BLG 561 E FALL 2021

Deep Learning

30.11.2021

Gözde ÜNAL

Supervised Learning: We've done this so far

$$\{x^i, y^i\}_{i=1}^{m}$$

Input Data

Labels

$$x^i \longrightarrow \boxed{f_\theta} \longrightarrow \hat{y}_i$$

Deep NN

We require lots of LABELED DATA

generate → LABELED DATA

Classification, Regression, Obj. Detection, Segmentation, Tracking.

Learn to map the Input ⟶ Label prediction

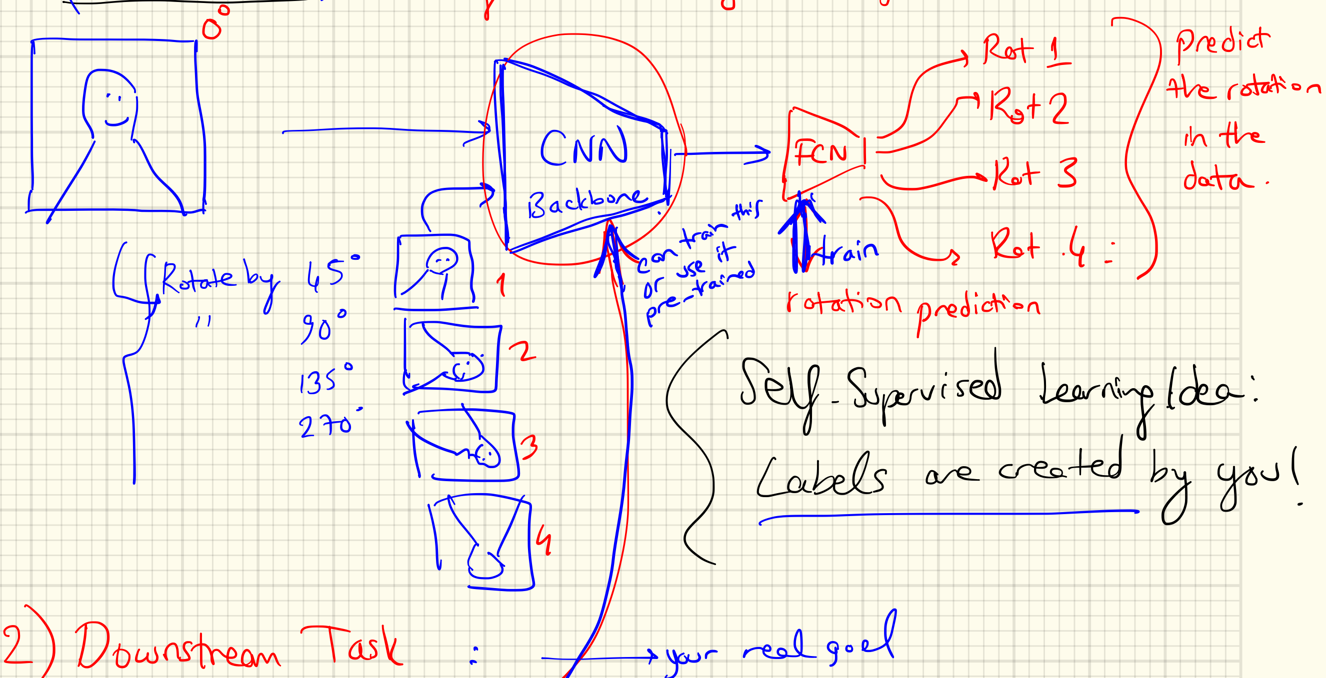
Self-Supervised Learning:

Let the data generate labels

1) Pretext Task ⇐

2) Downstream Task ⟵ final goal :

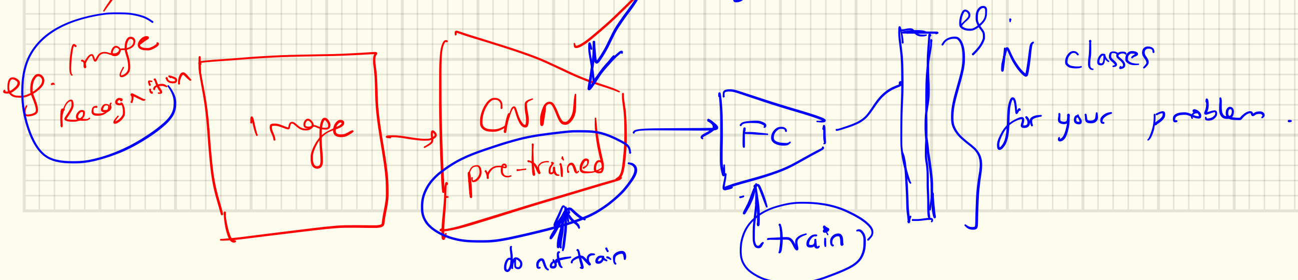
1) Pretext Task: Predicting Rotation of the Image



Self-Supervised Learning Idea:

