Constraints in Representation Learning

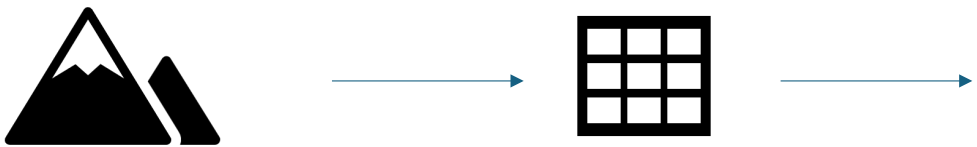
CAS AML 2024

Dr. Mykhailo Vladymyrov

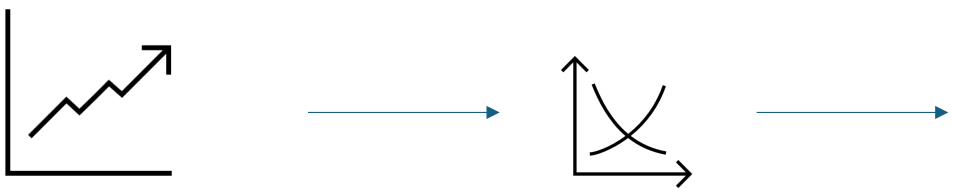
Data Science Lab
University of Bern

Representation learning

Representation Learning is an approach in machine learning where algorithms create *new data representations* from the data, that are easier to process.



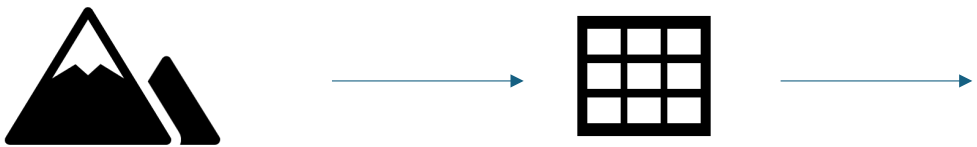
- Melting rate
- Temperature
- …
- Poetic description



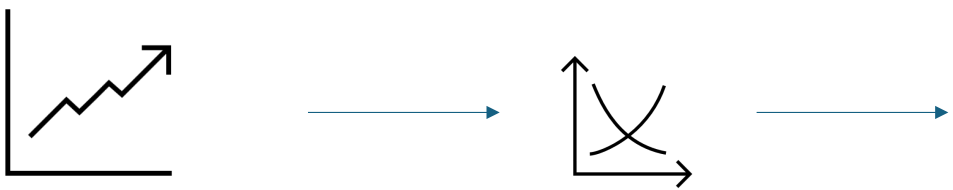
- Prediction
- Anomaly discovery
- …
- Understanding system dynamics

Representation learning

Representation Learning is an approach in machine learning where algorithms create *new data representations* from the data, that are easier to process.



- Melting rate
- Temperature
- …
- Poetic description

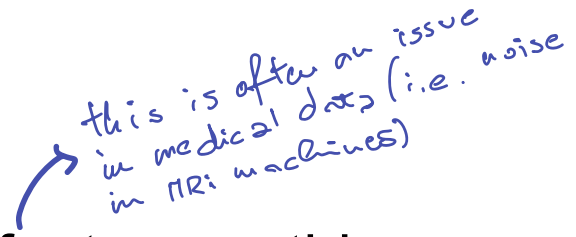


- Prediction
- Anomaly discovery
- …
- Understanding system dynamics

These representations can be designed for interpretability, reveal hidden features, or be used for transfer learning.

The data – defines “quality” of the representation

The model learns patterns in your data.



If the main pattern of your data is variety of noise features – this will be dominant feature learned.

If your dataset is too small – the model would learn features specific to your samples, rather than to the distribution they are sampled from.

The algorithm – defines properties of the representation

(1) The representation will have correlation with my original task weights on the last layer can be thought of as prototypes of the classes of my original classification task.

(2) Representation is not biased by the original classes as data in an AE are not labelled. (somehow unsup. learning) there is statistical constrains here.

