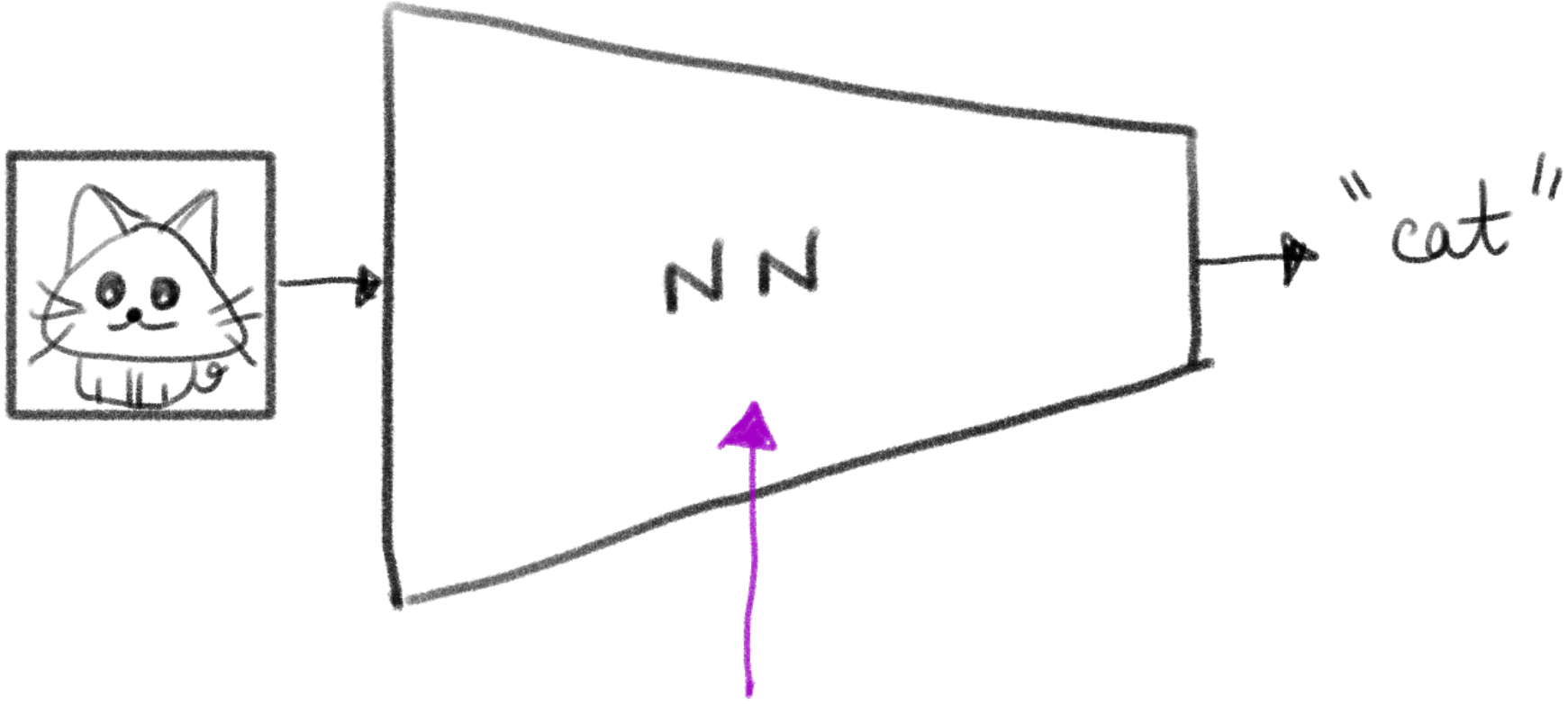
HDDL - LESSON 3

AUTOENCODERS

SELF-SUPERVISED LEARNING

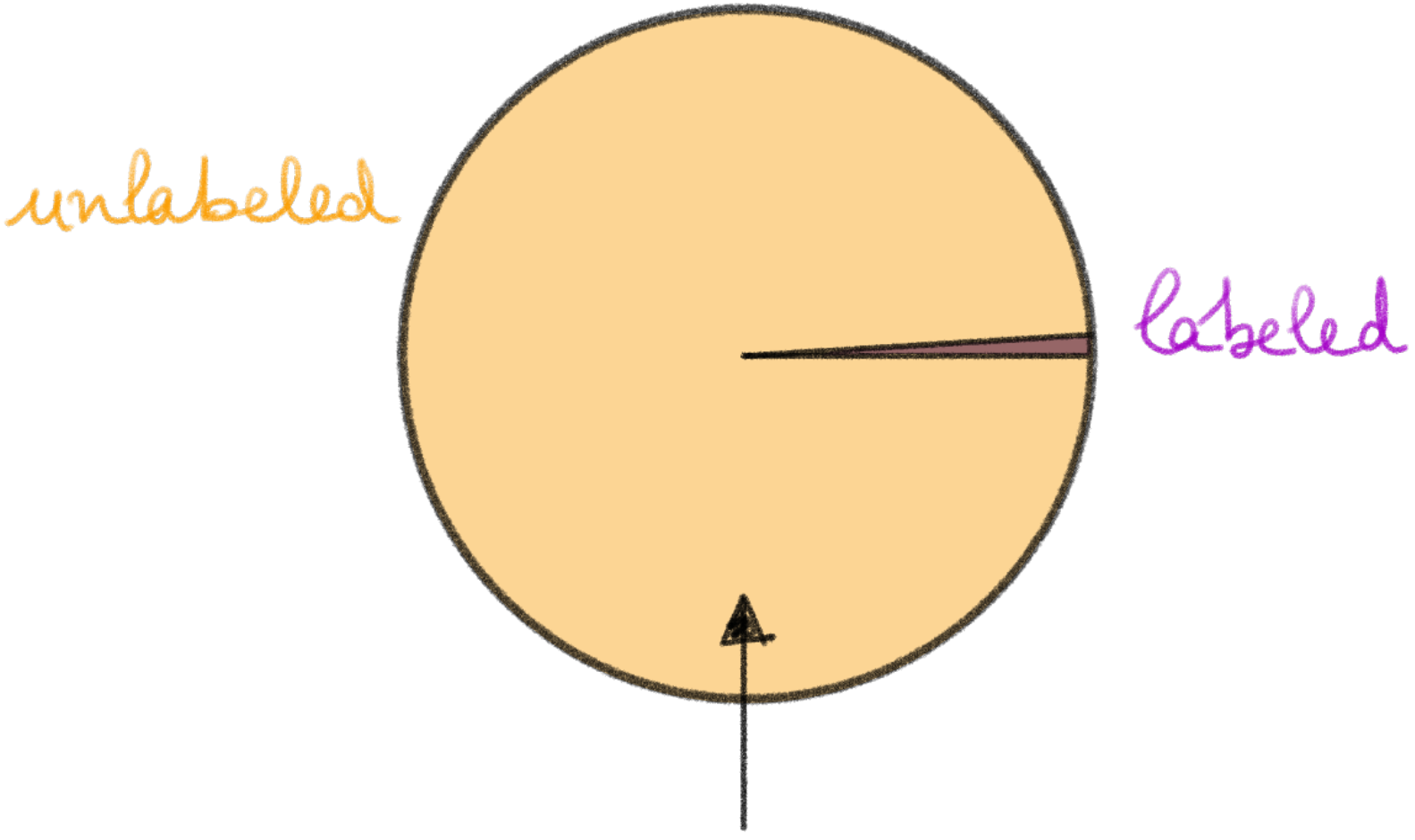
JOSEBA DALMAU

Supervised Learning



What do we need in order to train this neural network?

DATA



how do we leverage all this data?

What is an Autoencoder?

Autoencoder

An unsupervised learning technique for the task of representation learning.

Unsupervised learning

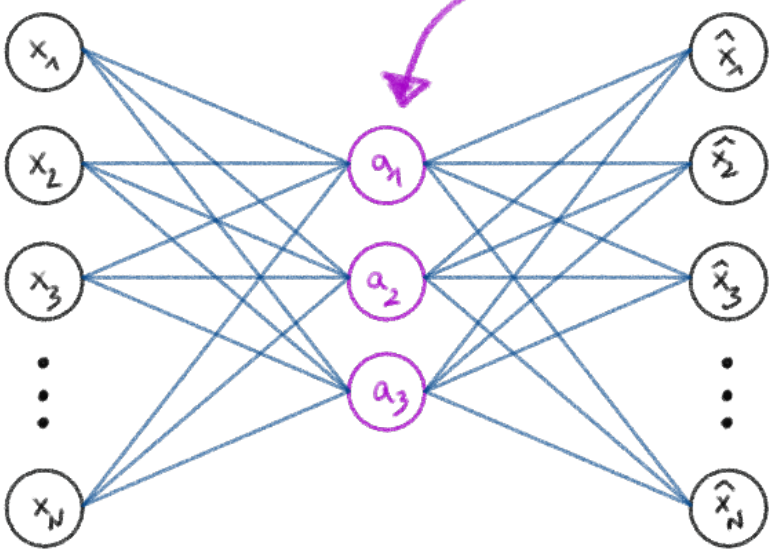
- no labels, only input!

Representation learning

- Latent representation (lower dimensions)
- Transferable representations

Main Idea

Architecture: NN with a bottleneck



Loss: reconstruction error

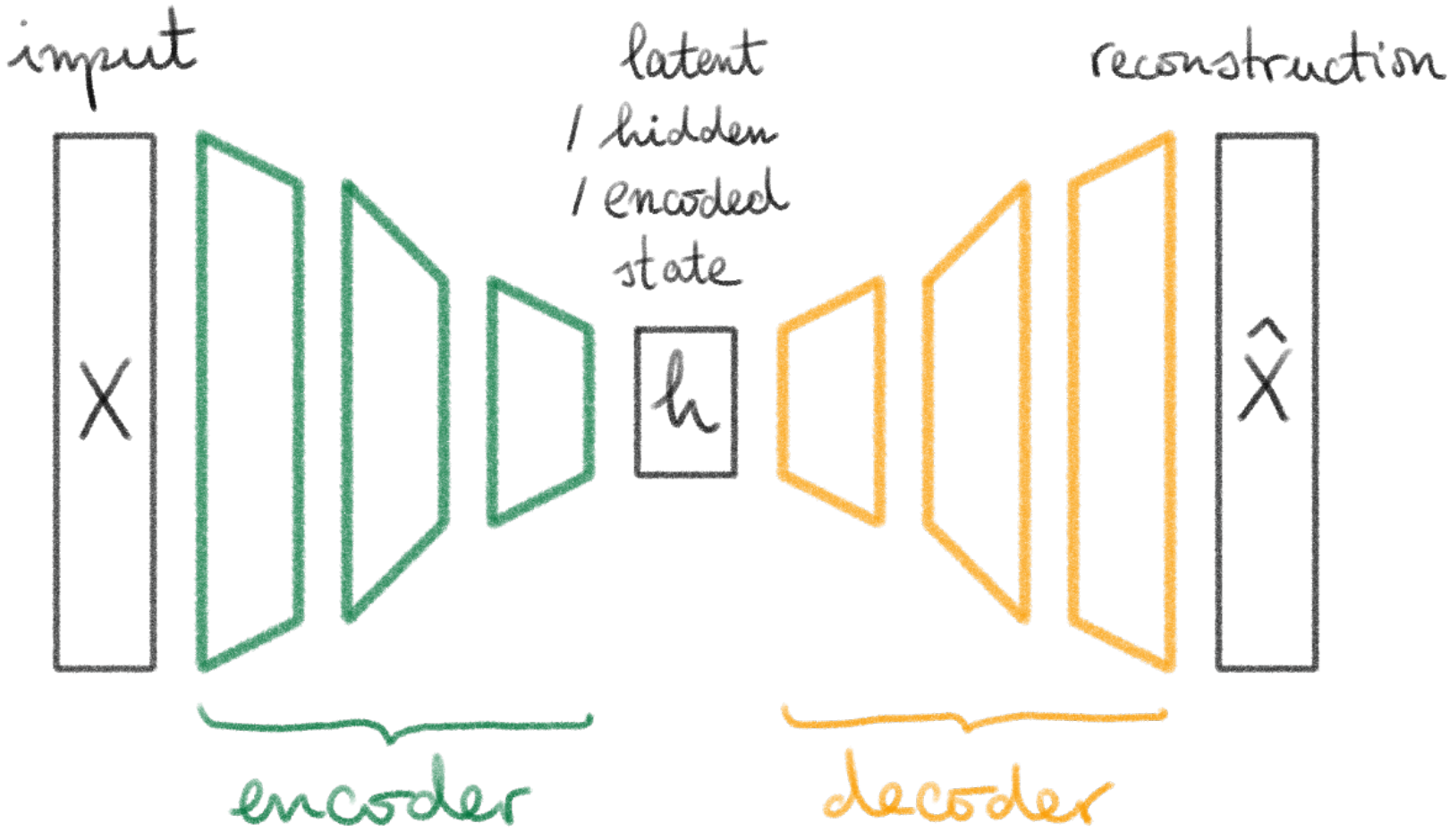
$$L(x, \hat{x})$$

QUESTIONS AND REMARKS

- Why is the bottleneck important?
- This architecture is:
  - → unsupervised
  - → provides latent representations
  - → are the latent representations transferable?
- Remark: linear autoencoder ≡ PCA
- What is the risk with a reconstruction error $L(x, \hat{x}) \simeq 0$?

