

深度学习

Lecture 7 Unsupervised Learning

YMSC, Pang Tongyao

Recaps: Supervised Learning

- Given labelled data $\{x_i, y_i\}$, the goal is to learn the map from x to y .

Step 1: Define the hypothesis space: $\mathcal{H} = \{f_\theta\}$

(f_θ can be MLP, CNN, RNN, Transformer...)

Step 2: Define the empirical loss

Step 3: Optimize the parameters to find the best approximation

Limitations: Dependency on Labelled Data, Poor Generalization, Vulnerability, Inflexibility.....

Unsupervised Learning

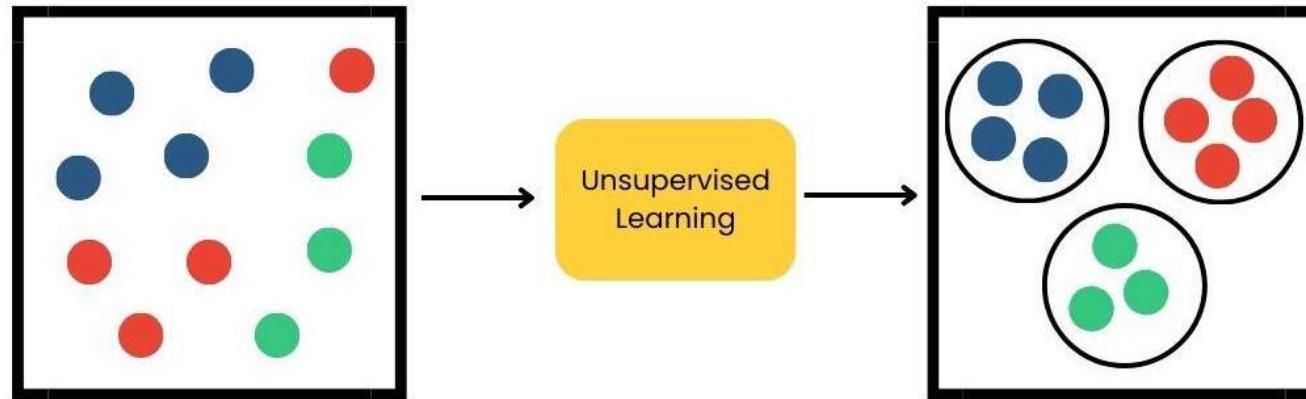
- Definition: Learn from unlabeled data!
- Motivation:
 - Labeling data is expensive, time-consuming and even unavailable
 - Discovering Hidden Patterns and Structures
 - Adaptability to New Domains and Tasks



“If intelligence was a cake, the bulk of the cake would be unsupervised learning”

Without Labels

Clustering



How to define the loss function?

Restoration



Clustering

- Divide the dataset $\mathcal{D} = \{\mathbf{x}\}$ into different categories:

$$\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \dots \cup \mathcal{D}_K$$

such that:

1. Data within the same category is similar.
 2. Data from different categories is dissimilar.
- Unlike supervised learning classification, clustering has no labelled data $\{y\}$.

K-means

Divide the data into K clusters:

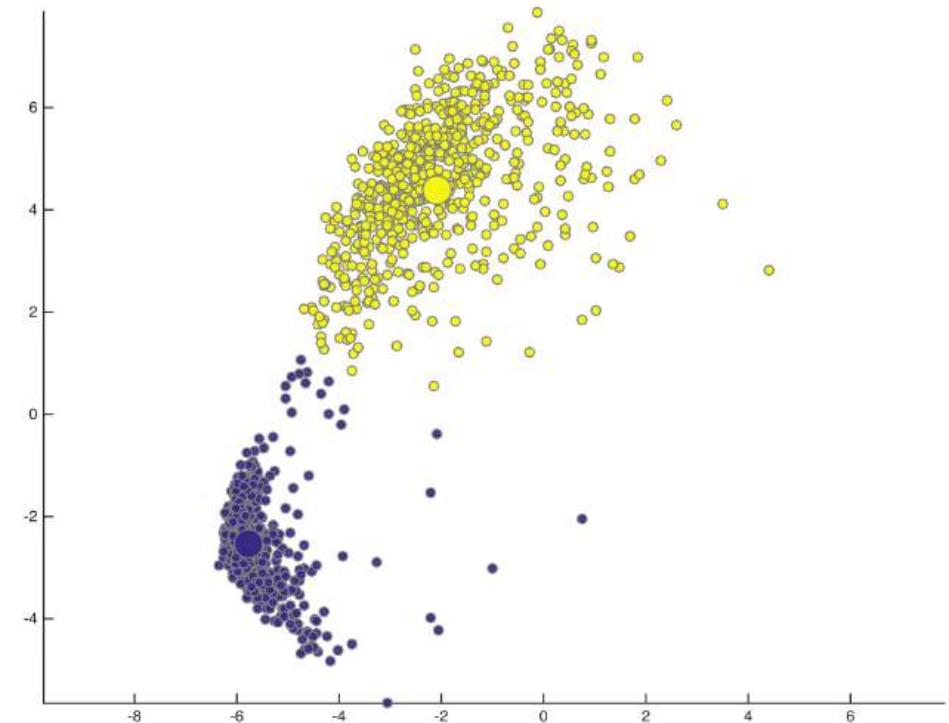
1. **Initialization:** Select K initial centroids.
2. **Assignment:** Assign each data point to the nearest centroid.

$$i(x) = \arg \min_j \|x - c_j\|_2^2$$

3. **Update Centroids:** Recalculate each centroid as the mean of all points assigned to that cluster.

$$c_i = \text{Mean}(\{x: i(x) = i\})$$

4. Repeat 2,3



K-means

- Assignment matrix

$$\gamma_{ij} = \begin{cases} 1 & \text{if } x_i \text{ is cluster } j \\ 0 & \text{else} \end{cases}$$

- Assignment

$$\gamma_{ij}^{(k)} = \arg \min_{\substack{\gamma_{ij} \in \{0,1\}, \\ \sum_j \gamma_{ij} = 1}} \frac{1}{2N} \sum_{ij} \gamma_{ij} \|x_i - c_j^{(k-1)}\|_2^2$$

- Update centroids

$$c_j^{(k)} = \arg \min_{c_j} \frac{1}{2N} \sum_{ij} \gamma_{ij}^{(k)} \|x_i - c_j\|_2^2$$

K-means

- K-means solves the following problem alternatively

$$\min_{\gamma_{ij} \in S, c_j} \frac{1}{2N} \sum_{ij} \gamma_{ij} \|x_i - c_j\|_2^2$$

K-means Clustering Algorithm

Input: $\mathcal{D} = \{x_i\}_{i=1}^N, x_i \in \mathbb{R}^d, K$ (number of clusters)

Initialize: $C = (c_1, c_2, \dots, c_K)$.

Iterate Until Converge:

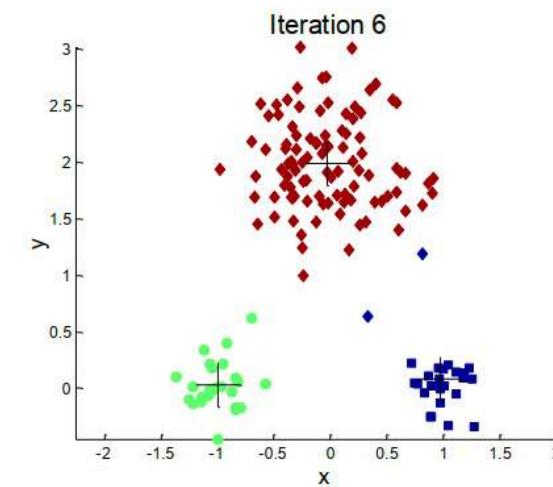
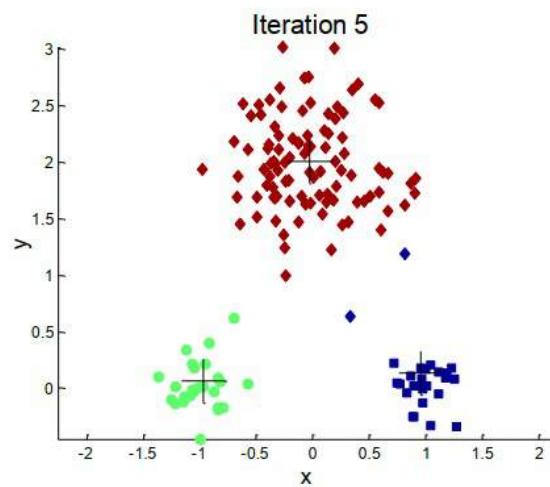
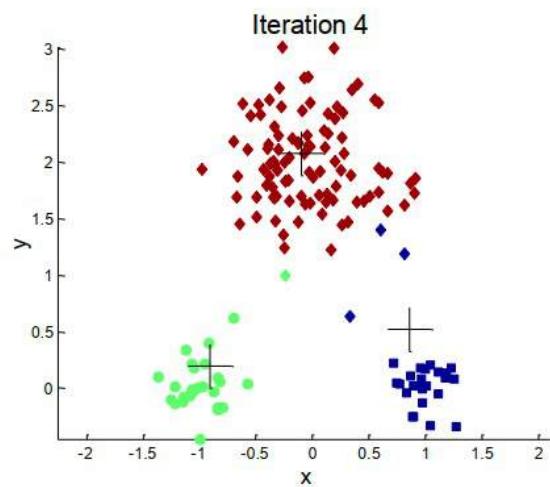
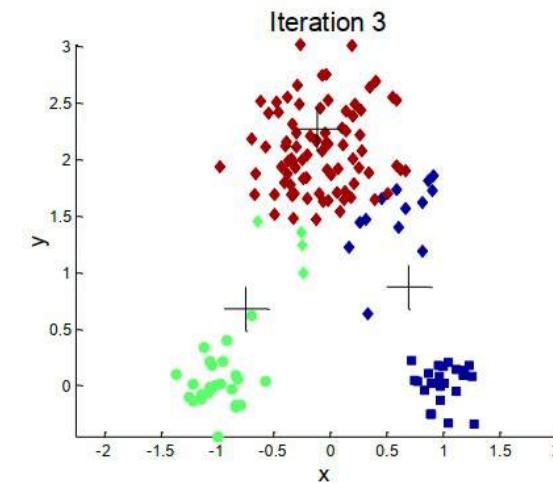
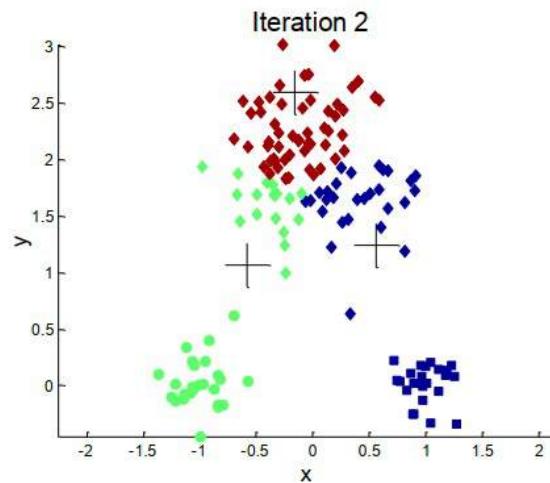
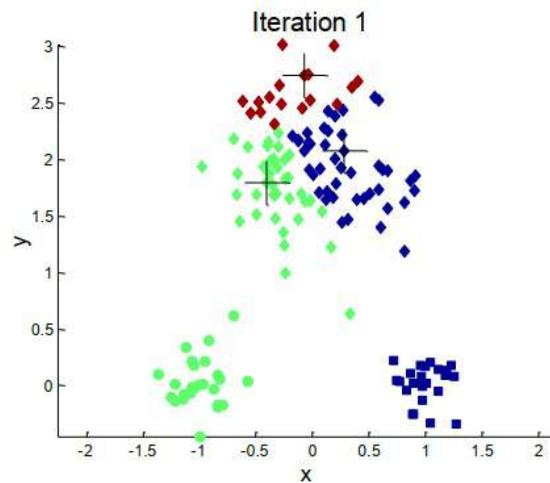
- **Update γ :** $\gamma_{ij} = \begin{cases} 1 & \text{if } j = \operatorname{argmin}_k \|x_i - c_k\|_2^2 \\ 0 & \text{else} \end{cases}$

