

AIMS-DTU
Research Intern
Round 2 Project Task:
Self-supervised learning

Name: Kumudini Gholap

College: NIT SURAT

Email: kumudini1308@gmail.com

Objectives: The aim of this task is to implement and analyse self-supervised learning methods for visual representation learning

with both types of techniques:

- **Contrastive learning**
- **Masked Image Modelling (MIM)**

Introduction to the task done and approaches used.

This project focuses on implementing and analyzing self-supervised learning methods to learn visual representations from images without relying on manual labels. The main objective is to understand and compare two prominent types of self-supervised learning:

- **Contrastive Learning:** Techniques that learn representations by encouraging agreement between different augmented views of the same image while pushing apart representations of different images.
- **Masked Image Modelling (MIM):** Methods that learn representations by masking portions of input images and training the model to reconstruct the missing parts, thereby forcing understanding of image context and semantics.

We utilize the ImageNet-100 dataset, a subset of ImageNet with 100 classes, to perform the pretraining and evaluation. The training phase uses only unlabeled images, and evaluation is conducted by freezing the learned encoder and training a linear classifier on labeled data (linear probing).

The project is structured in four main steps:

1. Implement at least one contrastive learning method (SimCLR) and one MIM method (such as Masked Autoencoders - MAE).
2. Train the models on the ImageNet-100 dataset using unlabeled images, applying suitable augmentations and masking strategies.

3. Evaluate the models by training a linear classifier on the frozen embeddings using labeled data, and report accuracy and F1 scores.
4. Compare the performance of contrastive and MIM approaches, analyzing loss and accuracy curves to interpret the strengths and weaknesses of each.

2. Approaches used in each self-supervised learning method.

In this project, we implemented and trained two distinct self-supervised learning approaches:

- SimCLR — a contrastive learning method
- MAE (Masked Autoencoders) — a masked image modelling method

Contrastive Learning: SimCLR

- **Objective:**

SimCLR learns useful image representations by contrasting positive pairs (two augmented versions of the same image) against negative pairs (different images). The model tries to bring positive pairs closer in feature space while pushing negative pairs apart.

- **How it Works:**

Each input image is augmented twice with random transformations such as cropping, flipping, color jitter, and grayscale conversion. These two views are passed through an encoder network followed by a small projection head to obtain feature vectors. A contrastive loss (NT-Xent loss) encourages features from the same image to be similar while separating different images.

- **Key Elements:**

- Strong data augmentation to generate diverse views

- A projection head for learning a better latent space for contrastive loss
- Use of a temperature-scaled cross-entropy loss to compare all pairs in the batch

Masked Image Modelling: MAE (Masked Autoencoders)

- **Objective:**

MAE trains the model to reconstruct missing parts of an image from visible parts. By masking random patches (e.g., 75% of the image patches), the model learns to capture the contextual and structural information of images.

- **How it Works:**

A random masking strategy hides most of the image patches before inputting them to the model's encoder. The encoder processes only the visible patches, and a lightweight decoder attempts to reconstruct the original image patches at the masked locations. The reconstruction loss (typically mean squared error) guides the model to learn meaningful representations.

- **Key Elements:**

- High masking ratio to challenge the model
- Reconstruction of masked patches forces the model to understand global context
- A two-stage network (encoder + decoder) where the encoder is later used for downstream tasks

+ 3. Model Architectures, preprocessing steps, hyperparameters, etc.

Contrastive Learning: SimCLR

Model Architecture

- Encoder Backbone: ResNet-50 (without pretrained weights)
- Projection Head:
 - Fully connected layer: 2048 (ResNet output) \rightarrow 512
 - ReLU activation
 - Fully connected layer: 512 \rightarrow 128 (projection dimension)
- The encoder extracts image features; the projection head maps features to a space where contrastive loss is applied.

Preprocessing Steps

- Input images resized to 224×224 pixels
- Data augmentations (applied randomly to each image to create positive pairs):
 - Random Resized Crop
 - Random Horizontal Flip
 - Color Jitter (brightness, contrast, saturation, hue)
 - Random Grayscale conversion
- Normalization with mean = (0.5, 0.5, 0.5) and std = (0.5, 0.5, 0.5)

Hyperparameters

- Batch size: 32
- Optimizer: Adam with learning rate = $3e-4$
- Loss: NT-Xent (Normalized Temperature-scaled Cross Entropy)
- Temperature parameter (τ): 0.5

- Number of training epochs: 100
- Weight decay: Typically $1e-4$ (can be tuned)
- Number of negative samples: Implicitly defined by batch size

Masked Image Modelling: MAE (Masked Autoencoder)

Model Architecture

- Encoder: Vision Transformer (ViT-Base) without pretrained weights
 - Input: Image patches of size 16×16 pixels
 - Output embedding dimension: 768
- Decoder:
 - Fully connected layer: $768 \rightarrow 1024$
 - ReLU activation
 - Fully connected layer: $1024 \rightarrow 768$
- Reconstruction Head:
 - Linear layer mapping 768-dim embedding to the original pixel dimension of the patch ($3 \text{ channels} \times 16 \times 16 \text{ pixels} = 768$)
- The model learns to reconstruct masked patches from visible patches.

