Variational Auto Encoders

LCP-B final exam

15/06/2023

Group 9 - Bacilieri, Barbiero, Bordin, Pitteri

Chapters

Chapter 1

What is a VAE

Auto-encoders

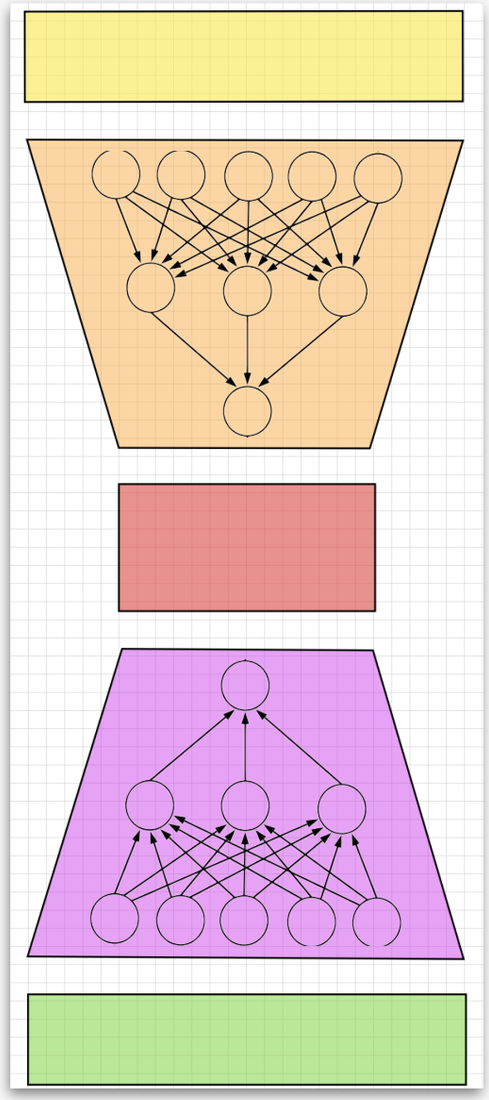
General structure

Generative neural network

Compress input features in a lower-dimensional *latent space*

Reconstruct an output similar to the input starting from the latent space

Mainly used for compression, de-noising and anomaly detection



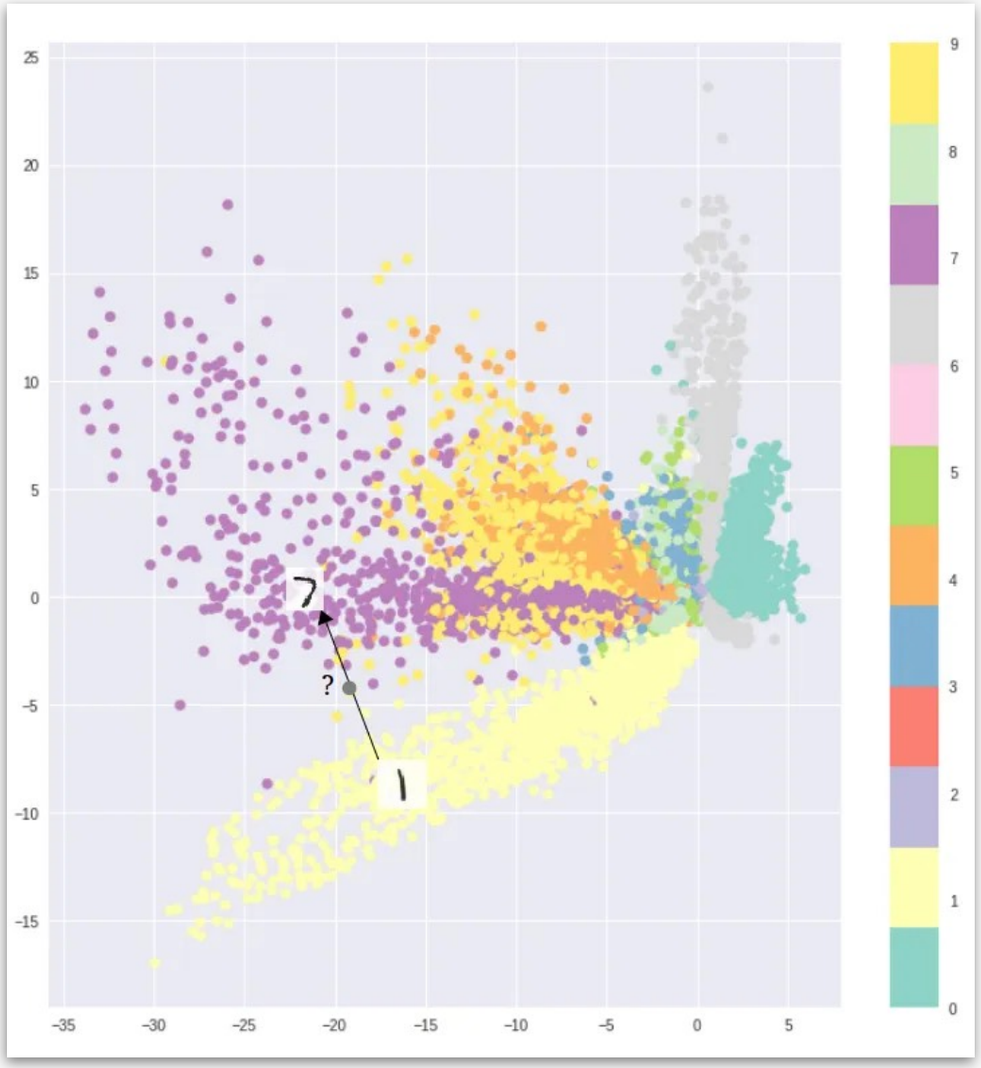
Auto-encoders

Latent space clustering

Encoder and decoder can be of any architecture depending on the problem at hand.

One notable feature to study is the latent space distribution of points, in this case numbers from the MNIST dataset (link to article)

Can you spot the issue?



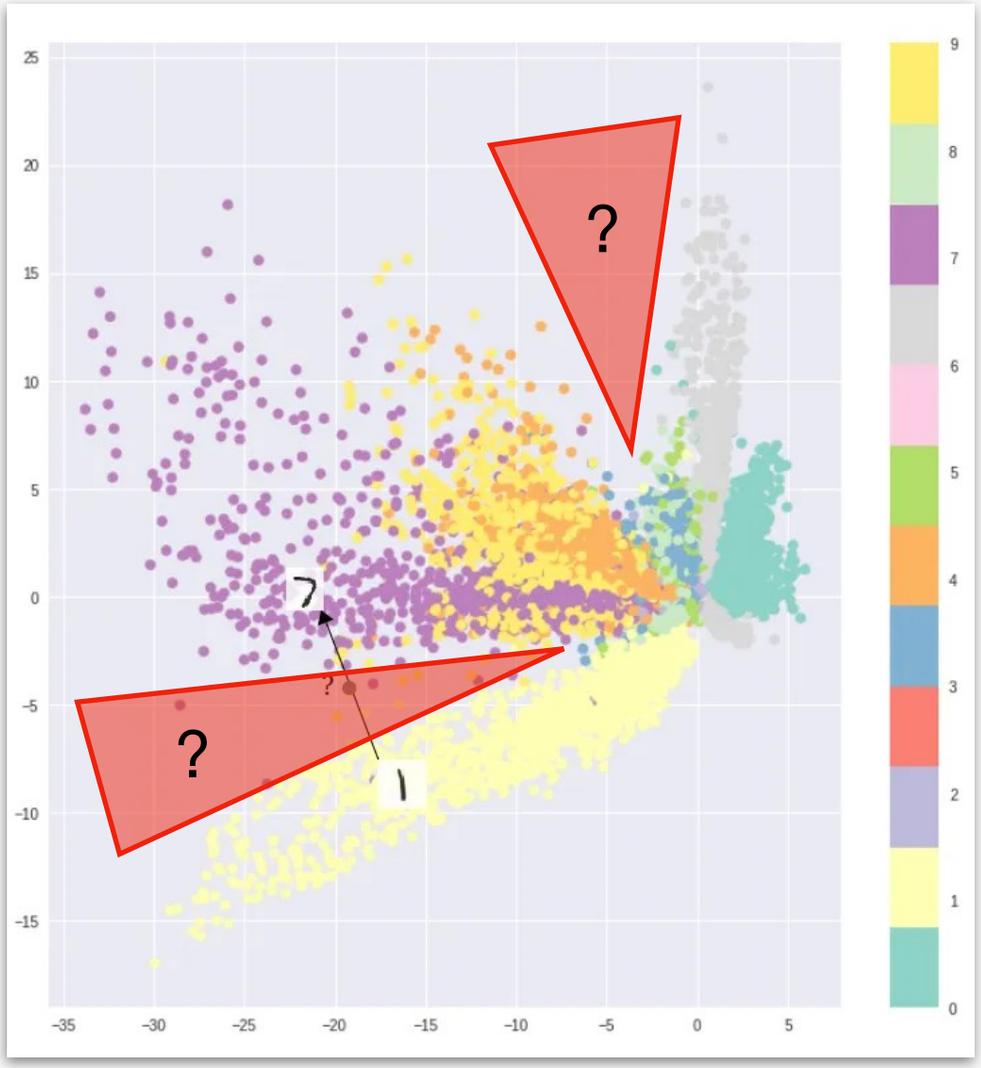
Auto-encoders

Latent space clustering

Encoder and decoder can be of any architecture depending on the problem at hand.

One notable feature to study is the latent space distribution of points, in this case numbers from the MNIST dataset (link to article)

Can you spot the issue?



Variational Auto-encoders

Sampling in latent space

An improvement: learning a posterior distribution in the latent space

$$p_\theta(\mathbf{z} \mid \mathbf{x}) = \frac{p_\theta(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})}{p_\theta(\mathbf{x})}$$

θ = model parameters
$\mathbf{z}$ = vector in latent space
$\mathbf{x}$ = vector in original space

The likelihood $p_\theta(\mathbf{x} \mid \mathbf{z})$ comes from a NN so it's impossible to evaluate it directly.

VAEs circumvent this issue with a *variational approach*

Variational Auto-encoders

Variational free energy

Starting from an approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$ we can define a variational free energy as

$$F_{q_\phi}(\mathbf{x}) = \underbrace{-\mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x} \mid \mathbf{z})]}_{\text{Prob. of data given latents}} + \underbrace{D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid p(\mathbf{z}))}_{\text{Regularizer term, encourages posterior ~ prior}}$$

and this acts as an upper-bound on the negative log-likelihood of the data.

This will serve as the objective function to minimize during training

Variational Auto-encoders

Sampling in latent space

Many choices are possible for $q_\phi(\mathbf{z} \mid \mathbf{x})$, in our case Gaussian

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \sim \mathcal{N}(\mu_\phi(\mathbf{x}), \mathrm{diag}(\sigma_\phi{}^2(\mathbf{x})))$$

and Gaussian latent priors $p(\mathbf{z})$

With this choice

$$D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid p(\mathbf{z})) = -\frac{1}{2} \sum_i (1 + \log \sigma^2_{\phi,i}(\mathbf{x}) - \mu^2_{\phi,i}(\mathbf{x}) - \sigma^2_{\phi,i}(\mathbf{x}))$$

Variational Auto-encoders

Sampling and stuff

For our application we also (implicitly) considered a Gaussian likelihood

$$p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \mathcal{N}(\mu_\theta(\mathbf{z}), \mathbb{I})$$

so that the log likelihood term of $F_{q_\phi}(\mathbf{x})$ can be written as

$$\log p_\theta(\mathbf{x} \mid \mathbf{z}) \sim \log \exp[-(\mathbf{x} - \mu_\theta(\mathbf{z}))^2] = -(\mathbf{x} - \mu_\theta(\mathbf{z}))^2$$

which can be seen as the MSE between the original and the reconstructed data

Variational Auto-encoders

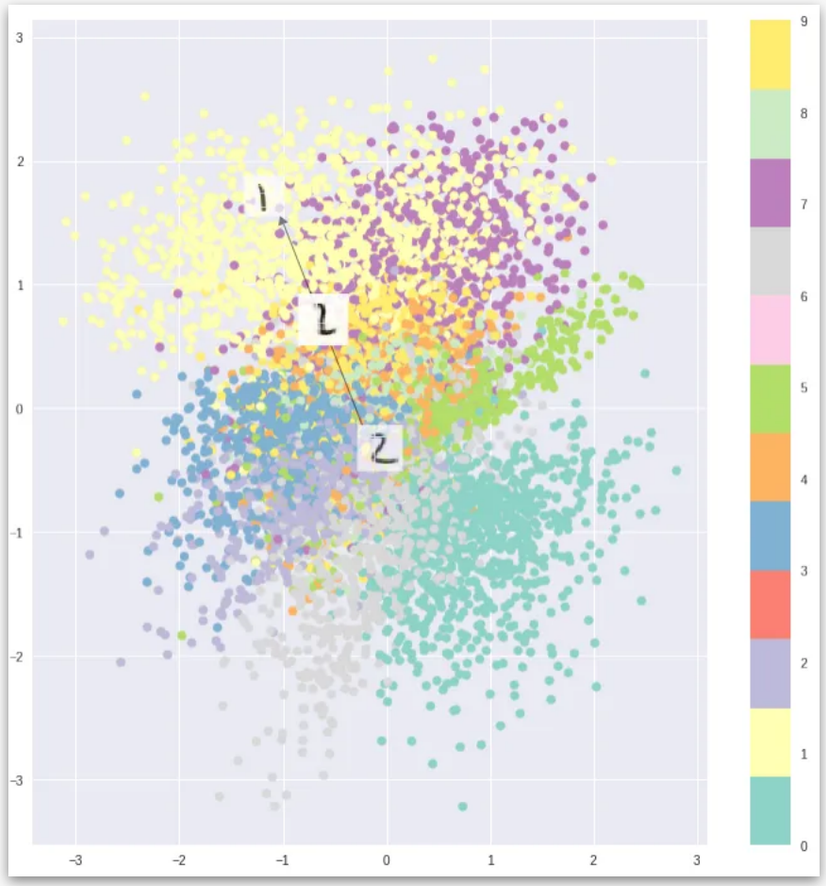
Wrapping up

To conclude, the function we minimize during training is

$$\mathrm{MSE}(\mathbf{x}, \mu_\theta(\mathbf{z})) + \lambda D_{\mathrm{KL}}(q_\phi(\mathbf{z} \mid \mathbf{x}) \mid p(\mathbf{z}))$$

