**TML Assignment – 2**

Team number – 39

Team token - 09596680

**Name1:** Soham Ritesh Tolwala

**Matriculation number:** 7076028

**Email id1:** [soto00002@stud.uni-saarland.de](mailto:soto00002@stud.uni-saarland.de)

**Name2:** Laxmiraman Dnyaneshwar Gudewar

**Matriculation number:** 7076312

**Email id2:** [lagu00003@stud.uni-saarland.de](mailto:lagu00003@stud.uni-saarland.de)

## **Introduction:**

This assignment focuses on stealing a high-utility encoder protected by the B4B (Bucks for Buckets) defense. The victim model exposes only 1024-dimensional representation vectors for given images through an API with a limited query quota. The goal was to reverse-engineer a surrogate encoder that closely mimics the victim’s embedding behavior, evaluated using the average L2 distance on a hidden test set via a public leaderboard.

## **Understanding the setup:**

|  |  |
| --- | --- |
| File / Directory | Description |
| data/modelstealingpub.pt | Public dataset of 1000 unlabeled images used to query the victim model. |
| Victim api (query access) | Provides 1024-dimensional representation vectors per image; no labels or logits exposed. |
| extract/api\_client.py | Handles API communication, caches embeddings, and logs quota usage. |
| extract/extract\_embeddings.ipynb | Executes image queries to the victim API, extracts and stores embeddings in .npz format. |
| train/train\_ssl.ipynb | Main training script for surrogate models using weighted MSE, SimCLR, and momentum teacher. |
| experiment\_colab.ipynb | Contains all the experiments and approaches tried out along with training graph plots. |
| models/ | Stores all trained PyTorch student models (with and without fixer layers). |
| results\_graphs/ | Contains matplotlib plots of training loss for each method and variation (SimCLR, EMA, etc). |
| submission\_simclr.py | Loads best model, converts to ONNX, runs validation, and submits to leaderboard. |

## **Approaches Attempted:**

1. **Direct MSE Regression**
   * **Goal**: Directly map student embeddings to the victim’s outputs using Mean Squared Error.
   * **Model**: ResNet18 + MLP head.
   * **Training**: Single view per image, no augmentation.
   * **Outcome**: Reasonable initial results but quickly plateaued due to overfitting and the noise introduced by B4B.
   * *Graph Ref*: MSE\_alpha09.png
2. **SimCLR-style Dual-View Weighted MSE**
   * **Goal**: Learn from two augmented views using a mix of alignment and consistency loss.
   * **Loss**:
   * **Alpha tuning**: Found α = 0.9 to be optimal.
   * **Outcome**: Smooth convergence with early saturation.
   * *Graph Ref*: SimCLR-style Stealing Loss and MSE\_alpha09.png
3. **SimCLR + Cosine Annealing Scheduler (300 epochs)**
   * **Goal**: Extend training to longer schedules using cosine LR decay.
   * **Observation**: Training remained stable and smooth for over 300 epochs.
   * **Outcome**: Further reduced loss, slightly improved generalization.
   * *Graph Ref*: MSE\_300optim.png
4. **Momentum Teacher (EMA) + Weighted MSE (Student-Teacher SimCLR)**
   * **Setup**: Student learns from both (i) victim and (ii) a teacher network (EMA of student weights).
   * **Effect**: Stabilizes training and reduces sharp gradient shifts.
   * **Outcome**: Best performing model (L2 ≈ 6.0301), highly stable after 100 epochs.
   * *Graph Ref*: Student-Teacher SimCLR Loss
5. **Momentum Teacher Fine-Tuning (Post 100 Epochs)**
   * **Improvement**: After base EMA model saturated, fine-tuned for another 50 epochs.
   * **Observation**: Minor dips in loss after epoch 100, then smooth flattening.
   * *Graph Ref*: Student-Teacher SimCLR Loss (with Fine-Tuning)
6. **NT-Xent Contrastive Loss (τ = 1.0)**
   * **Objective**: Use normalized temperature-scaled cross-entropy loss for contrastive training.
   * **Result**: High final loss (L2 ≫ 10); contrastive objectives failed to learn under B4B perturbations.
   * *Graph Ref*: NT-Xent Training Loss (τ=1.0)
7. **Cosine Similarity Contrastive Loss**
   * **Loss**: Minimize cosine distance between embeddings.
   * **Result**: Even with normalized projections, cosine loss failed to learn meaningful alignments.
   * *Graph Ref*: cosine\_norm.png
8. **Siamese Neural Network (SNN) Loss**
   * **Setup**: Distance-based loss using positive/negative pair sampling.
   * **Outcome**: Similar behavior to contrastive losses—sensitive to embedding distortion under B4B.
   * *Graph Ref*: SNN Training Loss
9. **DINO-style Predictive Coding (100 & 150 Epochs)**
   * **Goal**: Implement DINO’s self-distillation with vision transformers replaced by CNNs.
   * **Result**: Despite longer training, yielded poor L2 (~28+). DINO struggled with B4B-induced signal loss.
   * *Graph Refs*: DINO.png, DINO\_150.png
10. **Post-hoc Orthogonal Procrustes Fixer**
    * **Idea**: Fit a linear orthogonal transformation between student and victim embeddings using SVD.
    * **Result**: Significant post-training improvement; the fixer compensates for user-specific affine noise added by B4B.
    * **Code**: Implemented as a frozen nn.Linear(1024, 1024) layer added to the student encoder.