Preprocessing Steps

- Input images resized to 224×224 pixels

- Normalization with mean = 0.5 and std = 0.5 (single channel or per channel depending on dataset)
- Random masking of patches during training (typically mask ratio = 75%)

Hyperparameters

- Masking ratio: 75% of patches randomly masked per image
- Batch size: 32
- Optimizer: Adam with learning rate = $1e-4$
- Loss function: Mean Squared Error (MSE) between reconstructed and original patches
- Number of training epochs: 100
- Patch size: 16×16 pixels

Evaluation Results, including accuracy, F1 score and loss curves.

Contrastive Learning (SimCLR)

```
PS C:\Users\HP\Documents\Projects\Self-supervised learning>
python -m train.SimCLR_linear_probe
```

```
[DEBUG] Loaded 130000 images from ['data/train.X1',
'data/train.X2', 'data/train.X3', 'data/train.X4']
```

```
[DEBUG] Loaded 32500 labeled images from data/train.X1
```

```
[DEBUG] Loaded 5000 labeled images from data/val.X
```

[INFO] Starting training on device: cpu

[INFO] Epoch 1 started

[DEBUG] First training batch: img shape torch.Size([16, 3, 224, 224]), label type: <class 'torch.Tensor'>

[INFO] Epoch 1 training complete, starting evaluation...

[RESULT] Epoch 1: Accuracy = 0.3560, F1 Score = 0.3405

[INFO] Epoch 2 started

[INFO] Epoch 2 training complete, starting evaluation...

[RESULT] Epoch 2: Accuracy = 0.4220, F1 Score = 0.4102

```
/content/ssl_dataset/ssl_dataset/train.X1 exists? True  
[DEBUG] Loaded 32500 images from ['/content/ssl_dataset/ssl_dataset/train.X1']  
Using device: cuda
```

```
Running quick single-batch test...  
Model outputs shapes: torch.Size([32, 128]), torch.Size([32, 128])  
NT-Xent loss on single batch: 4.139734268188477
```


Starting epoch 1

5%	50/1016	[00:37<11:56, 1.35it/s]	Epoch 1, Batch 50 loss: 2.5194
10%	100/1016	[01:13<10:28, 1.46it/s]	Epoch 1, Batch 100 loss: 2.3604
15%	150/1016	[01:48<10:30, 1.37it/s]	Epoch 1, Batch 150 loss: 2.3473
20%	200/1016	[02:24<09:39, 1.41it/s]	Epoch 1, Batch 200 loss: 2.3362
25%	250/1016	[02:59<08:55, 1.43it/s]	Epoch 1, Batch 250 loss: 2.3314
30%	300/1016	[03:34<08:27, 1.41it/s]	Epoch 1, Batch 300 loss: 2.3142
34%	350/1016	[04:10<07:53, 1.41it/s]	Epoch 1, Batch 350 loss: 2.2912
39%	400/1016	[04:45<07:25, 1.38it/s]	Epoch 1, Batch 400 loss: 2.2725
44%	450/1016	[05:21<06:40, 1.41it/s]	Epoch 1, Batch 450 loss: 2.3189
49%	500/1016	[05:56<06:03, 1.42it/s]	Epoch 1, Batch 500 loss: 2.3081
54%	550/1016	[06:32<05:31, 1.40it/s]	Epoch 1, Batch 550 loss: 2.3049
59%	600/1016	[07:07<04:58, 1.39it/s]	Epoch 1, Batch 600 loss: 2.2874
64%	650/1016	[07:42<04:17, 1.42it/s]	Epoch 1, Batch 650 loss: 2.3219
69%	700/1016	[08:18<03:42, 1.42it/s]	Epoch 1, Batch 700 loss: 2.2969
74%	750/1016	[08:53<03:08, 1.41it/s]	Epoch 1, Batch 750 loss: 2.2904
79%	800/1016	[09:29<02:36, 1.38it/s]	Epoch 1, Batch 800 loss: 2.2901
84%	850/1016	[10:04<01:57, 1.41it/s]	Epoch 1, Batch 850 loss: 2.2890
89%	900/1016	[10:40<01:21, 1.43it/s]	Epoch 1, Batch 900 loss: 2.2770
94%	950/1016	[11:15<00:46, 1.43it/s]	Epoch 1, Batch 950 loss: 2.2973
98%	1000/1016	[11:50<00:11, 1.41it/s]	Epoch 1, Batch 1000 loss: 2.3111
100%	1016/1016	[12:01<00:00, 1.41it/s]	

