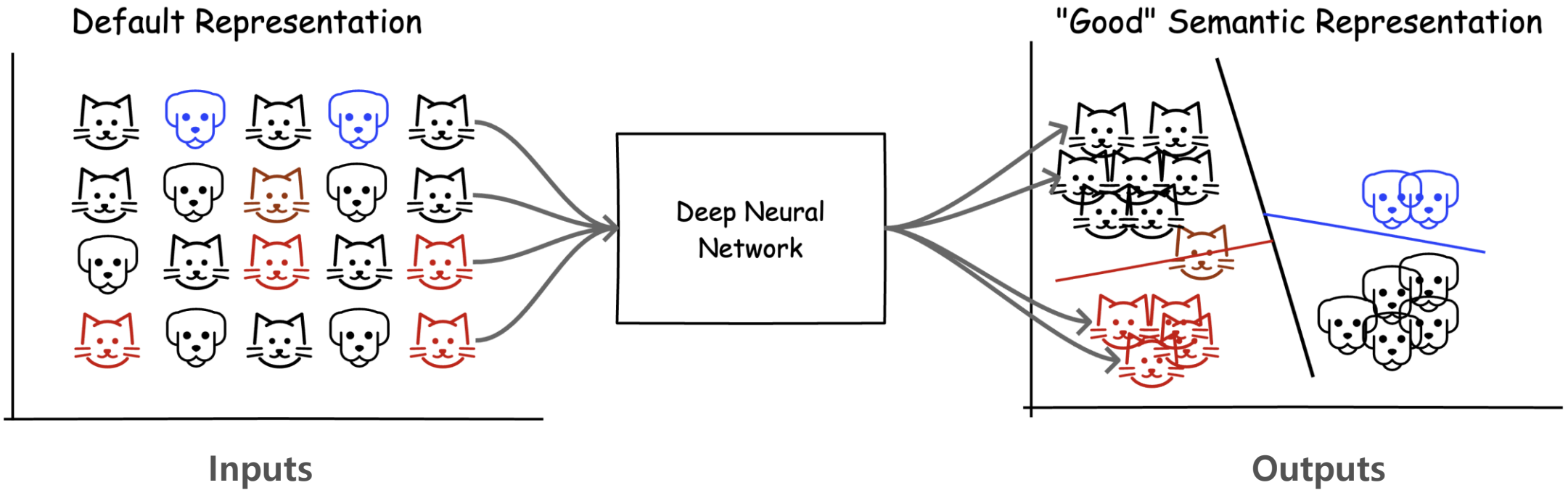
Multi-label Contrastive Predictive Coding

-Summary-

Introduction

- Is there a good way to use unlabeled data to boost performance of training model?
  - Current paradigm **[Representation learning]**
    : Use representation trained on large amounts of unlabeled images
    (turns out to improve performance on classification problem)



Introduction

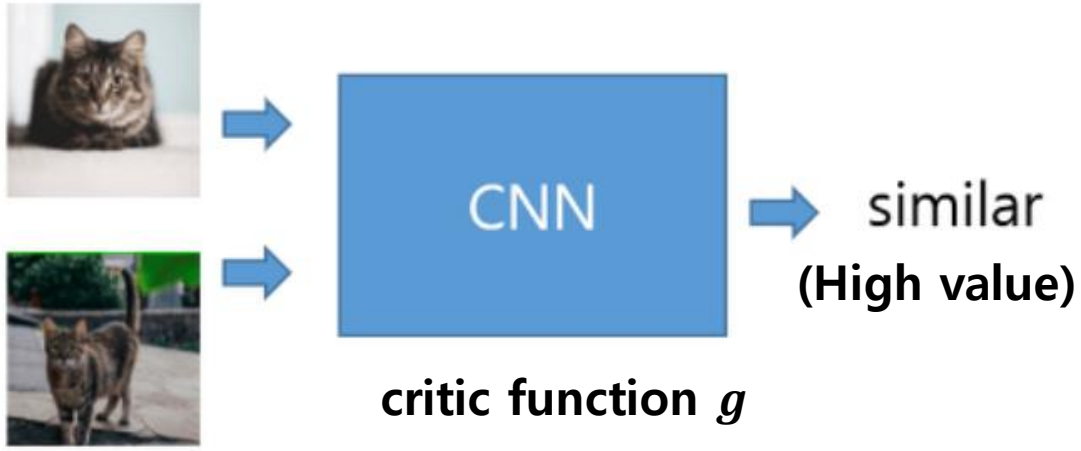
- Current well-known and successful approach : **Contrastive learning**

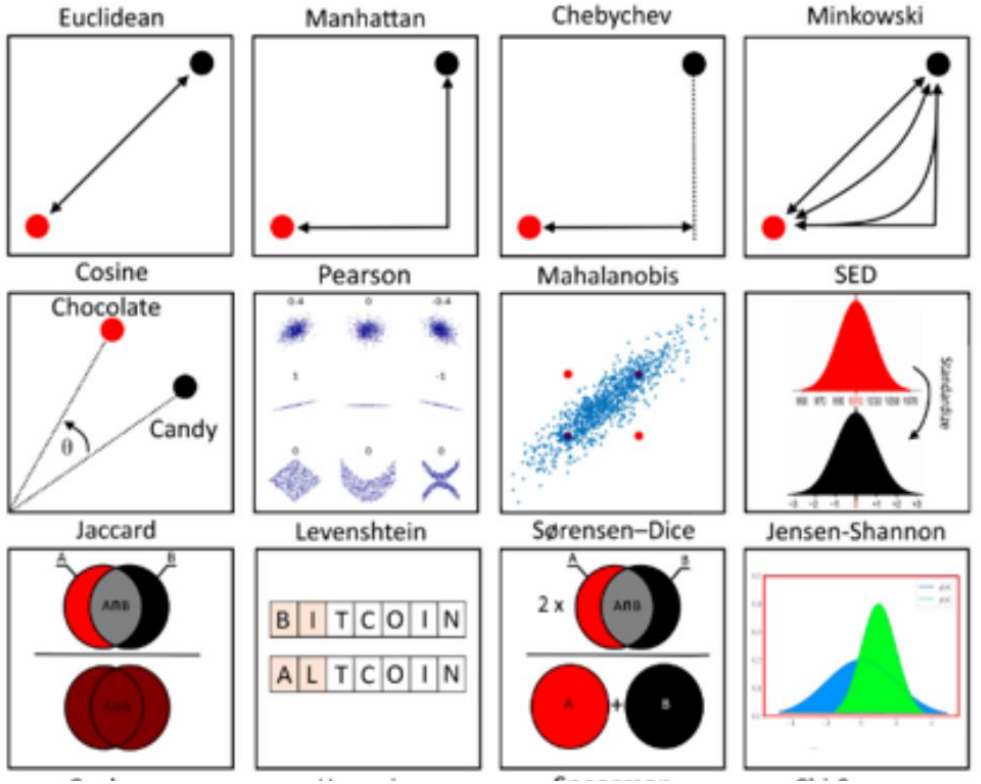
Notations & Backgrounds of Contrastive learning

