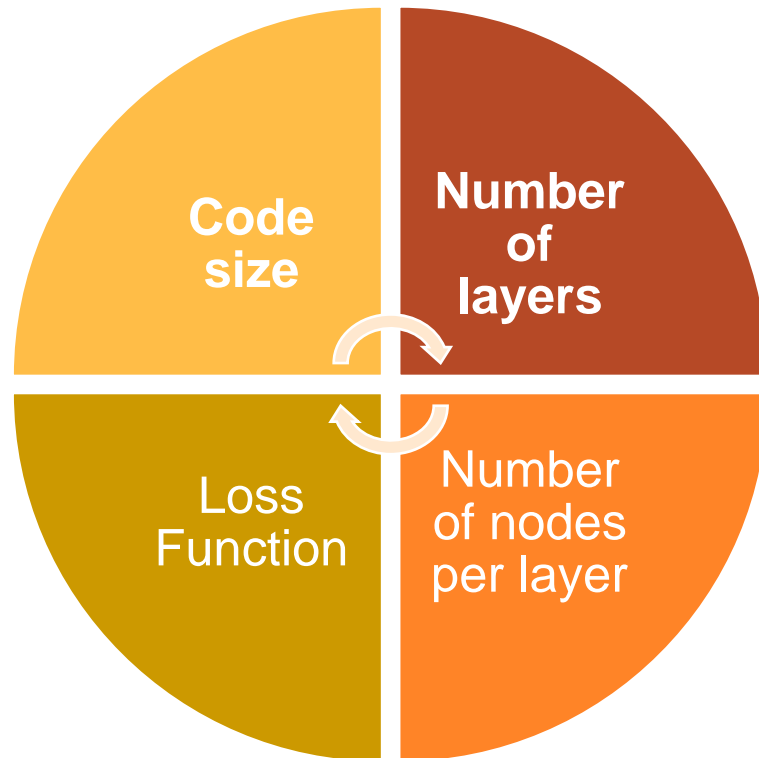
Number of layers

- The autoencoder consists of as many layers as we want.

Number of nodes per layer

- The number of nodes per layer decreases with each subsequent layer of the encoder, and increases back in the decoder

Hyperparameters of Autoencoders



Code size or Latent size

- Smaller size results in more compression.

Number of layers

- The autoencoder consists of as many layers as we want.

Number of nodes per layer

- The number of nodes per layer decreases with each subsequent layer of the encoder and increases back in the decoder.

Loss function

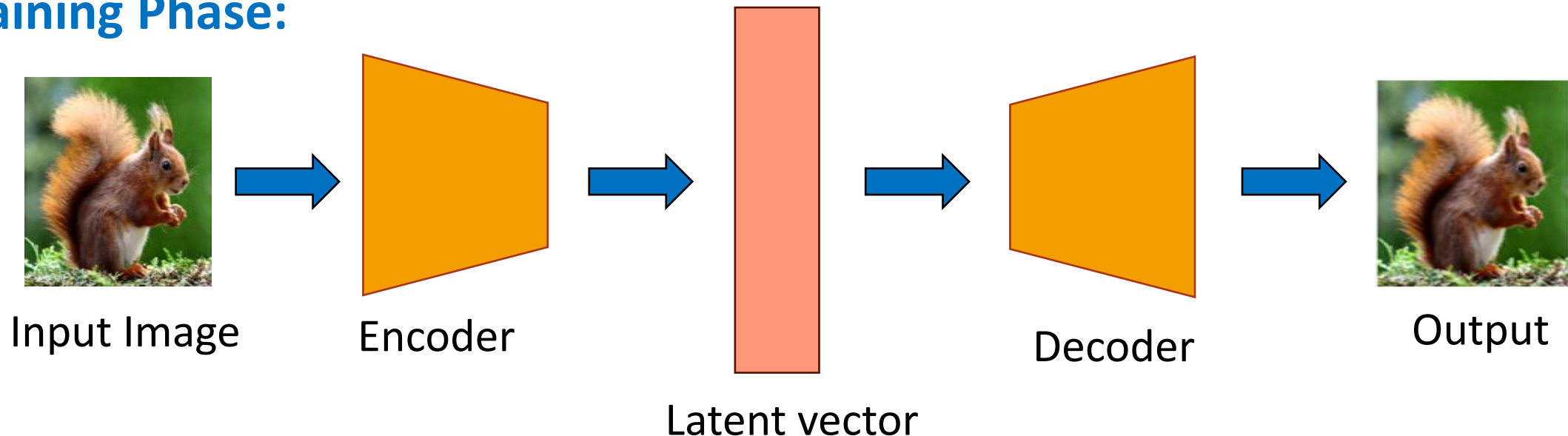
- Either use mean squared error or binary cross-entropy.

Properties of Autoencoders

Data-specific:

Autoencoders are only able to compress data like what they have been trained on.

Training Phase:

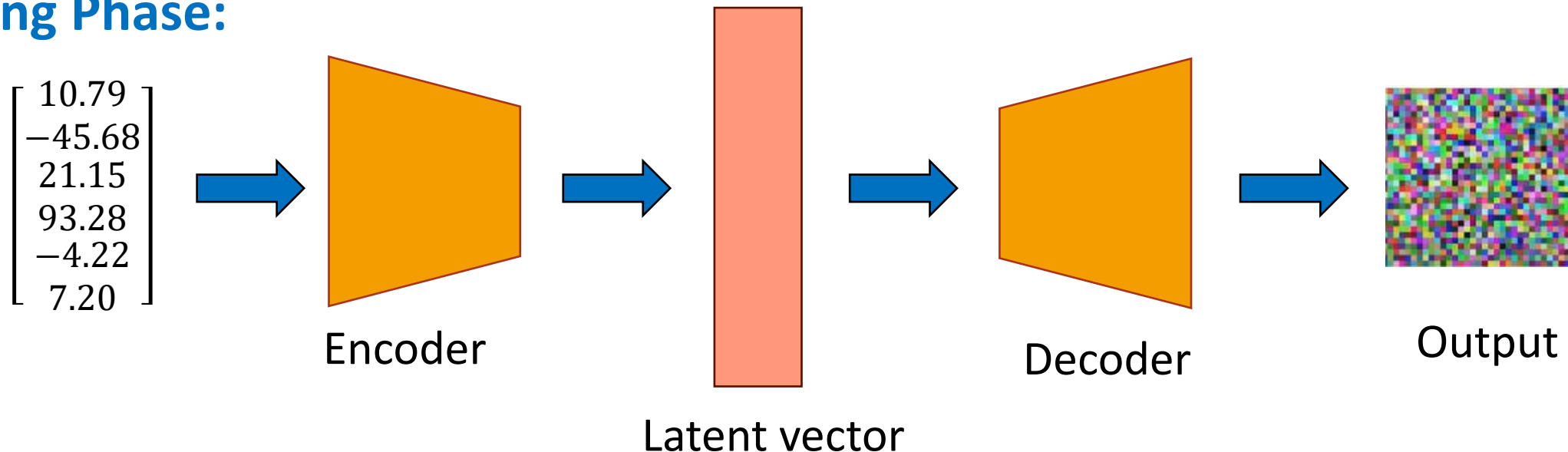


Properties of Autoencoders

Data-specific:

Autoencoders are only able to compress data similar to what they have been trained on.

Testing Phase:



Properties of Autoencoders

Data-specific:

Autoencoders are only able to compress data similar to what they have been trained on.

Lossy:

The decompressed outputs will be degraded compared to the original inputs.

Module III (20%)

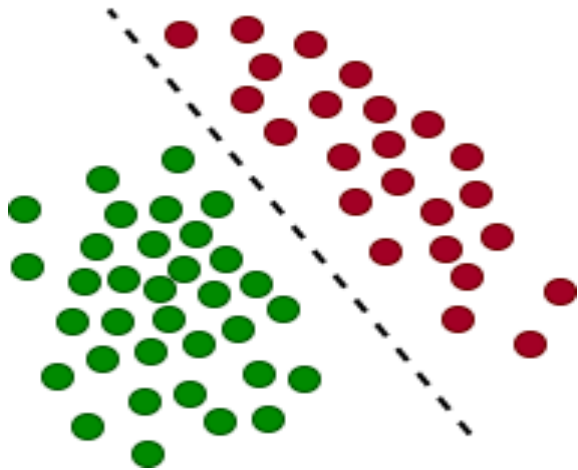
Generative Adversarial Networks

- Brief overview of Generative Models,
- Introduction to Autoencoders and Variational Autoencoders,
- Generative Adversarial Networks (GANs) for Realistic Data Synthesis,
- Conditional Generative Models for Controlled Data Generation,
- Dealing with common issues like mode collapse and instability,
- Use cases of GANs in image synthesis and data augmentation

Generative Models vs. Discriminative Models

Discriminative models:

- Learn the characteristics that distinguish different classes in a dataset.

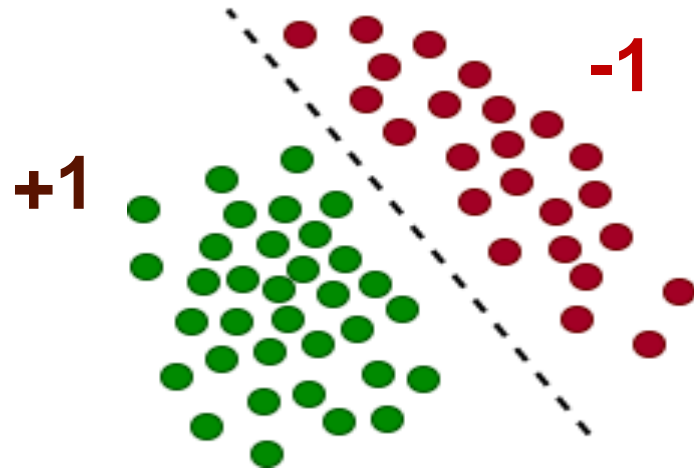


Discriminative

Generative Models vs. Discriminative Models

Discriminative models:

- Learn the characteristics that distinguish different classes in a dataset.



Discriminative

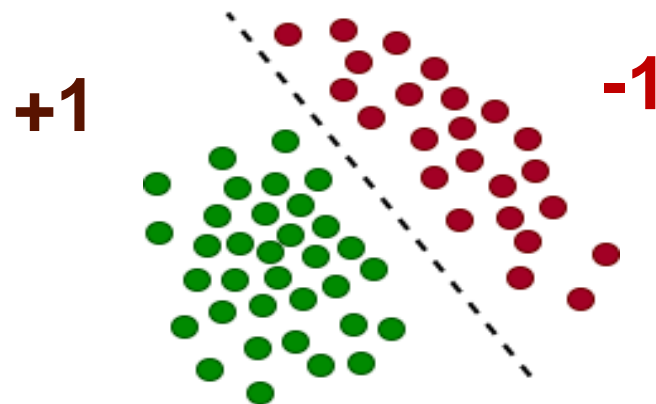
Generative Models vs. Discriminative Models

Discriminative models:

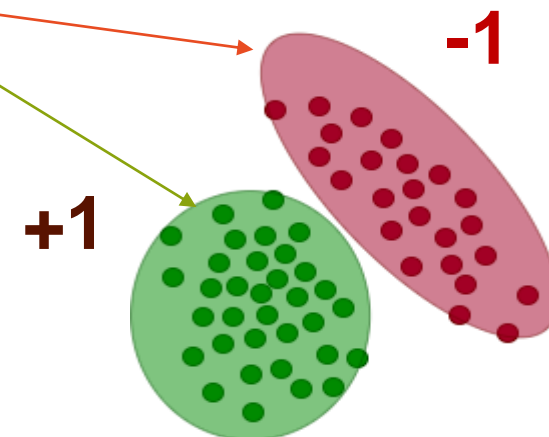
- Learn the characteristics that distinguish different classes in a dataset.

Generative models:

- Learn the fundamental distribution of each class inside a dataset.



Discriminative



Generative

Why Generative Models?



Image Super Resolution

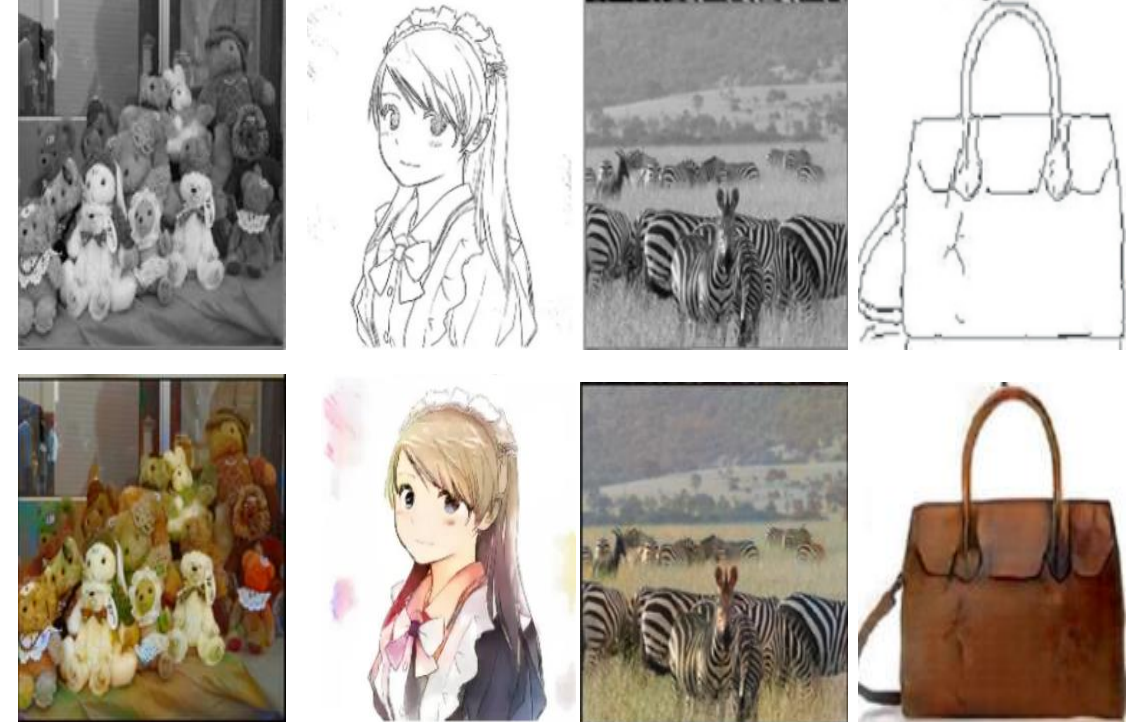


Image Colorization

Why Generative Models?