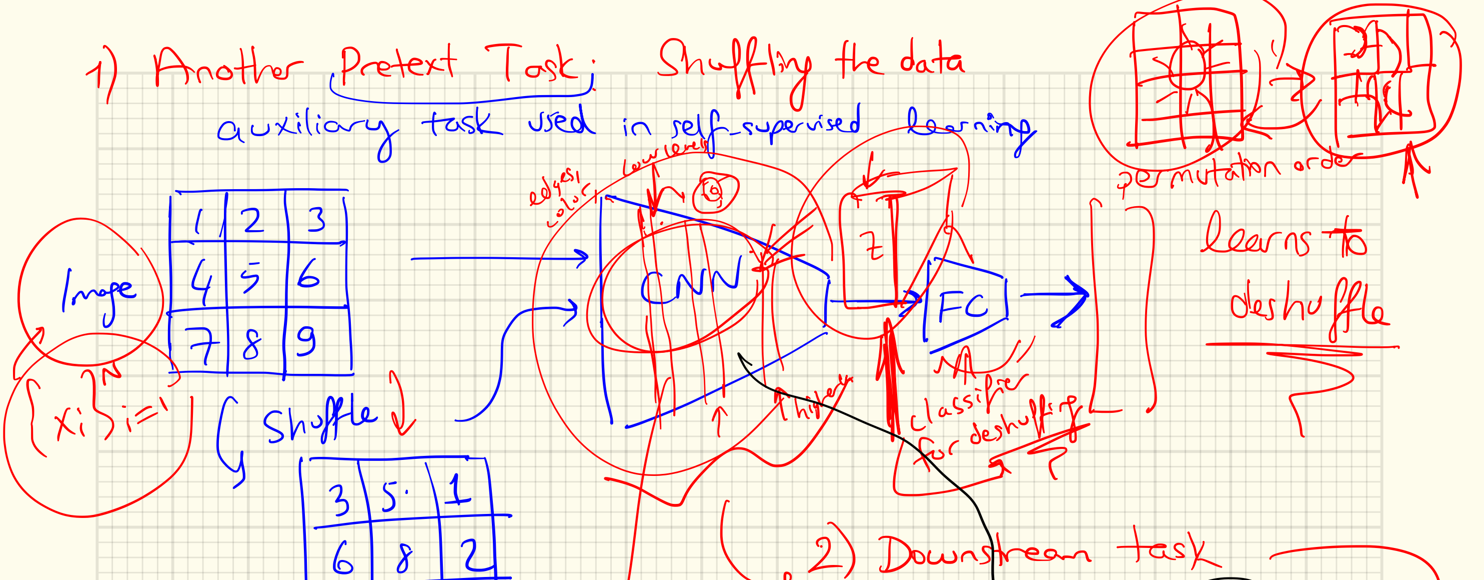
Labels are created by you!

2) Downstream Task : → your real goal

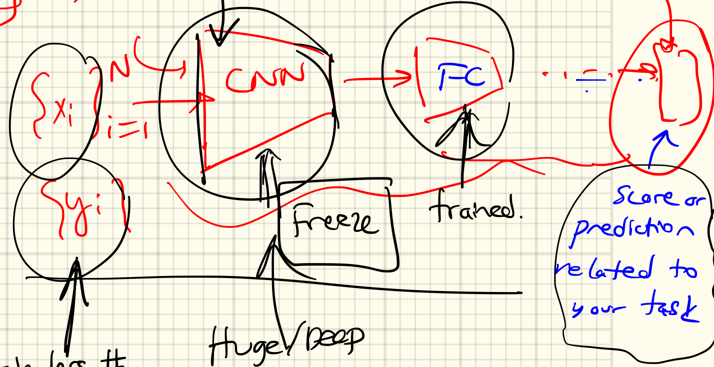


1) Another Pretext Task: Shuffling the data

auxiliary task used in self-supervised learning



2) Downstream task



much less # of labeled data.

Huge/Deep CNN is now pre-trained

Idea: Now, this CNN captures the input data characteristics better w/ self-supervision.

→ For a certain downstream task, I still need some labeled data, but not as much as before.

NLP (GPT) : Sentence : "I am going to school".

Self-supervision pretext task : Shuffle the words in the sentence

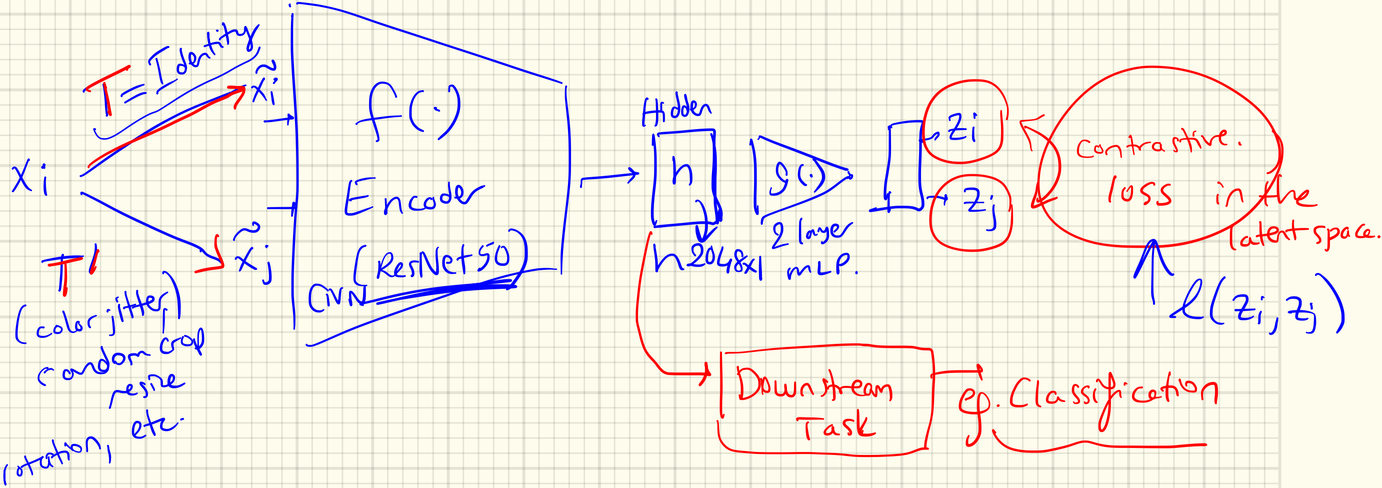
pretext task : 1.1) NN predicts the correct order of the sentence.