- Train **'critic'** function $g: \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^+$ to distinguish '**positive**' pair $(x, y) \sim p(x, y)$ and '**negative**' pair $(x, \bar{y}) \sim p(x)p(y)$, where $\mathcal{X}, \mathcal{Y}$ = input / feature sample space

- Basic principle of 'critic' function :
  - Similar samples should have high critic value
  - dissimilar samples should have low critic value

Introduction





- Note that the metric to define similarity is not unique :
  (L2-distance, Mahalanobis distance, JS-divergence …)
- However, CPC uses mutual information (KL-divergence between $p(x, y)$ and $p(x)p(y)$)
  and achieves good performance compared to the others

Contrastive Predictive Coding

- How to use mutual information to distinguish ‘positive’ and ‘negative’ pairs?
  - **InfoMax principle** : Maximizing mutual information $\Leftrightarrow$ increasing discrepancy between $p(x, y)$ and $p(x)p(y)$
    ($\because I(X;Y) = D_{KL}(P_X||P_Y) = 0 \Leftrightarrow P_X = P_Y$ almost everywhere)

  - Suggested lower bound of mutual information on CPC : I_{NCE} [Ooord et al, 2018]

$$I_{NCE} = L(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{m \cdot g(x_i, y_i)}{g(x_i, y_i) + \sum_{j=1}^{m-1} g(x_i, \overline{y_{i,j}})}\right]$$

    where n = # of positive pairs, $m-1$ = # of negative pairs and expectation is taken over n positive pairs $(x_i, y_i) \sim p(x, y)$ and $n(m-1)$ negative pairs $\left(x_i, \overline{y_{i,j}}\right) \sim p(x)p(y)$

  - Note : $L(g)$ is called ‘**CPC objective**’

Contrastive Predictive Coding

- Now, we maximize '**CPC objective**' $L(g)$ to achieve Mutual information estimate

$$I(X;Y) \cong \max_{g} L(g)$$

- But, it has one critical drawback :
  - Observe that $L(g)$ is upper bounded by $\log m$:

$$I_{NCE} = L(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{m \cdot g(x_i, y_i)}{g(x_i, y_i) + \sum_{j=1}^{m-1} g(x_i, y_{i,j})}\right] \leq \log m$$

  - When ground-truth MI is much bigger than $\log m$, the bias between ground-truth value and our estimate becomes drastic.

Re-weighted Contrastive Predictive Coding

- To avoid this limited upper bound problem, let's reweight the positive and negative samples terms in denominator on $L(g)$ [$\boldsymbol{\alpha-}$**CPC**] : $\alpha \in (0,1)$

$$L_\alpha(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{m \cdot g(x_i, y_i)}{\alpha g(x_i, y_i) + \frac{m-\alpha}{m-1}\sum_{j=1}^{m-1} g(x_i, \overline{y_{i,j}})}\right] \leq \log\frac{m}{\alpha}$$

  Now, we can achieve higher upper bound by lowering α, which leads to low bias.

- It turns out that there is a trade-off between bias and variance :
  (High bias ⇔ Low variance, Low bias ⇔ High variance)
- Critical drawback : It is not guaranteed to be lower bound of MI

Re-weighted Contrastive Predictive Coding

- [Example] Let $X, Y \sim Bern(\frac{1}{2})$ and $P(X = 1, Y = 1) = P(X = 0, Y = 0) = 0.5$, and assume $\alpha = 0.5$, $n = m = 3$, also set $g(x, y) = 1$ if $x = y$ and 0 otherwise

1) $I(X; Y) = \sum_{X \times y} p(x, y) \log \frac{p(x,y)}{p(x)p(y)} = \log 2 \cong 0.69$

2) $L_\alpha(g) =$ (by directly computing the given formula) $\cong 0.72$

which leads to $L_\alpha(g) \geq I(X; Y)$

- Although $\alpha - \text{CPC}$ can be useful empirically, we lack some principled way to select proper values of α, as $L_\alpha(g)$ may no longer be a lower bound of $I(X; Y)$

Example [detailed description]

- [Example] Let $X, Y \sim Bern(\frac{1}{2})$ and $P(X=1, Y=1) = p,\ P(X=0, Y=0) = 1-p$ and assume batch size = n, the number of negative samples = $n-1$

1) [Calculation of CPC – Inside expectation] :

$$L(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{n \cdot g(x_i, y_i)}{\sum_{j=1}^{n} g(x_i, y_j)}\right]$$

