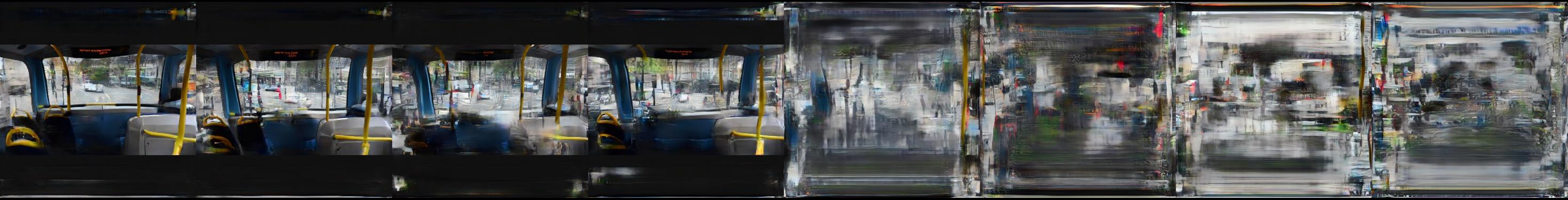
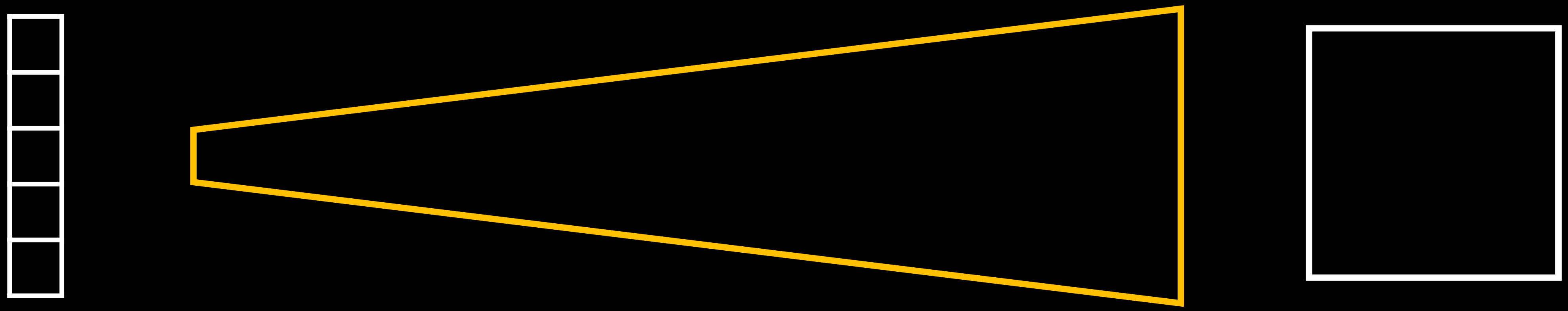
Exploring Machine Intelligence

Week 5, Generative Models

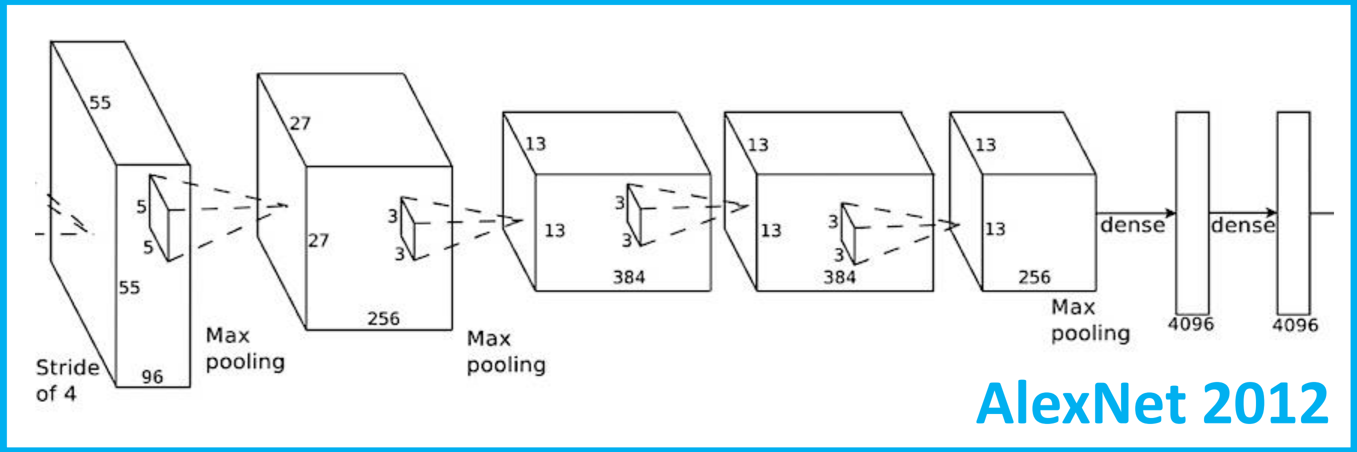


~ Vítek Růžička

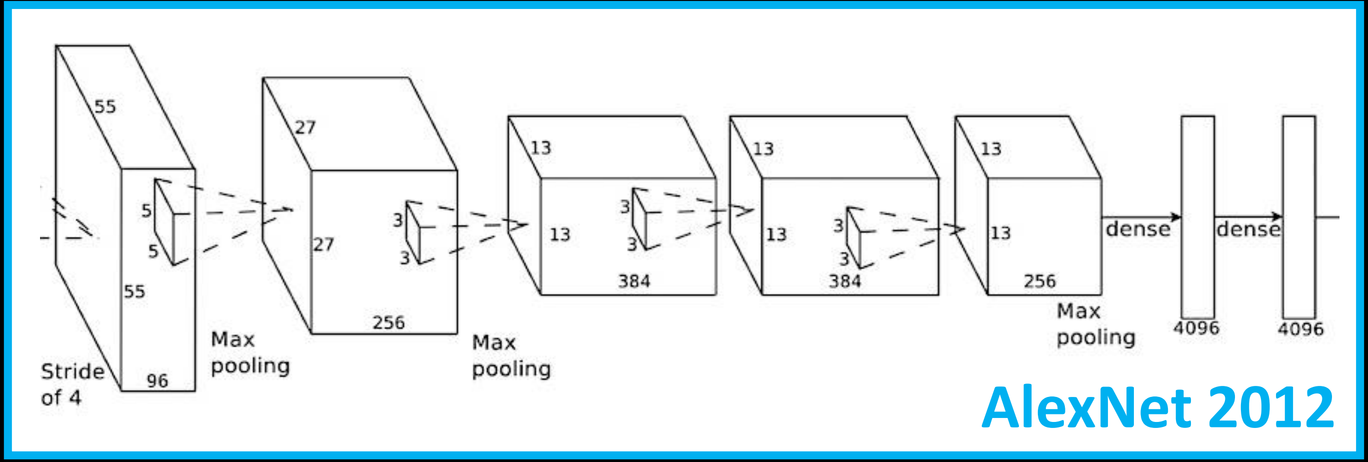
Motivation for today

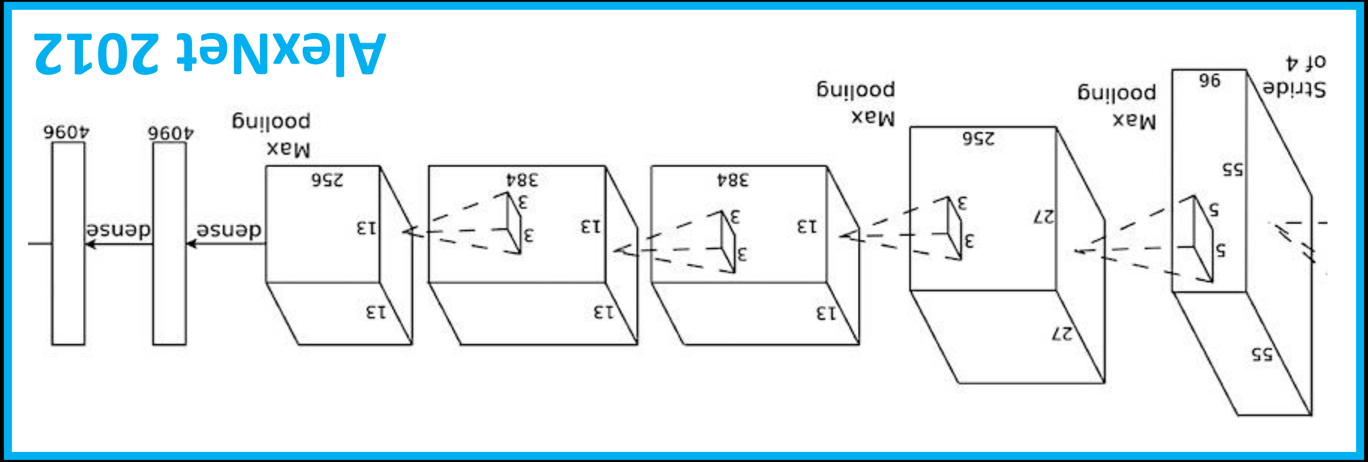


Classification

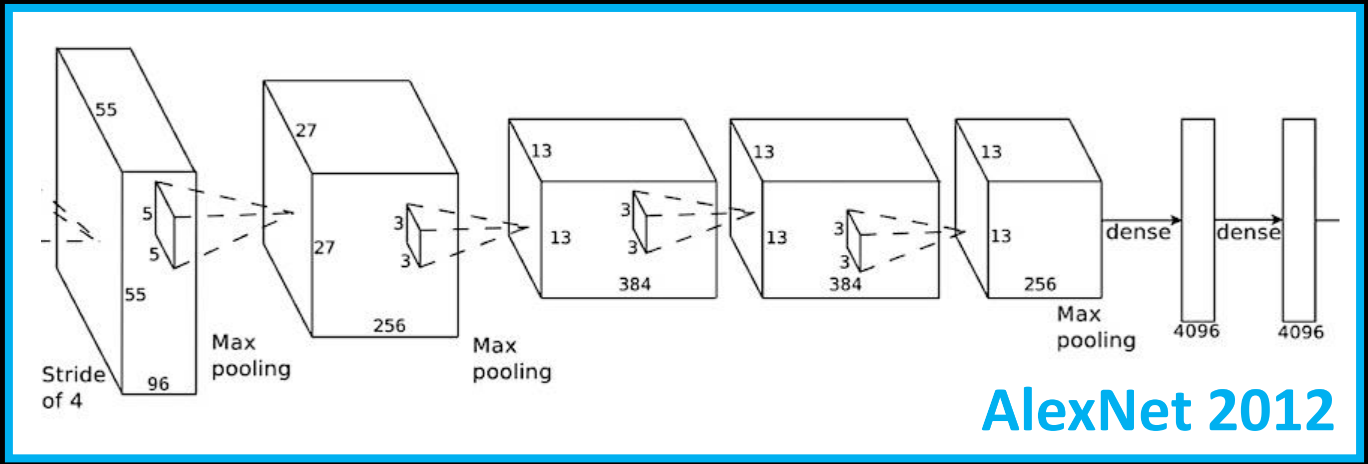


Classification



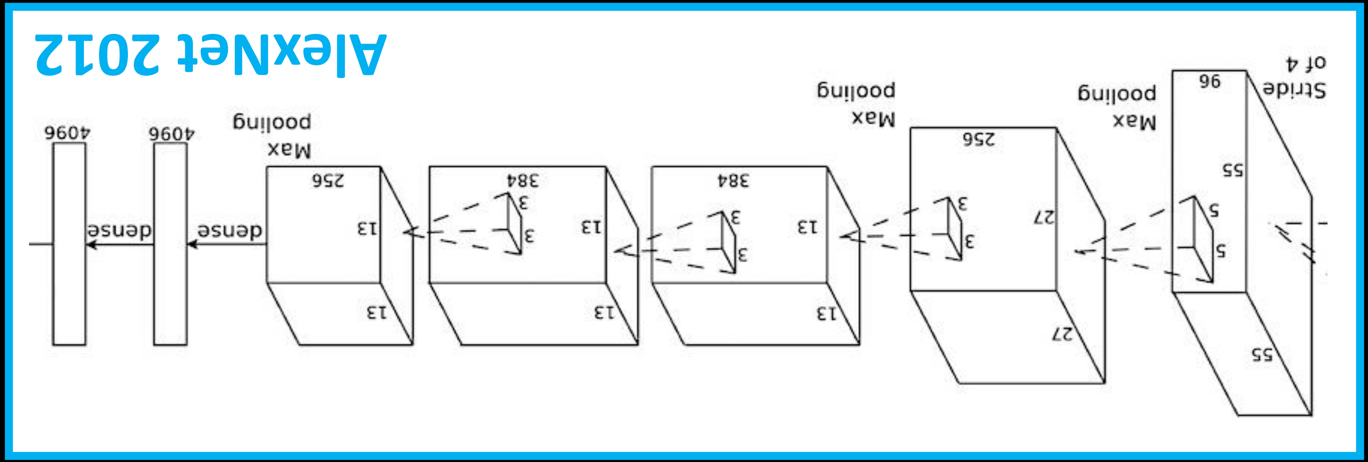


Classification



Generation

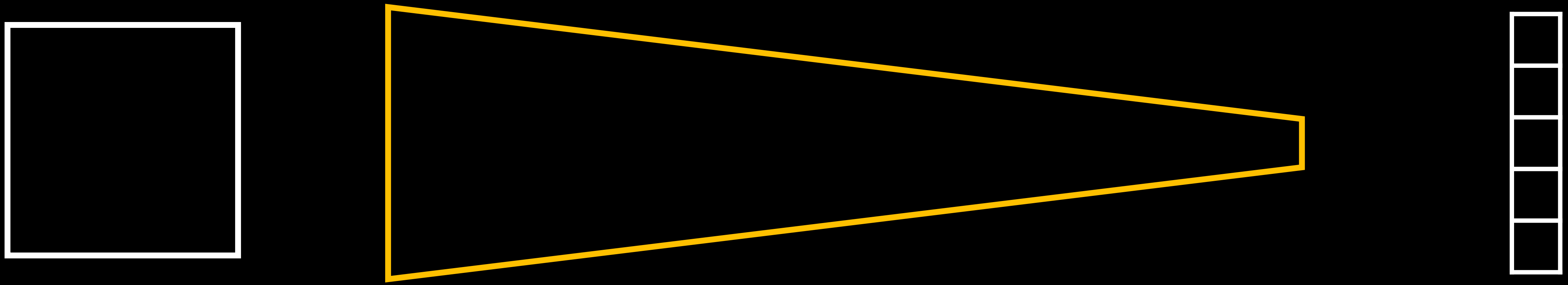
Target class



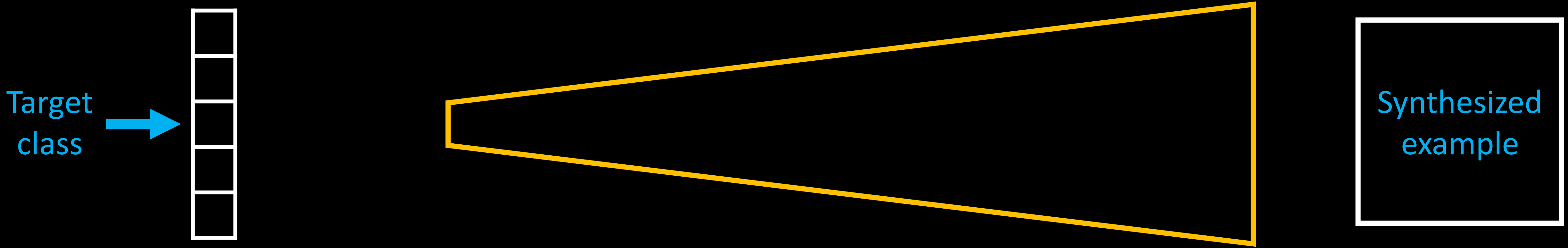
Synthesized example

- What we *more or less* want is to **flip the task** – but it turns out this is too difficult task

Classification

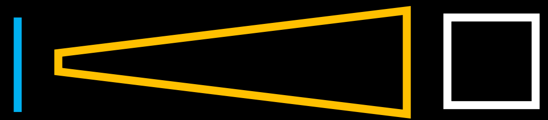


Generation



- What we *more or less* want is to **flip the task** – but it turns out this is too difficult task

Today



Generative Models:

- Turing the classification models up side down
- **AutoEncoders** and interaction
- **Generative Adversarial Networks** – interaction and artworks

Practical session:

- Scraping the Internet and Intro to Generative models

Machine Learning Models

- So far, we worked with classification models, feature descriptors, … etc. -> models used in **analysis**

Machine Learning Models

- So far, we worked with classification models, feature descriptors, … etc. -> models used in **analysis**

- We want to be able to create new imagery which would correspond to some training datasets -> **generation**

Machine Learning Models

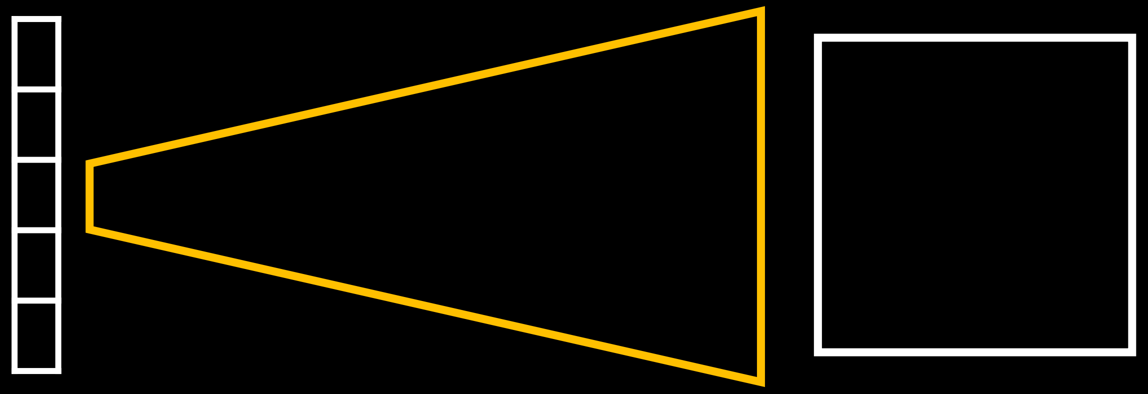
- So far, we worked with classification models, feature descriptors, … etc. -> models used in **analysis**

- We want to be able to create new imagery which would correspond to some training datasets -> **generation**

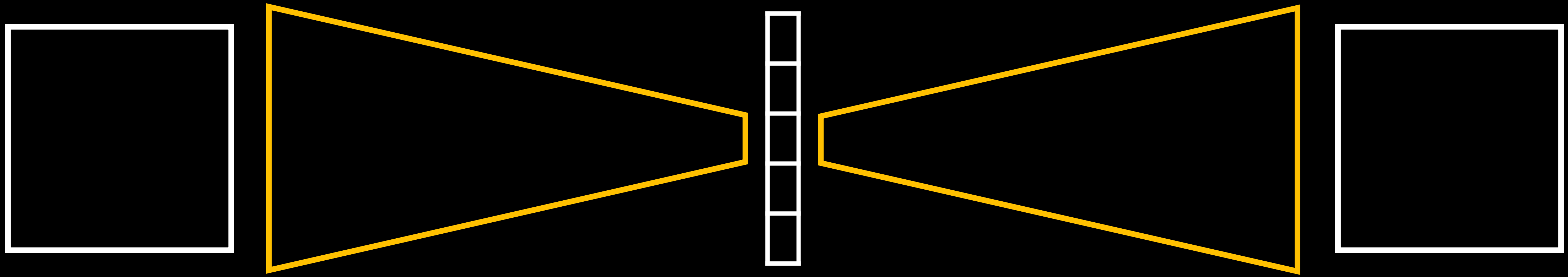
  creativity?

- *Keep in mind during today's class: What are the ways we can exercise control over these models? Where is the possibility of artistic input in these?*

Generative Architectures

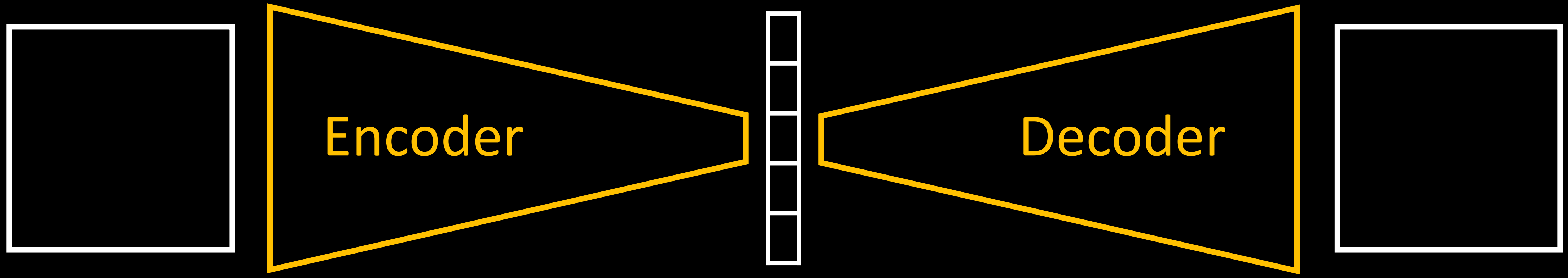


AutoEncoders



- Rephrase it as an **identity operation**, we want the model to encode image into some intermediate lower-dimensional representation and decoder to undo that work.

