Representation Learning Algorithms

Presenters :Mansi Tomer, Nirbhay Sharma

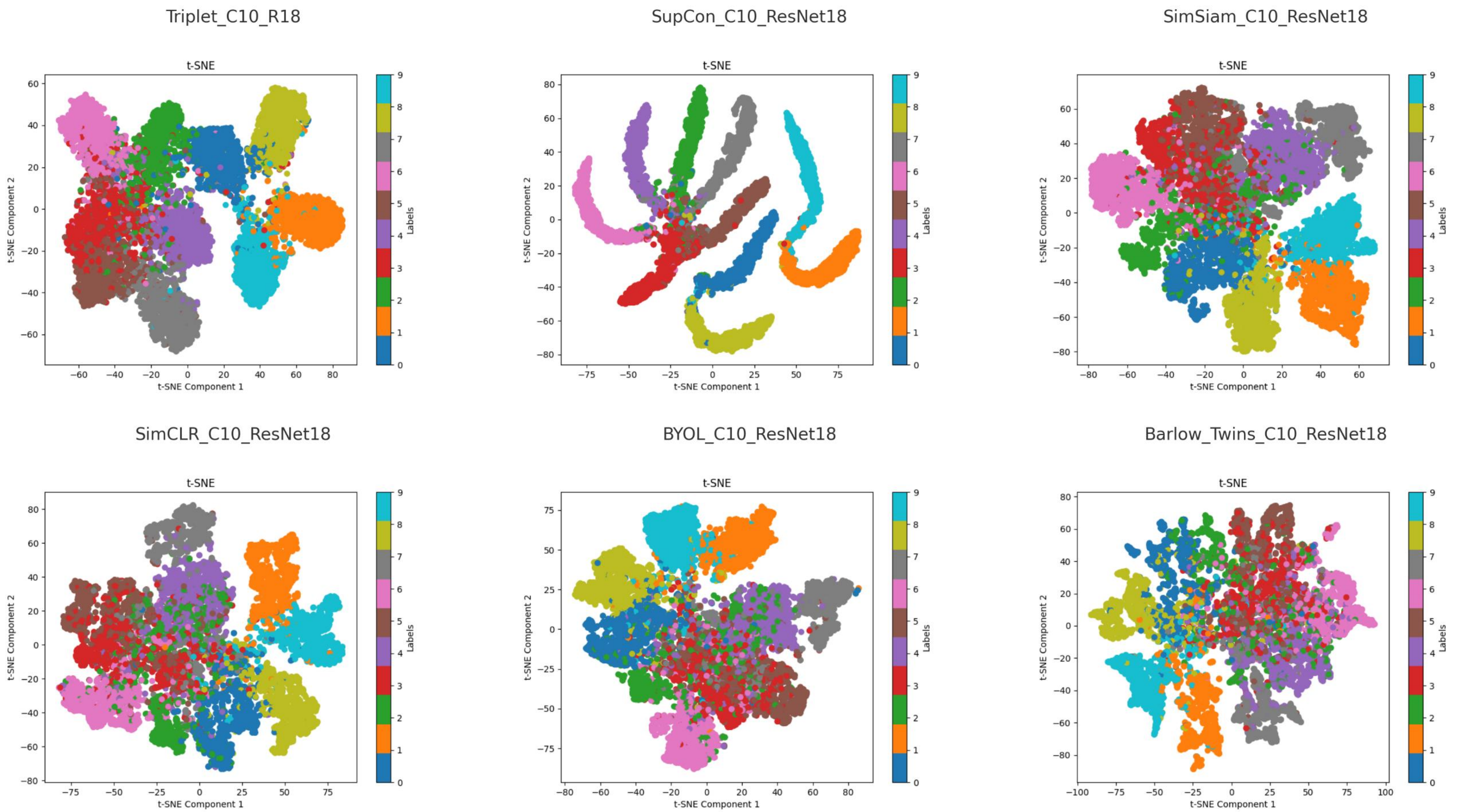
Emails: mansitomer@iisc.ac.in, nirbhays@iisc.ac.in

Problem definition & prior work

- Learning rich and generalizable feature representations is crucial for many vision tasks, especially in scarce label scenario
- We try to investigate and compare diverse self-supervised representation learning algorithms to identify which approach yield most effective features
- Prior Work Includes the following
  - SimCLR (ICML 2020) – Contrastive Loss Based
  - Barlow Twins (ICML 2021) - Loss Based
  - BYOL (NIPS 2020) – Network Based (Momentum Encoder)
  - SimSiam (CVPR 2021) - Network Based (Predictor)
  - Triplet Margin Loss (CVPR 2021) – Loss Based
  - SupCon (NIPS 2020) – Loss Based

