The plots and images illustrated in the report are saved in the Output folder.

Part I
SSL

Problem 1

Answer:

Implementation Details

Architecture

- Building and training a ResNet-50 from scratch

Labels

- For binary class, I changed the labels so that it classifies between non living and living thing
- For 5-class, I changed the classes so that bird and place(things that fly) are in one class, similarly for horse and car, etc.

Hyperparameters

Parameter	Value
Batch size	10000
Image size	32*32
Number of epochs	30
Optimizer	Adam(lr=0.002)

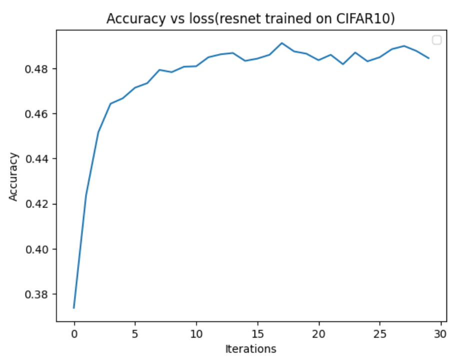
Table 1: Hyperparameters

Results

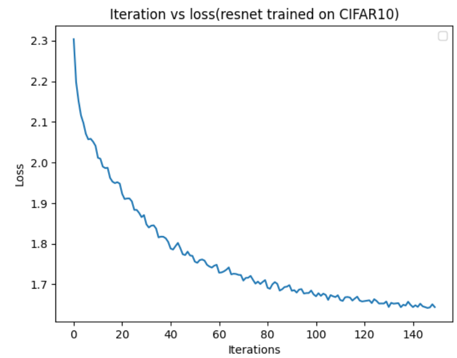
- The accuracies are in 2
- The training plots for just the 10 class classification head are presented in 1.
- The training plots for the 10 and 5 class classification head are presented in 2.
- The training plots for the 10 and 2 class classification head are presented in 3.

Heads	Accuracy
10	0.484
10,5	0.081
10,2	0.08

Table 2: CIFAR10 results with different heads

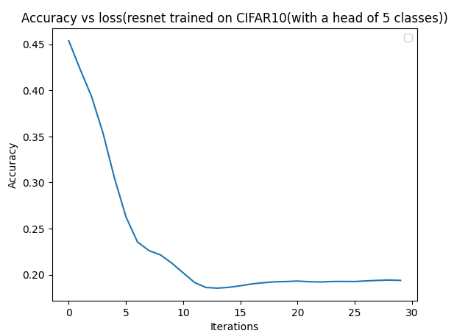


(a) Accuracy

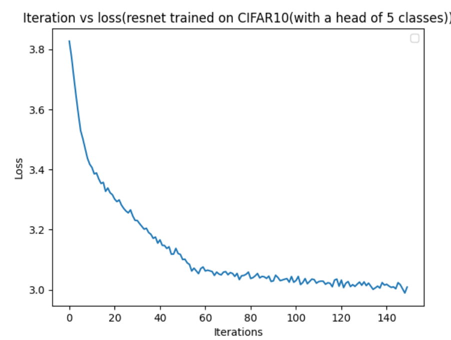


(b) Loss

Figure 1: Test Accuracy and Training Loss curves for 10 class

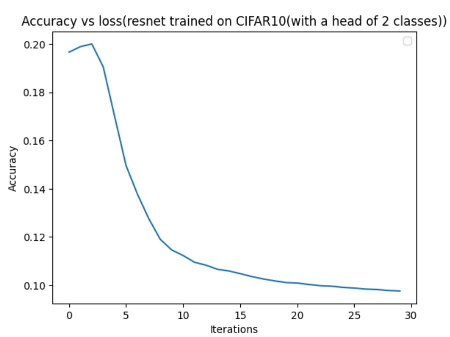


(a) Accuracy

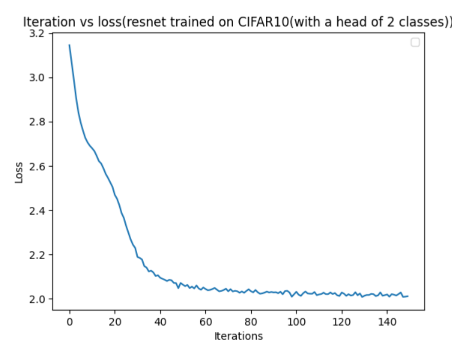


(b) Loss

Figure 2: Test Accuracy and Training Loss curves for training with 10 and 5 classes



(a) Accuracy



(b) Loss

Figure 3: Test Accuracy and Training Loss curves for training with 10 and 5 classes

T-SNE

- Figures 4, 6, 5 are the 3 TSNE plots.