AutoEncoders



- Rephrase it as an **identity operation**, we want the model to encode image into some intermediate lower-dimensional representation and decoder to undo that work.
- Seemingly this is a useless task – we are making a machine for nothing. But as we saw in previous class (feature extraction), **parts of the models can also be useful!**

AutoEncoders

Layers:

Convolutional

Pooling

Fully Connected

UnPooling (scale up)

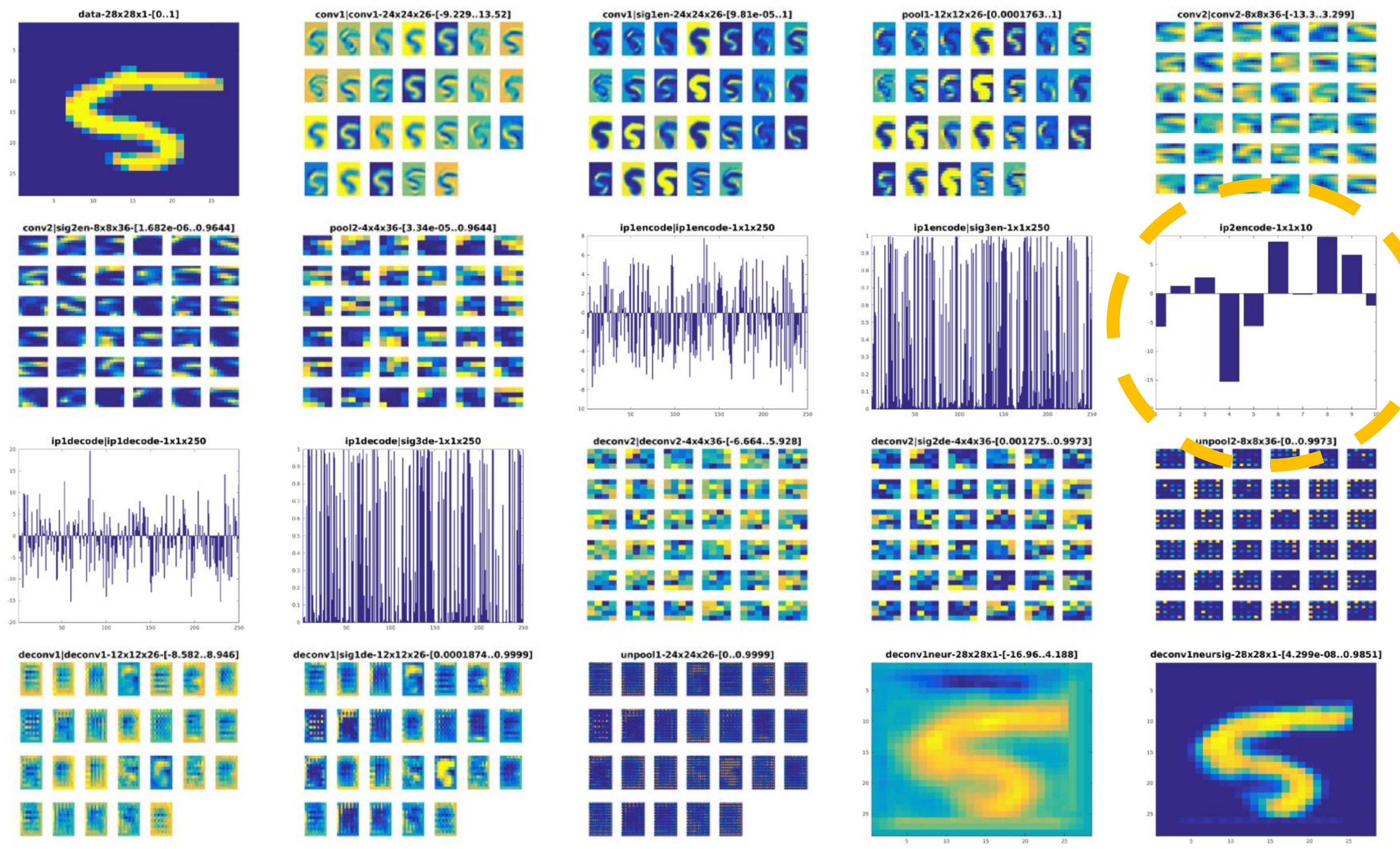
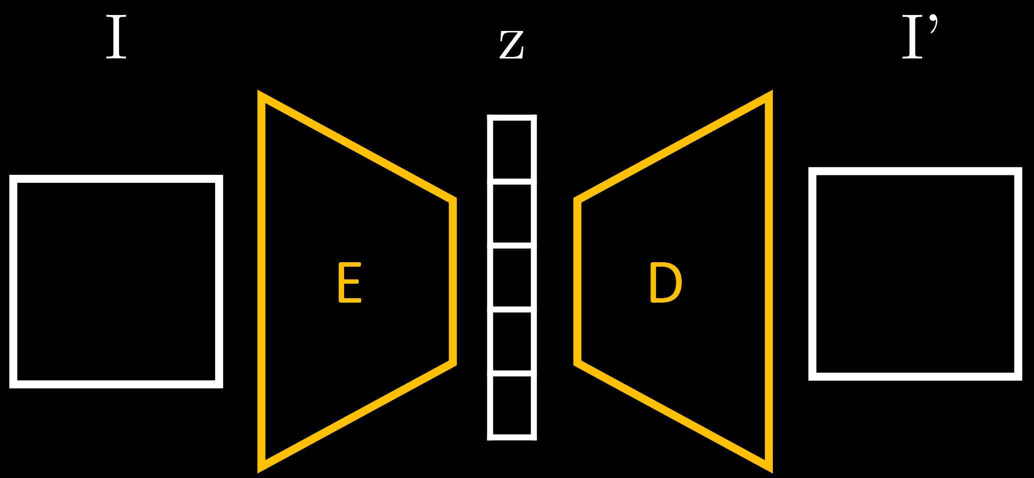


Fig. 11. Visualization of encoding and decoding digit '5' by 10D Model 4.
Each panel visualizes the output of appropriate layer of Model 4. The title of each panel describes a type of the layer (e.g. conv1, pool1, conv2, pool2, and so on), a size of the layer (e.g. 24x24x26 means 26 feature maps with 24x24 elements each) and a range of the layer's output [min..max]. See more explanation in the next to last paragraph of Section 4.3.

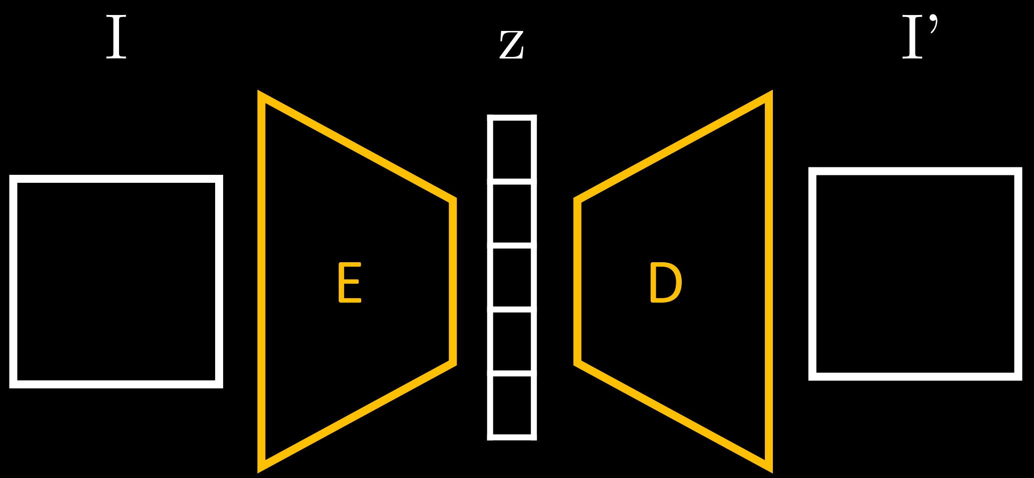
Latent vector: 10 numbers

AutoEncoder



< Training

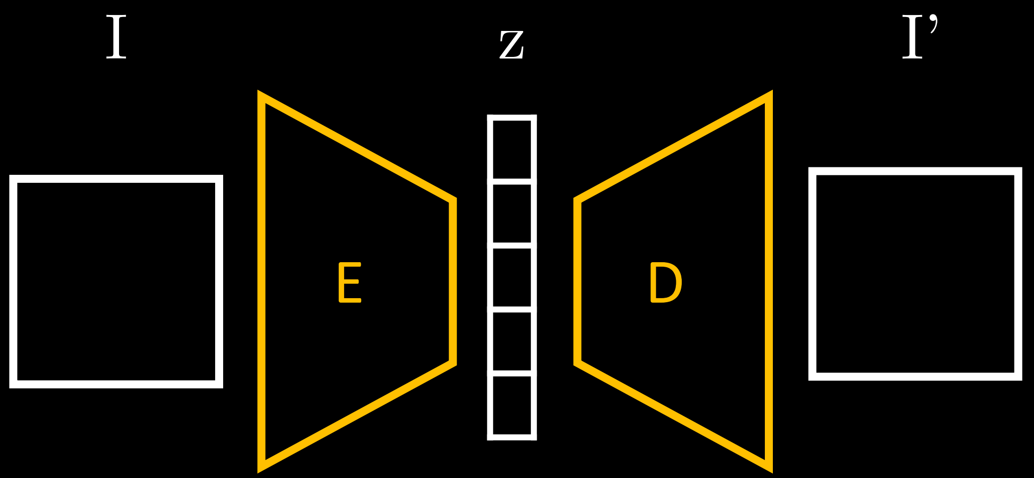
AutoEncoder



- Reconstructed image

$$I' = D(\underbrace{E(I)}_{z})$$

AutoEncoder



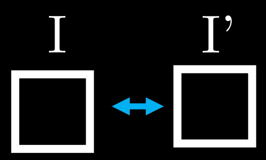
- Reconstructed image

$$I' = D\big(\underbrace{E(I)}_{z}\big)$$

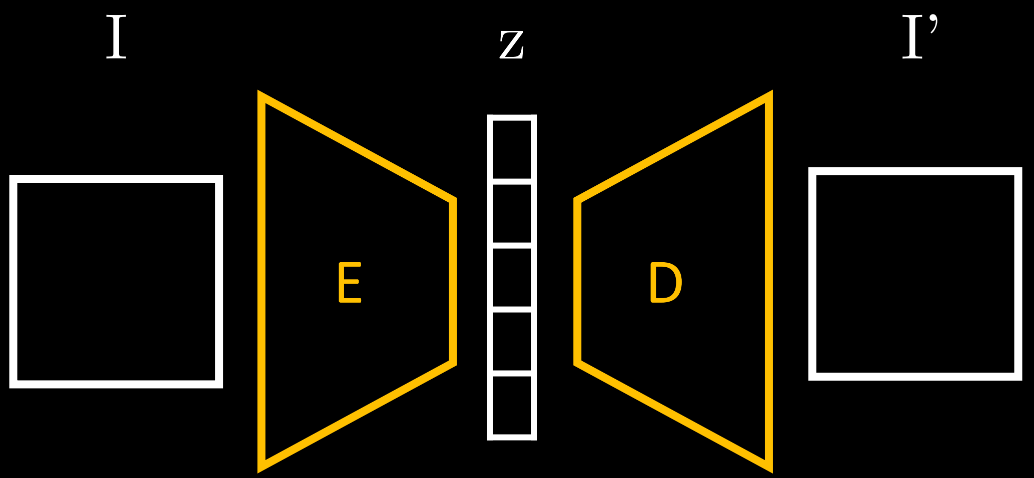
- Training with reconstruction loss:

$$\boldsymbol{distance\big(D\big(E(I)\big), I\big)}$$

(for example: MSE)



AutoEncoder



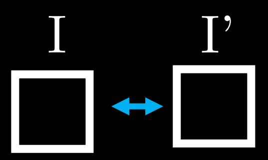
- Reconstructed image

$$I' = D(\underbrace{E(I)}_{z})$$

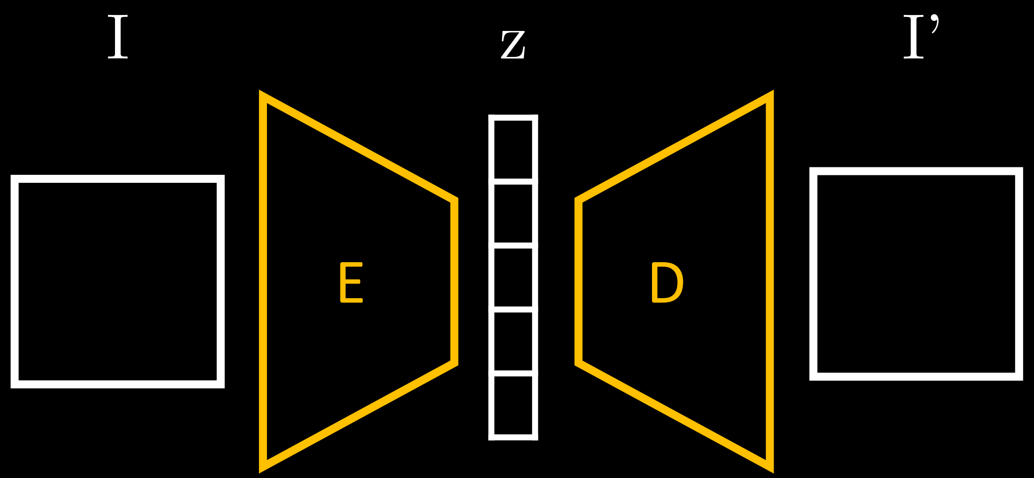
- Training with reconstruction loss:

$$\boldsymbol{distance(D(E(I)), I)}$$

(for example: MSE)

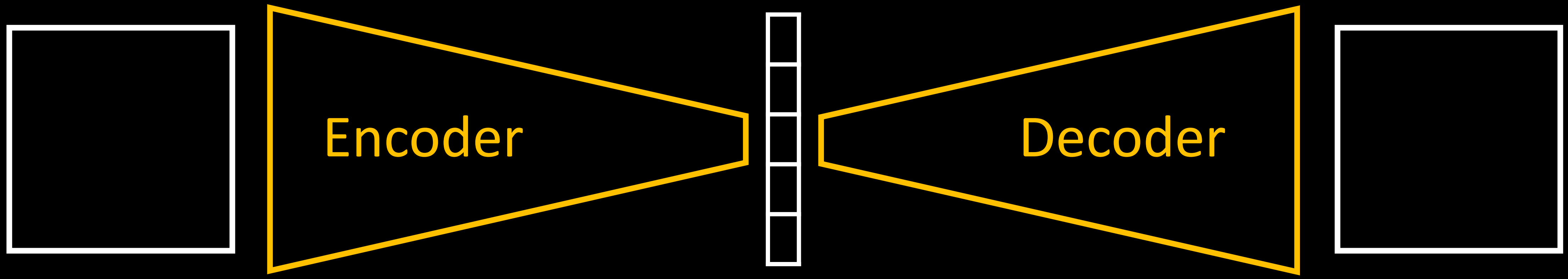


Variational AE



- We impose **additional restrictions on the latent representation** (z), such as that it has a Gaussian distribution: $z \sim N(0,1)$
- The loss function is more complicated, it comes from probabilistic theory (for further read see a blog)

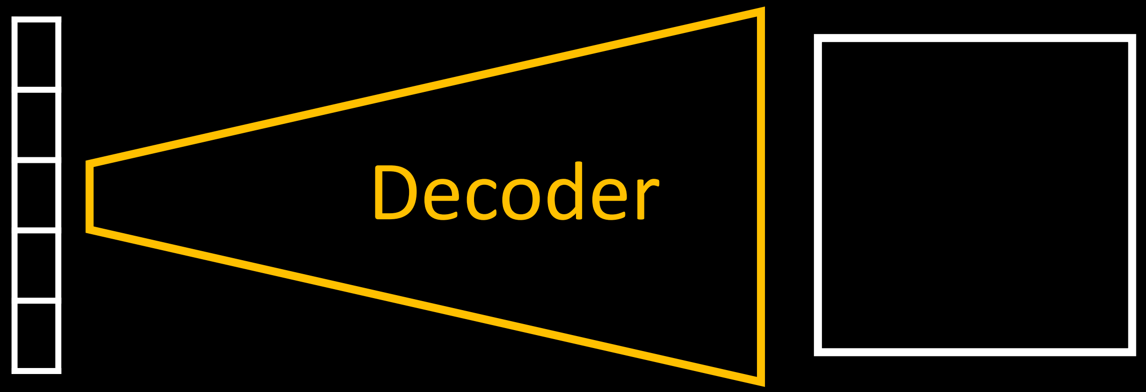
Interaction



- With a trained AutoEncoder model

Interaction

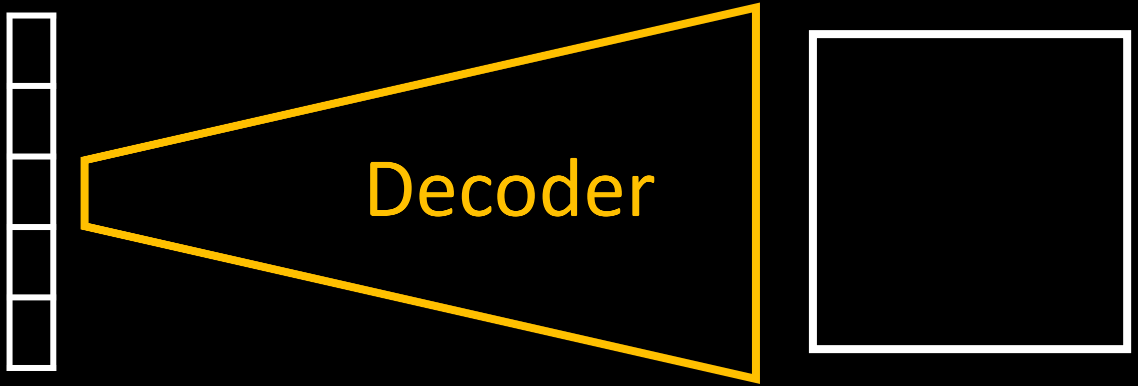
We want to **generate new images**:



Interaction

We want to **generate new images**:

- New random vector z_1 -> new random image I_1



$$z_1 \in \mathbb{R}^{512}$$
$$\downarrow$$
$$I_1$$

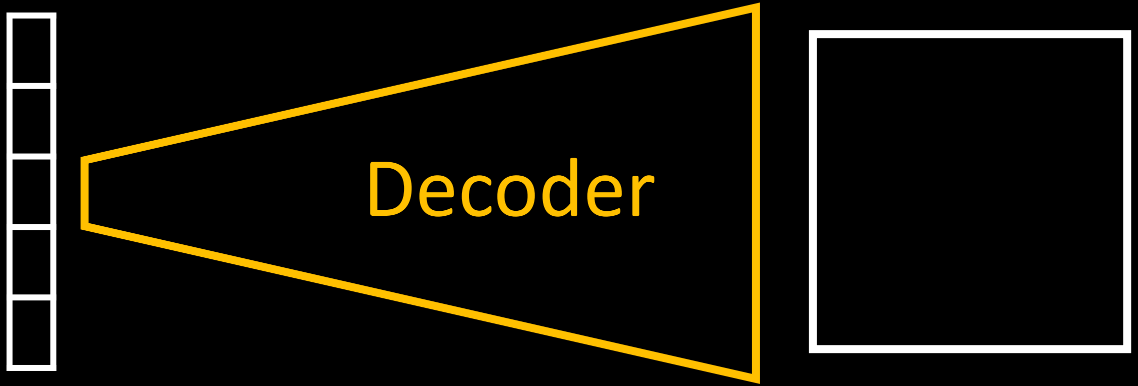
… z_1 is a vector of 512 real numbers

z1 = [0.1, 0.2, …, 0.9, -0.2, 0.3]

Interaction

We want to **generate new images**:

- New random vector z_1 -> new random image I_1
- Another random vector z_2 -> another random image I_2



$z_1 \in \mathbb{R}^{512}$

$\downarrow$

I_1

$z_2 \in \mathbb{R}^{512}$

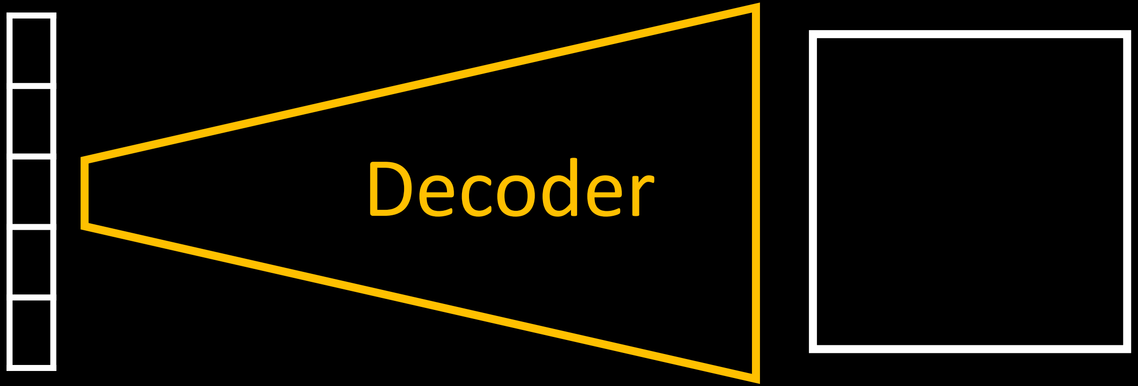
$\downarrow$

I_2

Interaction

We want to **generate new images**:

- New random vector z_1 -> new random image I_1
- Another random vector z_2 -> another random image I_2



$z_1 \in \mathbb{R}^{512}$ - - - - - → $0.4*z_1 + 0.6*z_2$ ← - - - - - $z_2 \in \mathbb{R}^{512}$

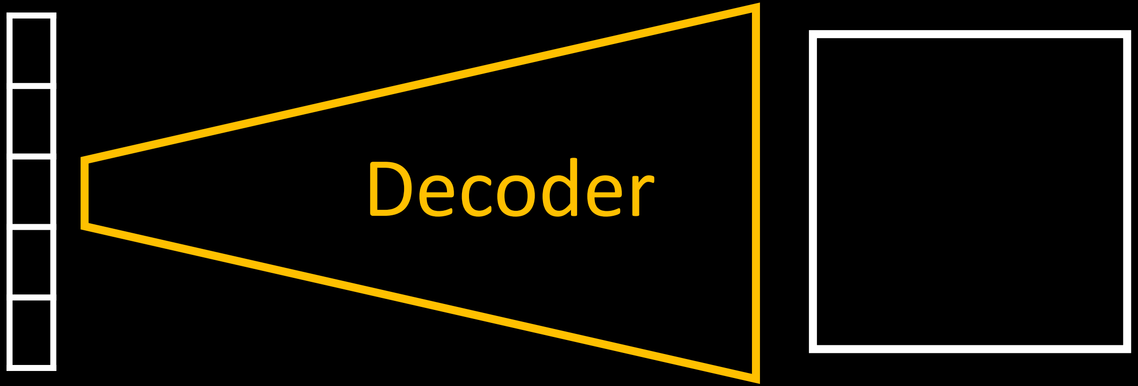
$\downarrow$ I_1 $\qquad\qquad$ $\downarrow$ I_2

We can easily mix these vectors and get a new vector of 512 numbers.