- **Update C :** $c_j = (\sum_i \gamma_{ij} x_i) / \sum_i \gamma_{ij}$

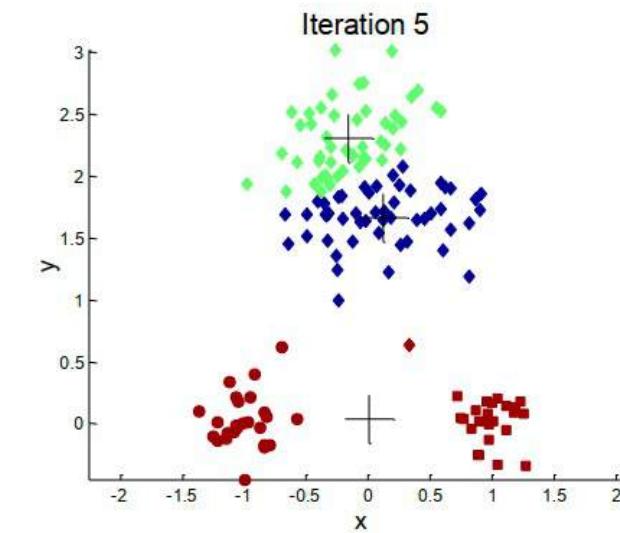
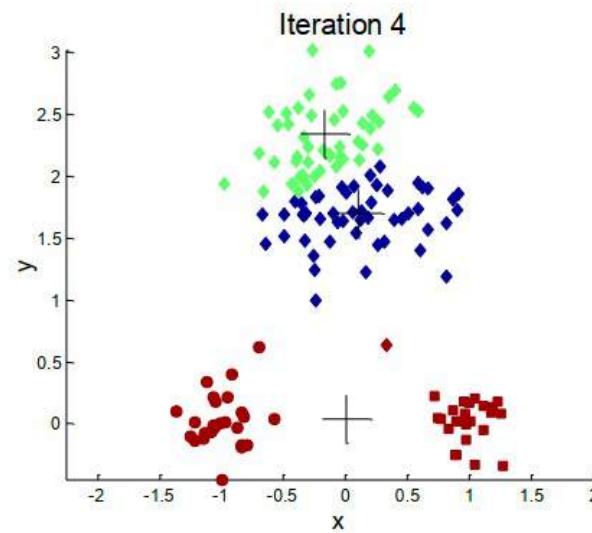
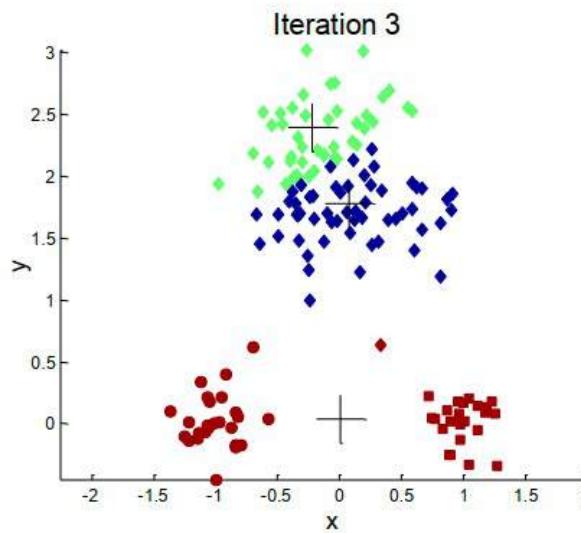
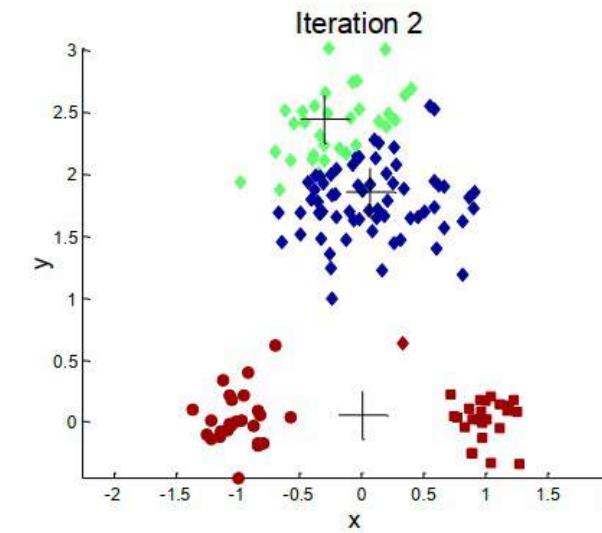
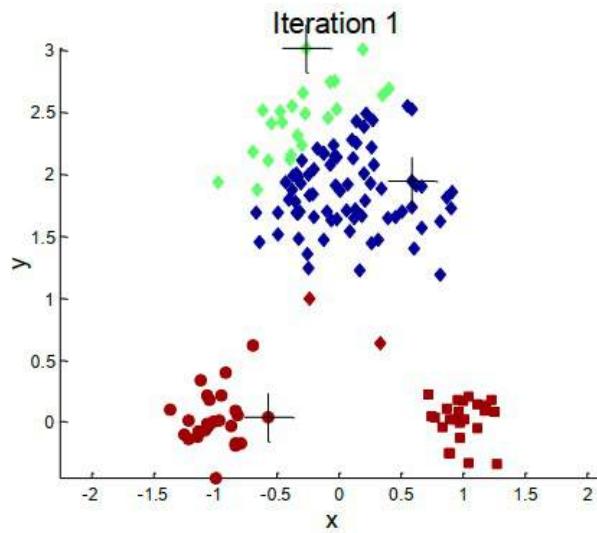
Output: Cluster centers C and cluster assignment γ .

K-means





The clustering results may vary depending on the selection of initial points.



The clustering results may vary depending on the selection of initial points.

Gaussian Mixture Model

- z : latent variable
- If z can only take finite number of values, e.g.

$$p(z = i) = \pi_i, i = 1, 2, \dots, N.$$

and $p(x|z)$ is Gaussian, then

$$p(x) = \sum_{i=1}^N \pi_i \mathcal{N}(\mu_i, \Sigma_i)$$

Gaussian Mixture Model

(Generalized K-means with soft assignments)

Gaussian Mixture Model

GMM optimization

- update assignment $\gamma_{ij} = p(z_i = j|x_i)$ (posterior)

$$\gamma_{ij}^{(k+1)} := p(z_i = j|x_i) = \frac{p(z_i = j)p(x_i|z_i = j)}{p(x_i)} = \frac{p(z_i = j)p(x_i|z_i = j)}{\sum_j p(z_i = j)p(x_i|z_i = j)} = \frac{\pi_j^{(k)} \mathcal{N}(x_i|\mu_j^{(k)}, \Sigma_j^{(k)})}{\sum_j \pi_j^{(k)} \mathcal{N}(x_i|\mu_j^{(k)}, \Sigma_j^{(k)})}$$

- update center (μ, Σ) and $\pi_i = p(z = i)$ (prior)

$$\pi_j^{k+1} = p(z = j) = \int p(z = j|x)p(x)dx \approx \frac{\sum_{i=1}^N p(z = j|x_i)}{N} = \frac{\sum_i \gamma_{ij}^{(k+1)}}{N}$$

$$\mu_j^{(k+1)} = \frac{1}{N_j^{(k+1)}} \sum_i \gamma_{ij}^{(k+1)} x_i, \quad \Sigma_j^{(k+1)} = \frac{1}{N_j^{(k+1)}} \sum_i \gamma_{ij}^{(k+1)} (x_i - \mu_j^{(k+1)}) (x_i - \mu_j^{(k+1)})^\top, \quad N_j^{(k+1)} = \sum_i \gamma_{ij}^{(k+1)}$$

Gaussian Mixture Model

Expectation Maximization (EM) Algorithm (Variational Inference)

$$\sum_i \log \left(\sum_j \gamma_{ij} \pi_j N(x_i | \mu_j, \Sigma_j) / \gamma_{ij} \right) \geq \sum_i \left(\sum_j \gamma_{ij} \log \frac{\pi_j N(x_i | \mu_j, \Sigma_j)}{\gamma_{ij}} \right) \quad \text{Jensen's Inequality}$$

- Expectation Step: Compute the expected value of the latent variable assignments given current parameters.

$$\gamma_{ij}^{(k+1)} = \arg \max_{\{\gamma_{ij}: \sum_j \gamma_{ij} = 1, \gamma_{ij} \in [0,1]\}} \sum_i \sum_j \gamma_{ij} \log \pi_i^{(k)} \mathcal{N}(x_i | \mu_j^{(k)}, \Sigma_j^{(k)}) - \gamma_{ij} \log \gamma_{ij}$$

- Maximization Step: Maximizing the expected complete-data log-likelihood under the responsibilities $\gamma_{ij}^{(k+1)}$ from the E-step.

$$(\pi_j^{(k+1)}, \mu_j^{(k+1)}, \Sigma_j^{(k+1)}) = \arg \max_{\theta} \sum_i \sum_j \gamma_{ij}^{(k+1)} \log \pi_j N(x_i | \mu_j, \Sigma_j)$$

Contrastive Learning

- Supervised classification is kind of Contrastive Learning

- maximize similarities of positive pairs
- minimize similarities of negative pairs

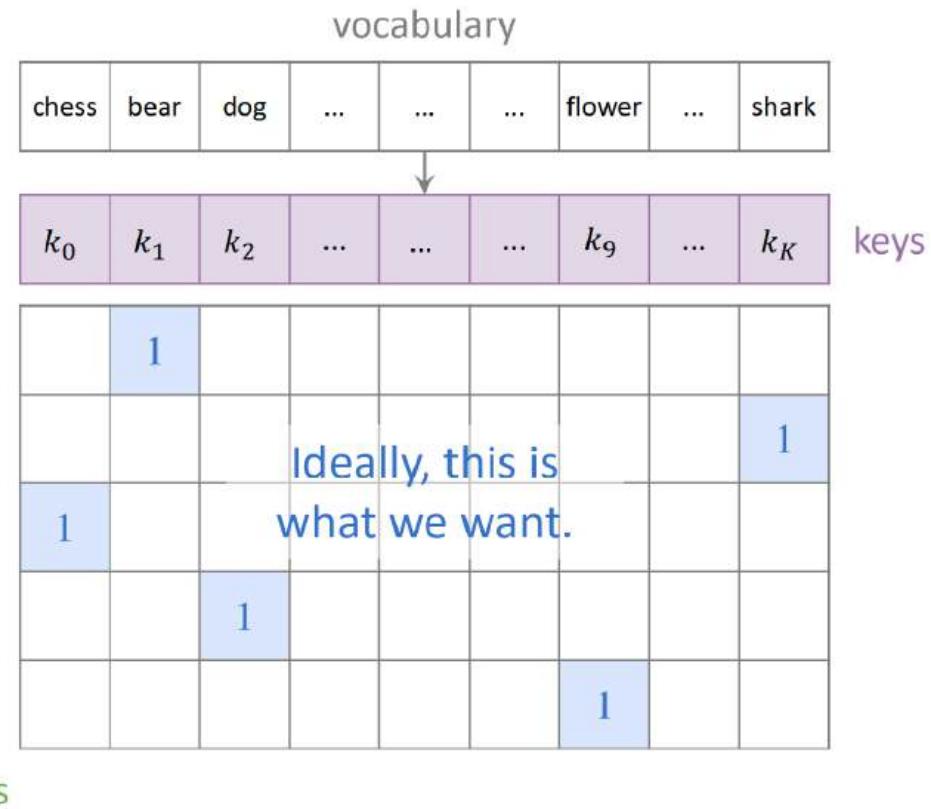
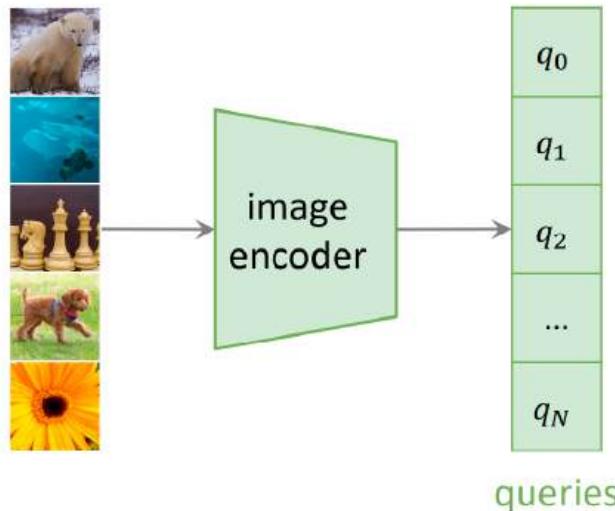


