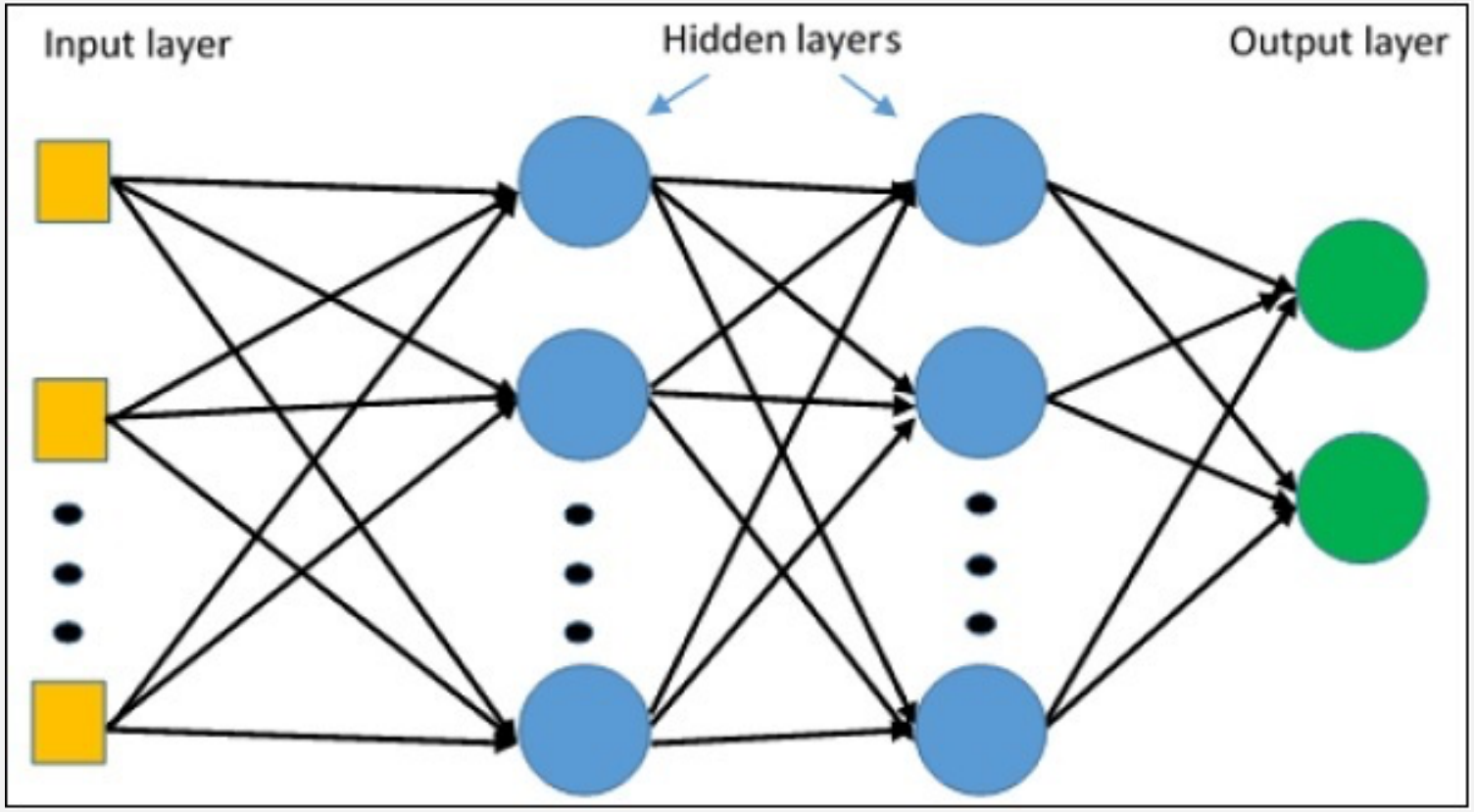
POLITECNICO
MILANO 1863

IMAGE AND SOUND
ISPG
PROCESSING GROUP

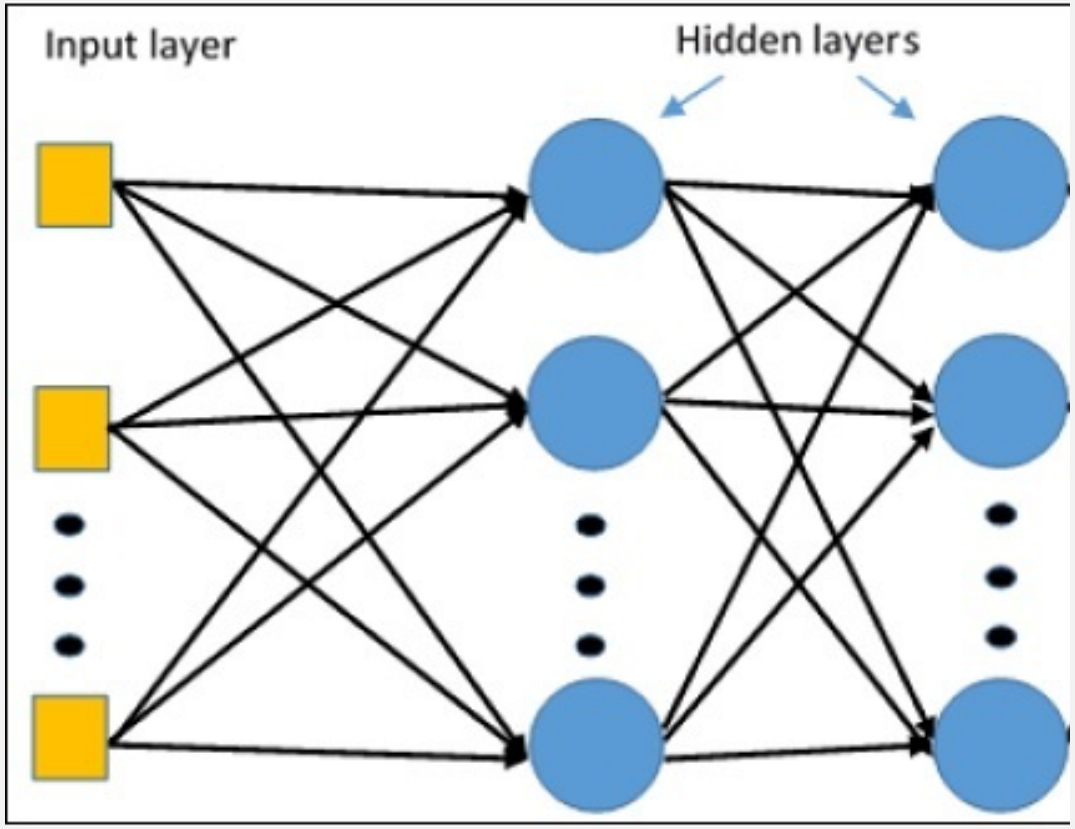
CREATIVE PROGRAMMING AND COMPUTING

Generative Neural Networks

FEATURE LEARNING

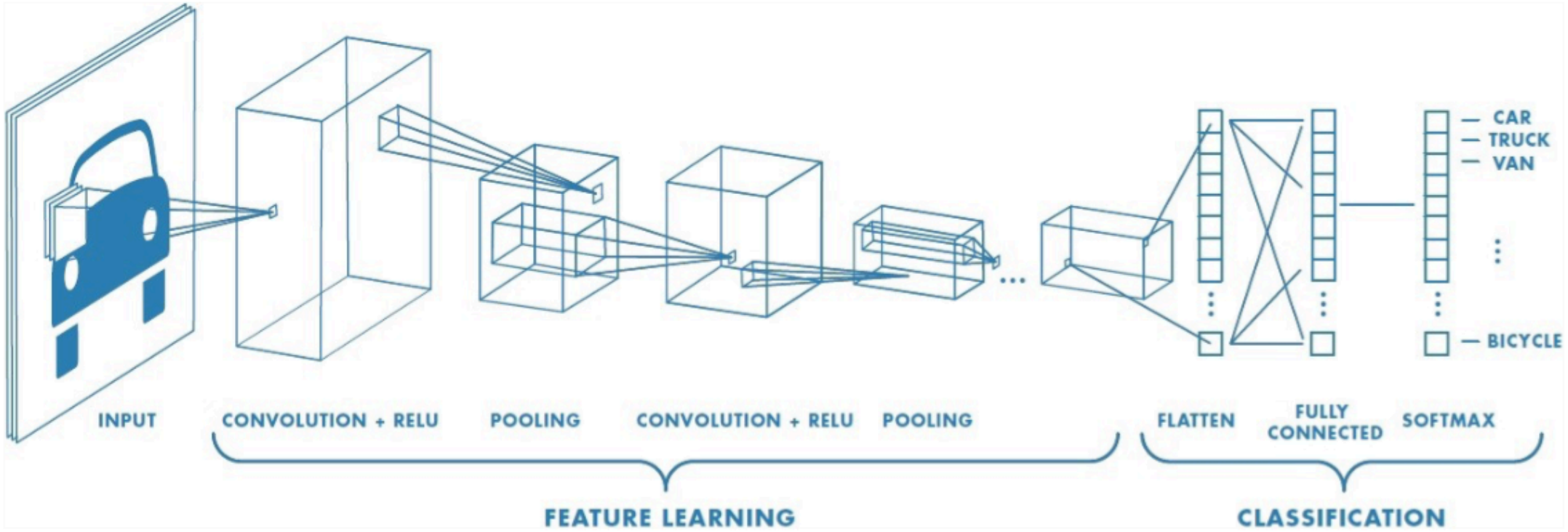


FEATURE LEARNING

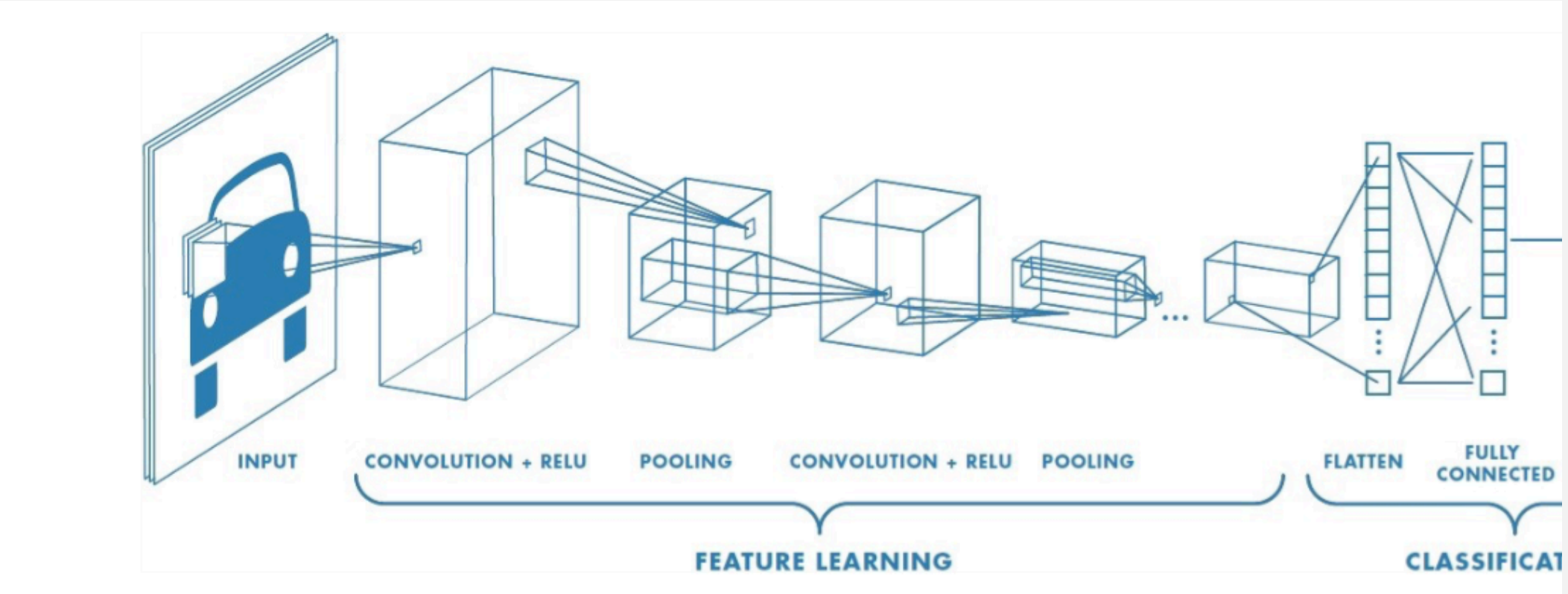


- The output of neurons in the hidden layers can be seen as a representation of the input
- Using a output layer and backpropagation in training we can learn a set of "learned features" that represents the input data
- In the test we can represent input data using a feature learned by the net with with the trained weights

FEATURE LEARNING

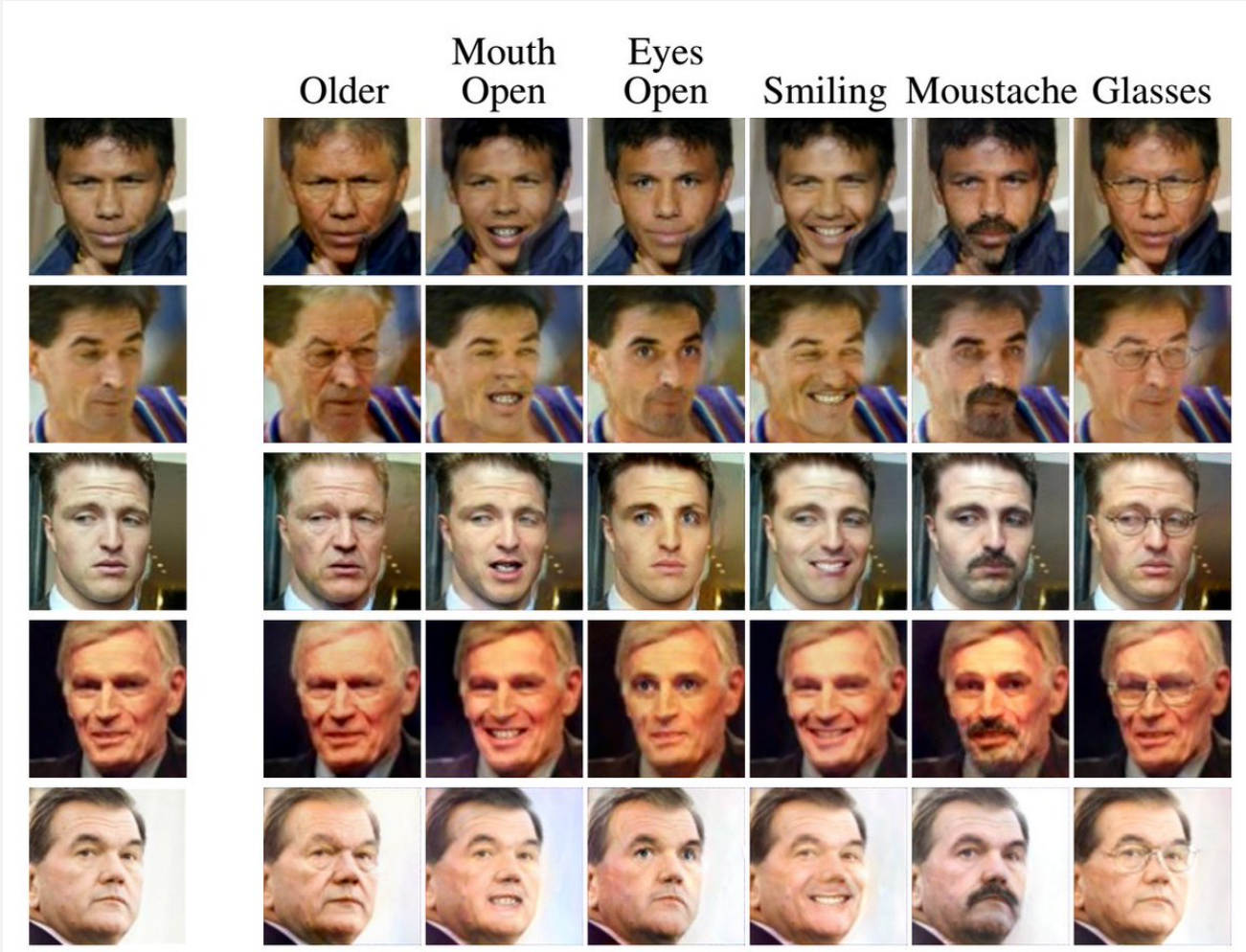


FEATURE LEARNING



GENERATIVE MODELS

- To train a generative model we first collect a large amount of data in some domain (e.g., think millions of images, sentences, or sounds, etc.) and **then train a model to generate data like it**
- Example: give 1.2 million of images, let's generate a new similar image from scratch



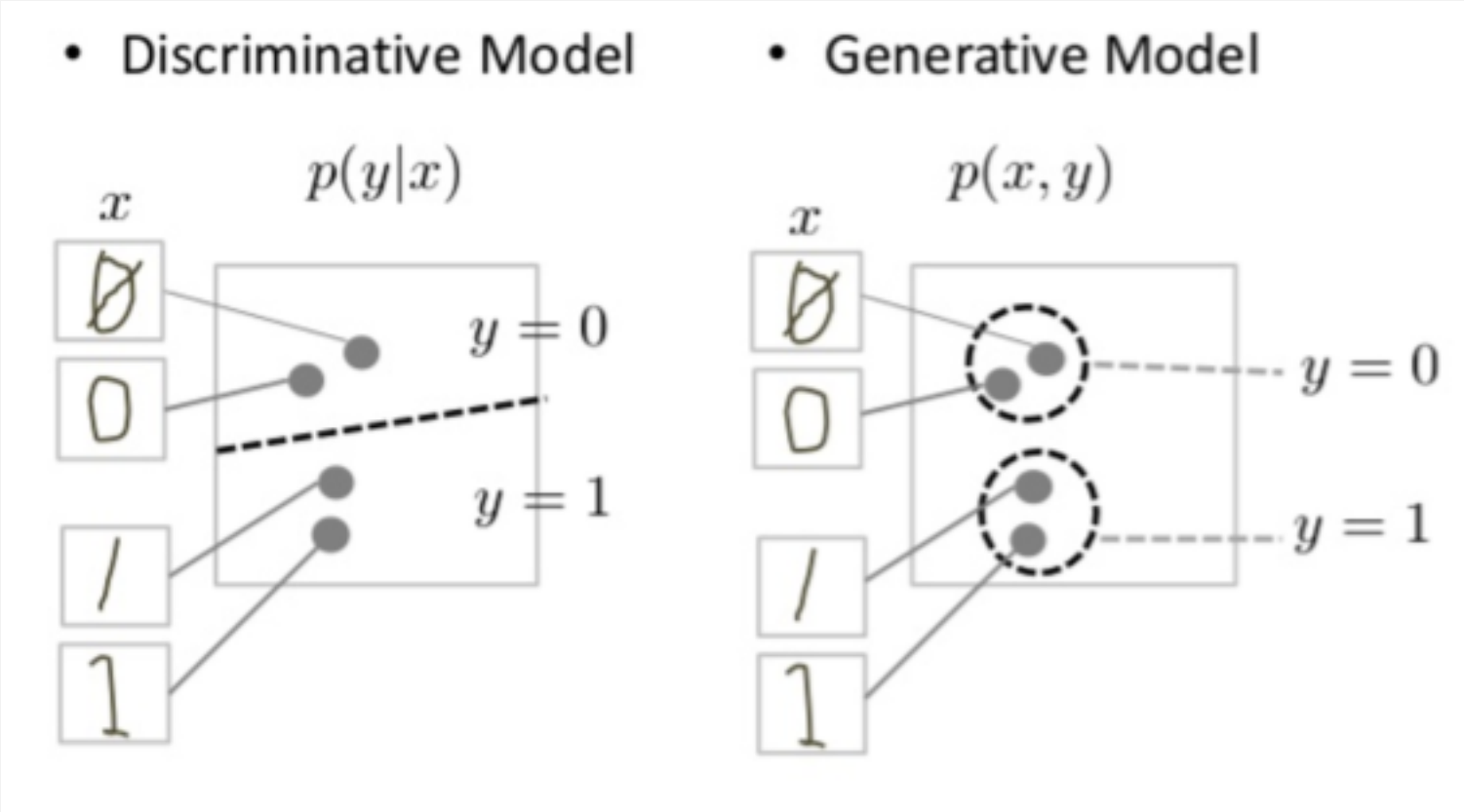
DISCRIMINATIVE VS GENERATIVE MODELS

- Discriminative VS generative models
- **Discriminative** (classification, regression, clustering)
  - discriminate between different kinds of data instances
  - learn the boundary between classes
  - map features to labels, they are concerned solely with that correlation. Instead of predicting a label given certain features, they attempt to predict features given a certain label
  - models capture the conditional probability p(Y | X)
- **Generative**
  - generate new data instances
  - model the distribution of individual classes
  - instead of predicting a label given certain features, they attempt to predict features given a certain label
  - models capture the joint probability p(X, Y), or just p(X) if there are no labels

DISCRIMINATIVE VS GENERATIVE MODELS

- The **discriminative** model tries to capture the difference between handwritten 0's and 1's by drawing a line in the data space
- No need to model exactly where the instances are placed in the data space on either side of the line.