Zebras \longleftrightarrow Horses

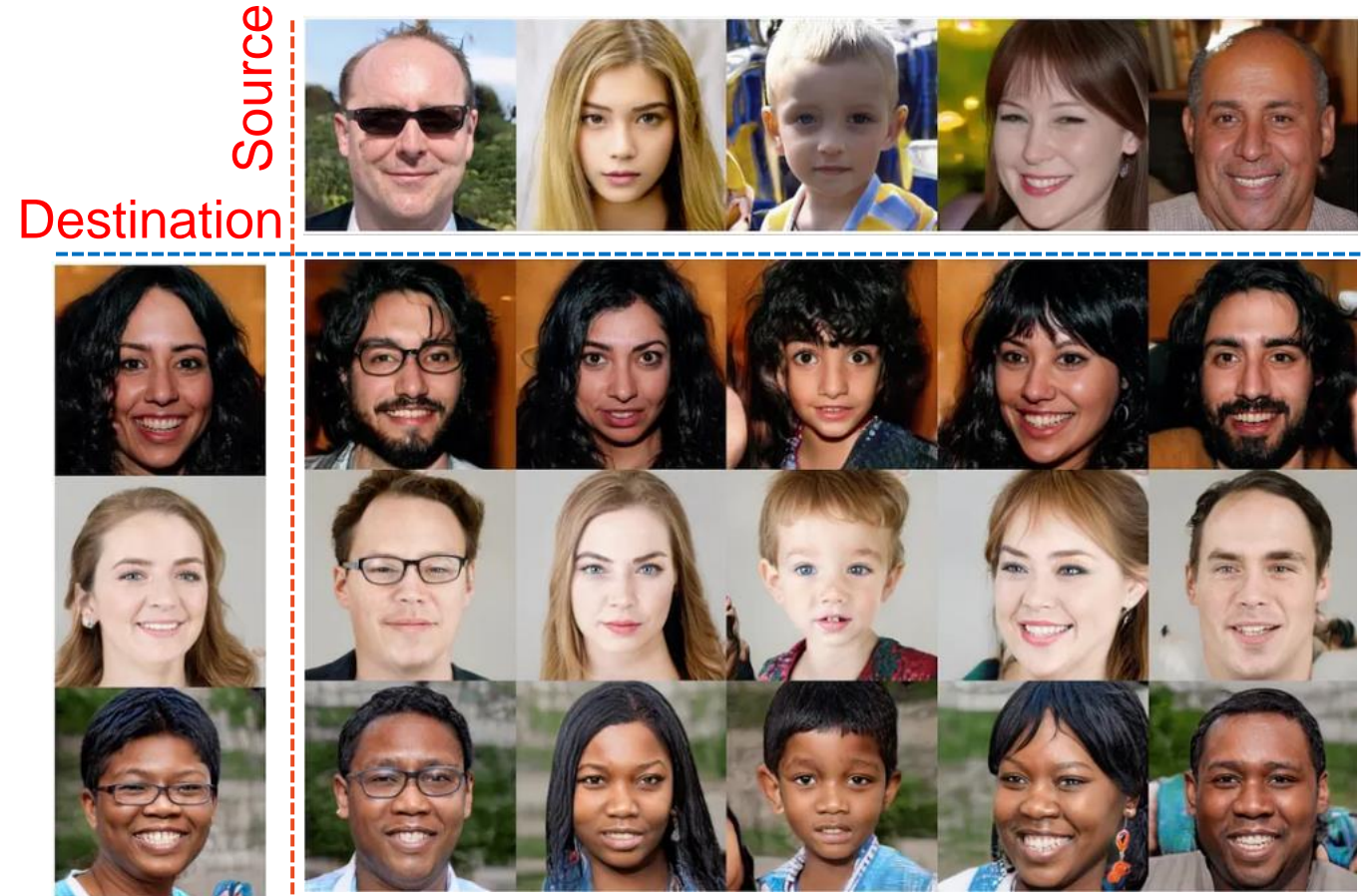


Horse \longrightarrow Zebra



Zebra \longrightarrow Horse

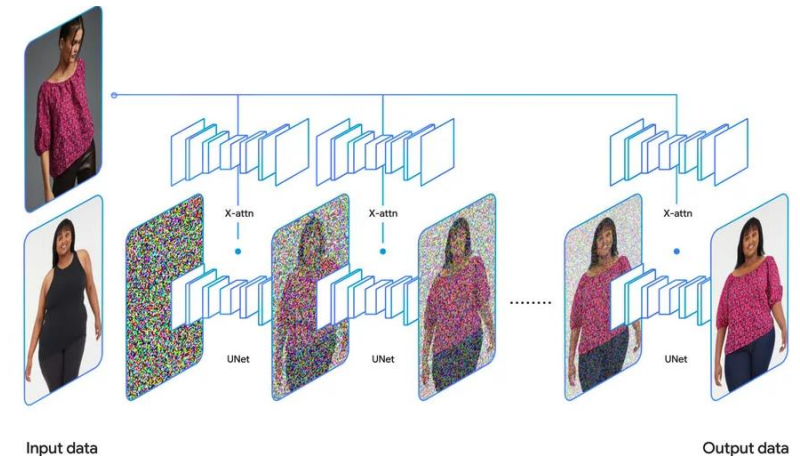
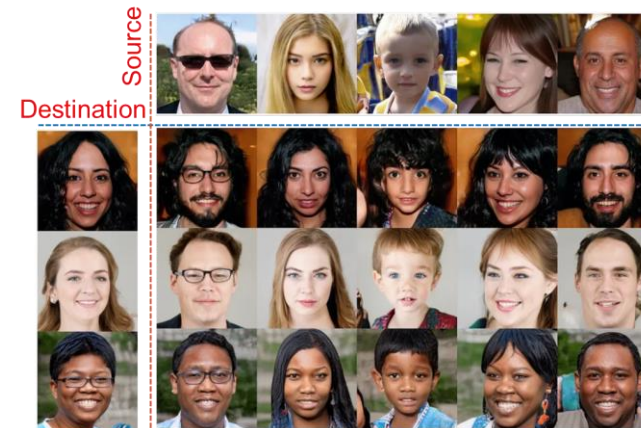
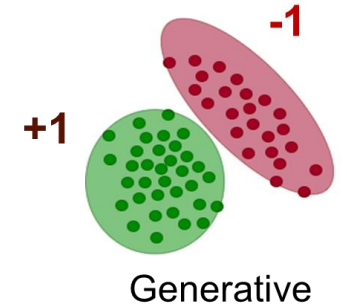
Cross-domain Image
Translation



Generating Realistic Face
Images

Why Generative Models?

- Learn significant generalizable latent characteristics.
- Augment small datasets.
- Facilitate mixed-reality applications like Virtual Try-on.
- Many more..



Autoencoder generative?

Before we start

- **Question?**
 - Are the previous Autoencoders generative model?

Autoencoder generative? NO

Before we start

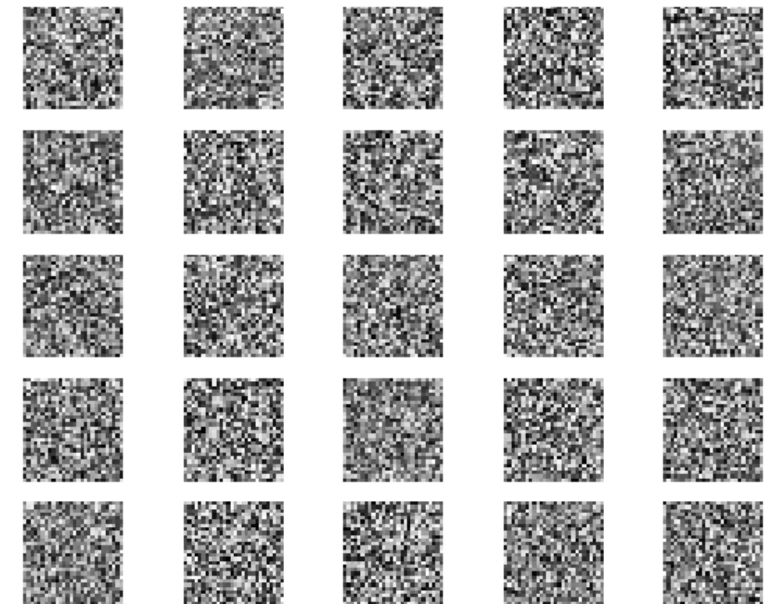
- **Question?**
 - Are the previous Autoencoders generative model?
(**NO, The compressed latent codes of autoencoders are not prior distributions, autoencoder cannot learn to represent the data distribution**). Autoencoders learn the feature representation.
- Generative models are typically meant for modelling data distribution, $P(X)$.

Generative Models

Goal: Given a training data, generate data samples similar to the one in the training set

- Assuming training data $X = \{x_1, x_2, \dots, x_n\}$ comes from an underlying distribution $P_{\text{data}}(X)$, and a generative model samples data from a distribution $P_{\text{model}}(X)$.
- Aim is to **minimize** the some notion of **distance** between $P_{\text{data}}(X)$ and $P_{\text{model}}(X)$.

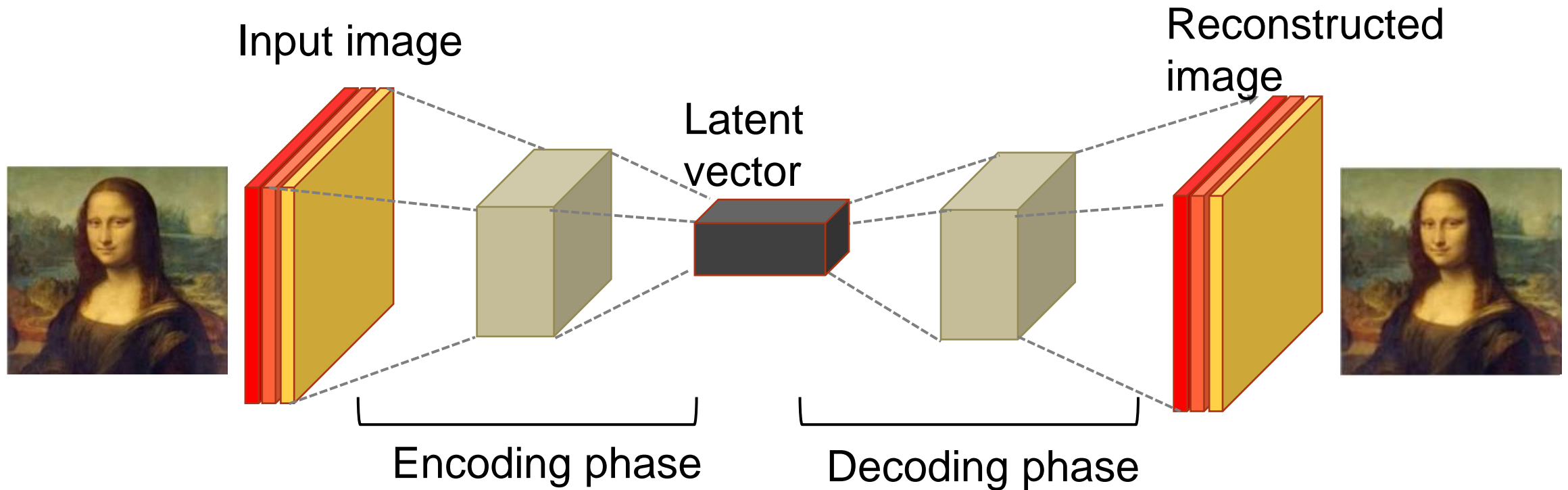
Generative Adversarial Network



Generation of hand written digits over several iterations

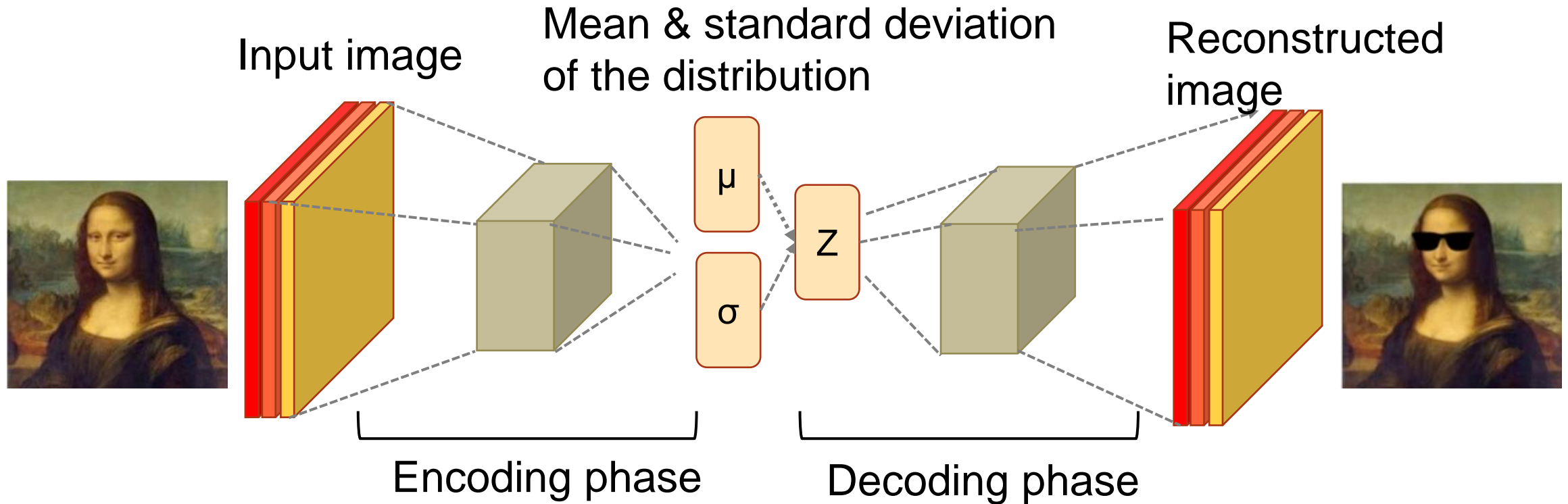
Variational Autoencoders (VAE)