Image credit: Kaiming He

Contrastive Learning

- when there is no label, form a vocabulary w/ data itself

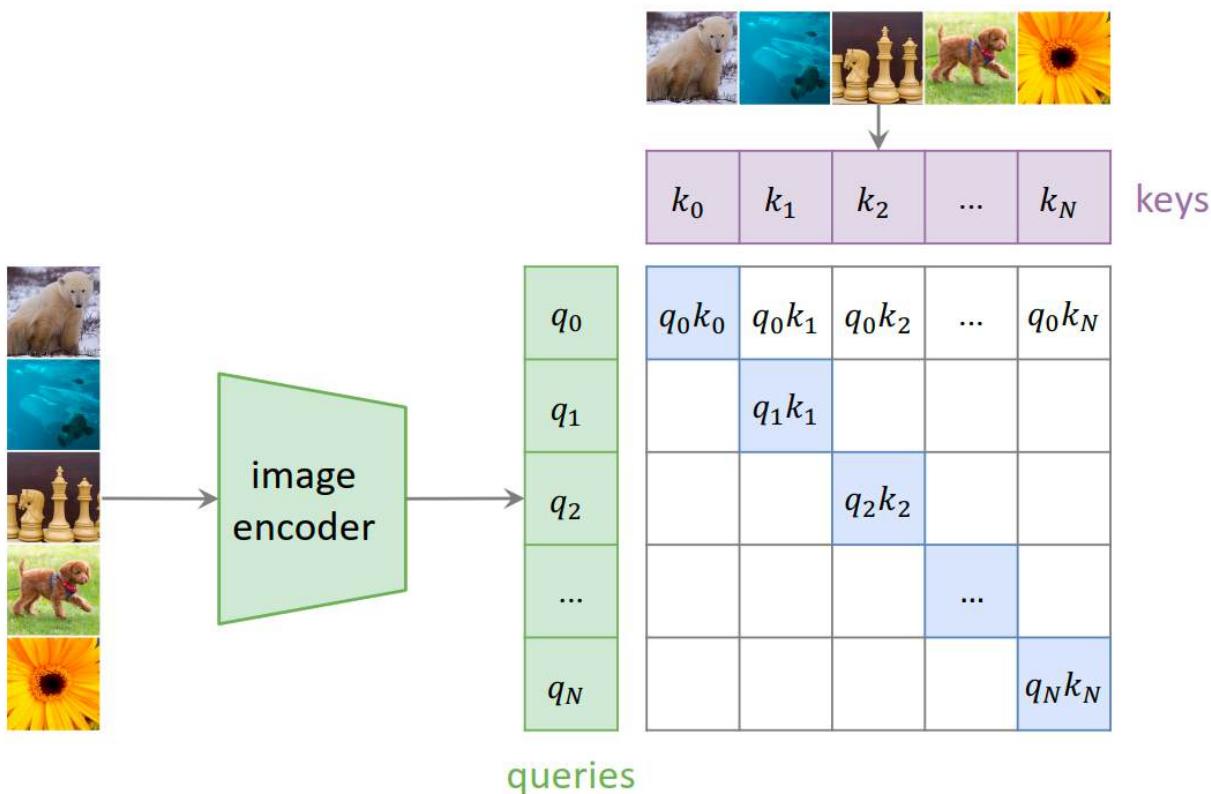


Image credit: Kaiming He

Contrastive Learning

- when there is no label, form a vocabulary w/ data itself

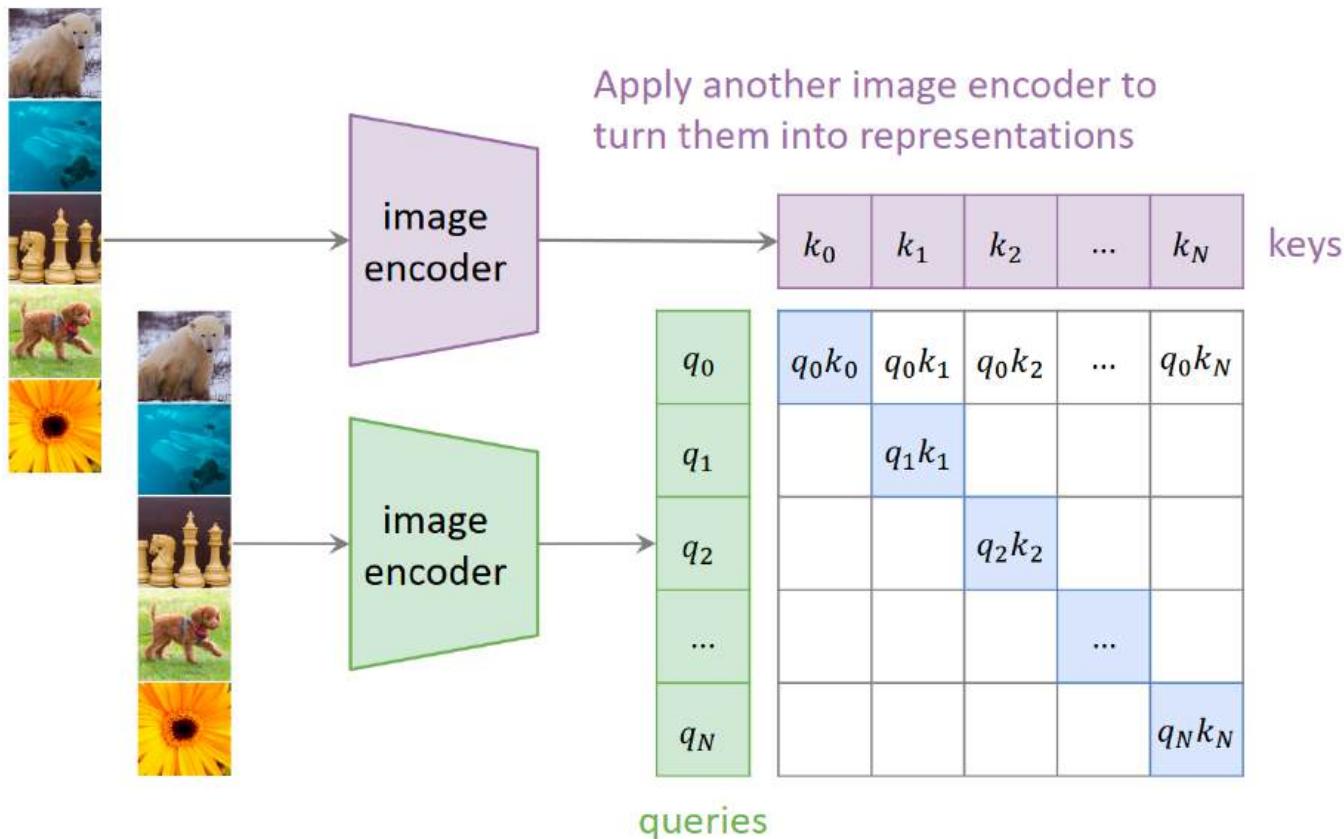
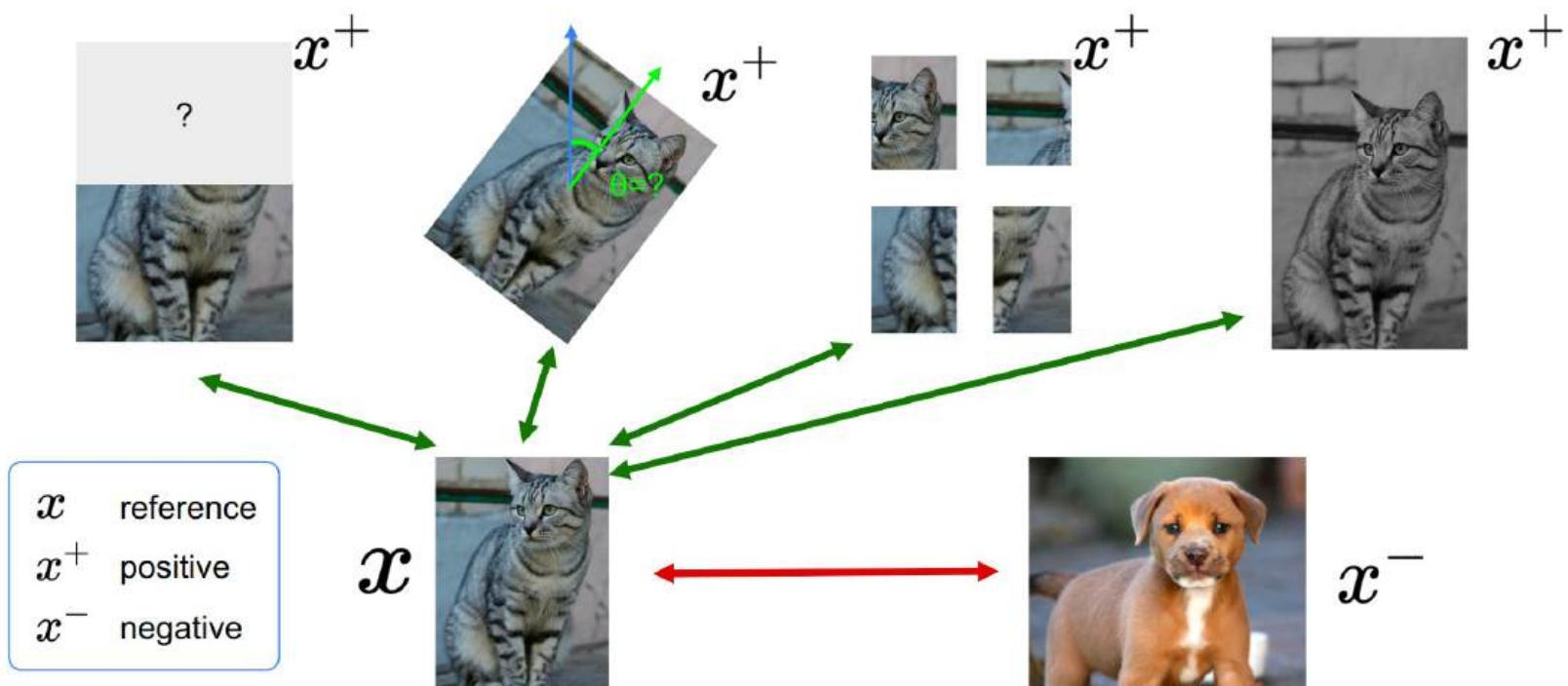


Image credit: Kaiming He

Contrastive Learning

Maximize(Minimize) similarities of positive(negative) pairs



Contrastive Learning

- Loss function

$$L = -\mathbb{E}_X \left[\log \frac{\exp(s(f(x), f(x^+)))}{\underbrace{\exp(s(f(x), f(x^+)))}_{\text{score for the positive pair}} + \sum_{j=1}^{N-1} \underbrace{\exp(s(f(x), f(x_j^-)))}_{\text{score for the N-1 negative pairs}}} \right]$$

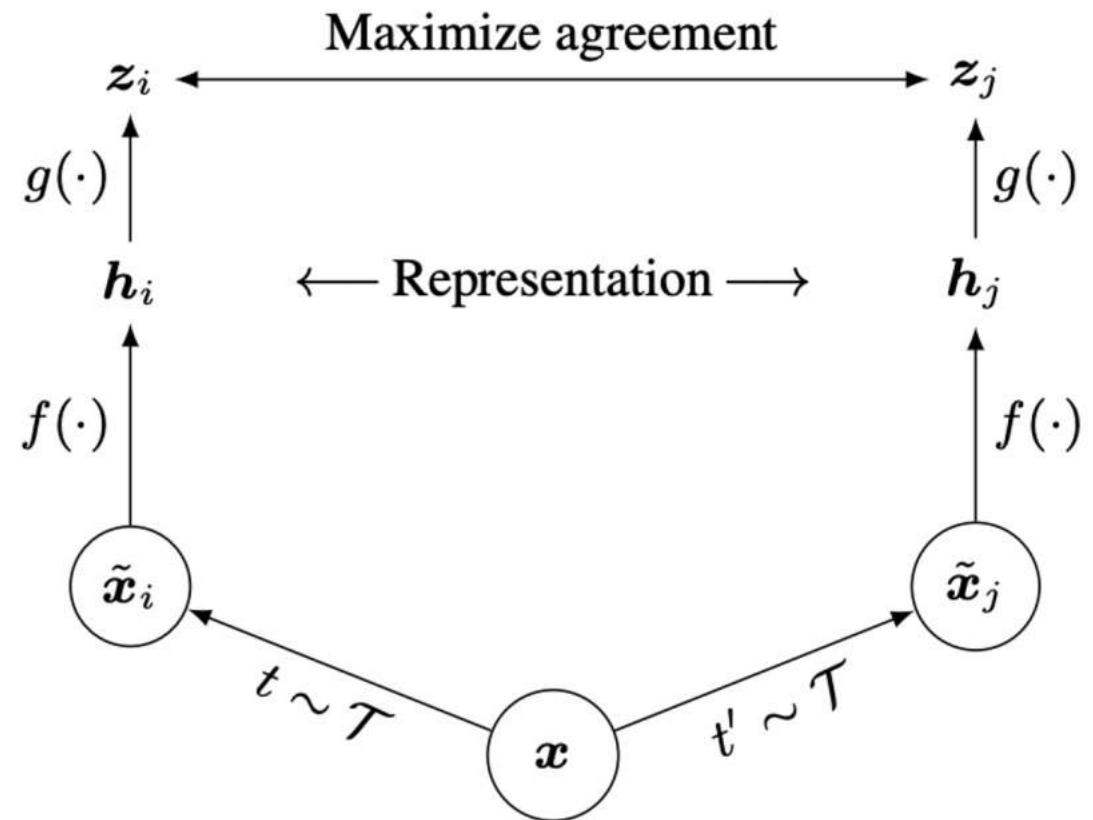
- The above loss corresponds to **cross-entropy loss** in supervised learning.
- In supervised learning, **positive pairs** refer to **the correct output and its corresponding label**.

Simple Contrastive Learning (SimCLR)

- Shared encoder $f(\cdot)$
- Non-linear projection $g(\cdot)$ to compress features further
- Measure similarities

$$s(u, v) = \frac{u^T v}{\|u\| \|v\|}$$

- Construct positive pairs by data augmentation



SimCLR



(a) Original



(b) Crop and resize



(c) Crop, resize (and flip)



(d) Color distort. (drop)



(e) Color distort. (jitter)



(f) Rotate $\{90^\circ, 180^\circ, 270^\circ\}$



(g) Cutout



(h) Gaussian noise

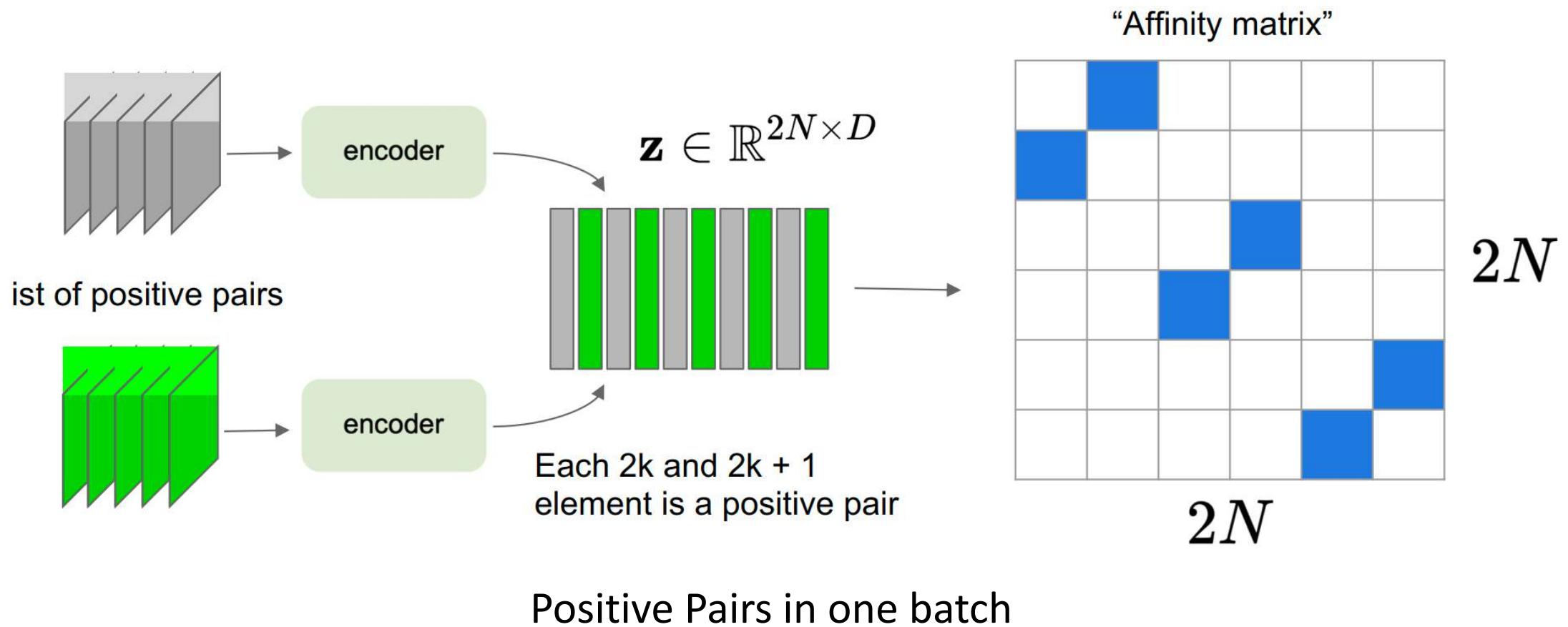


(i) Gaussian blur

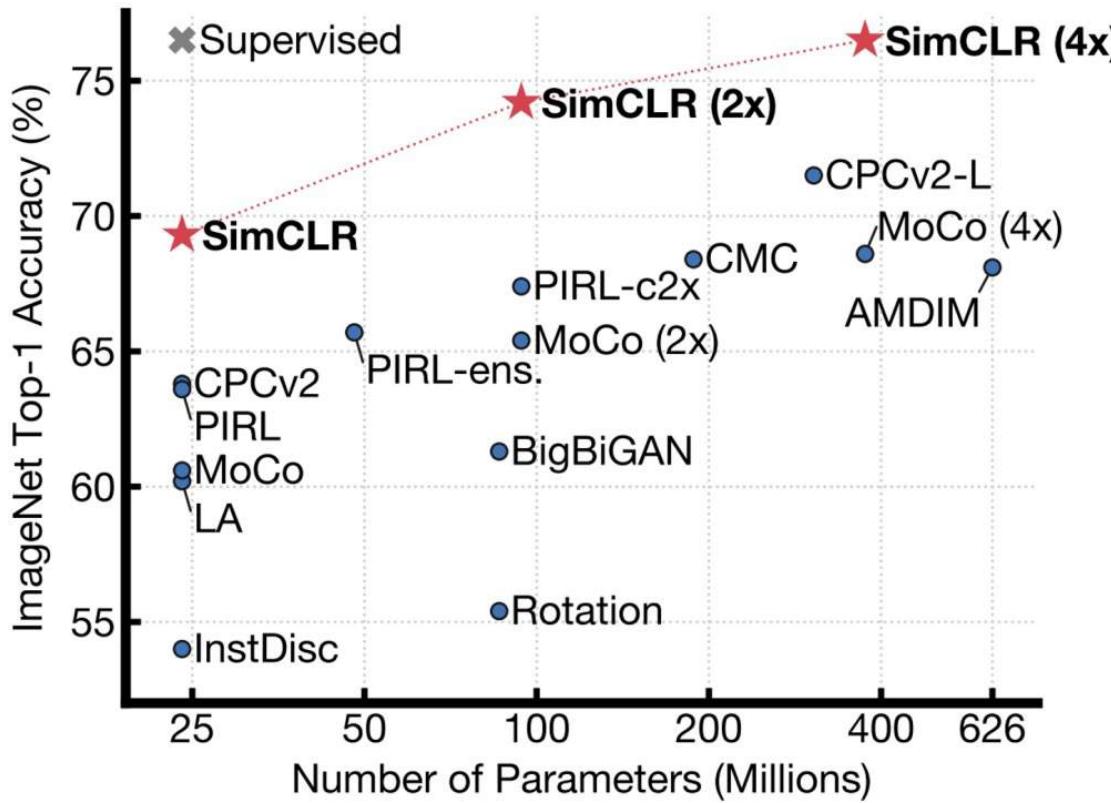


(j) Sobel filtering

SimCLR



SimCLR



Transfer Learning on ImageNet.
(Freeze feature encoder, train a
linear classifier on top with
labeled data.)