(3) the model could be trained/assigned to de-noise samples and i.e. define if a new sample has likelihood to be real or not. example: model trained on generation of butterflies, then used to determine the likelihood of a new butterfly picture to be real or false

The algorithm – defines properties of the representation

Let's discuss what does this imply for:

- Classification model (1) → bias toward specific task
- AE (and VAE, VQVAE) (2) → not biased tow. specific task but statistically constrained.
- DDM (3) → extend applicability of a trained model for a new irrelated Task

The task – defines *desired* property of the representation

The different algorithm do for sure different things, but other elements play a role as well, for example the kind of loss that we use. the loss used has a strong impact on the performance of the model us the task.

PERCEPTIVE LOSS:

Called perceptive as usually it's used for image domain.
Compare representation of the model at the lower levels with the latent representation.
Practically we end up comparing activation function of different representation (of the feature maps). We don't compare the outputs, but the activation of the values in a different model.

Representation produce by a low level of a ≠ model (pretrained) → to compare with the output of the AE.

a low level representation will have only high frequency components (i.e. edges)

this approach force the AE to pay more attention to the high frequency components of image

(butterfly with claws)

The task – defines *desired* property of the representation

e.g. consider the difference:

- AE with L2 vs L1 vs Perceptive Loss
- AE vs VAE vs VQVAE → imply some stat prop. of the latent space. Close together sample in the input space, far samples will be forced apart.
- DDM in original vs latent space

↓

Depending on what we do in the latent space there'll be as well a difference. DDM will give a gradient of likelihood.

perceptive loss, formula example:

$\arg\min_{\theta} |X - AE(X|\theta)|$

We have in principle ∞ way to represent the data and consequently infinite way to train the AE

there should be a difference between a property being unlikely and a property being unknown by the model. It should be possible to disentangle & encode and define which property are unknown and which are unlikely.

So how can we steer it?

How can we make the representation the way we want them?

We have:

- Constraints
- Transformations
- Conditions

Constraints

Constraints

→ Are defined by the model: → Architecture, → loss function, → activation function

- Size of representation
- Statistical properties – e.g. using variational inference (VAE handle statistic of the sample more than the sample itself).
- Space in which representations "Live":
  - Vector
  - Matrix
  - Vector field → diffusion (the model learn likelihood of feature in the vector space).

Spacial dimension of the latent space are continuous.
The representation can be different depending on what are we using: can be a vector of size 2 if we flatten in a 1D vector, but can be also something different. If we increase the dimensions, those new dimension will have some constraints in the vector space that depends from the model.

Transformations

Input transformations

the very augmentation that we apply will have an impact on the training of the model

Augmentations:

- Translation/rotation
- Noise
- Derivate explicit representation

=> help the model to generalize but at the same time tells the model that those properties are not important. (teach actually the model to ignore them).

Representation transformations

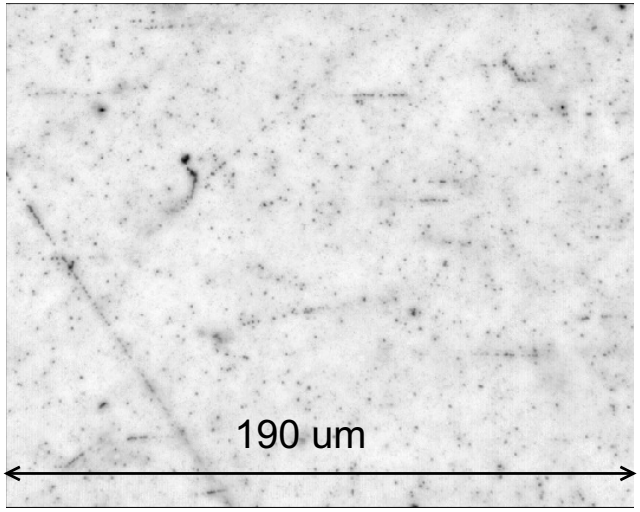
We somehow already saw: in the VAE we sample in a gaussian distribution to smooth out to a gaussian distrib.