λ is a regularization parameter

All the previous manipulations allow us to end up with a much more “coherent” latent space



Chapter 2

Data Generation

Data Generation

Physical problem

Fluid in a 2D square container of side L, subject to

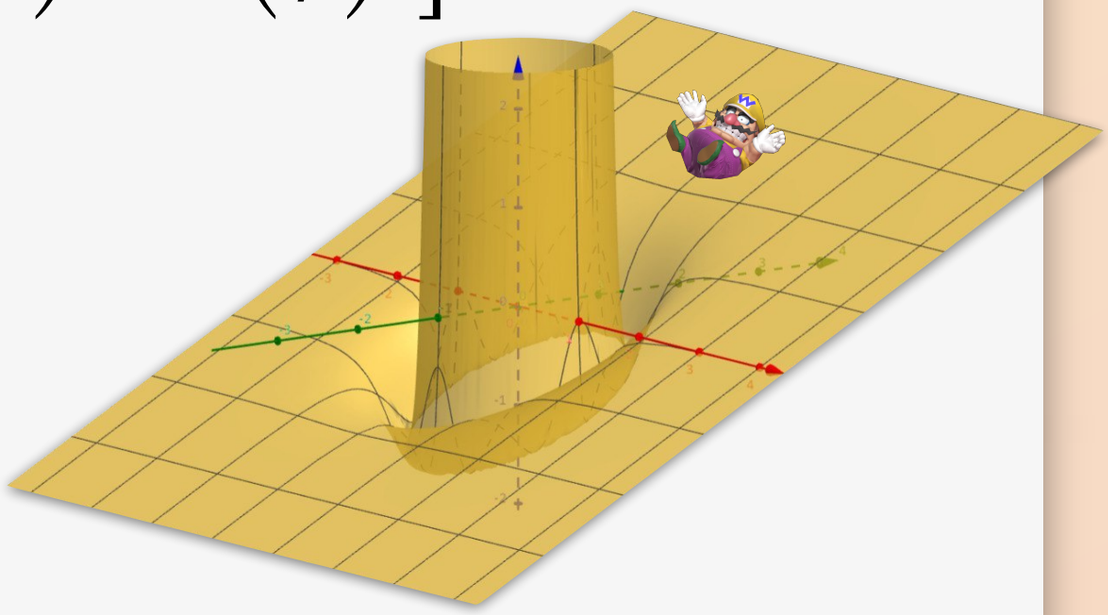
- gravitational potential $U_g = mgy$

- Lennard Jones Potential $U_{LJ} = 4\epsilon\left[\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right]$

Moving to reduced units $r' = r/\sigma$

$$U' = 4\left[\left(\frac{1}{r'}\right)^{12} - \left(\frac{1}{r'}\right)^{6}\right] + \frac{mg\sigma}{\epsilon}y'$$

The only trainable parameter is $\gamma = \frac{mg\sigma}{\epsilon}$



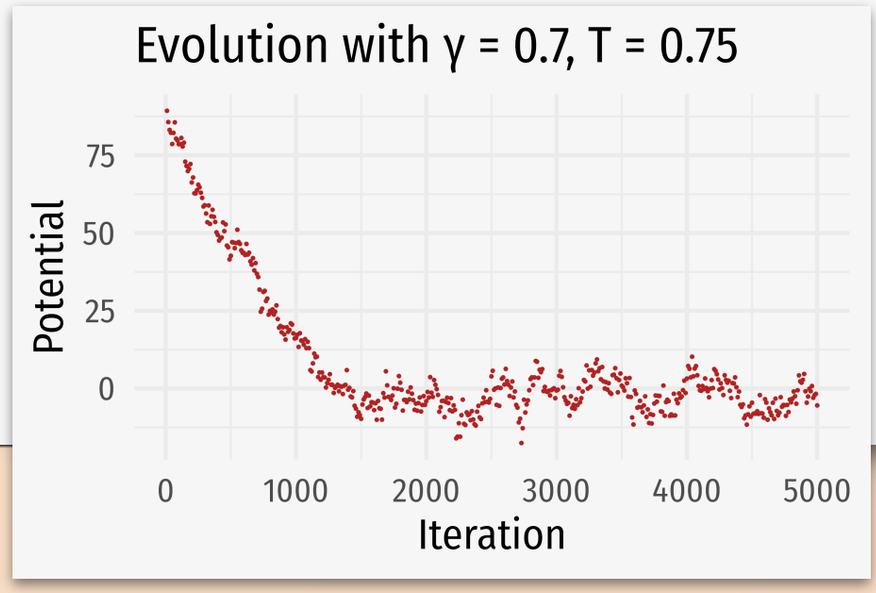
Data Generation

Metropolis Algorithm

Data is generated in C++ with the Metropolis Algorithm noting that

1 - Proposal moves on all data points iteratively rather than randomly

2 - Simulation is reset to an initial random configuration after every sampling

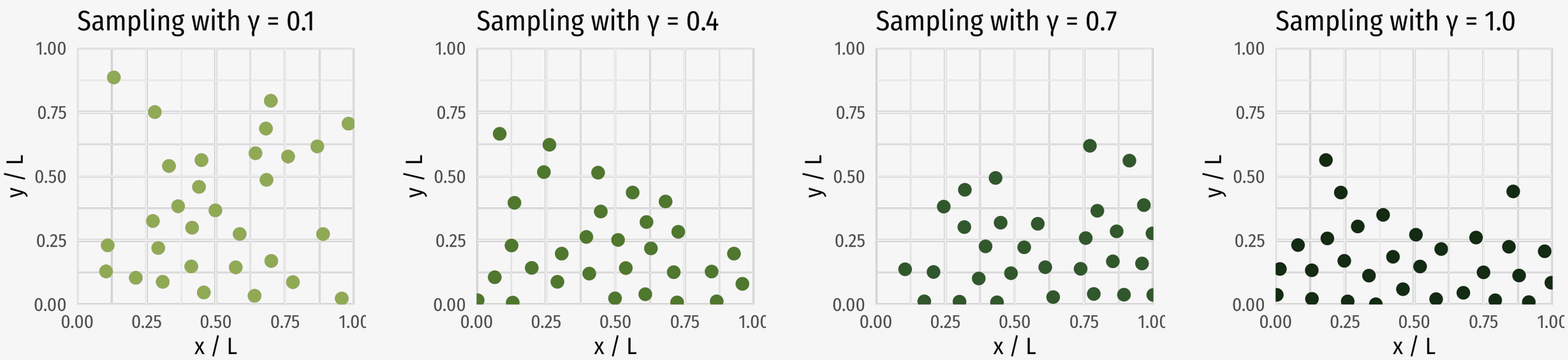


Data Generation

The dataset

To train and test the VAE we generated four datasets with $\gamma = \{0.1, 0.4, 0.7, 1.0\}$ and 10000 configurations each.

Note that for a real fluid (e.g. Argon) $\gamma \sim 10^{-13}$



Chapter 3

Building the VAE

Building the VAE

Core ideas

The two main factors to take into account while building the network were

1 - Data representation and manipulation

2 - Encoder and decoder architecture

Next the optimization would concern

3 - Latent dimension size

4 - Regularization lambda for the MSE/KL loss tradeoff

Building the VAE

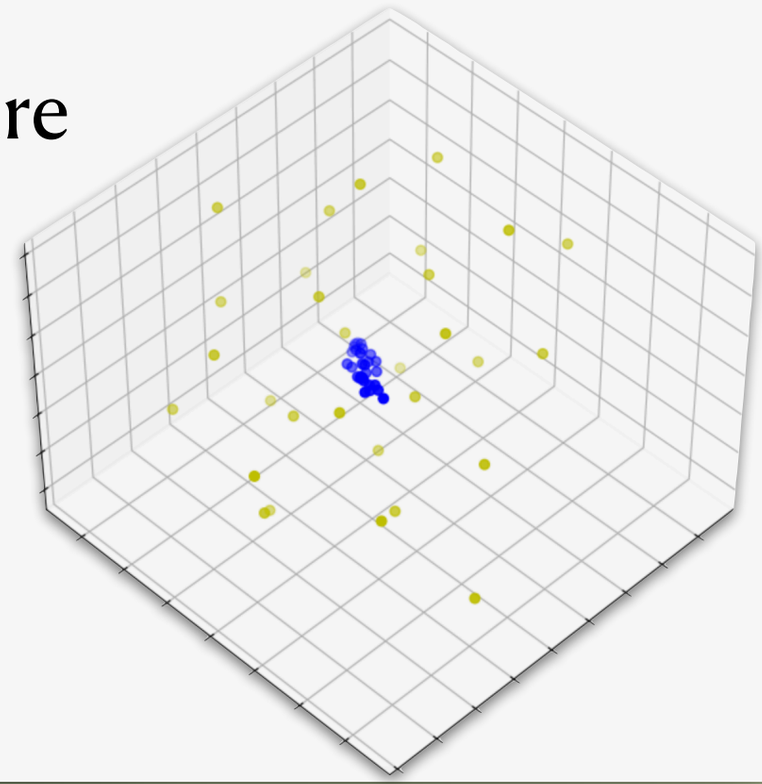
First unsuccessful attempts

The most simple idea and, naturally, the first one we tested was to use

1 - Coordinates representation for data points

2 - Feed forward fully connected architecture

Reconstructed data (blue) failed to capture any meaningful feature of the system no matter the sorting of points or the layers in the network

