

Autoencoders and Generative Adversarial Networks

Giacomo Boracchi

Advanced Neural Networks and Deep Learning

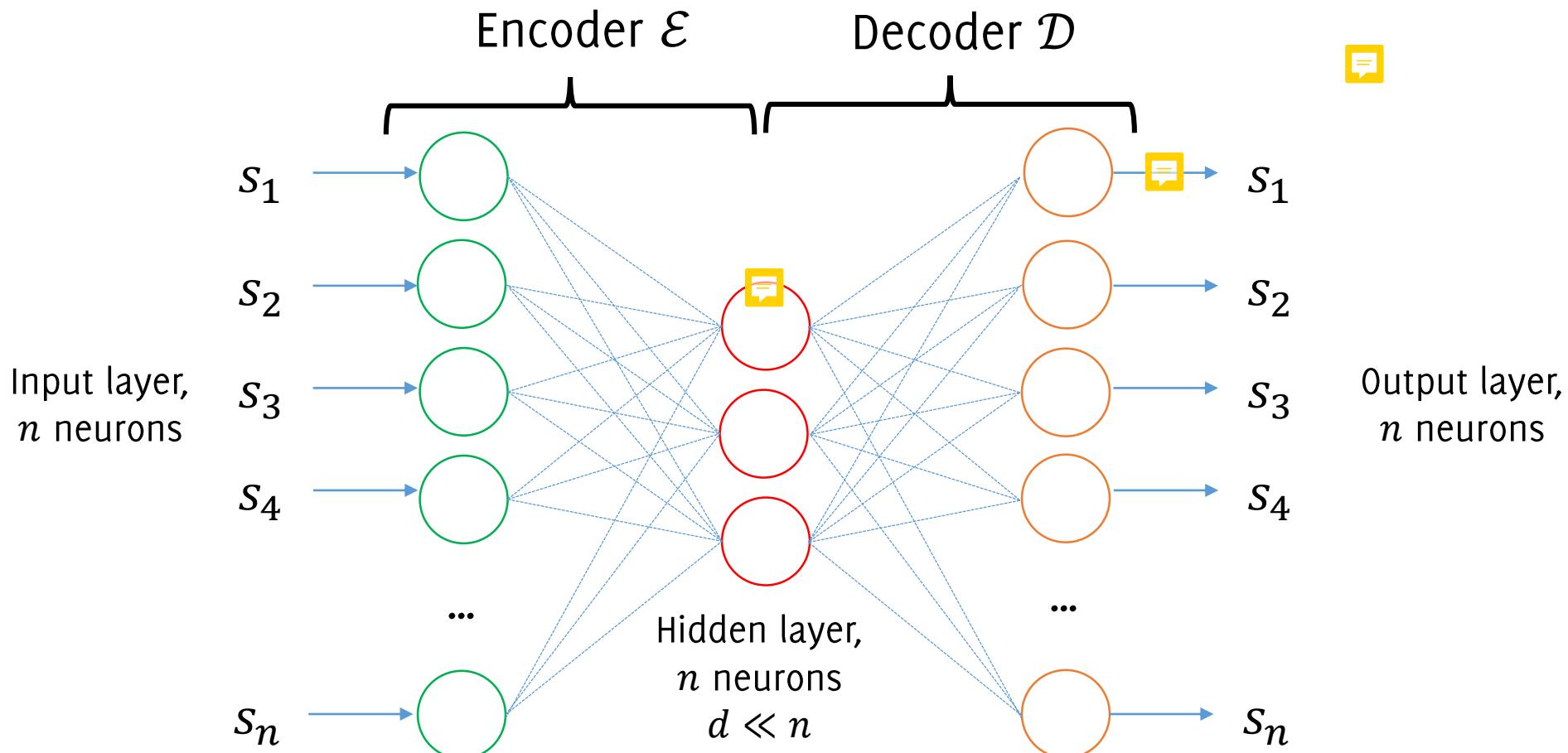
<https://boracchi.faculty.polimi.it/>

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

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

and training of $\mathcal{D}(\mathcal{E}(\cdot))$ 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

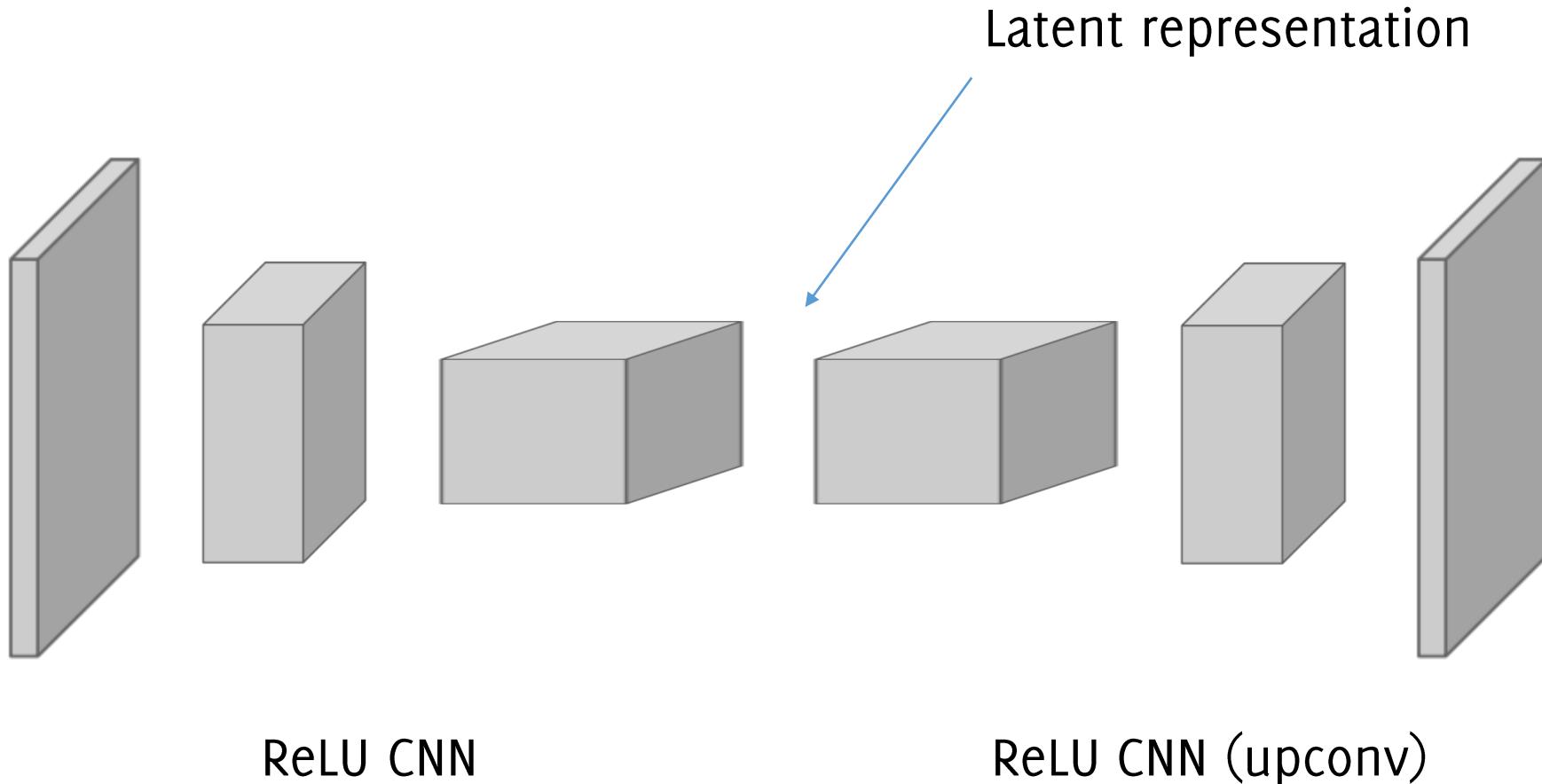
Autoencoders

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 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

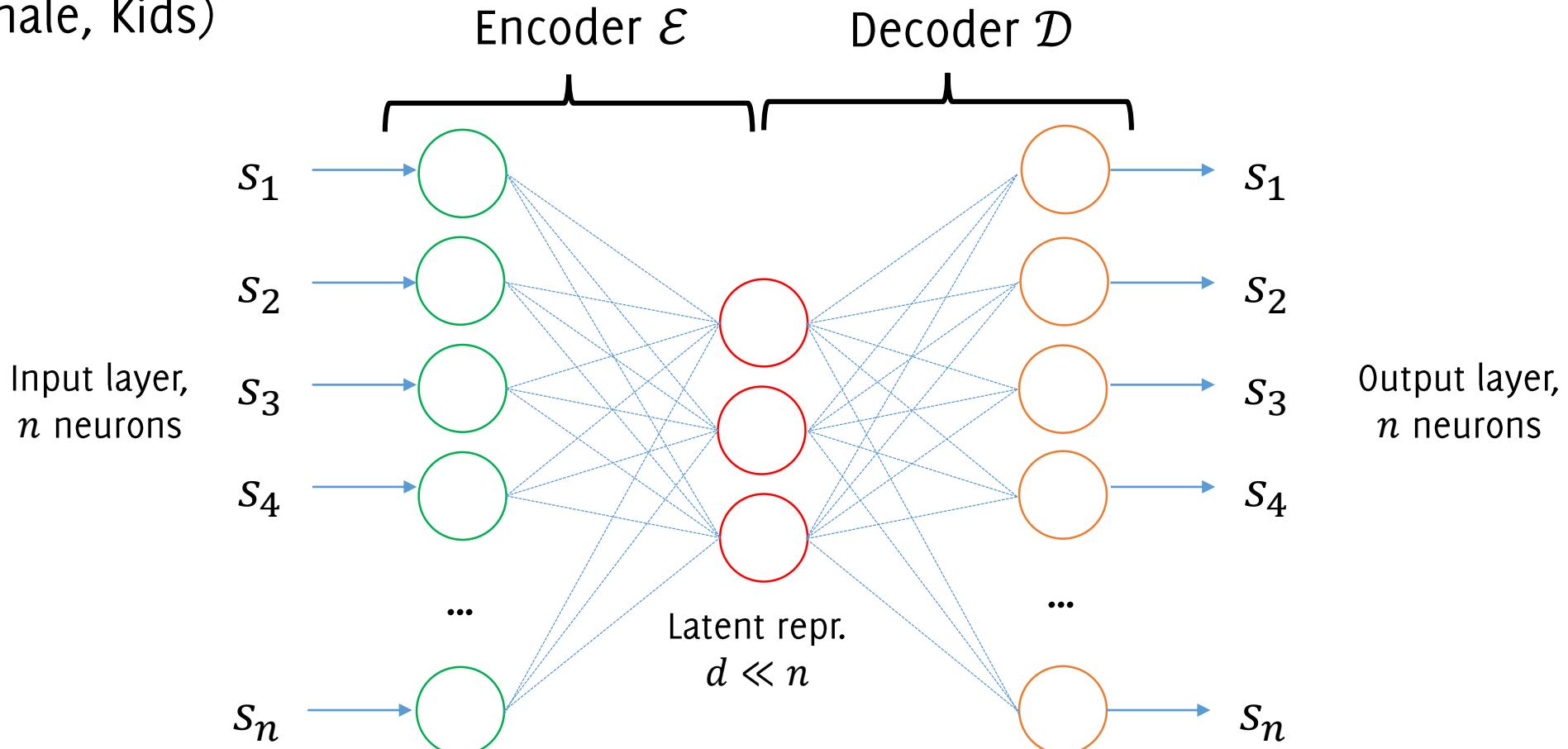
And of course it is possible to use convolutional layers and transpose convolution to implement a deep convolutional autoencoder



Using Autoencoders for classifier initialization

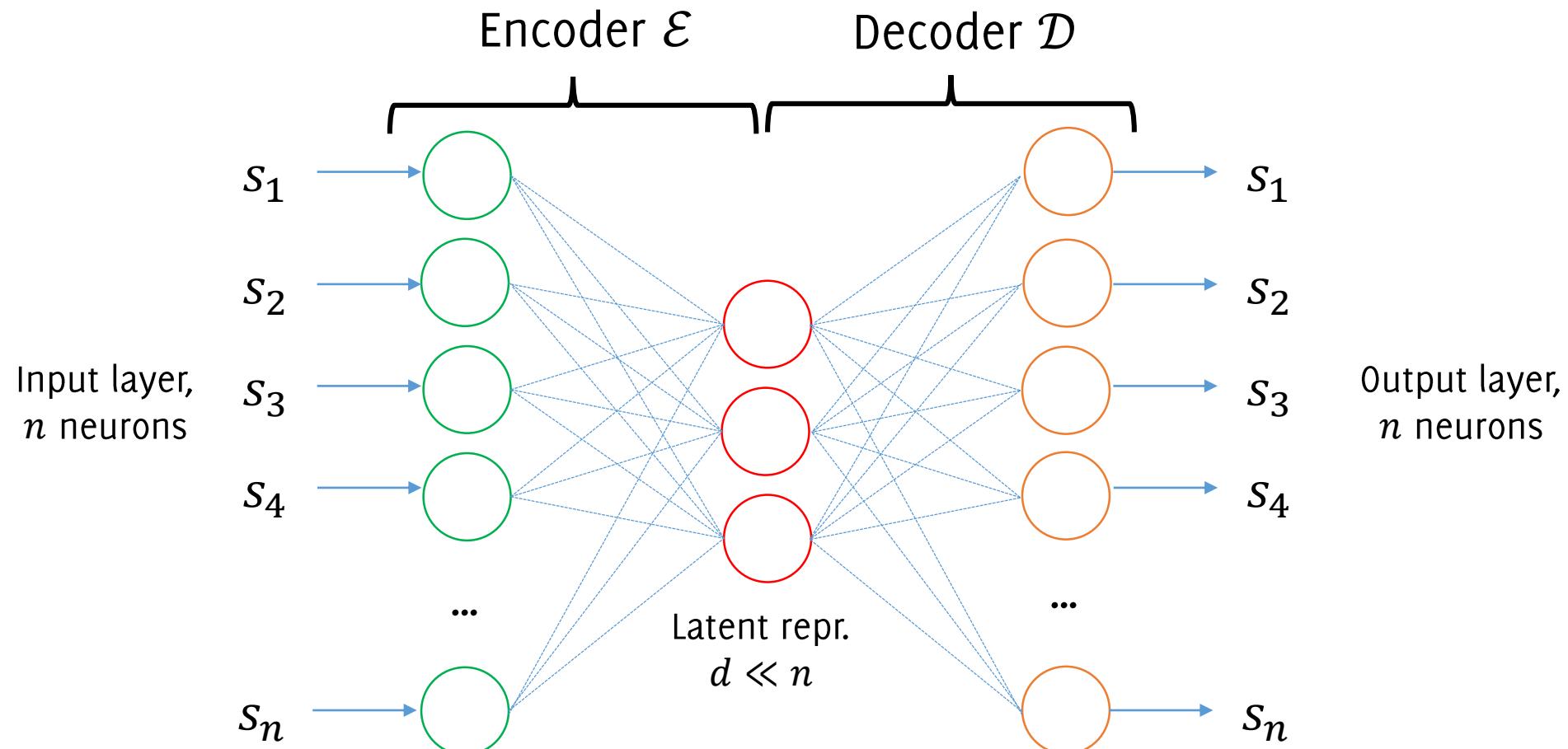
Autoencoders can be used to **inizialize the classifier** when the training set includes

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



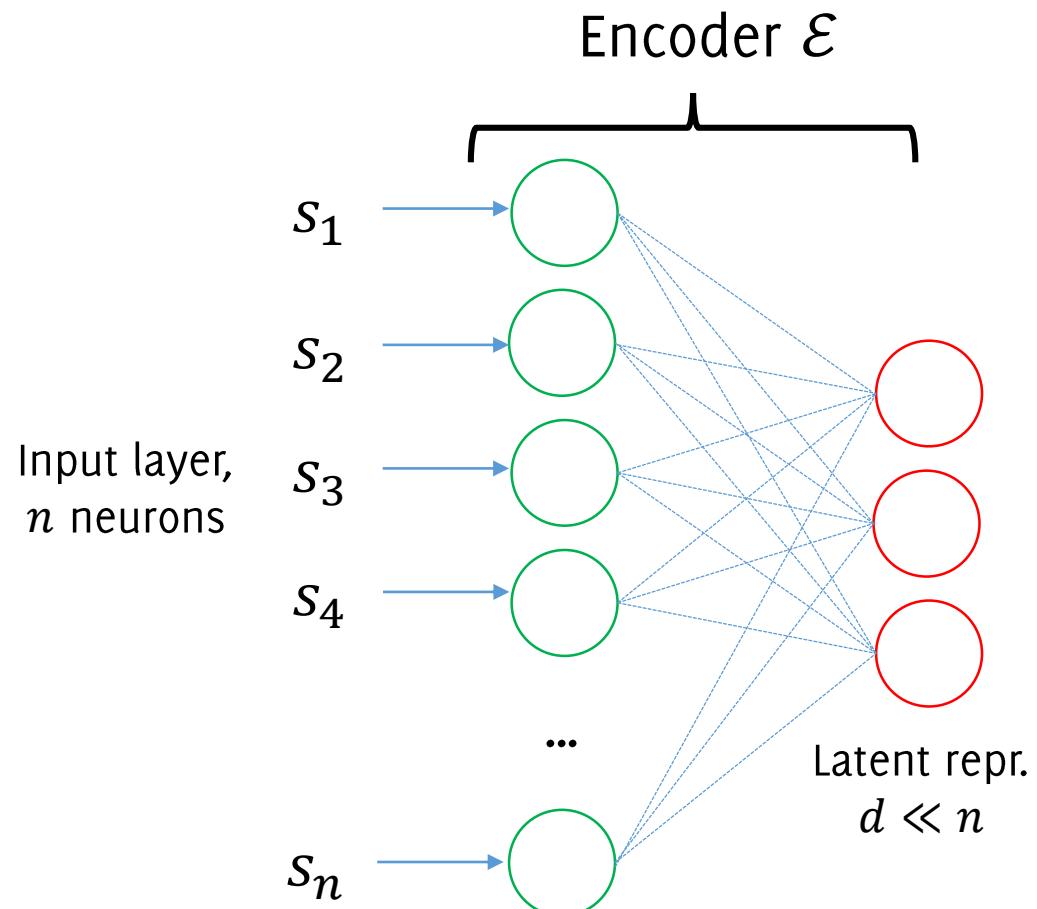
Using Autoencoders for classifier initialization

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



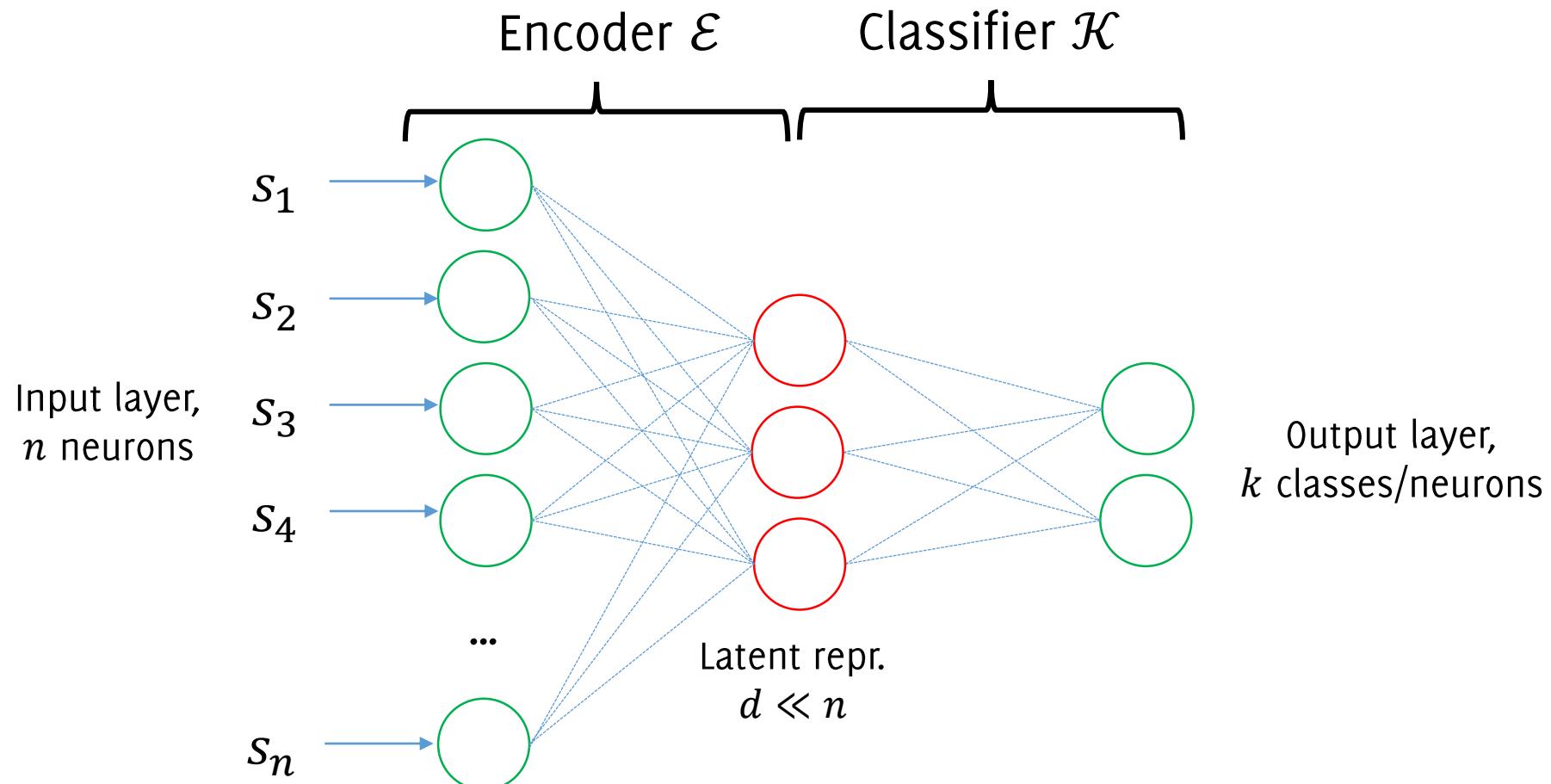
Using Autoencoders for classifier initialization

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



Using Autoencoders for classifier initialization

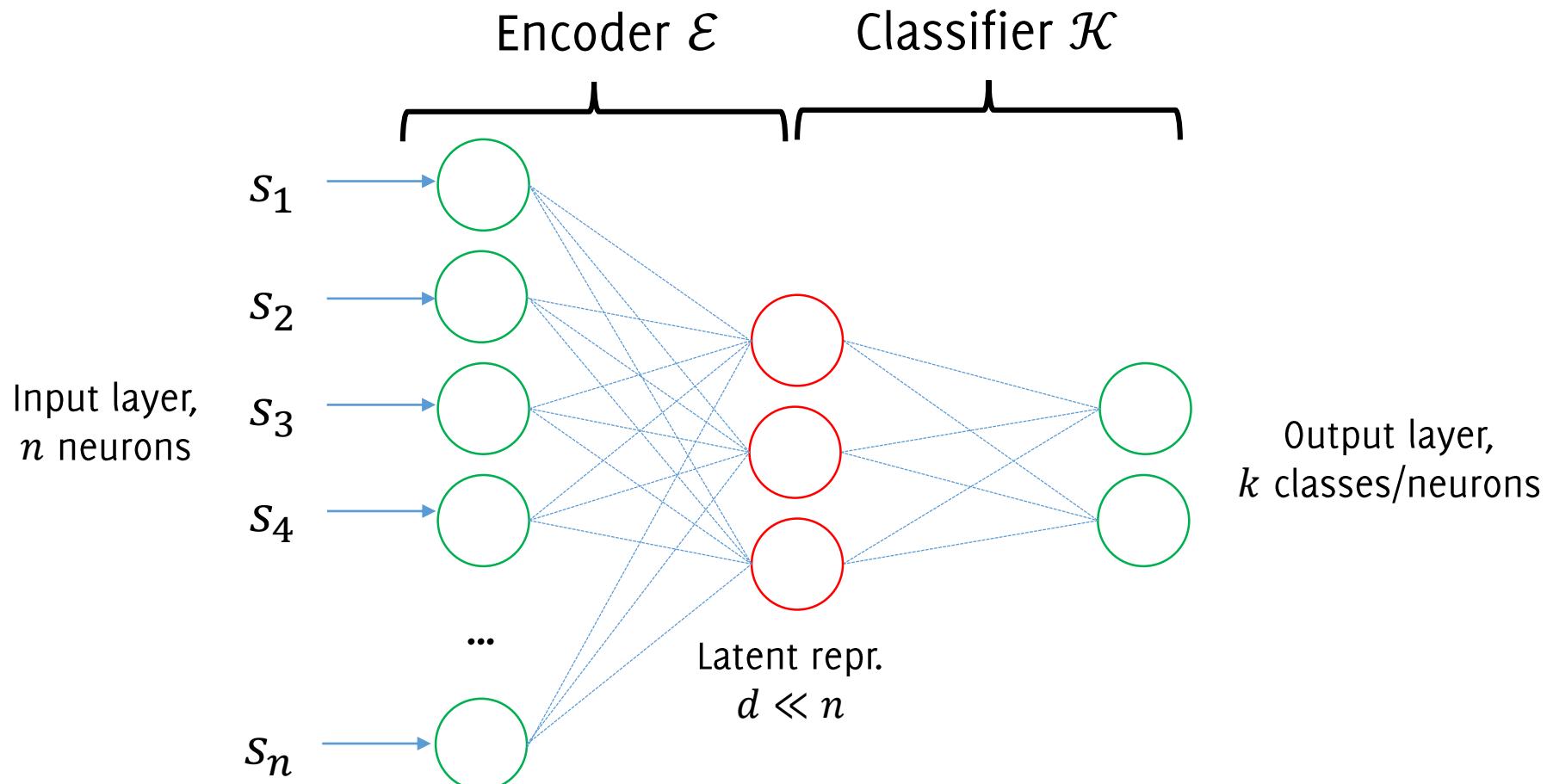
- 3) Plug in a FC layer for classifying samples from the latent representation



Using Autoencoders for classifier initialization

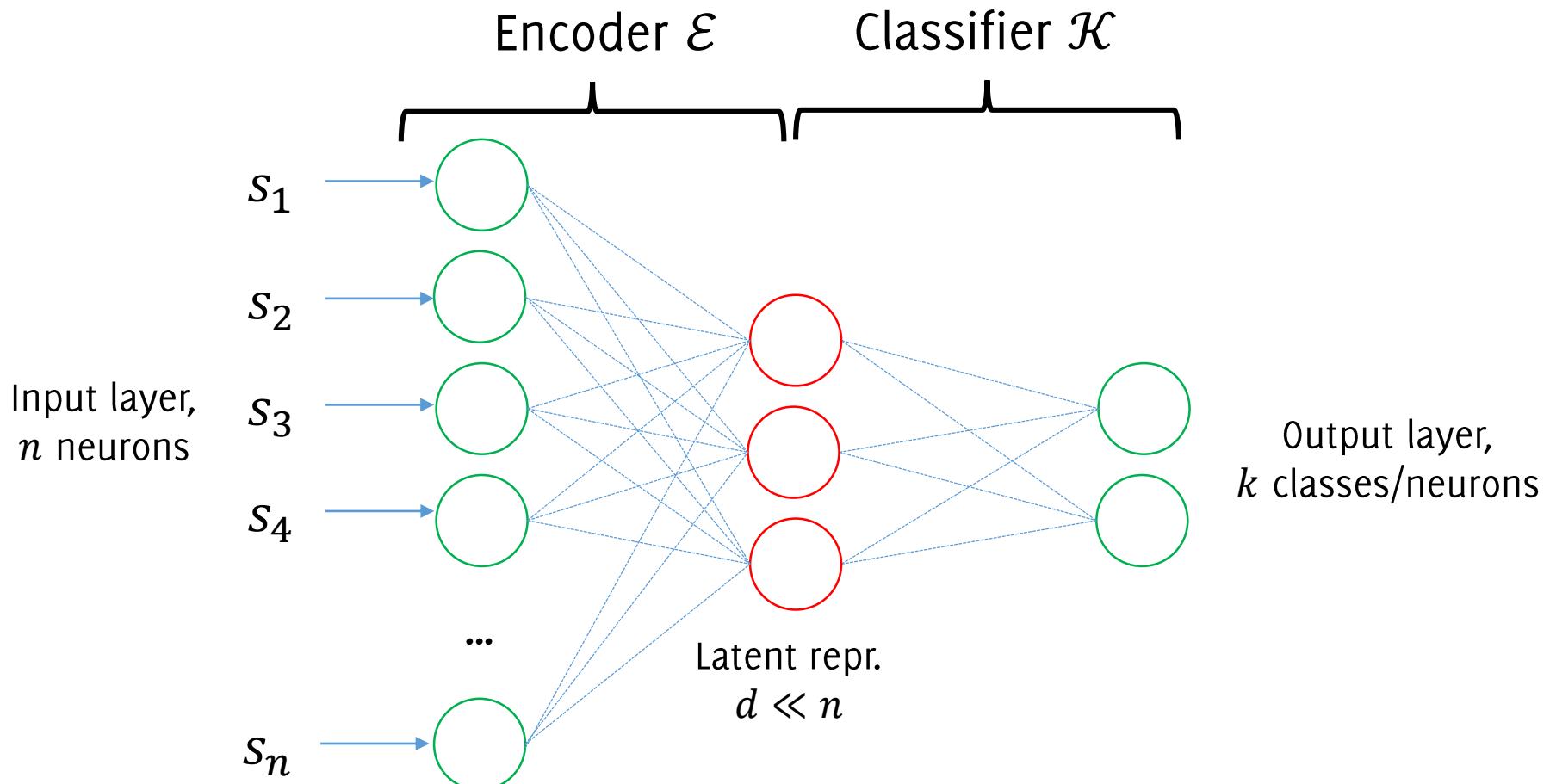
4) Fine tune the autoencoder using the few supervised samples provided L . This is perfectly in line with «Transfer Learning» and holds for whatever model.

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



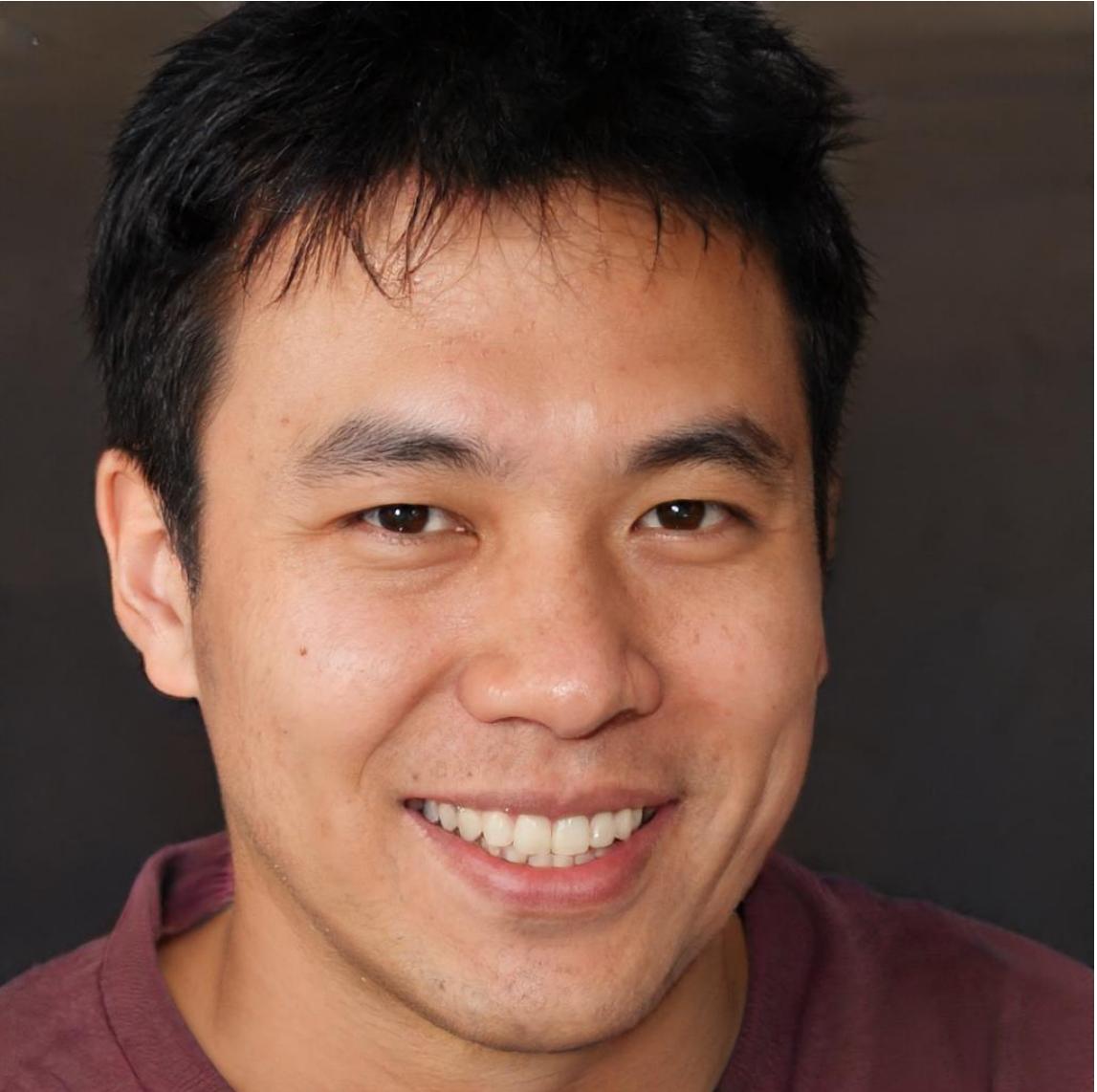
Using Autoencoders for classifier initialization

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.



**Generative Models:
Networks able to generate realistic images**

Which image is real and which one generated?



<https://thispersondoesnotexist.com/> 1024 x 1024



<https://thispersondoesnotexist.com/>



<https://thispersondoesnotexist.com/>



<https://thispersondoesnotexist.com/>



Sometimes there models are not perfect...



Sometimes there models are not perfect...



even with cats (lower resolution 512 x 512) ...