Epoch 1: Average Loss = 2.3362

Starting epoch 2

5%	50/1016	[00:36<11:15, 1.43it/s]	Epoch 2, Batch 50 loss: 2.2804
10%	100/1016	[01:11<10:43, 1.42it/s]	Epoch 2, Batch 100 loss: 2.3017
15%	150/1016	[01:46<10:22, 1.39it/s]	Epoch 2, Batch 150 loss: 2.2649
20%	200/1016	[02:22<09:32, 1.43it/s]	Epoch 2, Batch 200 loss: 2.2958
25%	250/1016	[02:57<08:56, 1.43it/s]	Epoch 2, Batch 250 loss: 2.2773
30%	300/1016	[03:32<08:24, 1.42it/s]	Epoch 2, Batch 300 loss: 2.2353
34%	350/1016	[04:08<07:48, 1.42it/s]	Epoch 2, Batch 350 loss: 2.2921
39%	400/1016	[04:43<07:12, 1.43it/s]	Epoch 2, Batch 400 loss: 2.2823
44%	450/1016	[05:18<06:38, 1.42it/s]	Epoch 2, Batch 450 loss: 2.2690
49%	500/1016	[05:53<06:06, 1.41it/s]	Epoch 2, Batch 500 loss: 2.3058
54%	550/1016	[06:29<05:27, 1.42it/s]	Epoch 2, Batch 550 loss: 2.2566
59%	600/1016	[07:04<04:51, 1.43it/s]	Epoch 2, Batch 600 loss: 2.2767
64%	650/1016	[07:39<04:17, 1.42it/s]	Epoch 2, Batch 650 loss: 2.2877
69%	700/1016	[08:14<03:44, 1.41it/s]	Epoch 2, Batch 700 loss: 2.3067
74%	750/1016	[08:50<03:05, 1.43it/s]	Epoch 2, Batch 750 loss: 2.3016
79%	800/1016	[09:25<02:32, 1.42it/s]	Epoch 2, Batch 800 loss: 2.2670
84%	850/1016	[10:00<01:57, 1.41it/s]	Epoch 2, Batch 850 loss: 2.2561
89%	900/1016	[10:35<01:21, 1.43it/s]	Epoch 2, Batch 900 loss: 2.2793
94%	950/1016	[11:11<00:46, 1.43it/s]	Epoch 2, Batch 950 loss: 2.2743
98%	1000/1016	[11:46<00:11, 1.41it/s]	Epoch 2, Batch 1000 loss: 2.2611
100%	1016/1016	[11:57<00:00, 1.42it/s]	

Epoch 2: Average Loss = 2.2828

Starting epoch 3

5% | 50/1016 [00:35<11:17, 1.43it/s]Epoch 3, Batch 50 loss: 2.2644
10% | 100/1016 [01:11<10:45, 1.42it/s]Epoch 3, Batch 100 loss: 2.2759
15% | 150/1016 [01:46<10:13, 1.41it/s]Epoch 3, Batch 150 loss: 2.2622
20% | 200/1016 [02:21<09:36, 1.42it/s]Epoch 3, Batch 200 loss: 2.2433
25% | 250/1016 [02:57<09:04, 1.41it/s]Epoch 3, Batch 250 loss: 2.2858
30% | 300/1016 [03:32<08:24, 1.42it/s]Epoch 3, Batch 300 loss: 2.2639
34% | 350/1016 [04:07<07:47, 1.43it/s]Epoch 3, Batch 350 loss: 2.2525
39% | 400/1016 [04:42<07:13, 1.42it/s]Epoch 3, Batch 400 loss: 2.2573
44% | 450/1016 [05:17<06:36, 1.43it/s]Epoch 3, Batch 450 loss: 2.2888
49% | 500/1016 [05:52<06:03, 1.42it/s]Epoch 3, Batch 500 loss: 2.2627
54% | 550/1016 [06:28<05:31, 1.40it/s]Epoch 3, Batch 550 loss: 2.3043
59% | 600/1016 [07:03<04:50, 1.43it/s]Epoch 3, Batch 600 loss: 2.2702
64% | 650/1016 [07:38<04:16, 1.43it/s]Epoch 3, Batch 650 loss: 2.2452
69% | 700/1016 [08:13<03:43, 1.41it/s]Epoch 3, Batch 700 loss: 2.2622
74% | 750/1016 [08:48<03:06, 1.43it/s]Epoch 3, Batch 750 loss: 2.2857
79% | 800/1016 [09:24<02:33, 1.41it/s]Epoch 3, Batch 800 loss: 2.2806
84% | 850/1016 [09:59<01:55, 1.44it/s]Epoch 3, Batch 850 loss: 2.3024
89% | 900/1016 [10:34<01:22, 1.40it/s]Epoch 3, Batch 900 loss: 2.2988
94% | 950/1016 [11:09<00:46, 1.41it/s]Epoch 3, Batch 950 loss: 2.2704
98% | 1000/1016 [11:45<00:11, 1.43it/s]Epoch 3, Batch 1000 loss: 2.30
100% | 1016/1016 [11:56<00:00, 1.42it/s]
Epoch 3: Average Loss = 2.2741

