AMMI_Paper_Implemntaion-1

i-MIX : A DOMAIN -A GNOSTIC STRATEGY

FOR CONTRASTIVE REPRESENTATION LEARNING

By: Amna Elmustafa

01

Core Idea

Why?

Domain Agnostic regularization/Augmentation in SSL problems with small data_size

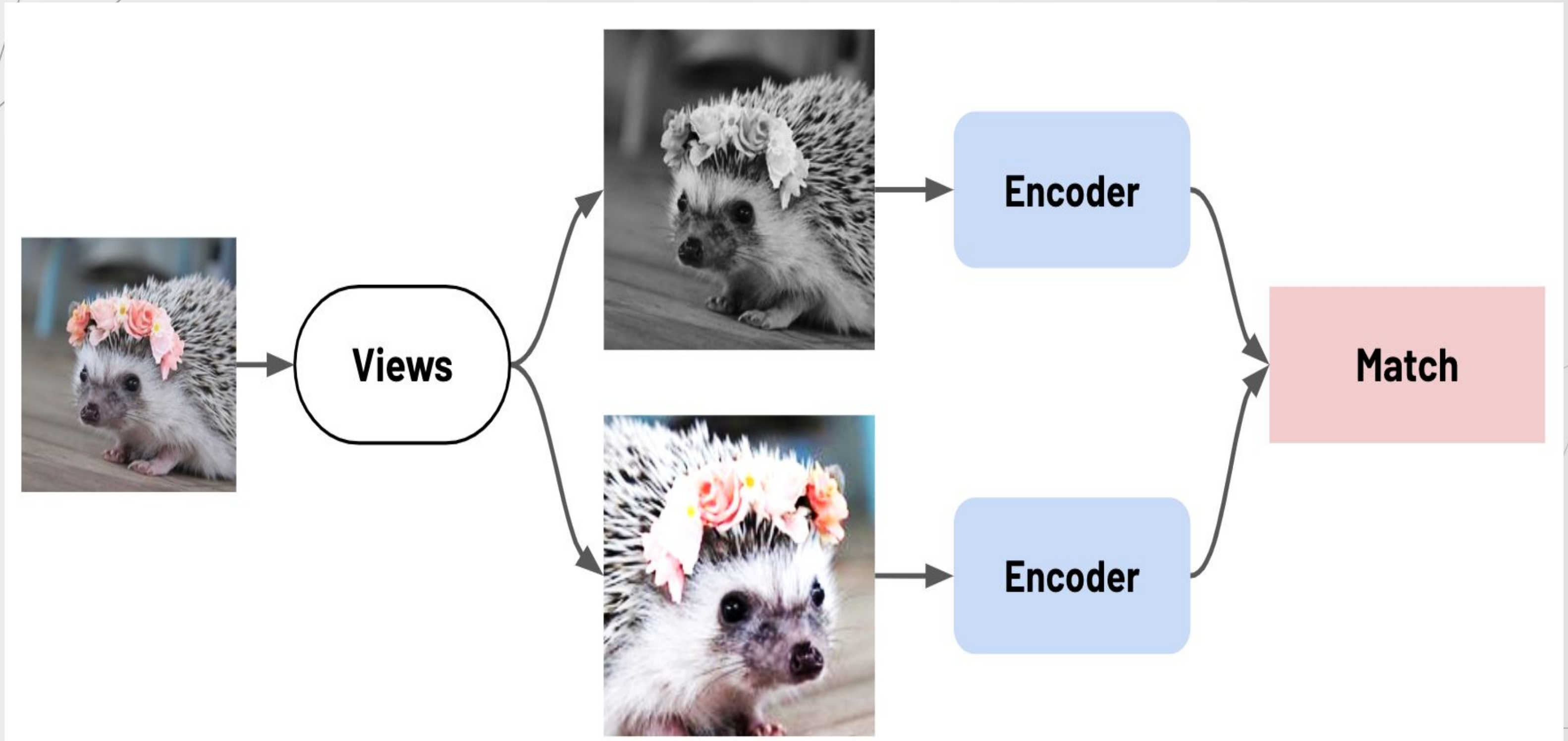
Better Representation of data → Better Performance at downstream Tasks