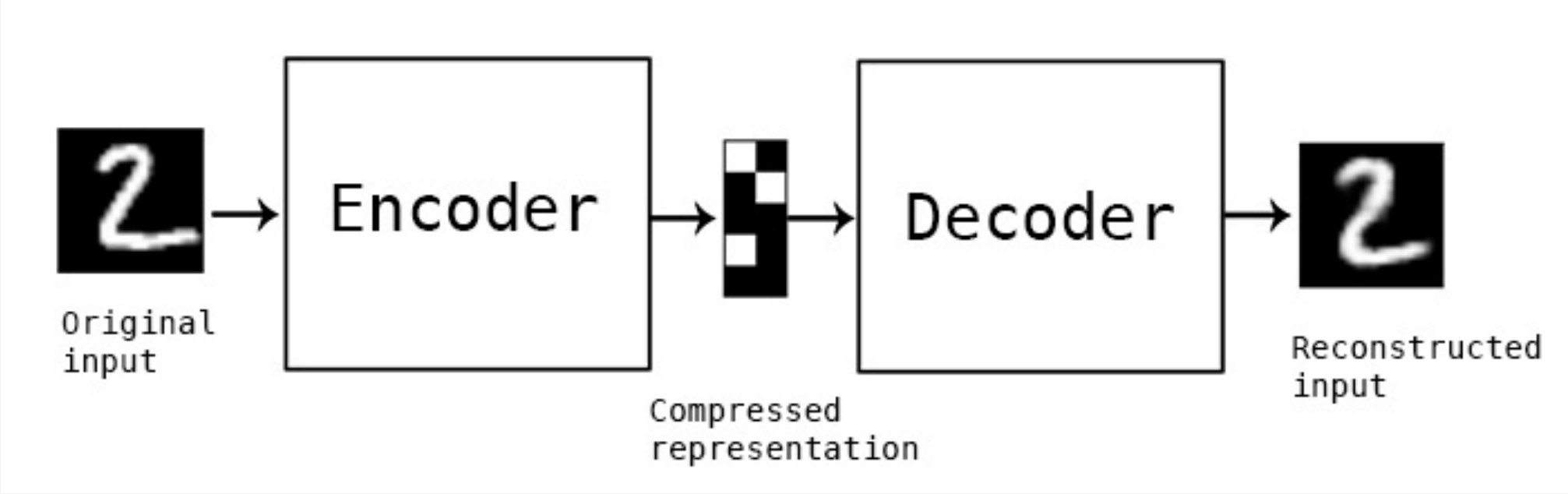
- The **generative** model tries to produce convincing 1's and 0's by generating digits that fall close to their real counterparts in the data space.
- It has to model the distribution throughout the data space



AUTOENCODERS

AUTOENCODER

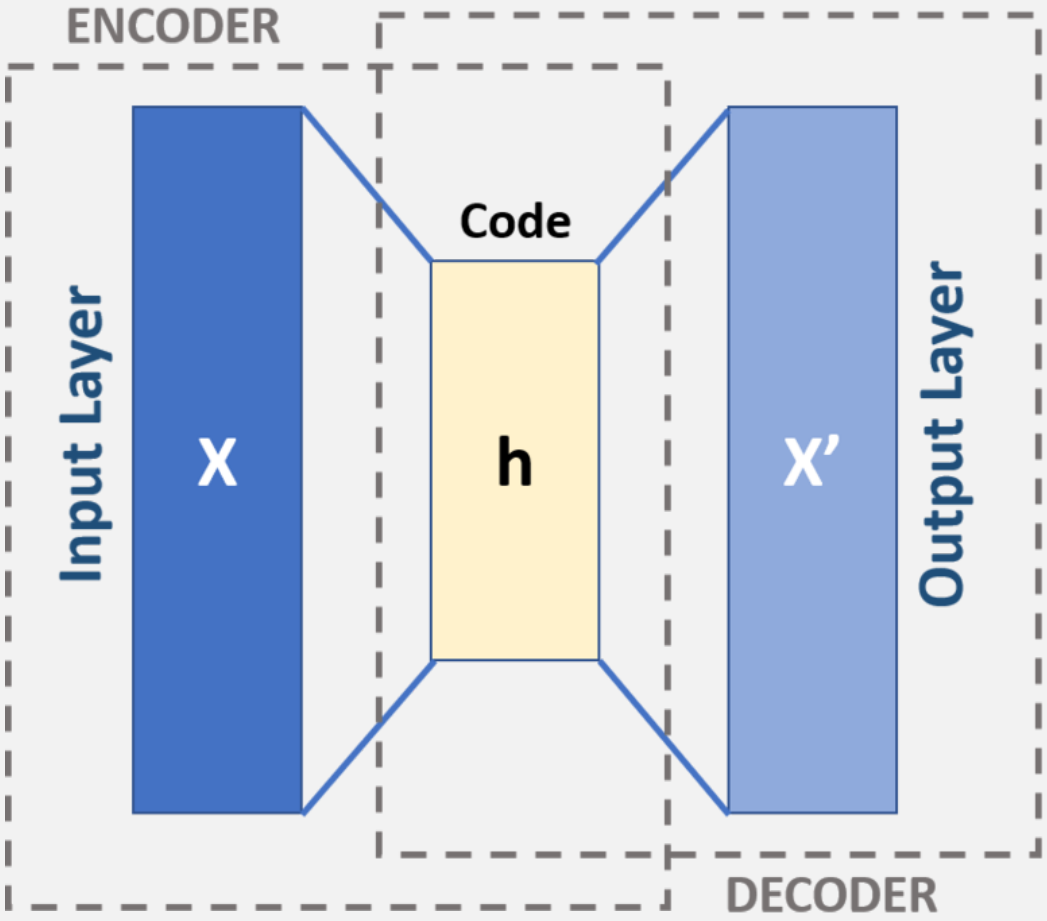
- **An autoencoder is a neural network that is trained to attempt to copy its input to its output**
- They create a hidden, or compressed, representation of the raw data
- The hidden layer produces a reduced learned (lossy) set of features able to represent the input data



AUTOENCODER

- **An autoencoder is a neural network that is trained to attempt to copy its input to its output**

- At each iteration:
  - feed the autoencoder architecture with some data
  - compare the encoded-decoded output with the initial data
  - backpropagate the error through the architecture to update the weights of the networks
- The goal is the hidden layer to ensures only the main structured part of the information



AUTOENCODER

- Given the input x
- r is the reconstruction of x
- h is the set of learned features

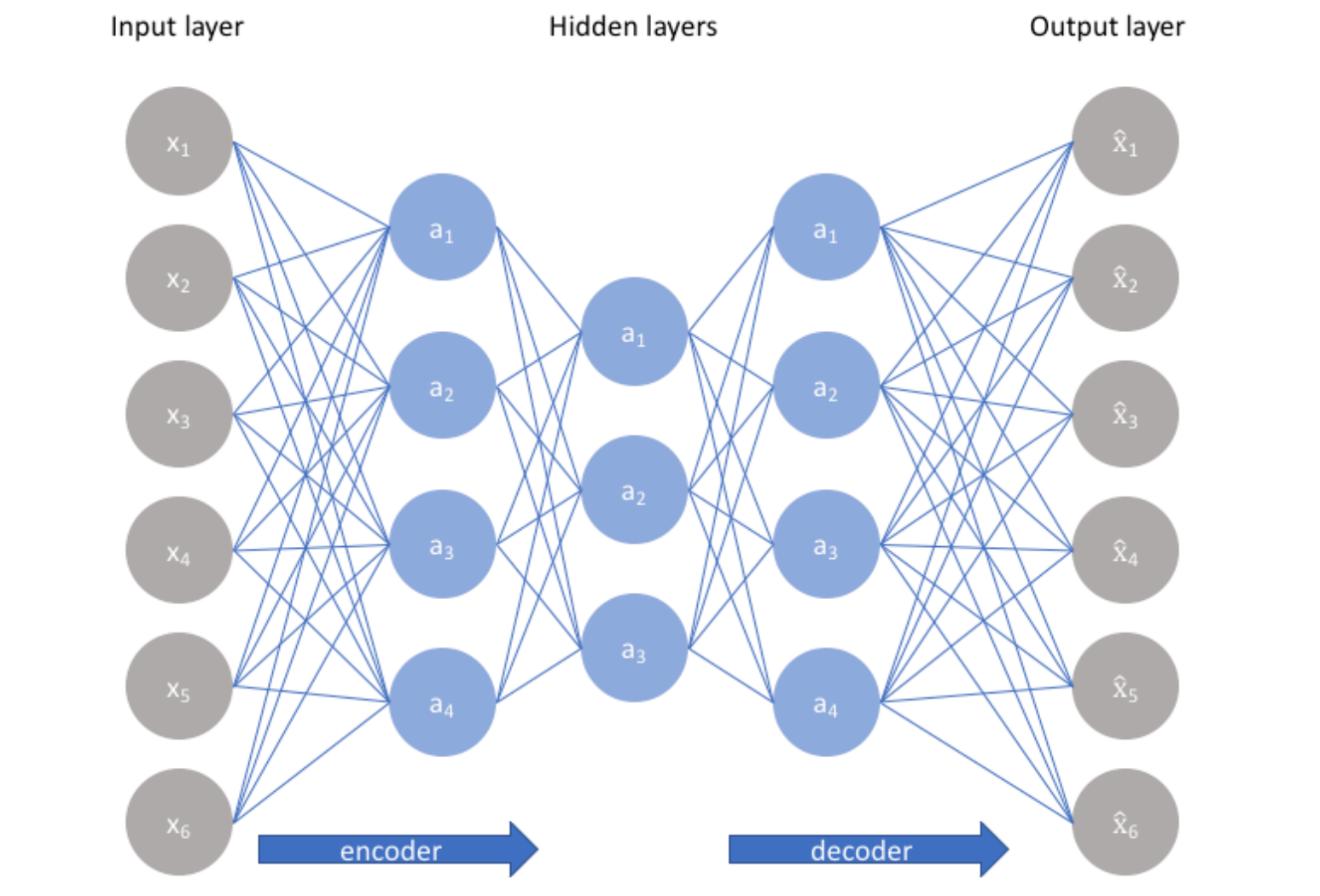
- Encoder $\boldsymbol{h} = f(\boldsymbol{x})$
- Decoder $\boldsymbol{r} = g(\boldsymbol{h})$
- Loss Function $L(\boldsymbol{x}, g(f(\boldsymbol{x})))$

- h is called **latent vector** and the feature space is called **latent space**

AUTOENCODER

- Encoder and Decoder can be implemented using MLPs o CNNs

- Autoencoder in unsupervised technique

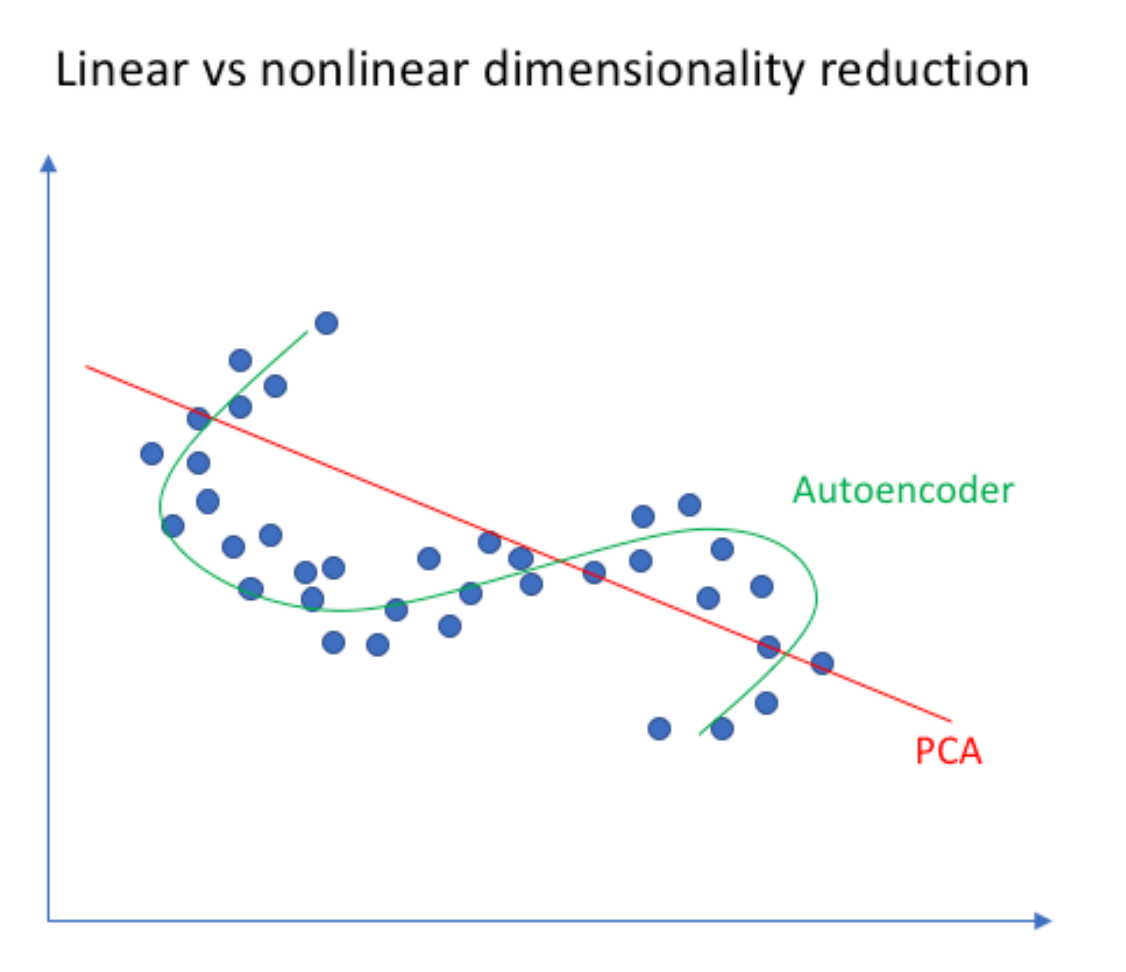


https://colab.research.google.com/github/AFAgarap/examples/blob/master/community/en/autoencoder.ipynb

AUTOENCODER – FEATURE LEARNING

- Autoencoder are much effective in feature learning and feature reduction
- The latent space is a reduced space of feature that well describe the input
- Features are learnt in a unsupervised manner

- Can be thought as a generalized PCA able to learn nonlinear relationships
- Whereas PCA attempts to discover a lower dimensional hyperplane which describes the original data, autoencoders are capable of learning nonlinear manifolds

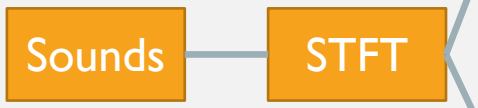


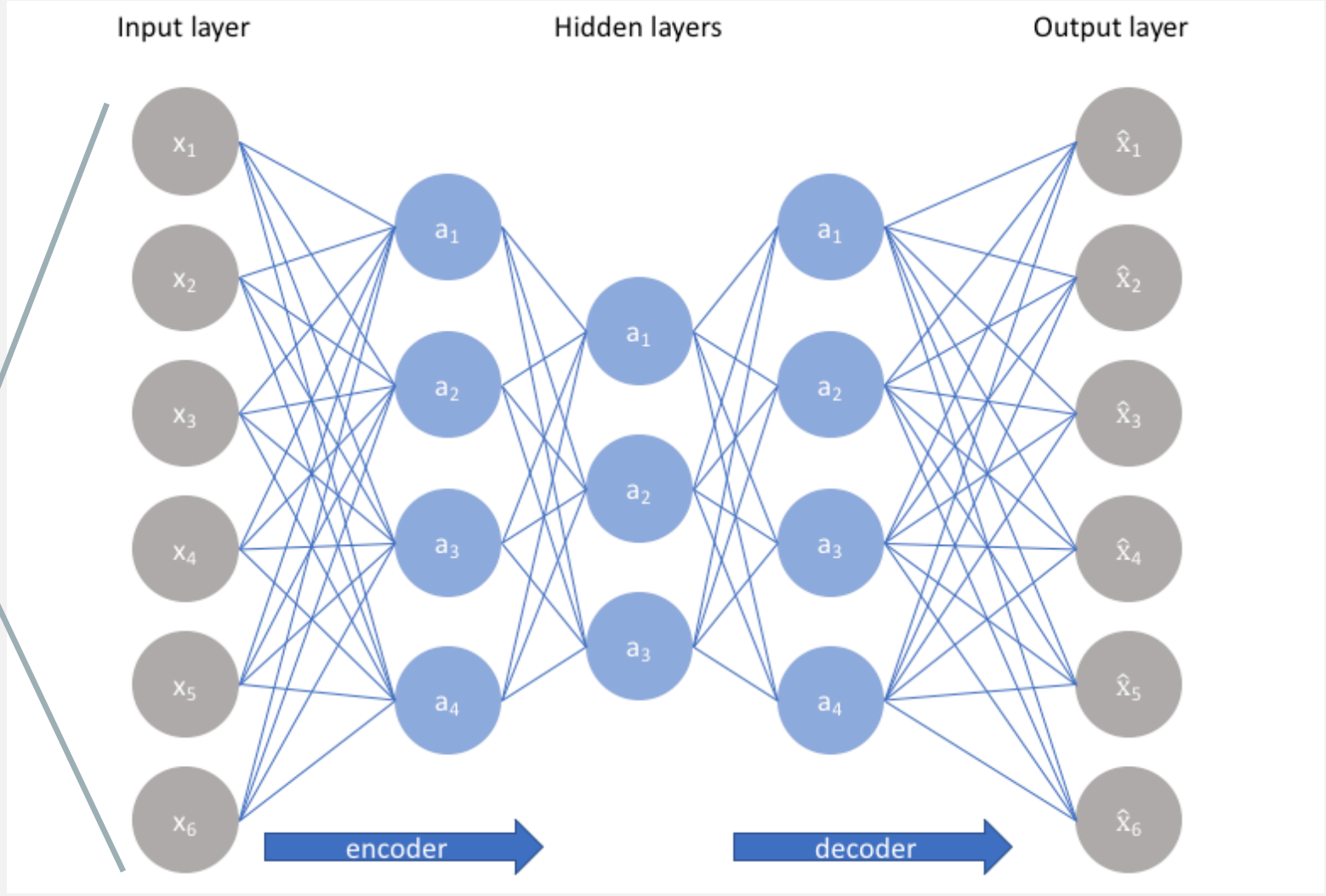
AUTOENCODER – SOUND SPACE

- **Goal**: organize sounds in a 2D space according to some similarity
- With MLPs the similarity were learnt in supervised manner (Ex. Genre)
- With Autoencoders the similarity is discovered (unsupervised manner)

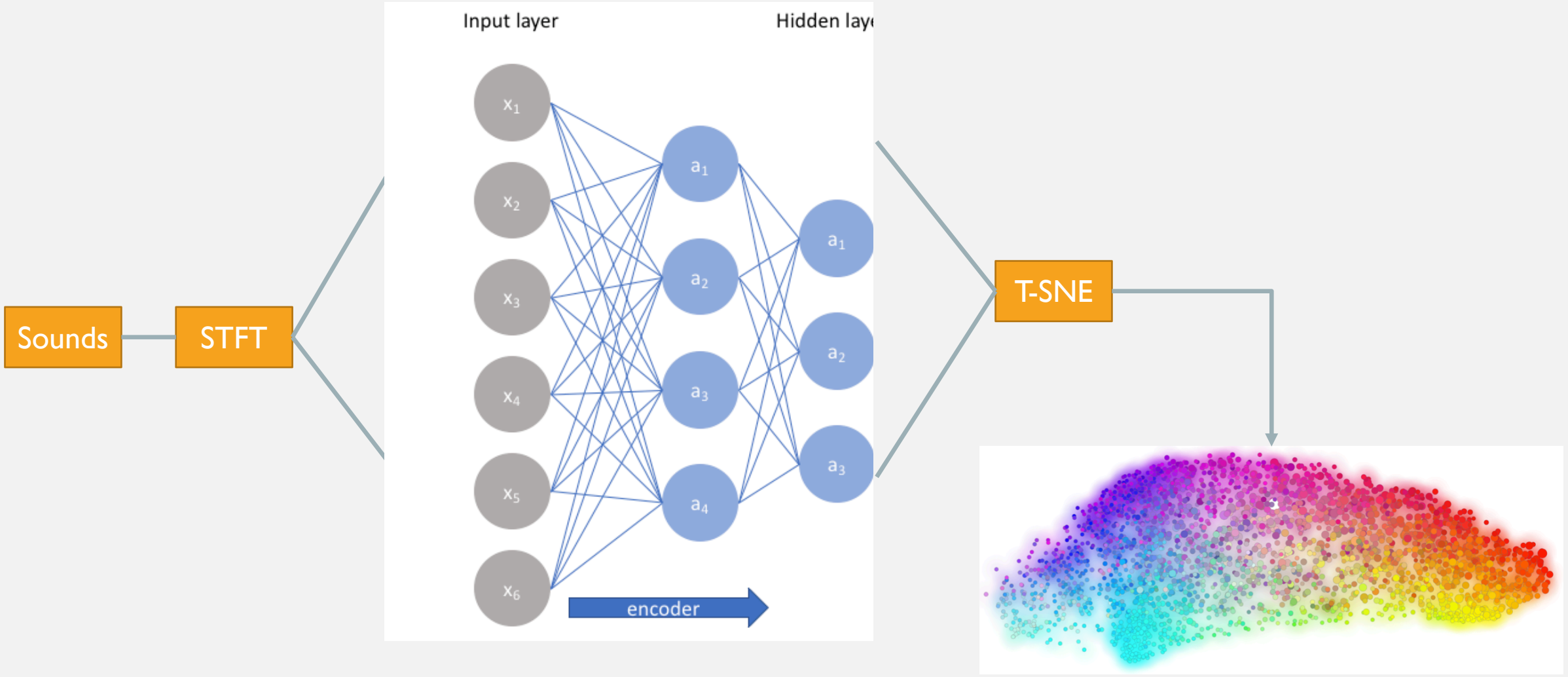
AUTOENCODER – SOUND SPACE

- Give a set of audio, the latent vector is learnt for each input





AUTOENCODER – SOUND SPACE

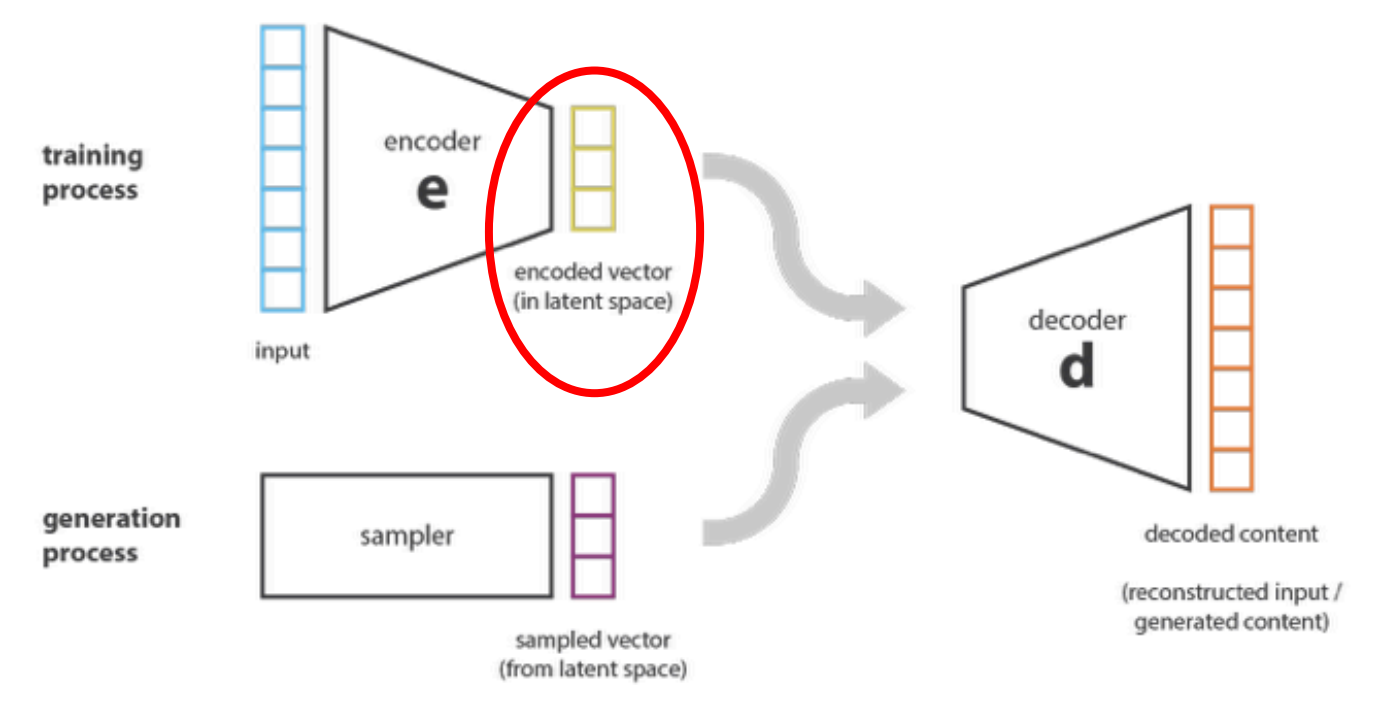


AUTOENCODER A GENERATIVE APPROACH

Are those real ?

