

Unsupervised Learning & Semi-supervised Learning

AI602: Recent Advances in Deep Learning

Lecture 13

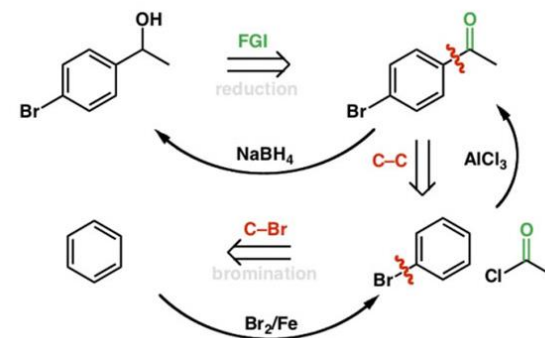
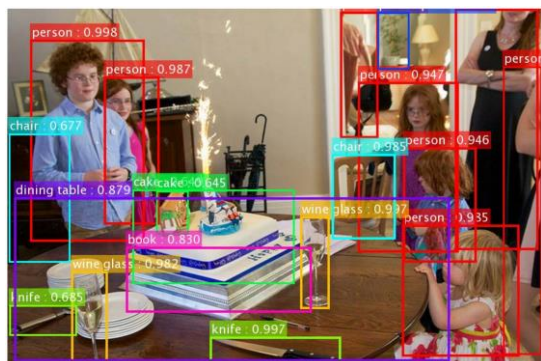
Slide made by

Hankook Lee

KAIST EE

Motivation

- DNNs achieve **remarkable success** on various applications
 - They usually **require massive amounts of manually labeled data**
 - The **annotation cost is high** because
 - It is **time-consuming**: e.g., annotating bounding boxes
 - It requires **expert knowledge**: e.g., medical diagnosis and retrosynthesis



- But, **collecting unlabeled samples is extremely easy** compared to annotation
- **Question:** How to utilize the unlabeled samples for learning?

- **Supervised learning**

$$\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\} \subseteq \mathcal{X} \times \mathcal{Y}$$

- \mathcal{X} and \mathcal{Y} are input and label spaces, respectively
- All training samples have ground-truth labels

- **Semi-supervised learning**

$$\mathcal{D}_l = \{(x^{(1)}, y^{(1)}), \dots, (x^{(n)}, y^{(n)})\} \subseteq \mathcal{X} \times \mathcal{Y}$$

$$\mathcal{D}_u = \{x^{(1)}, \dots, x^{(m)}\} \subseteq \mathcal{X}$$

- Only few samples have ground-truth labels, i.e., $m \gg n$
- **Q)** What is the difference from *weakly-supervised learning*?

- **Unsupervised learning** (or representation learning)

$$\mathcal{D} = \{x^{(1)}, \dots, x^{(m)}\} \subseteq \mathcal{X}$$

- No labeled samples, i.e., **no target task is defined**
- The goal is learning **good representations**
- **Q)** How to measure the quality of representations?

1. **Semi-supervised Learning**

- Entropy Minimization & Pseudo Labeling
- Consistency Regularization

2. **Unsupervised Learning**

- Self-supervised learning
- Clustering-based approaches
- Maximizing mutual information

1. **Semi-supervised Learning**

- Entropy Minimization & Pseudo Labeling
- Consistency Regularization

2. **Unsupervised Learning**

- Self-supervised learning
- Clustering-based approaches
- Maximizing mutual information

- **EntMin** [Grandvalet & Bengio, 2005]

- **Idea:** Make uncertain predictions of unlabeled samples be more certain

$$H(f(x; \theta)) = - \sum_k f(x; \theta)_k \log f(x; \theta)_k$$

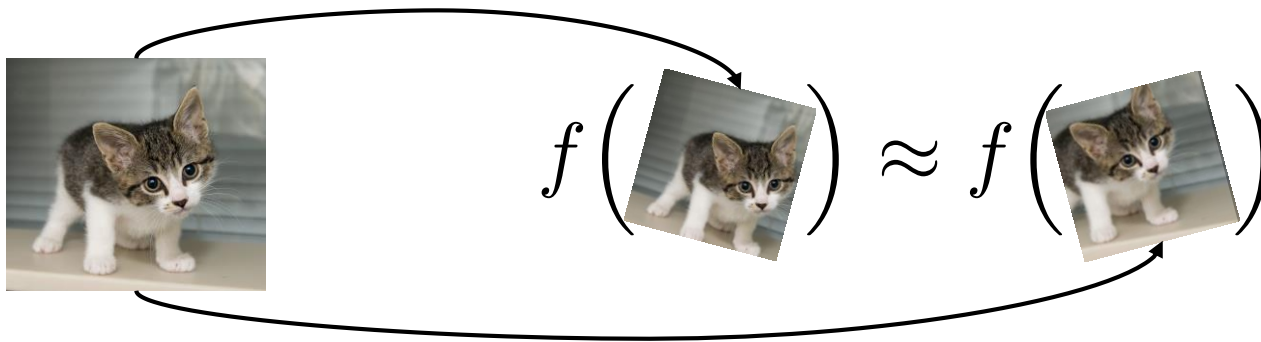
- Note that high-capacity models might overfit to a low-entropy solution is available
- EntMin cannot produce competitive results, but **it can be combined with others**

- **Pseudo-labeling** [Lee, 2013]

- **Idea:** Assign a pseudo-label if confidence of prediction is greater than a threshold
- Similar to EntMin, but it encourages only confident samples to be more confident
- Pseudo-labeling is closely related to learning on noisy datasets [Tanaka et al., 2018]

- **Extensive experimental analysis** on semi-supervised methods [Oliver et al., 2018]

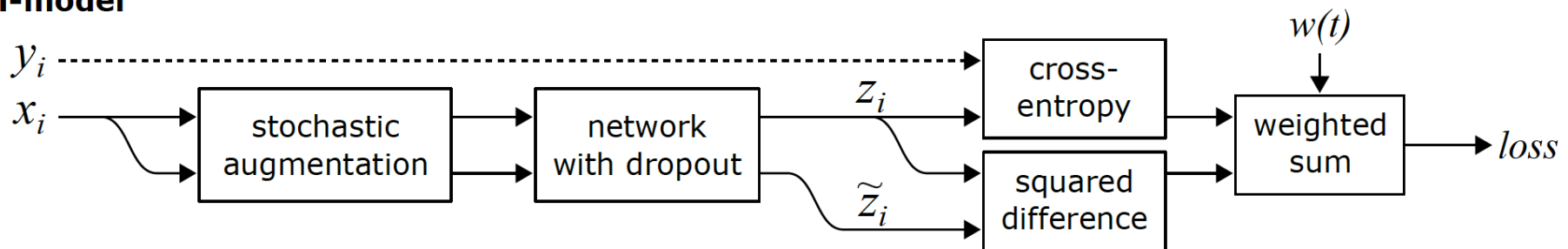
- **Idea:** Models should produce similar outputs for similar samples
- **Q)** How to know the samples are similar? How to generate similar samples?
- **A)** Use random data augmentation & stochastic networks



- For all samples, add a **regularization** term enforcing the **consistent** property
 - Temporal Ensembling [Laine & Aila, 2017]
 - Mean Teacher [Tarvainen & Harri Valpola, 2017]
 - Virtual Adversarial Training [Miyato et al., 2017]

- **Π -Model** [Laine & Aila, 2017]

Π -model



1. For each input x , construct two **stochastic** outputs z and \tilde{z}
2. If its label y exists, then add the standard supervised loss, e.g., cross-entropy
3. Add the **squared difference loss** $\|z - \tilde{z}\|^2$

$$\mathcal{L}_{\text{total}} = -\frac{1}{|B|} \sum_{i \in B \cap L} \mathcal{L}(z_i, y_i) + \frac{w(t)}{C|B|} \sum_{i \in B} \|z_i - \tilde{z}_i\|^2$$

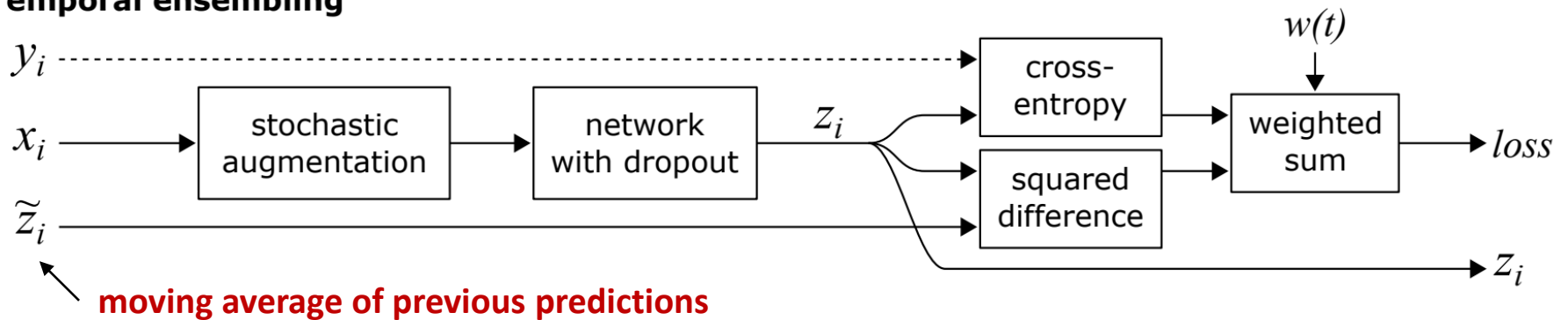
of classes

- $w(t) = \exp(-5(1 - t)^2)$ is a **time-dependent** weighting function
 - In the beginning ($t = 0$), \tilde{z} has no meaningful information, thus the ramp-up of the weight should be slow enough

Semi-supervised Learning: Consistency Regularization

- In Π -Model
 - The target \tilde{z} is generated randomly \Rightarrow It can be noisy
 - For each iteration, \tilde{z} should be computed \Rightarrow 2x longer training
- **Temporal Ensembling** [Laine & Aila, 2017]
 - Idea: **Ensemble the predictions** at the previous epochs

Temporal ensembling



- **Key difference** from Π -Model: A maintaining strategy of targets \tilde{z}
- Compute the **moving average of prediction** z_i for each i -th sample

$$Z \leftarrow \alpha Z + (1 - \alpha) z \quad \text{moving average}$$

$$\tilde{z} \leftarrow Z / (1 - \alpha^t) \quad \text{bias correction}$$

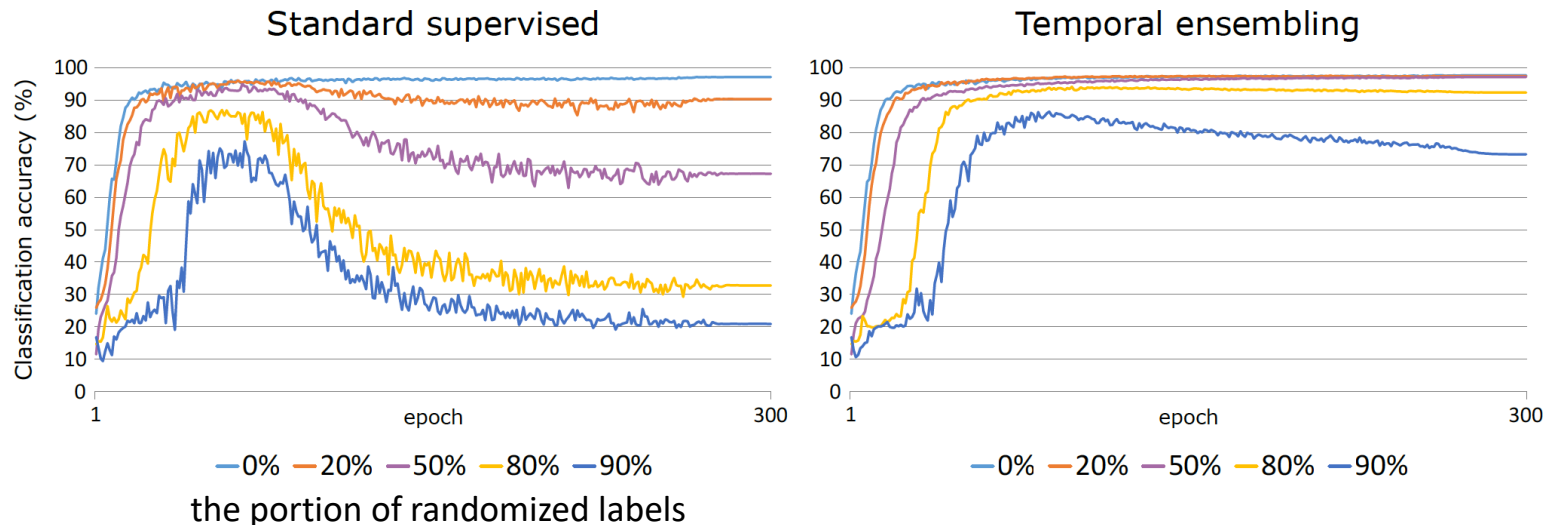
Semi-supervised Learning: Consistency Regularization