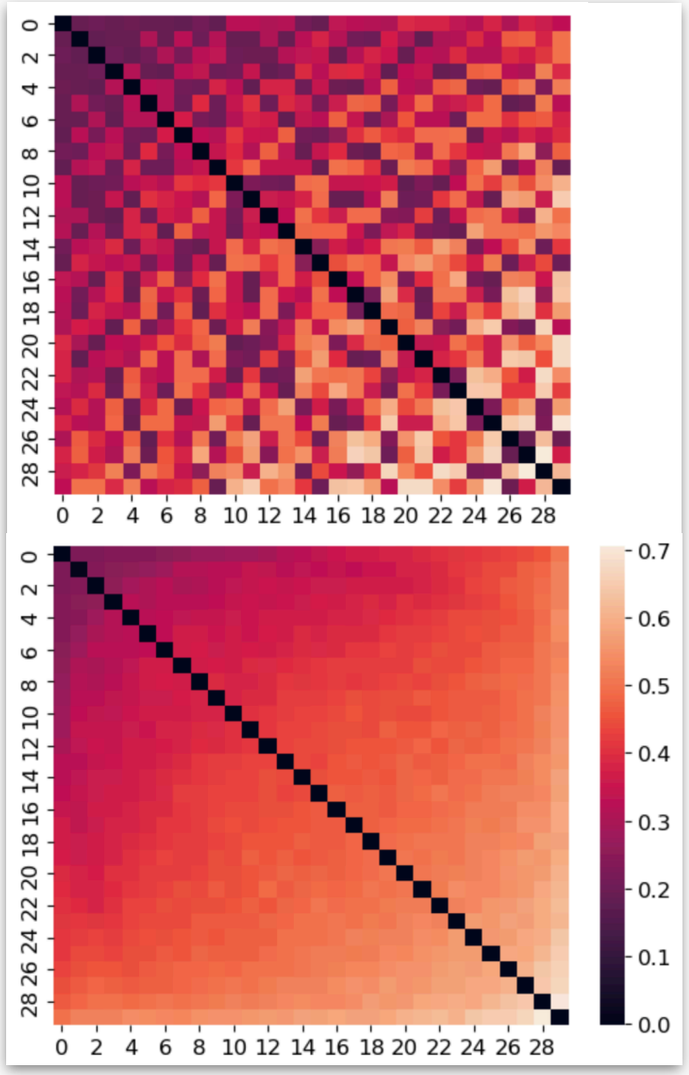


Building the VAE

First **unsuccessful** attempts

The next idea was

- FF architecture
- *distance matrix representation*
- sorted points



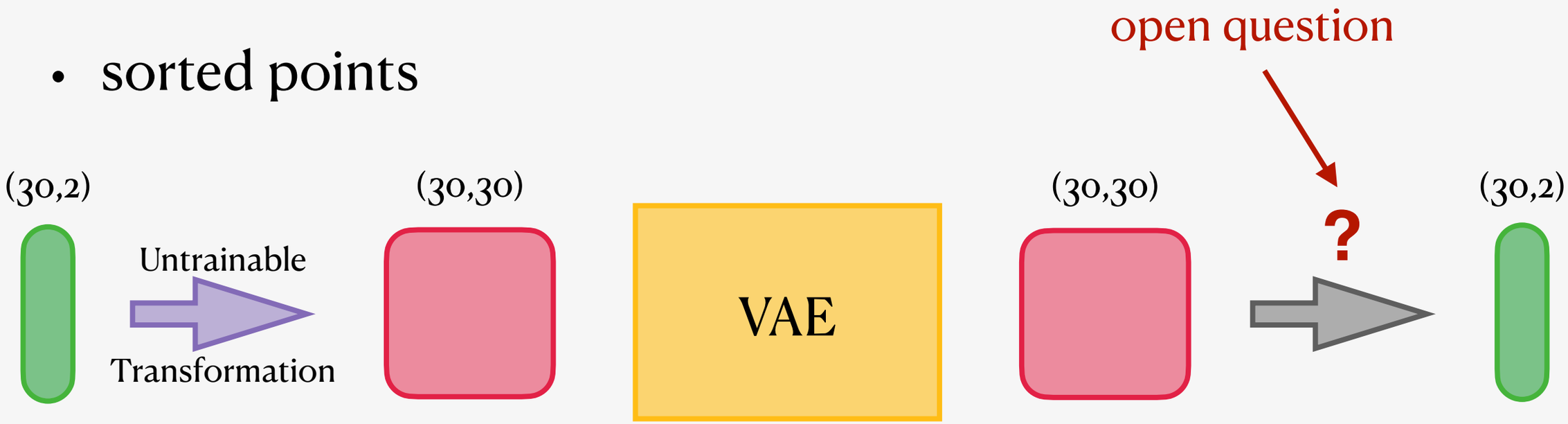
It didn’t learn anything...

Building the VAE

Going convolutional

Finally

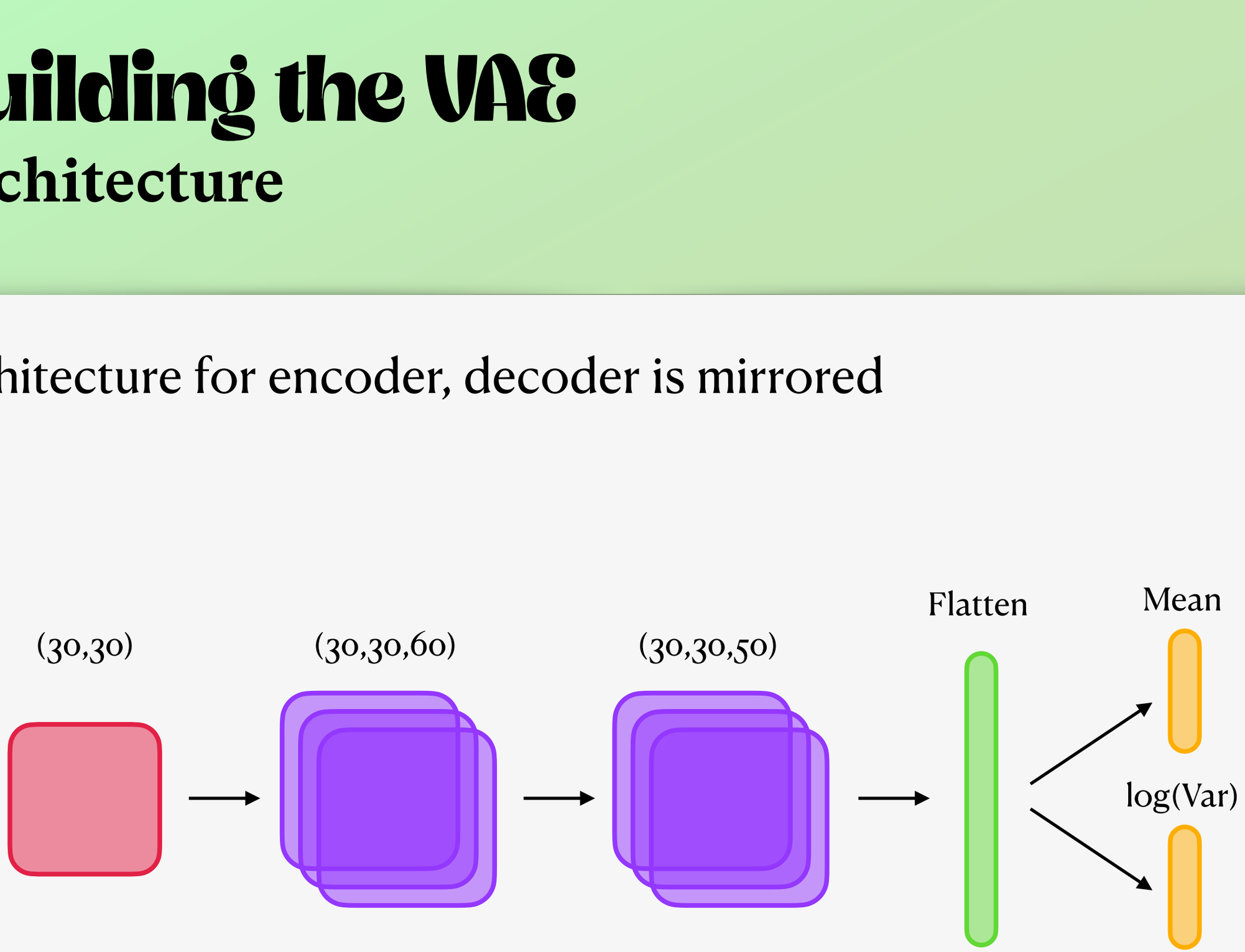
- *convolutional 2D*
- distance matrices
- sorted points



Building the VAE

Architecture

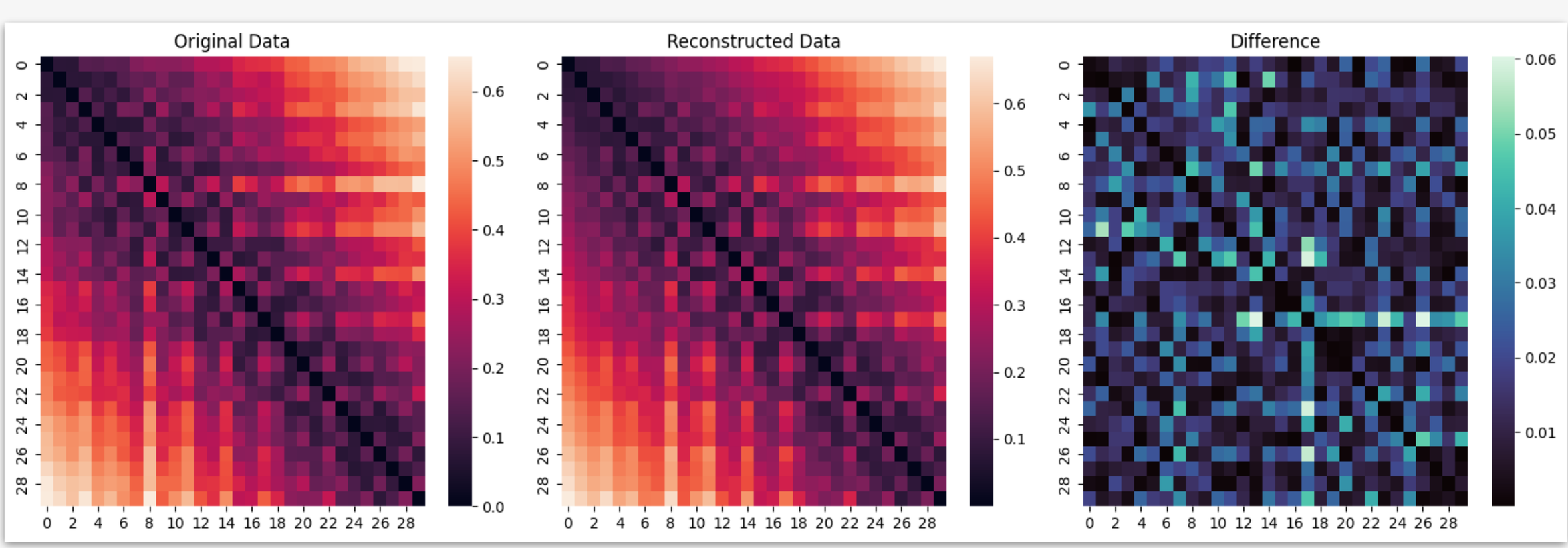
Architecture for encoder, decoder is mirrored



Building the VAE

Going convolutional

First promising results

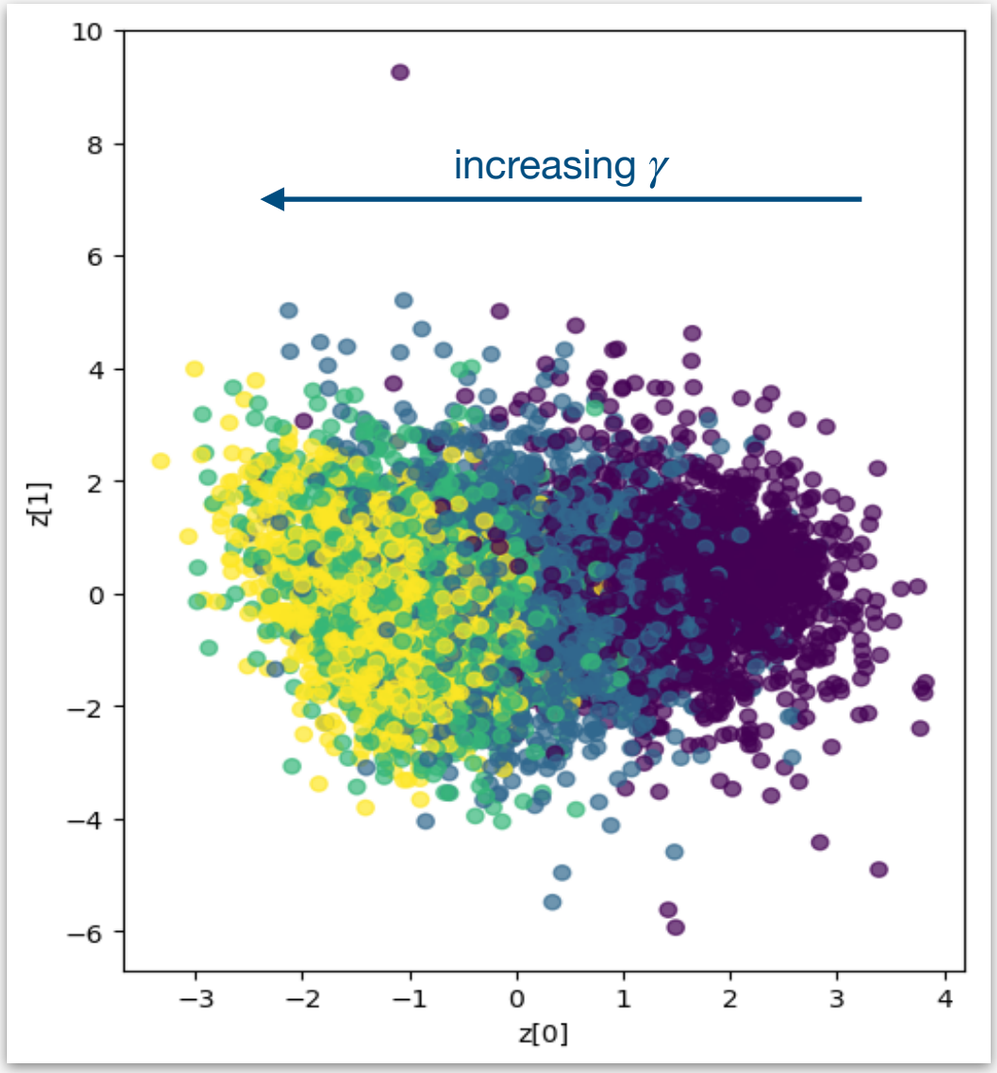


Building the VAE

Latent space clustering

Latent space clustering is in line with expectations for a VAE

However the first two PCA components accounted for only a small percentage of the total variance (8% – 15%)