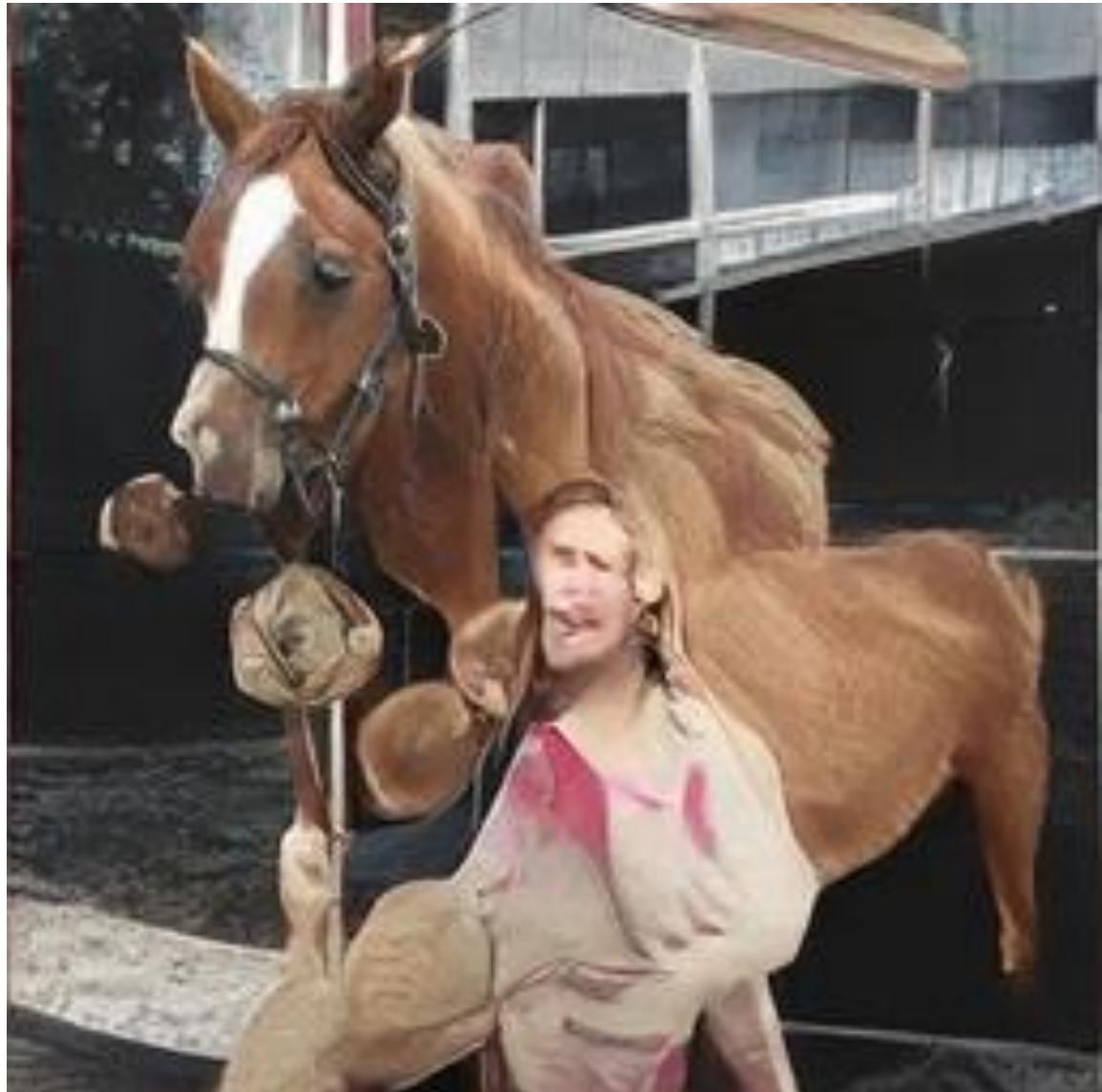
even with cats (lower resolution 512 x 512) ...



even with horses (lower resolution 256 x 256) ...



even with horses (lower resolution 256 x 256) ...



Generative Models

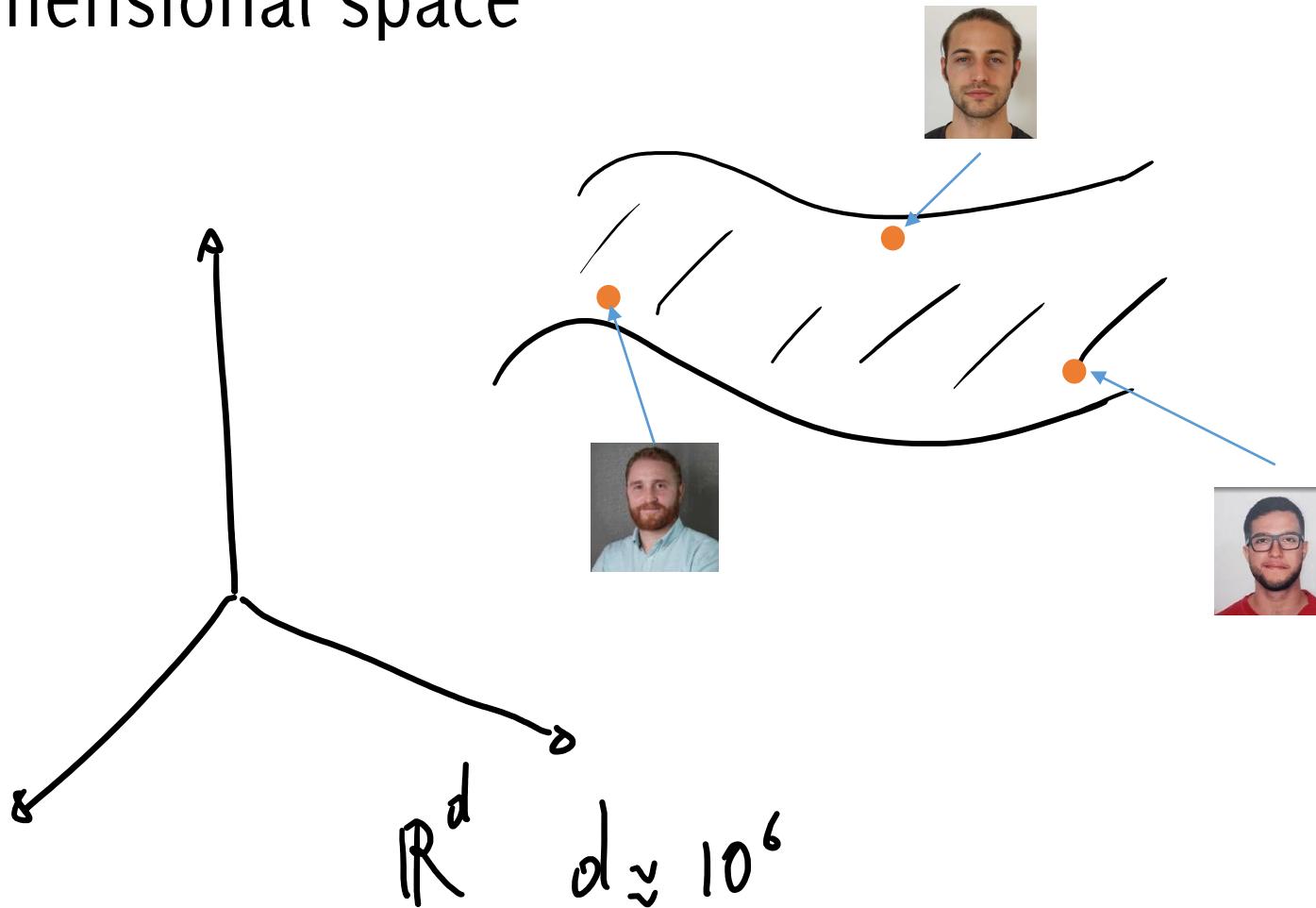
Goal:

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

$$TR = \left\{ \begin{array}{c} \text{[Image of man with red beard]} \\ , \quad \text{[Image of man with glasses]} \\ , \quad \text{[Image of man with dark hair and glasses]} \\ , \quad \text{[Image of man with light brown hair]} \end{array} \right\}$$

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
- Training generative models can also enable inference of latent representations that can be **useful as general features**
- **Realistic samples for artwork, super-resolution, colorization, etc.**
- 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 first effective generative model (i.e. GANs) give rise to new training paradigm and practices (adversarial training)

Images randomly generated by a GAN

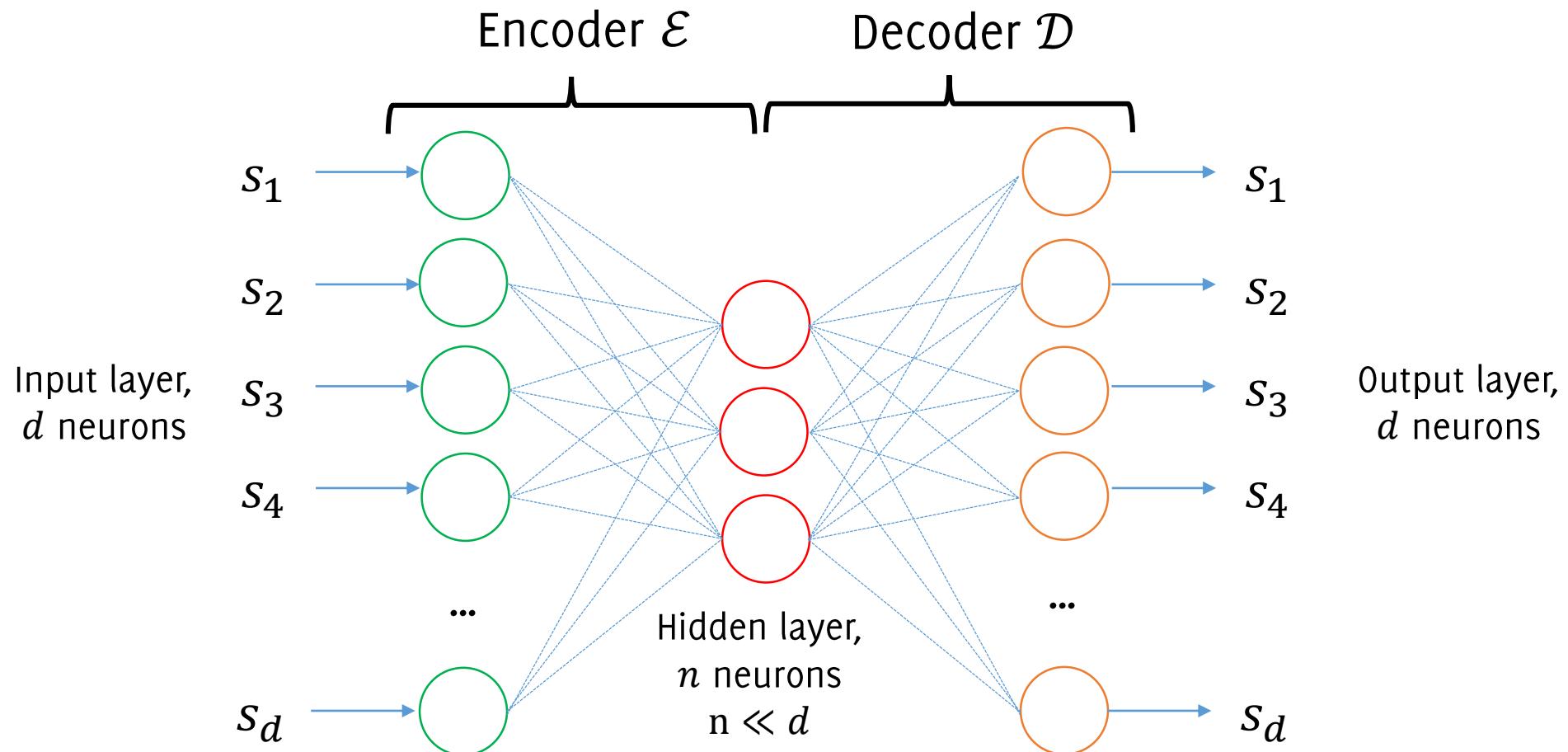


Why don't we use an autoencoder?

What about using Autoencoders as Generative Models?

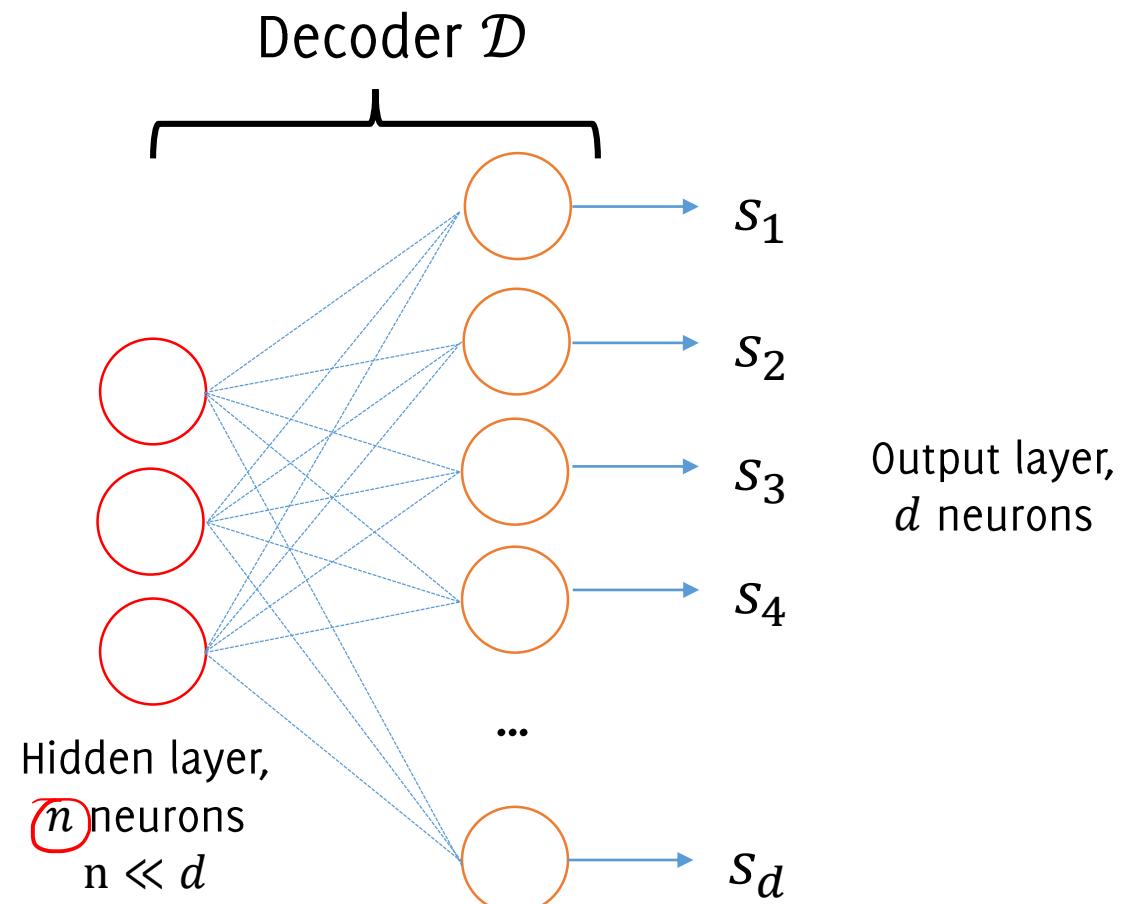
One option would be to

- 1) train an autoencoder on S



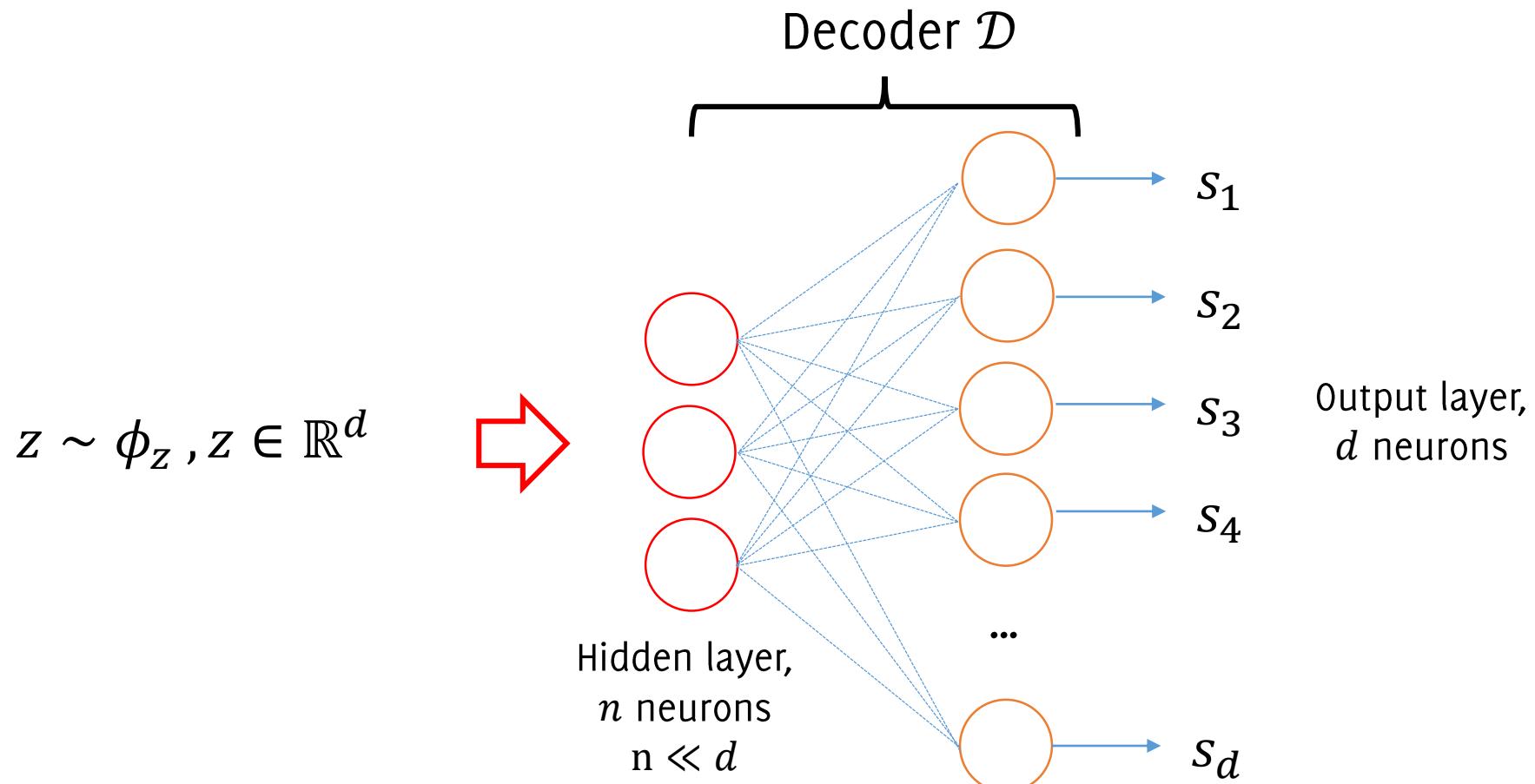
What about using Autoencoders as Generative Models?

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

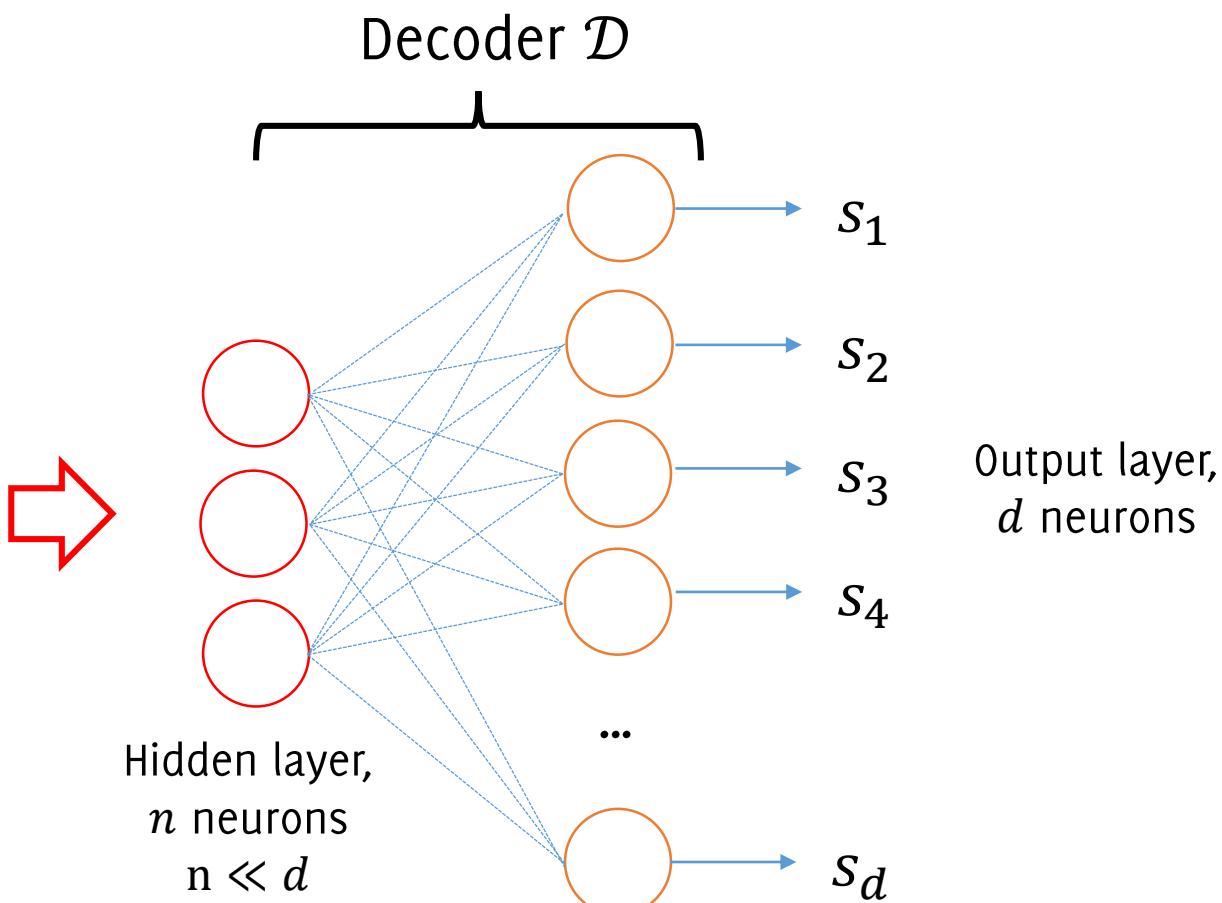


What about using Autoencoders as Generative Models?

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

Variational autoencoders leverages a prior over z to generate images using a similar architecture

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



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:

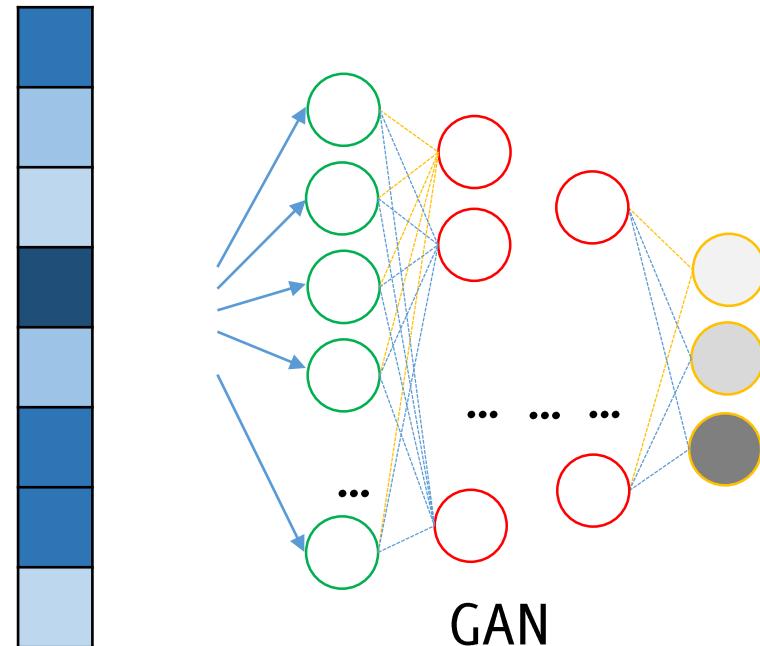
- Sample a seed from a known distribution ϕ_z . This is defined a priori and also referred to as noise.
- Feed this seed to a learned transformation that generates realistic samples, as if they were drawn from ϕ_S

Use a neural network to learn this transformation

The neural network is going to be trained in an unsupervised manner, no label needed

Generative Adversarial Networks (GAN)

The GAN approach:

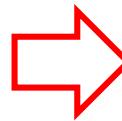
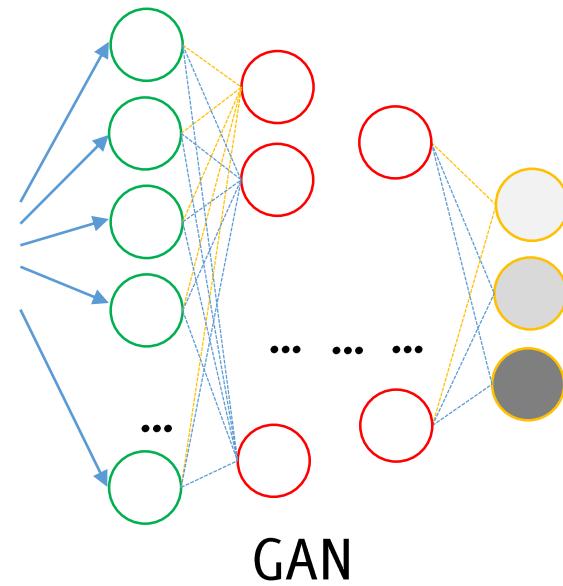


$$z \sim \phi_z$$

Draw a sample from the
noise distribution

Generative Adversarial Networks (GAN)

The GAN approach:

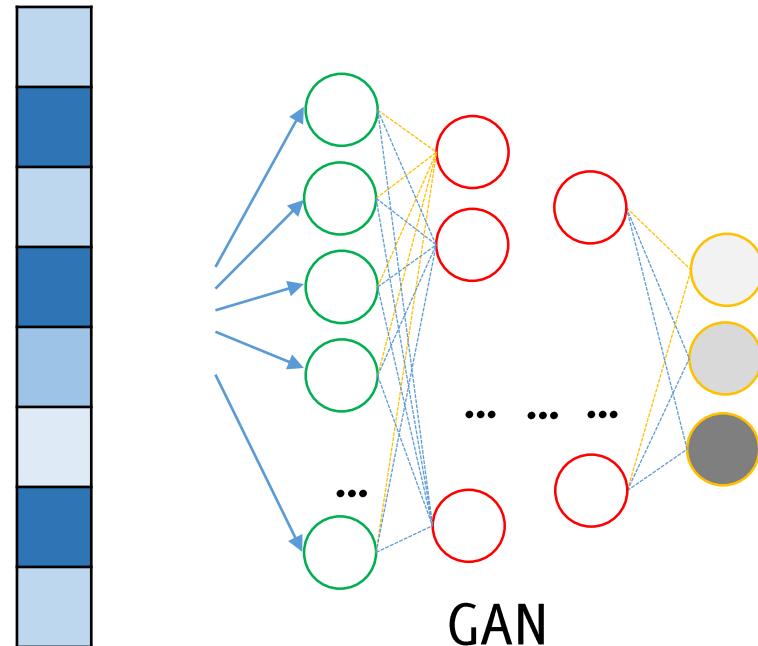


$$z \sim \phi_z$$

Draw a sample from the
noise distribution

Generative Adversarial Networks (GAN)

The GAN approach:

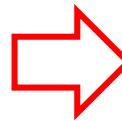
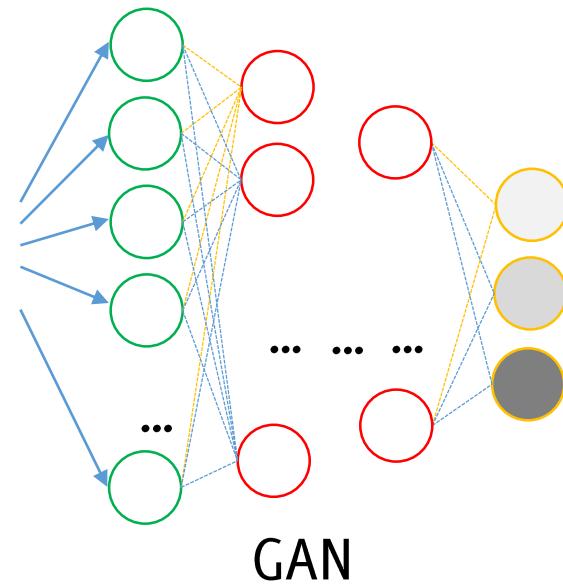


$$z \sim \phi_z$$

Draw a sample from the
noise distribution

Generative Adversarial Networks (GAN)

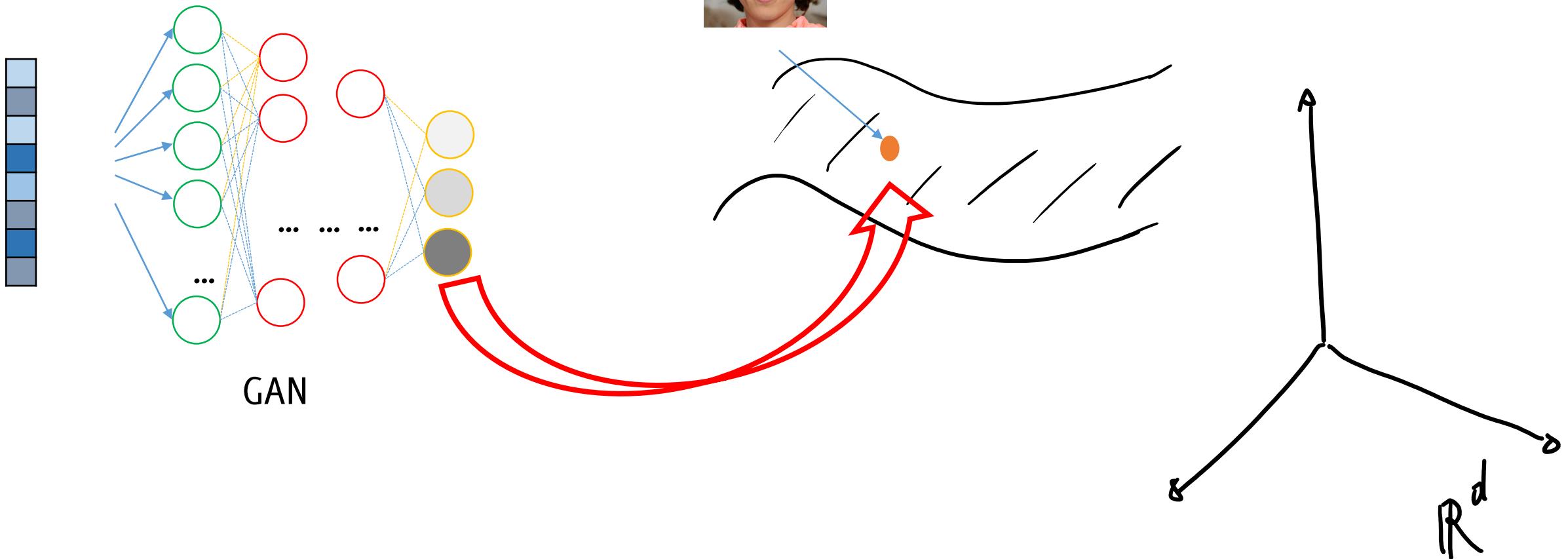
The GAN approach:



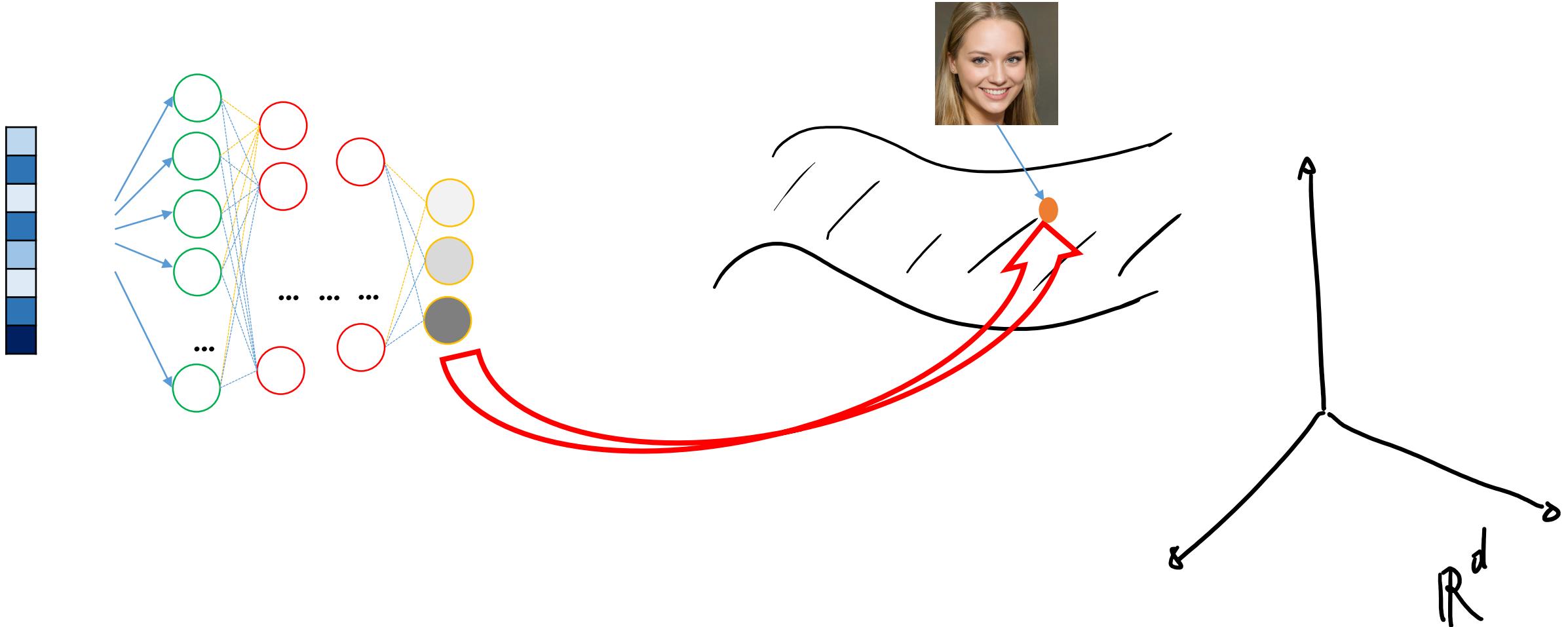
$$z \sim \phi_z$$

Draw a sample from the
noise distribution

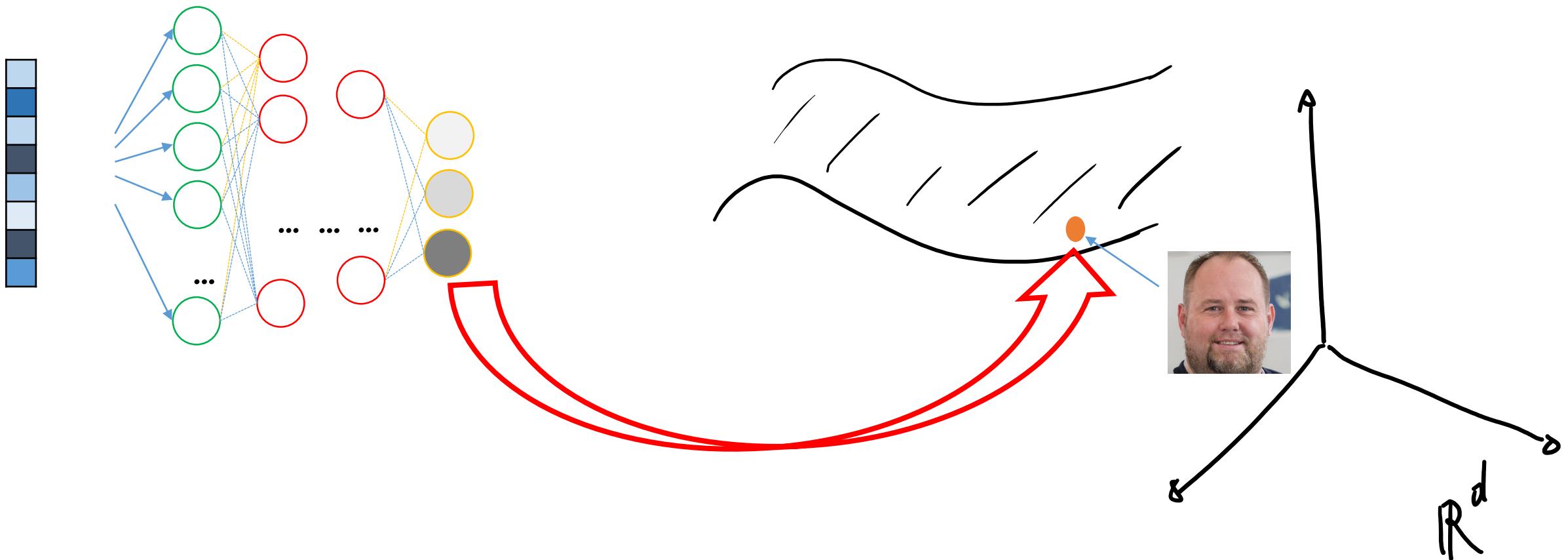
The GAN Approach



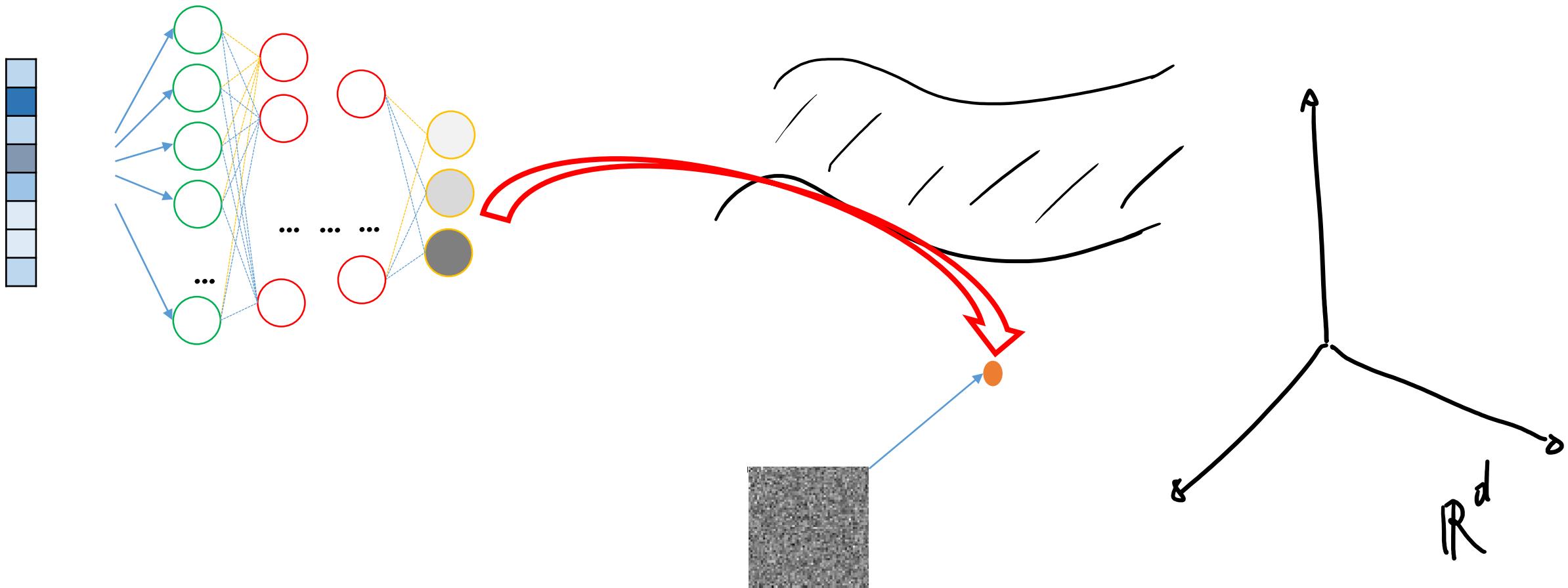
The GAN Approach



The GAN Approach

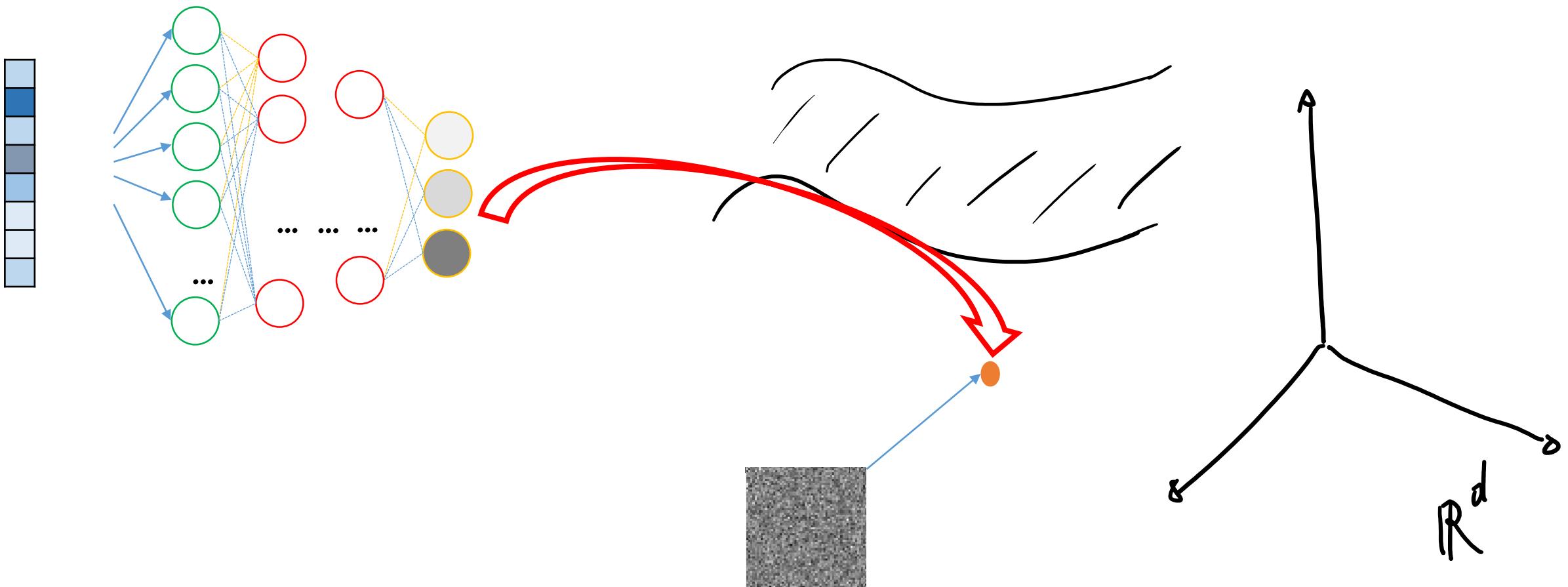


The GAN Approach



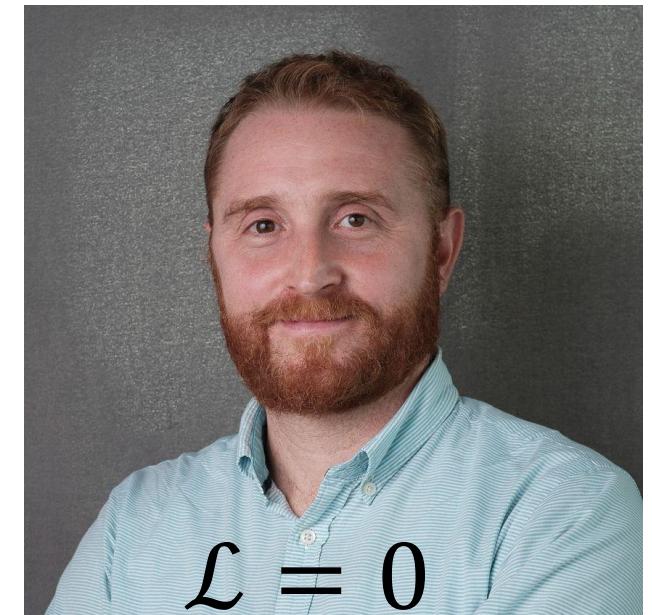
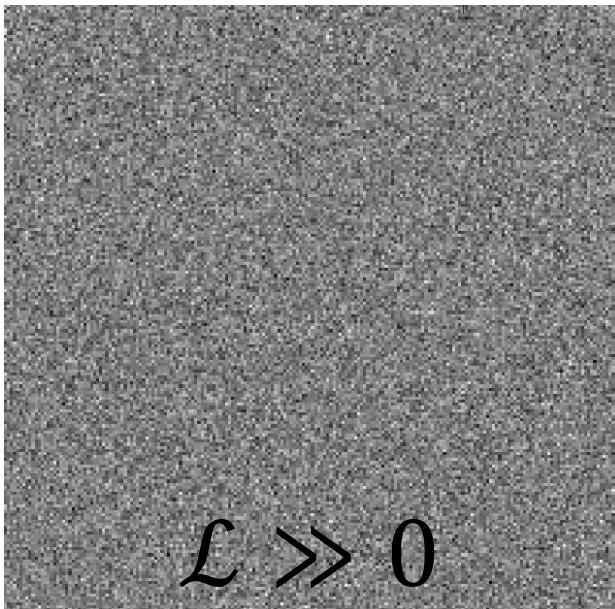
The GAN Approach

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?



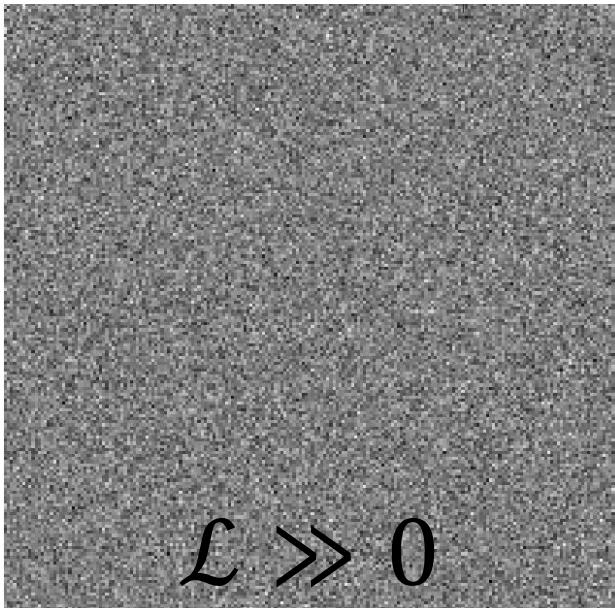
What a loss function?

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

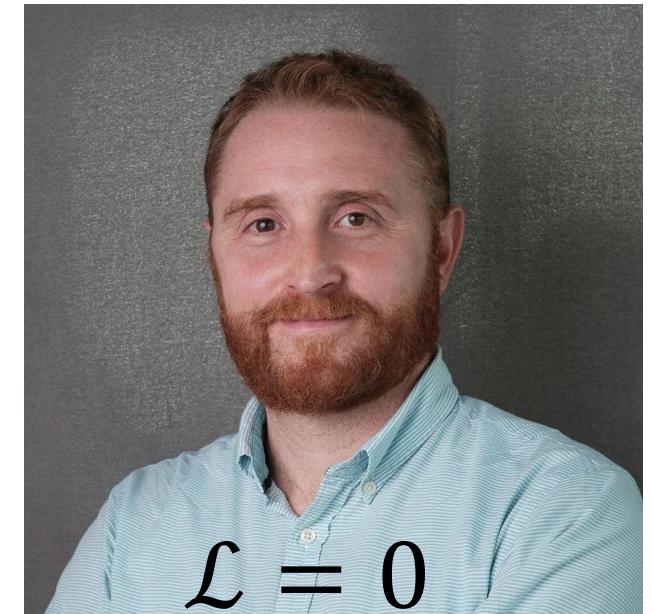
GAN solution: resort to a neural network to define the loss!



$$\mathcal{L} \gg 0$$



$$\mathcal{L} \approx 0$$



$$\mathcal{L} = 0$$

Generative Adversarial Networks (GAN)

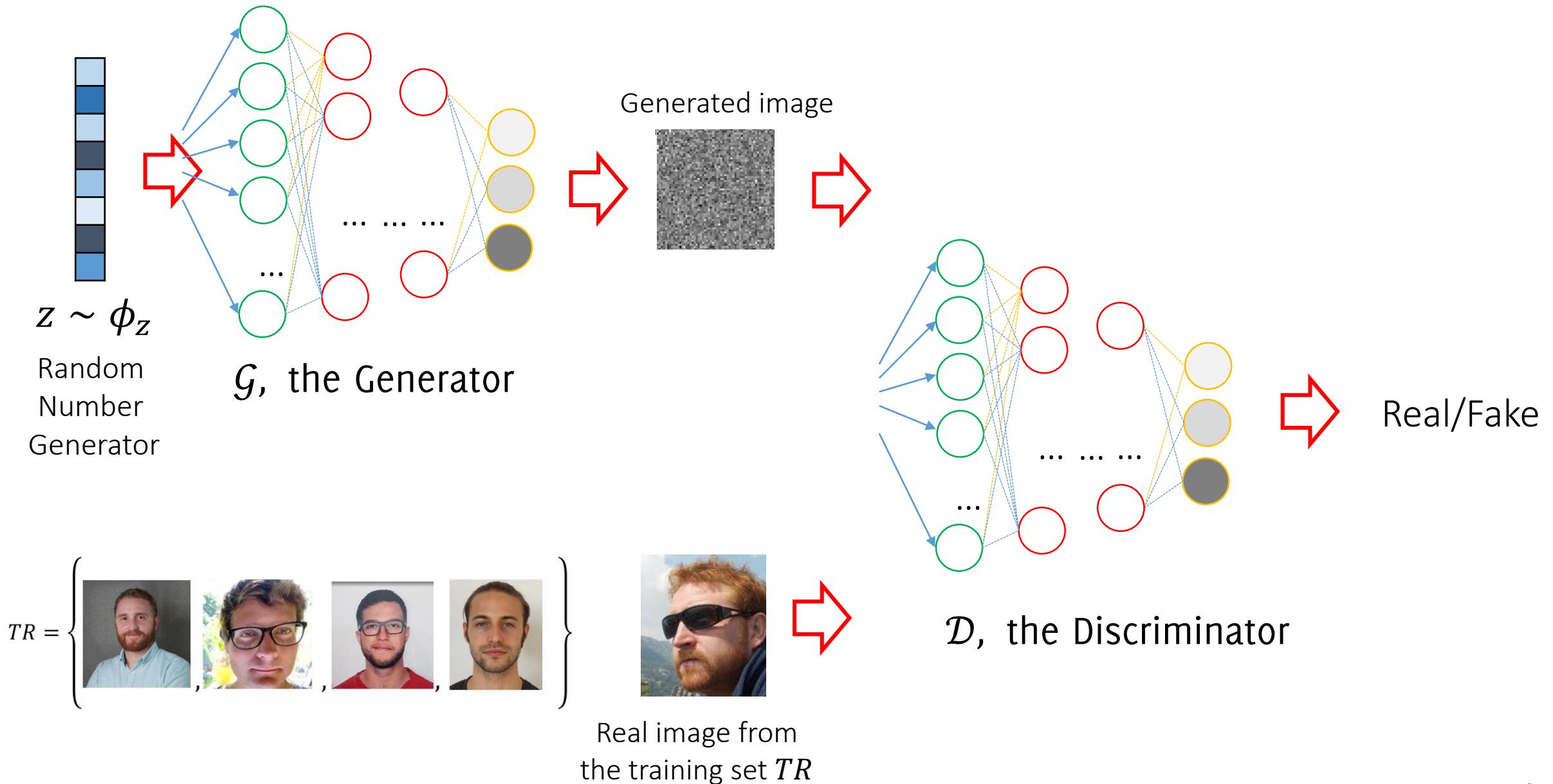
The GAN solution: Train a pair of neural networks addressing two different tasks that compete in a sort of two player (adversarial) game.

These models are:

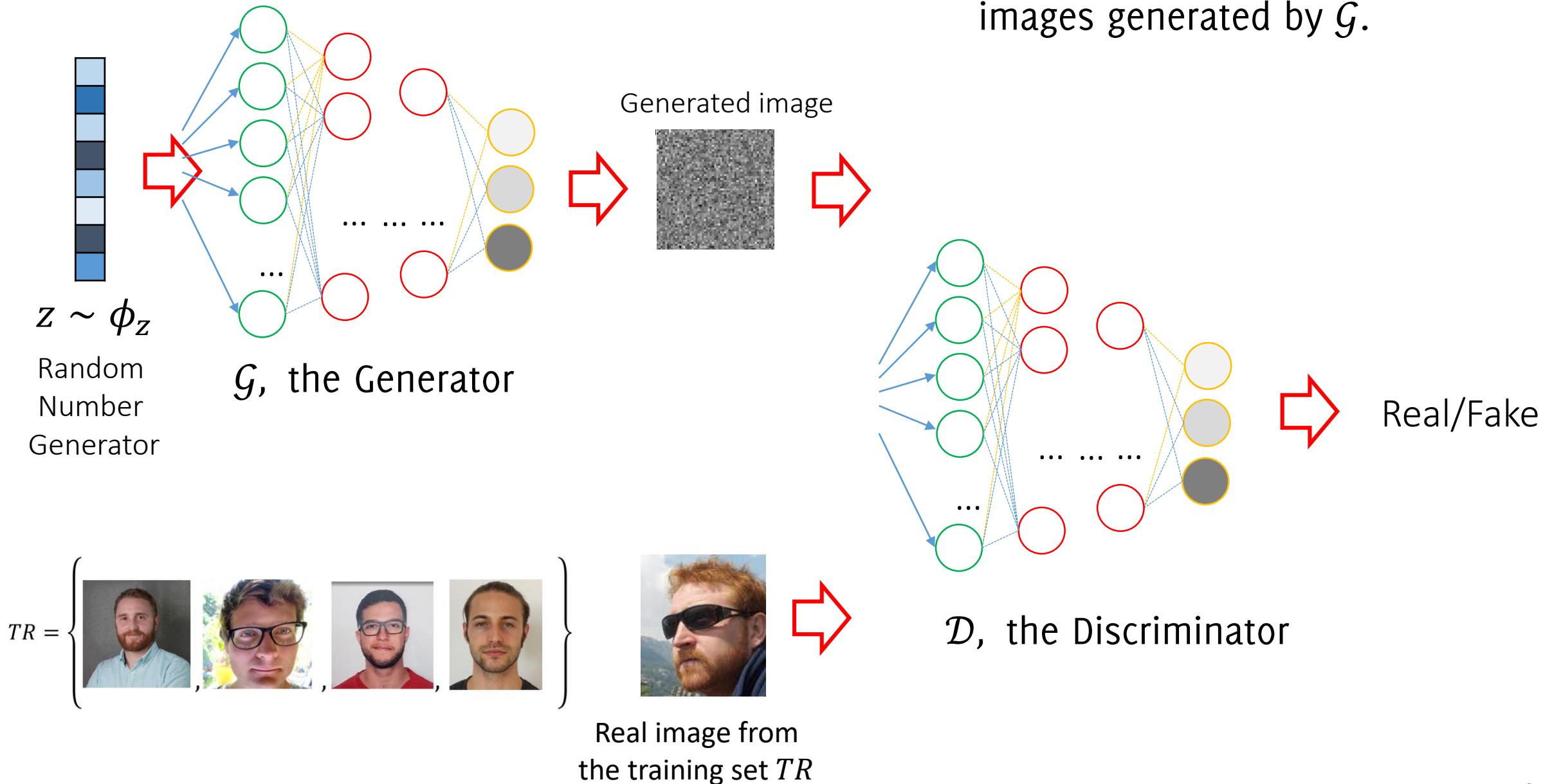
- Generator \mathcal{G} that produces realistic samples e.g. taking as input some random noise. \mathcal{G} tries to fool the discriminator
- Discriminator \mathcal{D} that takes as input an image and assess whether it is real or generated by \mathcal{G}

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

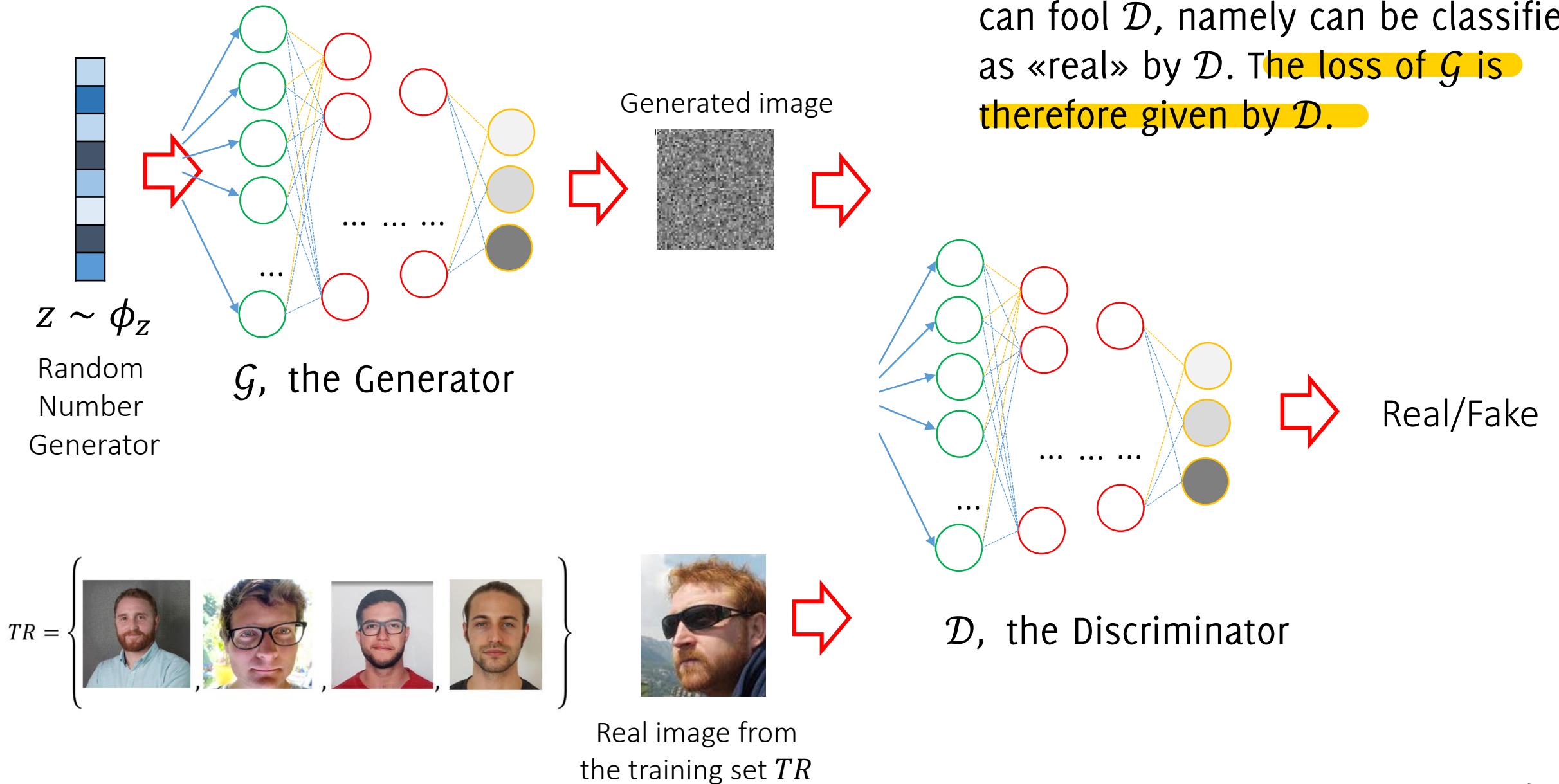
GAN Training



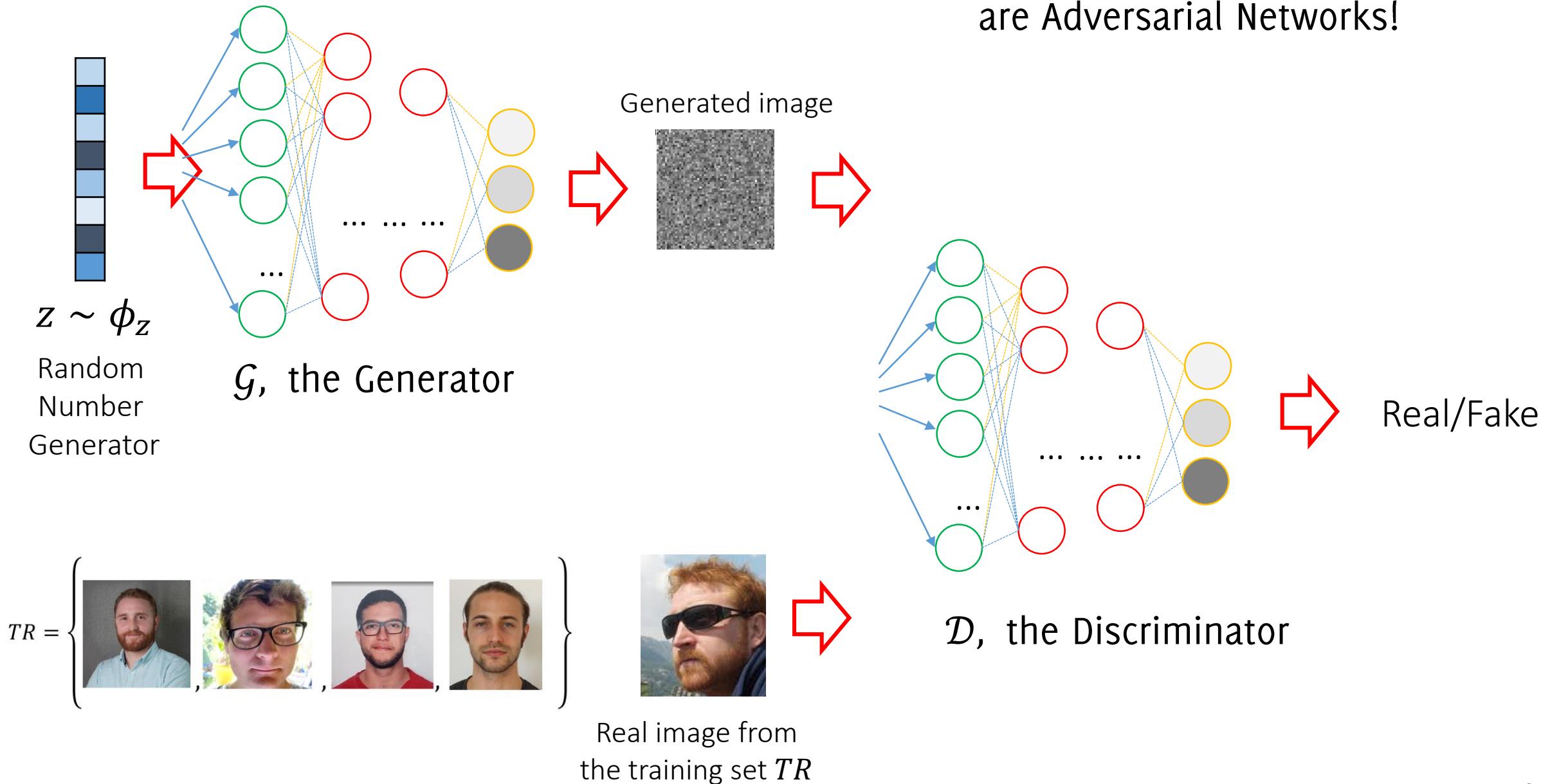
GAN Training



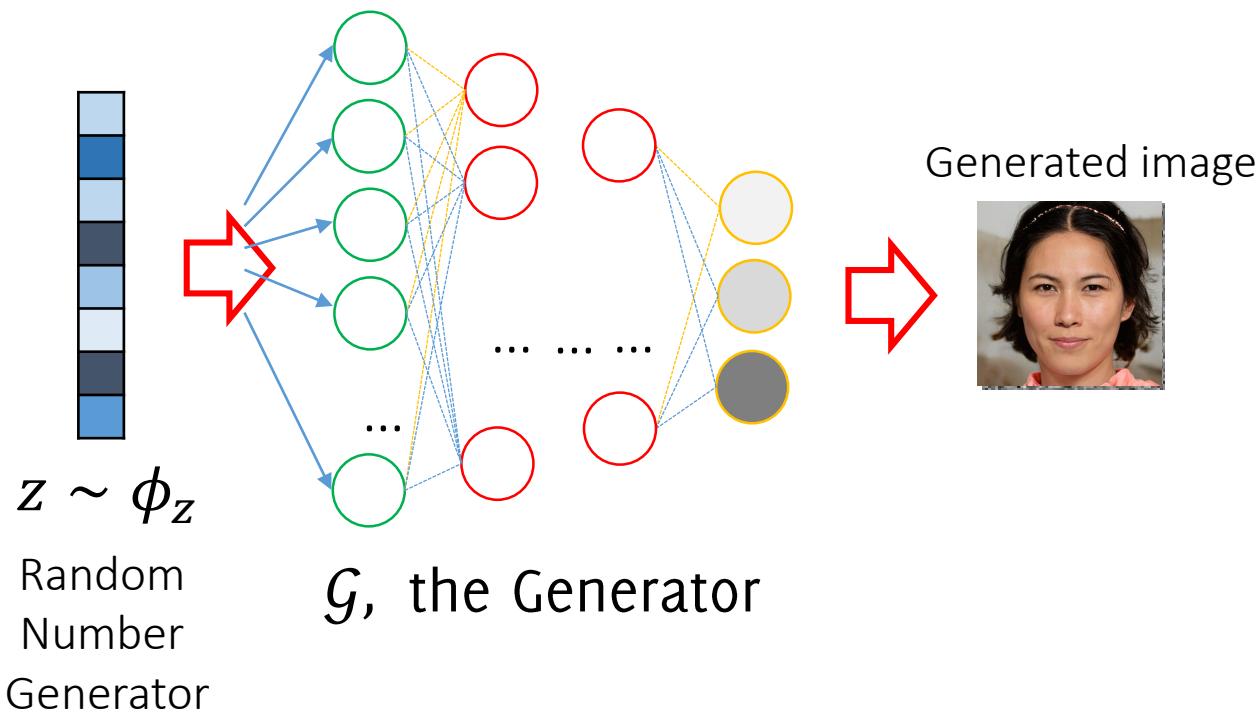
GAN Training



GAN Training



GAN Inference



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

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

\mathcal{D} is expected effective to distinguish real and fake images... if G can fool \mathcal{D} , this means that 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 the real/fake

GAN: Setting up the stage

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

Our networks take as input:

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

This notation is meant to visualize what are the NN parameters.. The network is taking a single input

θ_g and θ_d are network parameters, $s \in \mathbb{R}^n$ is an input image (either real or generated by \mathcal{G}) and $z \in \mathbb{R}^d$ 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

GAN Training

A good discriminator is such:

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

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

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

Written using expectations

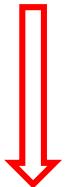
GAN Training

A good discriminator is such:

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

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

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



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



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

GAN Training

A good discriminator is such:

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

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

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

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

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

GAN Training

Solve by an iterative numerical approach

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

GAN Training

Solve by an iterative numerical approach

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

Alternate:

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

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

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

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

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

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

GAN Training

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

for k –times # gradient ascent steps for θ_d

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

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

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

Update \mathcal{G} by stochastic gradient descent:

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

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

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

end for

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

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

end for

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

GAN Training

```
for i = 1 ... #number of epochs
```

```
for k -times # gradient ascent steps for  $\theta_d$ 
```

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

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

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

Update \mathcal{G} by stochastic gradient descent:

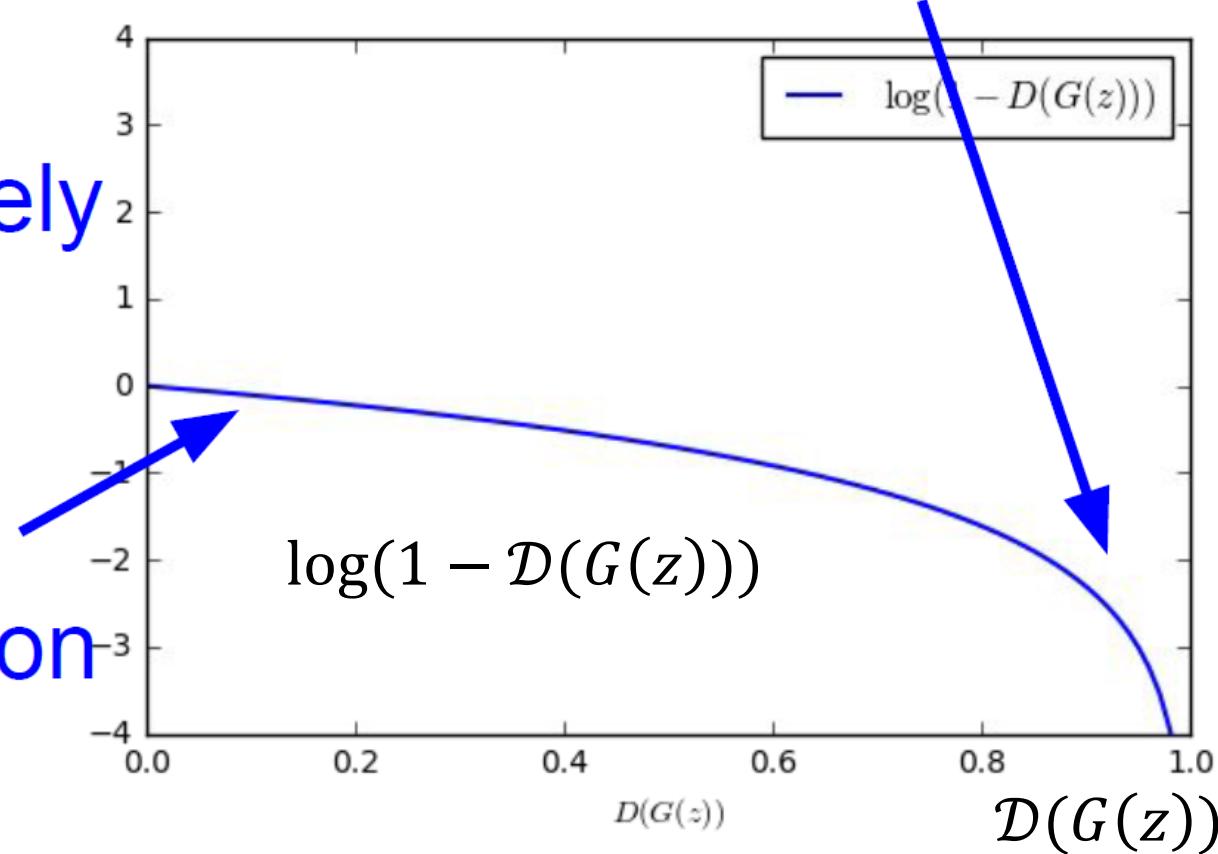
$$\nabla_{\theta_g} \left[\sum_i \log \left(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}_i, \theta_g), \theta_d) \right) \right]$$

This was presented as a best practice,
latest GANs such as Wasserstein GANs do
not use this

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!