Small data Size

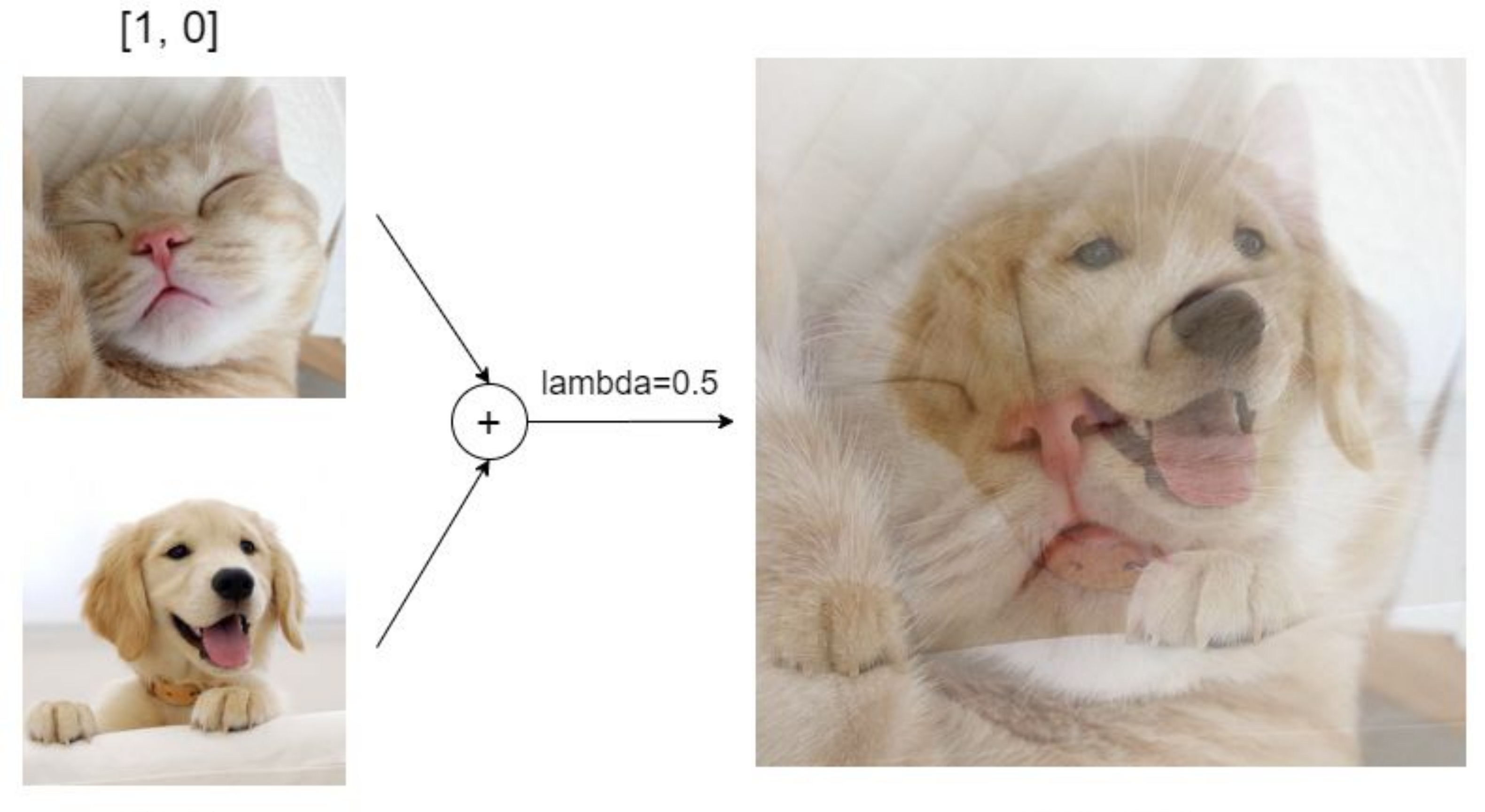
Methodology

- Combines Mixup augmentation with 3 different contrastive learning methods
- Methods are : simCLR , Moco & Byol
- Virtual labels rather than real labels representing location of the sample in the batch