Autoencoder



*Autoencoders learn the feature representation
They are Data-specific*

Variational Autoencoder

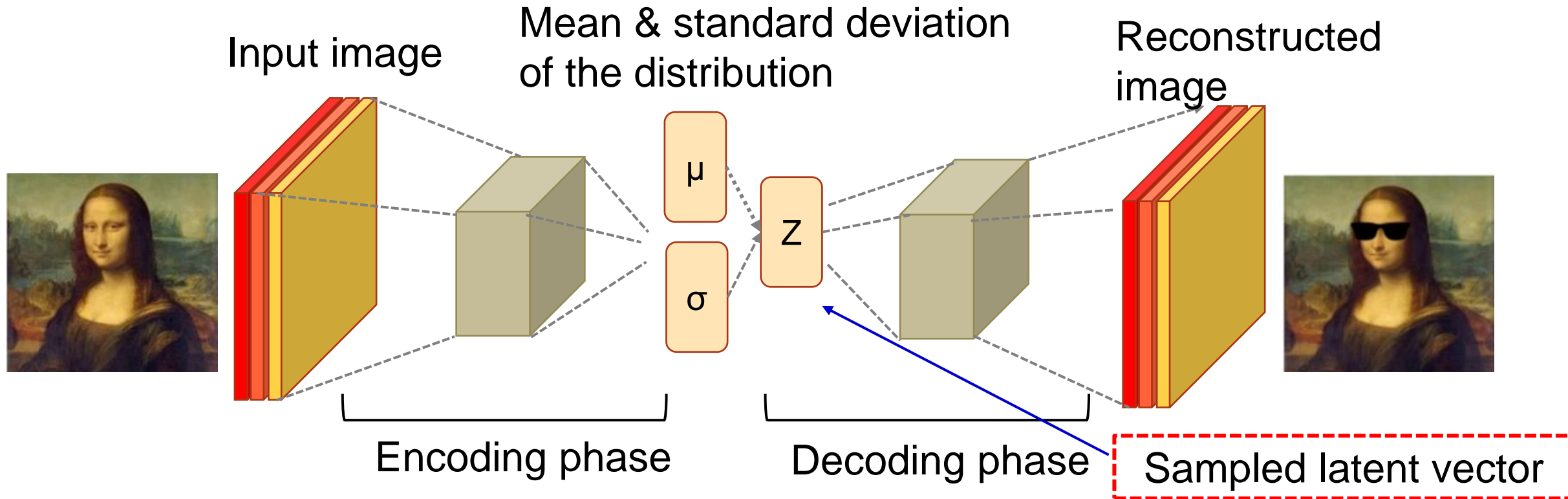


- Instead of learning the latent variables z directly for each input, VAE learns a mean and a variance associated with that latent variable.
- Mean and variance parameterize the probability distribution for that latent variable

<https://www.youtube.com/watch?v=BUNi0To1IVw> (start from 17:19)

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

Variational Autoencoder



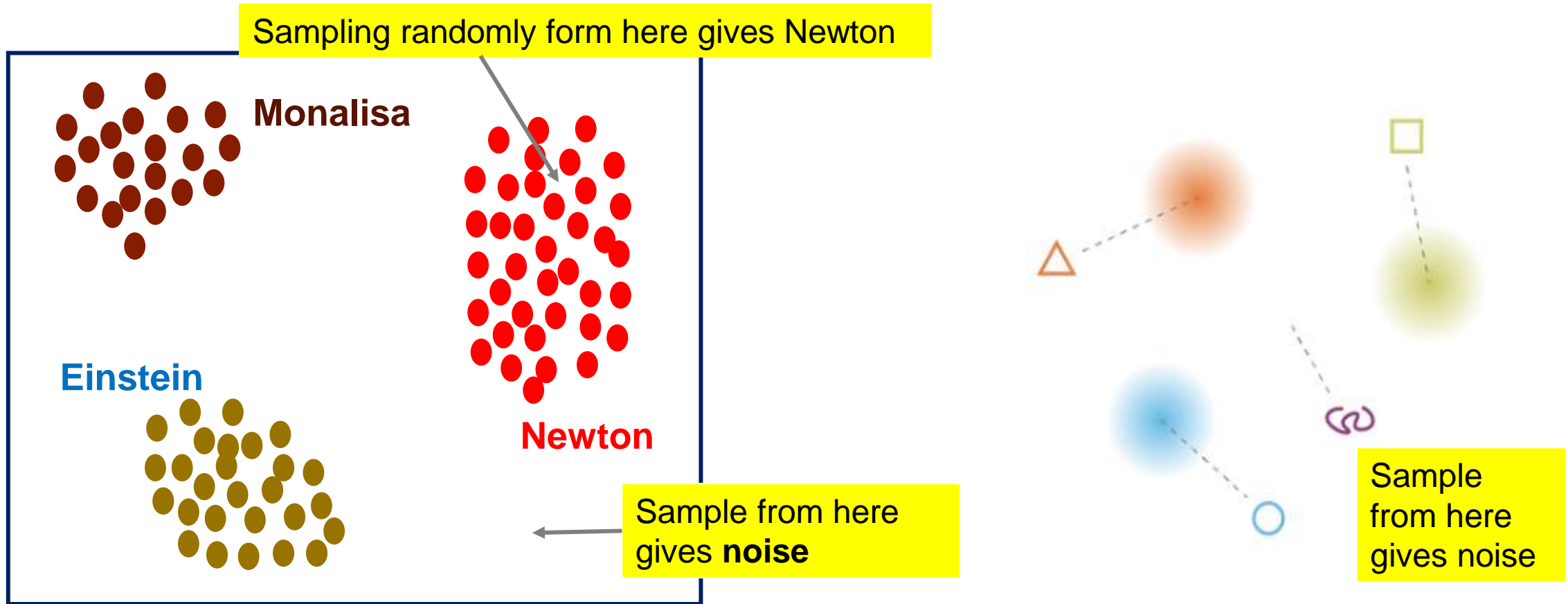
- Input to decoder is sampled from the probability distribution of the latent variable
- generate **new data** that is similar to but not direct reconstructions of the input data

<https://www.youtube.com/watch?v=BUNi0To1IVw> (start from 17:19)

<https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73>

Variational Autoencoder

Latent Distribution for Autoencoder

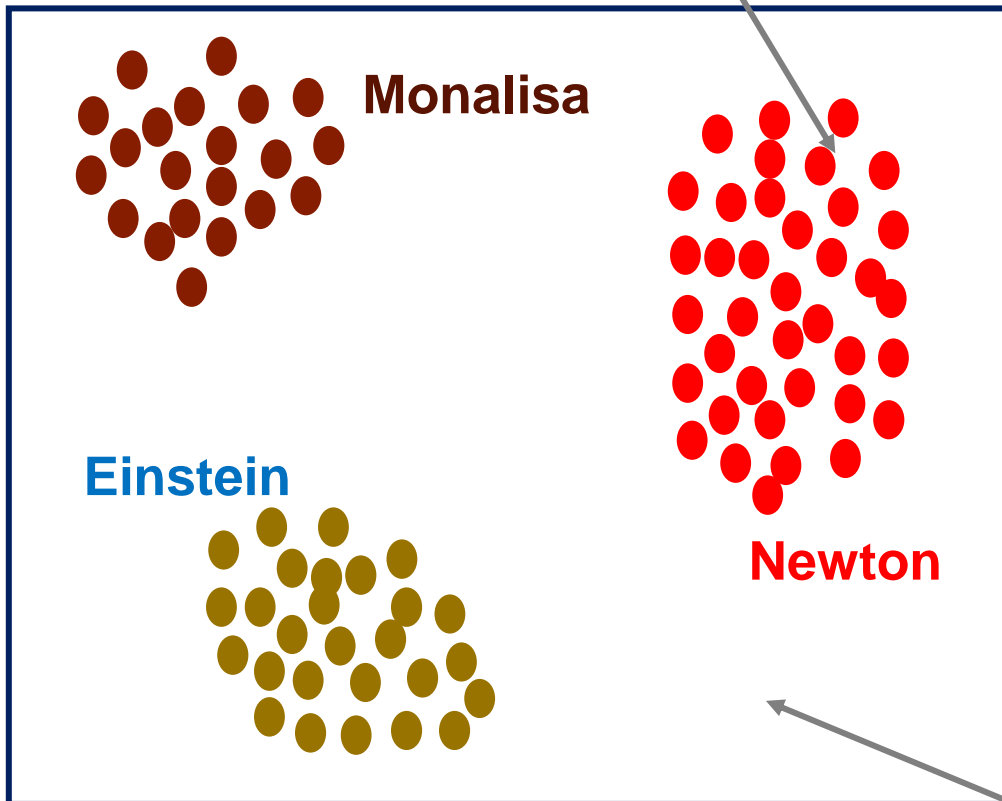


If we just randomly sample, then all the reconstructed value may mean nothing.

Variational Autoencoder

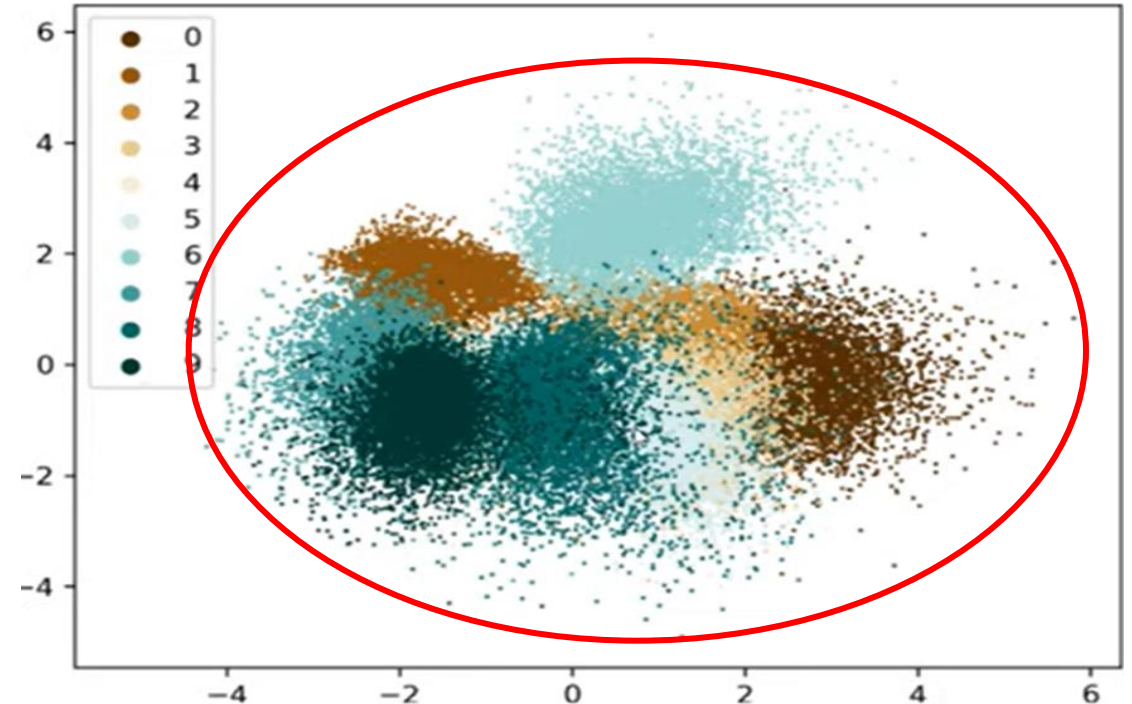
Latent Distribution for Autoencoder

Sampling randomly from here gives Newton



Sample from here gives noise

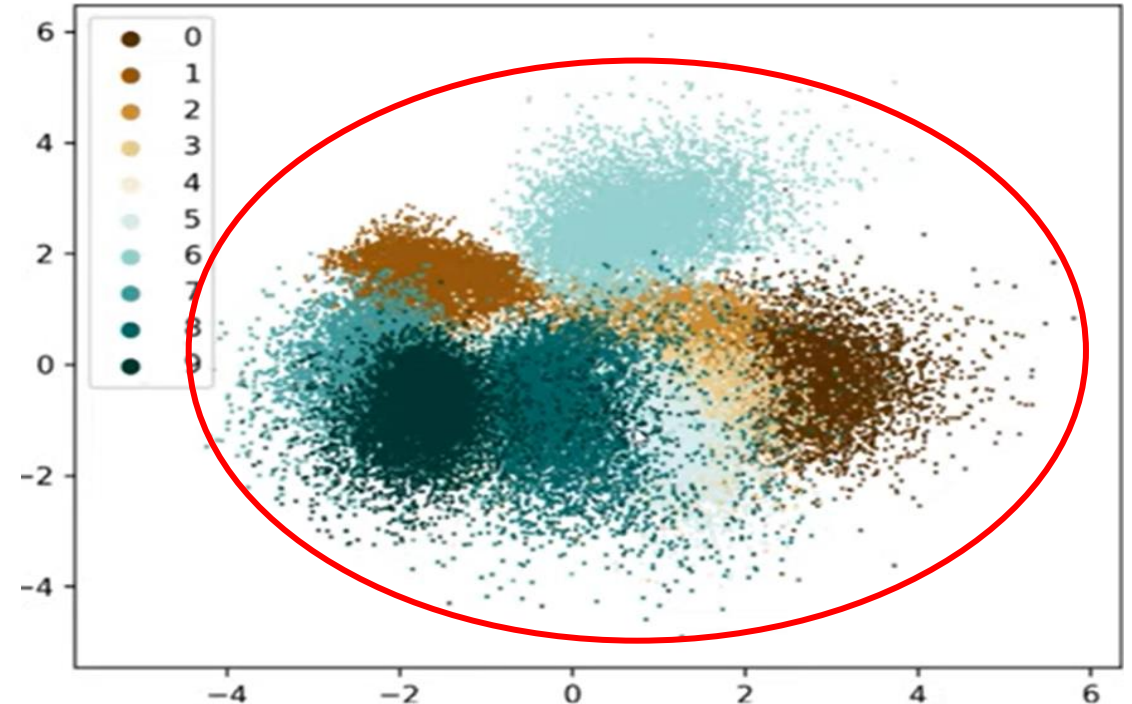
Constrain latent vector values to a specific continuous region