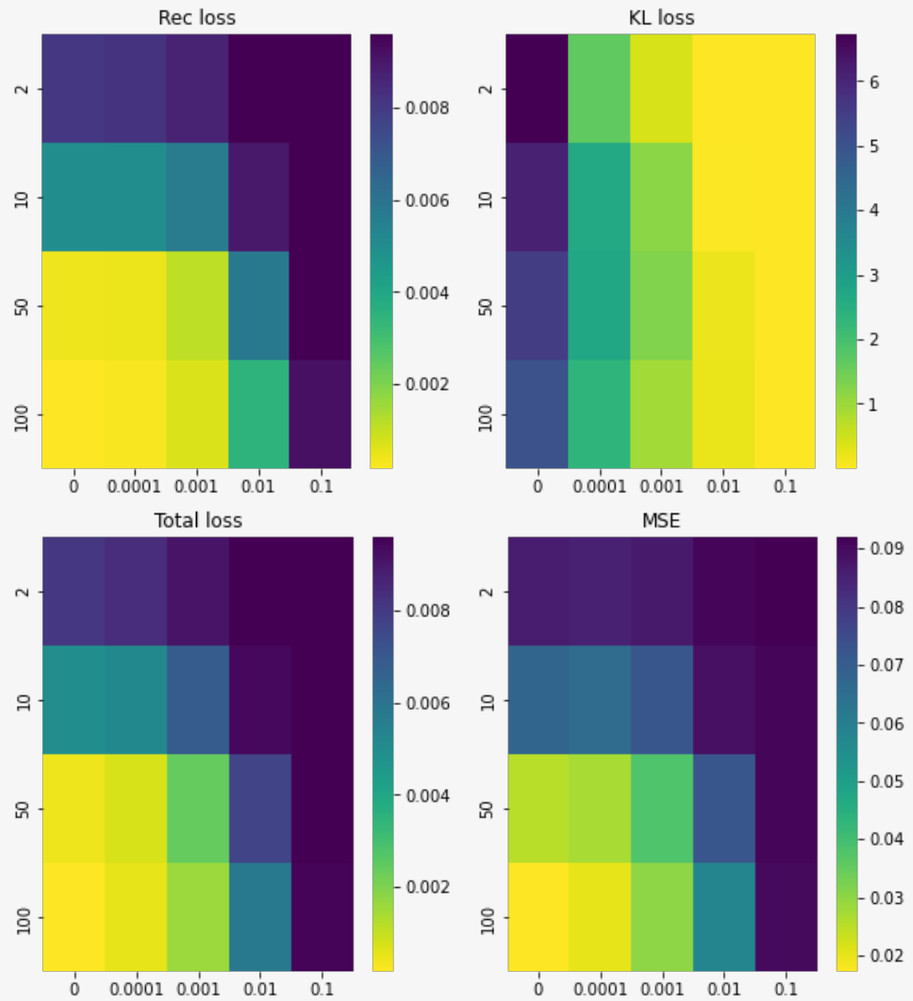


Building the VAE

Parameters optimization

- Lower MSE with higher lat dim
- Lower MSE with lower λ

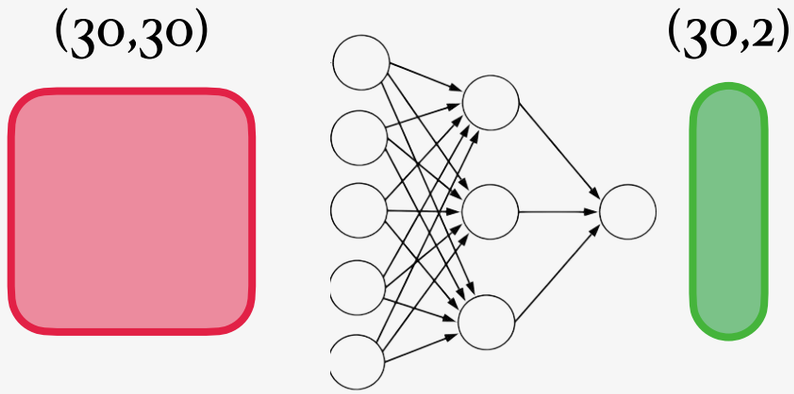
Potential under-fitting, exploring more complex convolutional layers for both encoder and decoder did not provide a meaningful difference so possibly a wholly different type of architecture would be needed.

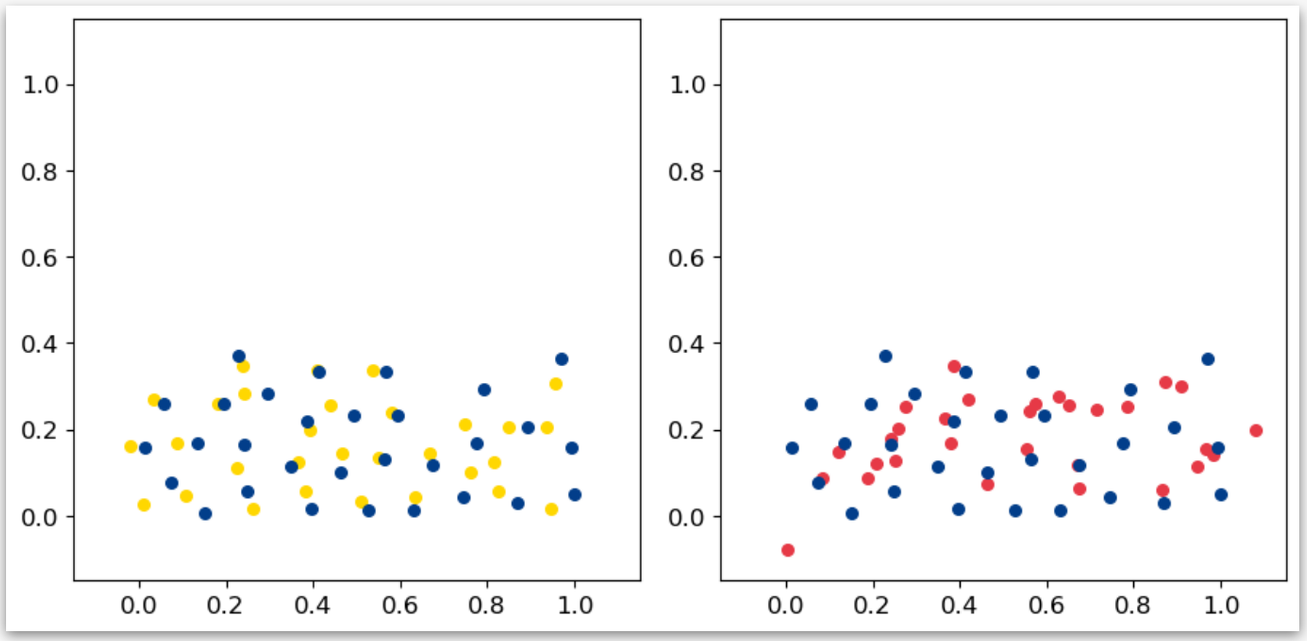


Building the VAE

Coordinates reconstruction

Tried learning based approaches with different architectures and levels of complexity but...

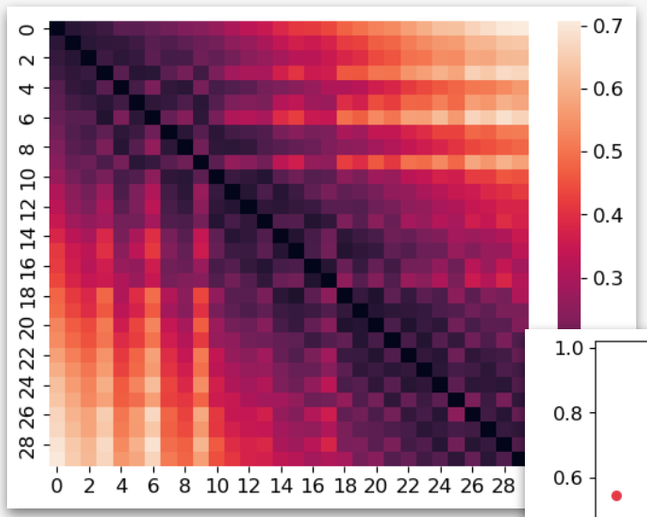




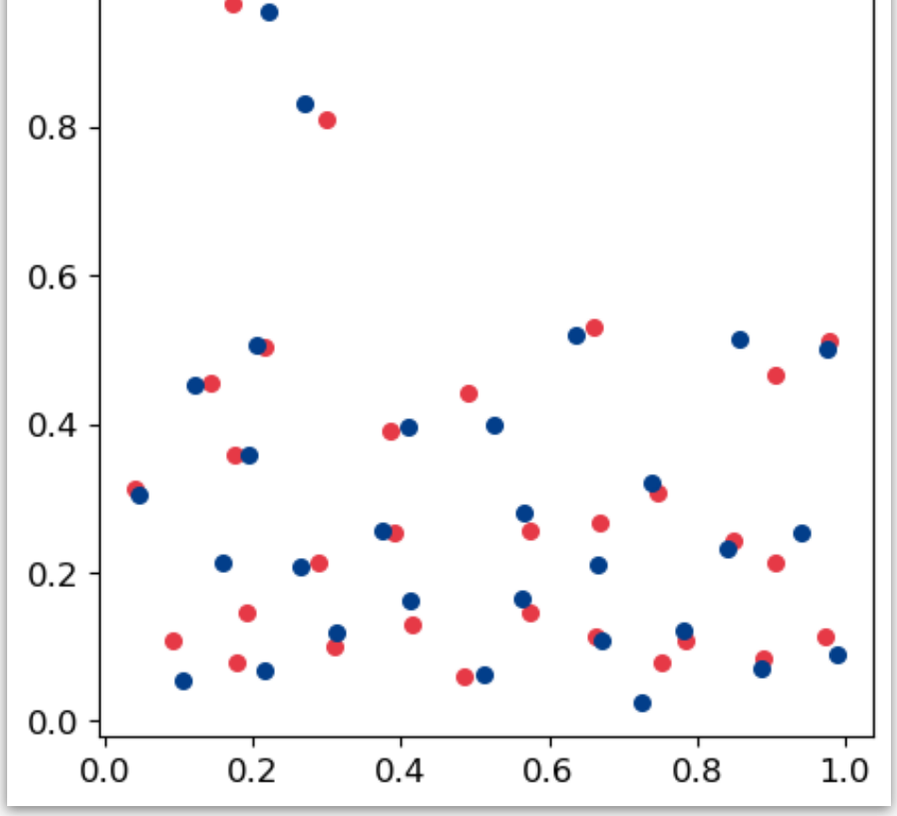
Building the VAE

Coordinates reconstruction

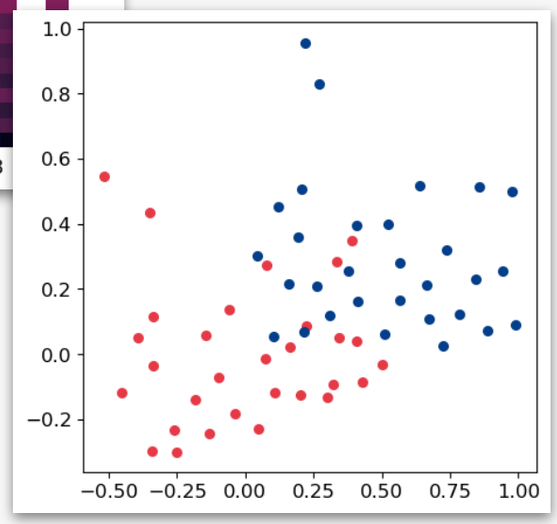
Fortunately, we found some linear algebra trickery (courtesy of Math Stack Exchange) to reconstruct deterministically



Trasl. + Rot.