From assumed joint distribution, $x_i = y_i$ for each $i \in [n]$. Now, assume there exists t pairs of (1,1) and $n-t$ pairs of (0,0), then

$$\frac{1}{n}\sum_{i=1}^{n}\log\frac{n \cdot g(x_i, y_i)}{\sum_{j=1}^{n} g(x_i, y_j)} = \frac{1}{n}\left(t\log\frac{n}{t} + (n-t)\log\frac{n}{n-t}\right)$$

Experiments

2) [Calculation of CPC – Taking expectation]:

Since $P(X=1,Y=1)=p,\ P(X=0,Y=0)=1-p$, we get following probability :

$$P\big(t\ pairs\ of\ (1,1)\big)=\binom{n}{t}p^t(1-p)^{n-t}$$

Finally, by taking expectation, we get $L(g)$ analytically :

$$L(g)=\sum_{t=0}^{n}\binom{n}{t}p^t(1-p)^{n-t}\times\left[\frac{1}{n}\left(t\log\frac{n}{t}+(n-t)\log\frac{n}{n-t}\right)\right]$$

- Given specified α, $\alpha-\mathrm{CPC}$, $\alpha-\mathrm{ML}-\mathrm{CPC}$ can be obtained in a similar way.

Multi-label Contrastive Predictive Coding (ML-CPC)

- To guarantee the lower-bound property, we slightly modify $L(g)$ while maintaining the computational cost [**ML-CPC**] :

$$J(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{nm\cdot g(x_i,y_i)}{\sum_{j=1}^{n} g\left(x_i,y_j\right)+\sum_{j=1}^{n}\sum_{j=1}^{m-1} g(x_i,\overline{y_{i,j}})}\right] \leq \log m$$

  where expectation is taken over n positive samples $(x_i, y_i)\sim p(x,y)$ for $i \in [n]$ and the $n(m-1)$ negative samples $\overline{y_{j,k}}\sim p(y)$ for $j \in [n]$, $k \in [m-1]$

- Recall that CPC required mn critic evaluations. Similarly, ML-CPC requires mn critic evaluations. (Also, it turns out that the computational cost for computing gradients in ML-CPC is similar to the one in CPC)

Multi-label Contrastive Predictive Coding (ML-CPC)

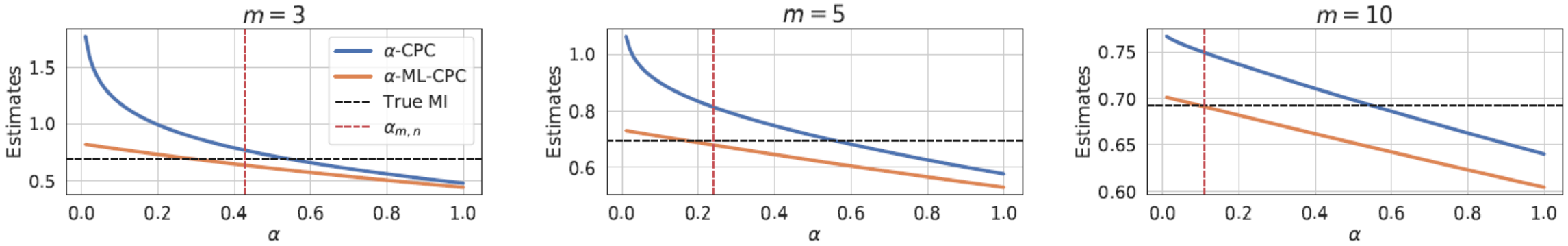
- Similarly as before, We want to increase the upper bound of ML-CPC by re-weighting the critic predictions [$\boldsymbol{\alpha-\mathrm{ML-CPC}}$]: $\alpha \in (0,1)$

$$J_\alpha(g) = \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}\log\frac{nm\cdot g(x_i,y_i)}{\alpha\sum_{j=1}^{n} g(x_i,y_j)+\frac{m-\alpha}{m-1}\sum_{j=1}^{n}\sum_{k=1}^{m-1} g(x_i,\overline{y_{j,k}})}\right] \le \log\frac{m}{\alpha}$$

- One important fact [proved in paper] :
  In contrast to $\alpha-\mathrm{CPC}$, $J_\alpha(g)$ is guaranteed to be a lower bound of $I(X;Y)$ for a wide range of α (specifically, $\alpha \in [\alpha_{m,n}, 1]$, where $\alpha_{m,n} = \frac{m}{n(m-1)+1}$)

Multi-label Contrastive Predictive Coding (ML-CPC)

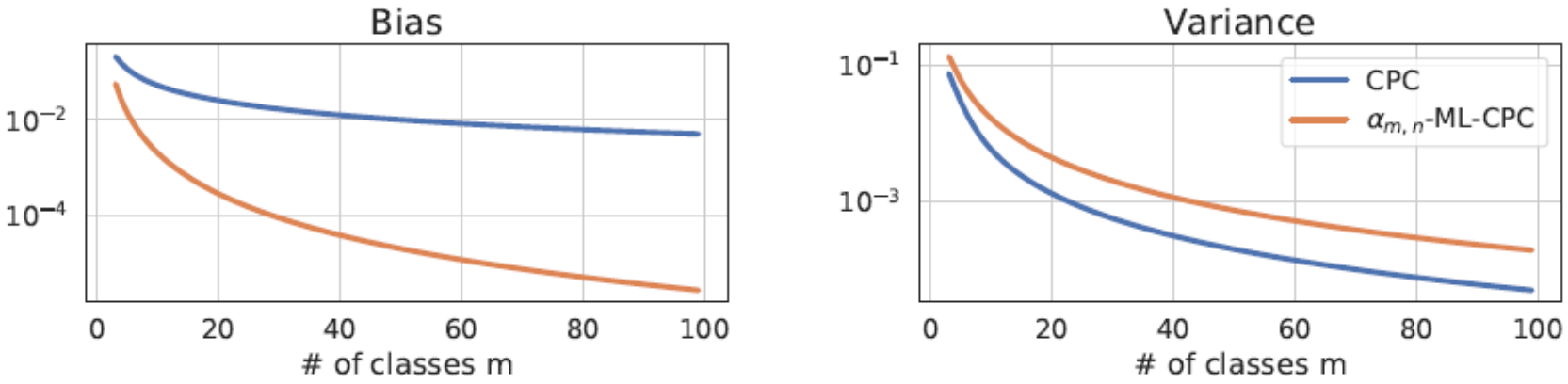
- One important fact [proved in paper] :
  In contrast to $\alpha - \mathrm{CPC}$, $J_\alpha(g)$ is guaranteed to be a lower bound of $I(X;Y)$ for a wide range of α (specifically, $\alpha \in [\alpha_{m,n}, 1]$, where $\alpha_{m,n} = \frac{m}{n(m-1)+1}$)