Names and Abbreviations

- Autoencoder → AE



The Ideal Autoencoder

Sensitive enough to the inputs
⇒ reconstruct accurately

Insensitive enough to the inputs
⇒ avoid memorizing

$$\text{Loss} = L(x, \hat{x}) + \lambda \text{Regularizer}$$

Undercomplete Autoencoder

Idea: Rely on the bottleneck

If $\dim(\text{latent representation}) << \dim(\text{input})$

we expect $\Rightarrow$
- Learn the most important attributes
- Be able to reconstruct input

Loss = $L(x, \hat{x})$ (no Regularizer)

WAIT A MINUTE...

... how is this different from PCA?

NNs can learn non-linear relationships

$\Rightarrow$ AE $\equiv$ more powerful (nonlineal) generalization of PCA

PCA

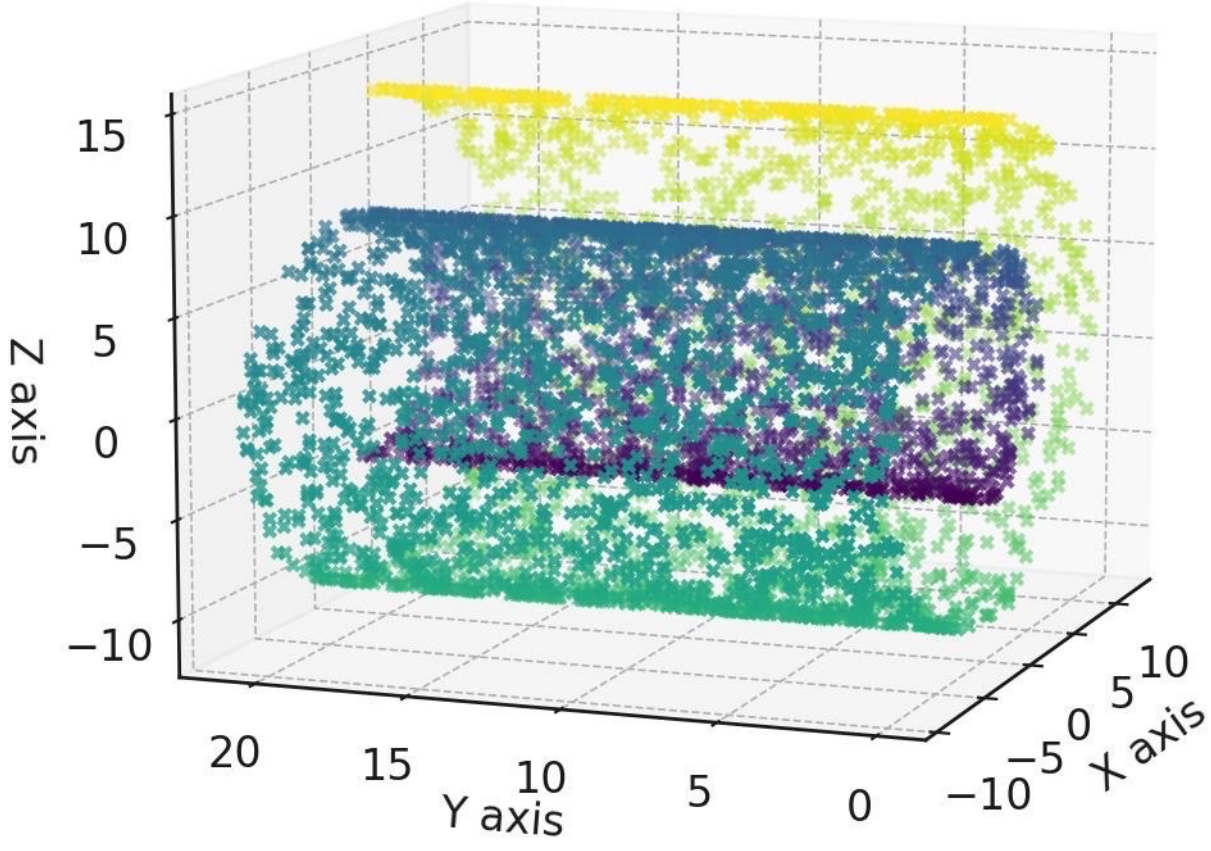
↓ learn

lower dimensional hyperplane

AE

↓ learn

lower dimensional non-linear manifold



Problem: "Capacity" of the Autoencoder

If the encoder and decoder are complex enough ...

... even with a very small number of latent dimensions ...

... input data can be memorized!

Sparse Autoencoders

Idea

~~Reduction in nb. of nodes of hidden layers~~

↓

Penalize activations within a layer

REMARKS

(i) Usually: Regularizer $\xrightarrow{\text{penalize}}$ weights

e.g. $\sum_{i,j} |W_j^i|$

Sparse AEs: Regularizer $\xrightarrow{\text{penalize}}$ activations

(ii) The neurons that activate are data-dependent

(iii) Latent dimension indep. of Regularization

Regularization

Given: sparsity parameter ρ

(average activation of a neuron over a collection of samples)

Define:

$$\hat{\rho}_j^{(h)} = \frac{1}{m} \sum_{i=1}^{m} a_j^{(h)}(x^{(i)})$$

- h: layer
- j: neuron
- m: nb. of training obs.
- $a_j^{(h)}$: activation on layer h neuron j
- $x^{(i)}$: i-th training obs

REGULARIZATION

Given: sparsity parameter ρ

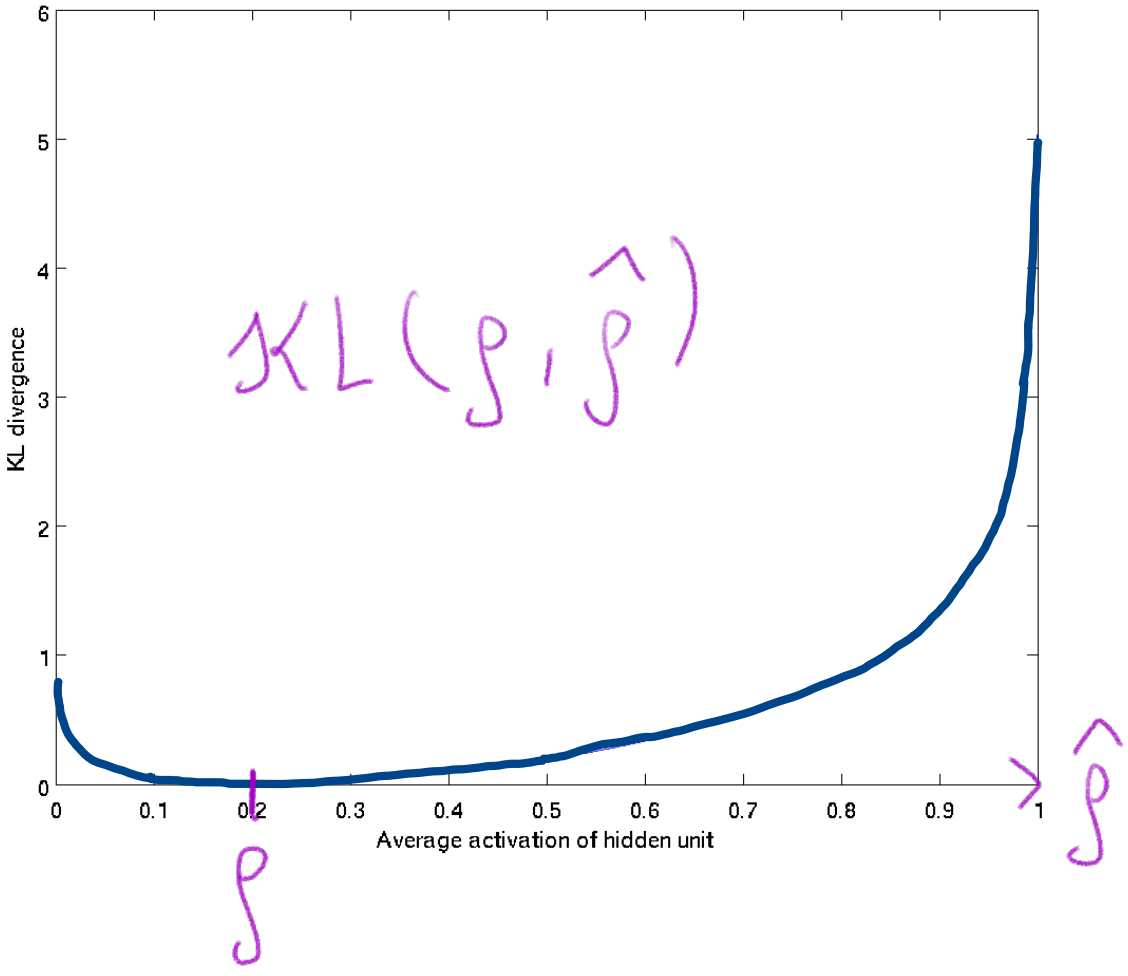
(average activation of a neuron over a collection of samples)

Define:

$$\hat{\rho}_j^{(h)} = \frac{1}{m} \sum_{i=1}^{m} a_j^{(h)}(x^{(i)})$$

$$\text{Loss} = L(x, \hat{x}) + \lambda \sum_{h,j} KL(\rho, \hat{\rho}_j)$$

KL-Divergence



$KL(\rho, \hat{\rho})$ has a minimum at $\hat{\rho} = \rho$ and is NOT symmetrical!

BACKPROPAGATION

Given that:

$$\hat{\rho}_j^{(h)} = \frac{1}{m} \sum_{i=1}^{m} a_j^{(h)}(x^{(i)})$$

$$\text{Loss} = L(x, \hat{x}) + \lambda \sum_{h,j} KL(\rho, \hat{\rho}_j)$$

How to perform backpropagation?

BACKPROPAGATION

Given that:

$$\hat{\rho}_j^{(h)} = \frac{1}{m} \sum_{i=1}^{m} a_j^{(h)}(x^{(i)})$$

$$\text{Loss} = L(x, \hat{x}) + \lambda \sum_{h,j} KL(\rho, \hat{\rho}_j)$$

How to perform backpropagation?

Sol: perform multiple forward passes per backward pass!