What is contrastive learning



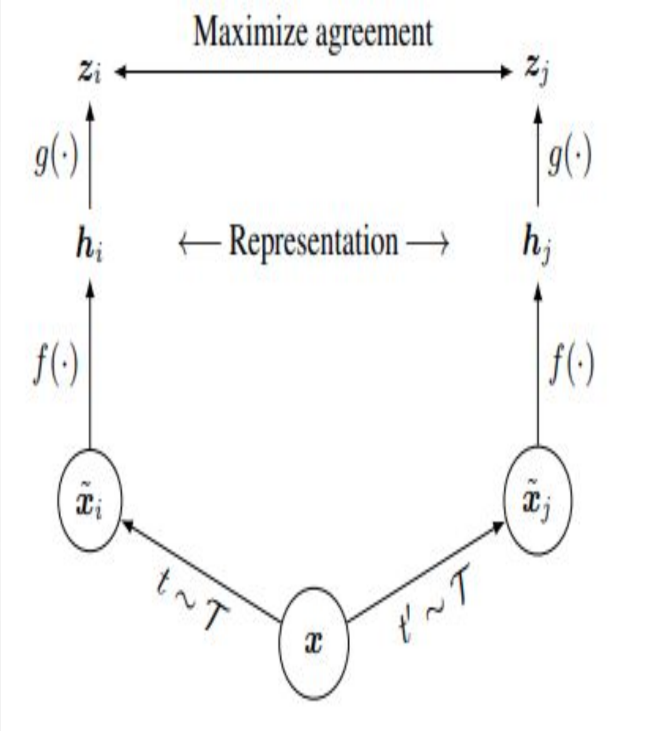
What is mixup ?



Contrastive Learning Methods

SimCLR (Contrastive Learning of Visual Representations)

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



Contrastive Learning Methods

Npair

SimCLR with N pairs rather than 2N

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

I-Mix loss

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

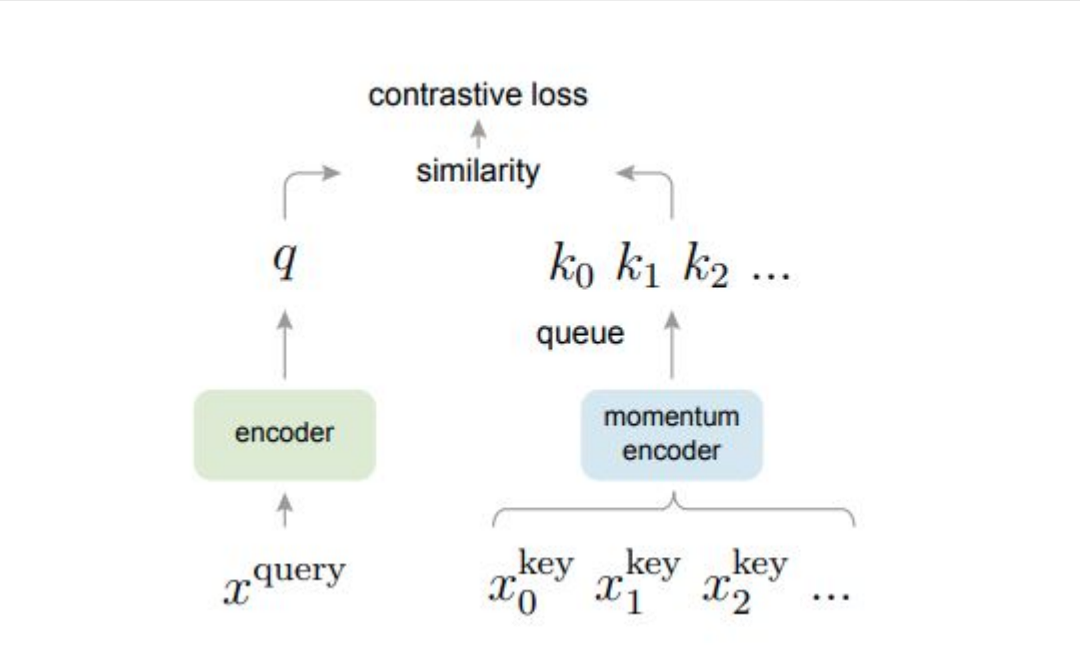
Contrastive Learning Methods

Npair + i-Mix ALgorithm

```
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)
```

Contrastive Learning Methods

Moco(Momentum Contrast for Unsupervised Visual Representation)



Contrastive Learning Methods

Moco **(Memory Bank method)**

$$\ell_{\text{MoCo}}(x_i; \mathcal{B}, \mathcal{M}) = -\log \frac{\exp\left(s(f_i, \tilde{f}_i^{\text{EMA}})/\tau\right)}{\exp\left(s(f_i, \tilde{f}_i^{\text{EMA}})/\tau\right) + \sum_{k=1}^{K} \exp\left(s(f_i, \mu_k)/\tau\right)}.$$