Reconstruct



Building the VAE

Gram matrices

For a distance matrix $$d_{ij}^2 = (\mathbf{x}_i - \mathbf{x}_j) \cdot (\mathbf{x}_i - \mathbf{x}_j) = \mathbf{x}_i^2 + \mathbf{x}_j^2 - 2\mathbf{x}_i \cdot \mathbf{x}_j$$

Considering the first particle to be in the origin $$\mathbf{x}_i^2 = d_{i1}^2$$

So, effectively $$\mathbf{x}_i \cdot \mathbf{x}_j = \frac{d_{i1}^2 + d_{j1}^2 - d_{ij}^2}{2}$$

We define the matrix G as $$g_{ij} = \mathbf{x}_i \cdot \mathbf{x}_j = \sum_{a=1}^{m} x_i^{(a)} x_j^{(a)} = \tilde{X}\tilde{X}^\intercal$$

If we decompose G then $$G = USU^\intercal = U\sqrt{S}\sqrt{S}U^\intercal = XX^\intercal$$

Note —> $\tilde{X} = X$ up to an orthogonal transformation

Building the VAE

Reconstruction, pros and cons

Pros:

- able to reconstruct without complicating the model
- most important feature learned by VAE is preserved (distance between points)

Big huge catastrophic con:

- direct reconstruction from the latent space is not possible anymore

turbo tumbler

turbo tumbler

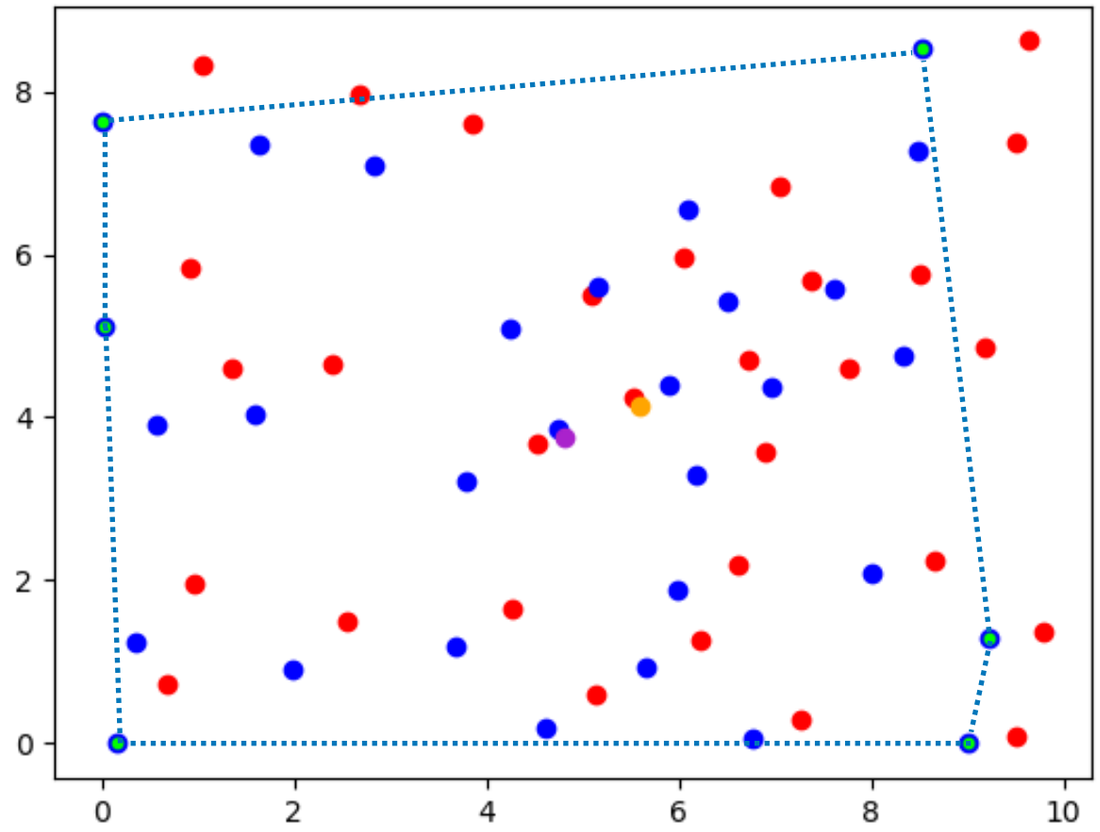
Losing a bit of generality we implemented an algorithm to minimize gravitational potential energy for reconstructed points.

The core concepts were as follows

- Find the smallest convex polygon enclosing all the points
- Look for the side that is closest to the center of mass
- Rotate the polygon to “let it sit” on that side

turbo tumbler

More on this later on...

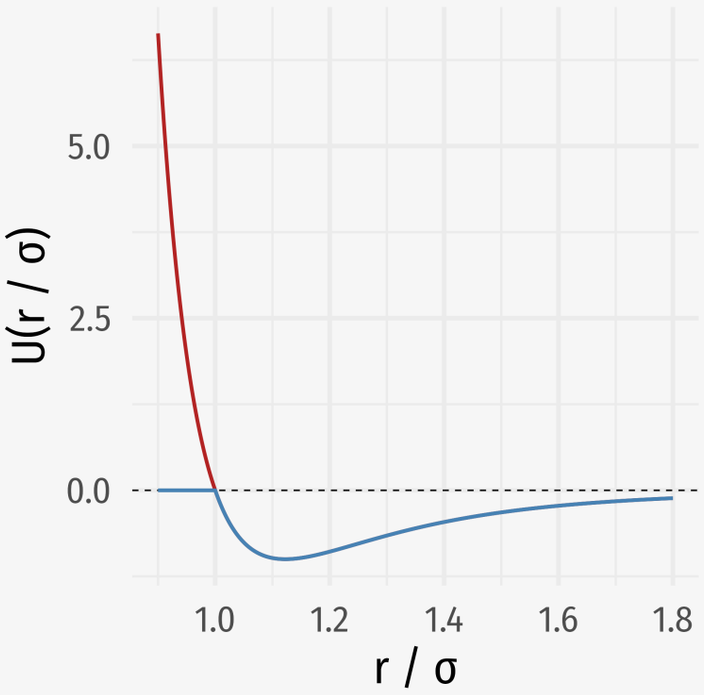


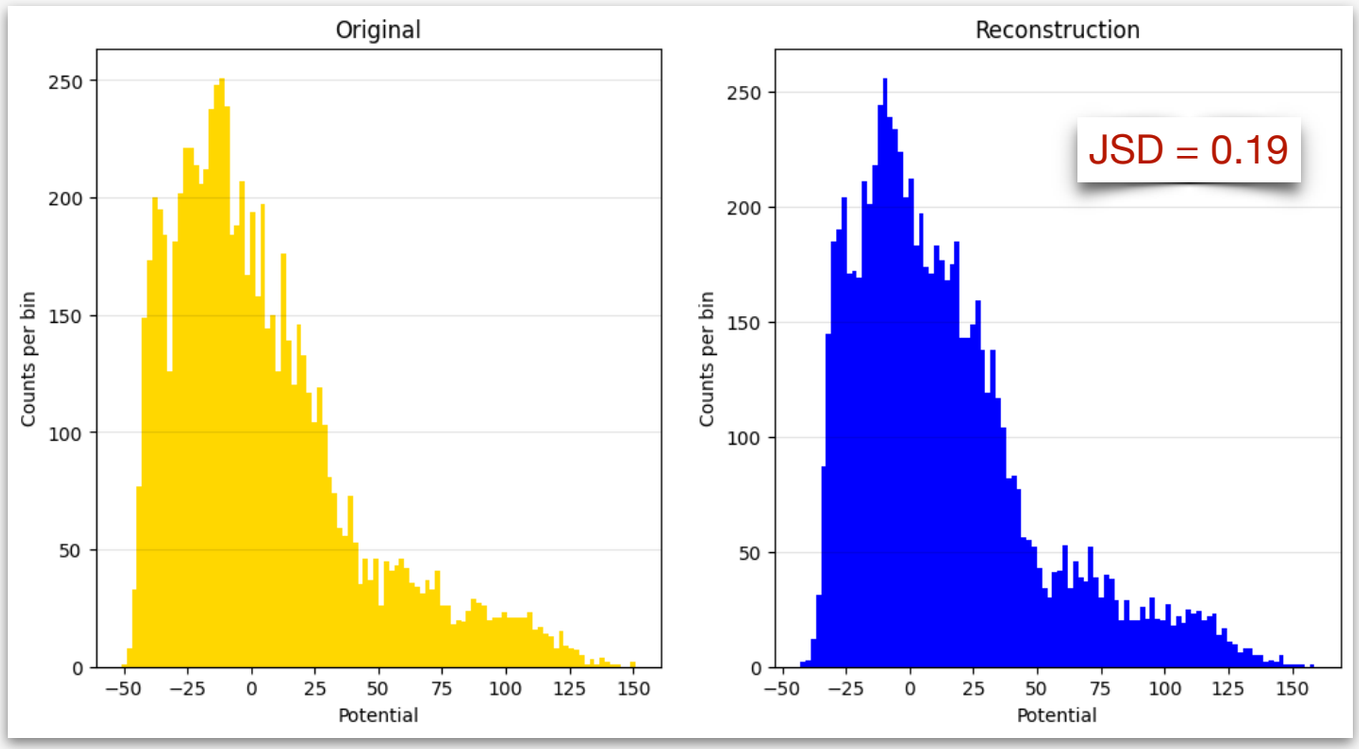
Chapter 4

Testing the VAE

Energy distribution

LJ potential proved to be quite unforgiving (*very* easily diverging) so we cut it at $r/\sigma = 1$ just to analyze the energy distribution





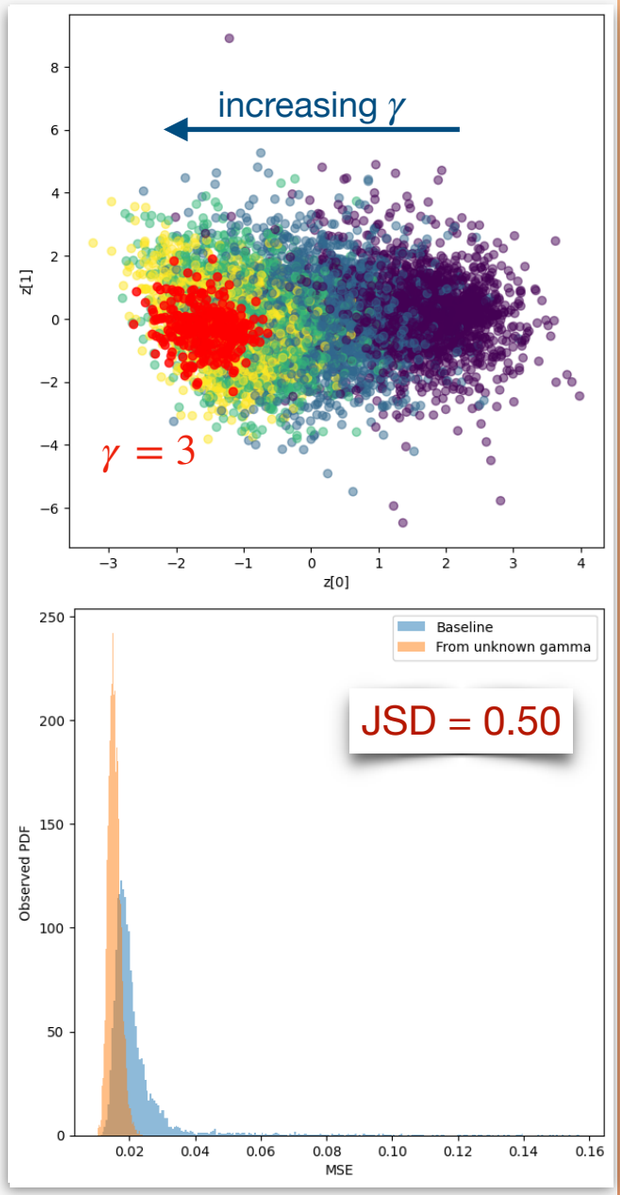
Testing with a different gamma

Anomaly detection

We fed the VAE data generated with a γ never seen during training.