MI estimates with CPC and ML-CPC under different α

Side Notes

- Q : Is the ML-CPC always better than CPC in terms of bias and variance?
  A: No, There is a trade off between them
  (CPC : higher bias, but smaller variance compared to ML-CPC)



Bias-variance trade-offs for different m

- Q : What about optimal critic for CPC / ML-CPC?

  A : $g^*(x,y) = c(x) \cdot \frac{p(x,y)}{p(x)p(y)}$ for CPC / $g^*(x,y) = c \cdot \frac{p(x,y)}{p(x)p(y)}$ for ML-CPC

  (Observe that ML-CPC optimal critic forces the similarity of any positive pair to be higher than that of any negative pair in contrast to CPC optimal critic)

Experiments

- Experiments setup:
  - Let $(X, Y) \sim MN(0, \Sigma)$, where

$$K = \begin{bmatrix} \sigma^2 I_d & \rho\sigma^2 I_d \\ \rho\sigma^2 I_d & \sigma^2 I_d \end{bmatrix}$$

    and $X, Y \in \mathbb{R}^d$ respectively. (i.e (X, Y) follows correlated standard multivariate normal)

  - Then, $X, Y \sim MN(0, I_d)$ and (differential) entropy $h(X) = h(Y) = \frac{1}{2}\log\left[(2\pi e)^d \sigma^{2d}\right]$
  - And, $h(X, Y) = \frac{1}{2}\log(2\pi e)^{2d} \det(K) = \frac{1}{2}\log\left[(2\pi e)^{2d}\left(1 - \rho^2\sigma^2\right)^d \sigma^{2d}\right]$
  - Hence, $I(X; Y) = h(X) + h(Y) - h(X, Y) = \frac{1}{2}\log\left[\left(1 - \rho^2\sigma^2\right)^{-d} \sigma^{2d}\right]$
  - If $\sigma = 1$, $I(X; Y) = -\frac{d}{2}\log(1 - \rho^2)$

Experiments

- Experiments setup:
  - During experiment ρ increase by 2 every 4k iterations (total 20k iterations)
  - Adopt two types of critic : joint / separable

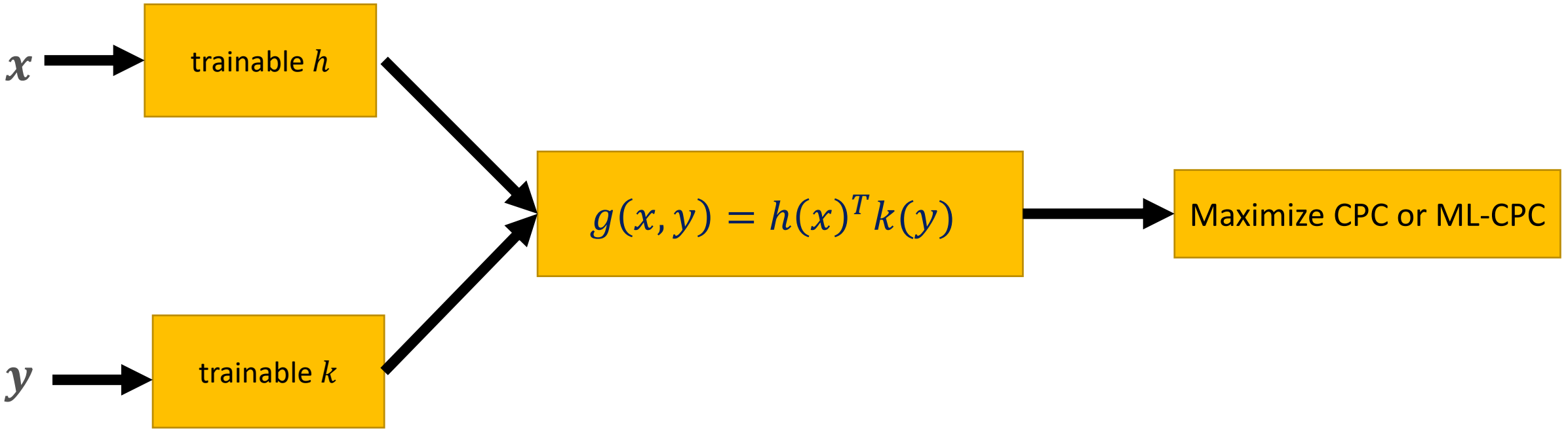
  - Our goal : estimate $I(X;Y)$ using CPC and ML-CPC with various choice of α

- **[Background]**
  two types of architecture for critic function g (separable / joint) :
  - **Separable architecture** : $g(x,y) = h(x)^T k(y)$ for some function h, k
    (Requires $2 \times batch\ size$ forward pass per batch) => much computationally efficient