One of the many GAN Training «trick»

When optimizing for θ_g , instead of minimizing the following

$$\min_{\theta_g} \left(E_{z \sim \phi_z} \left[\log \left(1 - D(G(z, \theta_g), \theta_d) \right) \right] \right)$$

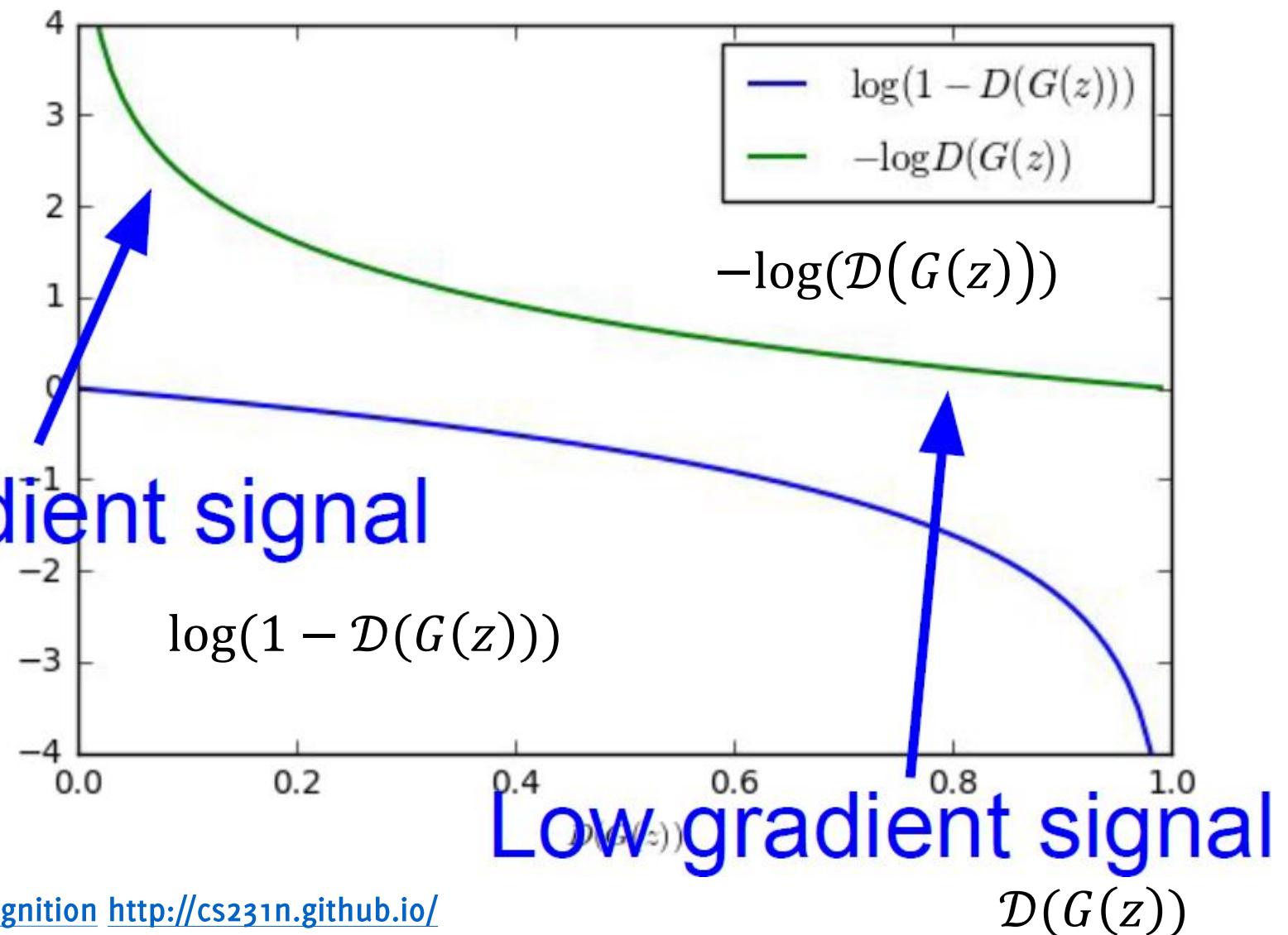
we maximize this

$$\max_{\theta_g} \left(E_{z \sim \phi_z} \left[\log \left(D(G(z, \theta_g), \theta_d) \right) \right] \right)$$

Which is equivalent in terms 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 $\log(D(G(z)))$ [or as in this figure, minimize $-\log(D(G(z)))$] This objective function results in **the same fixed point of the dynamics of G and D , but provides much stronger gradients early in learning.**

High gradient signal



GAN Training

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

for k –times # gradient ascent steps for θ_d

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

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

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

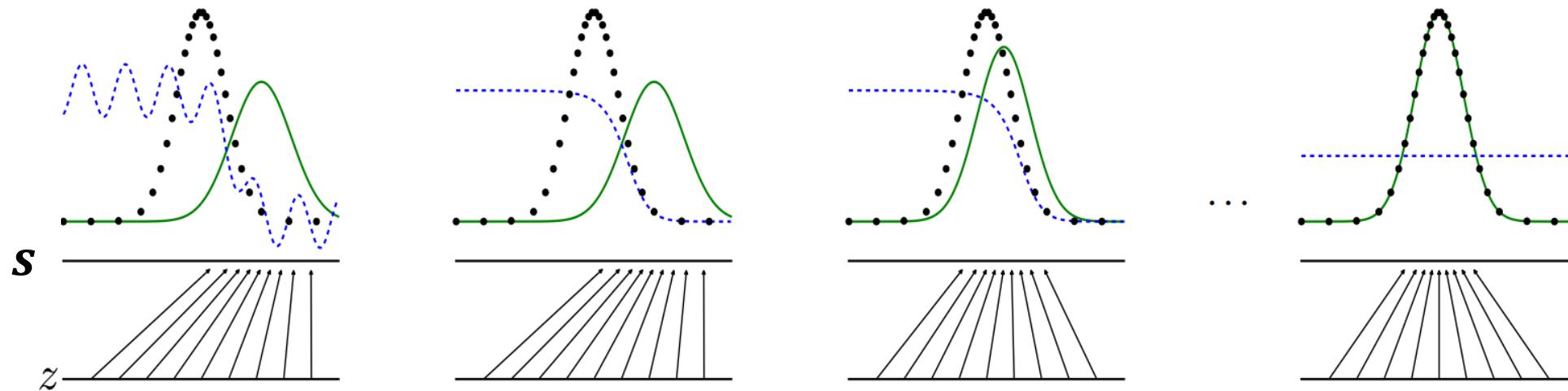
Update \mathcal{G} by stochastic gradient **ascent**:

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

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

..... ϕ_s , s real
— $\phi_{G(z)}$ $G(z)$ fake
- - - $D(\cdot)$ D posterior



At the end of the day...

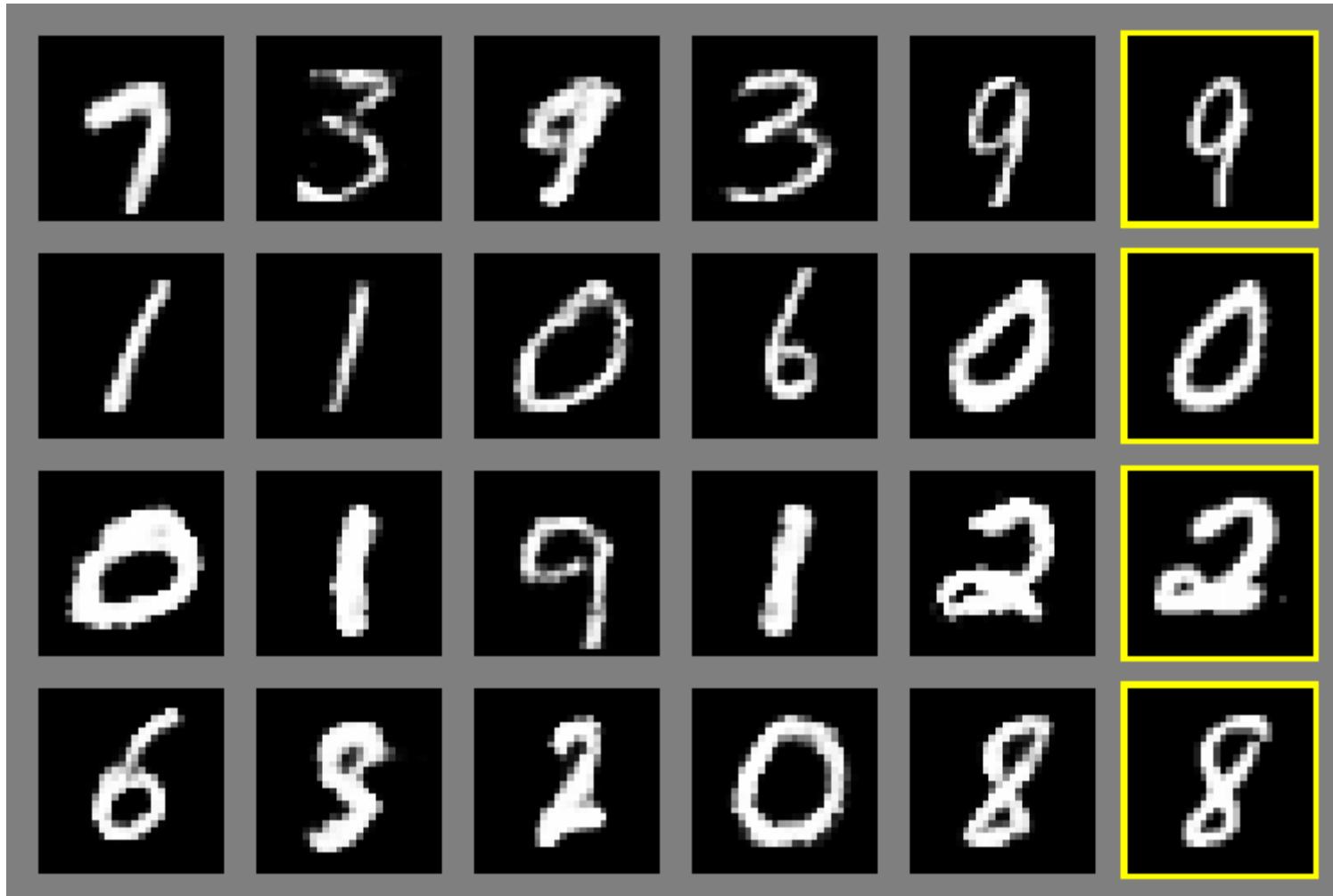
The discriminator \mathcal{D} is discarded

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

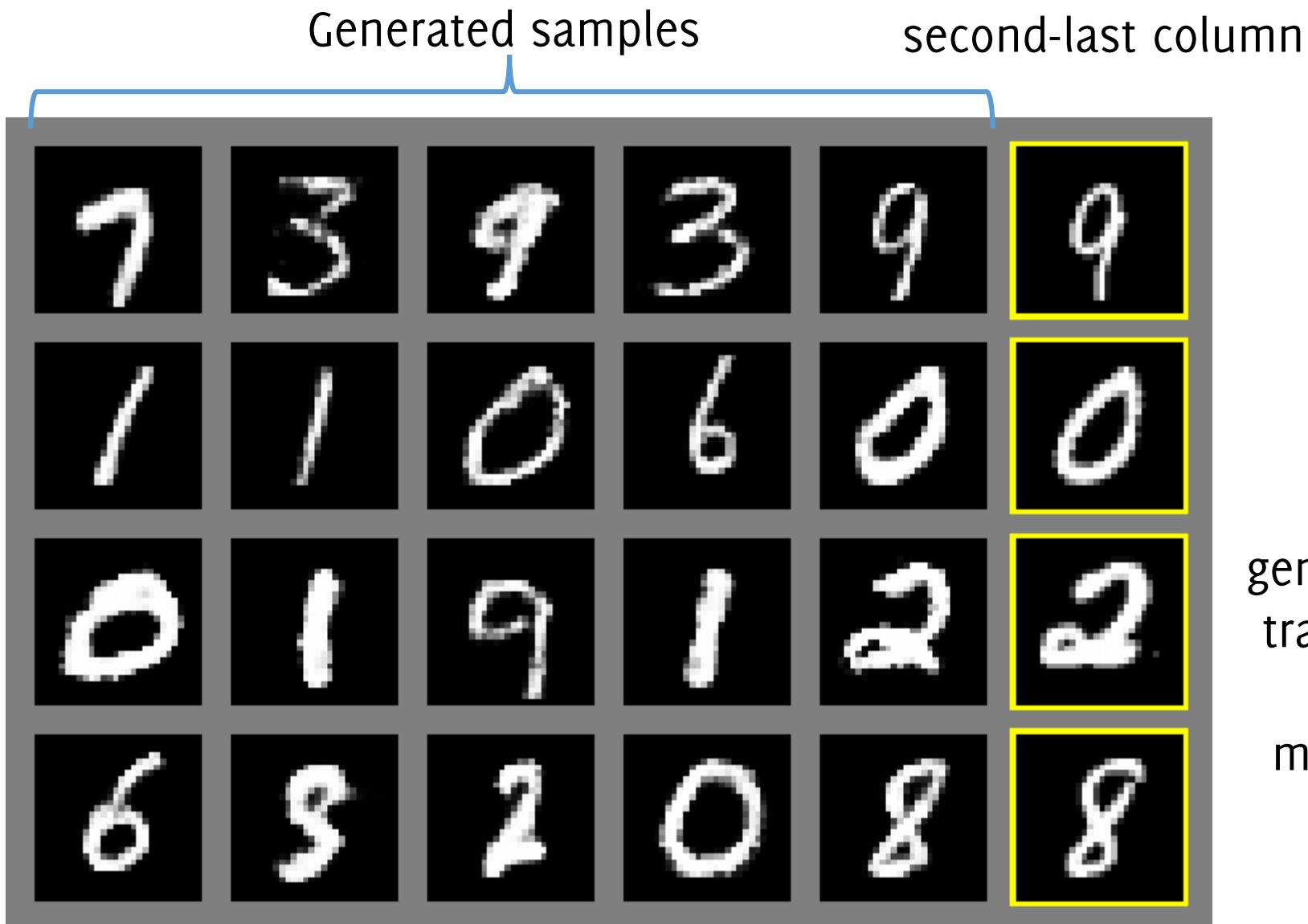
Remarks:

- 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 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

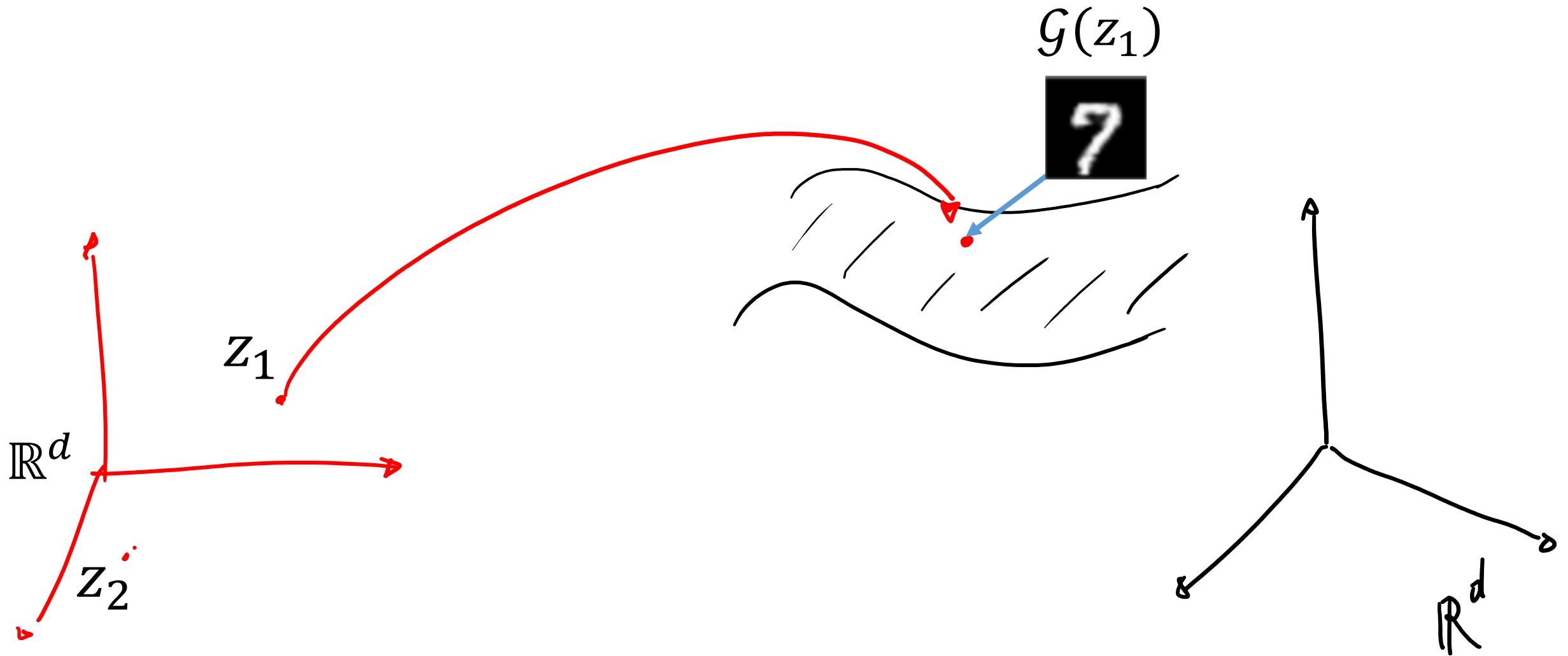
MNIST



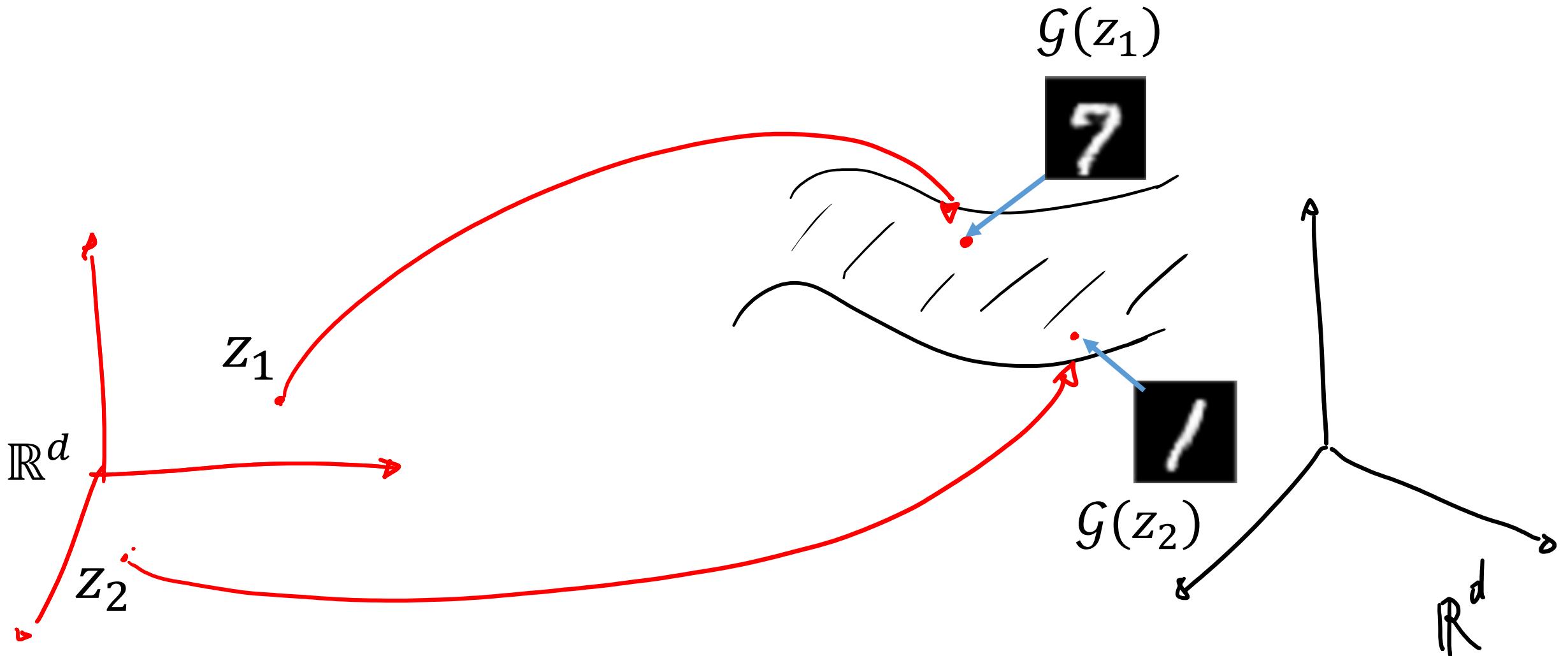
MNIST



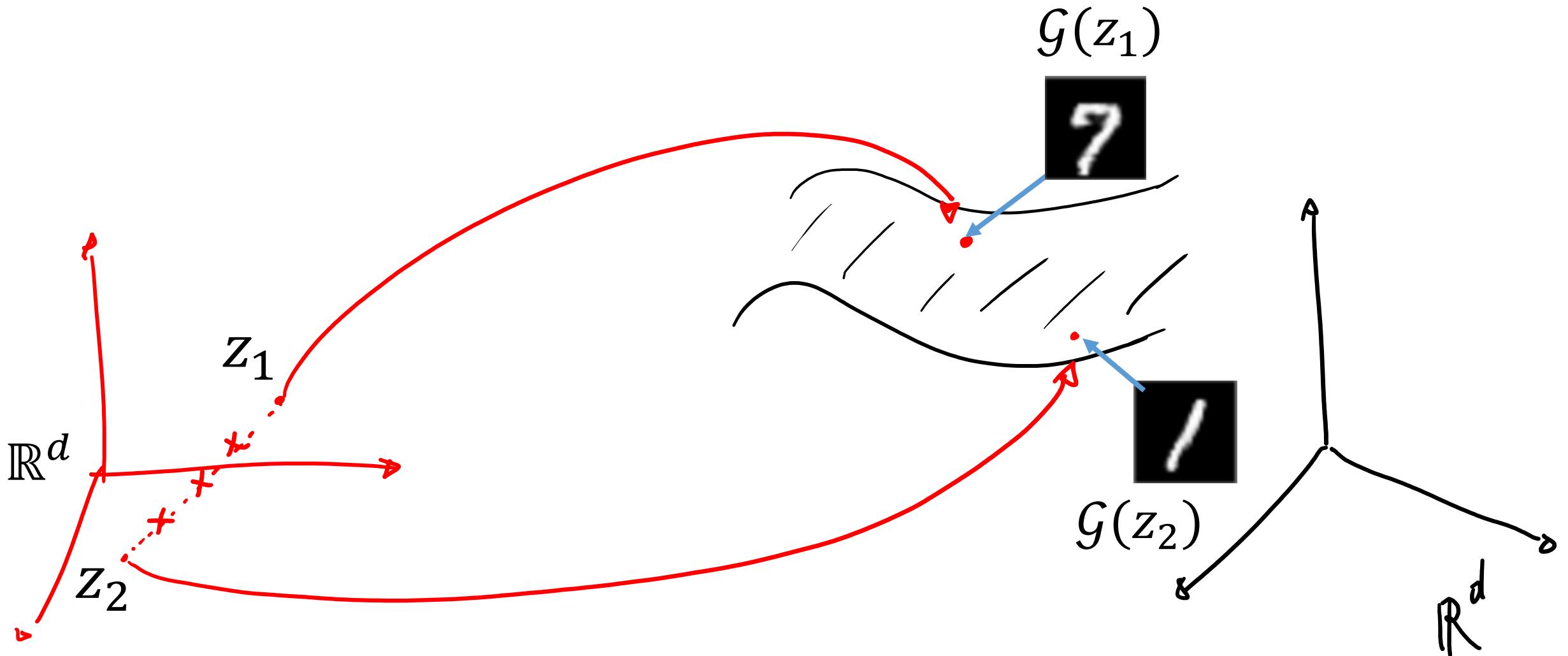
Interpolation experiment: the manifold was learned!



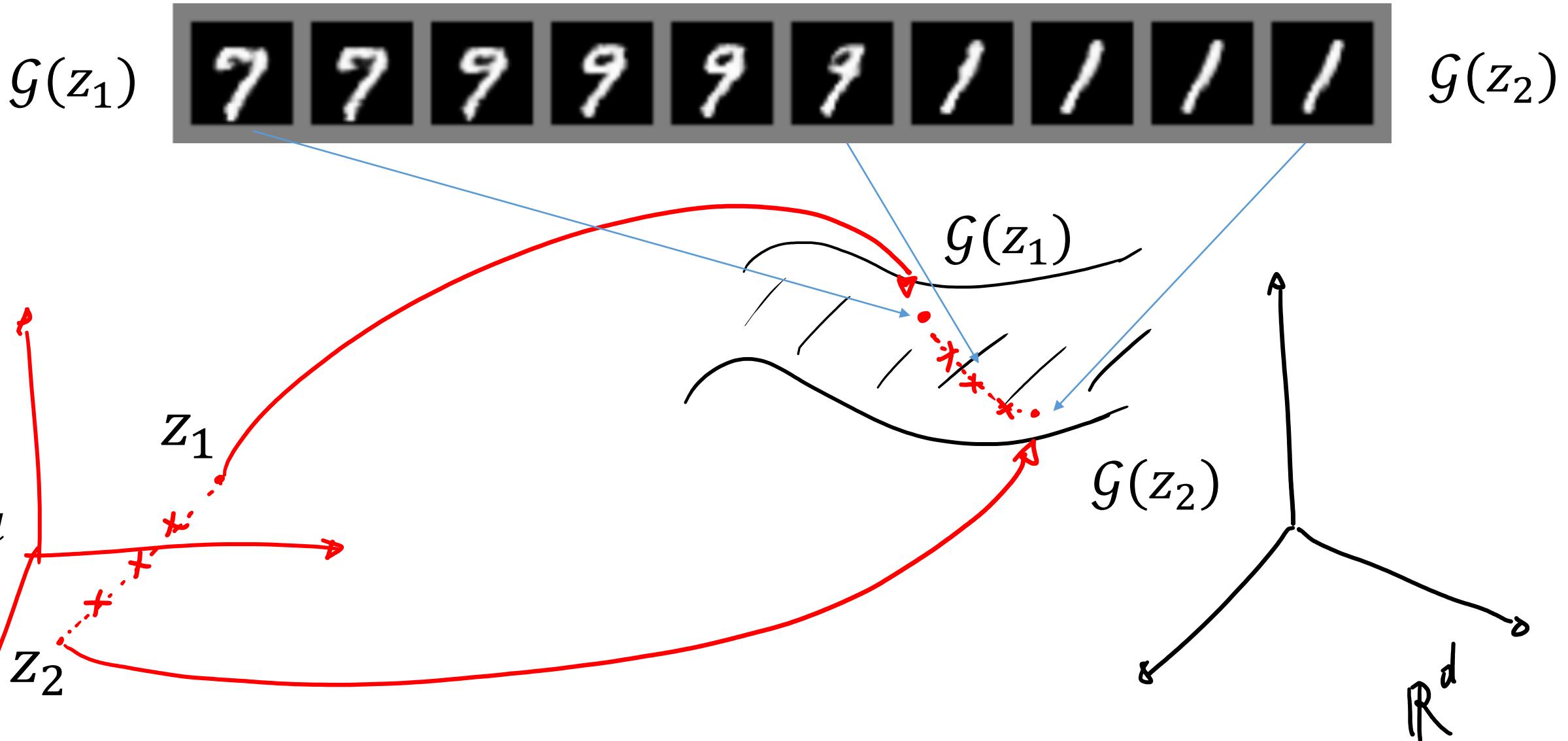
Interpolation experiment: the manifold was learned!



Interpolation experiment: the manifold was learned!



Interpolation experiment: the manifold was learned!



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

$\mathcal{G}(z_1)$



$\mathcal{G}(z_2)$

$\mathcal{G}(z_3)$

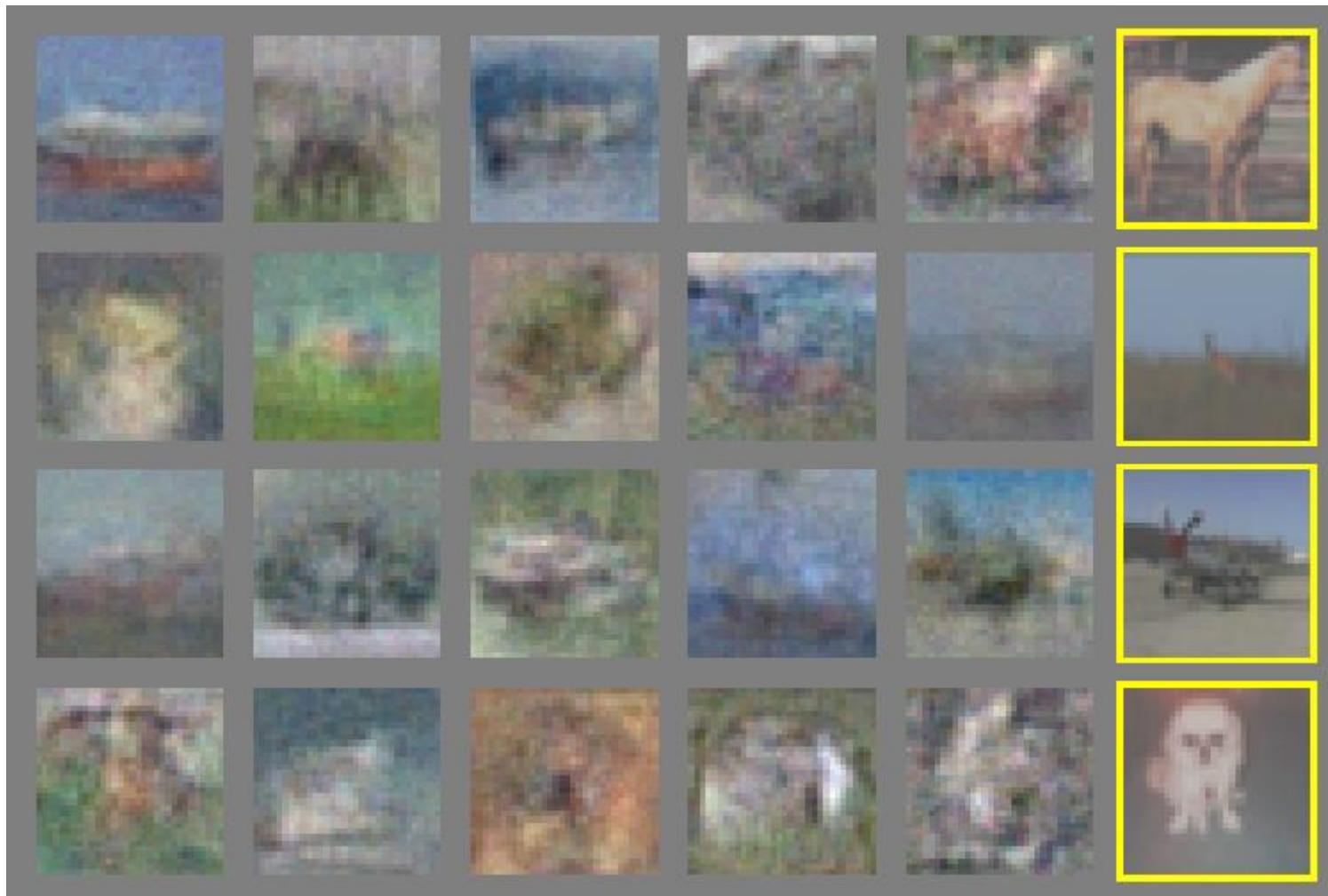


$\mathcal{G}(z_4)$

Toronto Face Database (TFD)



