Pattern Recognition
Lecture 04-2
Basic Deep Learning

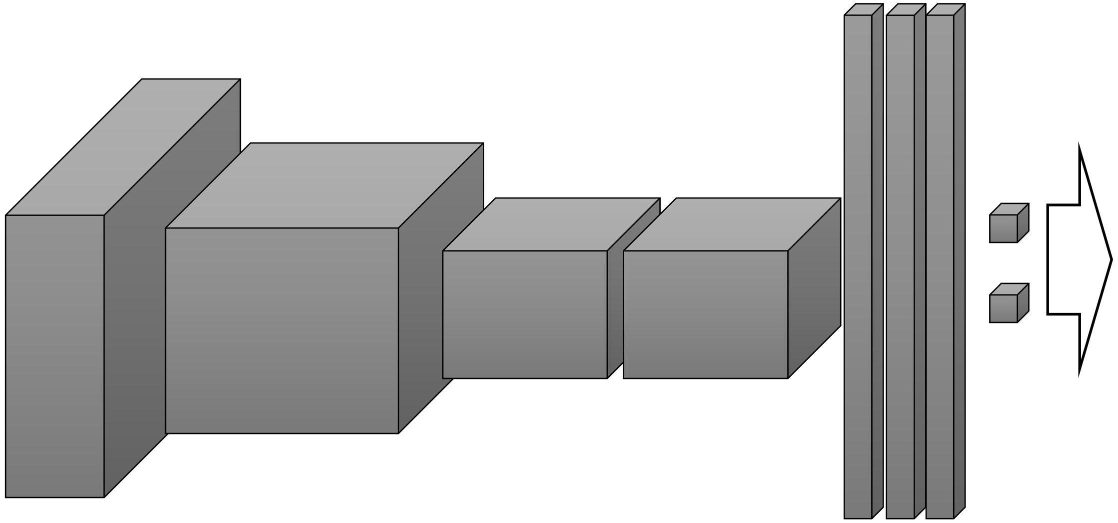
Prof. Jongwon Choi
Chung-Ang University
Fall 2022

This Class

- **Supervised Deep Learning**
  - Definition
  - Architecture
  - Prediction
  - Training
- **Unsupervised Deep Learning – Auto-encoder**
  - Definition
  - Architecture
  - Prediction
  - Training

Supervised Deep Learning - Training

- Cross-entropy Loss : $f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = -\sum_{c=1}^{N_c} \widetilde{y_c} \log p(\widehat{y_c}|\mathbf{x}, \mathbf{y})$
  - $\tilde{\mathbf{y}}$: One-hot vector of label (GT)
  - Good combination with softmax : $f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = -y_{(c=GT)}$