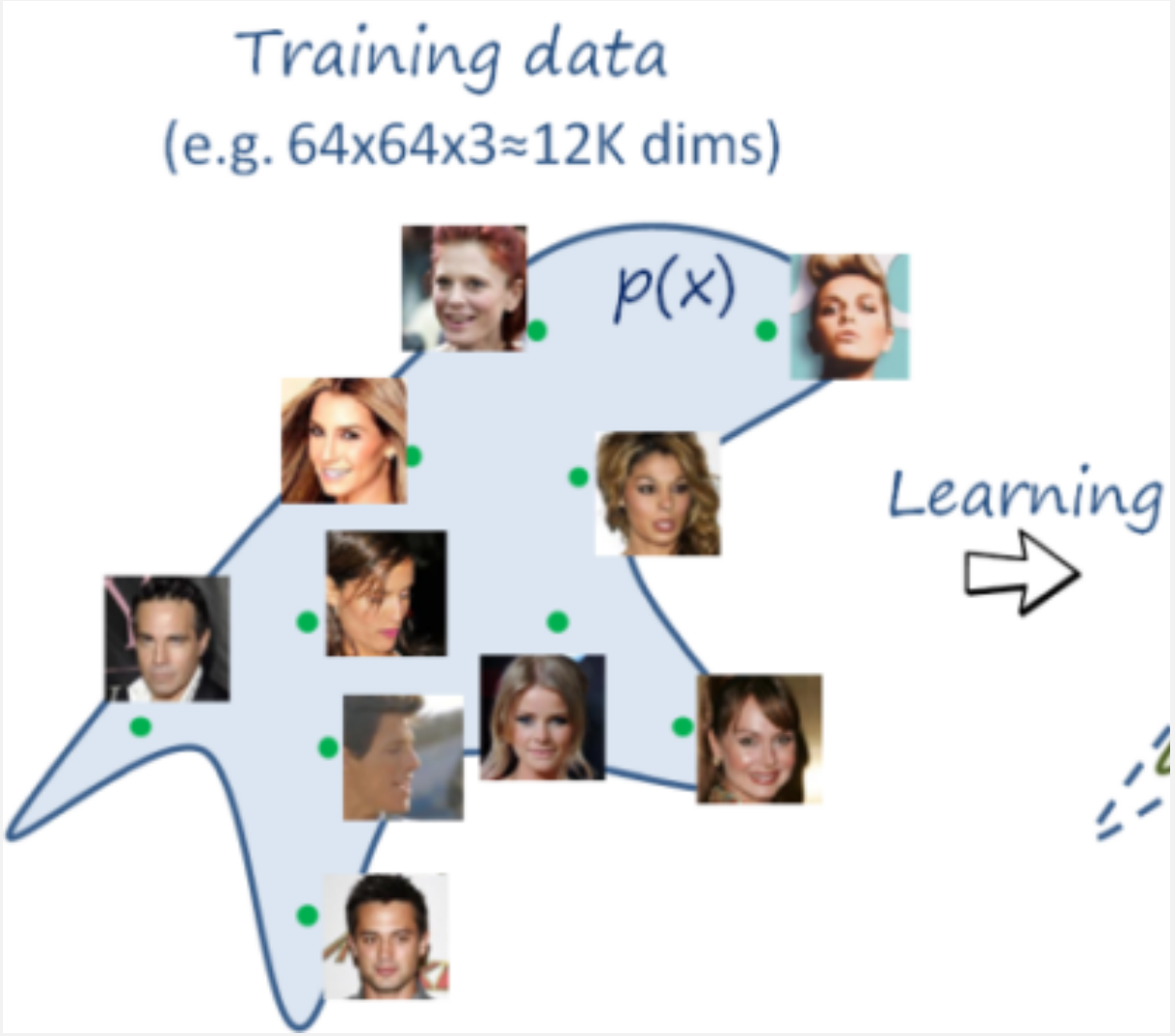
VARIATIONAL AUTOENCODER

- Given a training set, each input will produce a latent vector
- The set of latent vector can produce a multidimensional space (**latent space**)



VARIATIONAL AUTOENCODER

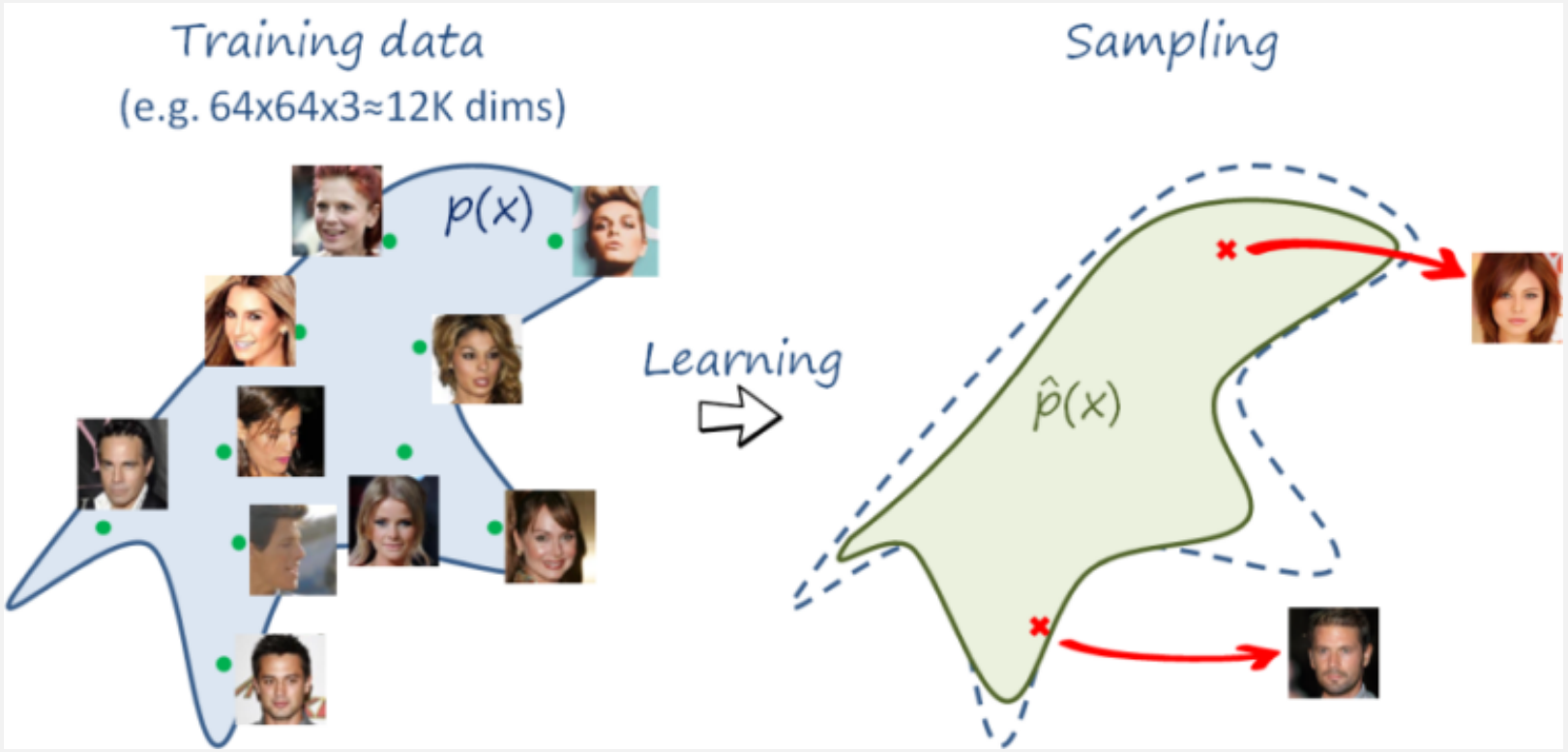
- Let's consider it as **not sparse**, the space will be composed by:
  - Points representing input data
  - Points do not representing input data



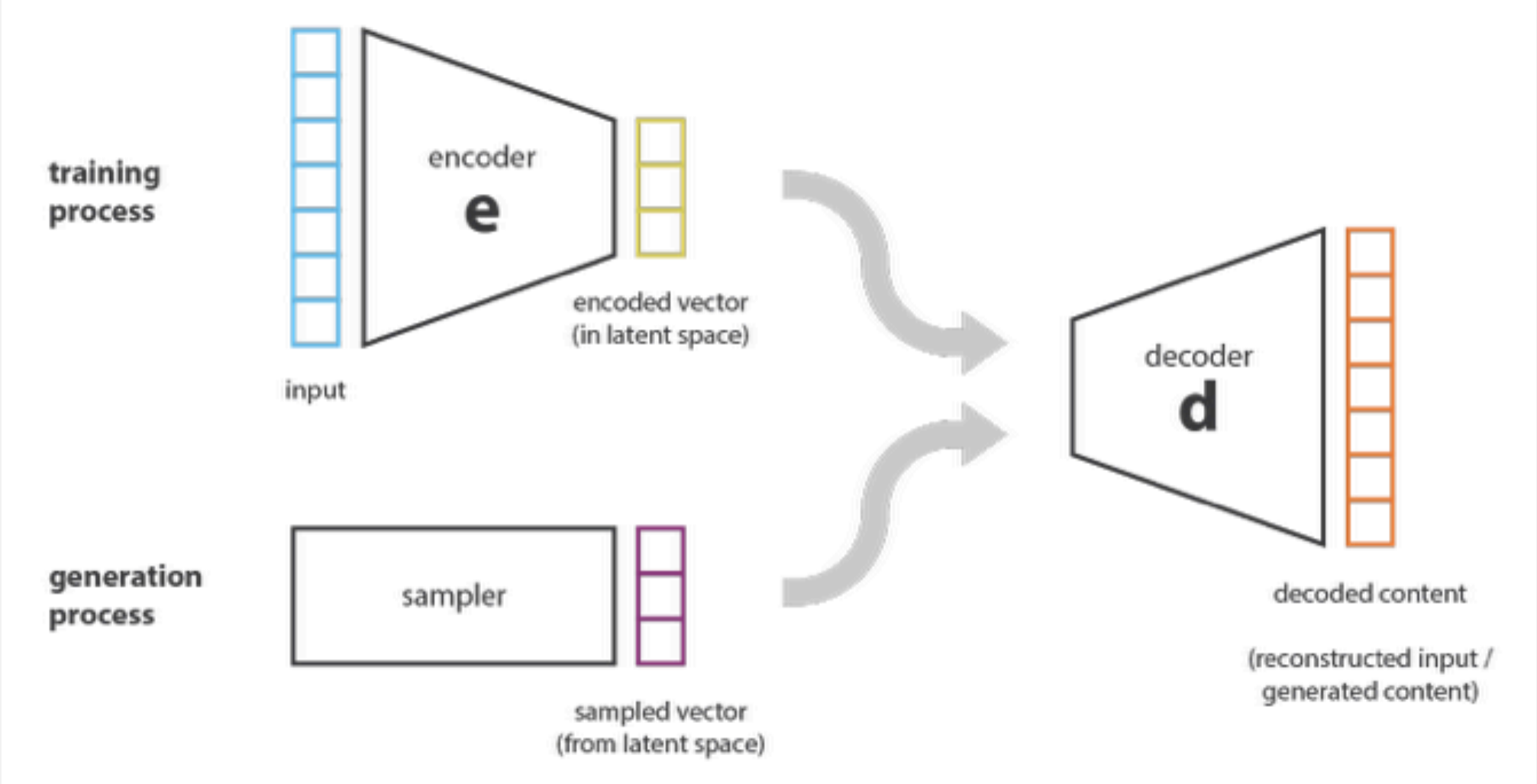
VARIATIONAL AUTOENCODER

- **We can generate new data by decoding points that are randomly sampled from the latent space**

- Example: give a latent space generated by a set of face images
- Generation (sampling): pick up a point in the space which is not the one part of the input data

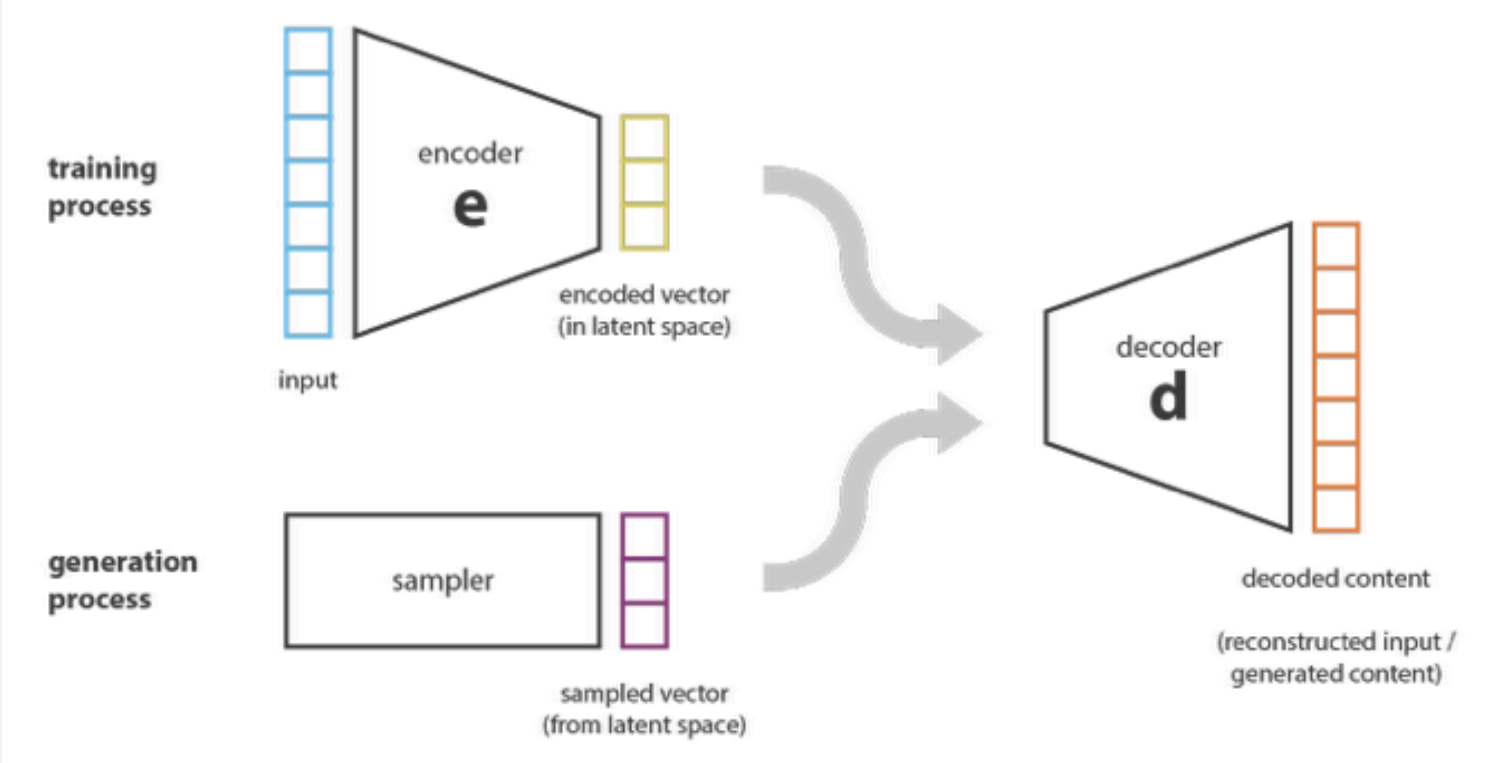


VARIATIONAL AUTOENCODER



VARIATIONAL AUTOENCODER

- If the latent space is not regular the final generation result will be low quality
- Regular: the space is well organized by the input data; no overfitting
- However -> autoencoder is build in order to overfit on input data
- -> regular latent space is not guarantee



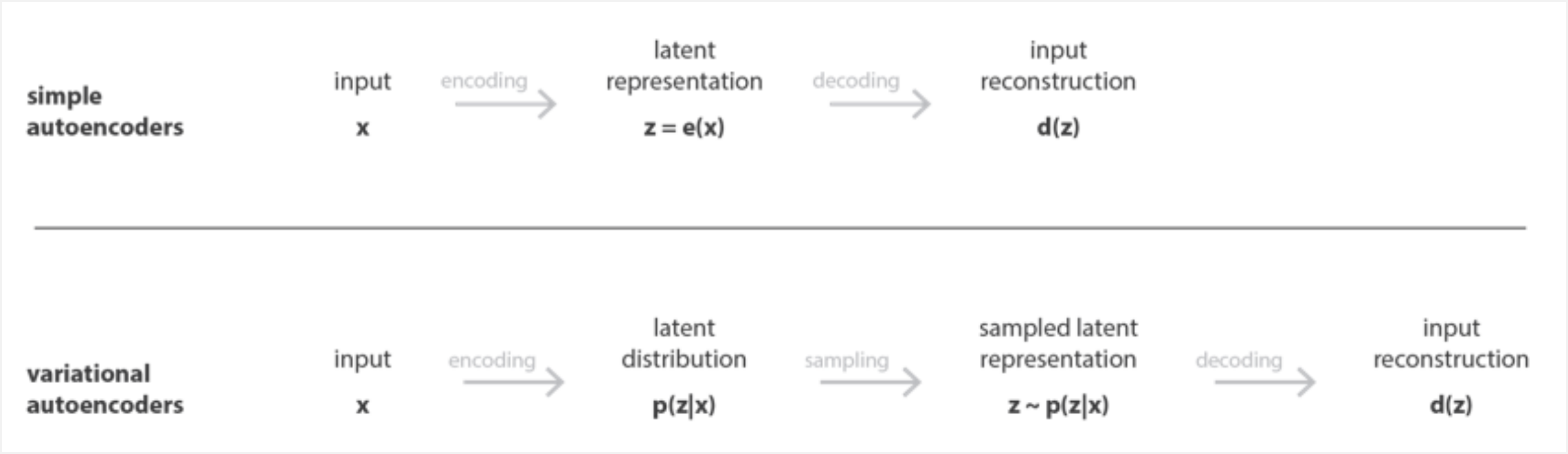
Variational Autoencoder

VARIATIONAL AUTOENCODER (VAE)

- VAE introduce explicit **regularisation** during the training process to avoid overfitting and ensure that the latent space has good properties that enable generative process
- Instead of encoding an input as a single point, we encode it as a distribution over the latent space

- In the training process:
  - the input is encoded as Gaussian distribution over the latent space
  - a point from the latent space is sampled from that distribution
  - the sampled point is decoded and the reconstruction error can be computed
  - the reconstruction error is backpropagated through the network

VARIATIONAL AUTOENCODER (VAE)

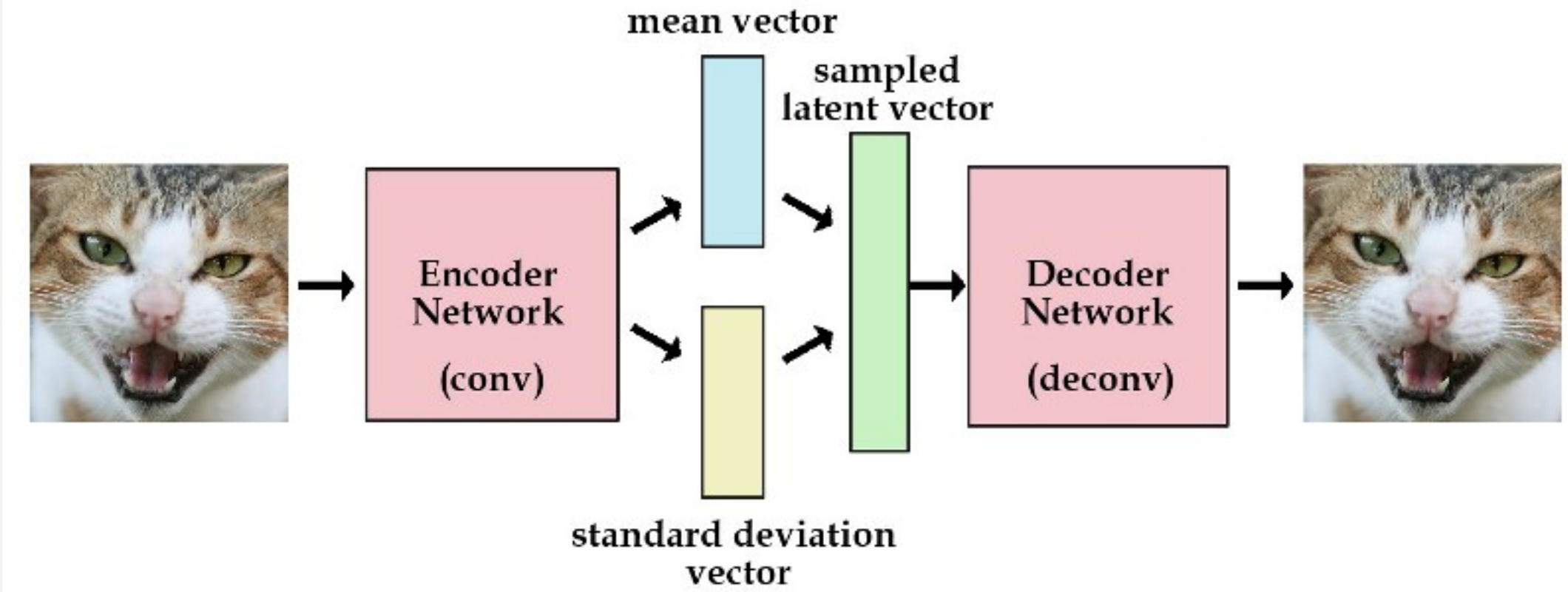


VARIATIONAL AUTOENCODER (VAE)

- In practice, the encoded distributions are chosen to be normal so that the encoder can be trained to return the mean and the covariance matrix that describe these Gaussians
- The distributions returned by the encoder are enforced to be close to a standard normal distribution.

