i-Mix

A DOMAIN-AGNOSTIC STRATEGY FOR CONTRASTIVE REPRESENTATION LEARNING

By MohamedElfatih MohamedElkhair

Abstract

- Contrastive learning approaches are well designed for vision domains only.

- Combine Contrastive learning approaches with Mixup.

- Improve the performance of contrastive learning approaches in across domains (image, speech, tabler)

What is contrastive learning?

Augmentation

Encoder

h_i

Maximize Agreement

h_i

SimCLR

What is Mixup?

0.4 x  + 0.6 x  = 

Cat: 1.0
Dog: 0.0

Cat: 0.0
Dog: 1.0

Cat: 0.4
Dog: 0.6

How to Mixup is applied in Supervised setting?

$$\ell_{\text{Sup}}(x_i, y_i) = -\sum_{c=1}^{C} y_{i,c} \log \frac{\exp(w_c^\top f_i)}{\sum_{k=1}^{C} \exp(w_k^\top f_i)}$$

$$\ell_{\text{Sup}}^{\text{MixUp}}\big((x_i, y_i), (x_j, y_j); \lambda\big) = \ell_{\text{Sup}}(\lambda x_i + (1-\lambda)x_j, \lambda y_i + (1-\lambda)y_j)$$

Then….

How to apply in Self-Supervised settings?

$$\mathbf{Mix}(x_i, x_j; \lambda) = \lambda x_i + (1-\lambda) x_j$$

$$\ell^{i\text{-Mix}}\big((x_i, v_i), (x_j, v_j); \mathcal{B}, \lambda\big) = \ell(\mathbf{Mix}(x_i, x_j; \lambda), \lambda v_i + (1-\lambda) v_j; \mathcal{B})$$

$$\mathbf{CutMix}(x_i, x_j; \lambda) = M_\lambda \odot x_i + (1-M_\lambda) \odot x_j$$

Contrastive Learning Approaches

- SimCLR
- Moco
- Byol

$$\ell_{\text{SimCLR}}(x_i; \mathcal{B}) = -\log \frac{\exp\left(s(f_i, f_{(N+i) \bmod 2N})/\tau\right)}{\sum_{k=1, k\neq i}^{2N} \exp\left(s(f_i, f_k)/\tau\right)}$$

$$\ell_{\text{N-pair}}(x_i, v_i; \mathcal{B}) = -\sum_{n=1}^{N} v_{i,n} \log \frac{\exp\left(s(f_i, \tilde{f}_n)/\tau\right)}{\sum_{k=1}^{N} \exp\left(s(f_i, \tilde{f}_k)/\tau\right)}$$

$$\ell_{\text{N-pair}}^{i\text{-Mix}}\left((x_i, v_i), (x_j, v_j); \mathcal{B}, \lambda\right) = \ell_{\text{N-pair}}(\lambda x_i + (1-\lambda)x_j, \lambda v_i + (1-\lambda)v_j; \mathcal{B})$$

Pseudocode

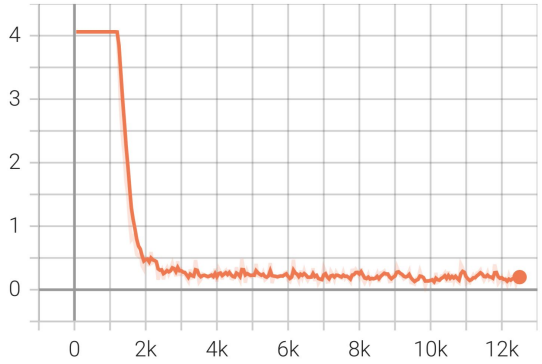
Algorithm 1 Loss computation for i-Mix on N-pair contrastive learning in PyTorch-like style.

```
a, b = aug(x), aug(x) # two different views of input x
lam = Beta(alpha, alpha).sample() # mixing coefficient
randidx = randperm(len(x))
a = lam * a + (1-lam) * a[randidx]
logits = matmul(normalize(model(a)), normalize(model(b)).T) / t
loss = lam * CrossEntropyLoss(logits, arange(len(x))) + \
       (1-lam) * CrossEntropyLoss(logits, randidx)
```

Results

NPair

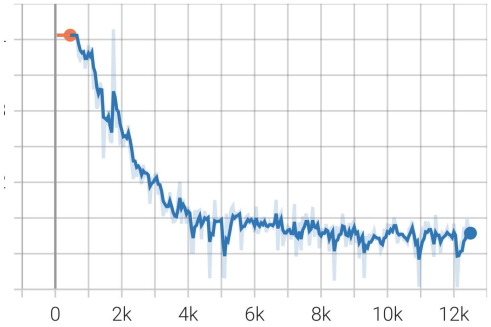
pretrain_loss



Test ACC 67.7

NPair + imix

pretrain_loss



Test ACC 74.8

Comparing with the paper

Test Accuracy

Implementation		Paper	
Npair	+ imix	Npair	+ imix
67.7	74.8	68.5	72.1

Future Works

- Implementing BYOL, MOCO
- Experimenting in Speech commands and CIFAR
- Organizing the code

ANY QUESTIONS?