SimCLR

- Large batch size (more negative pairs) is beneficial

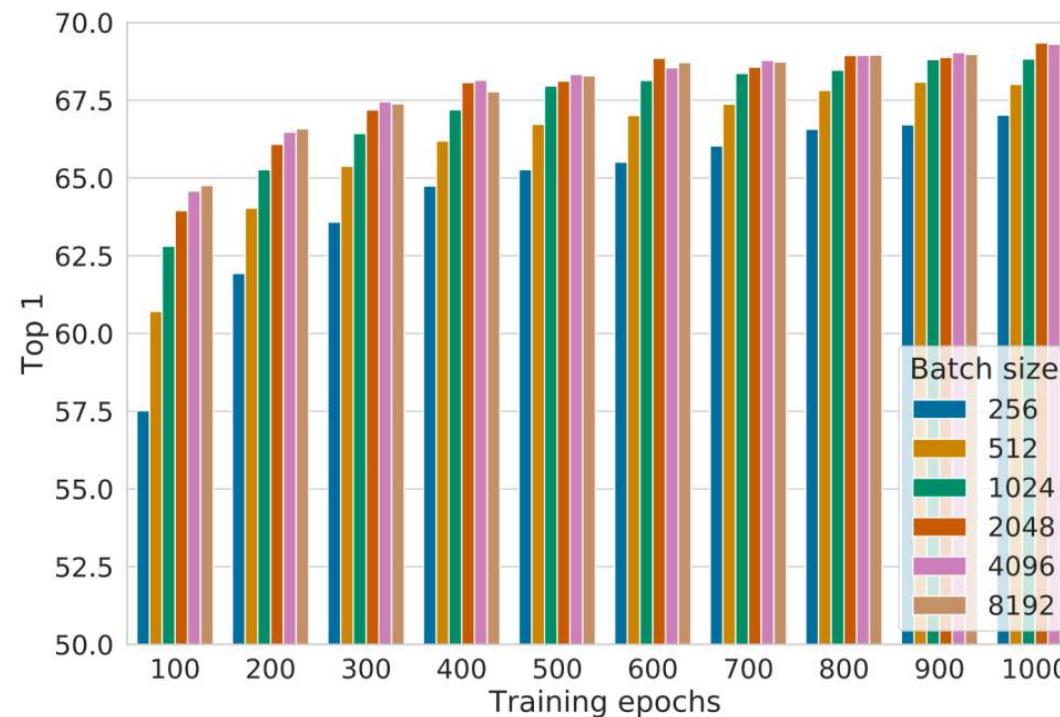


Figure 9. Linear evaluation models (ResNet-50) trained with different batch size and epochs. Each bar is a single run from scratch.¹⁰

Momentum Contrastive Learning (MoCo)

- negative pairs: dynamic queue

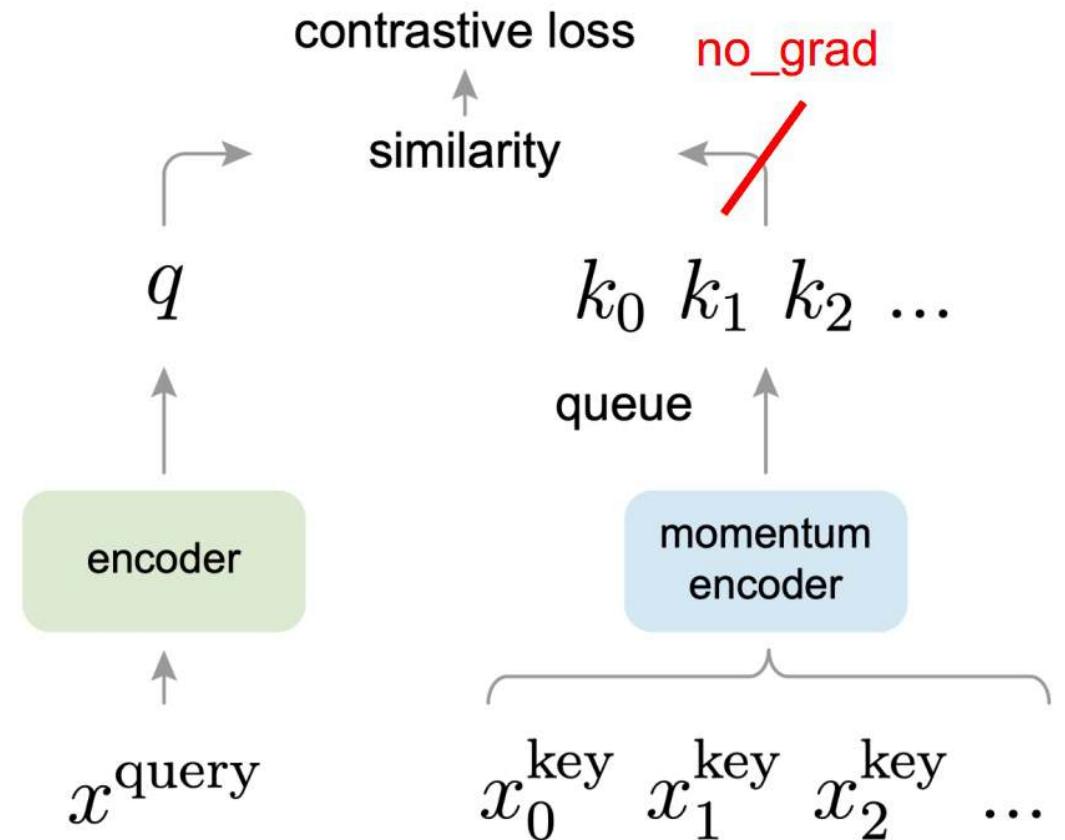
```
enqueue(queue, k) # enqueue the current minibatch  
dequeue(queue) # dequeue the earliest minibatch
```

- Update encoder only via query

- key encoder update: moving average

$$\theta_{key} = m\theta_{key} + (1 - m)\theta_{query}$$

- the number of negative pairs is irrelevant to batch size



MoCo V2

Combination of SimCLR and MoCo

- from SimCLR: non-linear projection+ strong data augmentation
- from MoCo: more negative pairs

case	MLP	unsup. pre-train				ImageNet acc.
		aug+	cos	epochs	batch	
MoCo v1 [6]				200	256	60.6
SimCLR [2]	✓	✓	✓	200	256	61.9
SimCLR [2]	✓	✓	✓	200	8192	66.6
MoCo v2	✓	✓	✓	200	256	67.5
<i>results of longer unsupervised training follow:</i>						
SimCLR [2]	✓	✓	✓	1000	4096	69.3
MoCo v2	✓	✓	✓	800	256	71.1

Table 2. **MoCo vs. SimCLR**: ImageNet linear classifier accuracy

mechanism	batch	memory / GPU	time / 200-ep.
MoCo	256	5.0G	53 hrs
end-to-end	256	7.4G	65 hrs
end-to-end	4096	93.0G [†]	n/a

Table 3. **Memory and time cost** in 8 V100 16G GPUs, implemented in PyTorch. [†]: based on our estimation.

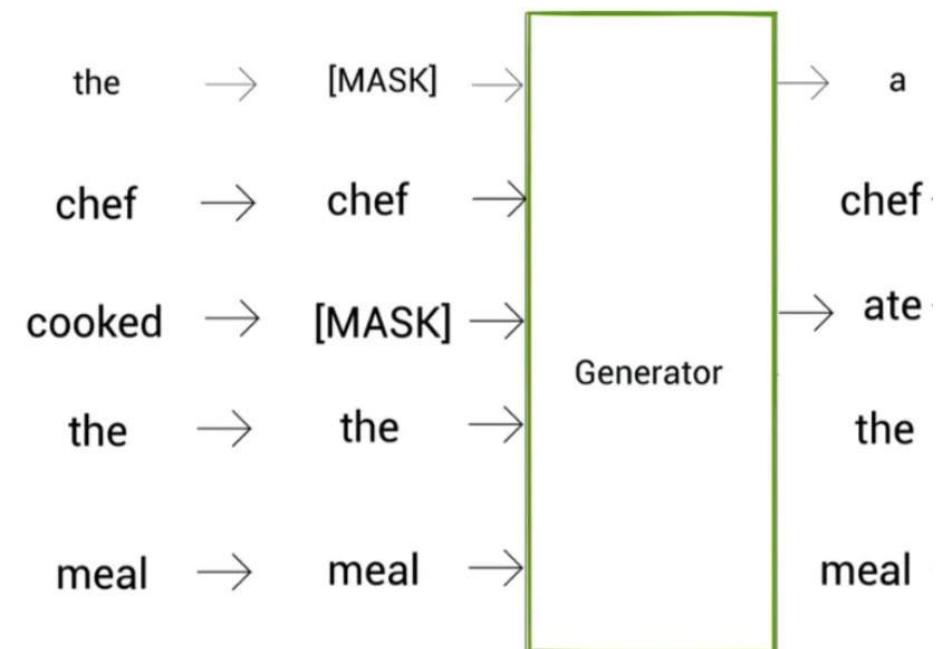
end-to-end =SimCLR

Masking

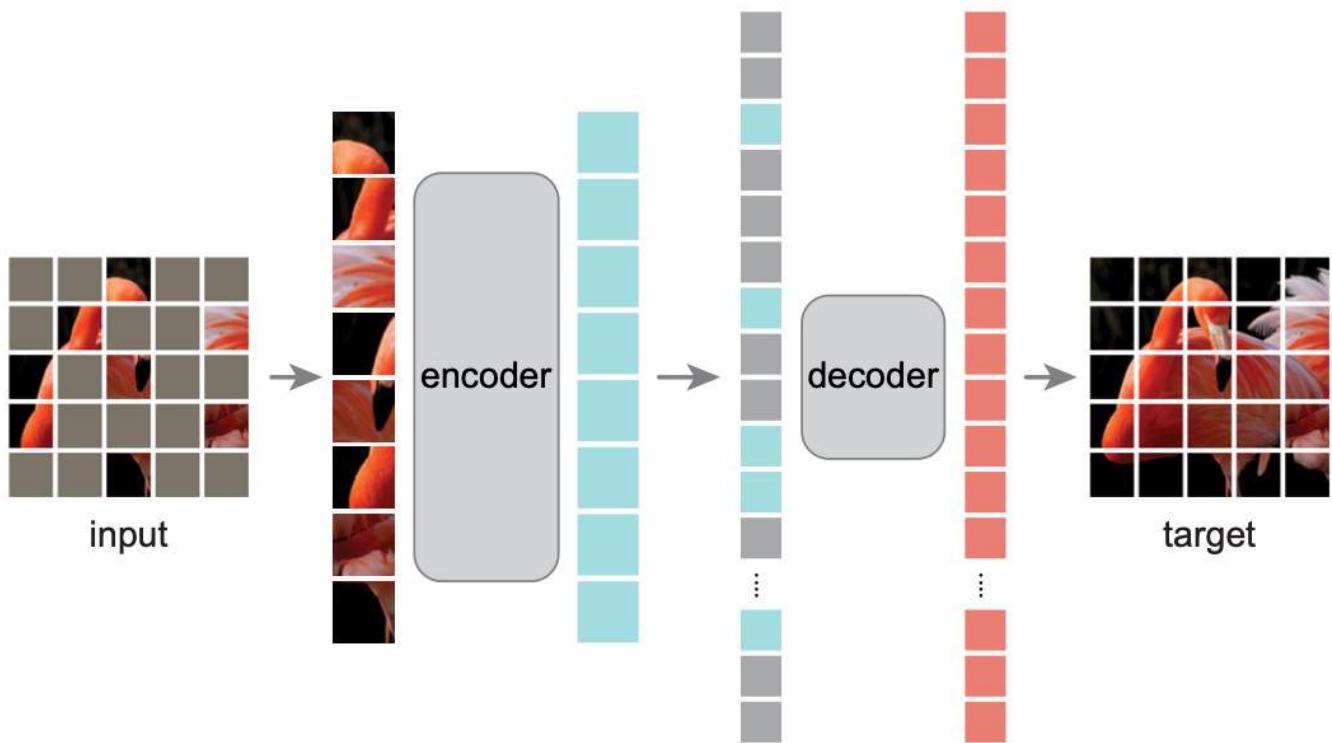
- Masking in **Self-supervised Learning**: hiding parts of the input data and training a model to predict or reconstruct the missing content.
- This forces the model to learn **meaningful internal representations** from context.

Masked-Language Model:

- Predict the masked tokens
- Cross entropy loss (output:a one-hot label vector of the vocabulary size.)

