Autoencoders and Generative Adversarial Networks

Giacomo Boracchi

Advanced Neural Networks and Deep Learning

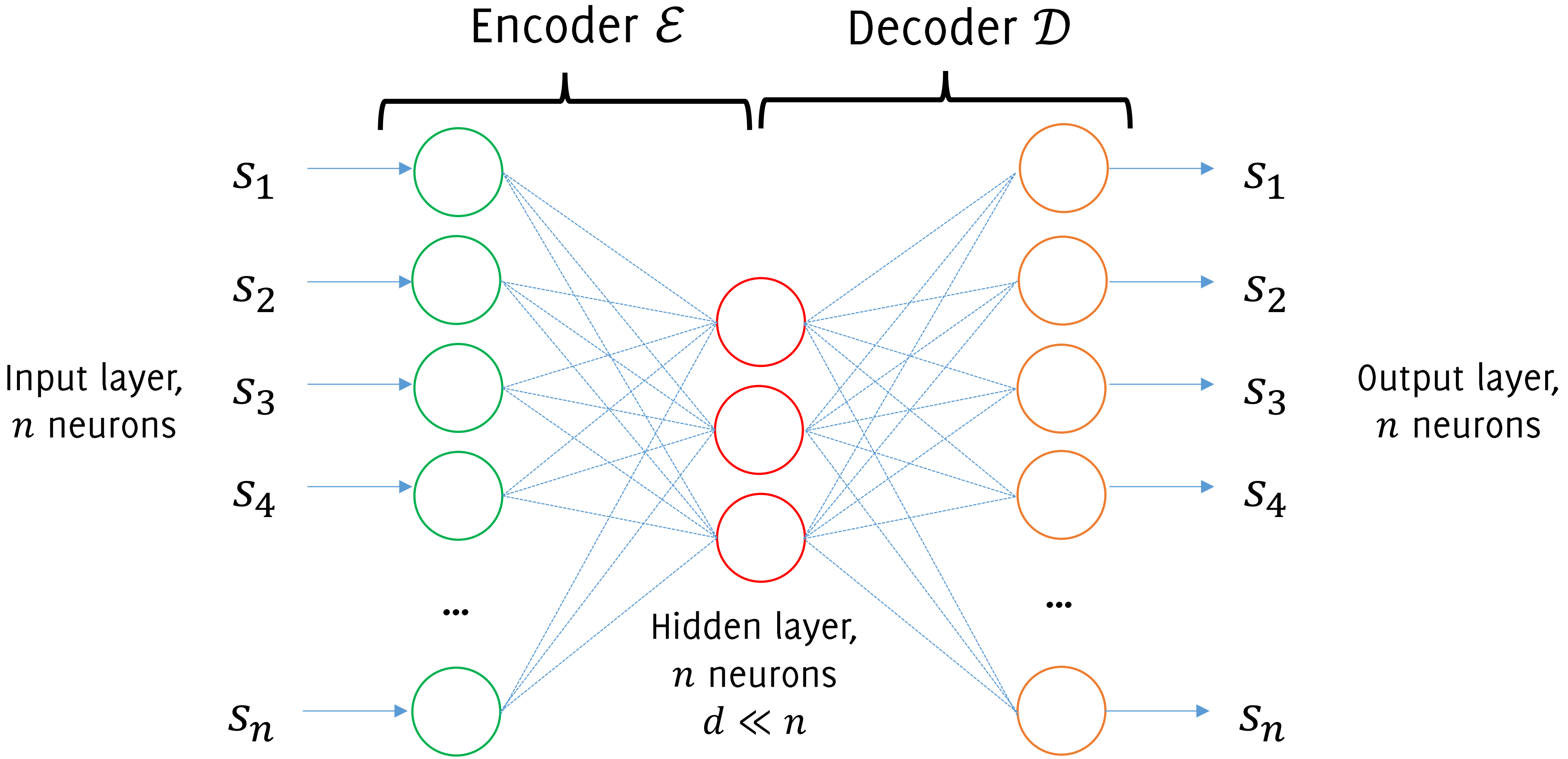
https://boracchi.faculty.polimi.it/

Autoencoders

Autoencoders using MLP

Autoencoders are neural networks used for data reconstruction (unsupervised learning)

The typical structure of an autoencoder is:



Autoencoders using MLP

Autoencoders can be trained to reconstruct all the data in a training set.

The reconstruction loss over a batch S is

$$\ell(S) = \sum_{\boldsymbol{s} \in S} \left\| \boldsymbol{s} - \mathcal{D}\big(\mathcal{E}(\boldsymbol{s})\big) \right\|_2$$

and training of $\mathcal{D}\big(\mathcal{E}(\cdot)\big)$ is performed through standard backpropagation algorithms (e.g. SGD).

The autoencoder thus learns the identity mapping.

Rmk there are **no external labels** involved in training the autoencoder, as it performs reconstruction of the input

Bengio, Y., Courville, A., Vincent, P. *"Representation learning: A review and new perspectives"*. IEEE TPAMI 2013

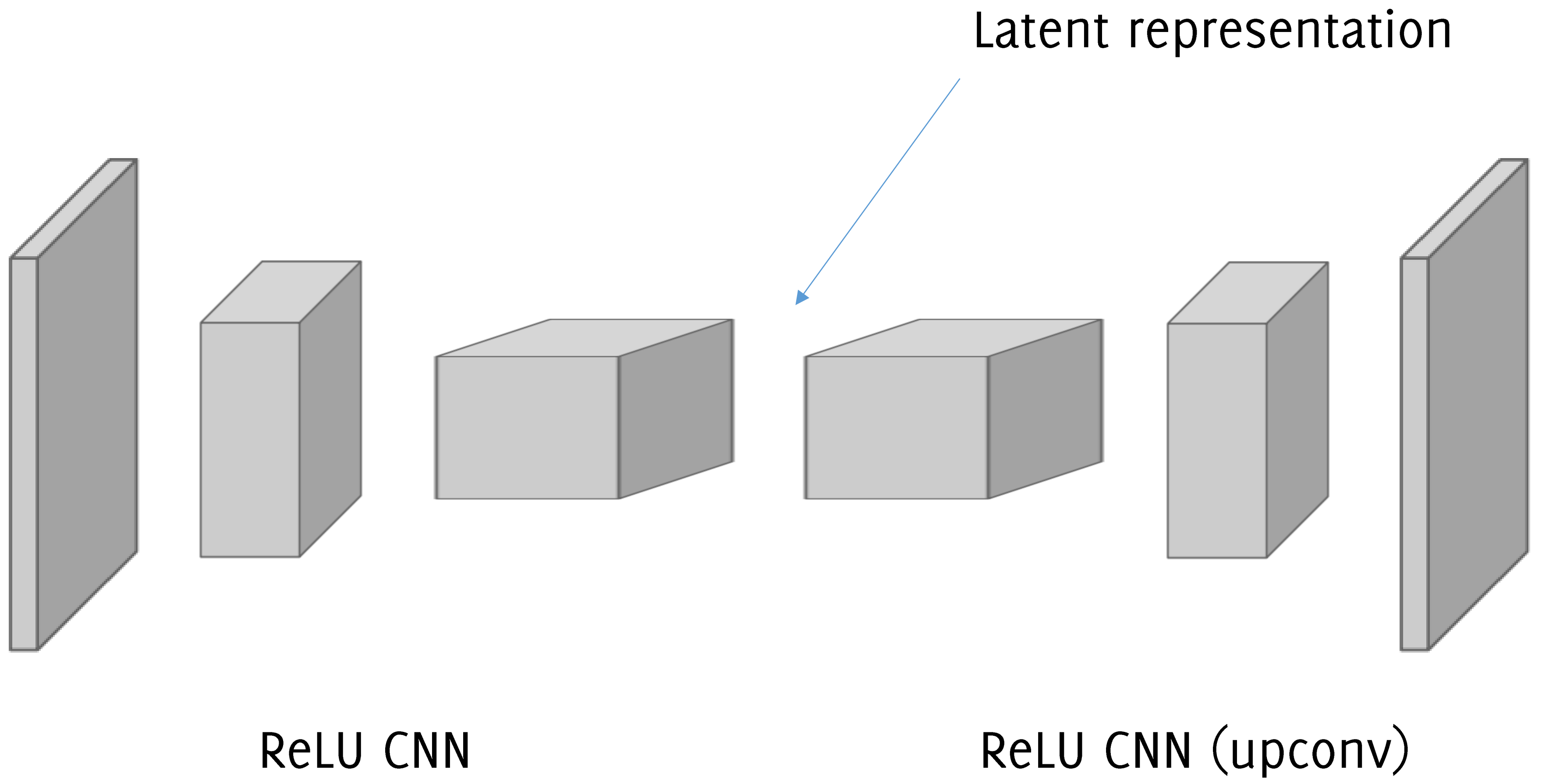
Autoencoders

Remark:

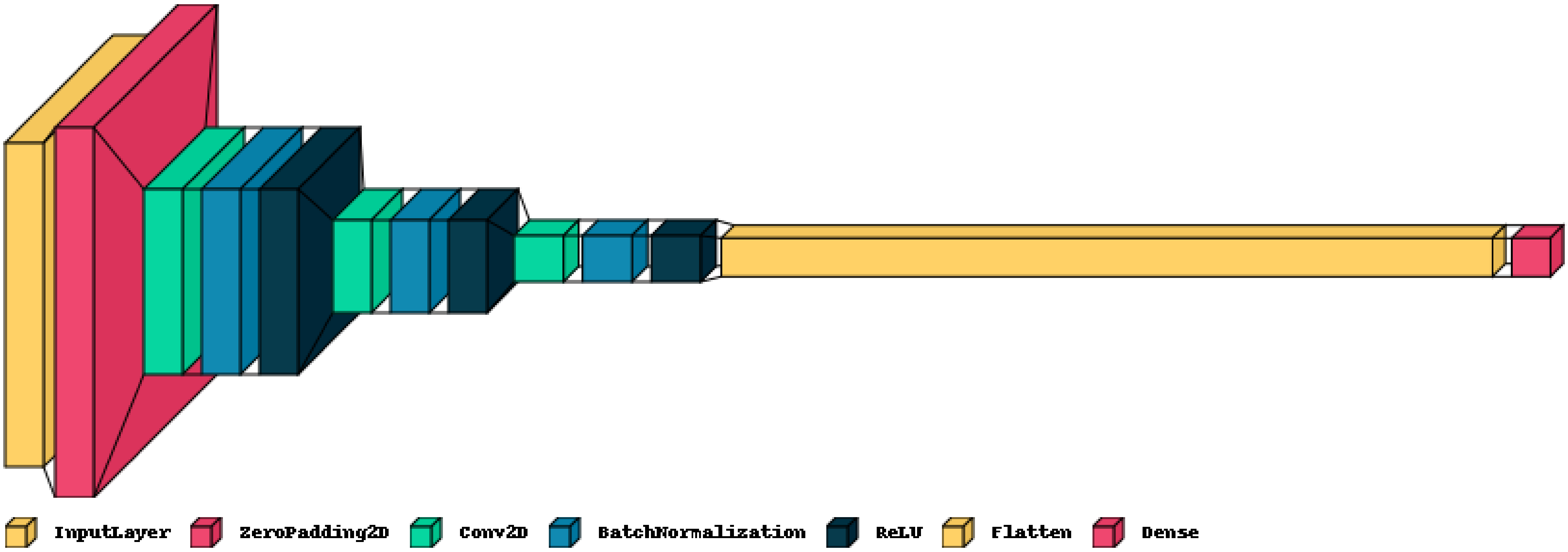
- Features $z = \mathcal{E}(s)$ are typically referred to as **latent representation**
- AE typically do not provide exact reconstruction since $n \ll d$, by doing so we **expect the latent representation to be a meaningful and compact representation** of the input
- **It is possible to add a regularization term** $+\lambda\mathcal{R}(s)$ to steer latent representation $\mathcal{E}(s)$ to satisfy desired properties (e.g. sparsity, or to follow a Gaussian distribution) or the reconstruction $\mathcal{D}(\mathcal{E}(s))$ (e.g. smoothness, sharp edges in case of images)
- More powerful and nonlinear representations can be learned by **stacking multiple hidden layers** (deep autoencoders)

Convolutional AutoEncoders

And of course it is possible to use convolutional layers and transpose convolution to implement a deep convolutional autoencoder



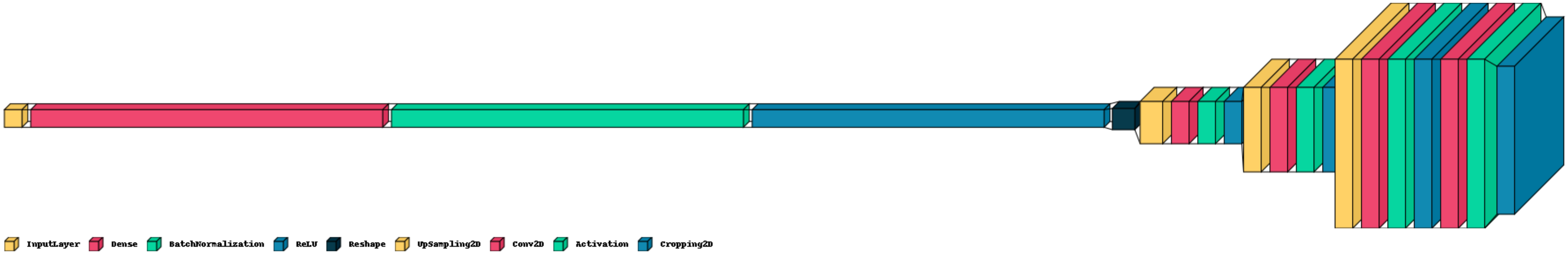
Training Autoencoders


InputLayer
ZeroPadding2D
Conv2D
BatchNormalization
ReLU
Flatten
Dense

G. Boracchi

Code for the encoder function

```
input_layer = tfkl.Input(shape=enc_input_shape, name='input_layer')

# block of conv+batchnorm+relu
x = tfkl.Conv2D(64, 3, padding='same', strides=2)(input_layer)
x = tfkl.BatchNormalization()(x)
x = tfkl.ReLU()(x)

# Another block of conv+batchnorm+relu

# Another block of conv+batchnorm+relu

# flattening and a dense layer to the latent_dim
x = tfkl.Flatten()(x)
output_layer = tfkl.Dense(enc_output_shape, name='output_layer')(x)

# the value returned by the output layer is the latent representation

# Connect input and output through the Model class
model = tfk.Model(inputs=input_layer, outputs=output_layer, name='encoder')
```


InputLayer
Dense
BatchNormalization
ReLU
Reshape
UpSampling2D
Conv2D
Activation
Cropping2D

G. Boracchi

Code for the decoder function

```
input_layer = tfkl.Input(shape=dec_input_shape, name='input_layer')

# adda a dense layer from the latent representation to a larger vector
x = tfkl.Dense(n_rows*n_cols*n_channels)(input_layer)
x = tfkl.BatchNormalization()(x)
x = tfkl.ReLU()(x)

# invert the flattening by reshaping
x = tfkl.Reshape((n_rows, n_cols, n_channels))(x)

# upsampling block: upsampling + convolution + batchnorm + relu
x = tfkl.UpSampling2D()(x)
x = tfkl.Conv2D(128, 3, padding='same')(x)
x = tfkl.BatchNormalization()(x)
x = tfkl.ReLU()(x)
```

Code for the decoder function

```
# Another upsampling block: upsampling + convolution + batchnorm + relu

# Another upsampling block: upsampling + convolution + batchnorm + relu

# the last block is a convolution returning to the image domain
x = tfkl.Conv2D(dec_output_shape[-1], 3, padding='same')(x)

x = tfkl.Activation('sigmoid')(x) #’ by doing so we clip values,
                                     linear is also fine

# Connect input and output through the Model class
model = tfk.Model(inputs=input_layer, outputs=output_layer, name='decoder')
```

Code for the autoencoder

```
def get_autoencoder(ae_input_shape=input_shape, ae_output_shape=input_shape):
    tf.random.set_seed(seed)

    # invoke functions to instantiate models
    encoder = get_encoder()
    decoder = get_decoder()

    # assemble the network
    input_layer = tfkl.Input(shape=ae_input_shape)
    z = encoder(input_layer)
    output_layer = decoder(z)

    model = tfk.Model(inputs=input_layer, outputs=output_layer, name='autoencoder')
    return model

# instantiate the autoencoder
autoencoder = get_autoencoder()
autoencoder.summary()
tfk.utils.plot_model(autoencoder, show_shapes=True, expand_nested=True, to_file='au
toencoder.png')
```

Training the autoencoder

```
# define training options
learning_rate = 1e-3
optimizer = tf.optimizers.Adam(learning_rate)

# the autoencoder needs to be trained by minimizing a
reconstruction loss
autoencoder.compile(optimizer=optimizer, loss=tfk.losses
.MeanSquaredError(), metrics=['mse', 'mae'])
```

Training the autoencoder

```
# define training options
learning_rate = 1e-3
optimizer = tf.optimizers.Adam(learning_rate)

# the autoencoder needs to be trained by minimizing a
reconstruction loss
autoencoder.compile(optimizer=optimizer, loss=tfk.losses
.MeanSquaredError(), metrics=['mse', 'mae'])
```

Training the autoencoder

```
# train the autoencoder
autoencoder.fit(
    X_train, # the input
    X_train, # the target for the autoencoder is the input itself
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_val,X_val),
    # the target for the autoencoder is the input itself
    callbacks=[tfk.callbacks.EarlyStopping(monitor='val_loss',
                  patience=10, restore_best_weights=True),
               tfk.callbacks.ReduceLROnPlateau(monitor='val_loss',
                  patience=5, factor=0.5, min_lr=1e-5),
              ]
)
```

Training the autoencoder

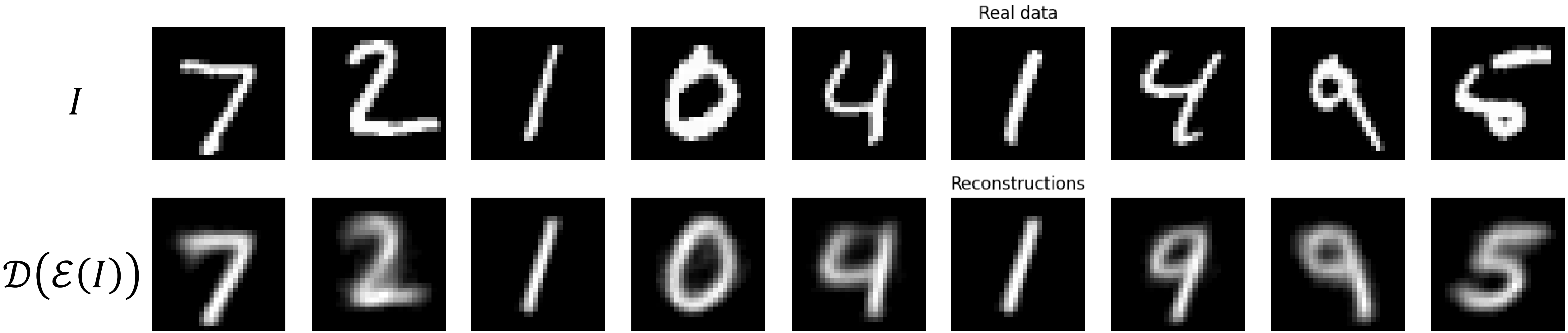
```
# train the autoencoder
autoencoder.fit(
    X_train, # the input
    X_train, # the target for the autoencoder is the input itself
    batch_size=batch_size,
    epochs=epochs,
    validation_data=(X_val,X_val),
    # the target for the autoencoder is the input itself
    callbacks=[tfk.callbacks.EarlyStopping(monitor='val_loss',
                  patience=10, restore_best_weights=True),
               tfk.callbacks.ReduceLROnPlateau(monitor='val_loss',
                  patience=5, factor=0.5, min_lr=1e-5),
              ]
)
```

The Latent Representation

The size of the latent representation

The larger the latent representation, the better images are reconstructed.

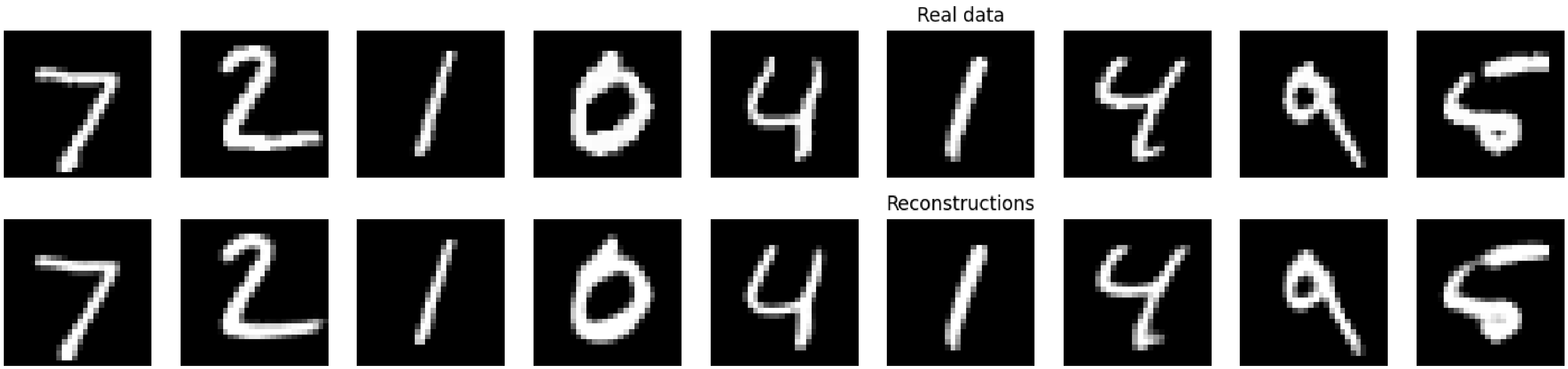
Reconstructions from $\mathcal{D}(\mathcal{E}(\cdot))$ trained on latent space having dimension $d = 2$



The size of the latent representation

The larger the latent representation, the better images are reconstructed.

Reconstructions from $\mathcal{D}(\mathcal{E}(\cdot))$ trained on latent space having dimension $d = 32$



I

$\mathcal{D}(\mathcal{E}(I))$

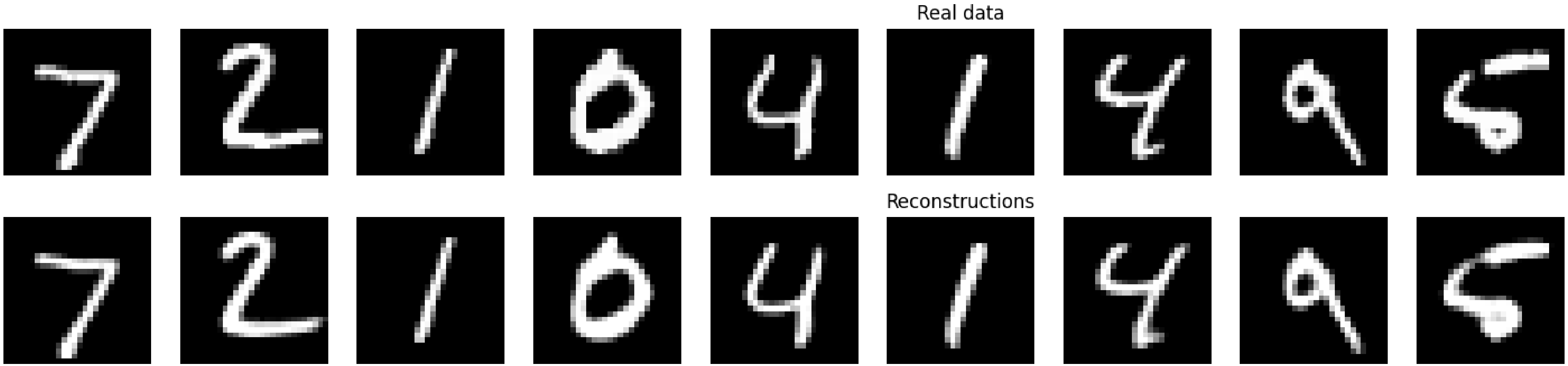
The size of the latent representation

The larger the latent representation, the better images are reconstructed.

Reconstructions from $\mathcal{D}(\mathcal{E}(\cdot))$ trained on latent space having dimension $d = 32$

I

$\mathcal{D}(\mathcal{E}(I))$



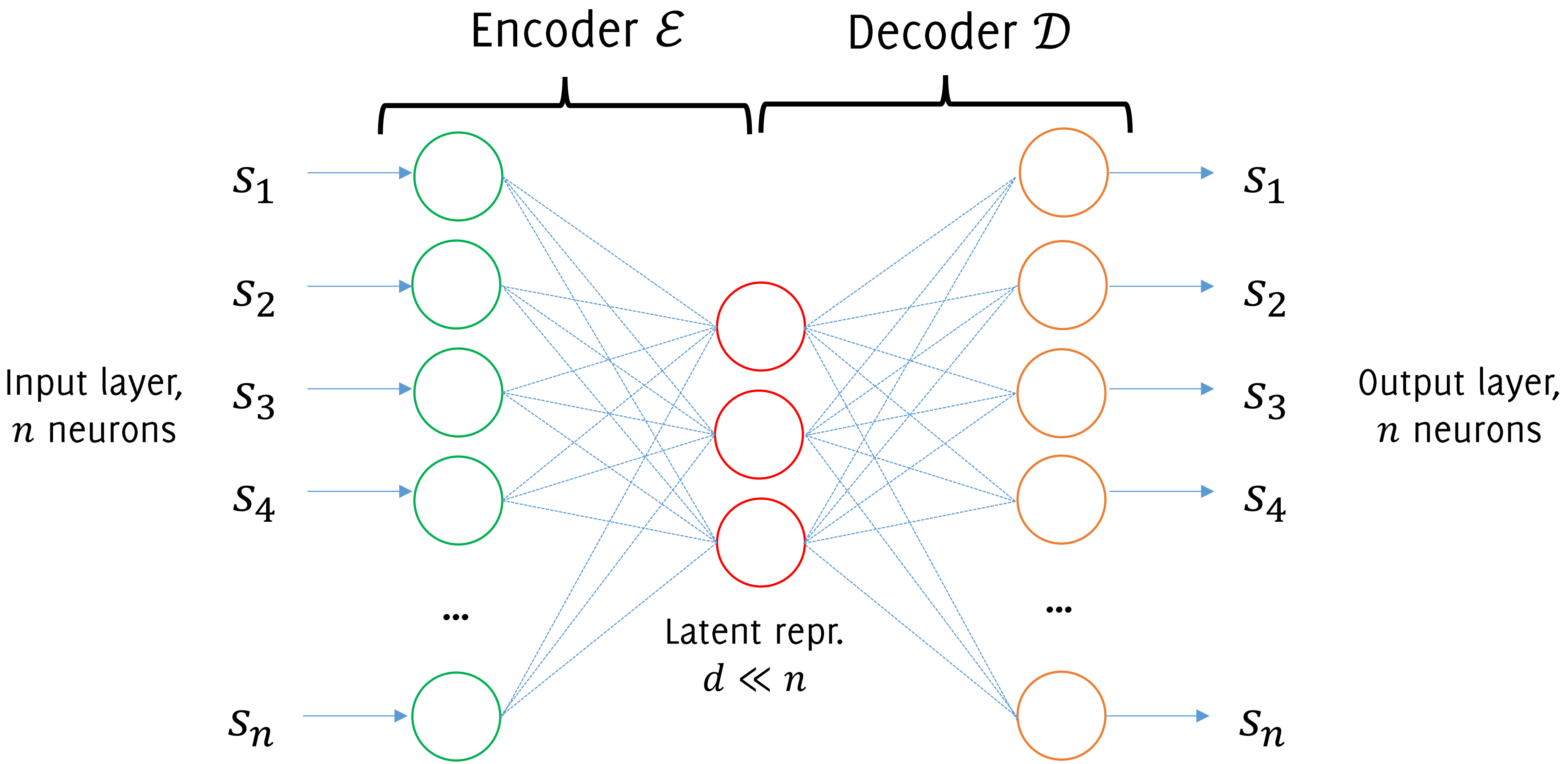
and, as a limit, when the latent dimension is as big as the input you can perfectly reconstruct it (learning the identity mapping from network input to network output)

Autoencoders for Classifier Initialization

Using Autoencoders for classifier initialization

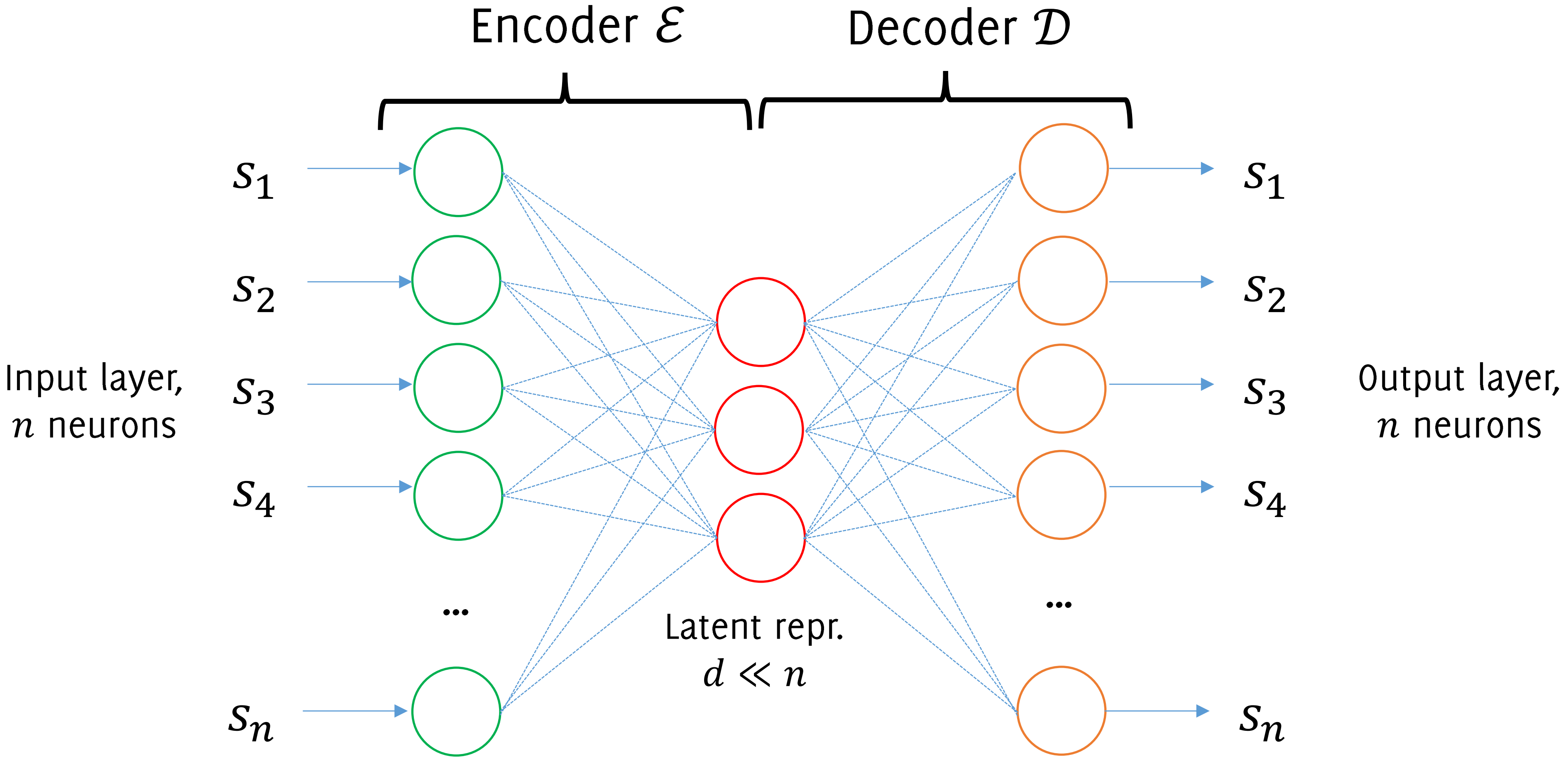
Autoencoders can be used to **inizialize the classifier when** the training set includes

- **few annotated data** (large set $S = \{s_i\}$ of unlabeled human faces)
- **many unlabeled ones** (small set $L = \{(s_i, y_i)\}$ of faces labelled as *Male, Female*)



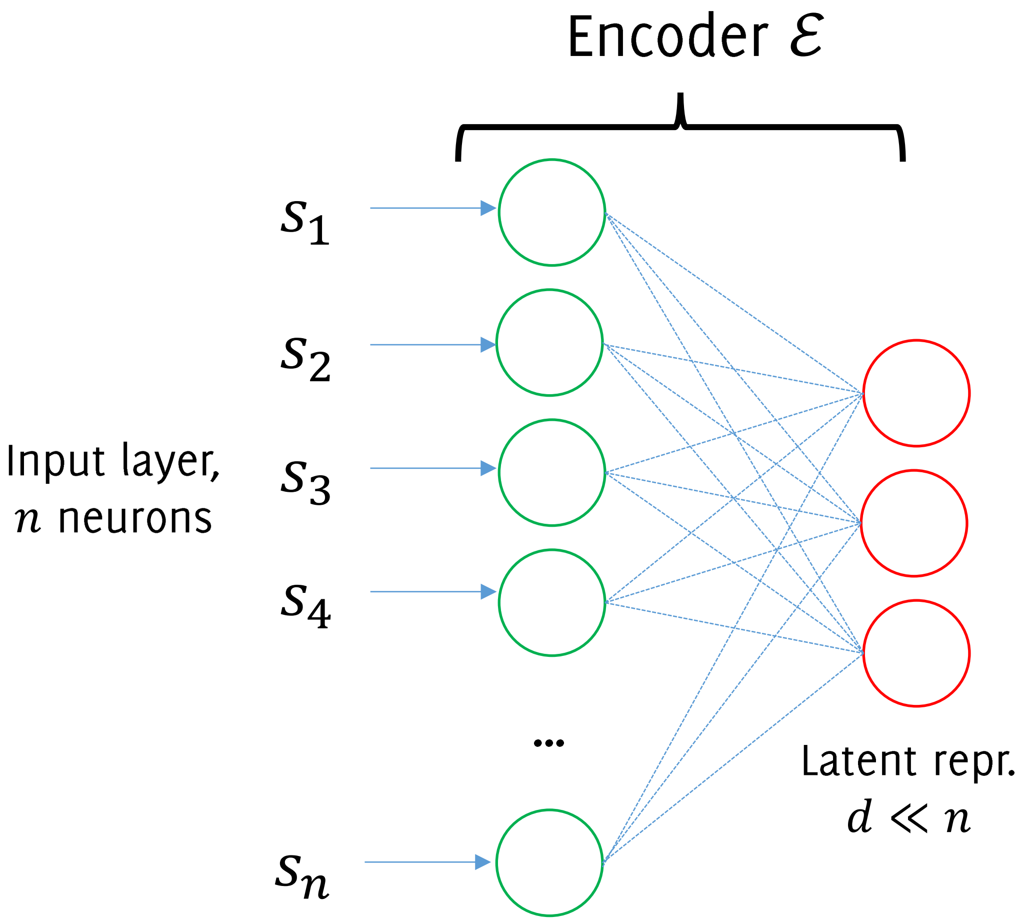
Using Autoencoders for classifier initialization

1) **Train the autoencoder** in a fully unsupervised way, using the unlabeled data S



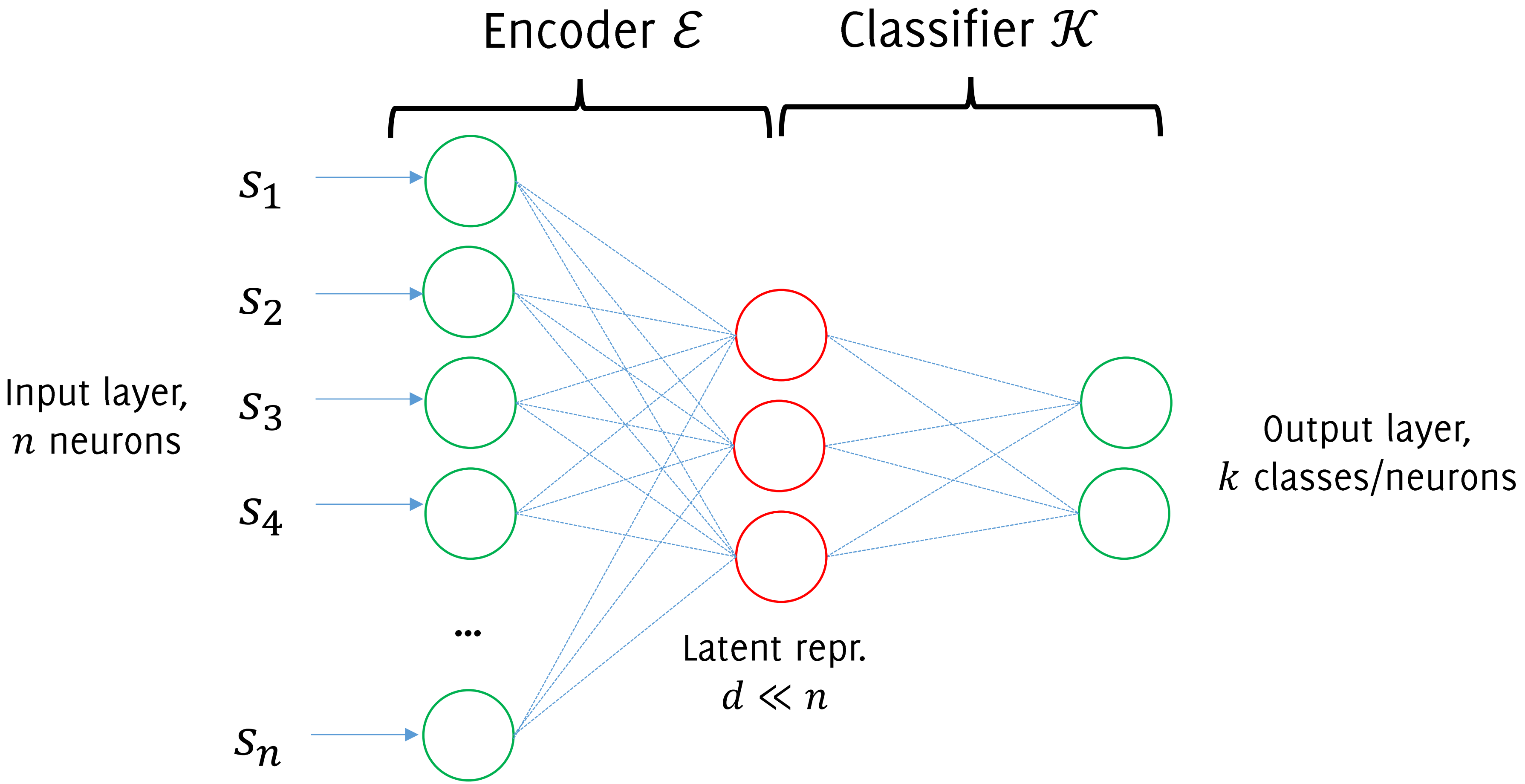
Using Autoencoders for classifier initialization

2) **Get rid of the decoder** and keep the encoder weights



Using Autoencoders for classifier initialization

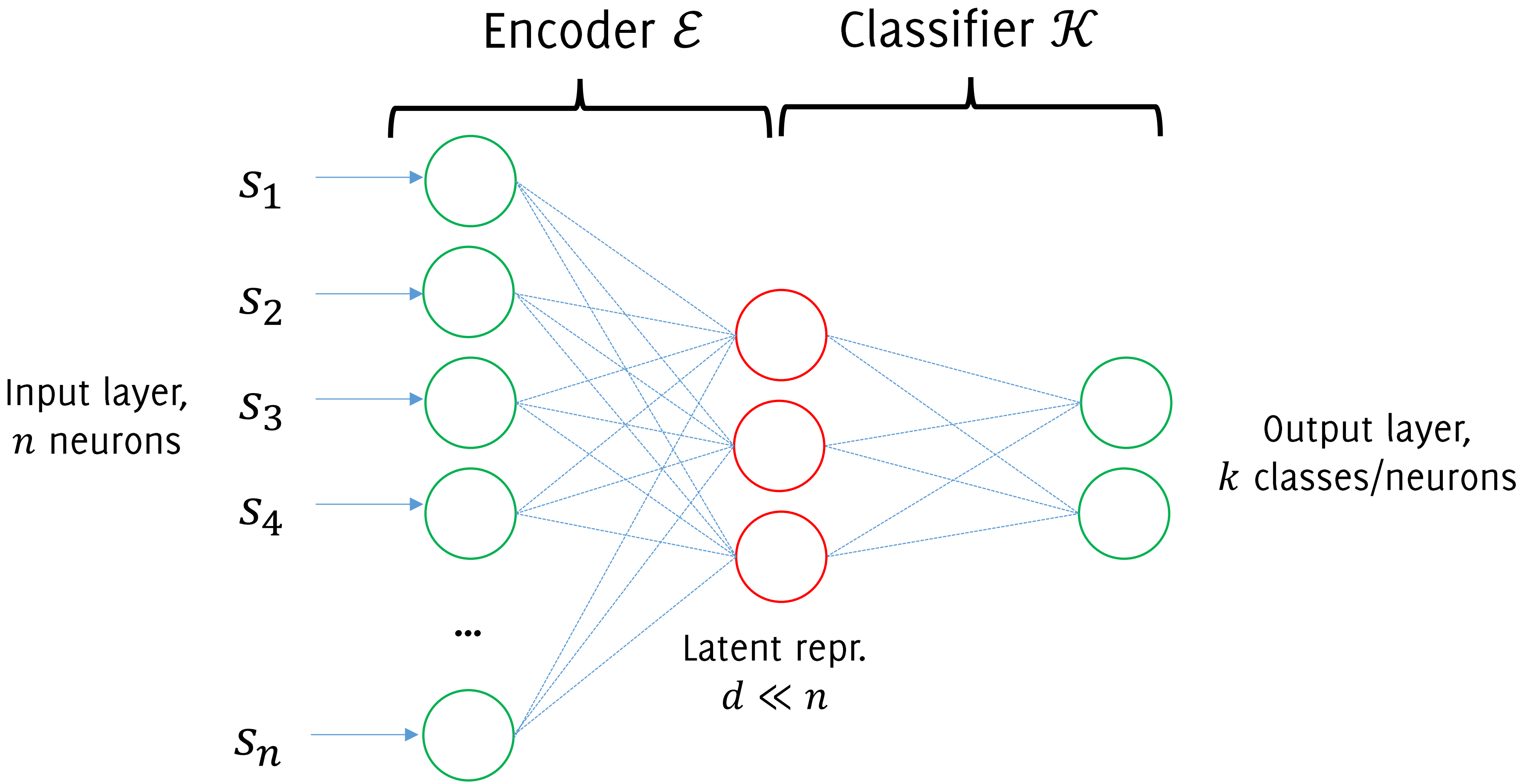
3) **Plug in a FC layer for classifying** samples from the latent representation



Using Autoencoders for classifier initialization

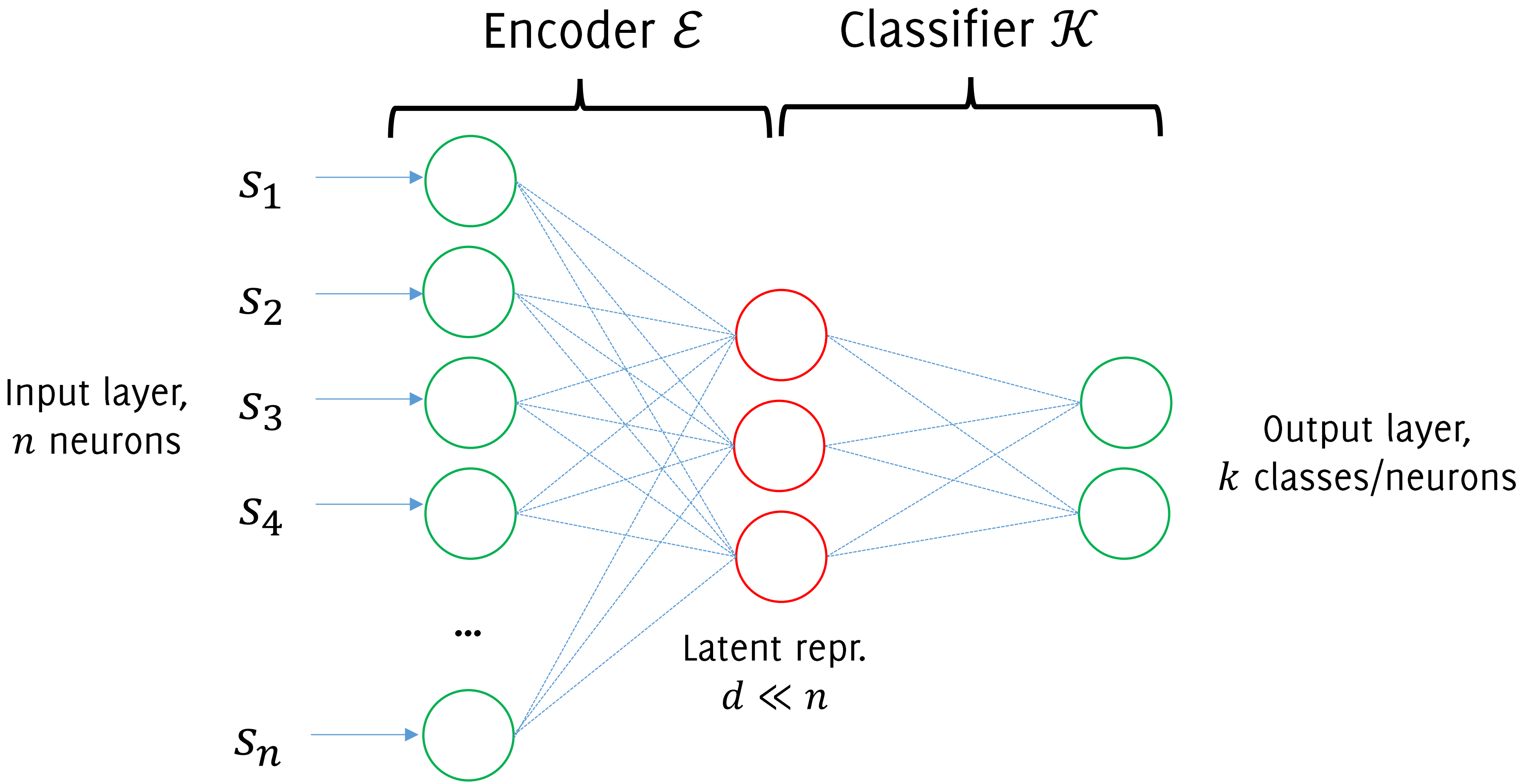
4) **Fine tune** the autoencoder using the few supervised samples provided L. This is perfectly in line with «Transfer Learning» and holds for whatever model.