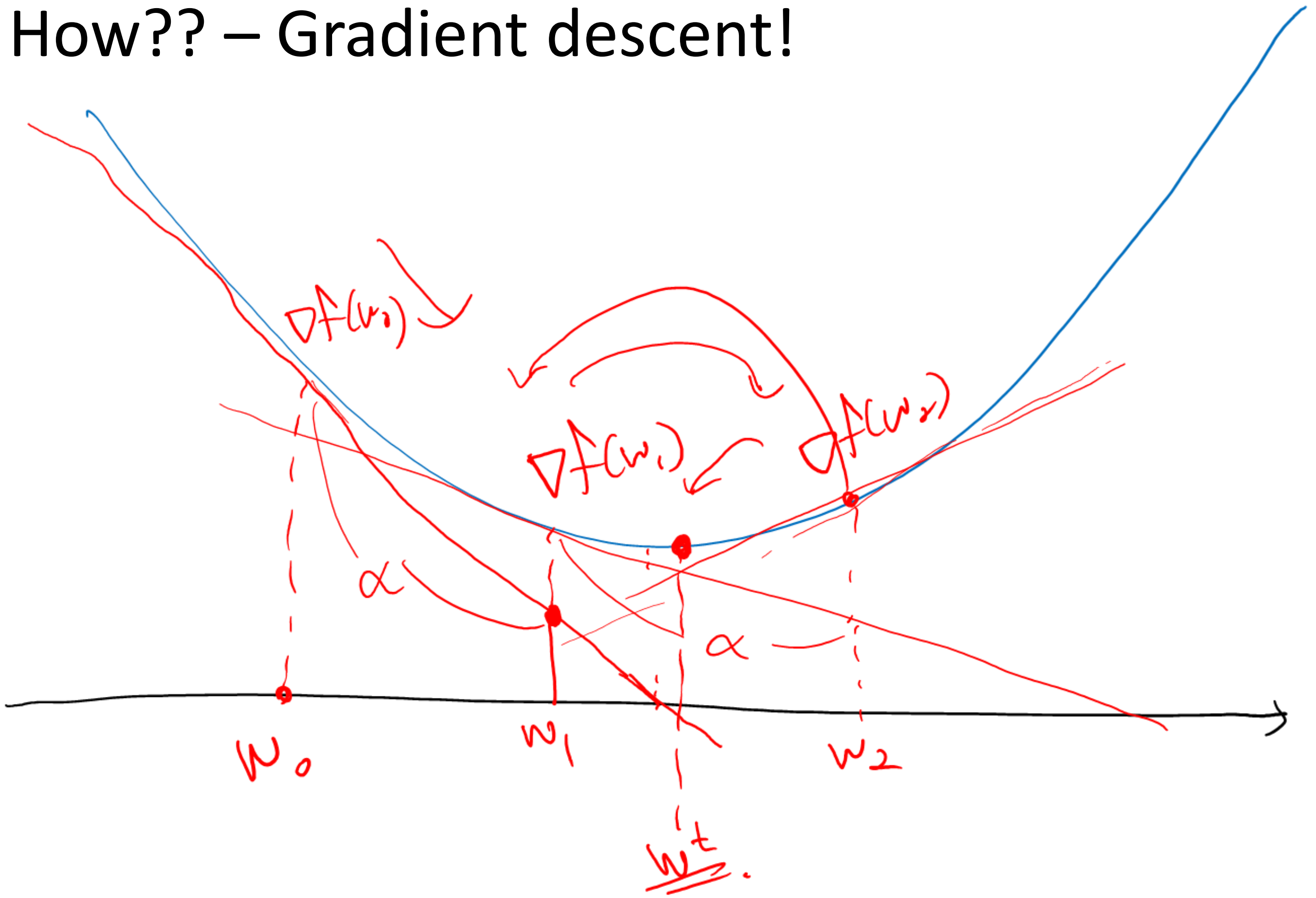


$$p(class\ 1|\mathbf{x}, \mathbf{y}) = \frac{\exp(y_1)}{\sum \exp(y_c)}$$

$$p(class\ 2|\mathbf{x}, \mathbf{y}) = \frac{\exp(y_2)}{\sum \exp(y_c)}$$

Supervised Deep Learning - Training

- **Let's start the training of last layer!**
  - How?? – Gradient descent!

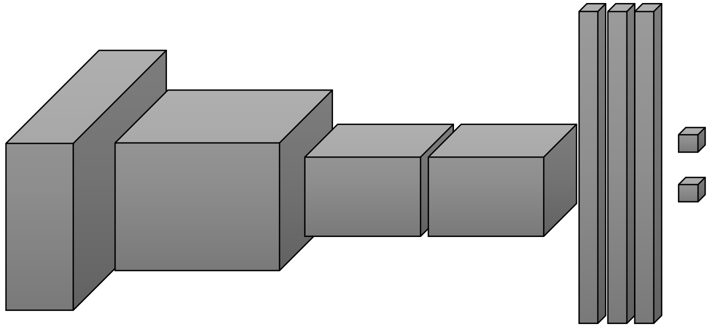


Gradient Descent for a Local Minimum

- We start with some initial guess, w^0
- Generate new guess by moving in the negative gradient direction:
  - $w^1 = w^0 - \alpha^0 \nabla f(w^0)$
    - This decreases 'f' if the "step size" α^0 is small enough
    - Usually, we decrease α^0 if it increases 'f'
- Repeat to successively refine the guess:
  - $w^{t+1} = w^t - \alpha^t \nabla f(w^t)$
- Stop if not making progress
  - $\|\nabla f(w^t)\| \leq \epsilon$