Latent space mapping is satisfactory (particles are squashed to the bottom similarly to $\gamma = 1$)

MSE distribution is surprisingly good



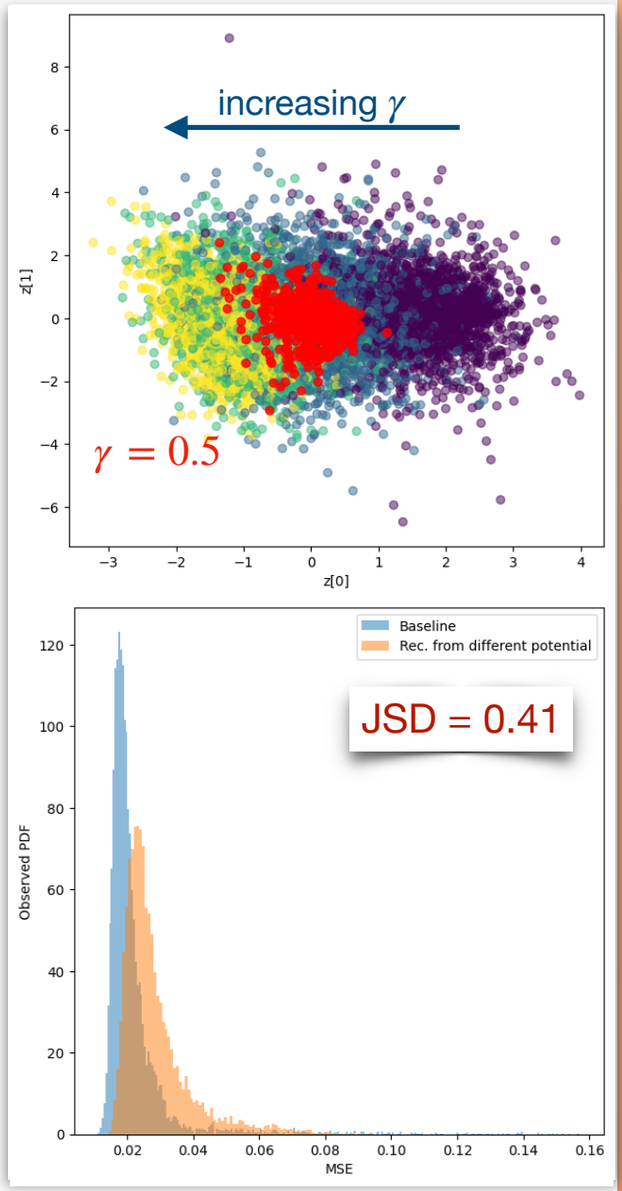
Different potential

Anomaly detection

We fed the VAE data generated with a “fake” Lennard Jones potential

$$U_i' = 4 \sum_{j \neq i} \left(\frac{1}{r_{ij}^4} - \frac{1}{r_{ij}^2} \right) + \gamma \, y_i$$

The performance still holds up surprisingly well both for gamma clustering and MSE distribution



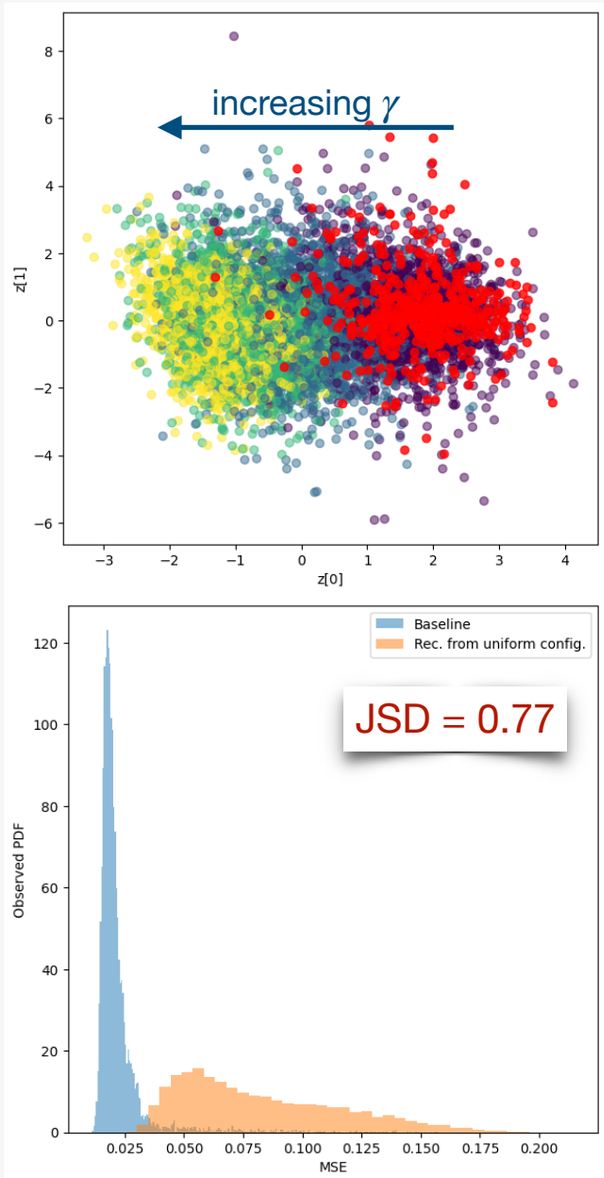
Random data

Anomaly detection

We fed the VAE data generated randomly from a uniform distribution inside the box

Latent space behavior is reasonable in the fact that it tends to cluster in the ... but with noticeably larger spread

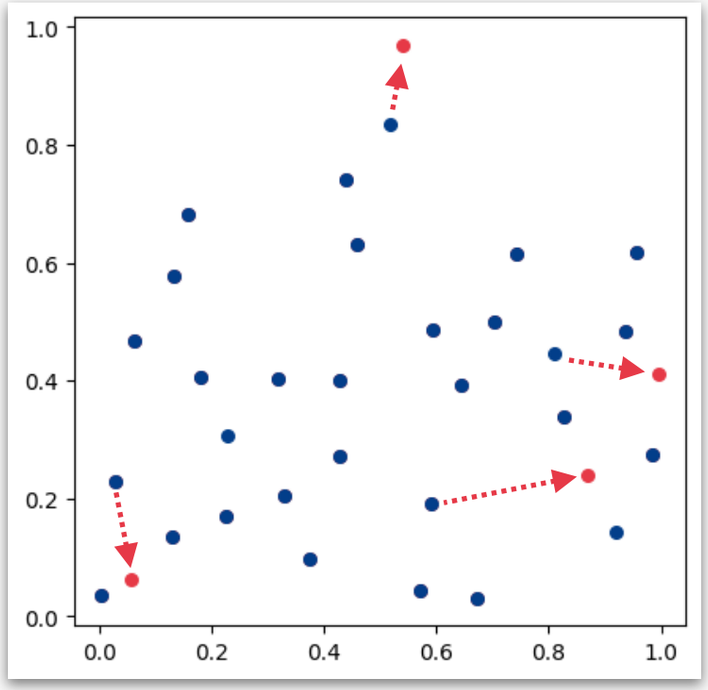
MSE distribution is all over the place



Corrupting data

Anomaly detection

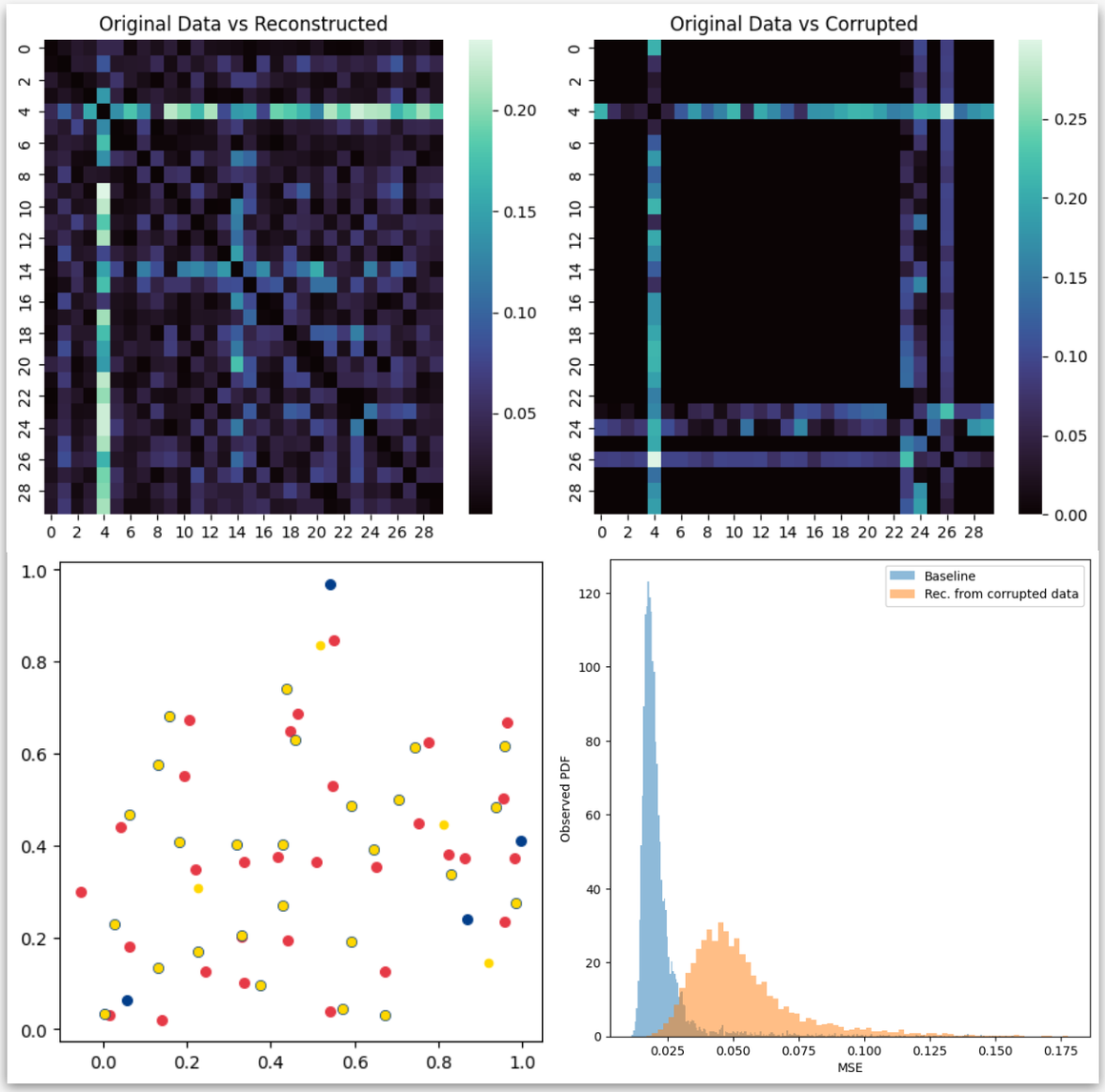
For each configuration of the test set four randomly chosen points were moved in a randomly chosen direction by a stochastic step length between 0 and 0.2.



Corrupting data

Anomaly detection

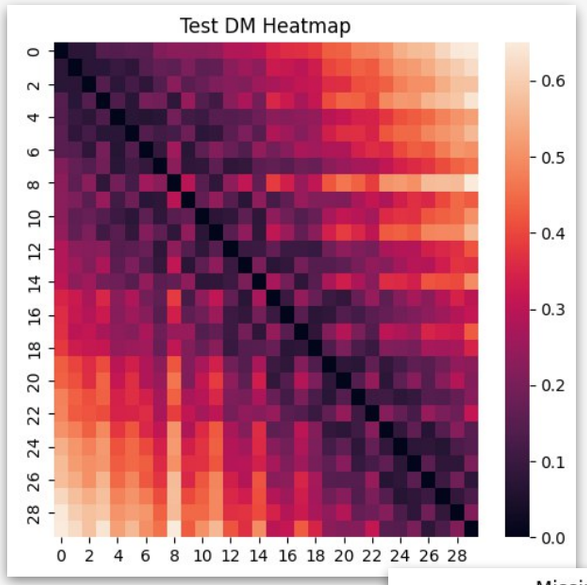
The network tries to regularize the four “faulty” points by shifting the whole distance matrix, however MSE distribution clearly reflects the anomaly of such configurations