Interaction

We want to **generate new images**:

- New random vector z_1 -> new random image I_1
- Another random vector z_2 -> another random image I_2



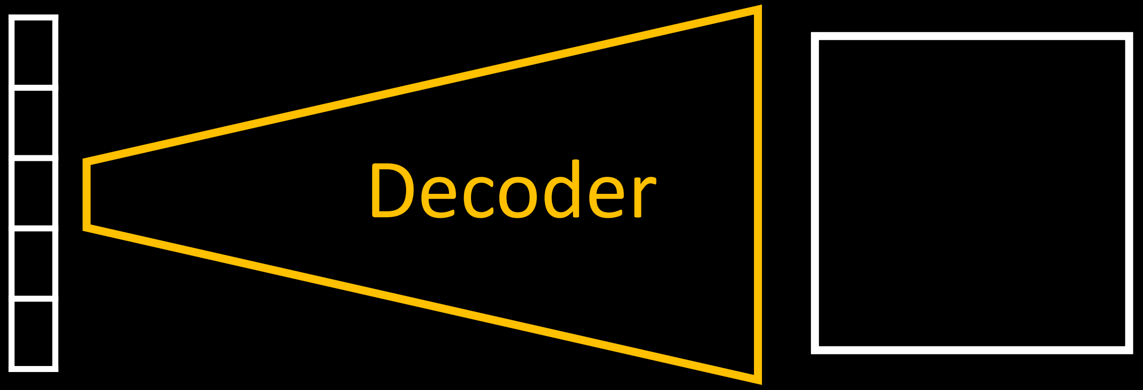
$z_1 \in \mathbb{R}^{512}$ | $0.4 * z_1 + 0.6 * z_2$ | $z_2 \in \mathbb{R}^{512}$

I_1 | Mix 0.4 of I_1 and 0.6 of I_2 | I_2

Interaction

We want to **generate new images**:

- New random vector z_1 -> new random image I_1
- Another random vector z_2 -> another random image I_2



$z_1 \in \mathbb{R}^{512}$ $\downarrow$ I_1

Interpolation!

$a^*z_1 + (1.0\text{-}a)^*z_2$ $\downarrow$ Mix of images I_1 and I_2

$z_2 \in \mathbb{R}^{512}$ $\downarrow$ I_2

Interpolation

- Mixing in the **pixel space**:

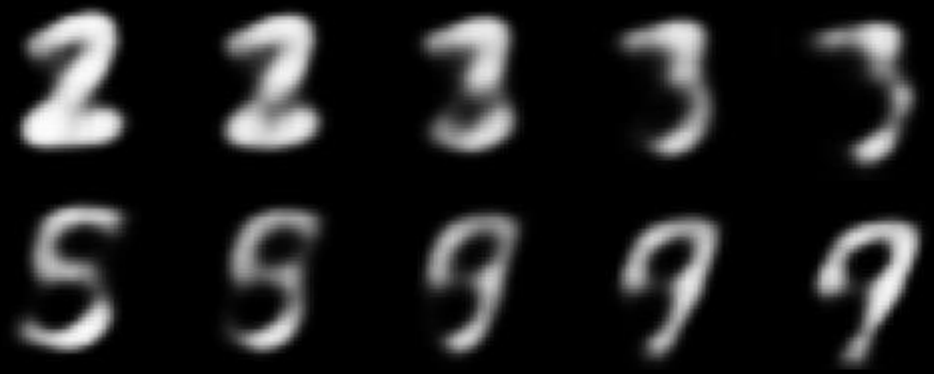
$$a^*I_1 + (1.0\text{-}a)^*I_2$$

… where each image is a matrix

- Mixing in the **latent space**:

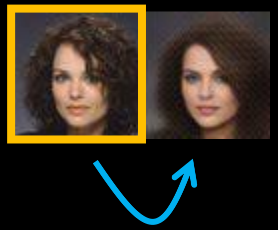
$$a^*z_1 + (1.0\text{-}a)^*z_2$$

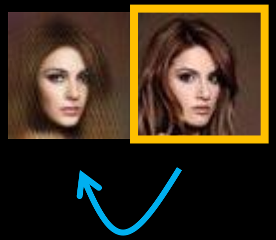
… where each image is generated from the mixed vector



AutoEncoder specific interaction I.

- **Encoding real samples** into their latent representations

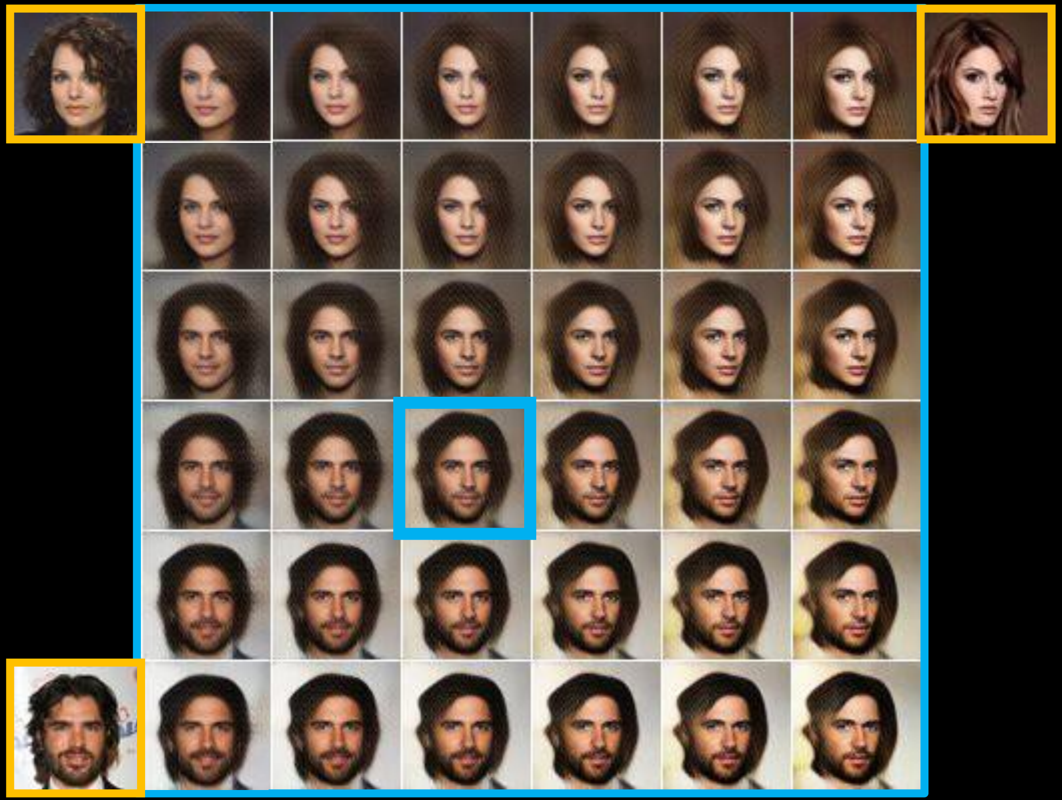




PS: the encoded image is often not reconstructed perfectly (we can notice a sort of blurry approximation)

AutoEncoder specific interaction I.

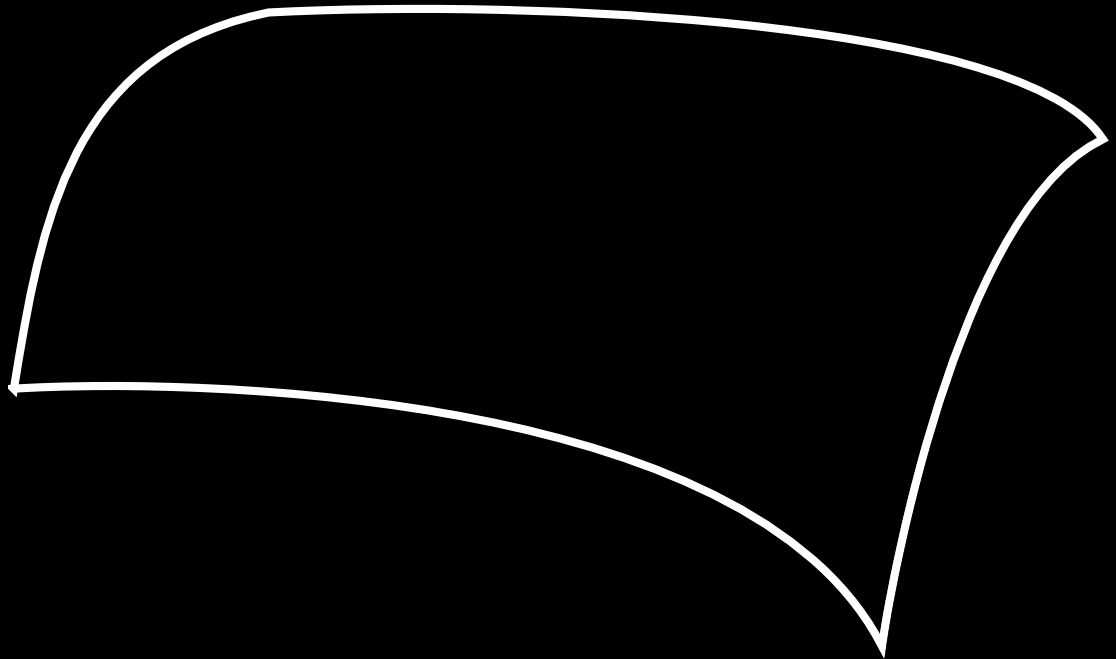
- **Encoding real samples** into their latent representations



$$\mathbf{z}_{\mathbf{mix}} = a^*z_1 + b^*z_2 + c^*z_3$$
$$a+b+c = 1.0$$

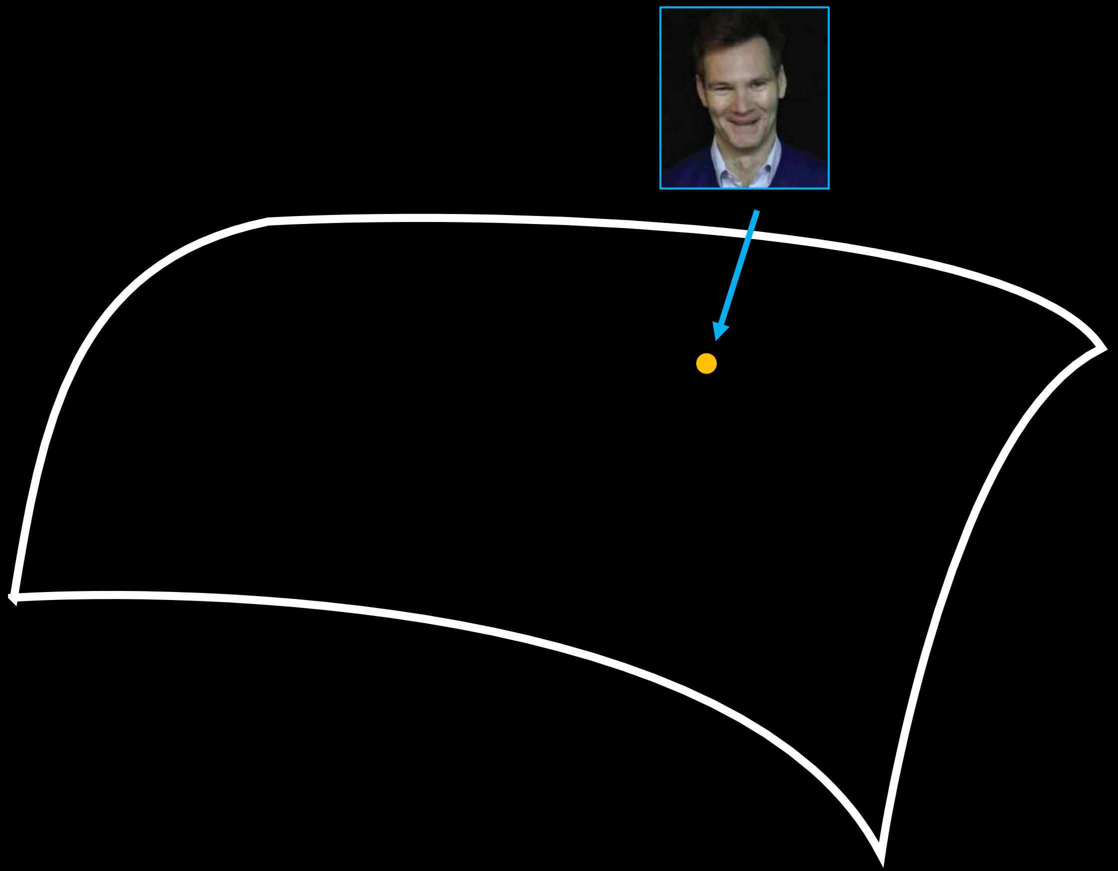
- Then using these, we can **generate interpolations**

AutoEncoder specific interaction II.



This is a **high dimensional space** (like this *cloth* shape in 3D, but in 512D instead), in which we can explore relations between data.

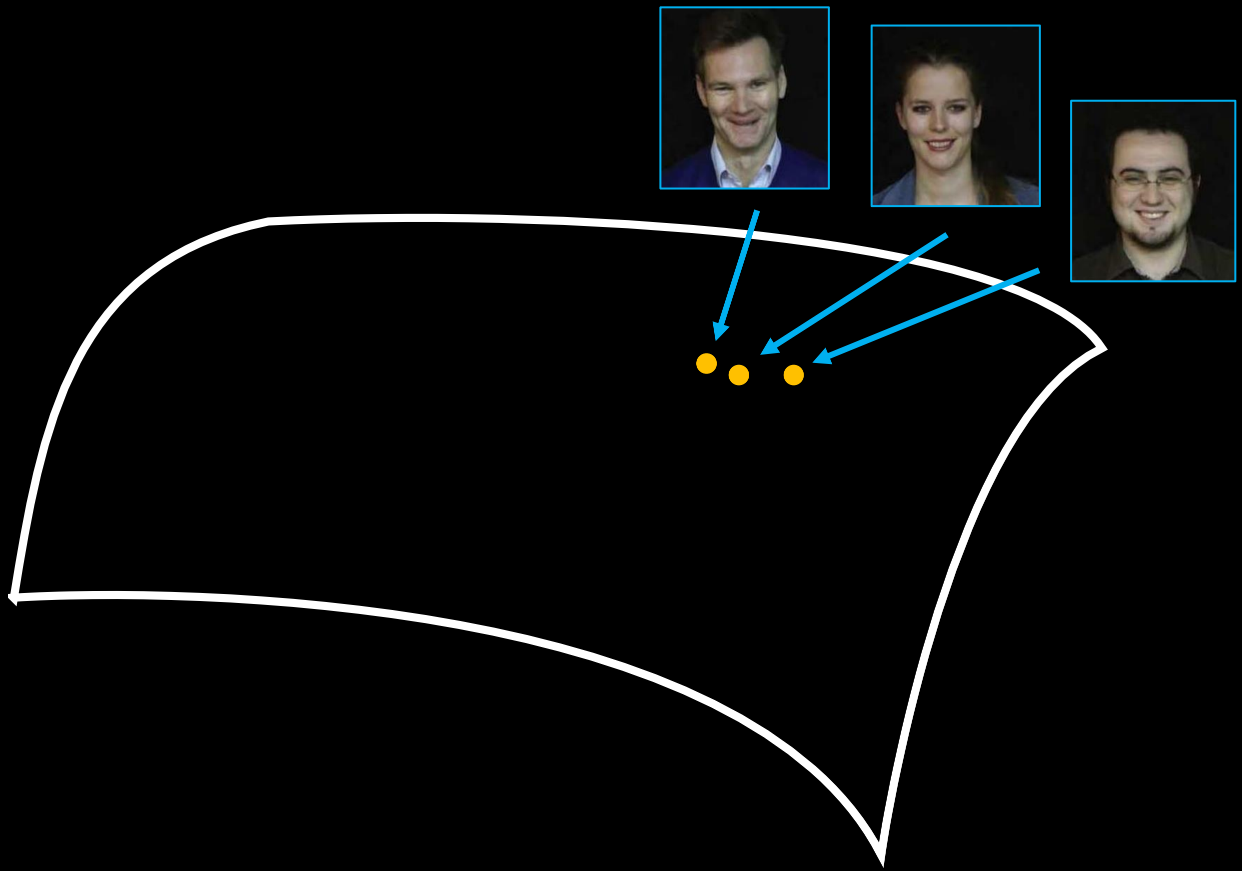
AutoEncoder specific interaction II.



This is a **high dimensional space** (like this *cloth* shape in 3D, but in 512D instead), in which we can explore relations between data.

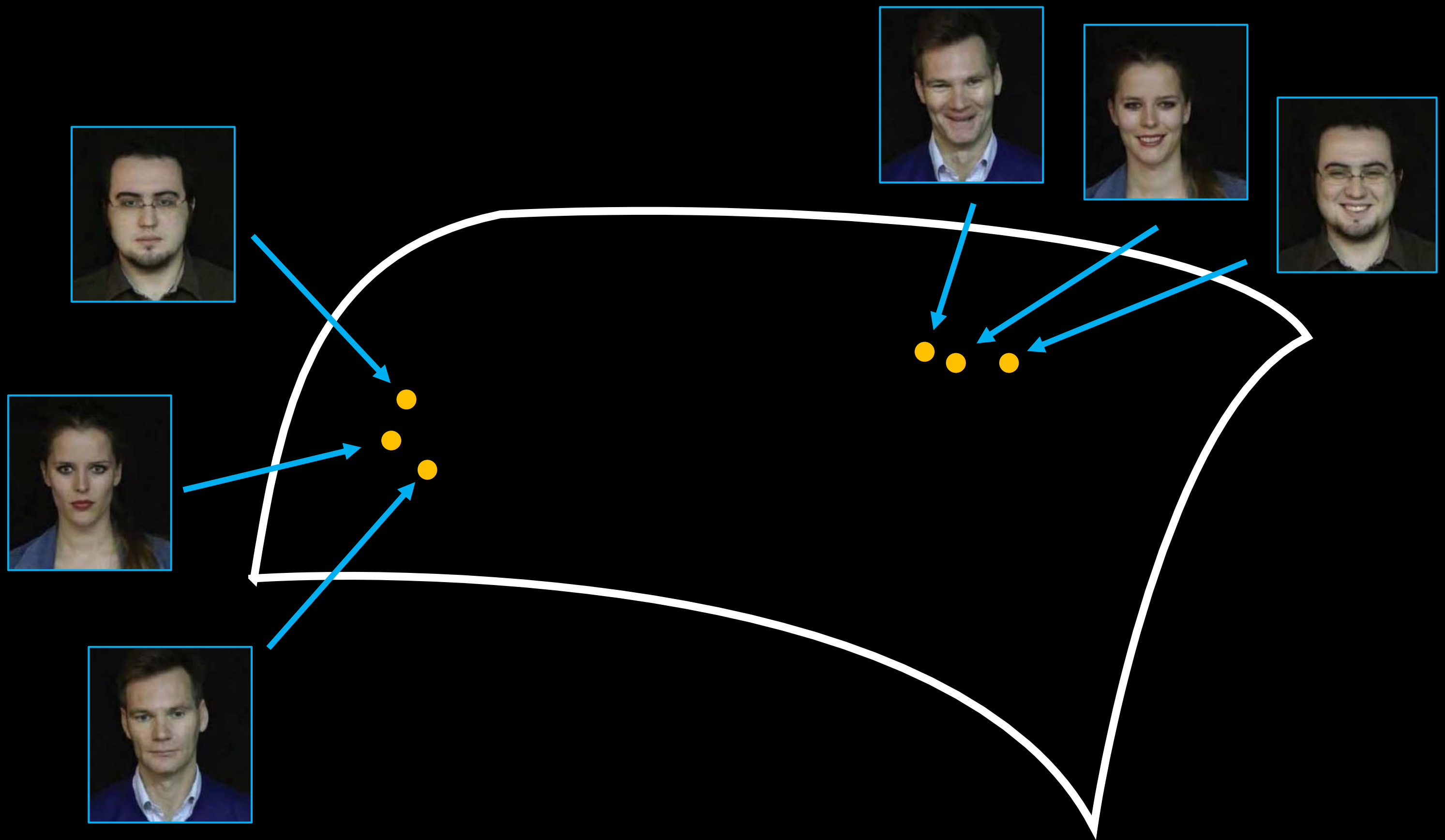
Each **point** in this space **is a vector of 512 numbers** (and so each point is an encoded real sample – and can also be used to generate an image)

AutoEncoder specific interaction II.