Moco v2 **(adapted in the paper)**

$$\ell_{\text{MoCo}}(x_i, \tilde{v}_i; \mathcal{B}, \mathcal{M}) = -\sum_{n=1}^{N} \tilde{v}_{i,n} \log \frac{\exp\left(s(f_i, \tilde{f}_n^{\text{EMA}})/\tau\right)}{\sum_{k=1}^{N} \exp\left(s(f_i, \tilde{f}_k^{\text{EMA}})/\tau\right) + \sum_{k=1}^{K} \exp\left(s(f_i, \mu_k)/\tau\right)}.$$

(

i-Mix loss

$$\ell_{\text{MoCo}}^{\text{i-Mix}}\left((x_i, \tilde{v}_i), (x_j, \tilde{v}_j); \mathcal{B}, \mathcal{M}, \lambda\right) = \ell_{\text{MoCo}}\left(\lambda x_i + (1-\lambda)x_j, \lambda\tilde{v}_i + (1-\lambda)\tilde{v}_j; \mathcal{B}, \mathcal{M}\right).$$

Contrastive Learning Methods

Moco Algorithm

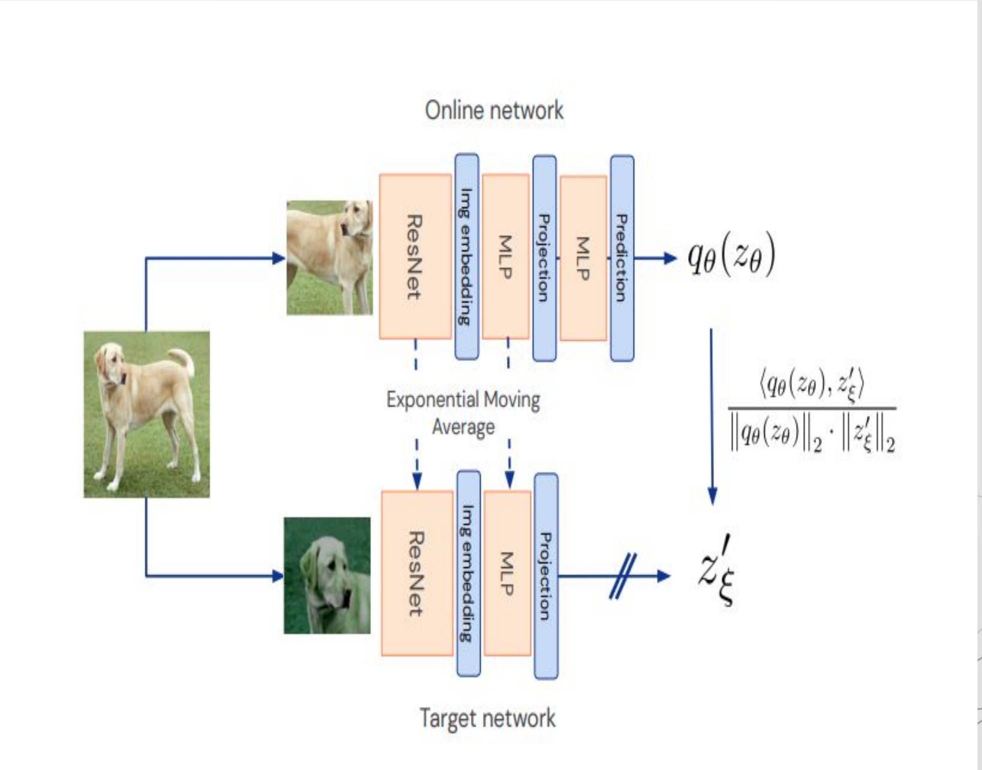
```
# f_q, f_k: encoder networks for query and key
# queue: dictionary as a queue of K keys (CxK)
# m: momentum
# t: temperature

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
```

Contrastive Learning Methods

BYOL (BootStrap your own Latent)