If L is large enough, the encoder weights $\mathcal{E}$ can also be fine-tuned



Using Autoencoders for classifier initialization

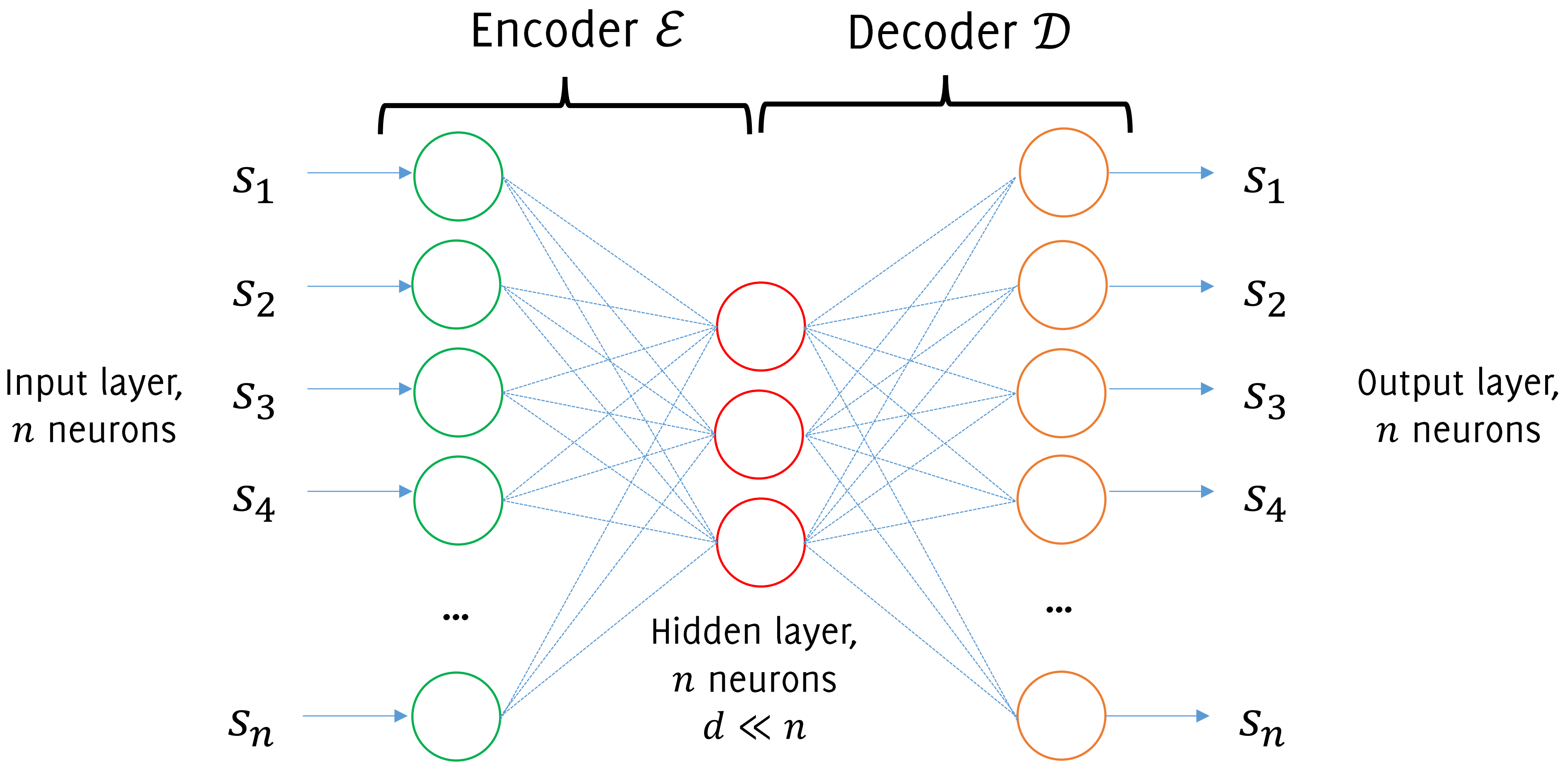
Autoencoders provide a **good initialization** (and reduce the risk of overfitting) because their latent vector is actually **a good (latent) representation** of the inputs used for training.



Autoencoders using MLP

Autoencoders are neural networks used for data reconstruction (unsupervised learning)

The typical structure of an autoencoder is:

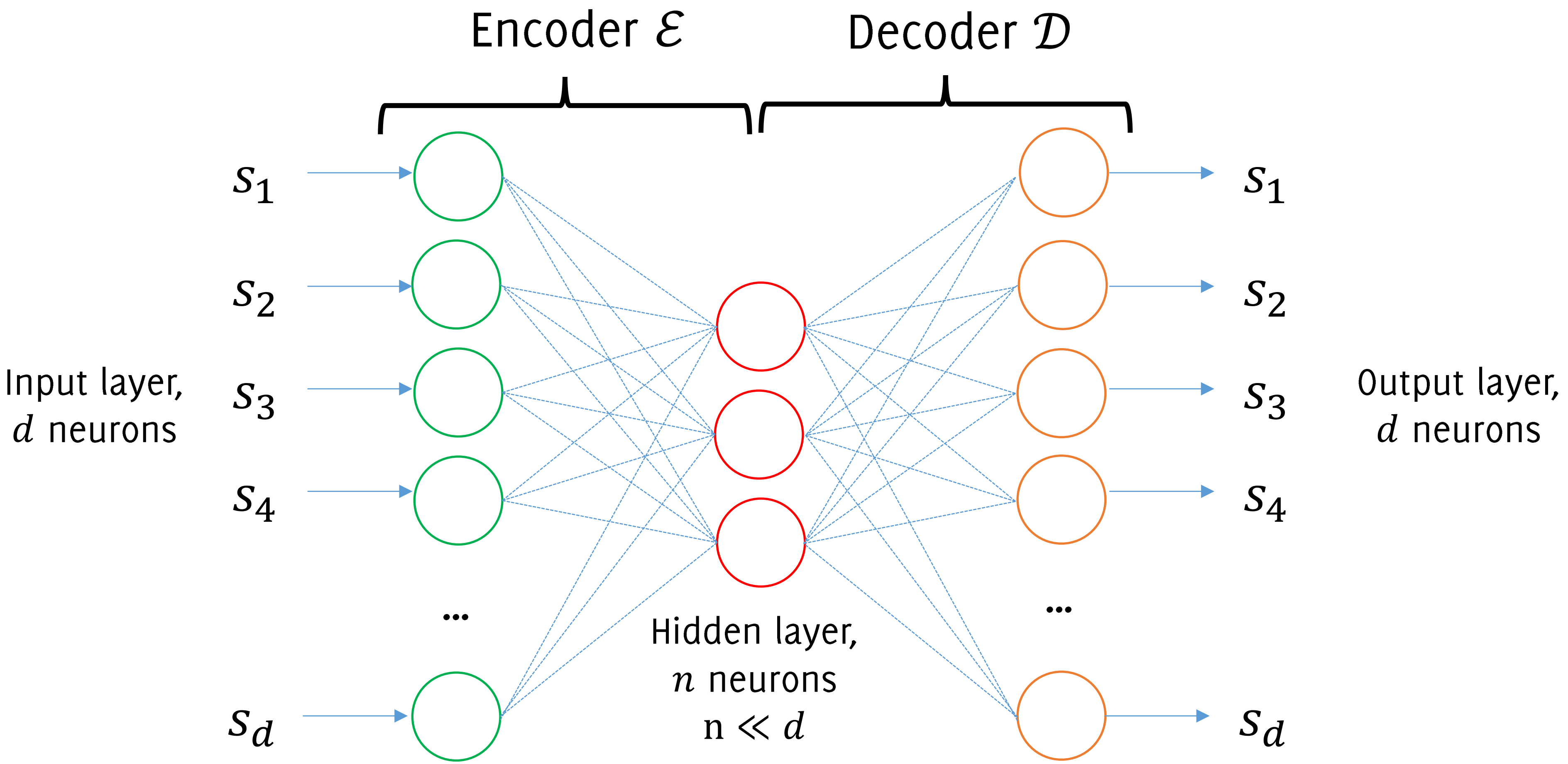


Sampling the Latent Space

What about using Autoencoders as Generative Models?

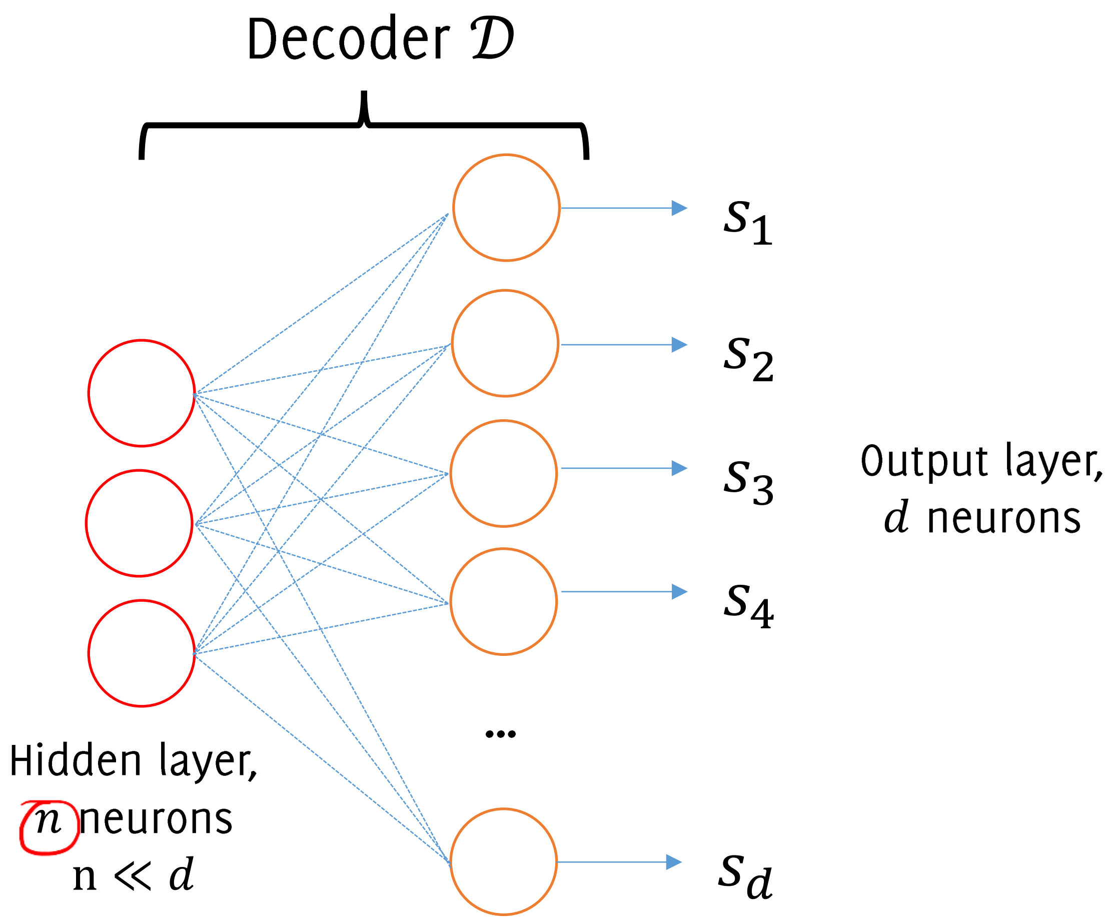
One option would be to

1) **train an autoencoder** on S



What about using Autoencoders as Generative Models?

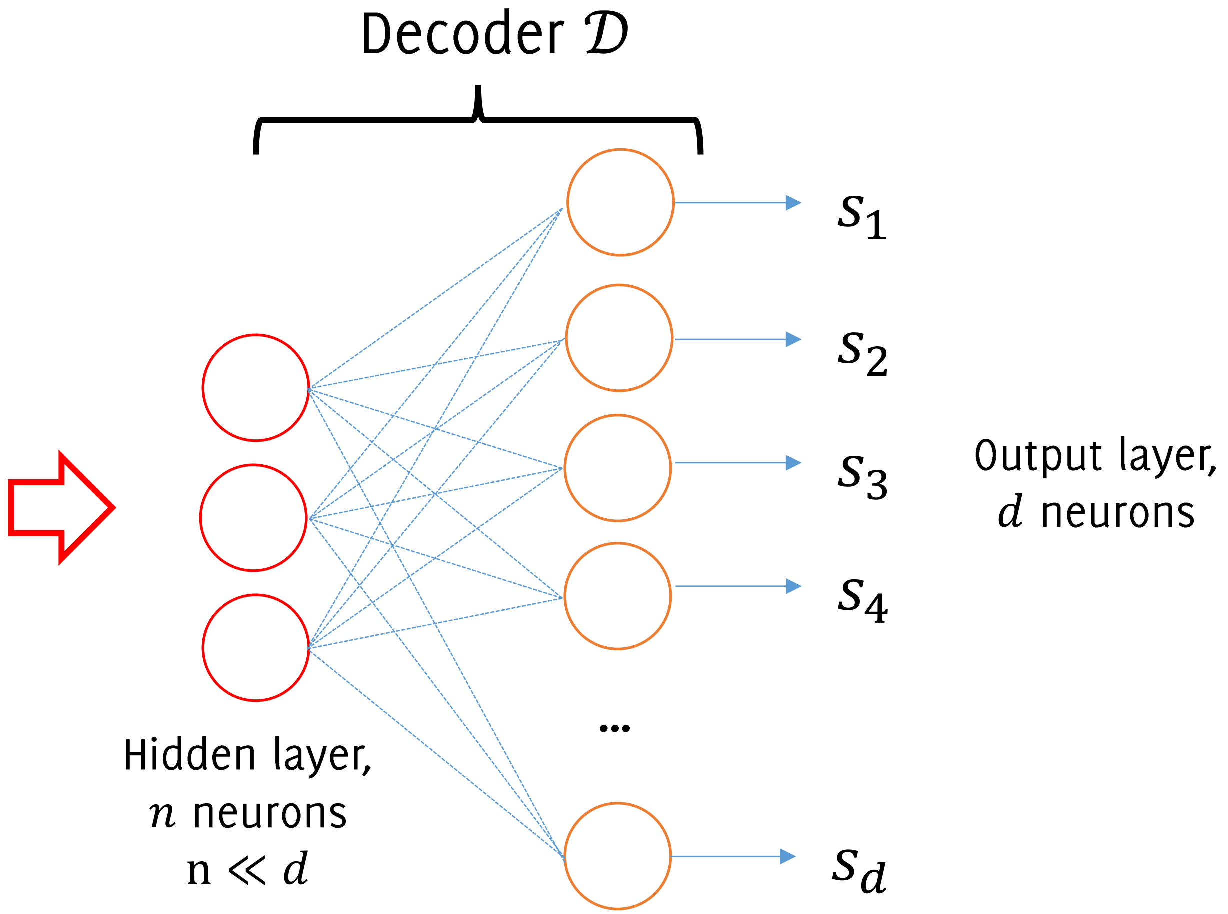
2) Discard the encoder



What about using Autoencoders as Generative Models?

3) Draw random vectors $z \sim \phi_z$, to mimic «a new latent representation» and feed this to the decoder input

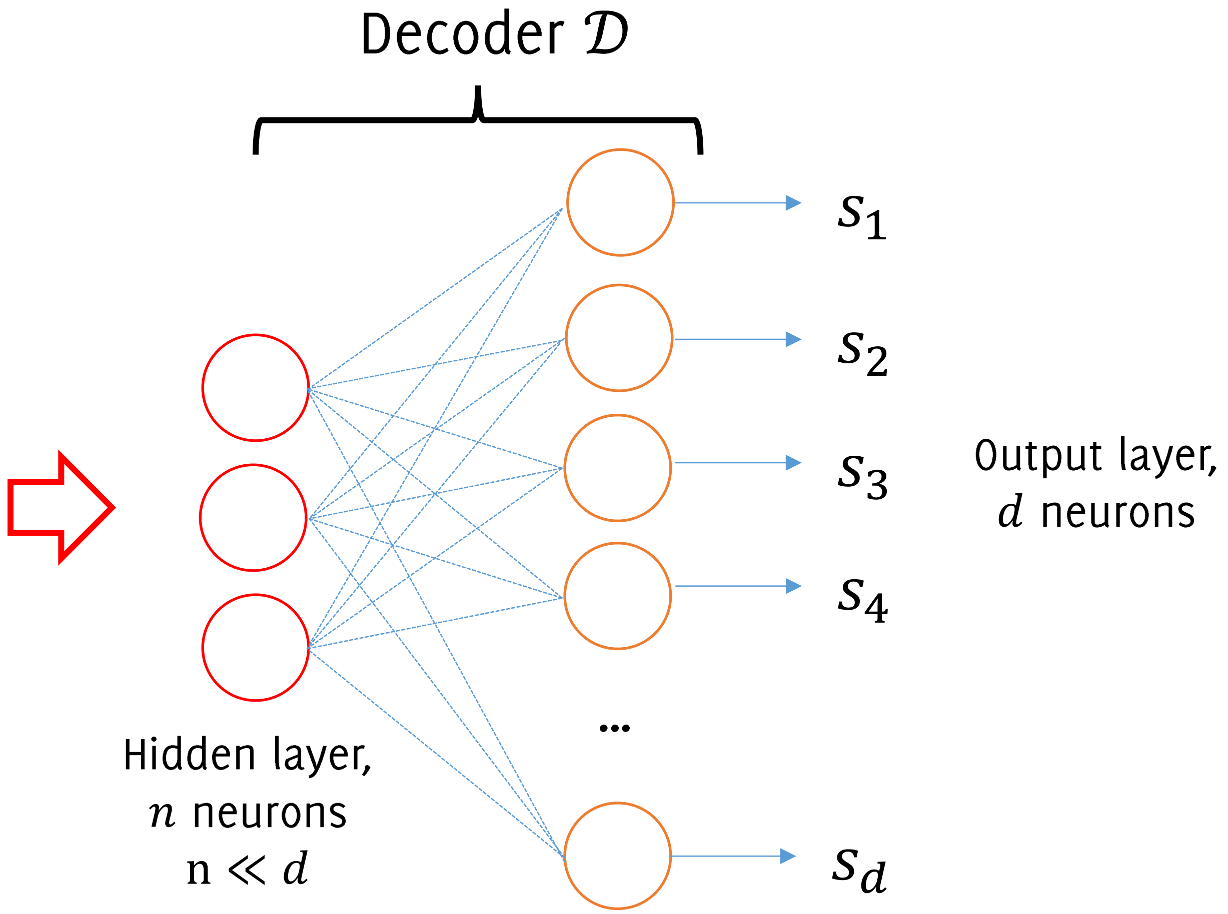
$$z \sim \phi_z , z \in \mathbb{R}^d$$



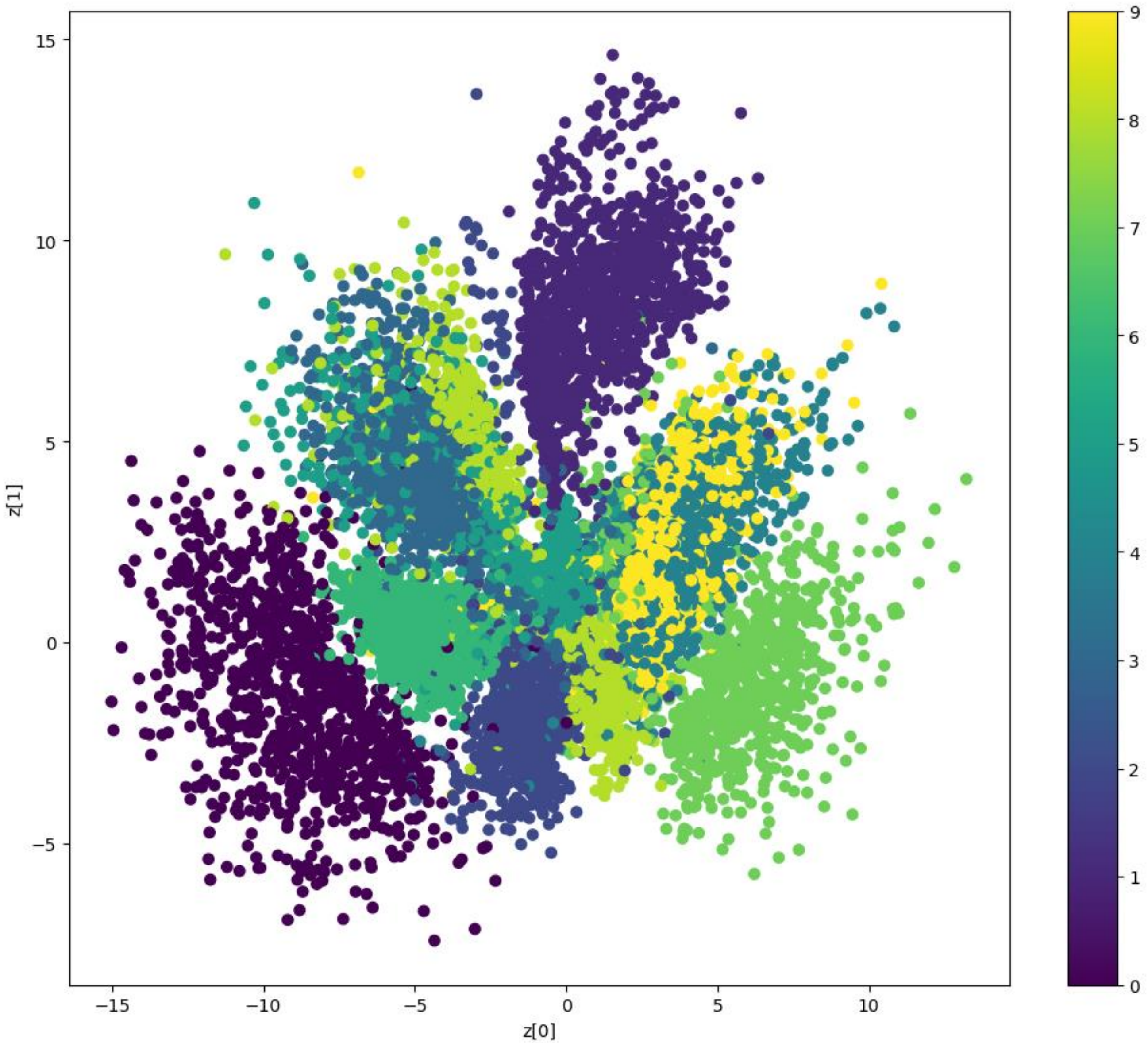
What about using Autoencoders as Generative Models?

This approach does not work since **we do not know the distribution of proper latent representation** (or it is very difficult to estimate).

$$z \sim \phi_z \, , z \in \mathbb{R}^d$$

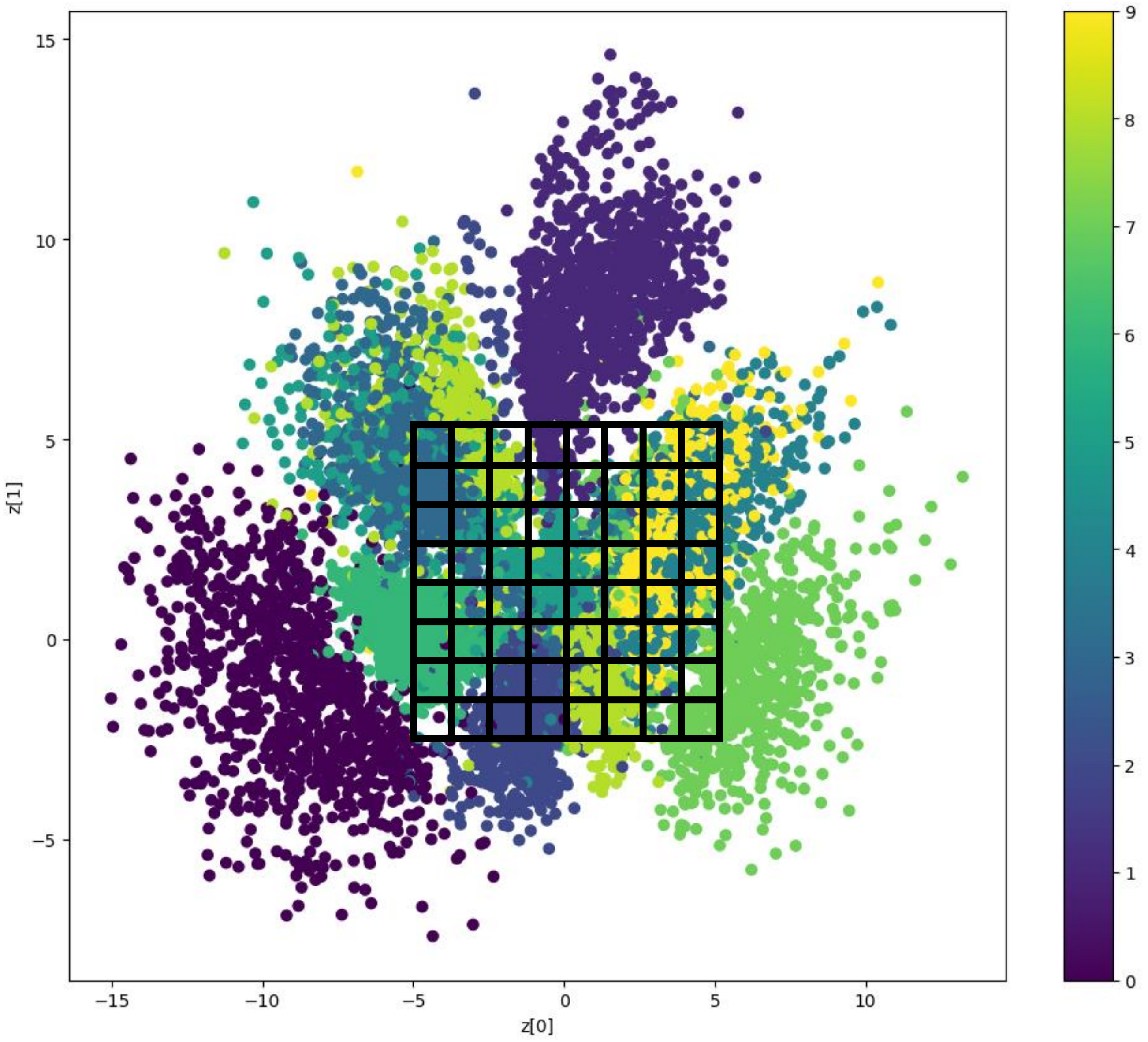


The latent representation of MNIST autoencoder $(d = 2)$



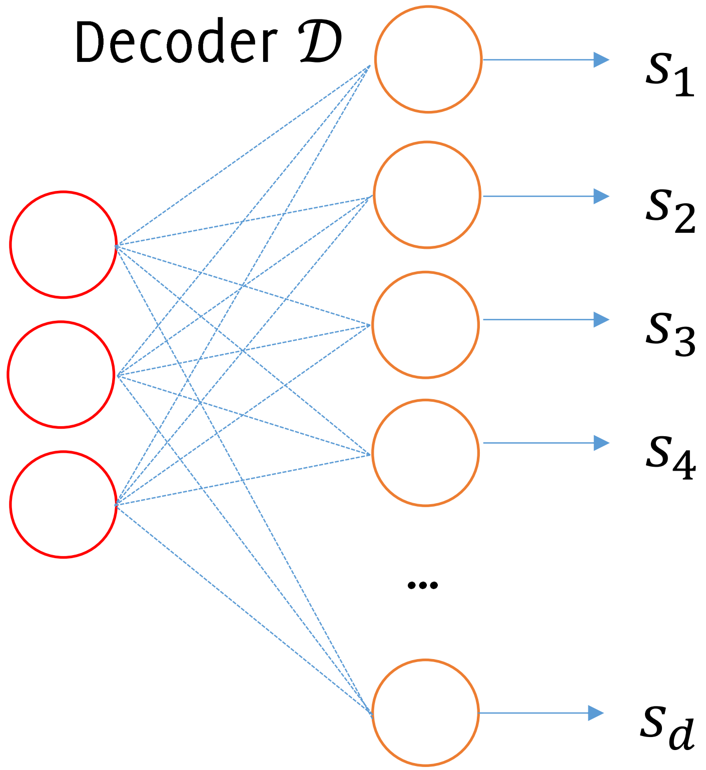
Easy to visualize, classes are somehow separated in the latent space

The latent representation of MNIST autoencoder ($d = 2$)

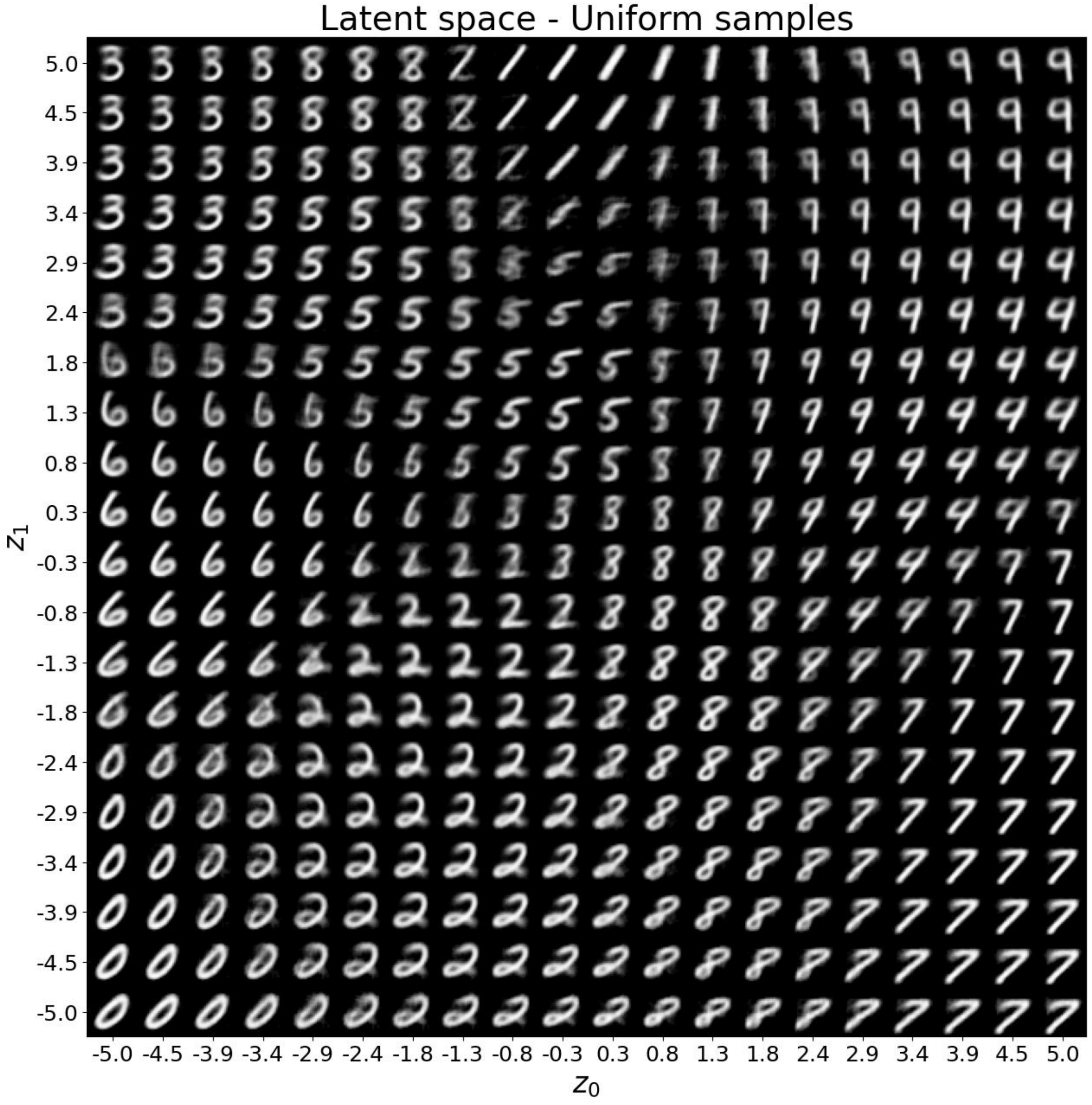


Define sampling locations as a grid in the latent space.

The grid need to cover a «populated» region of the space



The latent representation of MNIST autoencoder ($d = 2$)



As the latent space dimension grows, it is more likely to fall in a less populated area, thus to sample in regions which do not correspond to any class

Sampling the Latent Space

This is a viable image generation approach only in a **low dimensional latent space.**

When d increases, high density regions are rare, distributions ϕ_z is difficult to estimate.

Variational autoencoders forces z to follow a Gaussian distribuiton (on top of enabling accurate reconstruction). These are considered generative models.

Generative Models:
Networks able to generate realistic images

Which image is real and which one generated?

https://thispersondoesnotexist.com/ **1024 X 1024**

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

https://thispersondoesnotexist.com/

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

https://thispersondoesnotexist.com/

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

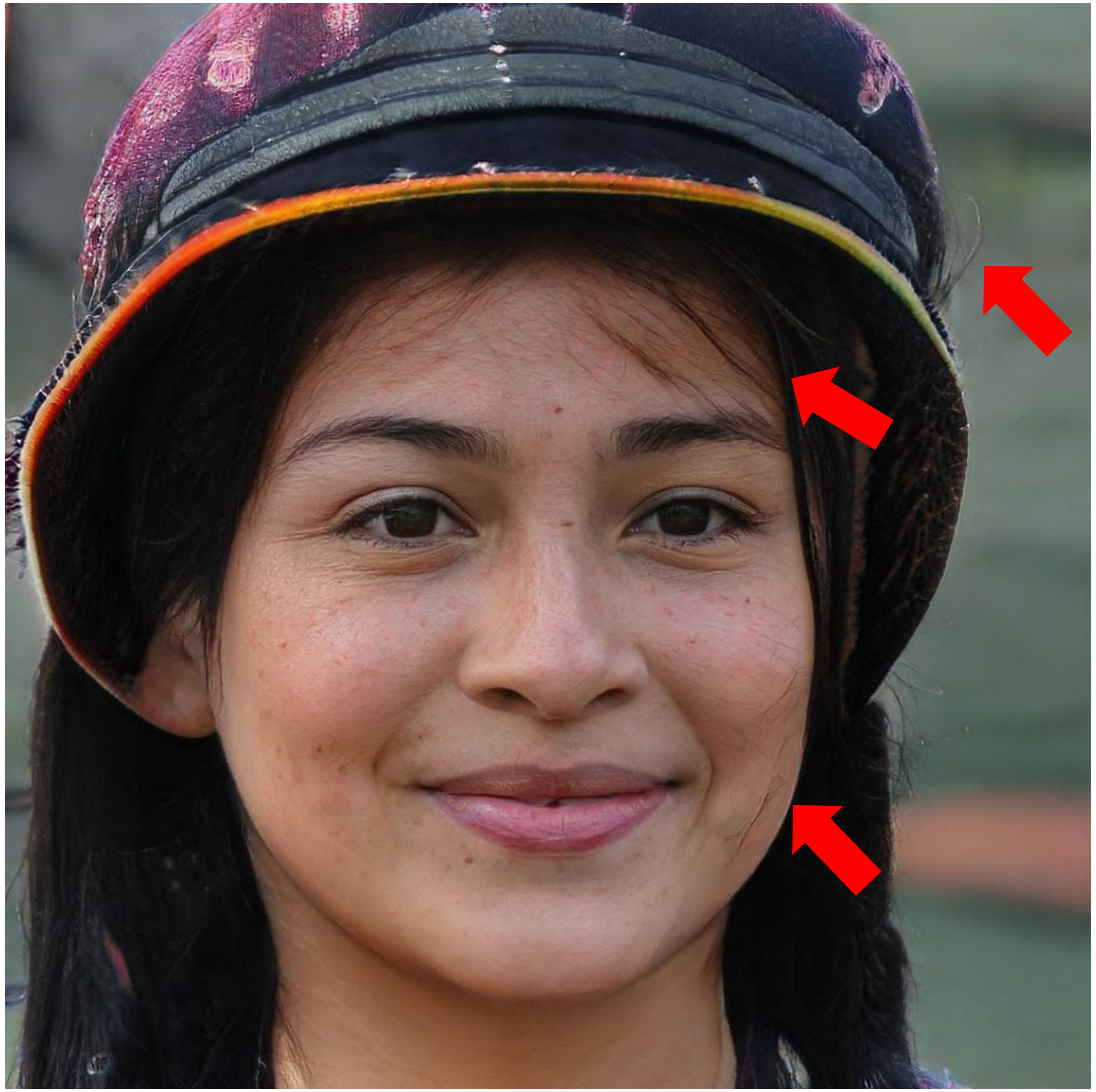
https://thispersondoesnotexist.com/

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

Sometimes there models are not perfect...

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

Sometimes there models are not perfect...



Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

even with cats (lower resolution 512 x 512) …

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

even with cats (lower resolution 512 x 512) …

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

even with horses (lower resolution 256 x 256) …

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

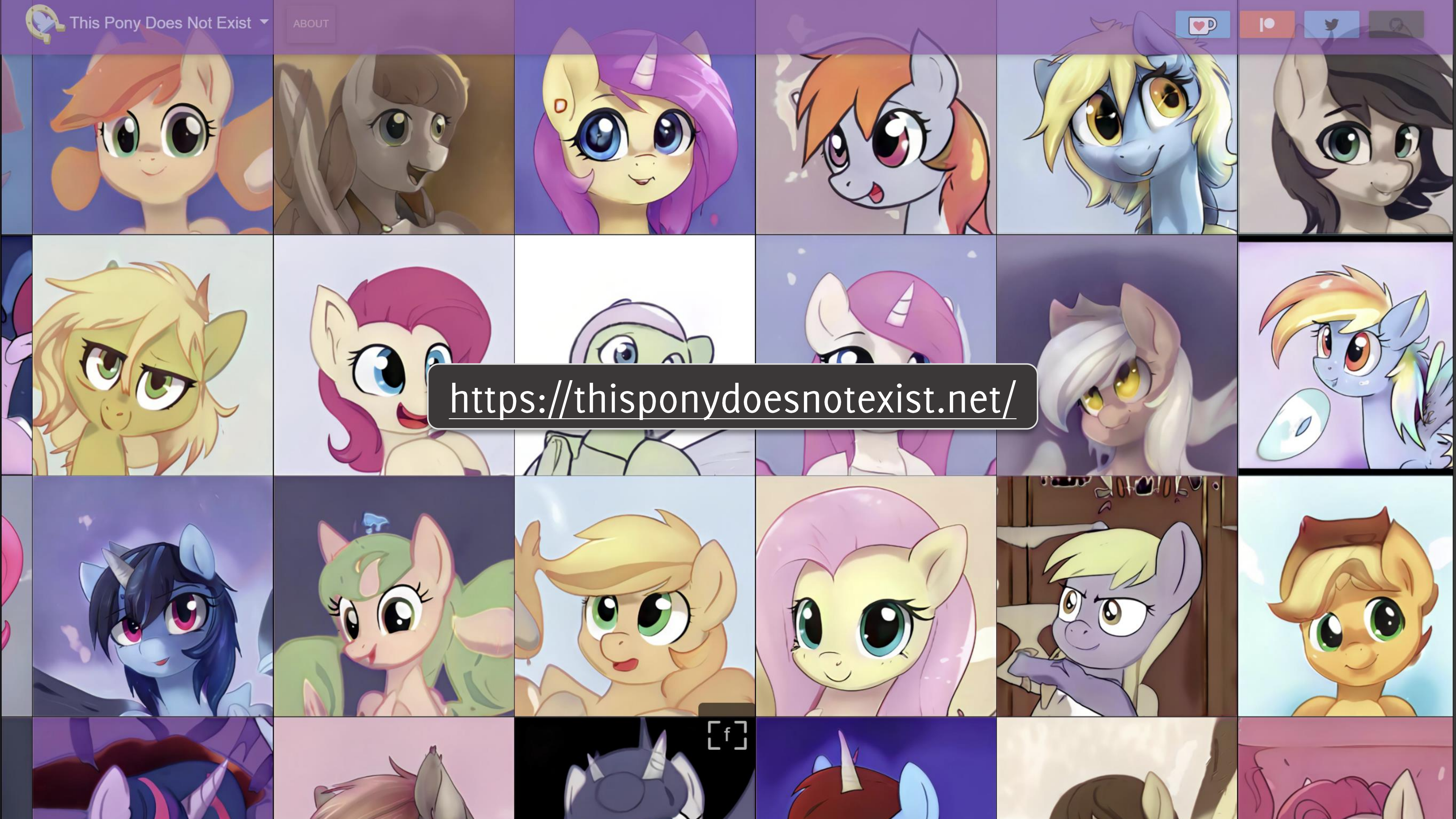
even with horses (lower resolution 256 x 256) …

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." *CVPR 2020*

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." CVPR 2020

thisnightskydoesnotexist

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." CVPR 2020


This Pony Does Not Exist
ABOUT
https://thisponydoesnotexist.net/

thissneakerdoesnotexist.com

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." CVPR 2020

Generative Models

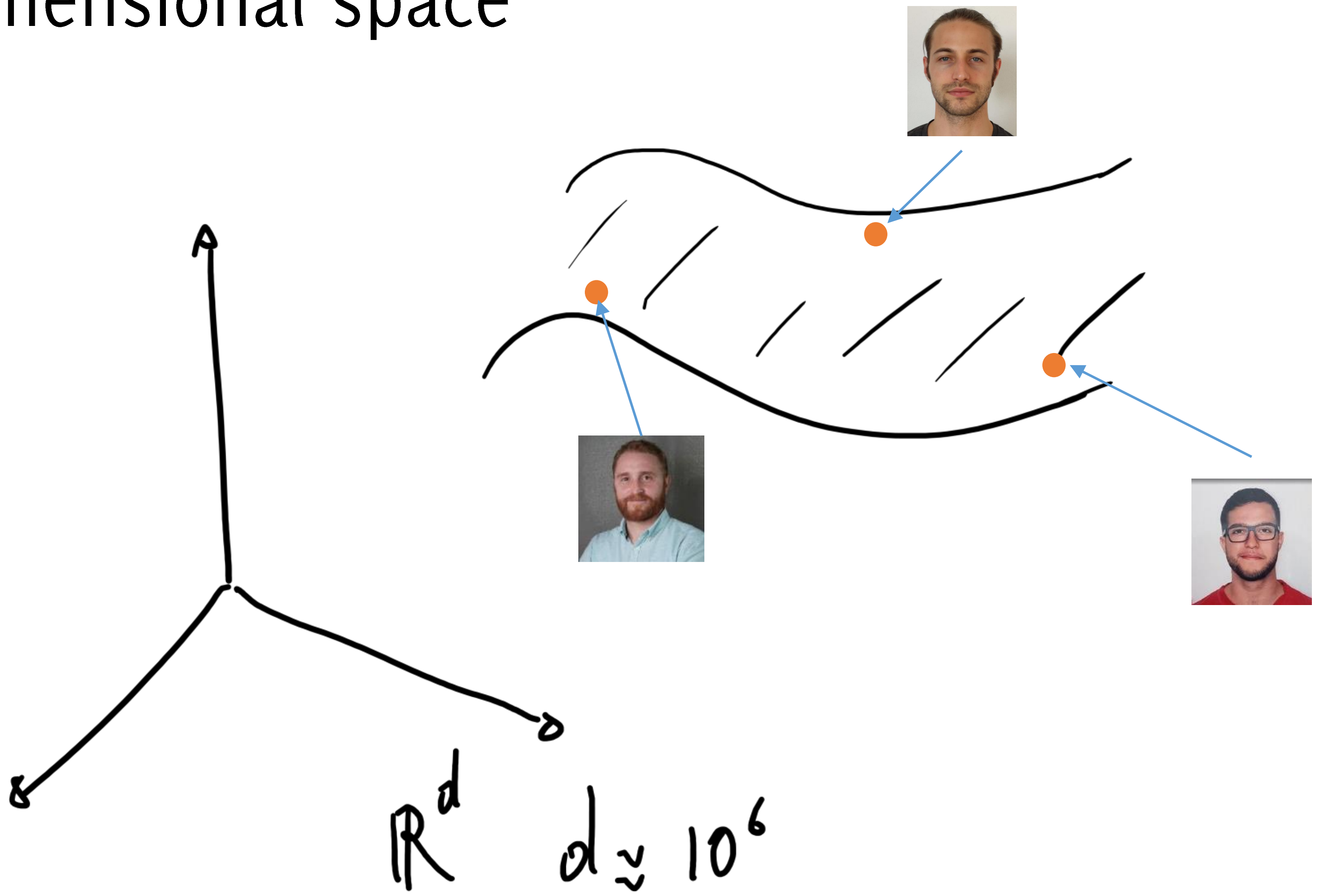
Goal:

generate, given a training set of images $TR = \{x_i\}$, generate other images that are similar to those in TR

$TR = \{$  ,  ,  ,  $\}$

The “holy grail” of image processing

Images live in a very «difficult to describe» manifold in a huge dimensional space



What for generative models?

- Generative models can be used for **data augmentation**, simulation and planning
- **Inverse problems like super-resolution, inpainting, colorization.**
- Realistic samples for artwork.
- Training generative models can also enable inference of latent representations that can be useful as **general features.**
- You are getting close to the "holy grail" of modeling the distribution of natural images
  - This can be a very useful regularization prior in other problems or to perform anomaly detection
- On top of specific application of image generation, the fist effective generative model (i.e. GANs) give rise to new training paradigm and practices (adversarial training)

Images randomly generated by a GAN

Radford, et al "Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

What for generative models?

- Generative models can be used for **data augmentation**, simulation and planning
- **Inverse problems like super-resolution, inpainting, colorization.**
- Realistic samples for artwork.
- Training generative models can also enable inference of latent representations that can be useful as **general features.**
- You are getting close to the "holy grail" of modeling the distribution of natural images
  - This can be a very useful regularization prior in other

Outdated! These were the major arguments in favour of generative models for images **before the advent of fundation models** like Dall-E, Midjourney,... Now we all know how realistic these models are, and their use in everyday life.

Images randomly generated by a GAN

Radford, et al "Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Generative Adversarial Networks

A very effective way to generate images

Generative Adversarial Networks (GAN)

The GAN approach:

- Do not look for an explicit density model ϕ_S describing the manifold of natural images.
- Just find out a model able to generate samples that «looks like» training samples $S \subset \mathbb{R}^n$.

Instead of sampling from ϕ_S, just:

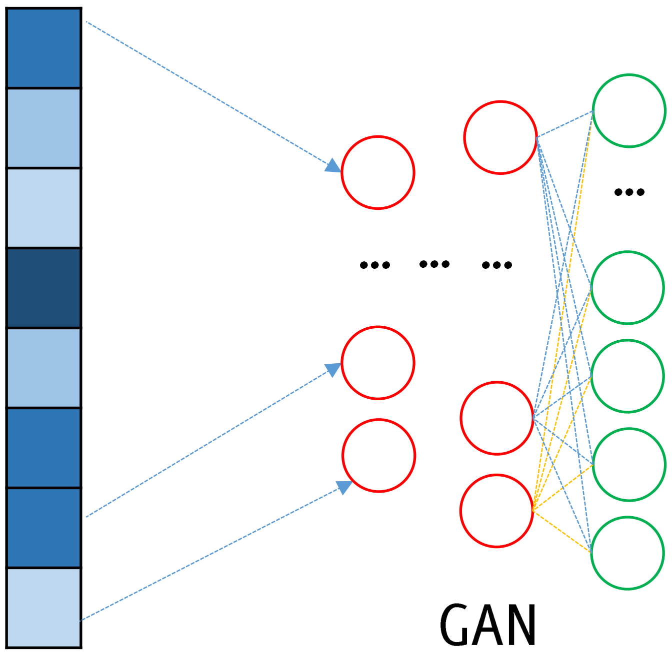
- **Sample a seed** from a known distribution ϕ_z . This is defined a priori and also referred to as **noise.**
- Feed this seed to a learned transformation that generates realistic samples, as if they were drawn from ϕ_S.

Use a neural network to learn this transformation. The neural network is going to be **trained in an unsupervised manner, no label needed**

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Generative Adversarial Networks (GAN)

The GAN approach:



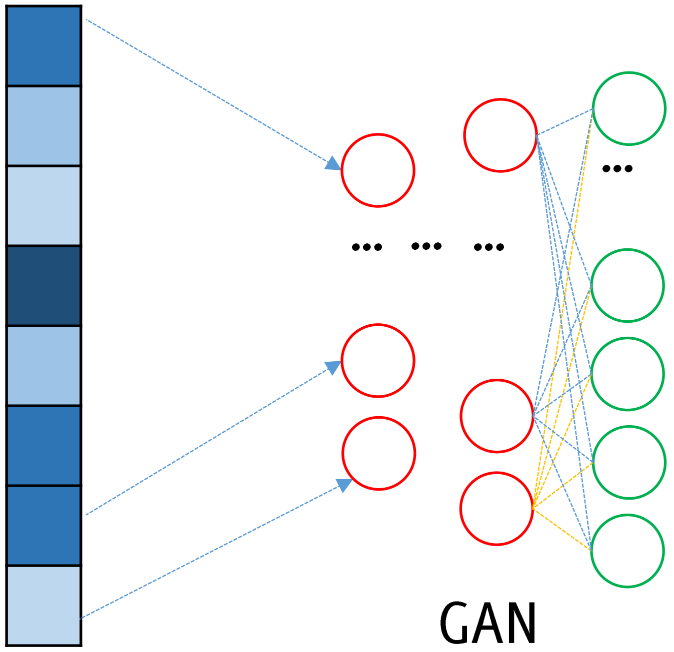
$$z \sim \phi_z$$

Draw a sample from the noise distribution

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Generative Adversarial Networks (GAN)

The GAN approach:



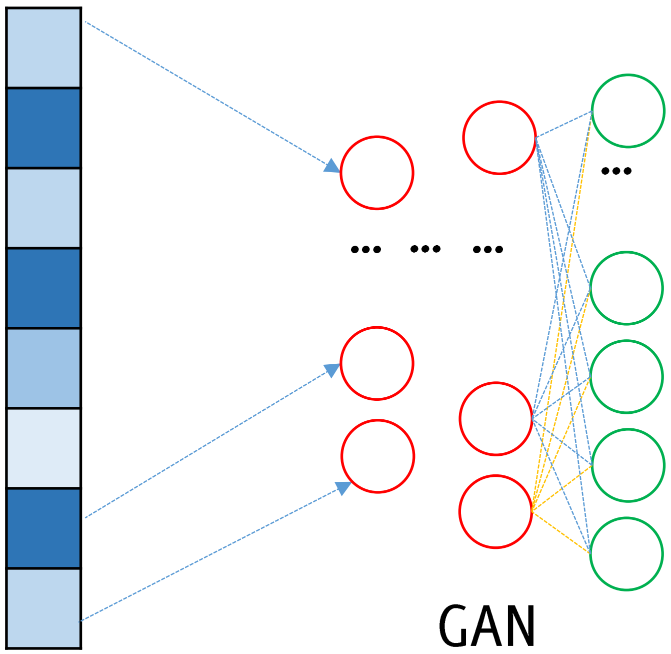
$z \sim \phi_z$

Draw a sample from the noise distribution

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Generative Adversarial Networks (GAN)

The GAN approach:



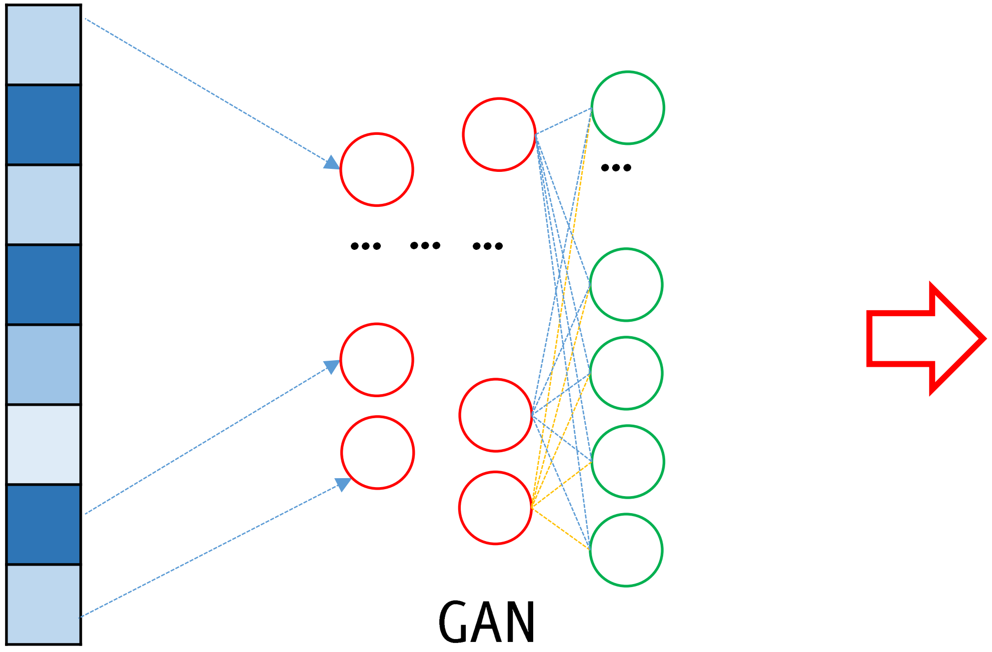
$z \sim \phi_z$

Draw a sample from the noise distribution

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Generative Adversarial Networks (GAN)

The GAN approach:

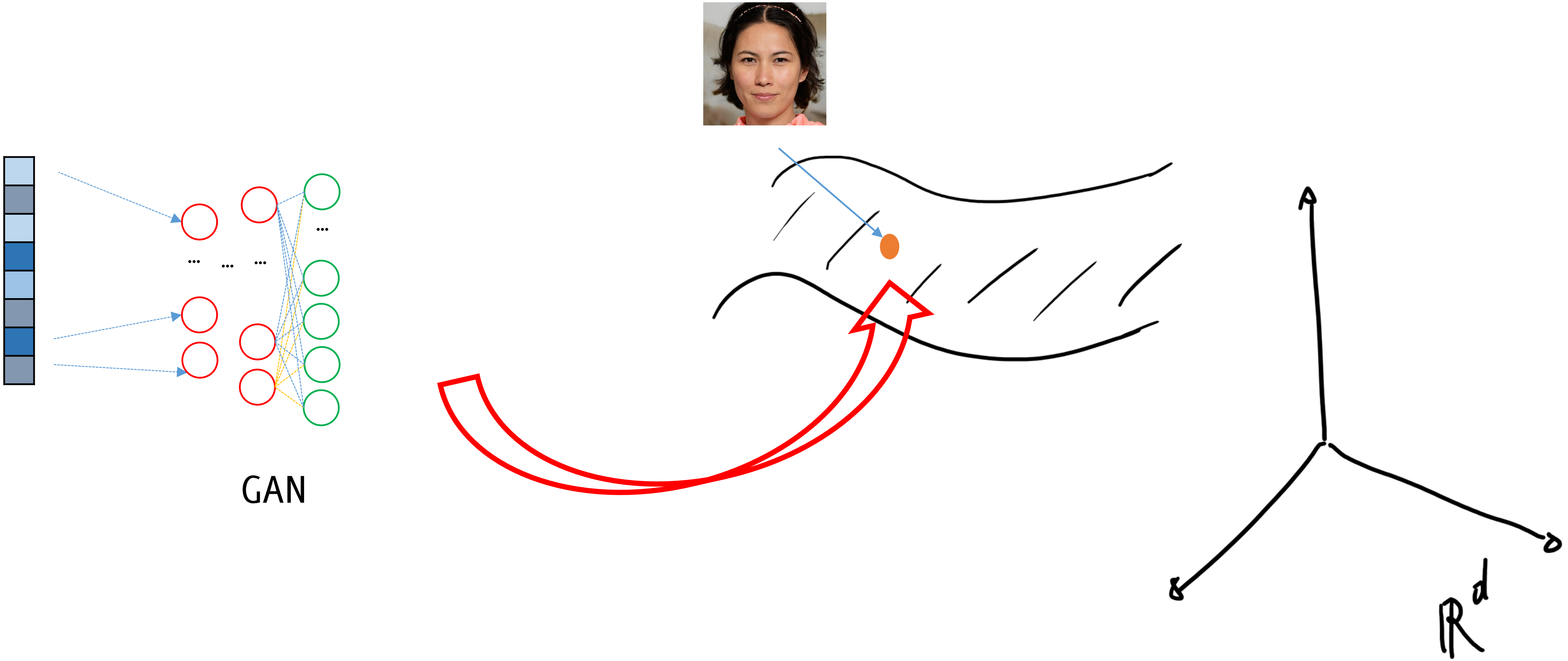


$z \sim \phi_z$

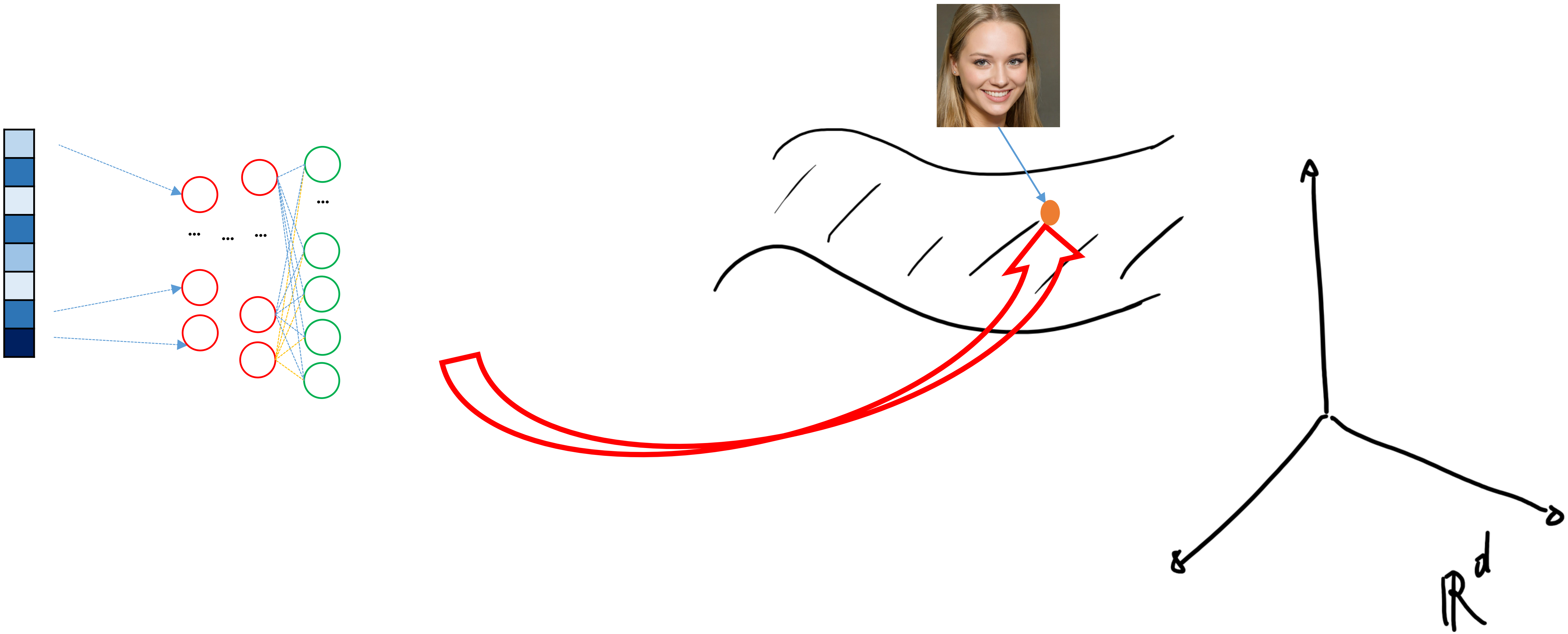
Draw a sample from the noise distribution

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

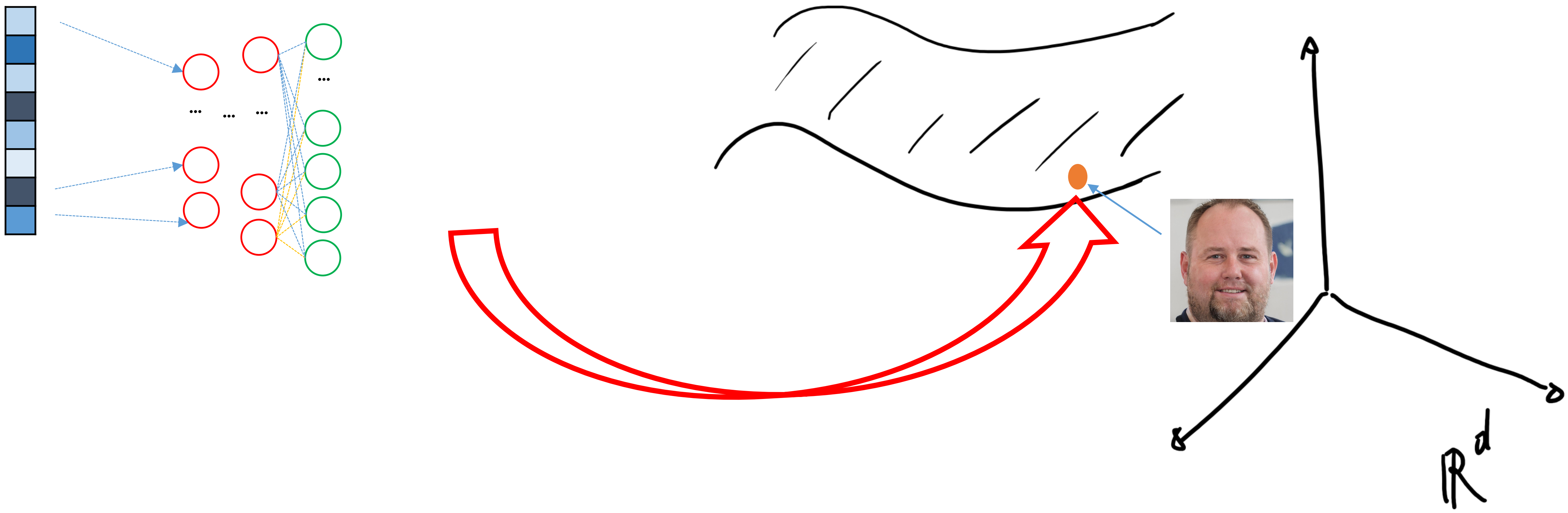
The GAN Approach



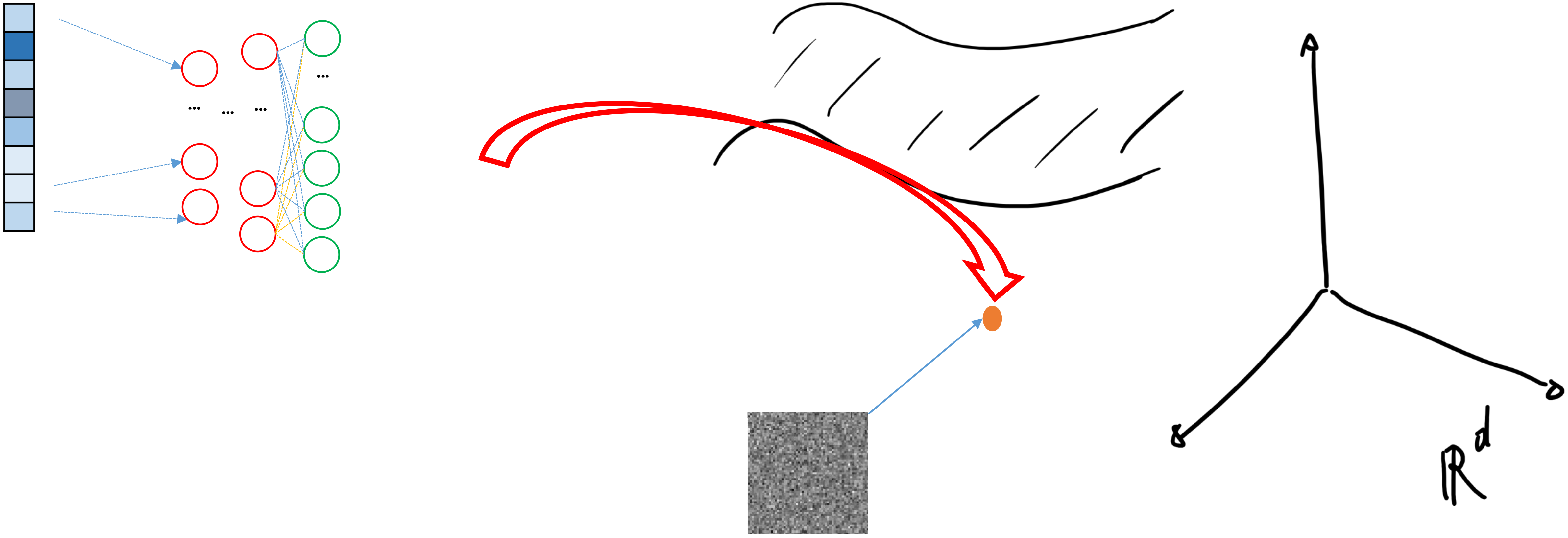
The GAN Approach



The GAN Approach

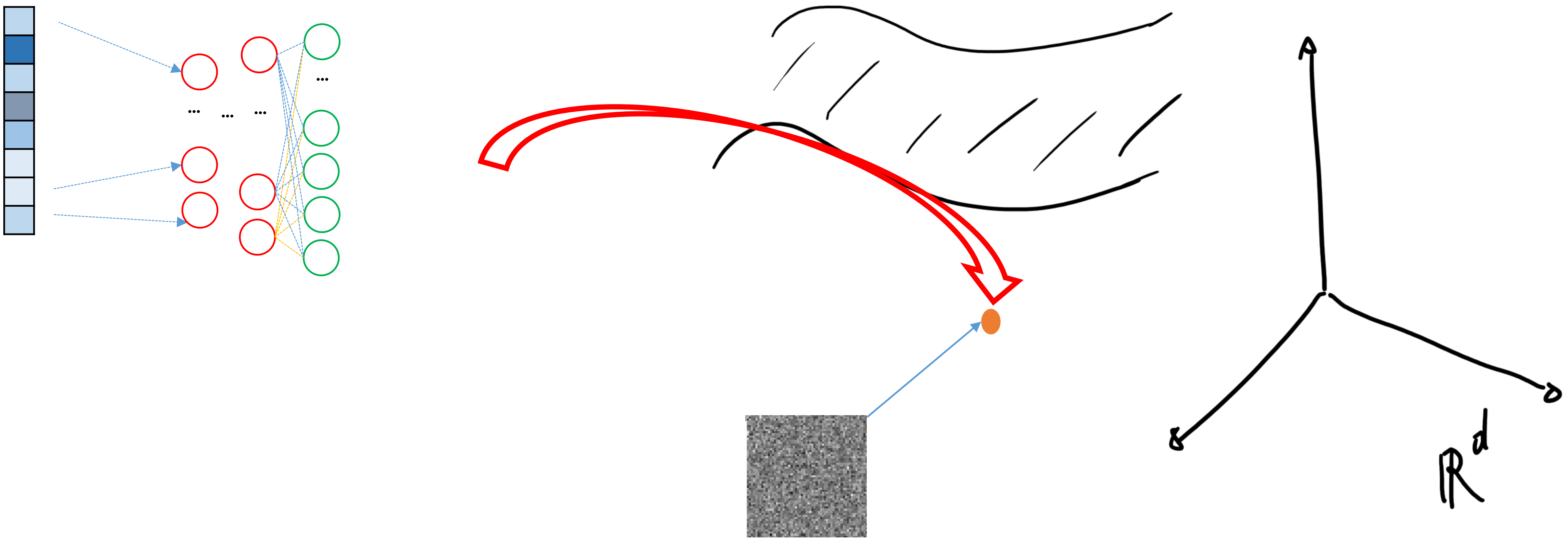


The GAN Approach



The GAN Approach

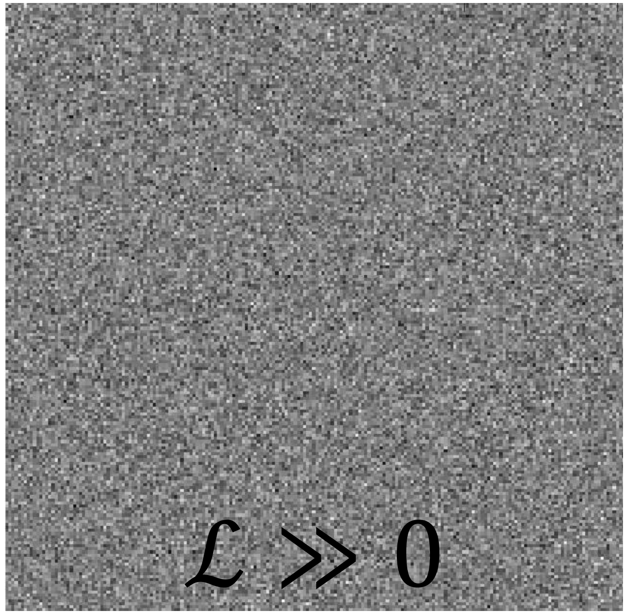
The biggest challenge is to define a suitable loss for assessing whether the output is a realistic image or not



What a loss function?

To train a neural network we need a loss function.

What would be a good loss function here?



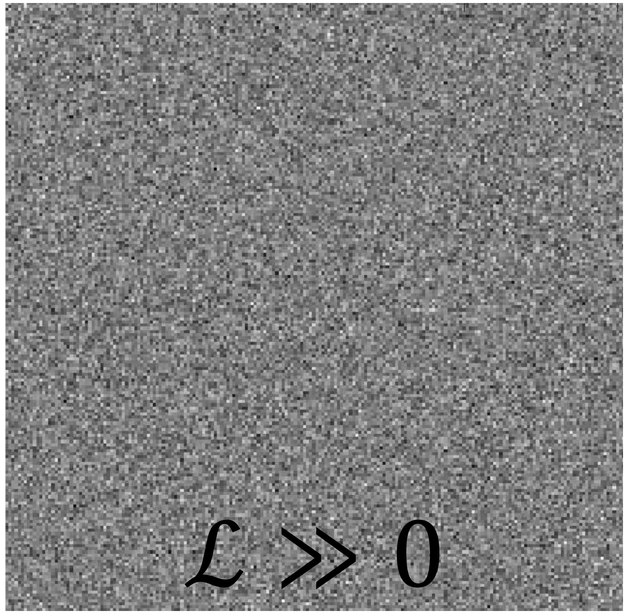
What a loss function?

To train a neural network we need a loss function.

What would be a good loss function here?

It is difficult to assess whether an image is real or not

GAN solution: **resort to a neural network to define the loss!**



Generative Adversarial Networks (GAN)

The GAN solution: Train a pair of neural networks addressing two different tasks that compete in a sort of **two player** (adversarial) **game**.

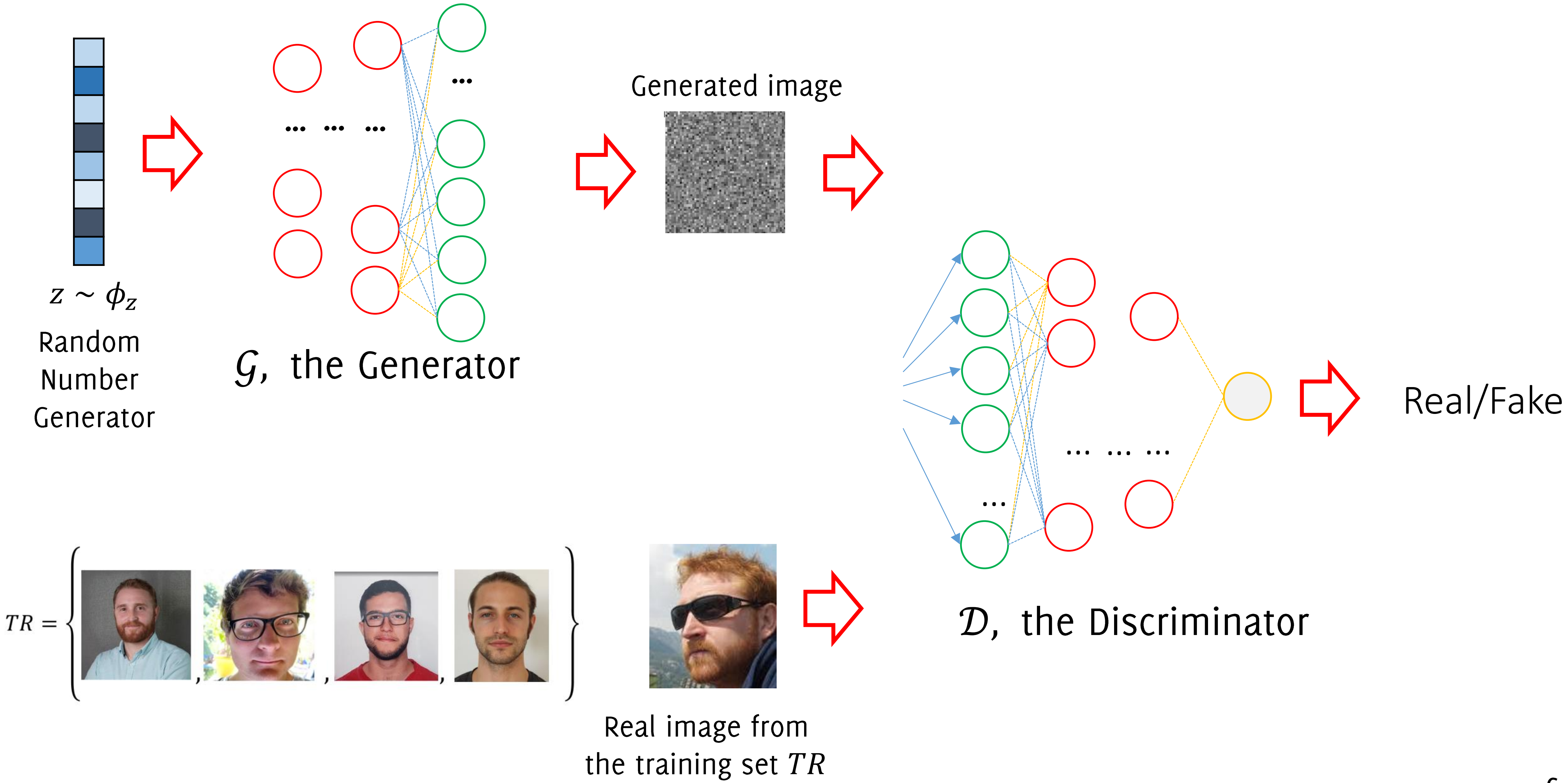
These models are:

- **Generator** $\mathcal{G}$ **that produces realistic samples** e.g. taking as input some random noise. $\mathcal{G}$ tries to fool the discriminator
- **Discriminator** $\mathcal{D}$ that takes as input an image **and assess whether it is real or generated by** $\boldsymbol{\mathcal{G}}$

Train the two and **at the end, keep only** $\mathcal{G}$

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

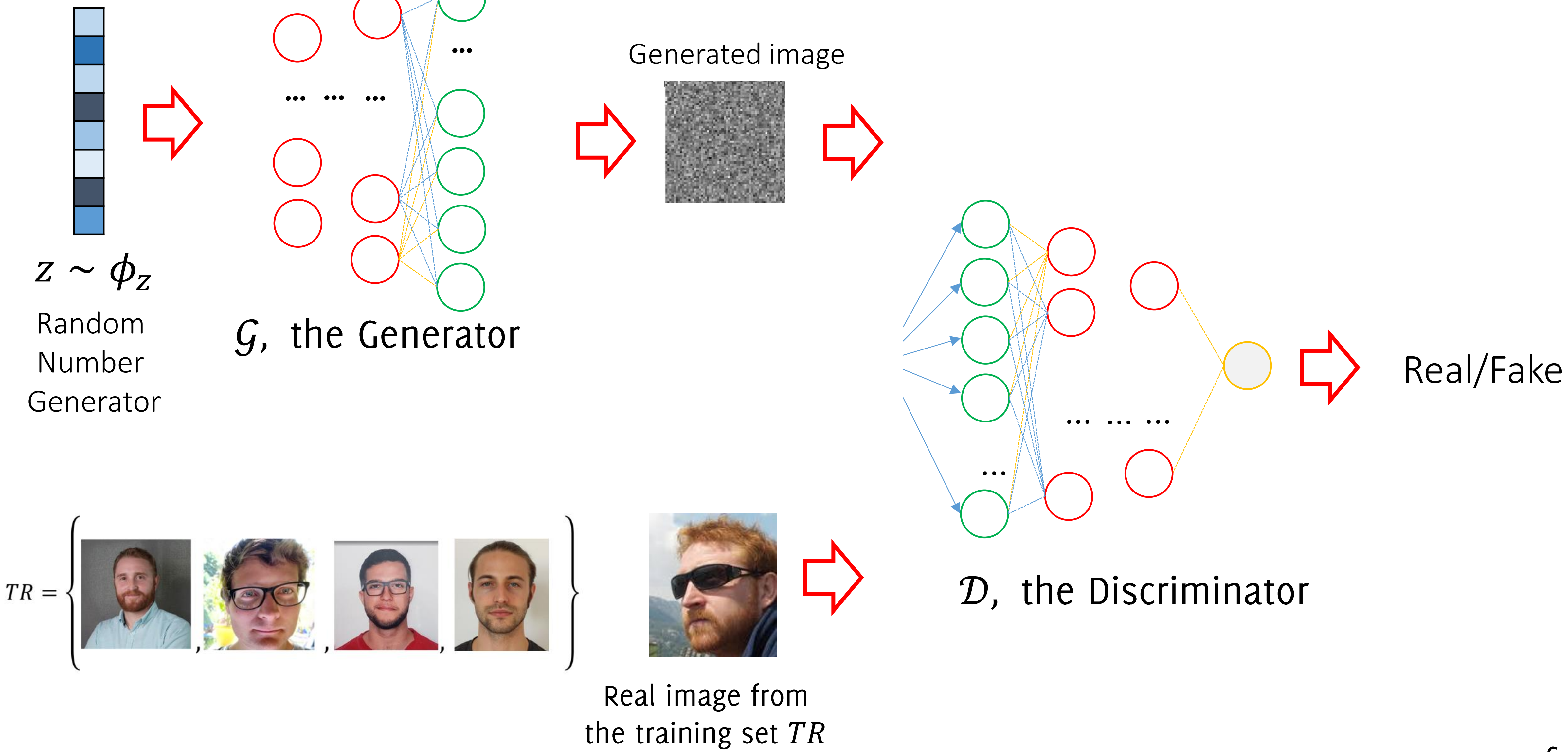
GAN Training



G. Boracchi

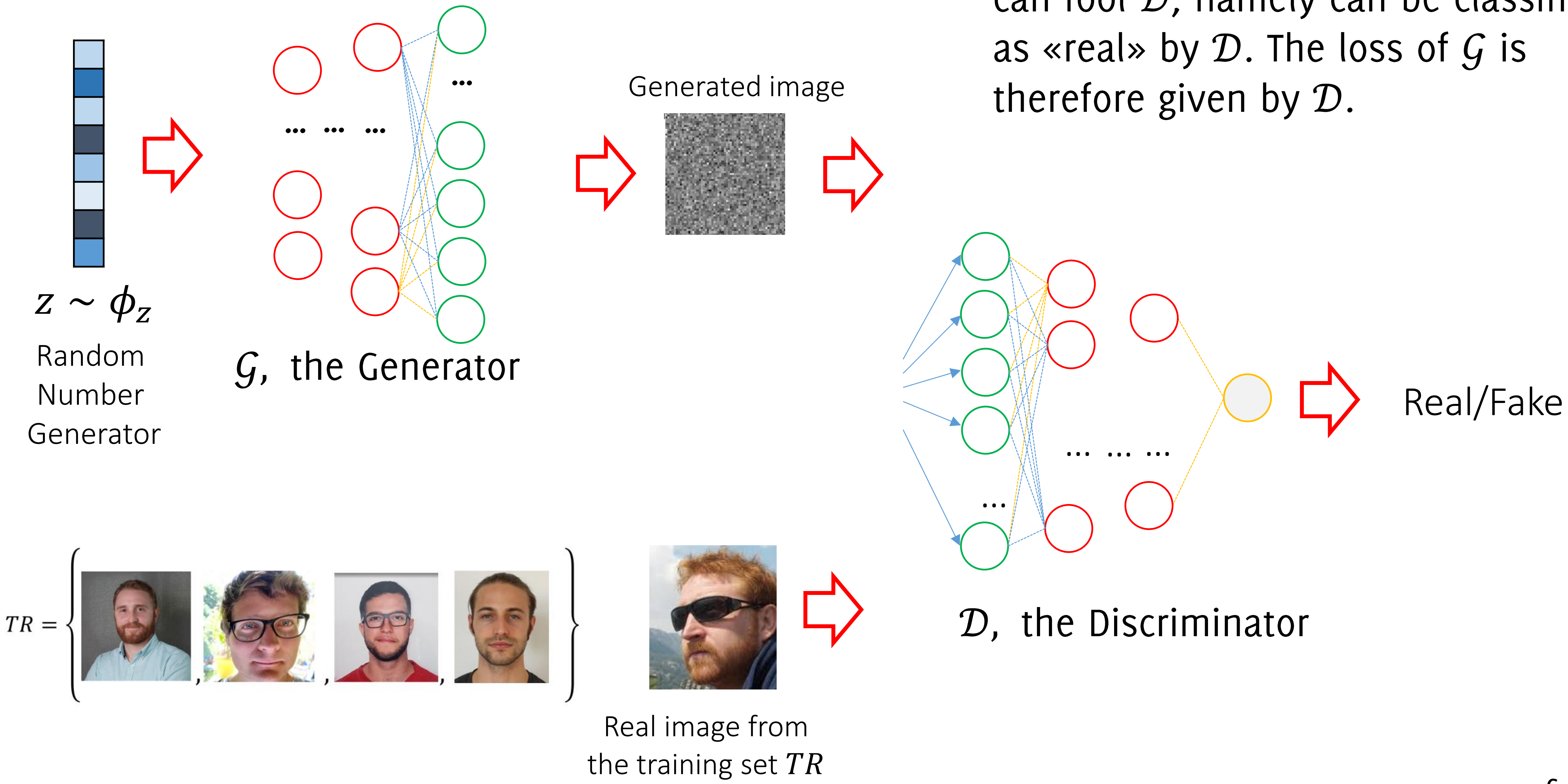
GAN Training

The goal of $\mathcal{D}$ is to recognize all the images generated by $\mathcal{G}$.