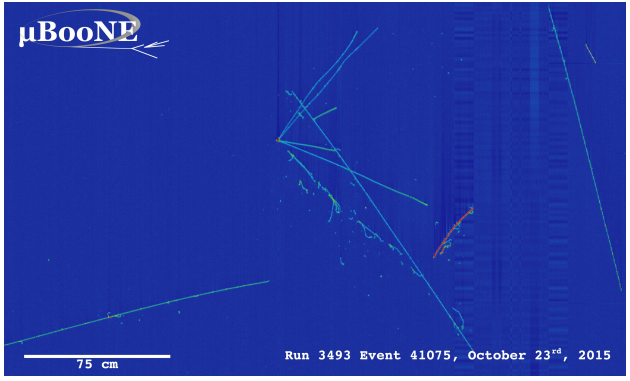
- Sampling from modelled distribution
- Equivariance
- Invariance

Some transformation in the latent space will not change the reconstruction.

EXAMPLE of APPLICATION:

Tracking: finding lines





Algorithms:

- Leverage small part of available information
- Tedious adaptation to new conditions

Supervised Machine Learning

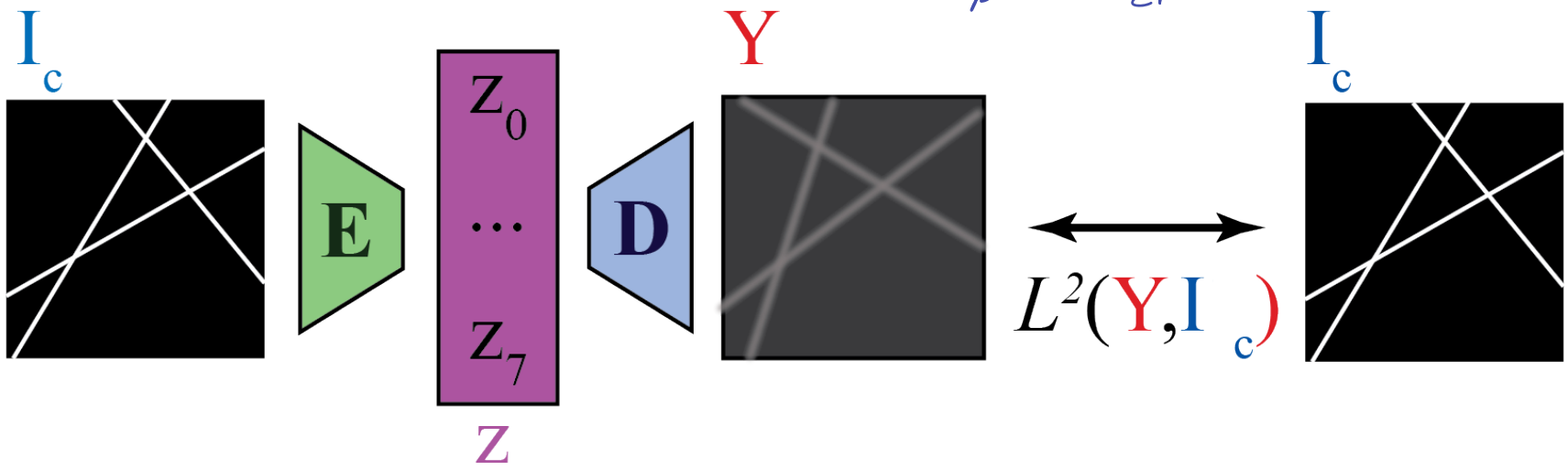
- Leverage all low-level information
- Requires labeled data / training on simulations