- Π -Model and Temporal ensembling **improve semi- & fully-supervised settings**

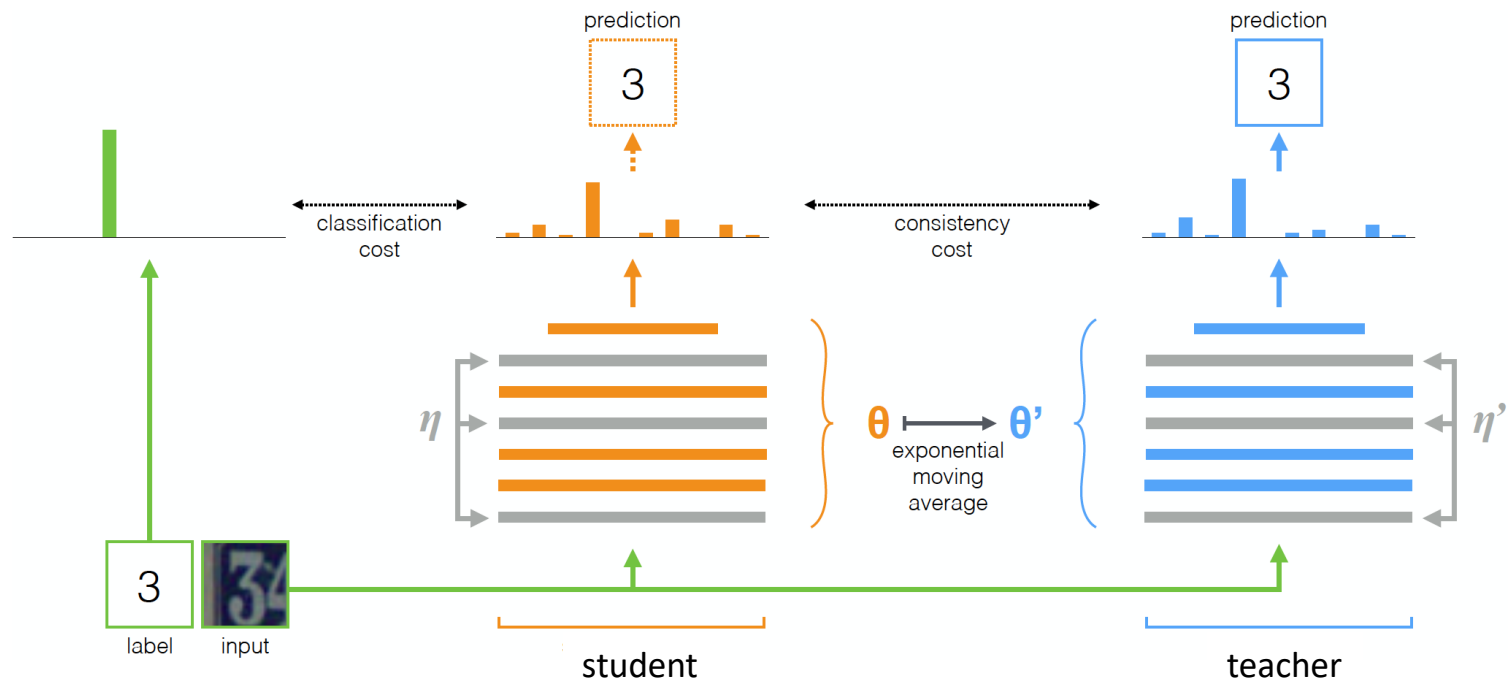
Table 3: CIFAR-100 results with 10000 labels, averages of 10 runs (4 runs for all labels).

	Error rate (%) with # labels	
	10000	All (50000)
Supervised-only	51.21 ± 0.33	29.14 ± 0.25
with augmentation	44.56 ± 0.30	26.42 ± 0.17
Π -model	43.43 ± 0.54	29.06 ± 0.21
Π -model with augmentation	39.19 ± 0.36	26.32 ± 0.04
Temporal ensembling with augmentation	38.65 ± 0.51	26.30 ± 0.15

- **Tolerance to incorrect labels**



- **Limitations** on Temporal Ensembling
 - For each sample, \tilde{z} is **updated only once per epoch**
 - In **on-line learning**, how to maintain the average of predictions?
- **Mean Teacher** [Tarvainen & Harri Valpola, 2017]
 - Idea: **averaging model weights** instead of predictions



- **Limitations** on Temporal Ensembling
 - For each sample, \tilde{z} is **updated only once per epoch**
 - In **on-line learning**, how to maintain the average of predictions?

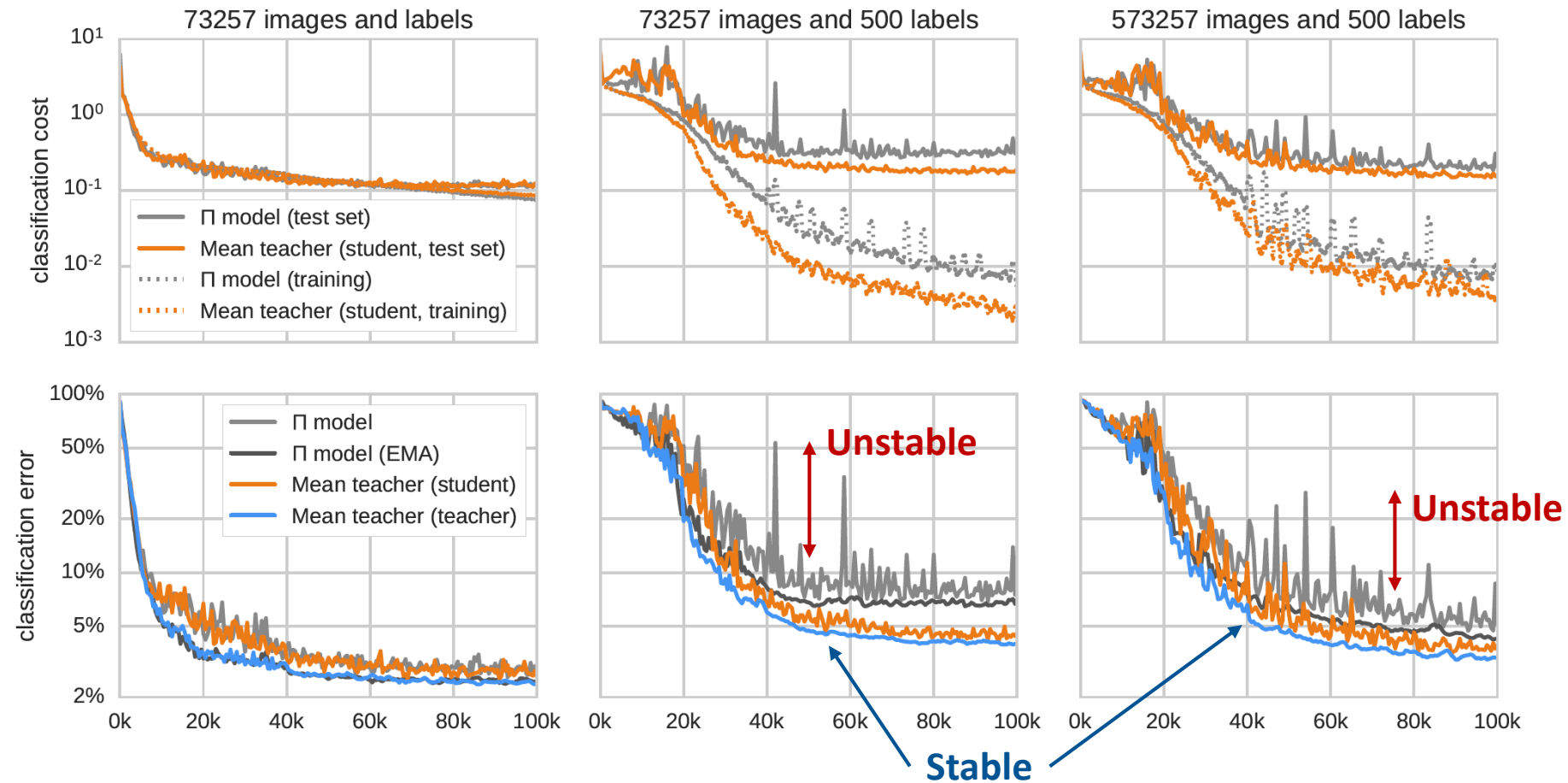
- **Mean Teacher** [Tarvainen & Harri Valpola, 2017]
 - **Idea: averaging model weights** instead of predictions

$$\mathcal{L}_{\text{unlabeled}} = \|f(x, \theta') - f(x, \theta)\|^2$$

$$\theta'_t = \alpha'_{t-1} + (1 - \alpha)\theta_t$$

- **Advantages**
 - More **accurate targets** \tilde{z} can be constructed from **a faster feedback**
 - **Scalability** to large datasets and on-line learning

- **Mean teacher** enables stable training



- **Virtual Adversarial Training** [Miyato et al., 2017]

- **Motivation:** How to choose better similar samples for consistency regularization?
 - Previous methods use randomly augmented samples

- **Idea:** Select an adversarial sample which can most greatly alter the outputs

1. Find the adversarial sample

$$r_{\text{vadv}}(x) := \arg \max_{r; \|r\|_2 \leq \epsilon} D \left[p(y|x, \hat{\theta}), p(y|x + r, \hat{\theta}) \right]$$

- D be a divergence between two distributions
- $\hat{\theta}$ be the current parameters, but it is considered as constant

- **Q)** How to solve the optimization problem?

- **A)** Use the second-order Taylor approximation at $r = 0$

$$D(r) \approx D(0) + \underbrace{r^\top \nabla_r D|_{r=0}}_{\text{always zero by definition}} + \frac{1}{2} r^\top H r = \frac{1}{2} r^\top \underset{\substack{\uparrow \\ \text{Hessian matrix}}}{H} r$$

always zero by definition

Hessian matrix

- **Virtual Adversarial Training** [Miyato et al., 2017]

- **Motivation:** How to choose better similar samples for consistency regularization?
 - Previous methods use randomly augmented samples

- **Idea:** Select an adversarial sample which can most greatly alter the outputs

1. Find the adversarial sample

$$r_{\text{adv}}(x) \approx \arg \max_{r; \|r\|_2 \leq \epsilon} r^\top H r = \epsilon \frac{u}{\|u\|}$$

- u be the first dominant eigenvector of the Hessian matrix
- The power iteration method: $d \leftarrow \frac{Hd}{\|Hd\|}$ converges to u
- The finite difference method:

$$Hd \approx \frac{\nabla_r D|_{r=\xi d} - \nabla_r D|_{r=0}}{\xi} = \frac{\nabla_r D|_{r=\xi d}}{\xi}$$

↖ We can compute this!