EXERCISE

Consider an AE with an input dim of n. Call the weights of the first (FC) layer $W^{(1)}$ so that activations after the 1st layer are

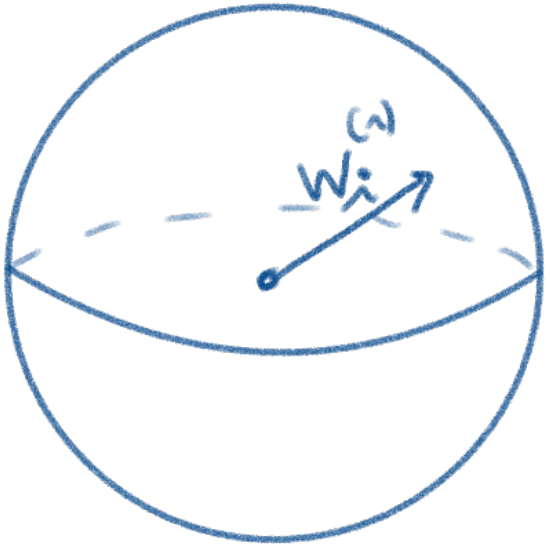
$$a_i^{(1)}(x) = \sum_{1 \le j \le n} W_{ij}^{(1)} x_j + b_i^{(1)}, \quad \text{for } i = 1, \dots, n_1$$

Assumming that the input satisfies $\|x\|_2^2 \le 1$. Find the input x that maximizes $a_i^{(1)}(x)$.

Solution

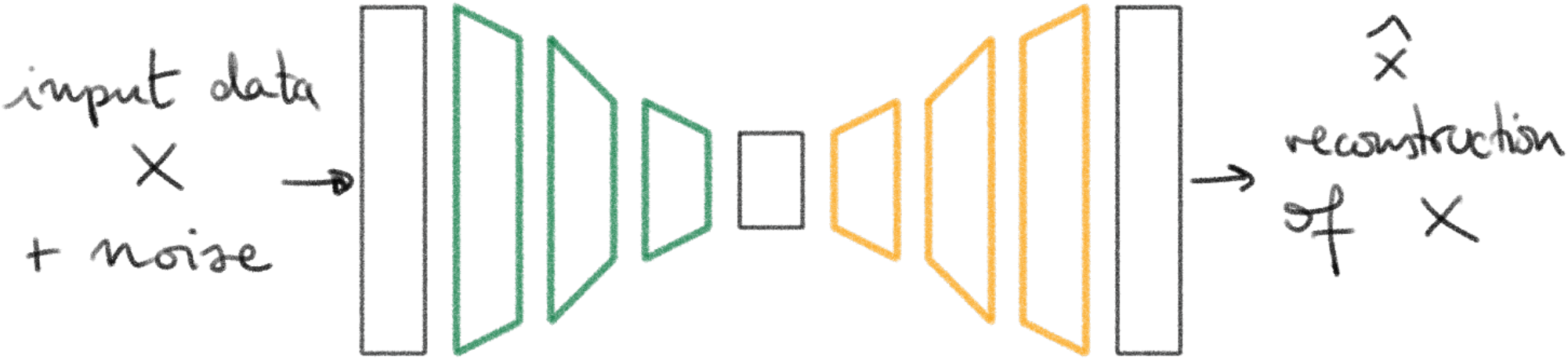
$$a_i(x) = \sum_{j=1}^{n} W_{ij}^{(1)} x_j + b_i = \langle W_i^{(1)}, x \rangle + b_i$$

is maximal when $x = \dfrac{W_i^{(1)}}{\| W_i^{(1)} \|_2}$



Denoising Autoencoders

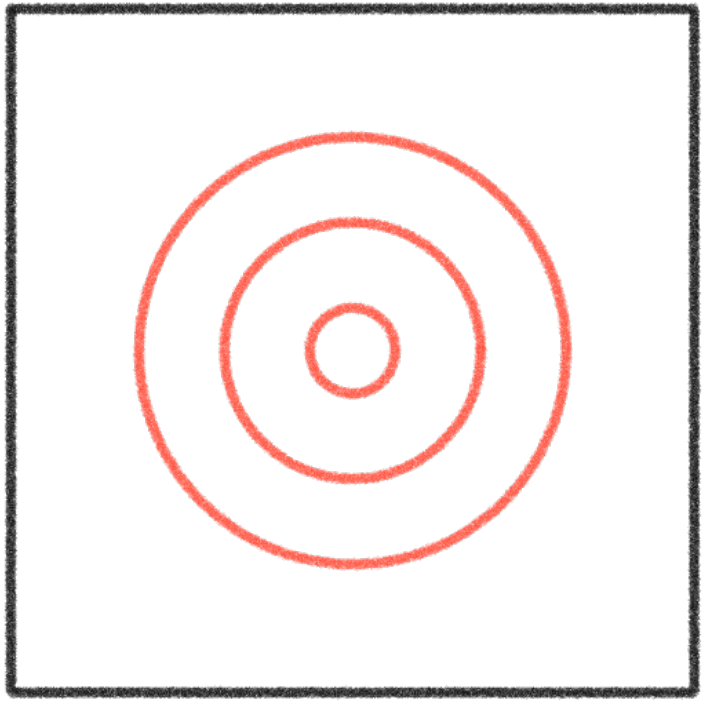
Idea



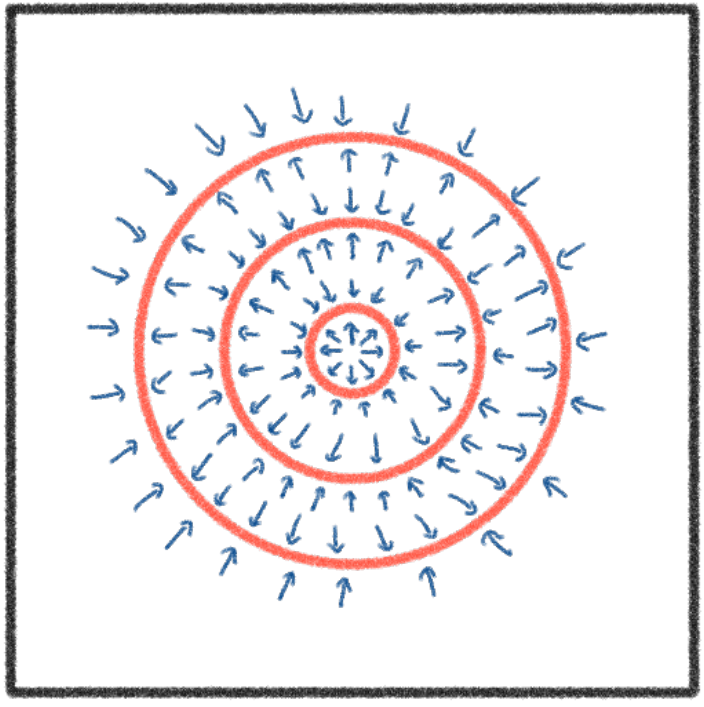
$\Rightarrow$ Prevent AE from memorizing

(why?)

LEARNED VECTOR FIELD



original data



learned vector field

What vector field do we learn in regions where no training data is available?

VARIATIONAL AUTOENCODERS

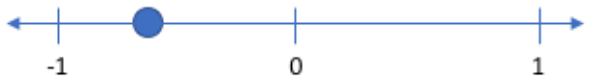
AE: input $\xrightarrow{\text{encoder}}$ representation (single value)

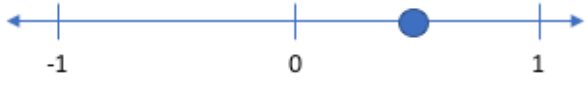
VAE: input $\xrightarrow{\text{encoder}}$ representation (probability distribution)

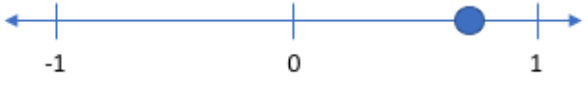
$\Rightarrow$ VAEs can easily generate new, unseen data.

Example

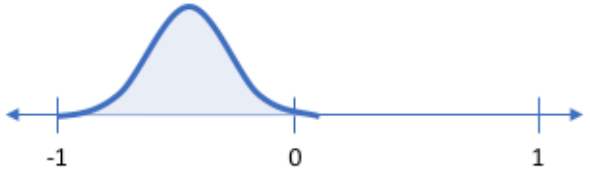
Smile (discrete value)



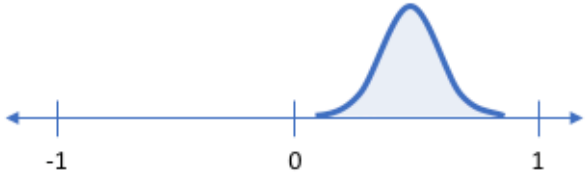




Smile (probability distribution)

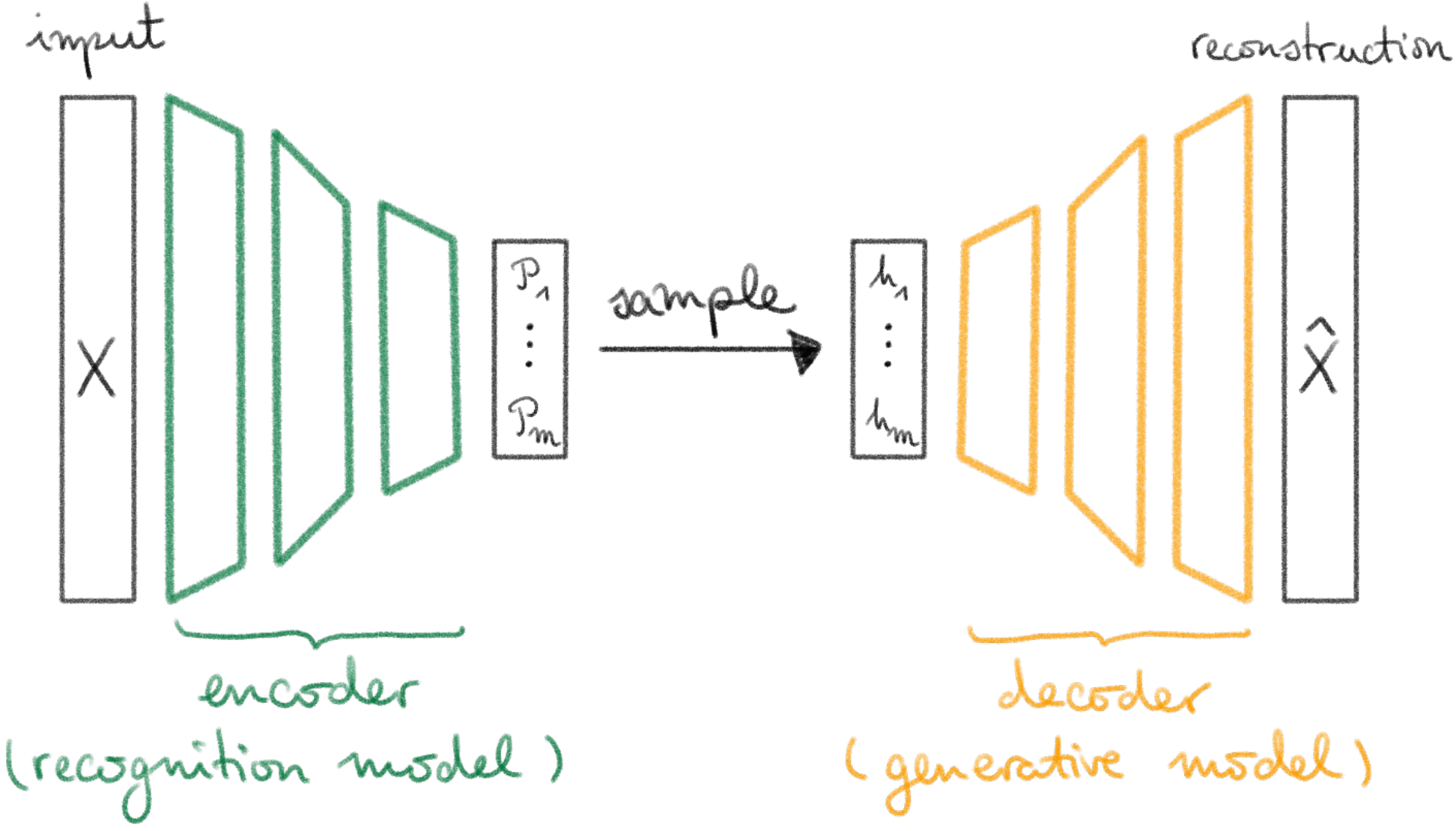


vs.