Unsupervised Machine Learning

- Leverage all low-level information
- No / small amount of labeled data

Proposed model

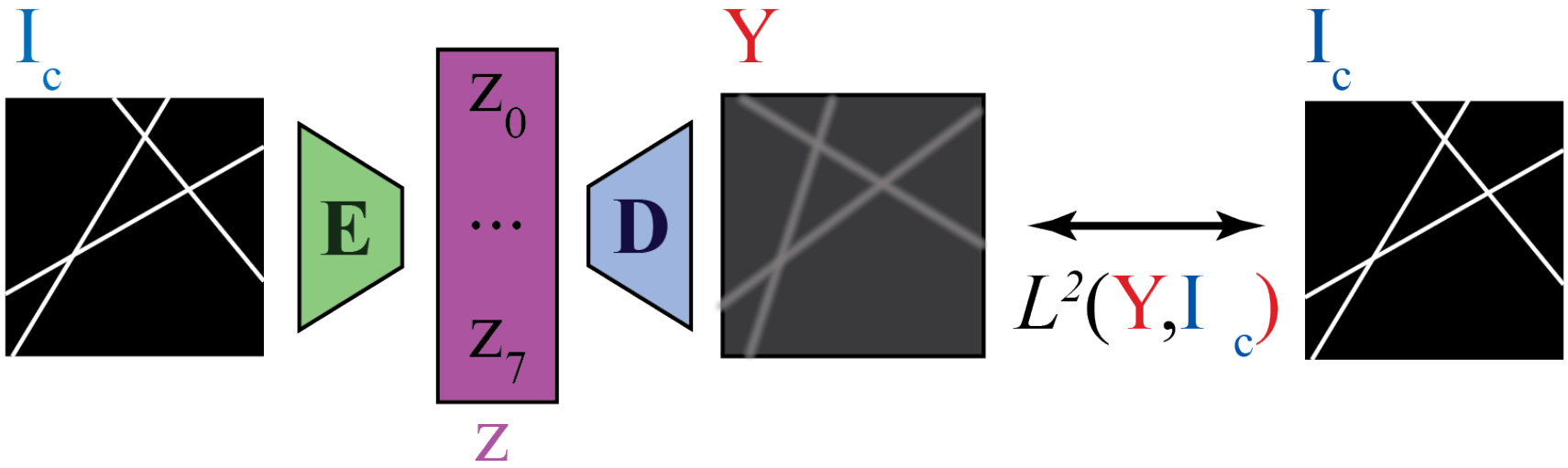
Autoencoder input image with some lines, decoder reconstruct from latent space. We want this (params/lines) to be a parameter



– parameters that *shall* correspond to the ith track

Proposed model

line as parameters are a constrain.