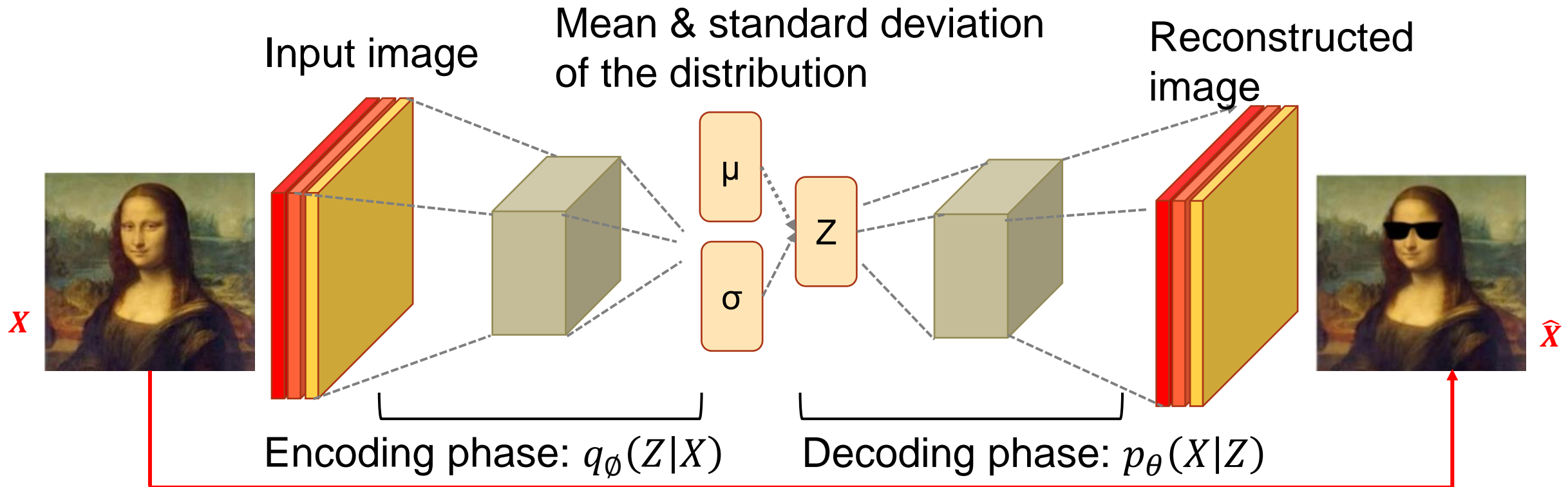
Variational Autoencoder

- Instead of mapping into a fixed latent vector, we map it to a **distribution**.
- **We constrain these latent variable to be normally distributed**
- Instead of passing the encoder output to the decoder, use **mean** and **standard deviation** describing the distribution.

Constrain latent vector values to a specific continuous region



Reconstruction Loss



$$\text{Loss function} = \underbrace{\|X - \hat{X}\|^2}_{\text{Reconstruction loss}} + \underbrace{D(q_{\phi}(Z|X) \| p(Z))}_{\text{KL divergence loss}}$$

Quantify the difference between the input and the reconstructed output

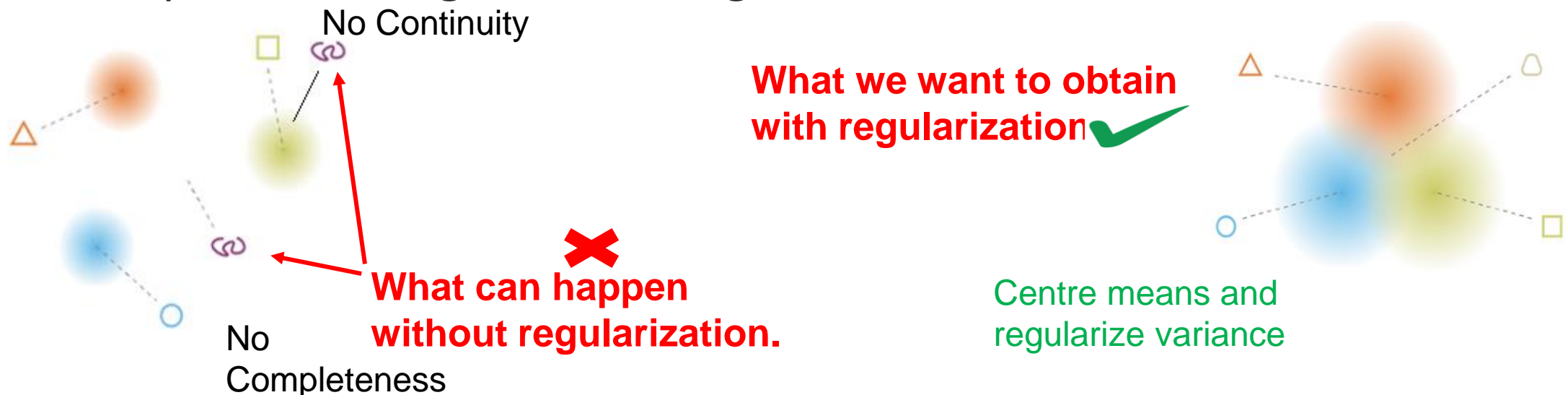
Quantify the distance between learned distribution and standard norm. distr. using Kullback-Leibler (KL) divergence

ϕ : weights of Encoder
 θ : weights of Decoder

Intuition on Regularization and Normal Prior

Which characteristics do we hope the regularization will produce?

- **Continuity:** Two close points in the latent space should not give two completely different contents once decoded.
- **Completeness:** For a chosen distribution, a point sampled from the latent space should give “meaningful” content once decoded



Variational Autoencoder

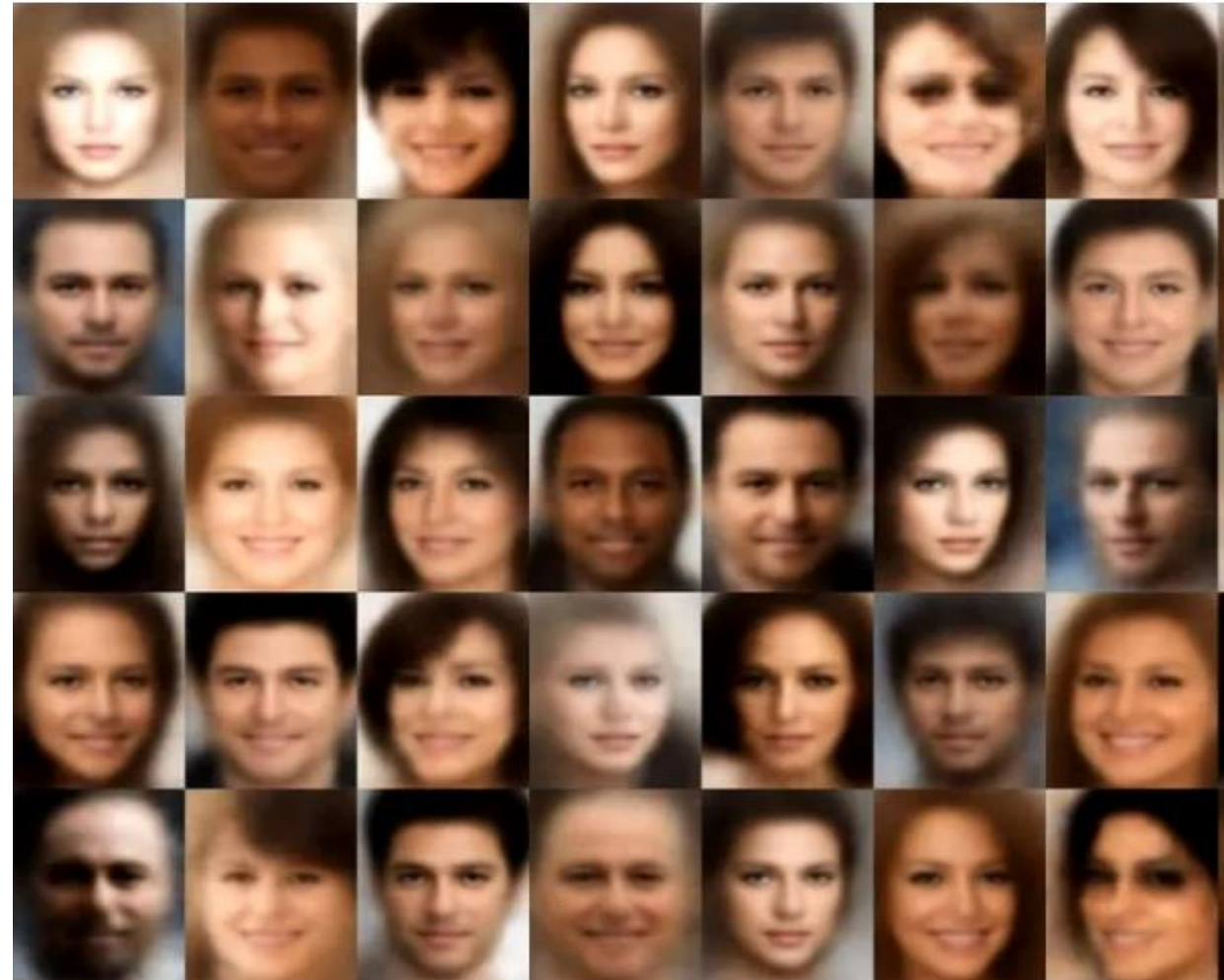
Advantages:

- ✓ Given an X , easy to find Z .
- ✓ Interpretable probability $P(X)$.

Disadvantages:

- ✗ Usually outputs blurry images.

Face images generated with a
Variational Autoencoder



Ref. <https://github.com/wojciechmo/vae>

So far we learned...

- Autoencoders for Representation Learning
- Latent Variables
- Types of Autoencoders
- Applications of Autoencoders
- Variational Autoencoders

Module III (20%)

Generative Adversarial Networks

- Brief overview of Generative Models,
- Introduction to Autoencoders and Variational Autoencoders,
- **Generative Adversarial Networks (GANs)** for Realistic Data Synthesis,
- Conditional Generative Models for Controlled Data Generation,
- Dealing with common issues like mode collapse and instability,
- Use cases of GANs in image synthesis and data augmentation

What are Generative Adversarial Networks?

Generative

Generates data
(Creates fake data)



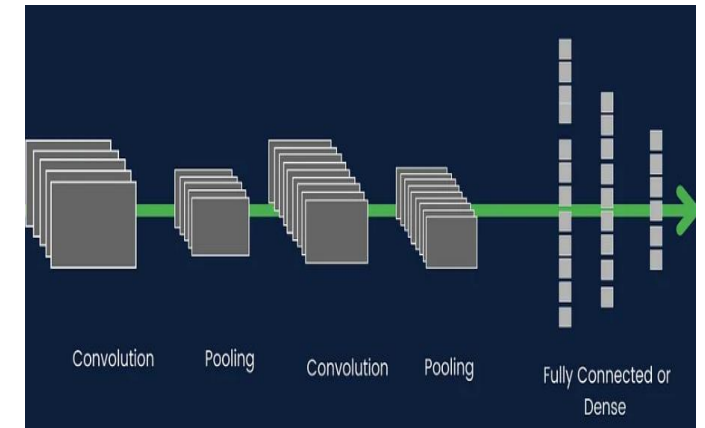
Adversarial



Generator and
Discriminator, each
competing to win.

Generator trying to fake, and
Discriminator trying not be fooled.