Work done

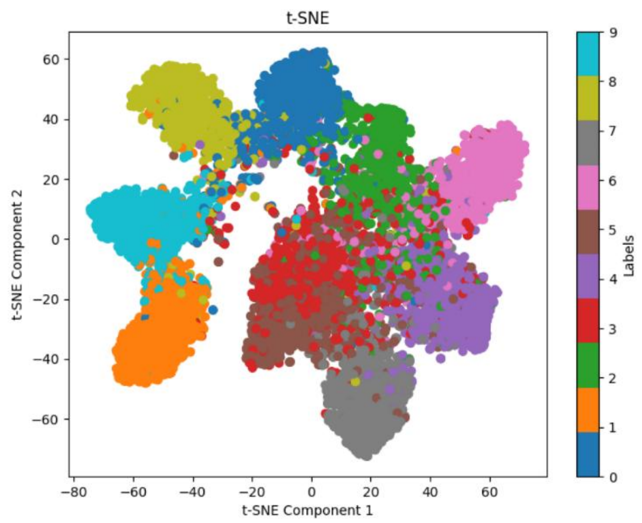
- We implement SimCLR, SupCon, BYOL, SimSiam, Triplet, Barlow_Twins **from Scratch in Pytorch**, TSNE plot is shown below (For Resnet18)



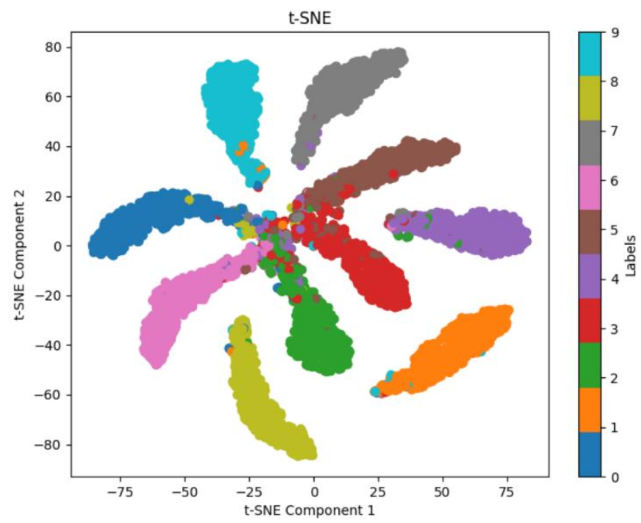
Work done

- We implement SimCLR, SupCon, BYOL, SimSiam, Triplet, Barlow_Twins **from Scratch in Pytorch**, TSNE plot is shown below (For Resnet50)