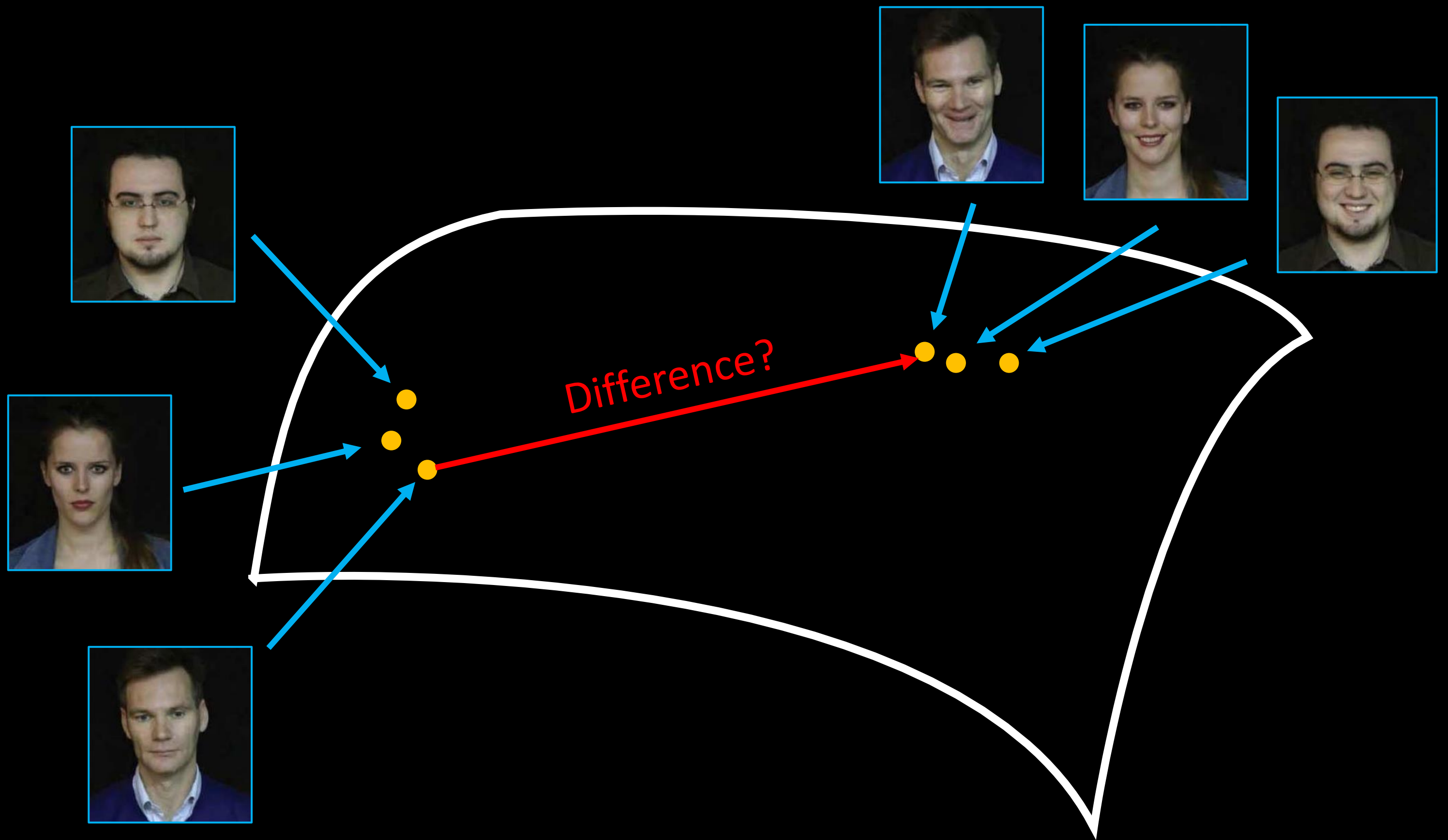
If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) …

AutoEncoder specific interaction II.



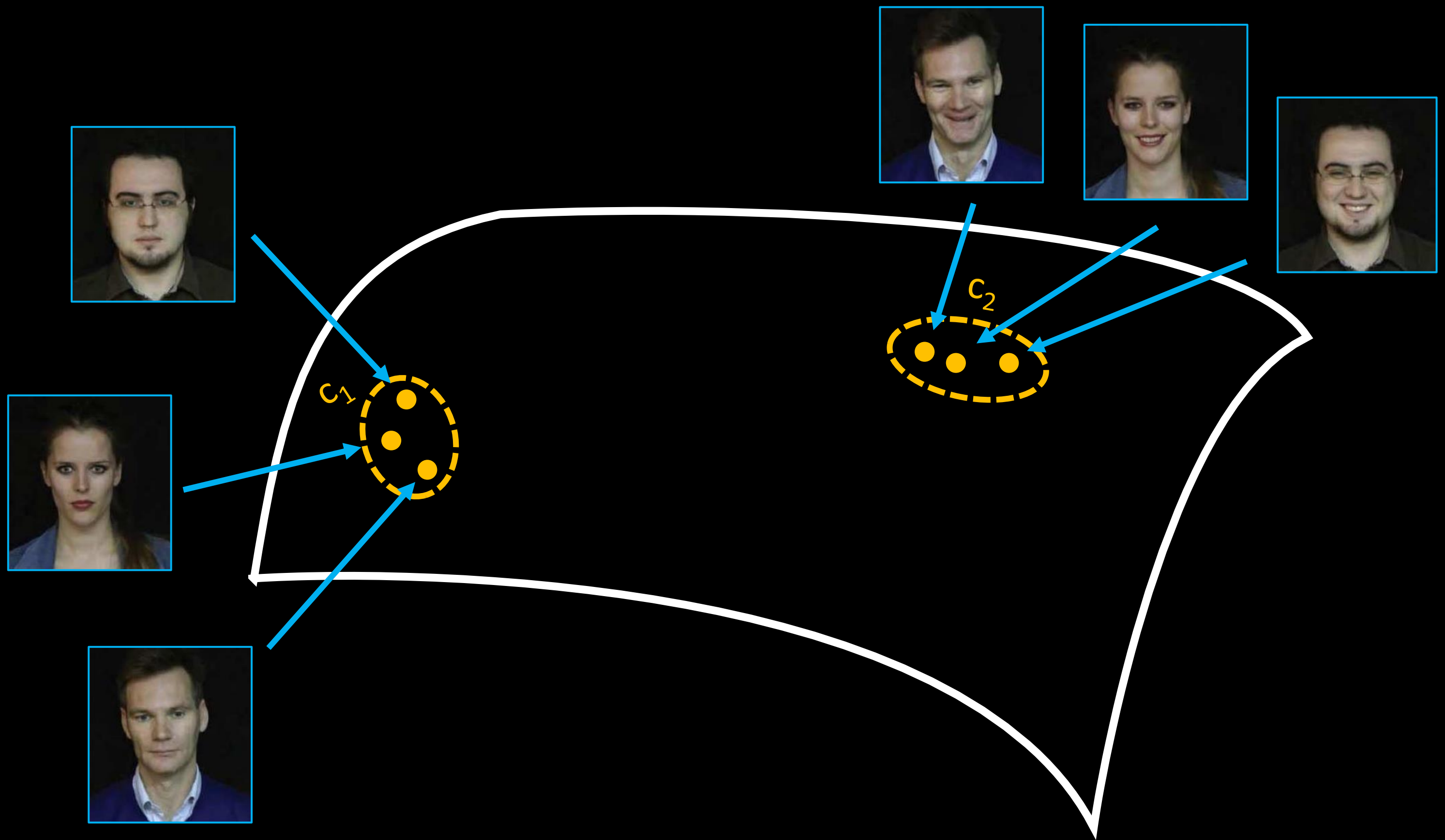
If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) ...

AutoEncoder specific interaction II.



If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) …

AutoEncoder specific interaction II.

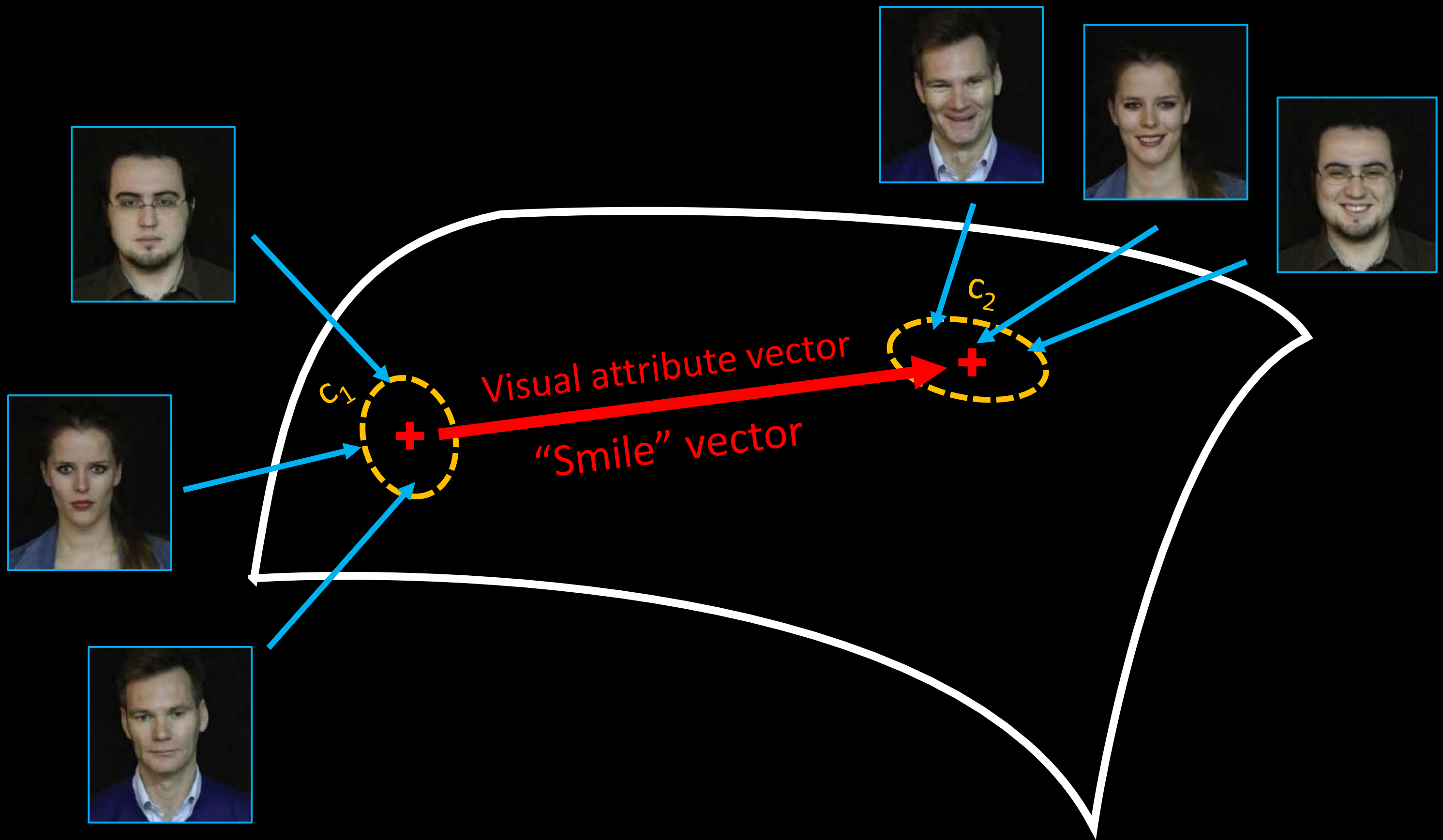


If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) …

We can find clusters and check their relative positions:

v = centroid of c_1 – centroid of c_2

AutoEncoder specific interaction II.

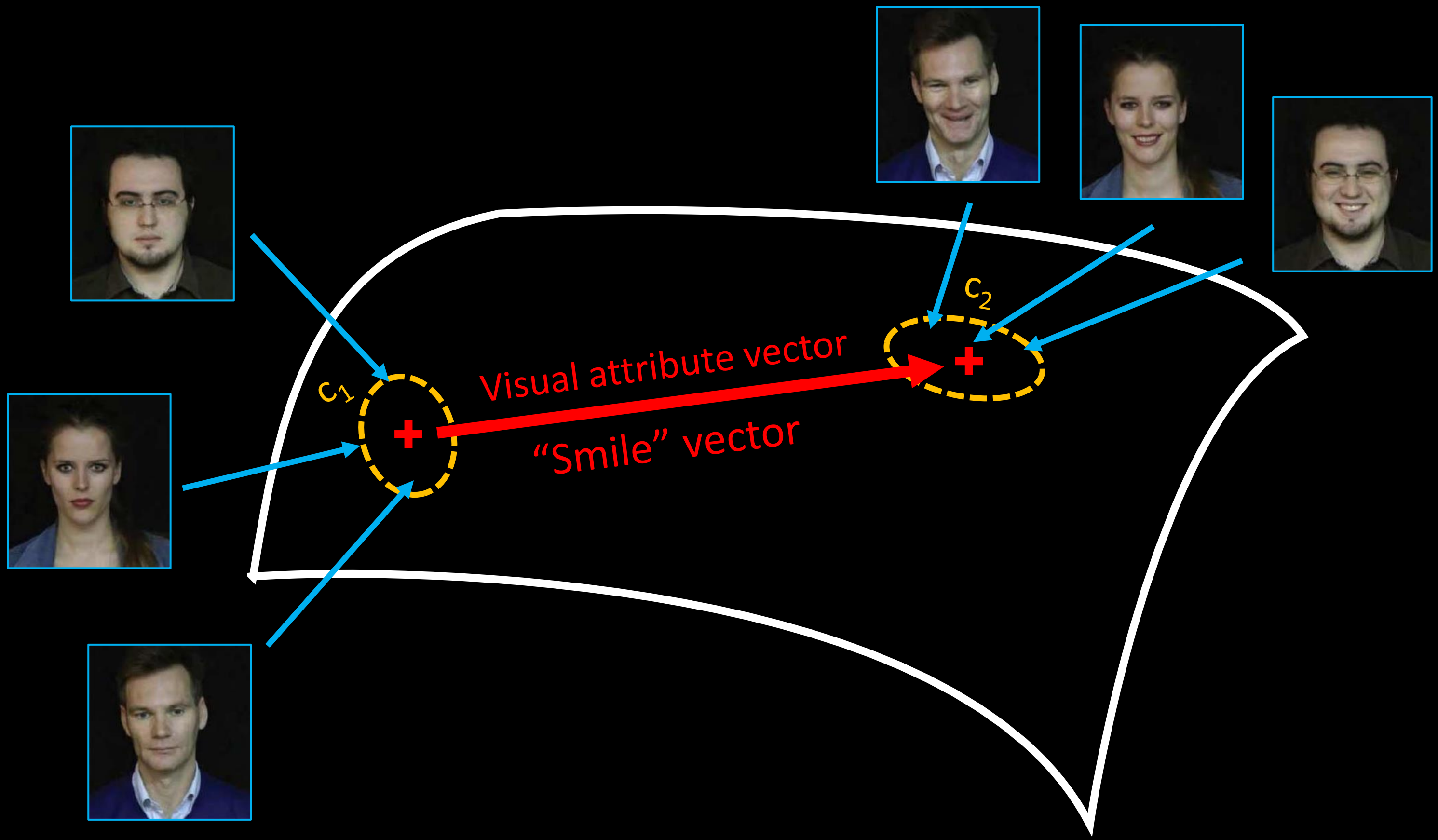


If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) …

We can find clusters and check their relative positions:

$\mathbf{v}$ = centroid of c_1 – centroid of c_2

AutoEncoder specific interaction II.



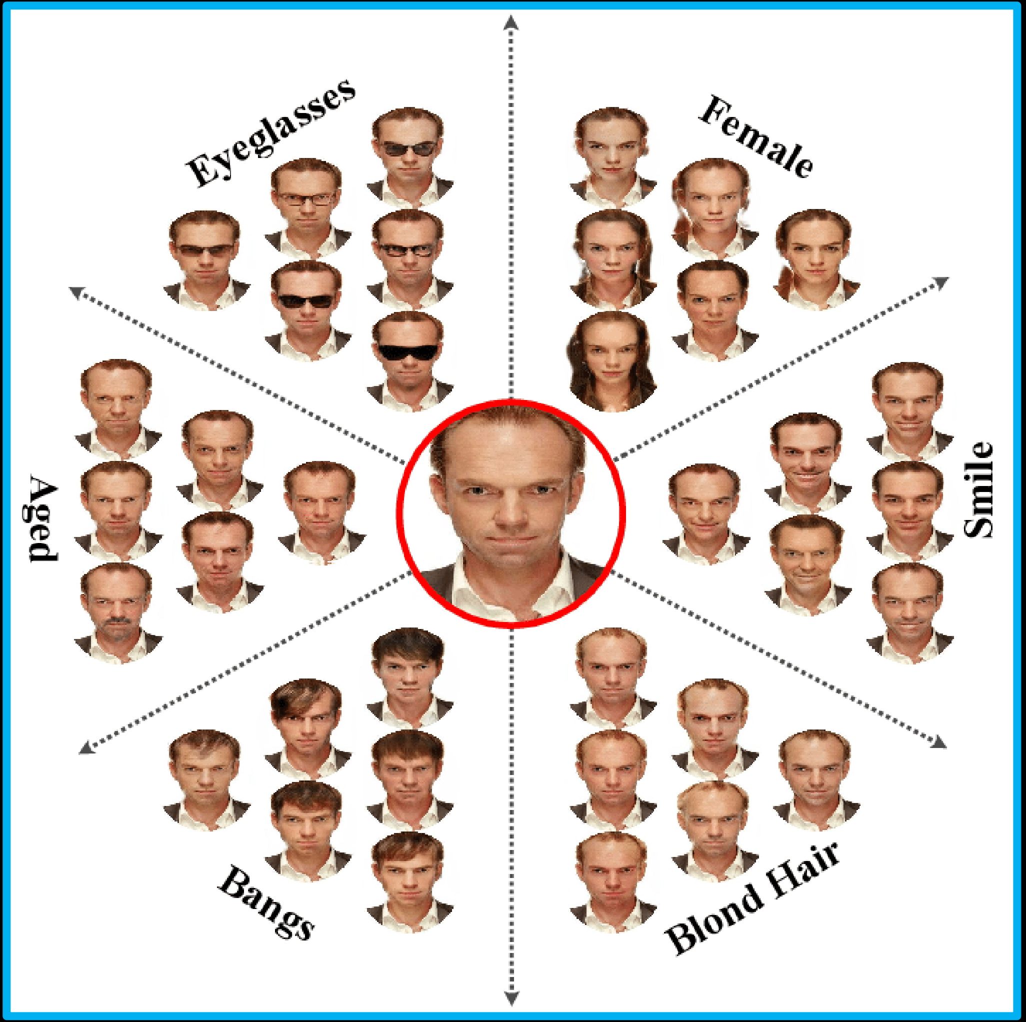
If we encoded images *with some property* and *without this property* (for example: **smiling / neutral** expression) …

We can find clusters and check their relative positions:

$\mathbf{v}$ = centroid of c_1 – centroid of c_2

- With labeled datasets we can extract **visual attribute vectors**
- We can call this **latent space arithmetic**

Visual attribute vectors



- **Visual attribute vectors:**
  - Eyeglasses vector
  - Smile vector
  - Blond hair vector
  - Bangs vector
  - Age vector
  - "Female" vector
- *PS: Heavily depends on what we labelled "smile", "blond" … or "female" …*

>> See [Animation link] <<

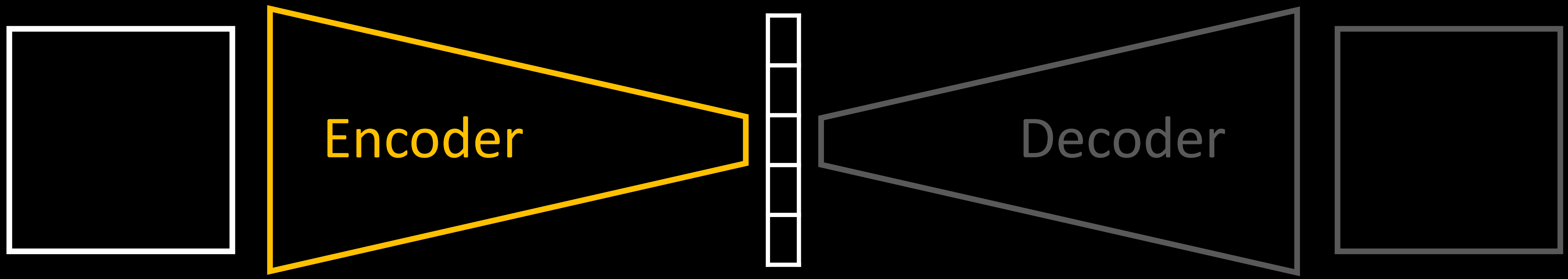
PS2: this is secretly a special implementation of a GAN ([link]) , but it illustrates this point nicely …

Side note: Diversity of datasets …

- **Celeb A** dataset
  - Diversity of samples
  - Choice of categories (planes of division)
  - Equal numbers of samples …
  - (etc … etc …)

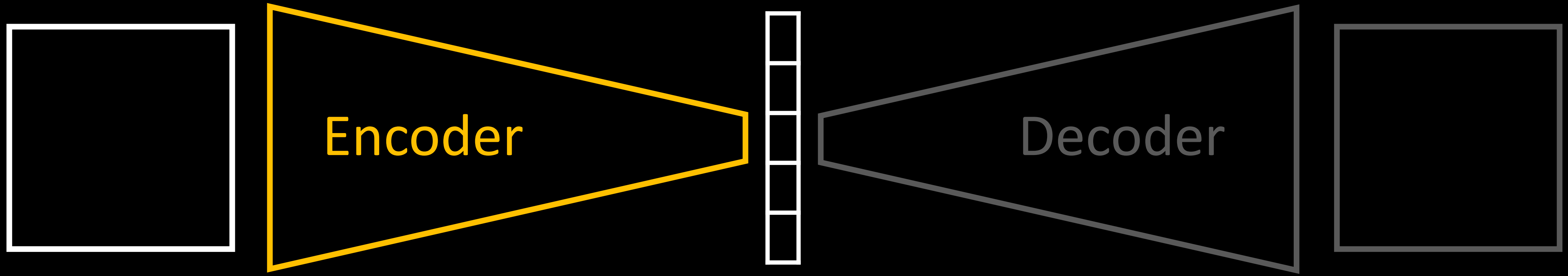
Weird uses of AutoEncoders I.