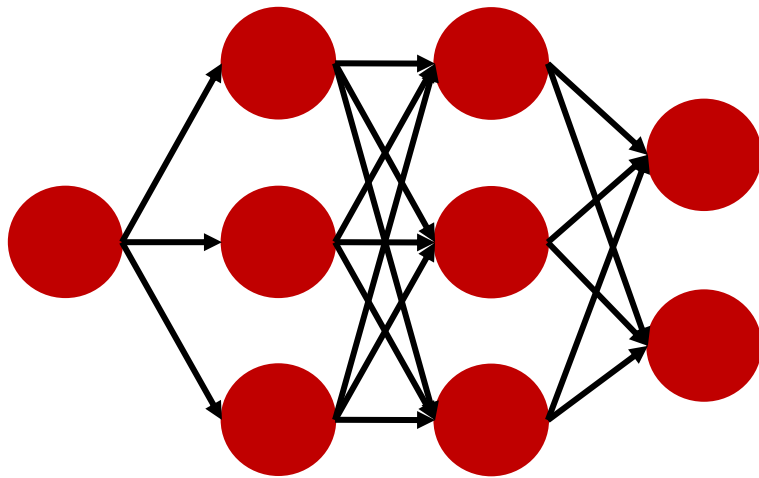
Networks



Deep Convolutional
Network

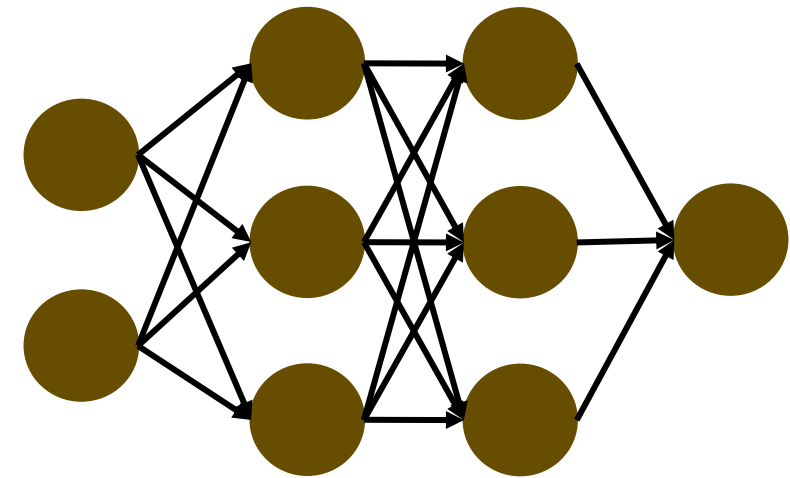
What are Generative Adversarial Networks?

Generative Adversarial Network is a two-player game



Generator

Try to fool the Discriminator by generating the real-looking samples.

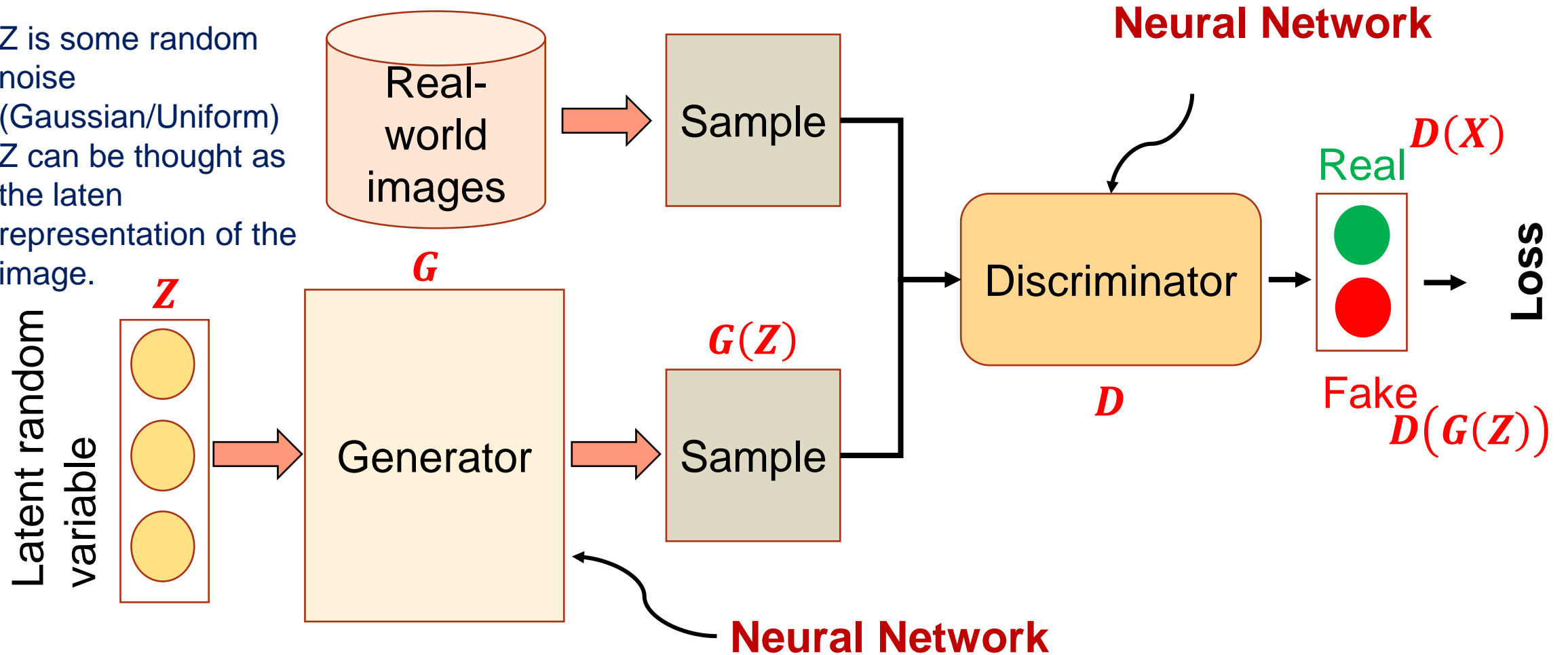


Discriminator

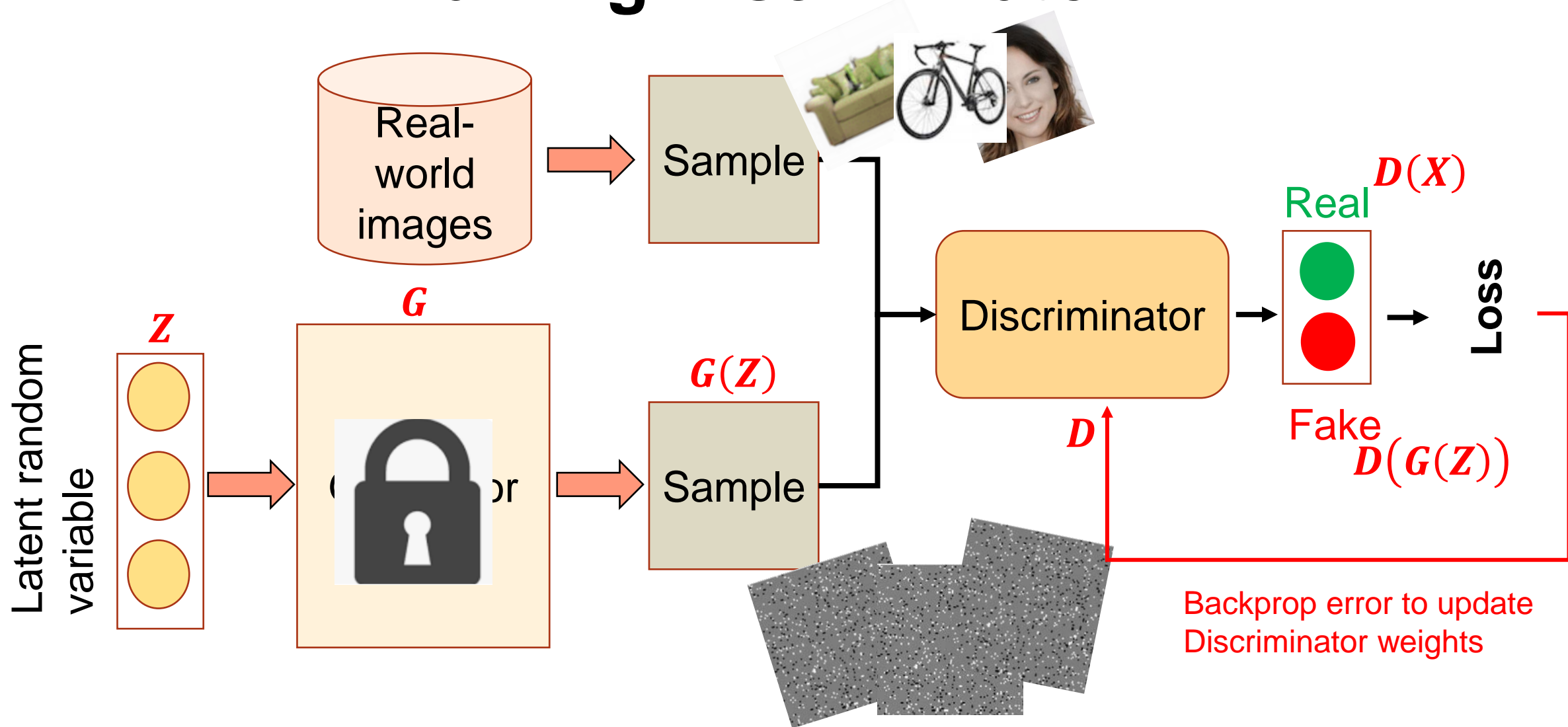
Try to distinguish between real and fake samples.

GAN's Architecture

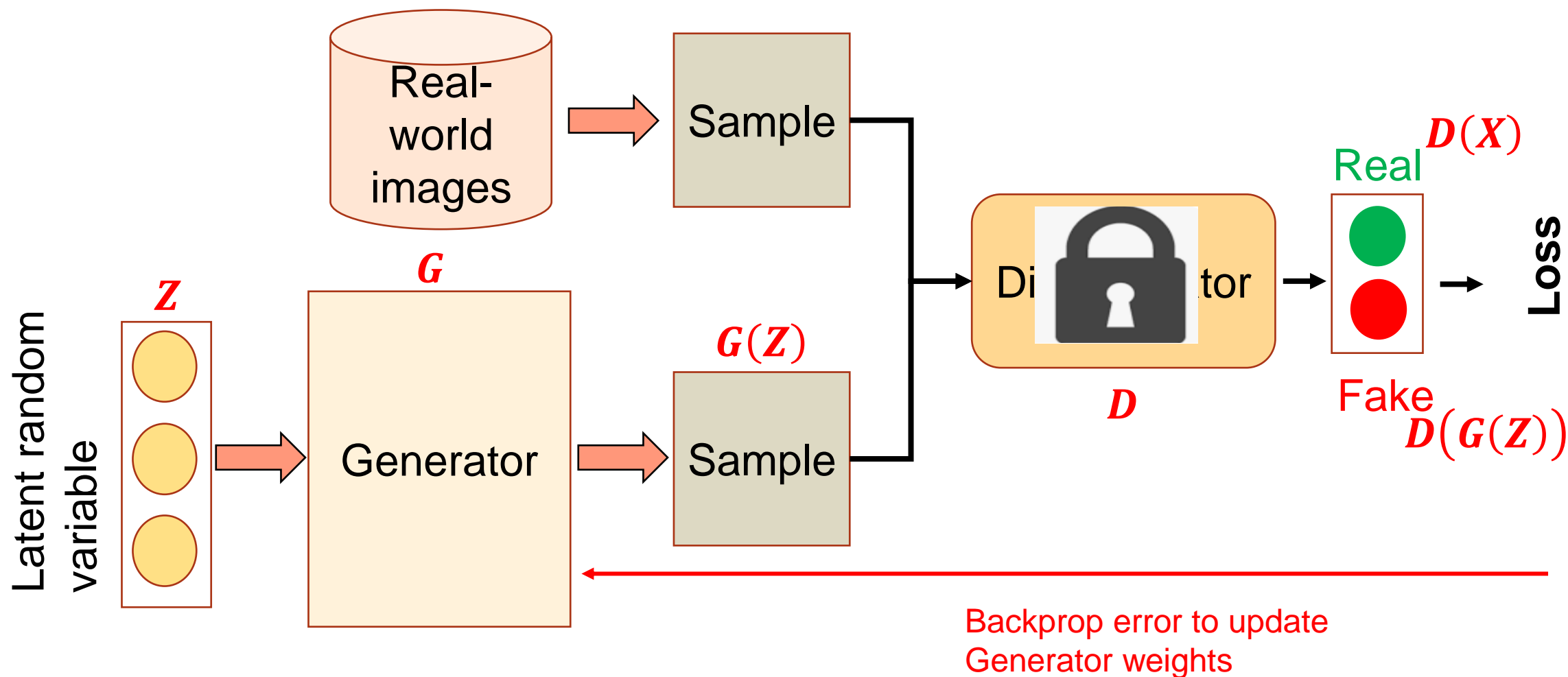
- Z is some random noise (Gaussian/Uniform)
- Z can be thought as the latent representation of the image.



Training Discriminator



Training Generator



Generative Adversarial Networks Steps

Steps of Generative Adversarial Networks:

- Define the GAN architecture based on the application.
- **Train Discriminator** to distinguish real vs fake.
- **Train the Generator** to fake data that can fool the Discriminator.
- **Continue** Discriminator and Generator **training** for multiple epochs.
- **Save Generator** model to create new, realistic fake data.

Loss Function of GANs

Where,

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

$$V(D, G) = E_{X \sim p_{data}(X)} [\log D(X)] + E_{Z \sim p_Z(Z)} [\log 1 - D(G(Z))]$$

Expectation over data of log of D given X.