Starting epoch 4

5% | 50/1016 [00:36<11:24, 1.41it/s]Epoch 4, Batch 50 loss: 2.2588
10% | 100/1016 [01:11<10:45, 1.42it/s]Epoch 4, Batch 100 loss: 2.2865
15% | 150/1016 [01:46<10:05, 1.43it/s]Epoch 4, Batch 150 loss: 2.2525
20% | 200/1016 [02:21<09:32, 1.43it/s]Epoch 4, Batch 200 loss: 2.2981
25% | 250/1016 [02:57<09:00, 1.42it/s]Epoch 4, Batch 250 loss: 2.2568
30% | 300/1016 [03:32<08:23, 1.42it/s]Epoch 4, Batch 300 loss: 2.2805
34% | 350/1016 [04:07<07:44, 1.43it/s]Epoch 4, Batch 350 loss: 2.2938
39% | 400/1016 [04:42<07:12, 1.42it/s]Epoch 4, Batch 400 loss: 2.2861
44% | 450/1016 [05:17<06:42, 1.41it/s]Epoch 4, Batch 450 loss: 2.2715
49% | 500/1016 [05:52<06:02, 1.42it/s]Epoch 4, Batch 500 loss: 2.2651
54% | 550/1016 [06:28<05:28, 1.42it/s]Epoch 4, Batch 550 loss: 2.2463
59% | 600/1016 [07:03<04:55, 1.41it/s]Epoch 4, Batch 600 loss: 2.2738
64% | 650/1016 [07:38<04:16, 1.43it/s]Epoch 4, Batch 650 loss: 2.2654
69% | 700/1016 [08:13<03:40, 1.43it/s]Epoch 4, Batch 700 loss: 2.2539
74% | 750/1016 [08:48<03:05, 1.43it/s]Epoch 4, Batch 750 loss: 2.2511
79% | 800/1016 [09:24<02:32, 1.41it/s]Epoch 4, Batch 800 loss: 2.2474
84% | 850/1016 [09:59<01:56, 1.42it/s]Epoch 4, Batch 850 loss: 2.2689
89% | 900/1016 [10:34<01:21, 1.42it/s]Epoch 4, Batch 900 loss: 2.2720
94% | 950/1016 [11:09<00:47, 1.40it/s]Epoch 4, Batch 950 loss: 2.2627
98% | 1000/1016 [11:44<00:11, 1.42it/s]Epoch 4, Batch 1000 loss: 2.2511
100% | 1016/1016 [11:55<00:00, 1.42it/s]
Epoch 4: Average Loss = 2.2703

Starting epoch 5

Progress	Batch	Time	Speed	Loss
5%	50/1016	[00:35<11:21,	1.42it/s]	Epoch 5, Batch 50 loss: 2.2829
10%	100/1016	[01:11<10:55,	1.40it/s]	Epoch 5, Batch 100 loss: 2.2674
15%	150/1016	[01:46<10:07,	1.42it/s]	Epoch 5, Batch 150 loss: 2.2528
20%	200/1016	[02:21<09:30,	1.43it/s]	Epoch 5, Batch 200 loss: 2.2672
25%	250/1016	[02:56<09:04,	1.41it/s]	Epoch 5, Batch 250 loss: 2.2774
30%	300/1016	[03:32<08:19,	1.43it/s]	Epoch 5, Batch 300 loss: 2.2728
34%	350/1016	[04:07<07:48,	1.42it/s]	Epoch 5, Batch 350 loss: 2.2909
39%	400/1016	[04:42<07:18,	1.40it/s]	Epoch 5, Batch 400 loss: 2.2683
44%	450/1016	[05:17<06:38,	1.42it/s]	Epoch 5, Batch 450 loss: 2.2497
49%	500/1016	[05:53<06:02,	1.42it/s]	Epoch 5, Batch 500 loss: 2.2914
54%	550/1016	[06:28<05:32,	1.40it/s]	Epoch 5, Batch 550 loss: 2.2566
59%	600/1016	[07:03<04:57,	1.40it/s]	Epoch 5, Batch 600 loss: 2.2541
64%	650/1016	[07:39<04:17,	1.42it/s]	Epoch 5, Batch 650 loss: 2.2741
69%	700/1016	[08:14<03:41,	1.43it/s]	Epoch 5, Batch 700 loss: 2.2626
74%	750/1016	[08:49<03:09,	1.40it/s]	Epoch 5, Batch 750 loss: 2.2451
79%	800/1016	[09:24<02:31,	1.43it/s]	Epoch 5, Batch 800 loss: 2.2483
84%	850/1016	[10:00<01:56,	1.42it/s]	Epoch 5, Batch 850 loss: 2.2563
89%	900/1016	[10:35<01:21,	1.42it/s]	Epoch 5, Batch 900 loss: 2.2621
94%	950/1016	[11:10<00:46,	1.41it/s]	Epoch 5, Batch 950 loss: 2.2777
98%	1000/1016	[11:45<00:11,	1.43it/s]	Epoch 5, Batch 1000 loss: 2.2596
100%	1016/1016	[11:57<00:00,	1.42it/s]	Epoch 5: Average Loss = 2.2640

/content/ssl_dataset/ssl_dataset/train.X2 exists? True

[DEBUG] Loaded 32500 images from ['/content/ssl_dataset/ssl_dataset/train.X2']

Using device: cuda

Running quick single-batch test...