– parameters that *shall* correspond to the ith track

We need to 1. disentangle different tracks; 2.disentangle track parameters

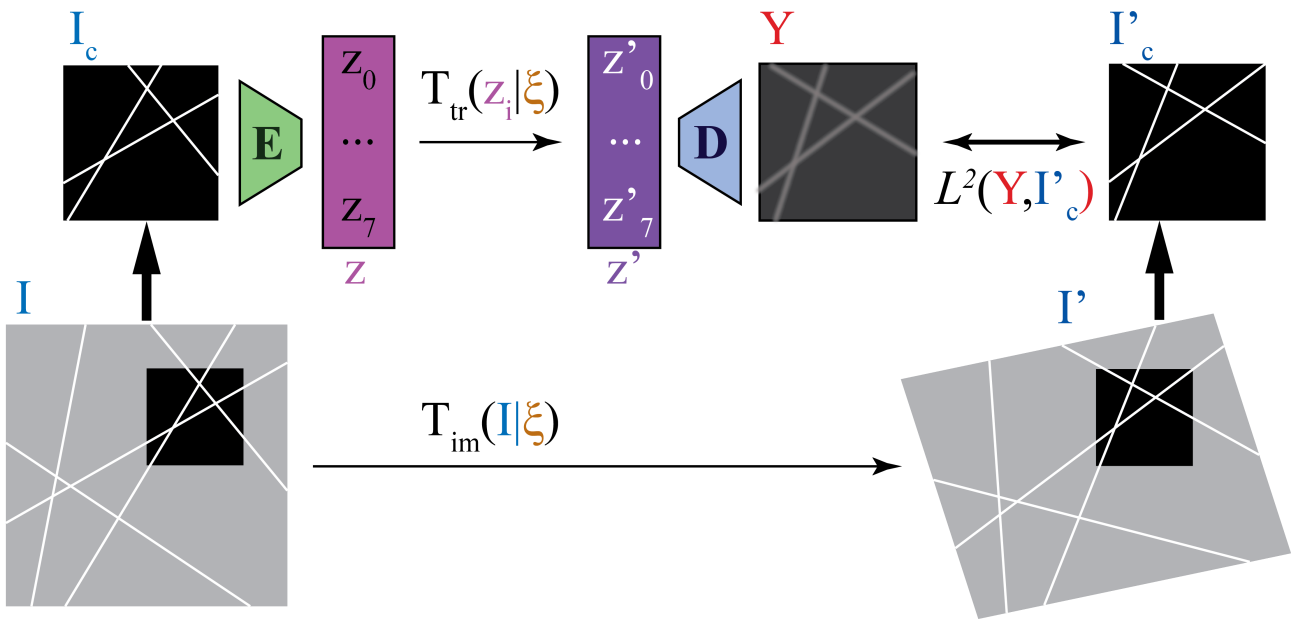
Equivariance constraint

, – geometrical transformations in the image and the latent representation space, parametrized by the same parameter set

Demand equivariance of encoder and decoder:

->

Equivariance constraint



$I' = T_{im}(I|\xi)$, $z'_i = T_{tr}(z_i|\xi)$ –

, – geometrical transformations in the image and the latent representation space, parametrized by the same parameter set

Demand equivariance of encoder and decoder:

$$T_t(E(I)|\xi) = E(T_{im}(I|\xi)), \forall \xi$$

$$D(T_{tr}(z|\xi)) = T_{im}(D(z)|\xi), \forall \xi$$

$$E, D = \underset{E,D}{\operatorname{argmin}} \mathbb{E}_{I,\xi} L^2(Y, I'_c) = \underset{E,D}{\operatorname{argmin}} \mathbb{E}_{I,\xi} L^2(D(T_{tr}(E(C(I))|\xi)), C(T_{im}(I|\xi)))$$

predicted angle of the lines

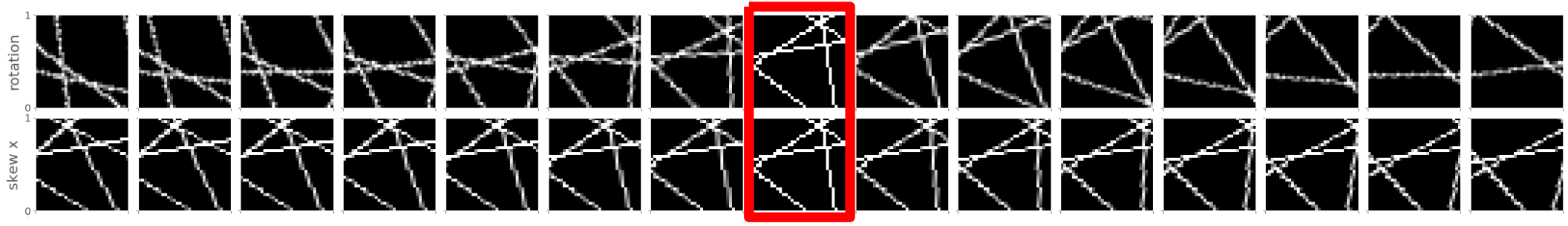
There is relation b/w the image space and the latent space