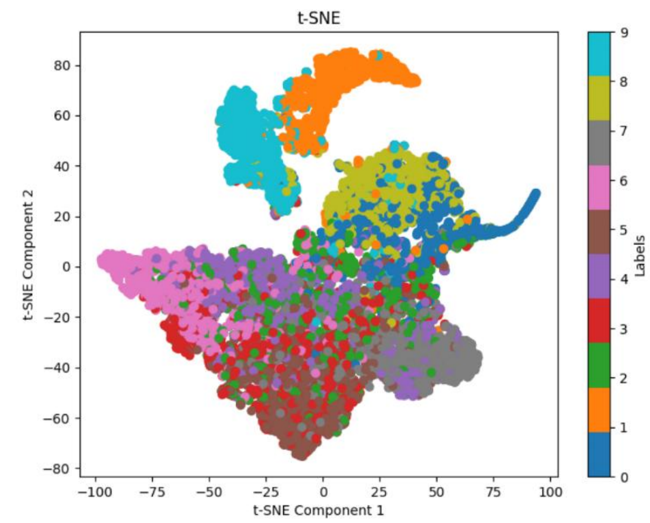
Triplet_C10_R50



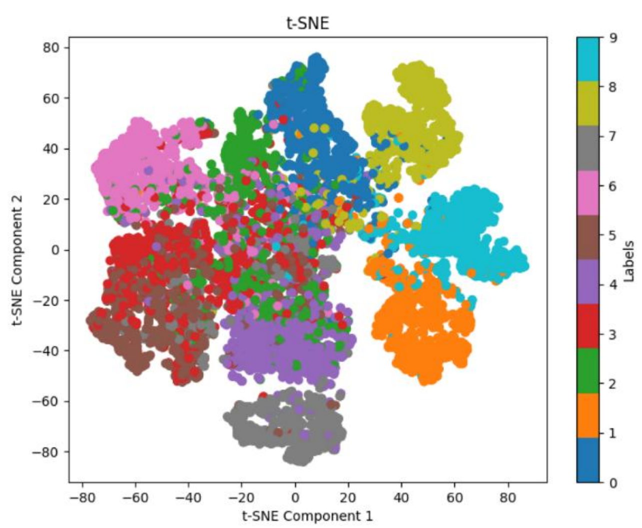
SupCon_C10_R50



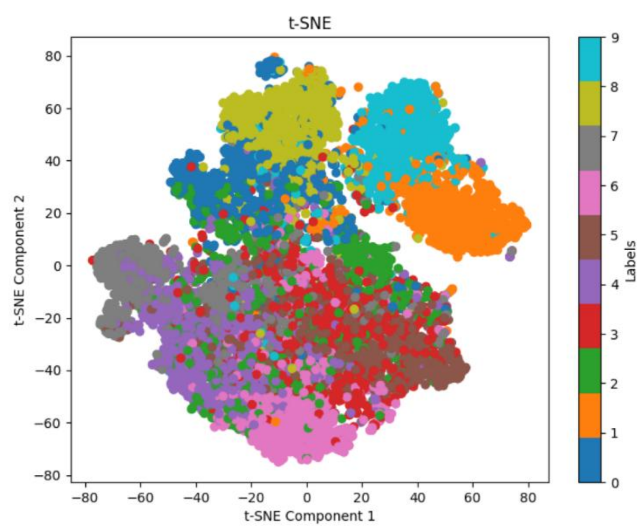
SimSiam_C10_R50



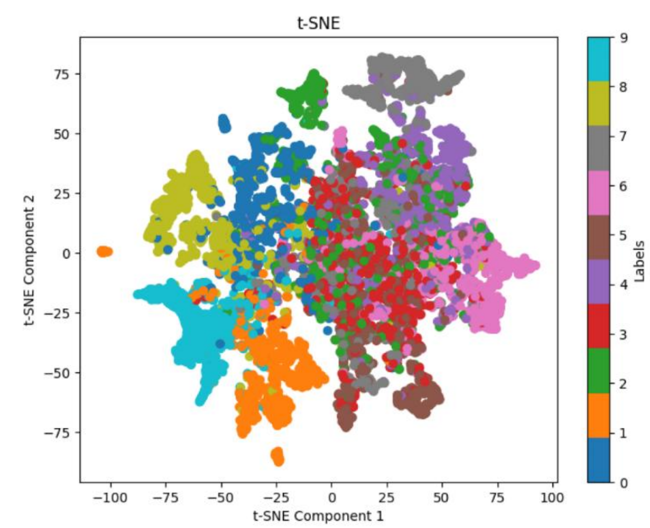
SimCLR_C10_R50



BYOL_C10_R50



Barlow_Twins_C10_R50



Work done

➢ Sample Code snippet of our **Scratch Implementation**

```
def train_network(**kwargs):
    train_algo = kwargs['train_algo']
    kwargs.pop("train_algo")
    if train_algo == "supcon" or train_algo == "simclr
        kwargs["train_algo"] = train_algo
        train_supcon(**kwargs)
    elif train_algo == "triplet":
        train_triplet(**kwargs)
    elif train_algo == "simsiam":
        train_simsiam(**kwargs)
    elif train_algo == 'byol':
        train_byol(**kwargs)
    elif train_algo == "barlow_twins":
        train_barlow_twins(**kwargs)
    elif train_algo == "dare":
        train_DARe(**kwargs)
```

```
def loss_function(loss_type = 'supcon', **kwargs):
    print(f"loss function: {loss_type}")
    loss_mlp = nn.CrossEntropyLoss()
    if loss_type == "simclr":
        return SimCLR(**kwargs), loss_mlp
    elif loss_type == 'supcon':
        return SupConLoss(**kwargs), loss_mlp
    elif loss_type == "triplet":
        return TripletMarginLoss(**kwargs), loss_mlp
    elif loss_type == "simsiam":
        return SimSiamLoss(), loss_mlp
    elif loss_type == 'byol':
        return BYOLLoss(), loss_mlp
    elif loss_type == "barlow_twins":
        return BarlowTwinLoss(**kwargs), loss_mlp
    elif loss_type == "dare":
        return DAReLoss(**kwargs), loss_mlp
    elif loss_type == "dial":
        return DiALLoss(**kwargs), loss_mlp
    else:
        print("{loss_type} Loss is Not Supported")
        return None
```