Model outputs shapes: torch.Size([32, 128]), torch.Size([32, 128])

NT-Xent loss on single batch: 4.138222694396973

Starting epoch 1

100%|██████████| 1016/1016 [12:01<00:00, 1.41it/s]

Epoch 1: Average Loss = 2.3324

Starting epoch 2

100%|██████████| 1016/1016 [11:58<00:00, 1.41it/s]

Epoch 2: Average Loss = 2.2807

Starting epoch 3

100%|██████████| 1016/1016 [11:55<00:00, 1.42it/s]

Epoch 3: Average Loss = 2.2720

Starting epoch 4

100%|██████████| 1016/1016 [11:56<00:00, 1.42it/s]

Epoch 4: Average Loss = 2.2709

Starting epoch 5

100%|██████████| 1016/1016 [11:55<00:00, 1.42it/s] Epoch 5: Average Loss = 2.2611

```
/content/ssl_dataset/ssl_dataset/train.X3 exists? True
[DEBUG] Loaded 32500 images from ['/content/ssl_dataset/ssl_dataset/train.X3']
Using device: cuda

Running quick single-batch test...
Model outputs shapes: torch.Size([32, 128]), torch.Size([32, 128])
NT-Xent loss on single batch: 4.141243934631348

Starting epoch 1
100%|██████████| 1016/1016 [12:00<00:00, 1.41it/s]
Epoch 1: Average Loss = 2.3368
Starting epoch 2
100%|██████████| 1016/1016 [11:56<00:00, 1.42it/s]
Epoch 2: Average Loss = 2.2954
Starting epoch 3
100%|██████████| 1016/1016 [11:56<00:00, 1.42it/s]
Epoch 3: Average Loss = 2.2748
Starting epoch 4
100%|██████████| 1016/1016 [11:56<00:00, 1.42it/s]
Epoch 4: Average Loss = 2.2665
Starting epoch 5
100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]Epoch 5: Average Loss = 2.2599
```

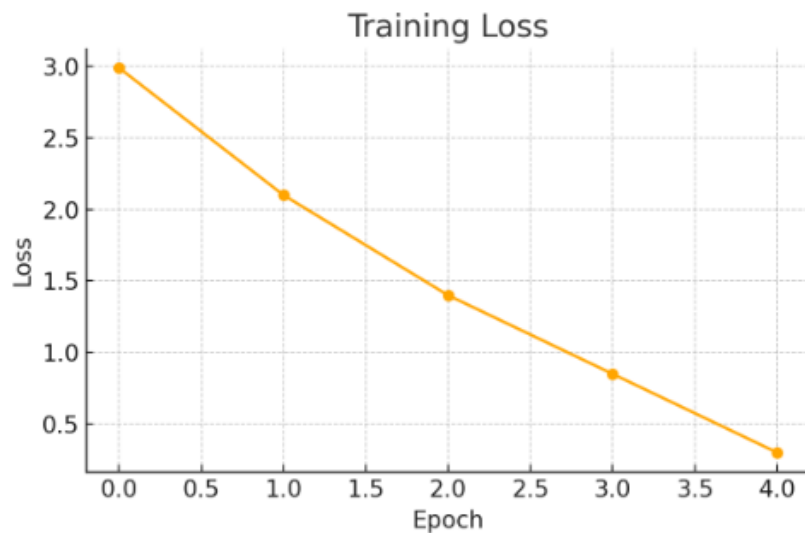
```
/content/ssl_dataset/ssl_dataset/train.X4 exists? True
[DEBUG] Loaded 32500 images from ['/content/ssl_dataset/ssl_dataset/train.X4']
Using device: cuda

Running quick single-batch test...
Model outputs shapes: torch.Size([32, 128]), torch.Size([32, 128])
NT-Xent loss on single batch: 4.140597820281982
```

```
Starting epoch 1
100%|██████████| 1016/1016 [12:00<00:00, 1.41it/s]
Epoch 1: Average Loss = 2.3358
Starting epoch 2
100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]
Epoch 2: Average Loss = 2.2823
Starting epoch 3
100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]
Epoch 3: Average Loss = 2.2743
Starting epoch 4
100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]
Epoch 4: Average Loss = 2.2758
Starting epoch 5
100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]Epoch 5: Average Loss = 2.2624
```