from Jeremy Jordan's blog

VAE Architecture



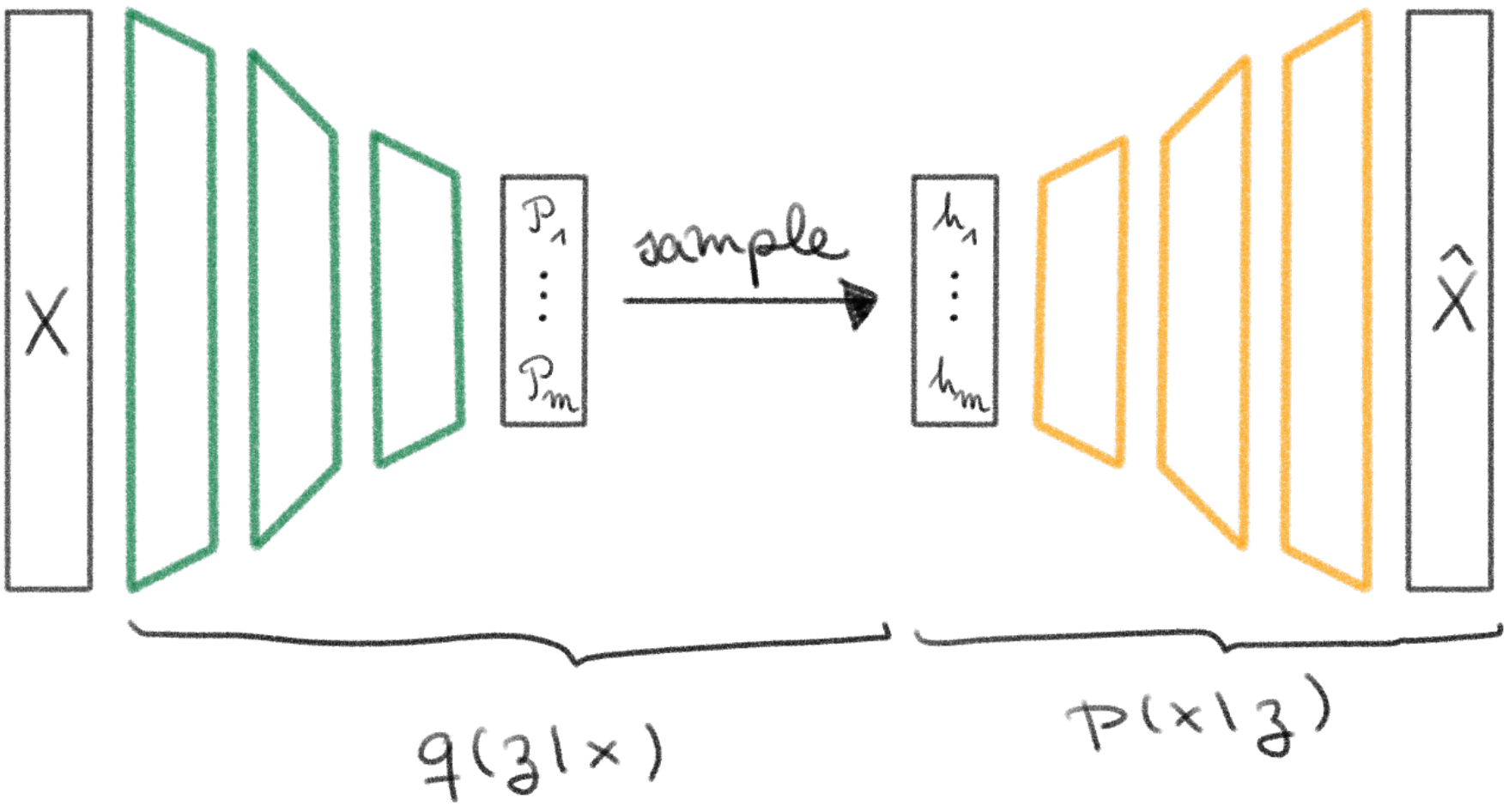
REMARKS

(i) We are enforcing a continuous, smooth latent representation.

(ii) If the decoder reconstructs well for any sampling, necessarily:

close in latent space $\Longrightarrow$ similar reconstructions

Statistical Motivation



Loss Function

$$\text{Loss} = L(x, \hat{x}) + \sum_{j=1}^{m} KL\left(q(z_j|x) \,\|\, p(z_j)\right)$$

$L(x, \hat{x})$: MSE or BCE

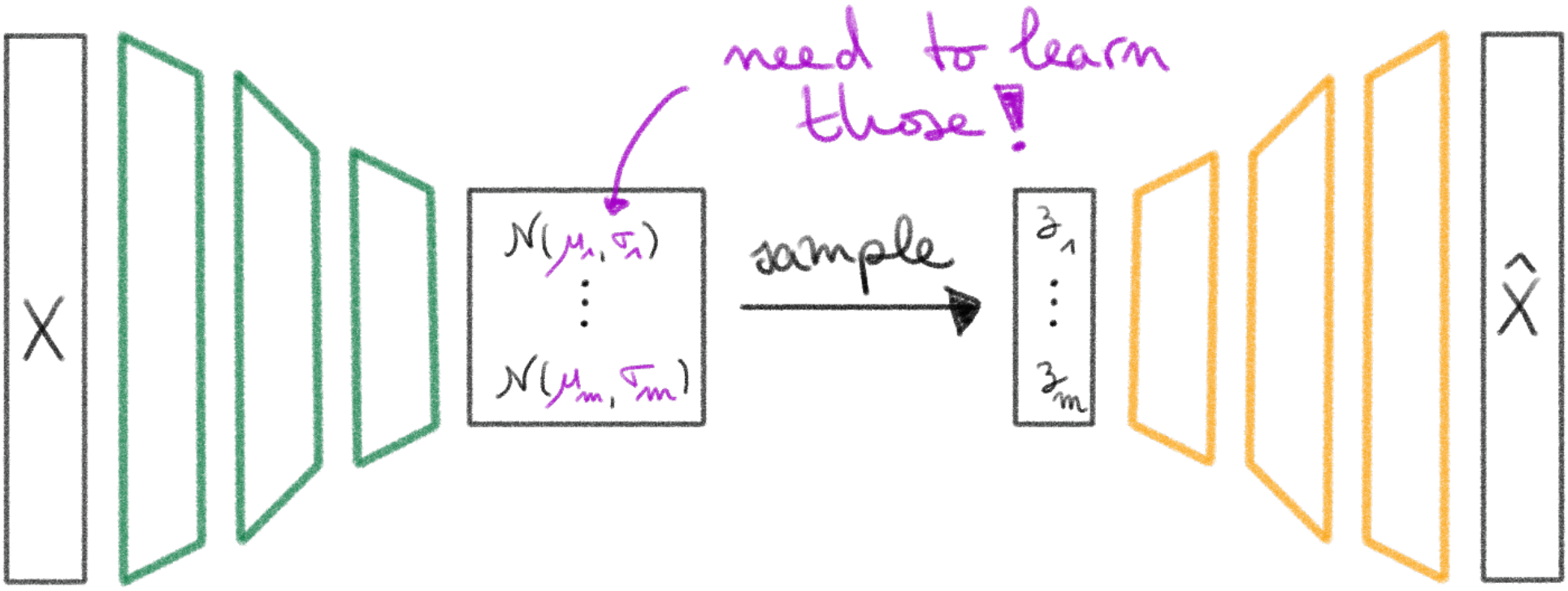
$p(z_j)$: true prior, usually assumed to be $N(0, 1)$

IMPLEMENTATION

We have to learn the mean vector and covariance matrix of the latent Gaussian distribution.

Simplifying Assumption .- The covariance matrix is diagonal, i.e. the latent features are uncorrelated.

Backpropagation



Since $z_1 = \mathcal{N}(\mu_1, \sigma_1)$ ...

... $\dfrac{\partial L(x, \hat{x})}{\partial \mu_1} = ?$

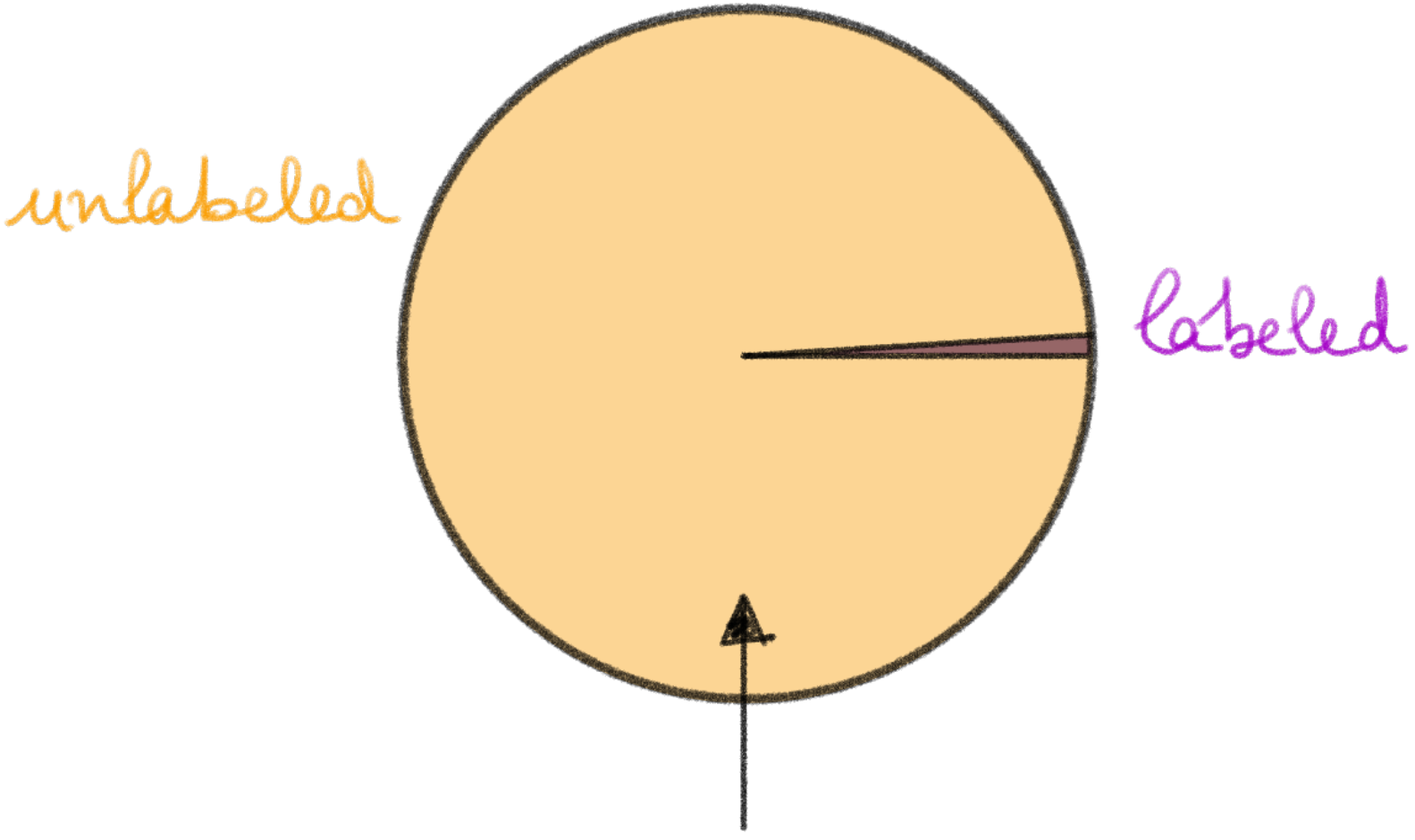
Reparametrization Trick

~~$z \sim \mathcal{N}(\mu, \sigma)$~~ instead $\longrightarrow$ $z = \mu + \sigma\varepsilon$, $\varepsilon \sim \mathcal{N}(0,1)$

$$\Rightarrow \begin{cases} \dfrac{\partial L}{\partial \mu} = \dfrac{\partial L}{\partial z} \cdot \dfrac{\partial z}{\partial \mu} \\ \dfrac{\partial L}{\partial \sigma} = \dfrac{\partial L}{\partial z} \cdot \dfrac{\partial z}{\partial \sigma} \end{cases}$$

We could learn negative σ!
Sol: learn $\log(\sigma)$ then take exp to get σ.