1.2) Delete a word from the sentence & predict the missing word(s).

Open Directions : Designing appropriate pretext tasks for your own problem.

- Incorporating self-supervision idea into your research work in a novel way.

→ Contrastive Learning (SimCLR paper Hinton et al, 2020.)

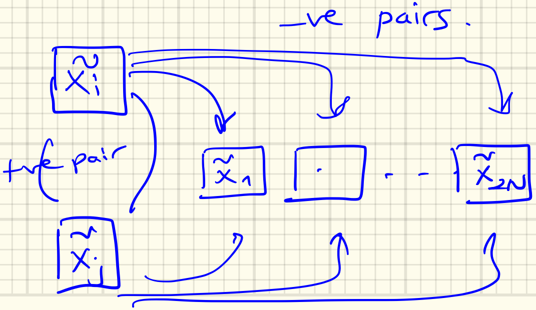


$\{\tilde{x}_i, \tilde{x}_j\}$: +ve pair

N Batchsize : 2N-1 -ves

2N pairs

$\{\tilde{x}_i, \tilde{x}_k\}$



$$\underbrace{\min}\ \text{loss}(\underset{\underset{\text{anchor}}{\uparrow}}{\underline{z_i}}, \underline{z_j}) = -\text{sim}(\underset{\uparrow}{\underline{z_i}}, \underset{\underset{\text{true pair}}{\uparrow}}{z_j})/\underset{\underset{\text{temperature}}{\uparrow}}{\tau} + \log \sum_{\substack{k=1\\k\neq i}}^{N} \exp(\text{sim}(z_i, \underset{\uparrow}{z_k})/\tau)$$

$$\begin{aligned}
&\log(\exp(\text{sim}(z_i, \overset{\swarrow}{z_1}))/\tau \\
&\qquad + \exp(\text{sim}(z_i, \overset{\swarrow}{z_2}))/\tau \\
&\qquad \vdots \\
&\qquad + \exp(\text{sim}(z_i, \overset{\swarrow}{z_N}))/\tau)
\end{aligned}$$

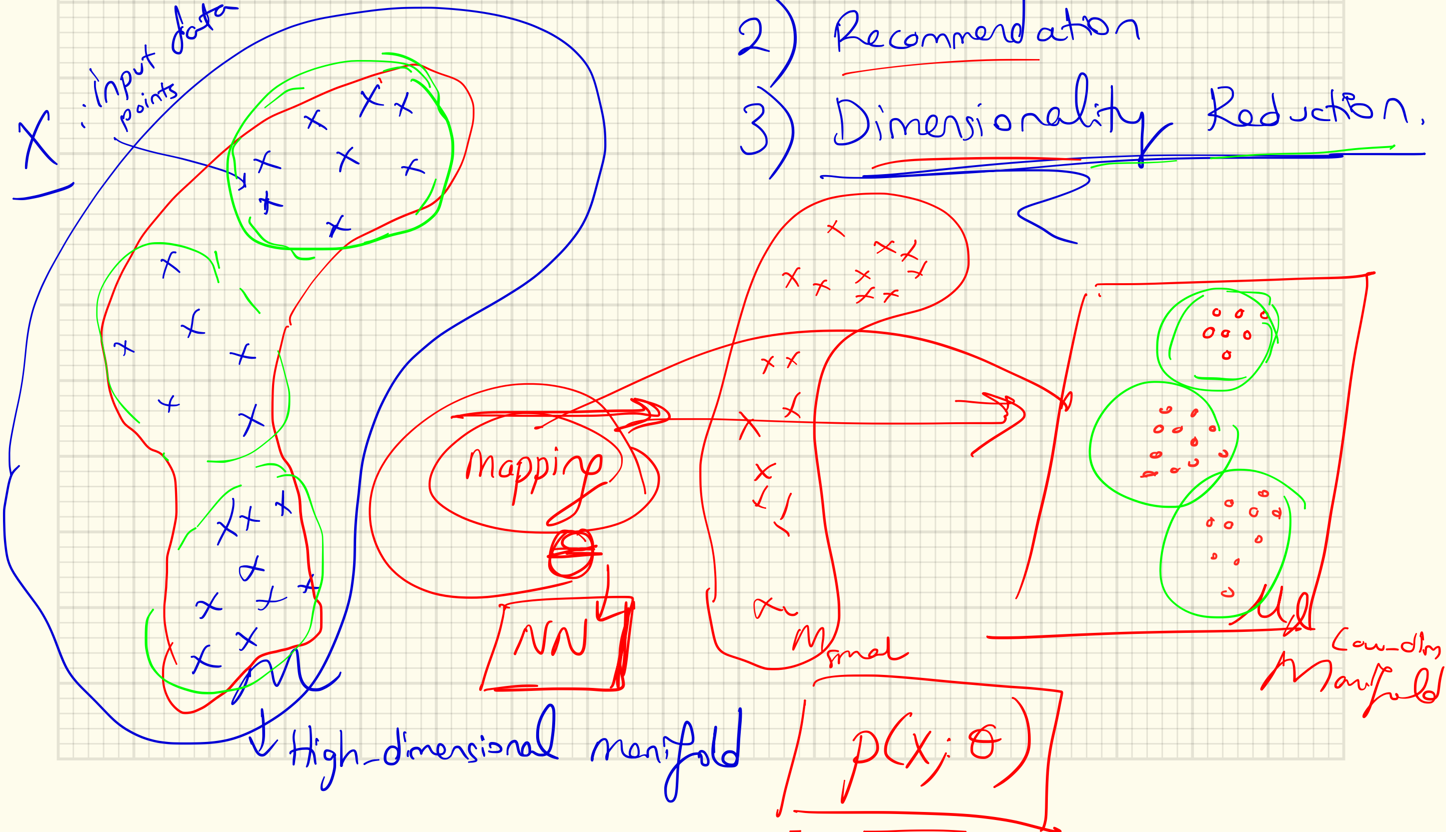
Unsupervised Learning : No Labels !!!