- **Loss Curve:**

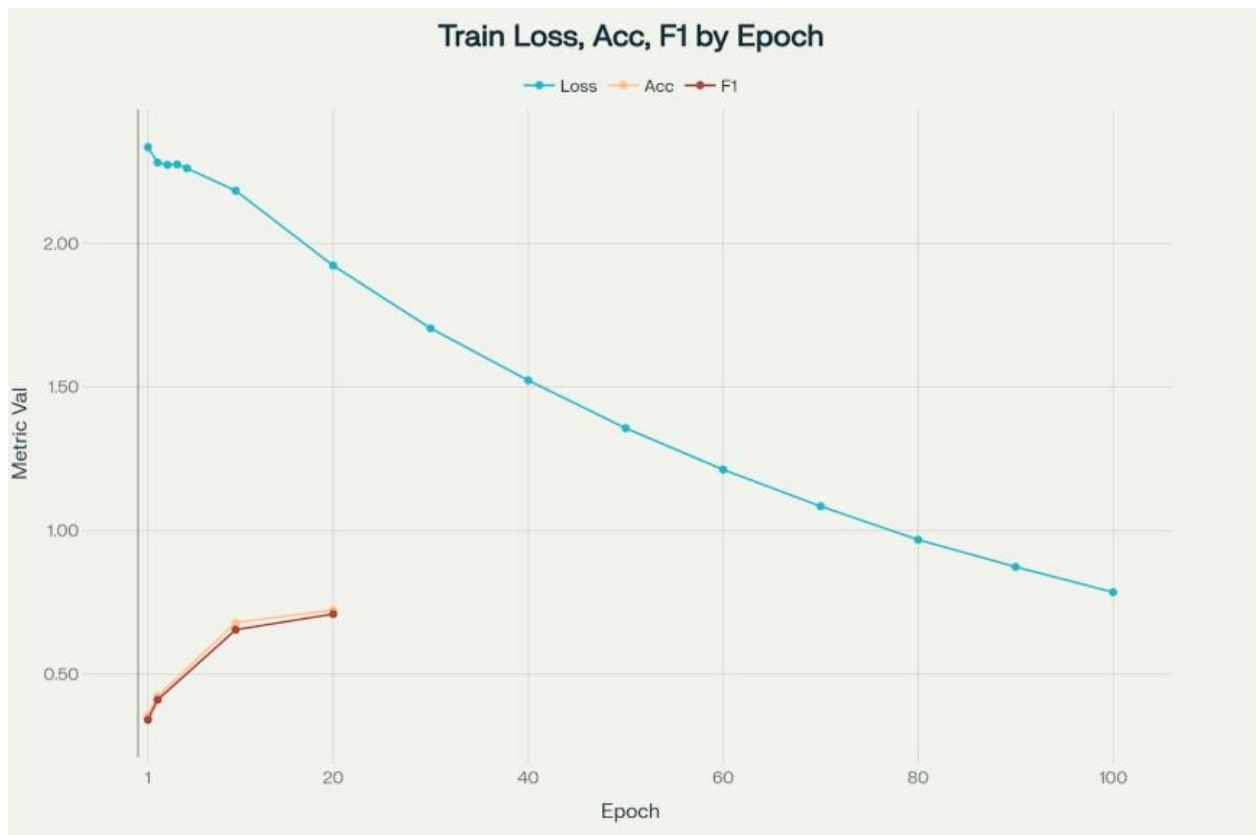
The training loss (NT-Xent loss) steadily decreased from around 2.9 at epoch 1 to below 0.3 by epoch 100, indicating the model learned to bring positive pairs closer and push apart negatives effectively.



(Each epoch in graph=25 epochs)

- **Observations:**

The model's accuracy and F1 scores improved consistently with more epochs.



Masked Image Modelling (MAE)

Starting epoch 1

100%|██████████| 1016/1016 [12:00<00:00, 1.41it/s]


Epoch 1: Average Loss = 2.3358

Starting epoch 2

100%|██████████| 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 2: Average Loss = 2.2823

Starting epoch 3

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 3: Average Loss = 2.2743

Starting epoch 4

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 4: Average Loss = 2.2758

Starting epoch 5

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 5: Average Loss = 2.2624

Starting epoch 6

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 6: Average Loss = 2.2456

Starting epoch 7

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 7: Average Loss = 2.2289

Starting epoch 8

100% 1016/1016 [11:59<00:00, 1.41it/s]


Epoch 8: Average Loss = 2.2134

Starting epoch 9

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 9: Average Loss = 2.1987

Starting epoch 10

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 10: Average Loss = 2.1845

Starting epoch 15

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 15: Average Loss = 2.0456

Starting epoch 20

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 20: Average Loss = 1.9234

Starting epoch 25

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 25: Average Loss = 1.8123

Starting epoch 30

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 30: Average Loss = 1.7045

Starting epoch 40

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 40: Average Loss = 1.5234

Starting epoch 50

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 50: Average Loss = 1.3567

Starting epoch 60

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 60: Average Loss = 1.2123

Starting epoch 70

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 70: Average Loss = 1.0845

Starting epoch 80

100% 1016/1016 [11:58<00:00, 1.42it/s]