- Only **one iteration** is enough (no improvement when using more iterations)

- **Virtual Adversarial Training** [Miyato et al., 2017]

- **Motivation:** How to choose better similar samples for consistency regularization?
 - Previous methods use randomly augmented samples

- **Idea:** Select an adversarial sample which can most greatly alter the outputs

1. Find the adversarial sample

$$r_{\text{vadv}}(x) := \arg \max_{r; \|r\|_2 \leq \epsilon} D \left[p(y|x, \hat{\theta}), p(y|x + r, \hat{\theta}) \right]$$

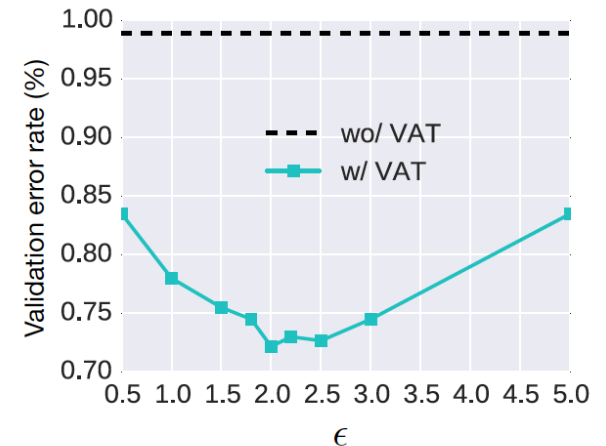
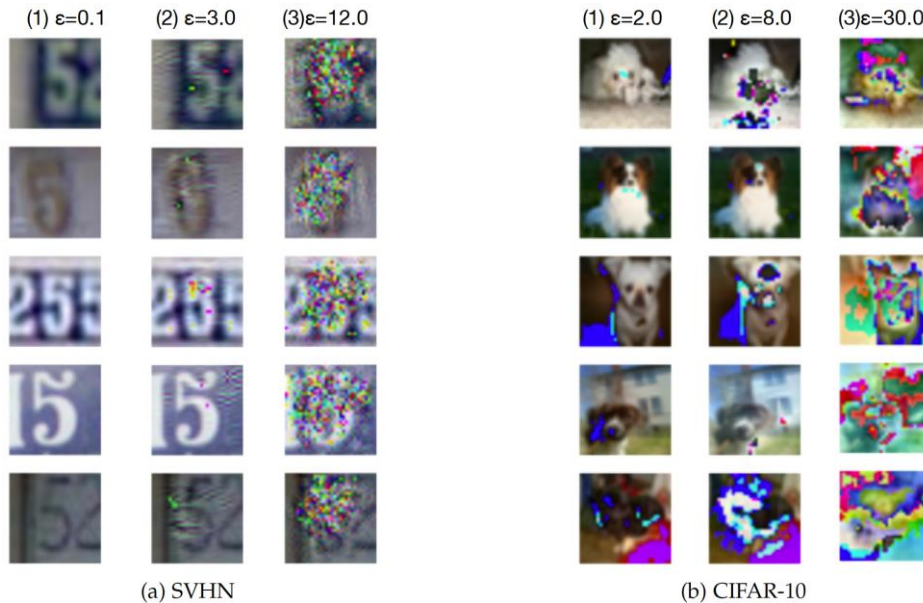
2. Compute the following gradient for minimizing adversarial divergence

$$\nabla_{\theta} D \left[p(y|x, \hat{\theta}), p(y|x + r_{\text{vadv}}, \theta) \right] \Big|_{\theta=\hat{\theta}}$$

- **Virtual Adversarial Training** [Miyato et al., 2017]
 - Combining with Entorpy Minimization improves performance further

Method	Test error rate(%)	
	SVHN $N_l = 1000$	CIFAR-10 $N_l = 4000$
Π model [24].	4.82 (± 0.17)	12.36 (± 0.31)
Temporal ensembling [24]	4.42 (± 0.16)	12.16 (± 0.24)
Sajjadi et al. [35]		11.29 (± 0.24)
(On Conv-Large used in [24])		
VAT	5.42 (± 0.22)	11.36 (± 0.34)
VAT+EntMin	3.86 (± 0.11)	10.55 (± 0.05)

- Virtual adversarial examples (large ϵ degrades performance)

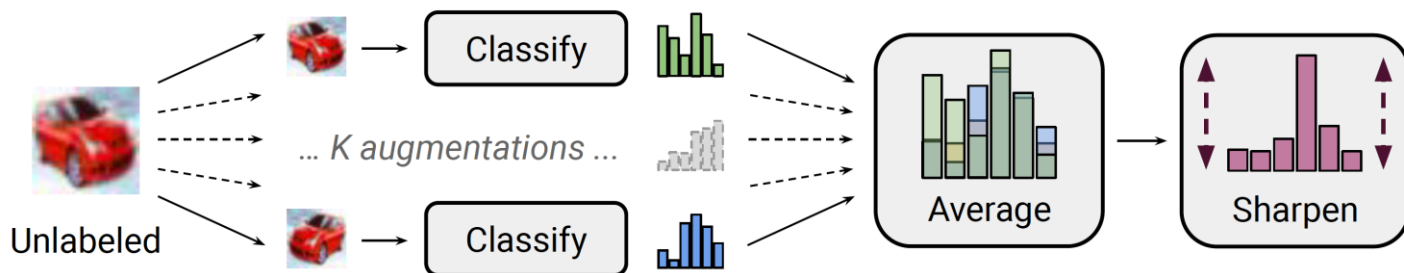


- **MixMatch** [Berthelot et al., 2019]

- It unifies the dominant paradigms for semi-supervised learning
- Using labeled samples \mathcal{X} and unlabeled samples \mathcal{U} , produce ...
 - **Augmented labeled samples \mathcal{X}'**
 - **Augmented unlabeled samples \mathcal{U}' with its guessed label**



1. For each labeled sample x , just apply random data augmentation $\hat{x} = \text{Augment}(x)$
2. For each unlabeled sample u ,
 - Construct K different augmented samples
 - Guess its label by averaging & sharpening



- **MixMatch** [Berthelot et al., 2019]

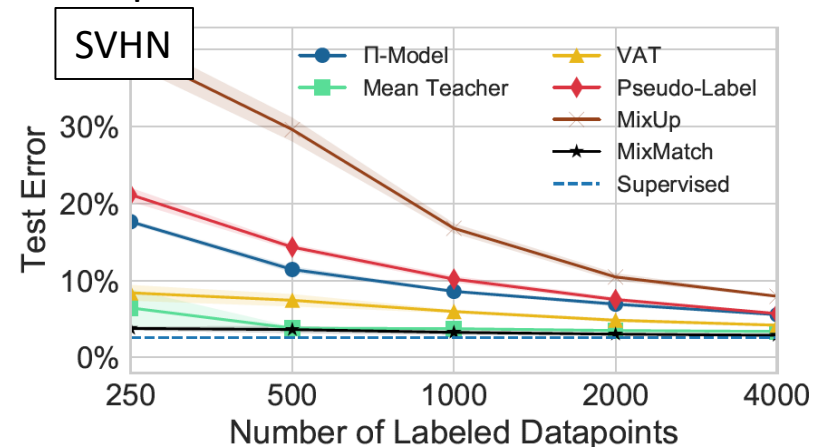
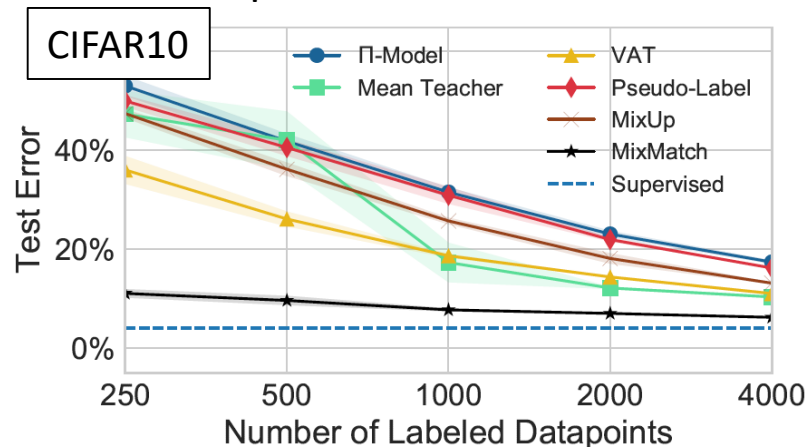
- It unifies the dominant paradigms for semi-supervised learning
- Using labeled samples \mathcal{X} and unlabeled samples \mathcal{U} , produce ...
 - **Augmented labeled samples \mathcal{X}'**
 - **Augmented unlabeled samples \mathcal{U}' with its guessed label**



1. $\hat{\mathcal{X}} = \{(\hat{x}_1, p_1), \dots\}$
2. $\hat{\mathcal{U}} = \{(\hat{u}_{1,k}, q_1), \dots\}$
3. $\text{Mixup}(\hat{\mathcal{X}} \cup \hat{\mathcal{U}}) \rightarrow \mathcal{X}' \cup \mathcal{U}'$
4. Using the labeled or guessed samples, **minimize typical cross-entropy loss**

• MixMatch [Berthelot et al., 2019]

- This simple method achieves **state-of-the-art** performance on benchmark datasets



- Ablation study: **all components are important**

Ablation		250 labels	4000 labels
Averaging	MixMatch	11.80	6.00
	MixMatch without distribution averaging ($K = 1$)	17.09	8.06
	MixMatch with $K = 3$	11.55	6.23
	MixMatch with $K = 4$	12.45	5.88
	MixMatch without temperature sharpening ($T = 1$)	27.83	10.59 — Sharpening
Mixup	MixMatch with parameter EMA	11.86	6.47
	MixMatch without MixUp	39.11	10.97
	MixMatch with MixUp on labeled only	32.16	9.22
	MixMatch with MixUp on unlabeled only	12.35	6.83
	MixMatch with MixUp on separate labeled and unlabeled	12.26	6.50
Interpolation Consistency Training [45]		38.60	6.81

1. Semi-supervised Learning

- Entropy Minimization & Pseudo Labeling
- Consistency Regularization

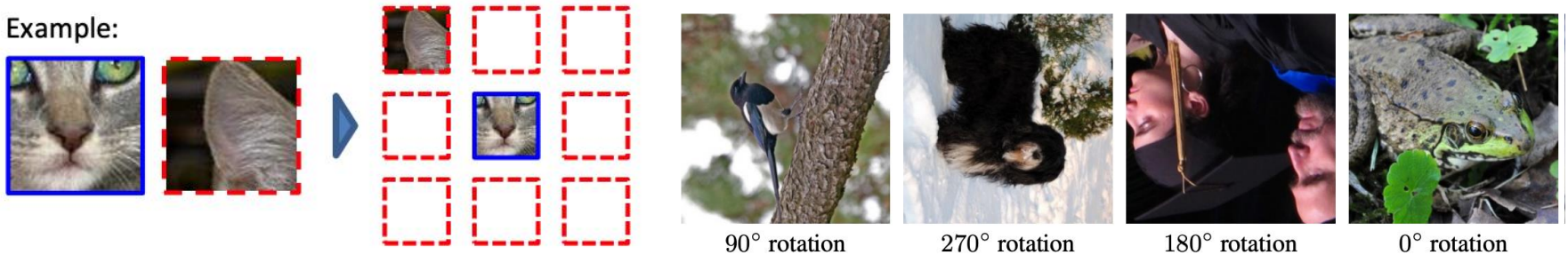
2. Unsupervised Learning

- Self-supervised learning
- Clustering-based approaches
- Maximizing mutual information


- **Self-supervision?**

- It is **a label** constructed **from only input signals** without human-annotation
- Using self-supervision, one can apply supervised learning approaches
- Examples: Predicting relative location of patches¹ or rotation degree²

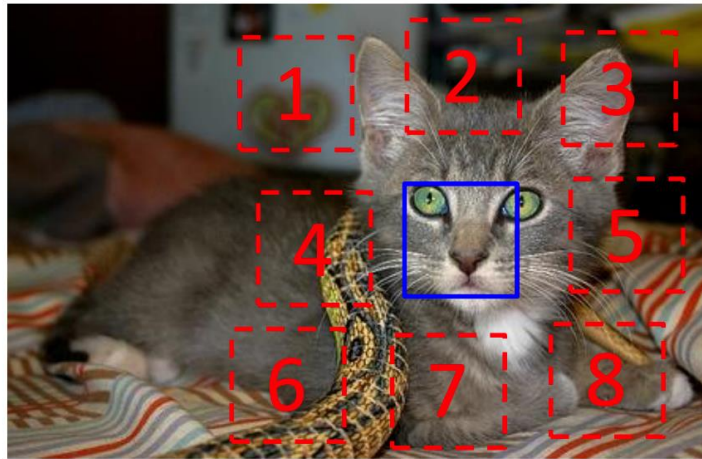
Example:



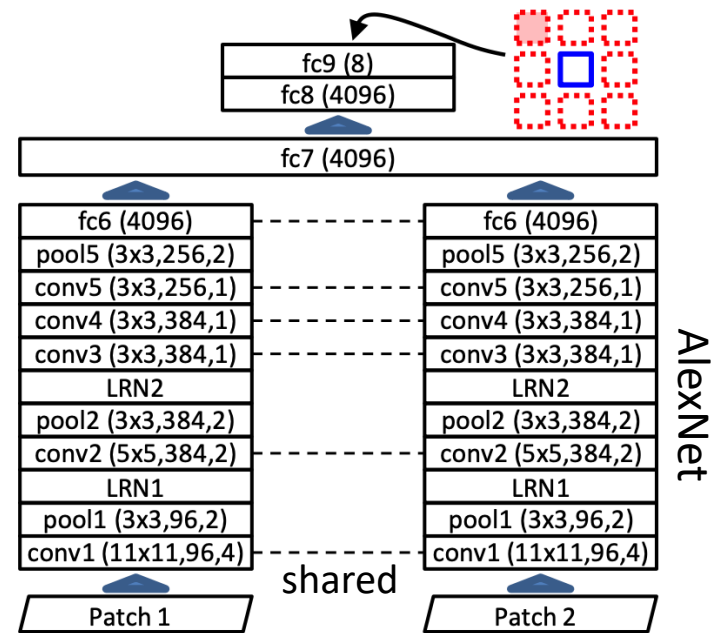
- What can we learn from self-supervised learning?