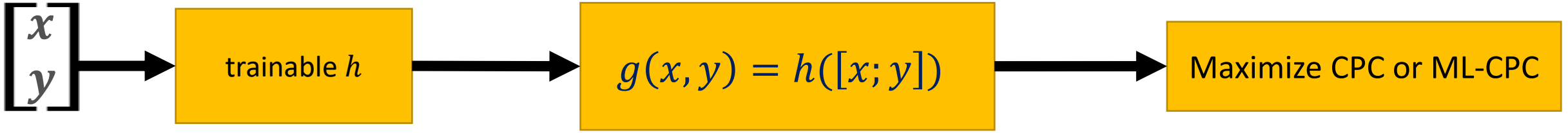
  - **Joint architecture** : $g(x,y) = h([x;y])$ where $[x;y]$ implies concatenation of x, y
    (Requires $batch\ size^2$ forward pass per batch)

Experiments

- Example (separable architecture)



- Example (Joint architecture)

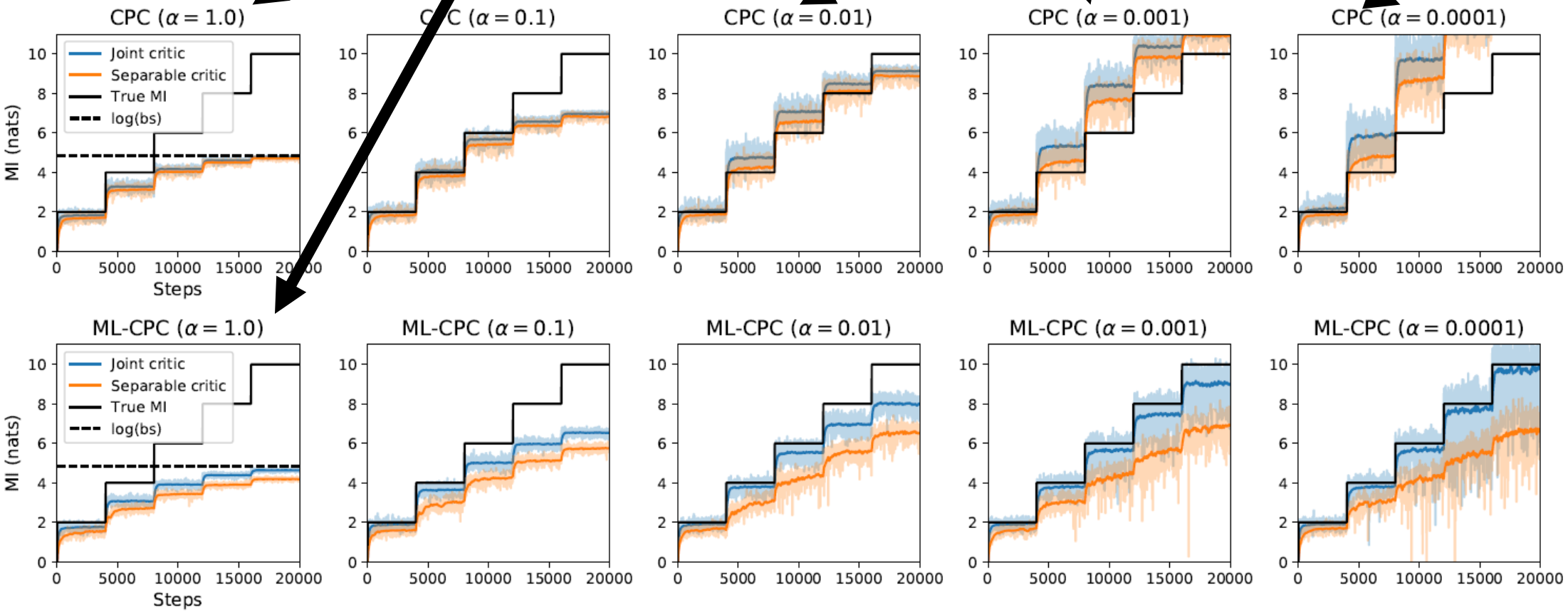


Experiments (

Upper bound by $\log m$

Lower bound property broke

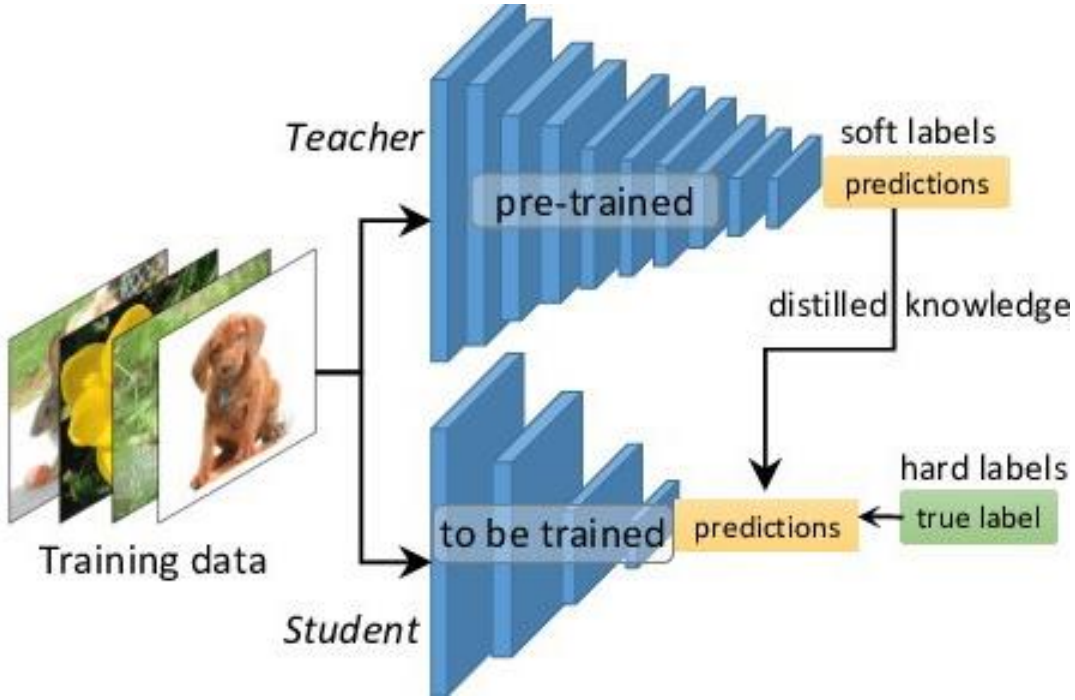
- (Back to) Experiment results:
  - (Recall) Our goal : estimate $I(X;Y)$ using CPC and ML-CPC with various choice of α



MI estimation with CPC and ML-CPC with different choices of α

Experiments (Knowledge distillation)

- **[Background] :**
  - Knowledge distillation
    = one network (called teacher) transfer its knowledge to another (typically smaller) model (called student) so that the student' performance is higher than training from labeled data alone.