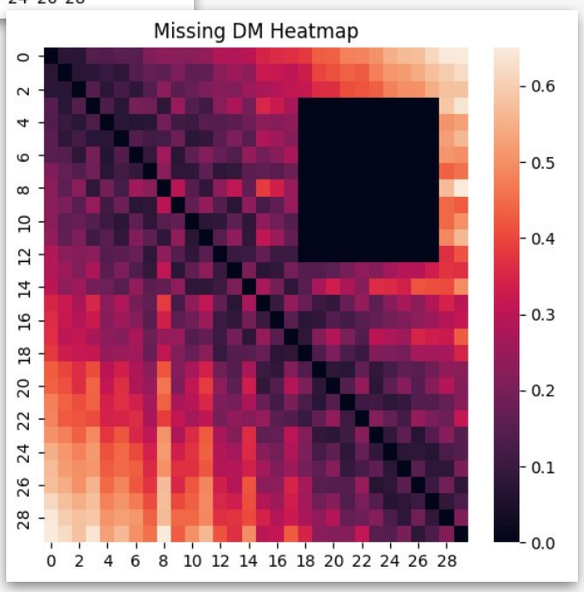


Missing data

Anomaly detection

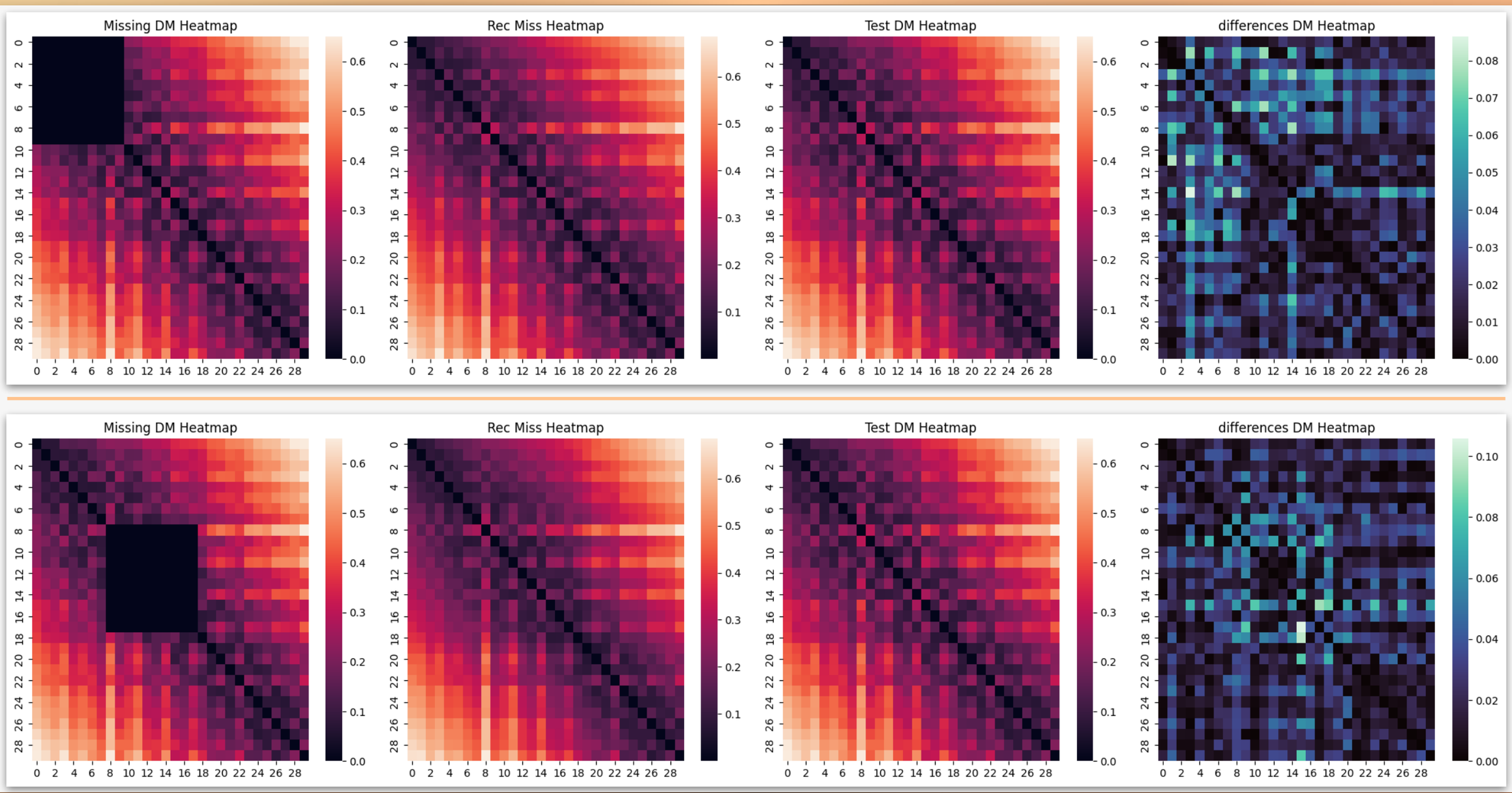
Just as a proof of concept, and to study what happens in the event of data loss at a later stage we “blacked out” regions of the distance matrices and tried to exploit the convolutional nature of the network for reconstruction





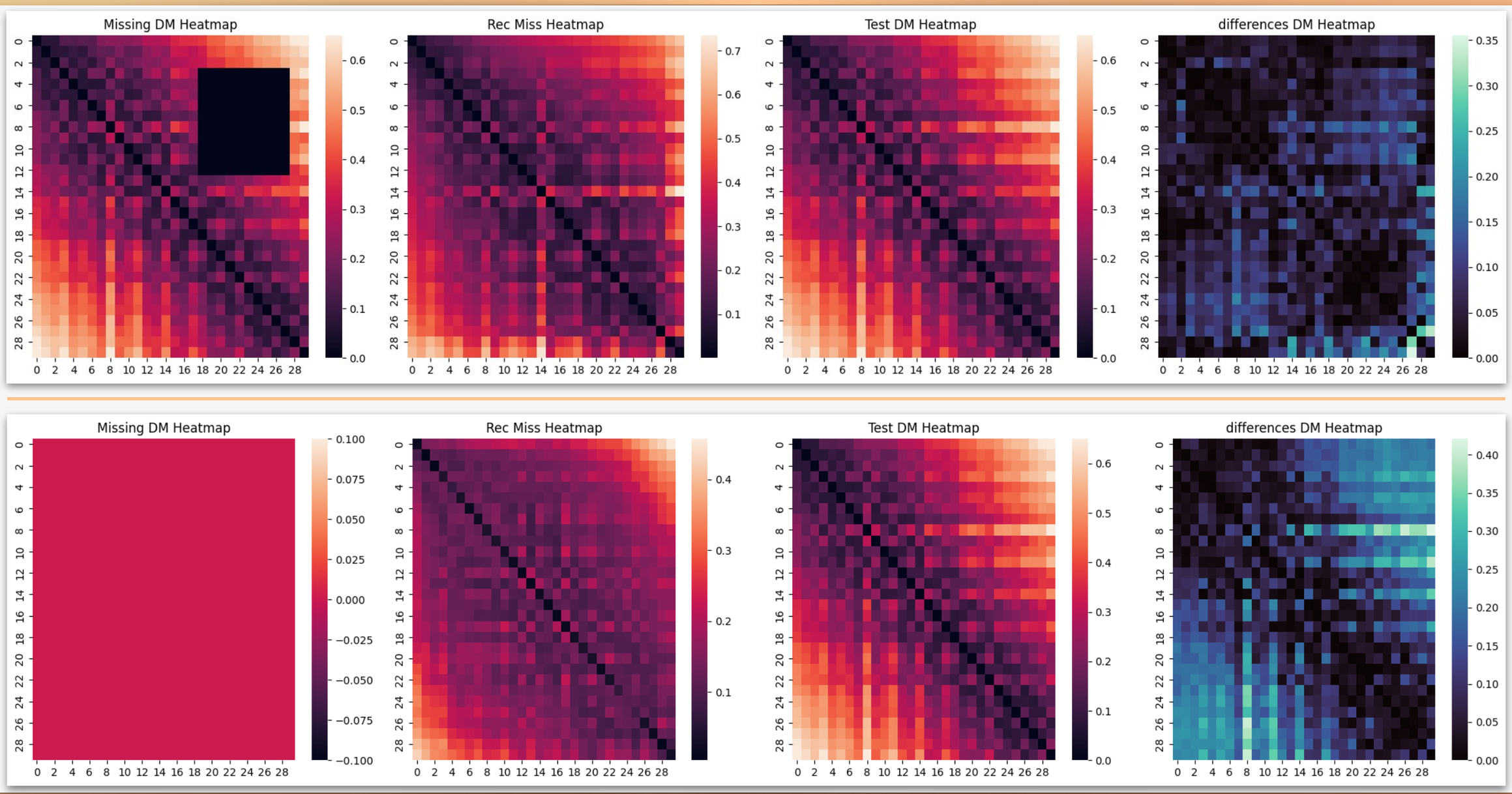
Missing data

Anomaly detection



Missing data

Anomaly detection



Chapter 5

~~Bloopers~~

Open problems

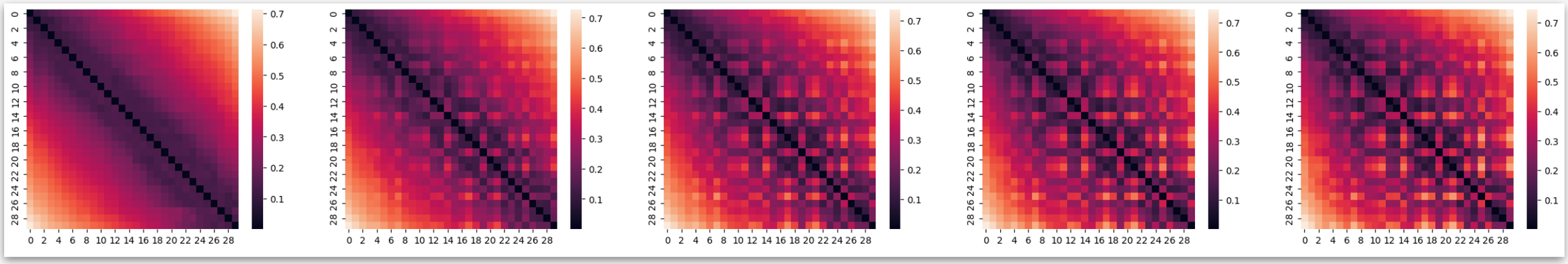
Latent space probing

40 dimensions under the latent sea

We tried to expose a possible link between each latent dimension and pattern emergence in the reconstructed matrices.

First notable discoveries

- **0** in latent space is completely featureless
- Moving along the most significant dims gradually reveals a pattern

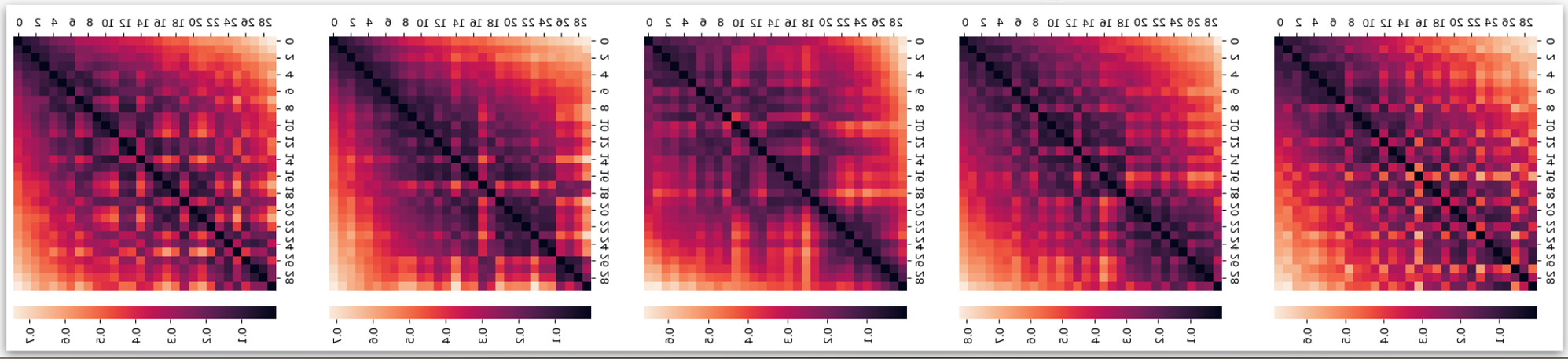


Latent space probing

40 dimensions under the latent sea

First notable discoveries

- **0** in latent space is completely featureless
- Moving along the most significant dims gradually reveals a pattern
- Each dimension encodes a different pattern