- To predict (well-designed) self-supervision, one might **require high-level understanding of inputs**,
- E.g., we should know  is the right ear of the cat for predicting locations
- Thus, **high-level representations could be learned w/o human-annotation**

- **Context Prediction** [Doersch et al., 2015]
 - From a natural image, extract 3x3 patches
 - **Patch1**: The center patch & **Patch2**: Choose one of other patches randomly
 - **Task**: Given **Patch1-2**, predict its label (1-8)

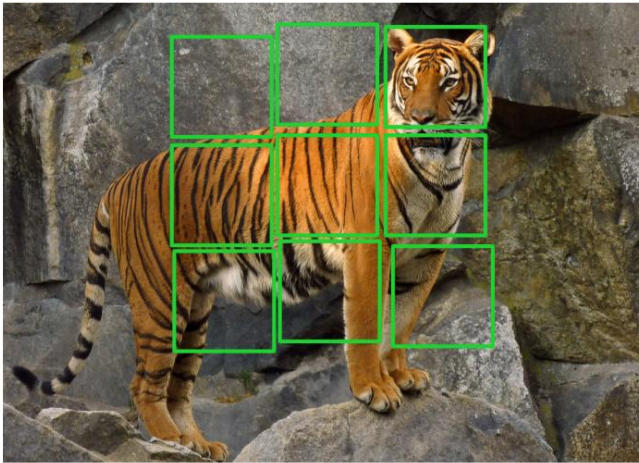


$$X = (\text{cat face patch}, \text{cat ear patch}); Y = 3$$



- Each patch's embedding is computed by one **shared** embedding function

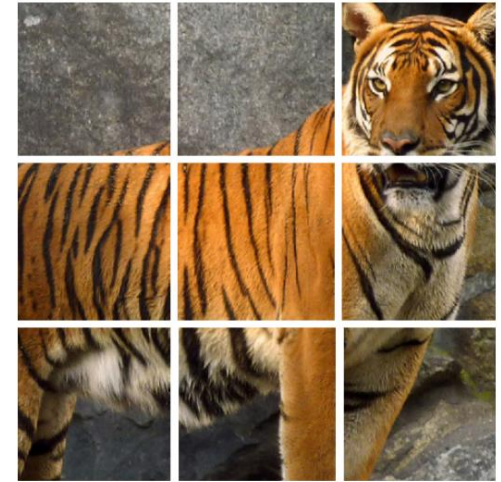
- **Solving Jigsaw Puzzles** [Noroozi & Favaro, 2016]
 - Extension from [Doersch et al., 2015]
 - From (a) a natural image, extract 3x3 patches and (b) **shuffle** them
 - **Task:** From (b) the shuffled patches, find **which permutation is applied**



(a)

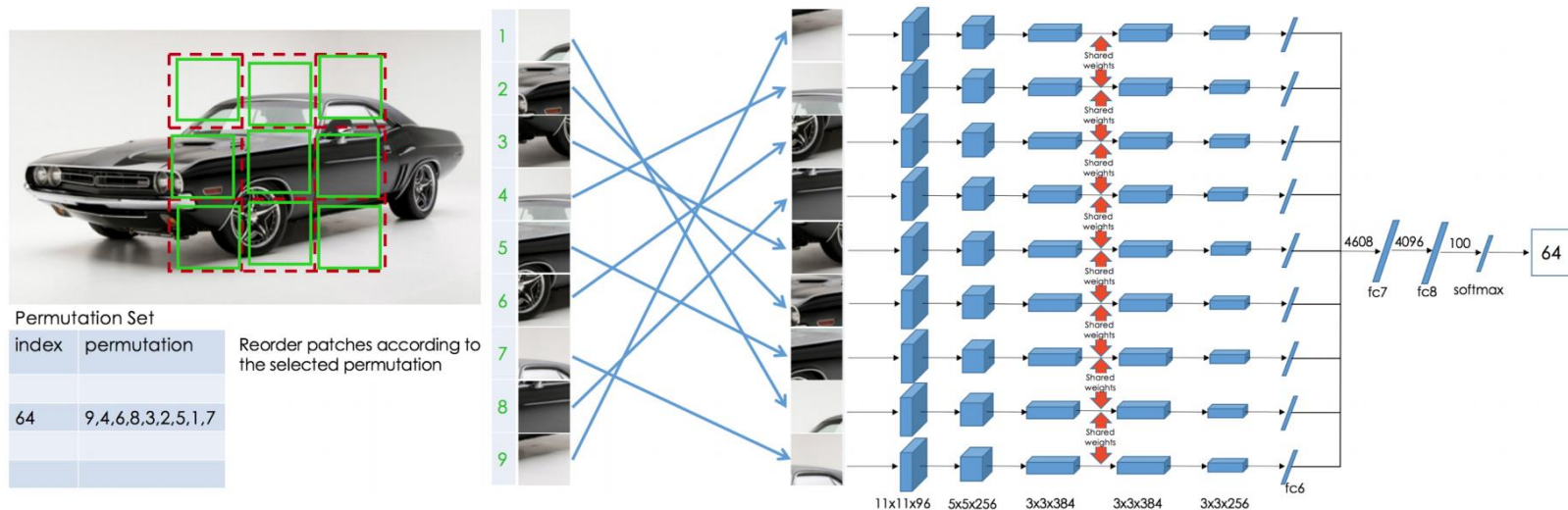


(b)



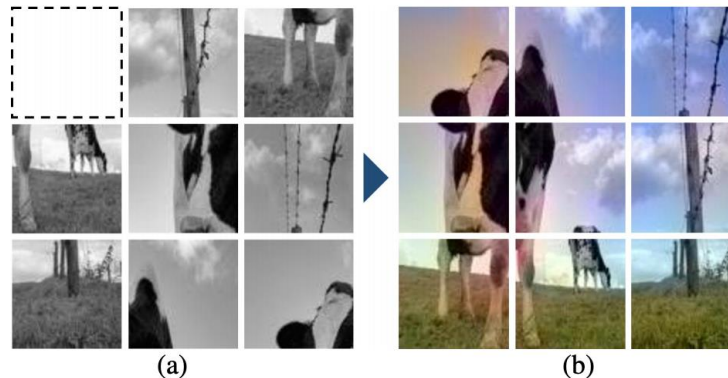
(c)

- **Solving Jigsaw Puzzles** [Noroozi & Favaro, 2016]
 - Extension from [Doersch et al., 2016]
 - From a natural image, extract 3x3 patches and **shuffle** them
 - **Task:** From the shuffled patches, find **which permutation is applied**

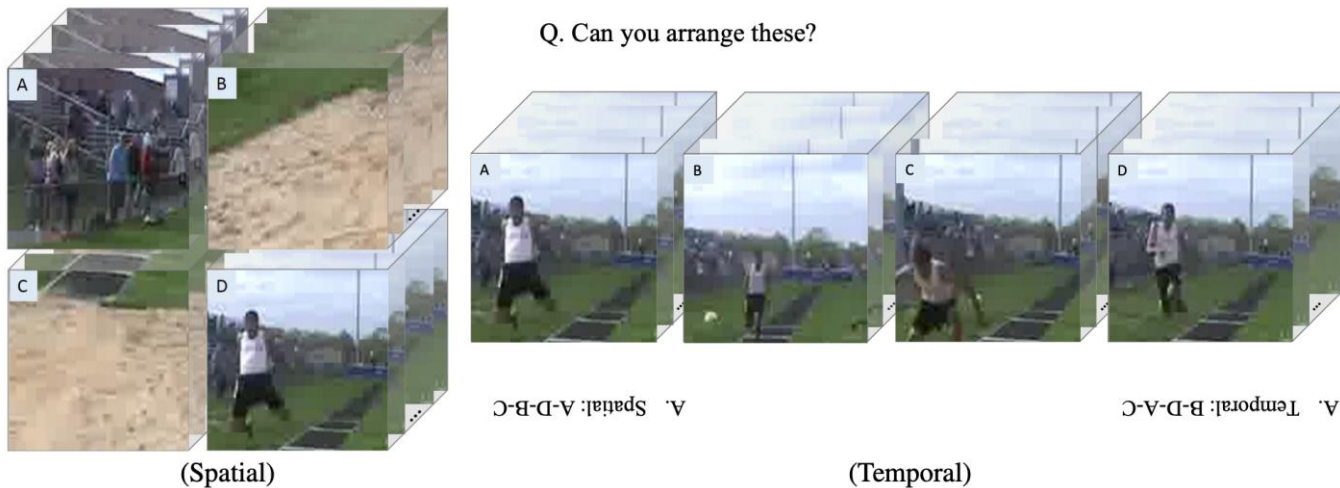


- Each patch's embedding is computed by one **shared** embedding function
- There are too many permutations ($9! = 362k$) \Rightarrow choose a subset of them
 - Empirically, **neither simple nor ambiguous tasks** achieve better performance

- **Solving Jigsaw Puzzles** [Noroozi & Favaro, 2016]
 - Extension 1: **Completing Damaged Jigsaw Puzzles** [Kim et al., 2018]



- Extension 2: **Space-Time Cubic Puzzles** [Kim et al., 2019] for **video representation**

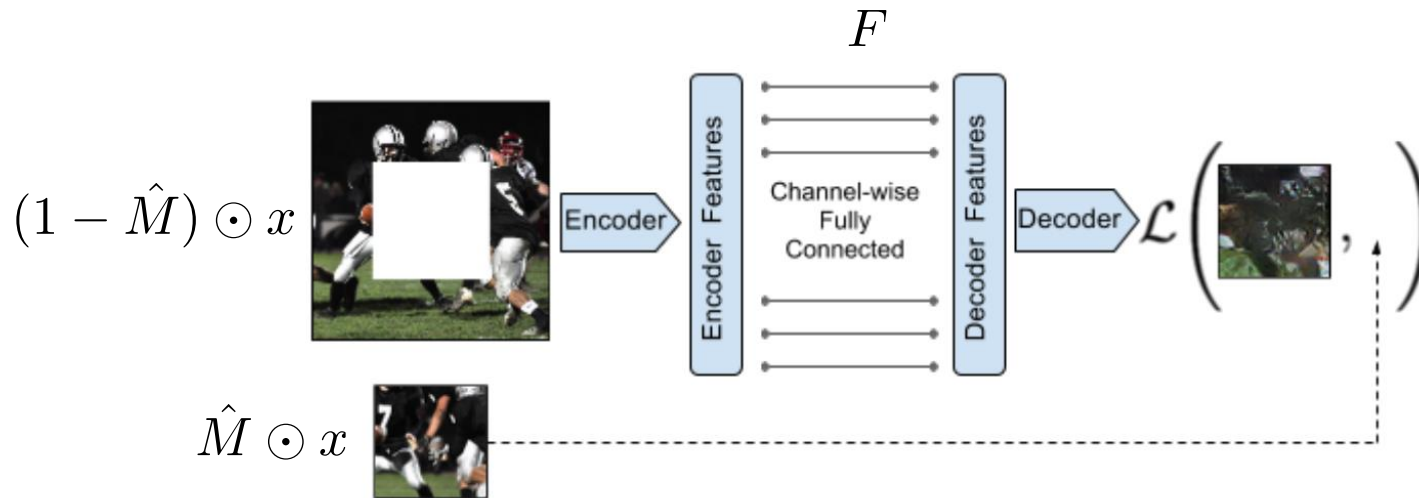


- **Feature Learning by Inpainting** [Pathak et al., 2016]

- **Task:** Predict the masked region using **its surrounding information**

- The auto-encoder is trained via **reconstruction loss**

$$\mathcal{L}_{\text{rec}}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$



- **Feature Learning by Inpainting** [Pathak et al., 2016]

- **Task:** Predict the masked region using **its surrounding information**

- The auto-encoder is trained via **reconstruction loss**

$$\mathcal{L}_{\text{rec}}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

- With **adversarial loss**, reconstruction quality is improved further

$$\mathcal{L}_{\text{adv}} = \max_D \mathbb{E}_{x \in \mathcal{X}} \left[\log D(x) + \log(1 - D(F((1 - \hat{M}) \odot x)) \right]$$



(a) Input context



(b) Human artist



(c) Context Encoder
(L2 loss)



(d) Context Encoder
(L2 + Adversarial loss)

- **Feature Learning by Inpainting** [Pathak et al., 2016]

- **Task:** Predict the masked region using **its surrounding information**

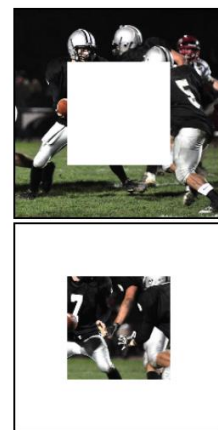
- The auto-encoder is trained via **reconstruction loss**

$$\mathcal{L}_{\text{rec}}(x) = \|\hat{M} \odot (x - F((1 - \hat{M}) \odot x))\|_2^2$$

- With **adversarial loss**, reconstruction quality is improved further

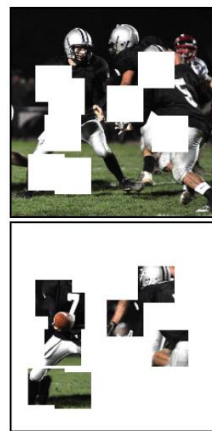
$$\mathcal{L}_{\text{adv}} = \max_D \mathbb{E}_{x \in \mathcal{X}} \left[\log D(x) + \log(1 - D(F((1 - \hat{M}) \odot x)) \right]$$

- How to **construct** the masks?