↕ Representation Learning

1) Clustering ⇒ Unsupervised Segmentation.

2) Recommendation

3) Dimensionality Reduction.



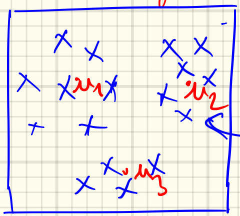
Goal : Learn the distrib. of the input data $p(x; \theta)$ from training data $\boxed{\{x_i\}_{i=1}^{m}}$ w/ NO LABELS.

Unsupervised Learning Tasks :

1) Clustering. 2) Dimensionality Reduction.

[1] K-Means Clustering : Hypothesis function.

Reformulate K-means in an ML framework



θ : cluster means : $\mu_1, \ldots, \mu_K$

$=$

$\{x_i\}_{i=1}^{m}$: Dataset w/ no Labels

① Hypothesis Class : $\mu_{\underset{j}{\arg\min}} \|x^i - \mu_j\|_2^2 = h_\theta(x^i)$

→

② Loss function: $\mathcal{L}(x^i, h_\theta(x^i)) =$

$$\mathcal{L} = \sum_{i=1}^{m} \left\| x^i - \mu_{\left(\arg\min_j \|x^i - \mu_j\|_2^2\right)} \right\|_2^2$$

③ Optimization: Q. Is the loss $\mathcal{L}$ a convex loss fn? No!

$$\arg\min_{\underline{\underline{\theta}} = \{\mu_1, \dots, \mu_K\}} \mathcal{L}(x, h_\theta(x))$$ → non-convex optim. problem.

We solve it approximately by an iterative approach.

Alternate between (i) & (ii):

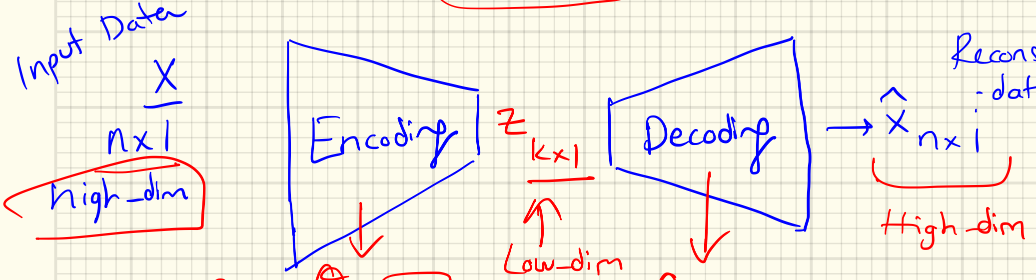
(i) Find optimal μ_i for each data instance $i = 1, \dots K$.

(ii) Then we update μ_i to be average of all samples assigned to μ_i

until "convergence".

$\boxed{2}$ PCA (Dimensionality Reduction)

Given $\{x_i\}_{i=1}^{m}$, $x_i \in \mathbb{R}^n$, $n \uparrow\uparrow$.



$\theta = \theta_2\ \boxed{k \times n}\quad \theta_1\ \boxed{n \times k}$

Idea: Reconstr the data so that "most of the informa tion in x is preserved.