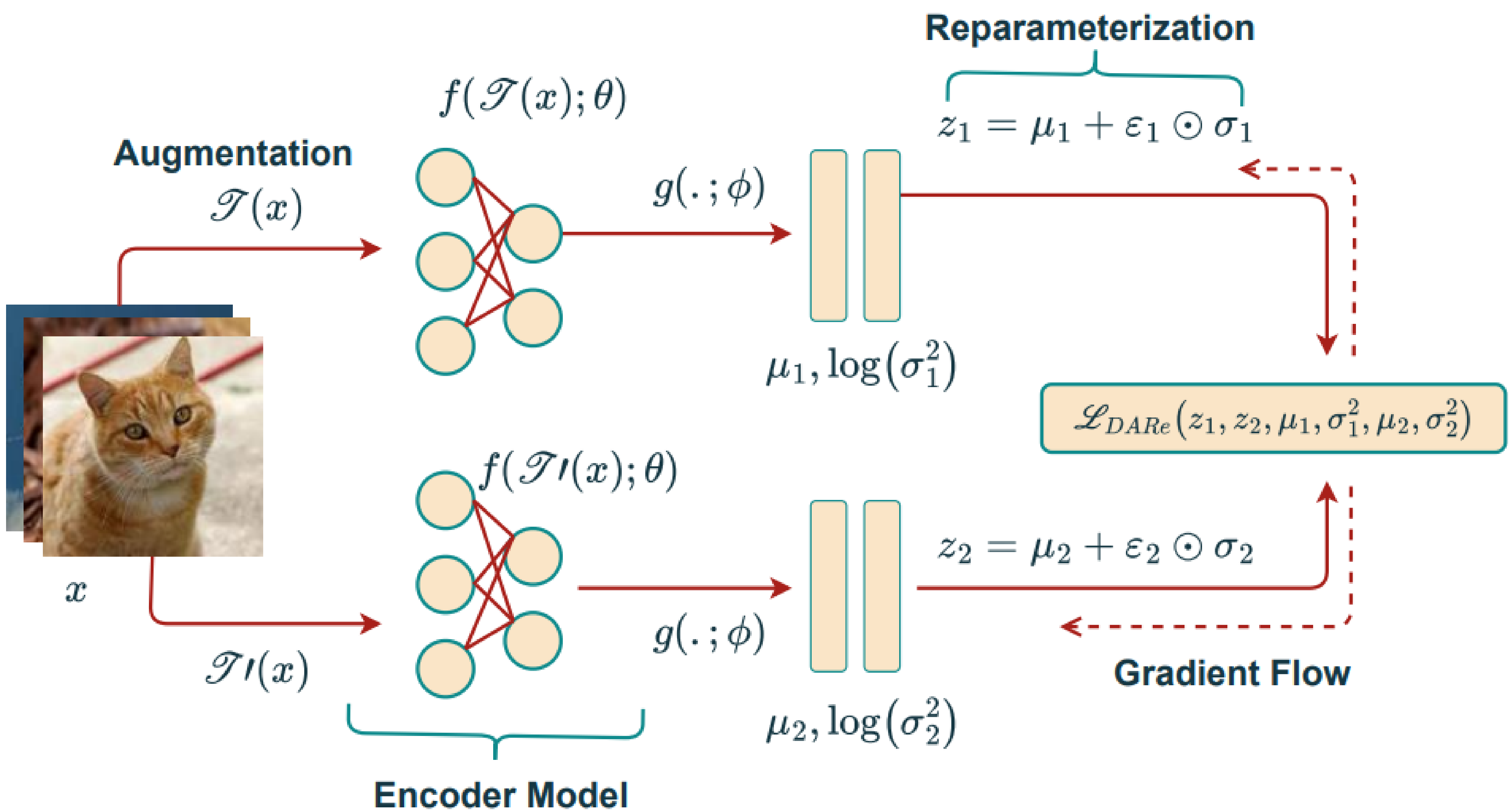
Key results and summary

- Results of all methods Top-1 accuracy on CIFAR10/100 datasets
- **Our Implementation from scratch is available at:** https://github.com/nirbhay-design/RepresentationLearningAlgorithms