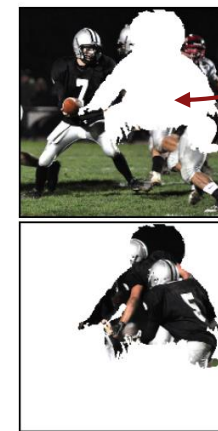
(a) Central region

<



(b) Random block

≈

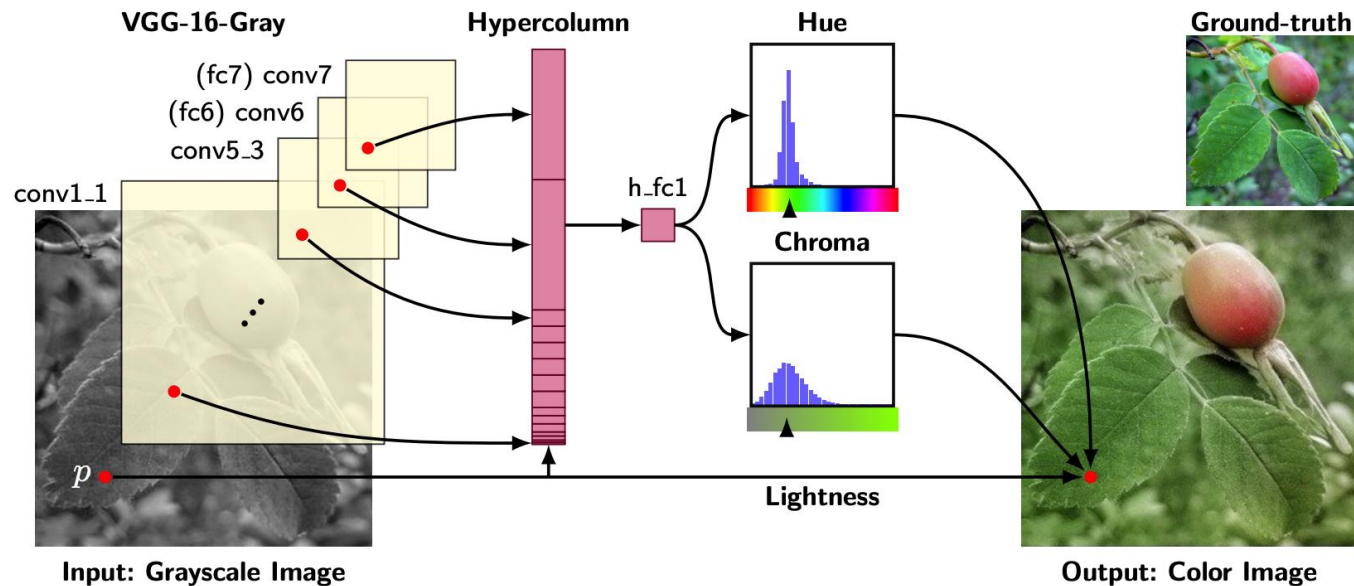


(c) Random region

A segmentation mask in other dataset

- **Colorization** [Larsson et al., 2017]

- **Task:** Predict color information for each pixel from gray images
- **Dense prediction** is required for colorization \Rightarrow which type of architecture?
- **Hypercolumn:** for each pixel, concatenate all feature vectors in feature map

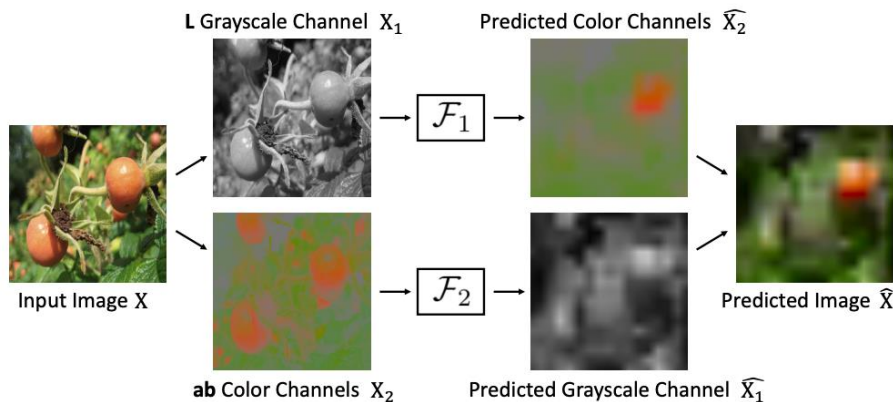


- Aft
- This can handle **multimodal** color distributions well

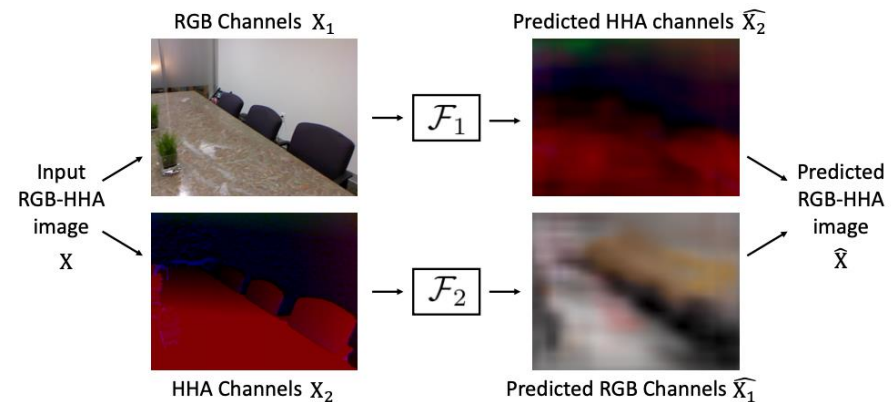
- **Split-Brain Autoencoders** [Zhang et al., 2017]

- **Task:** Cross-channel auto-encoding

- **Split** the input data: $X \in \mathbb{R}^{H \times W \times C} = \text{concat}(X_1 \in \mathbb{R}^{H \times W \times C_1}, X_2 \in \mathbb{R}^{H \times W \times C_2})$
 - Examples: colors & depth, L & ab (in Lab color space)



(a) **Lab Images**



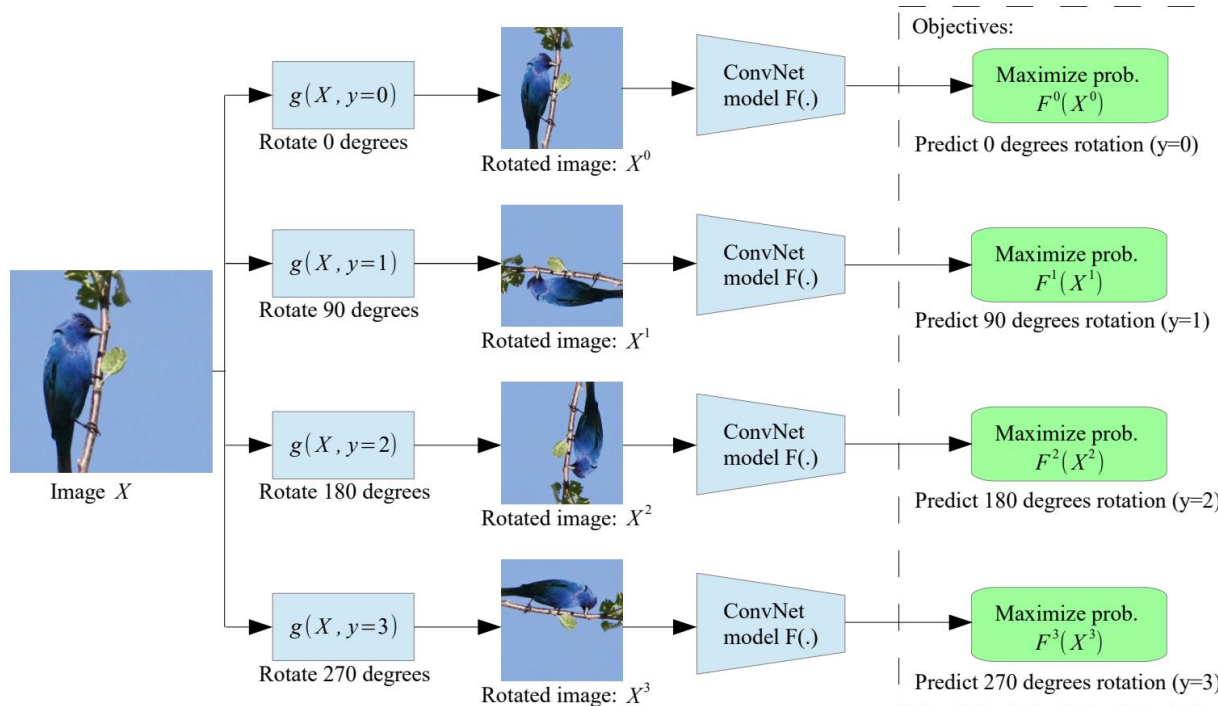
(b) **RGB-D Images**

- Train multiple **cross-channel auto-encoders**

$$\mathcal{F}_1^* = \arg \min_{\mathcal{F}_1} \mathcal{L}_1(\mathcal{F}_1(X_1), X_2)$$

$$\mathcal{F}_2^* = \arg \min_{\mathcal{F}_2} \mathcal{L}_2(\mathcal{F}_2(X_2), X_1)$$

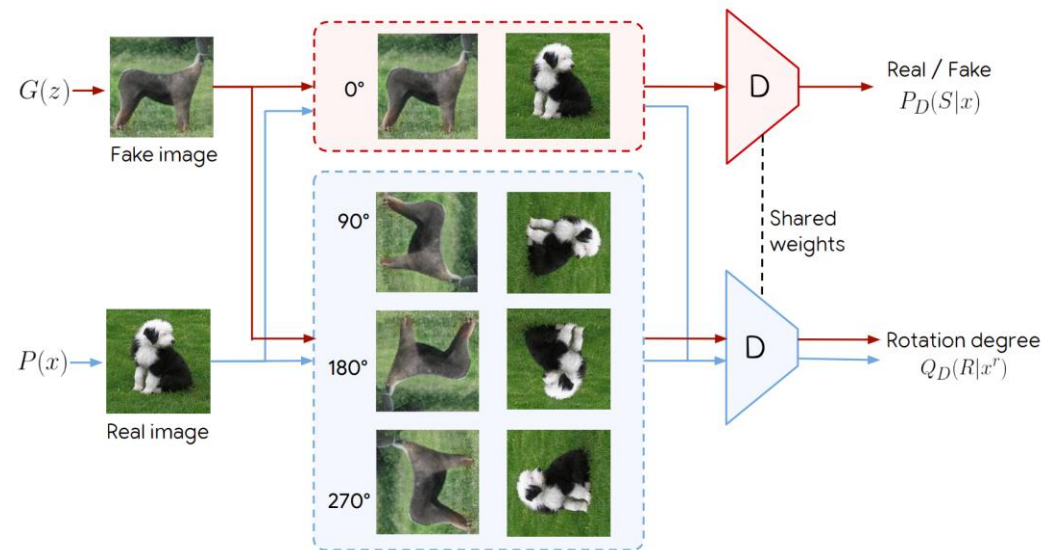
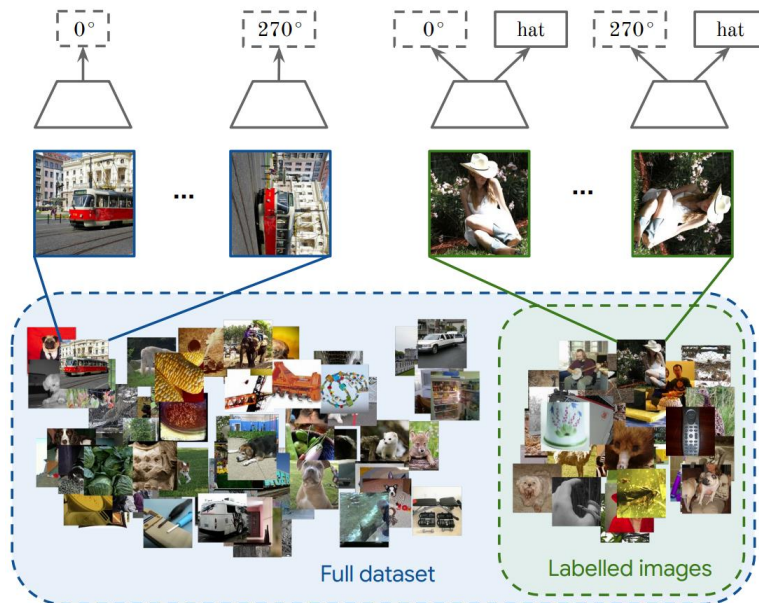
- **Rotation** [Gidaris et al., 2018]
 - **Task:** Predict the rotation degree from a rotated image