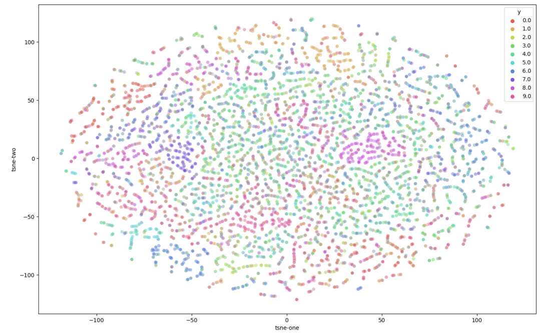


Figure 4: CIFAR10 TSNE

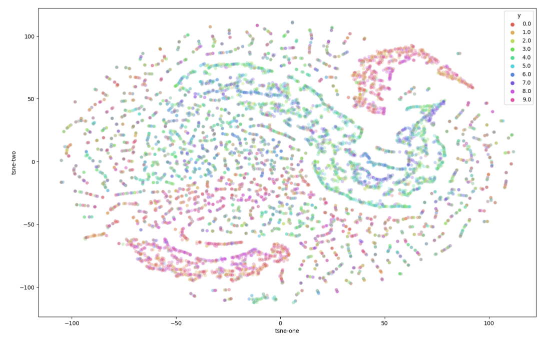


Figure 5: CIFAR2 TSNE

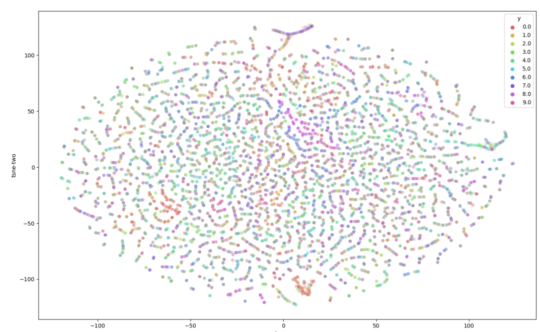


Figure 6: CIFAR5 TSNE

Problem 2
Implement MOCO for CIFAR 10.

Answer:

Implementation Details

Architecture

- Building and training a ResNet-50 from scratch

Augmentation

- All the augmentations mentioned in the question are done randomly. This is just during training.

- During testing, no augmentations are done.

Hyperparameters

Parameter	Value
Batch size	512
Image size	32*32
Number of epochs	200
Optimizer	Variable(initial 0.6)
k in KNN	200
Dict sizes	2048, 4096

Table 3: Hyperparameters for MOCO

Results

- The accuracies in the plots are calculated using a KNN classifier with k = 200
- The training plots for just the 2000 dict size are presented in 7.

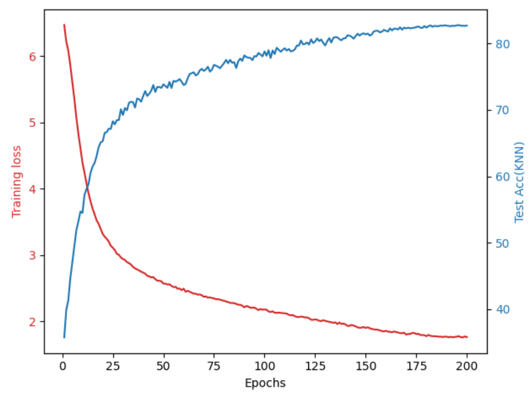


Figure 7: MOCO 2000

- The training plots for just the 4000 dict size are presented in 8.
- After learning features, the data was classified using 5 different subsets of data. The results are in 4.