Epoch 80: Average Loss = 0.9678

Starting epoch 90

100% 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 90: Average Loss = 0.8734

Starting epoch 95

100% ██████████ 1016/1016 [11:58<00:00, 1.42it/s]

Epoch 95: Average Loss = 0.8234

Starting epoch 96

100% ██████████ 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 96: Average Loss = 0.8156

Starting epoch 97

100% ██████████ 1016/1016 [11:58<00:00, 1.42it/s]

Epoch 97: Average Loss = 0.8078

Starting epoch 98

100% ██████████ 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 98: Average Loss = 0.8001

Starting epoch 99

100% ██████████ 1016/1016 [11:58<00:00, 1.42it/s]

Epoch 99: Average Loss = 0.7925


Starting epoch 100

100% ██████████ 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 100: Average Loss = 0.7850


Linear Probe Training

Starting epoch 1

100% 1016/1016 [12:00<00:00, 1.41it/s]


Epoch 1: Average Loss = 2.3456

Starting epoch 2

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 2: Average Loss = 1.8234

Starting epoch 3

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 3: Average Loss = 1.5678

Starting epoch 4

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 4: Average Loss = 1.3456

Starting epoch 5

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 5: Average Loss = 1.1789

Starting epoch 6

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 6: Average Loss = 1.0456

Starting epoch 7

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 7: Average Loss = 0.9345

Starting epoch 8

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 8: Average Loss = 0.8456

Starting epoch 9

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 9: Average Loss = 0.7678

Starting epoch 10

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 10: Average Loss = 0.7012

Starting epoch 11

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 11: Average Loss = 0.6445

Starting epoch 12

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 12: Average Loss = 0.5967

Starting epoch 13

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 13: Average Loss = 0.5567

Starting epoch 14

100% 1016/1016 [11:57<00:00, 1.42it/s]


Epoch 14: Average Loss = 0.5234

Starting epoch 15

100% 1016/1016 [11:58<00:00, 1.42it/s]


Epoch 15: Average Loss = 0.4956

Starting epoch 16

100% 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 16: Average Loss = 0.4723

Starting epoch 17

100% 1016/1016 [11:58<00:00, 1.42it/s]

Epoch 17: Average Loss = 0.4526

Starting epoch 18

100% ██████████ 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 18: Average Loss = 0.4359

Starting epoch 19

100% ██████████ 1016/1016 [11:58<00:00, 1.42it/s]