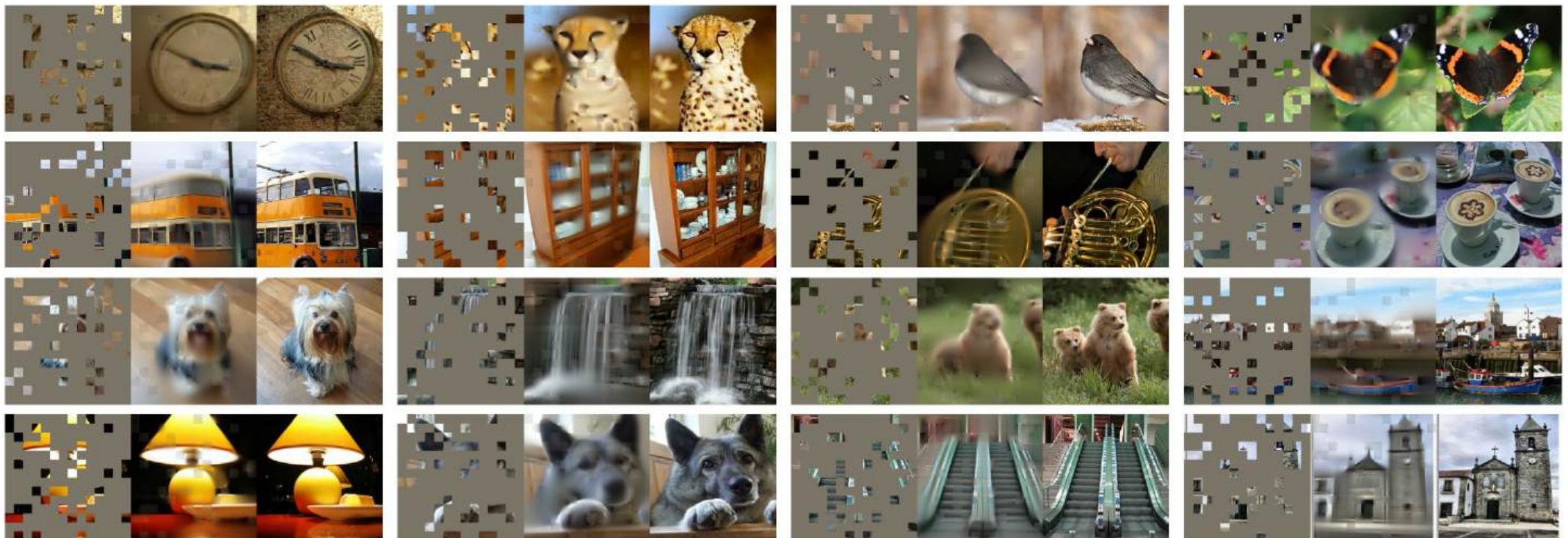


Masked VAE



- Mask 75% of patches
- Use encoder on visible patches only
- Train with MSE loss in pixel space (use visual tokens and cross entropy loss has no advantages)

Masked VAE



Masked Image (Left), Reconstruction(Middle), Truth(Right)

Masked VAE

- Learned Features for Downstream Tasks Perform Better Than End-to-End Supervised Learning

method	pre-train data	AP ^{box}		AP ^{mask}		object detection and segmentation
		ViT-B	ViT-L	ViT-B	ViT-L	
supervised	IN1K w/ labels	47.9	49.3	42.9	43.9	
MoCo v3	IN1K	47.9	49.3	42.7	44.0	
BEiT	IN1K+DALLE	49.8	53.3	44.4	47.1	
MAE	IN1K	50.3	53.3	44.9	47.2	
dataset		ViT-B	ViT-L	ViT-H	ViT-H ₄₄₈	prev best
iNat 2017		70.5	75.7	79.3	83.4	75.4 [55]
iNat 2018		75.4	80.1	83.0	86.8	81.2 [54]
iNat 2019		80.5	83.4	85.7	88.3	84.1 [54] [†]
Places205		63.9	65.8	65.9	66.8	66.0 [19] [†]
Places365		57.9	59.4	59.8	60.3	58.0 [40] [‡]

Classification

Masked VAE

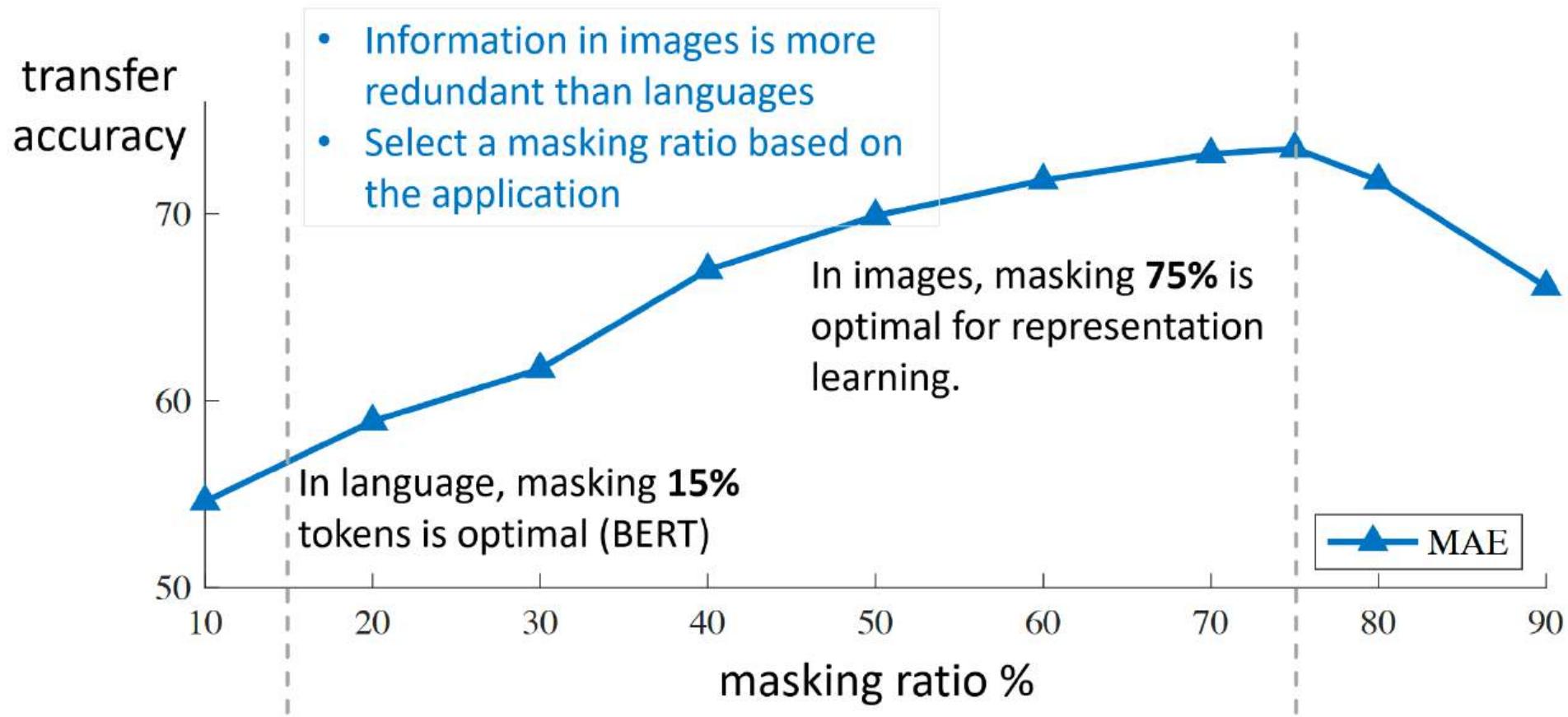
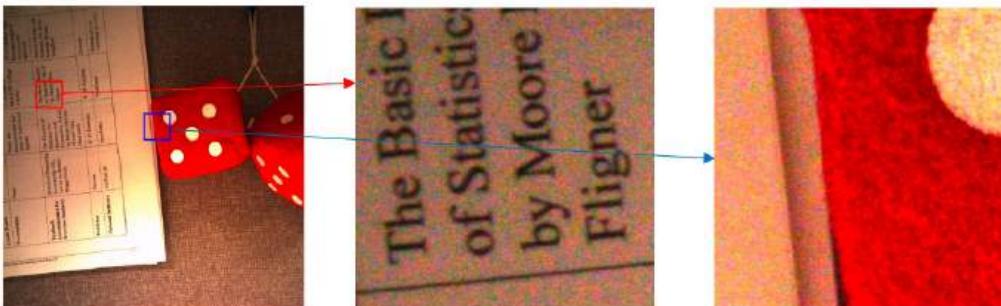


Image Restoration

- Recovering high-quality images from insufficient or degraded measurements, e.g. denoising, deblurring...



Out of focus (Defocus blur)

Unsupervised Restoration

- Supervised learning: $\min \|f_\theta(y) - x\|_2^2$
- Unsupervised learning: only $\{y_i\}$
 - Construct paired data $\{(\hat{y}, \tilde{y})\}$ over which we can define the loss
 - Self-supervised learning with regularization
- Restoration focus on local information rather than high-level semantics. **Even a single image contains numerous small patches that can be used for learning to restore images.**

Denoise

- Noise2Noise: Training over paired noisy data $\{\hat{y}, \tilde{y}\}$

$$\begin{cases} \hat{y} = x + \hat{n} \\ \tilde{y} = x + \tilde{n} \end{cases}, \rightarrow \min \mathbb{E} \|f_\theta(\hat{y}) - \tilde{y}\|_2^2$$

Assume \hat{n} and \tilde{n} are independent, then

$$\begin{aligned} \mathbb{E} \|f_\theta(\hat{y}) - \tilde{y}\|_2^2 &= \mathbb{E} (\|f_\theta(\hat{y}) - x\|_2^2 + 2\tilde{n}^T(f_\theta(\hat{y}) - x) + \|\tilde{n}\|_2^2) \\ &= \mathbb{E} \|f_\theta(\hat{y}) - x\|_2^2 + \text{const} \end{aligned}$$


0

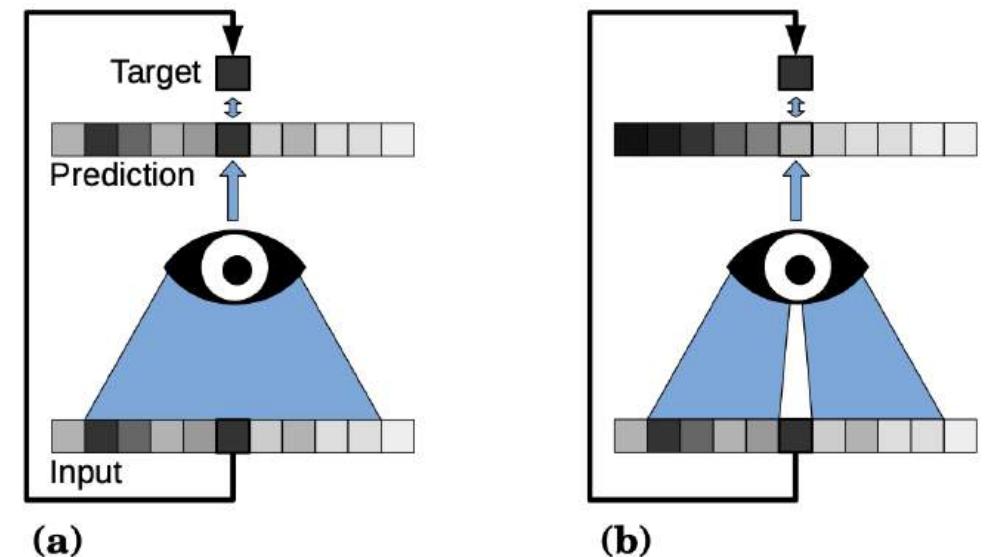
- It is equivalent to supervised training.

Blind-spot

- Noise2Void/Noise2Self: predict the center pixels using the neighboring pixels

Assume n_i ($i = 1, 2, \dots, N$) are independent, then noise in the center pixel y^c and its neighbors y^{bs} are independent.

$$\begin{aligned} & \mathbb{E} \|f_\theta(y^{bs}) - y^c\|_2^2 \\ &= \mathbb{E}(\|f_\theta(y^{bs}) - x^c\|_2^2 + 2(n^c)^T f_\theta(y^{bs}) \\ &+ \|n^c\|_2^2) = \mathbb{E}\|f_\theta(y^{bs}) - x^c\|_2^2 + c. \end{aligned}$$



Masking techniques

Recorrupted-to-Recorrupted(R2R)

- R2R constructs positive pairs from noisy image $y = x + n, n \sim N(0, \sigma^2 I)$
 $z \sim N(0, \sigma^2 I), \hat{y} = y + 2z, \tilde{y} = y - \frac{1}{2}z.$



- For Gaussian noise, zero covariance means independence.

$$\mathbb{E}\|f_\theta(\hat{y}) - \tilde{y}\|_2^2 = \mathbb{E}\|f_\theta(\hat{y}) - x\|_2^2 + \text{const}$$

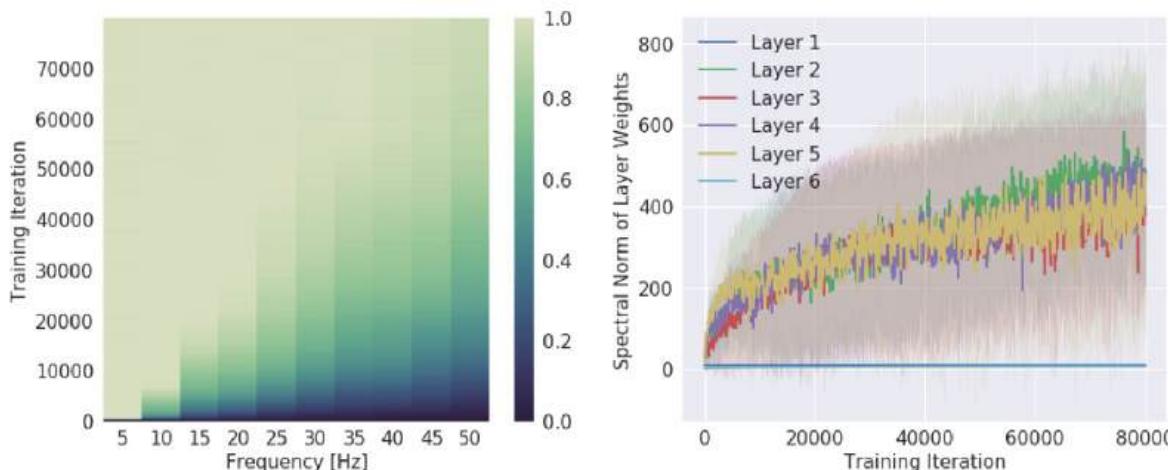
R2R

	Single-image-based Methods				Noisy/Noisy	Noisy/Clean
	BM3D	WNNM	DIP	S2S	N2N	DnCNN
$\sigma = 25$	28.56/0.801	28.80/0.809	27.96/0.774	28.57/0.802	28.86/0.823	29.19/0.830
	Trained on Unpaired Noisy Images					
	N2V	N2S	SURE	Nr2N	Laine <i>et al.</i>	R2R
	27.72/0.794	28.12/0.792	28.94/0.818	28.55/0.808	28.84/0.814	29.14/0.822
$\sigma = 50$	Single-image-based Methods				Noisy/Noisy	Noisy/Clean
	BM3D	WNNM	DIP	S2S	N2N	DnCNN
	25.62/0.687	25.87/0.698	25.04/0.645	25.93/0.698	25.77/0.700	26.22/0.720
	Trained on Unpaired Noisy Images					
	N2V	N2S	SURE	Nr2N	Laine <i>et al.</i>	R2R
	25.12/0.684	25.62/0.678	25.93/0.678	25.61/0.681	25.78/0.698	26.13/0.709