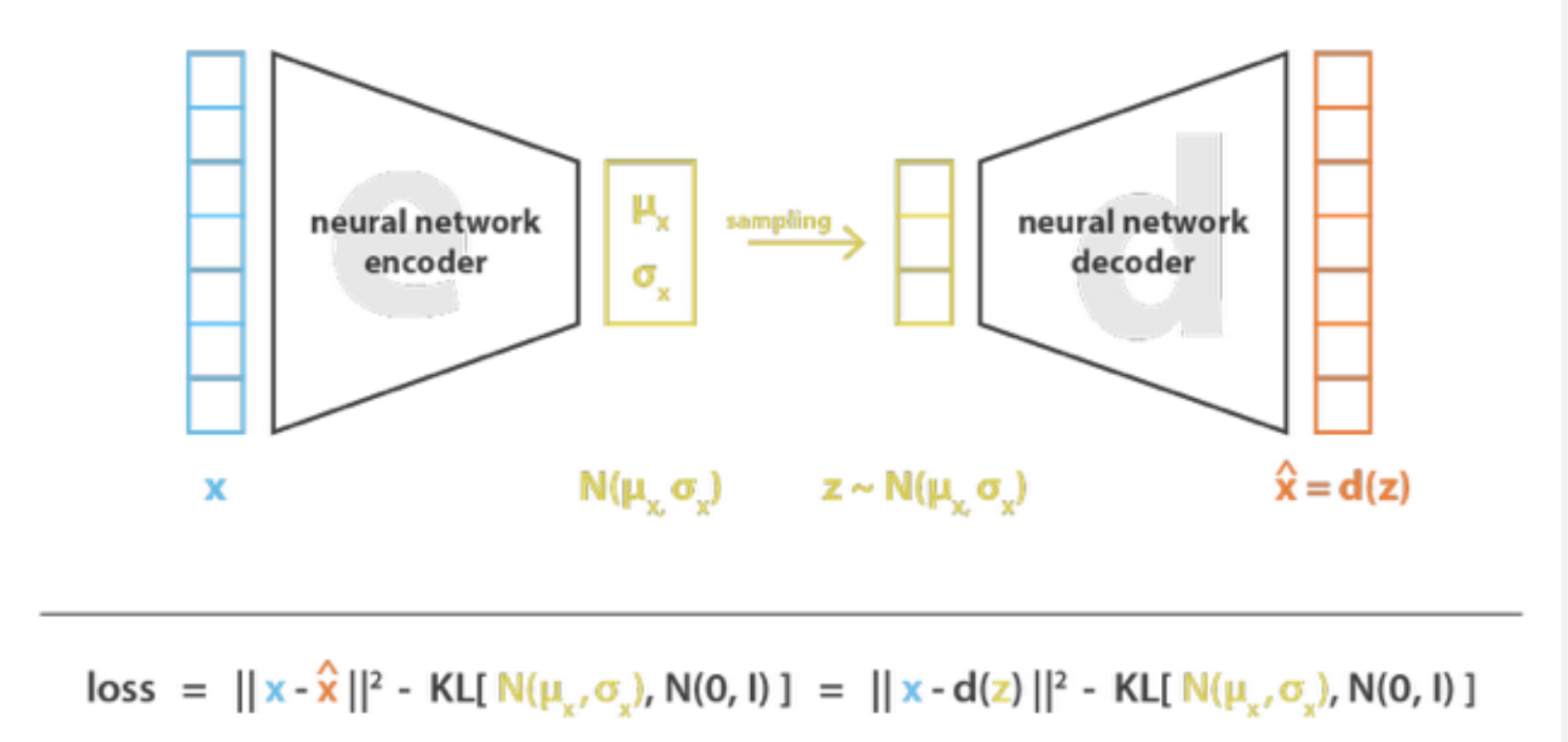
- The loss function that is minimised when training a VAE is composed of
  - a "**reconstruction term**" (on the final layer), that tends to make the encoding-decoding scheme as performant as possible
  - a "**regularisation term**" (on the latent layer), that tends to regularise the organisation of the latent space by making the distributions returned by the encoder close to a standard normal distribution

VARIATIONAL AUTOENCODER



VARIATIONAL AUTOENCODER (VAE)

- That regularisation term is expressed as the Kulback-Leibler divergence between the returned distribution and a standard Gaussian



- The backpropagation will tune the encoder to also increase the regularity of the latent space

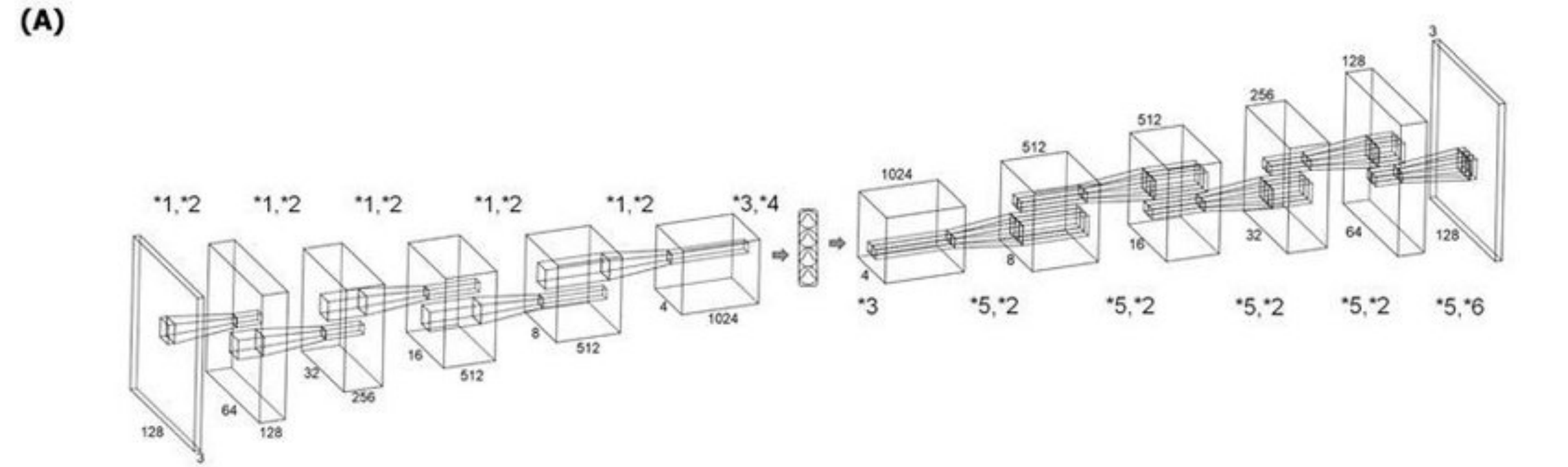
VARIATIONAL AUTOENCODER (VAE)

In the generation process:

- The problem of generating a new image of face is equivalent to the problem of generating a new vector following the "face probability distribution" over the N dimensional latent vector space.

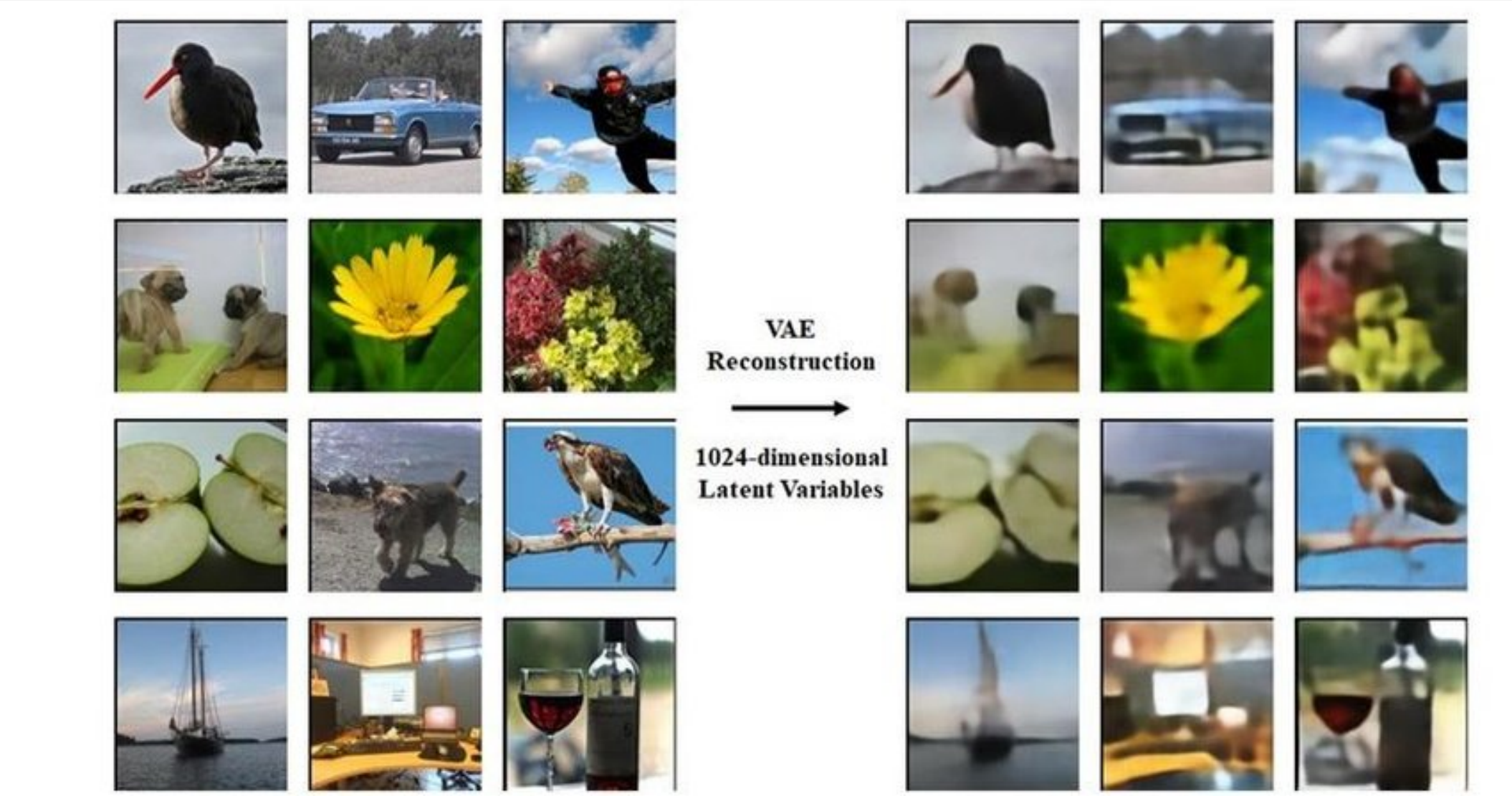
- So we are, in fact, facing a problem of generating a random variable with respect to a specific probability distribution

VAE – NEW FACE GENERATION



- Use of convNet as encoders and decoders
- The decoder implements upsampling instead of maxpooling

VAE – NEW FACE GENERATION



VAE – MUSIC

Magenta:

An open source research project exploring the role of machine learning as a tool in the creative process.

https://magenta.tensorflow.org

EXAMPLE: MAGENTA STUDIO

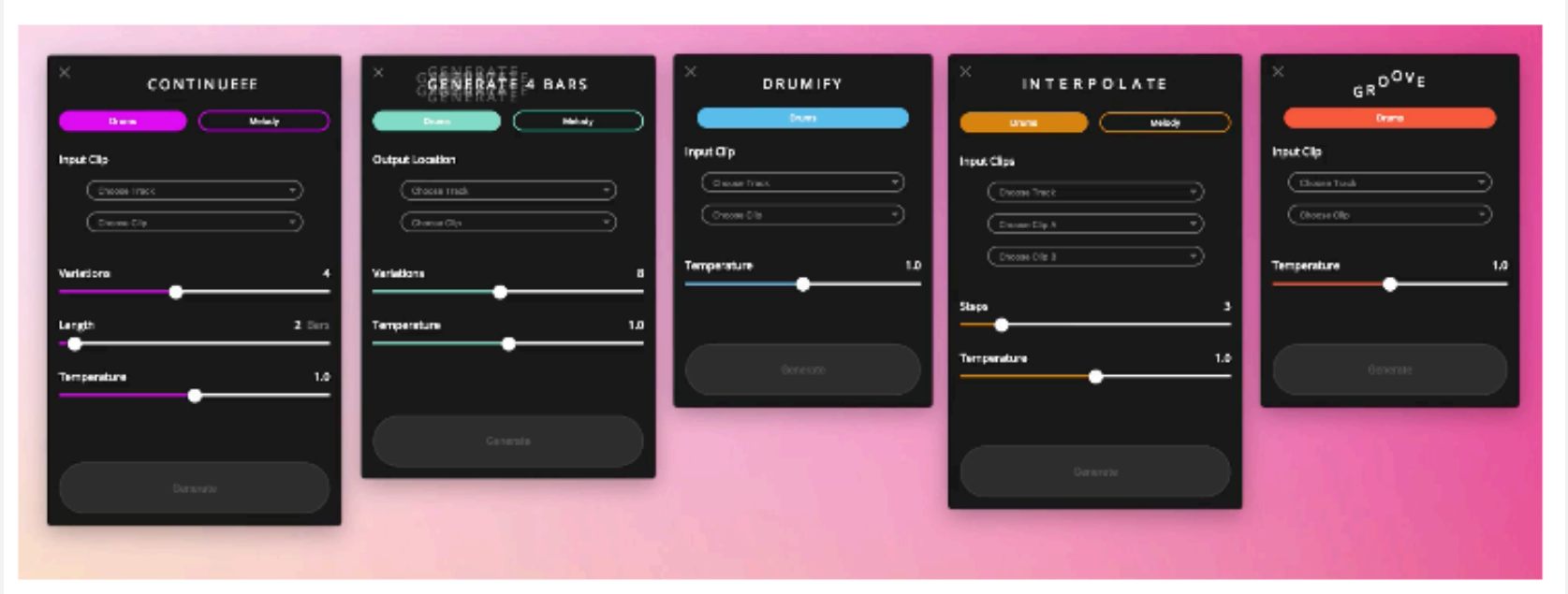
Magenta Studio is a collection of music plugins built on Magenta's open source tools and models. They use cutting-edge machine learning techniques for music generation

https://magenta.tensorflow.org/studio

Use it as plugin for **Ableton Live** o **Standalone**
Plugins uses MusicVAE

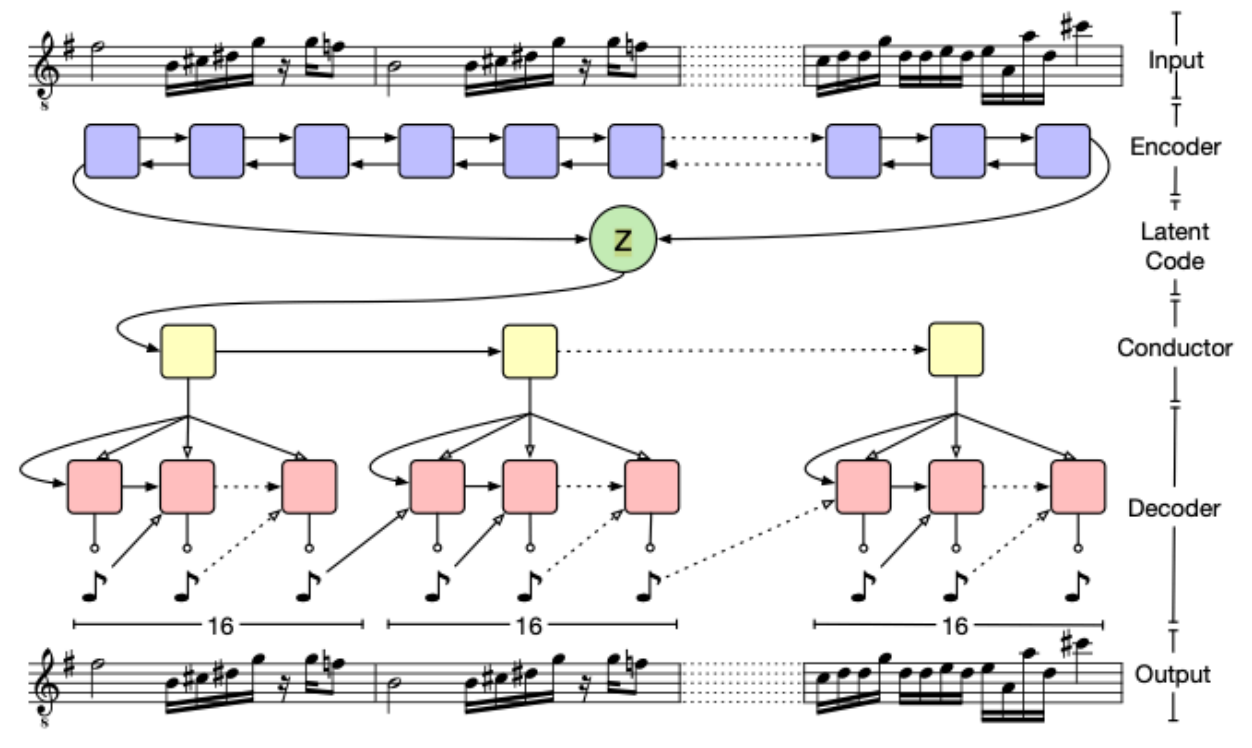
https://www.youtube.com/watch?v=G5JT16flZwM&list=PLBUMAYA6kvGU8Cgqh709o5SUvo-zHGTxr

https://www.youtube.com/watch?v=guNrrl9xpxs



VAE - MUSICVAE

- In the classical approach, VAE are not effective to model sequences of data
- -> **Bidirectional RNN-based Encoder**
- Each cell in the encoder receives a note from the sequence with respect to time. Hence there are 256 cells for a 16 bar (of eights) melody input.
- **-> Hierarchical Decoders**



https://www.youtube.com/watch?v=G5JT16fIZwM&list=PLBUMAYA6kvGU8Cgqh709o5SUvo-zHGTxr
https://www.youtube.com/watch?v=guNrrl9xpxs