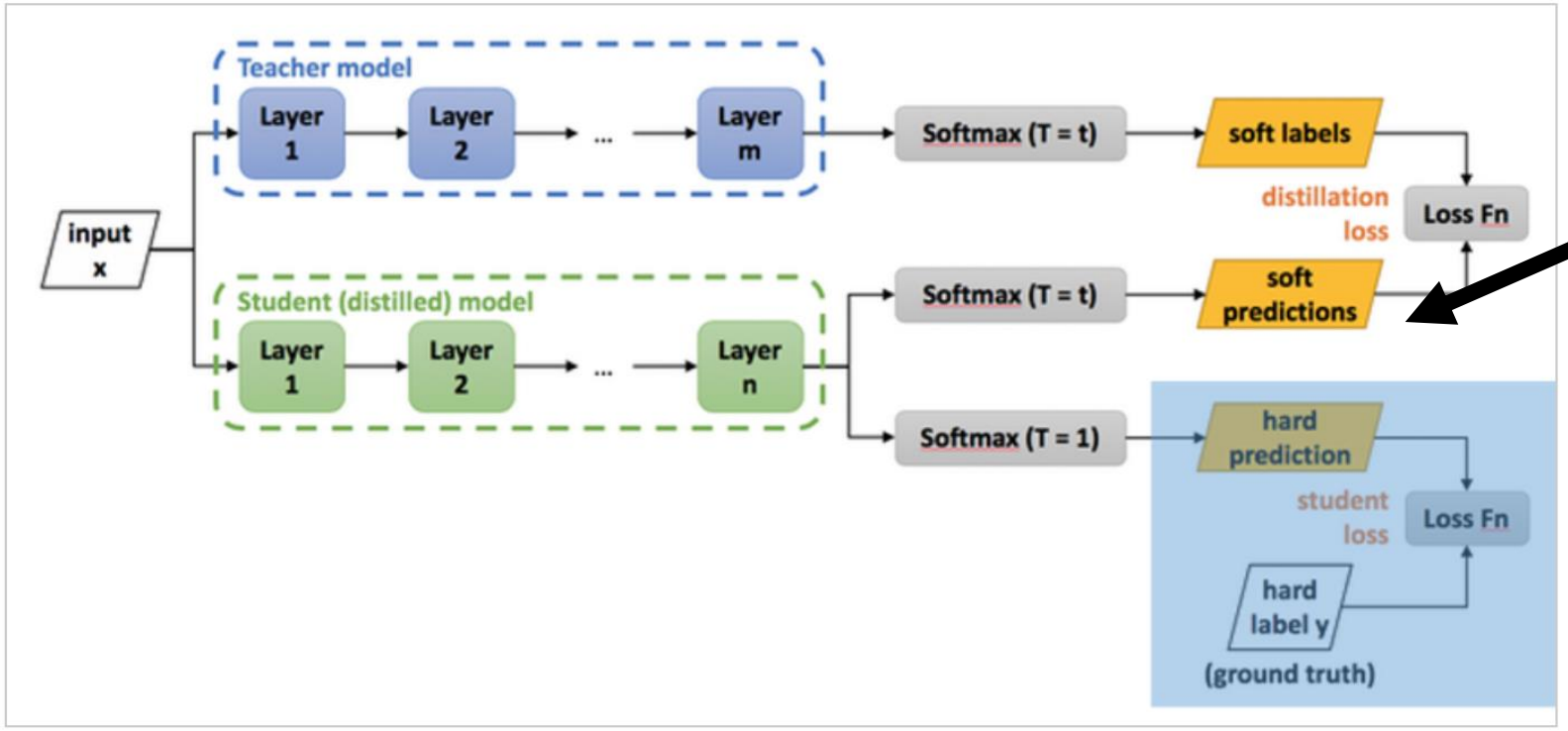


Flow of Knowledge distillation

Current SOTA methodology :
Contrastive representation distillation (CRD)

Experiments (Knowledge distillation)

- **[Background] :**
  - Knowledge distillation (Detailed flow)



$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

Soft prediction with T

$$Softmax\begin{pmatrix}1\\2\\9\end{pmatrix} = \begin{pmatrix}0.000335\\0.000911\\0.998754\end{pmatrix}, \quad Softmax_{T=3}\begin{pmatrix}1\\2\\3\end{pmatrix} = \begin{pmatrix}0.059\\0.083\\0.857\end{pmatrix}$$

Loss on KD : $L = \sum_{(x,y)\in D} L_{KD}\big(S(x;\theta_s), T(x,\theta_T)\big) + \lambda \cdot L_{CE}(\hat{y}_s, y)$

(Here, $\hat{y}_s$ = top-1 class of $T(x,\theta_T)$ => Hard prediction of student w.r.t x)

Experiments (Knowledge distillation)

- **[Background] :**
  - Contrastive representation distillation (CRD):
    regularizes the student model so that its features have higher mutual information with that of the teacher