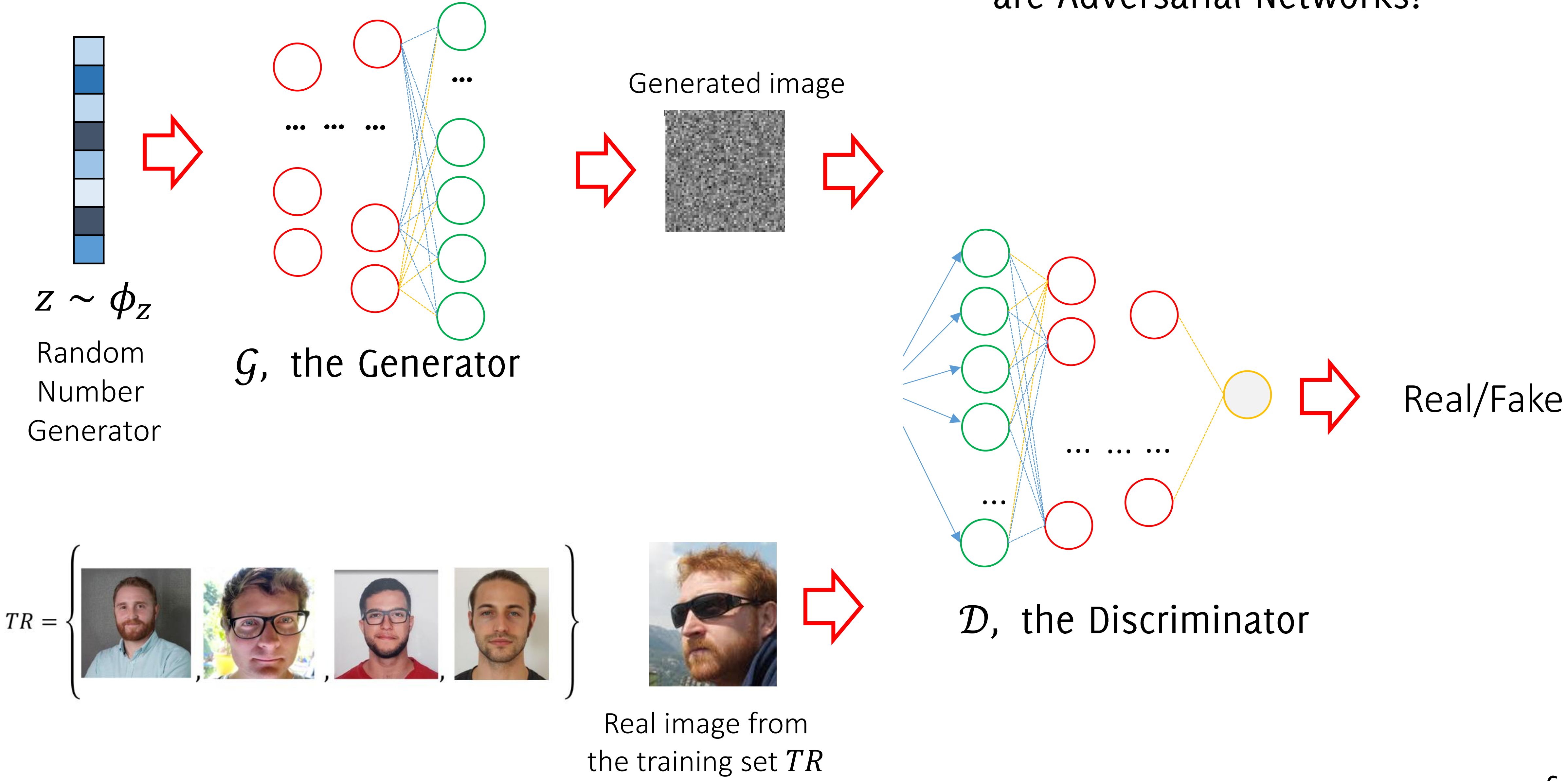
GAN Training

$\mathcal{G}$ is trained to generate images that can fool $\mathcal{D}$, namely can be classified as «real» by $\mathcal{D}$. The loss of $\mathcal{G}$ is therefore given by $\mathcal{D}$.

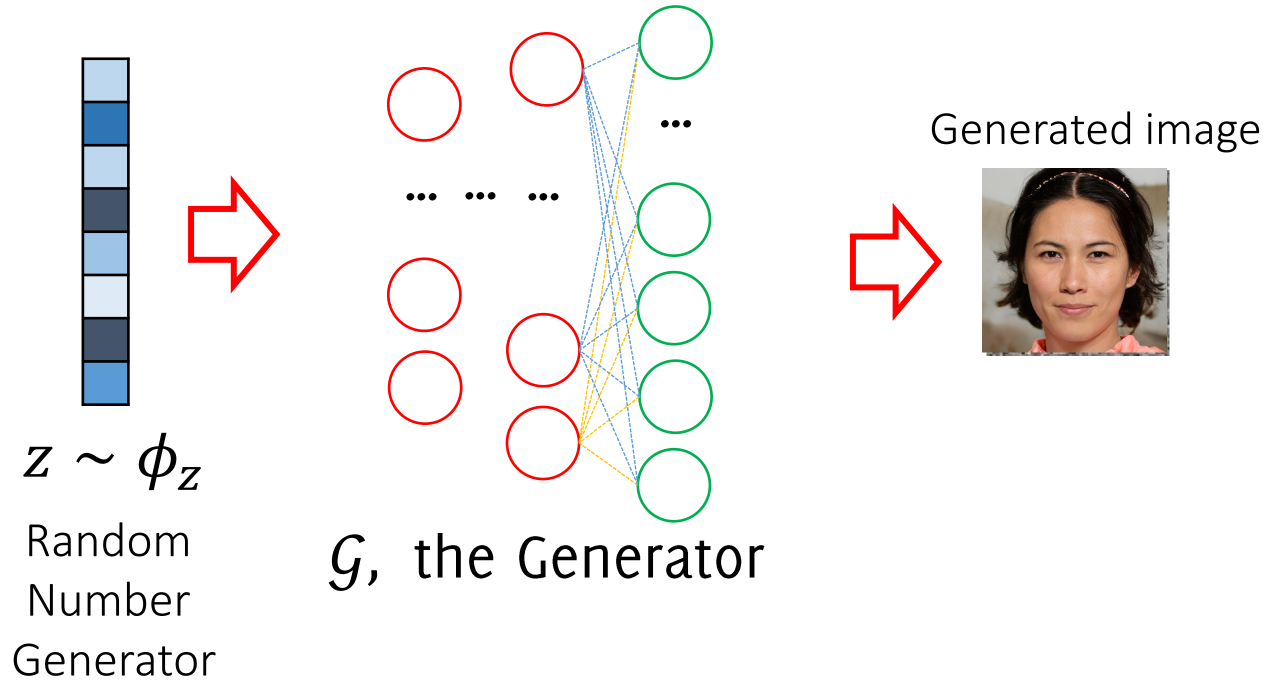


GAN Training

$\mathcal{D}$ and $\mathcal{G}$ play opposite games: they are Adversarial Networks!



GAN Inference

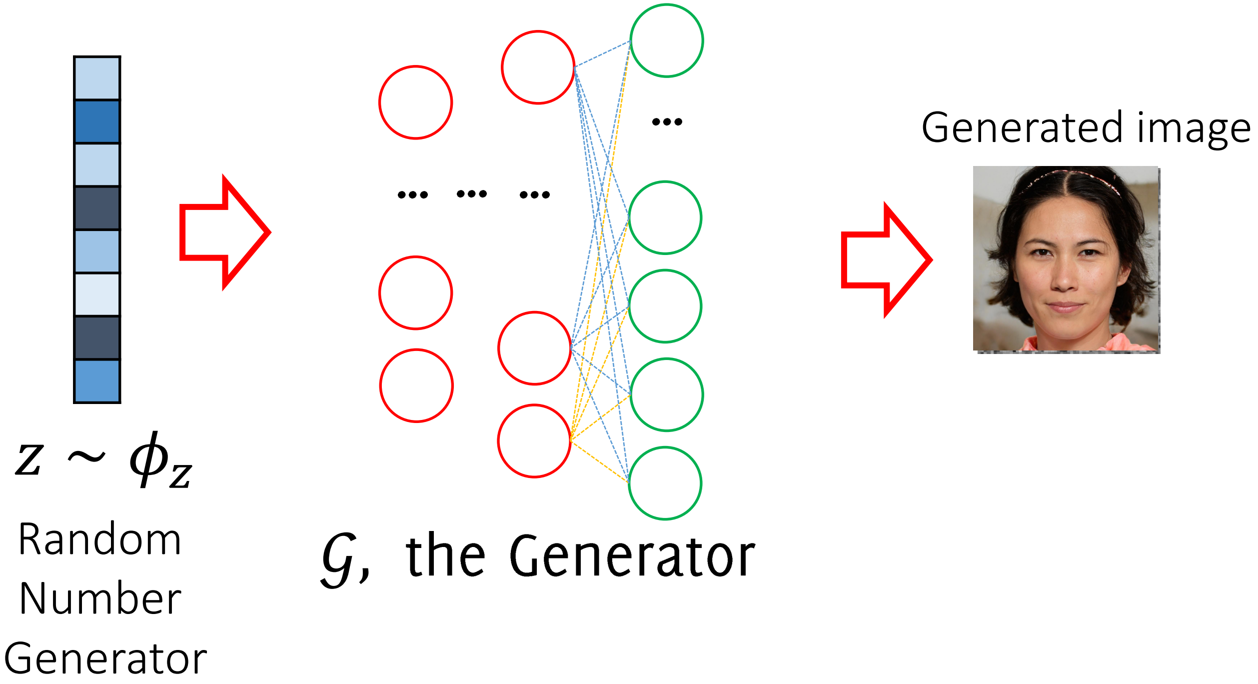


At the end of training, we hope $\mathcal{G}$ to succeed in fooling $\mathcal{D}$ consistently.

We discard $\mathcal{D}$ and keep $\mathcal{G}$ as generator

$\mathcal{D}$ is expected effective to distinguish real and fake images… if $\mathcal{G}$ can fool $\mathcal{D}$, this means that $\mathcal{G}$ is a generator.

GAN Inference



At the end of training, we hope $\mathcal{G}$ to succeed in fooling $\mathcal{D}$ consistently.

We discard $\mathcal{D}$ and keep $\mathcal{G}$ as generator

$\mathcal{D}$ is expected effective to distinguish real and fake images… if $\mathcal{G}$ can fool $\mathcal{D}$, this means that $\mathcal{G}$ is a generator.

- **Discriminator $\mathcal{D}$ is completely useless and as such dropped.**
- **After a successful GAN training, $\mathcal{D}$ is not able to distinguish fake images.**
- **The generative network $\mathcal{G}$ has never seen a single image from S**

GAN: Setting up the stage

Both $\mathcal{D}$ and $\mathcal{G}$ are conveniently chosen as MLP or CNN

This notation is meant to visualize what are the NN parameters (θ_d or θ_g). Networks take a single input $\boldsymbol{s}$ or $\boldsymbol{z}$

Our networks take as input:

- $\mathcal{D} = \mathcal{D}(\boldsymbol{s}, \theta_d)$,
- $\mathcal{G} = \mathcal{G}(\boldsymbol{z}, \theta_g)$,

θ_g and θ_d are network parameters, $\boldsymbol{s} \in \mathbb{R}^n$ is an input image (either real or generated by $\mathcal{G}$) and $\boldsymbol{z} \in \mathbb{R}^d$ is some random noise to be fed to the generator.

Our networks give as output:

- $\mathcal{D}(\cdot, \theta_d): \mathbb{R}^n \rightarrow [0,1]$ gives as output the posterior for the input be a true image
- $\mathcal{G}(\cdot, \theta_d): \mathbb{R}^d \rightarrow \mathbb{R}^n$ gives as output the generated image

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

GAN Training

A good discriminator is such:

- $\mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s} \in S$ (true image from the training set)
- $1 - \mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s}$ was generated from $\mathcal{G}$
- $1 - \mathcal{D}\big(\mathcal{G}(\boldsymbol{z}, \theta_g), \theta_d\big)$ is maximum when $\boldsymbol{z} \sim \phi_Z$

Training $\mathcal{D}$ consists in maximizing the **binary cross-entroy**

$$\max_{\theta_d}\Big(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\big[\log(1 - \mathcal{D}\big(\mathcal{G}(\boldsymbol{z}, \theta_g), \theta_d\big))\big]\Big)$$

Written using mathematical expectation rather than sum on minibatches

GAN Training

A good discriminator is such:

- $\mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s} \in S$ (true image from the training set)
- $1 - \mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s}$ was generated from $\mathcal{G}$
- $1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right)$ is maximum when $\boldsymbol{z} \sim \phi_Z$

Training $\mathcal{D}$ consists in maximizing the binary cross-entroy

$$\max_{\theta_d}\left(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\left[\log(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right))\right]\right)$$

This has to be 1 since $s \sim \phi_S$, thus images are real

This has to be 0 since $\mathcal{G}\left(\boldsymbol{z}, \theta_g\right)$ is a generated (fake) image

GAN Training

A good discriminator is such:

- $\mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s} \in S$ (true image from the training set)
- $1 - \mathcal{D}(\boldsymbol{s}, \theta_d)$ is maximum when $\boldsymbol{s}$ was generated from $\mathcal{G}$
- $1 - \mathcal{D}\big(\mathcal{G}(\boldsymbol{z}, \theta_g), \theta_d\big)$ is maximum when $\boldsymbol{z} \sim \phi_Z$

Training $\mathcal{D}$ consists in maximizing the binary cross-entroy

$$\max_{\theta_d}\Big(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\big[\log(1 - \mathcal{D}\big(\mathcal{G}(\boldsymbol{z}, \theta_g), \theta_d))\big]\Big)$$

A good generator $\mathcal{G}$ makes $\mathcal{D}$ to fail, thus minimizes the above

$$\min_{\theta_g}\max_{\theta_d}\Big(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\big[\log(1 - \mathcal{D}\big(\mathcal{G}(\boldsymbol{z}, \theta_g), \theta_d))\big]\Big)$$

GAN Training

Solve by an iterative numerical approach

$$\min_{\theta_g} \max_{\theta_d} \left(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\left[\log(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right))\right] \right)$$

GAN Training

Solve by an iterative numerical approach

$$\min_{\theta_g} \max_{\theta_d} \left(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\left[\log(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right))\right]\right)$$

Alternate:

- k-steps of Stochastic Gradient Ascent w.r.t. θ_d, keep θ_g fixed and solve

$$\max_{\theta_d} \left(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\left[\log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right)\right)\right]\right)$$

- 1-step of Stochastic Grandient Descent w.r.t. θ_g being θ_d fixed

$$\min_{\theta_g} \left(\mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(\boldsymbol{s}, \theta_d)] + \mathrm{E}_{z\sim\phi_Z}\left[\log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right)\right)\right]\right)$$

and since the first term does not depend on θ_g, this consists in minimizing

$$\min_{\theta_g} \left(\mathrm{E}_{z\sim\phi_Z}\left[\log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}, \theta_g\right), \theta_d\right)\right)\right]\right)$$

GAN Training

for $i = 1 \dots$ #number of epochs

for k −times # gradient ascent steps for θ_d

- Draw a minibatch $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}$ of noise realization
- Sample a minibatch of images $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_m\}$
- Update θ_d by stochastic gradient ascend:

$$\nabla_{\theta_d}\left[\sum_i \log \mathcal{D}(\boldsymbol{s_i}, \theta_d) + \log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}_i, \theta_g\right), \theta_d\right)\right)\right]$$

Draw a minibatch $\{\boldsymbol{z}_1, \dots, \boldsymbol{z}_m\}$ of noise realizations # gradient descent steps for θ_g

Update $\mathcal{G}$ by stochastic gradient descent:

$$\nabla_{\theta_g}\left[\sum_i \log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}_i, \theta_g\right), \theta_d\right)\right)\right]$$

Algorithm outline

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

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

end for

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

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

end for

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

Goodfellow, I. et al “Generative adversarial nets” NIPS 2014

GAN Training

This was presented as a best practice, later GANs such as Wasserstein GANs do not use.

for $i = 1 \ldots$ #number of epochs

for k –times # gradient ascent steps for θ_d

- Draw a minibatch $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ of noise realization
- Sample a minibatch of images $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\}$
- Update θ_d by stochastic gradient ascend:

$$\nabla_{\theta_d} \left[\sum_i \log \mathcal{D}(\boldsymbol{s}_i, \theta_d) + \log\left(1 - \mathcal{D}\left(\mathcal{G}(\boldsymbol{z}_i, \theta_g), \theta_d\right)\right) \right]$$

Draw a minibatch $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ of noise realizations # gradient descent steps for θ_g

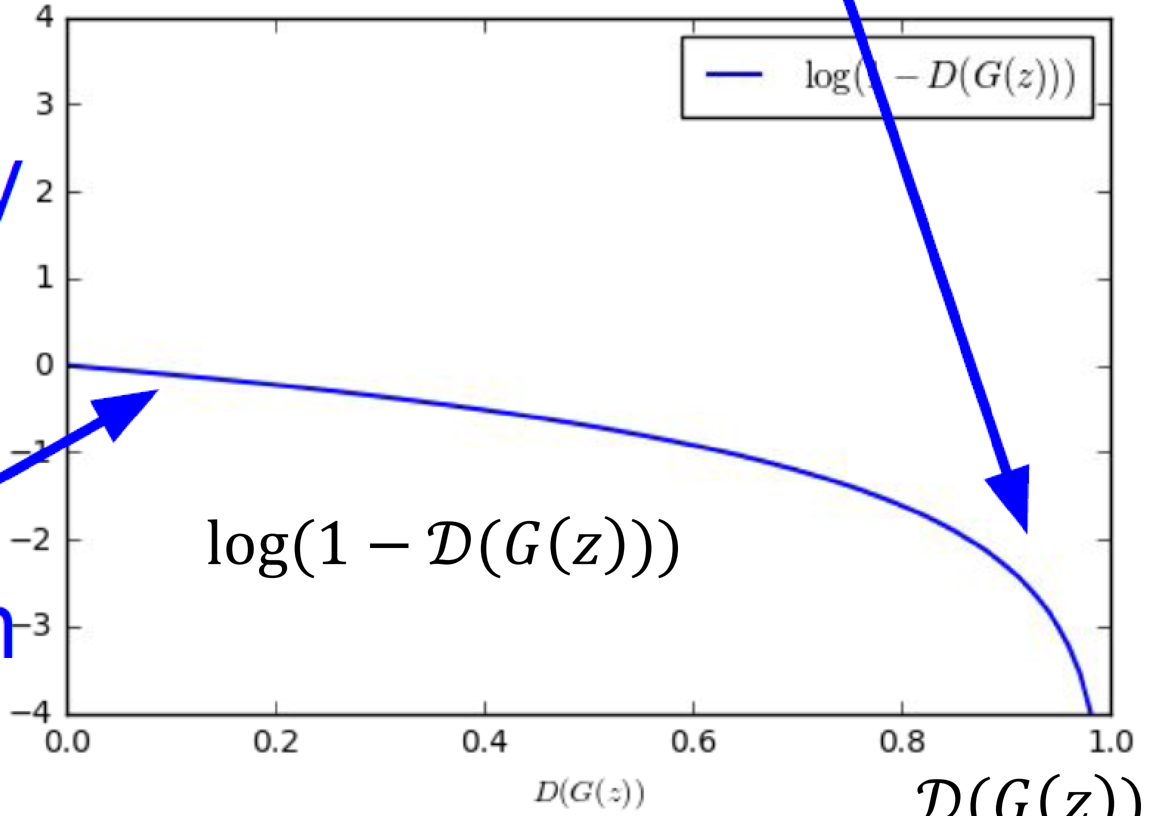
Update $\mathcal{G}$ by stochastic gradient descent:

$$\nabla_{\theta_g} \left[\sum_i \log\left(1 - \mathcal{D}\left(\mathcal{G}(\boldsymbol{z}_i, \theta_g), \theta_d\right)\right) \right]$$

During early learning stages, when G is poor, D can reject samples with high confidence because they are clearly different from the training data (thus $D(G(z)) \approx 0$) . In this case, $\log(1 - D(G(z)))$ is flat, thus has very low gradient.

Gradient signal dominated by region where sample is already good

When sample is likely fake, want to learn from it to improve generator. But gradient in this region is relatively flat!



$\log(1 - \mathcal{D}(G(z)))$

$\mathcal{D}(G(z))$

One of the many GAN Training «trick»

When optimizing for θ_g, instead of minimizing the following

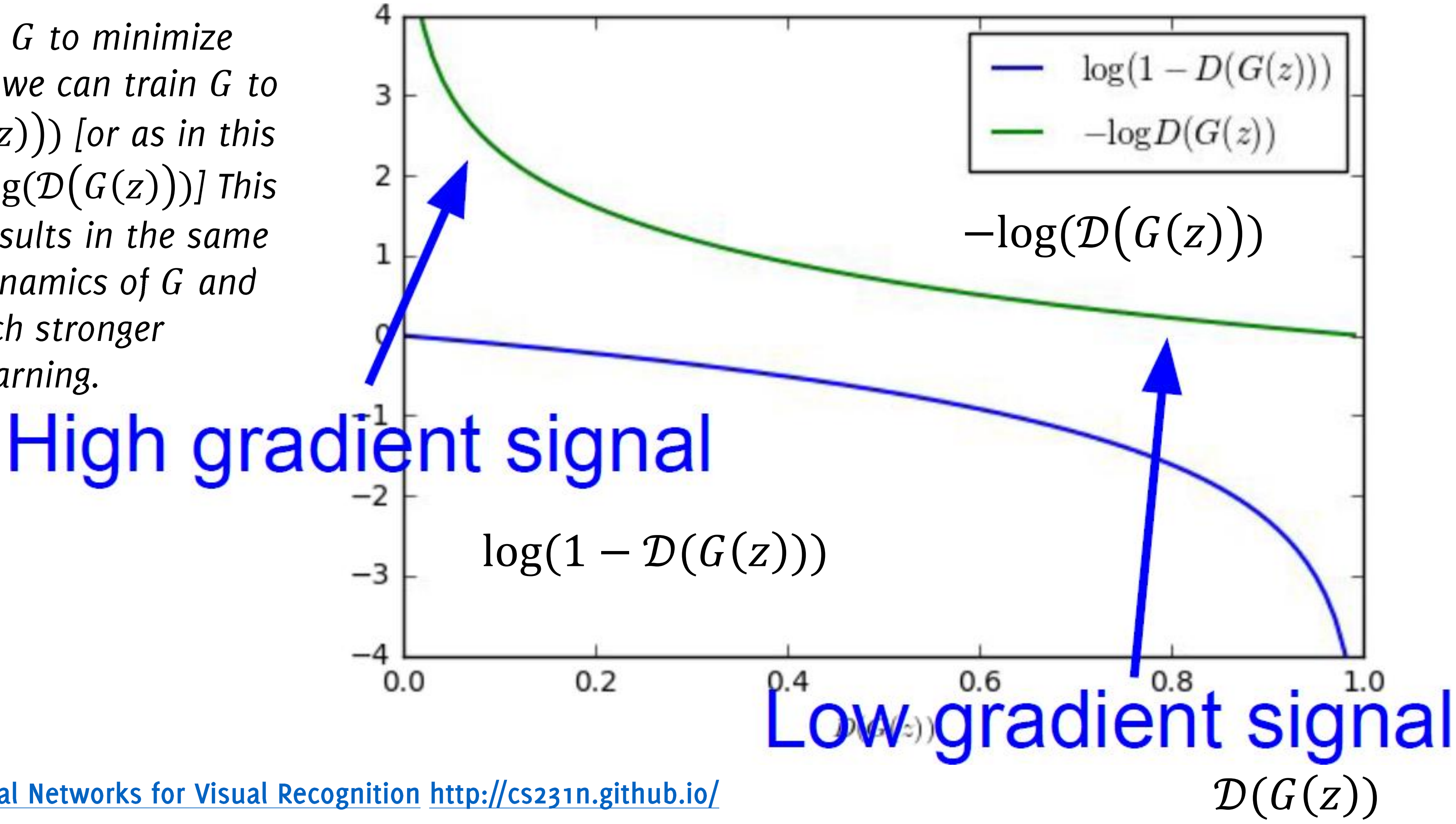
$$\min_{\theta_g} \left(\mathrm{E}_{z\sim\phi_Z} \left[\log\left(1 - \mathcal{D}(\mathcal{G}(\boldsymbol{z},\theta_g),\theta_d)\right) \right] \right)$$

we maximize this

$$\max_{\theta_g} \left(\mathrm{E}_{z\sim\phi_Z} \left[\log\left(\mathcal{D}(\mathcal{G}(\boldsymbol{z},\theta_g),\theta_d)\right) \right] \right)$$

Which is equivalent in therms of loss function.. provides a stronger gradient during the early learning stages

Rather than training G to minimize $log(1 - D(G(z)))$ we can train G to maximize $\log(\mathcal{D}(G(z)))$ [or as in this figure, minimize $-\log(\mathcal{D}(G(z)))$] This objective function results in the same fixed point of the dynamics of G and D, but provides much stronger gradients early in learning.



GAN Training

for $i = 1 \ldots$ #number of epochs

for k −times # gradient ascent steps for θ_d

- Draw a minibatch $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ of noise realization
- Sample a minibatch of images $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_m\}$
- Update θ_d by stochastic gradient ascend:

$$\nabla_{\theta_d}\left[\sum_i \log \mathcal{D}(\boldsymbol{s_i}, \theta_d) + \log\left(1 - \mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}_i, \theta_g\right), \theta_d\right)\right)\right]$$

Draw a minibatch $\{\boldsymbol{z}_1, \ldots, \boldsymbol{z}_m\}$ of noise realizations # gradient descent steps for θ_g

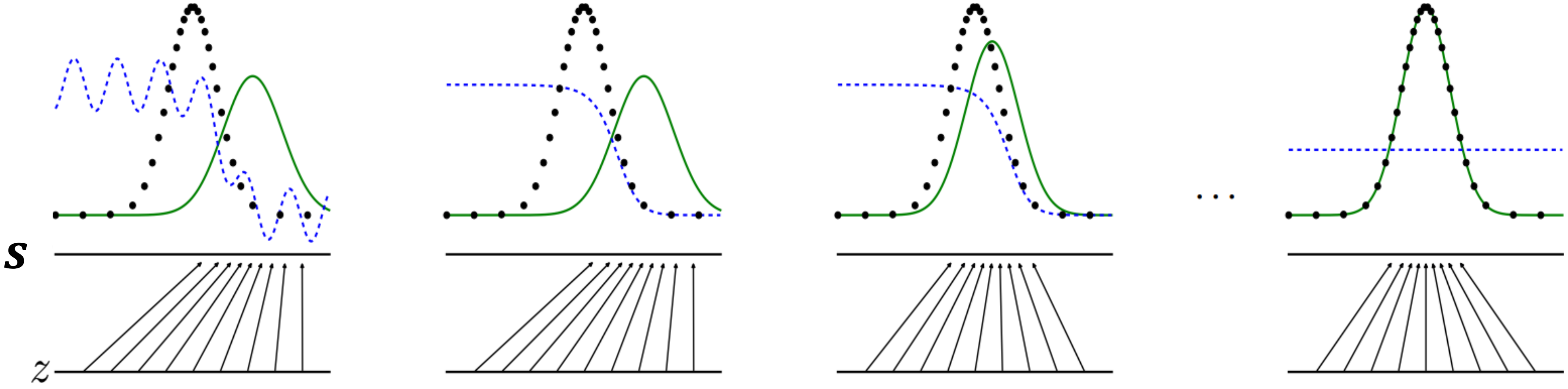
Update $\mathcal{G}$ by stochastic gradient **ascent**:

$$\nabla_{\theta_g}\left[\sum_i \log\left(\mathcal{D}\left(\mathcal{G}\left(\boldsymbol{z}_i, \theta_g\right), \theta_d\right)\right)\right]$$

Illustration of the GAN Training Process

In this illustration $\mathbb{R}^d$ and $\mathbb{R}^n$ are collapsed into 1d points
this allows also the visualization of their distribution

- dotted (dark): $\boldsymbol{\phi}_{\boldsymbol{s}}$, $\boldsymbol{s}$ real
- solid (green): $\boldsymbol{\phi}_{\mathcal{G}(\boldsymbol{z})}$ $\mathcal{G}(\boldsymbol{z})$ fake
- dotted (blue): $\mathcal{D}(\cdot)$ $\mathcal{D}$ posterior



Goodfellow, I. et al "Generative adversarial nets" NIPS 2014

At the end of the day…

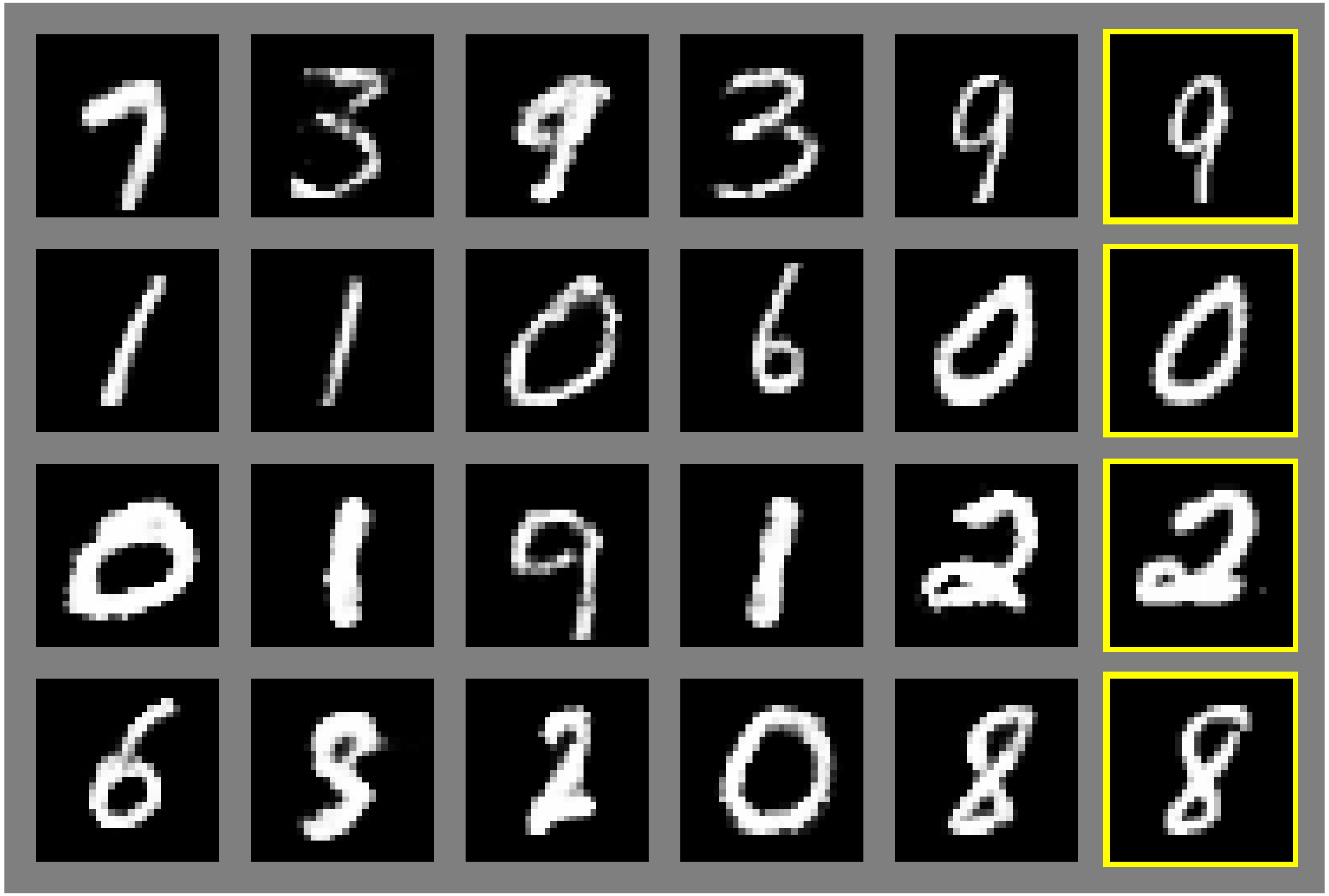
The discriminator $\mathcal{D}$ is discarded

The generator $\mathcal{G}$ and ϕ_Z are preserved as generative model

Remarks:

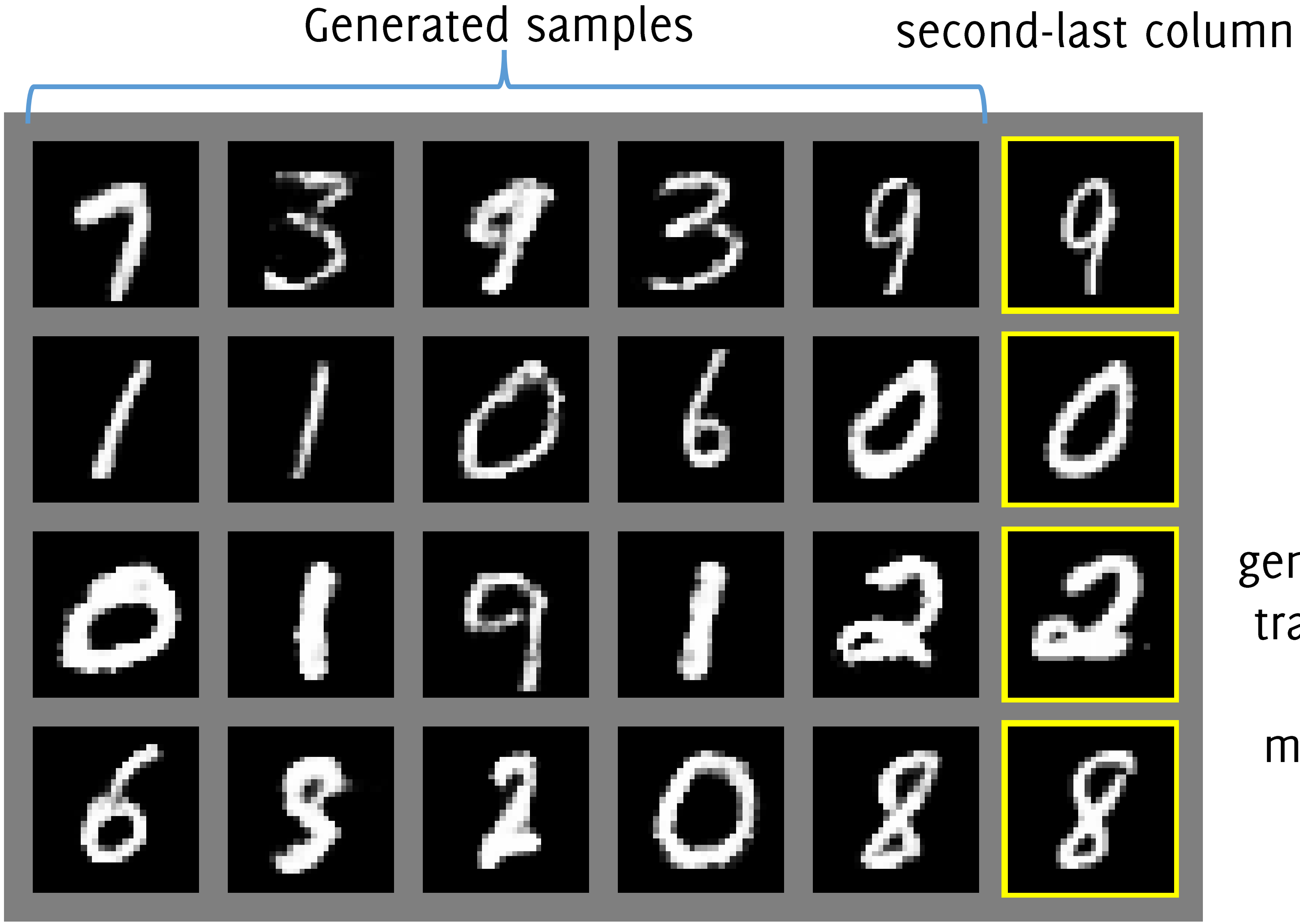
- **The training is rather unstable**, need to carefully synchronize the two steps (many later works in this direction, e.g. Wasserstein GAN)
- **Training by standard tools**: backpropagation and dropout
- Theoretical analysis provided in the paper
- **Generator does not use $\boldsymbol{S}$ directly during training**
- **Generator performance is difficult to assess quantitatively**
- There is **no explicit expression for the generator**, it is provided in an implicit form -> you cannot compute the likelihood of a sample w.r.t. the learned GAN

MNIST



Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

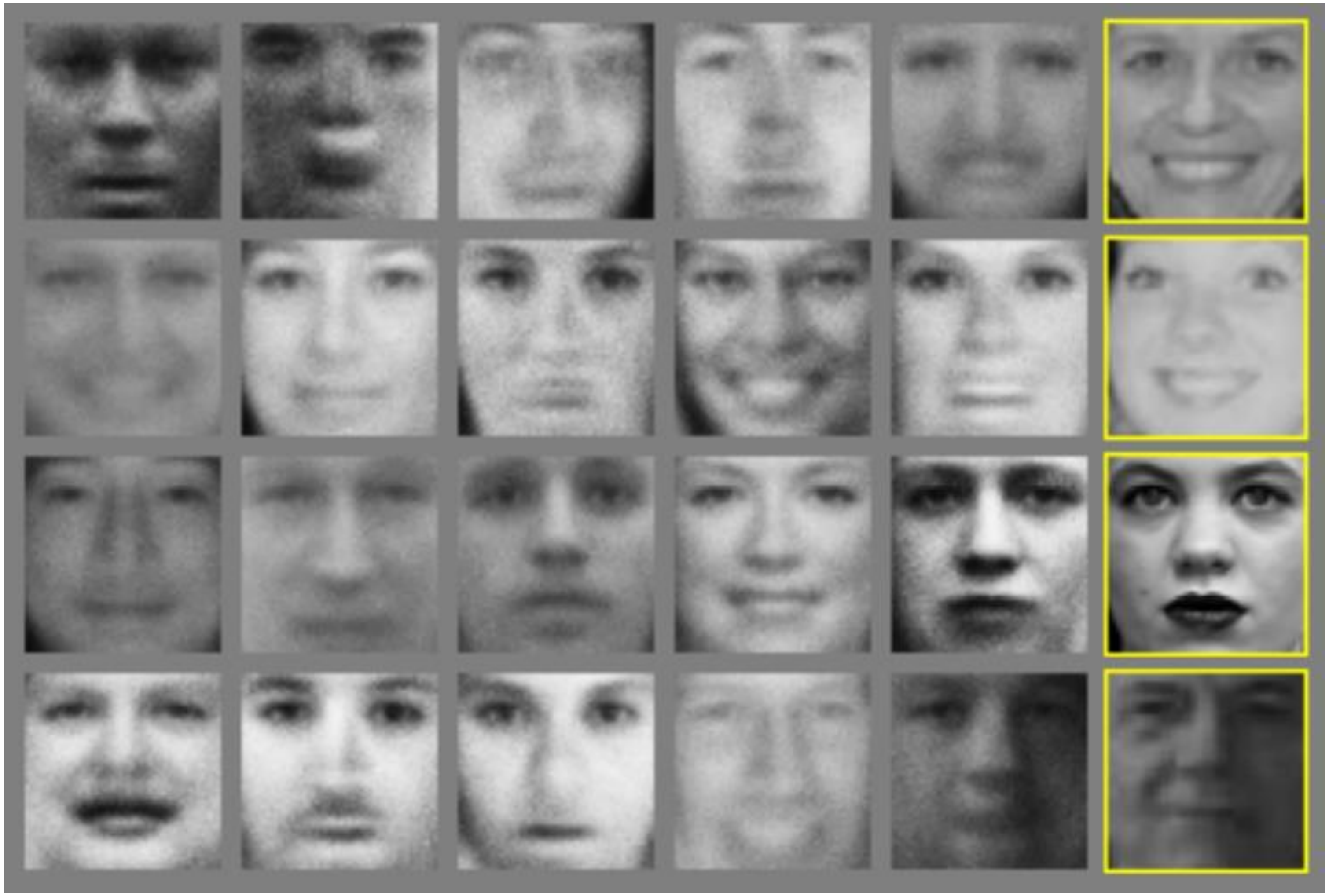
MNIST



This GAN generates realistic training samples without memorizing the training set

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Toronto Face Database (TFD)

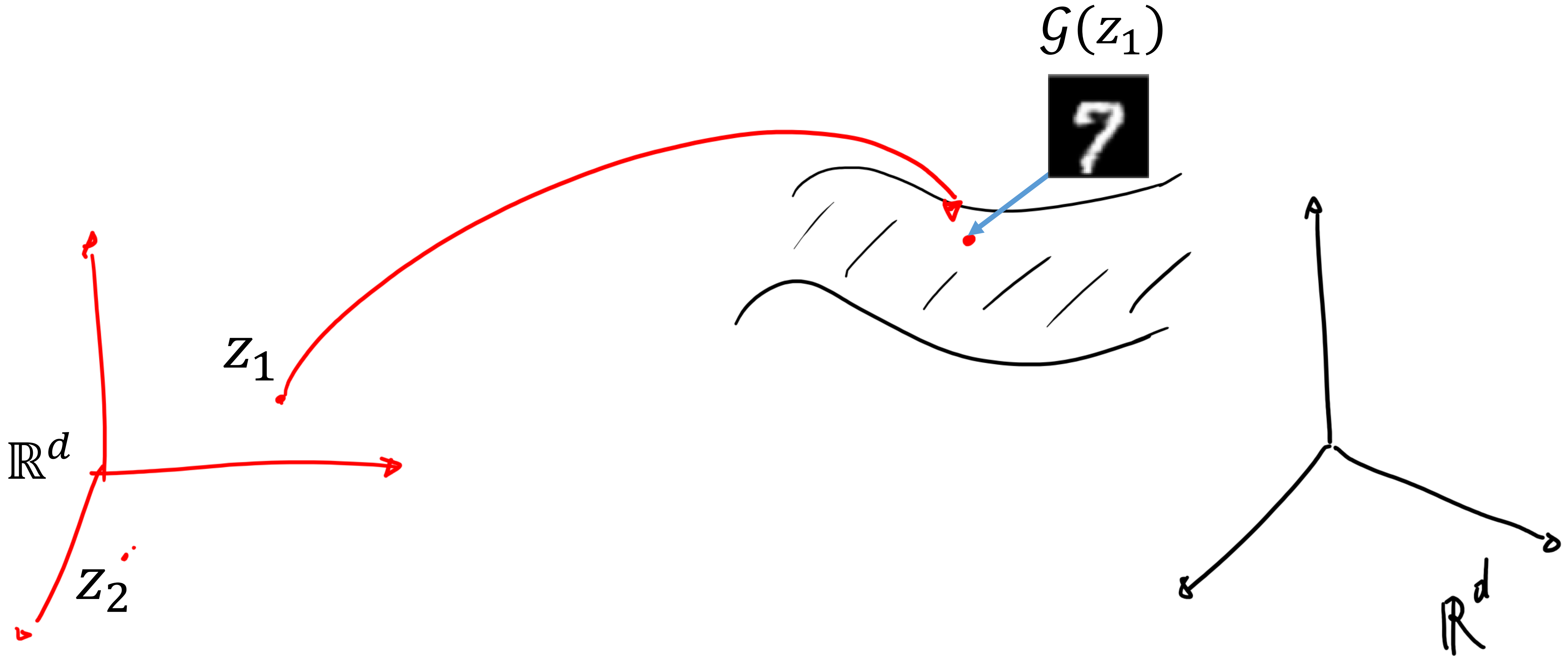


Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

CIFAR-10

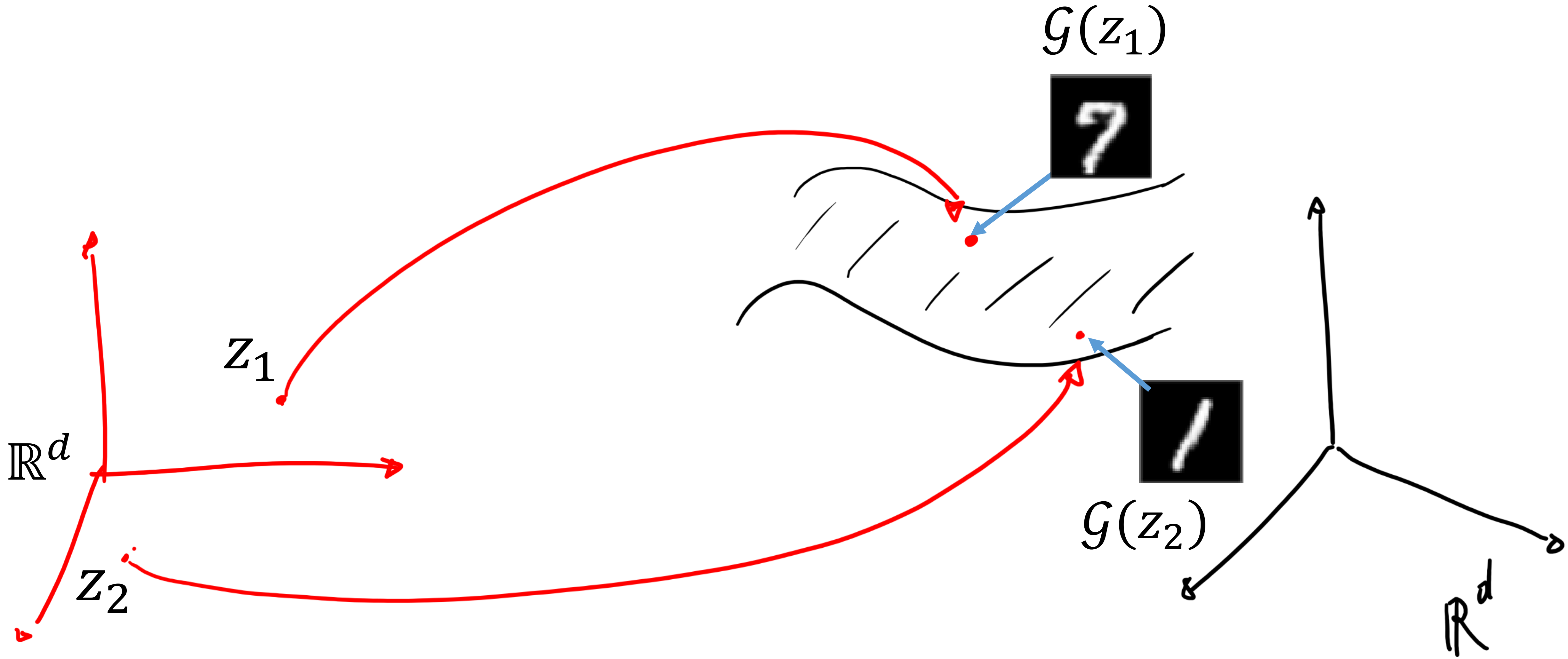
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Interpolation experiment: the manifold was learned!