EXAMPLE: SAMPLEVAE

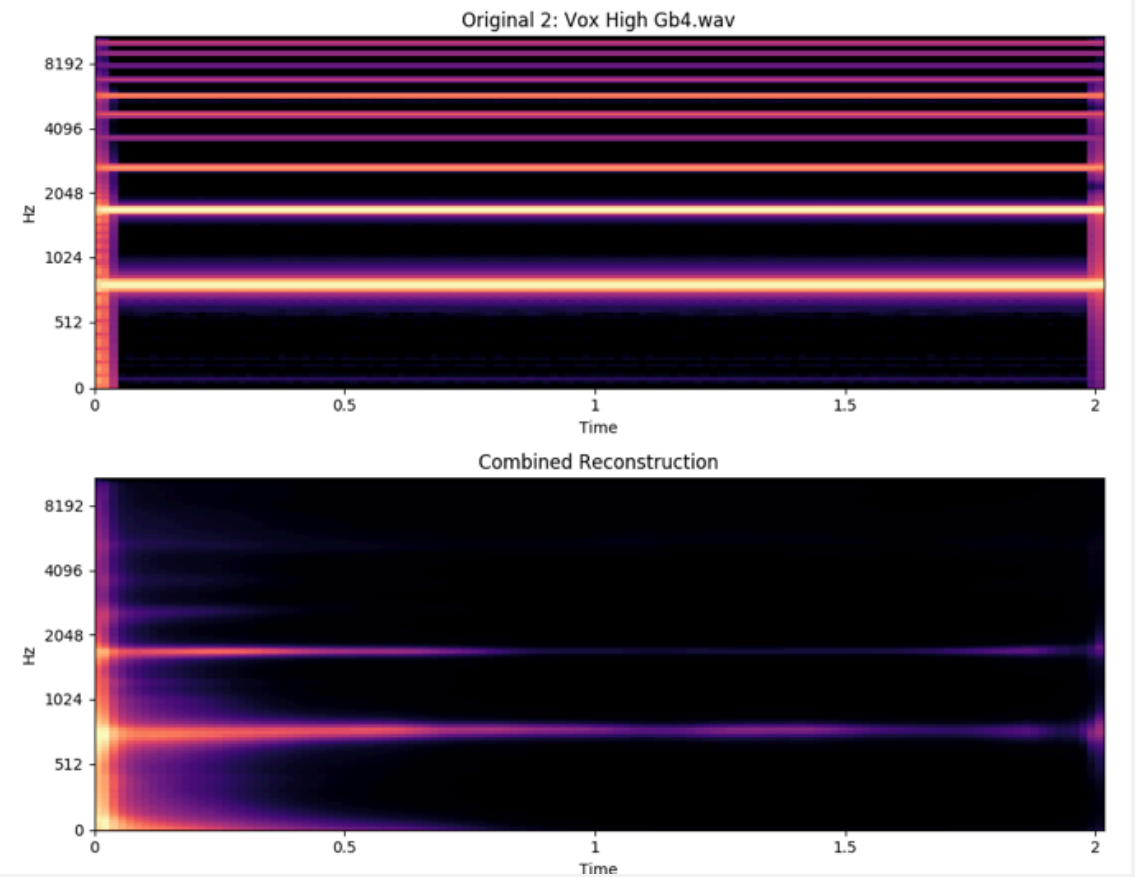
- Tool for sound designer created by Max Frenzel

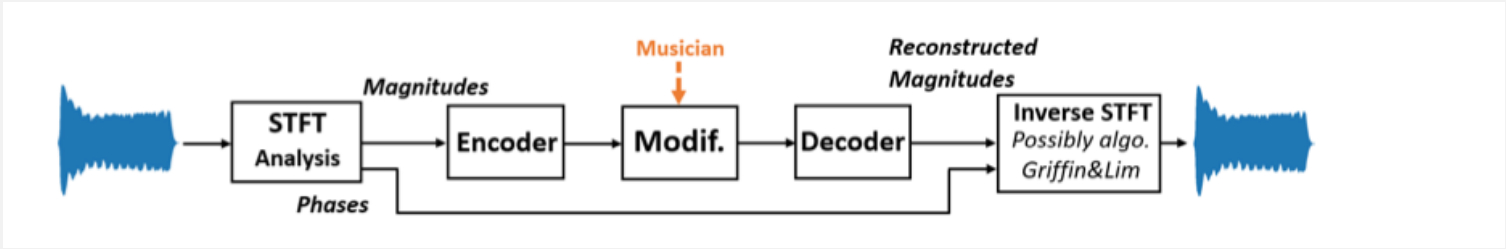
- It is composed by three tools
  - **Generate sounds in several unique ways**
  - **Classify sounds into distinct classes**
  - **Find similar sounds in a large sample library**

https://github.com/maxfrenzel/SampleVAE

EXAMPLE: SAMPLEVAE

- **Generate sounds in several unique ways**
  - https://soundcloud.com/taktilednb/neuralfunk
  - Encode spectrograms into a latent space and decode spectrograms from latent space
  - Based on the samples used to generate the latent space we can have interesting and hybrid sounds
  - Econder and decoder are CNN





https://github.com/maxfrenzel/SampleVAE

EXAMPLE: SAMPLEVAE

Classify sounds into distinct classes

https://vimeo.com/360731244

Find similar sounds in a large sample library

Sound close in the latent space a similar sounds

GANS

GENERATIVE ADVERSARIAL NETWORK

- Christie's sold a portrait for **$432,000** that had been generated by a GAN
- Based on open-source code written by Robbie Barrat of Stanford
- Programmer they didn't not receive any money
- French company, Obvious that run the GAN earned the money

GENERATIVE ADVERSARIAL NET (GAN)

Is this real ?

GENERATIVE ADVERSARIAL NET (GAN)

- Generative Adversarial Networks belong to the set of generative models. It means that they are able to produce / to generate (we'll see how) new content.

- Generate a new female face means to have a matrix (NxM) of pixels in a certain disposition.
- Let's convert the matrix into a vector. The vector can a point into a N*M-dimentional vector space
- In the vector space we have vector representing all possible images of NxM dimensions

- Female-face-related vectors will be closer in the space: we can say that are distributed according to a very specific probability distribution

- … we will have the probability distribution for cats, dogs, etc.

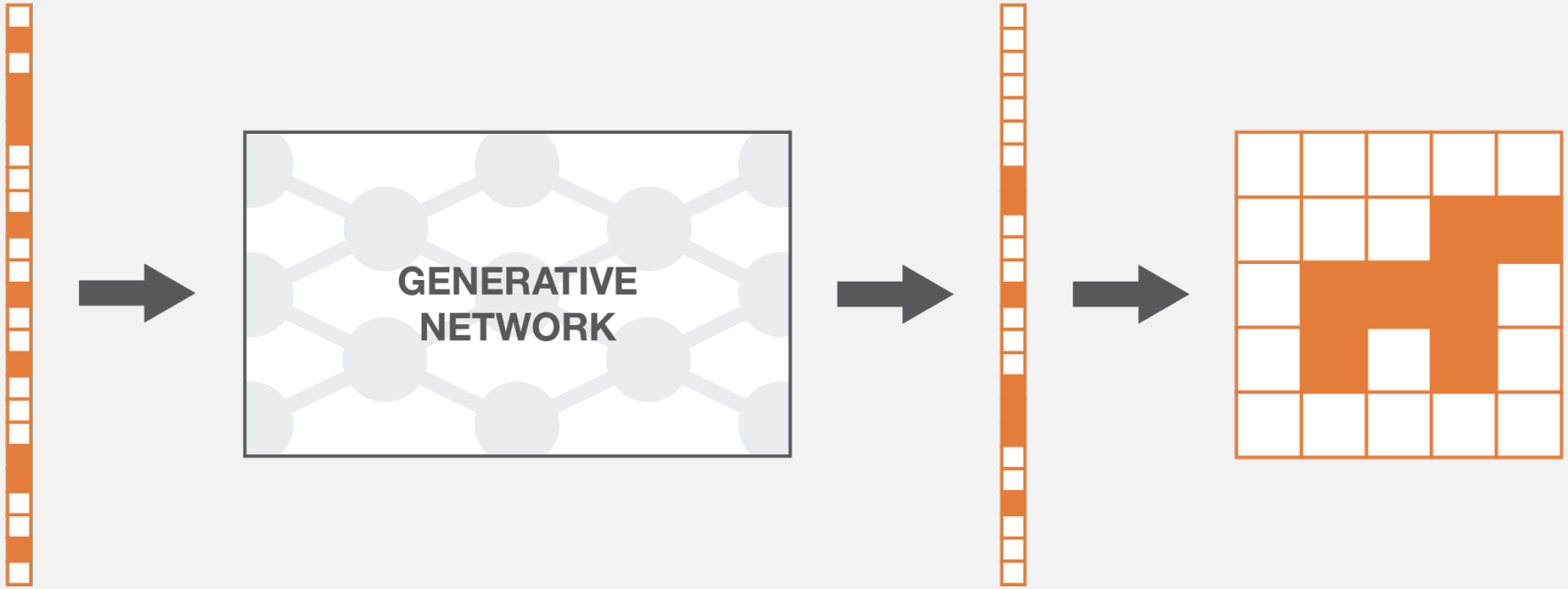
GENERATIVE ADVERSARIAL NET (GAN)

- Then, the problem of generating a new female face image is equivalent to the problem of generating a new vector following the "female face probability distribution".
- Pickup a vector in the probability distribution is easy
- **Issue:** probability distributions are very complex to define and we don't know how to directly generate complex random variables (vector)

- However we know how to generate N*M uncorrelated uniform random variables.
- Each vector in the space is the result of the application of a transformation function to a N*M vector generated trough N*M uncorrelated uniform random variables

GENERATIVE ADVERSARIAL NET (GAN)

- We ca use Deep Learning to learn the transformation function



Input random variable (drawn from a simple distribution, for example uniform).

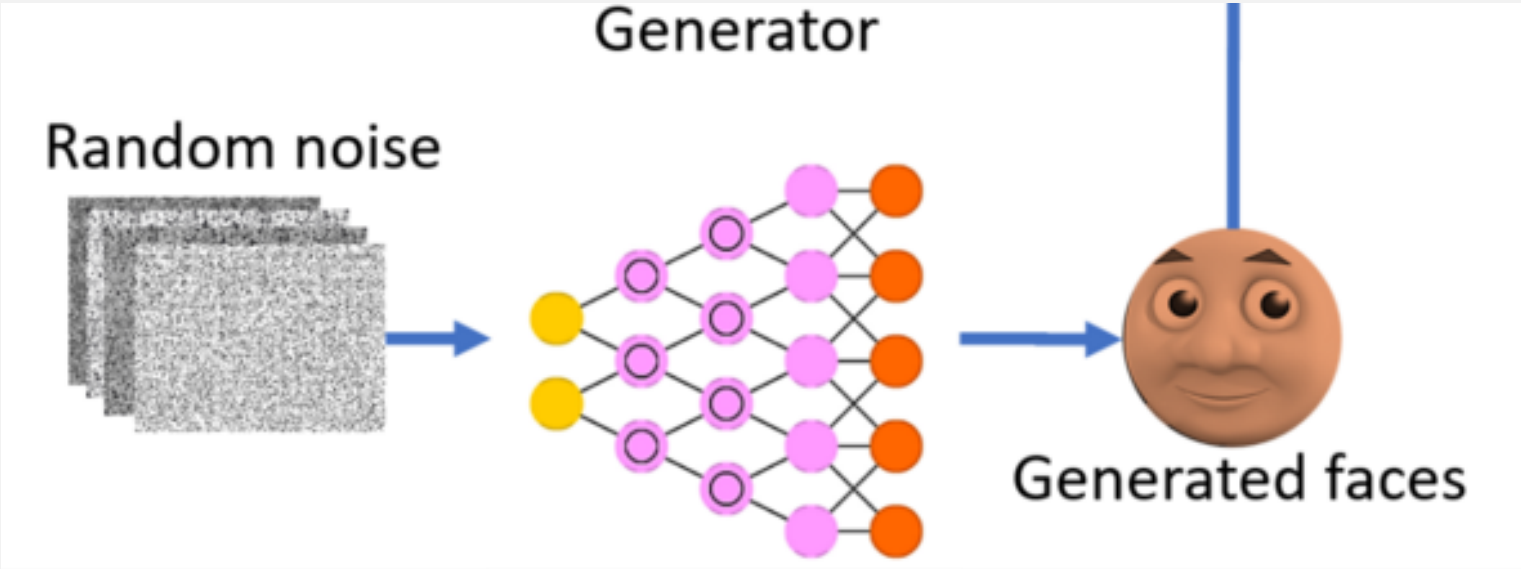
The generative network transforms the simple random variable into a more complex one.

Output random variable (should follow the targeted distribution, after training the generative network).

The output of the generative network once reshaped.

GENERATIVE ADVERSARIAL NET (GAN)

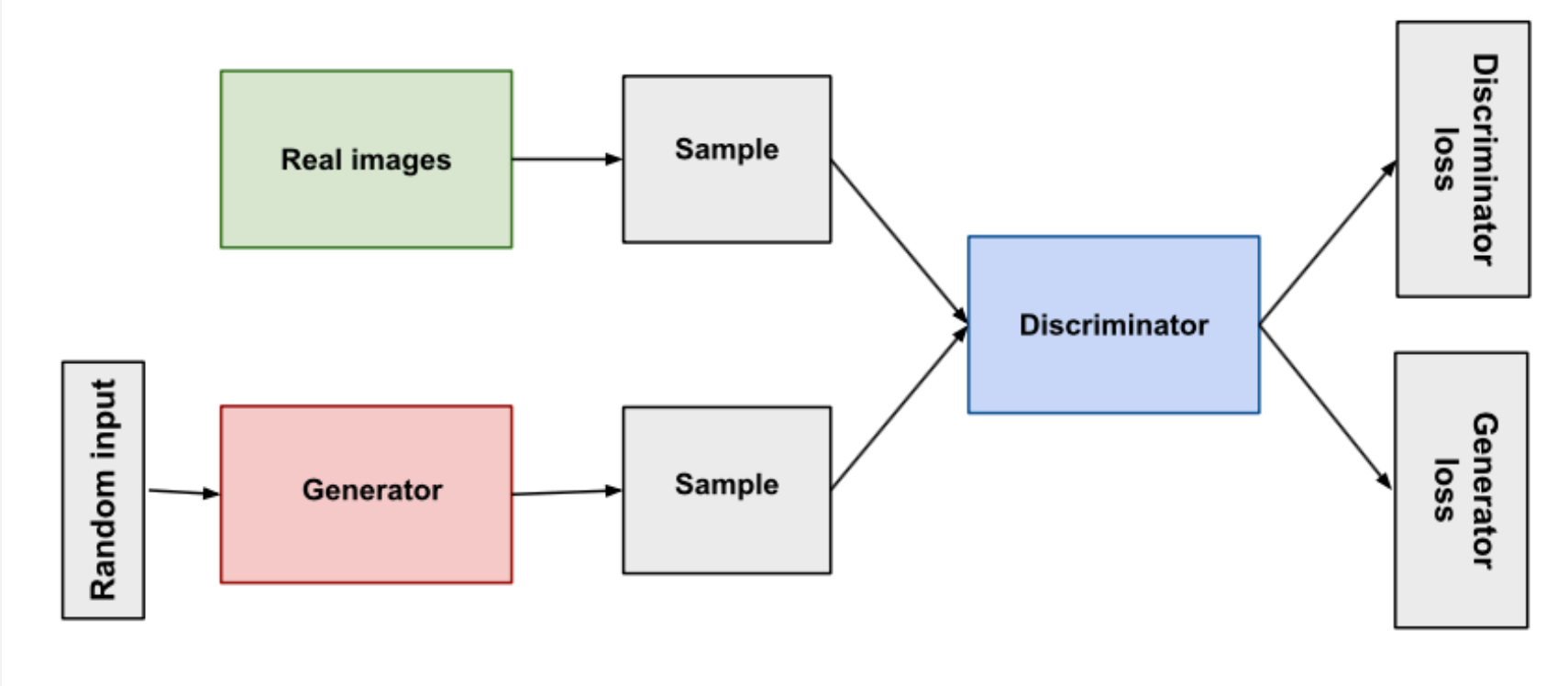
- We ca use Deep Learning to learn the transformation function



- Then we need a method to compare the generated vector and the distribution
- The error is back propagated to "sculpture" the random noise

GENERATIVE ADVERSARIAL NET (GAN)

- GANs frames the problem of unsupervised learning (generative) as a supervised learning problem with two sub-models:
- **Generator -** Model that is used to generate new plausible examples from the problem domain.
- **Discriminator -** Model that is used to classify examples as real (from the domain) or fake (generated).



GENERATIVE ADVERSARIAL NET (GAN)

- The two models are trained together in a zero-sum game, **adversarial**, in order to fool each other

- *Generative adversarial networks are based on a game theoretic scenario in which the generator network must compete against an adversary. The generator network directly produces samples. Its adversary, the discriminator network, attempts to distinguish between samples drawn from the training data and samples drawn from the generator.*

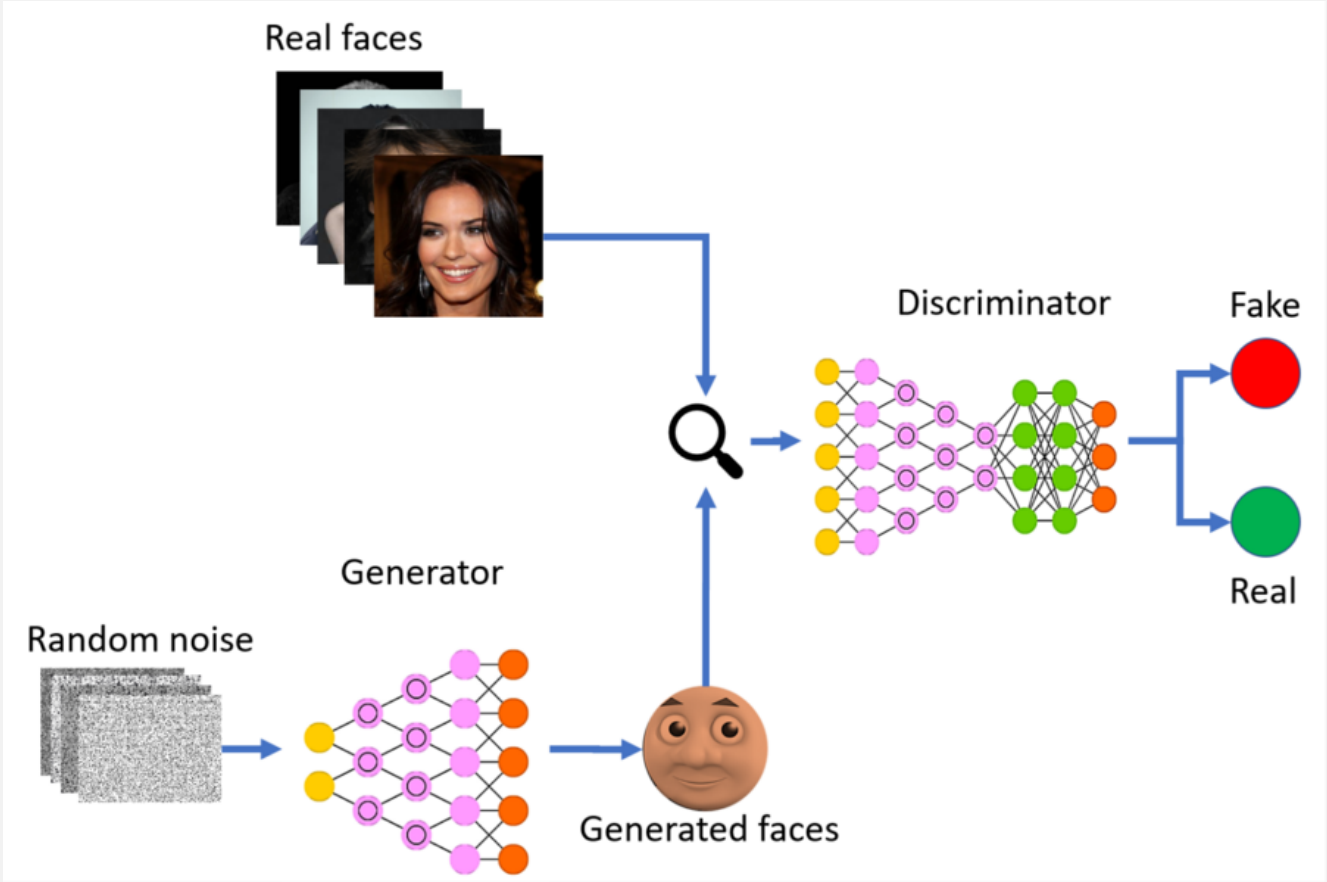
  — [Goodfellow] Page 699, Deep Learning, 2016

- In training, the competition lead the learning process of the two networks

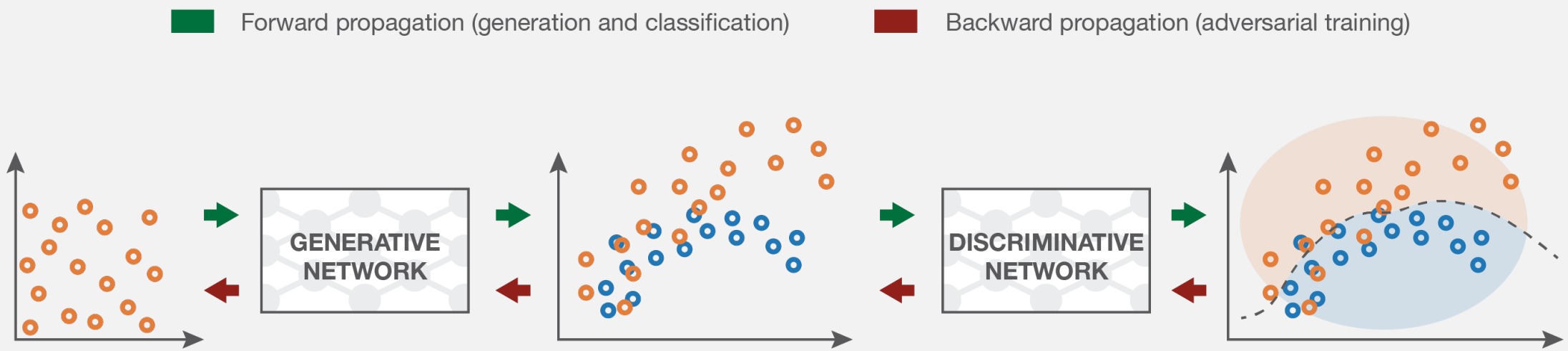
GAN – THE DISCRIMINATOR MODEL

- Supervised learning problem with two sub-models:
- the generator model that we train to generate new examples,
- the discriminator model that tries to classify examples as either real (from the domain) or fake (generated).

- The two models are trained together in a **zero-sum game**, adversarial, until the discriminator model is fooled about half the time, meaning the generator model is generating plausible examples.