if the image rotate, the latent space should rotate too

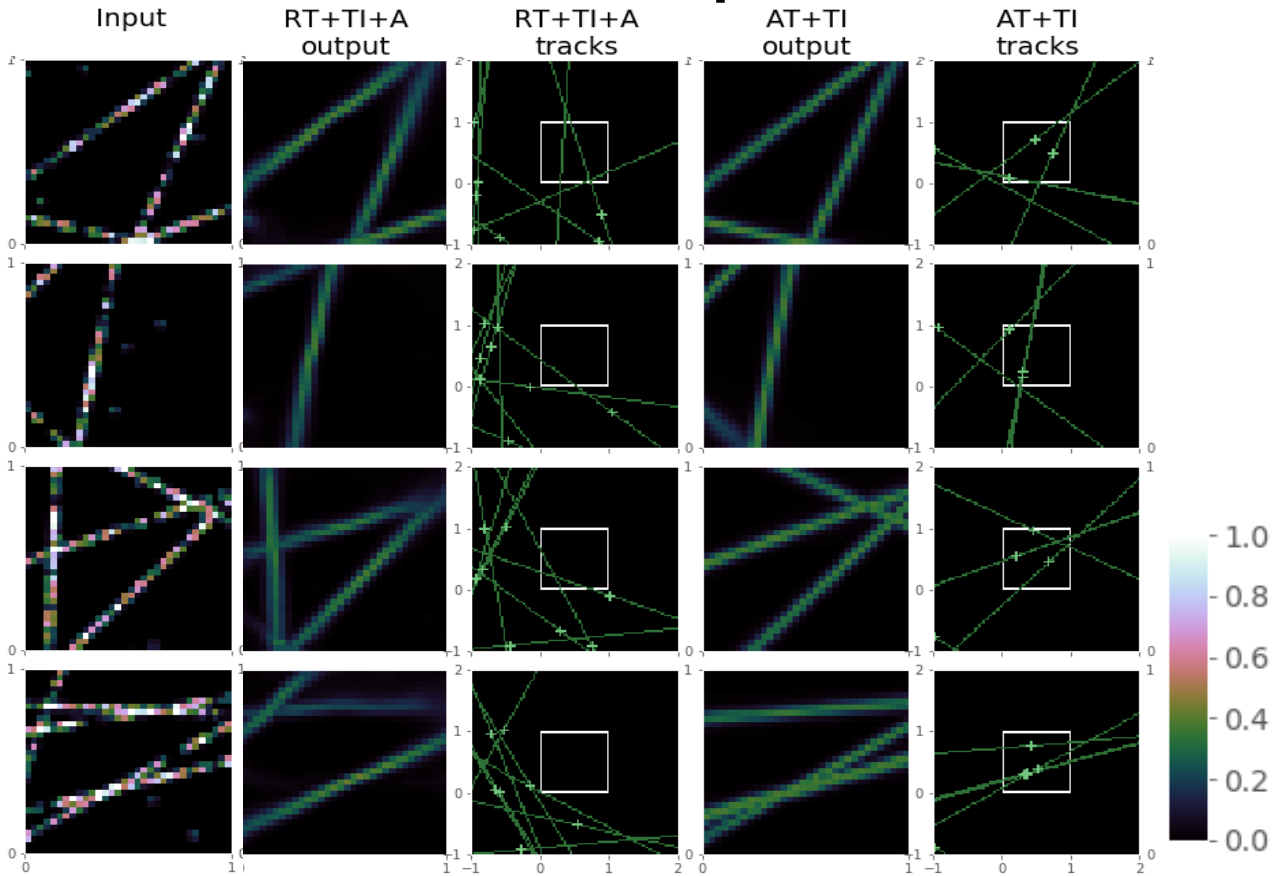
Representation transformations

By ignoring some coherent transformation we can obtain meaningful representations also without labelling data

Affine Transformations: rotation, translation, skew, scale



Autoencoder performance



Imposing equivariance constraint on the autoencoder under geometrical transformations in the image and latent representation domains enables the model to “discover” the existence of multiple lines in the presented images in a fully unsupervised manner.