- When we need an **arbitrary feature extractor / transformer** of image -> feature



Weird uses of AutoEncoders I.

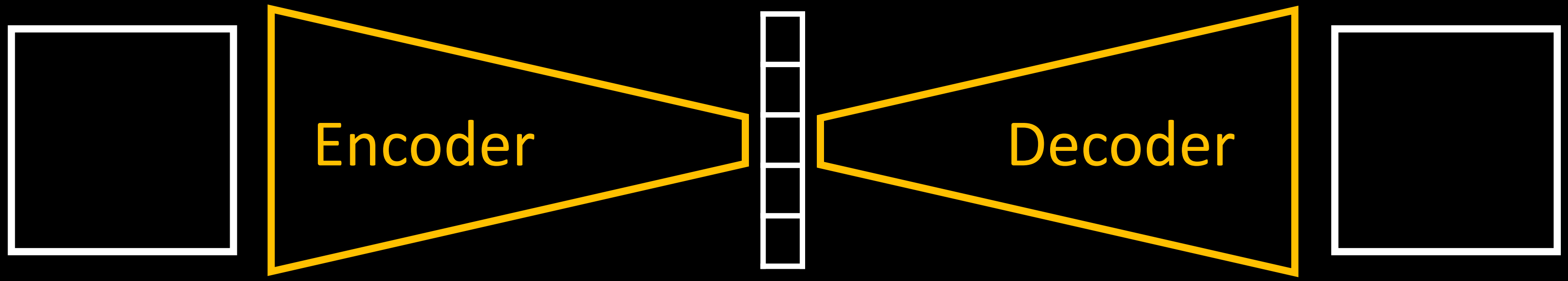
- When we need an **arbitrary feature extractor / transformer** of image -> feature
  - If we had labels (**supervised scenario**), we could used something similar like AlexNet …
  - Without labels (**unsupervised scenario**) we can instead use these methods – AEs or GANs



Intuition: if the encoder can create a “good enough” representation, that it can be used for reconstruction -> then we hope that it will be good in other scenarios as well

Weird uses of AutoEncoders II.

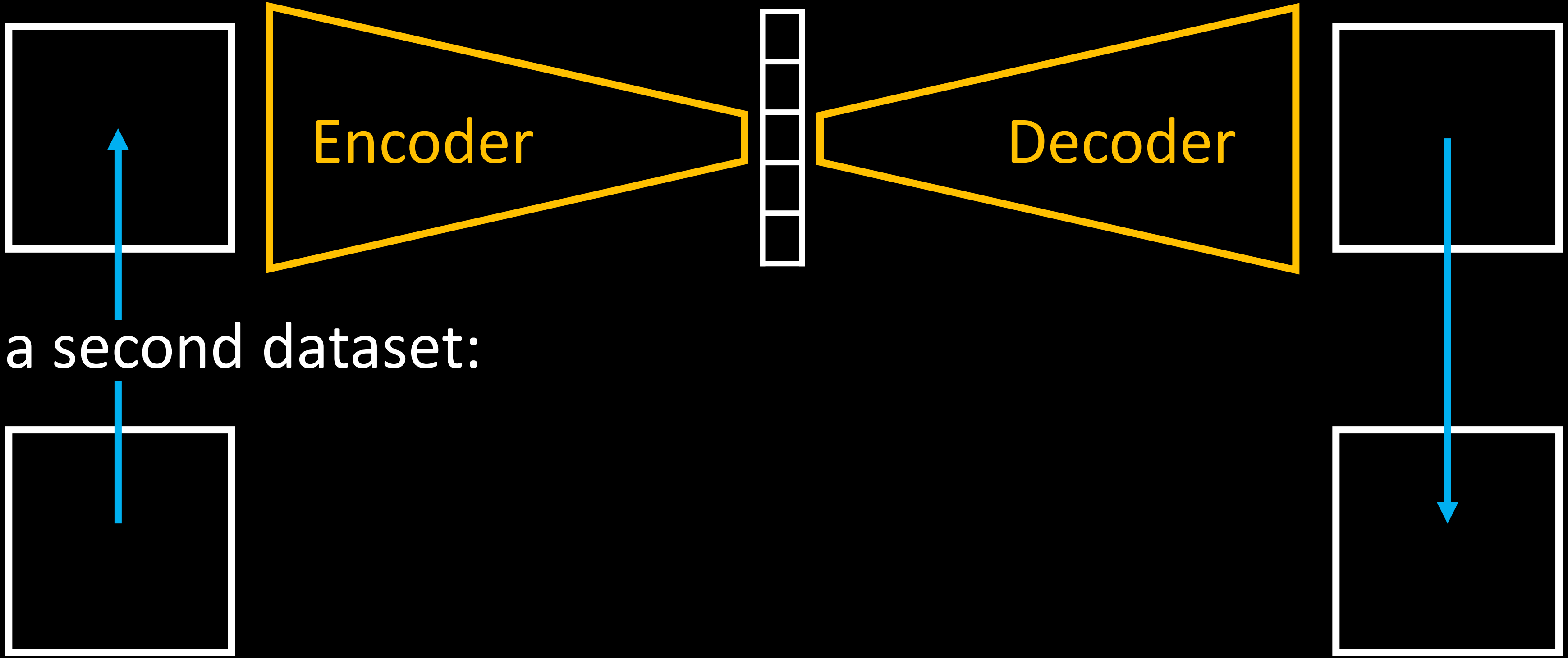
Train on a first dataset:



Intuition: **reinterpret material** with shapes/details/imagery of another material

Weird uses of AutoEncoders II.

Train on a first dataset:

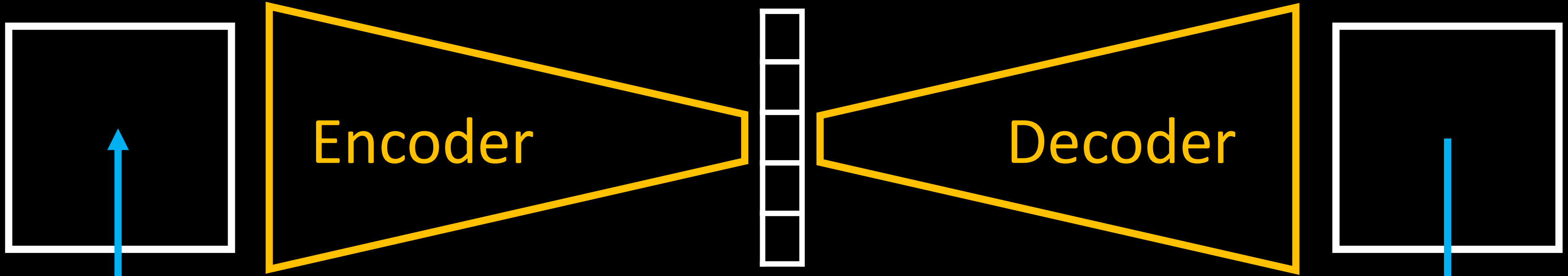


Use with a second dataset:

Intuition: **reinterpret material** with shapes/details/imagery of another material

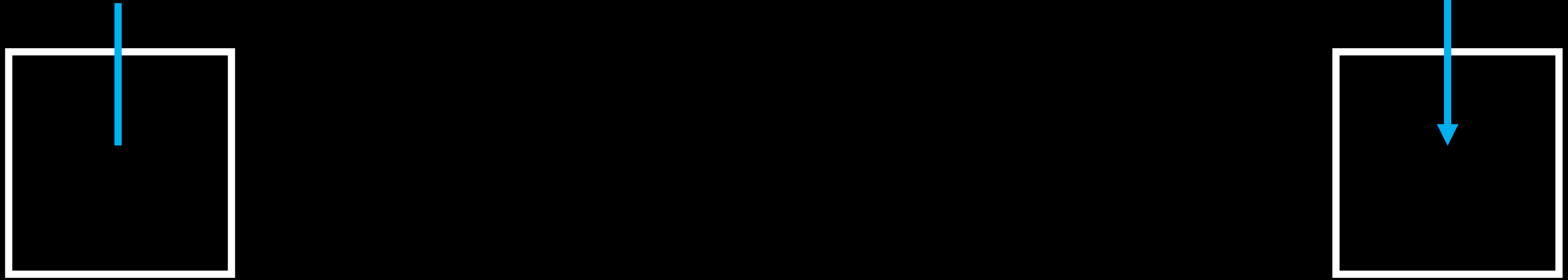
Weird uses of AutoEncoders II.

Train on a first dataset:



Example [video 1]

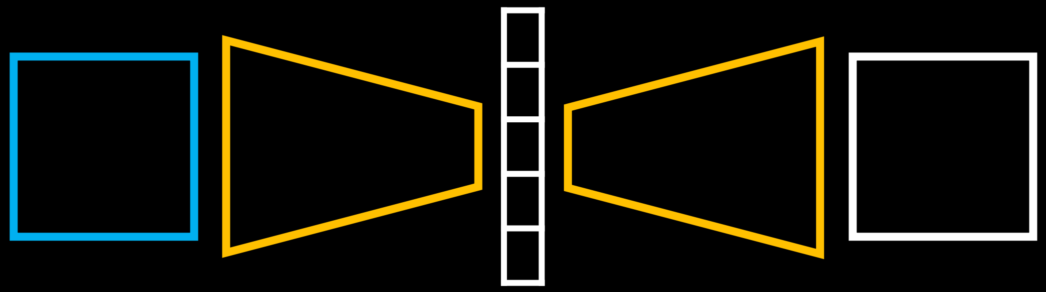
Use with a second dataset:



Example [video 2]

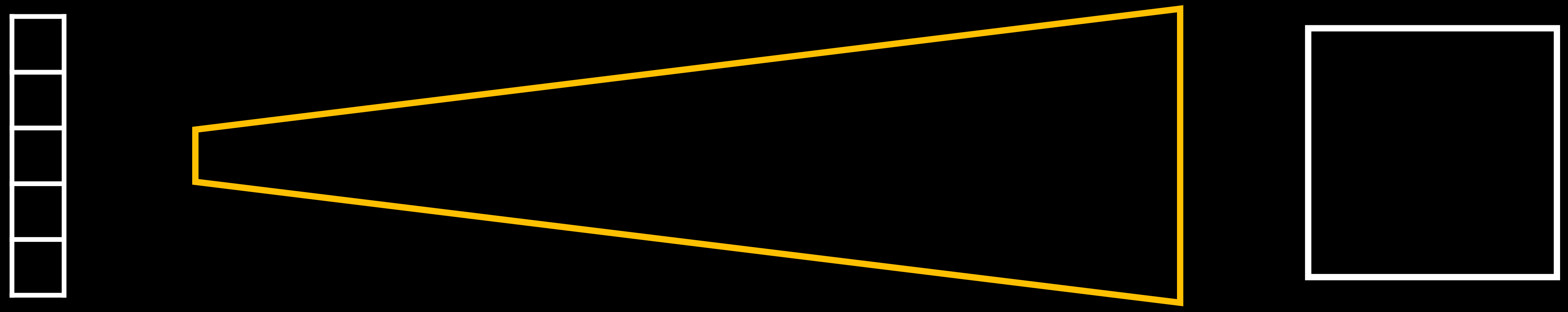
Intuition: **reinterpret material** with shapes/details/imagery of another material

More on: [blog]

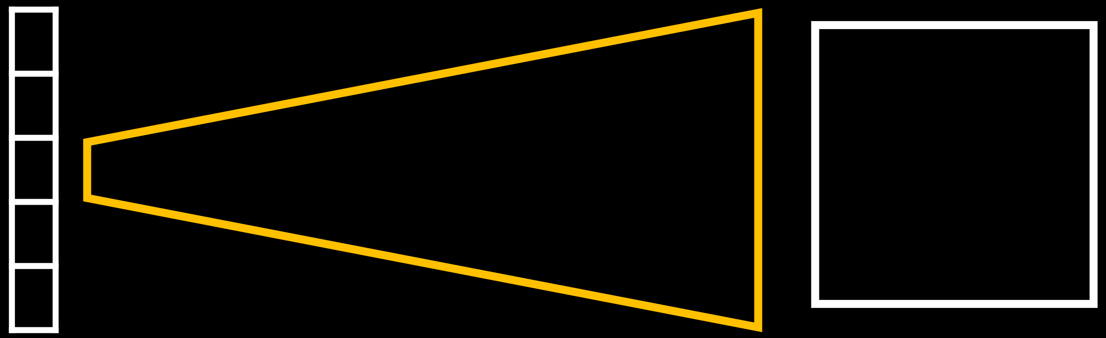


Pause 1

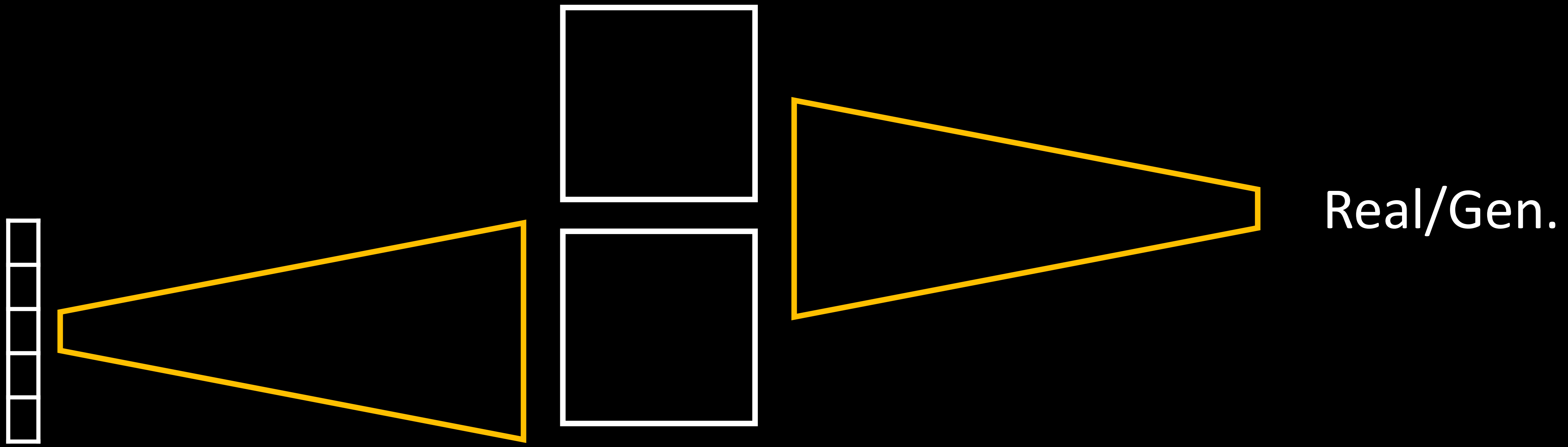
Back to the beginning ...



Generative Architectures

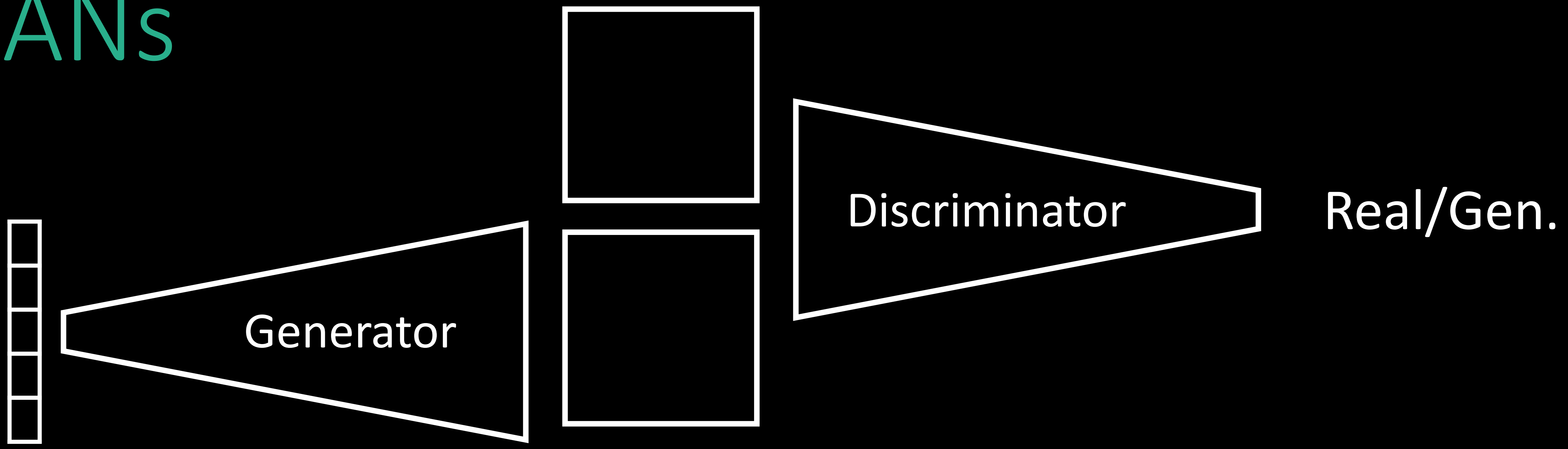


Generative Adversarial Networks



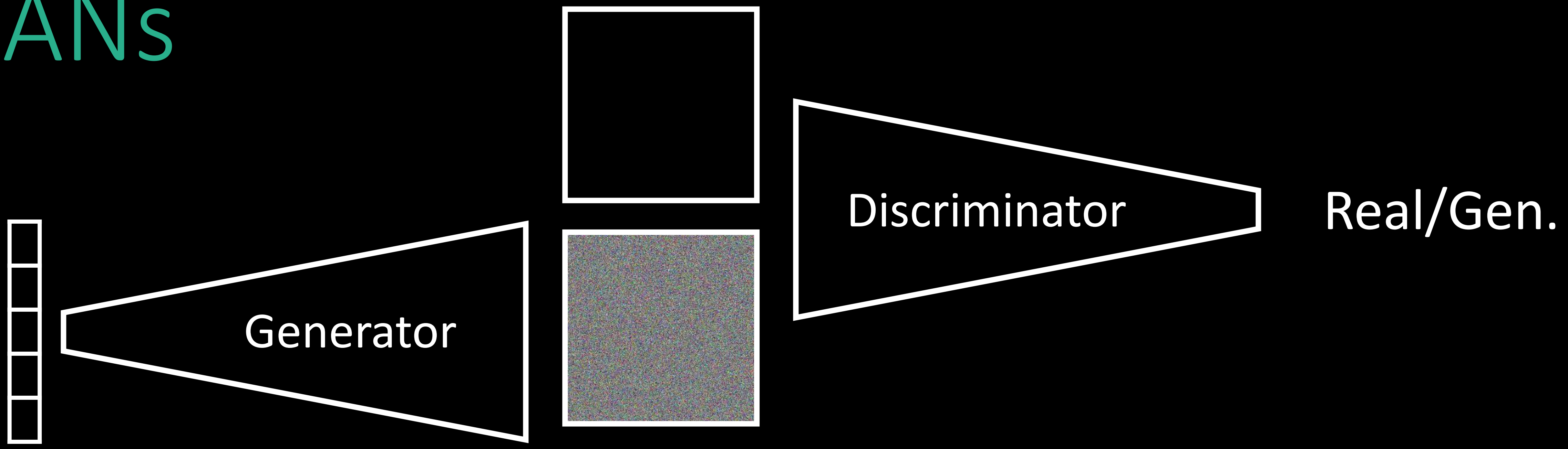
- Rephrase it as a **two-player game**. One network is supposed to generate images – and the other one is responsible for discriminating if they are real or generated.

GANs



Furthermore this architecture uses a smart **training scheme**:

GANs



Furthermore this architecture uses a smart **training scheme**:

- The **Generator network starts with** generating basically **noise**

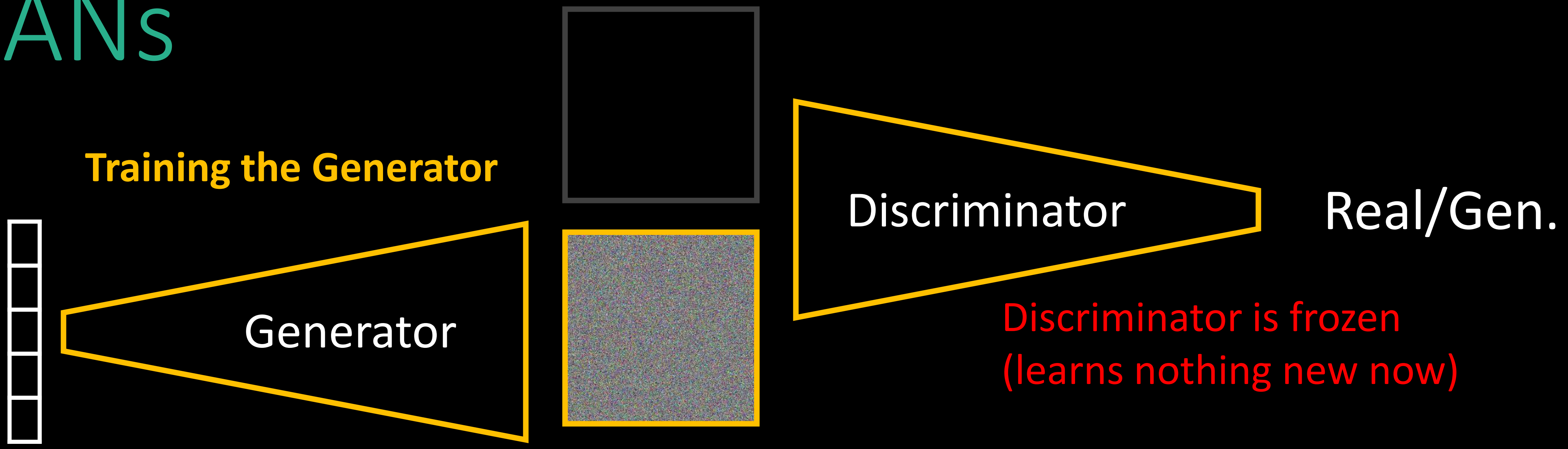
(it has learned nothing yet,
its completely rubbish)

GANs

Furthermore this architecture uses a smart **training scheme**:

- The **Generator network starts with** generating basically **noise**
- We **train the Discriminator** on the task to distinguish between the real dataset and the noise – afterwards we freeze it!