① Hypothesis Class: $\hat{x} = h_\theta(x) = \theta_1\ \theta_2\ x$ ✓

② (Reconstruction) Loss: $\mathcal{L}_r(\hat{x}, x) = \| x - h_\theta(x) \|_2^2$

★ most widely used Unsupervised Learning loss

③ Minimize the Reconstruction Loss: $\theta = [\theta_1\ \theta_2]$

$$\min_{\theta_1, \theta_2} \sum_{i=1}^{m} \| x^i - \underbrace{\theta_1 \theta_2 x^i}_{\hat{x}^i} \|_2^2 :$$

$\hat{x}^i$: reconstructed data

Q: Is this a convex problem? →

→ Not a convex problem: but ∃ a solution through EVD
(Eigen Value Decomposition)

→ Given $\{\underline{x}_{i\,n\times 1}\}_{i=1}^{m}$: $\underline{\underline{X}} = \begin{bmatrix} | & | & & | \\ x^1 & x^2 & \cdots & x^n \\ | & | & & | \end{bmatrix}_{n\times m}$

Q. How do you perform EVD (↓ analysis) on $\cancel{\underline{\underline{X}}}$?

Calculate COVARIANCE matrix of the input (training) data matrix.

$$\underline{\underline{C}} = E\Big[\underbrace{\underline{\underline{\tilde{X}}}_{n\times m}}\ \underbrace{\underline{\underline{\tilde{X}}}^{T}_{m\times n}}\Big]$$

⇐ of course, there's centering of the data.

perform EVD on $\underline{\underline{C}} = \underbrace{\underline{\underline{U}}}_{\text{matrix of evectors : orthogonal matrix}}\ \underline{\underline{\Sigma}}\ \underline{\underline{U}}^{T}$

$$\underline{\underline{\Sigma}} = \begin{bmatrix} \overset{\max}{d_1} & & & 0 \\ & d_2 & & \\ & & \ddots & \\ 0 & & & \overset{\min}{d_n} \end{bmatrix}_{n\times n}$$

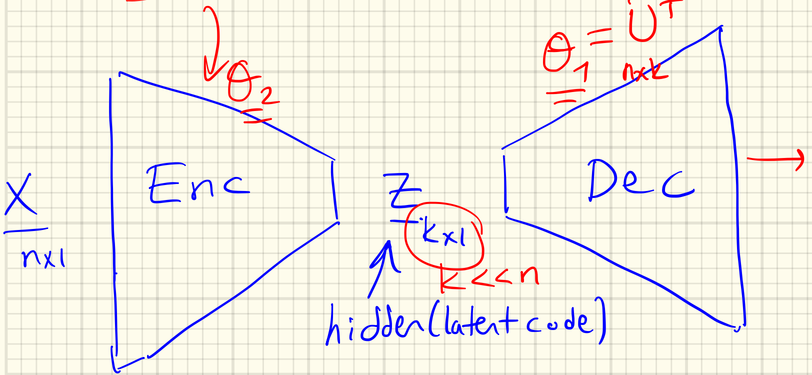
e.value

Set $\underline{\underline{\theta}}_2 = \underline{\underline{U}} \longrightarrow \underline{\underline{\tilde{U}}}$ → eliminating "small" dimensions w.r.t small e.values

$$\underline{\underline{\tilde{U}}} = \begin{bmatrix} u_1 & \cdots & u_k \end{bmatrix}_{k\times n}$$

$$\underline{\underline{\tilde{\Sigma}}} = \begin{bmatrix} d_1 & & & & 0 \\ & d_2 & & & \\ & & \ddots & & \\ & & & d_k & \\ 0 & & & & 0 \end{bmatrix}$$

$$\rightarrow \underline{\underline{\theta_2}} = \underline{\underline{\tilde{U}}}_{k\times n} \longrightarrow \text{Choose } k \text{ eigenvectors} \quad k \ll n$$



$$\hat{x}_{n\times1} = h_\theta(x) = \underline{\underline{\theta_1}}\,\underline{\underline{\theta_2}}\,x.$$

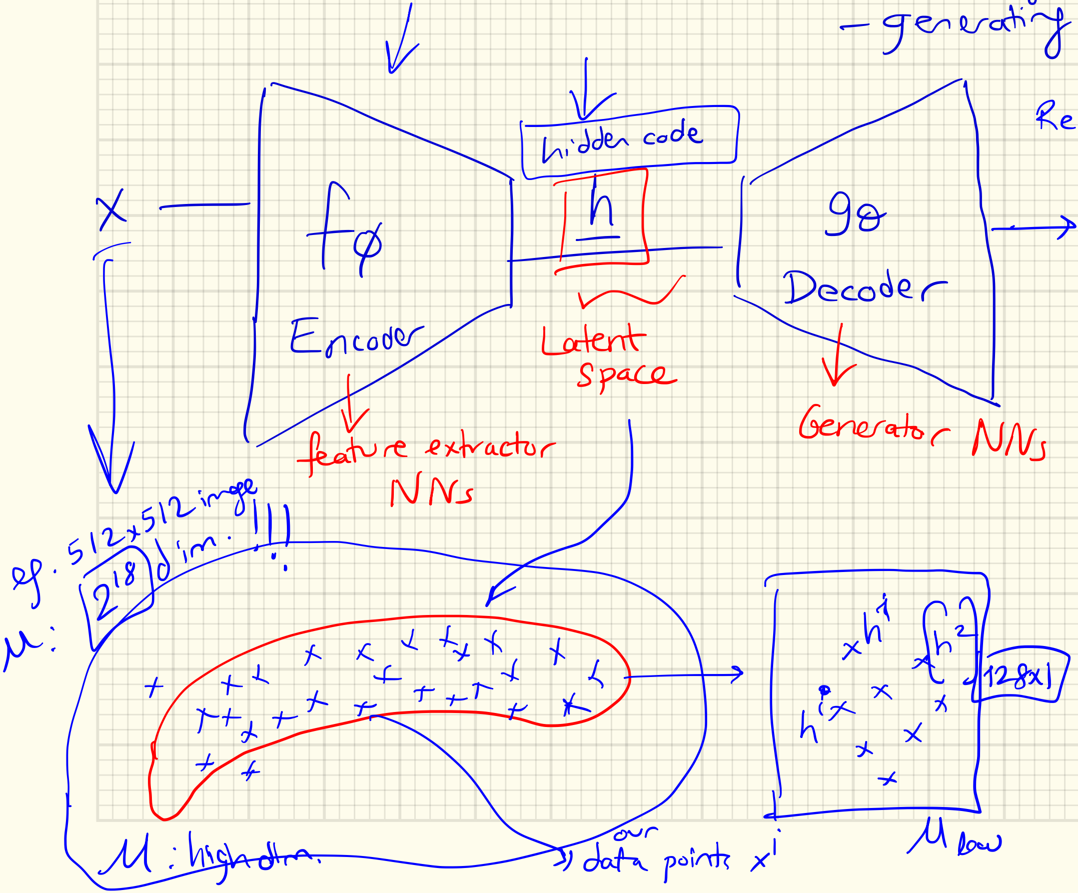
$$\underline{\underline{\theta_2}} = \begin{bmatrix} & & \end{bmatrix}_{k\times n} \qquad \underline{\underline{\theta_1}} = \begin{bmatrix} \\ \\ \end{bmatrix}_{n\times k}$$