M.Vladymyrov, A. Ariga,
Novel tracking approach based on fully-unsupervised disentanglement of the geometrical factors of variation

JINST 15 (2020) 03, P03009

Conditions

Conditions

- Distribution properties

  e.g. for VAE we had:

Conditions

- Sparseness – like Lasso regression
  Impose by augmenting the loss function:

We can imply ~~illest~~ loss ways

$$L = L_{task} + \lambda \cdot \sum_i (z_i - 1)$$

project on a unit sphere

$$z_i \longrightarrow \frac{z_i}{|z_i|}$$

We can imply some normalization

Conditions

- Integrity:

e.g. terms sum up to 100%:

project on a unit sphere:

Conditions

We can, the same way, imply also any physical property.

- Physical properties

Any property in fact can be added similar way (known as Physics informed Neural Networks). Given equation

$$f(z) = 0$$

$$L = L_0 + \lambda \cdot f(z)$$

Coeff. lambda balance the order of my equation.

λ

Coefficient balances scale of the function value range

$f(z)$

Conditions

- Physical properties

Any property in fact can be added similar way (known as Physics informed Neural Networks). Given equation

$$f(z) = 0$$

$$L = L_0 + \lambda \cdot f(z)$$

Coefficient balances scale of the function value range

- Static – features of the output.
- Dynamic – gradient of features

PINNs

differential equation

Consider searching for a PDE solution:

$$u_t + uu_x - (0.01/\pi)u_{xx} = 0, \quad x \in [-1, 1], \quad t \in [0, 1],$$
$$u(0, x) = -\sin(\pi x),$$
$$u(t, -1) = u(t, 1) = 0.$$

$$f := u_t + uu_x - (0.01/\pi)u_{xx},$$

Train should minimize the total loss.

```
def u(t, x):
    u = neural_net(tf.concat([t,x],1), weights, biases)
    return u
```

Z with t and x the model predict U

Correspondingly, the *physics informed neural network* $f(t, x)$ takes the form

```
def f(t, x):
    u = u(t, x)
    u_t = tf.gradients(u, t)[0]
    u_x = tf.gradients(u, x)[0]
    u_xx = tf.gradients(u_x, x)[0]
    f = u_t + u*u_x - (0.01/tf.pi)*u_xx
    return f
```

The shared parameters between the neural networks $u(t, x)$ and $f(t, x)$ can be learned by minimizing the mean squared error loss

$$MSE = MSE_u + MSE_f, \quad \text{Total loss} \tag{4}$$

where

$$MSE_u = \frac{1}{N_u}\sum_{i=1}^{N_u} |u(t_u^i, x_u^i) - u^i|^2, \quad \leftarrow \text{boundary condition}$$

and

$$MSE_f = \frac{1}{N_f}\sum_{i=1}^{N_f} |f(t_f^i, x_f^i)|^2. \quad \leftarrow \text{use of the function } f.$$

Total Loss will have the loss on the boundary condition + the loss of making eq=0

arXiv:1711.10561v1

PINNs

Instead of the NN to solve the eq, authors try to predict the Hamiltonian of the system.

$$\mathcal{L}_{HNN} = \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{p}} - \frac{\partial \mathbf{q}}{\partial t} \right\|_2 + \left\| \frac{\partial \mathcal{H}_\theta}{\partial \mathbf{q}} + \frac{\partial \mathbf{p}}{\partial t} \right\|_2$$