Supervised Deep Learning - Training

- **Gradient descent on the last layer**
  - Weight initialization – Gaussian random (Xavier's initialization)
  - for the iterative update: $w^{t+1} = w^t - \alpha^t \nabla f(w^t)$
    - α^t : Learning rate. Hyperparameter
    - $\nabla_{w_L^t} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \nabla_{w_L^t}\left(-y_{(c=GT)}\right) = \nabla_{w_L^t}\left(-{\mathbf{w}_c^t}^T \mathbf{x}_{L-1}\right)$
    - Thus, $\nabla_{w_L^t} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = -\mathbf{x}_{L-1}$



Supervised Deep Learning - Training

- **Gradient Descent on the remaining layers – Chain Rule!!**
  - Weight initialization – Gaussian random (Xavier's initialization)
  - for the iterative update: $w_{L-1}^{t+1} = w_{L-1}^{t} - \alpha^{t} \nabla f(w_{L-1}^{t})$
    - $\nabla_{\mathbf{w}_{L-1}^{t}} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \dfrac{\partial f_{CE}}{\partial \mathbf{w}_{L-1}^{t}} = \dfrac{\partial \mathbf{x}_{L-1}^{t}}{\partial \mathbf{w}_{L-1}^{t}} \times \dfrac{\partial f_{CE}}{\partial \mathbf{x}_{L-1}^{t}}$
    - $\nabla_{\mathbf{w}_{L-2}^{t}} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \dfrac{\partial f_{CE}}{\partial \mathbf{w}_{L-2}^{t}} = \dfrac{\partial \mathbf{x}_{L-2}^{t}}{\partial \mathbf{w}_{L-2}^{t}} \times \dfrac{\partial \mathbf{x}_{L-1}^{t}}{\partial \mathbf{x}_{L-2}^{t}} \times \dfrac{\partial f_{CE}}{\partial \mathbf{x}_{L-1}^{t}}$
    - $\nabla_{\mathbf{w}_{L-3}^{t}} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \dfrac{\partial f_{CE}}{\partial \mathbf{w}_{L-3}^{t}} = \dfrac{\partial \mathbf{x}_{L-3}^{t}}{\partial \mathbf{w}_{L-3}^{t}} \times \dfrac{\partial \mathbf{x}_{L-2}^{t}}{\partial \mathbf{x}_{L-3}^{t}} \times \dfrac{\partial \mathbf{x}_{L-1}^{t}}{\partial \mathbf{x}_{L-2}^{t}} \times \dfrac{\partial f_{CE}}{\partial \mathbf{x}_{L-1}^{t}}$