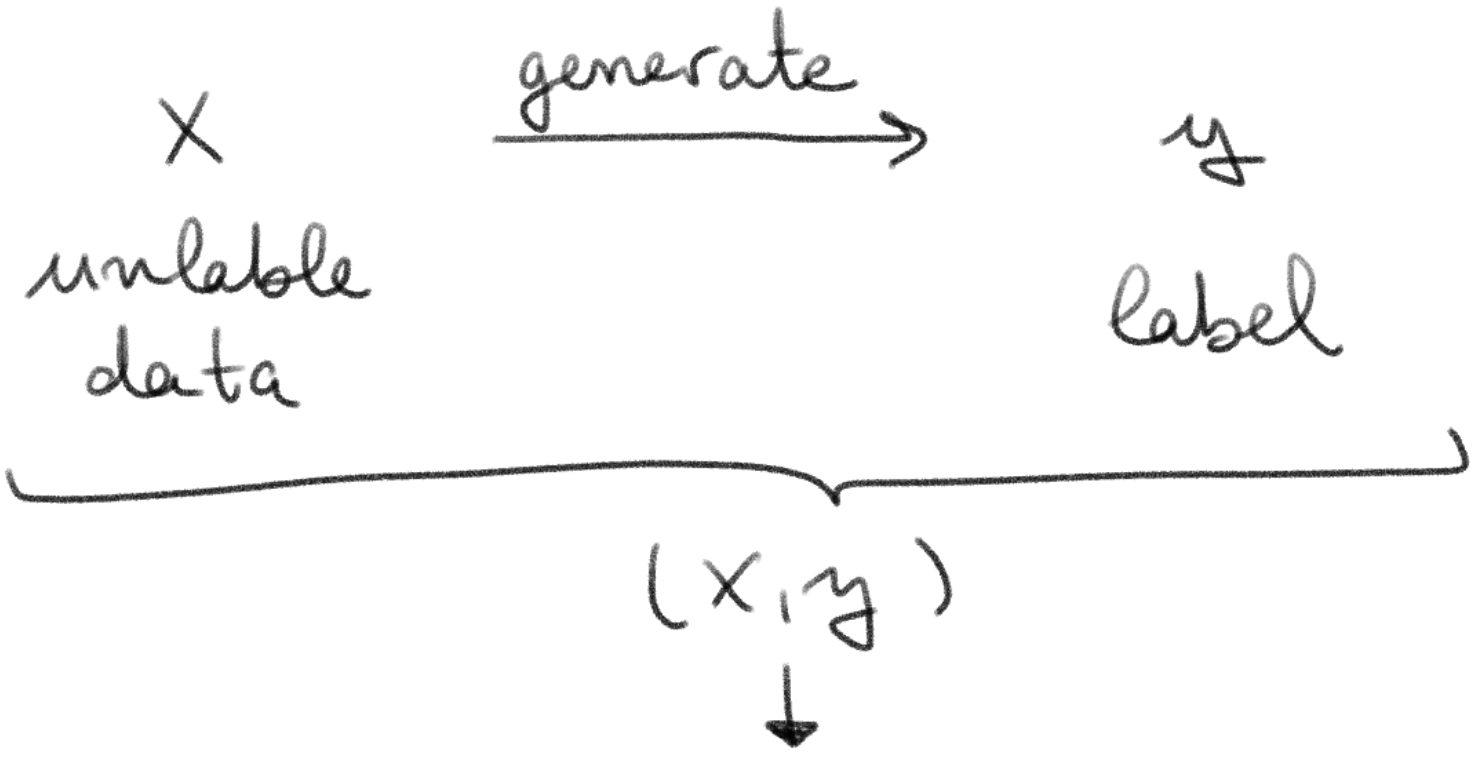
DATA



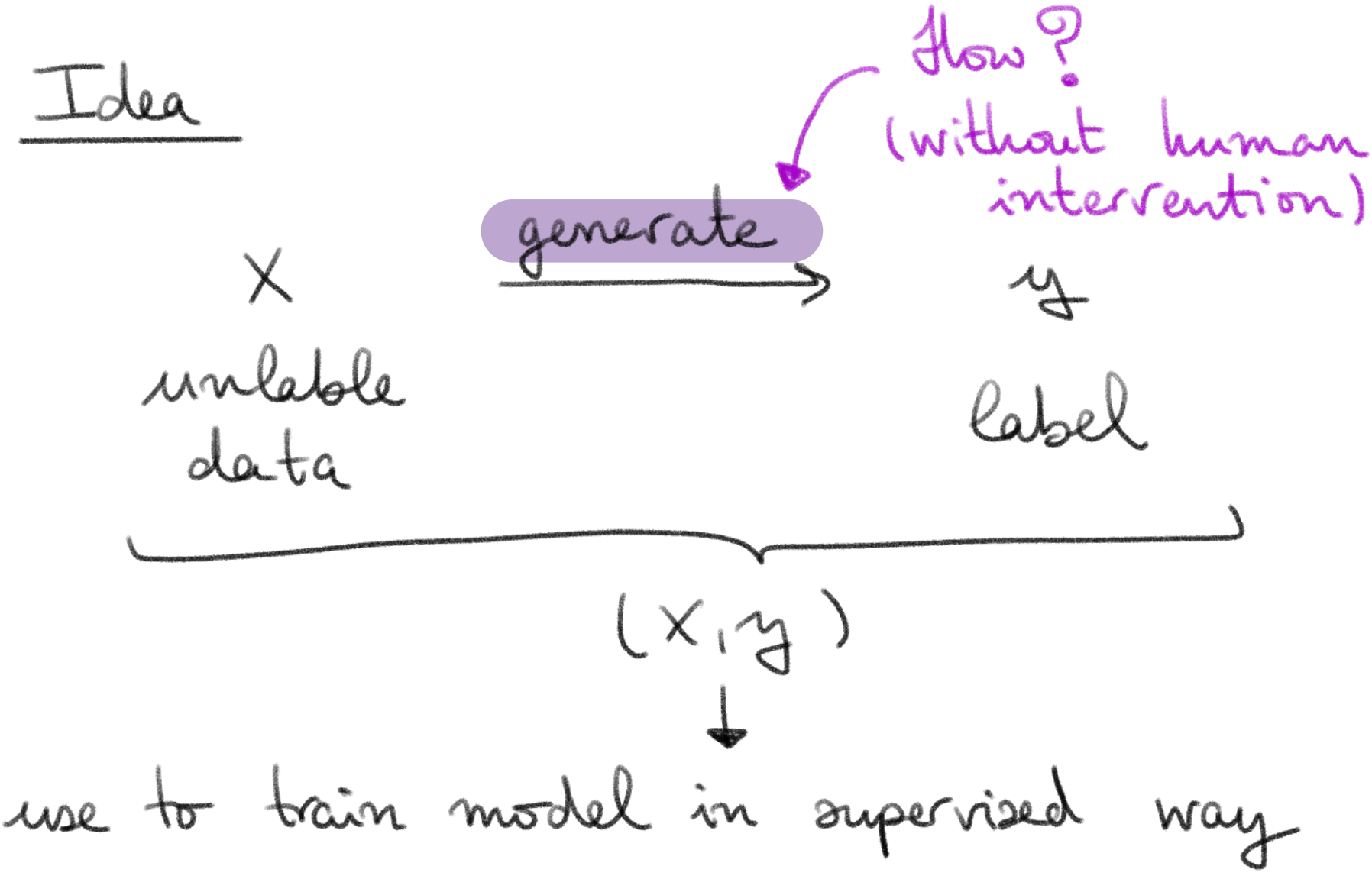
how do we leverage all this data?

Self-Supervised Learning

Idea



Self-Supervised Learning



Pretext vs Downstream Tasks

Pretext Task.- Designed so that NN learns visual features.

Downstream Task.- Has real world applications and human-labeled data.

EXAMPLES: DOWNSTREAM TASKS

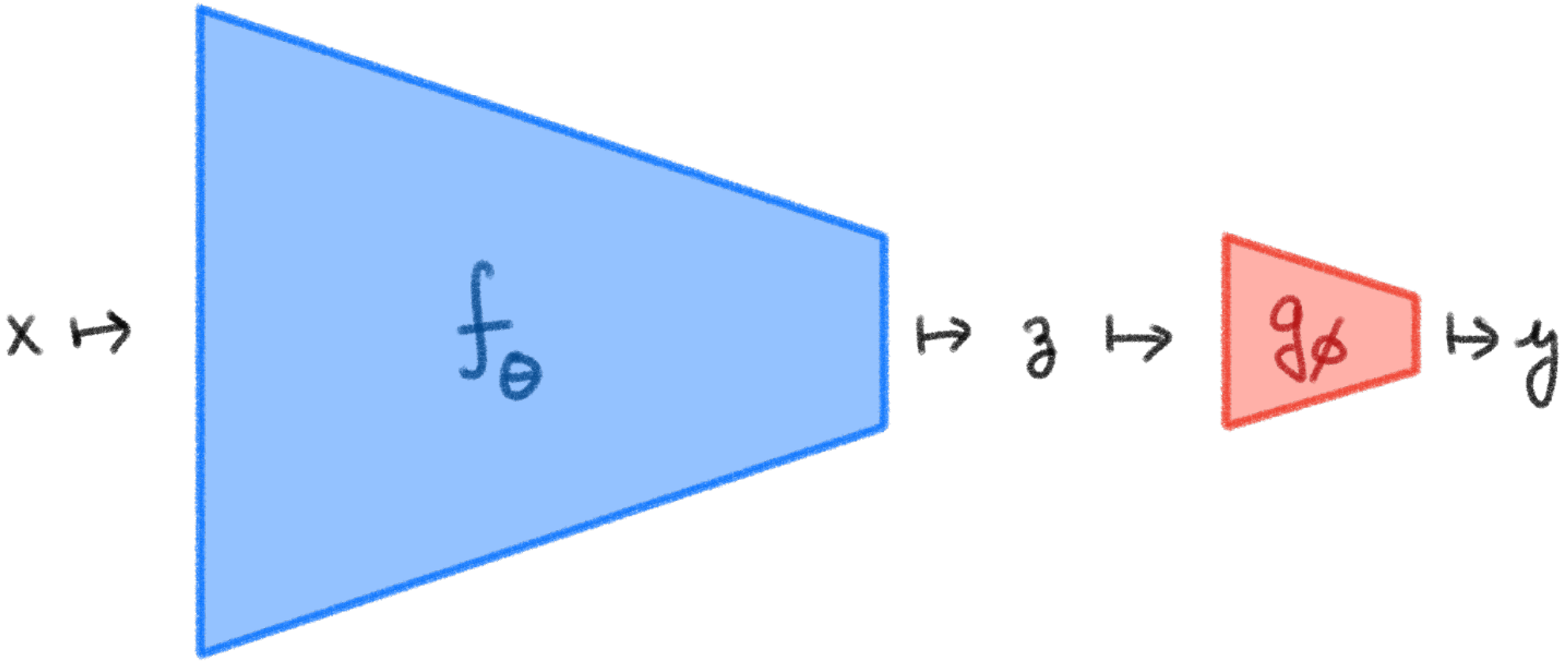
- Classification
- Regression
- Object Detection
- Segmentation
- Anomaly Detection
- ...