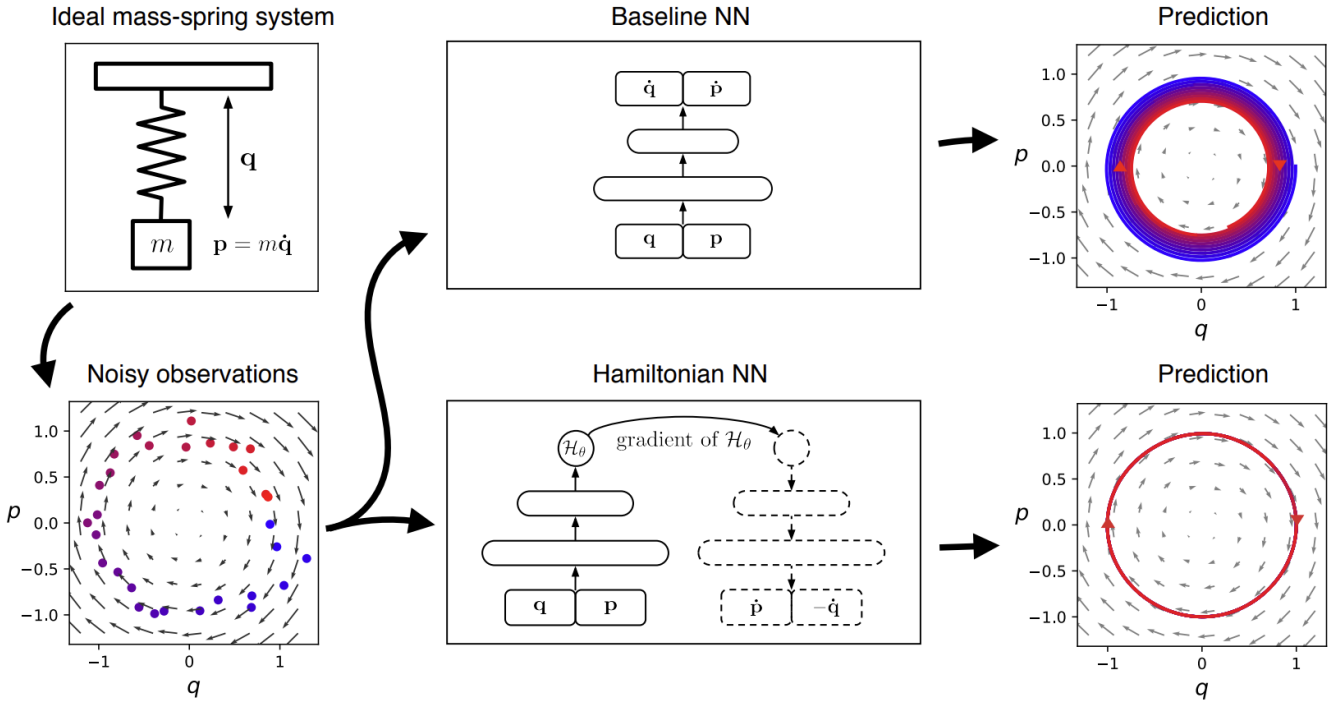


Figure 1: Learning the Hamiltonian of a mass-spring system. The variables q and p correspond to position and momentum coordinates. As there is no friction, the baseline's inner spiral is due to model errors. By comparison, the Hamiltonian Neural Network learns to *exactly* conserve a quantity that is analogous to total energy.

The way you shape your model will determine how your model learn.

As you can in principle do everything

check the paper

arXiv:1906.01563v3

Further readings

- PINNs:
  - https://arxiv.org/pdf/1906.01563
  - https://towardsdatascience.com/physics-informed-neural-networks-pinns-an-intuitive-guide-fff138069563
  - https://arxiv.org/pdf/1711.10561

links

- NN important tools:
  - Normalizations: https://www.pinecone.io/learn/batch-layer-normalization/
  - Gradient propagation: https://mrinalwalia.medium.com/understanding-the-vanishing-gradient-problem-in-deep-learning-c648a4f16b05
  - Stop gradient

Tutorials, topics to look into

- Interpreting learned features in CNNs:
  - https://distill.pub/2017/feature-visualization/
  - https://distill.pub/2018/building-blocks/
  - https://distill.pub/2018/differentiable-parameterizations/

some more reading on interpretation of the latent space & meaningful representation

Workshop

1h + 15min

Plan the project

- What task are you attempting?
- Which representations would be useful?
- Which tools would you use?
- Which constraints are imposed by the model?
- Which conditions would you apply?

PROJECT:

- do smth about represent. learning
- ideally adding some constrains

- which could be the properties of my repres?
- which simmetry?
- how would my latent space look like?