Supervised Deep Learning - Training

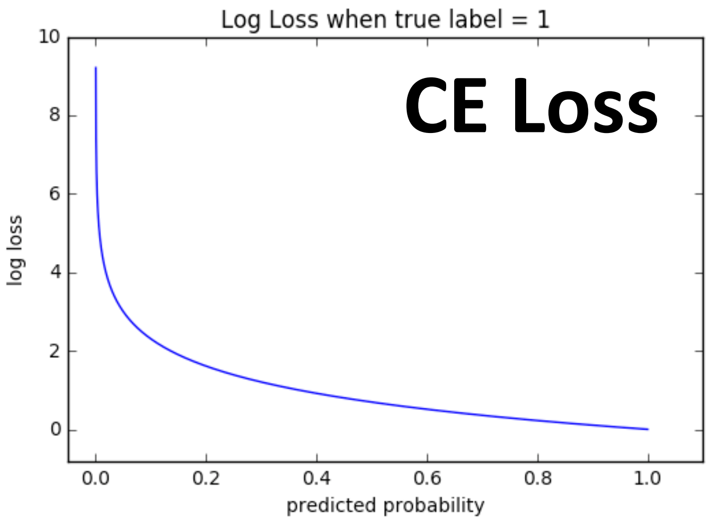
- **Gradient Descent on the remaining layers – Chain Rule!!**
    - $\nabla_{\mathbf{w}_{L-2}^t} f_{CE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \frac{\partial f_{CE}}{\partial \mathbf{w}_{L-2}^t} = \frac{\partial \mathbf{x}_{L-2}^t}{\partial \mathbf{w}_{L-2}^t} \times \frac{\partial \mathbf{x}_{L-1}^t}{\partial \mathbf{x}_{L-2}^t} \times \frac{\partial f_{CE}}{\partial \mathbf{x}_{L-1}^t}$
- $\frac{\partial f_{CE}}{\partial \mathbf{x}_{L-1}} = \frac{\partial}{\partial \mathbf{x}_{L-1}} \left(-{\mathbf{w}_c^t}^T \mathbf{x}_{L-1} \right) = -\mathbf{w}_c^t$
- $\frac{\partial \mathbf{x}_{L-1}}{\partial \mathbf{x}_{L-2}^t} = \frac{\partial}{\partial \mathbf{x}_{L-2}^t} \left(h(\mathbf{W}\mathbf{x}_{L-2}^t) \right) = \frac{\partial}{\partial \mathbf{x}_{L-2}^t} (\mathbf{W}\mathbf{x}_{L-2}^t) \times \frac{\partial h(\mathbf{W}\mathbf{x}_{L-2}^t)}{\partial (\mathbf{W}\mathbf{x}_{L-2}^t)} = \mathbf{W} \times \frac{\partial h\left(\mathbf{W}\mathbf{x}_{L-2}^t\right)}{\partial (\mathbf{W}\mathbf{x}_{L-2}^t)}$
- $\frac{\partial \mathbf{x}_{L-2}^t}{\partial \mathbf{w}_{L-2}^t} = \frac{\partial}{\partial \mathbf{w}_{L-2}^t} \left({\mathbf{w}_{L-2}^t}^T \mathbf{x}_{L-3}^t \right) = \mathbf{x}_{L-3}^t$

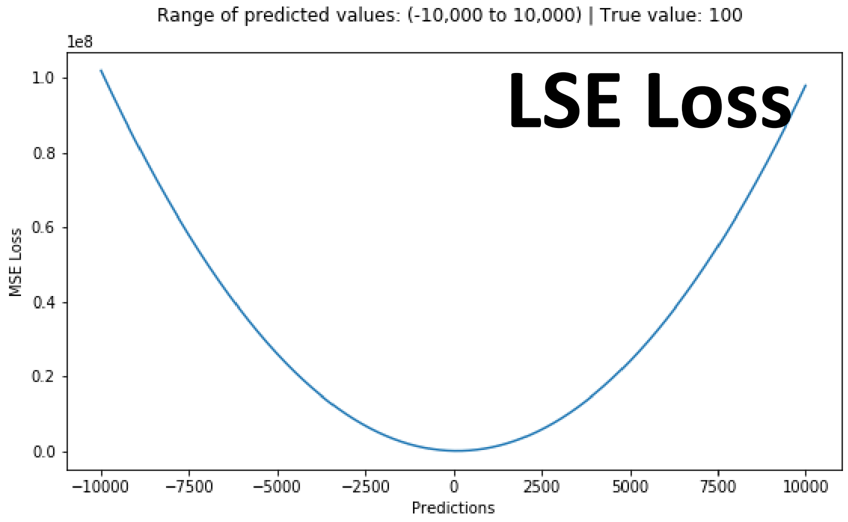
Supervised Deep Learning - Training

- **Gradient Descent on the remaining layers – Chain Rule!!**
  - $\frac{\partial h\left(\mathbf{W}\mathbf{x}_{L-2}^{t}\right)}{\partial\left(\mathbf{W}\mathbf{x}_{L-2}^{t}\right)}$

Supervised Deep Learning - Training

- Why not LSE Loss?
  - $f_{LSE}(\tilde{\mathbf{y}},\ \mathbf{y},\ \mathbf{x}) = \sum_{c=1}^{N_c}\left(\tilde{y}_c - p(\hat{y}_c|\mathbf{x},\mathbf{y})\right)^2$
  - Contrary to LSE, CE does not have the point of gradient=0
  - This leads the model to be trained continuously!