★ PCA: ≈ a NN w/o (without) nonlinear activation functions.

↓ AutoEncoders are like PCA but w/ NONLINEAR NN models.

AUTOENCODERS (AEs):

- feature extraction
- dimensionality reduction
- clustering
- generating new samples



Reconstruct the input

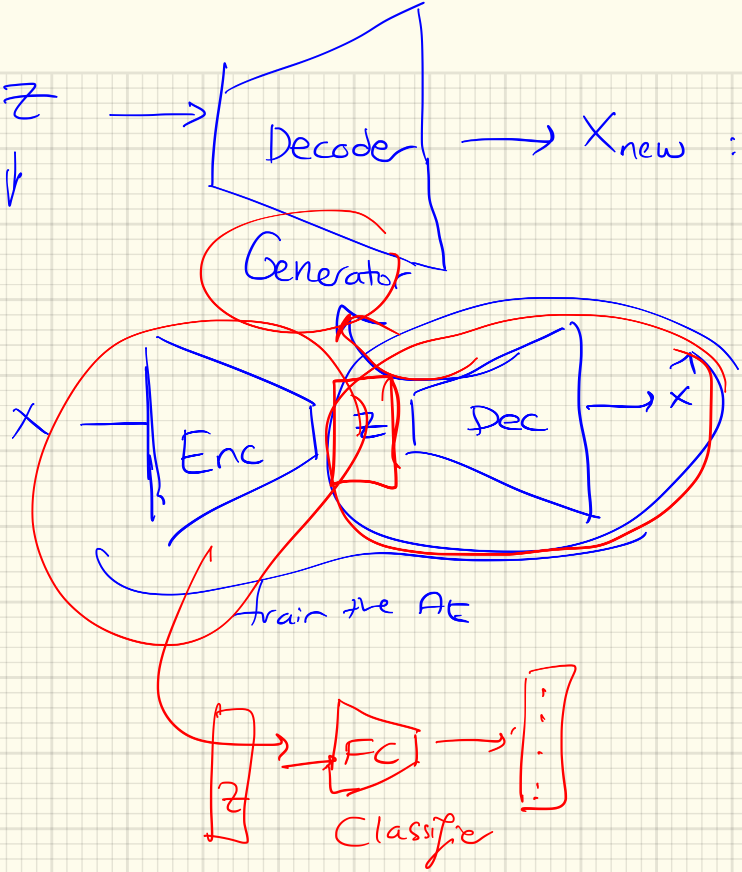
$$\hat{x} = g_\theta(f_\phi(x))$$

$$= h_\Theta(x)$$

★ AE tries to copy the input to the output.

★ Input Data:

- Images (Face)
- Places
- Text Document
- music/Audio Librarie
- Time Series Data

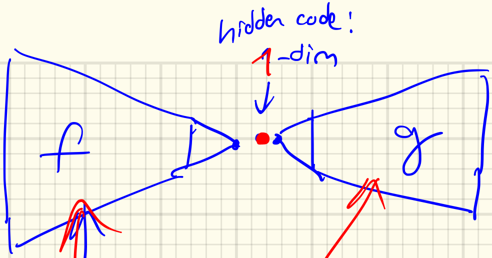


Unsupervised Learning tries to
: synthesize new data.

Extreme ex: AE:

$x^i_{n\times 1}$

$\{x^i\}_{i=1}^{m}$: millions of data



high capacity NNs

$\hat{x}^i \approx x^i$ ← normally

$\hat{x}^i = x^i$

exact reconst.

This AE does not learn anything useful about the data

!! f : maps each input in the training set to its INDEX in the dataset.