No Negative pairs

Predictive layer for positive examples to maximize similarity



Contrastive Learning Methods

Byol loss

$$\ell_{\text{BYOL}}(x_i, v_i; \mathcal{B}) = \left\| g(f_i)/\|g(f_i)\| - \tilde{F}v_i \right\|^2 = 2 - 2 \cdot s(g(f_i), \tilde{F}v_i).$$

i-Mix loss

$$= \lambda \ell_{\text{BYOL}}(\lambda x_i + (1-\lambda)x_j, v_i; \mathcal{B}) + (1-\lambda)\ell_{\text{BYOL}}(\lambda x_i + (1-\lambda)x_j, v_j; \mathcal{B})$$

02
Approach

Steps

1 Implement the baseline Loss Function (N-pair)

2 Implement the augmentation Method

3 Build Model Architecture

Steps

4 Pre-Training On the N-pair Loss

5 Fine Tune a linear classifier

6 Implement Npair+i-Mix loss -Experiment again

Experiment details

Tabular data used

15k training -566k Test

Pretriain -Finetune

To investigate the effect of regularization longer training is done

500 epochs /512 batch

5 layer MLP +2 projection heads

With batch normalization & maxout layer

03

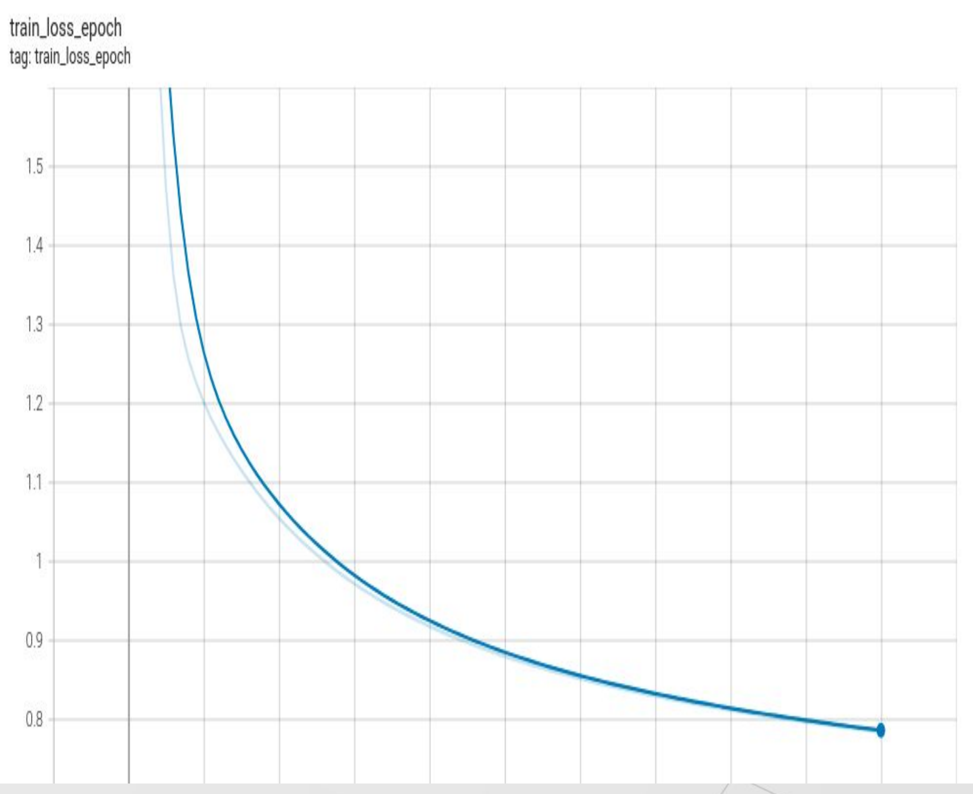
Results

What is done so far?