- What is the optimal number of classes (rotations)?
 - Empirically, using 4 rotations (0°, 90°, 180°, 270°) is best

Unsupervised Learning: Self-supervised Learning

- **Rotation** [Gidaris et al., 2018]
 - **Task:** Predict the rotation degree from a rotated image
 - Due to **its simplicity**, this approach is widely used for other applications
 - Semi-supervised Learning [Zhai et al., 2019]
 - Training GAN [Chen et al., 2019]



- **How to measure the quality of self-supervision? (quantitative)**
 - 1. Self-supervised Learning on a large-scale dataset**
 - 2. Fine-tuning for a downstream task**
 3. Compare with
 - Without step 1, pre-training (lower bound)
 - With supervised learning with labels in step 1 (upper bound)
 - Other unsupervised learning methods
- **Popular benchmark 1:**
 - Pretraining on ImageNet
 - Fine-tuning on VOC Pascal tasks (Classification, Detection, Segmentation)
- **Popular benchmark 2:**
 - Pretraining on ImageNet
 - Training only classifiers on ImageNet/Places datasets with **freezing embedding functions**

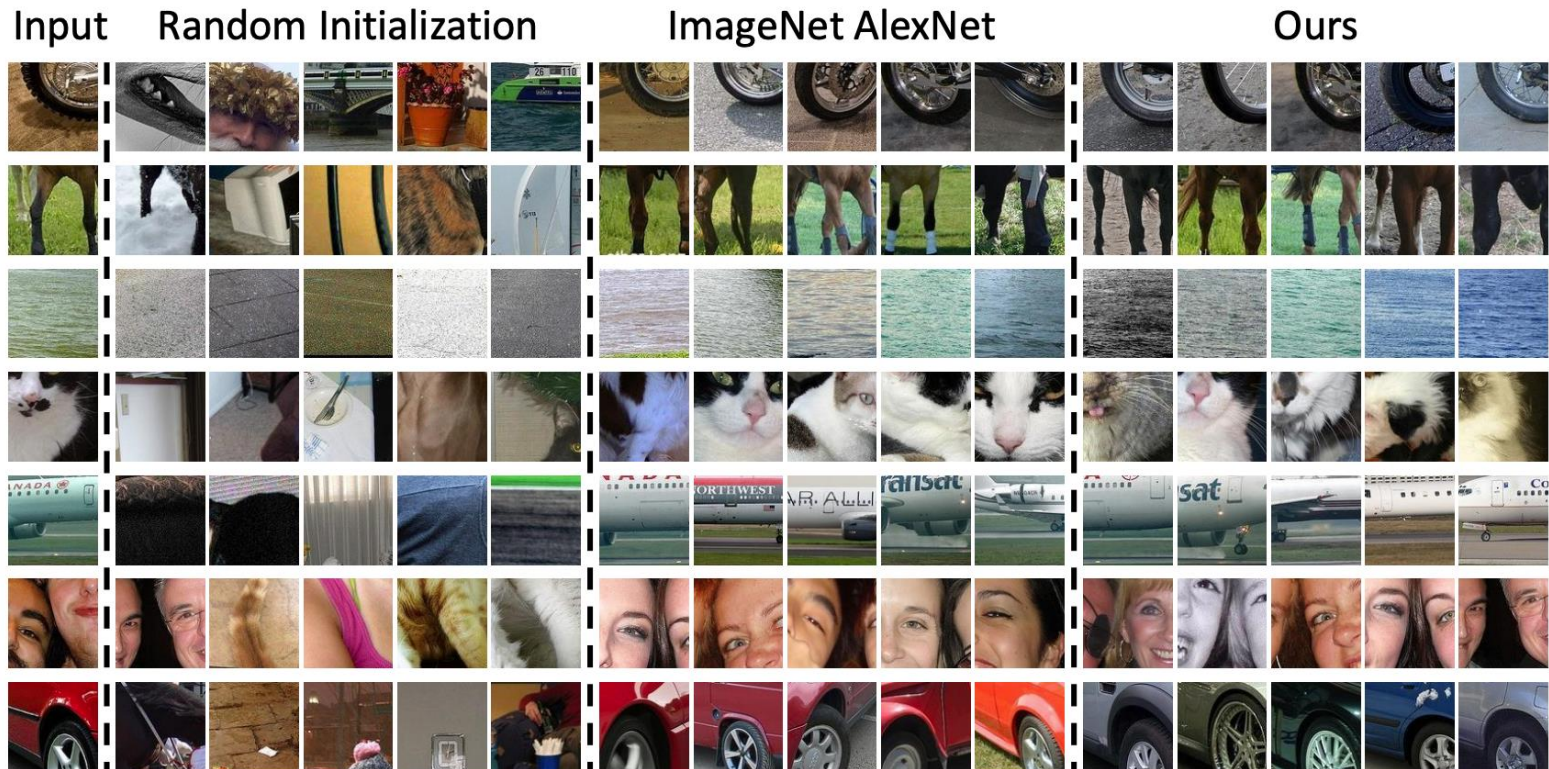
- **How to measure the quality of self-supervision? (quantitative)**
 - **Popular benchmark 1:**
 - Pretraining on ImageNet
 - Fine-tuning on VOC Pascal tasks (Classification, Detection, Segmentation)

	Classification (%mAP)		Detection (%mAP)	Segmentation (%mIoU)
Trained layers	fc6-8	all	all	all
ImageNet labels	78.9	79.9	56.8	48.0
Random		53.3	43.4	19.8
Random rescaled Krähenbühl et al. (2015)	39.2	56.6	45.6	32.6
Egomotion (Agrawal et al., 2015)	31.0	54.2	43.9	
Context Encoders (Pathak et al., 2016b)	34.6	56.5	44.5	29.7
Tracking (Wang & Gupta, 2015)	55.6	63.1	47.4	
Context (Doersch et al., 2015)	55.1	65.3	51.1	
Colorization (Zhang et al., 2016a)	61.5	65.6	46.9	35.6
BIGAN (Donahue et al., 2016)	52.3	60.1	46.9	34.9
Jigsaw Puzzles (Noroozi & Favaro, 2016)	-	67.6	53.2	37.6
NAT (Bojanowski & Joulin, 2017)	56.7	65.3	49.4	
Split-Brain (Zhang et al., 2016b)	63.0	67.1	46.7	36.0
ColorProxy (Larsson et al., 2017)		65.9		38.4
Counting (Noroozi et al., 2017)	-	67.7	51.4	36.6
(Ours) RotNet	70.87	72.97	54.4	39.1

- **How to measure the quality of self-supervision? (quantitative)**
 - **Popular benchmark 2:**
 - Pretraining on ImageNet
 - Training only classifiers on ImageNet/Places datasets with **freezing embedding functions**
 - It measures the quality of representations more directly

Method	Conv1	Conv2	Conv3	Conv4	Conv5
ImageNet labels	19.3	36.3	44.2	48.3	50.5
Random	11.6	17.1	16.9	16.3	14.1
Random rescaled Krähenbühl et al. (2015)	17.5	23.0	24.5	23.2	20.6
Context (Doersch et al., 2015)	16.2	23.3	30.2	31.7	29.6
Context Encoders (Pathak et al., 2016b)	14.1	20.7	21.0	19.8	15.5
Colorization (Zhang et al., 2016a)	12.5	24.5	30.4	31.5	30.3
Jigsaw Puzzles (Noroozi & Favaro, 2016)	18.2	28.8	34.0	33.9	27.1
BIGAN (Donahue et al., 2016)	17.7	24.5	31.0	29.9	28.0
Split-Brain (Zhang et al., 2016b)	17.7	29.3	35.4	35.2	32.8
Counting (Noroozi et al., 2017)	18.0	30.6	34.3	32.5	25.7
(Ours) RotNet	18.8	31.7	38.7	38.2	36.5

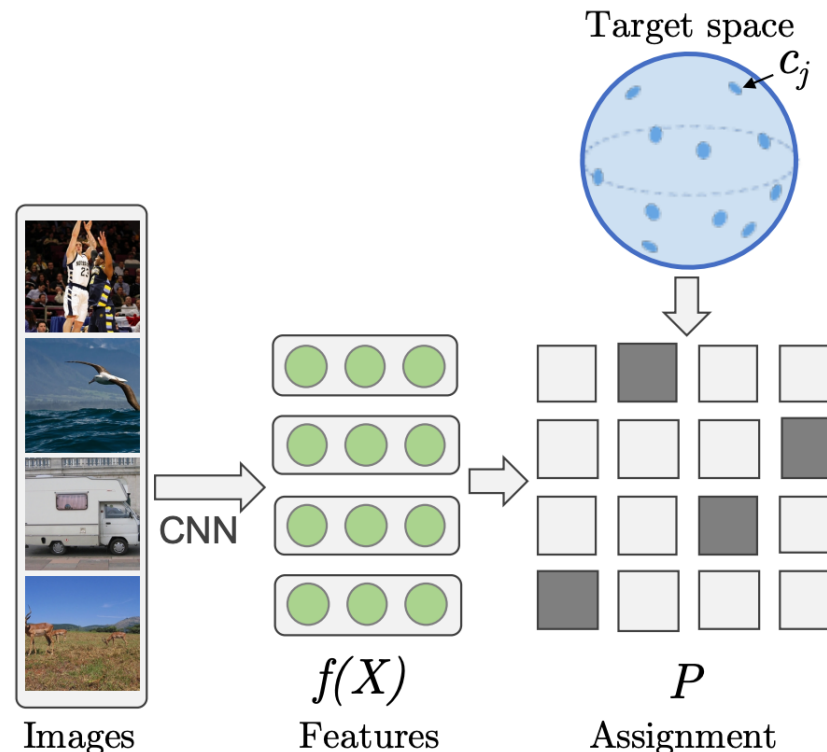
- **How to measure the quality of self-supervision? (qualitative)**
 - Find the nearest samples from a query on embedding space



- **Limitations** on self-supervised learning
 1. **Domain-specific knowledge is required to design self-supervision**
 - For different domains (e.g., audio), existing methods might be not working
 2. **The use of self-supervision is limited**
 - Patch-based tasks for small-sized datasets, e.g., CIFAR
 - Colorization-based tasks for single-channel inputs, e.g., gray images
 3. **Pre-processing is important to avoid trivial solutions**
 - In some cameras, one color channel (commonly green) is shrunk toward the image center relative to the others
 - CNN can capture the difference, so predicting location is available without high-level understanding
- **Next:** more general approaches, Clustering & Mutual Information-based

- **Noise As Target** [Bojanowski & Joulin, 2017]

- **Idea:** Assume that (oracle) representation vectors lie on the unit sphere uniformly
 - It is similar that each image belongs to a unique class
- $f(X) \in \mathbb{R}^{n \times d}$: the predicted representation vectors
- $C \in \mathbb{R}^{n \times d}$: the fixed true representation vectors
- $P \in \{0, 1\}^{n \times n}$: Assignment matrix, i.e., if $p_{ij} = 1$, then we aim to learn $f(x_i) = c_j$



- **Noise As Target** [Bojanowski & Joulin, 2017]

- **Idea:** Assume that (oracle) representation vectors lie on the unit sphere uniformly
 - It is similar that each image belongs to a unique class