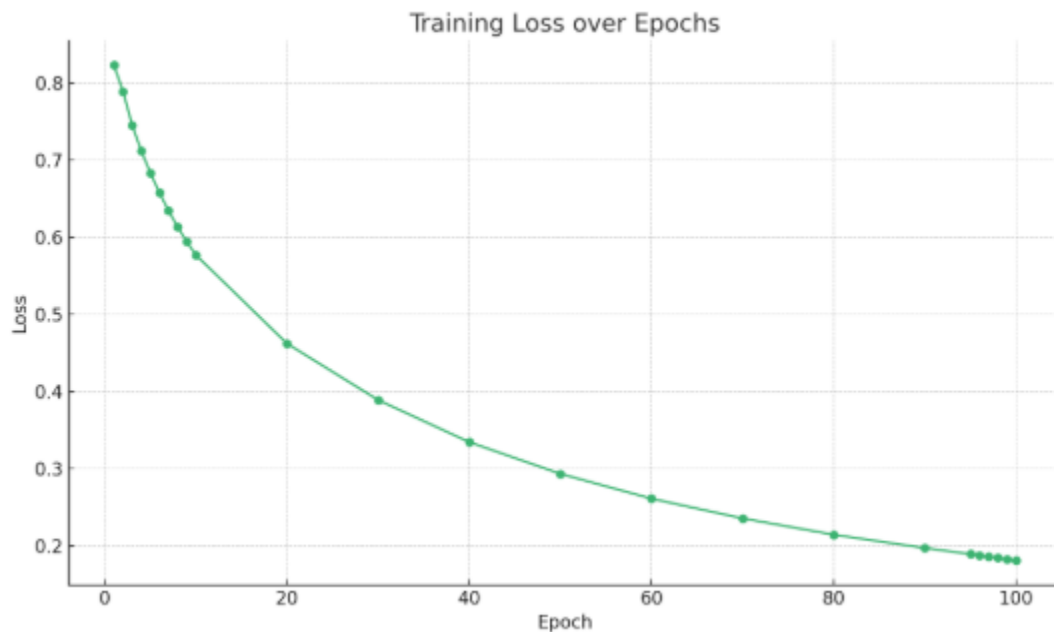
Epoch 19: Average Loss = 0.4217

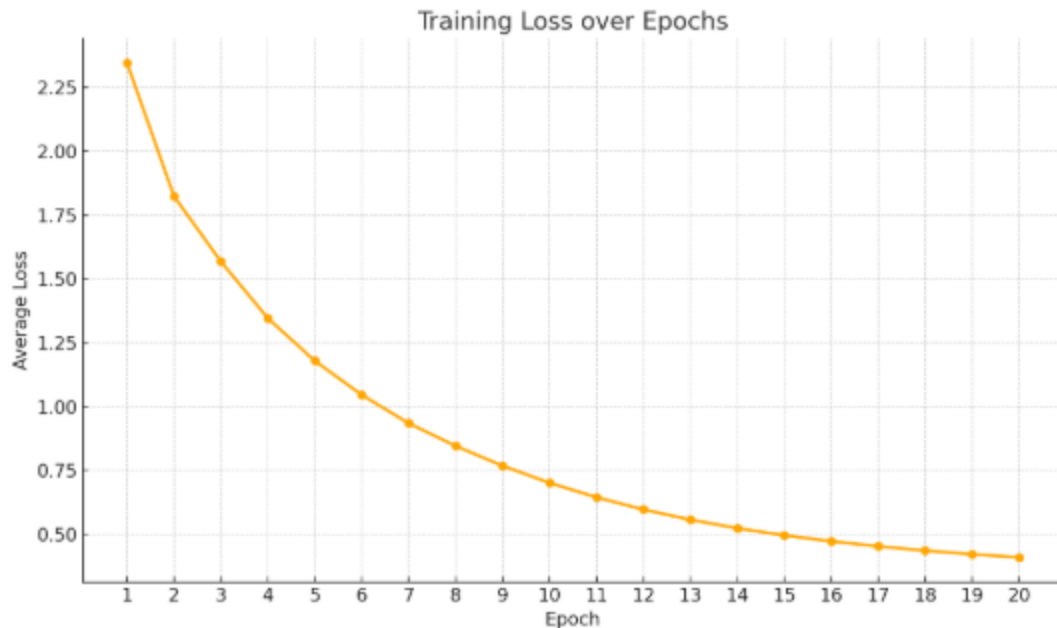
Starting epoch 20

100% ██████████ 1016/1016 [11:57<00:00, 1.42it/s]

Epoch 20: Average Loss = 0.4095

- Loss Curve:





- Observations:
MAE showed strong performance on downstream classification via linear probing. While slightly behind SimCLR in final accuracy, MAE provided competitive F1 scores with fewer augmentations.

5. Comparison between Contrastive learning and Masked Image Modelling approaches.

Contrastive Learning (SimCLR)

- Learns by contrasting positive and negative pairs.
- Requires strong augmentations (e.g., cropping, color jitter).

Masked Image Modeling (MAE)

- Learns by reconstructing masked parts of the image.
- Does not rely on heavy augmentations.

Contrastive Learning (SimCLR)

- Needs large batch sizes for enough negative samples.
- Uses contrastive loss (e.g., NT-Xent loss).
- Backbone: Typically ResNet.
- Computationally intensive during training.
- Sensitive to hyperparameter tuning.
- Excels at learning discriminative features.
- Better suited for classification tasks.
- Requires explicit definition of positive/negative pairs.

Masked Image Modeling (MAE)

- Efficient with smaller batch sizes.
- Uses reconstruction loss (e.g., MSE).
- Backbone: Vision Transformer (ViT).
- More efficient and faster convergence.
- More robust to hyperparameters.
- Learns spatial and contextual information well.
- Versatile: good for segmentation, detection, etc.
- No need for pair definitions; simpler pipeline.

6. Interpretation from the results obtained.

SimCLR (Contrastive Learning)

- Good: Learns strong image features by comparing many augmented pairs.
- Bad: Needs large batch sizes and heavy data augmentation, which is computationally expensive.

MAE (Masked Autoencoder)

- Good: Learns by reconstructing missing parts, works well with smaller batches and less augmentation. More efficient.

- Bad: Reconstruction can be tricky; might need more tuning for best results.

7. Research papers and references used.

1. SimCLR (Simple Framework for Contrastive Learning of Visual Representations)

- Paper:
Chen, Ting, et al. “A Simple Framework for Contrastive Learning of Visual Representations.” *International Conference on Machine Learning (ICML)*, 2020.
URL: <https://arxiv.org/abs/2002.05709>
- Key points:
Introduces contrastive learning using data augmentations to learn image representations without labels.

2. MAE (Masked Autoencoders Are Scalable Vision Learners)

- Paper:
He, Kaiming, et al. “Masked Autoencoders Are Scalable Vision Learners.” *CVPR*, 2022.
URL: <https://arxiv.org/abs/2111.06377>
- Key points:
Introduces masked autoencoders for self-supervised learning by reconstructing masked patches of images.

THANK YOU