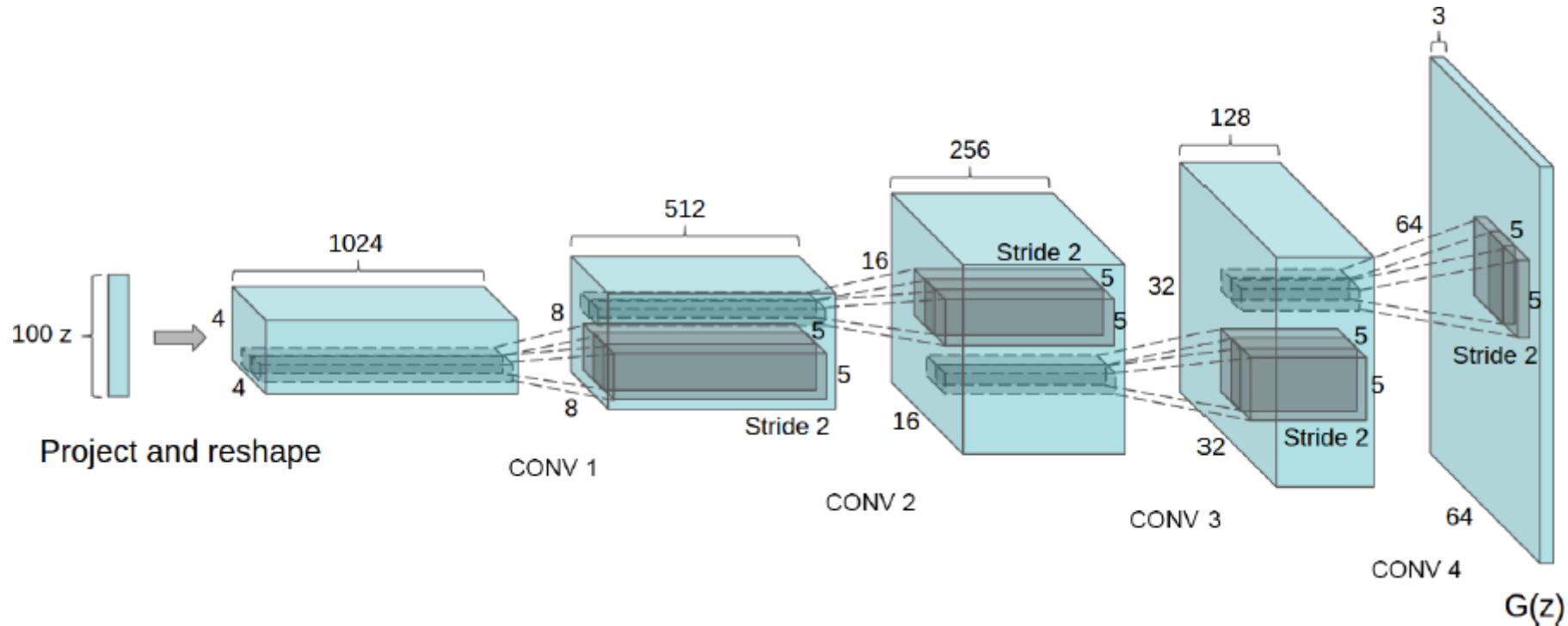
CIFAR-10



GANs have much improved in the last few years

DC-GAN: Deep Convolutional GANs

RGB Image



GANs have much improved in the last few years

Images generated after 1 training epochs

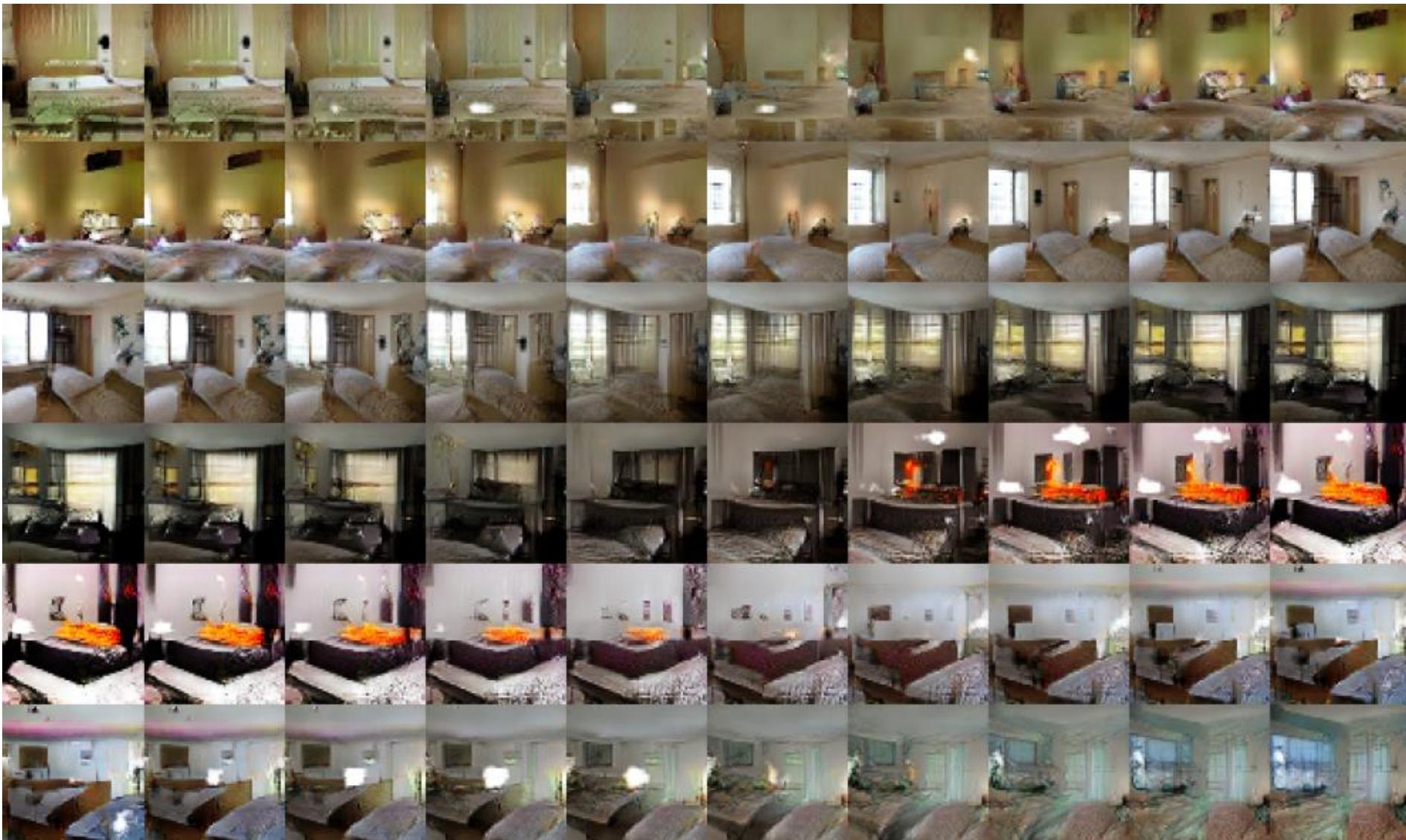


GANs have much improved in the last few years

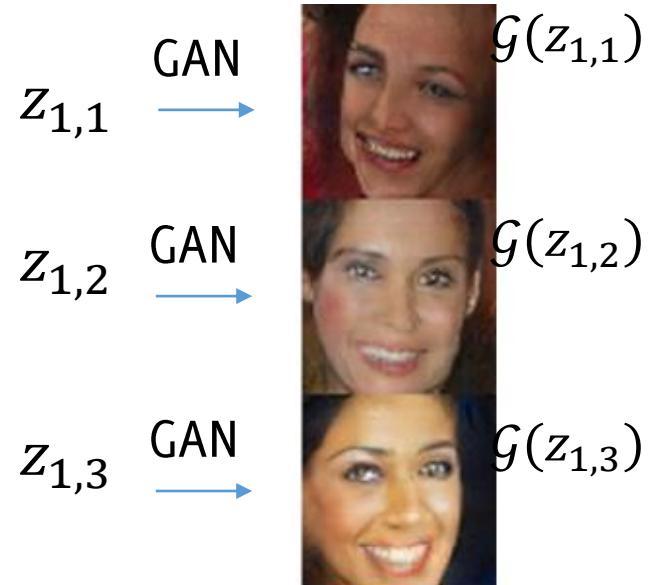
Images generated after 5 training epochs



Interpolation between a series of 9 random points

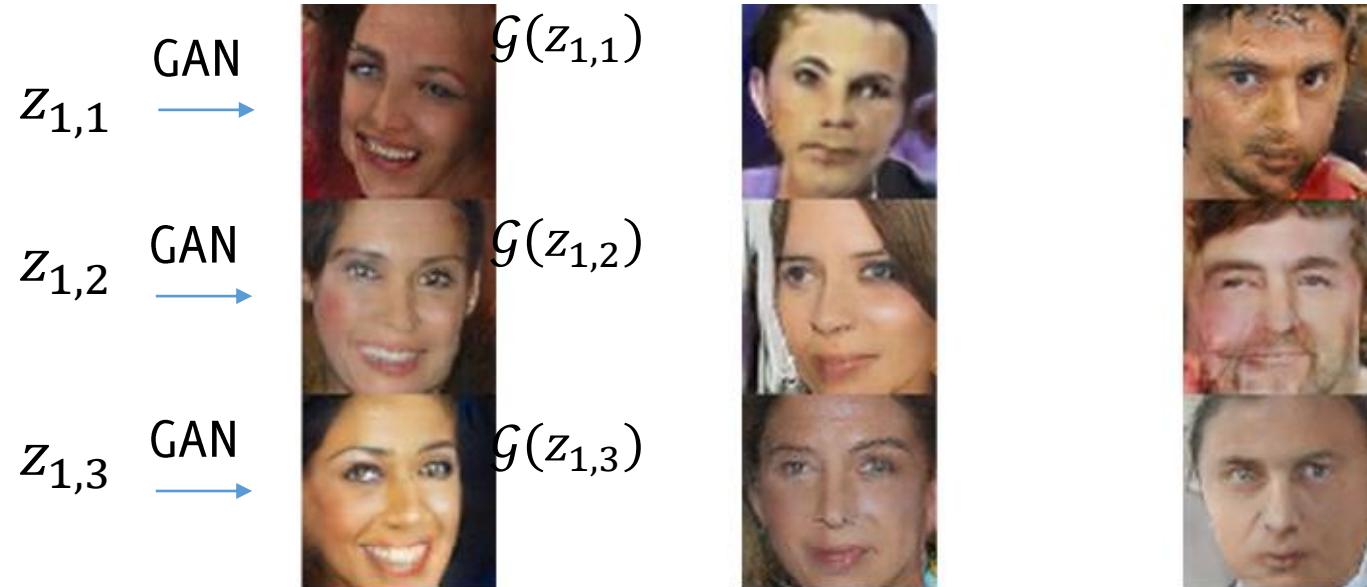


Vector Arithmetic



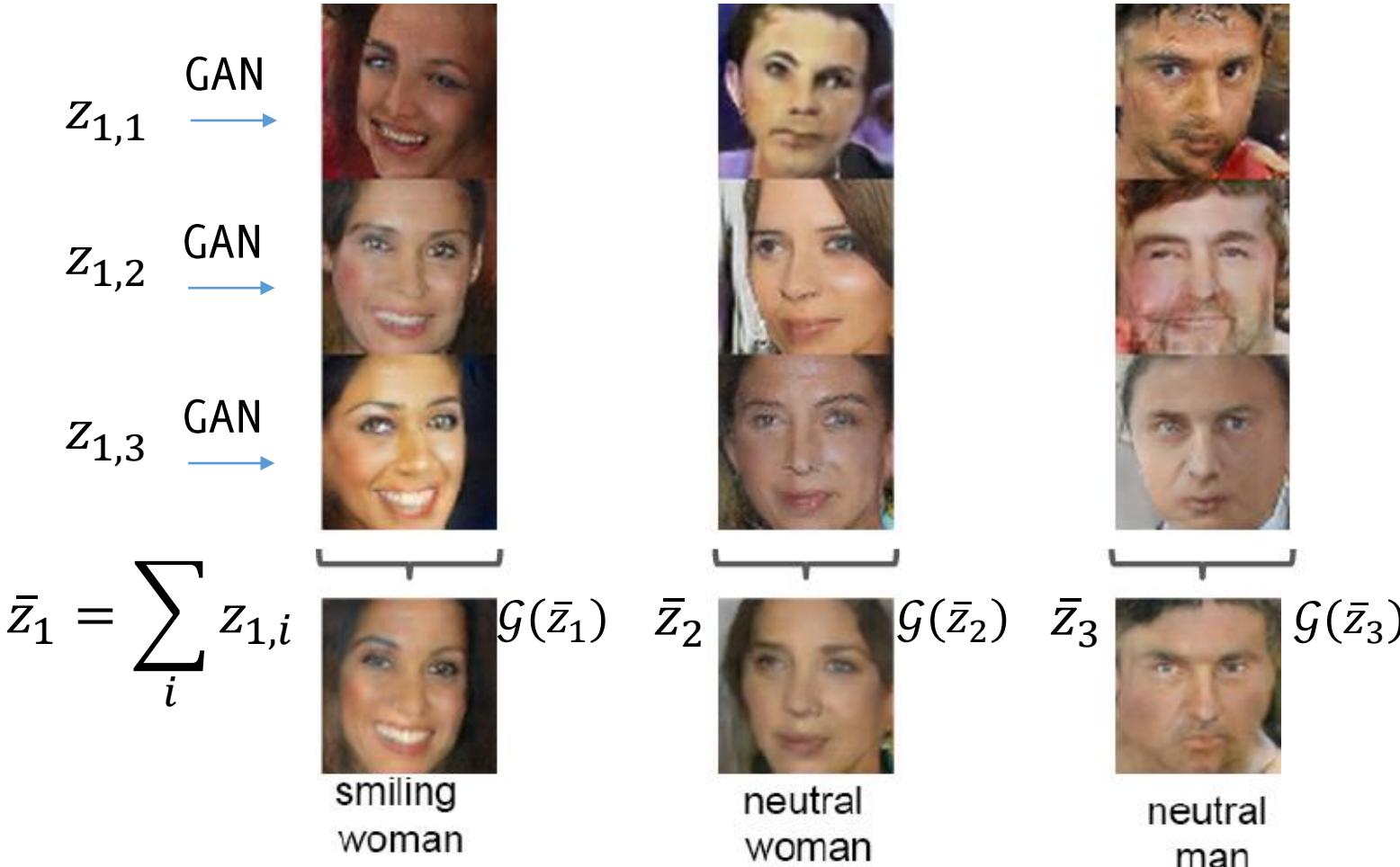
Take randomly
generated samples of
smiling women,
neutral women, and
neutral men

Vector Arithmetic



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

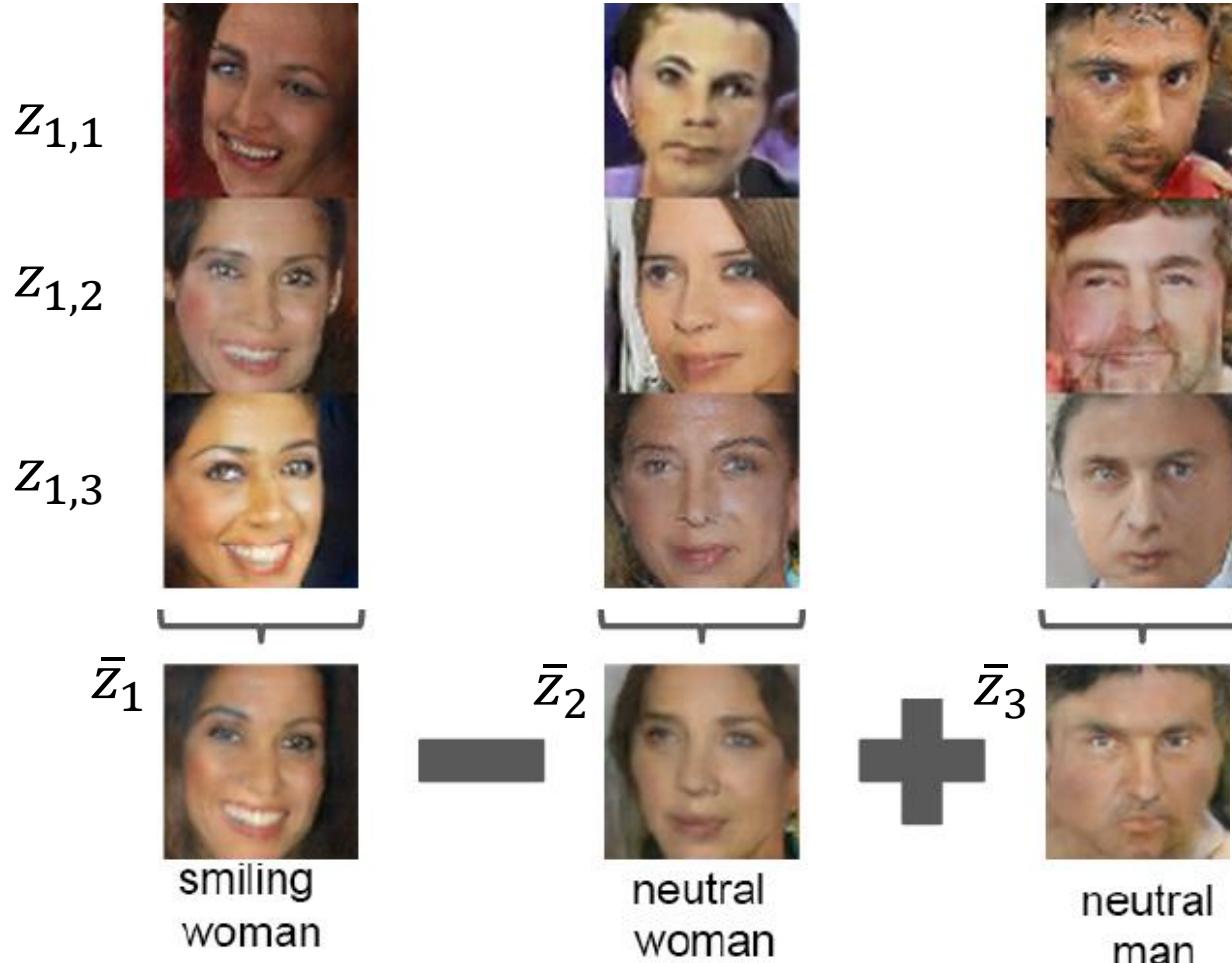
Vector Arithmetic



Average the corresponding noise seeds

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

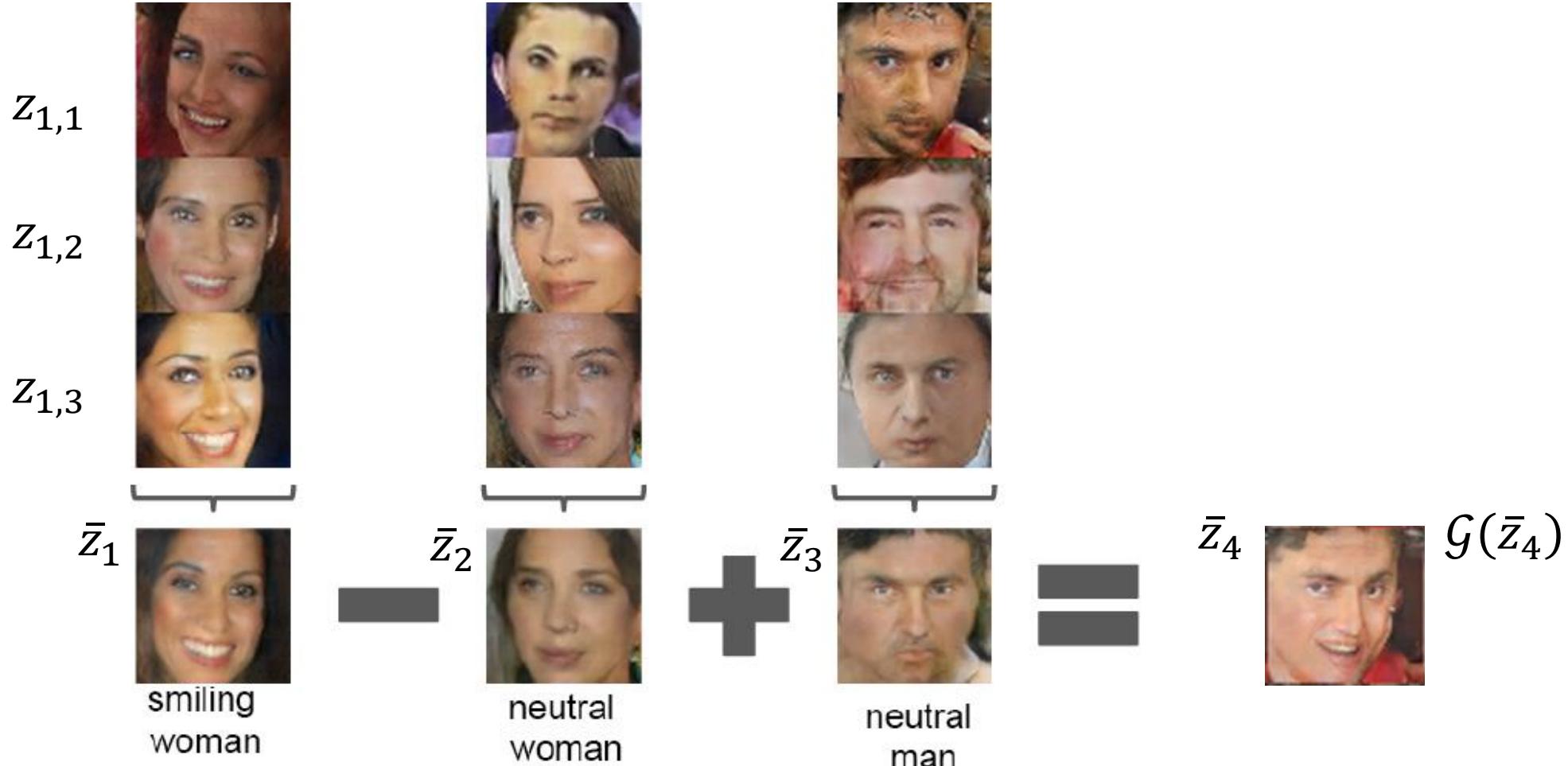
Vector Arithmetic



Perform some arithmetic

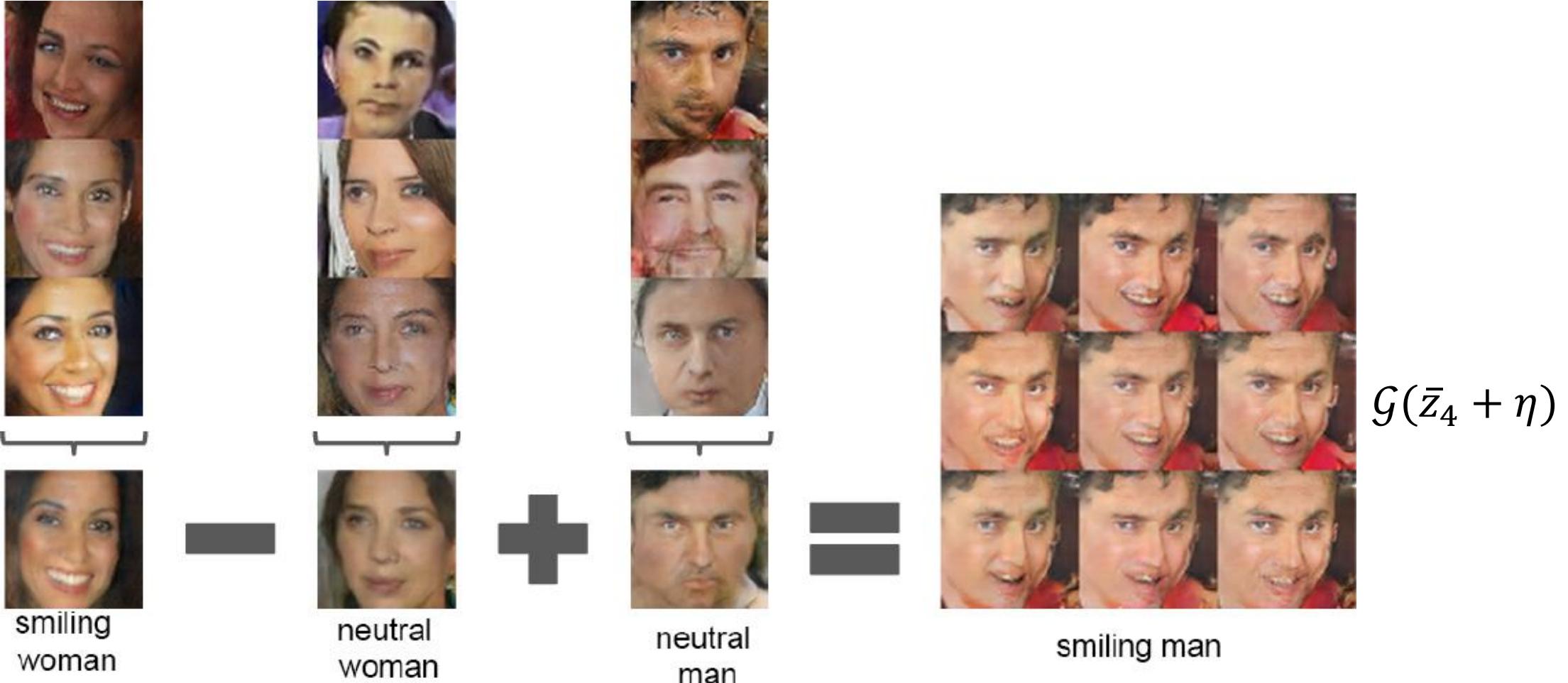
Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Vector Arithmetic



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

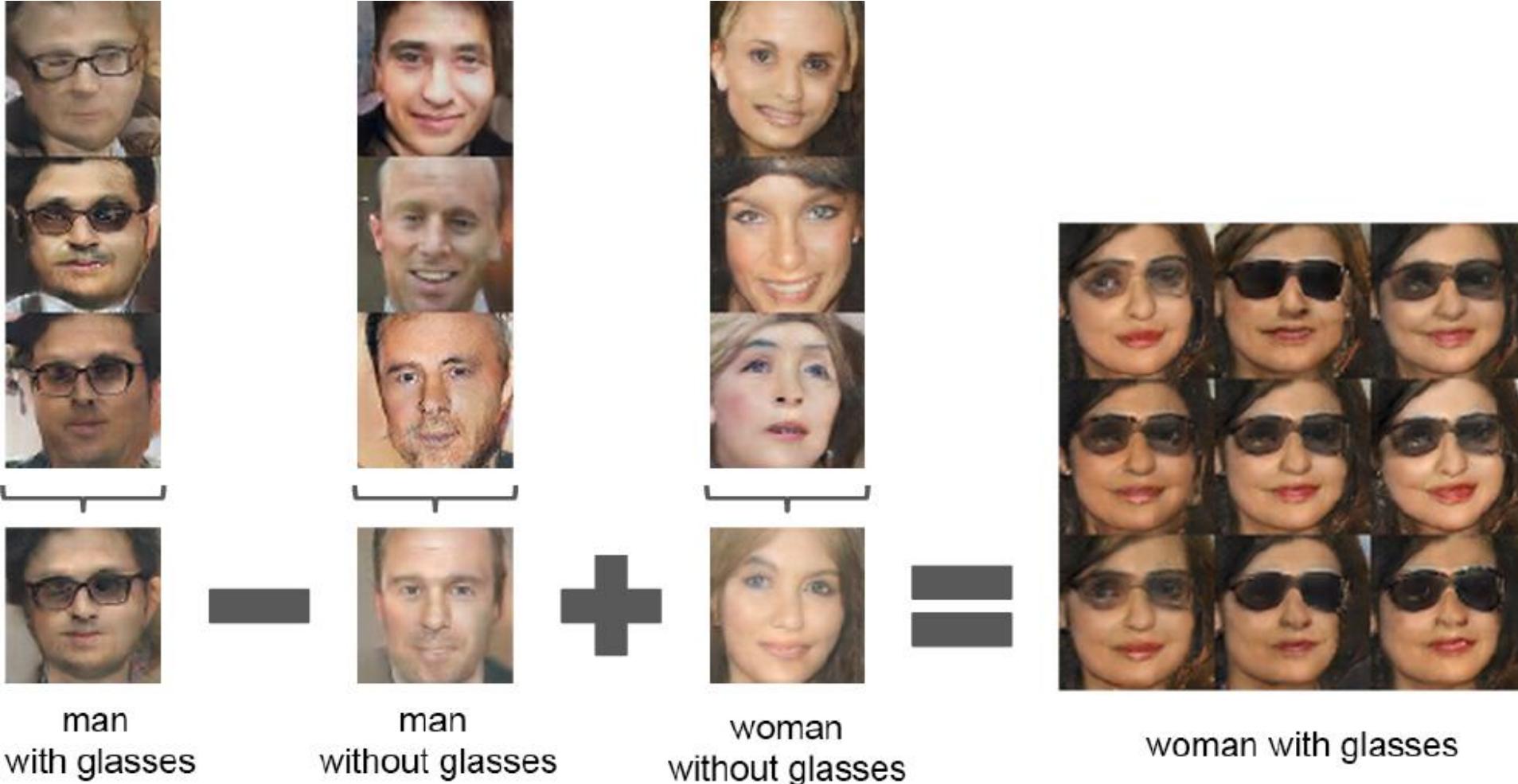
Vector Arithmetic



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

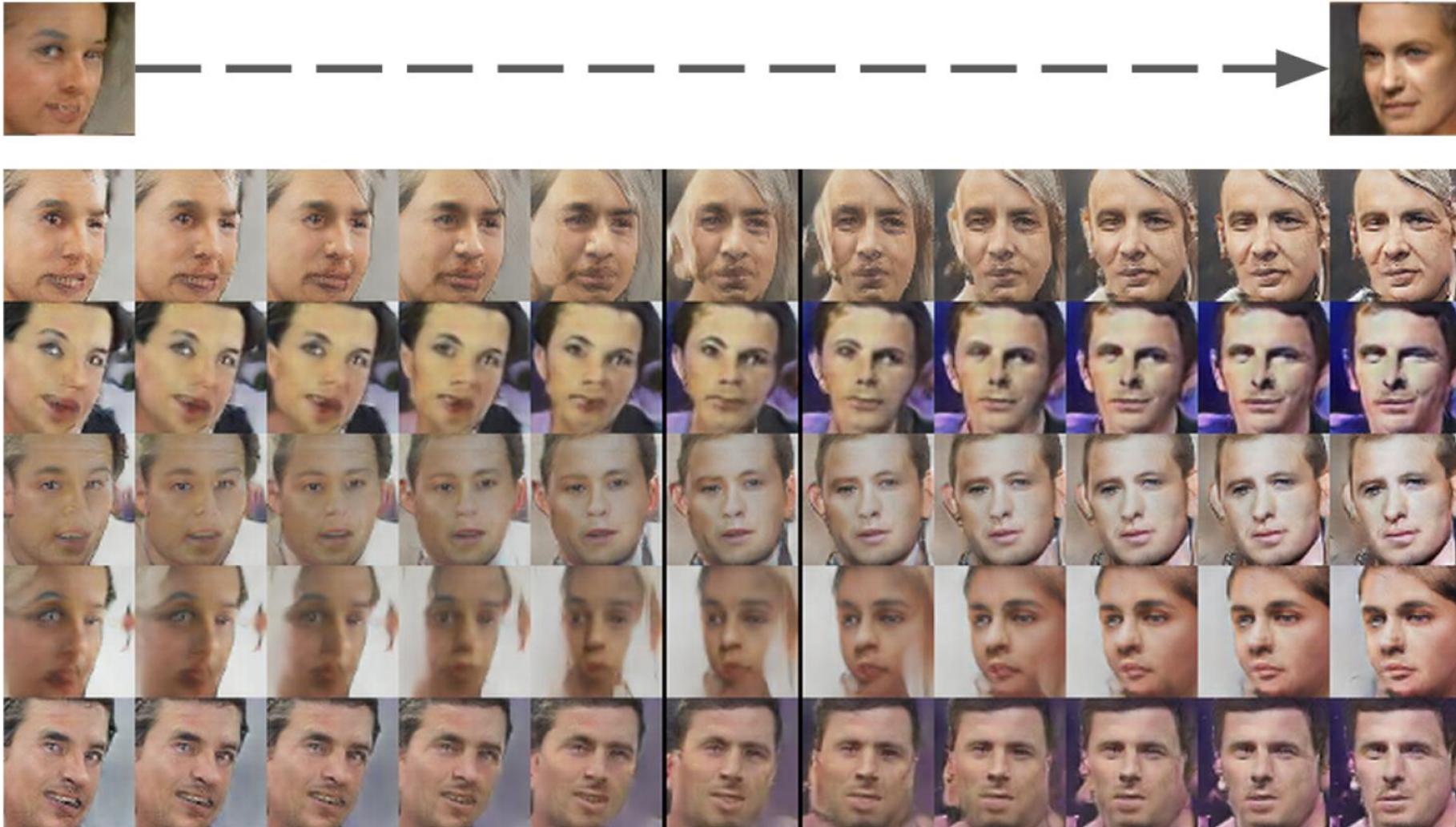


Similar example as word embedding

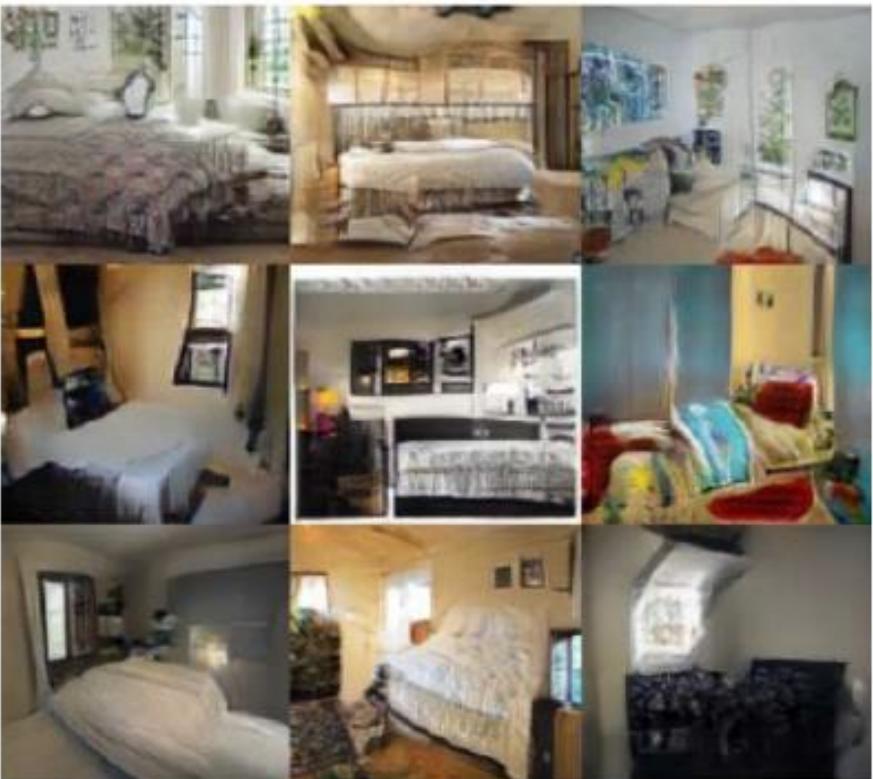
Radford, A., Metz, L., & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

Vector Arithmetic

Interpolation of view changes



GAN has been a very active research field



LSGAN, Zhu 2017.



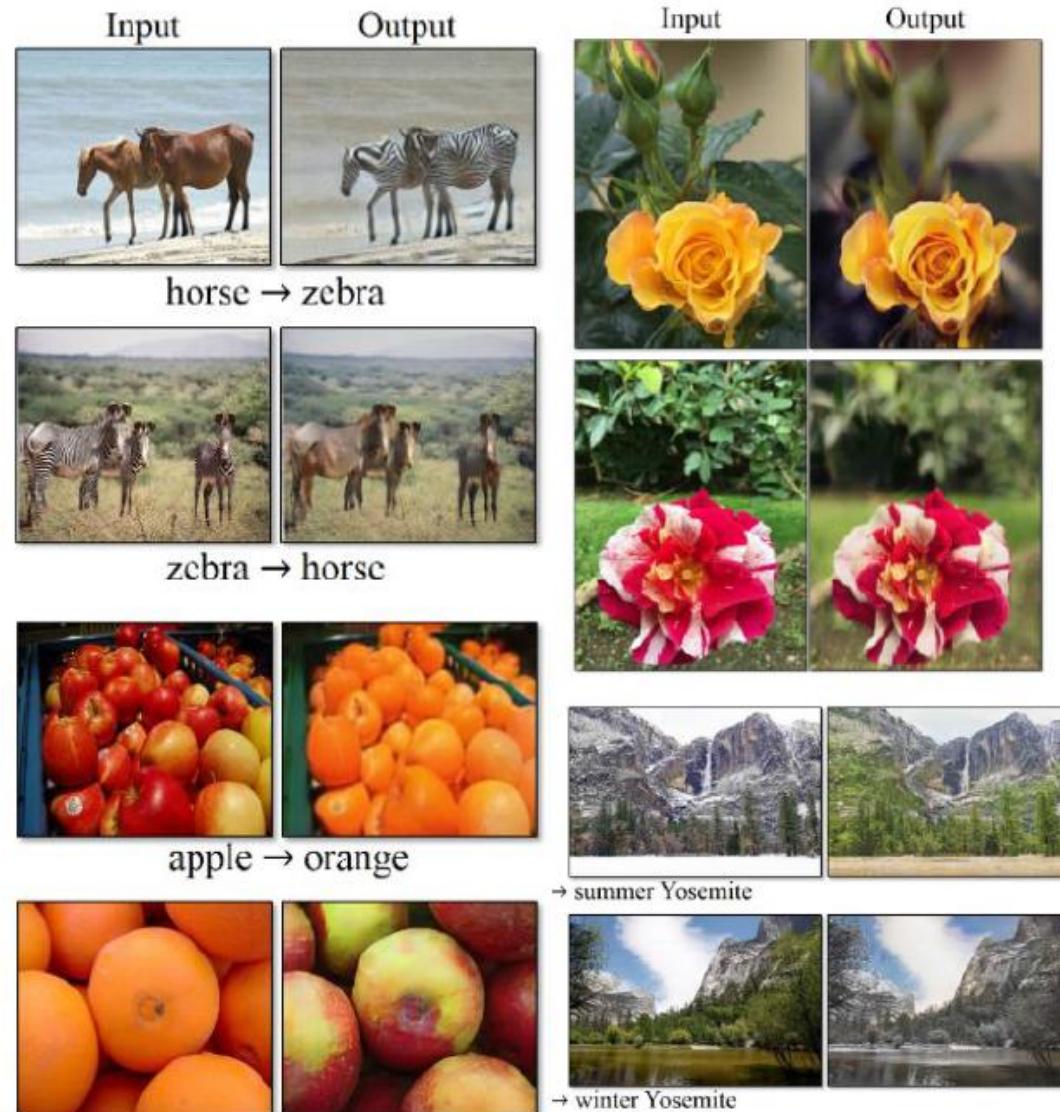
Wasserstein GAN,
Arjovsky 2017.
Improved Wasserstein
GAN, Gulrajani 2017.