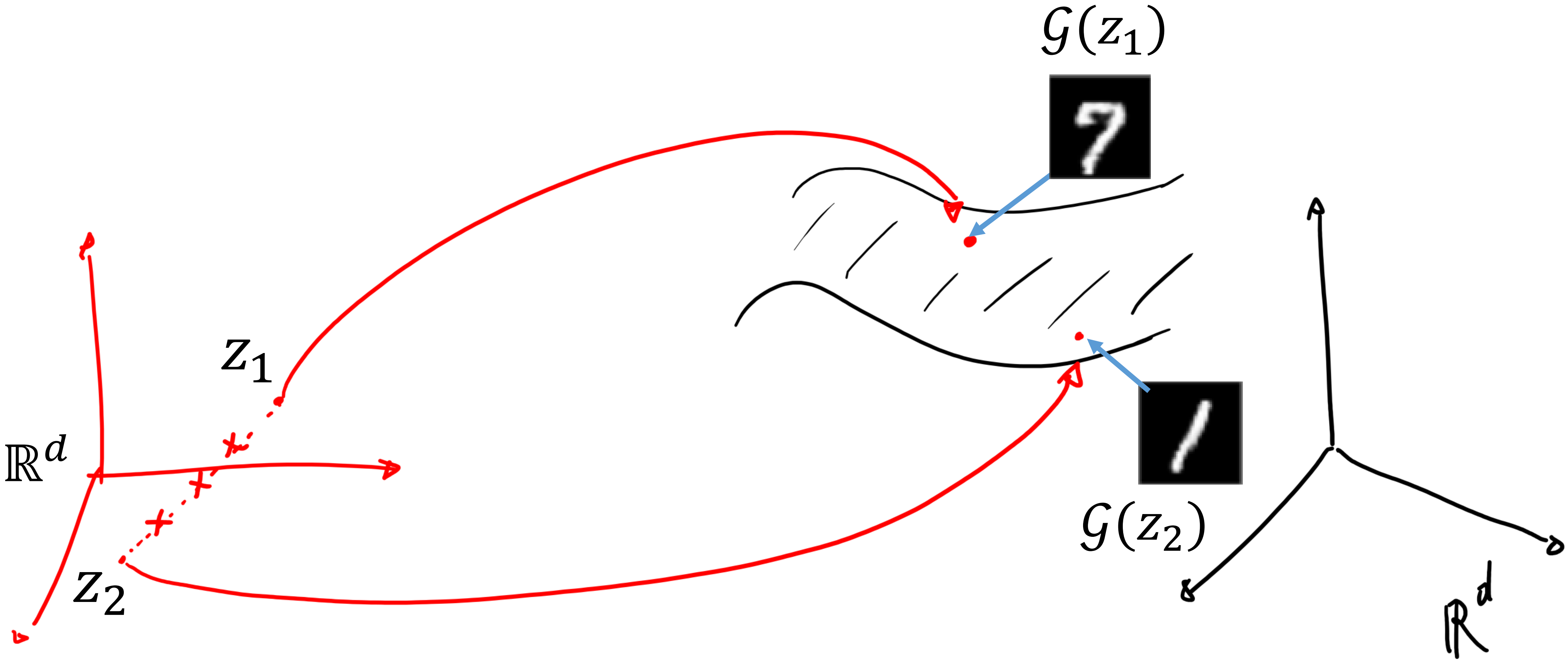
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Interpolation experiment: the manifold was learned!



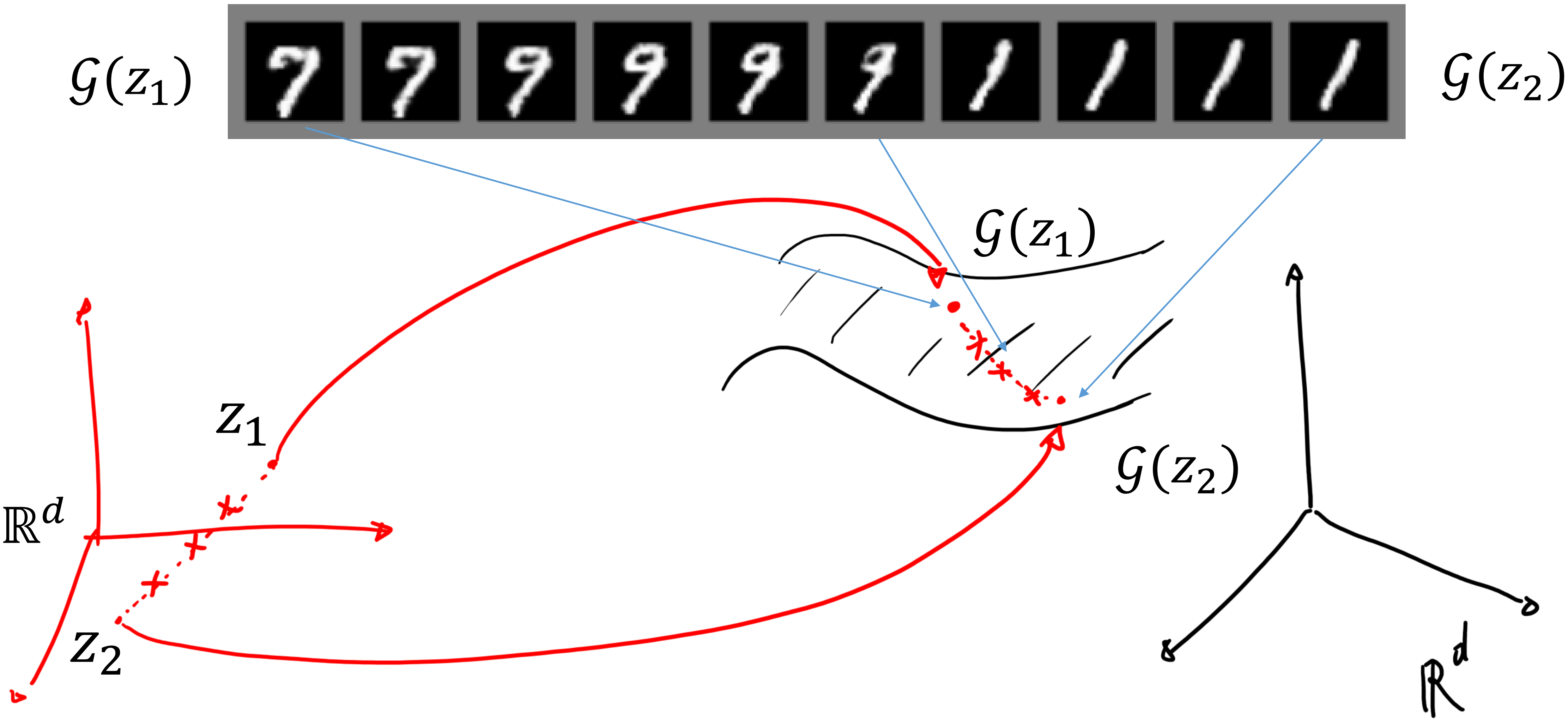
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Interpolation experiment: the manifold was learned!



Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

Interpolation experiment: the manifold was learned!

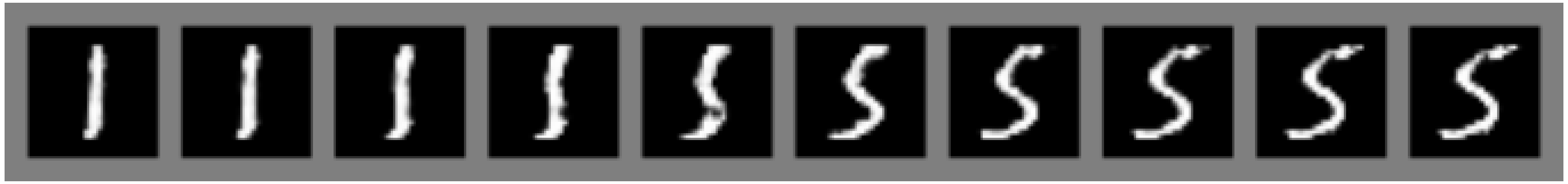


Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

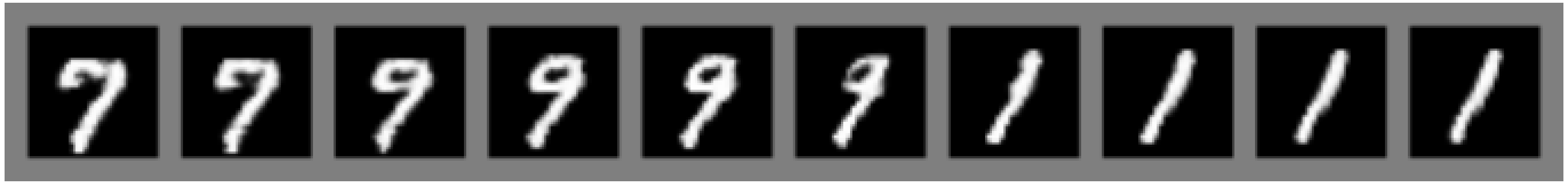
Outputs of interpolated trajectories

Select two noise realization z_1 and z_2 yielding reasonable outputs, and interpolate among the two. Generate the images of intermediate values

$\mathcal{G}(z_1)$ $\mathcal{G}(z_2)$



$\mathcal{G}(z_3)$ $\mathcal{G}(z_4)$

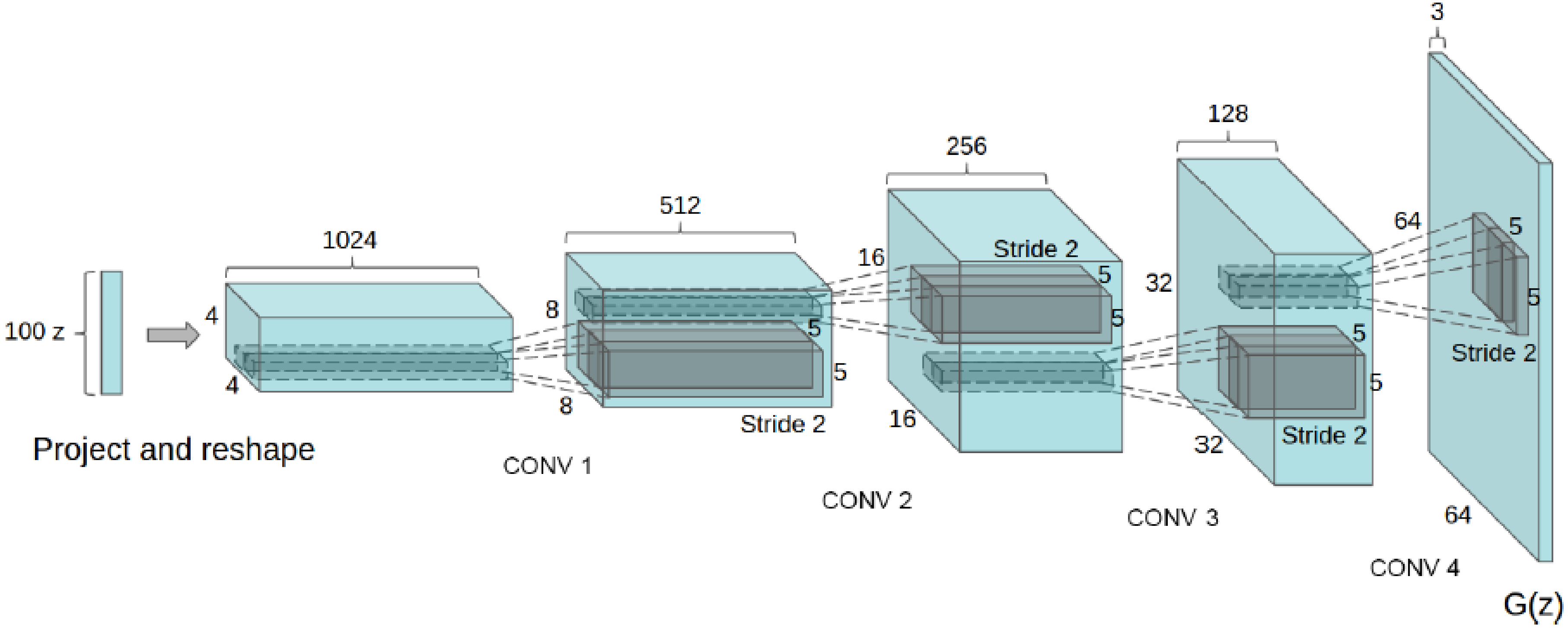


Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, ... & Bengio, Y. Generative adversarial nets NIPS 2014

GANs have much improved over the years

DC-GAN: Deep Convolutional GANs

RGB Image



Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

GANs have much improved in the last few years

Images generated after 1 training epochs

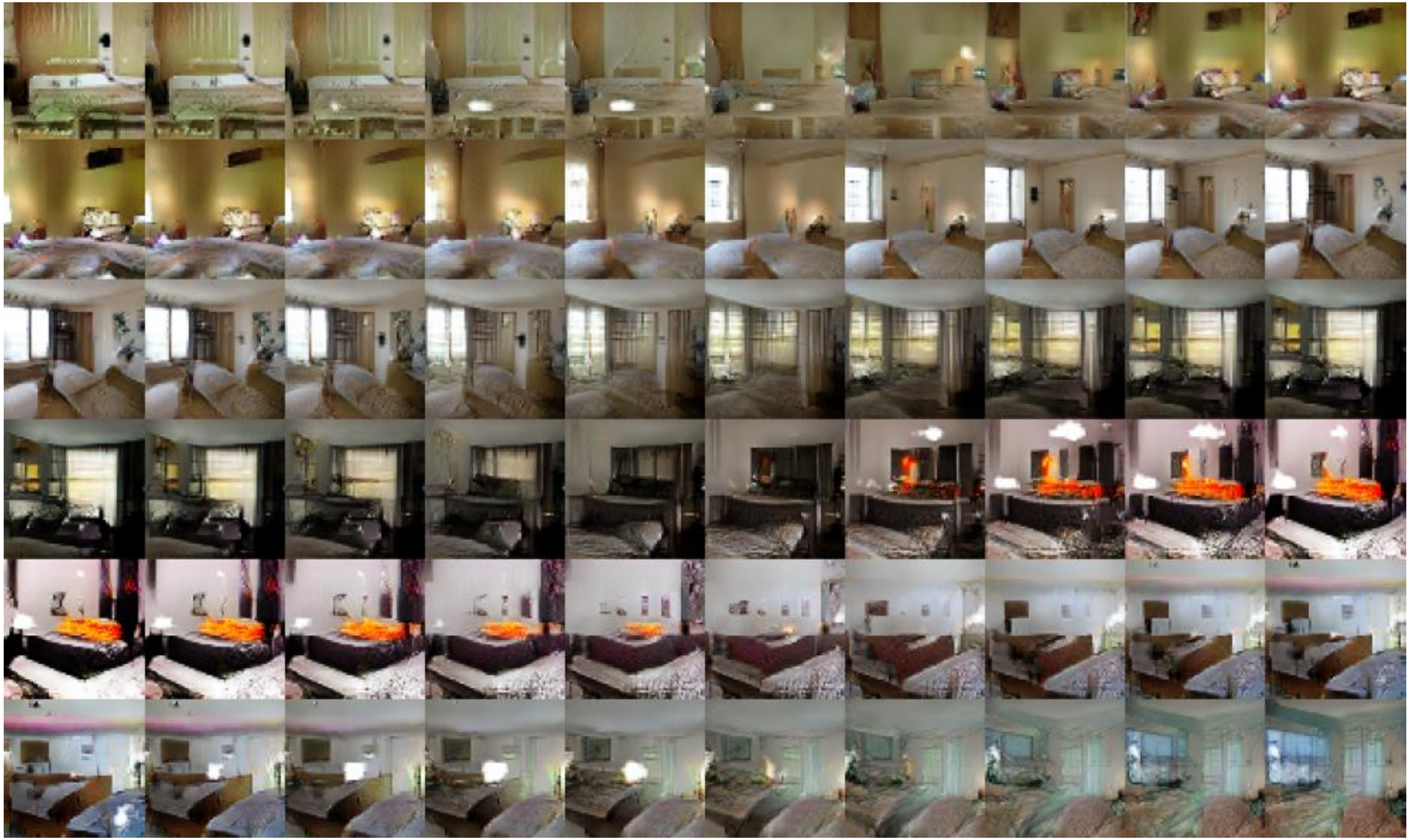
Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

GANs have much improved in the last few years

Images generated after 5 training epochs

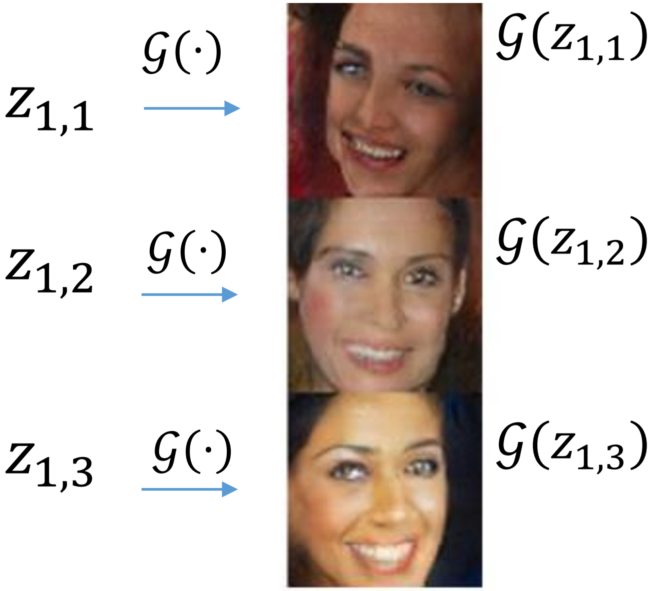
Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Interpolation between a series of 9 random points



Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

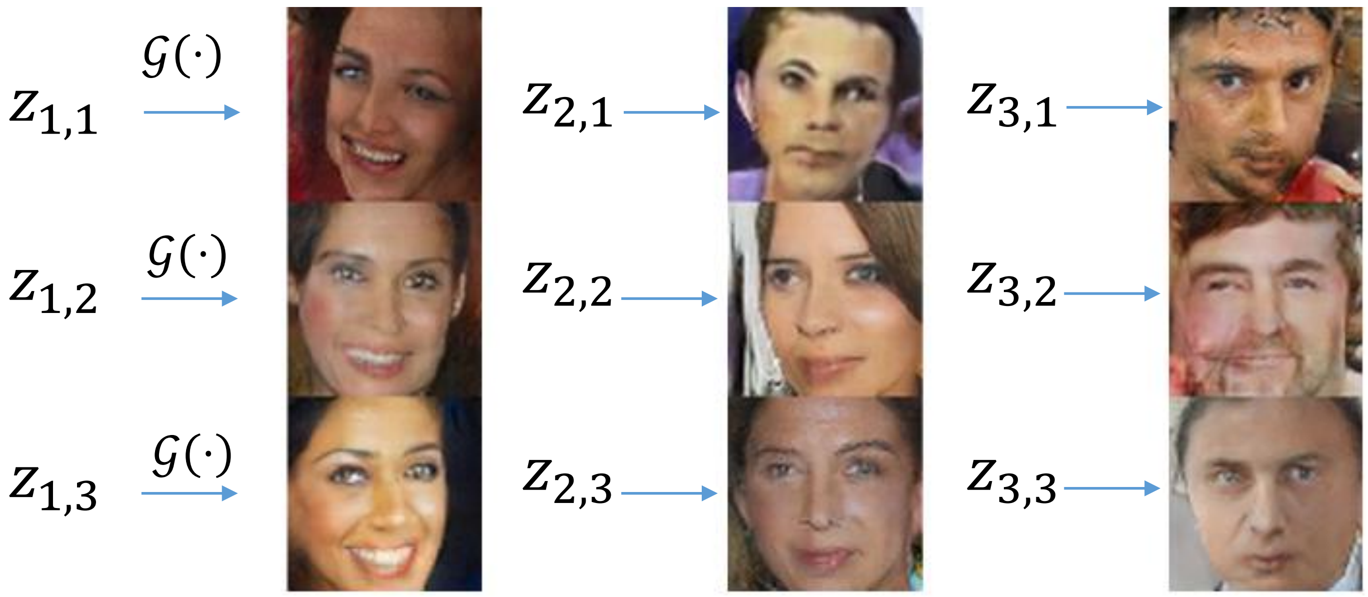
Vector Arithmetic



Take randomly generated samples of smiling women, neutral women, and neutral men

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

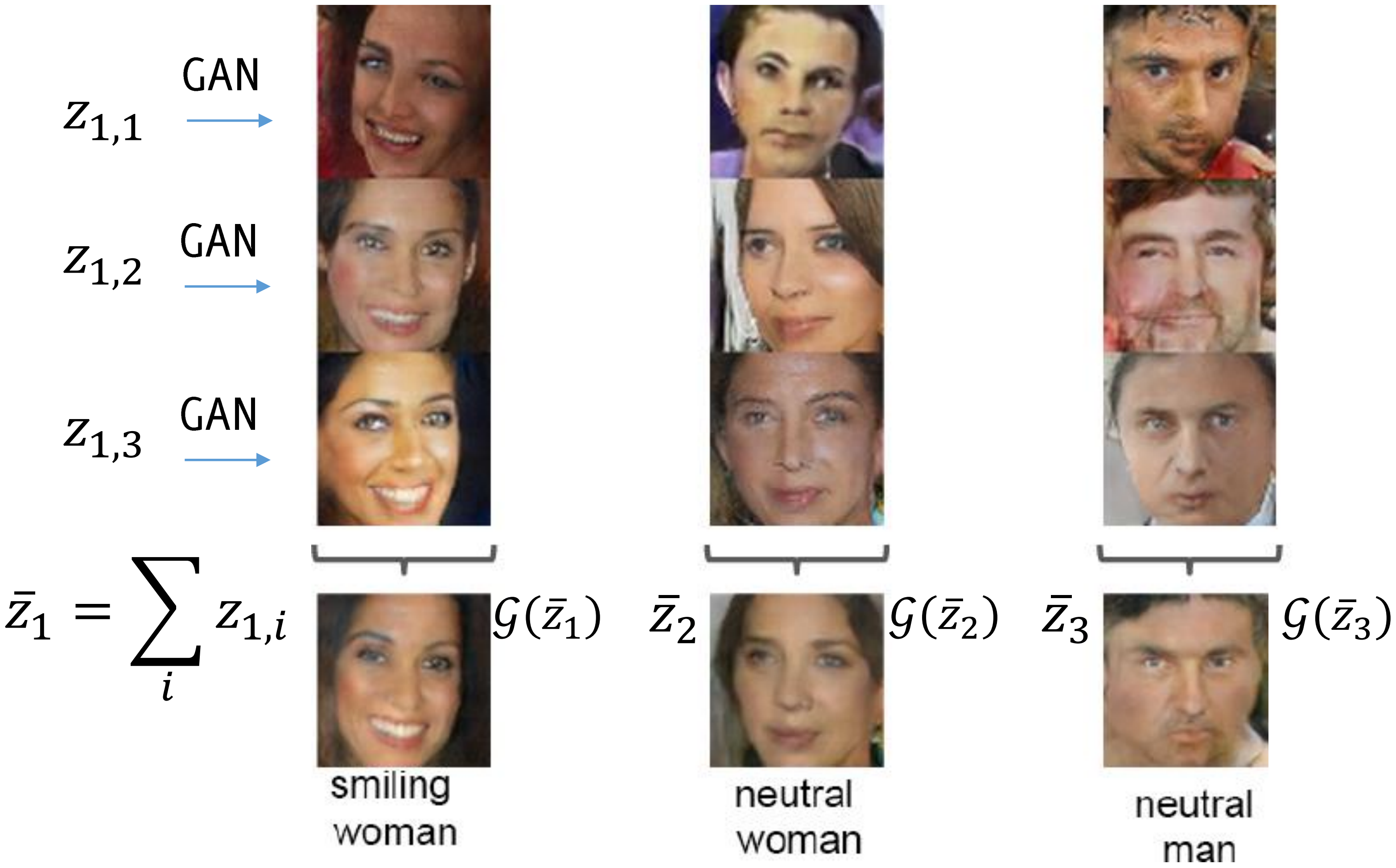
Vector Arithmetic



Take randomly generated samples of smiling women, neutral women, and neutral men

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

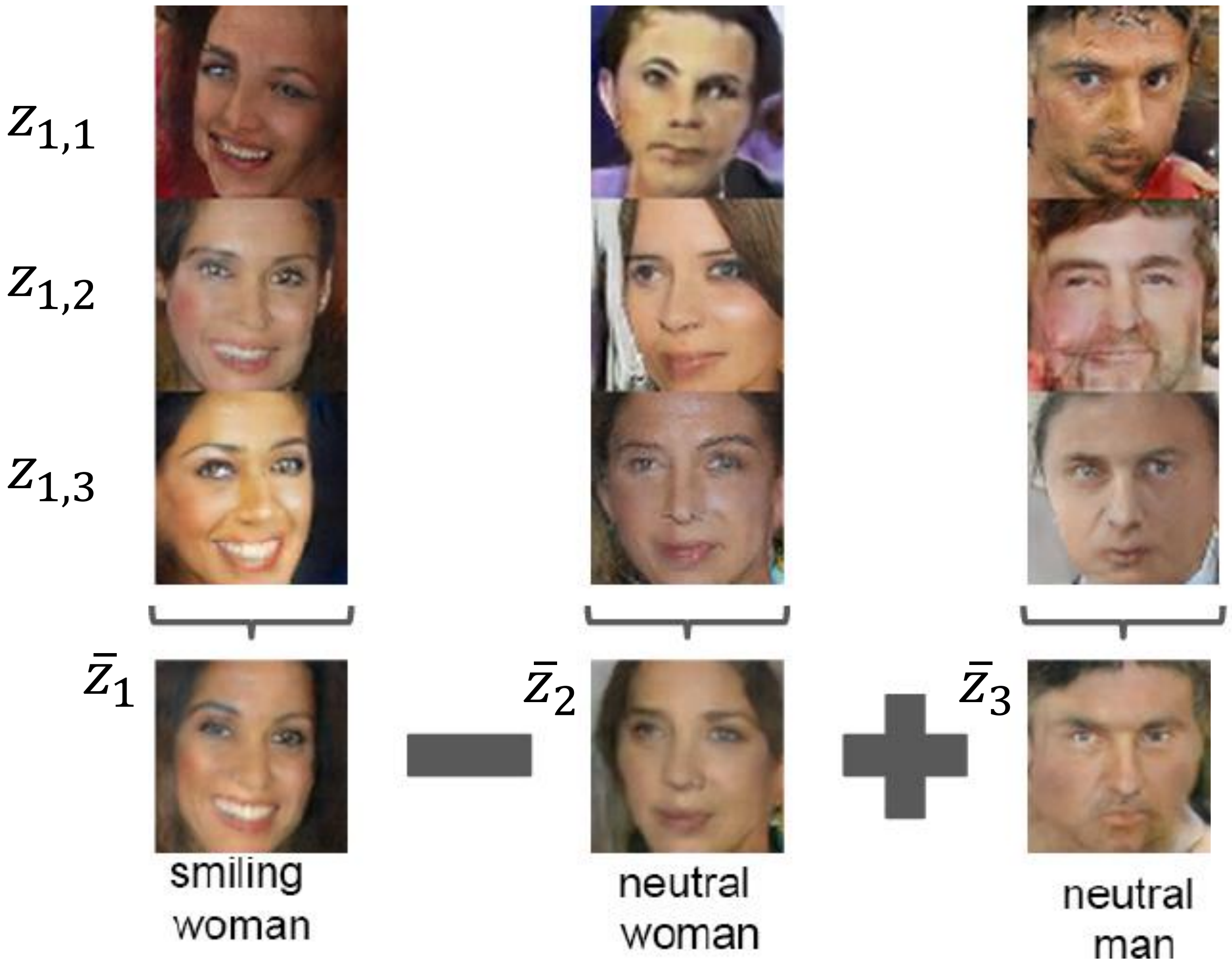
Vector Arithmetic



Average the corresponding noise seeds

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

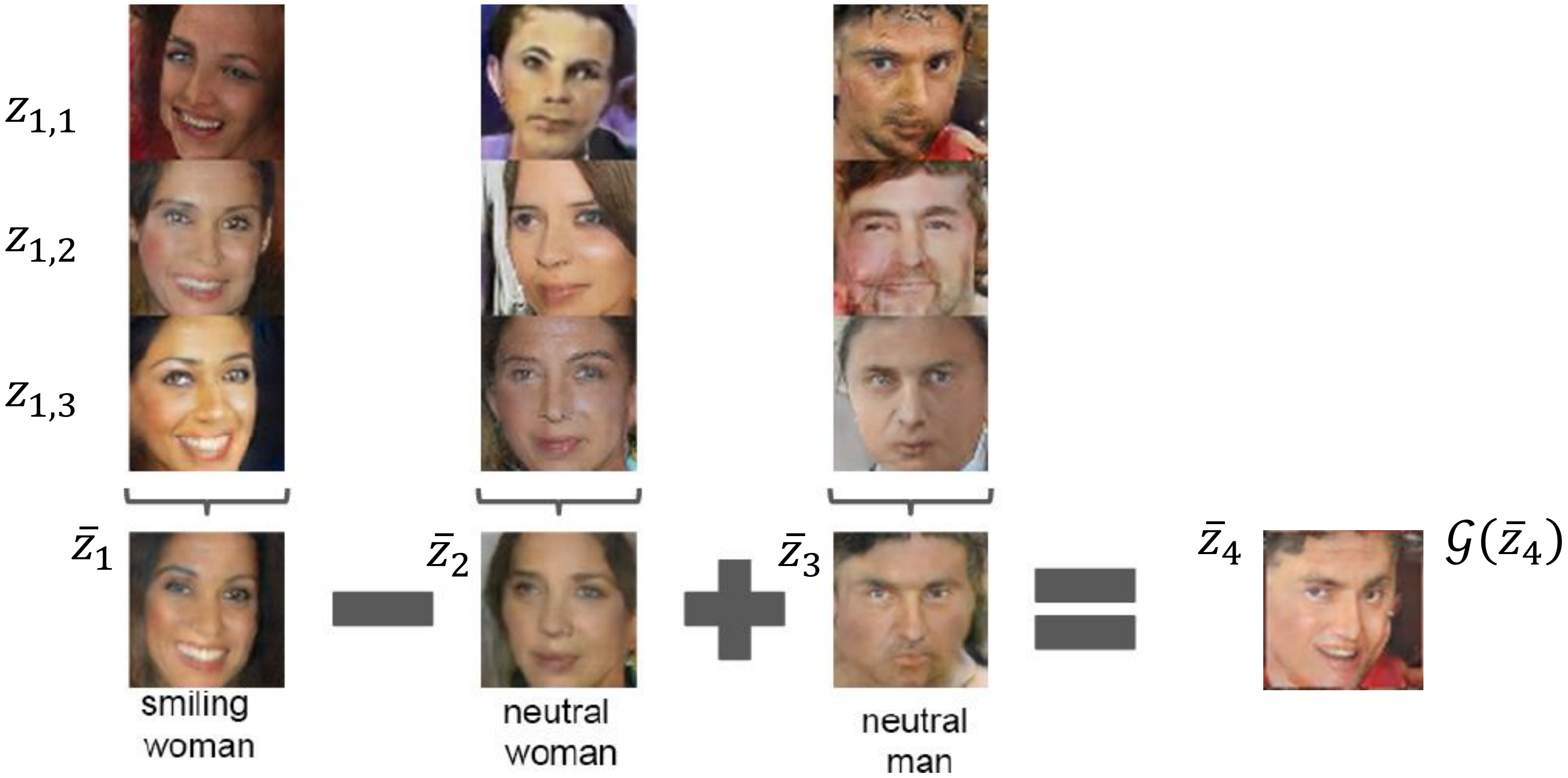
Vector Arithmetic



Perform some arithmetic

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

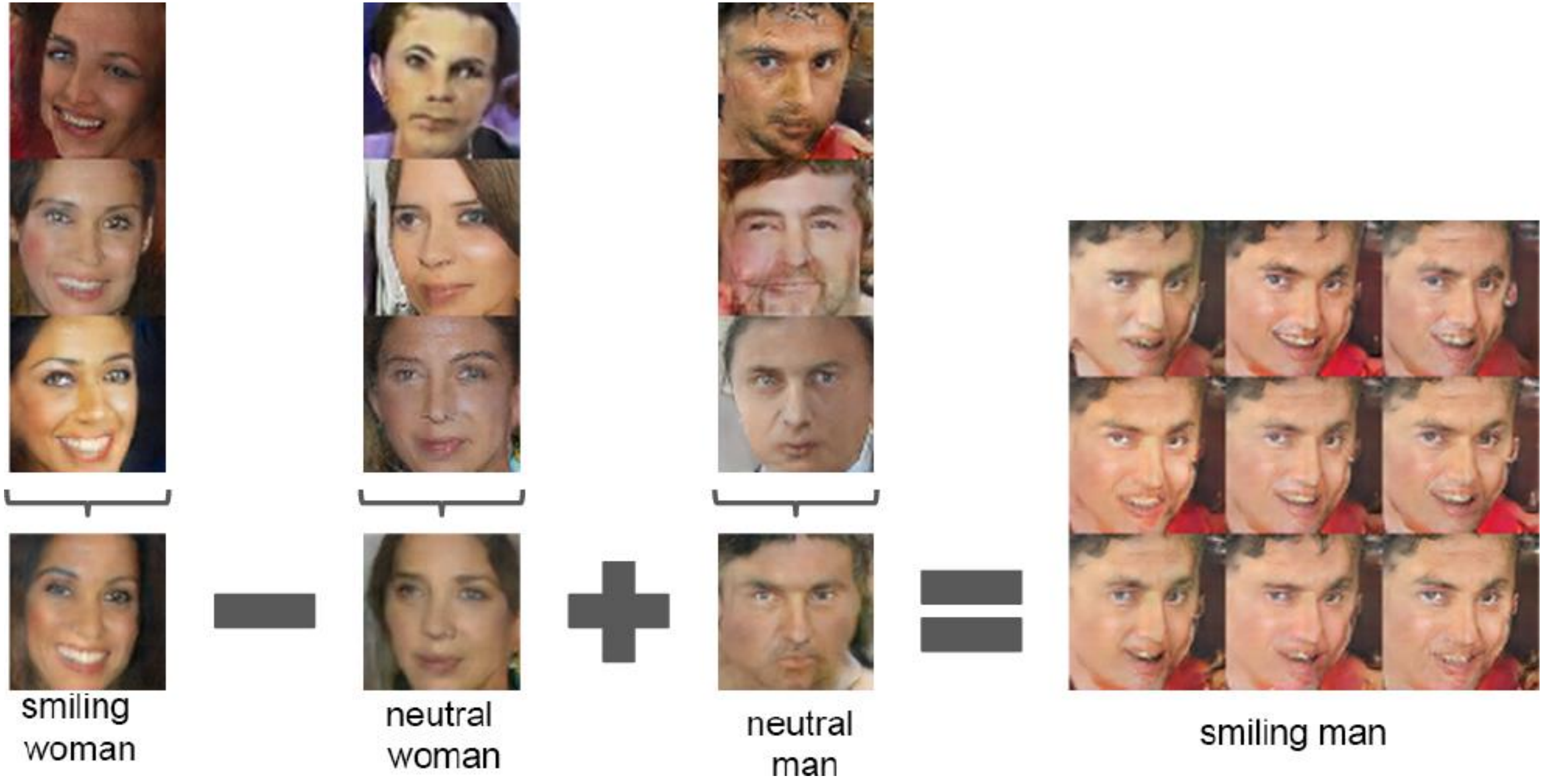
Vector Arithmetic



Perform some arithmetic $\bar{z}_1 - \bar{z}_2 + \bar{z}_3 = \bar{z}_4$

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Vector Arithmetic

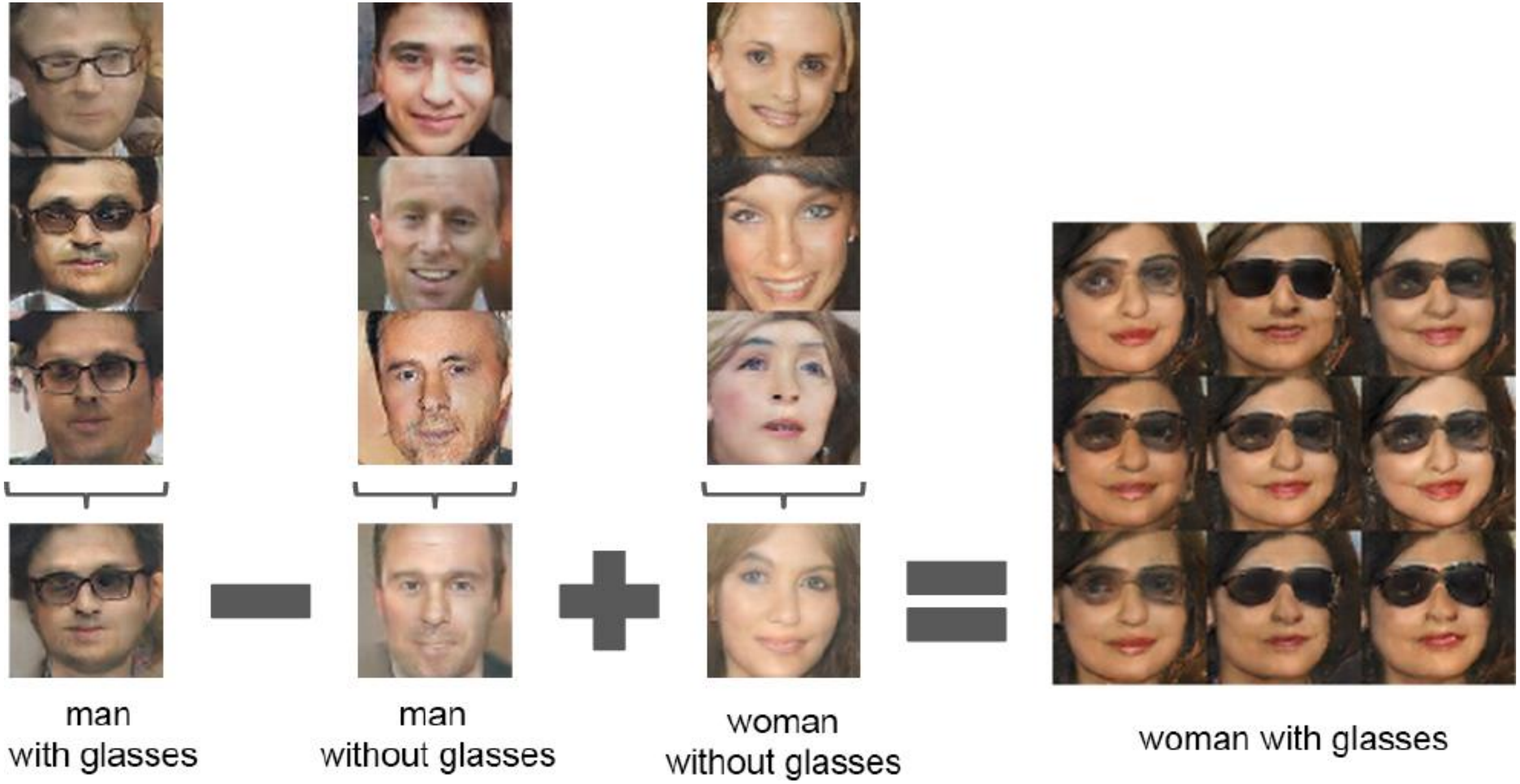


$\mathcal{G}(\bar{z}_4 + \eta)$

Add some noise to the input vector and that's pretty robust

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Vector Arithmetic



Similar example as word embedding

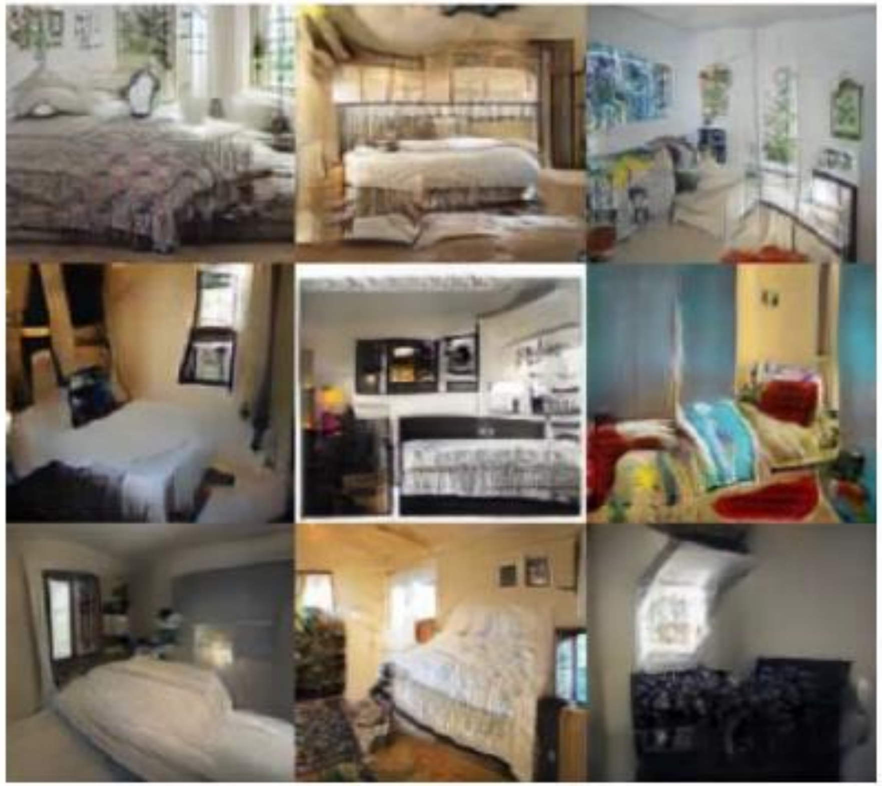
Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Vector Arithmetic

Interpolation of view changes

Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

GAN has been a very active research field



LSGAN, Zhu 2017.