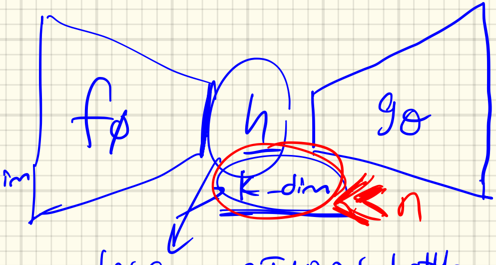
g : maps the index back to x^i again

★ Think about the generalization problem of this extreme case.

HOURGLASS model:

x

n-dim



reconstruct.

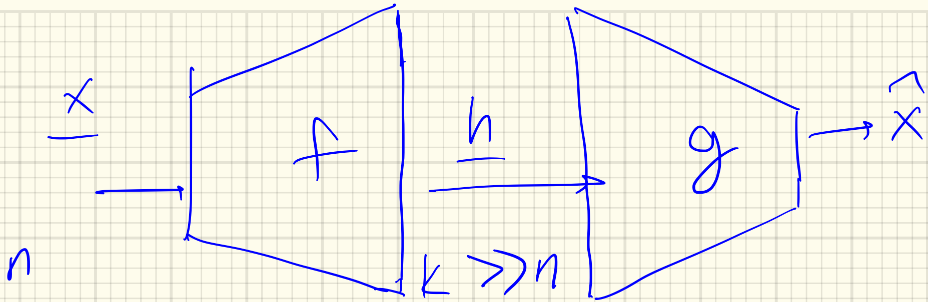
$\hat{x} = h_\theta(x) \approx x$

n-dim.

x goes thru INFORMATION bottleneck to get reconstructed.

★ UNDERCOMPLETE Representation:

Q.



overcomplete representation

? not preferred ?

Typically h spans a subspace of the $\mathcal{X}$.

Undercomplete representation: → tends to capture the "most salient features" of the input data distribution

→ How ?

By

→ ① Limiting the $\underline{\text{capacity}}$ of the AE (hourglass model)

→ ② $\underline{\text{Regularizing}}$ the AE ≡ adding some $\underline{\text{constraints}}$

↳ Unconstrained Optimization Objective:

$$\mathcal{L}(\theta) = \underbrace{\mathcal{L}_{Data}(\theta)} + \lambda \cdot \mathcal{L}_{\underline{Regularize}}(\theta)$$

$\underline{\text{Data Term of the Loss (objective) Function:}}$

$$\mathcal{L}_{Data}(\theta) = \mathcal{L}_{reconstruction}(x^i, h_\theta(x^i)) \underset{\text{typically}}{=} \sum_{i=1}^{m} \underline{\| x^i - h_\theta(x^i) \|_2^2}$$

1) Hypothesis fn for the AE,

$$\text{eg.}\ \underbrace{h_\theta(x)}_{\substack{L\text{-layer} \\ \text{FCN (MLP) or CNN}}} = f^L\underbrace{(\underline{\underline{W}}^L * (f^{L-1}(\underline{\underline{W}}^{L-1} *}_{\underline{\text{upconv}}} \underbrace{(\dots f^2(W^2 * \overset{\swarrow \text{activation}}{f^1}(\underline{\underline{W}}^1 * \underset{a^0}{x}))) \dots)}_{\underline{\text{conv-layers}}}$$

upconv ← conv-layers

Regularizers for AEs:

i) Sparse representation : Sparse AE

ii) Contracting AE : CAE ←.

iii) Denoising AE : → robustness to noise

push the AE to have other properties in addition to its ability to copy the input to the input.

① Sparse AE (SAE): adds a "sparsity" penalty in $\underline{h}$ code:

$$\arg\min_{\phi,\theta} \mathcal{L} = \mathcal{L}_{rec}(x, \underbrace{g_\theta(f_\phi(x))}_{\hat{x}}) + \boxed{\lambda} \underbrace{Reg(\underline{h})}_{\text{put sparseness constraint.}}$$

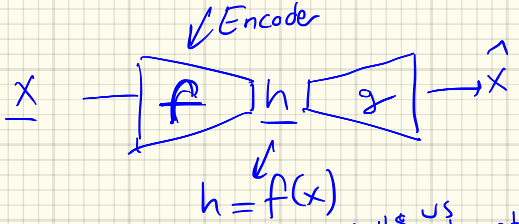
→ SAE : $Reg(\underline{h}) = \|\underline{h}\|_1$ → L1 norm.

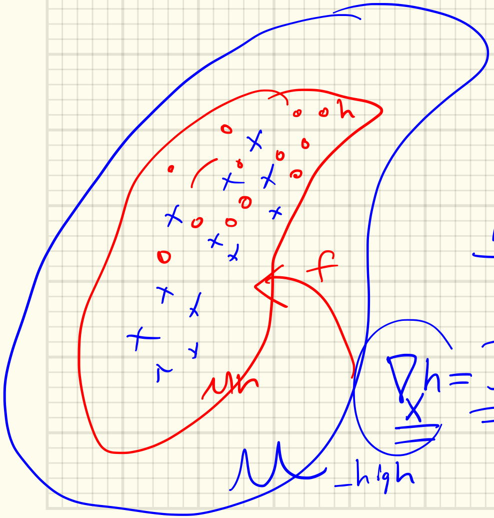
λ: regularizer weight → important hyperparameter in the optimization.
to be tuned.

→ $\mathcal{L}_{reg} = \|\underline{h}\|_1 = \sum_i |h_i|$: L1 regularization on the latent space.