Algorithm	CIFAR10 (R50)	CIFAR100 (R50)	CIFAR10 (R18)	CIFAR100 (R18)
SimCLR	87.5 (91.8)	57.7 (68.3)	85.9 (91.8)	55.0 (66.83)
SupCon	**94.0** (96.0)	74.7 (76.5)	**93.5**	**70.4**
Triplet	83.4	**76.3**	86.0	64.5
Barlow Twins	81.2 (90.8)	47.7	80.3 (84.7)	45.8
BYOL	83.0 (91.3)	47.0 (78.4)	84.8 (83.2)	54.8
SimSiam	76.5	34.5	88.6 (91.9)	62.3
DARe (Ours)	**89.4**	**62.3**	**87.3**	**61.6**

Contributions and novelty

- We design a novel loss function inspired from VAE
- The loss composed of Contrastive Loss + Jenson Shannon divergence term



Contributions and novelty

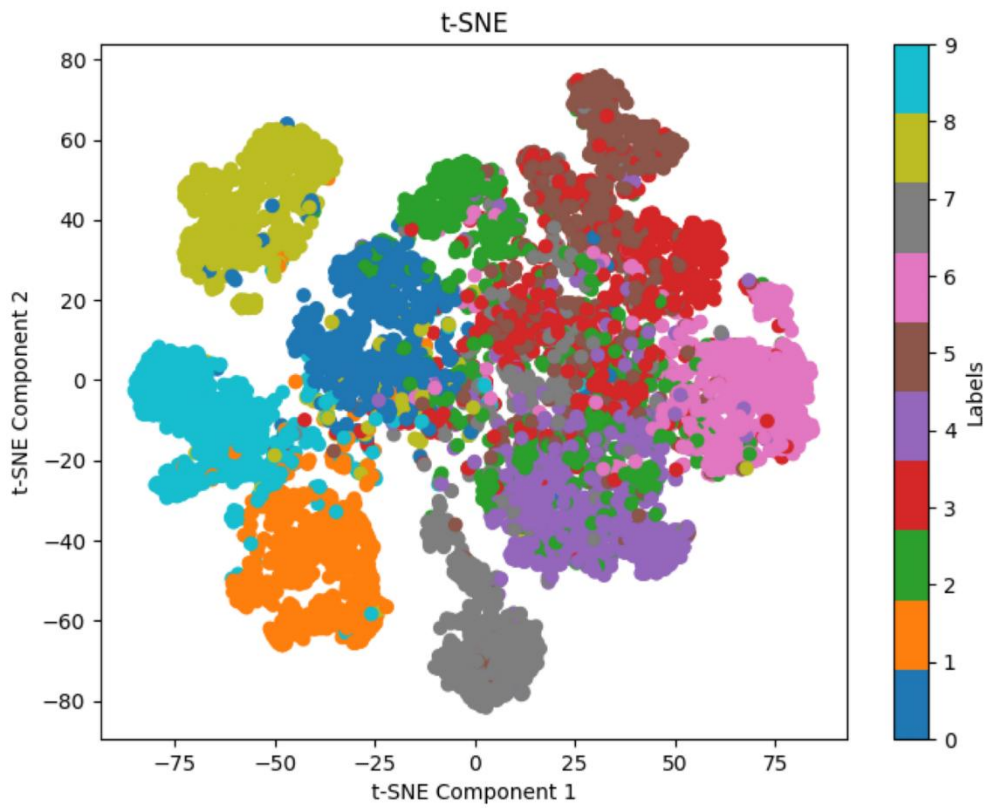
- Equations are described as follows

$$\mathcal{L}_{DARe} = \mathcal{L}_{con}(z_1, z_2) + \lambda \mathcal{L}_{JSD}(\mathcal{N}(\mu_1, \sigma_1^2 I), \mathcal{N}(\mu_2, \sigma_2^2 I))$$

$$\mathcal{L}_{con}(z_1, z_2) = -\sum_{i=1}^{n} \log\left(\frac{e^{sim(z_{1i}, z_{2i})/\tau}}{\sum_{j \neq i} e^{sim(z_{1i}, z_{2j})/\tau}}\right)$$

$$\mathcal{L}_{KL}(\mathcal{N}(\mu_1, \sigma_1^2 I), \mathcal{N}(\mu_2, \sigma_2^2 I)) = \frac{1}{2}\left(\sum_i \left(\frac{\sigma_{1i}^2}{\sigma_{2i}^2} + \log\left(\frac{\sigma_{2i}^2}{\sigma_{1i}^2}\right) + \frac{(\mu_{2i} - \mu_{1i})^2}{\sigma_{2i}^2}\right)\right)$$

DARe_C10_ResNet50



DARe_C10_ResNet18