GAN has been a very active research field



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>

this small bird has a pink breast and crown, and black primaries and secondaries.



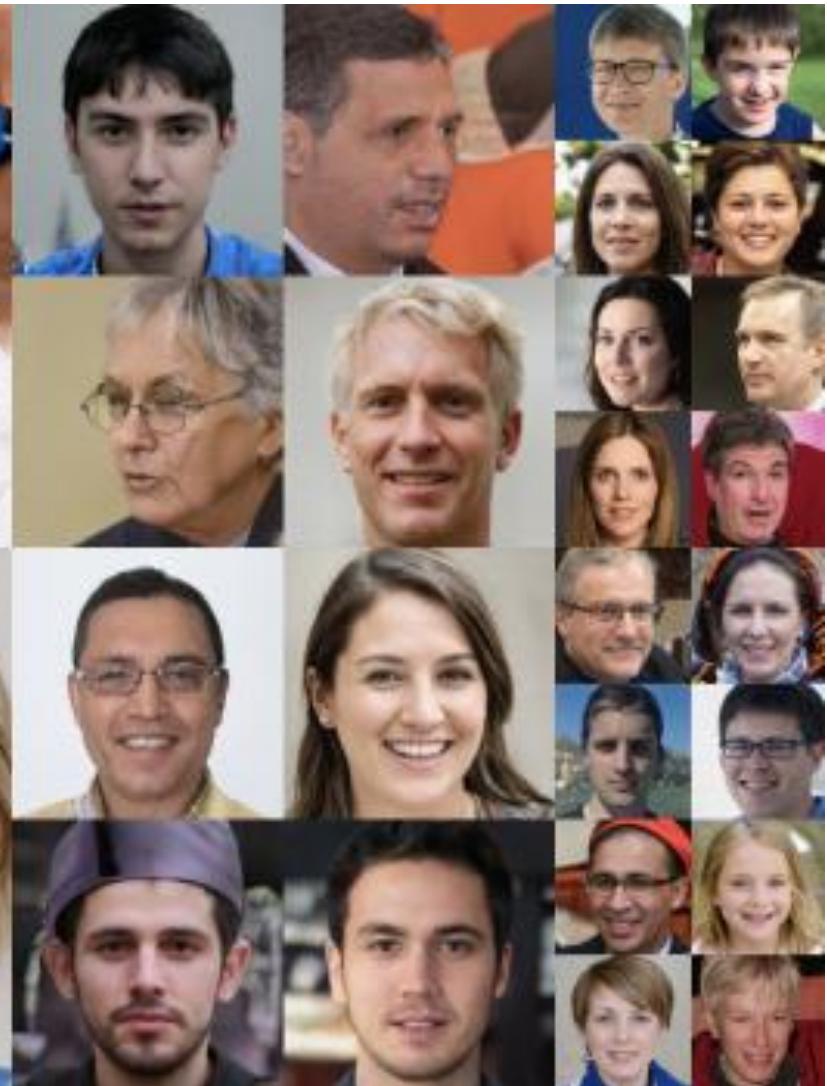
Reed et al. 2017.

this magnificent fellow is almost all black with a red crest, and white cheek patch.



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

Generative Adversarial Networks (these people do not exist)



DALL-E2:
generate images from text description

DALL-E2



a propaganda poster depicting a cat dressed as french emperor napoleon holding a piece of cheese



a teddy bear on a skateboard in times square

DALL-E2

DALL-E 2 a powerful machine learning models from OpenAI that generates digital images from natural language descriptions

It is not based on GANs but on diffusion models, a more recent generative model.

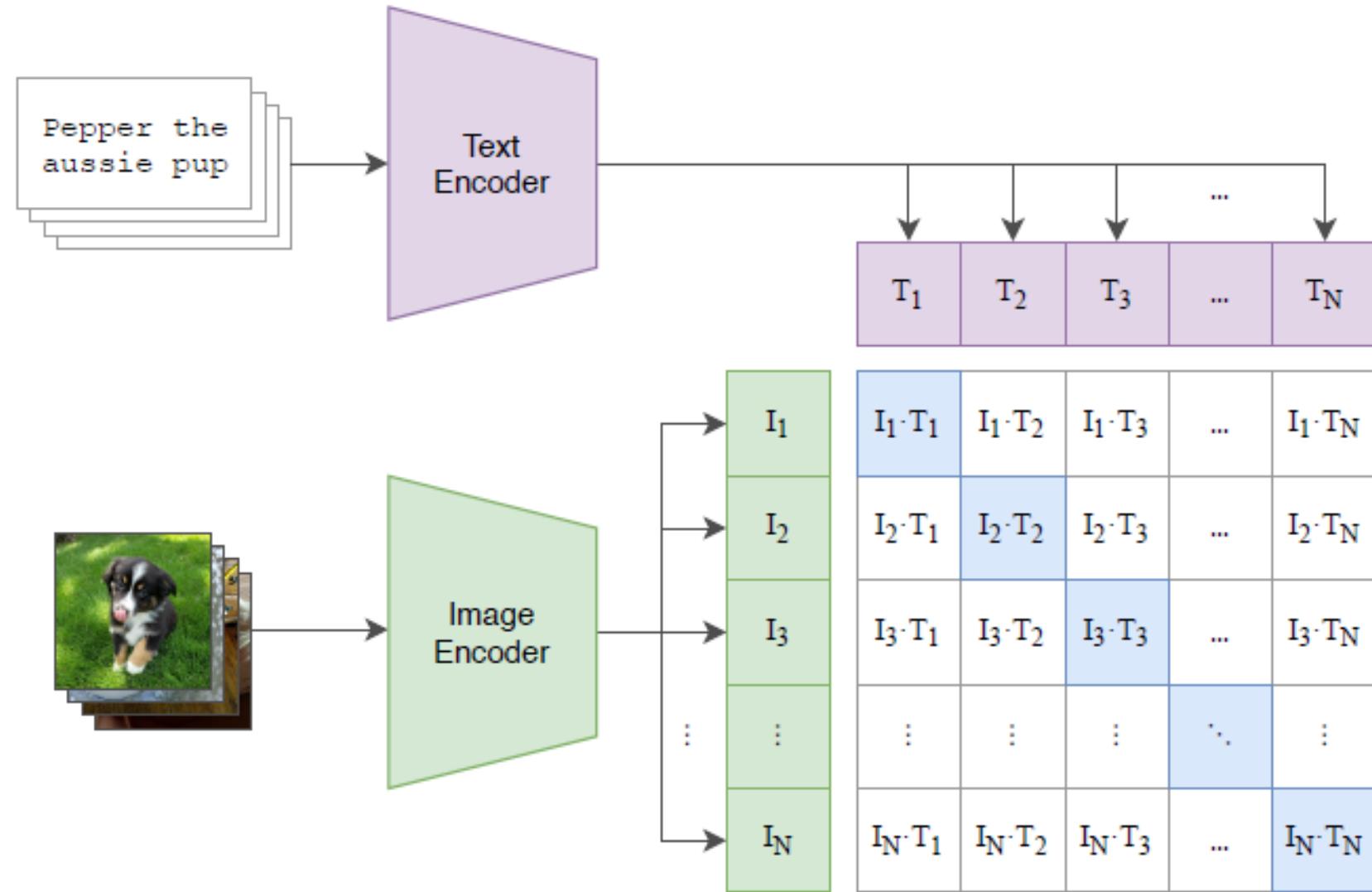
... but how to feed text to a generative model?

We need some help from an external model: CLIP

CLIP

Clip is a huge model, trained from image captions to generate consistent representations of both images and text

CLIP is trained by contrastive loss from a training set of images and their caption

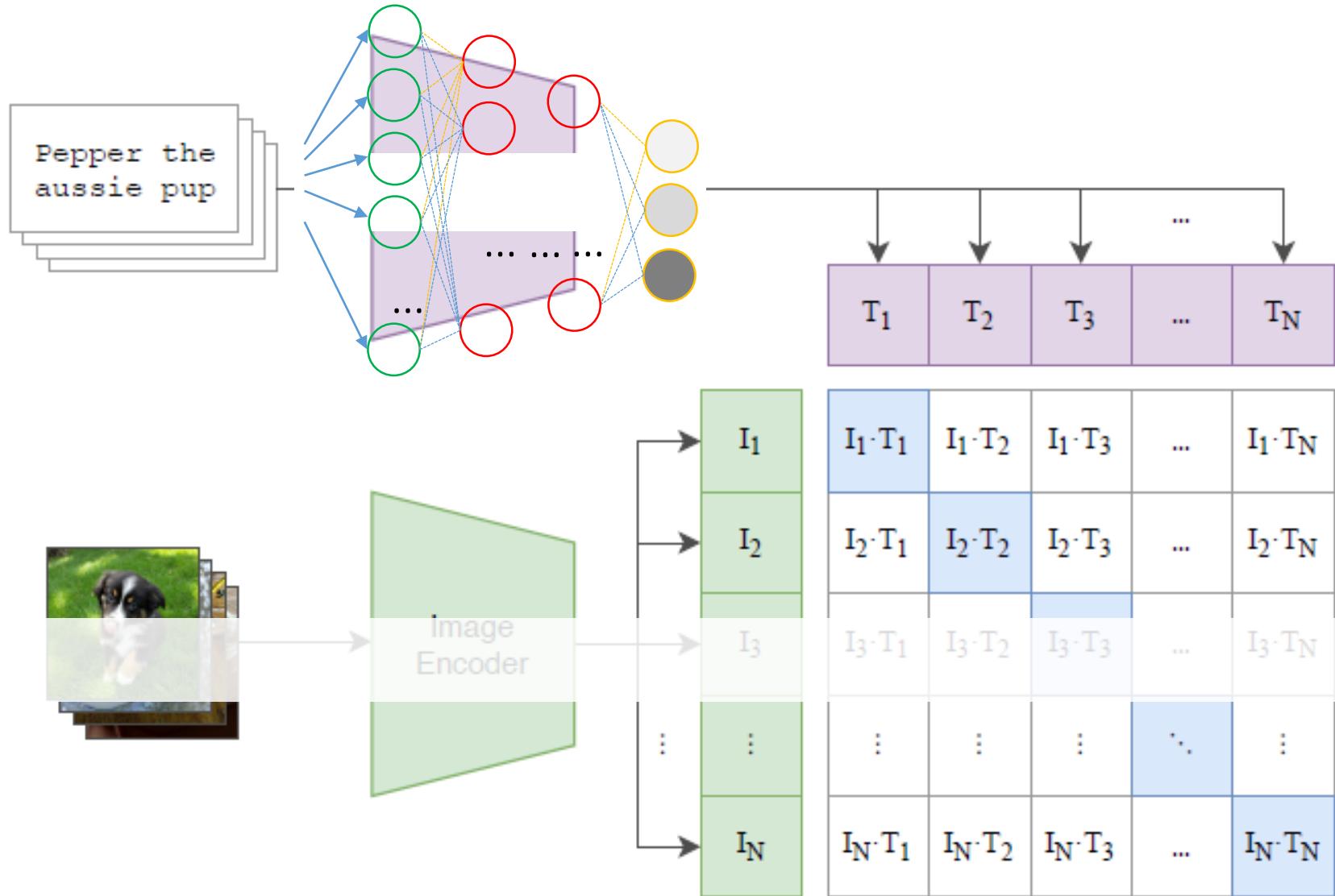


CLIP

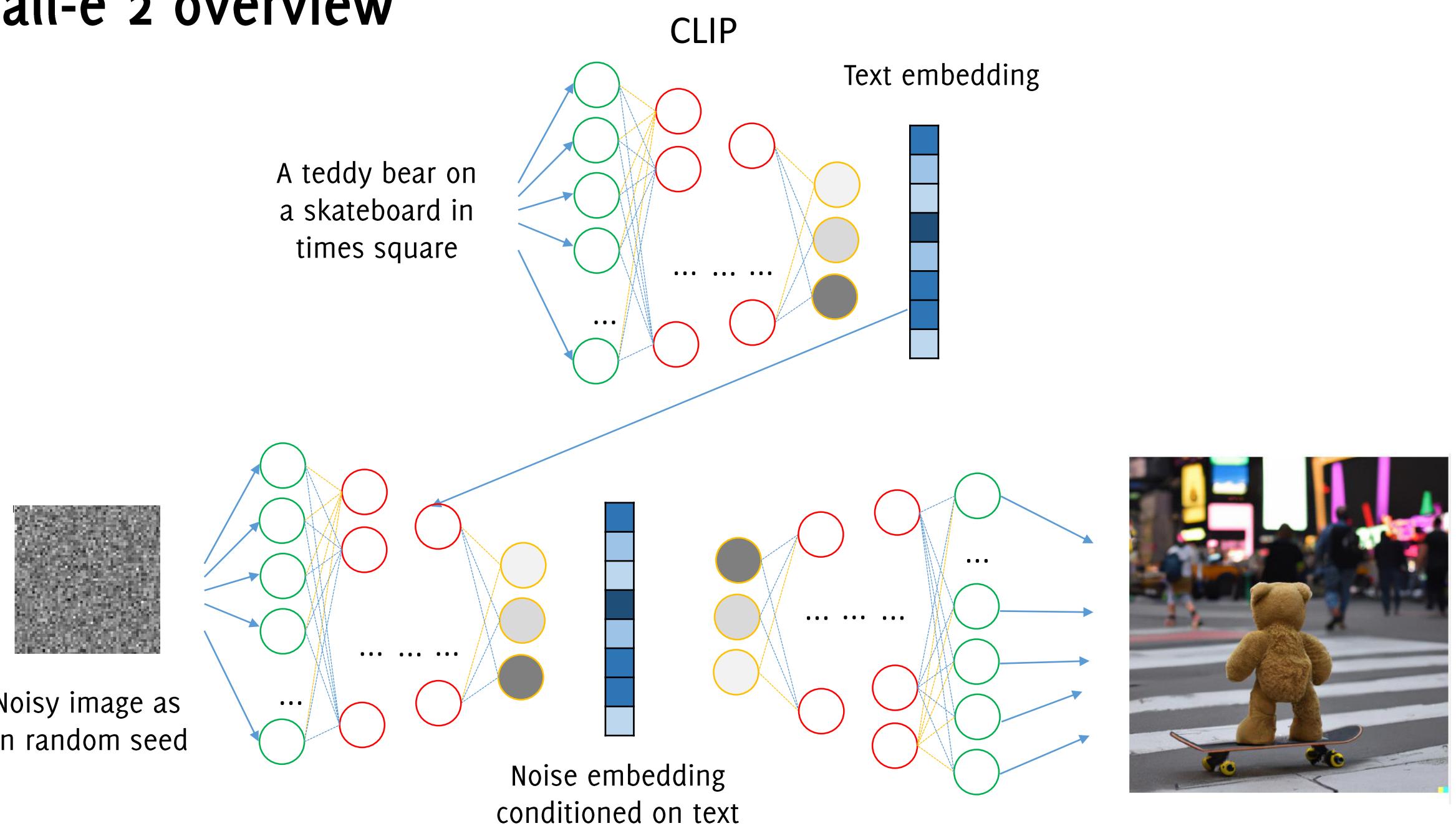
Clip is a huge model, trained from image captions to generate consistent representations of both images and text

CLIP is trained by contrastive loss from a training set of images and their caption

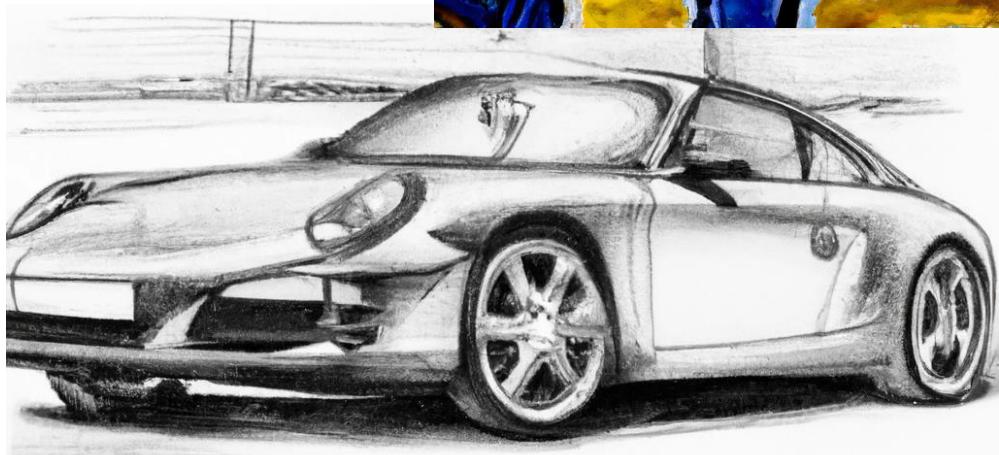
We just keep the text encoder from CLIP



Dall-e 2 overview

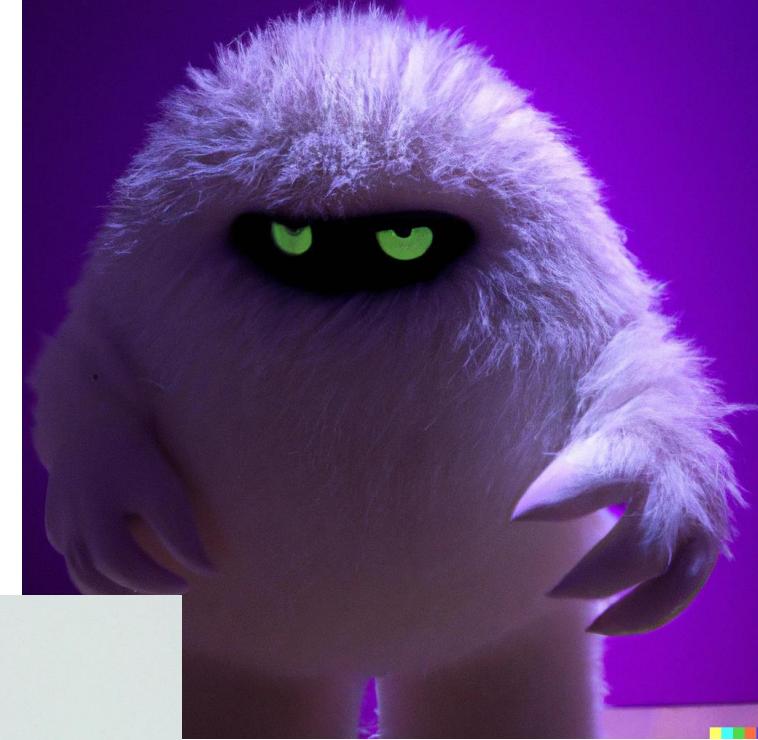


A van Gogh style painting of an American football player



A hand drawn sketch of a Porsche 911

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



A handpalm with a tree growing on top of it

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 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!

GAN for Anomaly Detection

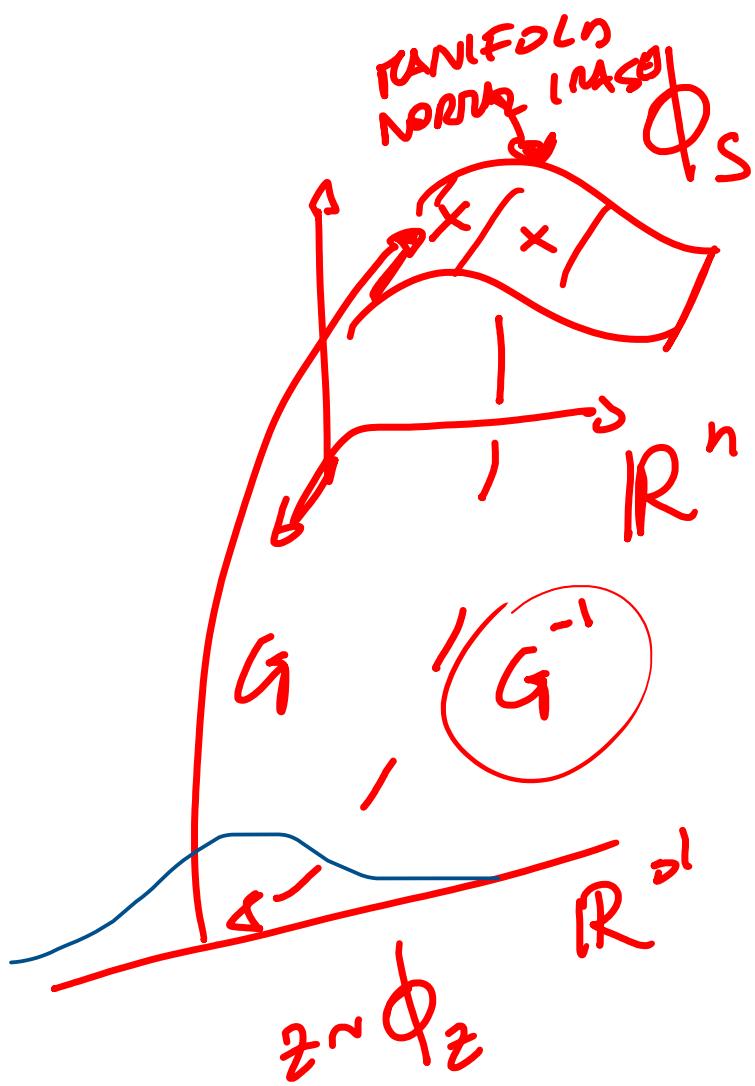
Thanks to Stefano Pecchia, former AN2DL student!

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



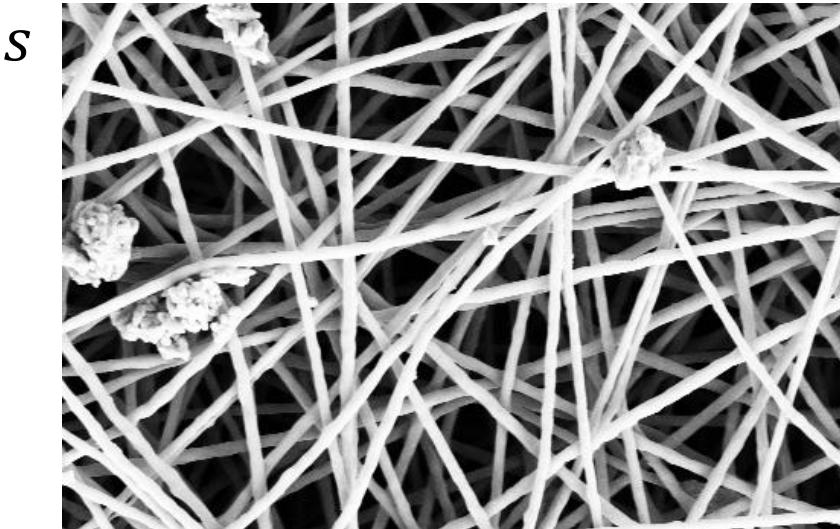
Anomaly Detection in images

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 Ω

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

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



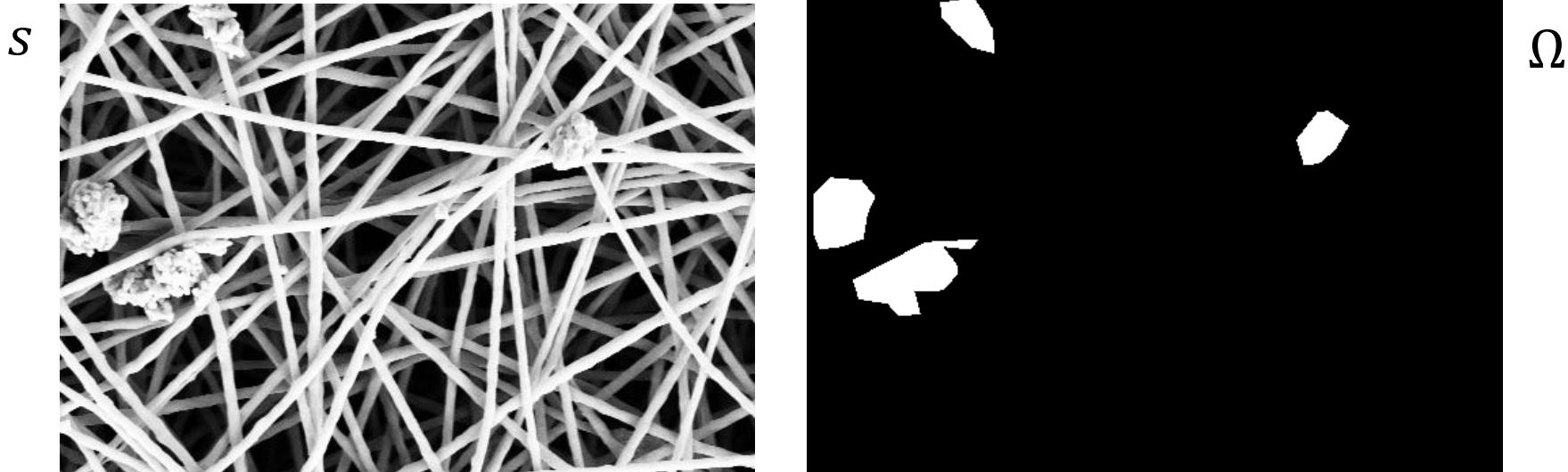
Anomaly Detection in images

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.

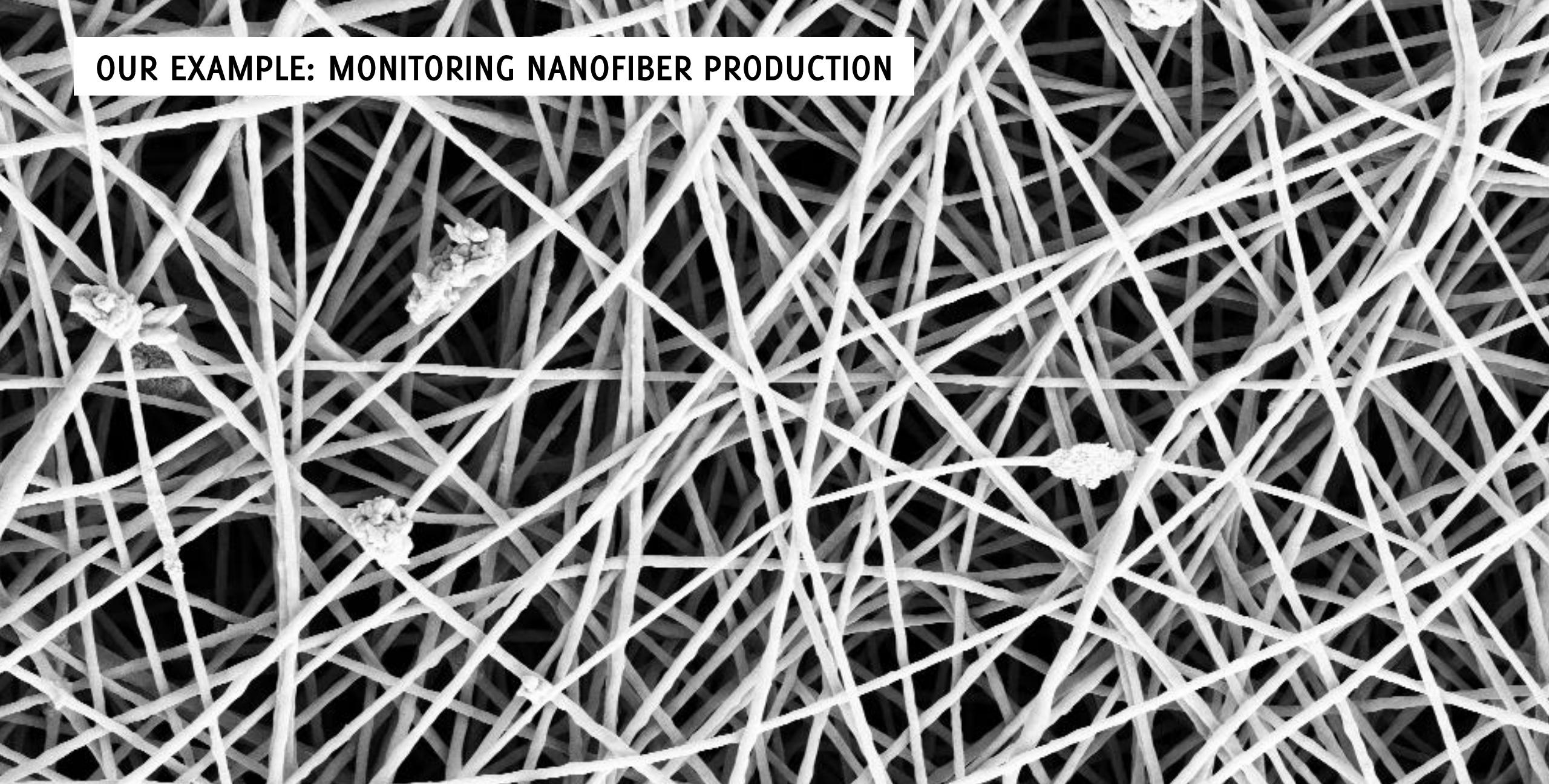
We want to locate any anomalous region in s , i.e. estimating the anomaly mask Ω

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

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

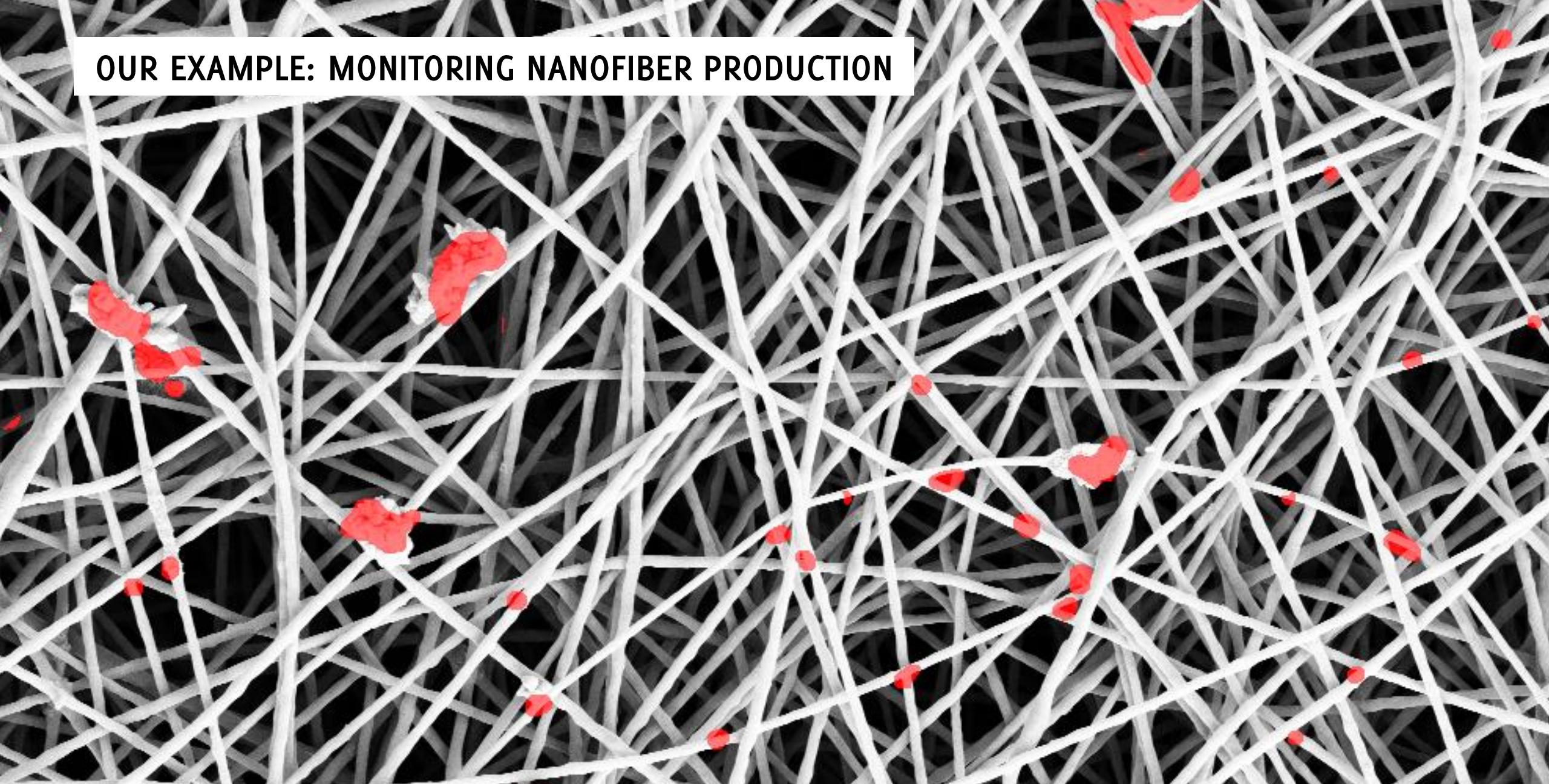


OUR EXAMPLE: MONITORING NANOFIBER PRODUCTION



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

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

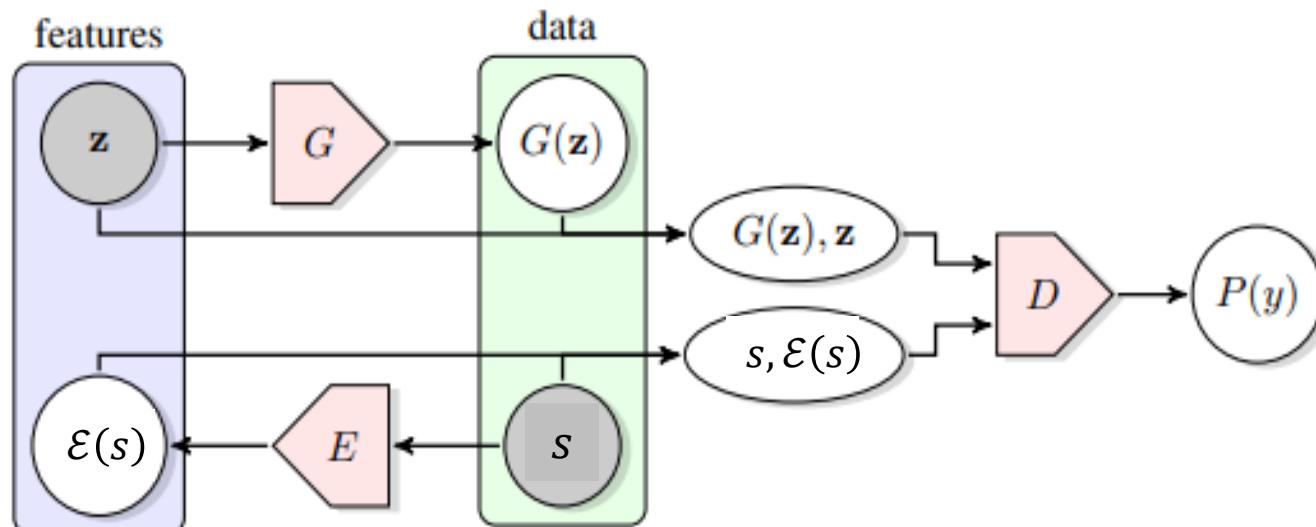
Unfortunately, it is not possible to invert \mathcal{G} ... some neural network need to be trained for this purpose!