GAN – THE DISCRIMINATOR MODEL



Input random variables.

The generative network is trained to **maximise** the final classification error.

The generated distribution and the true distribution are not compared directly.

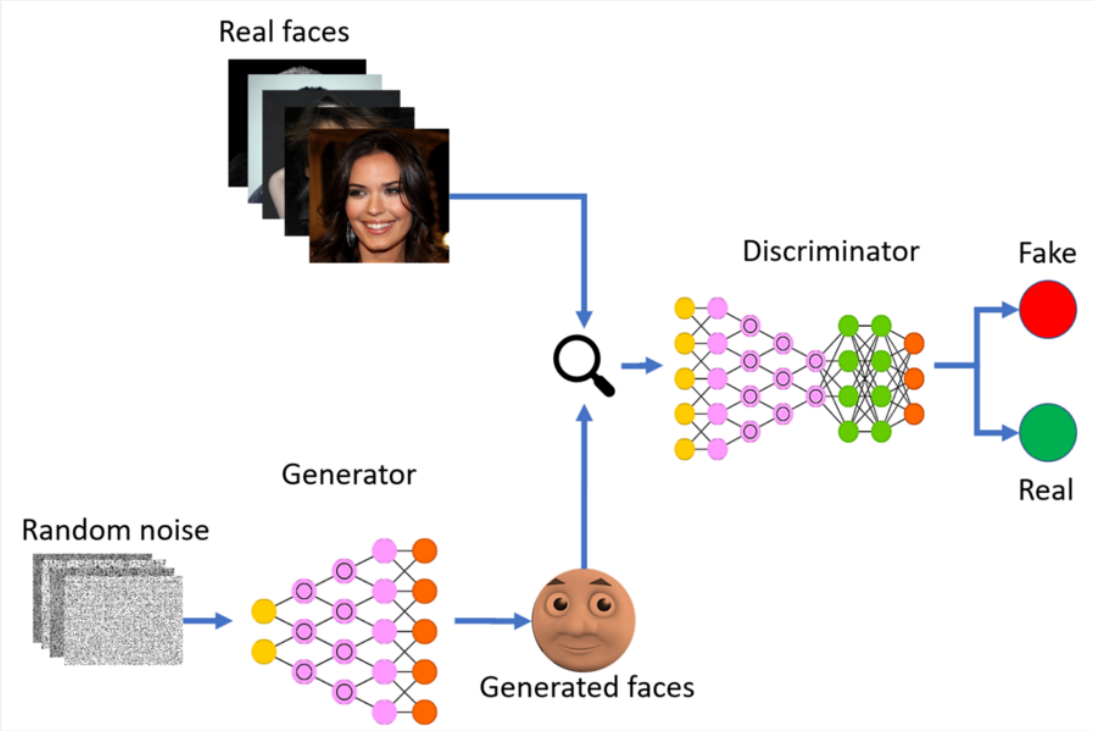
The discriminative network is trained to **minimise** the final classification error.

The classification error is the basis metric for the training of both networks.

GAN – THE DISCRIMINATOR MODEL

- In the training process, the real example comes from the training dataset.
- At each iteration a new randomly chosen real example is considered

- Hence, the most representative the training set is, the more difficult is to fool the discriminator, the more realistic is the generated image



GAN – THE TWO PLAYER GAME

- The two models, the generator and discriminator, are trained together

- The **discriminator** is then updated to get better at discriminating real and fake samples in the next round (it knows which is the real one and which is not)
- The **generator** is updated based on how well, or not, the generated samples fooled the discriminator – it will updated in order create a better latent space from the initial random vector
- That is
  - The generator has to learn how to **fool** the the discriminator
  - The discriminator has to learn how to **not be fooled** by the generator

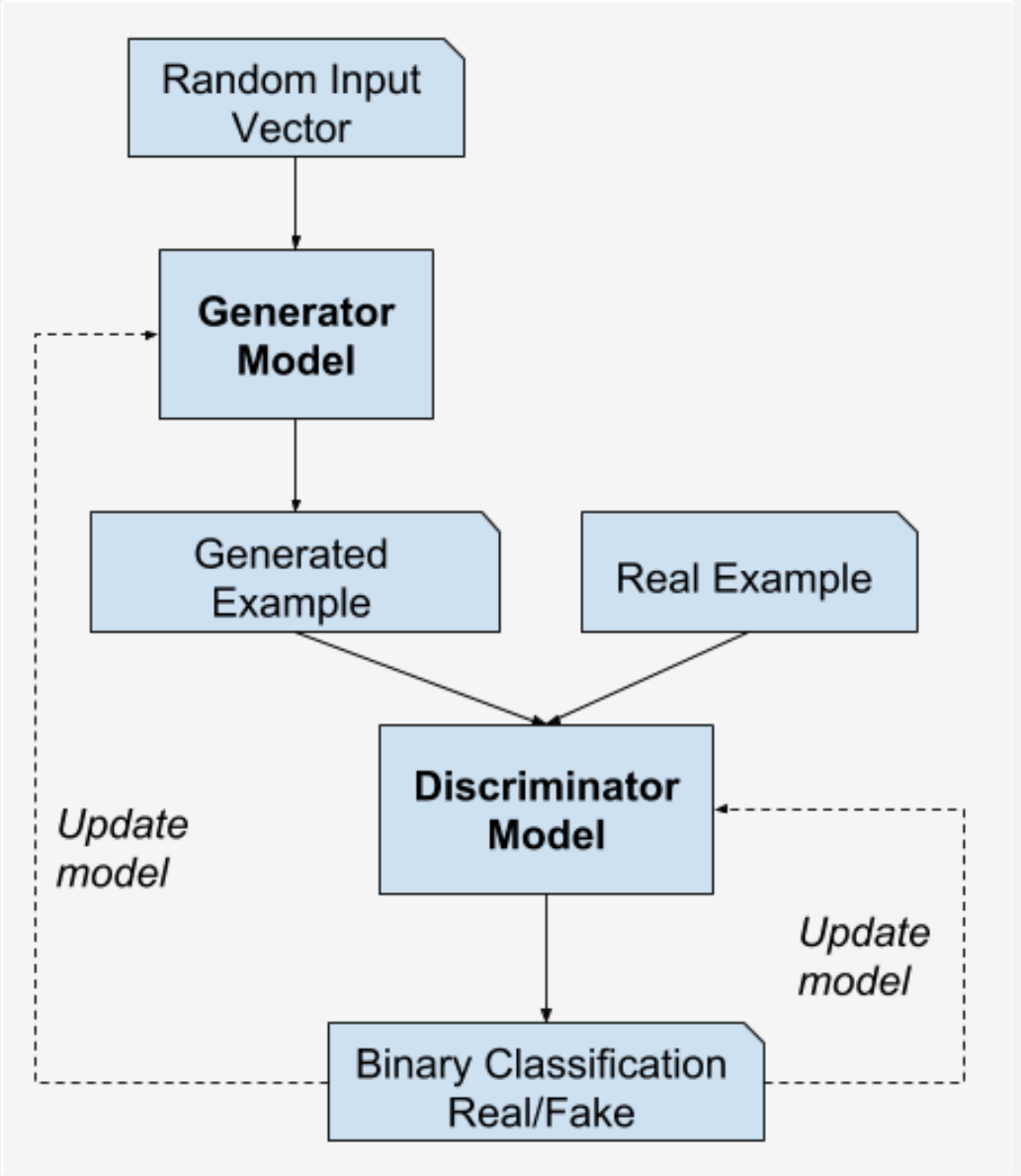
- The process is a **backpropagation** process

GAN – THE TWO PLAYER GAME

- So, they are adversarial in the game theory sense, and are playing a **zero-sum game**

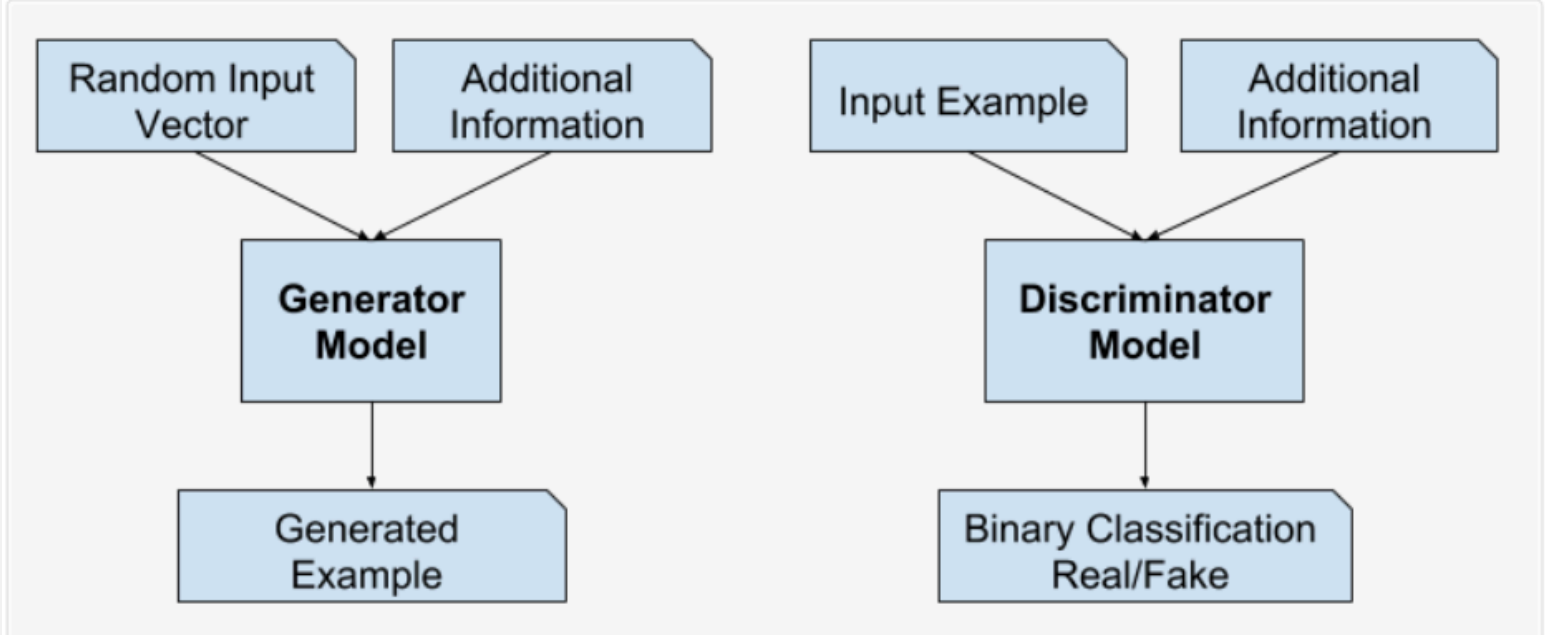
- **Zero-sum** means that when the discriminator successfully identifies real and fake samples, it is rewarded or no change is needed to the model parameters, whereas the generator is penalized with large updates to model parameters.

- Alternately, when the generator fools the discriminator, it is rewarded, or no change is needed to the model parameters, but the discriminator is penalized and its model parameters are updated.



CONDITIONAL GAN

- The generative model can be trained to generate new examples from the input domain, where the input, the random vector from the latent space, is provided with (conditioned by) some additional input
- The additional information can be for instance the class male or female of the image
- The discriminator also receives a label associated with each of the real input
- The discriminator do not performs fakeVSreal claffication, but (fake,label)VS(real,label)
- -> a Conditional GAN can be used to generate examples from a domain of a given type.



CONDITIONAL GAN

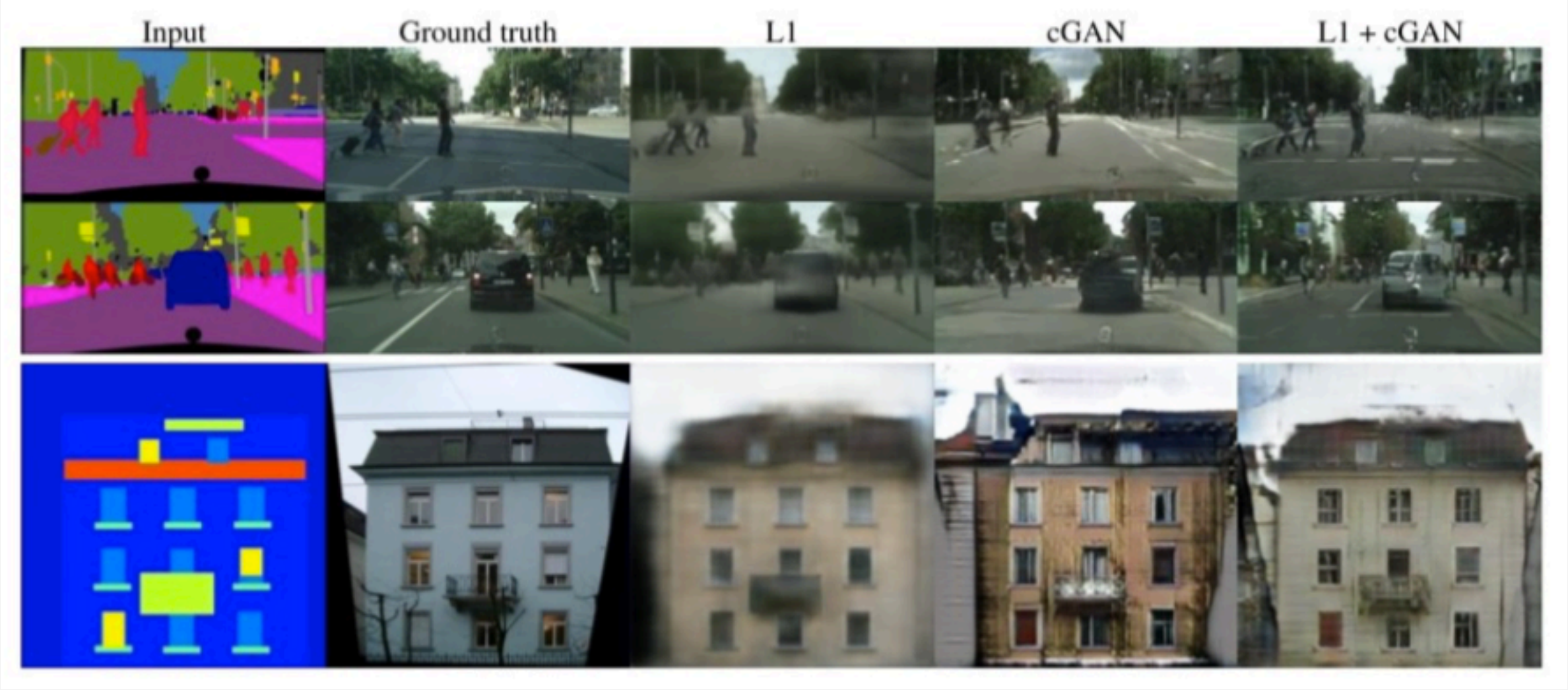
- What if the conditional info is an image ? ->>> image style transformation
- **Goal**: transform an image from day to night time

- The **discriminator** is provided examples of **real** and **generated** night time photos as well as (conditioned on) real daytime photos as input.
- The **generator** is provided with a random vector from the latent space as well as (conditioned on) real daytime photos as input

- The discriminator will teach to the generator how to generate an image very similar to the real daytime photos (condition) in the style of night time photos

CONDITIONAL GAN

CONDITIONAL GAN



GAN VS VAE

	GAN	VAE
Objective	(generator) fool the discriminator (discriminator) tell real data from fake ones	reconstruct real data using pixel-wise loss
Results	tend to be sharper 	tend to be more blurred 
Diversity	Higher	Lower
Stability	Lower	Higher

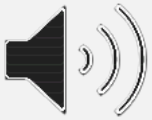
EXAMPLE: GAN SYNTH

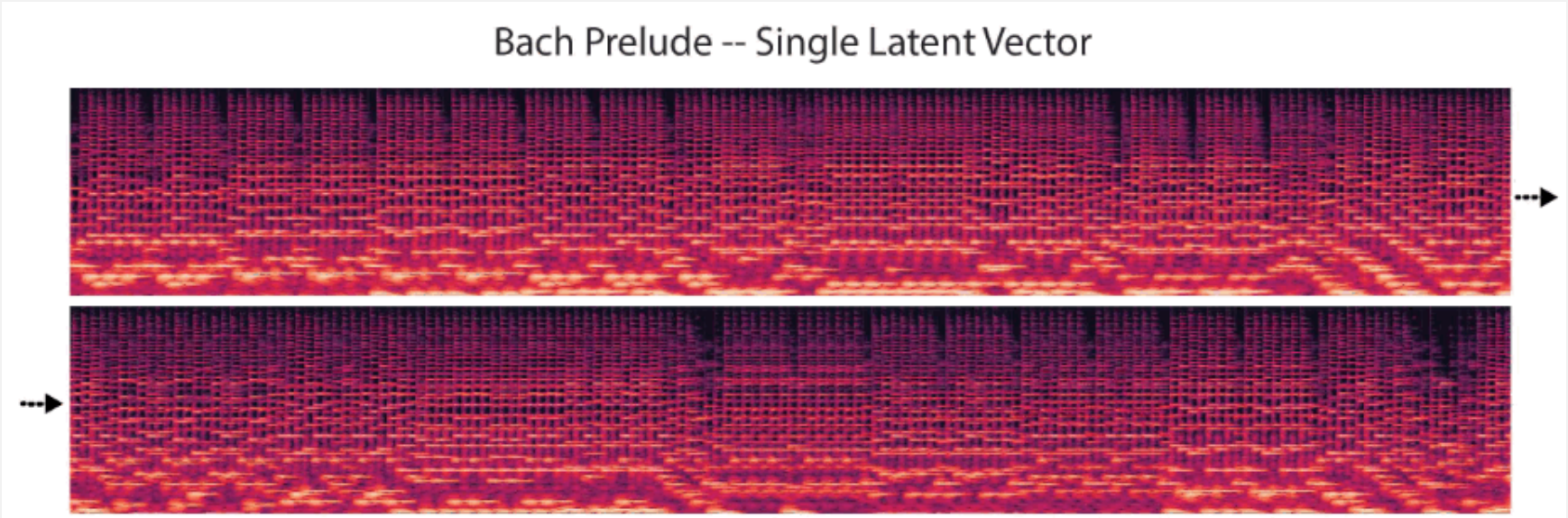
- **Generator** and **Discriminator** are **CNN**
- The model samples a random vector z from a spherical Gaussian, and runs it through a stack of transposed convolutions to upsample and generate output data x = G(z)
- X is fed into a discriminator network of downsampling convolutions (whose architecture mirrors the generator's) to estimate a divergence measure between the real and generated distributions
- GANSynth can be conditioned introducing the **pitch condition**

https://storage.googleapis.com/magentadata/papers/gansynth/index.html#consistent_timbre

EXAMPLE: GAN SYNTH

- ***Neural audio synthesis***
- Method for generating high-fidelity audio with Generative Adversarial Networks (GANs).
- Hard to apply to audio Generation
  - Classic autoregressive models predict one sample at a time (Wavenet, Transformers)
  - GANSynth generates the entire audio clip from a single latent vector, allowing for easier disentanglement of global features such as pitch and timbre.
  - GANSynth generates an entire sequence in parallel

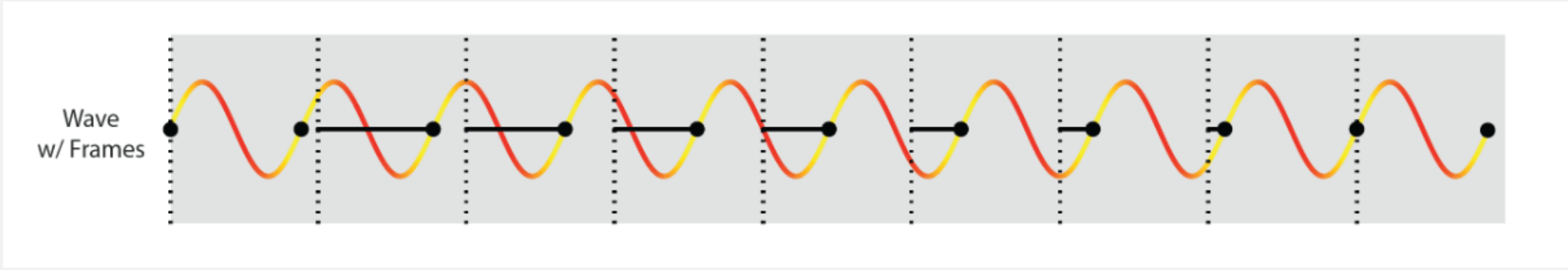




EXAMPLE: GAN SYNTH

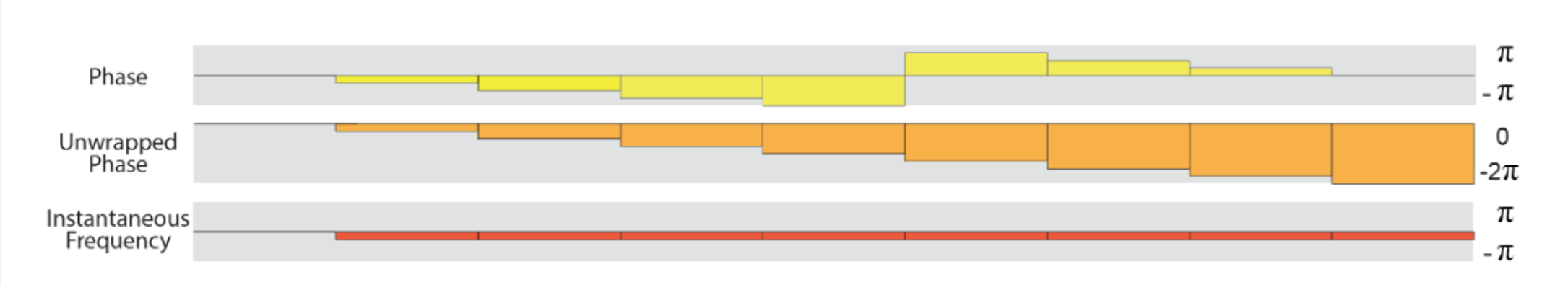
- **Data**
  - Trained on the NSynth Dataset
  - audio dataset containing 305,979 musical notes, each with a unique pitch, timbre, and envelope
  - 1,006 instruments from commercial sample libraries
  - four second, monophonic 16kHz audio snippets, referred to as notes, by ranging over every pitch of a standard MIDI pian o (21-108) as well as five different velocities (25, 50, 75, 100, 127)