Supervised Deep Learning - Training

- **Mini-batch Scheme Update**
  - $f^t\left(\widetilde{\mathbf{Y}},\ \mathbf{Y},\ \mathbf{X}\right) = \sum_{i=1}^{N_b} f_{CE}(\tilde{\mathbf{y}}_i,\ \mathbf{y}_i,\ \mathbf{x}_i)$
  - Mini-batch : Randomly sampled subset of input data
    - $\left\{(\tilde{\mathbf{y}}_1,\ \mathbf{y}_1,\ \mathbf{x}_1), \dots, \left(\tilde{\mathbf{y}}_{N_b},\ \mathbf{y}_{N_b},\ \mathbf{x}_{N_b}\right)\right\} \subset (\widetilde{\mathbf{Y}},\ \mathbf{Y},\ \mathbf{X})$
- **Drop-out**
  - Set randomly chosen response values to 0
  - Avoid the overfitting problem and gradient vanishing

3 Key Points of Deep Learning

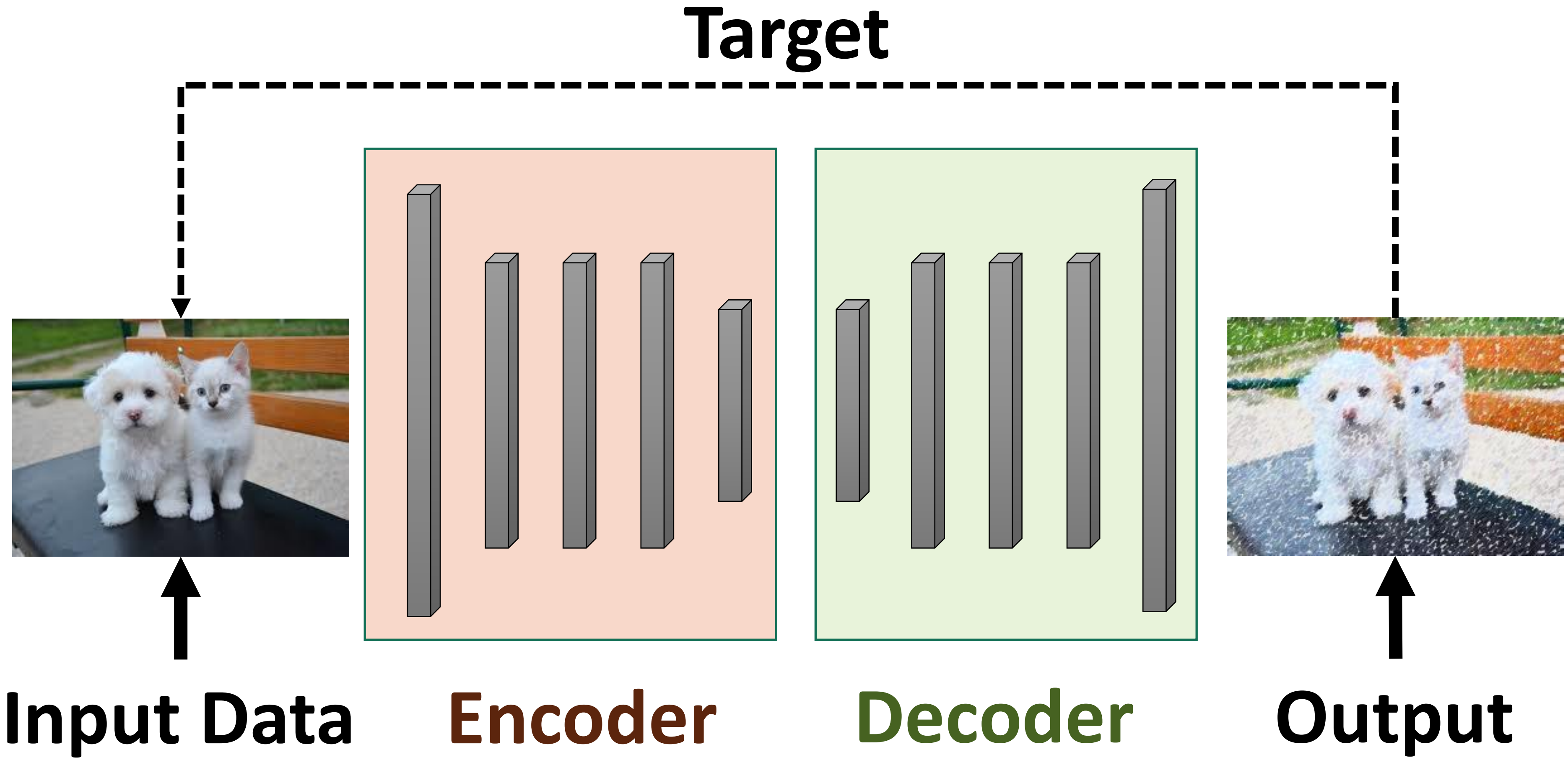
- Very very large model (Numerous weight parameters)
  - Parallel computation (especially matrix computation) by **GPU**
- Severe overfitting with the large model
  - **Big data** (ImageNet etc.)
- Gradient vanishing / exploding problem
  - Initialize the weight parameters by **Xavier's initialization**