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) $G(\mathcal{E}(s))$
- The discriminator \mathcal{D} takes as input (*image, latent repr.*) as in conditional GAN

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



BidirectionalGANs (BiGANs) and Anomaly Detection

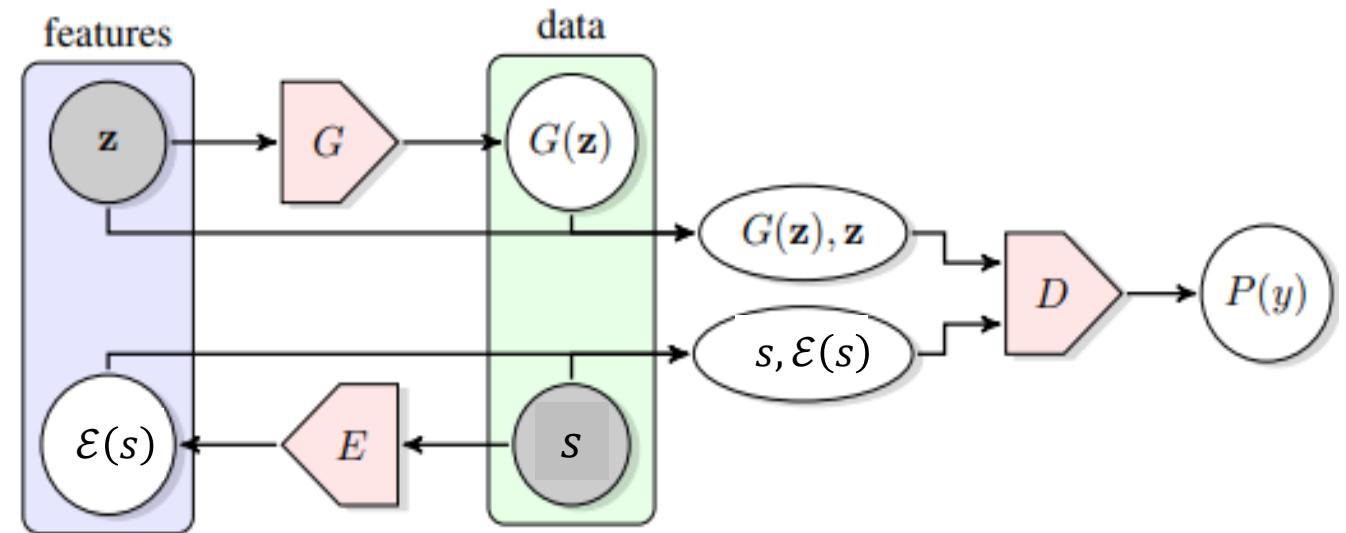
In principle, the encoder $\mathcal{E}(\cdot)$ can be used for anomaly detection by computing the likelihood of $\phi_z(\mathcal{E}(s))$ and consider as anomalous all the images s corresponding to a **low likelihood** (provided that ϕ_z was not a uniform distribution)

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

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

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

since the discriminator will consider the anomalous sample as fake.

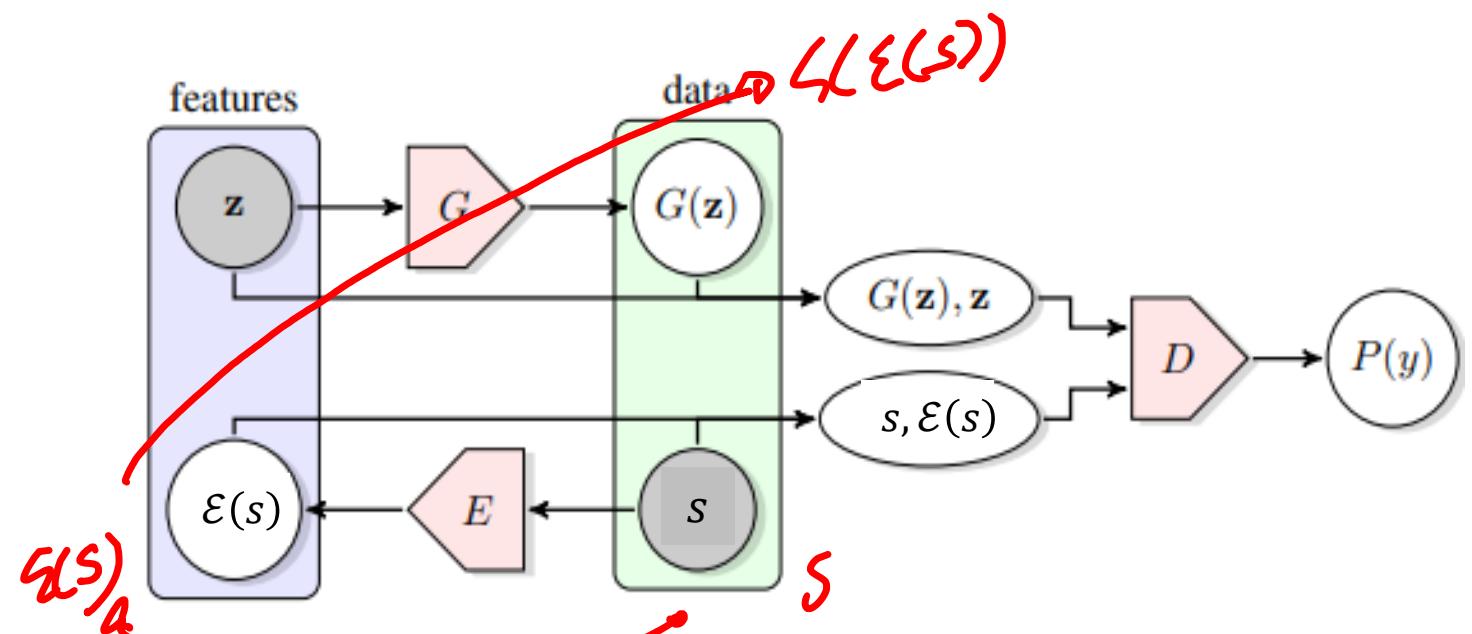


Anomaly detection with BidirectionalGANs (BiGANs)

However, there are more effective anomaly scores

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

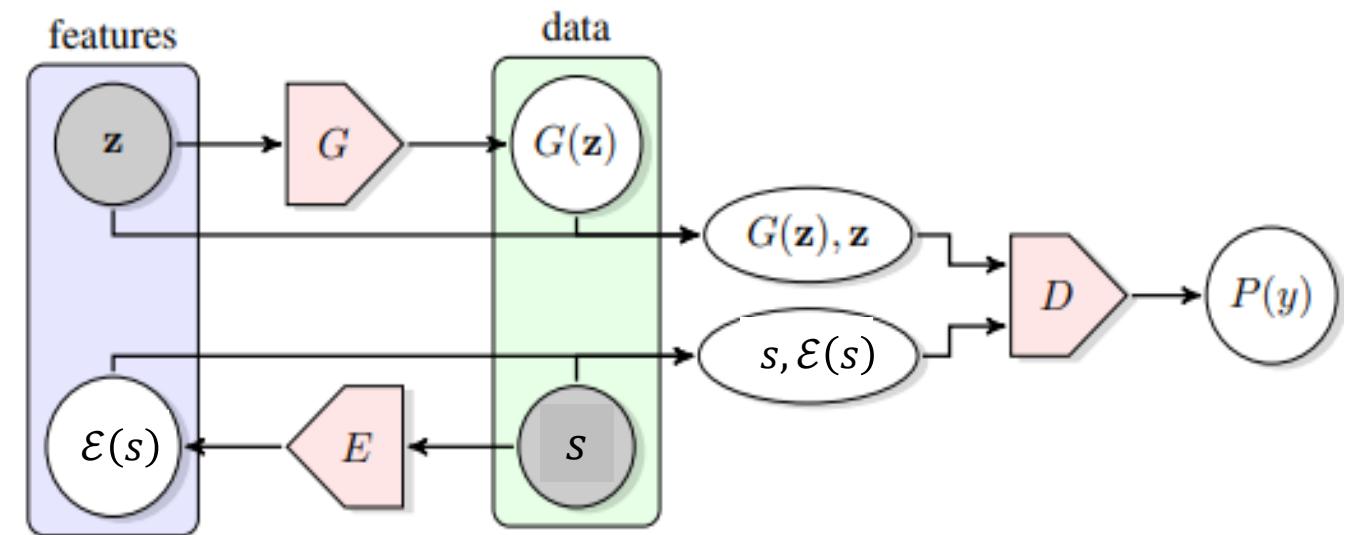
Reconstruction Loss Distance among latent representations of \mathcal{D} .
 f is a CNN extracting a latent representation



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

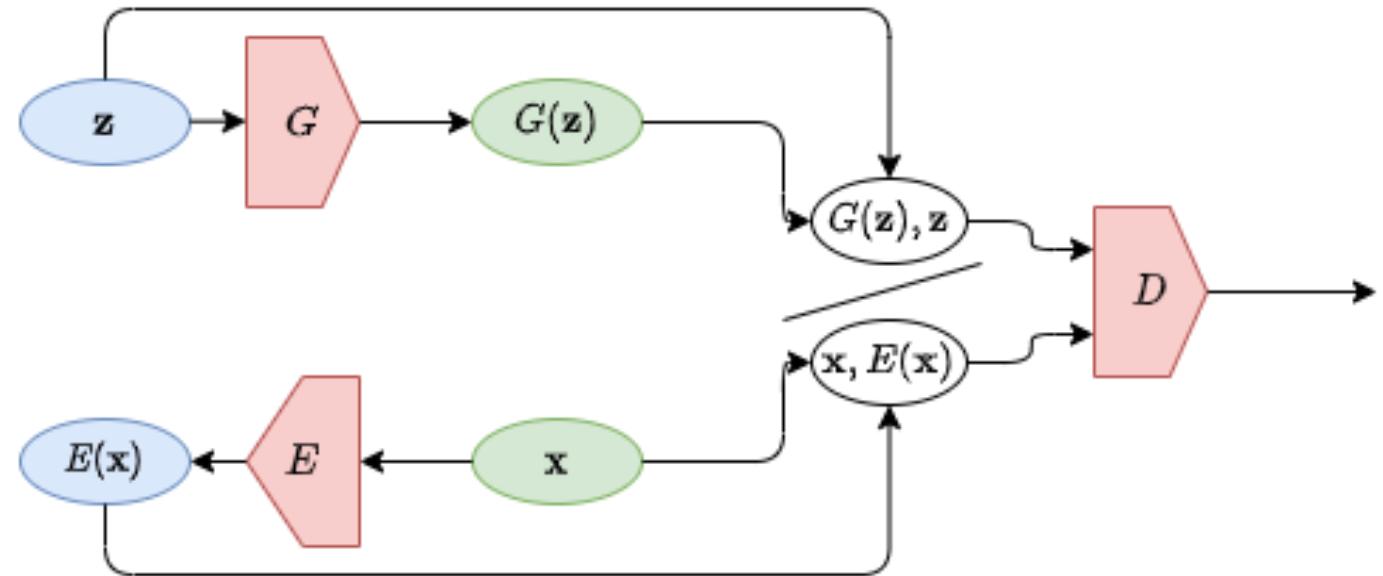


Fully Convolutional Anomaly Detection by GANs

Training

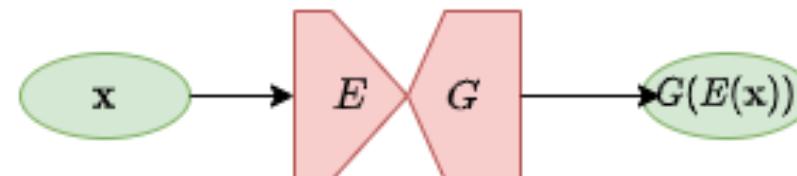
- All the layers are made fully convolutional (much more efficient processing)
- LS-losses used to train the model (this improves stability)

Adversarial Loss



Training

Reconstruction Penalty



Fully Convolutional Anomaly Detection by GANs

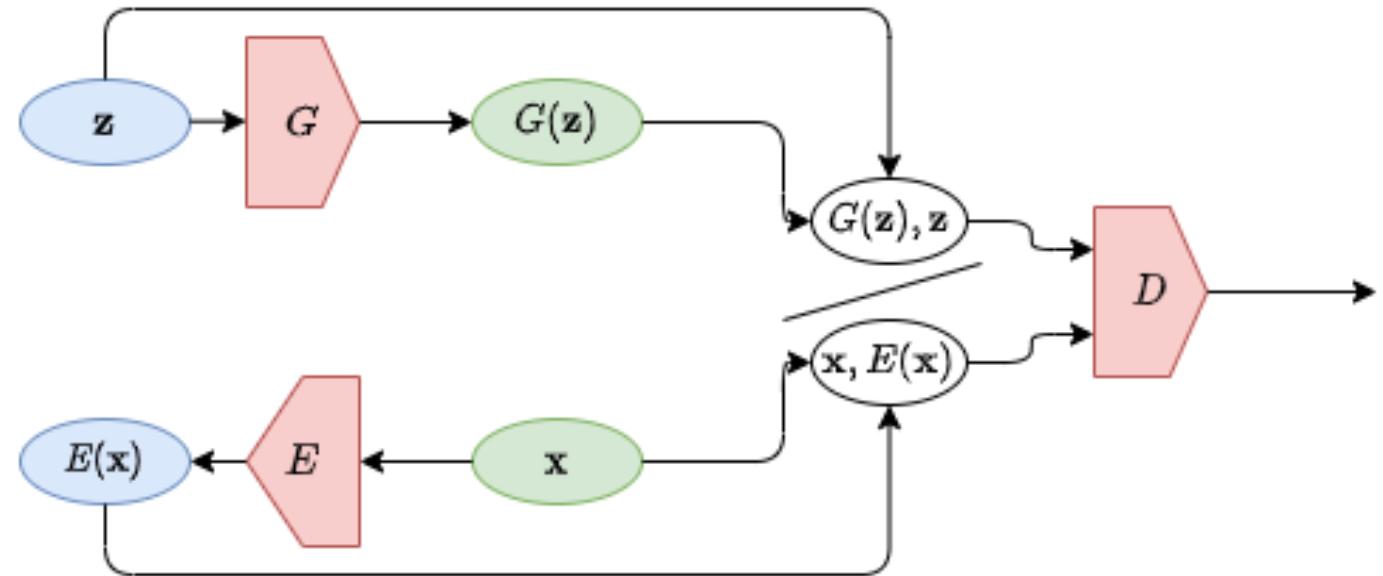
Inference

- **Anomaly score:** combines
 - I. image reconstruction error
$$\|G(\mathcal{E}(s)) - s\|_2$$
 - II. discriminator loss
$$(\mathcal{D}(s, \mathcal{E}(s)) - 1)^2$$

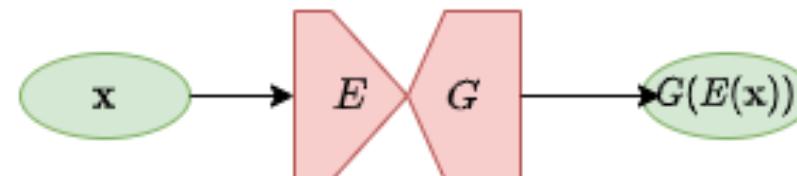
(since normal images are assumed to return 1)

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

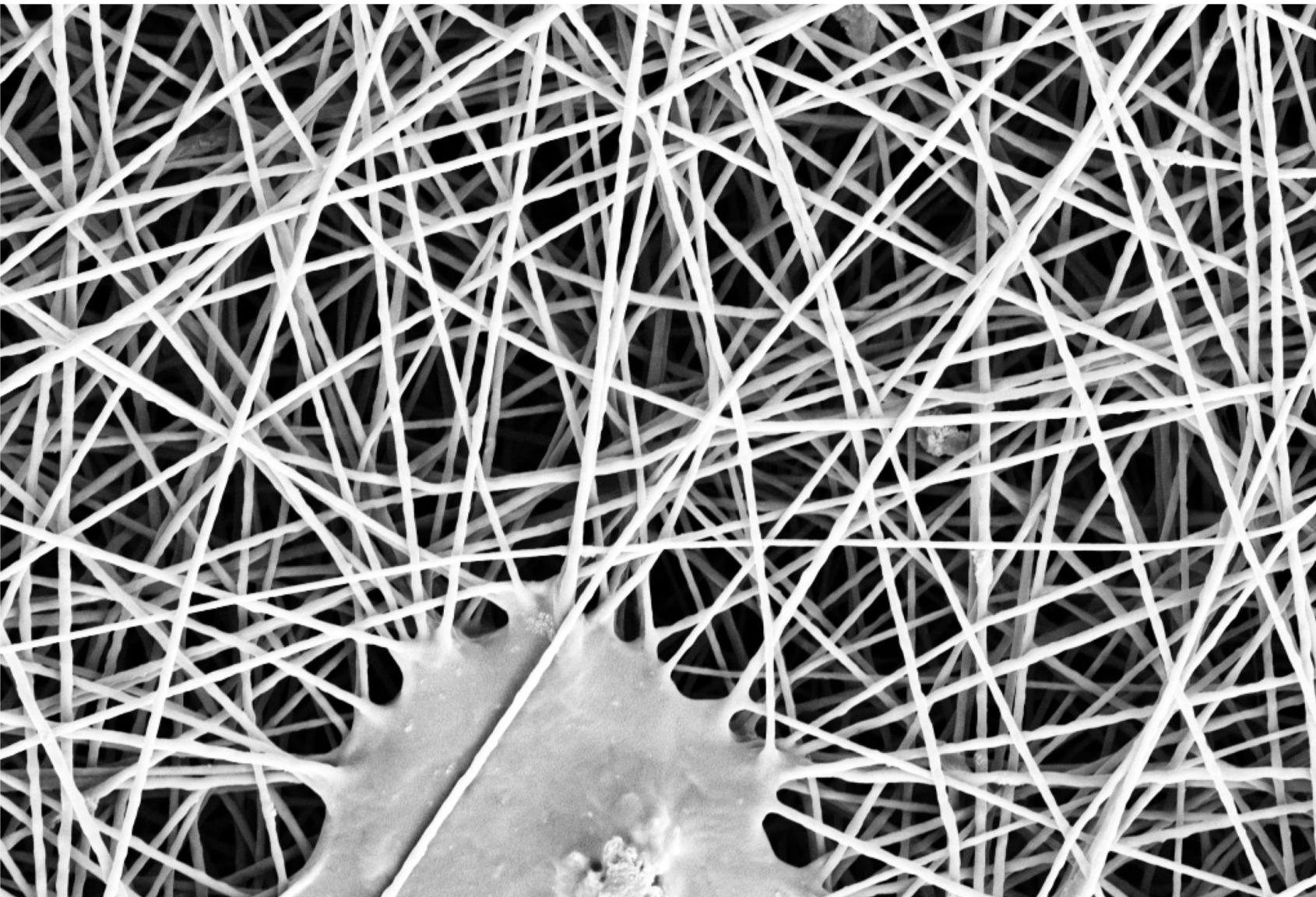
Adversarial Loss



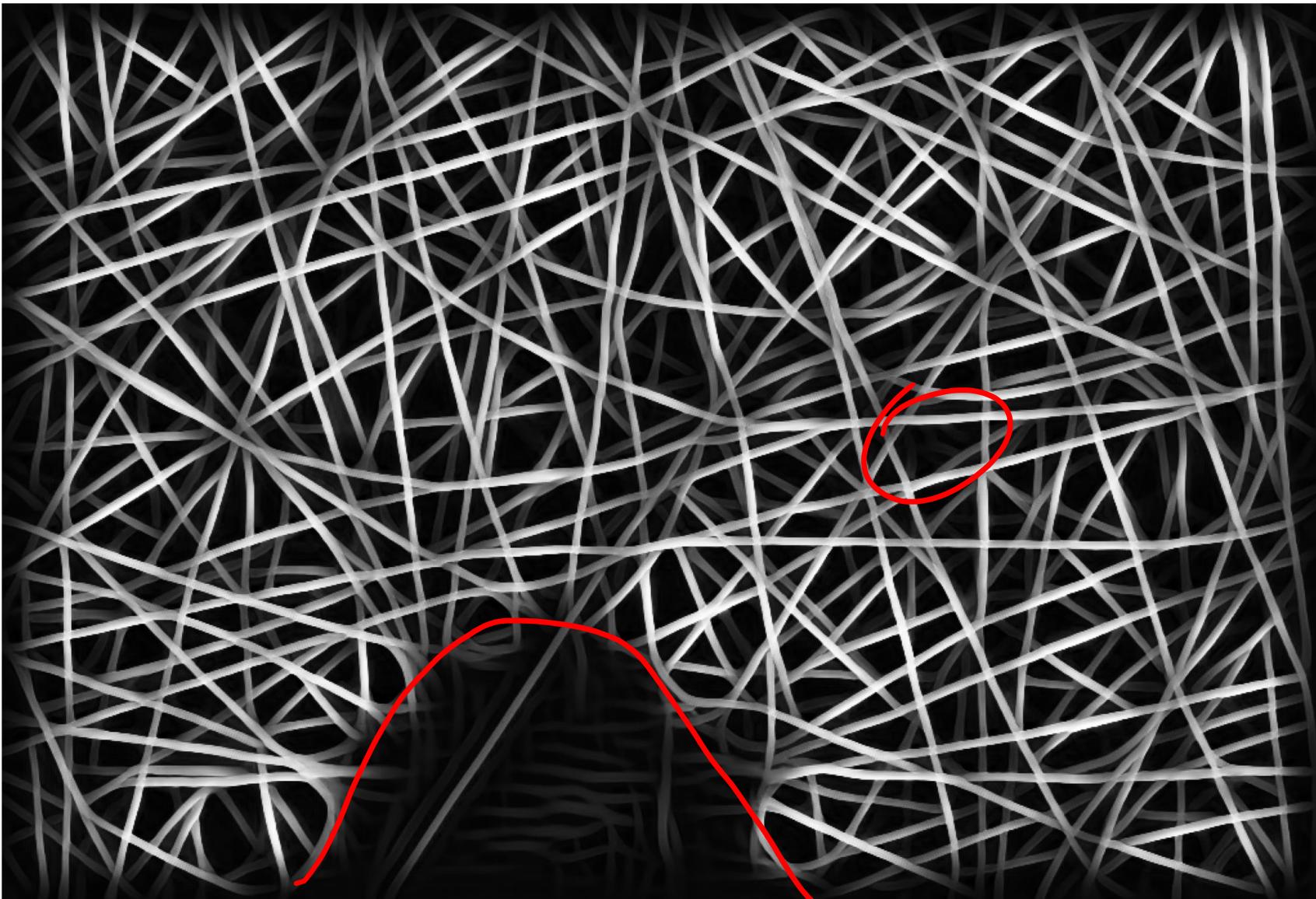
Reconstruction Penalty



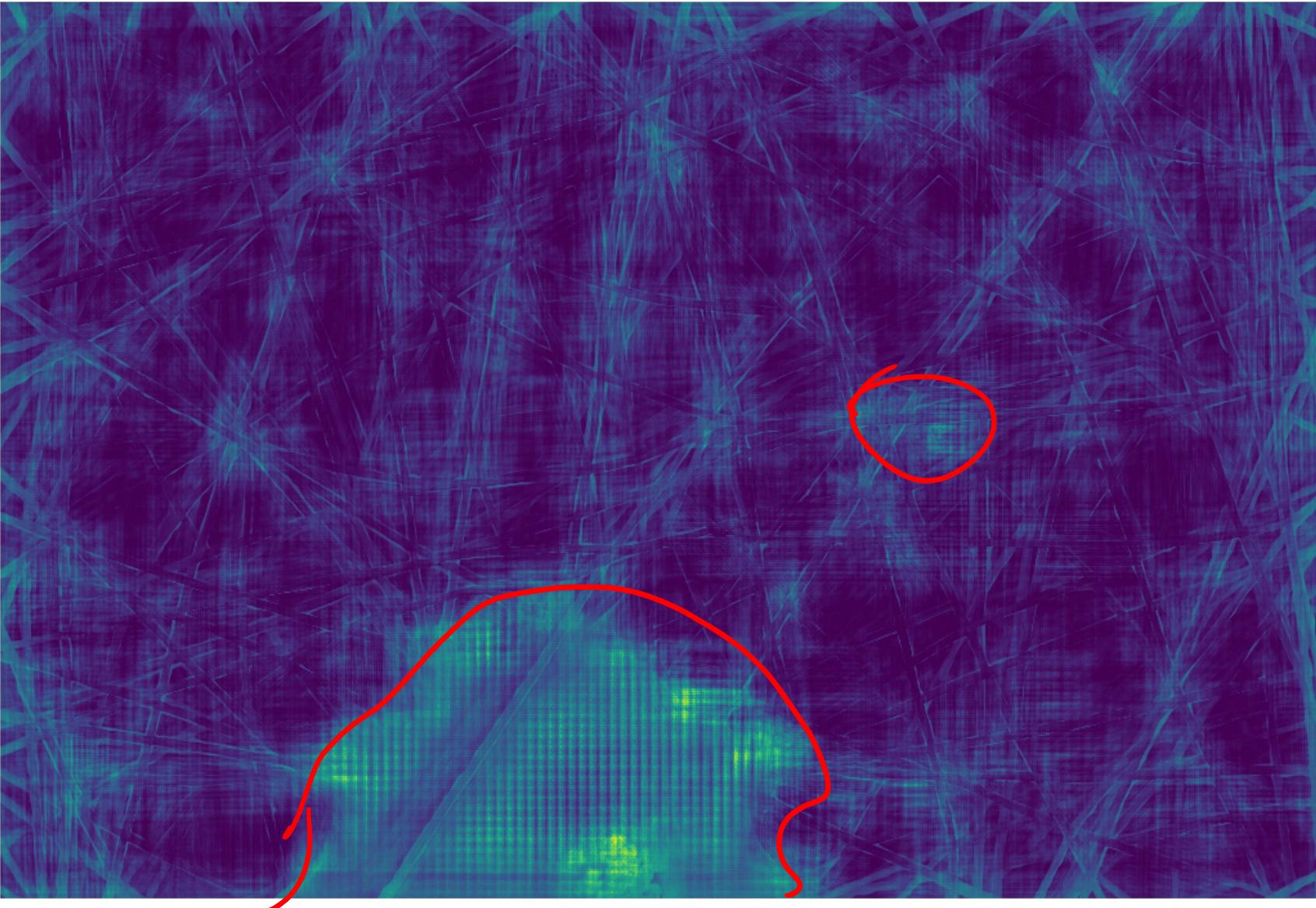
Input Image



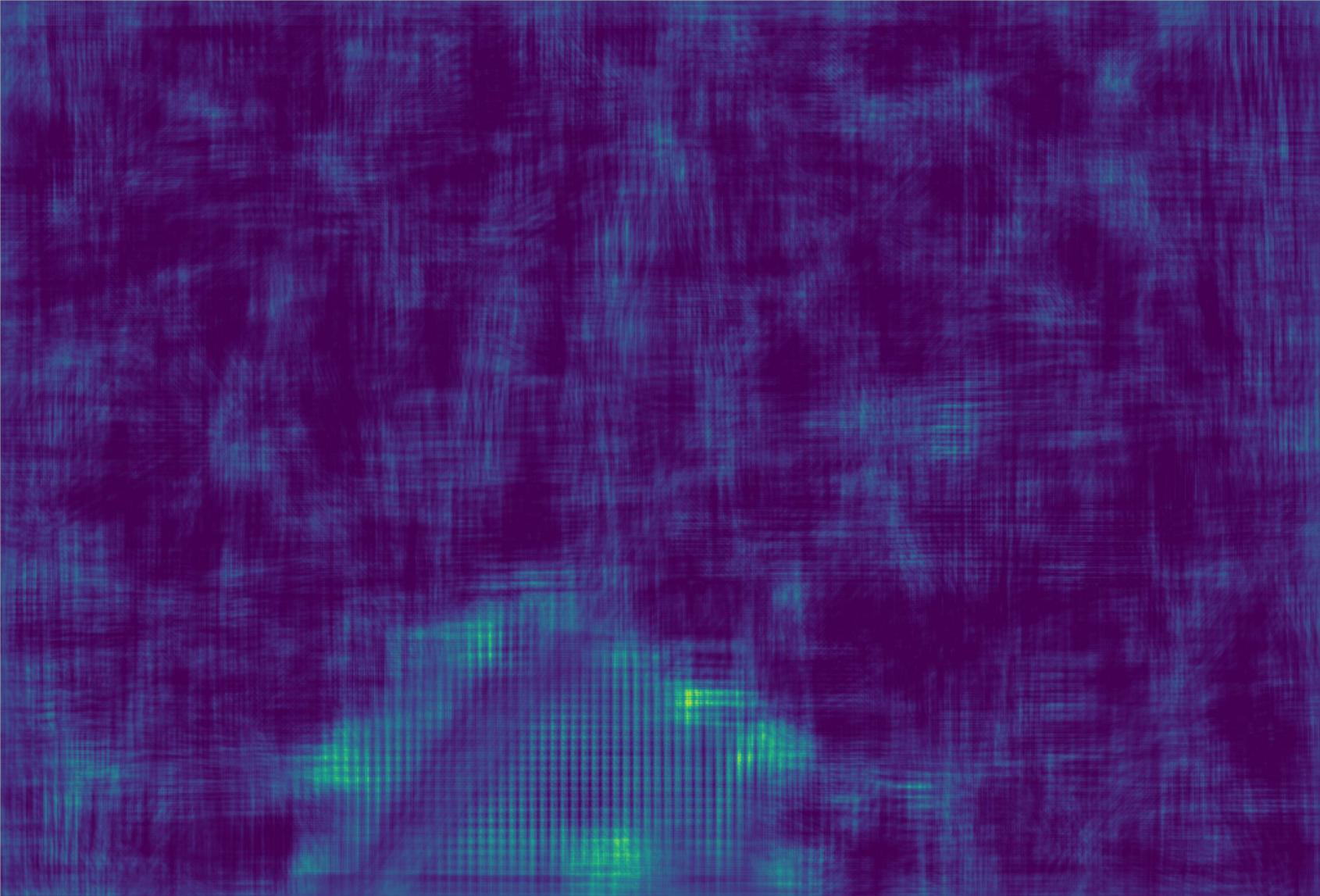
Reconstruction $\mathcal{G}(\mathcal{E}(s))$



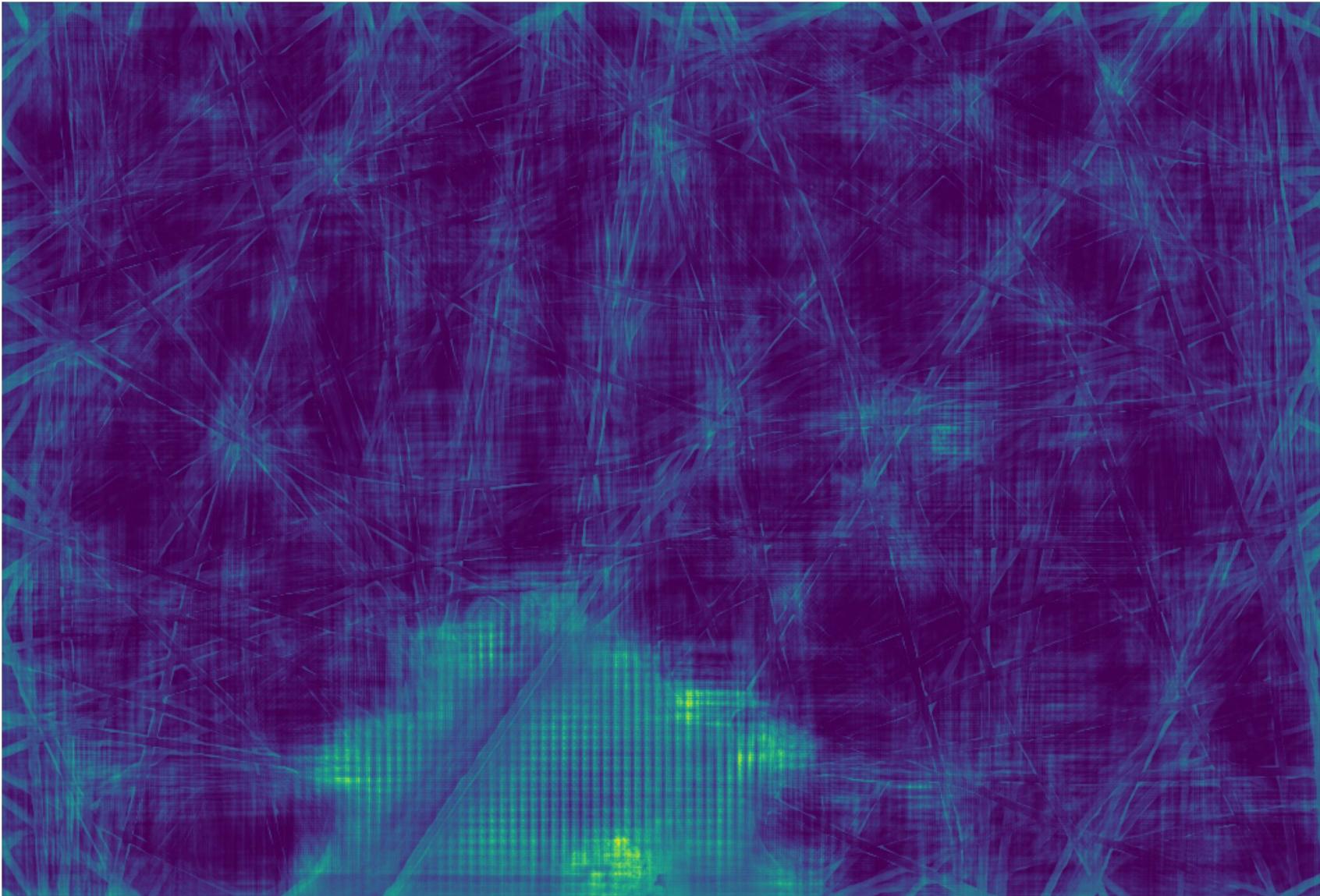
Reconstruction Loss $\|g(\varepsilon(s)) - s\|_2$