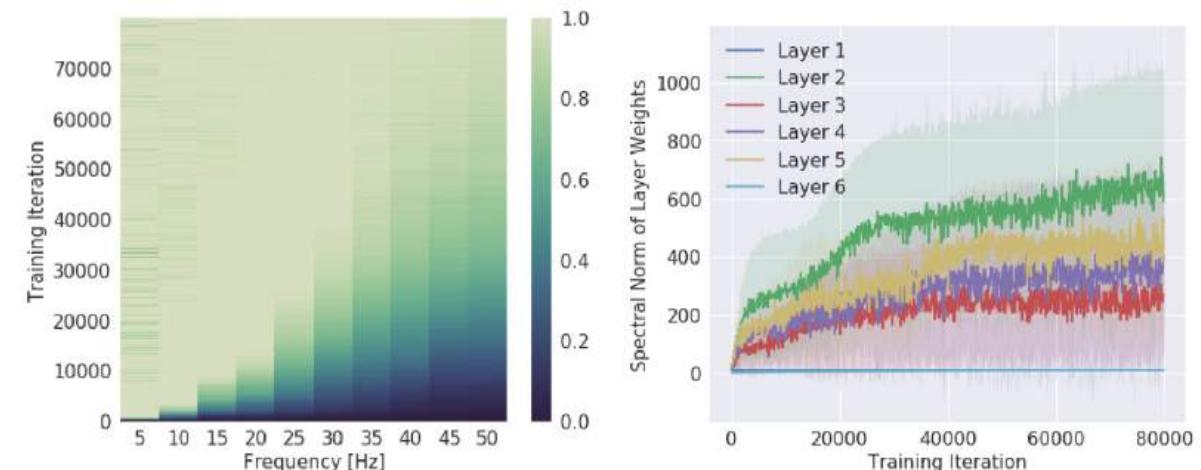
Table: Gaussian denoising results in PSNR(dB)/SSIM on BSD68.

Low Spectral Bias

- Neural networks tend to learn low spectral features firstly.



(a) Equal Amplitudes



(b) Increasing Amplitudes

Train an NN to fit the function: $\lambda(z) = \sum_i A_i \sin(2\pi k_i z + \varphi_i)$.

“On the Spectral Bias of Neural Networks”

Low Spectral Bias

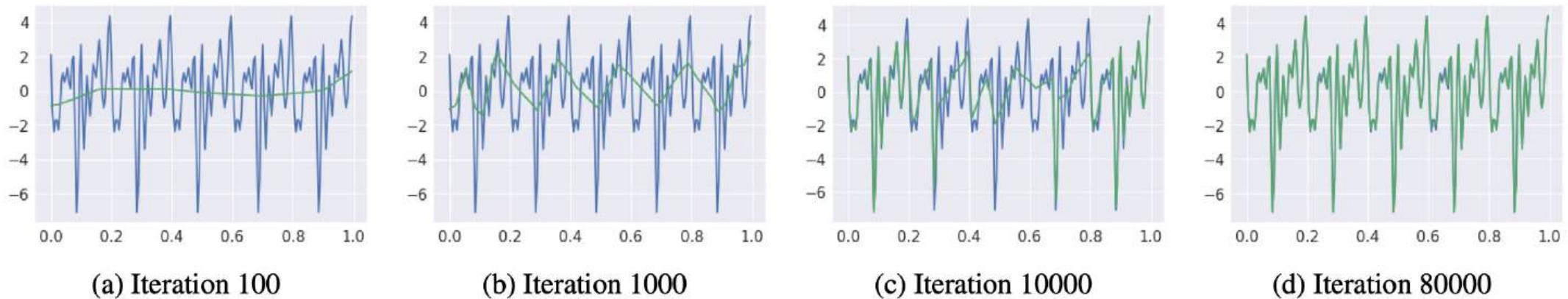
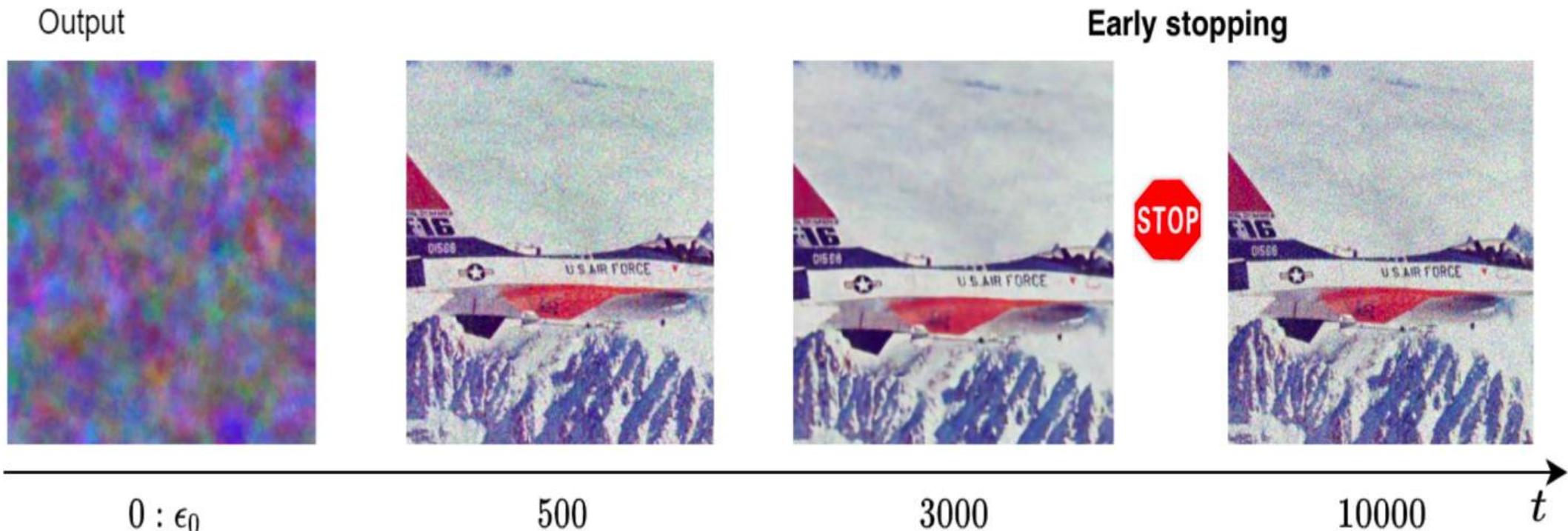


Figure 2. The learnt function (green) overlayed on the target function (blue) as the training progresses. The target function is a superposition of sinusoids of frequencies $\kappa = (5, 10, \dots, 45, 50)$, equal amplitudes and randomly sampled phases.

Deep Image Prior (DIP)

- When fitting an image, NN learns the smooth patterns before noise patterns. Early stopping can avoid overfitting to noise.



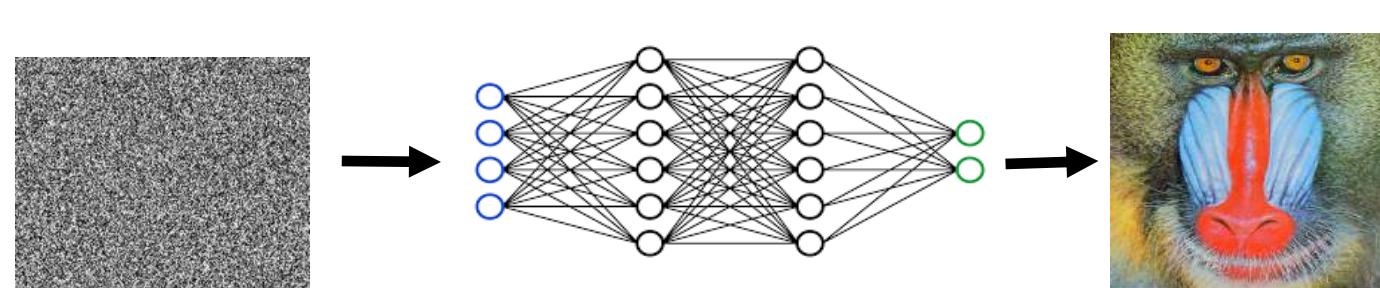
Deep Image Prior

- Image restoration problem: recovering x from $y = \phi(x) + n$
- DIP training loss

$$\min_{\theta} \|\phi(f_{\theta}(\epsilon_0)) - y\|_2^2$$

ϵ_0 : randomly generated latent code

decoder-only:



DIP is trained on single test image. No external training dataset is employed.

Deep Image Prior

Super-resolution

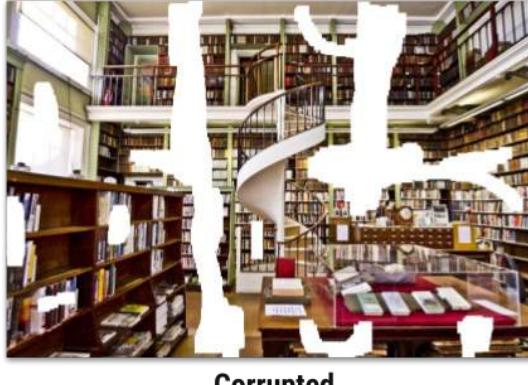


Corrupted

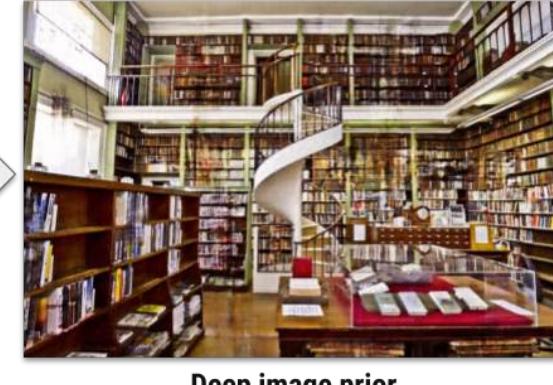


Deep image prior

Inpainting



Corrupted



Deep image prior

Denoising

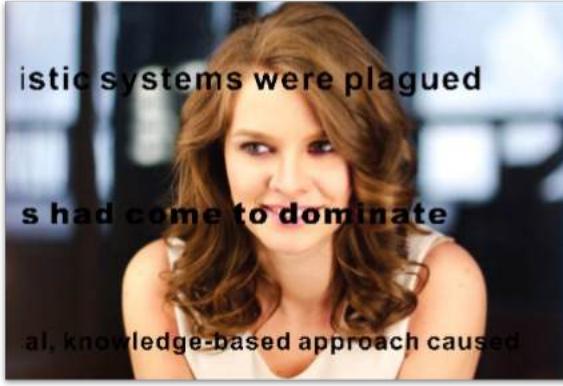


Corrupted



Deep image prior

Inpainting



Corrupted



Deep image prior

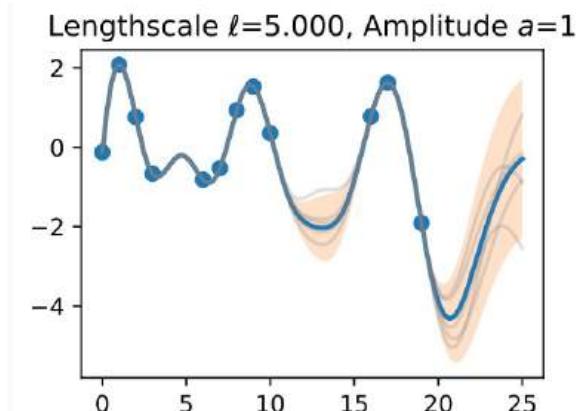
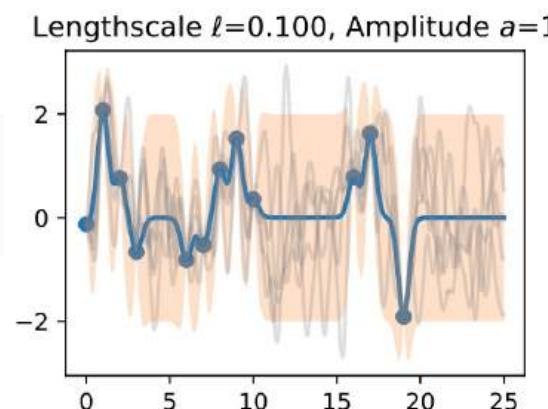
Gaussian Process Prior

- Gaussian Process: $\{f(x), f(x_1), \dots, f(x_n)\}$ follows joint Gaussian distribution for any sampling points.

$$\begin{bmatrix} f(x) \\ f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\mu, \begin{bmatrix} k(x, x) & k(x, x_1) & \dots & k(x, x_n) \\ k(x_1, x) & k(x_1, x_1) & \dots & k(x_1, x_n) \\ \vdots & \vdots & \ddots & \vdots \\ k(x_n, x) & k(x_n, x_1) & \dots & k(x_n, x_n) \end{bmatrix}\right)$$

- $k(x, x')$ controls the smoothness of the function

$$k_{\text{RBF}}(x, x') = \text{Cov}(f(x), f(x')) = a^2 \exp\left(-\frac{1}{2\ell^2} \|x - x'\|^2\right)$$