% of traning data	Dict size 1	Dict Size 2
10	0.789	0.785
20	0.791	0.79
30	0.795	0.792
40	0.799	0.796
50	0.797	0.797

Table 4: MOCO Performance

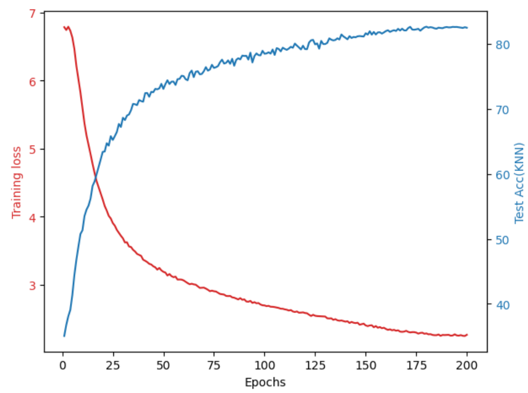


Figure 8: MOCO 4000

Part II
Few Shot Learning (FSL)

> **Problem 1**
> Implement prototypical networks for the above problem of FSL

Answer:

Implementation Details

Architecture

- The basic convolutional block in the architecture is -
  nn.Sequential(nn.Conv2d(in_channels, out_channels, kernel_size=3, padding=1), nn.BatchNorm2d(out_channels), nn.ReLU(), nn.MaxPool2d(2))
- 5 such blocks are added sequentially in which the last one does not have MaxPool(2) operation

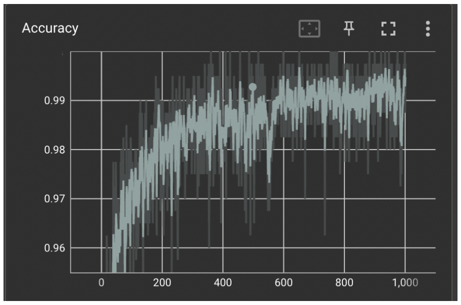
Hyper Parameters

Parameter	Value
Batch size	16
hidden-size	32
Number of epochs	1000
Optimizer	Adam(lr=1e-3)

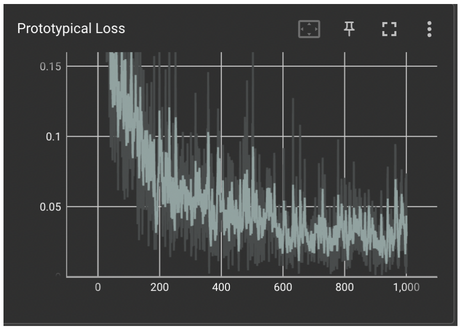
Table 5: Hyperparameters for Prototypical Networks

Plots

- Figure 9 shows test accuracy and training loss as a function of number of epochs for 5 way 5 shot classification
- Figure 10 shows test accuracy and training loss as a function of number of epochs for 5 way 1 shot classification

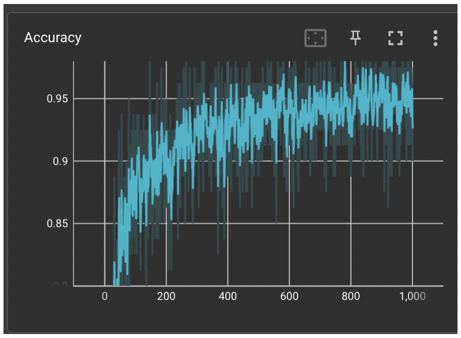


(a) Accuracy

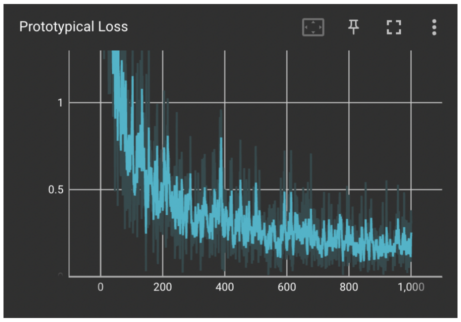


(b) Loss

Figure 9: Test Accuracy and Training Loss curves for N=5 K=5



(a) Accuracy



(b) Loss

Figure 10: Test Accuracy and Training Loss curves for N=5 K=1

Results

Few Shot Setting	Accuracy	Training Time	Computational Cost Per Epoch
N:5 K:5	98.8%	597.5 seconds	0.59 seconds
N:5 k:1	98.2%	186.8 seconds	0.18 seconds

Table 6: Results for Prototypical Networks

Conclusions

- The computational cost for Prototypical networks in 5 way, 5 shot setting is significantly higher than that for 5 way 1 shot setting.
- The accuracy numbers are also higher for 5 way 5 shot setting.