Unsupervised Deep Learning – Auto-encoder

- **Definition of Auto-encoder**
  - A neural network that results in (Output = Input)
  - Thus, the network can be trained in the unsupervised scheme

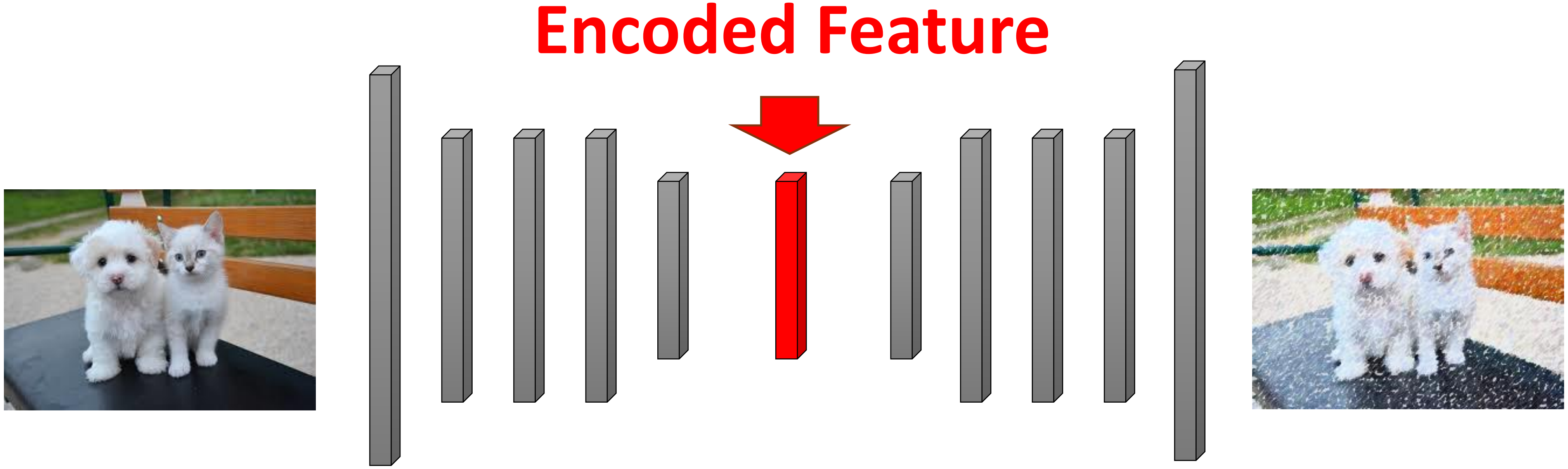
- **Objective of Auto-encoder**
  - At first, it is developed for initializing the layers of supervised deep network (Before Xavier's initialization)
  - In these days, it is utilized for data analysis, data clustering, data generation, etc.