- If P is given, then the optimization is written as

$$\min_{\theta} \|f_{\theta}(X) - PC\|_F^2$$

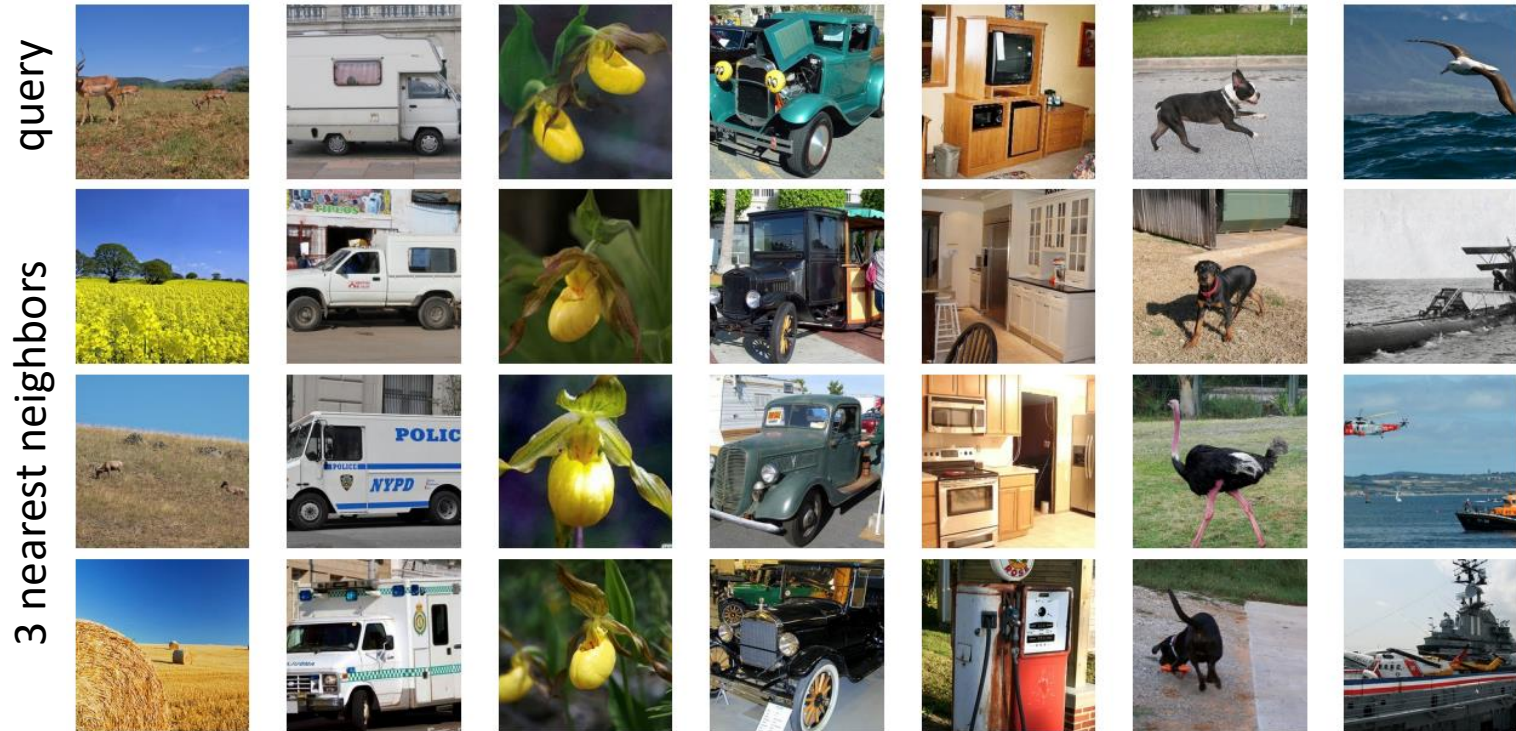
- Which P is **optimal**?

- Given θ , optimal P can be found in $O(n^3) \Rightarrow$ **inefficient** for large datasets

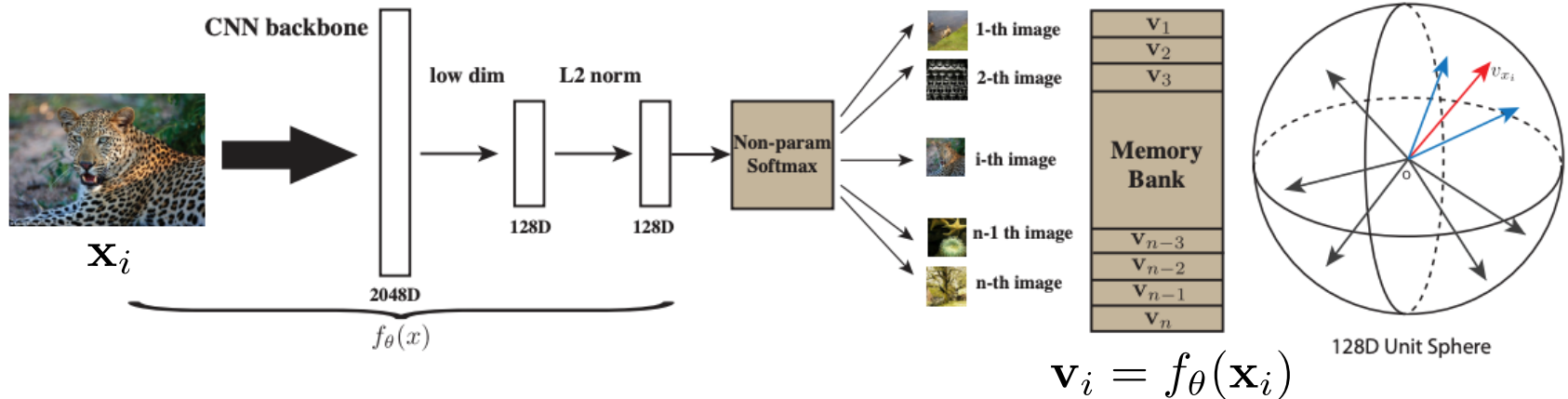
- **Stochastic update** on P with each mini-batch

- Maintain P during training
 - For each mini-batch, compute $f(X_b)$ and its corresponding targets
 - Find P_b^*
 - Update θ minimizing $\|f_{\theta}(X_b) - P_b^* C\|_F^2$
 - For each epoch, the update can be done in $O(b^3 \times n/b) = O(nb^2)$

- **Noise As Target** [Bojanowski & Joulin, 2017]



- **Instance Discrimination** [Wu et al., 2018]
 - **Idea:** Each image belongs to an unique class



- **Non-parameteric classifier**

$$P(i|\mathbf{v}) = \frac{\exp(\mathbf{v}_i^{\top} \mathbf{v} / \tau)}{\sum_{j=1}^n \exp(\mathbf{v}_j^{\top} \mathbf{v} / \tau)}$$

- Each class has only one instance $\Rightarrow \mathbf{v}_i$ can be used directly as a class prototype

- **Instance Discrimination** [Wu et al., 2018]

- **Idea:** Each image belongs to an unique class

- **Non-parameteric classifier**

$$P(i|\mathbf{v}) = \frac{\exp(\mathbf{v}_i^\top \mathbf{v} / \tau)}{\sum_{j=1}^n \exp(\mathbf{v}_j^\top \mathbf{v} / \tau)}$$

- Computing $P(i|\mathbf{v})$ is **inefficient** because it requires all $\mathbf{v}_j = f_\theta(\mathbf{x}_j)$ and $\mathbf{v}_j^\top \mathbf{v}$
- **Solution 1:** Memory bank
 - Store all \mathbf{v}_j in memory and update them for each mini-batch
- **Solution 2:** Noise-Contrastive Estimation [Gutmann & Hyvarinen, 2010]
 - It casts multi-class classification into a set of binary classification problems

Positive sample: $P(D = 1|i, \mathbf{v}) = P(i|\mathbf{v}) = \frac{\exp(\mathbf{v}_i^\top \mathbf{v})}{\exp(\mathbf{v}_i^\top \mathbf{v}) + \sum_{k=1}^m \exp(\mathbf{v}_{j_k}^\top \mathbf{v})}$

m negative samples

Objective: $\mathcal{L}_{\text{NCE}} = -\mathbb{E}_{P_d}[\log P(D = 1|i, \mathbf{v})] - m\mathbb{E}_{P_n}[\log P(D = 0|i, \mathbf{v}')]$

\swarrow data distribution \swarrow noise distribution (uniform)

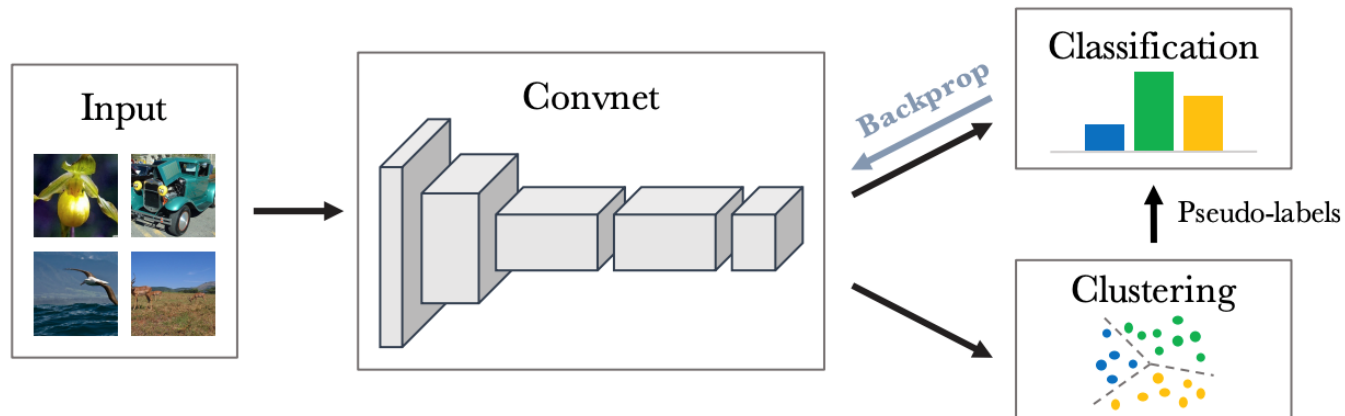
- **Instance Discrimination** [Wu et al., 2018]
 - Ablation Study
 - **Non-parametric softmax is better** than the parametric version
 - **NCE with many negative samples** approaches to the no-approximation version

Training / Testing	Linear SVM	Nearest Neighbor
Param Softmax	60.3	63.0
Non-Param Softmax	75.4	80.8
NCE $m = 1$	44.3	42.5
NCE $m = 10$	60.2	63.4
NCE $m = 512$	64.3	78.4
NCE $m = 4096$	70.2	80.4

- Large embedding size increases the performance, but it is saturated at 256

embedding size	32	64	128	256
top-1 accuracy	34.0	38.8	41.0	40.1

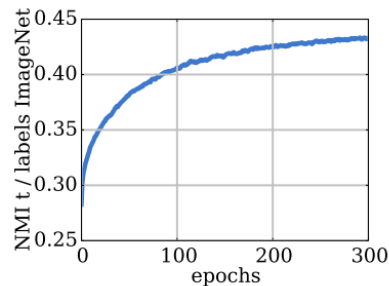
- **DeepCluster** [Caron et al., 2018]
 - **Idea:** Clustering on embedding space provides pseudo-labels



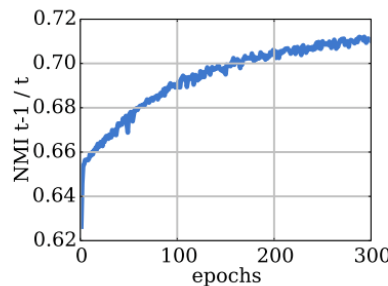
- **Simple method:** Alternate between
 1. Clustering the features to produce pseudo-labels
 2. Updating parameters by predicting these pseudo-labels
- What are **trivial solutions**? and how to avoid them?
 - Empty cluster \Leftarrow feature quantization (it reassigns empty clusters)
 - Imbalanced sizes of clusters \Leftarrow over-sampling

- **DeepCluster** [Caron et al., 2018]

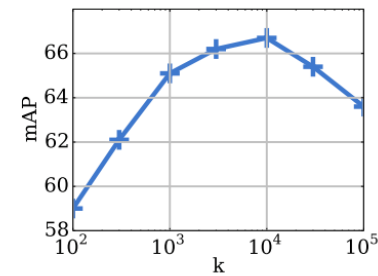
- Is the **clustering quality** improved during training?
 - a. Clustering overlap between DeepCluster and ImageNet
 - b. Clustering overlap between the current and previous epochs
 - c. Influence of the number of clusters



(a) Clustering quality



(b) Cluster reassignment



(c) Influence of k

- **Which images activate the target filters** in the last convolutional layer?