$\log D(X)$ is the discriminator output for real data X

$E_{X \sim p_{data}(X)} [\log D(X)] \Rightarrow$ likelihood of real data being real from the data distribution

p_{data}

Expectation of Z drawn from p_Z

$Z \sim p_Z(Z)$ means samples from generator network

$D(G(Z))$ is the output of our discriminator for generated fake data

$E_{Z \sim p_Z(Z)} [\log 1 - D(G(Z))]$ \Rightarrow likelihood of generated fake data being labelled real

G = Generator

D = Discriminator

$p_{data}(X)$ = Distribution of real data

$p(Z)$ = Distribution of Generator

X = Sample from

$p_{data}(X)$

Z = Sample from $p(Z)$

$D(X)$ = Discriminator

$G(Z)$ = Generator

Loss Function of GANs

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

$$V(D, G) = E_{X \sim p_{data}(X)} [\log D(X)] + E_{Z \sim p_Z(Z)} [\log 1 - D(G(Z))]$$

Training of GAN is designed as a minimax game, using the following rules:

- Generator can't directly affect the $\log(D(x))$ term in the function, so, for the generator, minimizing the loss is equivalent to minimizing $\log(1 - D(G(z)))$
 - generator seeks to minimize the log of the inverse probability predicted by the discriminator for fake images. This has the effect of encouraging the generator to generate samples that have a low probability of being fake.
- Discriminator tries to maximize $\log D(x) + \log(1 - D(G(z)))$
 - discriminator seeks to maximize the average of the log probability of real images and the log of the inverse probability for fake images

Optimization of GANs

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

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

end for

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

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

end for

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

**Discriminator
updates**

**Generator
updates**

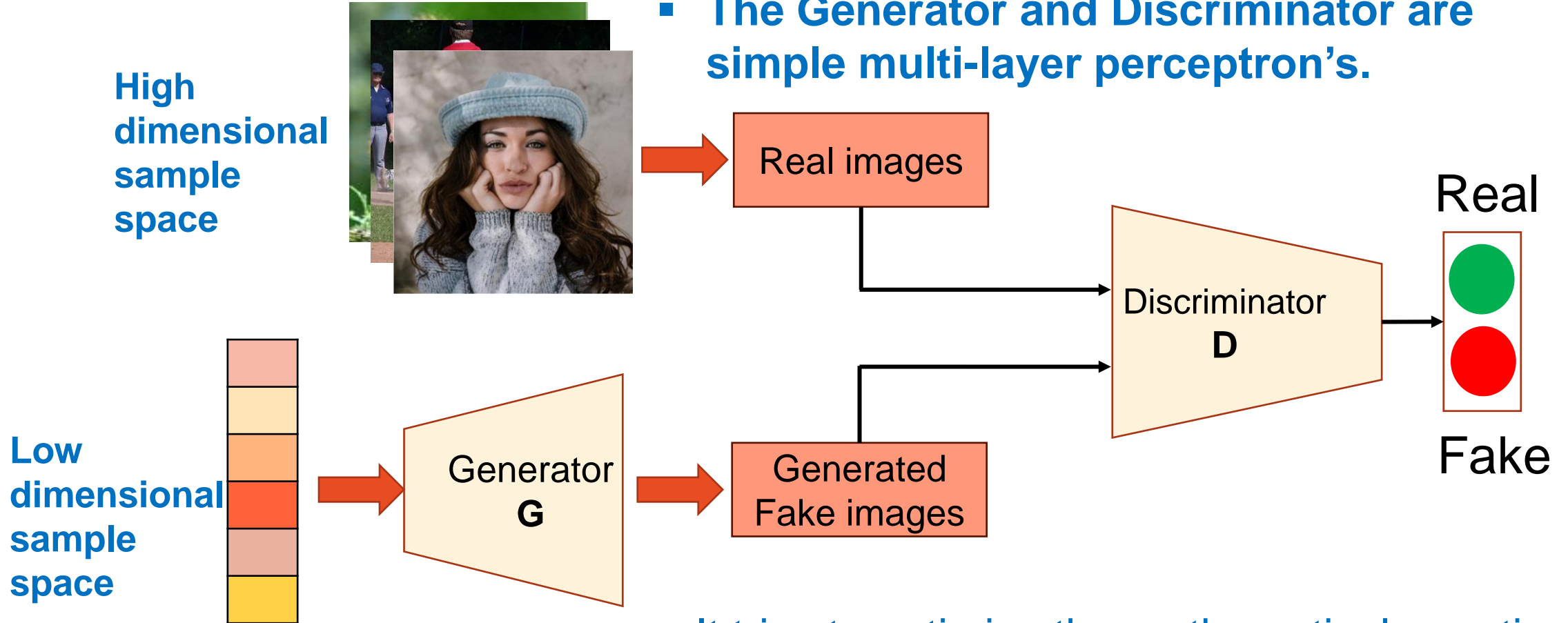
Advantages of GANs

- GANs are more powerful and simple to implement
 - Generation of real-like images is straightforward.
 - Training does not involve Maximum Likelihood Estimation.
 - Robust to Overfitting since Generator never sees the training data.
 - Empirically, GANs are good at capturing the modes of the distribution.

Types of GANs

Vanila GAN

- The Generator and Discriminator are simple multi-layer perceptron's.

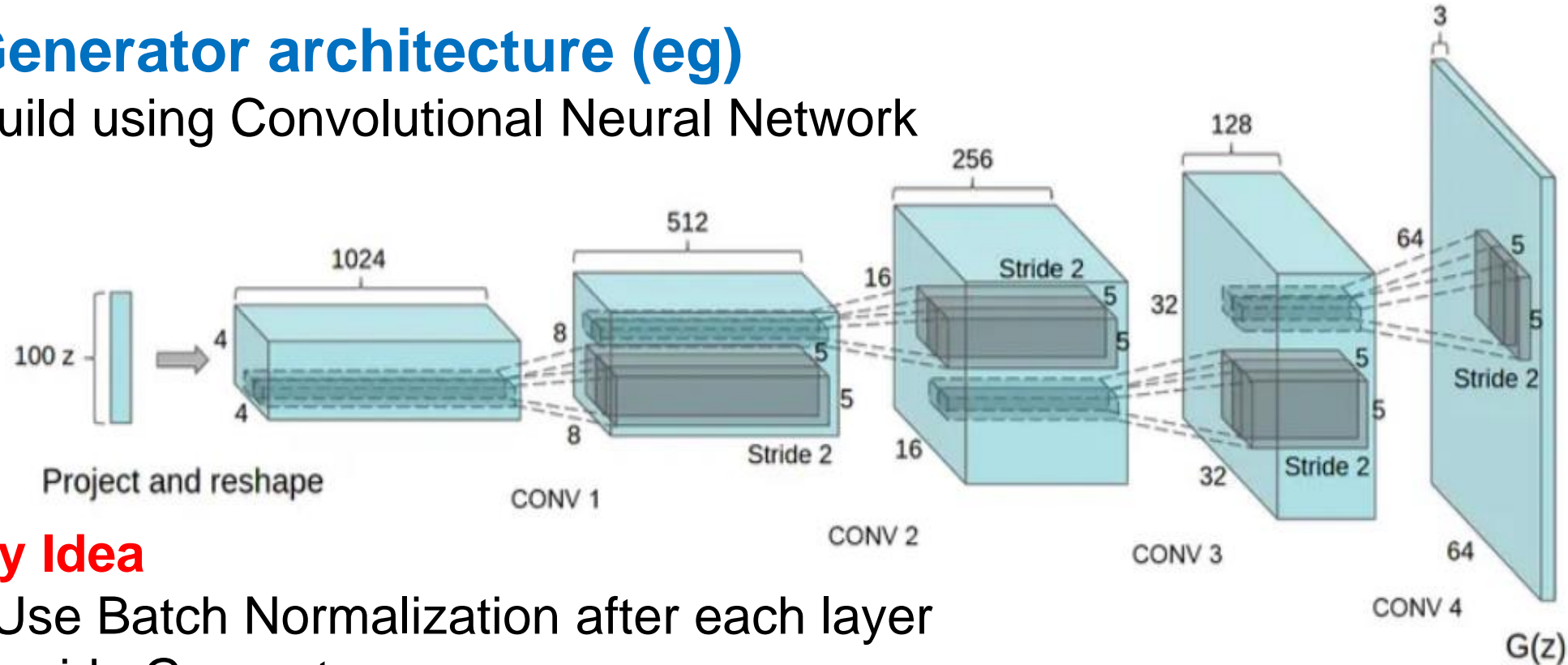


- It tries to optimize the mathematical equation using stochastic gradient descent.

Deep convolutional GAN (DCGAN)

Generator architecture (eg)

Build using Convolutional Neural Network



Key Idea

- Use Batch Normalization after each layer
- Inside Generator
 - Use ReLU for hidden layers
 - Use Tanh for the output layer

Advantages of DCGAN

- DCGAN can generate high-quality images with realistic textures and details, which is useful for many applications such as **image generation**, **style transfer**, and **data augmentation**.
- DCGAN can learn complex representations of the input data.
(which allows it to capture the underlying structure and patterns of the dataset, and generate new samples that are consistent with this structure)
- DCGAN is relatively easy to implement and can be trained on large datasets with parallel computing, which reduces the training time and memory requirements.

Disadvantages of DCGAN

- DCGAN is prone to **mode collapse**.
(which occurs when the generator produces a limited set of samples that do not cover the full range of the input space. This can result in the generator producing repetitive or unrealistic samples.)
- DCGAN requires a large amount of high-quality data for training, which can be expensive and time-consuming to obtain.
- DCGAN is sensitive to the choice of hyperparameters such as learning rate, batch size, and network architecture, which can affect the quality and stability of the generated samples.

Challenges with GANs/ DCGANs Training

- **Mode collapse**: the generator collapses which produces limited varieties of samples.

epoch 350k



epoch 400k



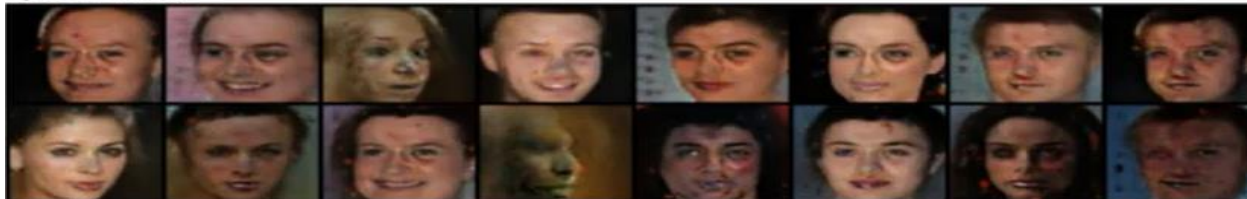
epoch 390k



epoch 480k



epoch 396k



At epoch 350k the generator producing reasonable images. While going to more epochs, generator starts producing same face every time.

Challenges with GANs / DCGANs Training

- **Mode collapse**: the generator collapses which produces limited varieties of samples.

MNIST

epoch 400k

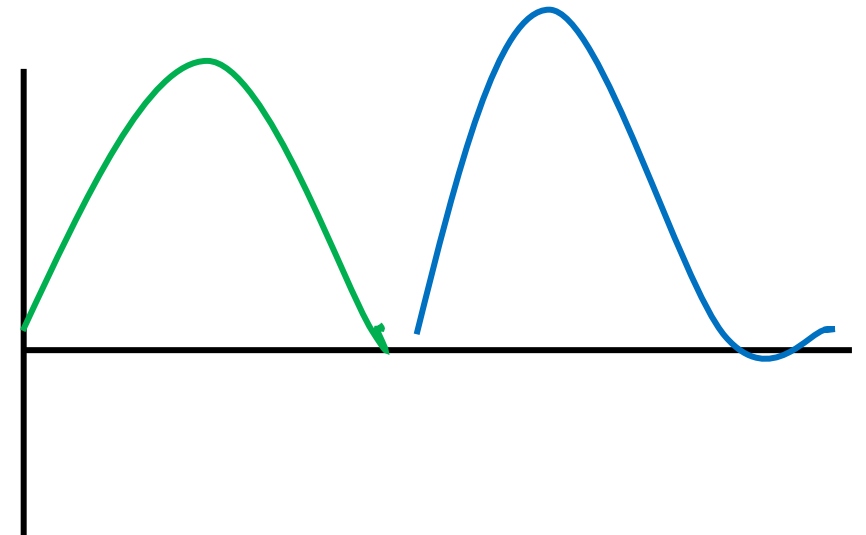


epoch 480k



0

9



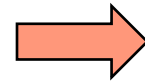
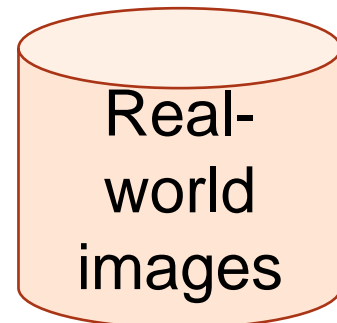
Module III (20%)

Generative Adversarial Networks

- Brief overview of Generative Models,
- Introduction to Autoencoders and Variational Autoencoders,
- Generative Adversarial Networks (GANs) for Realistic Data Synthesis,
- **Conditional Generative Models** for Controlled Data Generation,
- Dealing with common issues like mode collapse and instability,
- Use cases of GANs in image synthesis and data augmentation

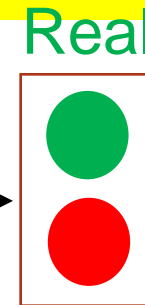
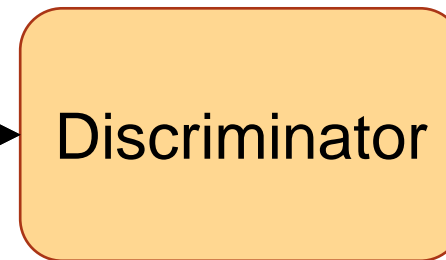
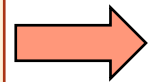
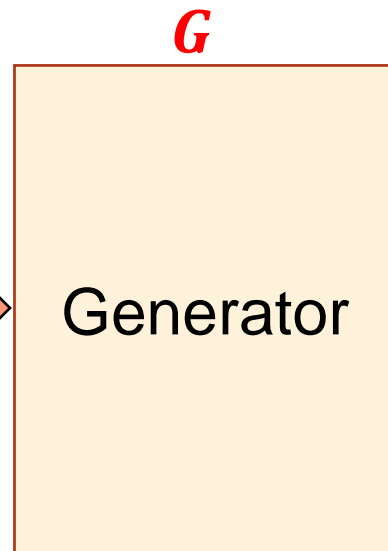
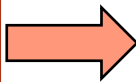
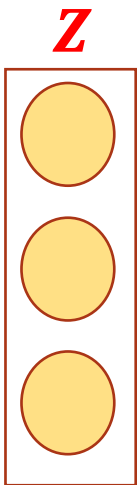
Generative Adversarial Networks

A GAN model generates a random image from the domain



The relationship between the points in the latent space and the generated images is hard to map.