Examples: Pretext Tasks

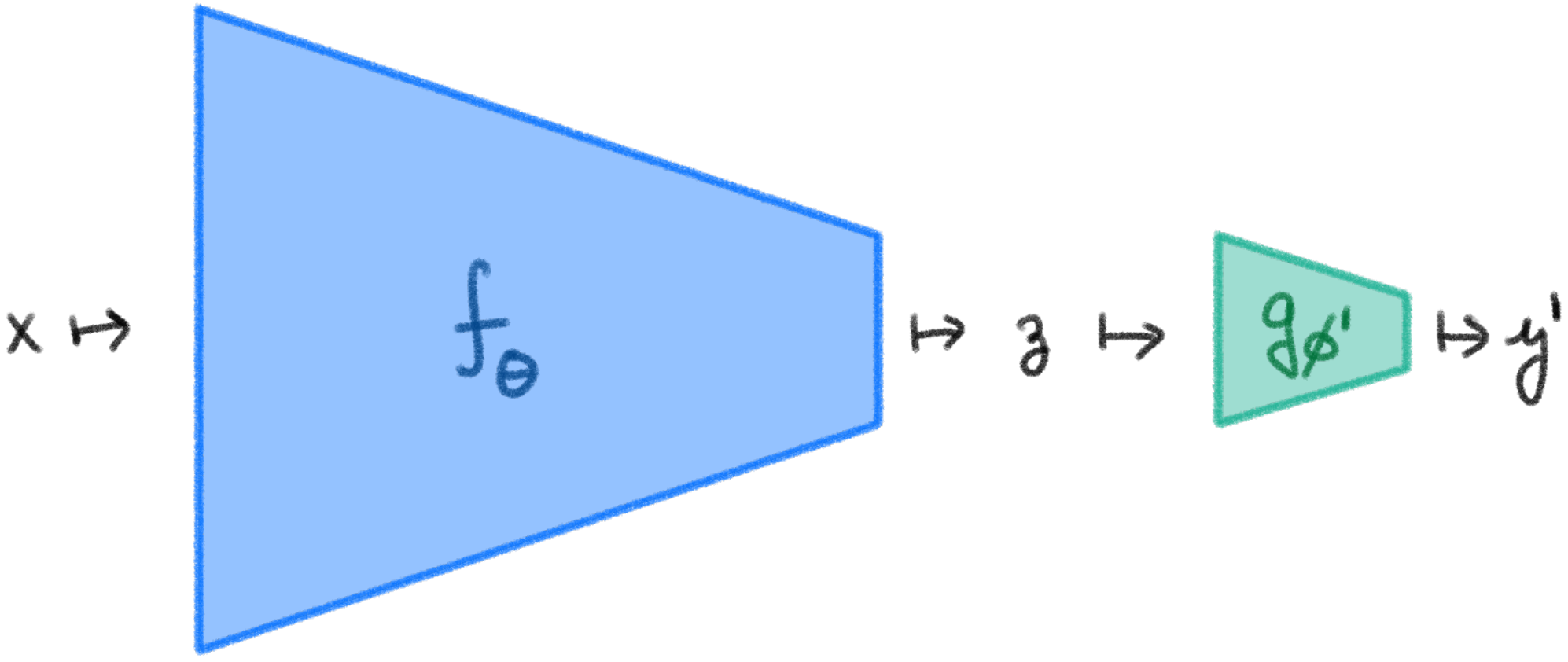
- Identifying augmentations
- Colorizing
- Masking
- Inpainting
- Contrastive Learning

How can a model trained on a (pretext) task be useful for a different (downstream) task?

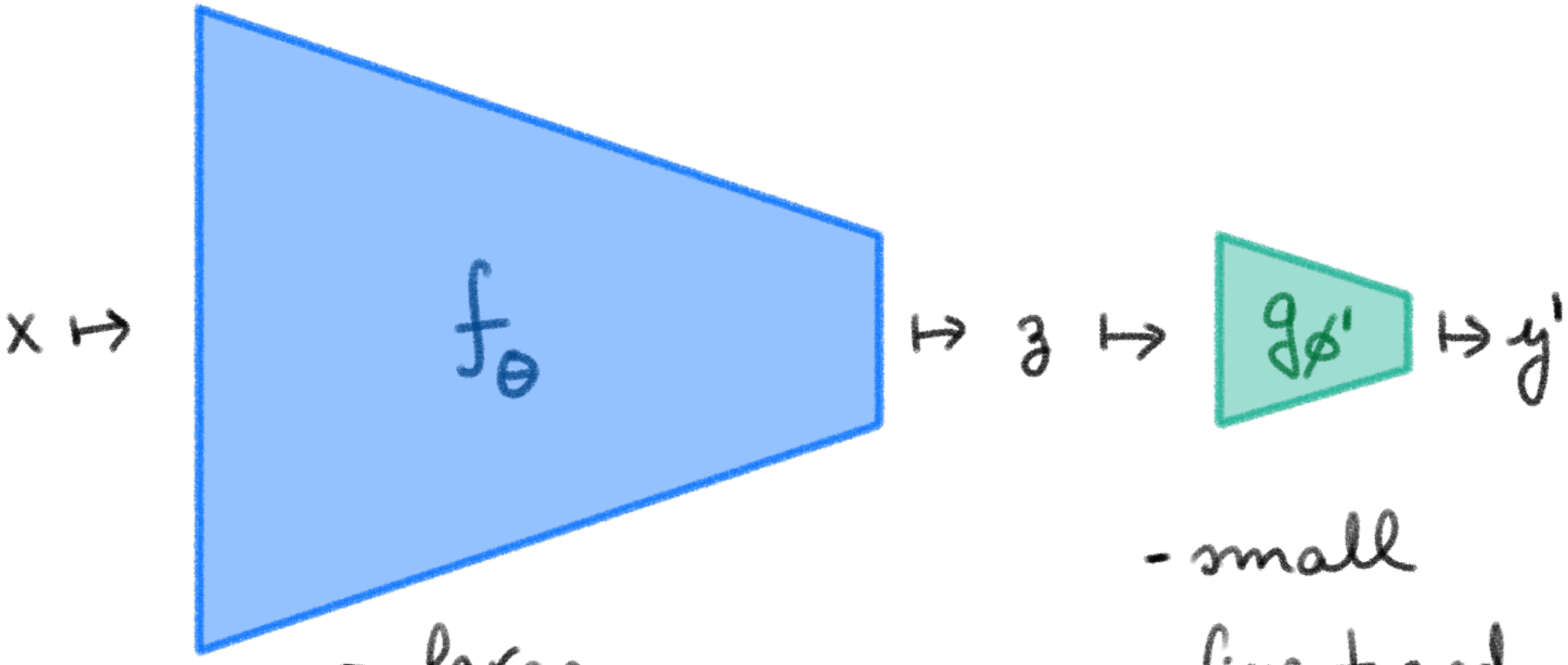
Backbone & Head



Backbone & Head



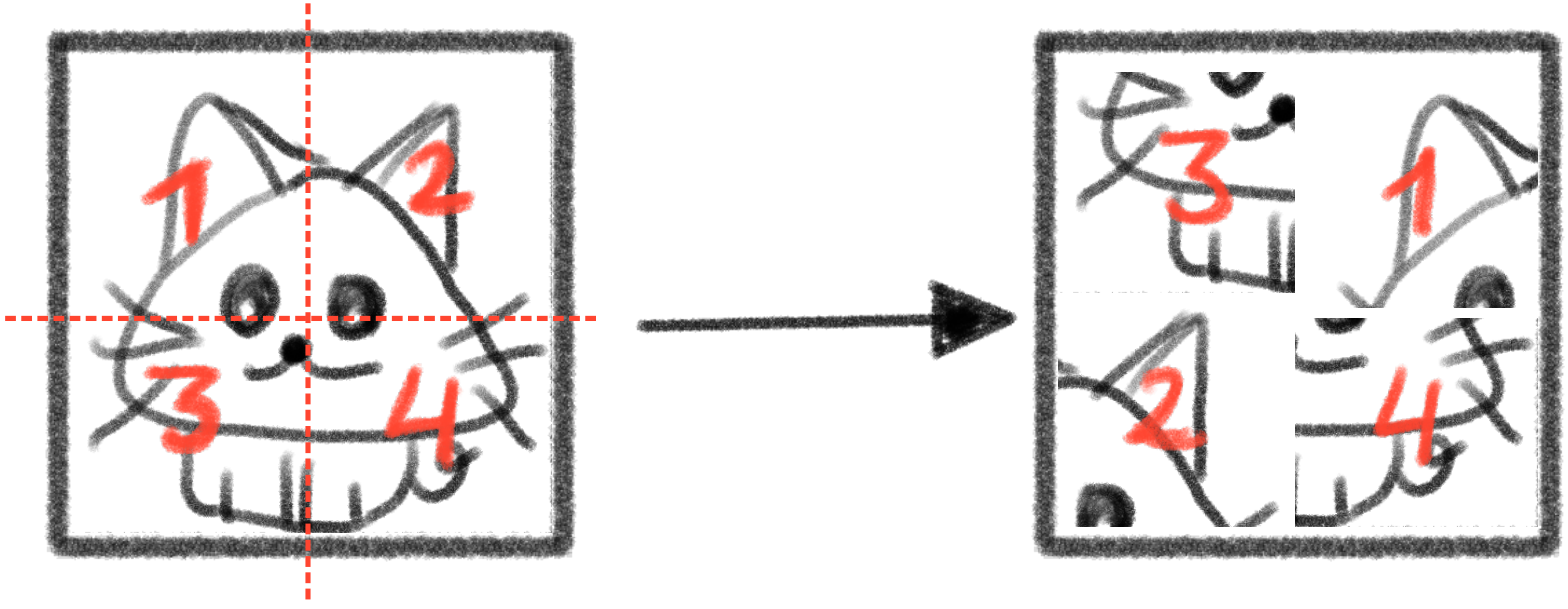
Backbone & Head



- large
- frozen

- small
- fine-tuned

Identifying Augmentations:
— Jigsaw Puzzle —

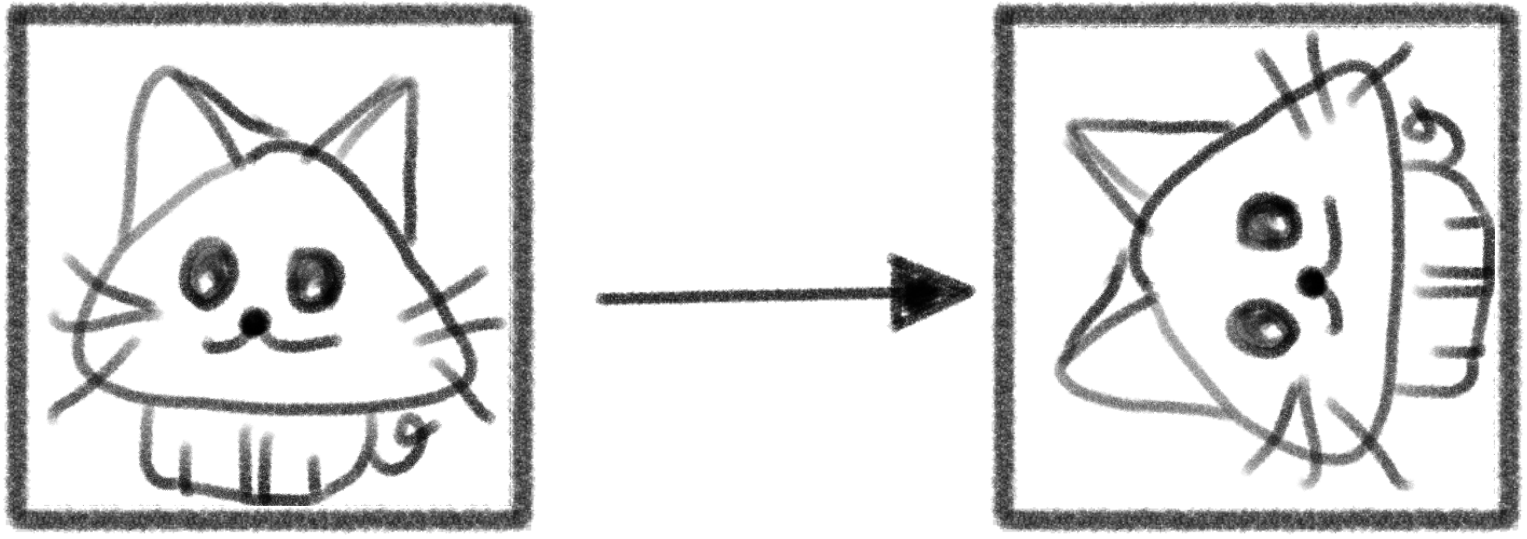


$$y = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 1 & 4 \end{pmatrix}$$

Loss?