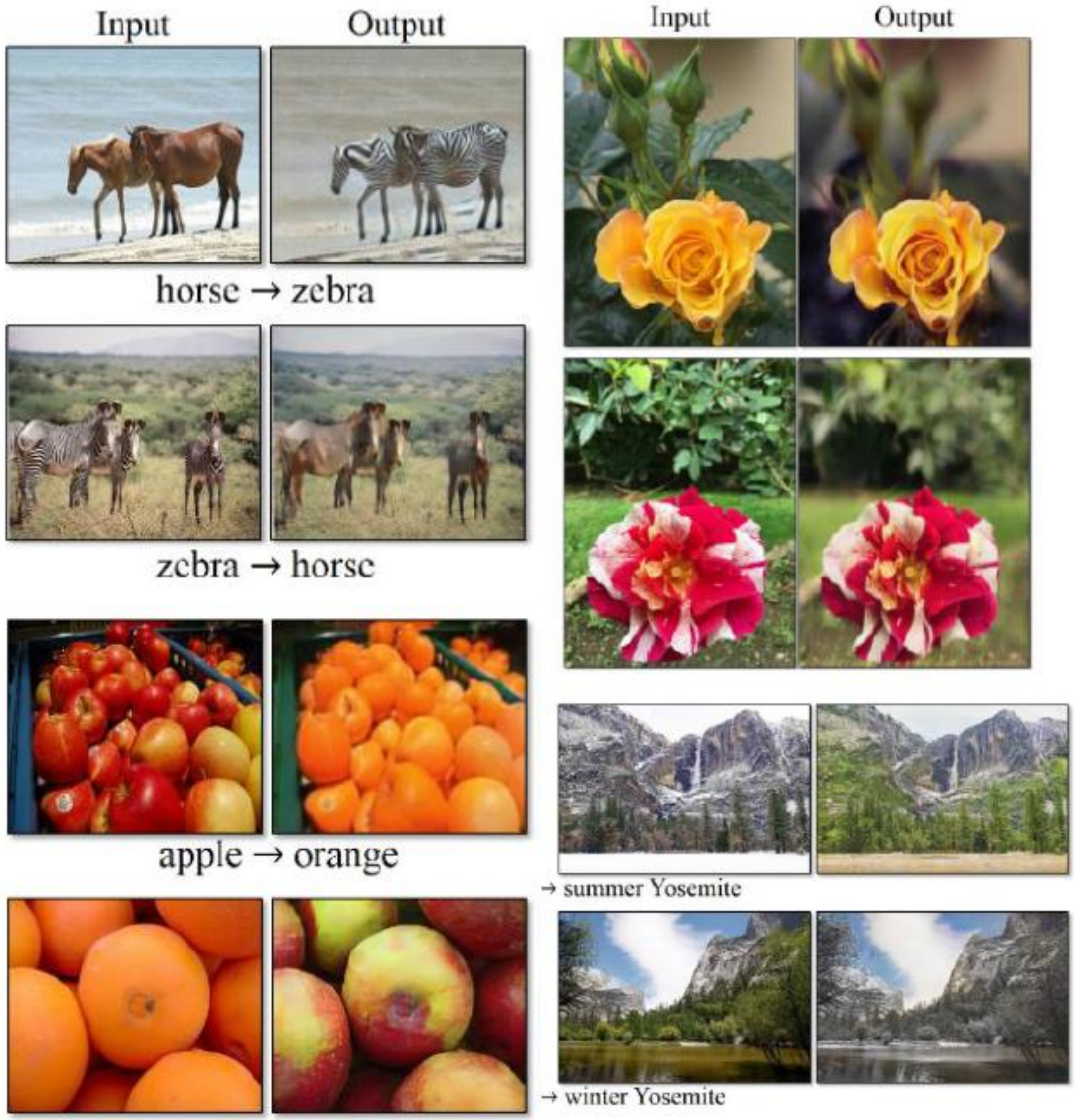
Wasserstein GAN, Arjovsky 2017. Improved Wasserstein GAN, Gulrajani 2017.

GAN has been a very active research field

Karras, Tero, et al. "Progressive growing of gans for improved quality, stability, and variation." *ICLR 2018*

GANs for source-target domain transfer

This can also be used for photo enhancement and data augmentation



CycleGAN. Zhu et al. 2017.

… «The GAN ZOO» and https://github.com/soumith/ganhacks

Reed et al. 2017.

Pix2pix. Isola 2017. Many examples at https://phillipi.github.io/pix2pix/

Generative Adversarial Networks (these people do not exist)

Tero Karras, Samuli Laine, Timo Aila «A Style-Based Generator Architecture for Generative Adversarial Networks" CVPR 2019

Karras, Tero, et al. "Analyzing and improving the image quality of stylegan." CVPR 2020

https://thissneakerdoesnotexist.com/editor/?seed=1403&style=2

This sneaker does not exist

The Grid | 3D demo (new) | Info | Contact

Sneaker Editor

Use the sliders below the image to edit the sneaker

Normal — Futuristic

Low creativity — High creativity

Lighter color — Darker color

Return to grid

https://thissneakerdoesnotexist.com/editor/?seed=1403&style=2

This sneaker does not exist

The Grid | 3D demo (new) | Info | Contact

Sneaker Editor

Use the sliders below the image to edit the sneaker

Normal — Futuristic

Low creativity — High creativity

Lighter color — Darker color

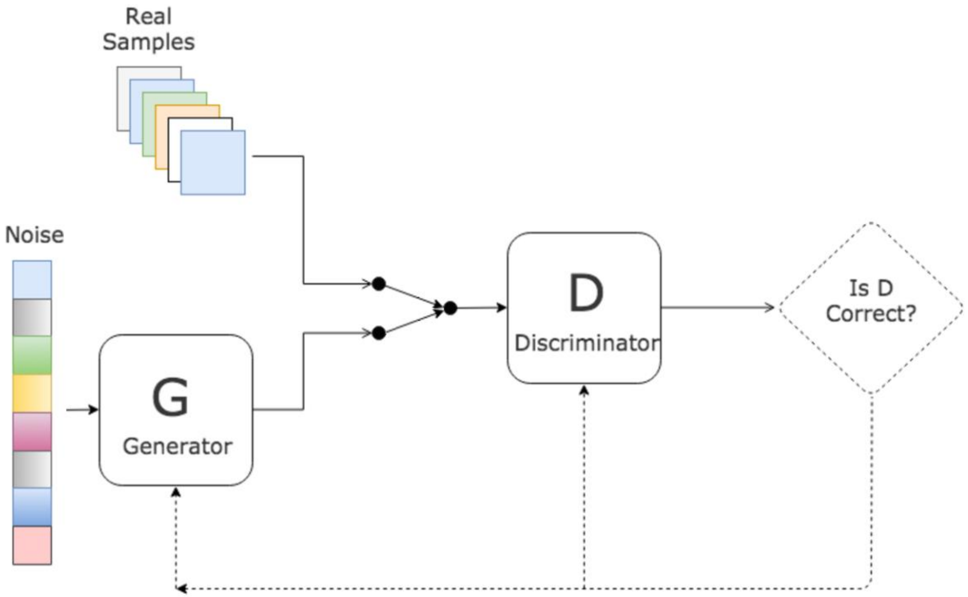
Return to grid

Intuition behind Conditional GANs

Deep Convolutional Conditional GAN

Suppose each images in S are connected with any auxiliary information y, such as class labels (e.g. digits images + the digit number)

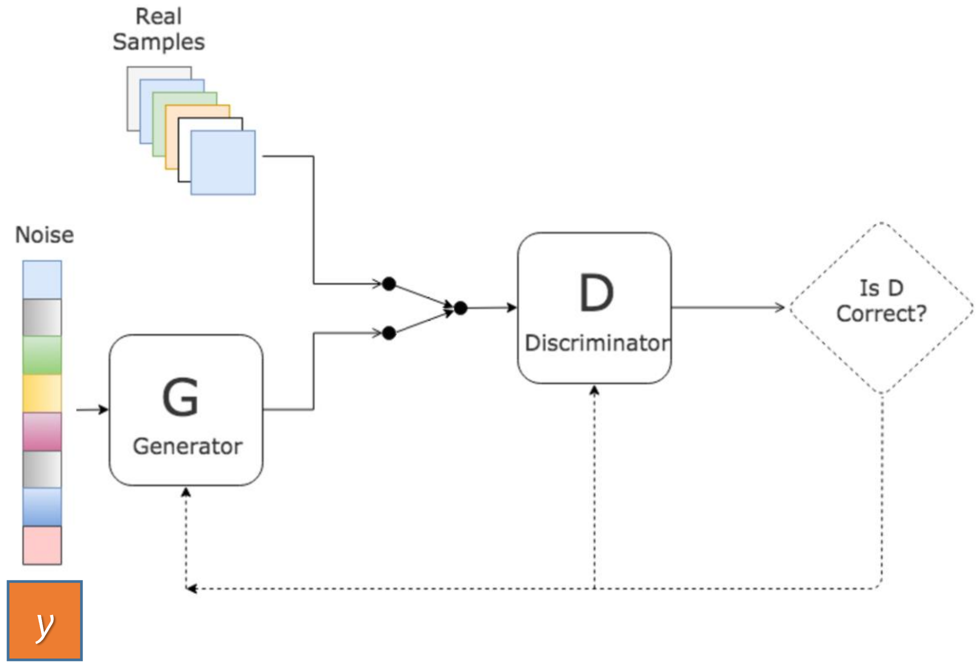
Where should this information be inserted for steering image generation?



Deep Convolutional Conditional GAN

We can concatenate this one-hot-encoded at the end of input noise.

Hopefully, the Generator $\mathcal{G}$ will learn to generate an image of the same class….

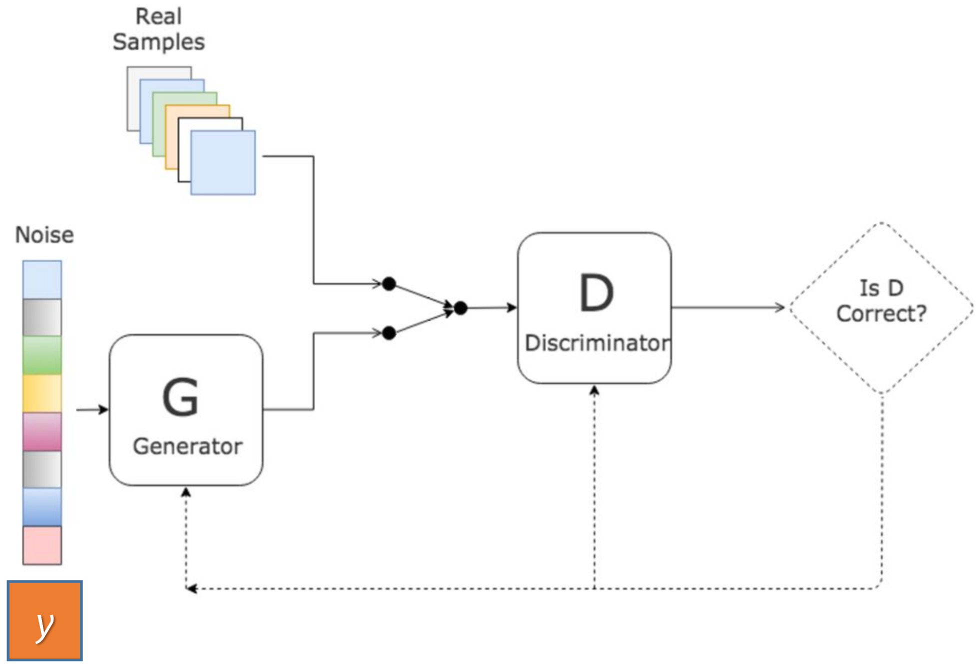


Deep Convolutional Conditional GAN

We can concatenate this one-hot-encoded at the end of input noise.

Hopefully, the Generator $\mathcal{G}$ will learn to generate an image of the same class….

How to make sure about this?

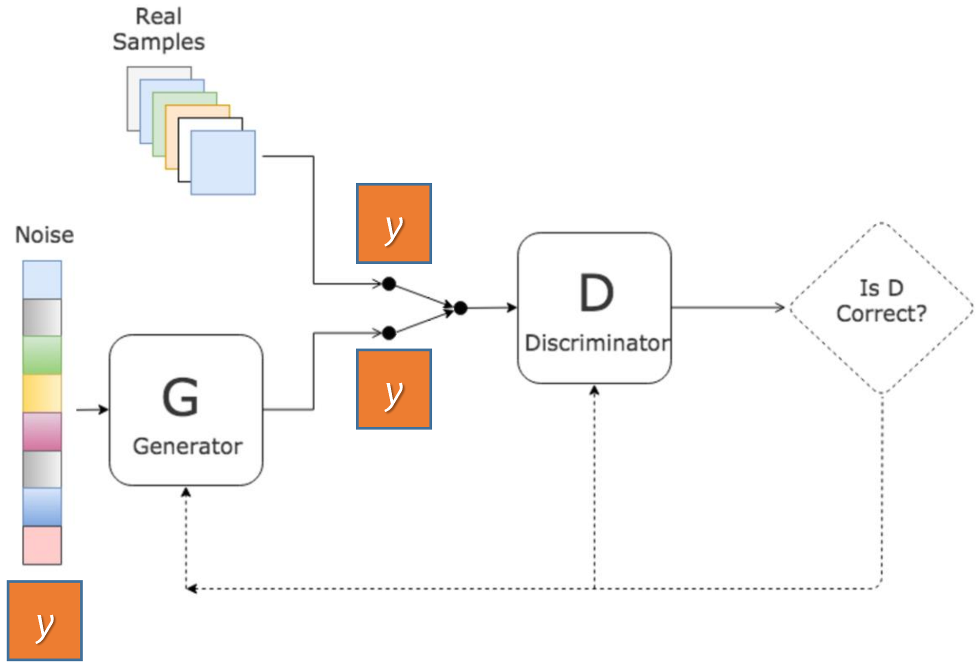


Deep Convolutional Conditional GAN

We also **append the label information in a one-hot-encoded channels at the end of real images** and generated images as well.

Generated images will also have this additional column.

This will allow the discriminator to easily classify as “fake” generated images whose content is not consistent with the encoded class label. Indeed, such consistency is guaranteed on real images.



GAN for Anomaly Detection

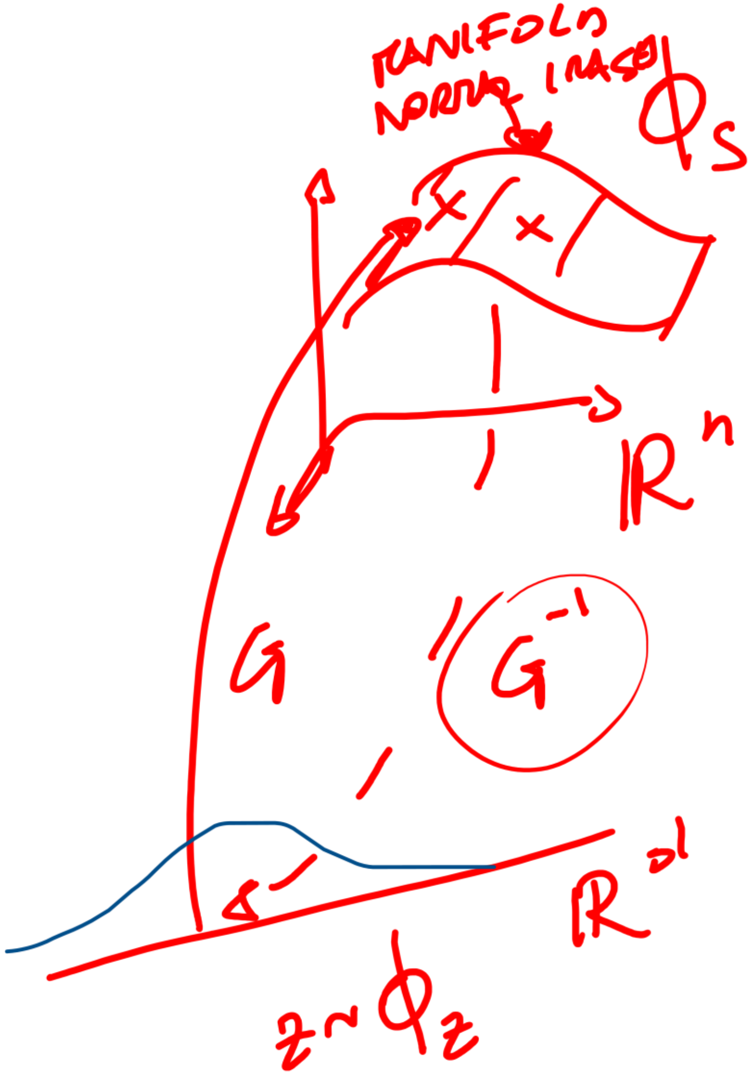
Thanks to Stefano Pecchia, former AN2DL student!

The intuition

GANs can successfully establish a mapping between random variables and the manifold of images

We might have a wonderful anomaly detection model **if**:

- we train a GAN $\mathcal{G}$ to generate normal images (an in particular texture images)
- we invert the GAN mapping and get $\mathcal{G}^{-1}$



Anomaly Detection in images

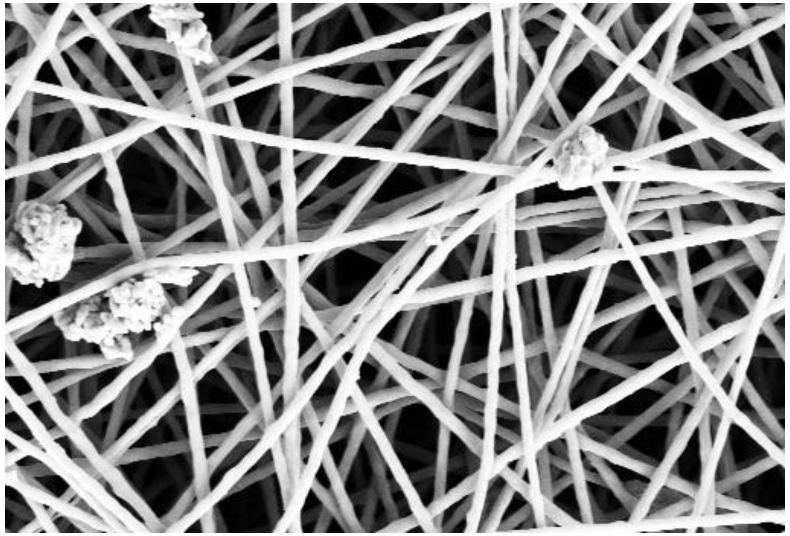
Let s be an image defined over the pixel domain $\mathcal{X} \subset \mathbb{Z}^2$, let $c \in \mathcal{X}$ be a pixel and $s(c)$ the corresponding intensity.

We want to **locate any anomalous region** in s, i.e. **estimating the anomaly mask $\boldsymbol{\Omega}$**

$$\Omega(c) = \begin{cases} 0 & \text{if } c \text{ falls in a normal region} \\ 1 & \text{if } c \text{ falls in an anomalous region} \end{cases}$$

We assume that a **training set TR containing only normal images is given.**

s



Anomaly Detection in images

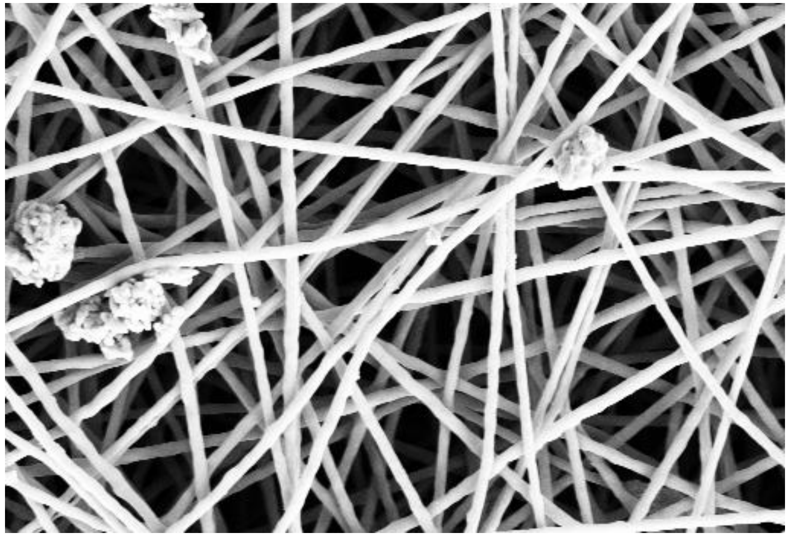
Let s be an image defined over the pixel domain $\mathcal{X} \subset \mathbb{Z}^2$, let $c \in \mathcal{X}$ be a pixel and $s(c)$ the corresponding intensity.

We want to **locate any anomalous region** in s, i.e. **estimating the anomaly mask $\boldsymbol{\Omega}$**

$$\Omega(c) = \begin{cases} 0 & \text{if } c \text{ falls in a normal region} \\ 1 & \text{if } c \text{ falls in an anomalous region} \end{cases}$$

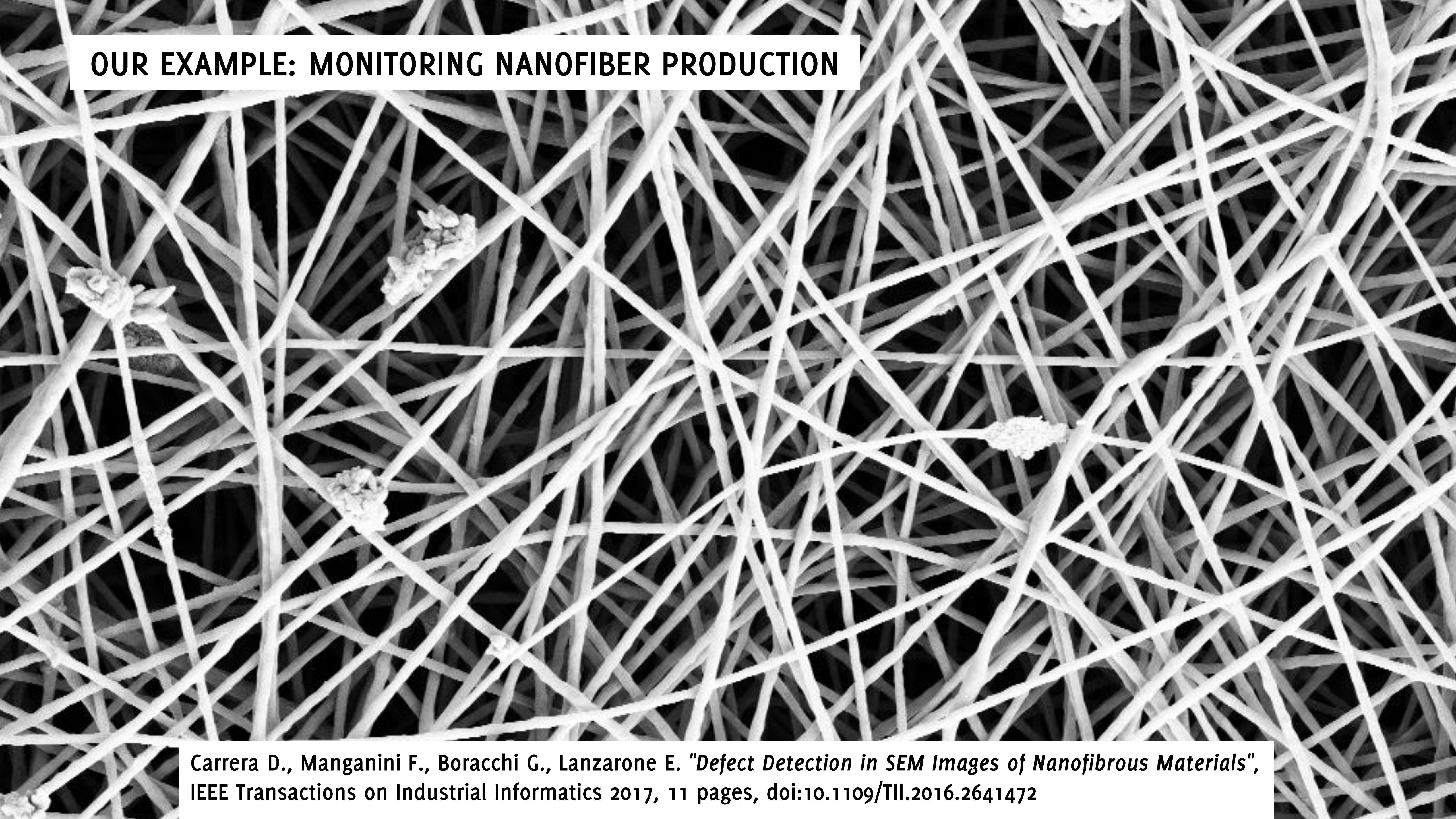
We assume that a **training set** TR **containing only normal images is given.**

s



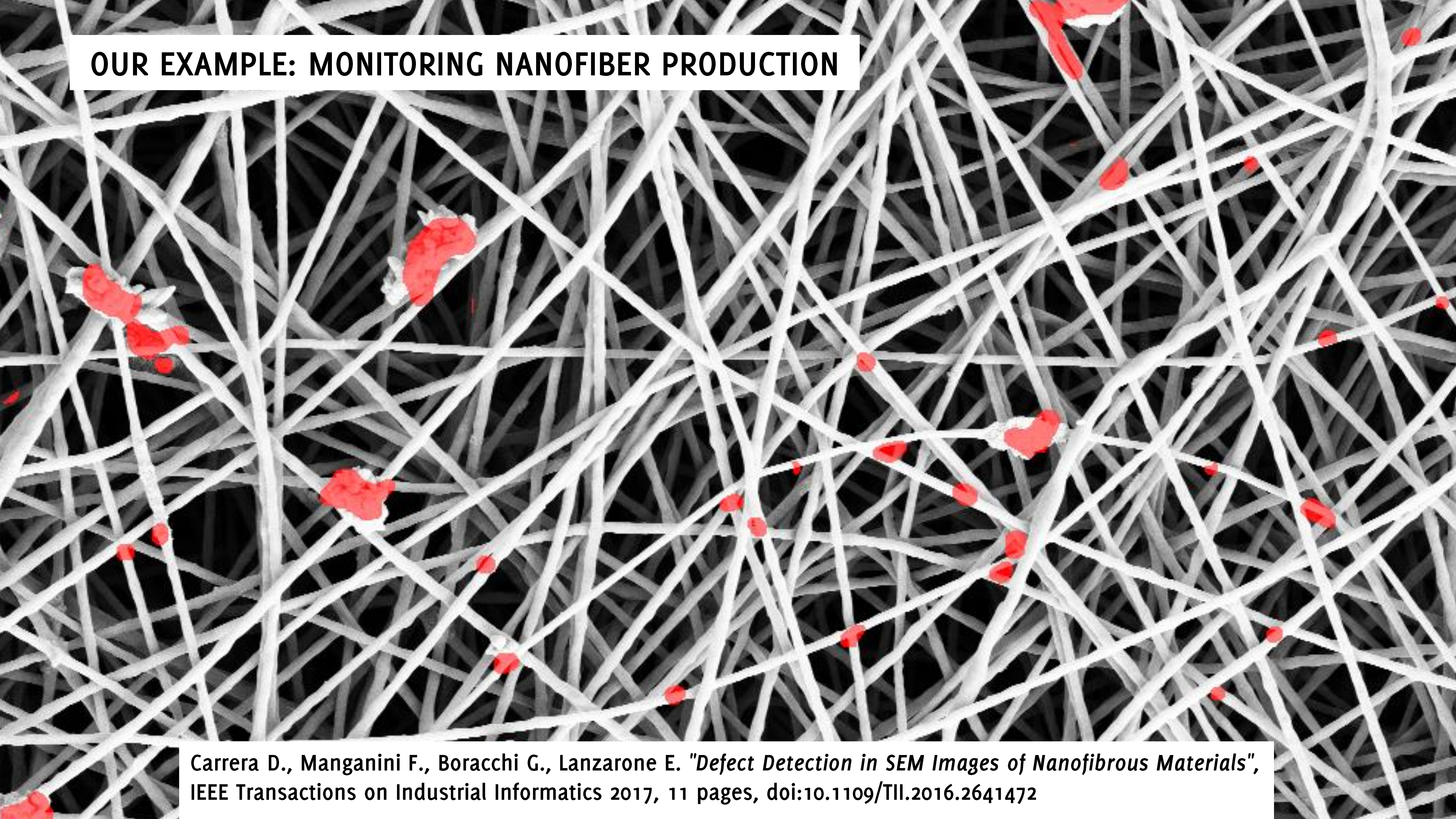
Ω

OUR EXAMPLE: MONITORING NANOFIBER PRODUCTION



Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

OUR EXAMPLE: MONITORING NANOFIBER PRODUCTION



Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

GANs and Anomaly Detection

A Generator $\mathcal{G}$ trained exclusively on normal images in TR, already **provides a mapping**

- From the space of random vectors $z \sim \phi_z$
- To the manifold where images live $s \sim \phi_s$

Thus, **if we could invert the GAN**, we would have already an AD model

An anomaly score for a test image s would be

$$s \rightarrow \mathcal{G}^{-1}(s) \rightarrow \phi_z\left(\mathcal{G}^{-1}(s)\right)$$

Unfortunately, it is not possible to invert $\mathcal{G}$… **some neural network need to be trained for this purpose!**

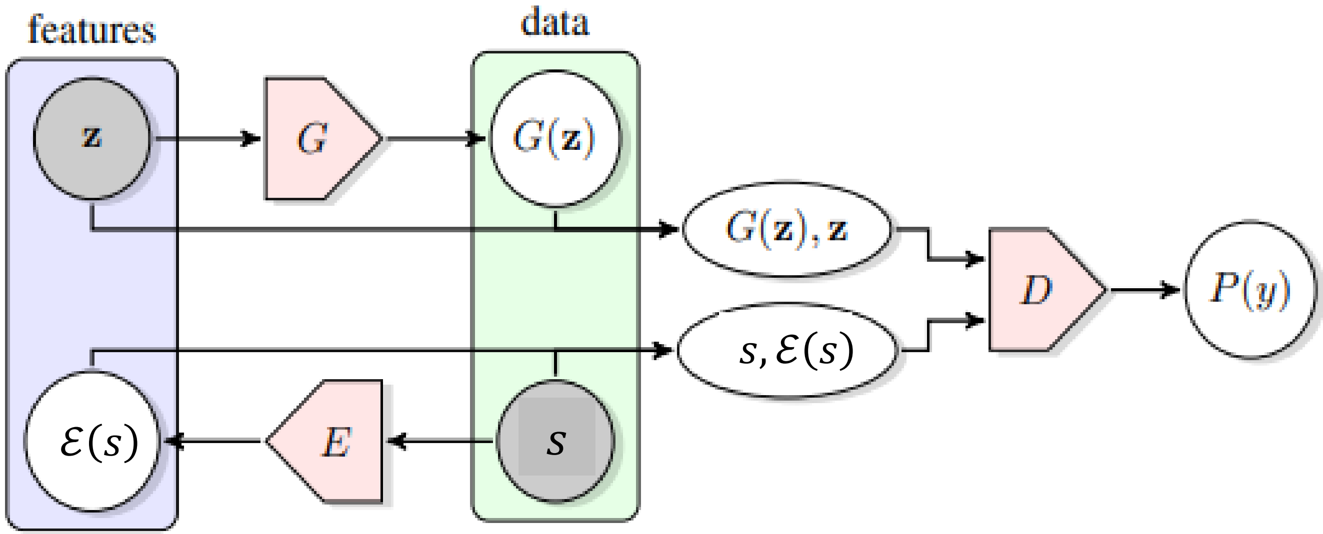
BidirectionalGANs (BiGANs)

The BiGAN adds an encoder $\mathcal{E}$ to the adversarial game which

- Brings an image back to the space of "noise vectors"
- Can be used to reconstruct an input image s (as in autoencoders) $\mathcal{G}(\mathcal{E}(s))$
- The discriminator $\mathcal{D}$ takes as input *(image, latent repr.)* as in conditional GAN

$$\min_{G,E} \max_{\mathcal{D}} V(\mathcal{D}, E, G)$$

$$V(\mathcal{D}, E, G) = \mathrm{E}_{s\sim\phi_S}[\log \mathcal{D}(s, E(s))] + \mathrm{E}_{z\sim\phi_z}[1 - \log \mathcal{D}(G(z), z))]$$

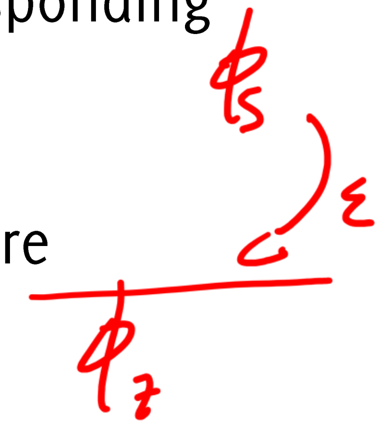


Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

BidirectionalGANs (BiGANs) and Anomaly Detection

In principle, the encoder $\mathcal{E}(\cdot)$ can be used for anomaly detection by computing the likelihood of $\phi_z(\mathcal{E}(\boldsymbol{s}))$ and consider as anomalous all the images $\boldsymbol{s}$ corresponding to a **low likelihood** (provided that ϕ_z was not a uniform distribution)

$$\phi_z(\mathcal{E}(\boldsymbol{s}))$$



Another option is to use the **posterior of the discriminator** as anomaly score

$$\mathcal{D}(\boldsymbol{s}, \mathcal{E}(\boldsymbol{s}))$$

since the discriminator will consider the anomalous sample as fake.

In principle, the encoder $\mathcal{E}(\cdot)$ can be used for anomaly detection by computing the likelihood of $\phi_z(\mathcal{E}(\boldsymbol{s}))$ and consider as anomalous all the images $\boldsymbol{s}$ corresponding to a **low likelihood** (provided that ϕ_z was not a uniform distribution)

$$\phi_z(\mathcal{E}(\boldsymbol{s}))$$

Another option is to use the **posterior of the discriminator** as anomaly score

$$\mathcal{D}(\boldsymbol{s}, \mathcal{E}(\boldsymbol{s}))$$

since the discriminator will consider the anomalous sample as fake.