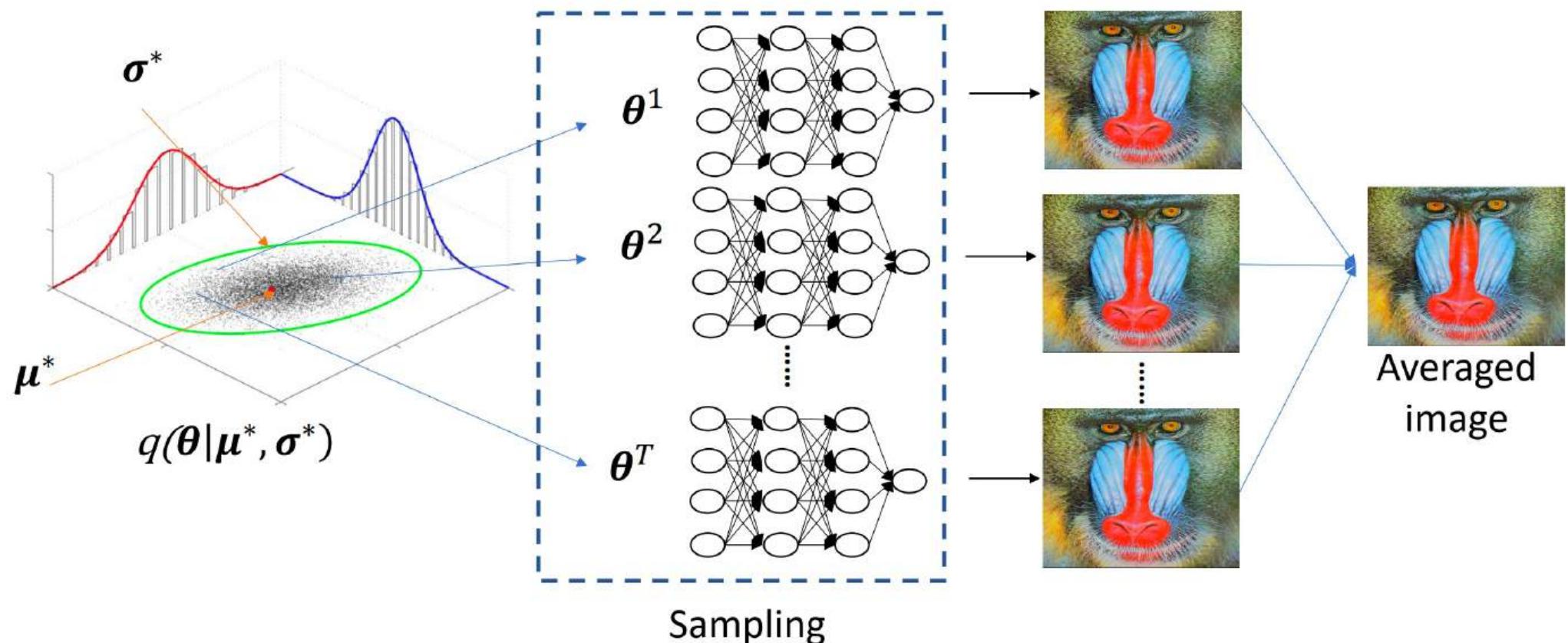


Gaussian Process Prior

- For a very wide NN f_θ , if $\theta \sim N(0, \sigma^2 I)$, then $f_\theta \sim GP$
- Assume a Gaussian prior on θ : $p(\theta) = N(0, \sigma^2 I)$
 - Bayesian inference: MAP(hard to solve here), MMSE...
 - MMSE estimate of x : $\hat{x} = \int x p(x|y) dx = \int f_\theta(x) p(\theta|y) d\theta$
 - $p(\theta|y)$: the posterior distribution of θ ; can be solved via variational inference

$$\begin{aligned}\mu^*, \sigma^* &= \arg \min_{\{\mu, \sigma\}} KL(q(\theta|\mu, \sigma)||p(\theta|y)) \\ \hat{x} &\approx \int f_\theta(x) q(\theta|\mu^*, \sigma^*) d\theta\end{aligned}$$

Gaussian Process Prior



Dimensionality

Reduction

Dimensionality Reduction

- Dimensionality reduction is an unsupervised learning technique: transform high-dimensional data into a **lower-dimensional** representation
 - reduce data complexity while retaining essential information.
 - can learn meaningful representations

Principle Component Analysis (PCA)

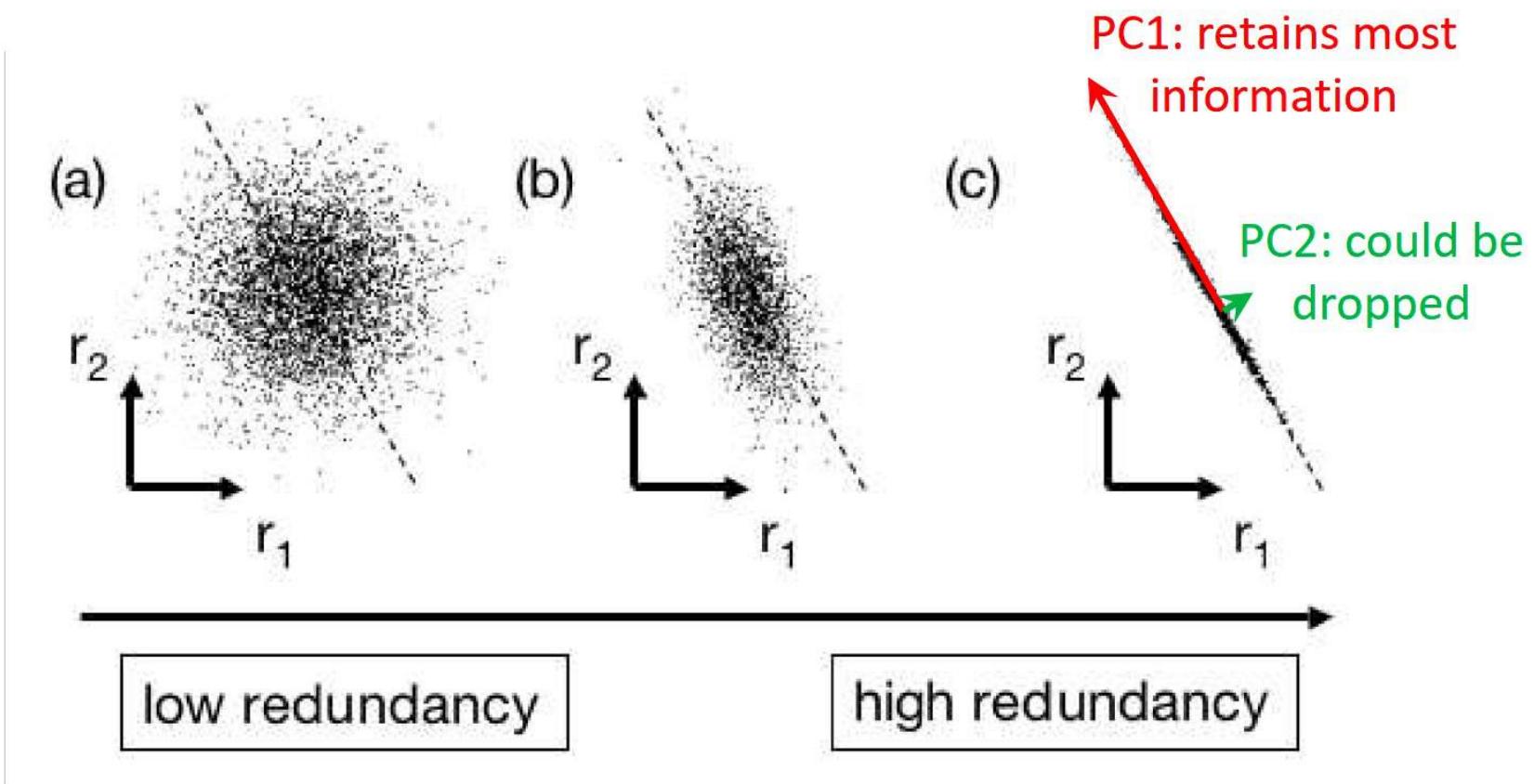


Figure credit: Jonathon Shlens, “A Tutorial on Principal Component Analysis”, 2005

PCA

- **Retaining Only Principal Components**
 - Can be used for denoising.
 - Helps facilitate subsequent high level vision, such as classification.
 - Reduces computational cost.
- **Principal Components:** The direction with maximum variance or the highest information content.
 - Maximum Variance: Principal components capture the most variance in the data.
 - Minimum Dimensionality Reduction Loss: Reducing dimensions while minimizing information loss.

PCA

- Maximum Variance
 - High variance is mainly caused by significant features in the data.
 - Low variance is often associated with noise.
- Given a dataset X with zero mean assumption, we seek a projection w that maximizes the variance of the projected data:

$$w = \arg \max_{\|w\|=1} \text{Var}(w^T X) = \arg \max_{\|w\|=1} w^T X X^T w$$

- w is the eigenvector associated with the largest eigenvalue of the covariance matrix ($V = \frac{1}{n} X X^T$), which can be solved numerically via power iteration: $u^{k+1} = \frac{V u^k}{\|V u^k\|}$
- W/o zero mean assumption, decentralize: $V = \frac{1}{n} (X - EX)(X - EX)^T$

PCA

- Minimum Dimensionality Reduction Loss: represent the original data x using only the first k principal components while minimizing the loss of information.

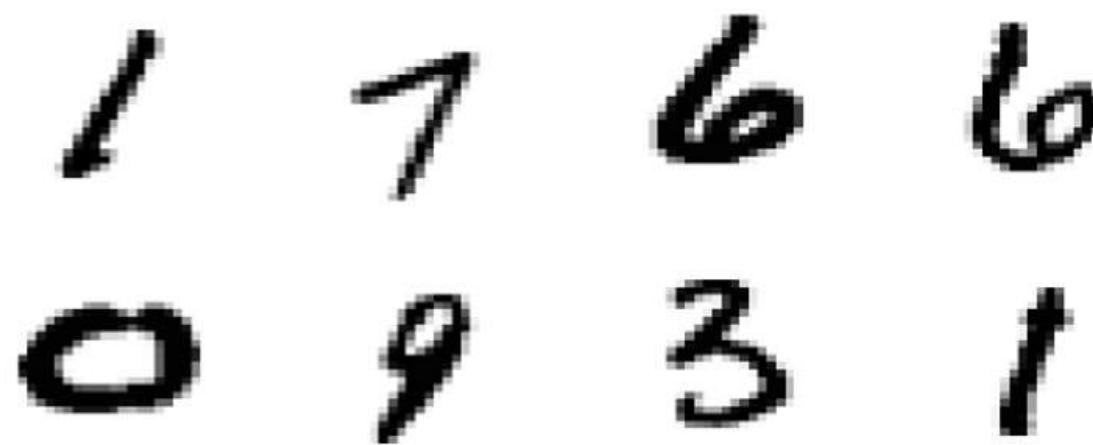
$$\min_W \sum_x ||x - W^T W x||_2^2$$

s.t. $W W^T = I_{k \times k}$

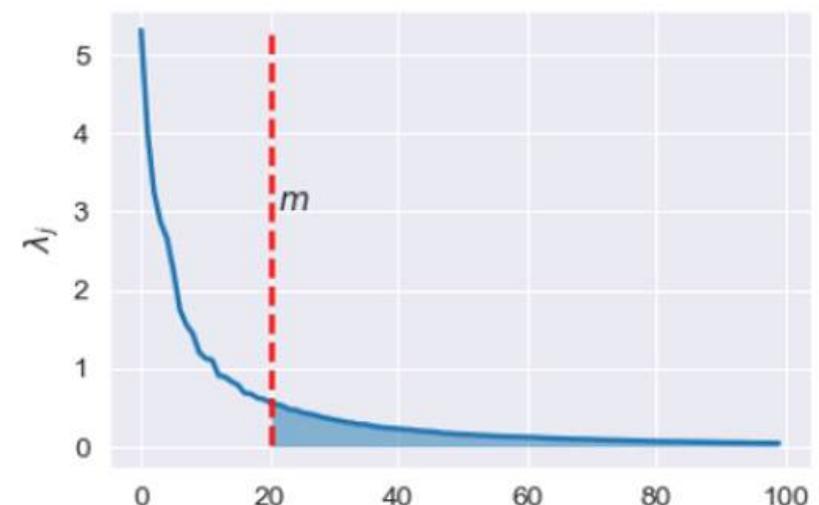
$$||x - W^T W x||_2^2 = x^T x - (Wx)^T (Wx)$$

$$\min_x \sum_x ||x - W^T W x||_2^2 \leftrightarrow \max_W \sum_x ||Wx||_2^2 = \sum_{i=1}^k ||w_i X||_2^2$$

PCA



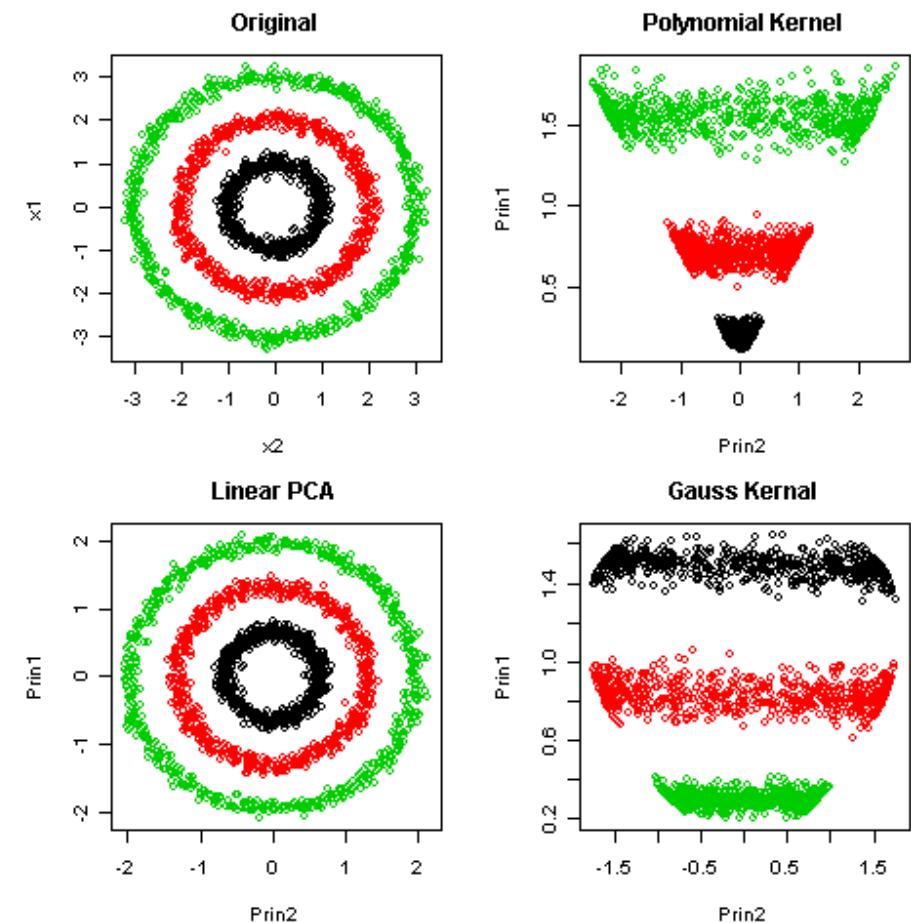
Principle components of MNIST.