(Recall: L1 norm is induced by a Laplacian prior on the latent code h. from probabilistic ML interpretation before)

ii) Contractive AE : (CAE)





Want $\boxed{\frac{\partial f}{\partial x}}$ small ! → Jacobian: → tells us contracting space
→ expanding
→ identity

$$\underline{h} = f(\underset{\partial h/\partial x}{\underline{x}}) \Rightarrow \frac{\partial f}{\partial x} = \underline{\underline{J}} = \text{Jacobian.}$$

$$\nabla_{\underline{x}} h = \underline{\underline{J}} = \begin{bmatrix} \frac{\partial f(x_1)}{\partial x_1} & \frac{\partial f(x_2)}{\partial x_1} & \cdots & \frac{\partial f(x_n)}{\partial x_1} \\ \vdots & & & \\ \frac{\partial f(x_1)}{\partial x_n} & \cdots & \cdots & \frac{\partial f(x_n)}{\partial x_n} \end{bmatrix}_{n \times n}$$

$$\mathcal{L} = \mathcal{L}_{data}(x, g(f(x))) + \lambda \cdot \underbrace{Reg(\underline{h})}_{\text{CAE} \quad \sum_i \|\nabla_x h_i\|^2}$$

Intuition : $f(x+\Delta x) \approx f(x) + \Delta x \underbrace{\boxed{\frac{\partial f}{\partial x}}}_{\underline{\underline{J}}}$

- A linear approximation to the nonlinear $f(\cdot)$ encoder.
- For a linear operator, it is contractive if $\|Jx\| \leq \underline{1} \quad \forall \|x\|=1$

$f \approx$ RHS $\longrightarrow f$ $\overset{\sim}{\text{is}}$ contractive.

→ CAE introduces the explicit regularizer on the hidden code h:

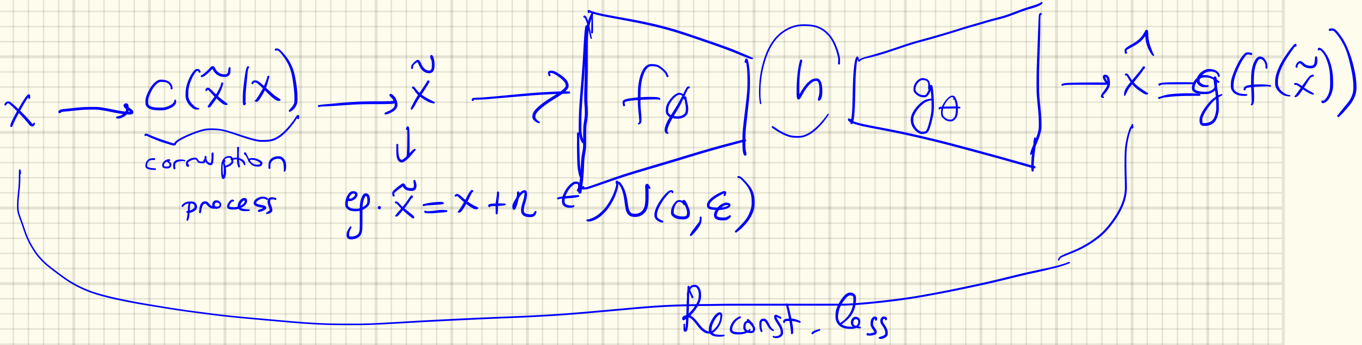
$$\underbrace{\mathcal{L}_{reg}(h)} = \left\| \underbrace{\frac{\partial f(x)}{\partial x}}_{\underline{\underline{J}}} \right\|_F^2$$

: Squared Frobenius norm of the Jacobian matrix.

makes derivatives of the encoder f as small as possible.

(iii) Denoising AEs (DAE)

: Assumes the input data is noisy, Tries to make the AE less sensitive to noise perturbations in the input.

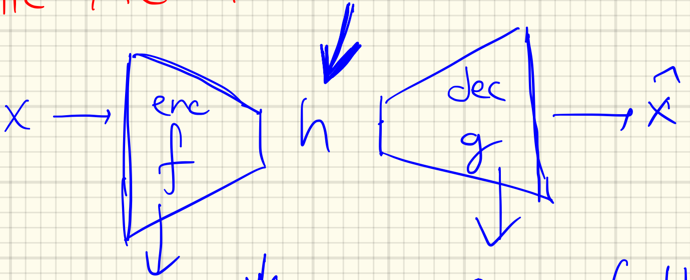


DAE:

1) Samples a training example $x^{(i)}$ from training data
2) Sample a corrupted version of $\tilde{x}^{(i)}$ ←
3) Train AE w/ $(x,\tilde{x})$ w/ $\text{Rec. Loss}\ \underbrace{(x, g_\theta(f_\phi(\tilde{x})))}_{\mathcal{L}(x,\hat{x})}$

↑

Stochastic AEs :



$$\underbrace{P_{\text{encoder}}(h|x)}$$

= posterior distrib./density

(inference)

$$\underbrace{P_{\text{decoder}}(x|h) = \text{generator density}}$$

ideally ≈ inverse of the encoder distribution.

$$\begin{tikzcd} x \arrow[r, bend left, "\text{inference}"] & h = z \arrow[l, bend left, "\text{generation.}"] \end{tikzcd}$$

Next time :

$\underline{\text{VAEs}}$