- Identify best audio representation
  - convolution struggles with phase alignment for highly periodic signals



EXAMPLE: GAN SYNTH

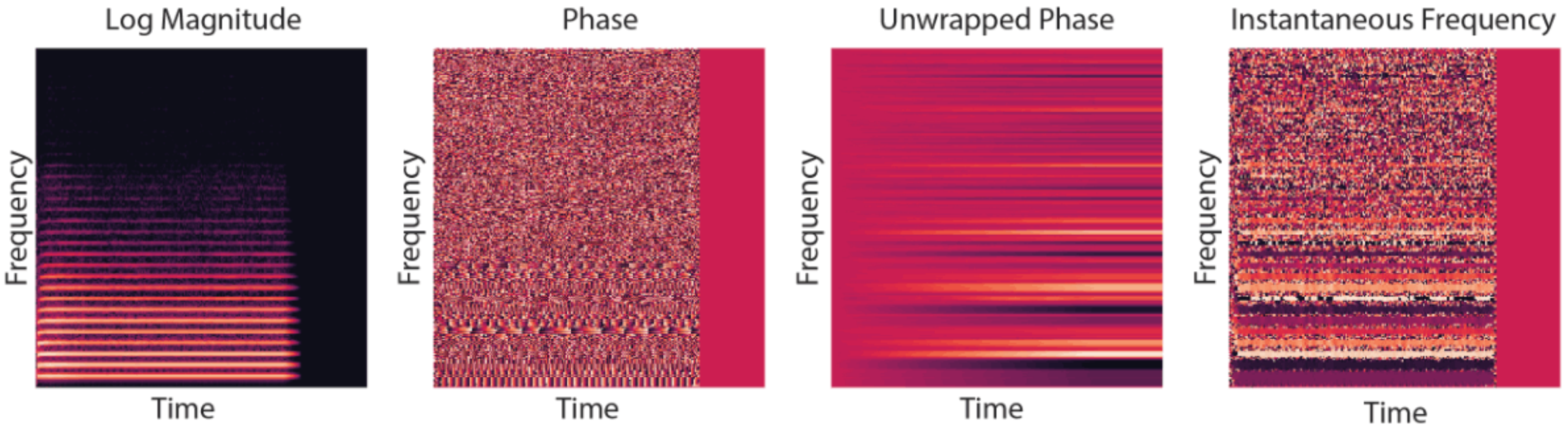
- Phase unwrapping and instantaneous frequency are a possible solution



- Pure tone produces a phase that precesses
- Unwrapping the phase causes the precessing phase to grow linearly
- Taking the derivative of the unwrapped phase is defined as *instantaneous angular frequency*

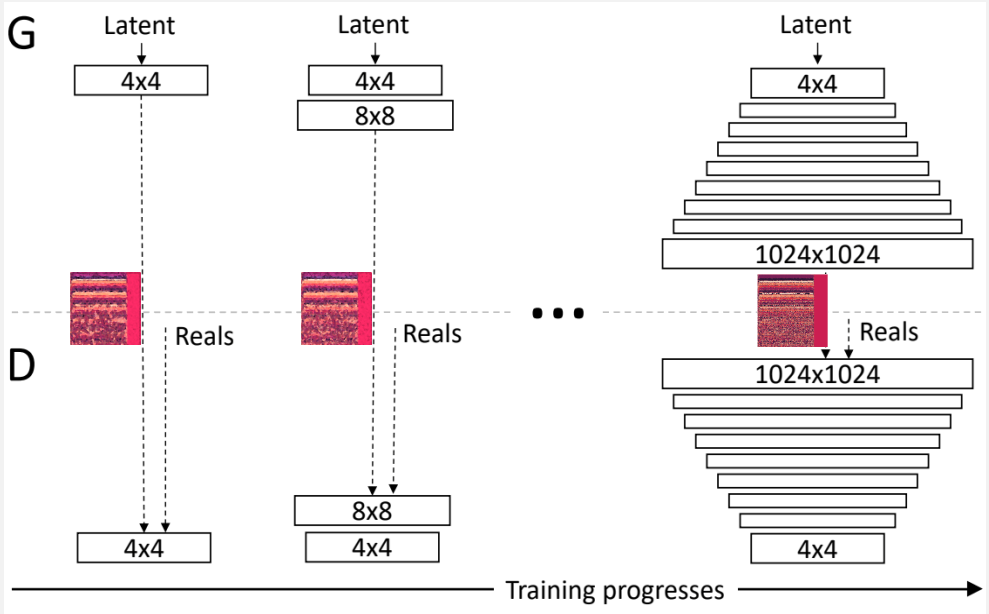
EXAMPLE: GAN SYNTH

- Instantaneous frequency is the one that performs better
- example trumpet note from the NSynth dataset



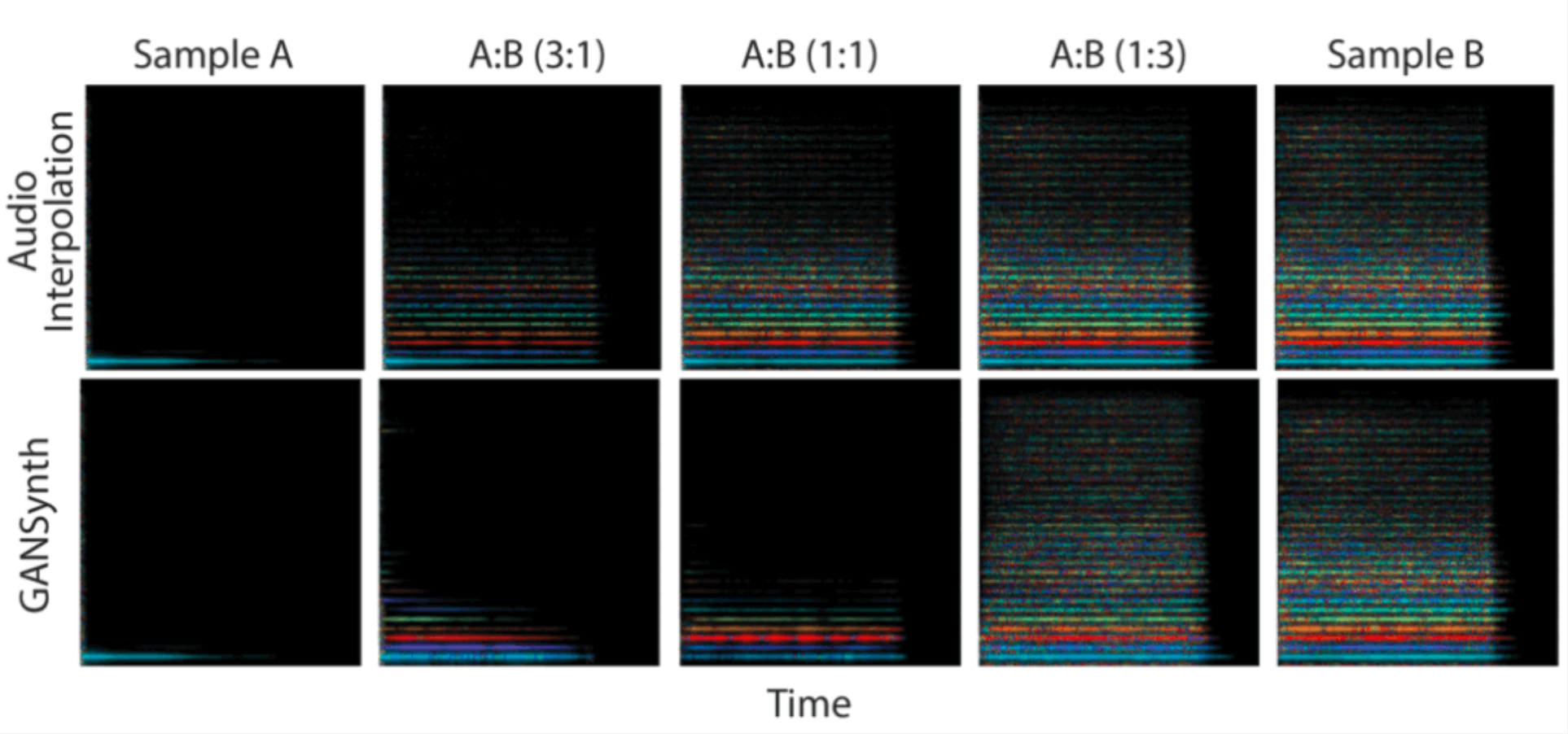
EXAMPLE: GAN SYNTH

- **Architecture**
  - ProgressiveGAN
    - Incrementally Add layers during training
  - Model samples a random vector z from a spherical Gaussian
  - **z** runs through a stack of transposed convolutions to upsample and generate output data $x = G(z)$
  - x is fed into a discriminator network of downsampling convolutions
  - A one-hot vector is appended representing musical pitch
    - *independent control of pitch and timbre*



EXAMPLE: GAN SYNTH

- **Interpolation**
  - GANs also allow conditioning on the same latent vector the entire sequence
  - Interpolating the waveform is perceptually equivalent to mixing between the amplitudes of two distinct sounds



EXAMPLE: GAN MUSIC COMPOSER

- **C-RNN-GAN** [1]: use RNNs for the generator and discriminator
- **JazzGAN** [2]: given chords, compose melody; compare 3 representations
- **Conditional LSTM-GAN** [3]: given lyrics, compose melody
- **SSMGAN** [4]: use GAN to generate a self-similarity matrix (SSM) to represent self-repetition in music, and then use LSTM to generate melody given the SSM

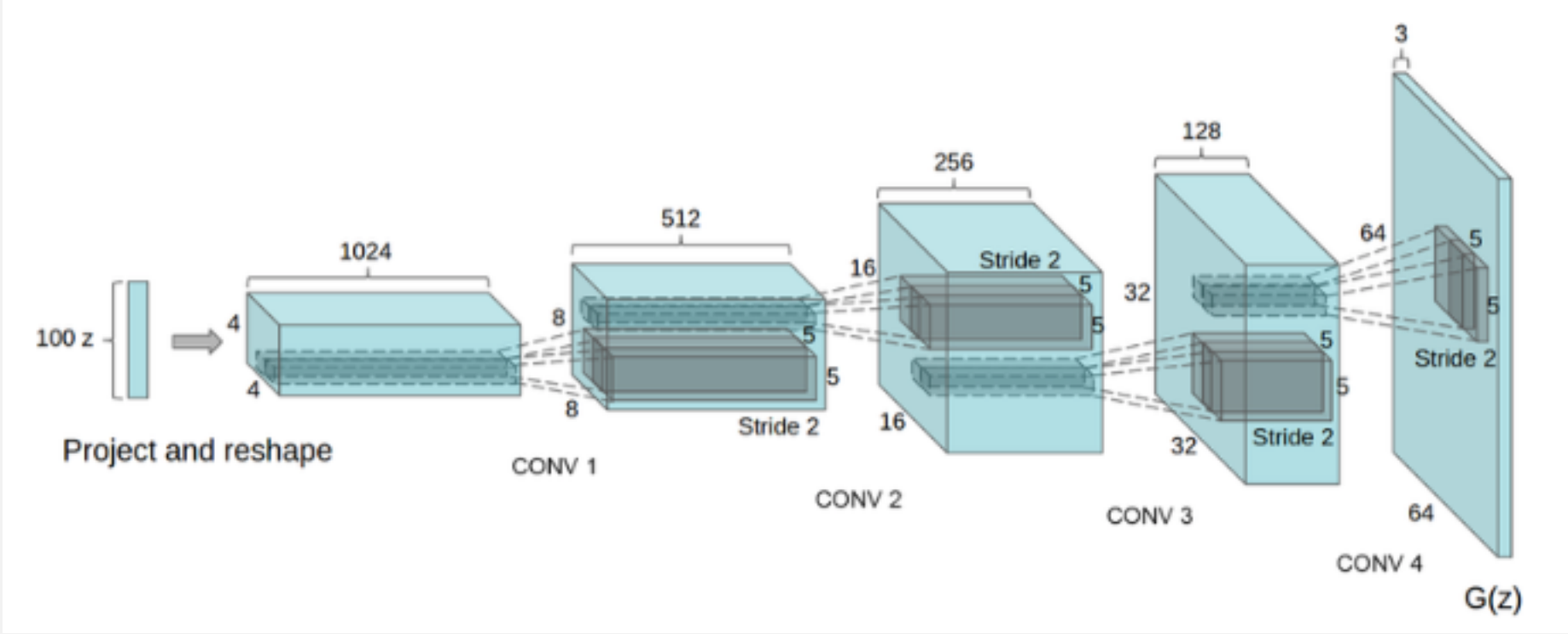
- [1] Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," CML 2016
- [2] Trieu and Keller, "JazzGAN: Improvising with generative adversarial networks," MUME 2018
- [3] Yu and Canales, "Conditional LSTM-GAN for melody generation from lyrics," arXiv 2019
- [4] Jhamtani and Berg-Kirkpatrick, "Modeling self-repetition in music generation using structured adversaries" ML4MD 2019

SOME EXAMPLES VIDEO, IMAGES, GRAPHICS

- Generate Examples for Image Datasets
- Generate Photographs of Human Faces
- Generate Realistic Photographs
- Generate Cartoon Characters
- Image-to-Image Translation
- Text-to-Image Translation
- Semantic-Image-to-Photo Translation
- Face Frontal View Generation
- Generate New Human Poses
- Photos to Emojis
- Photograph Editing
- Face Aging
- Photo Blending
- Super Resolution
- Photo Inpainting
- Clothing Translation
- Video Prediction
- 3D Object Generation

EXAMPLE: GENERATE NEW IMAGES

- *"Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" by Alec Radford, et al*
- Uses a DCGANs (Deep Convolutional Generative Adversarial Networks)
- The generation model is:



EXAMPLE: GENERATE NEW IMAGES

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

EXAMPLE: ANNA RIDLER "TULIPMANIA"

https://www.fusefactory.it/projects/artificial-botany/

EXAMPLE: GENERATE NEW IMAGES

Progressively-Growing GAN architecture released from NVIDIA and published at ICLR 2018

EXAMPLE: GENERATE NEW IMAGES

Progressively-Growing GAN architecture released from NVIDIA and published at ICLR 2018

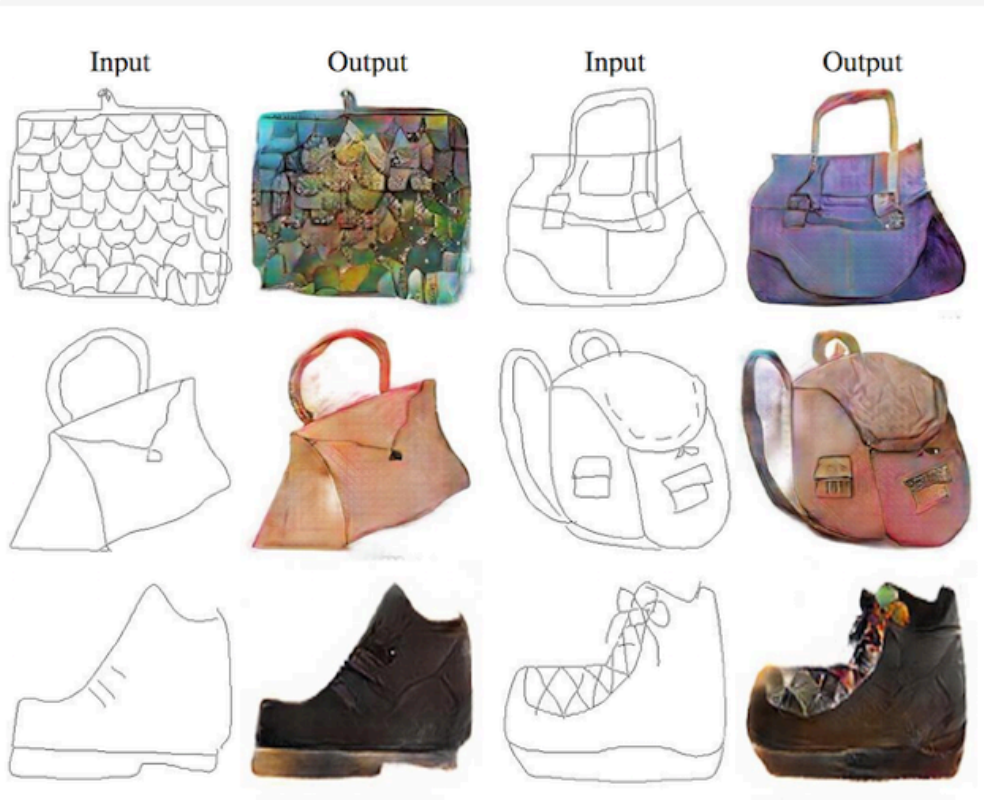
During training, new blocks of convolutional layers are systematically added to both the generator model and the discriminator models.

Generator	Act.	Output shape	Params
Latent vector	–	512 × 1 × 1	–
Conv 4 × 4	LReLU	512 × 4 × 4	4.2M
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Upsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Upsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Upsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Upsample	–	512 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	1.2M
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Upsample	–	256 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	295k
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Upsample	–	128 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	74k
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Upsample	–	64 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	18k
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Upsample	–	32 × 1024 × 1024	–
Conv 3 × 3	LReLU	16 × 1024 × 1024	4.6k
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 1 × 1	linear	3 × 1024 × 1024	51
Total trainable parameters			**23.1M**

Discriminator	Act.	Output shape	Params
Input image	–	3 × 1024 × 1024	–
Conv 1 × 1	LReLU	16 × 1024 × 1024	64
Conv 3 × 3	LReLU	16 × 1024 × 1024	2.3k
Conv 3 × 3	LReLU	32 × 1024 × 1024	4.6k
Downsample	–	32 × 512 × 512	–
Conv 3 × 3	LReLU	32 × 512 × 512	9.2k
Conv 3 × 3	LReLU	64 × 512 × 512	18k
Downsample	–	64 × 256 × 256	–
Conv 3 × 3	LReLU	64 × 256 × 256	37k
Conv 3 × 3	LReLU	128 × 256 × 256	74k
Downsample	–	128 × 128 × 128	–
Conv 3 × 3	LReLU	128 × 128 × 128	148k
Conv 3 × 3	LReLU	256 × 128 × 128	295k
Downsample	–	256 × 64 × 64	–
Conv 3 × 3	LReLU	256 × 64 × 64	590k
Conv 3 × 3	LReLU	512 × 64 × 64	1.2M
Downsample	–	512 × 32 × 32	–
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Conv 3 × 3	LReLU	512 × 32 × 32	2.4M
Downsample	–	512 × 16 × 16	–
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Conv 3 × 3	LReLU	512 × 16 × 16	2.4M
Downsample	–	512 × 8 × 8	–
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Conv 3 × 3	LReLU	512 × 8 × 8	2.4M
Downsample	–	512 × 4 × 4	–
Minibatch stddev	–	513 × 4 × 4	–
Conv 3 × 3	LReLU	512 × 4 × 4	2.4M
Conv 4 × 4	LReLU	512 × 1 × 1	4.2M
Fully-connected	linear	1 × 1 × 1	513
Total trainable parameters			**23.1M**