Latent random variable

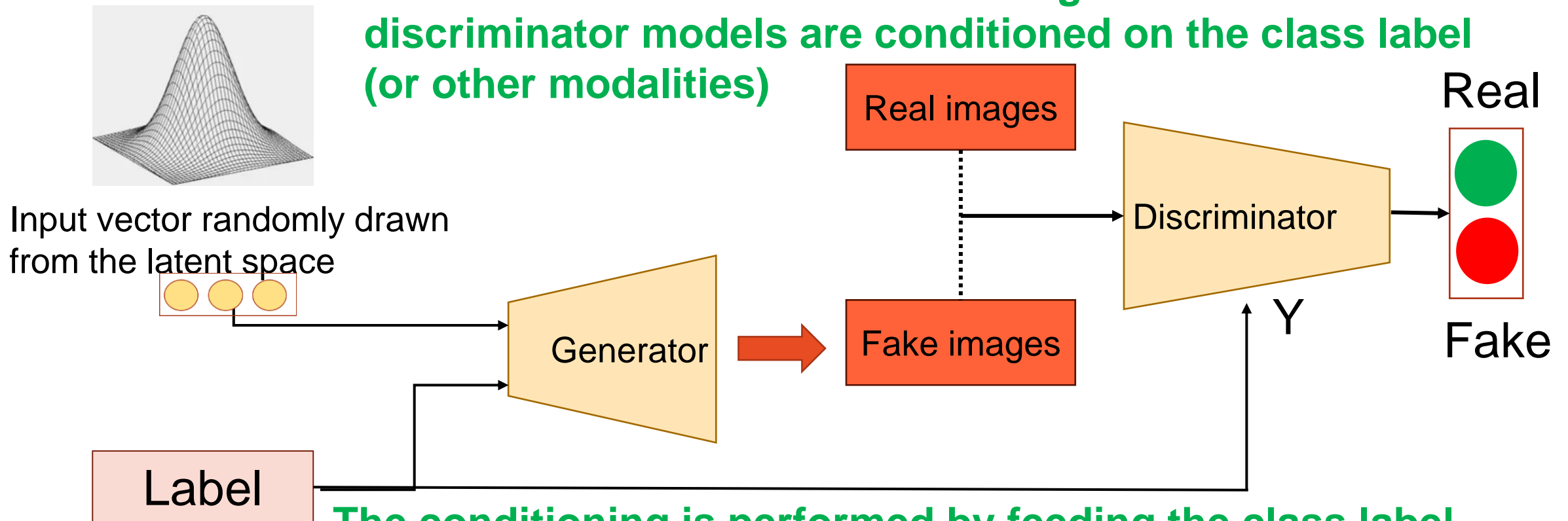


Fake
 $D(G(Z))$

Loss

Conditional Generative Adversarial Networks

A GAN can be trained so that both generator and discriminator models are conditioned on the class label (or other modalities)



The conditioning is performed by feeding the class label into both discriminator and generator with the input layer.

Conditional Generative Adversarial Networks

- Loss function consists of conditional Probabilities $D(x|y)$, $G(z|y)$, where y is the label. Label is included in both generator and discriminator.

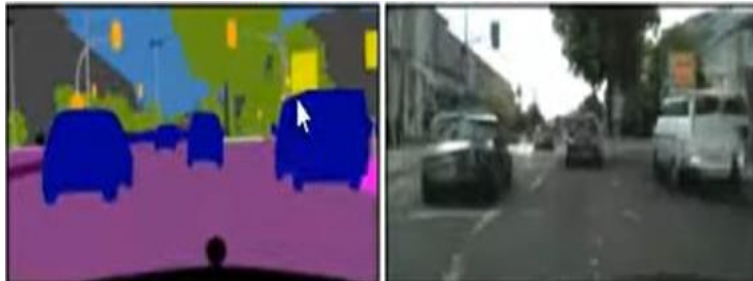
$$\min_G \max_D V(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{\text{data}}(\mathbf{x})} [\log D(\mathbf{x}|\mathbf{y})] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z}|\mathbf{y})))]$$

- It requires a large training dataset of input and its corresponding label

Conditional Generative Adversarial Networks

- The system requires **pairwise** correspondences between images during training.
- Takes an image as input and maps it to a generated output image with different properties.
 - e.g. Train CGAN to take sketches of handbags and turn them into photorealistic images of handbags.

Label to street scene



Input

Output

Day to night



Input

Output

Edges to Photo



Input

Output

Advantages of CGANs

- **Controlled generation:** CGAN can generate new data that meets specific conditions, making it useful in applications where controlled data generation is needed, such as image synthesis or text generation.
- **Diversity:** CGAN can generate multiple outputs for the same input, increasing the diversity of generated data.

Disadvantages of CGANs

- **Instability:** The training of GANs can be unstable, and it can be challenging to find the right hyperparameters and architecture for the model.
- **Computational cost:** CGAN requires a lot of computational power and time to train, making it challenging to use for larger datasets or real-time applications.

Applications of GANs in Image Generation and Computer Vision

Image to Image Translation



- convert images from one domain to another
- generator generates transformed images, while the discriminator distinguishes between real and generated images
- Through adversarial training, the generator learns to produce realistic images in the target domain
- applications in style transfer, image colorization, super-resolution, semantic segmentation, and day-to-night translation

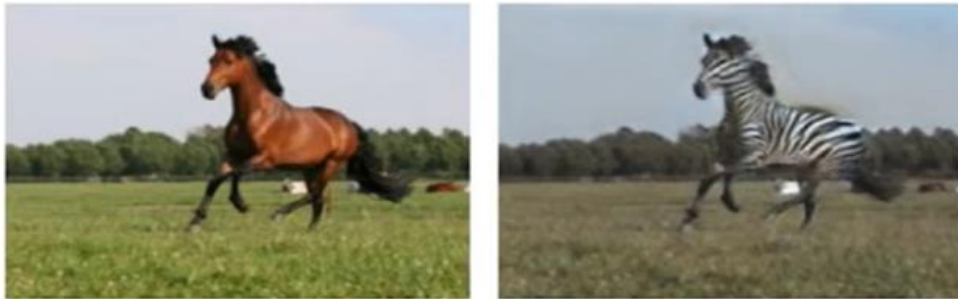
Cross-Domain Translation

Unpaired Image Translation (Zhu et al. 2017)

Zebras ↔ Horses



Zebras → Horse



Horse ← Zebra

Summer ↔ Winter



Summer → Winter



Winter → Summer

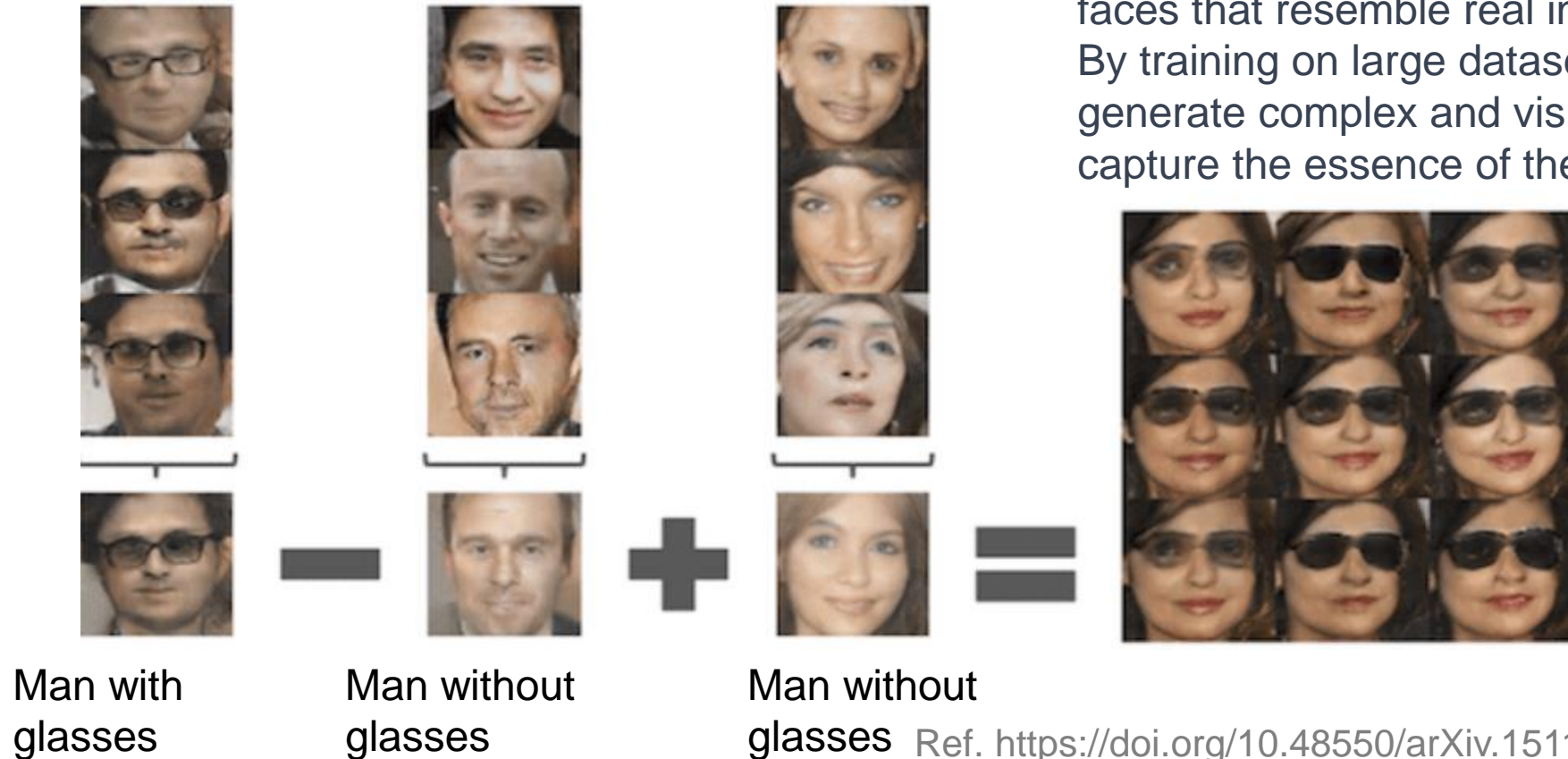
Train a generator to convert images while preserving essential characteristics

Image Generation

Generate examples of image dataset

Generator learns to produce new, never-seen-before faces that resemble real individuals

By training on large datasets, GANs can learn to generate complex and visually appealing images that capture the essence of the training data



Synthetic Human Face Generation



Generation of realistic human faces using GANs

Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

Cartoon Character Generation



(a)

(b)



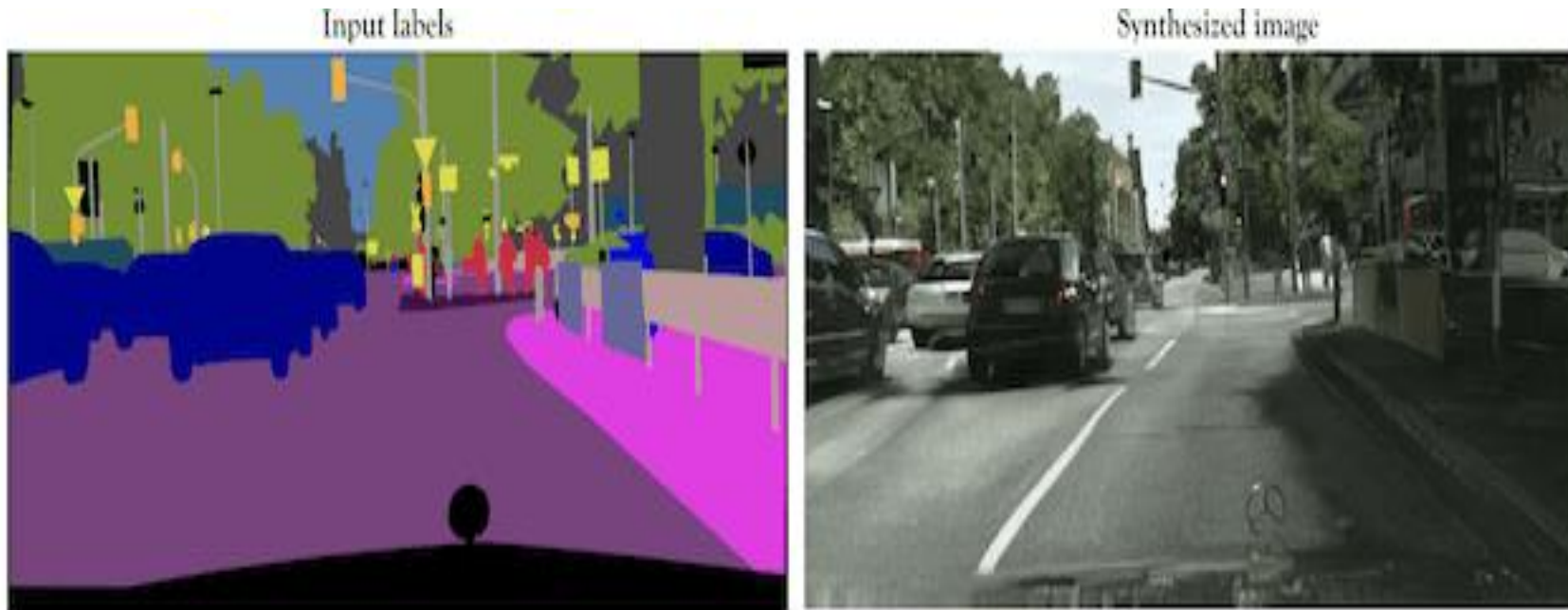
(c)

(d)

Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

GANs can be trained to generate new, unique cartoon characters that exhibit various styles, traits, and expressions

Semantic Image to Realistic Image Translation



Convert images that are labeled with semantic information, such as object classes or scene segmentation, into visually realistic representations

Side View to Frontal View Generation



Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

GANs that aims to transform images captured from a side perspective into visually realistic frontal view representations. This technology allows for the automatic generation of frontal view images, even when only side view images are available.