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



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



Normal Image Generation By Our GAN

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

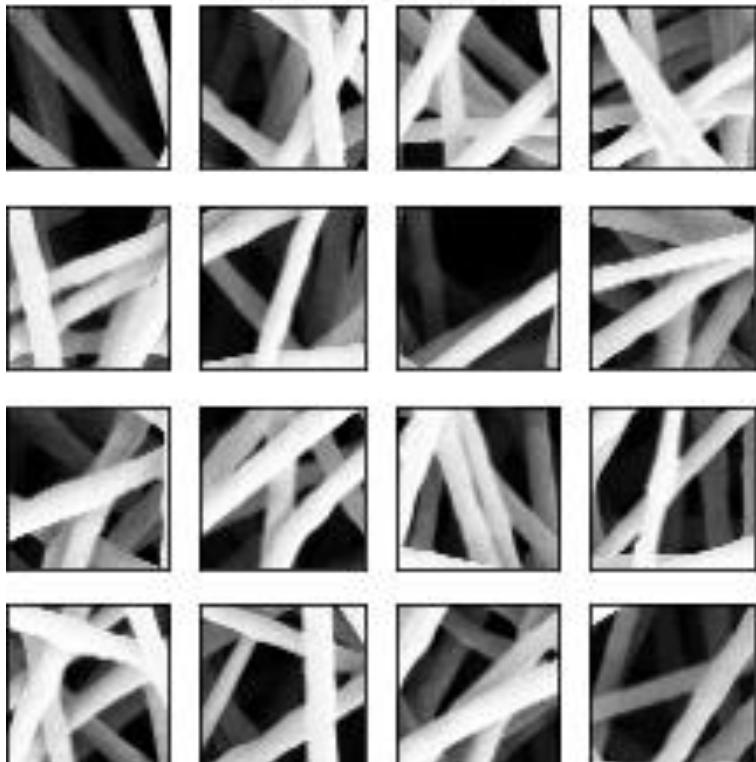
s

$$g(\epsilon(s))$$

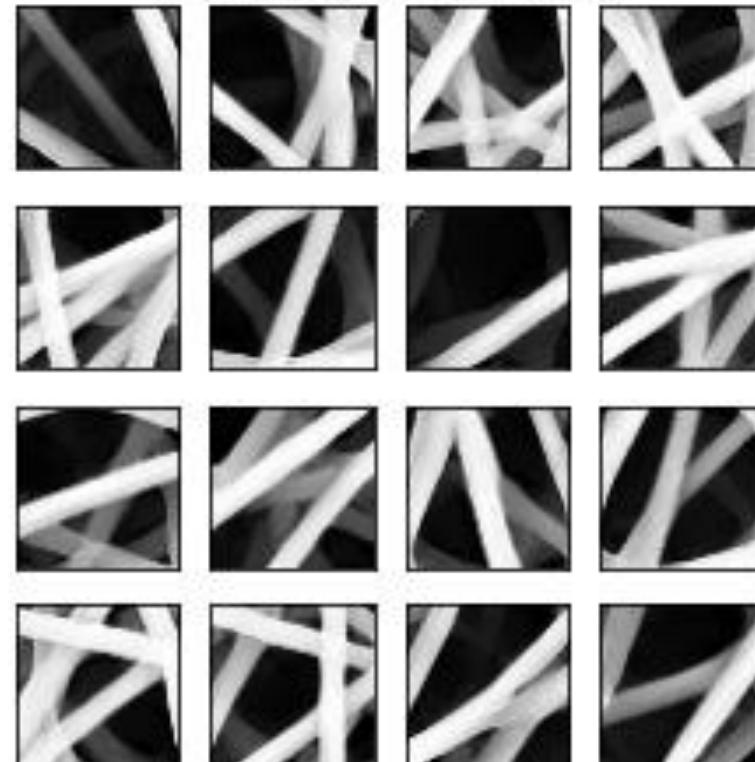
$$g(z)$$

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

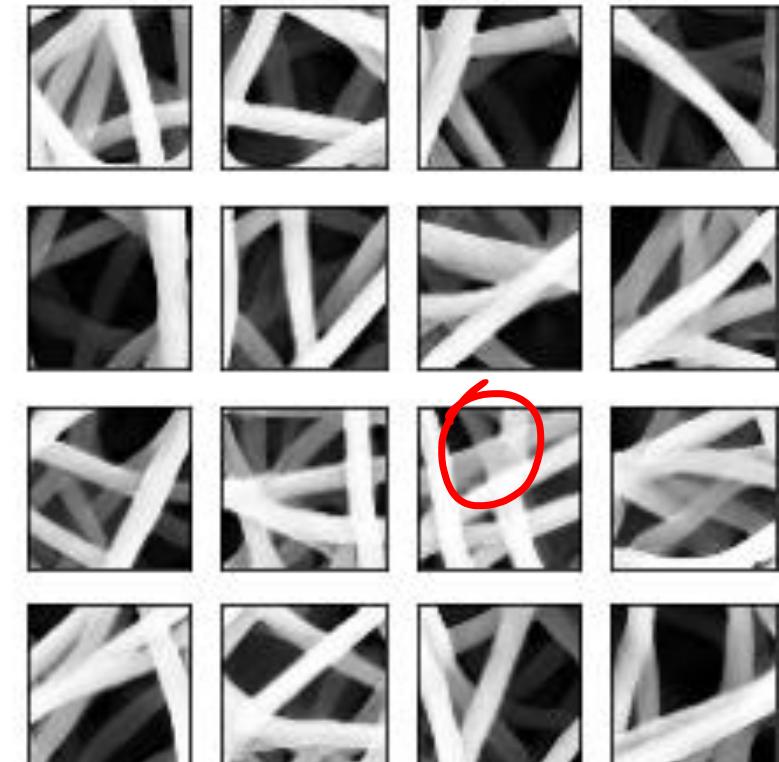
Original images



Reconstructed Images

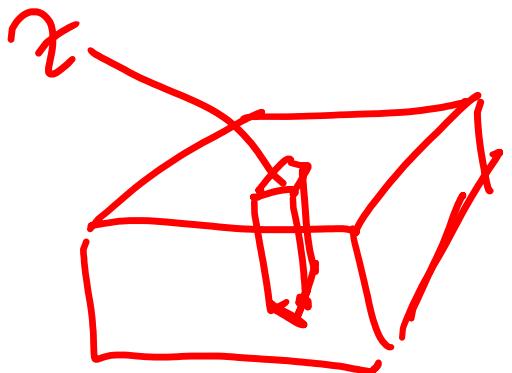


Random Generated Images

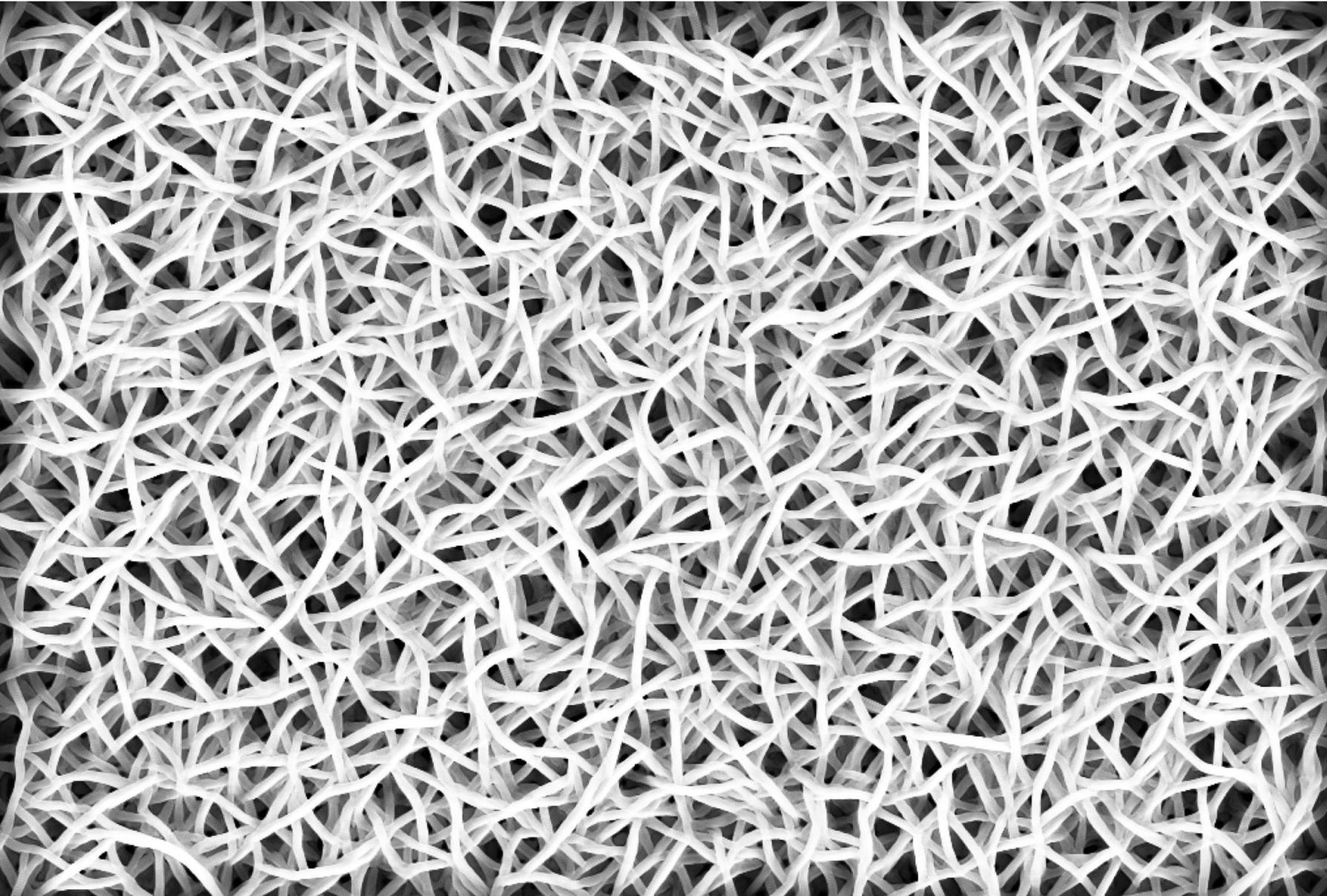


It is a Generative model!

$$z \in \mathbb{R}^{80 \times 120 \times 128}$$



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



A Few Opportunities...

Option 1: Join the Team for a Thesis

The Team

We are 3 faculties, 7 PhD students, 2 Research Assistants



Giacomo Boracchi



*Luca Magri
(Researcher)*



*Federica Arrigoni
(Researcher)*



Filippo Leveni



Antonino Rizzo



Michele Craighero



Andrea Schillaci



Diego Stucchi



Loris Giulivi



*Andrea Porfiri
Dal Cin*



Giuseppe Bertolini



Edoardo Peretti

Research Collaborations

Major research collaborations:



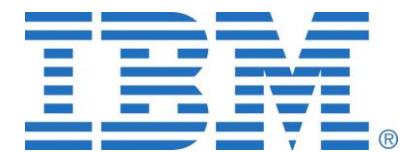
National Research
Council of Italy



Major research projects:



.Cleafy



Thesis Information

- We typically illustrate thesis opportunities in February and September
- Thesis concern either Deep Learning or Computer Vision.
- Thesis are primarily research thesis, or thesis on industrial projects
- Sometimes we open internship with companies we are collaborating with
- We typically send proposals for Honours Program in Research (for those of you interested in research perspectives)
- We are always interested in brilliant candidates and perspective PhD students

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.*



What is this course about?

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

..like denoising



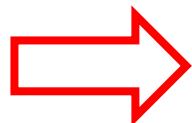
Who cares about images?

Who cares about images?

Everybody!

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

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

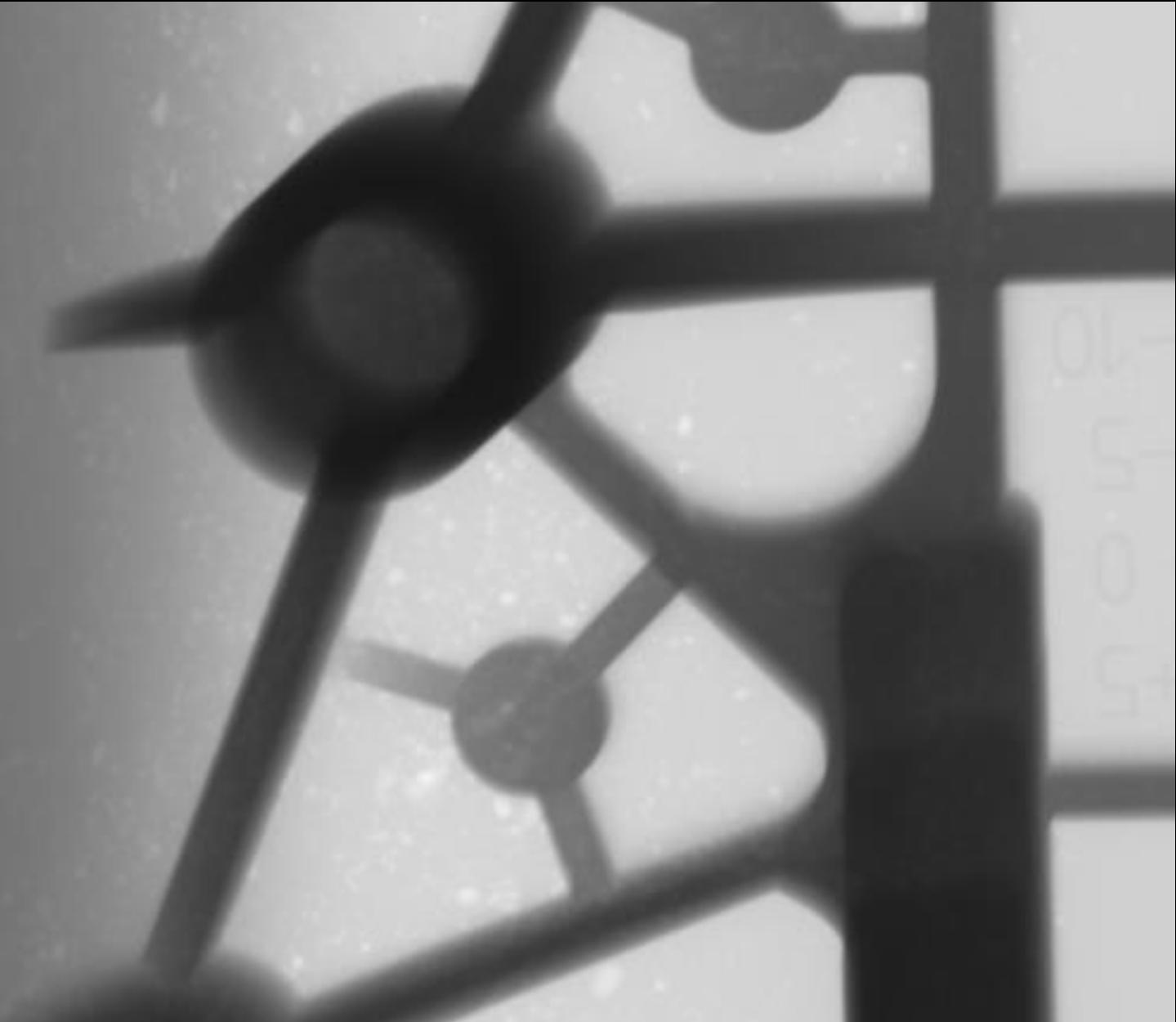


$$\hat{y} \approx y$$

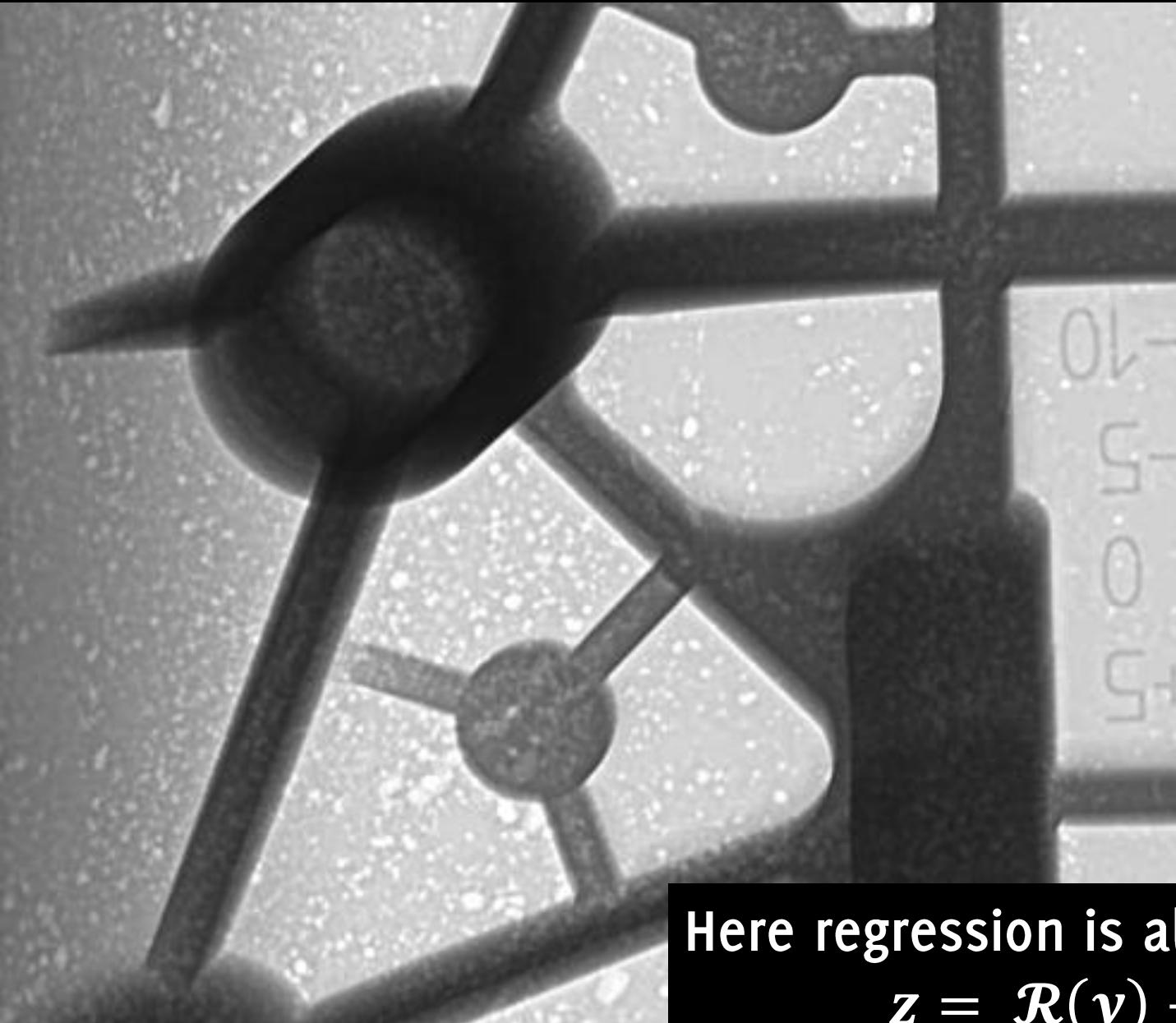


Denoising is a regression problem: given the noisy z , estimate \hat{y} close to the unknown y

Who cares about images? Quality Inspection

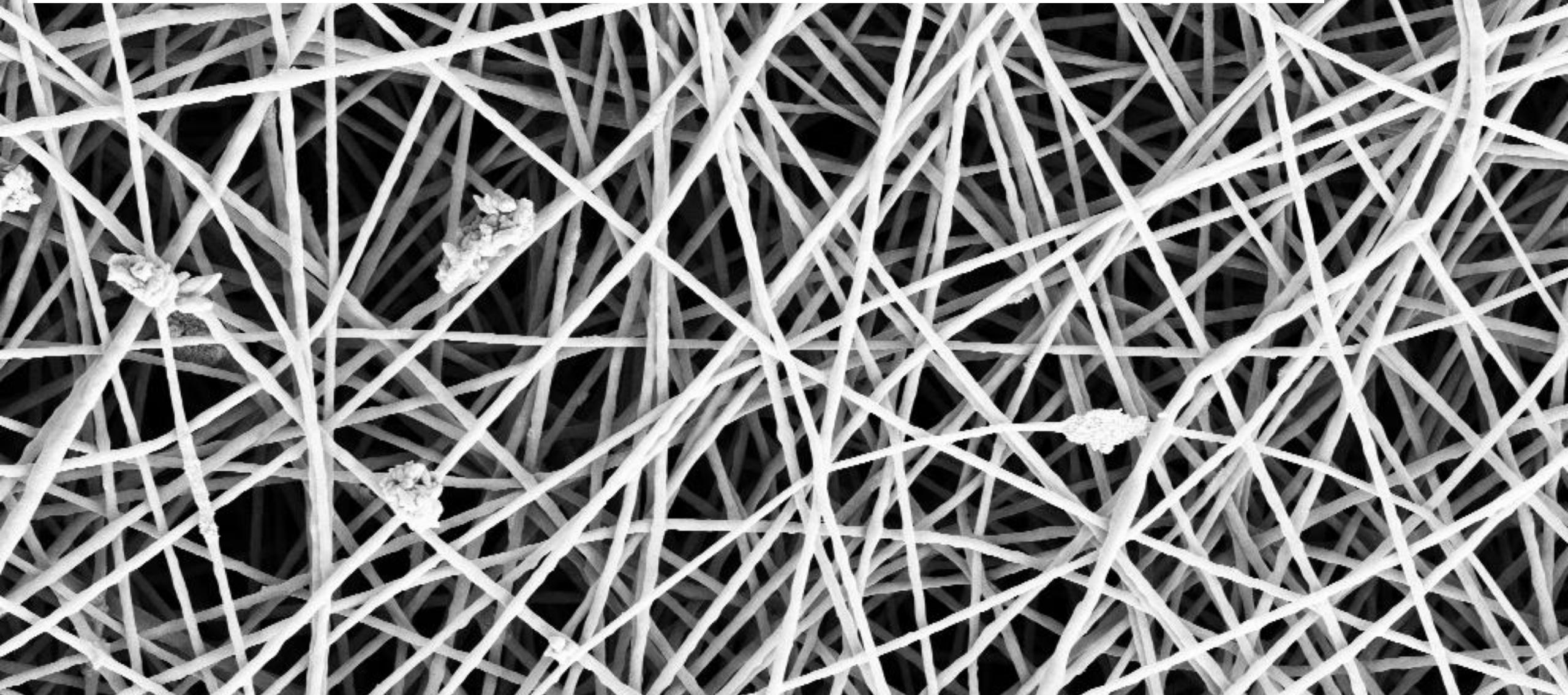


Who cares about images? Quality Inspection

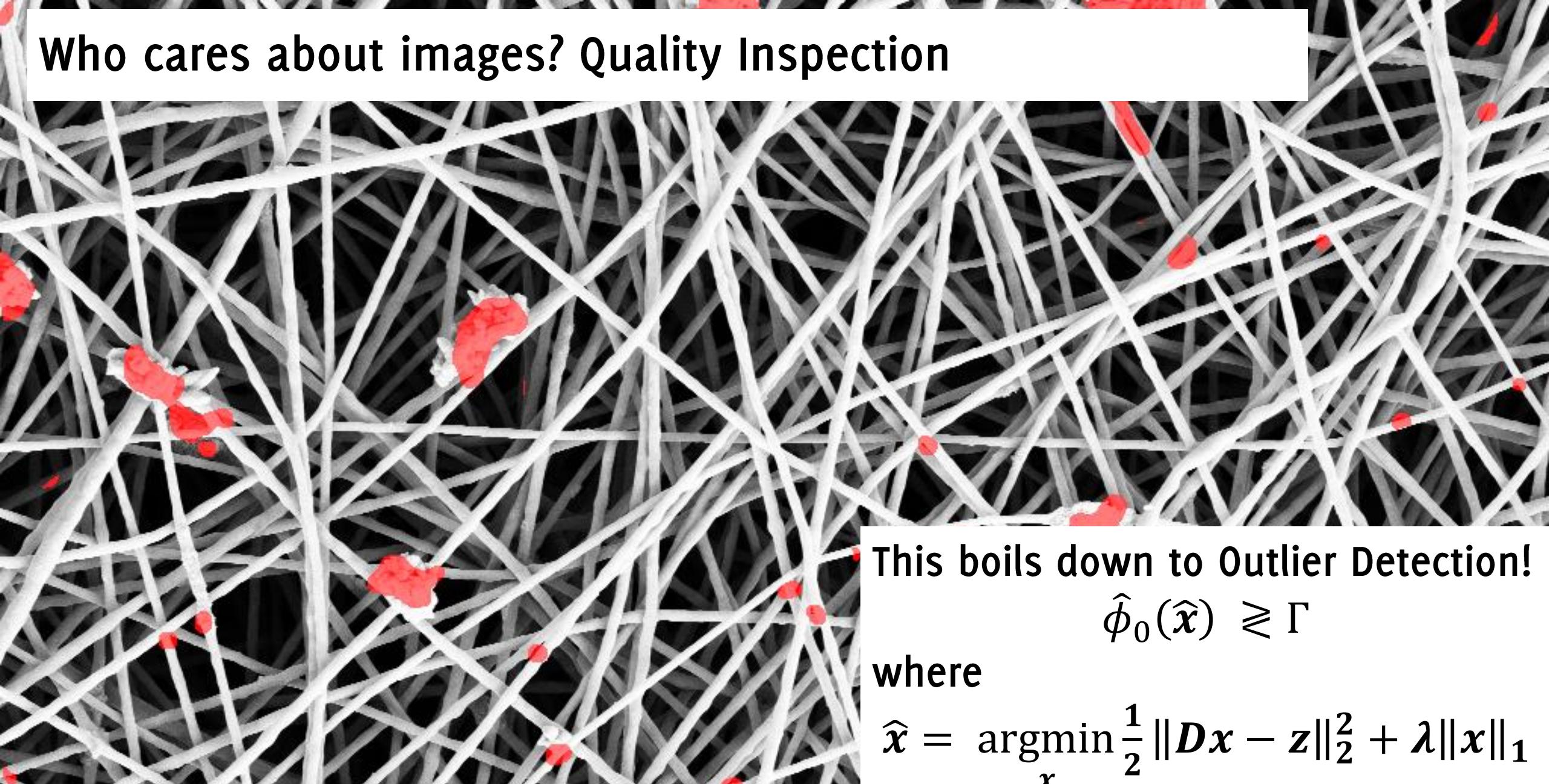


Here regression is also crucial
 $\mathbf{z} = \mathcal{R}(\mathbf{y}) + \boldsymbol{\eta}$

Who cares about images? Quality Inspection



Who cares about images? Quality Inspection



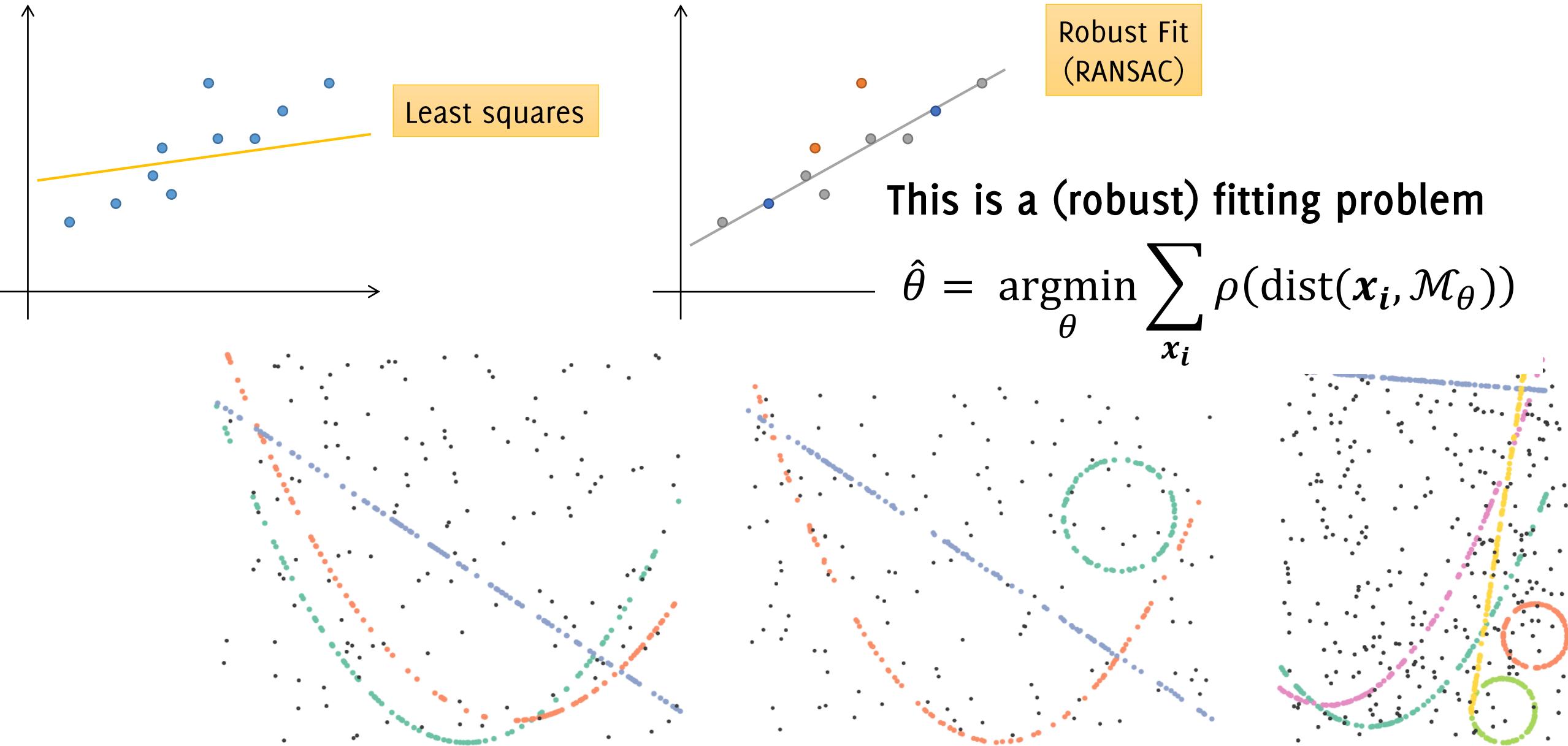
This boils down to Outlier Detection!

$$\hat{\phi}_0(\hat{x}) \geq \Gamma$$

where

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

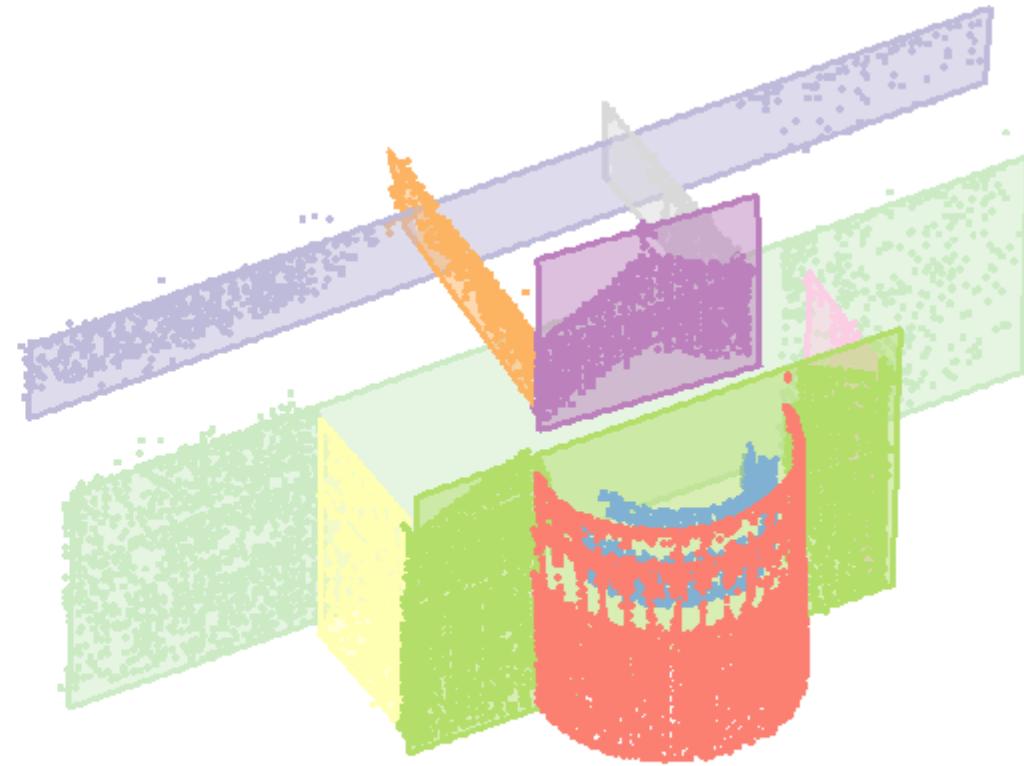
Who cares about images? *visual recognition systems*



Who cares about images? *visual recognition systems*



(a) Input point cloud



(b) Recovered structures

This is a (robust) fitting problem

Who cares about images? visual recognition systems

12:30 Mar 19 mar

36% 



Who cares about images? visual recognition systems

12:30 Mar 19 mar

36%



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!

0k, 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.

Questions?

Denoising over adaptively defined neighborhoods
for local polynomial regression



Option 3: Advanced Deep Learning PhD Courses

Every year we offer a PhD course

The Raise of Transformers

- Mon 23/01/2023 (14:00-19:00): Attention is all you need (Prof. Matteo Matteucci): From Recurrent Neural Networks (RNN) to Transformers; attention mechanism in recurrent models; the mathematics of Transformer and the self-attention mechanism. Multi-head attention. Positional encoding.
- Wed 25/01/2023 (14:00-19:00): Transformer in NLP with applications (Prof. Mark Carman): Using Transformer for Language Modeling; Bidirectional Encoder Representations from Transformers (BERT); Generative Pre-trained Transformer (GPT); Application of NLP Transformers in Question Answering, Language Translation, Summarization, Model explanation, Information Retrieval.
- Fri 27/01/2023 (14:00-19:00): Visual Transformers with applications (Prof. Giacomo Boracchi): Attention Mechanism in CNN and Deep Models for Visual Recognition; Vision Transformers: architecture, differences from CNN, self-supervised training; Application of Vision Transformer in visual recognition (Object Detection, Semantic Segmentation, Anomaly Detection)
- Mon 30/01/2023 (14:00-19:00): Time series Transformers with applications (Prof. Manuel Roveri): Learnable time representations; Transformers for time series; Preprocessing and embedding of time series; Dealing with seasonalities; Resampling in time and feature spaces; Applications in time series classification. Prediction and anomaly detection.
- Wed 01/02/2023 09:00-19:00: Hands-on Transformers (all teachers ...): The last day of the course aims at providing a practical experience on developing transformers. Students will learn to develop a simple Transformer model in python for NLP, Image Understanding, and Time-series prediction.