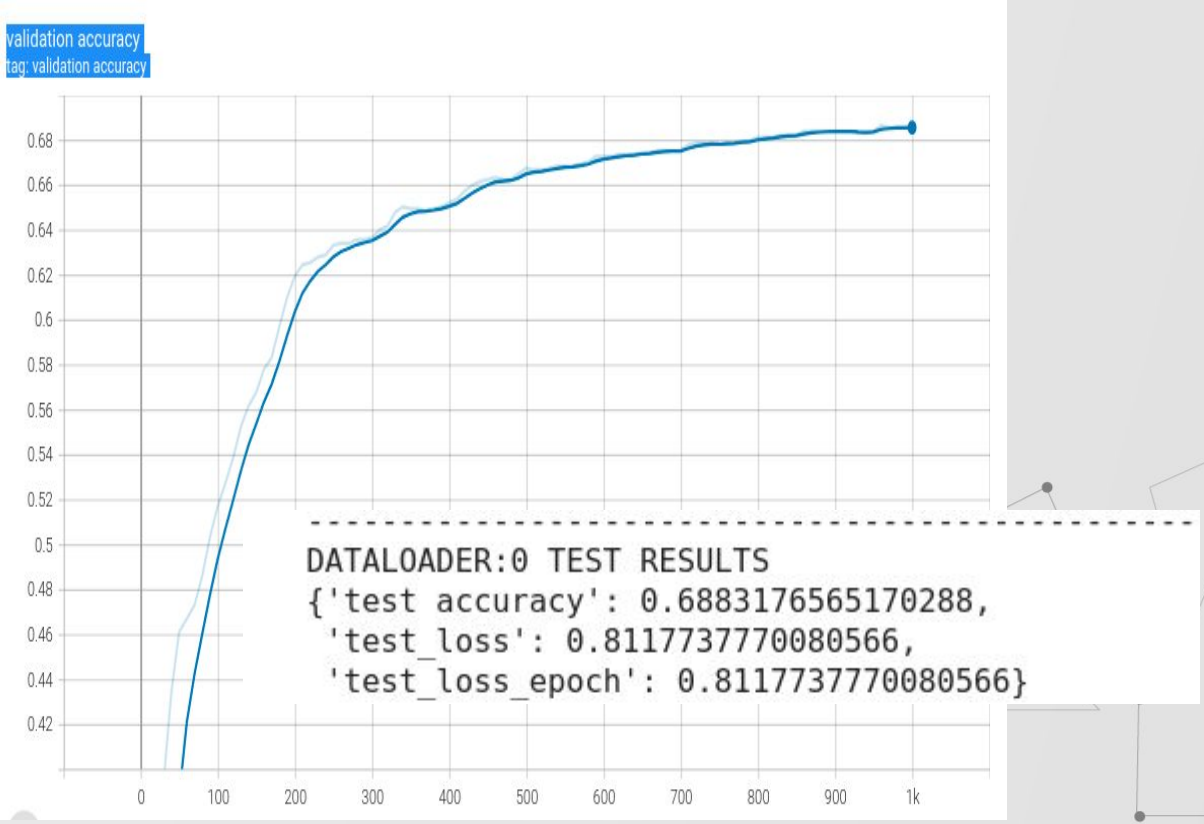
Original Paper Results

Domain	Dataset	N-pair	+ i-Mix
Image	CIFAR-10	93.3 $\pm$ 0.1	**95.6** $\pm$ 0.2
	CIFAR-100	70.8 $\pm$ 0.4	**75.8** $\pm$ 0.3
Speech	Commands	94.9 $\pm$ 0.1	**98.3** $\pm$ 0.1
Tabular	CovType	68.5 $\pm$ 0.3	**72.1** $\pm$ 0.2