IDENTIFYING AUGMENTATIONS:

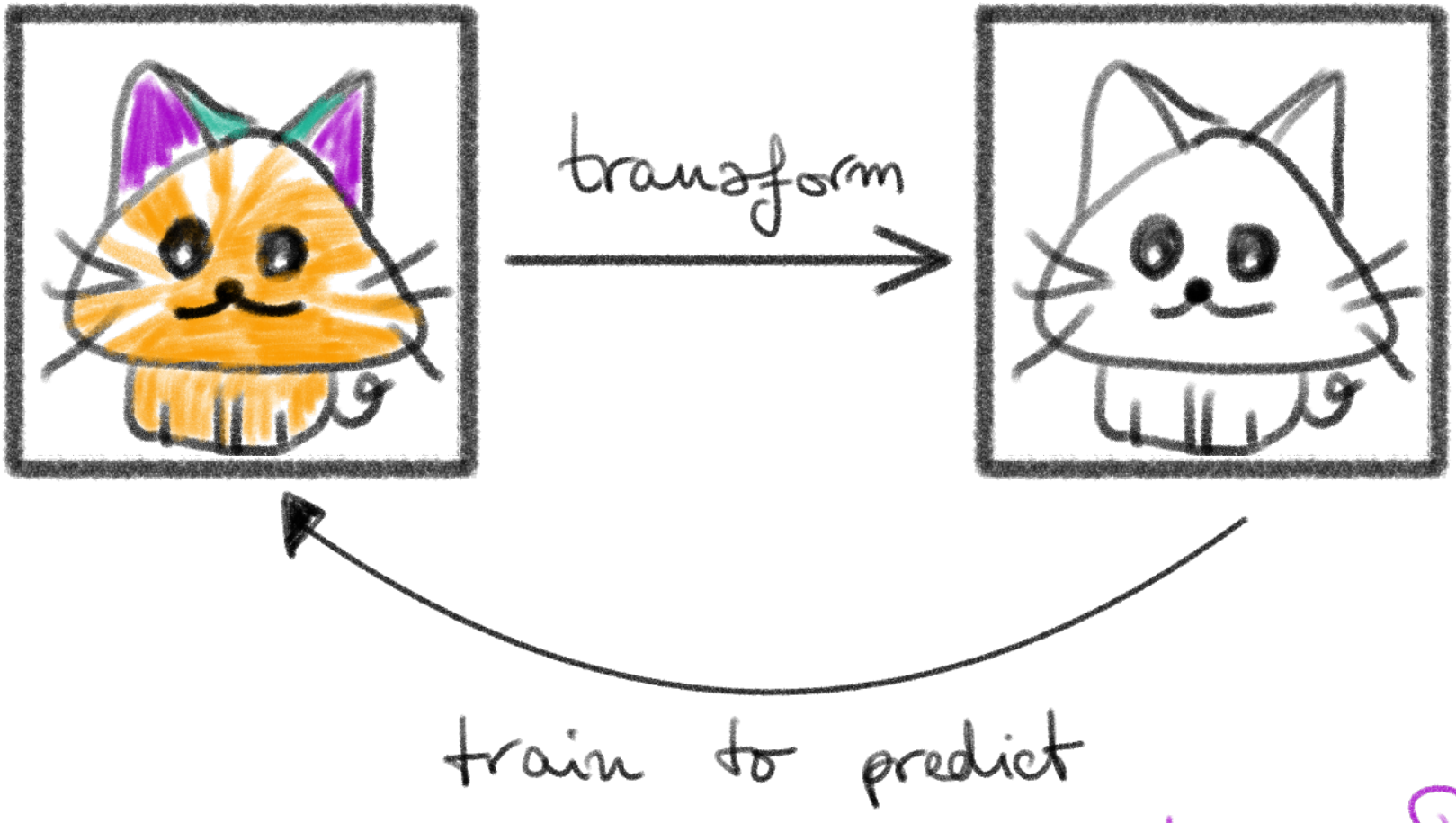
— ROTATIONS —



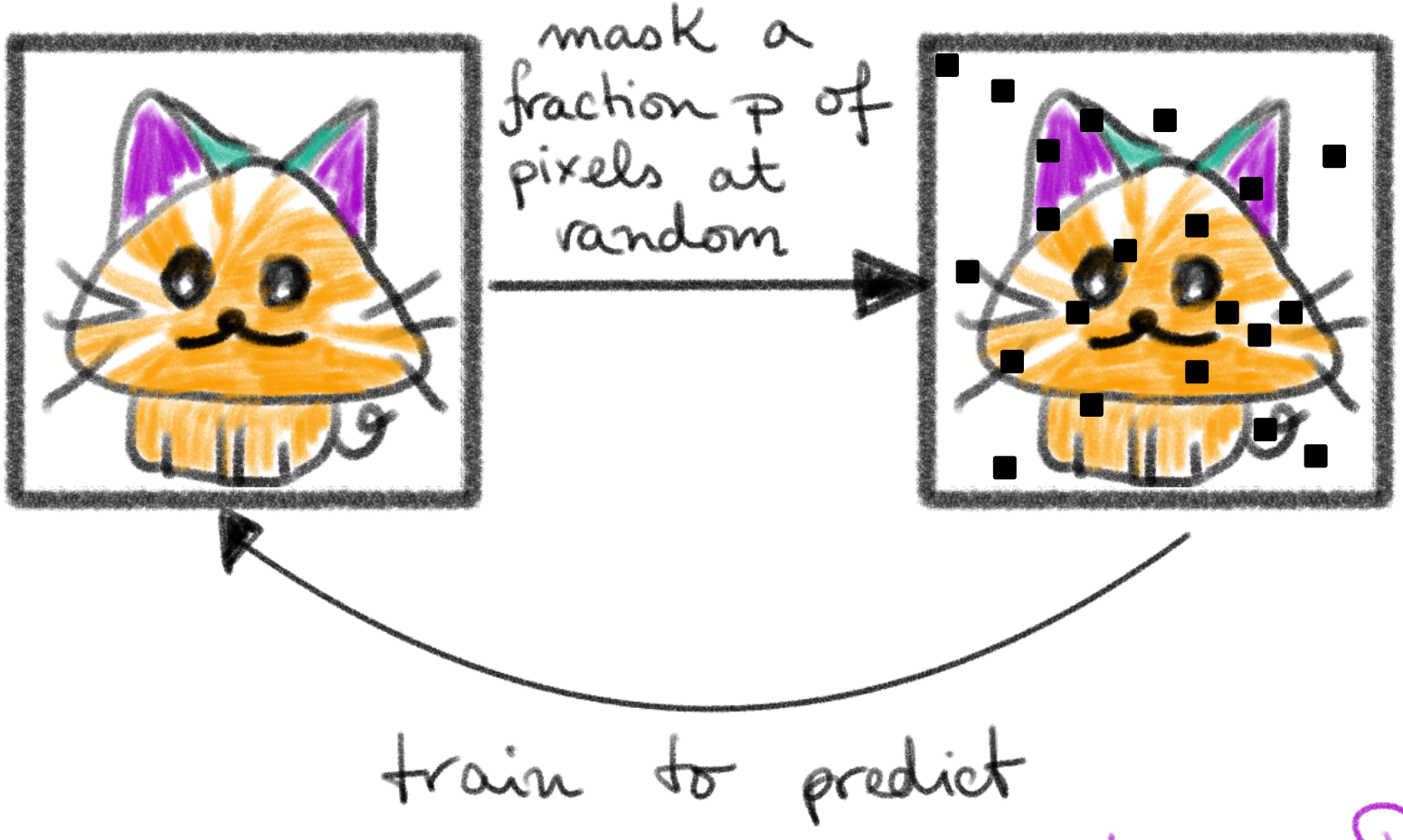
Loss?