Problem 2
MAML
Implement MAML on the same 5-layer CNN as above with a classif ication head at the end

Answer:

1 Implementation Details

Architecture

- I used pytorch meta to implement dataloaders, model and training loop in this problem.
- The basic convolutional block in the architecture is -
MetaSequential(MetaConv2d(in_channels, out_channels, kernel_size=3, padding=1), MetaBatchNorm2d(out_channels), nn.ReLU(), nn.MaxPool2d(2))
- 5 such blocks are added sequentially in which the last one does not have MaxPool(2) operation
- This architecture is the same as the one used in Prototypical Networks

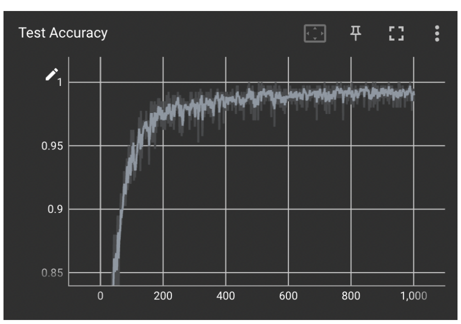
Hyper Parameters

Parameter	Value
Batch size	16
hidden-size	32
Number of epochs	1000
Optimizer	Adam(lr=1e-3)

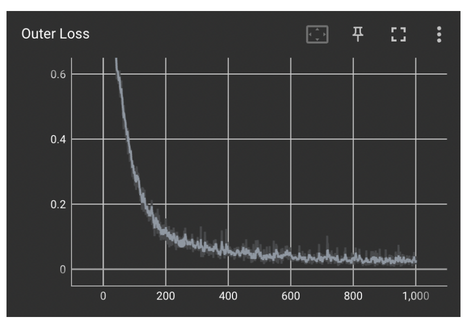
Table 7: Hyperparameters for MAML

Plots

- Figure 11 shows test accuracy and training loss as a function of number of epochs for 5 way 5 shot classification



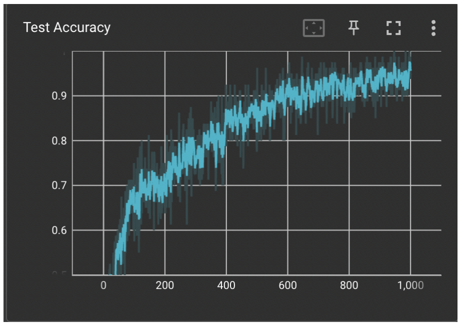
(a) Accuracy



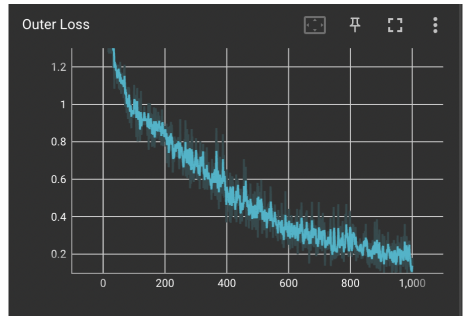
(b) Loss

Figure 11: Test Accuracy and Training Loss curves for N=5 K=5

- Figure 12 shows test accuracy and training loss as a function of number of epochs for 5 way 1 shot classification



(a) Accuracy



(b) Loss

Figure 12: Test Accuracy and Training Loss curves for N=5 K=1

Few Shot Setting	Accuracy	Training Time	Computational Cost Per Epoch
N:5 K:5	99.6	309.5 seconds	0.3 seconds
N:5 K:1	98.1	260.26 seconds	0.26 seconds

Table 8: Results for MAML

Results and Conclusions

- The computational cost of running MAML is significantly lower than Prototypical networks for 5 way, 5 shot setting, while being higher for 5 way 1 shot setting.
- The accuracy of MAML is higher than Prototypical networks for 5 way, 5 shot setting, while being lower for 5 way 1 shot setting.