$s, \mathcal{E}(s)$

$\mathcal{E}(s)$ s

Donahue, J., Krähenbühl, P., & Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782.*

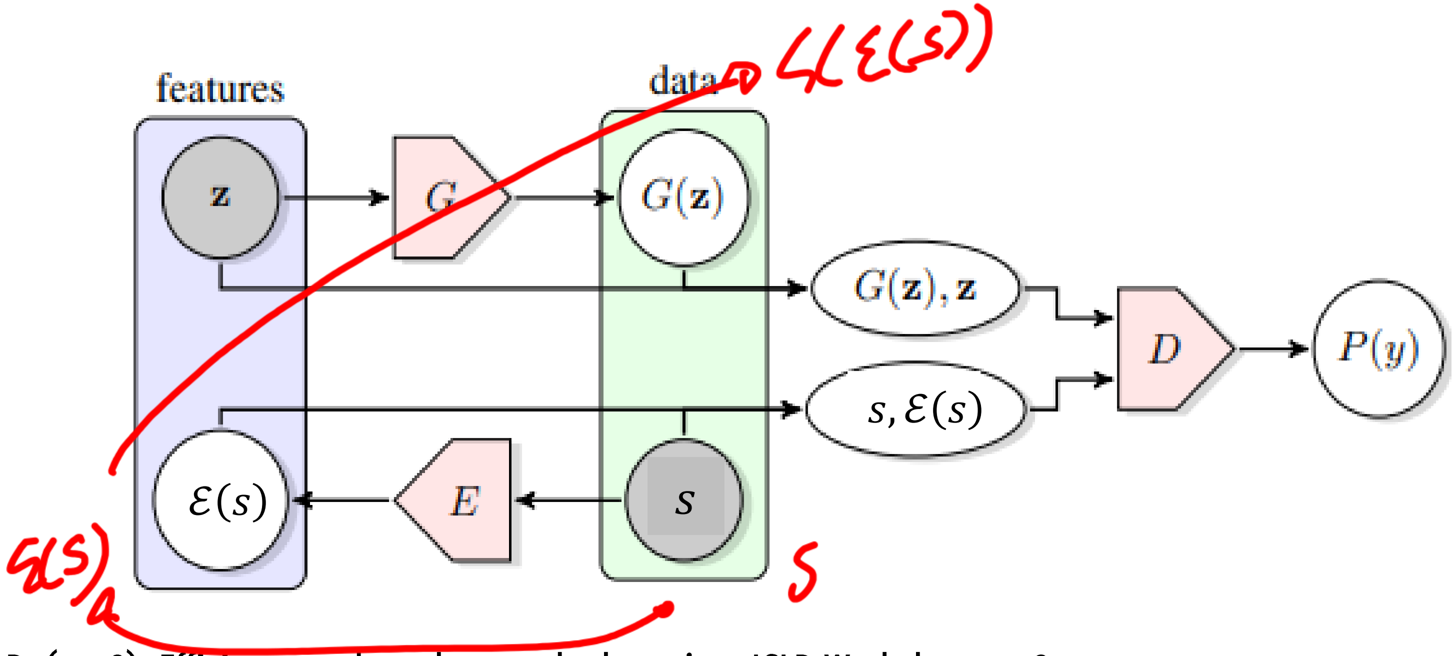
Anomaly detection with BidirectionalGANs (BiGANs)

However, there are more effective anomaly scores

$$A(s) = (1-\alpha)\left|\left|\mathcal{G}(\mathcal{E}(s)) - s\right|\right|_2 + \alpha\left|\left|f\left(\mathcal{D}(s,\mathcal{E}(s))\right) - f\left(\mathcal{D}\left(\mathcal{G}(\mathcal{E}(s)),\mathcal{E}(s)\right)\right)\right|\right|_2$$

Reconstruction Loss

Distance among latent representations of $\mathcal{D}$.
f is a CNN extracting a latent representation

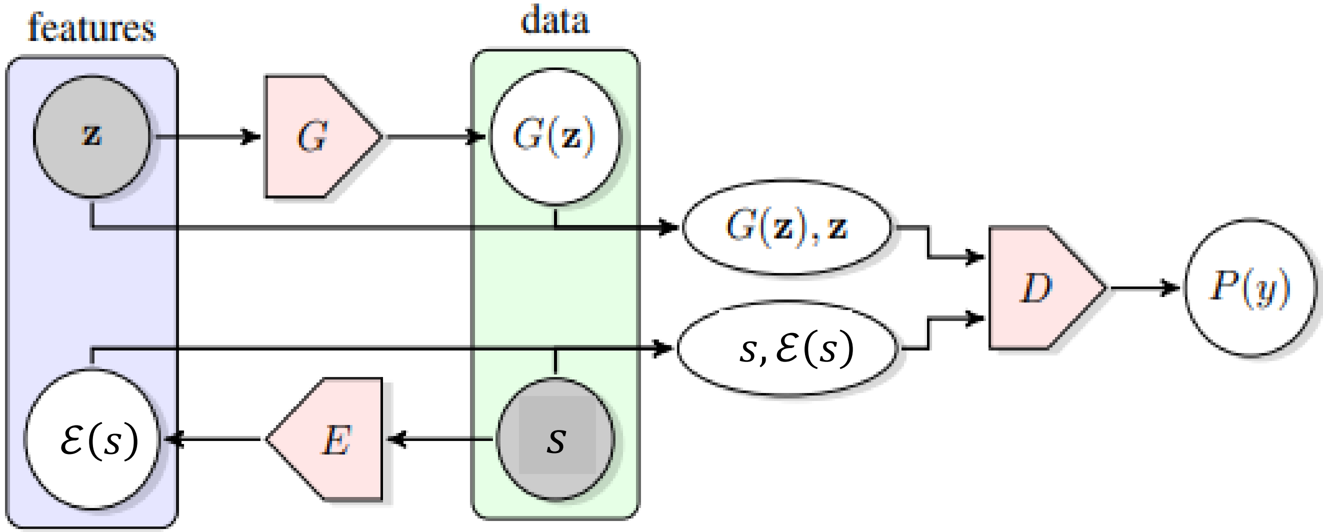


Zenati, H., Foo, C. S., Lecouat, B., Manek, G., & Chandrasekhar, V. R. (2018). Efficient gan-based anomaly detection. ICLR Workshop 2018

Anomaly detection with BidirectionalGANs (BiGANs)

Limitations

- Image-wise / patch-wise training and testing
- Little stability during training
- No way to promote better quality of reconstructed images

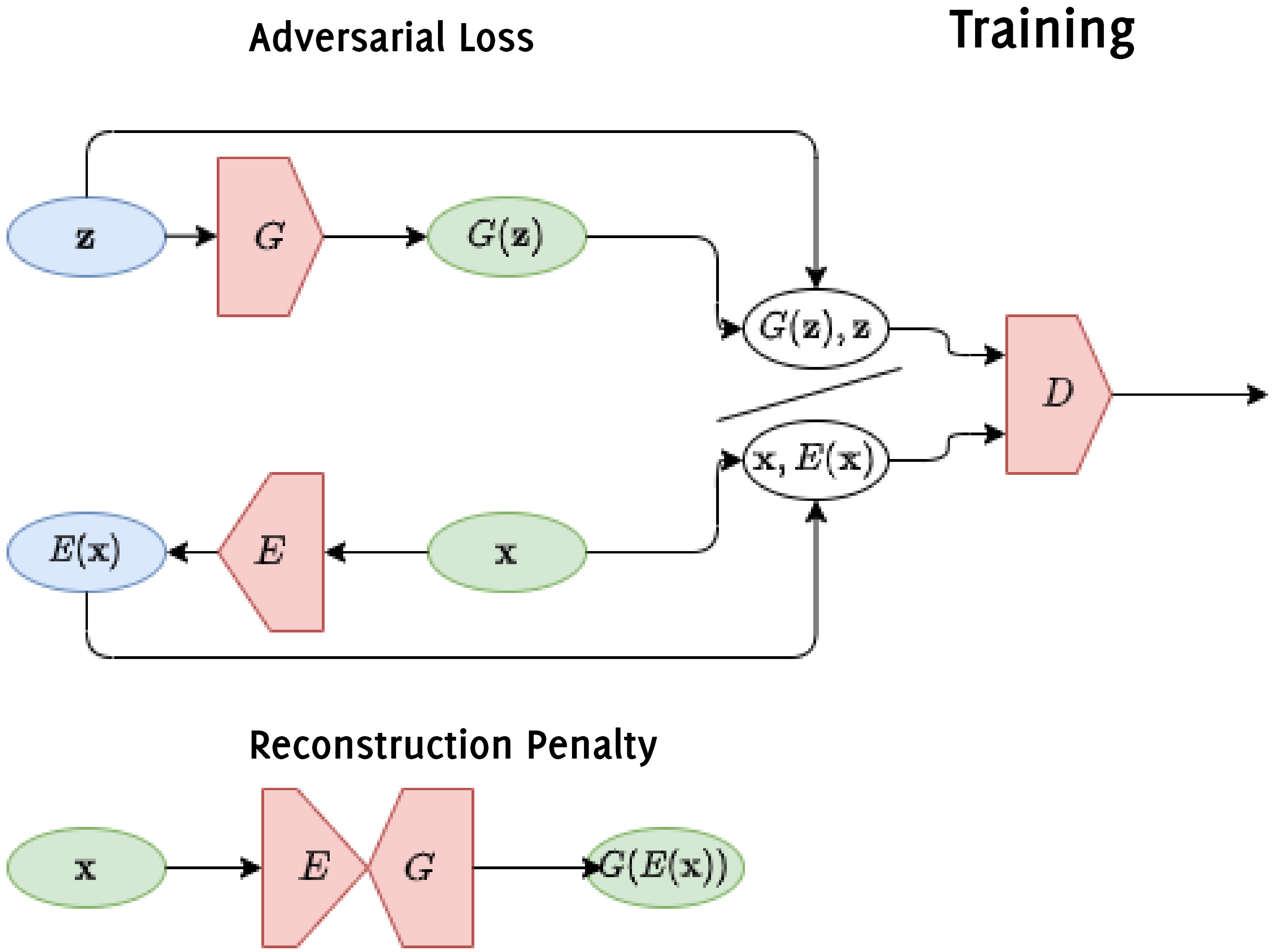


Zenati, H., Foo, C. S., Lecouat, B., Manek, G., & Chandrasekhar, V. R. (2018). Efficient gan-based anomaly detection. ICLR Workshop 2018

Fully Convolutional Anomaly Detection by GANs

Training

- All the layers are made fully convolutional (much more efficient processing)
- LS-losses used to train the model (this improves stability)



Credits Stefano Pecchia

Fully Convolutional Anomaly Detection by GANs

Inference

- **Anomaly score:** combines

I. image reconstruction error

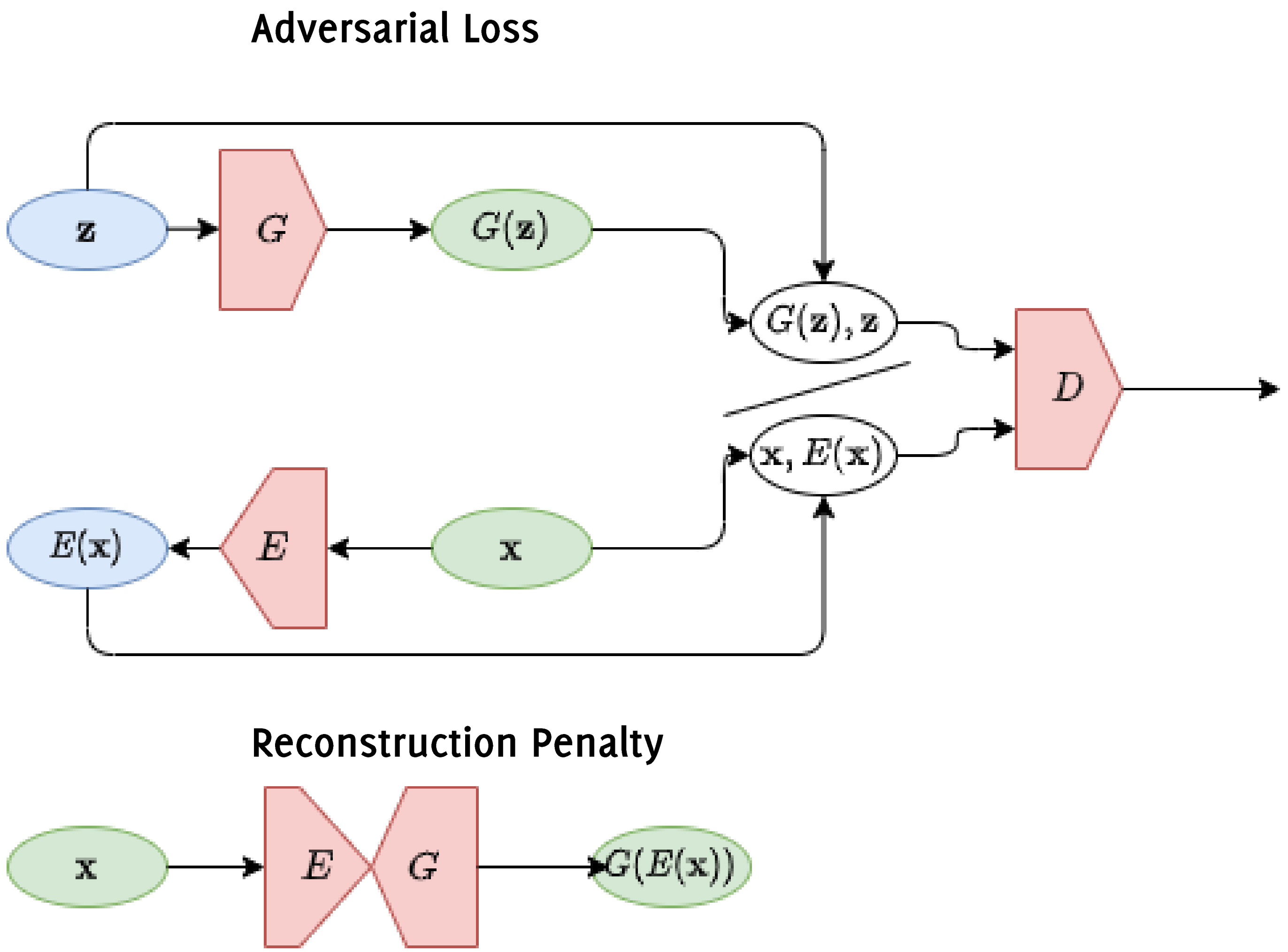
$$||\mathcal{G}(\mathcal{E}(\boldsymbol{s})) - \boldsymbol{s}||_2$$

II. discriminator loss

$$(\mathcal{D}(s, \mathcal{E}(s)) - 1)^2$$

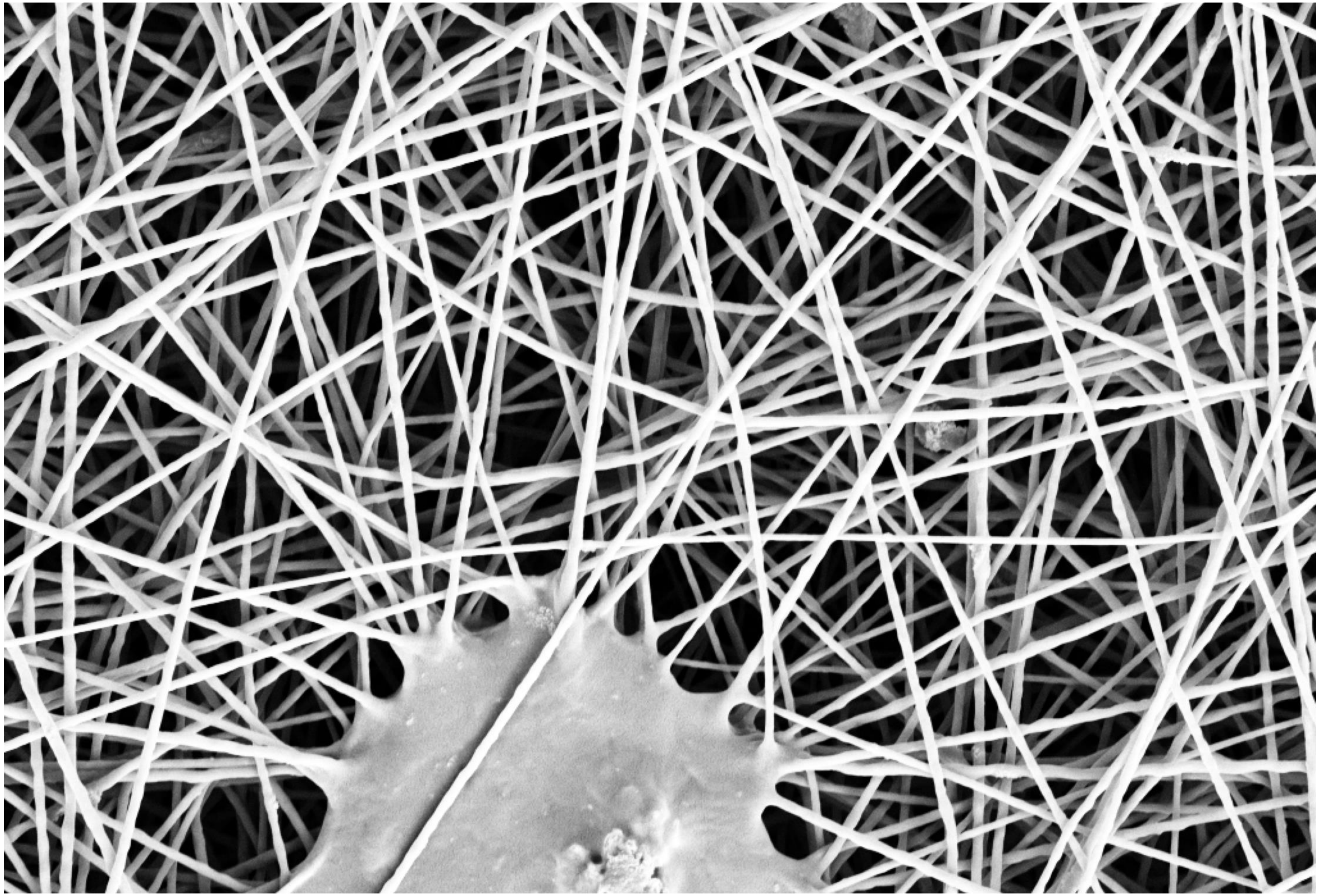
(since normal images are assumed to return 1)

- Possibly also likelihood w.r.t estimated distribution of $\mathcal{E}(s)$



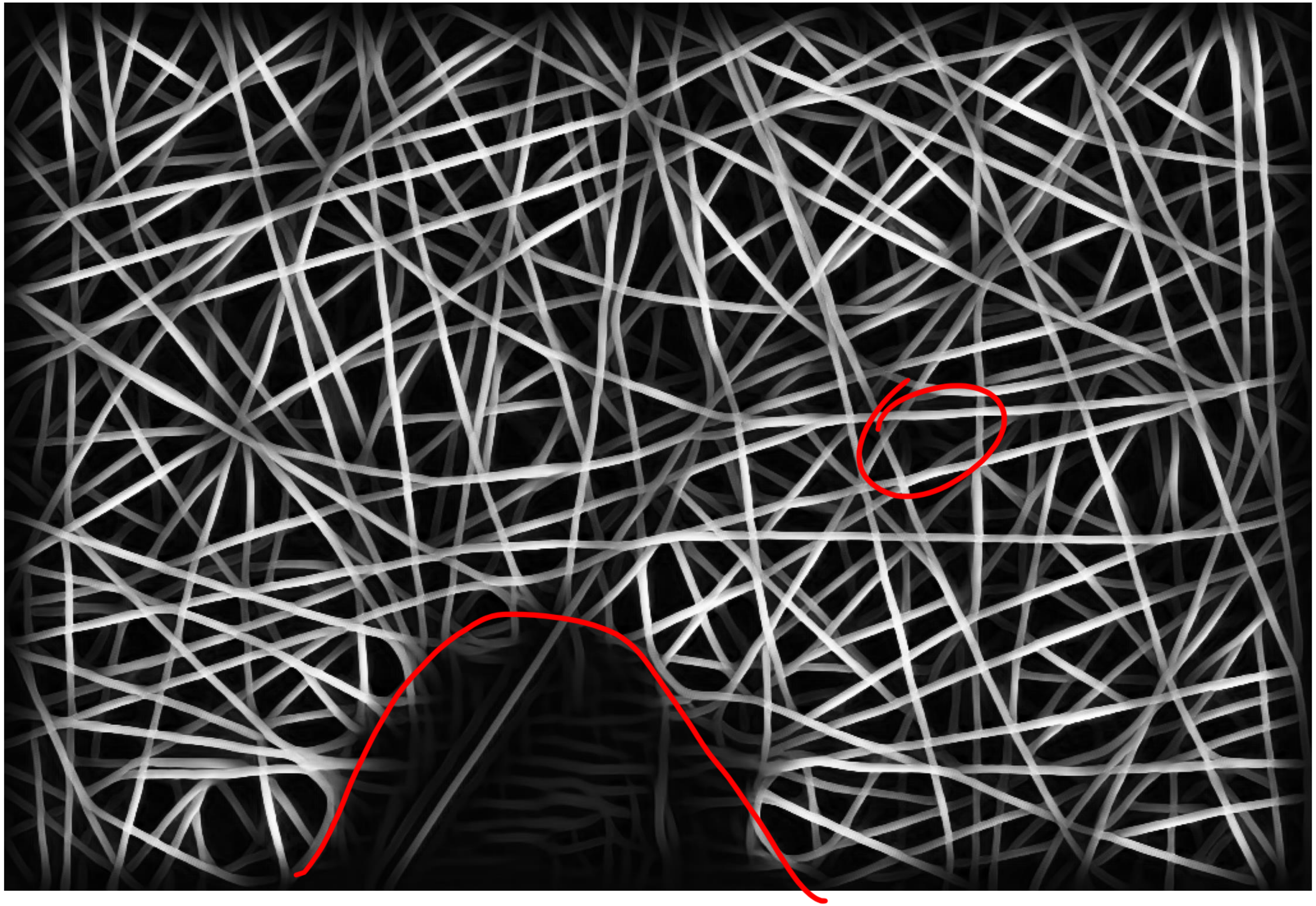
Credits Stefano Pecchia

Input Image



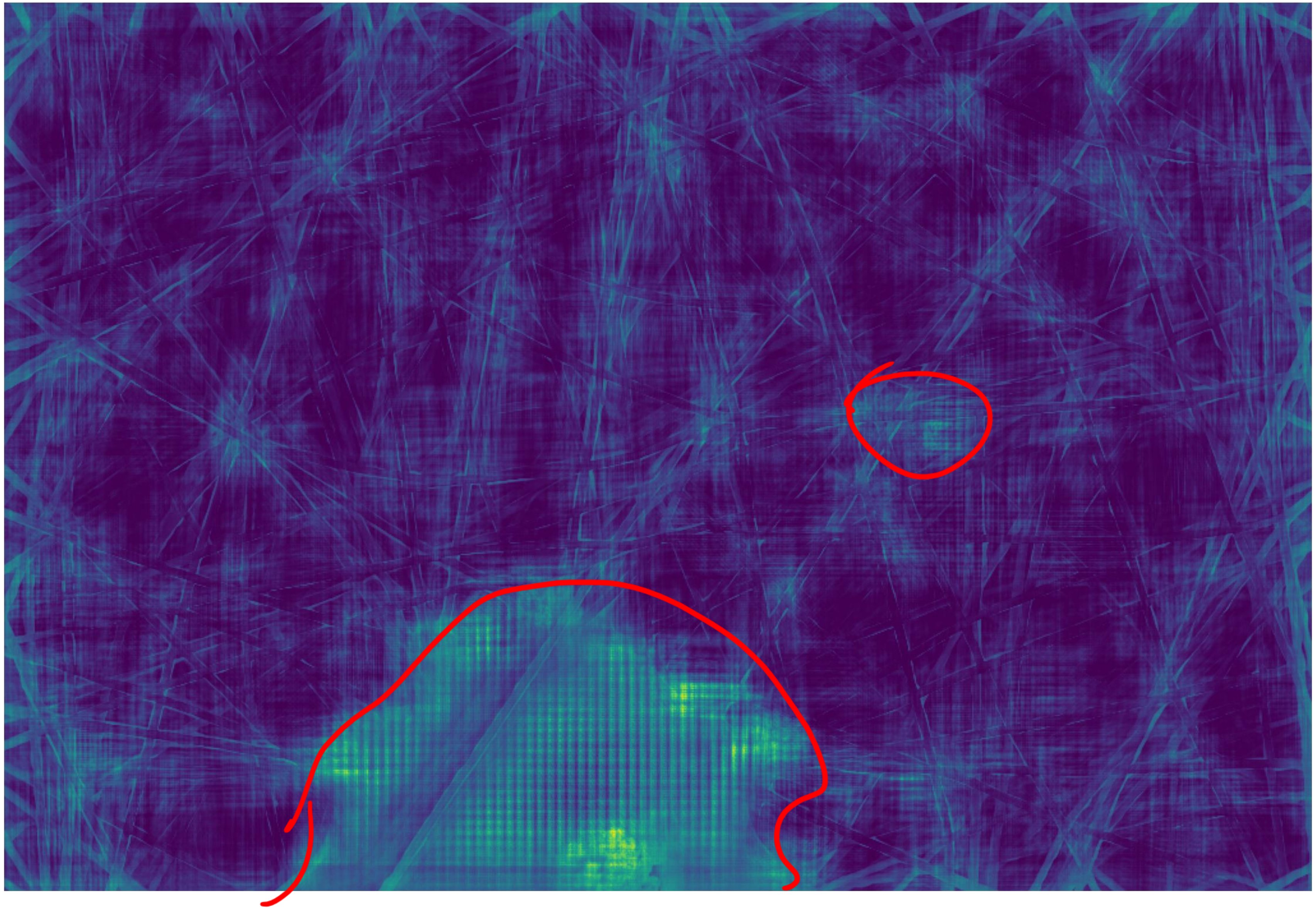
Credits Stefano Pecchia

Reconstruction $\mathcal{G}(\mathcal{E}(\boldsymbol{s}))$



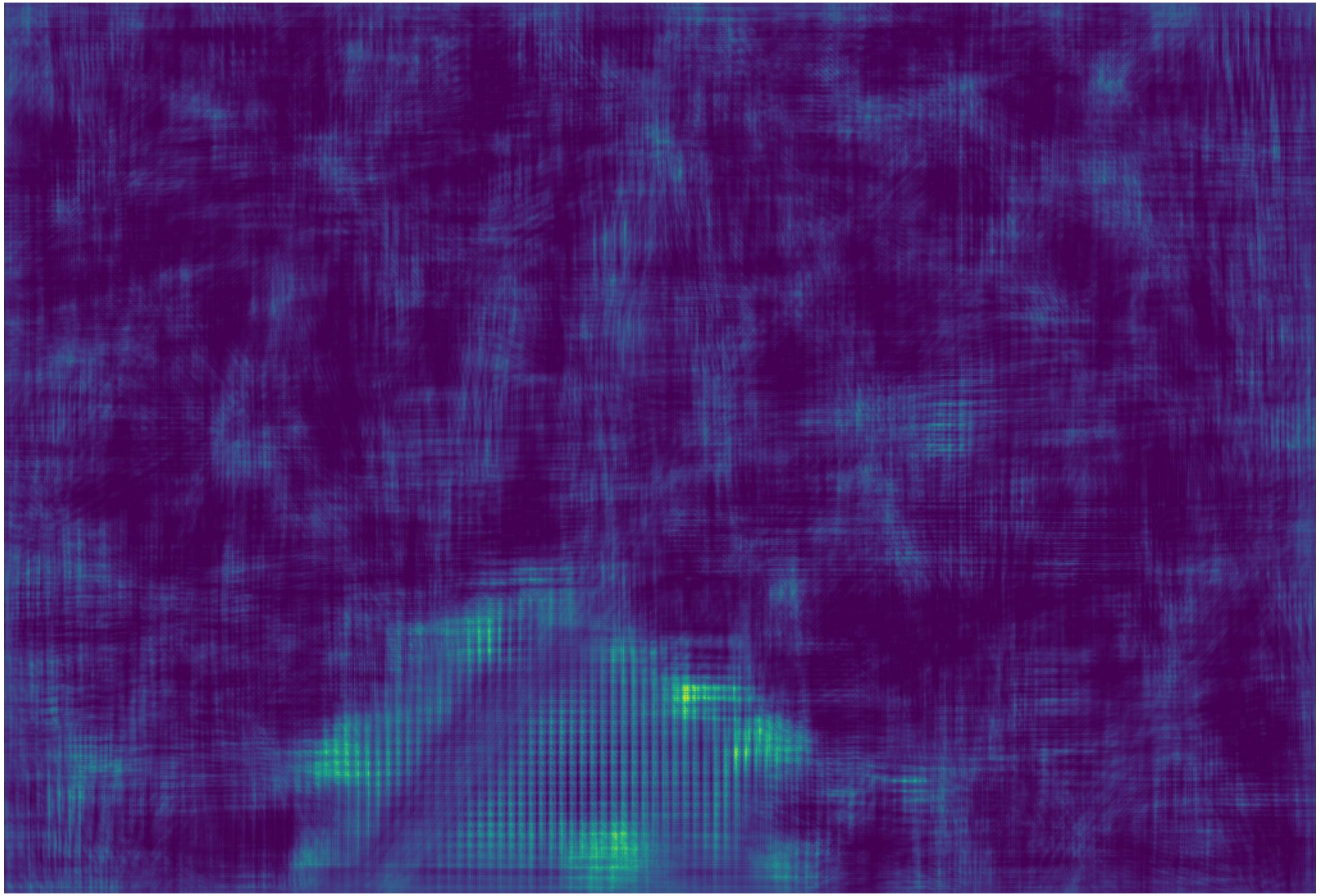
Credits Stefano Pecchia

Reconstruction Loss $||\mathcal{G}(\mathcal{E}(\boldsymbol{s})) - \boldsymbol{s}||_2$



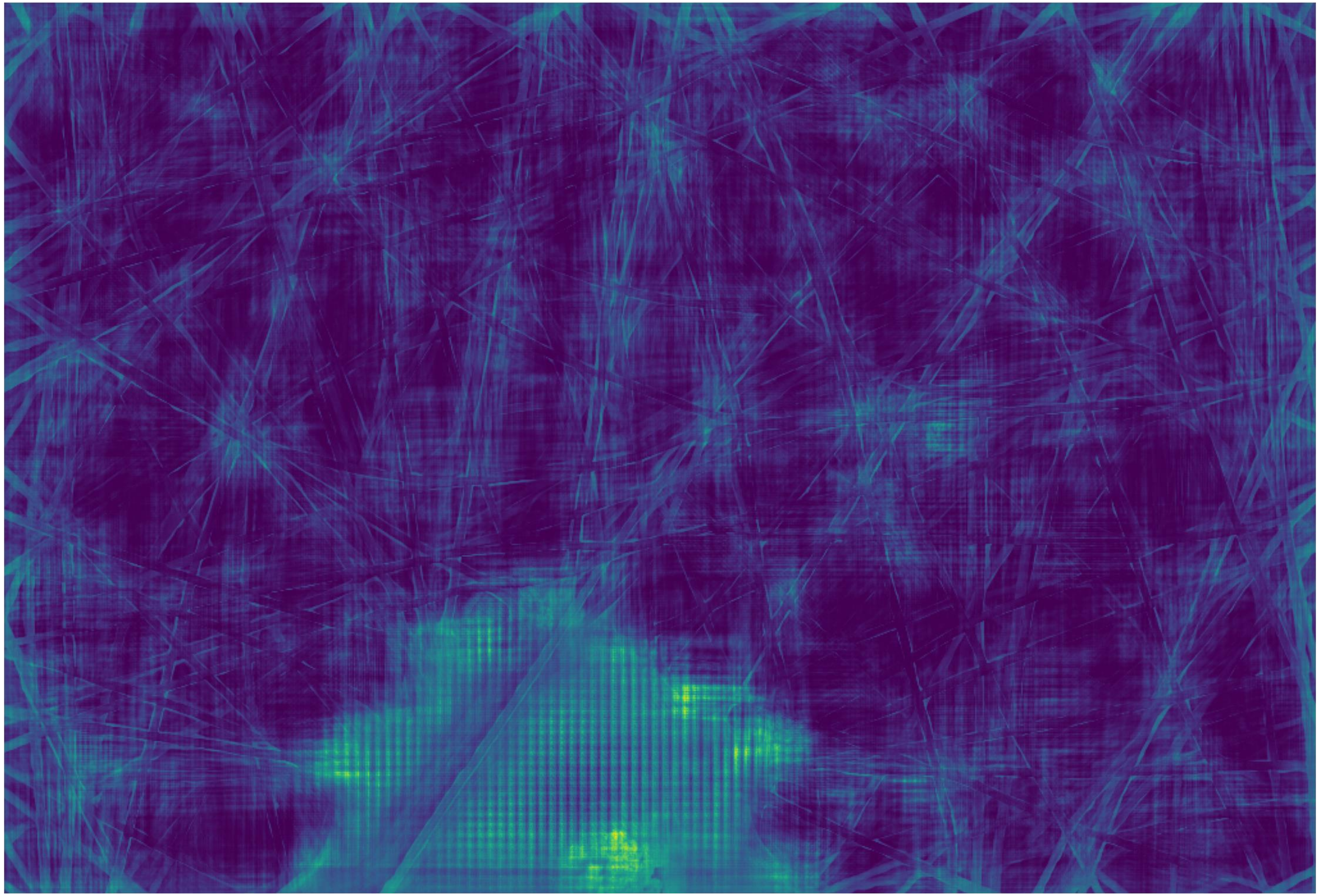
Credits Stefano Pecchia

Discriminator Score $\left(\mathcal{D}\left(s, \mathcal{E}(s)\right) - 1\right)^2$



Credits Stefano Pecchia

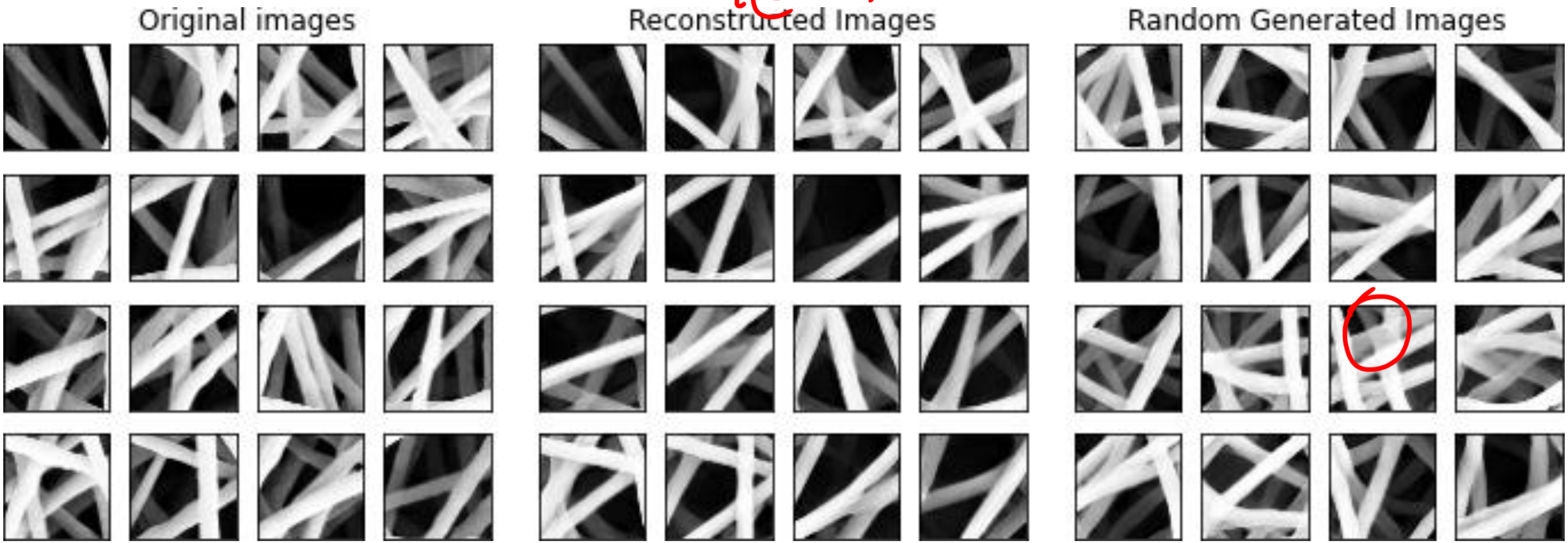
Anomaly Score $\alpha\left(\mathcal{D}\left(s, \mathcal{E}(s)\right) - 1\right)^2 + (1-\alpha)||\mathcal{G}\left(\mathcal{E}(\boldsymbol{s})\right) - \boldsymbol{s}||_2$



Credits Stefano Pecchia

Normal Image Generation By Our GAN

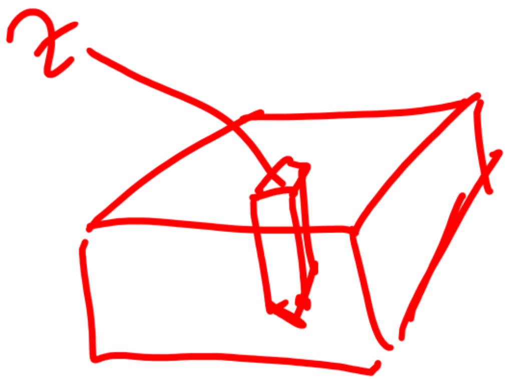
$z \in \mathbb{R}^{128}$

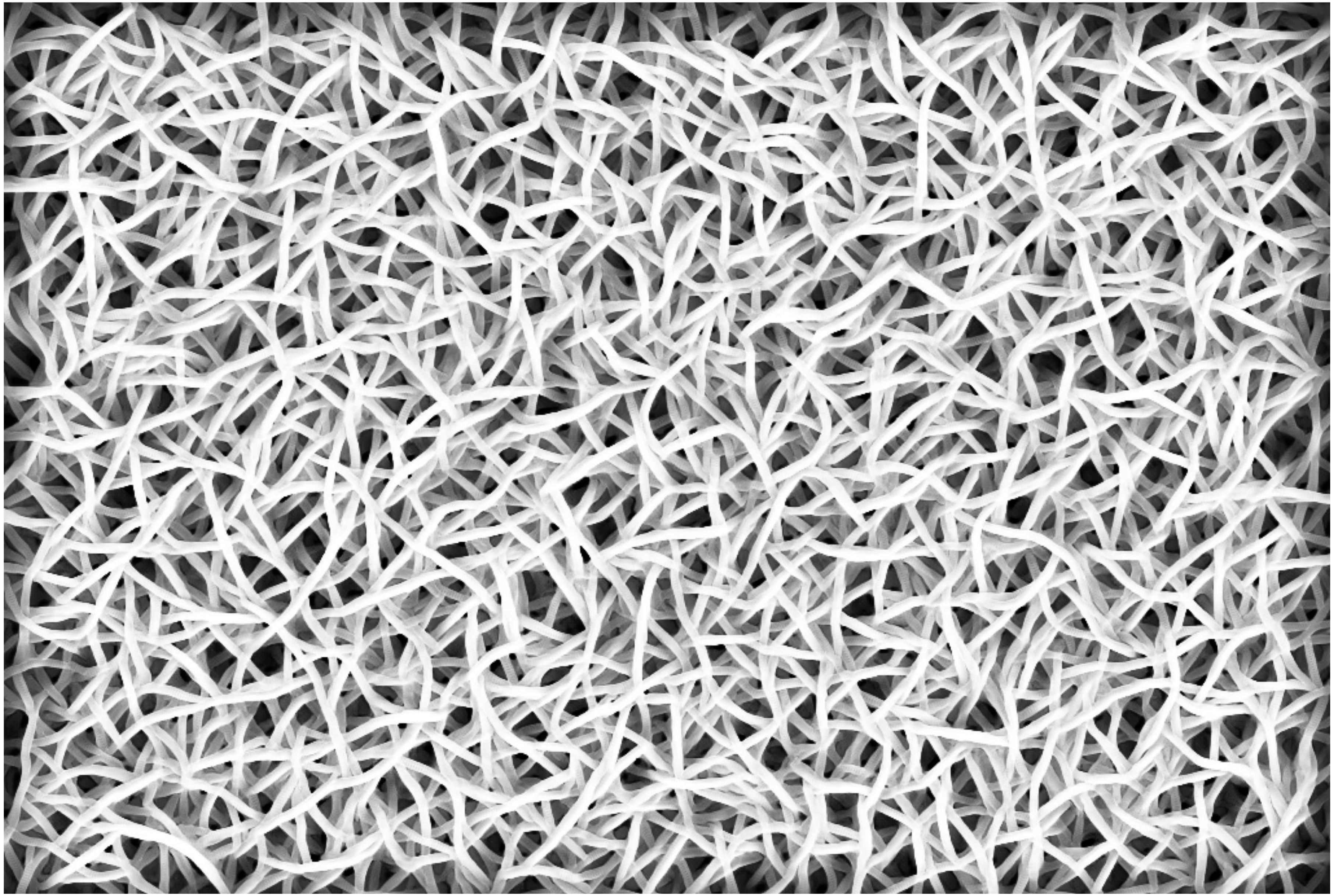


Credits Stefano Pecchia Thesis

It is a Genertive model!

$z \in \mathbb{R}^{80\times120\times128}$



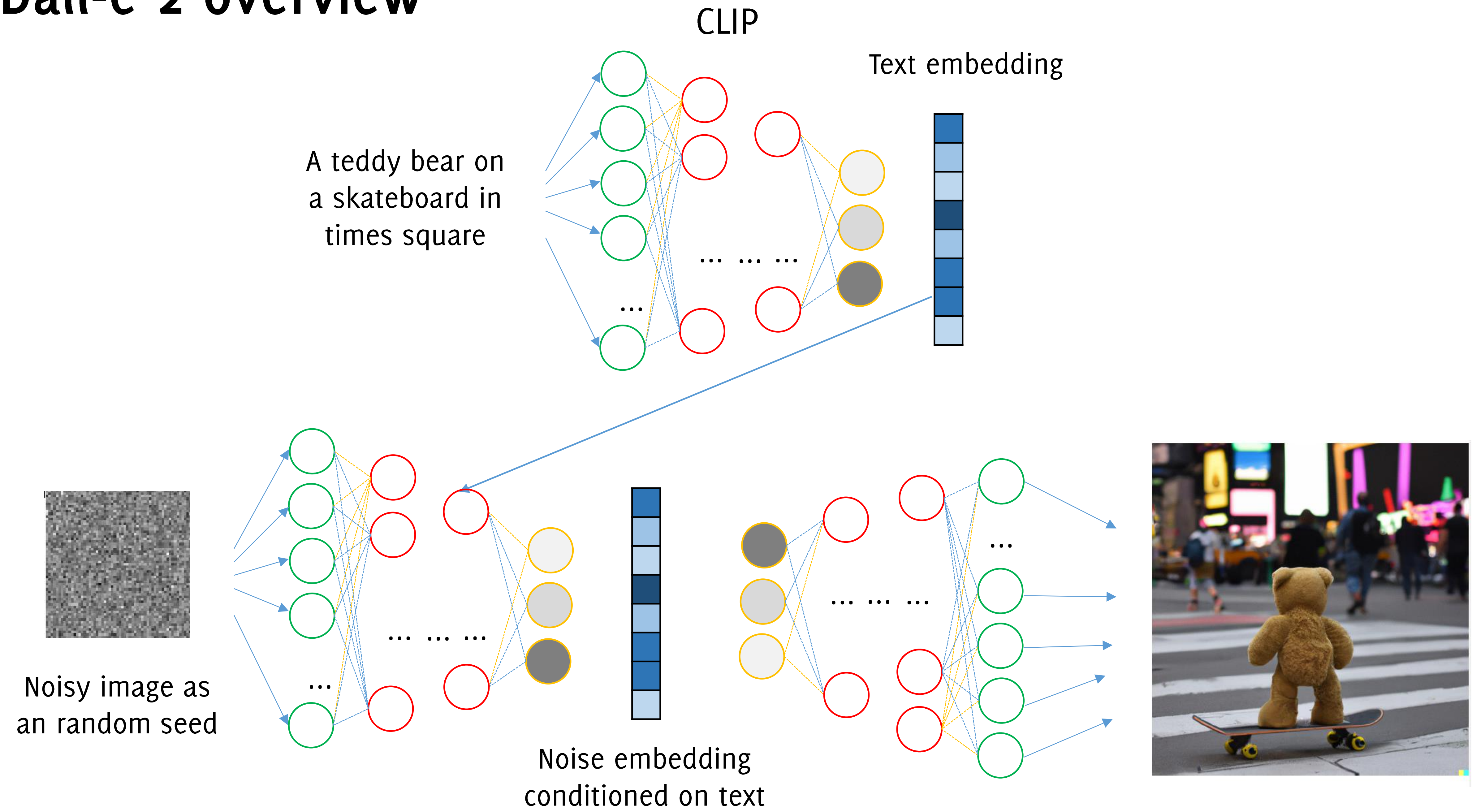


Local regions are well connected, but the GAN do not enforce a global image structure

Credits Stefano Pecchia

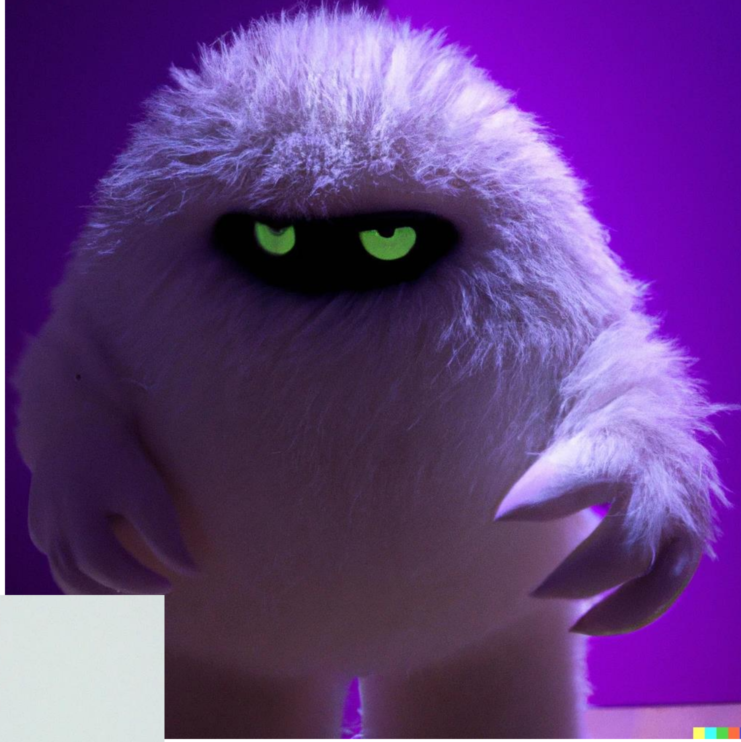
DALL-E2:
generate images from text description

Dall-e 2 overview



G. Boracchi

A photo of a white fur monster standing in a purple room



A van Gogh style painting of an American football player

A hand drawn sketch of a Porsche 911

A handpalm with a tree growing on top of it

https://openai.com/dall-e-2/

Generative Foundation Models

"Salmon in River"

Very poweful generators of never-seen contents

Slide Credits: Matteo Matteucci

Sora, 2024 Video Generator

nighttime footage of a hermit crab using an incandescent lightbulb as its shell

Photorealistic closeup video of two pirate ships battling each other as they sail inside a cup of coffee.