$y = 90°$

Colorizing

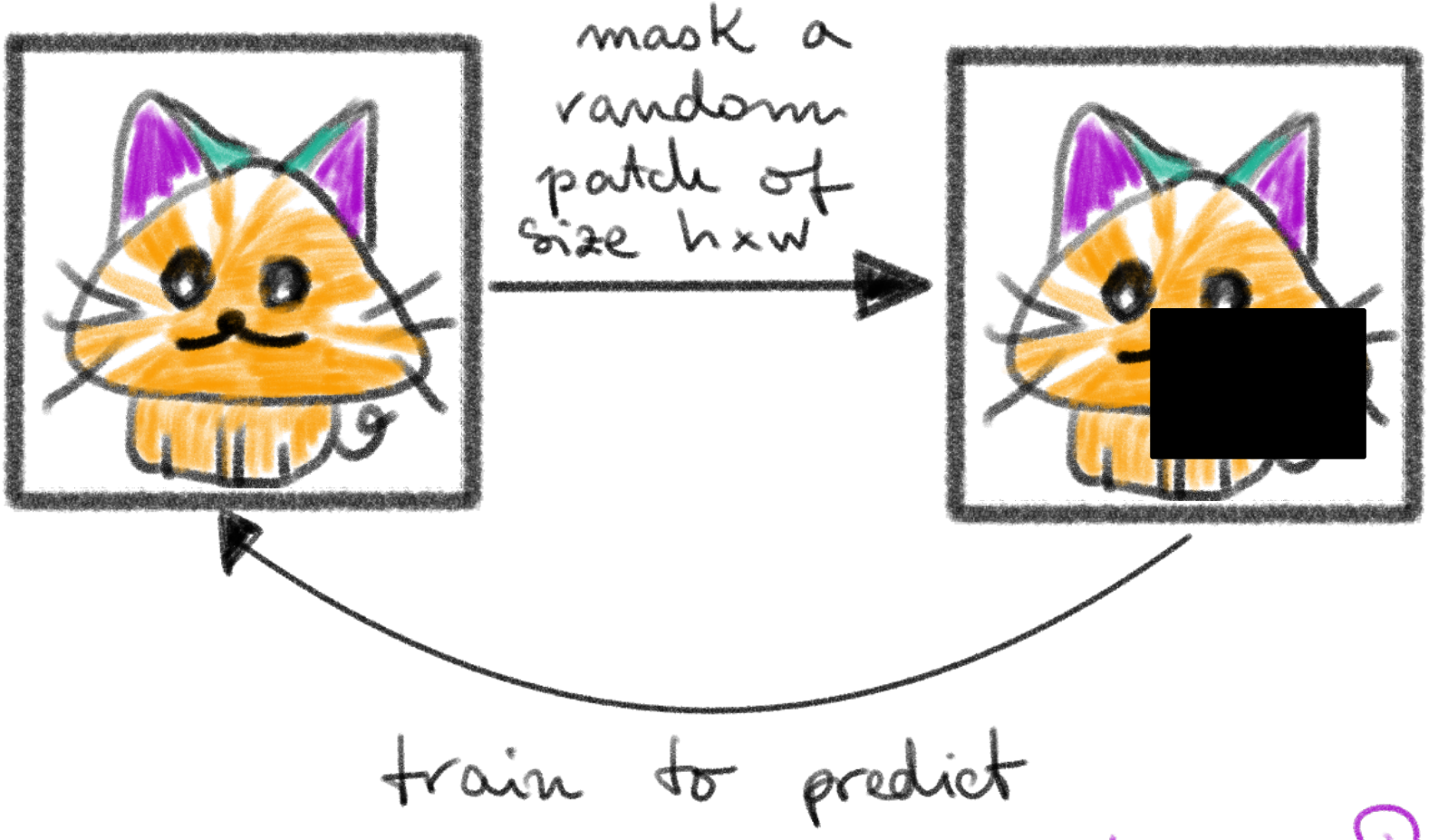


Masking



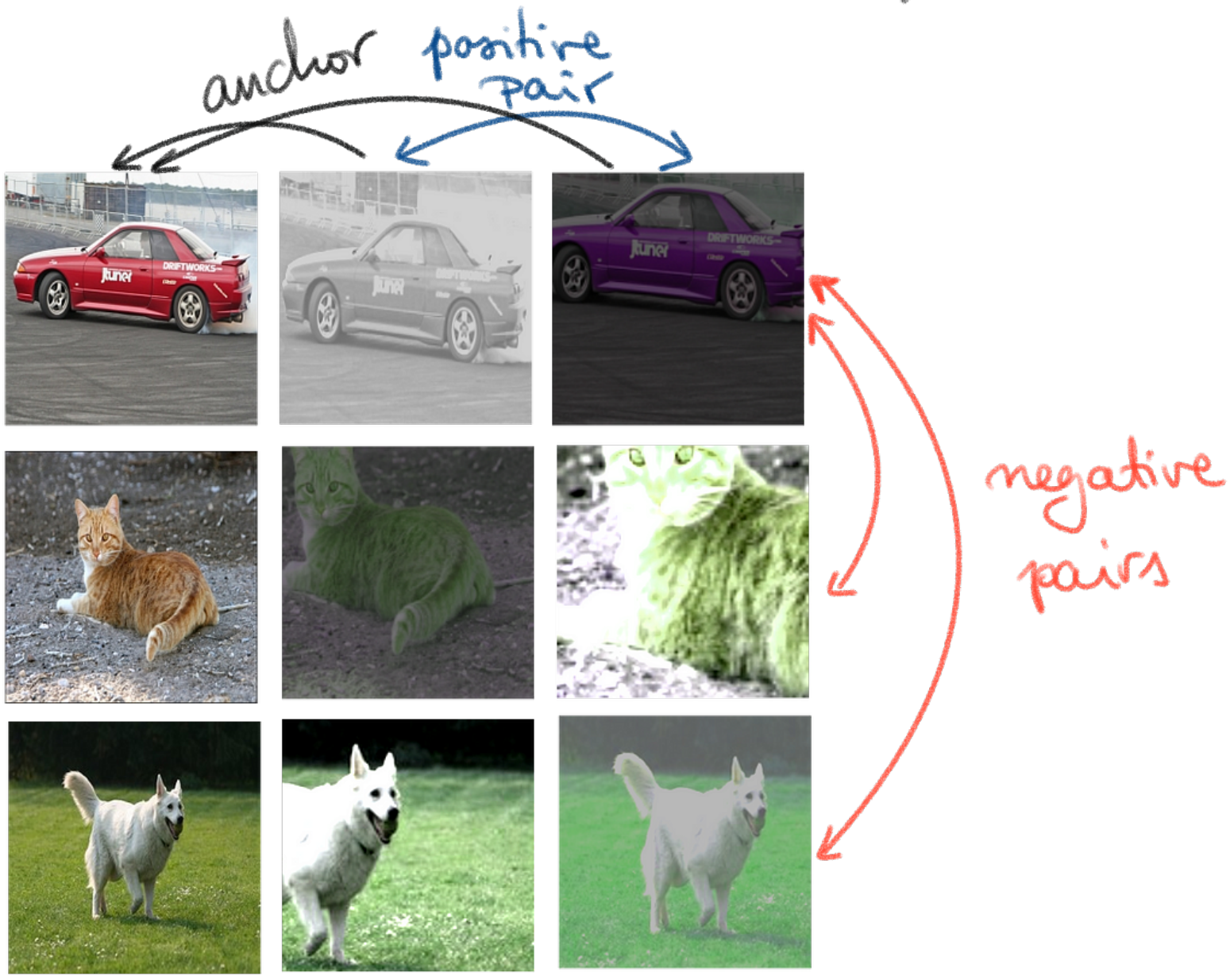
Loss?

INPAINTING



Loss?

CONTRASTIVE LEARNING



Contrastive Learning

Idea

Contrastive loss :

"pull together" positive pairs

"push apart" negative pairs

Contrastive Loss

$$\text{Loss}(x, x') = \mathbb{1}_{x \sim x'} \left\| E(x) - E(x') \right\|_2^2 + \mathbb{1}_{x \not\sim x'} \max\left(0,\ m - \left\| E(x) - E(x') \right\|_2\right)^2$$

positive pair: $x \sim x'$

negative pair: $x \not\sim x'$

margin: m

Contrastive Loss

$$\text{Loss}(x, x') = \mathbb{1}_{x \sim x'} \left\| E(x) - E(x') \right\|_2^2 + \mathbb{1}_{x \not\sim x'} \max\left(0,\ m - \left\| E(x) - E(x') \right\|_2\right)^2$$

positive pair: $x \sim x'$

negative pair: $x \not\sim x'$

m: margin

$$\text{Loss} \simeq 0 \Rightarrow$$

if $x \sim x' \Rightarrow E(x) \simeq E(x')$

if $x \not\sim x' \Rightarrow \left\| E(x) - E(x') \right\|_2 \geq m$

TRIPLET LOSS

anchor

negative input

$$Loss = \max(0, d(A, P) - d(A, N) + \alpha)$$

similarity metric: euclidean / cosine / ...

positive input

margin

Triplet Loss

$$\text{Loss} = \max\left(0,\ d(A,P) - d(A,N) + \alpha\right)$$

anchor

negative input

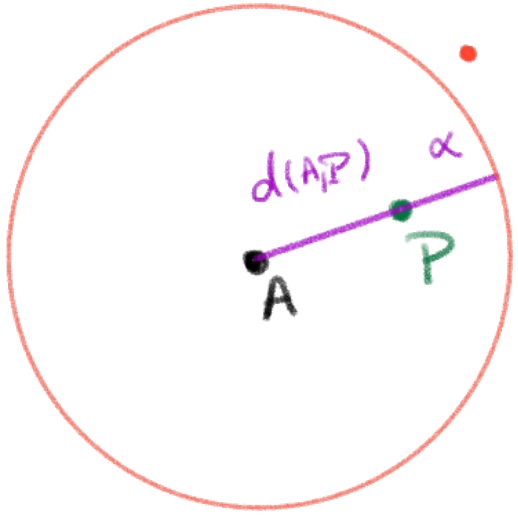
similarity metric: euclidean / cosine / ...

positive input

margin

$$\text{Loss} = 0 \Rightarrow$$

$$d(A,P) + \alpha \leq d(A,N)$$



Kahoot

Time!

Is SSL REALLY USEFUL ?

- Really useful for text data
- Really useful for multi-modal data
- For Computer Vision, struggles to beat purely supervised learning in most downstream tasks (e.g. classification) *
- Really useful for Anomaly Detection

JEPA (JOINT EMBEDDING PREDICTIVE ARCHITECHTURE)

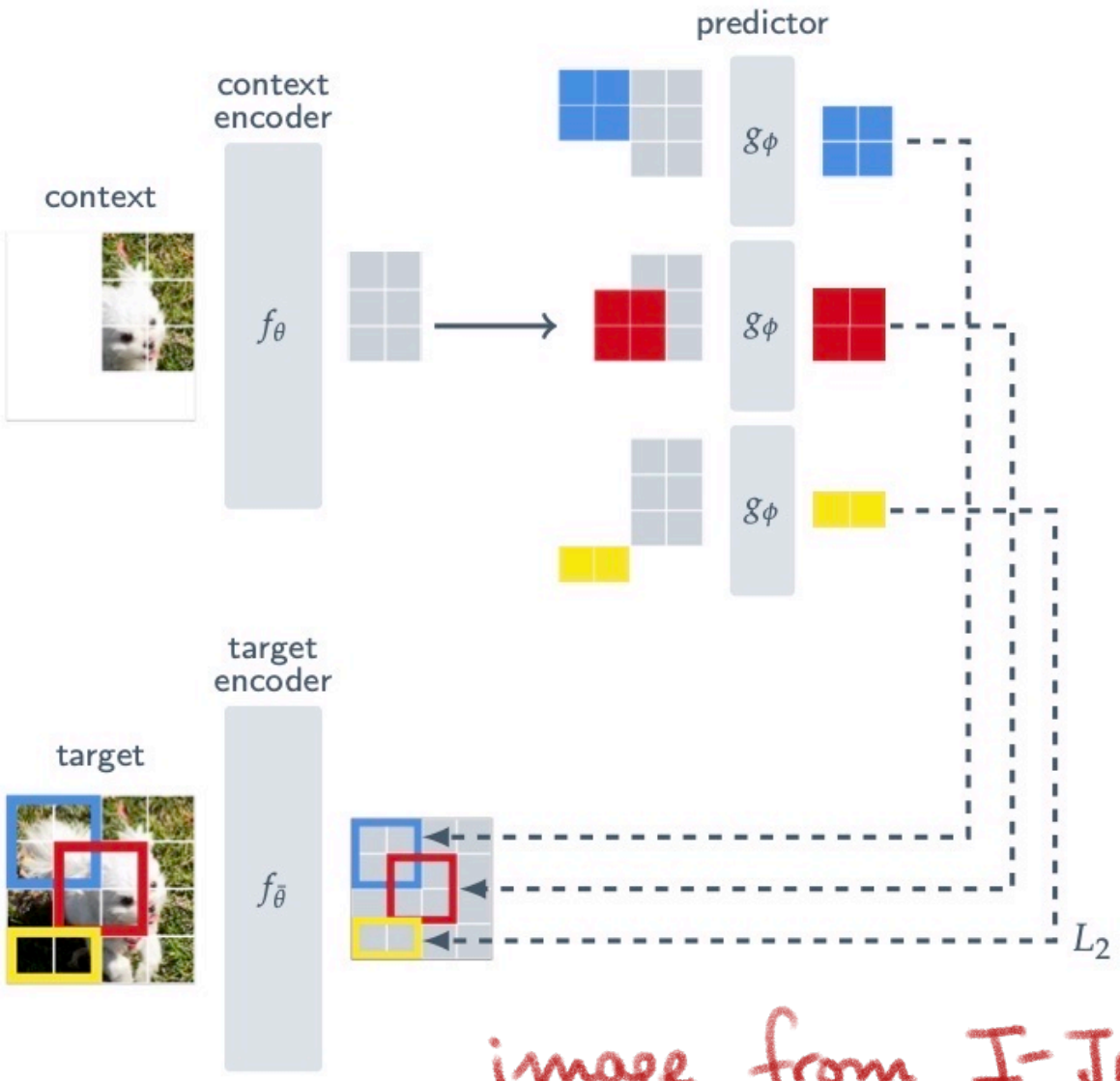


image from I-JEPA paper

DINO-v3

- Very good at Dense Prediction Tasks
- Good at semantic tasks
- Universal, frozen backbone

  ⇒ robust accross domains

Promising for medical imaging, satellite imaging ...