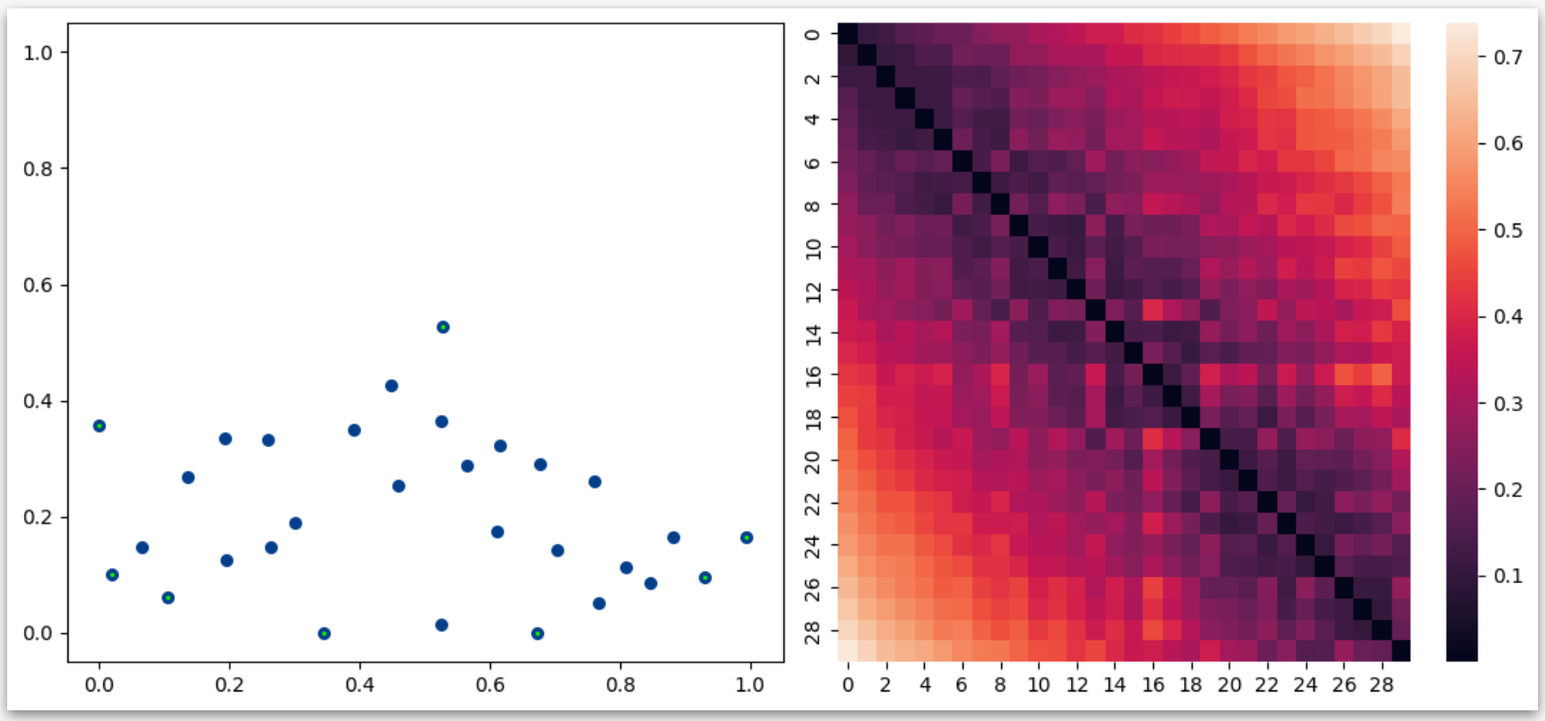


Five most relevant dims

Latent space probing

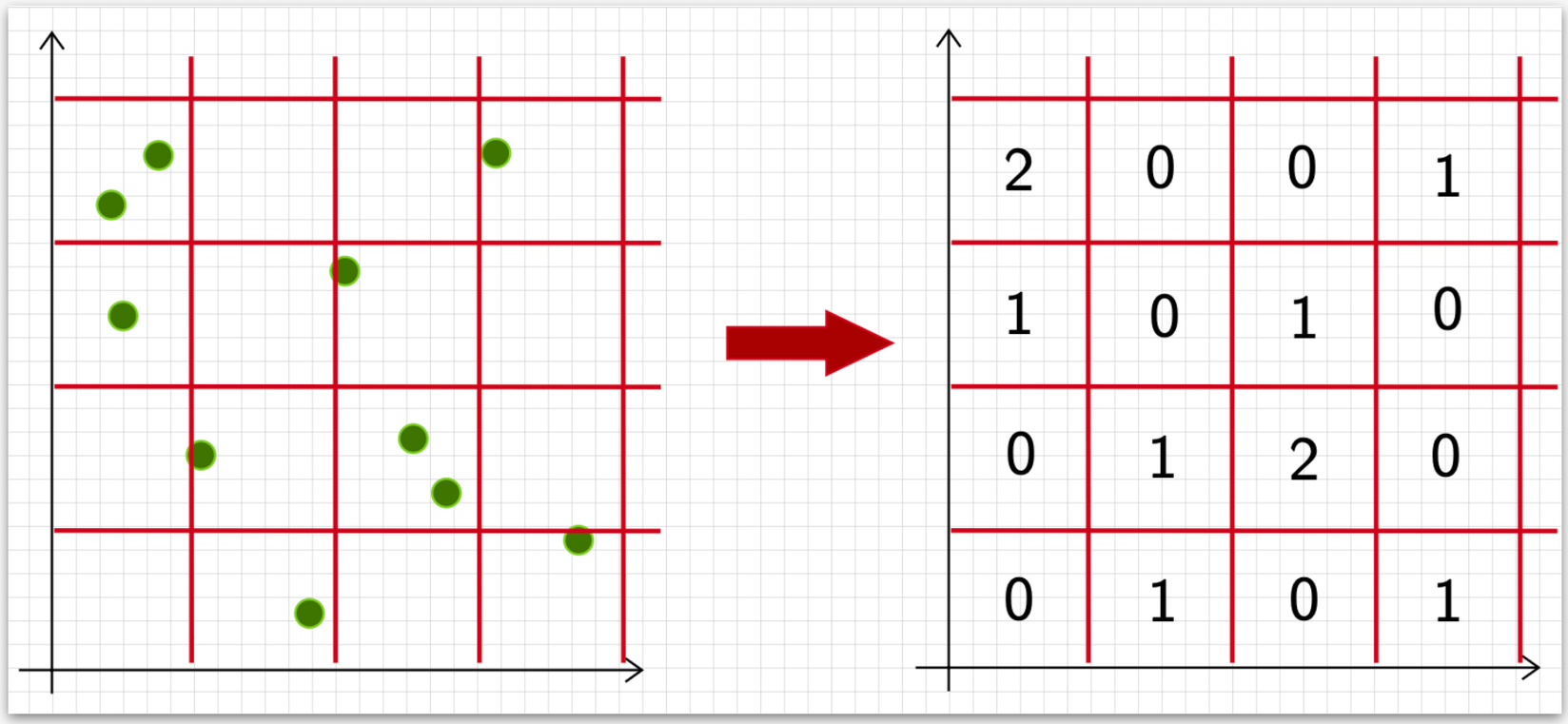
40 dimensions under the latent sea

Knowing this, using the turbo tumbler we tried reconstructing configurations sampling from those first dimensions but no apparent pattern emerged



Alternative Representation

An alternate project we worked on involved the representation of our data in an "image-like" format, in which the space is divided by a grid and the value of pixels reflects the number of particles in a cell.

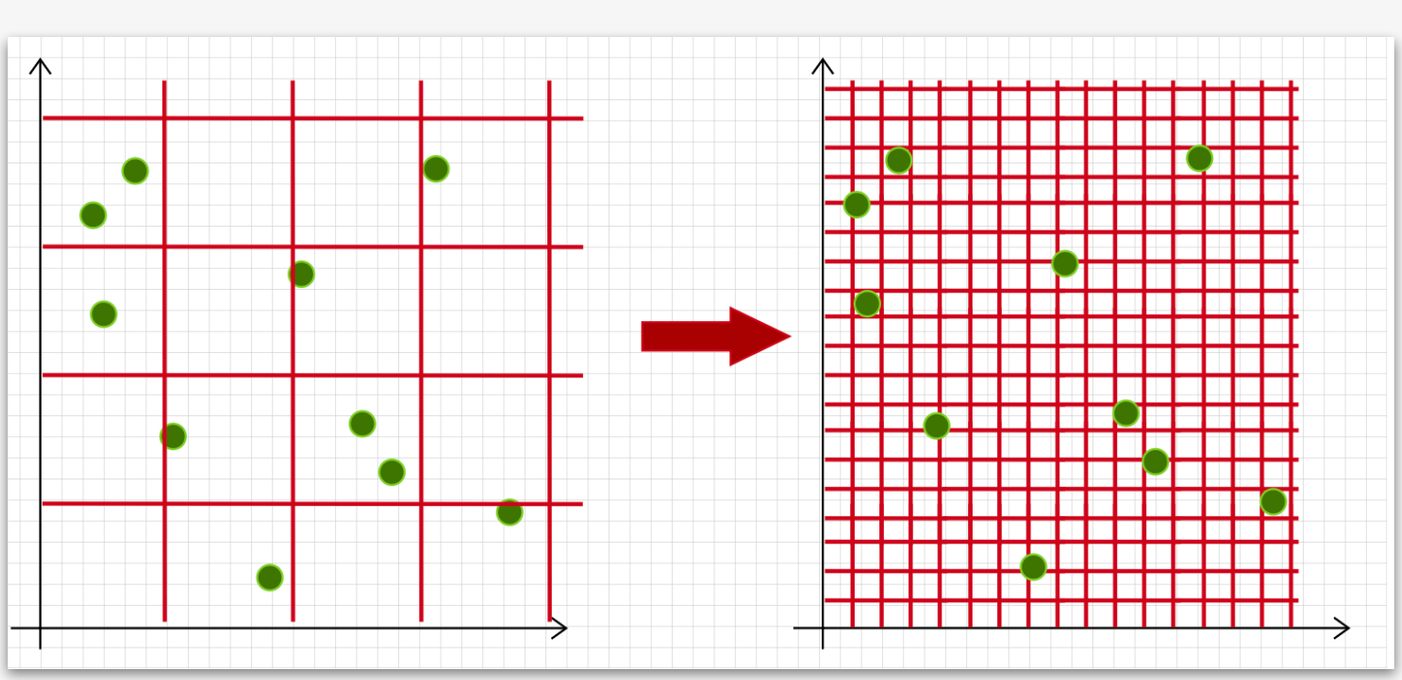


Alternative Representation

Cells have been made smaller in order to

- Only have one particle per cell
- Reduce the uncertainty in the position of the particle

The obvious drawback is an increased input dimension

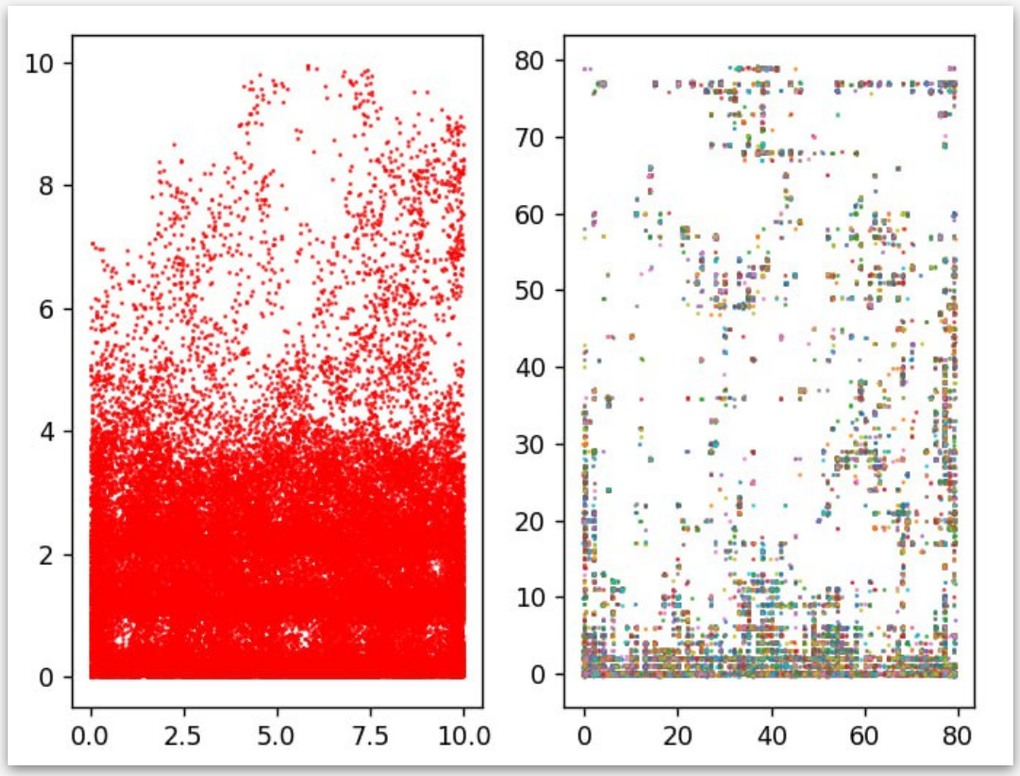


Alternative Representation

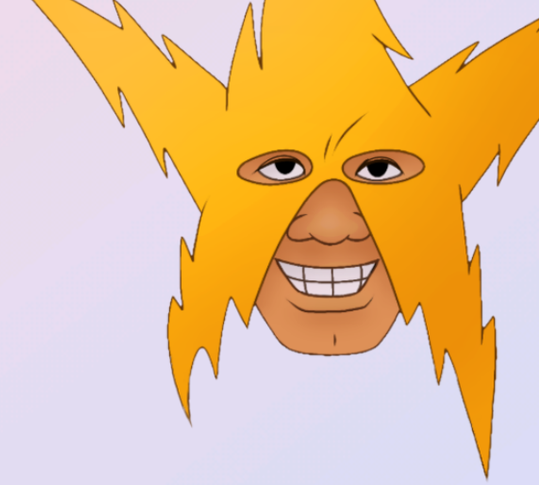
Despite our *(our = Davide's and his now charred laptop)* best efforts the network simply interpreted "live" pixels as random noise and, at most, learned some sort of y direction gradient.

Notable changes

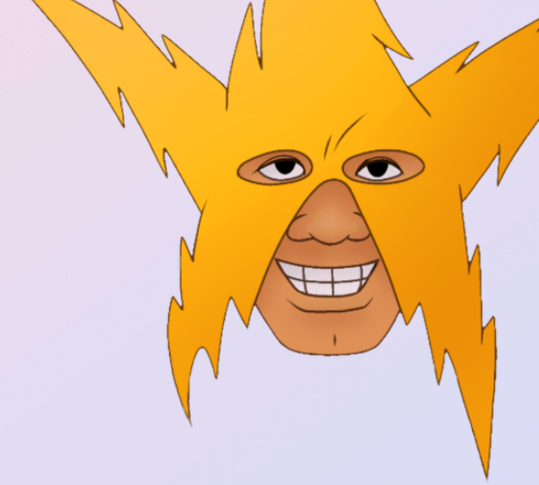
- Binary cross entropy loss
- Epoch dependent parameters



fin.



fin.



fin.