Auto-encoder - Architecture



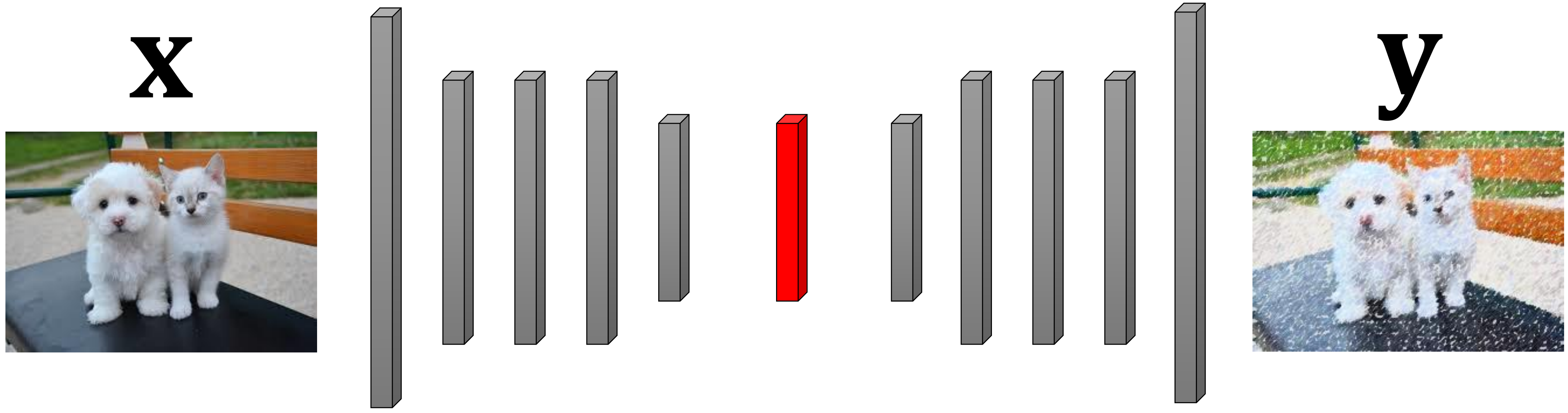
Auto-encoder - Architecture

- When the encoded feature is smaller than the input data,
- the information of input data is compressed in the encoded feature
- because the input data should be reconstructed from that!