EXAMPLE: PIX2PIX

Pix2PIx -> Image to Image translation

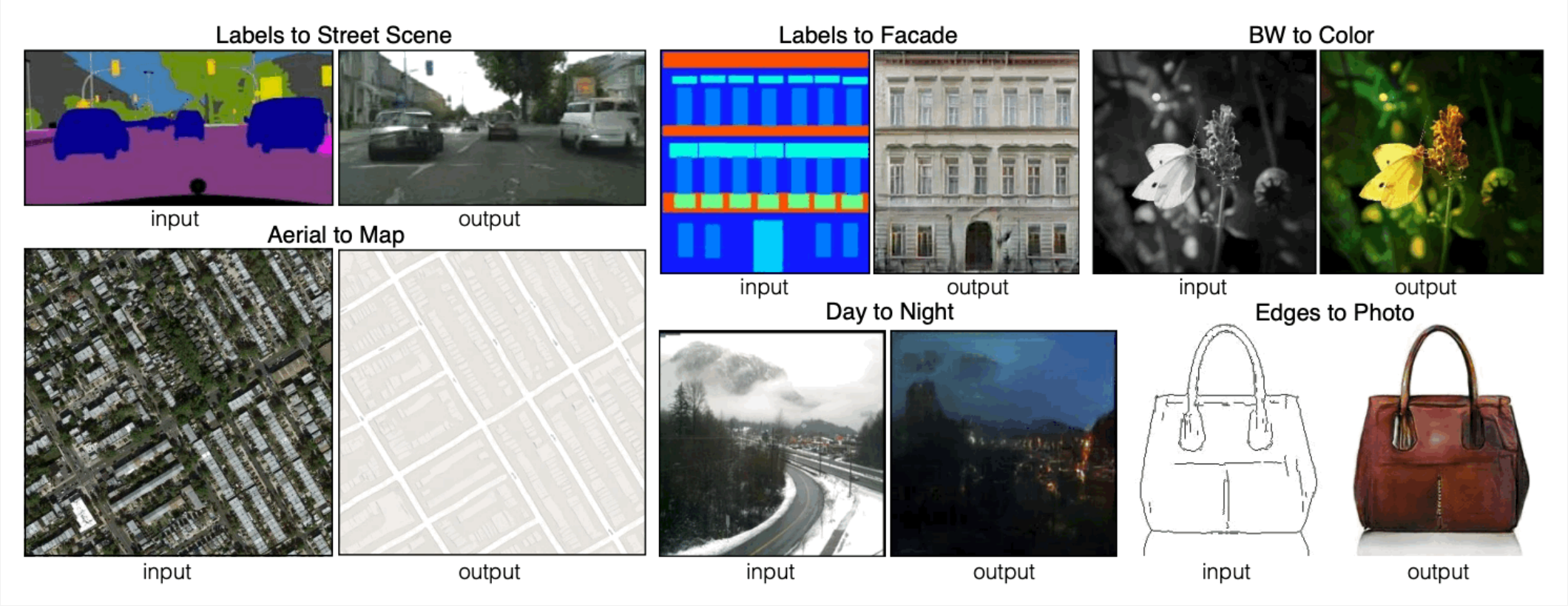


- The sketch is the condition
- Use of CNN for edge detection
- Large set of real images

https://affinelayer.com/pixsrv/

EXAMPLE: PIX2PIX

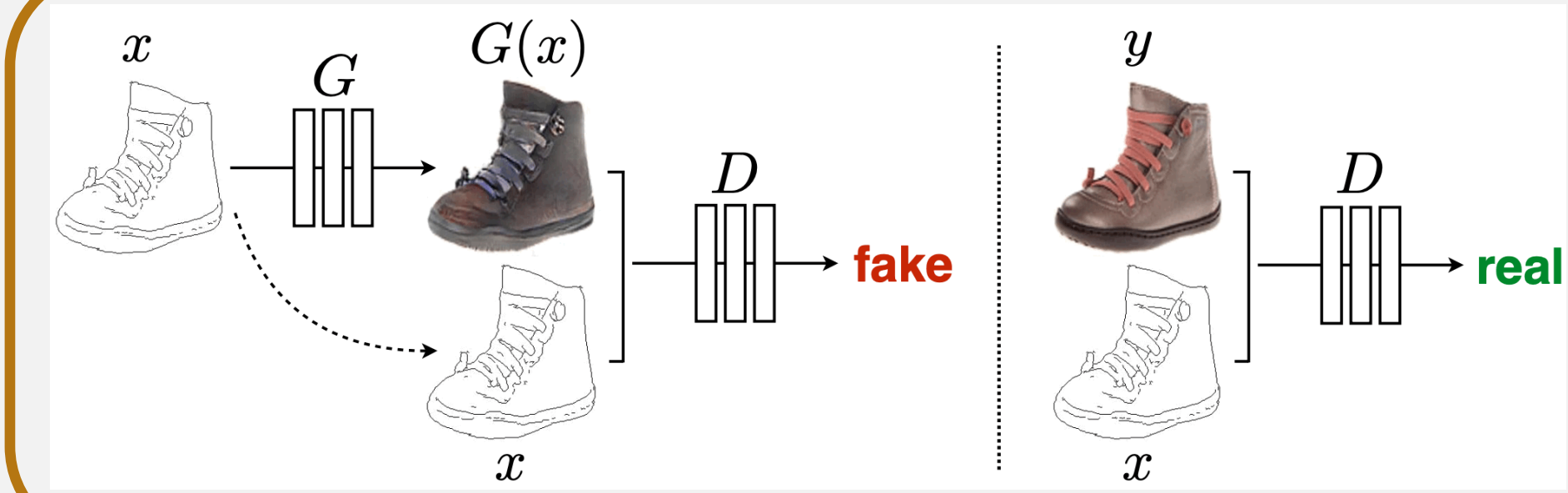
- Image-to-Image Translation with **Conditional Adversarial Networks**
  - Convert an image into another one
  - GAN is conditioned on the input image



EXAMPLE: PIX2PIX

- A naive approach such as CNN + Euclidean distance as Loss would produce blurry results
- GANs do the following:
  - Set a high level goal: ”make output indistinguishable from reality”
  - Automatically learn the loss function to do it

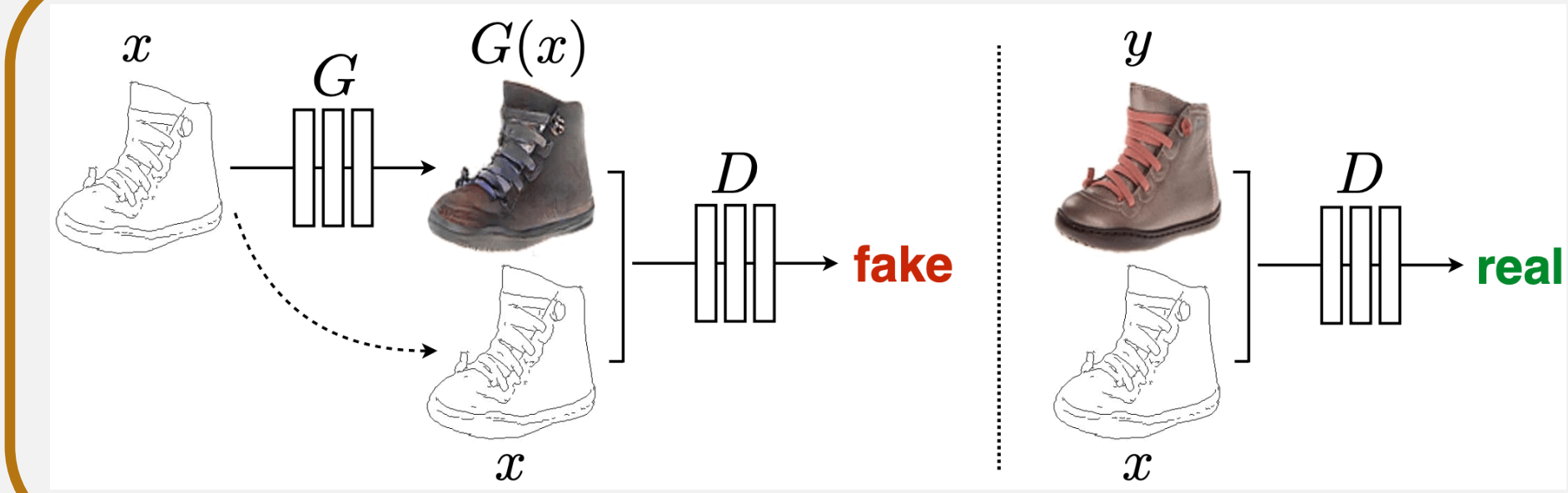
Training Procedure



EXAMPLE: PIX2PIX

- A naive approach such as CNN + Euclidean distance as Loss would produce blurry results
- GANs do the following:
  - Set a high level goal: ”make output indistinguishable from reality”
  - Automatically learn the loss function to do it

Training Procedure



EXAMPLE: PIX2PIX

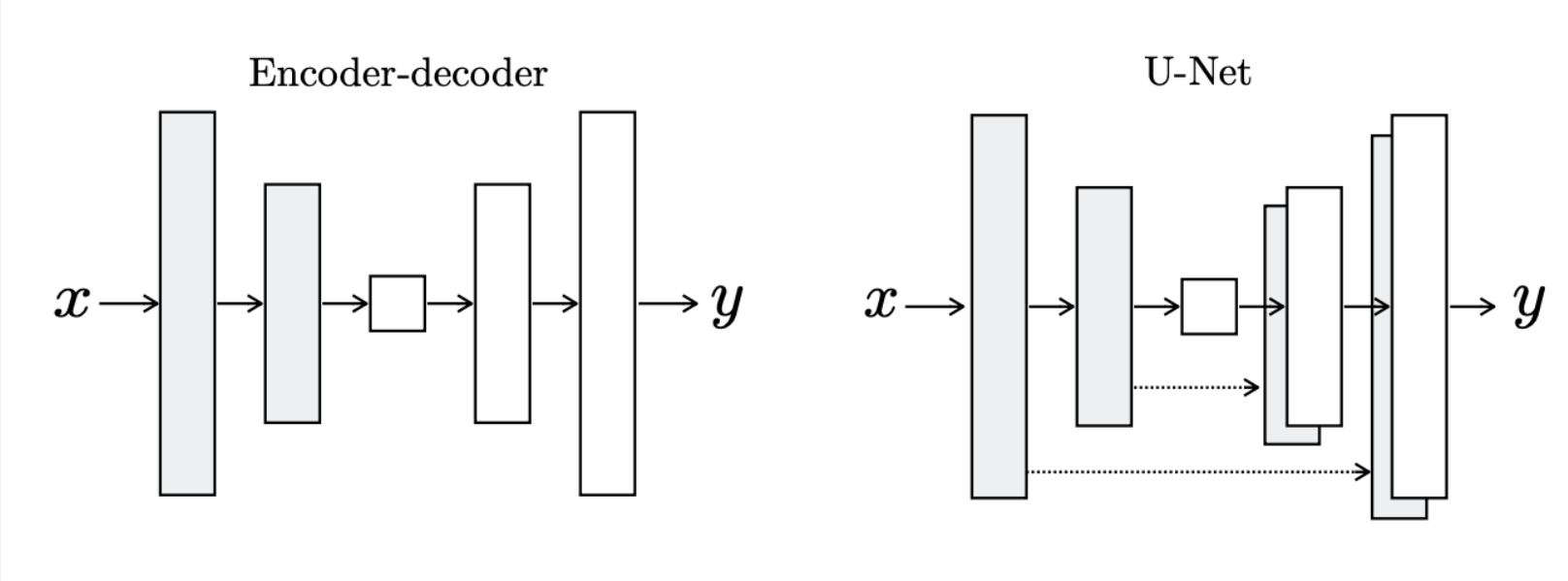
- Objective of a conditional GAN:

$$\mathcal{L}_{cGAN}(G, D) = \mathbb{E}_{x,y}[\log D(x, y)] + \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z)))]$$

- The conditioning is given by the fact that the discriminator observes both input and desired image

EXAMPLE: PIX2PIX

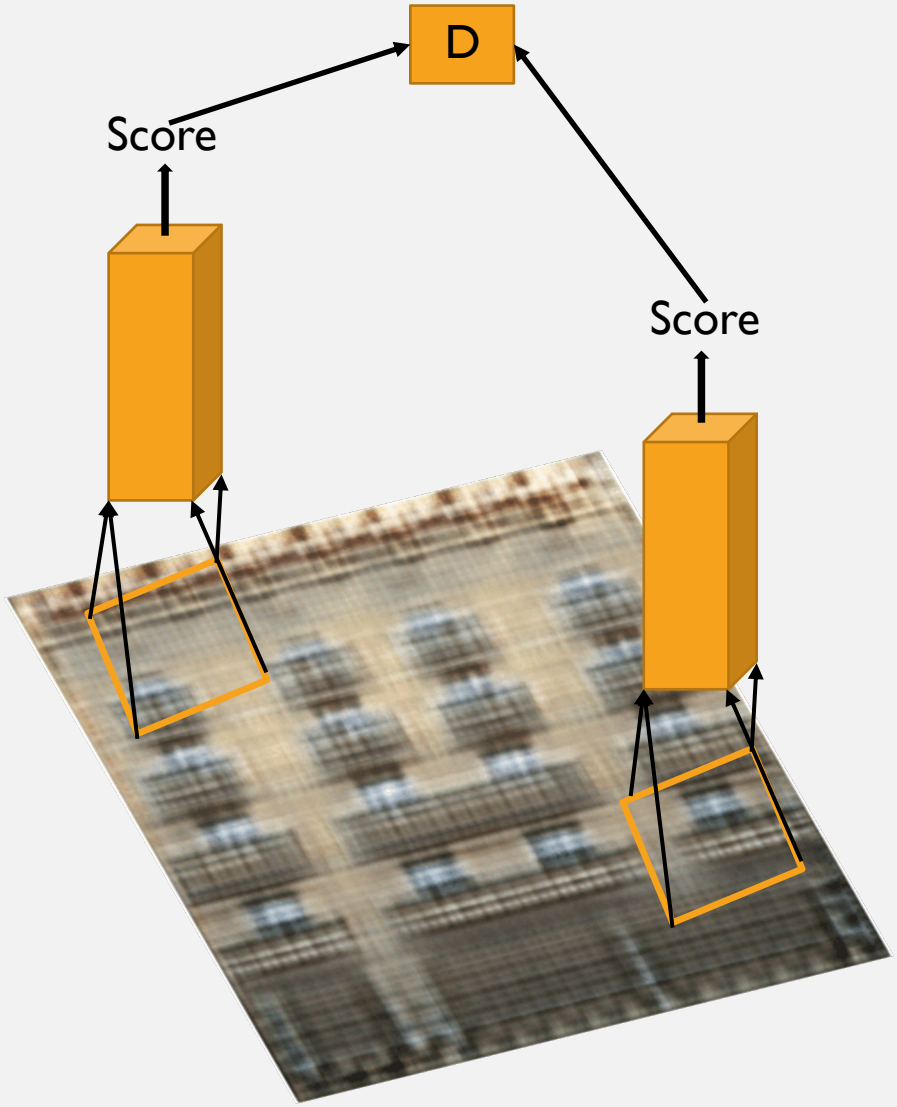
- **Generator**:
  - **U-Net** architecture
  - Autoencoder with skipped connections
  - both input and output are renderings of the same underlying structure



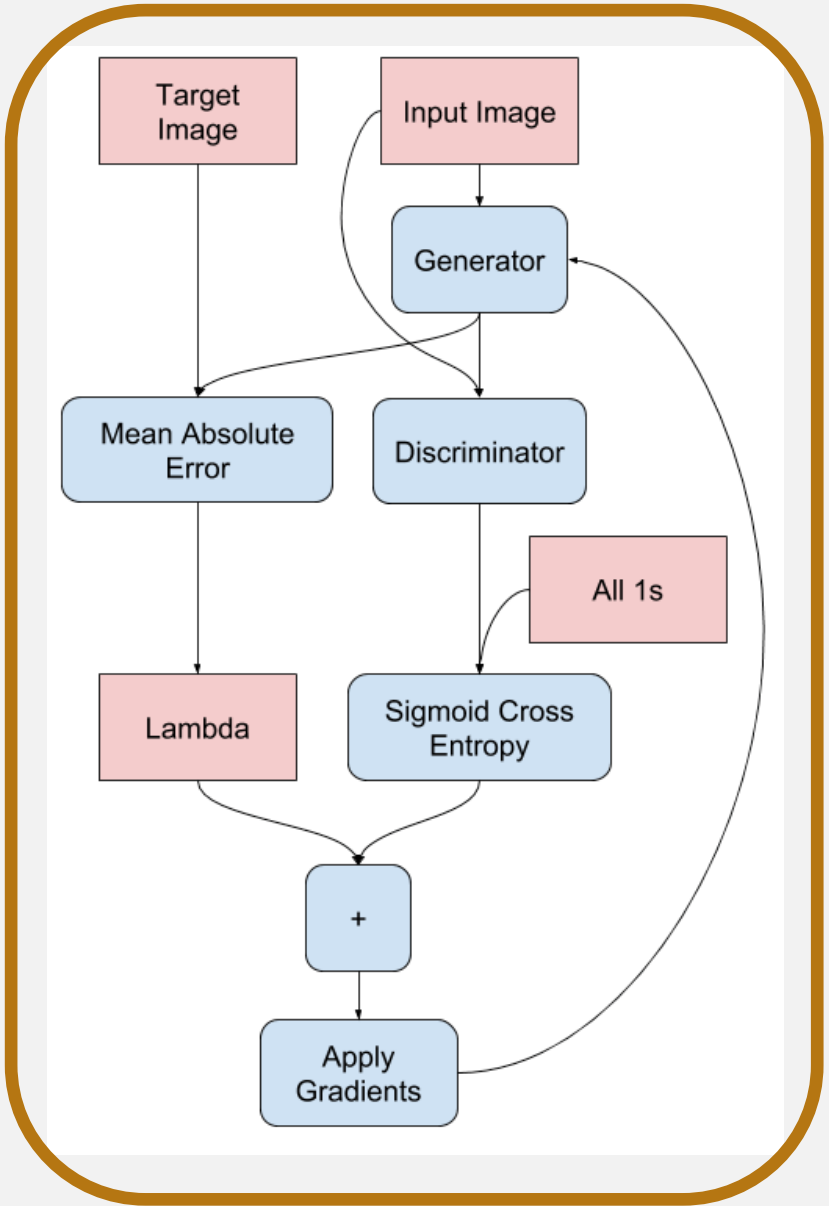
- In many image translation problems a lot of low-level information is shared between input and output
- Skip connections between each layer i and $n - i$ cirmunvents this issue

EXAMPLE: PIX2PIX

- **Discriminator**:
  - Patch GAN
    - Penalizes structure at the scale of patches
    - Tries to classify if each $N \times N$ patch in an image is real or fake
    - Run across the image, averaging all responses to provide ultimate output of the Discriminator
    - Can be understood as a form of texture/style loss

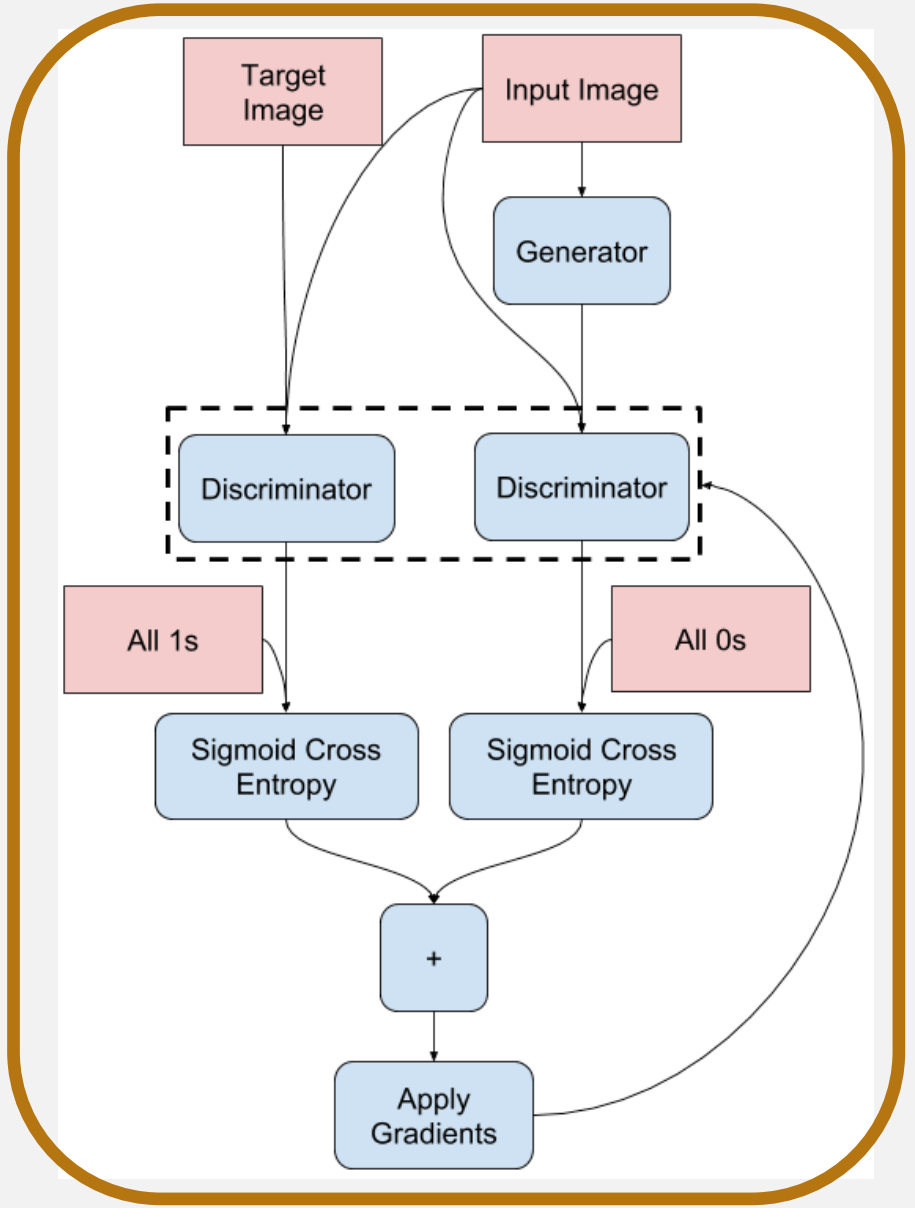


EXAMPLE: PIX2PIX



Generator training

Discriminator training



EXAMPLE: PIX2PIX

EXAMPLE: PIX2PIX

Background removal

by Kaihu Chen

Palette generation

by Jack Qiao

Sketch→ Portrait

by Mario Klingemann

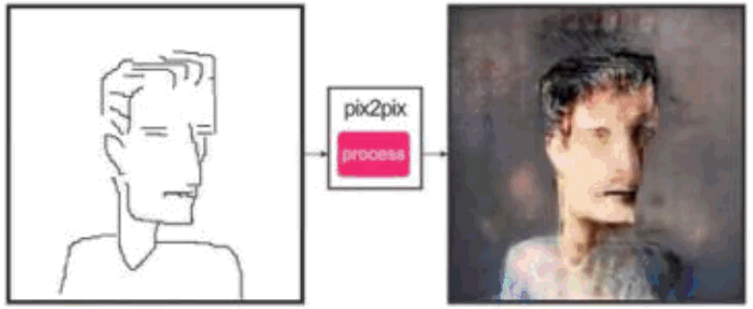
Sketch → Pokemon

by Bertrand Gondouin

"Do as I do"

by Brannon Dorsey

#fotogenerator



sketch by Yann LeCun

REFERENCES

- **Magenta**
  - https://magenta.tensorflow.org/research
- **Performance RNN**
  - Original paper
  - Blog post
  - Web Demo
- **Sketch RNN**
  - Original paper
  - Blog post
  - Demo
  - Sketch RNN and tensorflow.js
- **Pix2Pix**
  - Original paper
  - Tensorflow tutorial
  - Image-to-image Demo
- **GanSynth**
  - Original paper
  - Blog post
  - Audio Examples