Several giant wooly mammoths approach treading through a snowy meadow, their long wooly fur lightly blows in the wind as they walk, [..]

https://openai.com/index/sora/

G. Boracchi

Video generation

Generation of complex and consistent motion among different entities

A large orange octopus is seen resting on the bottom of the ocean floor, [...] The octopus is unaware of a king crab that is crawling towards [...]

https://openai.com/index/sora/

Concluding Remarks on Image Generation

- Image generation was considered the «holy grail» of imaging research up to less than 10 years ago
- Different architectures of neural networks made this possible.
- Still, the **practical impact** of the first generative models was **kind of limited**.
- Text embedding and superior quality in image generation has lead to astonishing performance, opening **new perspective applications**
- Behind these models there is no black magic or «human-like feelings», but rather expert training from **a huge amount of data…** it is important to know how these work!

A Few Opportunities…

Option 1: Join the Team for a Thesis

The Team

We are 3 faculties, 10 PhD students, 1 Research Assistant… and 20+ MSc students!

Giacomo Boracchi

Luca Magri

Federica Arrigoni

Filippo Leveni

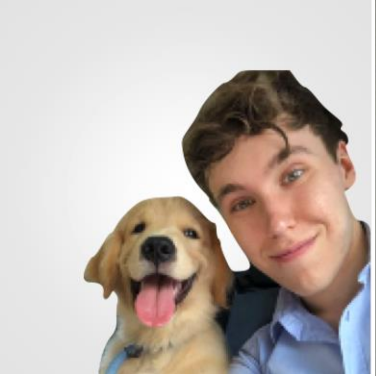
Loris Giulivi

Antonino Rizzo

Michele Craighero

Edoardo Peretti

Roberto Basla



Andrea Porfiri Dal Cin

Andrea Diecidue

Olmo Notarianni

Rakshith Madhavan

Research Collaborations

Major research collaborations:

Major research projects:

Thesis Information

- We typically illustrate thesis opportunities in February and September, typically during the first week of lectures.
- Thesis topics primarily concern Computer Vision, including both Deep Learning, Image processing and Geometric Computer Vision.
- Thesis are primarily research thesis, or thesis on industrial projects.
- Sometimes we open internship with companies we are collaborating with.
- We are always interested in brilliant candidates and perspective PhD students

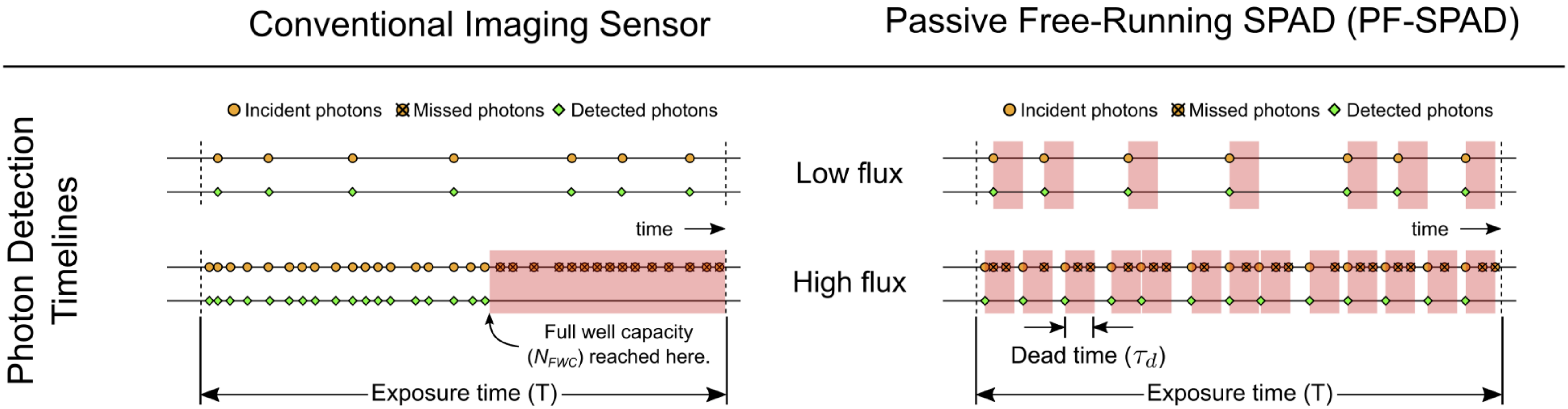
Thesis Information

- We have sent a proposal for **Honours Program in Research** (for those of you interested in research perspectives).

  http://www.honours-programme.deib.polimi.it/ (2025 call will probably open in January)

  Proposers from our team: Giacomo Boracchi, Luca Magri, Federica Arrigoni

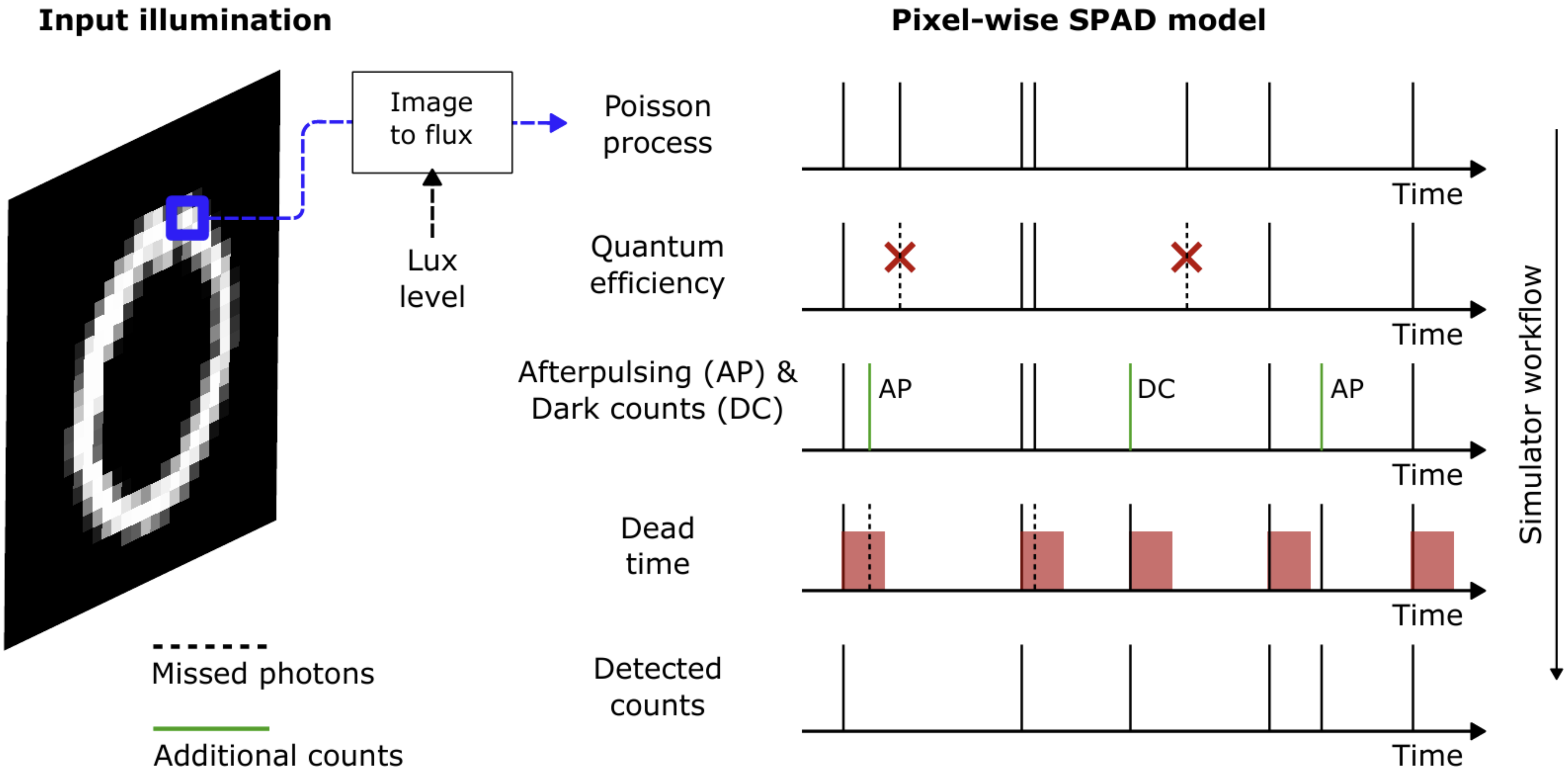
Probably next proposal will be on DL for TR-SPAD Imaging



A new imaging modality, requiring new image processing algorithms and probably….
new deep learning models!

Ingle, Atul, Andreas Velten, and Mohit Gupta. "*High flux passive imaging with single-photon sensors*" *CVPR 2019*

Our TR-SPAD Imaging Simulator



A. Suonsivu, L. Salmela, E. Peretti, L. Uosukainen, R. Ciprian Bilcu, and G. Boracchi, "*Time-Resolved MNIST Dataset for Single-Photon Recognition*", ECCV 2024 Workshop

Research Directions

Design of new:

- Image restoration algorithms for extremely low-light environments
- Deep learning models able to process streams of photons arrivals and address visual recognition *while the image is being acquired!*
- Parsimonious image acquisition procedures for high-flux conditoins….

- …if you want to know more on this, please drop us an email!

Option 2: Mathematical Models and Methods for Image Processing

Spring 2022, for Mathematical Engineering and Computer Science Engineering

What is this course about?

What is this course about?

*It is about **algorithms** for processing **images** and solving image-related problems.*

Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K. *"Image denoising by sparse 3-D transform-domain collaborative filtering"* TIP 2007

What is this course about?

*It is about **algorithms** for processing **images** and solving image-related problems.*

..like denoising

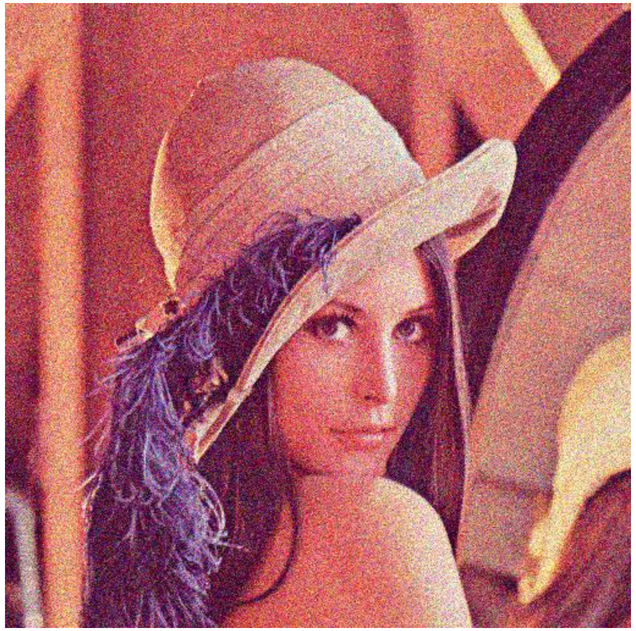
Dabov, K., Foi, A., Katkovnik, V., Egiazarian, K. *"Image denoising by sparse 3-D transform-domain collaborative filtering"* TIP 2007

Example of problems we will address here

Denoising

We will see algorithms solving problems customarily addressed in our phones,

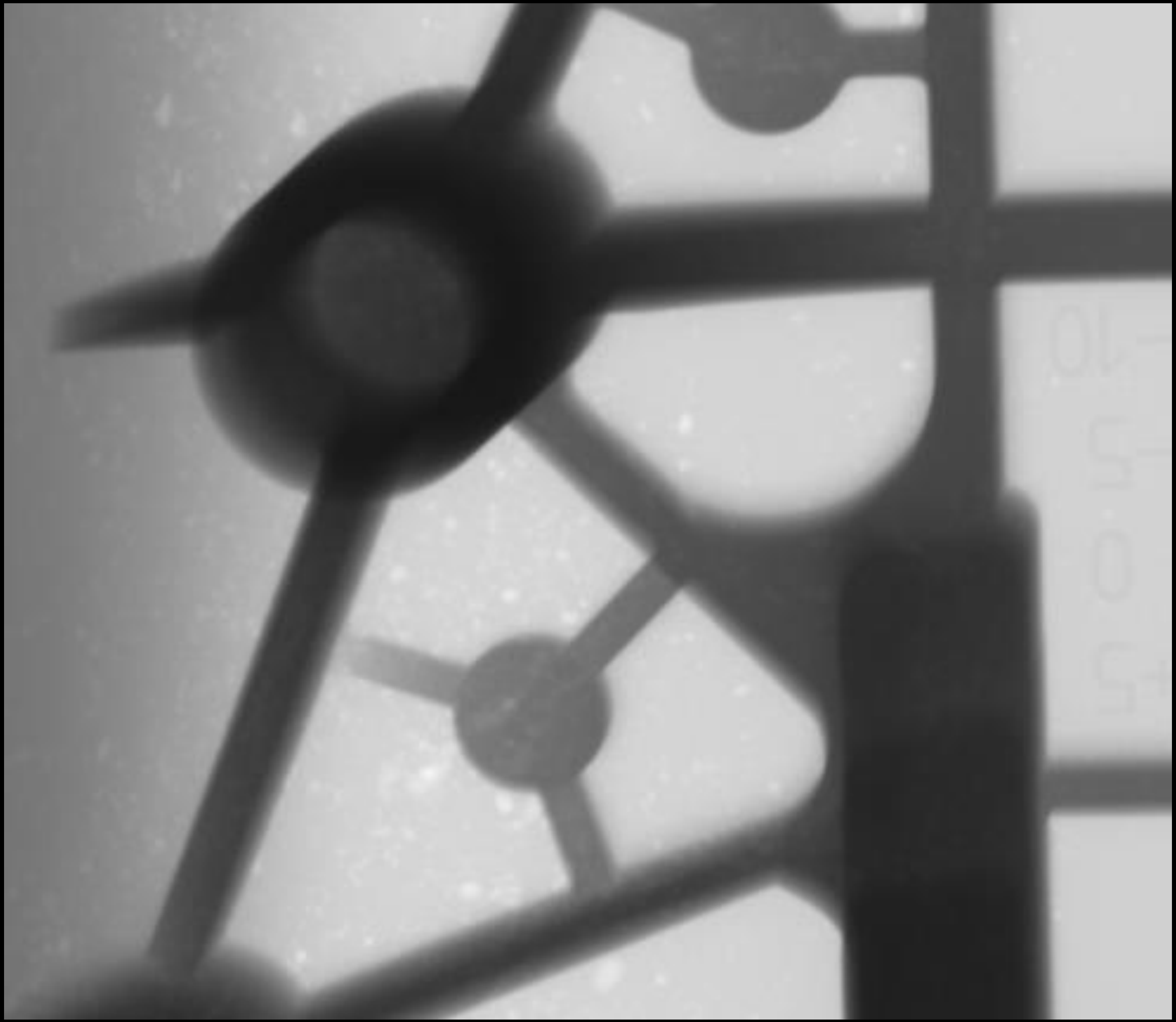
$z = y + \eta\,, \qquad \eta \sim \mathcal{N}(0, \sigma^2)$



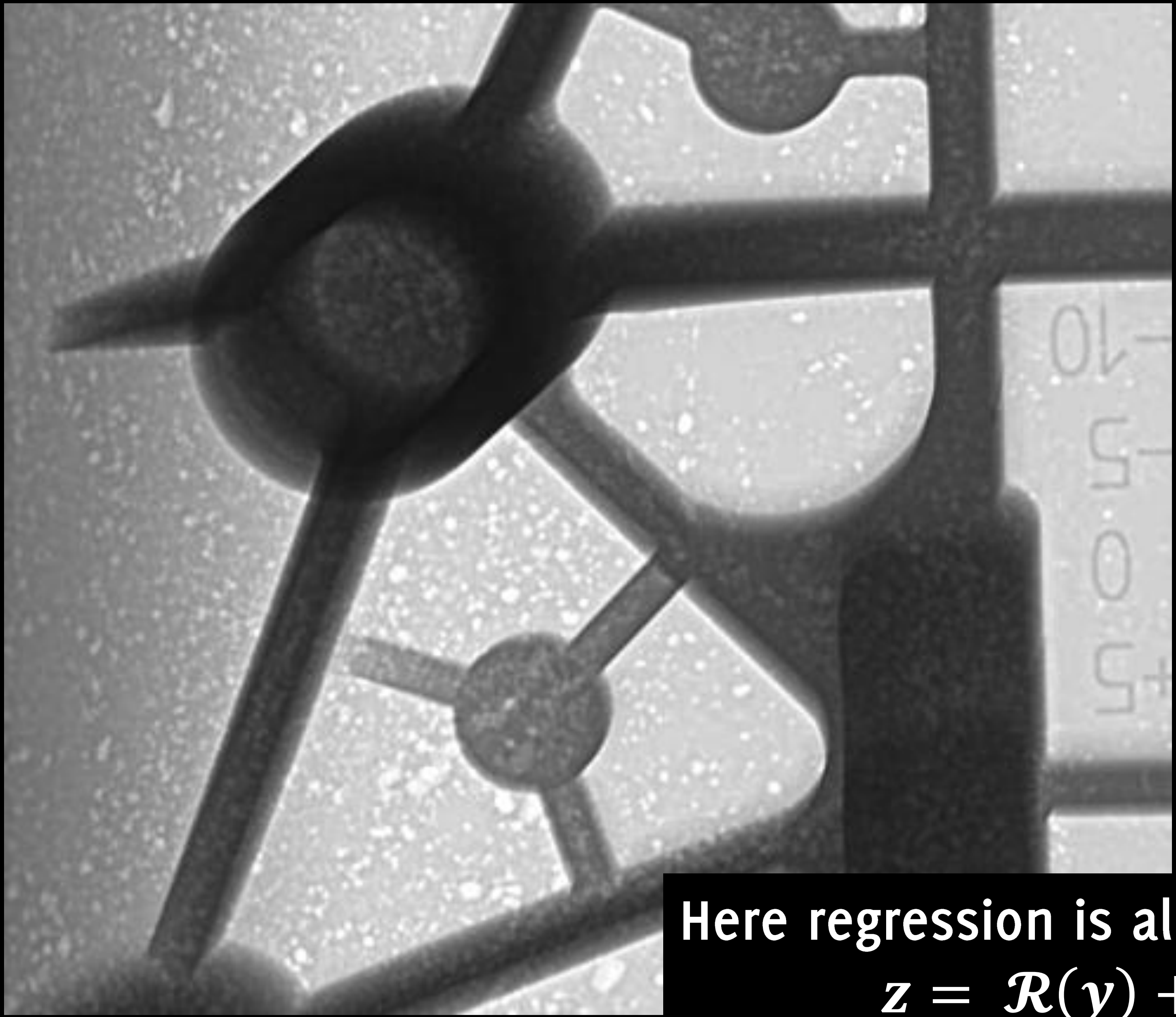
$\hat{y} \approx y$

Denoising is a regression problem: given the noisy z, estimate $\hat{y}$ close to the unknown y

Quality Inspection



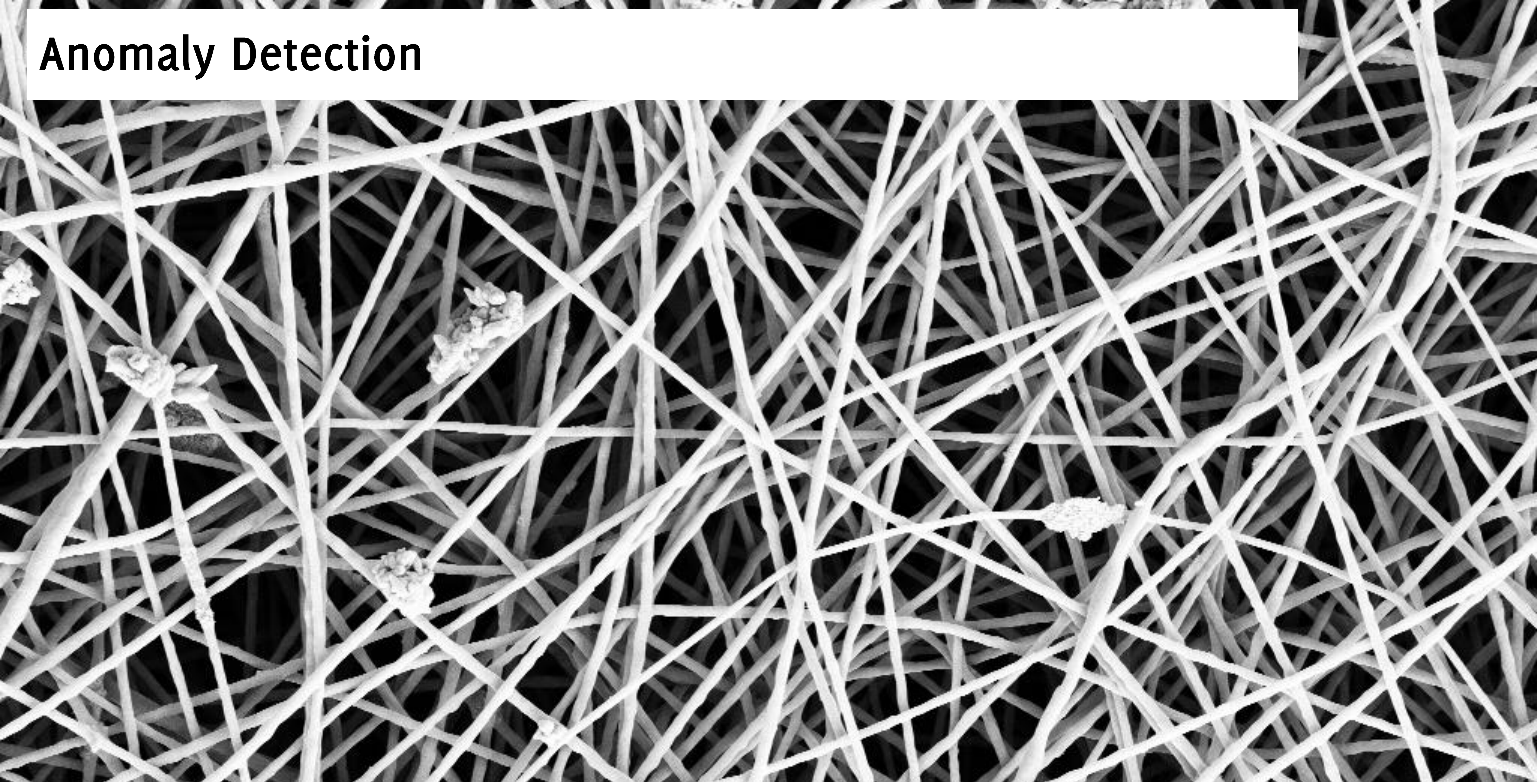
Quality Inspection



Here regression is also crucial

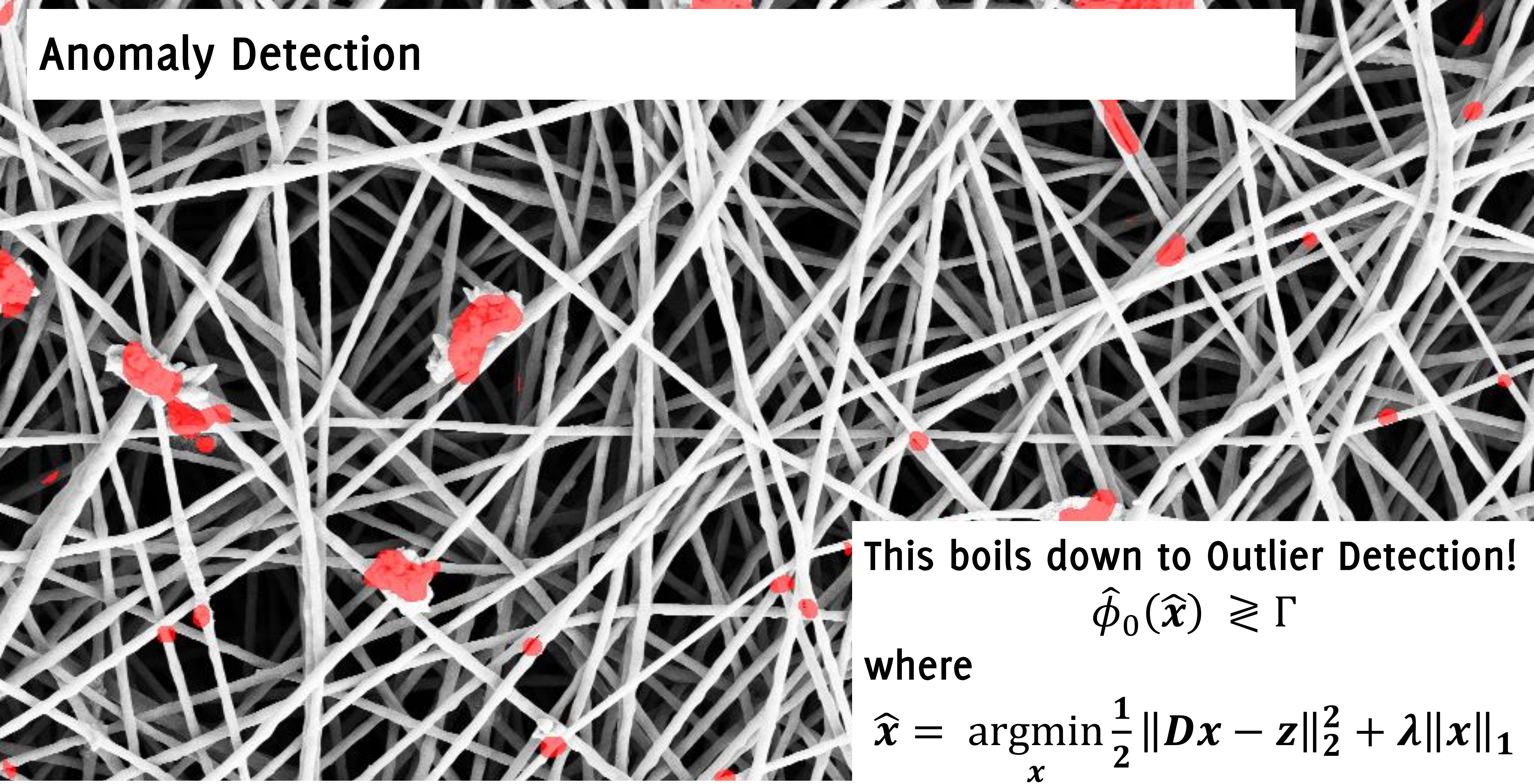
$$z = \mathcal{R}(y) + \eta$$

Anomaly Detection



Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

Anomaly Detection



This boils down to Outlier Detection!

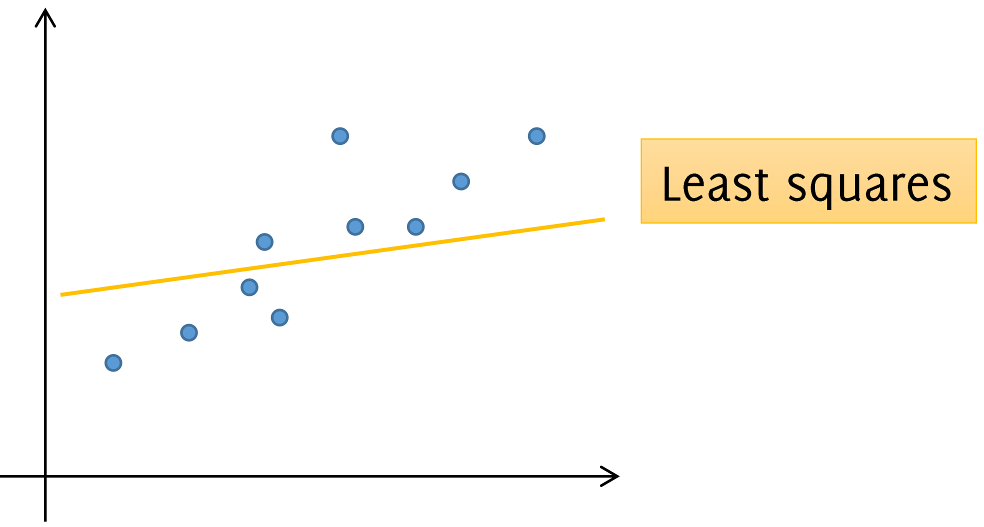
$$\hat{\phi}_0(\hat{\boldsymbol{x}}) \gtrless \Gamma$$

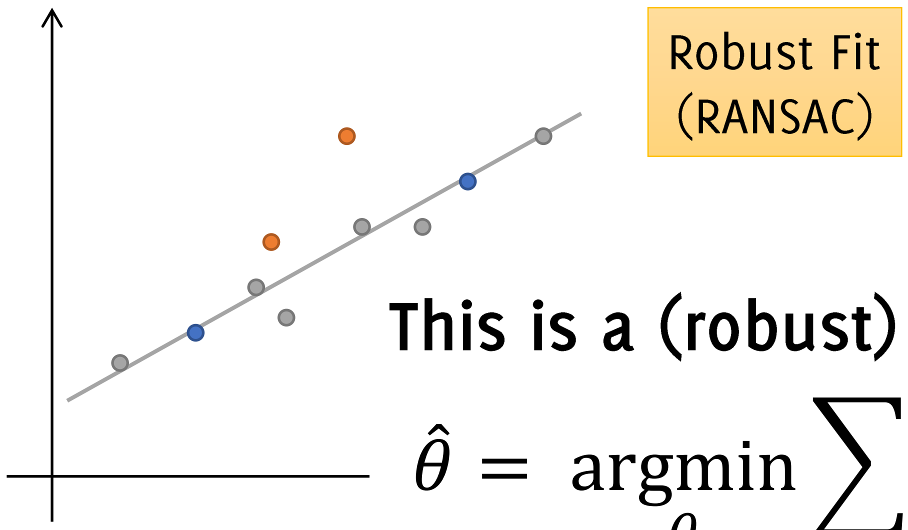
where

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \frac{1}{2} \|\boldsymbol{D}\boldsymbol{x} - \boldsymbol{z}\|_2^2 + \boldsymbol{\lambda}\|\boldsymbol{x}\|_1$$

Carrera D., Manganini F., Boracchi G., Lanzarone E. *"Defect Detection in SEM Images of Nanofibrous Materials"*, IEEE Transactions on Industrial Informatics 2017, 11 pages, doi:10.1109/TII.2016.2641472

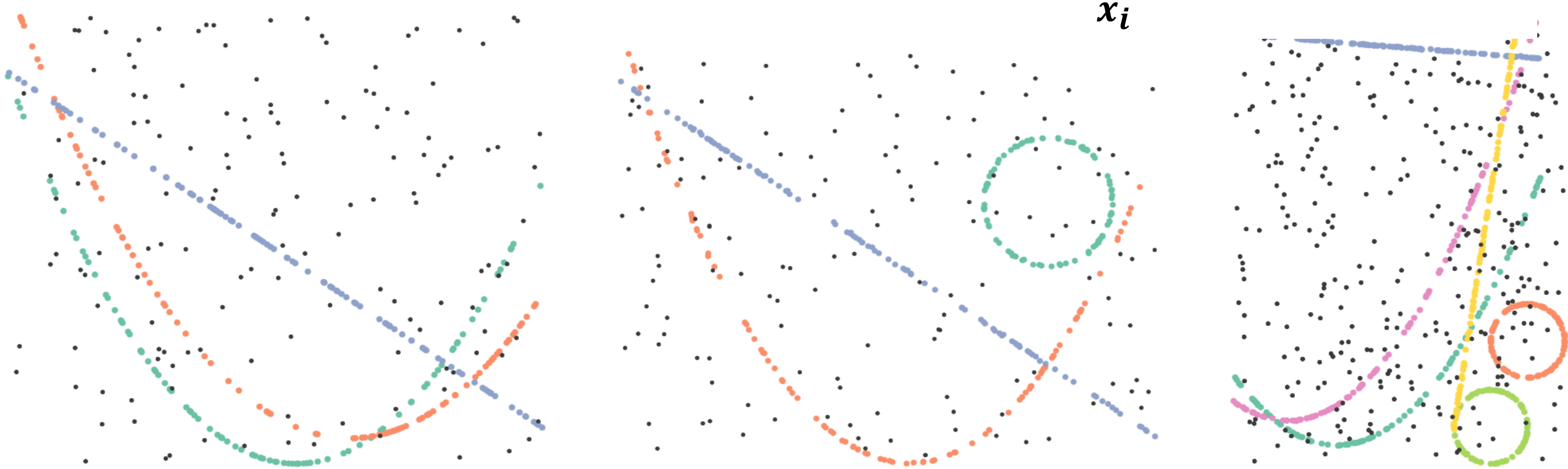
Robust Fitting



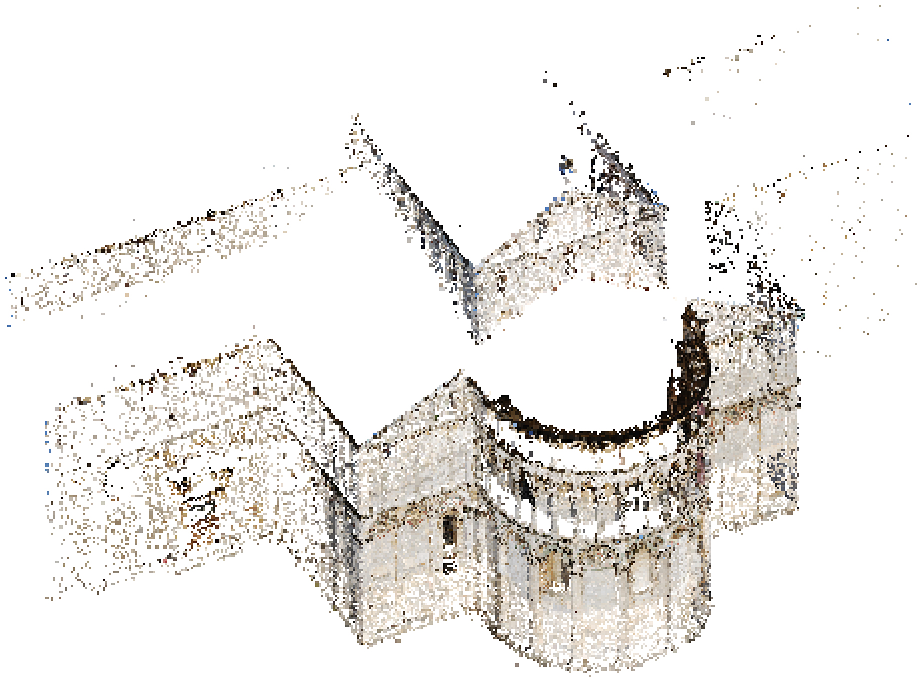


This is a (robust) fitting problem

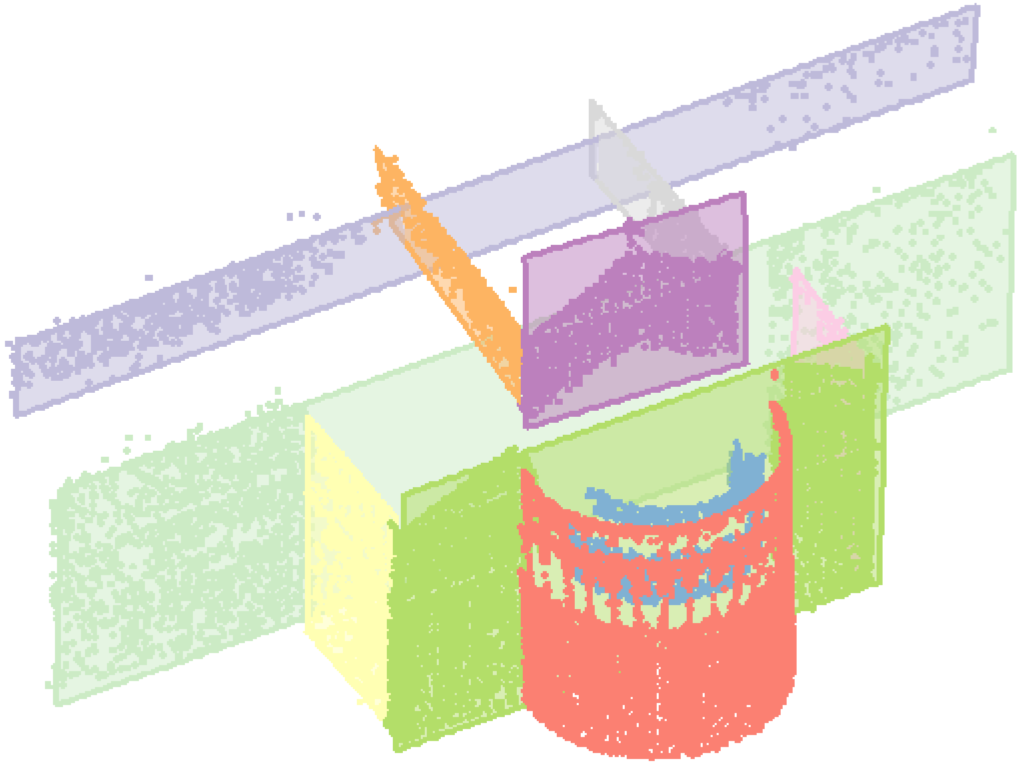
$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \sum_{\boldsymbol{x_i}} \rho(\operatorname{dist}(\boldsymbol{x_i}, \mathcal{M}_\theta))$$



Robust Fitting



(a) Input point cloud



(b) Recovered structures

This is a (robust) fitting problem

Magri, Leveni, Boracchi "MultiLink: Multi-class Structure Recovery via Agglomerative Clustering and Model Selection", CVPR 2021

Robust Fitting

Robust Fitting

This is a (robust) fitting problem

Is this interesting for a (perspective) Mathematical / Computer Science Engineer?

Is this interesting? Sure!

All the algorithms build upon:

- a **clear problem formulation**
- a **simple mathematical model (…often linear combinations!)**
- Sound mathematical solutions (linear algebra, least squares, convex optimization)

…and the result is not just a number… it's an image!

Ok, to recap

Mathematical Models and Methods for Image Processing (5 CFU)

*The primary **goal** of this laboratory course is to **let the students design, implement and practice algorithms** based **on***

simple mathematical models** from **linear algebra and convex optimization,

*and **solve** challenging inverse **problems in image processing** (denoising, deblurring, inpainting, anomaly detection)*

Mathematical Models and Methods for Image Processing (5 CFU)

The course **topics include**:

- **Image models based on orthonormal bases** (Fourier, wavelets), **data-driven basis** (PCA, Gram-Schmidt) and **local polynomial approximation**.

- **Sparsity and redundancy**.
    - Away from Orthonormal Basis, redundant set of generators
    - Sparse coding with ℓ^0 (OMP) or ℓ^1 norm (convex optimization ISTA, IRLS, LASSO)
    - Dictionaries yielding sparse representations and dictionary learning (KSVD)
- **Applications of sparse models** to image denoising, inpainting, anomaly detection and classification.

- **Robust fitting** methods (RANSAC, LMEDS, HOUGH) and their sequential counterparts for object detection in images.

Course Organization

Lectures: 20 hours

Laboratory: 30 hours

There will be short theory recap and then you will be invited to develop and practice presented algorithms. Some demo code to fill in will be provided.

Simple assignment provided during lectures, oral exam.

Frequently Asked Questions

Q: Any specific background?

A: linear algebra, statistics and calculus

Q: Any programming skill required?

A: Proficiency in Matlab or Python

Q: Plenty of neural networks then?

A: No way. No neural networks allowed here * ☺
Only expert-driven algorithms designed upon a clear mathematical modeling that admits closed-form solutions / sound optimization schemes.

* Interested in neural networks? Refer to «Artificial Neural Network and Deep Learning» in the first semester

Questions?

Denoising over adaptively defined neighborhoods for local polynomial regression

LASIP c/o Tampere University of Technology http://www.cs.tut.fi/~lasip/

Option 3: Advanced Deep Learning PhD Courses

Every year we offer a PhD course

ADVANCED DEEP LEARNING

A PhD course from Prof. Boracchi, Matteucci, Mentasti, Papini

Advanced Deep Learning course aims at ***exploring two major directions*** *in deep learning to provide an advanced ground to engineers aiming at up-to-date deep learning expertise that goes beyond a master-level course in deep learning:*

- ***Advanced Deep Learning Architectures****, such as Graph Neural Networks, Point Convolutional Networks, and Transformers, have recently introduced a breakthrough in DL research*
- ***Learning non-conventional tasks*** *(image generation with and without text conditioning) and from* **limited supervision** *(e.g., unsupervised / self-supervised / zero-shot learning). In particular, we will describe the mainstream models for generating images with d Diffusion models.*

More info here: link

ADVANCED DEEP LEARNING

The following program will be covered via the ***six half days of in-presence lectures***

Course Introduction: *a historical perspective on Deep Learning with key steps in the evolution of learning techniques, deep learning models, and deep models investigation techniques.*

Deep learning in non-supervised settings*: Unsupervised DL models (AutoEncoders), self-supervised learning practices for pre-training, metric-based and zero-shot learning, knowledge distillation. Deep Learning Models for Anomaly Detection and Image Restoration.*

The Transformers*: The Attention Mechanism and the Transformers (in natural language processing). The Attention mechanism in images and Vision Transformers, Self-supervised Learning for Images, Contrastive Learning / Multimodal Learning (e.g., DINO, CLIP, etc).*

Generative AI: *Advanced models for Image generation, Normalizing Flows, Diffusion Models, DALL-E and text-conditional image generation.*

Graph Neural Networks: *Learning on Graphs, Node Embedding, Network Embedding, Graph Convolutional Networks, etc.*

Lectures are accompanied by practical lab sessions where students can practice on Colab the materials seen during lectures and implement models for specific applications.

Course Calendar

Day 1 (Ven 28/2 -- 14:00-19:00) - Matteucci / Boracchi / Mentasti
- Course Introduction
- Deep learning in non-supervised settings
- Unsupervised Deep Learning / Self-supervised / Metric Learning
- Deep Learning for Image Restoration (Denoising/Inpainting)
- Anomaly Detection (Restoration-based, Student Teacher, Self-supervised)
- Coding labs
- Day 1 evaluation

Day 2 (Ven 7/3 -- 14:00-19:00) and Day 3 (Ven 14/3 -- 14:00-19:00) - Matteucci / Mentasti
- Transformers and multimodal learning
- Attention and Transformers
- Vision Transformers
- CLIP, DINO + Zero / Few Shot Learning
- Coding labs
- Day 2/ Day3 evaluation

Course Calendar

Day 4 (Ven 21/3 / 14:00 - 19:00) - Boracchi / Papini
- Generative AI (4h+4h)
- Generative Models: VAE Normalizing flows, and Diffusion Models, DALL-E, etc.
- Coding labs
- Day 4 evaluation

Days 5 (Ven 28/3 / 14:00 - 19:00) and Day 6 (Ven 4/4 -- 14:00-19:00) - Matteucci / Papini
- Deep Learning beyond images
- Graph Neural Networks (mesh and graphs, node embedding)
- Coding labs
- Day 5 / Day 6 evaluation