GANs



Furthermore this architecture uses a smart **training scheme**:

- The **Generator network starts with** generating basically **noise**
- We **train the Discriminator** on the task to distinguish between the real dataset and the noise – afterwards we freeze it!
- We **train the Generator** to create images which would fool the Discriminator – in doing so, we propagate error all the way from Disc.

GANs

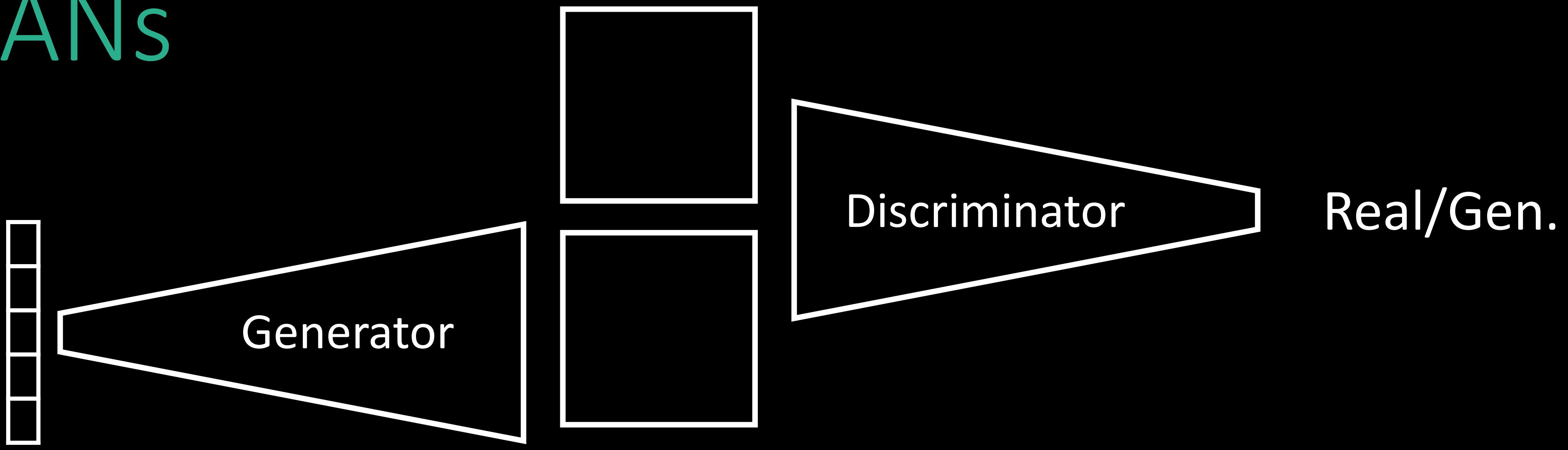
One iteration:

- **Train the Discriminator + Train the Generator**

We need to **repeat this training process** by switching between Discriminator and Generator while **freezing** the other one

- They both get better at their tasks.

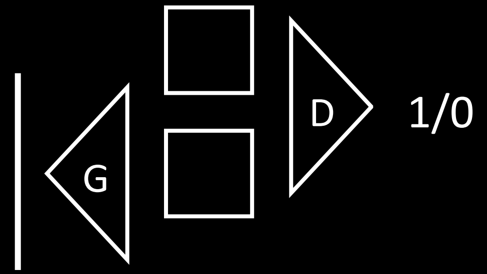
GANs

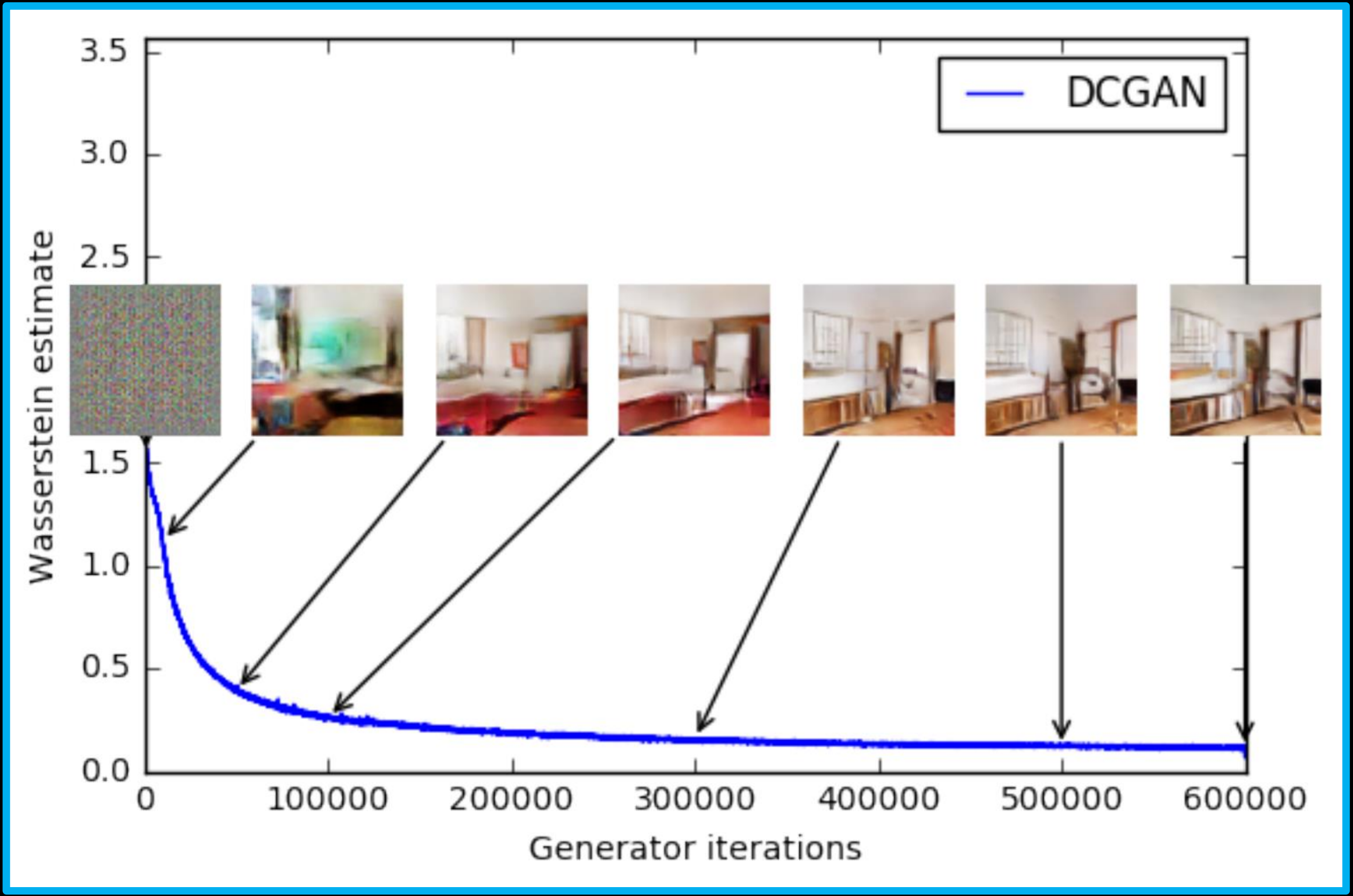


Metaphor:

Generator *network is a* ***counterfeit artist****, it's perfecting the craft of mimicking real artworks.* ***Discriminator*** *is an* ***art curator*** *– they are trying to spot the fake (generated) images. It's an arms race in which both get better at their job!*

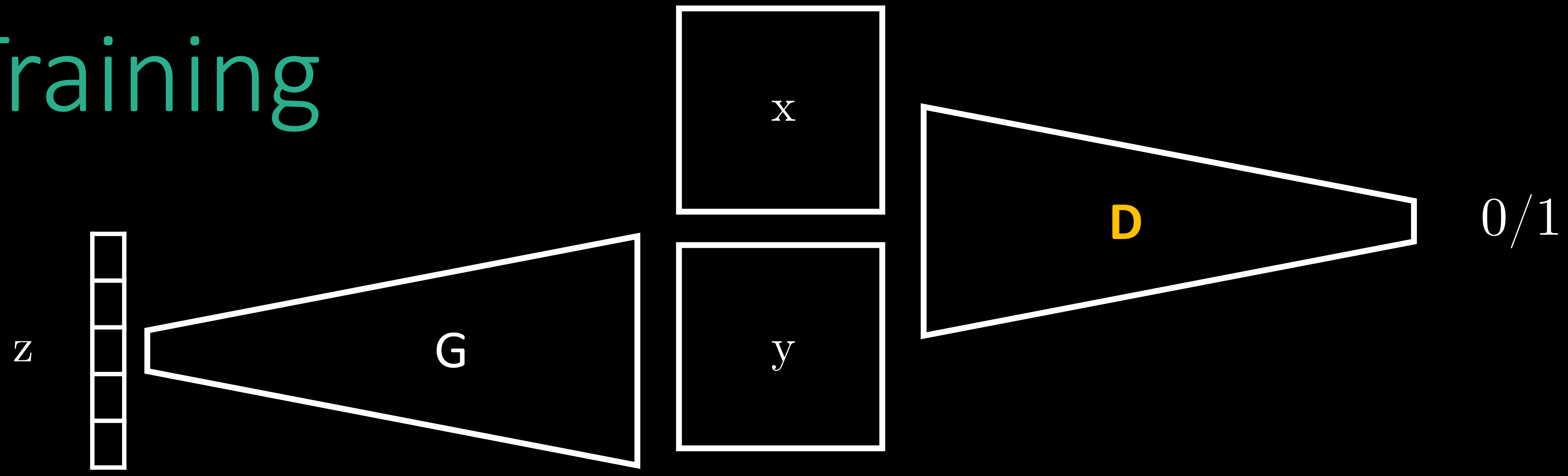
GANs





Over the iterations, the Generator get's better in mimicking the real data (without actually ever having a direct access to them) …

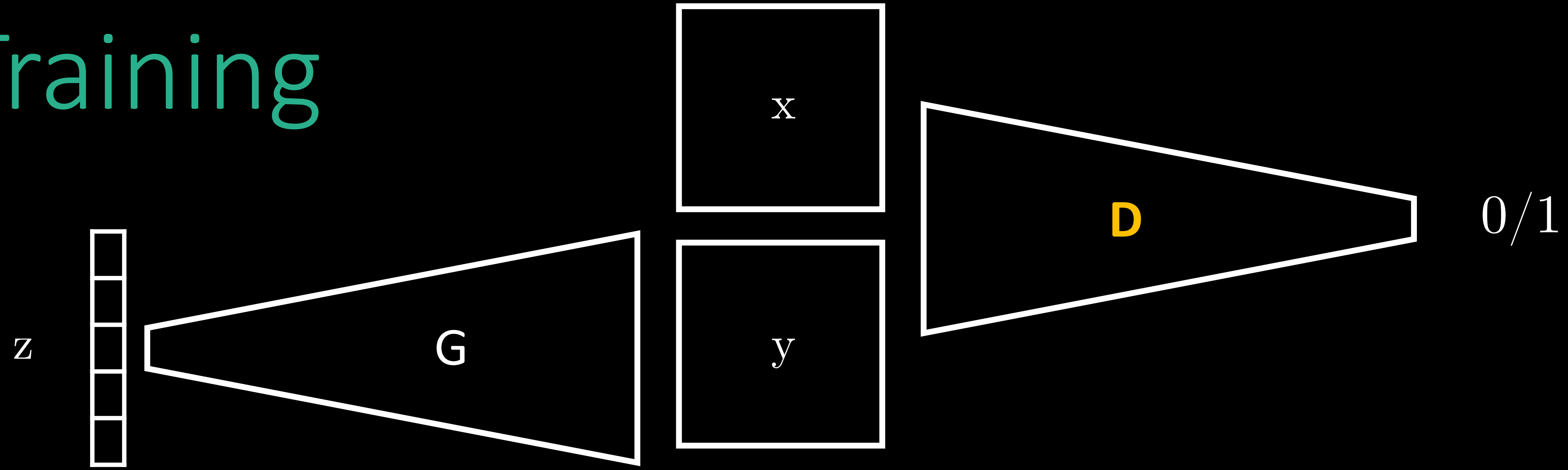
Training



- **Discriminator**

$$\mathrm{x}$$
$$\mathrm{y} = \mathrm{G}(\mathrm{z})$$

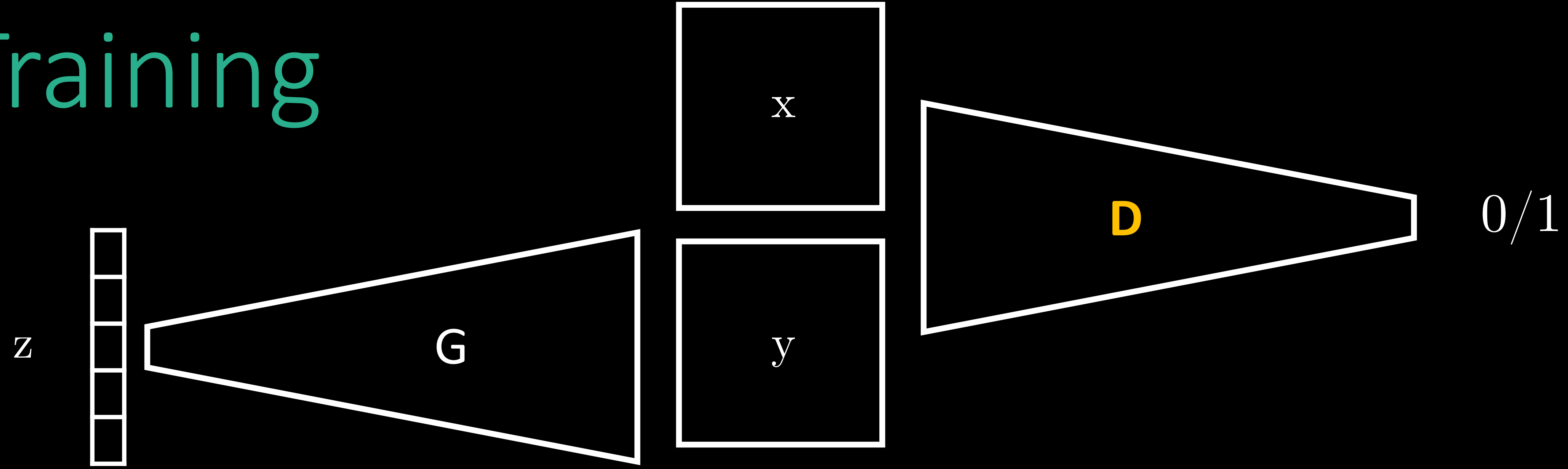
Training



- **Discriminator**

$$\left.\begin{array}{ll} \mathrm{x} & \ldots \text{ labeled with 1 (real)} \\ \mathrm{y} = \mathrm{G}(\mathrm{z}) & \ldots \text{ labeled with 0 (fake)} \end{array}\right\} \text{Classification task!}$$

Training

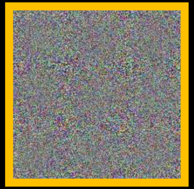


- **Discriminator**

$$\left.\begin{array}{lll} x & & \text{... labeled with 1 (real)} \\ y &= G(z) & \text{... labeled with 0 (fake)} \end{array}\right\}$$

Classification task!

Data:

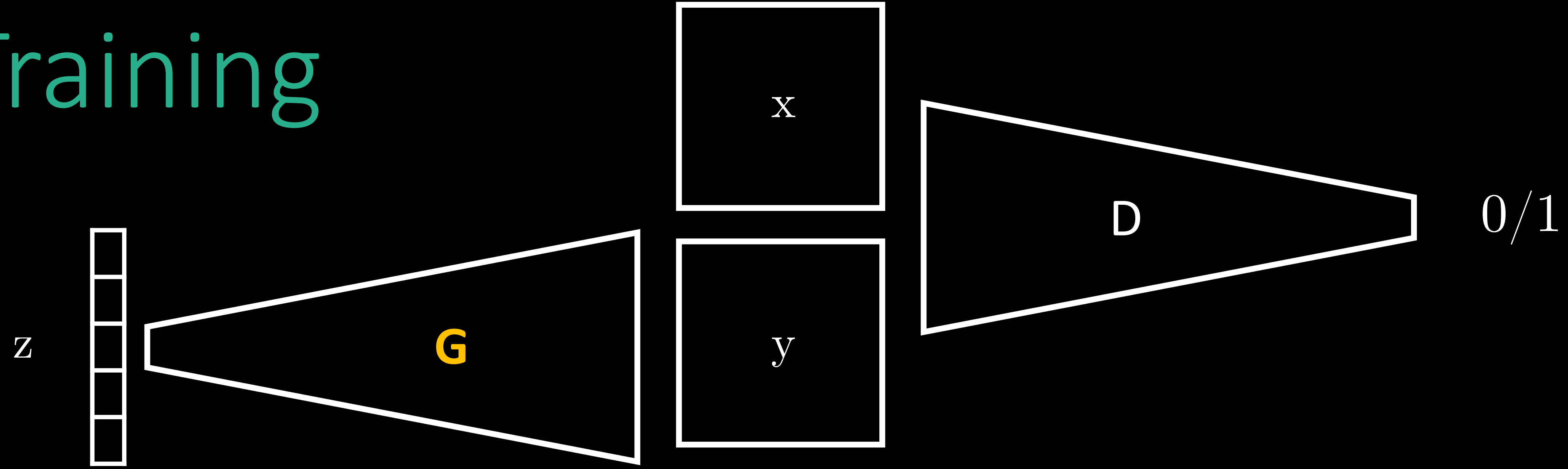


Labels:

1

0

Training

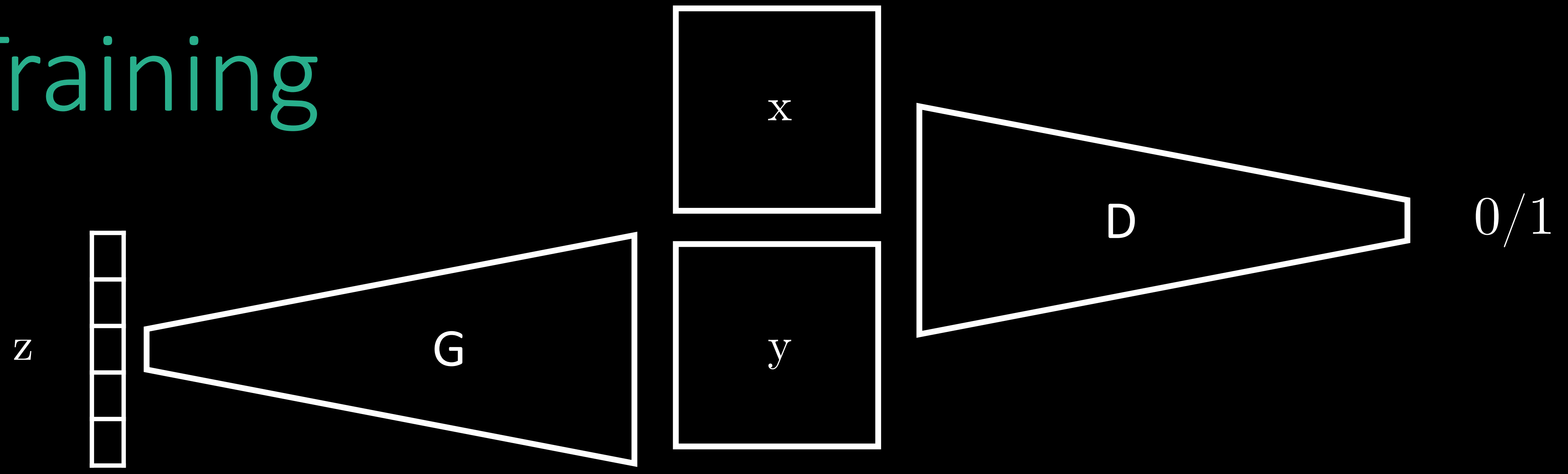


- Discriminator $\quad \mathrm{x} \quad$ ... labeled with 1 (real)
  $\mathrm{y} = \mathrm{G(z)}$ ... labeled with 0 (fake)
  $\Big\}$ Classification task!