Emoji Generation



Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

Training a GAN on paired datasets of human face images and corresponding emoji images, the generator network can learn to map facial features to emoji expressions

Generating Filtered Images

Real image



Reconstructed images



Blonde ↑

Bangs ↑

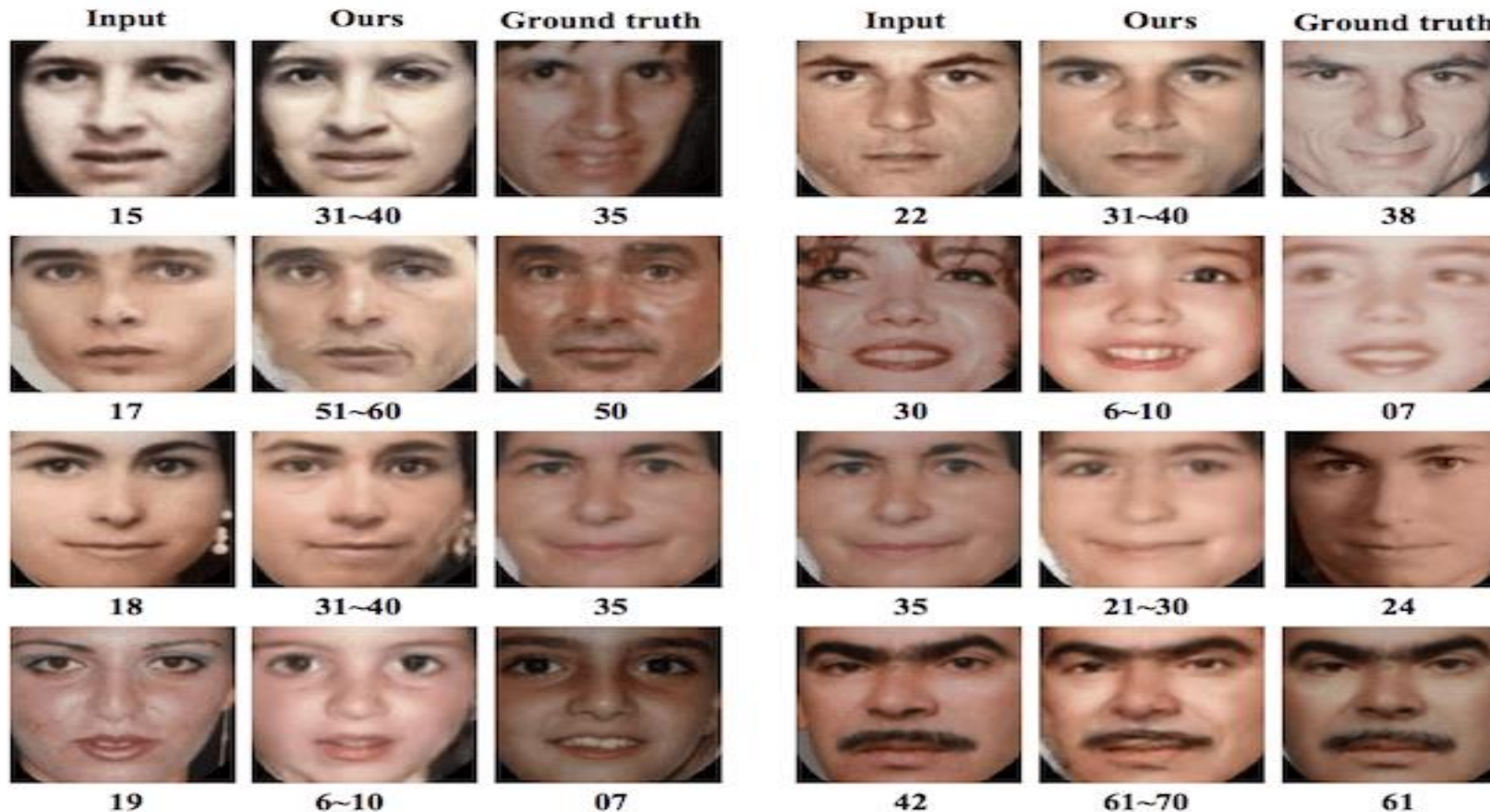
Smile ↑

Male ↑

Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

The generator is conditioned on additional input information, such as class labels or desired filter characteristics

Face Aging



Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

training GANs model to transform the appearance of a given face to simulate the effects of aging

Photo Blending



(a)



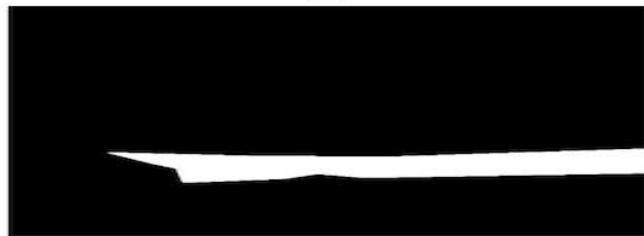
(b)



(c)

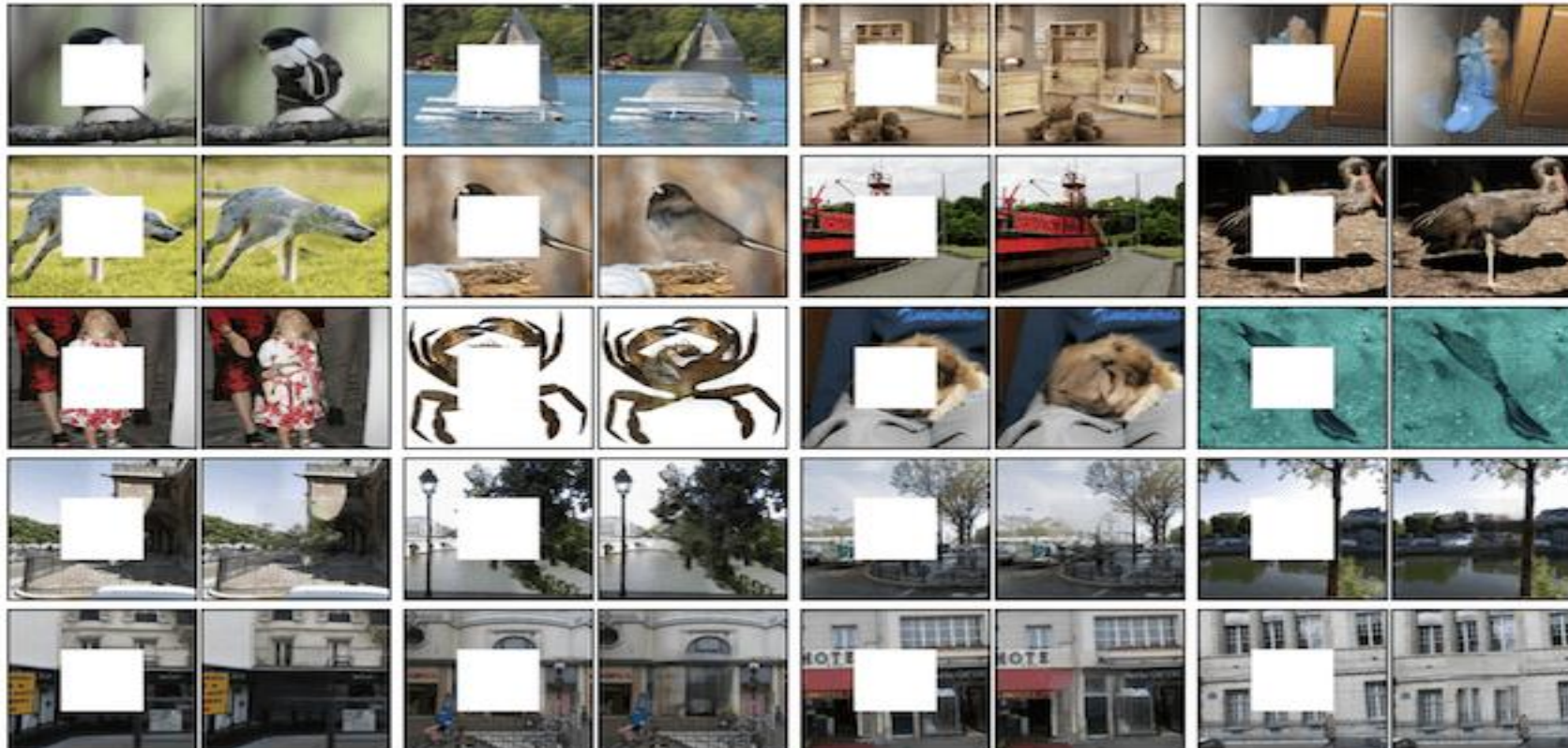


(d)



Allows seamless and realistic integration of multiple images.

Photo Inpainting



Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

Automatic filling of missing or damaged regions in images with visually coherent and plausible content.

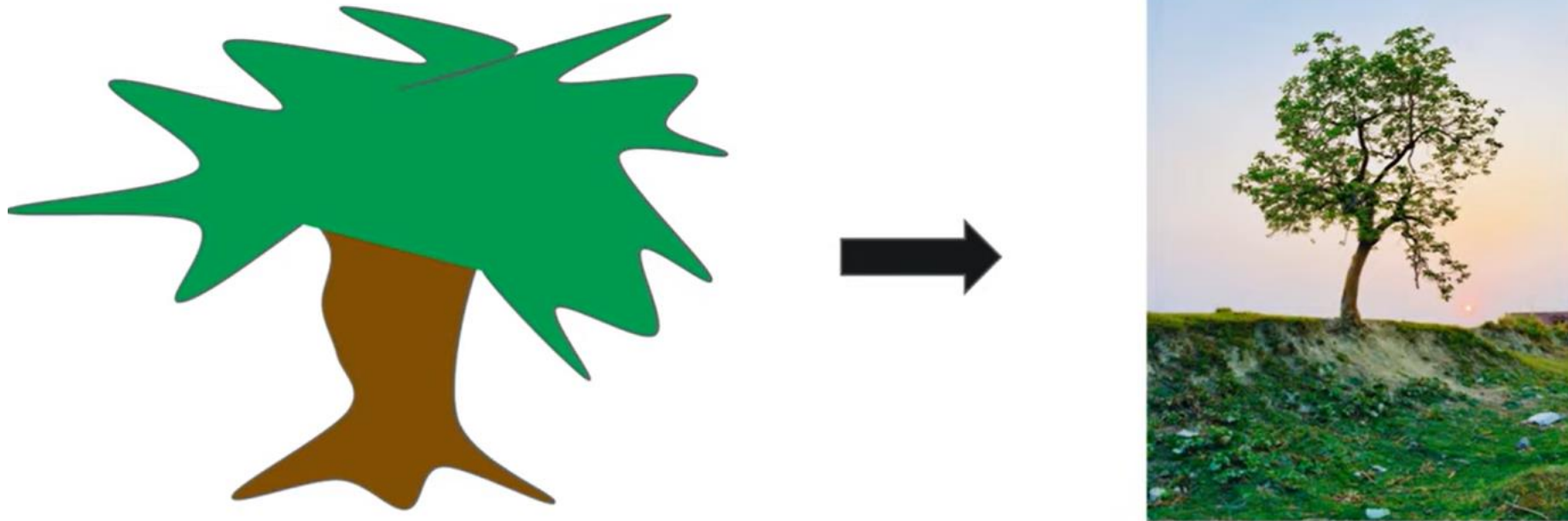
Clothing Translation



transformation of clothing items from one style to another

Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

Interactive Image Generation



Enables users to actively participate in the image synthesis process. GANs allow users to have control and influence over the generated images through interactive inputs, such as sketches, textual descriptions, or spatial masks, into the generation process

Ref. <https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/>

Topics to Cover

- Working of Autoencoder, VAE, Autoencoder vs. VAE.
Applications of Autoencoder
- Working of Generative Adversarial Networks(GAN), its challenges (mode collapse) and solutions
- Applications of GANs, GANs for image synthesis
- Difference between GAN and conditional GAN