  - Implementation of CRD :
    Use '**noisy contrastive estimation objective**' (NCE) [Gutmann., 2012] :
    Let p_m and p_n be data distribution / noise injected data distribution, and $(x_1, \dots, x_T), (y_1, \dots y_T)$ be data / noise injected data. Then, NCE objective :

$$J_{NCE} = \frac{1}{2T} \sum_{t=1}^{T} \ln[h(x_t; \theta)] + \ln[1 - h(y_t; \theta)]$$

    where $h(x; \theta) = logistic(\log p_m(x; \theta) - \log p_n(x))$

Experiments (Knowledge distillation)

- In this experiment, we replace NCE objective to CPC / ML-CPC objective with various α :

	Teacher Student	WRN-40-2 WRN-16-2	WRN-40-2 WRN-40-1	resnet56 resnet20	resnet110 resnet20	resnet110 resnet32	resnet32x4 resnet8x4	vgg13 vgg8
	Teacher	75.61	75.61	72.34	74.31	74.31	79.42	74.64
	Student	73.26	71.98	69.06	69.06	71.14	72.50	70.36
	KD	74.92	73.54	70.66	70.67	73.08	73.33	72.98
	CRD	75.48	74.14	71.16	**71.46**	73.48	**75.51**	73.94
CPC	$L_{1.0}$	75.42 (↓)	74.16 (↑)	71.32 (↑)	71.39 (↓)	73.57 (↑)	75.50 (↓)	73.60 (↓)
CPC	$L_{0.1}$	75.69 (↑)	74.17 (↑)	71.48 (↑)	71.38 (↓)	73.66 (↑)	75.41 (↓)	73.61 (↓)
ML-CPC	$J_{1.0}$	75.39 (↓)	74.18 (↑)	71.28 (↑)	71.28 (↓)	73.58 (↑)	75.32 (↓)	73.67 (↓)
ML-CPC	$J_{0.05}$	75.64 (↑)	**74.27** (↑)	71.33 (↑)	71.24 (↓)	73.57 (↑)	75.50 (↓)	**74.01** (↑)
ML-CPC	$J_{0.01}$	**75.83** (↑)	74.24 (↑)	**71.50** (↑)	71.27 (↓)	**73.90** (↑)	75.37 (↓)	73.95 (↑)

Top-1 test accuracy of student networks on CIFAR-100 (same type of model)

Experiments (Knowledge distillation)

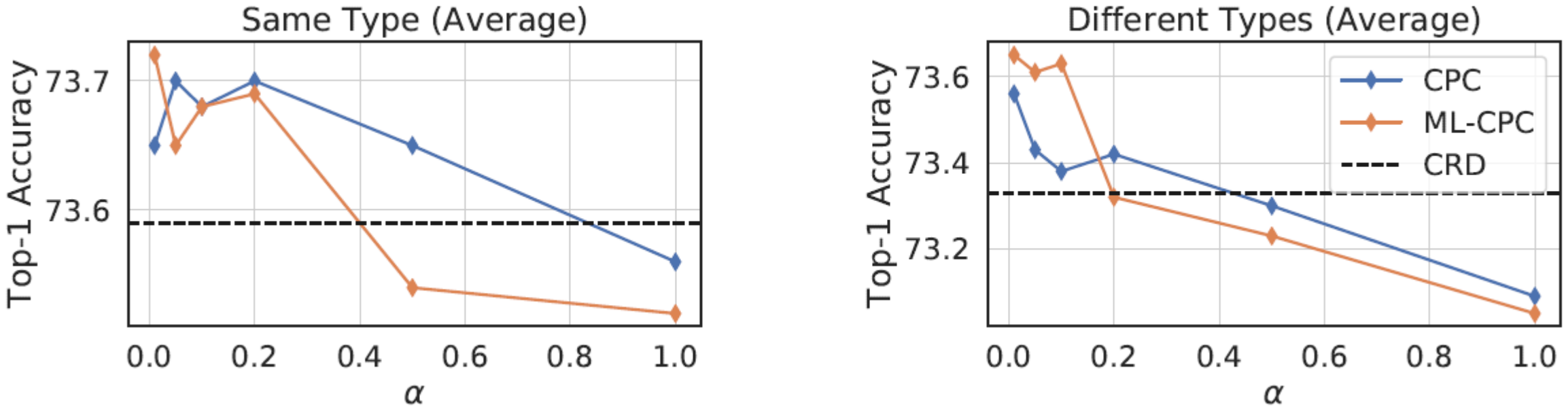
- In this experiment, we replace NCE objective to CPC / ML-CPC objective with various α :

	Teacher Student	vgg13 MobileNetV2	ResNet50 MobileNetV2	ResNet50 vgg8	resnet32x4 ShuffleNetV1	resnet32x4 ShuffleNetV2	WRN-40-2 ShuffleNetV1
	Teacher	74.64	79.34	79.34	79.42	79.42	75.61
	Student	64.60	64.60	70.36	70.50	71.82	70.50
	KD	67.37	67.35	73.81	74.07	74.45	74.83
	CRD	**69.73**	69.11	74.30	75.11	75.65	76.05
CPC	$L_{1.0}$	69.24 (↓)	69.02 (↓)	73.66 (↓)	75.00 (↓)	75.93 (↑)	75.72 (↓)
CPC	$L_{0.1}$	69.26 (↓)	69.33 (↑)	74.24 (↓)	75.34 (↑)	76.01 (↑)	76.12 (↑)
ML-CPC	$J_{1.0}$	68.92 (↓)	68.80 (↓)	73.65 (↓)	75.39 (↑)	75.88 (↑)	75.70 (↓)
ML-CPC	$J_{0.05}$	69.25 (↓)	**70.04** (↑)	**74.84** (↑)	75.51 (↑)	76.24 (↑)	76.03 (↑)
ML-CPC	$J_{0.01}$	69.25 (↓)	69.90 (↑)	74.81 (↑)	**75.47** (↑)	**76.04** (↑)	**76.19** (↑)

Top-1 test accuracy of student networks on CIFAR-100 (different type of model)