- **Generator** $\quad$ Train G so that G(z) is **mislabeled by the discriminator**
  $\mathrm{y} = \mathrm{G(z)}$ ... labeled with 1 (real)

PS: Generator has no access to the real data (x)

Training

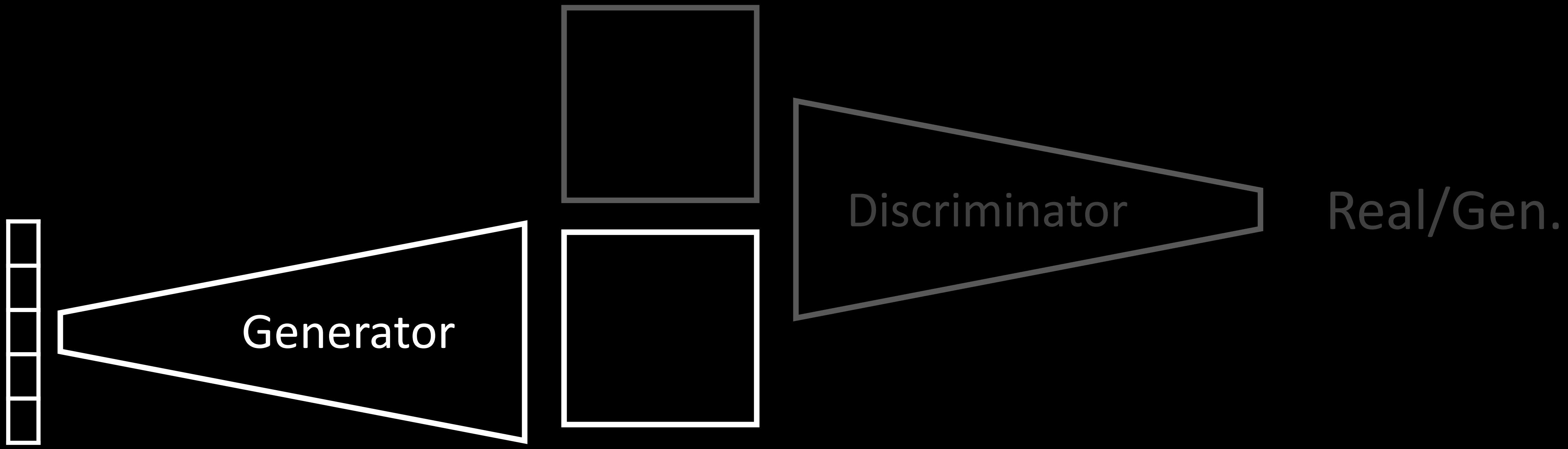


- Discriminator $\quad \mathrm{x} \quad$ ... labeled with 1 (real)
  $\mathrm{y} = \mathrm{G}(\mathrm{z}) \quad$ ... labeled with 0 (fake)
  Classification task!

- Generator $\quad$ Train G so that G(z) is **mislabeled by the discriminator**
  $\mathrm{y} = \mathrm{G}(\mathrm{z}) \quad$ ... labeled with 1 (real)

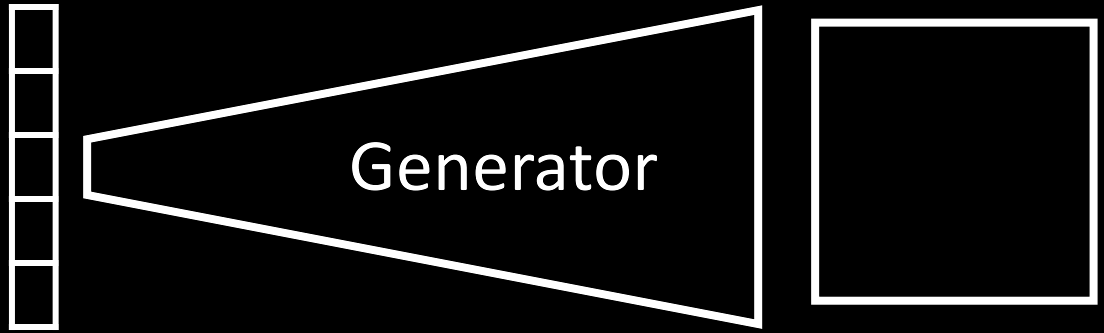
- This **training is a *bit* fragile** ... we need to balance between training D and G (so that one can't completely overtake the other) ... (*there are lots and lots of techniques*)

Interaction with a GAN



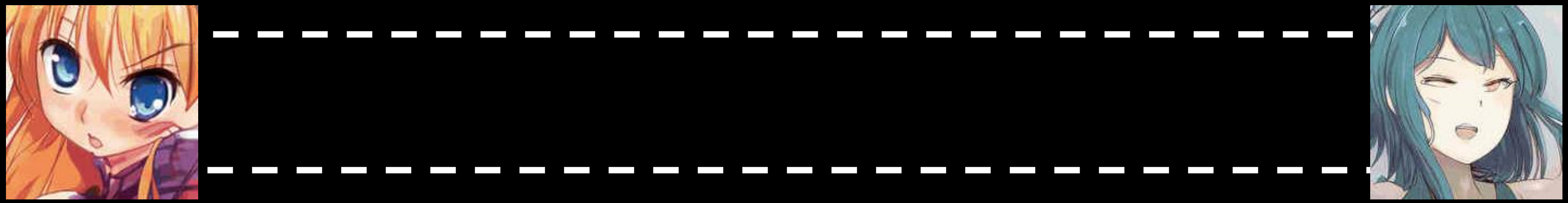
Interaction with a GAN

- We can do almost the same as with AutoEncoders before …



$z_1 \in \mathbb{R}^{512}$ $z_2 \in \mathbb{R}^{512}$

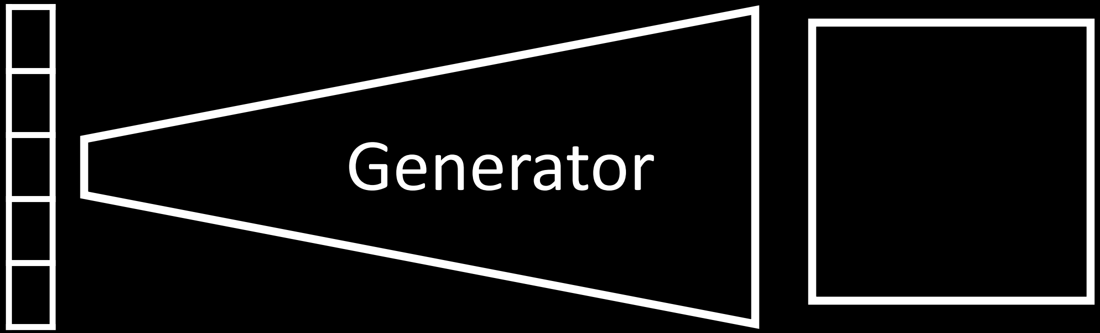
I_1 ->



<- I_2

Interaction with a GAN

- We can do almost the same as with AutoEncoders before …

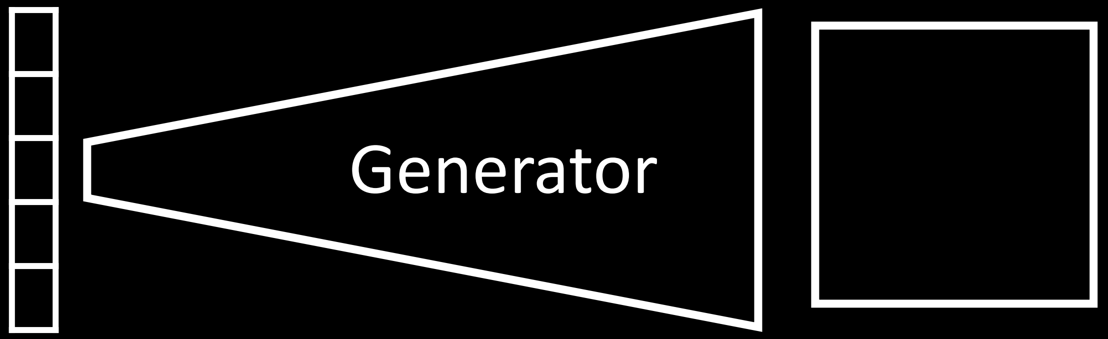


$z_1 \in \mathbb{R}^{512}$ $a * z_1 + (1.0 - a) * z_2$ $z_2 \in \mathbb{R}^{512}$

I_1 ->  <- I_2

Interaction with a GAN

- We can do almost the same as with AutoEncoders before …



…… surprisingly in-depth analysis: blog

$z_1 \in \mathbb{R}^{512}$ $\quad a*z_1 + (1.0\text{-}a)*z_2$ $\quad z_2 \in \mathbb{R}^{512}$

I_1 ->

<- I_2

Differences between GANs and AEs

- In the default version GANs don’t have an Encoder (*image to latent vector translation* network) = we can’t encode real samples
- **Why would we use GANs?**

Differences between GANs and AEs

- In the default version GANs don't have an Encoder (*image to latent vector translation* network) = we can't encode real samples
- **Why would we use GANs?**

(pairs of original and generated/reconstructed images)

AE

GAN

Differences between GANs and AEs

- In the default version GANs don't have an Encoder (*image to latent vector translation* network) = we can't encode real samples
- **Why would we use GANs?**

AE

-> **Blurry** results

Error from the square distance

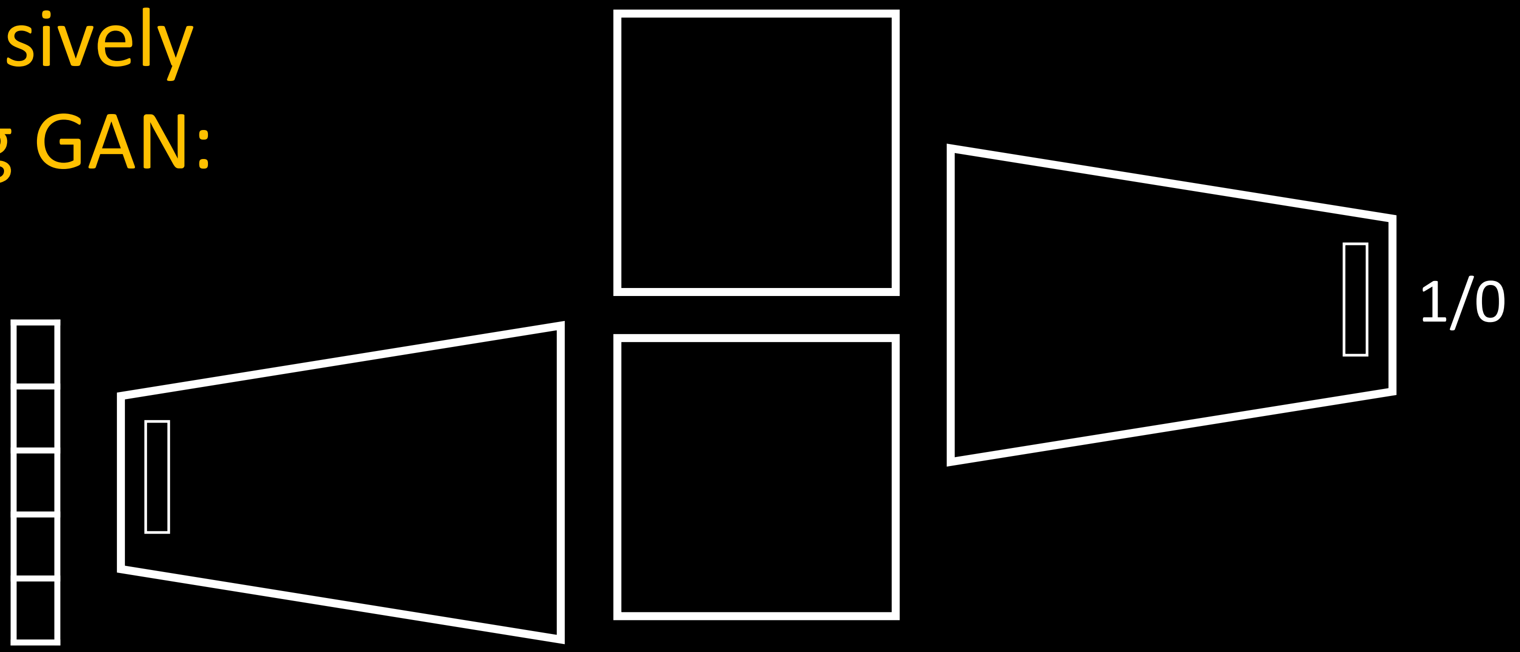
GAN

-> **Sharper**, more details

Error is propagated from G/D

Famous GAN architectures I.

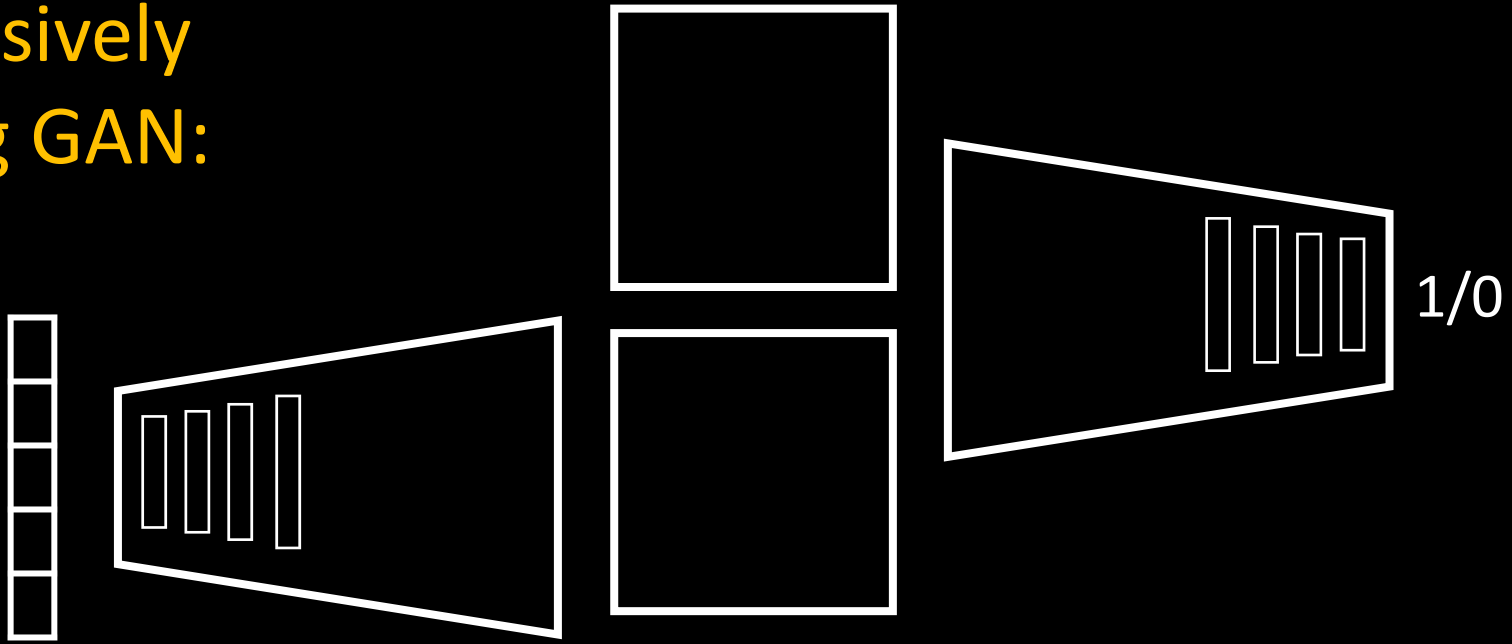
Progressively Growing GAN:



- Achieving high resolution through **iteratively adding layers into G and D**.

Famous GAN architectures I.

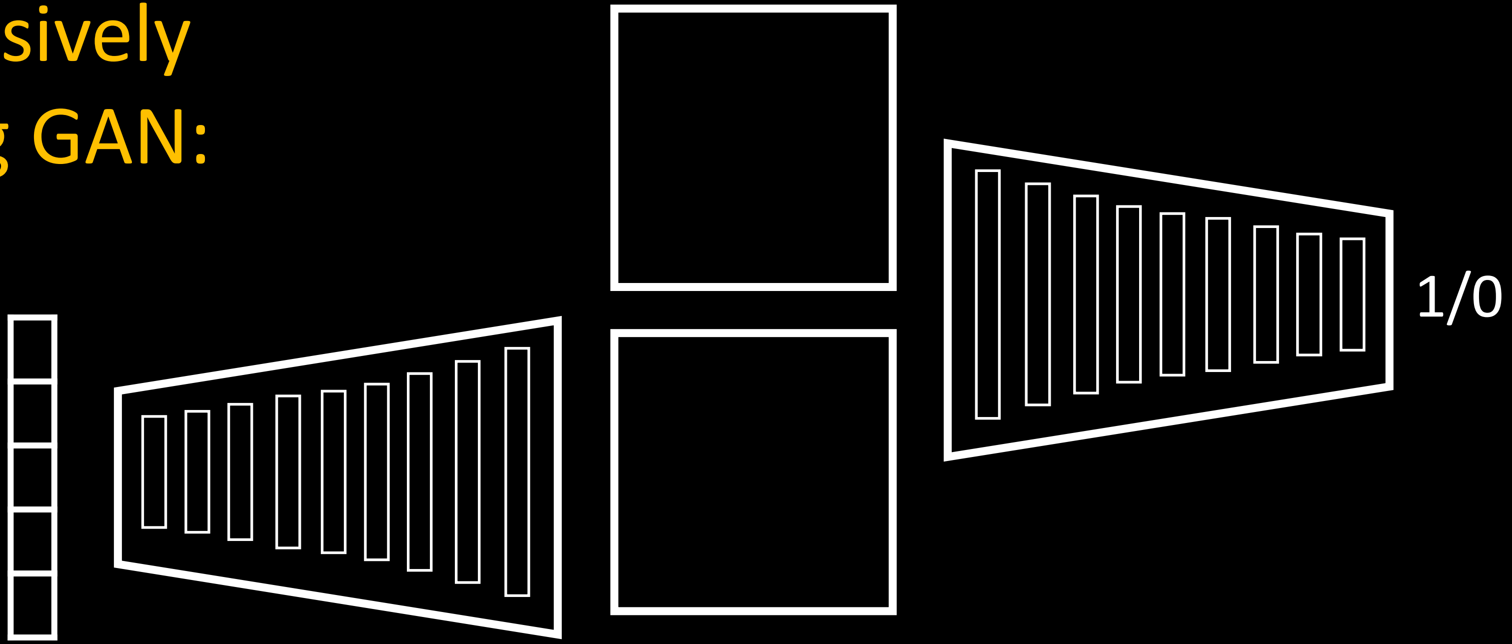
Progressively Growing GAN:



- Achieving high resolution through **iteratively adding layers into G and D**.

Famous GAN architectures I.

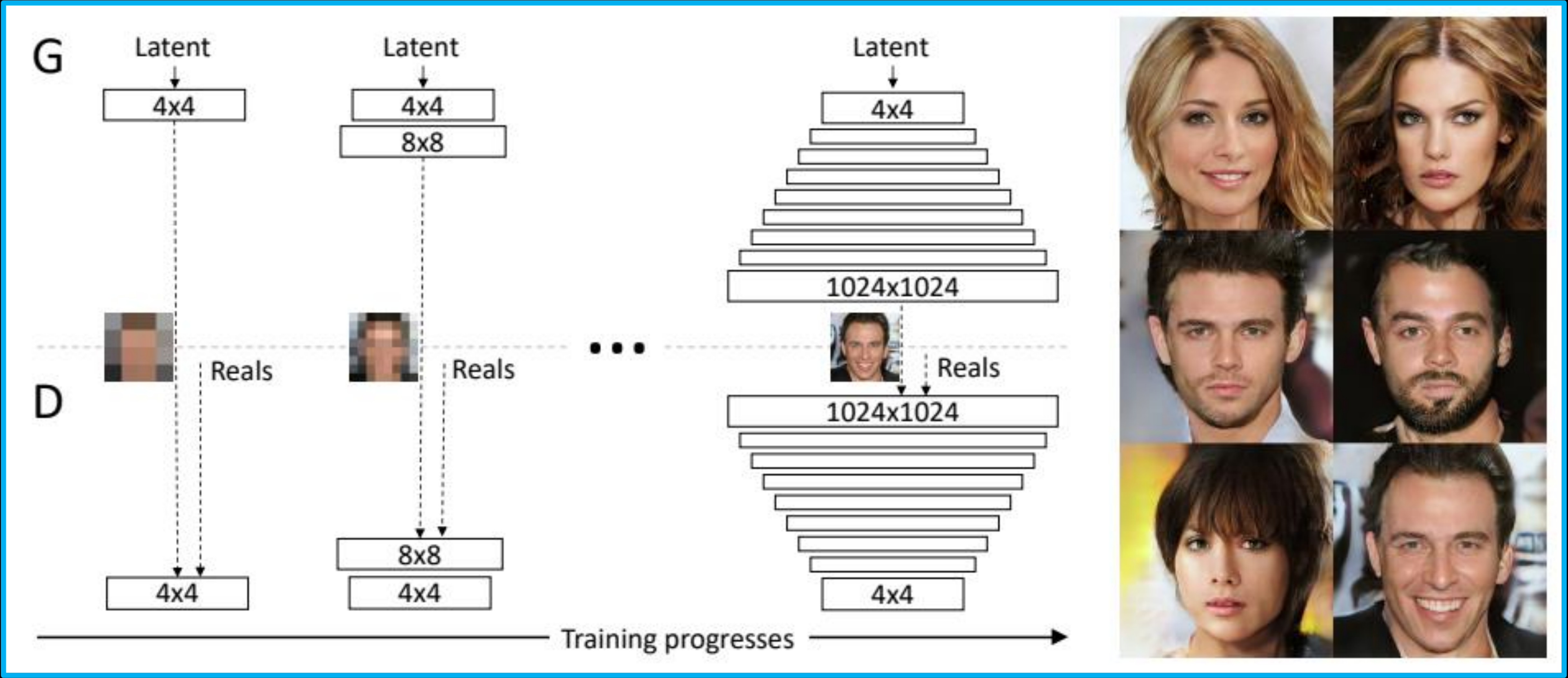
Progressively Growing GAN:



- Achieving high resolution through **iteratively adding layers into G and D**.
- *PS: Seems really stable even with custom datasets*

Progressively Growing GAN:

Video link



Famous GAN architectures II.