- **Deep InfoMax** [Hjelm et al., 2019]

- **Idea:** Maximizing mutual information between inputs and features
- $Y = E_\psi(X)$ is the feature vector of input X where E_ψ is an embedding function
- **How to optimize mutual information?** [Donsker & Varadhan, 1983]

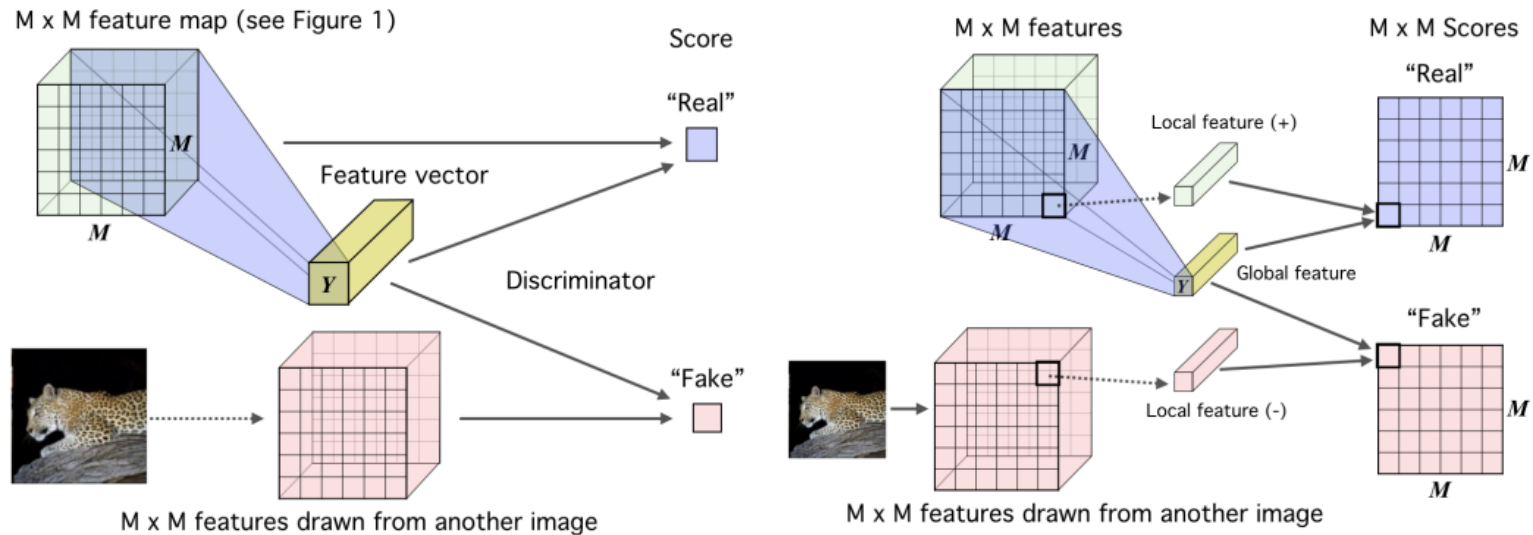
$$\mathcal{I}(X; Y) := \mathcal{D}_{\text{KL}}(\mathbb{J} \parallel \mathbb{M}) \geq \widehat{\mathcal{I}}_w^{(\text{DV})}(X; Y) := \mathbb{E}_{\mathbb{J}}[T_w(x, y)] - \log \mathbb{E}_{\mathbb{M}}[e^{T_w(x, y)}]$$

- Optimize the embedding function E_ψ and discriminator T_w simultaneously

$$\hat{w}, \hat{\psi} = \arg \max_{w, \psi} \widehat{\mathcal{I}}_w(X; E_\psi(X))$$

- **Deep InfoMax** [Hjelm et al., 2019]

- **Idea:** Maximizing mutual information between inputs and features
- $Y = E_\psi(X)$ is the feature vector of input X where E_ψ is an embedding function

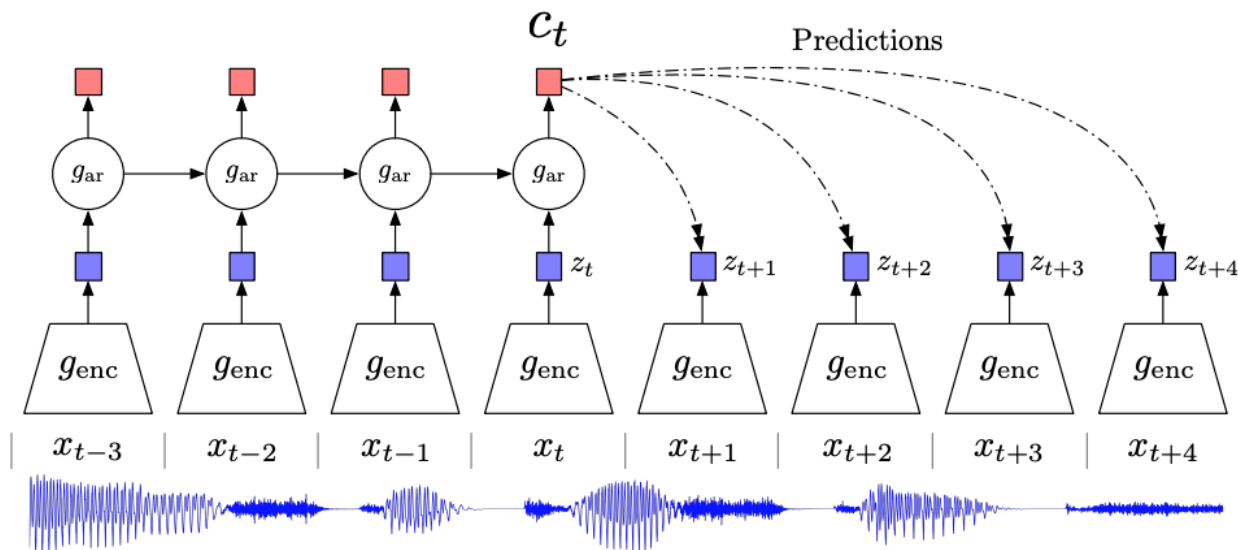


- Instead of (left) maximizing MI between global features, **(right) doing MI between global and local features** achieves better performance

- **Contrastive Predictive Coding** [Oord et al., 2018]

- **Idea:** Predicting future information with discarding low-level information

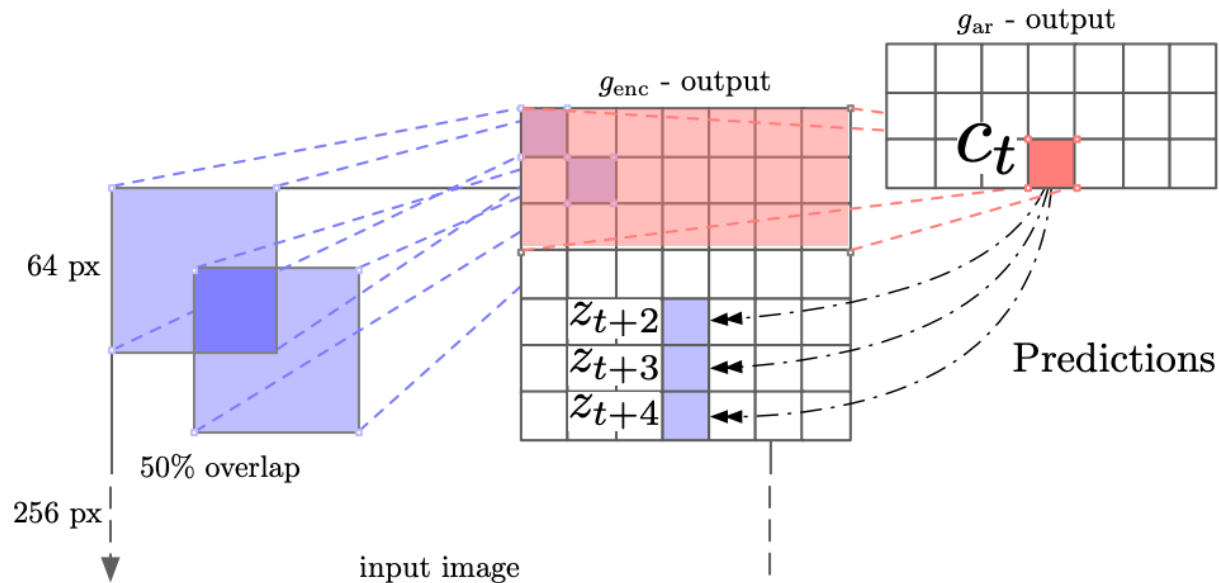
- x_t : data at time t
- $z_t = g_{\text{enc}}(x_t)$: high-level latent representation of x_t
- $c_t = g_{\text{ar}}(x_1, x_2, \dots, x_t)$: context latent representation summarizing all $z_{\leq t}$



- **Contrastive Predictive Coding** [Oord et al., 2018]

- **Idea:** Predicting future information with discarding low-level information

- x_t : data at time t
- $z_t = g_{\text{enc}}(x_t)$: high-level latent representation of x_t
- $c_t = g_{\text{ar}}(x_1, x_2, \dots, x_t)$: context latent representation summarizing all $z_{\leq t}$



- **Contrastive Predictive Coding** [Oord et al., 2018]

- **Idea:** Predicting future information with discarding low-level information
- How to maximize mutual information between x_{t+k} and c_t ?
 - Randomly choose one positive sample x_{t+k} and N-1 negative samples $\{x\}$
 - Minimize the following **NCE**-based loss:

$$\mathcal{L}_N = -\mathbb{E}_X \left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_x f_k(x, c_t)} \right]$$

where $f_k(x, c) = \exp(z^\top W_k c)$

- $I(x_{t+k}, c_t) \geq \log(N) - \mathcal{L}_N$ and it becomes tighter as N becomes larger

References

- [Laine & Aila, 2017] Temporal Ensembling for Semi-Supervised Learning, ICML2017
- [Tarvainen & Harri Valpola, 2017] Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results, NIPS 2017
- [Miyato et al., 2017] Virtual Adversarial Training: A Regularization Method for Supervised and Semi-Supervised Learning, TPAMI 2019
- [Berthelot et al., 2019] MixMatch: A Holistic Approach to Semi-Supervised Learning, NIPS 2019
- [Doersch et al., 2015] Unsupervised Visual Representation Learning by Context Prediction, ICCV 2015
- [Larsson et al., 2016] Learning Representations for Automatic Colorization, ECCV 2016
- [Noroozi & Favaro] Unsupervised Learning of Visual Representations by Solving Jigsaw Puzzles, ECCV 2016
- [Pathak et al., 2016] Context Encoders: Feature Learning by Inpainting, CVPR 2016
- [Zhang et al., 2017] Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction, CVPR 2017
- [Larsson et al., 2017] Colorization as a Proxy Task for Visual Understanding, CVPR 2017
- [Bojanowski & Joulin, 2017] Unsupervised Learning by Predicting Noise, ICML 2017
- [Wu et al., 2018] Unsupervised Feature Learning via Non-parameteric Instance Discrimination, CVPR 2018
- [Kim et al., 2018] Learning Image Representations by Completing Damaged Jigsaw Puzzles, WACV 2018
- [Gidaris et al., 2018] Unsupervised Representation Learning by Predicting Image Rotations, ICLR 2018
- [Caron et al., 2018] Deep Clustering for Unsupervised Learning of Visual Features, ECCV 2018

References

- [Oord et al., 2019] Representation Learning with Contrastive Predictive Coding, 2019
- [Kolesnikov et al., 2019] Revisiting Self-Supervised Visual Representation Learning, CVPR 2019
- [Kim et al., 2019] Self-Supervised Video Representation Learning with Space-Time Cubic Puzzles, AAAI 2019
- [Donsker & Varadhan, 1983] Asymptotic evaluation of certain markov process expectations for large time. Communications on Pure and Applied Mathematics, 1983
- [Gutmann & Hyvärinen, 2010] Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. AISTATS 2010
- [Oliver et al., 2018] Realistic Evaluation of Deep Semi-Supervised Learning Algorithms, NIPS 2018
- [Lee, 2013] Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. ICML Workshop on Challenges in Representation Learning, 2013.
- [Grandvalet & Bengio, 2005] Semi-supervised learning by entropy minimization. NIPS 2005.
- [Tanaka et al., 2018] Joint optimization framework for learning with noisy labels. CVPR 2018