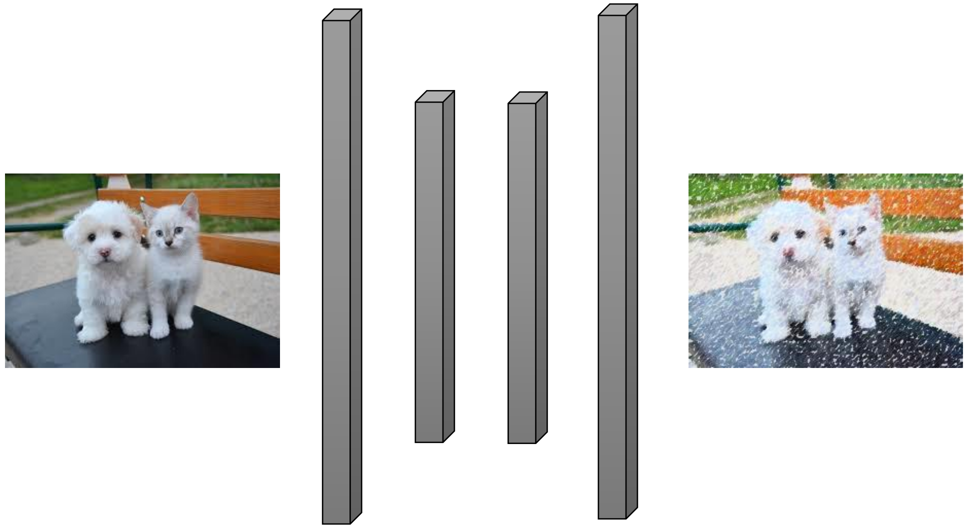
Supervised Deep Learning - Training

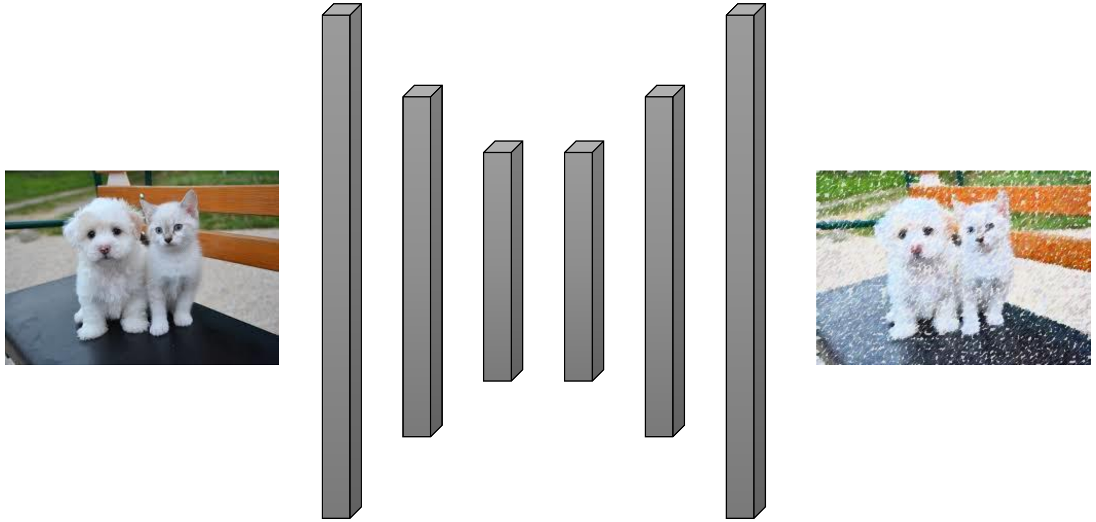
- Reconstruction Loss
  - $f_{MSE}(\mathbf{y}, \mathbf{x}) = -\sum(\mathbf{x}_i - \mathbf{y}_i)^2$
  - $f_{MAE}(\mathbf{y}, \mathbf{x}) = -\sum|\mathbf{x}_i - \mathbf{y}_i|$



Auto-encoder - Training

- Initial Auto-encoder (Denoising auto-encoder)
  - Stacked auto-encoder



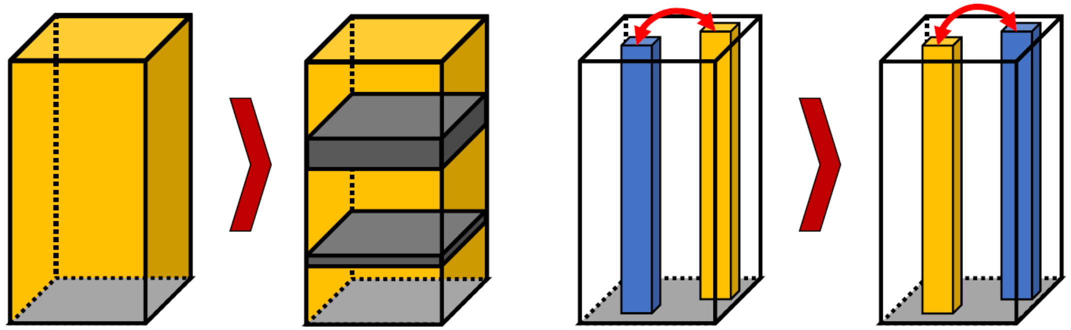


Auto-encoder - Training

- Conventional Auto-encoder

  - Make noisy input
  - Even with the noisy input, the auto-encoder should reconstruct the de-noised original input

- Special case – TRACA ***



(a) Channel corrupting process (b) Feature vector exchange process

*** "Context-aware Deep Feature Compression for High-speed Visual Tracking", CVPR2018

Summary

- **Supervised Deep Learning**
  - Definition
  - Architecture
  - Prediction
  - Training
- **Unsupervised Deep Learning – Auto-encoder**
  - Definition
  - Architecture
  - Prediction
  - Training