Big GAN

- Progressive GAN faced criticism from being able to generate data only from one domain …
- Here comes Google with ∞ clusters and compute. Training a model with added batch size (needs tons of memory) and training time -> **GAN with diverse dataset** (idea of having an *universal GAN rather than focused one*)

Famous GAN architectures II.

Big GAN

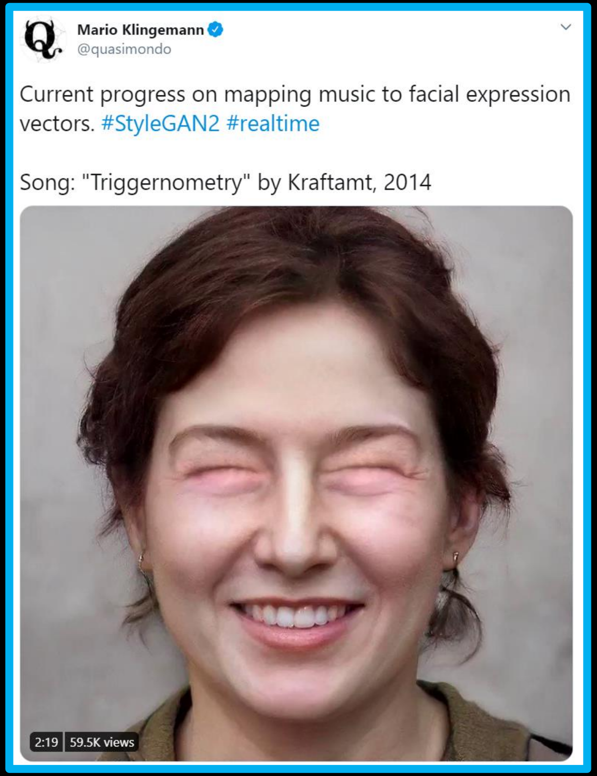
- Released models contain **many categories** (rather than focusing just on a faces dataset) …
- However training your own Big GAN model is not feasible … (whereas with Progressive GAN it is)

Weird uses of GANs I.

- Searching through latent space of existing models and finding *weird* vectors

Weird uses of GANs I.

- Searching through latent space of existing models and finding *weird* vectors



And mapping them … (for example from music analysis):

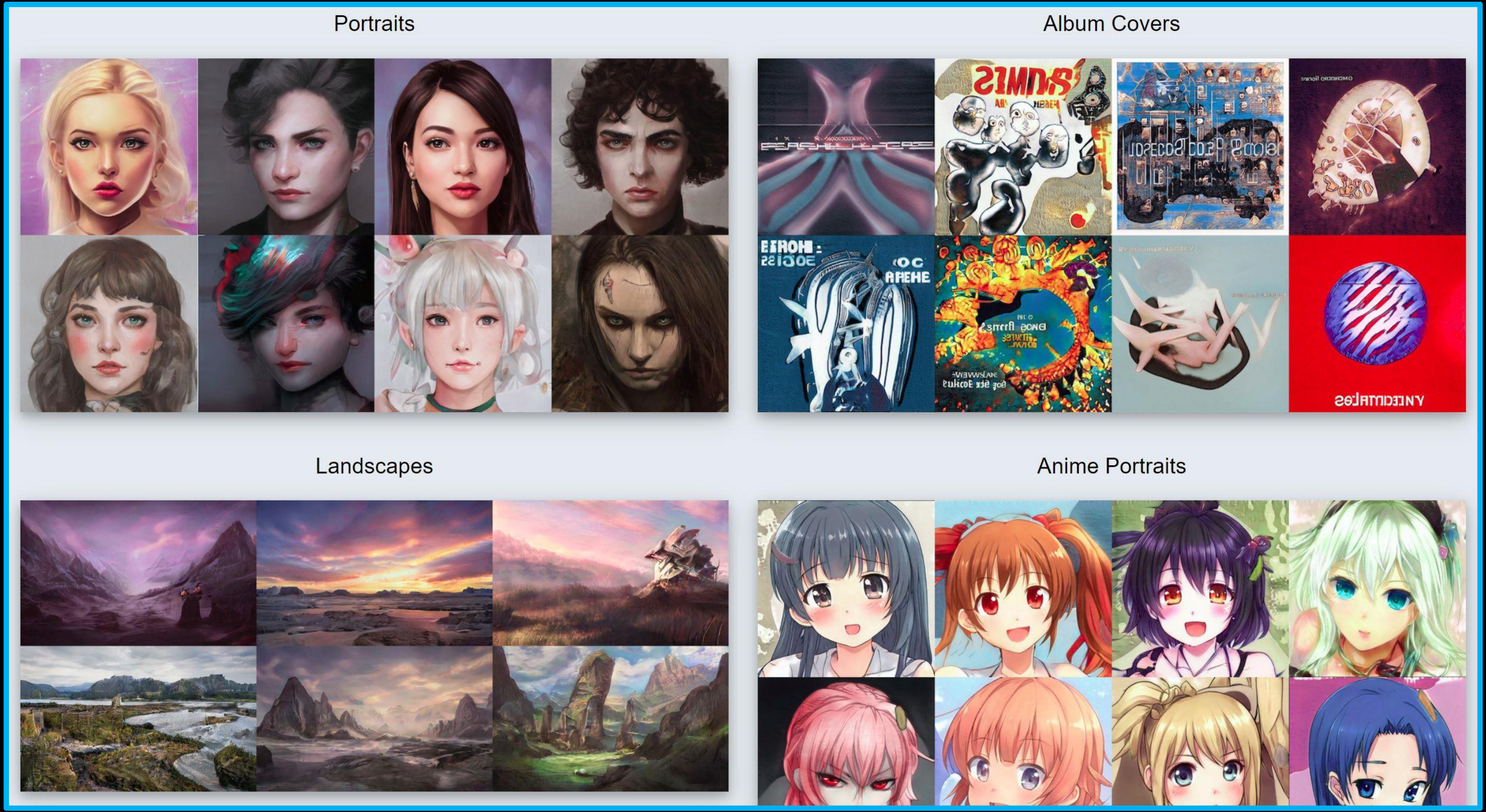
Watch: youtube.com/watch?v=A6bo_mIOto0

PS: Klingemann has tons of these, try: a, b, c

Weird uses of GANs I.

- Searching through latent space of existing models and finding *weird* vectors
- **Project ArtBreeder** (*GAN Breeder*)
  - Given the fact that BIG GAN contains a lot of imagery, we can almost endlessly search through it…
  - This project lets people on the internet look through the space, select their favorite samples and add mutations (inspired by **genetic algorithms**)
  - Mixing the latent vectors or adding small permutations …
  - Start with one sample, see where it evolves – for example: artbreeder.com/i?k=095f4e2843239046bbb45d1c

Project ArtBreeder – artbreeder.com/

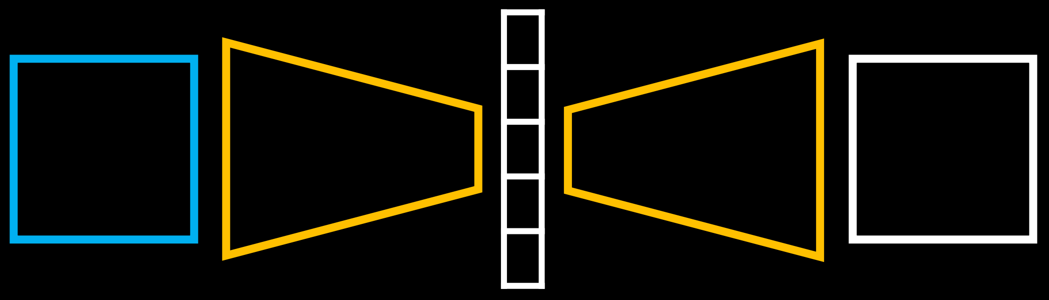


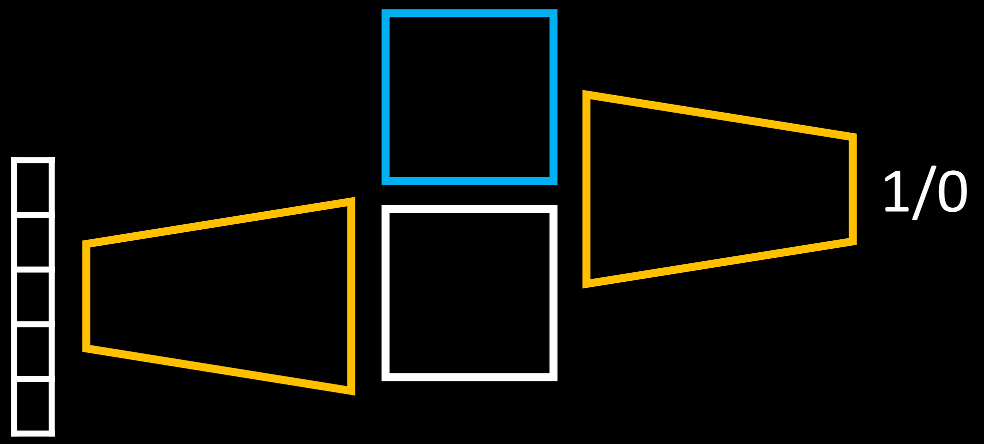
Weird uses of GANs II.

- Plugging back the capabilities of AutoEncoders into GANs:
  - **Encoding real images into** the **latent space** of the model
  - Therefore making use of the better quality of generated images

Figure 1: Top row: input images. Bottom row: results of embedding the images into the StyleGAN latent space.

Paper here





Pause 2

Big picture

There is a certain *neural aesthetic* when you train any generative model on your data. But is that an artistic act on its own?

- I see it rather as a **tool** one can use … paintbrush or camera of sorts (*photo–graphy -> neuro–graphy*)

We should **study the tool**, explore its capabilities. See where we can have some curatorial impact over it.

Curatorial control over generative models

Ways of interacting with generative models:

1. Choice of the **dataset**
2. Choice of the **architecture**
3. Interaction with the **latent space**
4. Editing the Neural Networks directly (*NN hacking / NN bending*)

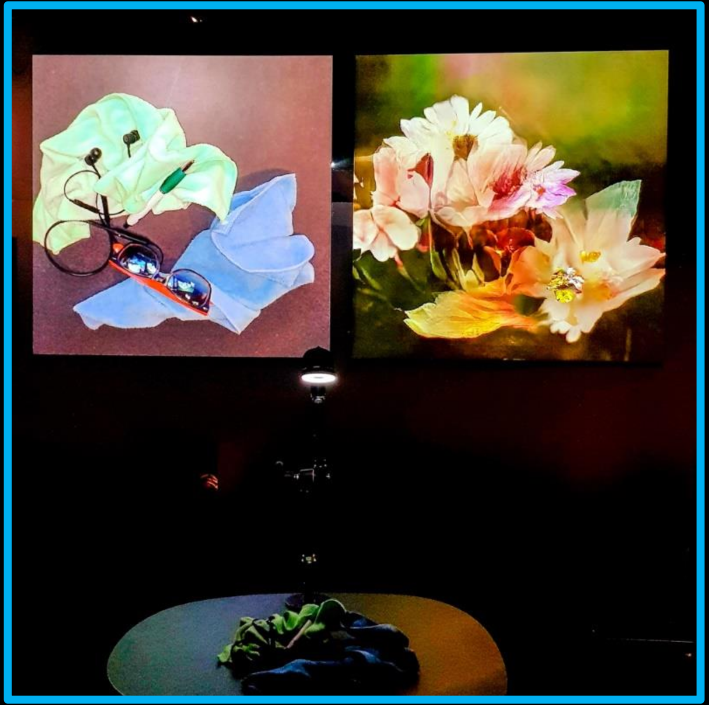
1. Choice of the dataset

Anna Ridler's work:

- Mosaic Virus (2018)
- Fall of the House of Usher (2017)

Memo Akten's work:

- Learning to See (2017)



2. Choice of the architecture

(Technical consideration…)

AE

-> **Blurry** results

-> **Encodes real images**

Error from the square distance

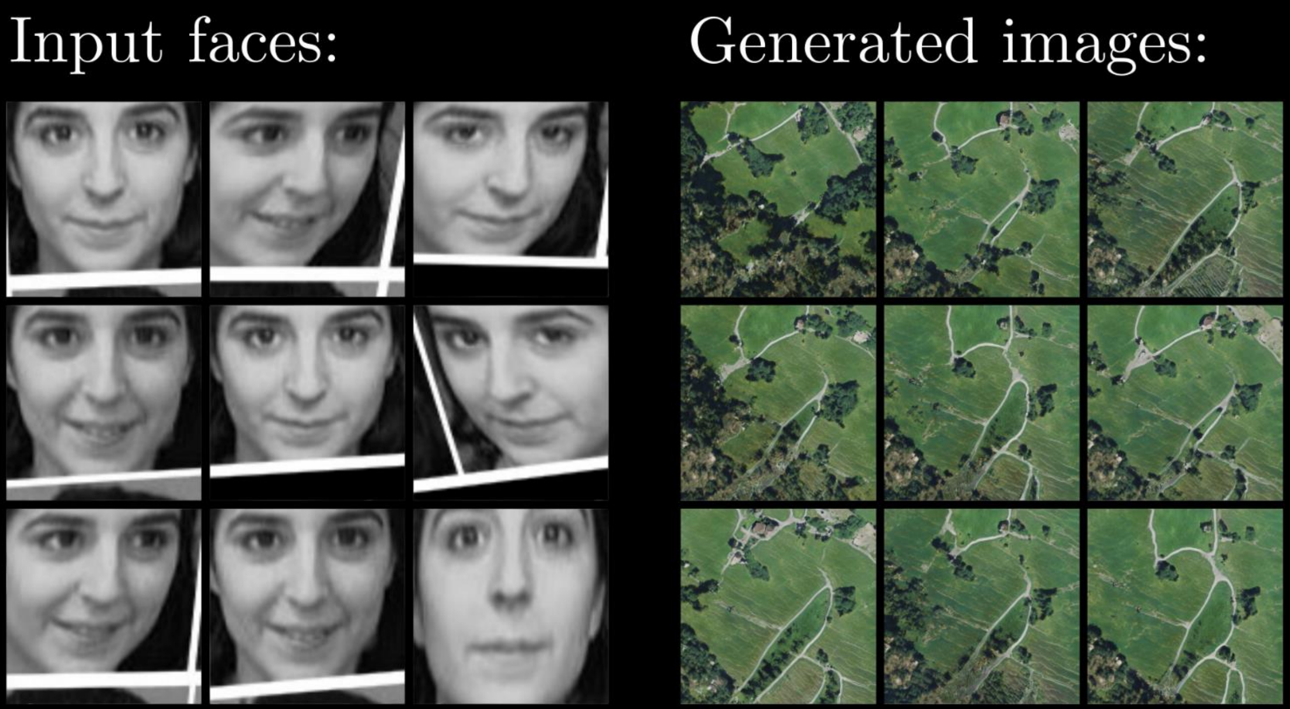
GAN

-> **Sharper**, more details

Error is propagated from G/D

*+ **Many versions of implementations** and used tricks …*

3. Latent Space Exploration

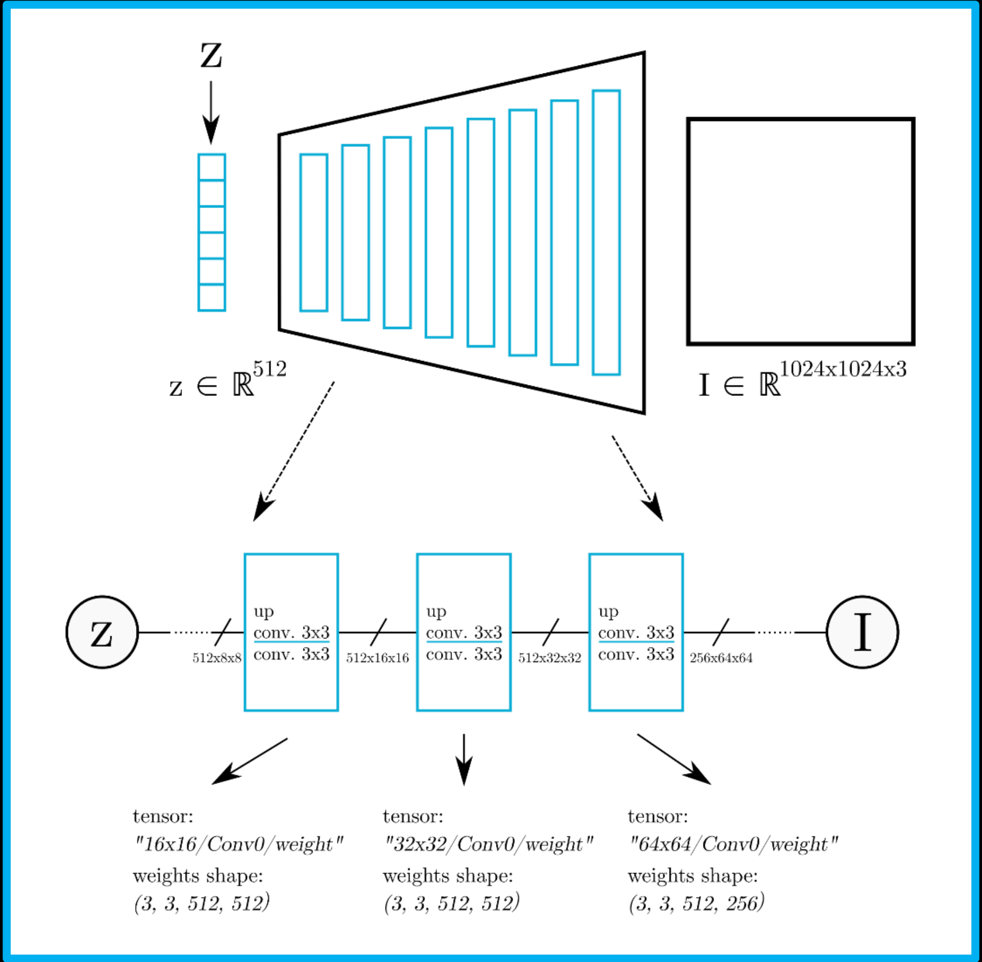


< Intuition: What is similar falls near each other

+ Everything you saw before (Mario K., …)

Video here:
youtube.com/watch?v=WncPWHE36S8

4. Neural Network editing ...

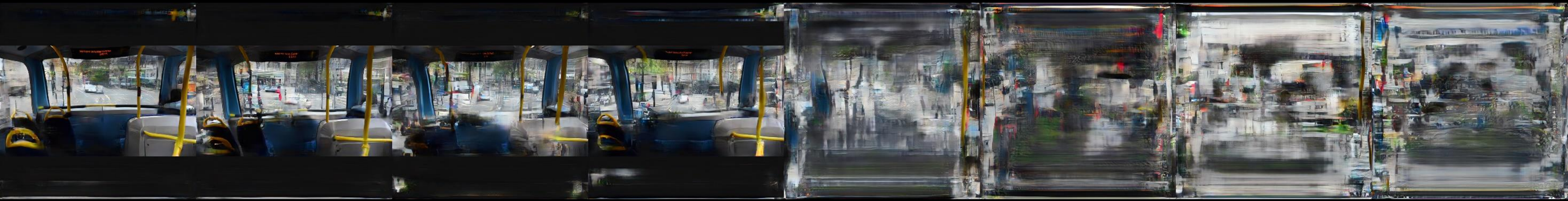


< Intuition: Targeting the Generative Model's building blocks

< Treatment of these blocks as if they could be reconnected, further edited, rewired ...
... (hacked!)

Exploring Machine Intelligence

Week 5, Generative Models



Scraping the Internet & Generative models

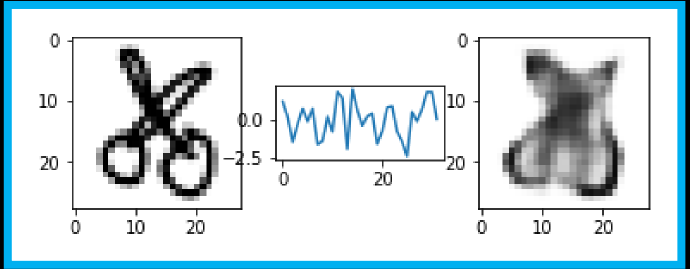
~ Vítek Růžička

Practicum: Generative Models

Scraping the Internet and Intro to Generative models (VAE)

Continue with code on our Github:

- github repo: [github.com/previtus/cci_exploring_machine_intelligence]
- Scraping the Internet: [ml05_scraping_internet.ipynb]
- VAE notebook: [ml05_convolutional_VAE.ipynb]



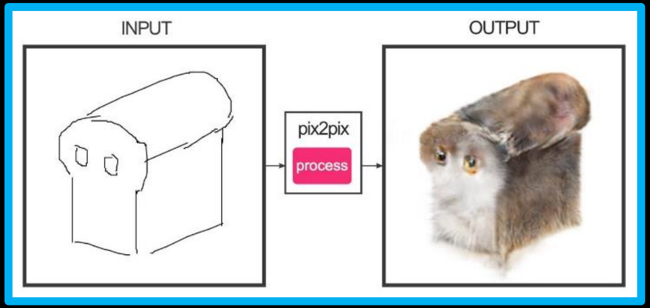
Links and additional readings:

- Details about types of VAEs: lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html
- VAE visualizations of latent space: hackernoon.com/latent-space-visualization-deep-learning-bits-2-bd09a46920df
- Very in-depth explanation of GANs on Anime (includes novel ideas, models, approaches): www.gwern.net/Faces

Next class

More generative models:

- **Pix2pix** (which we saw both the work by Anna Ridler, Memo Atken and others), **Style transfer**



- **Sequential modelling**

The end