Experiments (Knowledge distillation)

- Ablation study : Knowledge distillation with CPC and ML-CPC with various α



Top-1 accuracy according to different α
(Left : student - teacher are of same model / Right : student - teacher are from different model)

Experiments (Representation Learning)

- Naturally, We can use CPC objective to perform representation learning
- And, One famous method adopting CPC objective is MoCo:

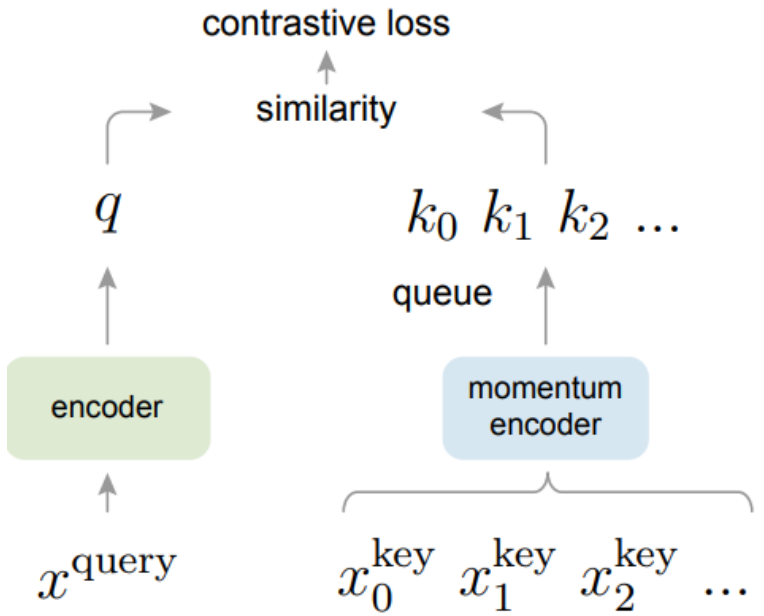
- **[Background]** MoCo (Momentum Contrast)
  Dictionary look-up contrastive learning using dynamic dictionary with a queue and a moving-averaged encoder

  1) Use InfoNCE contrastive loss : (K : length of queue)

  $$L_q = -\log \frac{\exp(q \cdot k_+ / \tau)}{\sum_{i=0}^{K} \exp(q \cdot k_i / \tau)}$$ (similar to CPC objective)

  2) Update query network using SGD and update key network using momentum :

  $$\theta_k = m\theta_k + (1-m)\theta_q$$



Experiments (Representation Learning)

- **[Background]** MoCo (Momentum Contrast) V1 [Pseudo Code]

Algorithm 1 Pseudocode of MoCo in a PyTorch-like style.

```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

f_k.params = f_q.params # initialize
for x in loader: # load a minibatch x with N samples
   x_q = aug(x) # a randomly augmented version
   x_k = aug(x) # another randomly augmented version

   q = f_q.forward(x_q) # queries: NxC
   k = f_k.forward(x_k) # keys: NxC
   k = k.detach() # no gradient to keys

   # positive logits: Nx1
   l_pos = bmm(q.view(N,1,C), k.view(N,C,1))

   # negative logits: NxK
   l_neg = mm(q.view(N,C), queue.view(C,K))

   # logits: Nx(1+K)
   logits = cat([l_pos, l_neg], dim=1)

   # contrastive loss, Eqn.(1)
   labels = zeros(N) # positives are the 0-th
   loss = CrossEntropyLoss(logits/t, labels)

   # SGD update: query network
   loss.backward()
   update(f_q.params)

   # momentum update: key network
   f_k.params = m*f_k.params+(1-m)*f_q.params

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

bmm: batch matrix multiplication; mm: matrix multiplication; cat: concatenation.

Data augmentation of batch (flip, noise …)

Forward process using key / query network

Calculate $q \cdot k^+$ and $q \cdot k_i$ for $i = 1 \dots K$

Calculate L_q using CrossEntropy

Update key / query network

Enqueue incoming batch / Dequeue oldest batch

Experiments (Representation Learning)

- Naturally, We replace the L_q (which is essentially the same as the CPC objective) by ML-CPC with various α
- One heuristic is to use α-scheduler : (improves test acc ~ 2.5% for 200 epoch) (ex : $\alpha = 10 \rightarrow 0.1$ for front half epoch, $\alpha = 1$ for behind half epoch)
- Use 'linear evaluation protocol' : evaluate learned representation by test top-1 acc when a model is trained to predict labels from representations

(a) CIFAR-10

	Epochs	200	500	1000
CPC	$L_{1.0}$	83.28	89.31	91.20
ML-CPC	$J_{1.0} \rightarrow J_{1.0}$	83.61 (↑)	89.43 (↑)	91.48 (↑)
	$J_{2.0} \rightarrow J_{0.5}$	84.31 (↑)	89.47 (↑)	91.43 (↑)
	$J_{5.0} \rightarrow J_{0.2}$	85.52 (↑)	**89.85** (↑)	91.50 (↑)
	$J_{10.0} \rightarrow J_{0.1}$	**86.16** (↑)	89.49 (↑)	**91.86** (↑)

(b) CIFAR-100

Epochs	200	500	1000
$L_{1.0}$	61.42	67.72	69.63
$J_{1.0} \rightarrow J_{1.0}$	61.80 (↑)	67.68 (↓)	**70.85** (↑)
$J_{2.0} \rightarrow J_{0.5}$	62.92 (↑)	68.01 (↑)	70.22 (↑)
$J_{5.0} \rightarrow J_{0.2}$	63.58 (↑)	**68.04** (↑)	70.07 (↑)
$J_{10.0} \rightarrow J_{0.1}$	**64.05** (↑)	67.94 (↑)	70.03 (↑)

Top-1 accuracy of unsupervised representation learning (setting = MoCo V2 with ML-CPC)