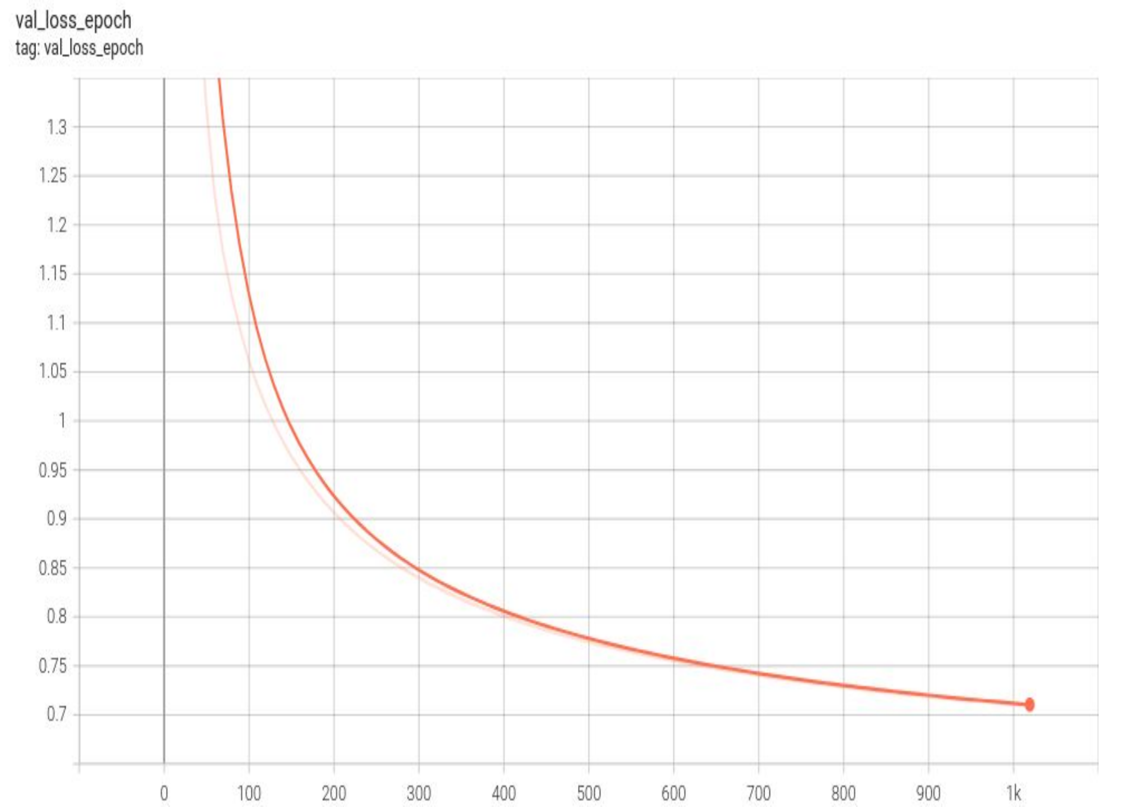
Our Results-Baseline (N-Pair)



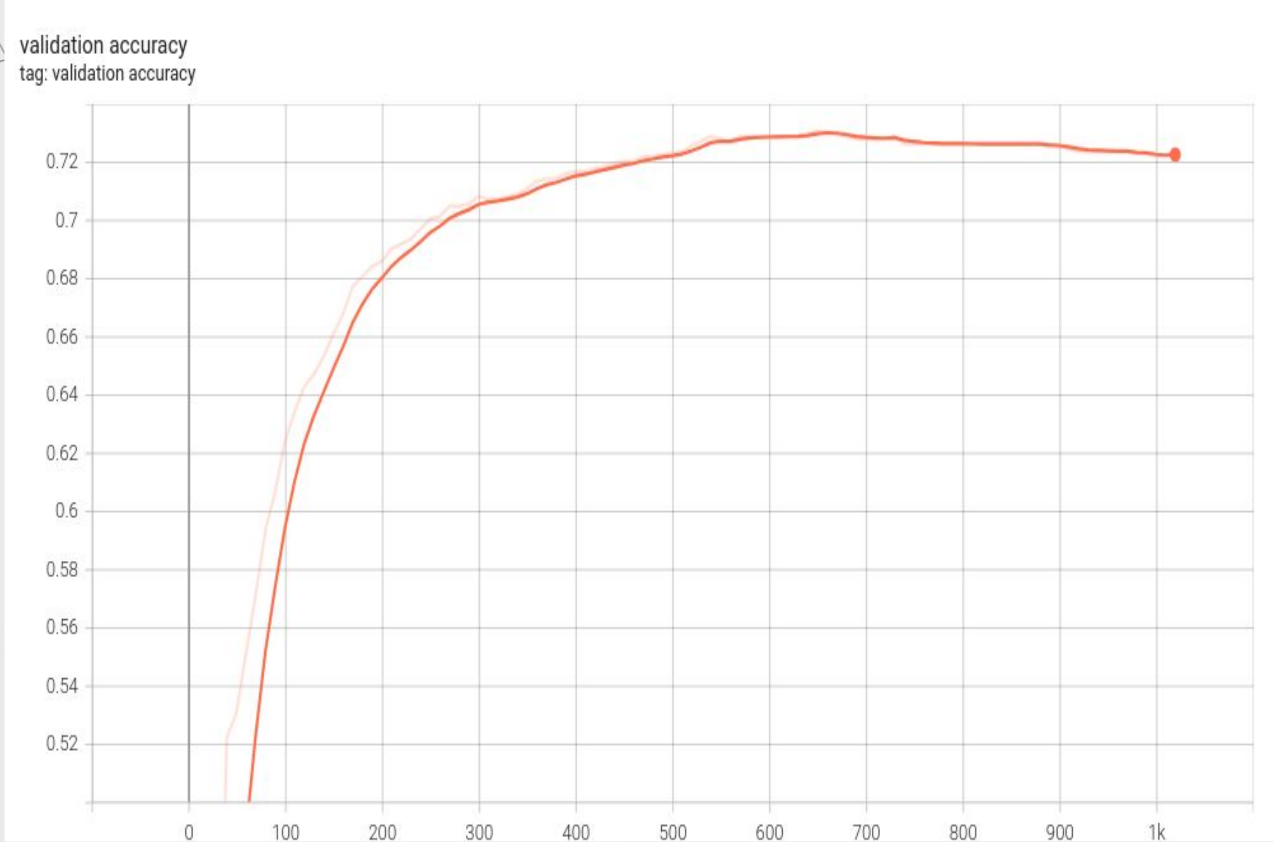
Our Results-BaseLine(Npair)