Effective dimension = 20

Kernel PCA

- Non-linear dimensionality reduction: $\Phi = (\phi_i(x_j))$, $V = \frac{1}{n} (\Phi - \mathbb{E}\Phi)(\Phi - \mathbb{E}\Phi)^T$, find the top k eigenvectors for projection
- Kernel PCA: $V_{ij} = k(x_i, x_j)$

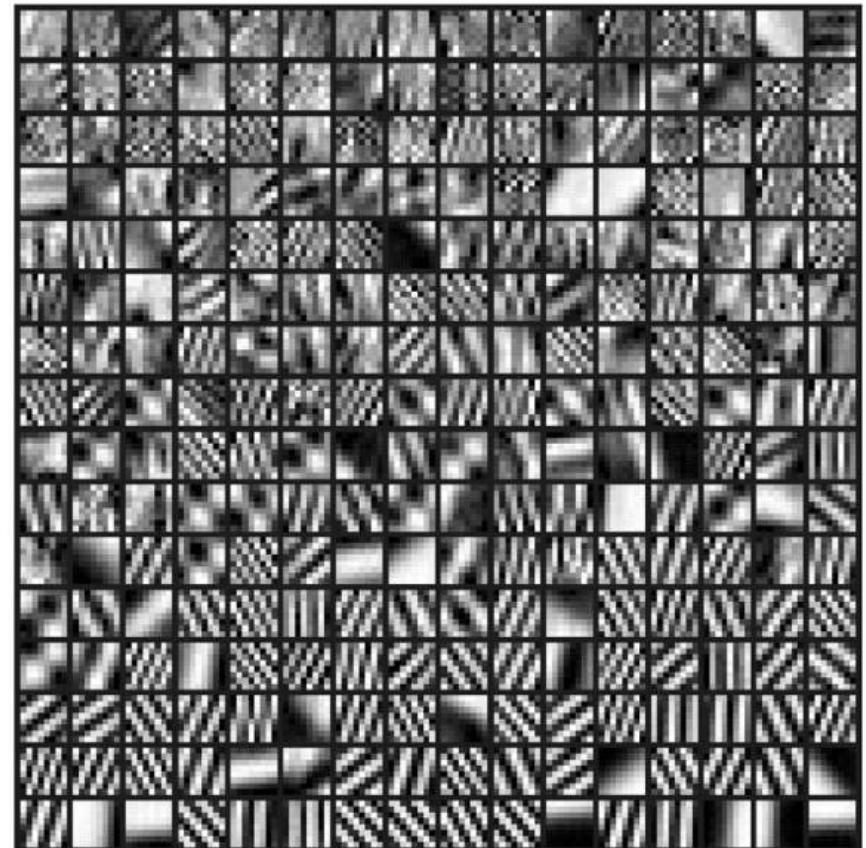


Dictionary Learning

- Learn Dictionary for Sparse Representation:

$$\min_{D, \{c_k\}_{k=1}^p} \sum_{k=1}^p \frac{1}{2} \|y_k - D c_k\|_2^2 + \lambda \|c_k\|_0$$
$$s.t. \|d_k\|_2^2 = 1, k = 1, 2, \dots, p$$

ℓ_0 -norm regularization purses sparsity.



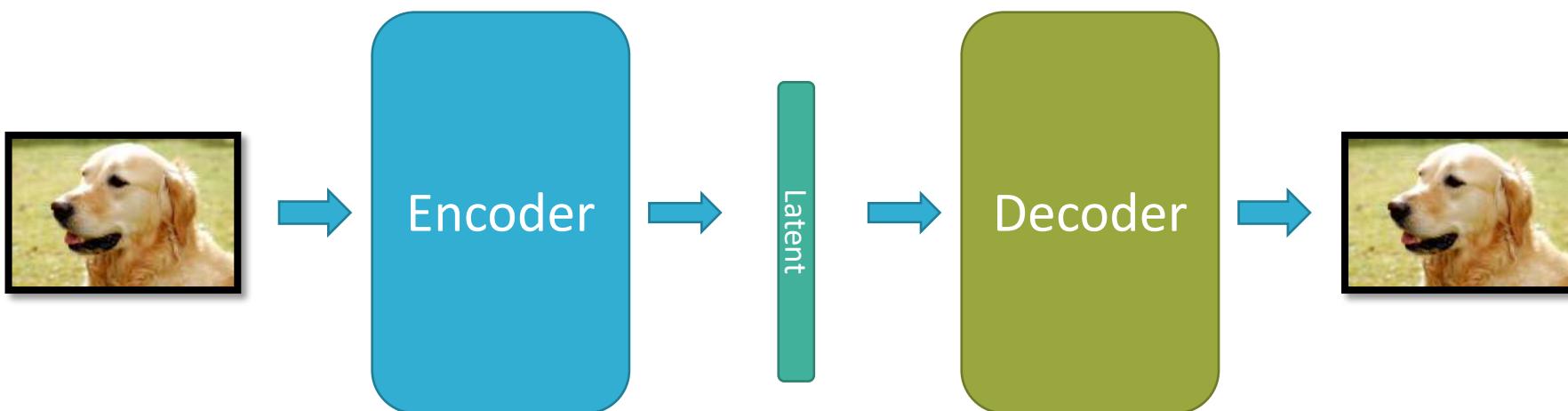
Learned dictionary for images

Autoencoders (AE)

PCA:

$$Z_k = W_k X$$

$$X' = W_k^T Z_k$$



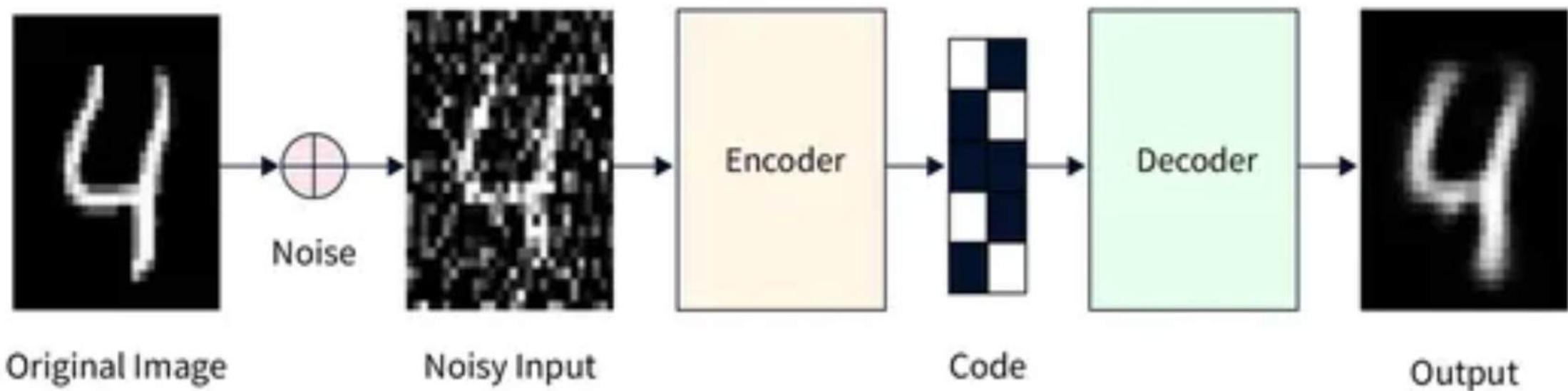
AE:

$$Z_k = E(X; \theta)$$

$$X' = D(Z_k; \phi)$$

$$\text{AE: } \min_{\theta} \sum_x ||X - D(E(X))||_2^2$$

Autoencoders



Autoencoders

- K-means, PCA, Dictionary Learning are like autoencoders

$$\text{K-means: } \min_{c,i} \|x - c(i(x))\|_2^2$$

$$\text{PCA: } \min_{W: WW^T = I_k} \sum_x \|x - W^T W x\|_2^2$$

$$\text{AE: } \min_{\theta} \sum_x \|X - D(E(X))\|_2^2$$

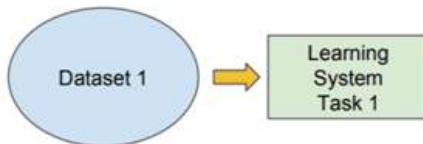
$$DL: \min_D \frac{1}{2} \|Y - Dc\|_2^2 + \lambda \|c\|_0 \text{ (decoder only)}$$

Transfer Learning

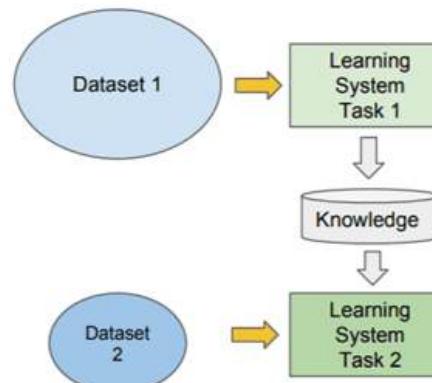
- **Transfer learning:** taking a model trained on one task (usually with a large dataset) and adapting it to a different, but related task.

Traditional ML vs Transfer Learning

- Isolated, single task learning:
 - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

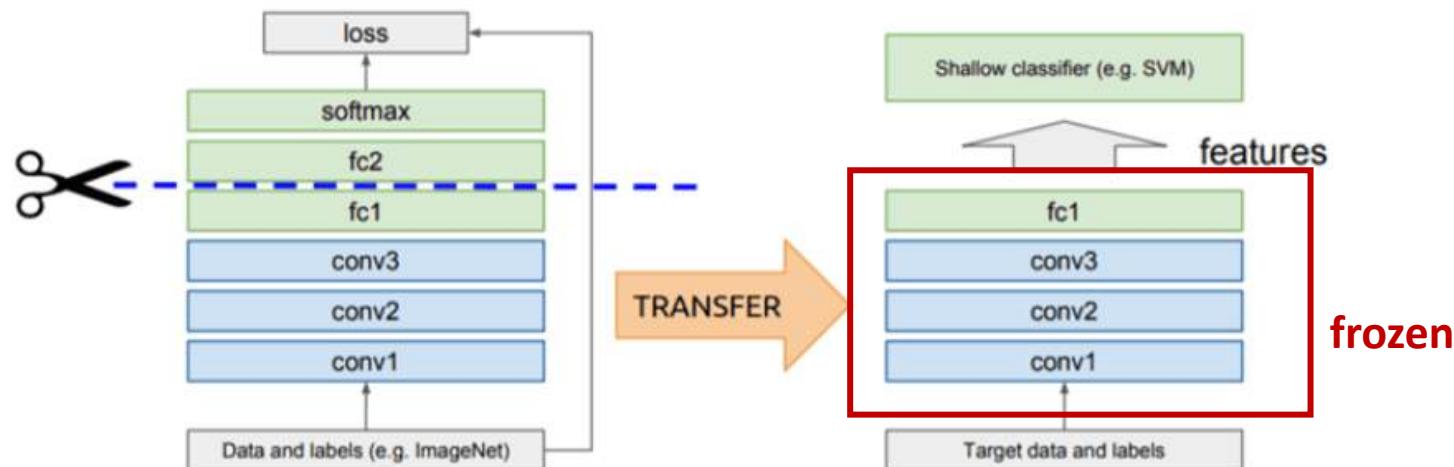


- Learning of a new tasks relies on the previous learned tasks:
 - Learning process can be faster, more accurate and/or need less training data



Strategies for Transfer Learning

- **Feature Extraction:** Leveraging rich representations from big models (e.g., MoCo, BERT, Masked VAE) and only train a new task(e.g. classification, segmentation) head.

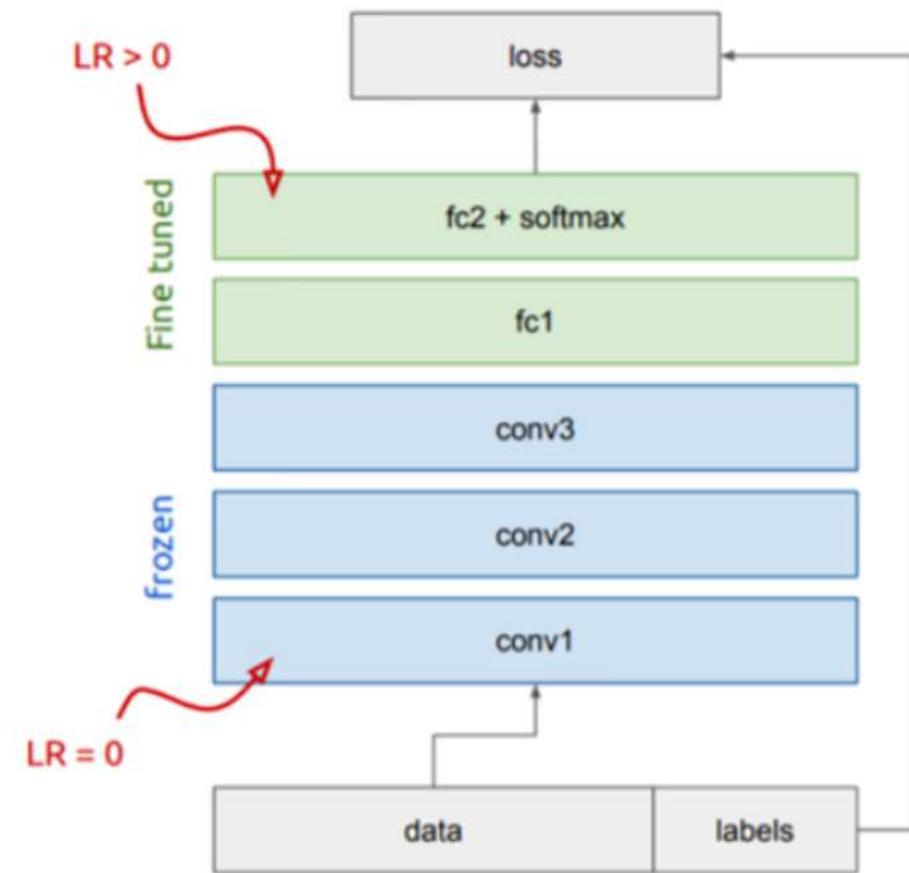


Transfer Learning with Pre-trained Deep Learning Models as Feature Extractors

Strategies for Transfer Learning

- **Fine-Tuning:**

- Initialize with pretrained weights
- Unfreeze some or all layers
- Train with lower learning rate.



Strategies for Transfer Learning

- **Adaptor: LoRA (Low-Rank Adaptation)**

$$W_0 + \Delta W = W_0 + BA, \text{ where } \hat{B} \in \mathbb{R}^{d \times r}, \hat{A} \in \mathbb{R}^{r \times k}$$