Our Results i-Mix+Npair



Our Results i-Mix+Npair



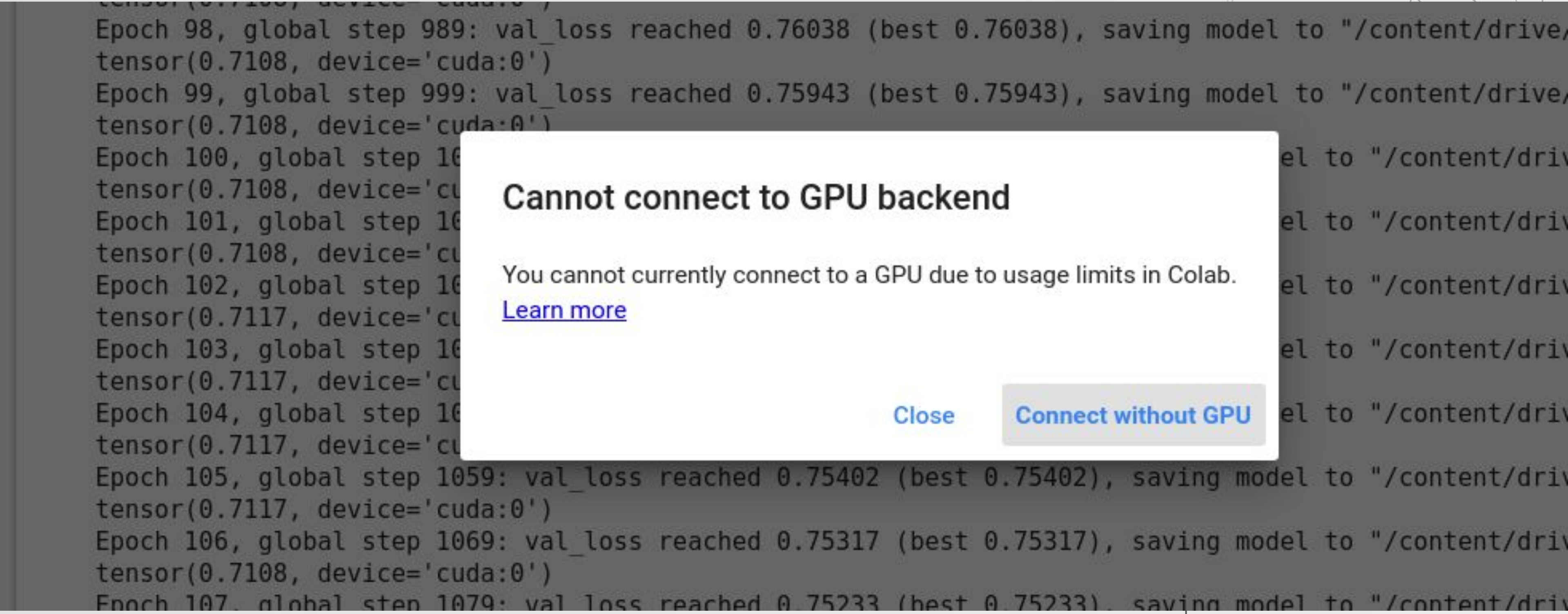
The Same RESULTS….

04

Challenges

Compute Resource !

Long Time spent in an Experiment



05

Future Work

More Experiments!

- Experiment BYol+iMix and Mocov2
- Experiment with images & speech commands with/without augmentation
- Experiment different data set sizes
- Compare with other regularization methods , weight decay , … etc.

Questions?