## **Results:**

|  |  |  |
| --- | --- | --- |
| **Model File Name** | **Method Description** | **L₂ Distance** |
| model\_Sim\_CLR.onnx | SimCLR-style, α = 0.5 | 6.81 |
| model\_Sim\_CLR\_07.onnx | SimCLR-style, α = 0.7 | 6.6186 |
| model\_Sim\_CLR\_09.onnx | SimCLR-style, α = 0.9 | 6.6148 |
| model\_Sim\_CLR\_MSE\_loss\_opt.onnx | Optimized SimCLR with scheduler (ResNet18) | **6.0300** |
| model\_Sim\_CLR\_Procrustes.onnx | Post-hoc Procrustes alignment | *6.41* |
| model\_Sim\_CLR\_Procrustes200.onnx | Post-hoc Procrustes with more epochs and increased α = 0.98 | *6.22* |
| model\_Sim\_CLR\_whitening\_procrustes.onnx | Added Embed Whitening / Covariance Alignment. | *6.24* |
| model\_Sim\_CLR\_cosine\_norm\_loss.onnx | Cosine norm loss with α = 0.9 | 19.93 |
| model\_Sim\_CLR\_cosine.onnx | Cosine loss with α = 0.7 | *Not tested* |
| model\_Sim\_CLR\_NT\_xent.onnx | NT-Xent, τ = 1.0 | 35.20 |
| model\_Sim\_CLR\_resnet50.onnx | Weighted MSE (α tuned), ResNet50 backbone | 22.044 |
| model\_Sim\_CLR\_DINO\_100.onnx | DINO-style SSL (100 epochs) | *Not tested* |
| model\_Sim\_CLR\_DINO\_150\_optim.onnx | DINO-style SSL, optimized (150 epochs) | 28.60 |

**Takeaway:**

Despite multiple variants, the Optimised SimCLR + scheduler run (α = 0.98, ResNet-18) remains the top performer at *L₂ = 6.03*. Procrustes plus whitening gave small but insufficient gains; deeper backbones and contrastive/DINO objectives consistently under-performed under B4B noise.

**Challenges Faced**

1. **B4B Noise vs. Data Budget**  
   The adaptive noise scales with the number of hash buckets we touch. Re-querying the victim for strong augmentations pushed bucket coverage up faster than expected, so we had to redesign the pipeline to reuse one embedding for KKK offline views.
2. **Whitening Stability**  
   Online (per-batch) SVD whitening froze the GPU after a few minutes. Switching to a *single* offline SVD and caching a fixed whitening matrix solved the runtime issue but required careful ε-regularisation to avoid exploding values.
3. **Torch < 2.0 vs. torchvision 0.16**  
   A hidden import of torch.\_dynamo.config inside torchvision broke the optimiser call. We added a tiny stub module (two lines) to satisfy the import without upgrading CUDA wheels mid-project.
4. **PIL Collation Crash**  
   The default PyTorch collate function cannot batch raw PIL images. A one-line ImgTensorDataset wrapper that converts every image to a tensor in \_\_getitem\_\_ fixed the DataLoader error during the Procrustes step.
5. **Loss-scale Interpretation**  
   Whitened MSE values are orders of magnitude larger than raw L₂, making early stopping non-intuitive. We logged both raw and whitened losses to verify real progress.

**Potential improvements:**

We can still squeeze the L₂ further by (i) re-using each victim embedding for dozens of offline augmentations (the “StolenEncoder” trick), which multiplies training signal while hardly increasing B4B bucket coverage; (ii) swapping point-wise MSE for a Soft Nearest-Neighbour loss that is more tolerant to B4B’s affine-plus-noise distortions; (iii) fitting Procrustes in a 256-D PCA sub-space to remove noisy directions before projecting back; and (iv) averaging two independent projection heads at inference for quick, decorrelated error reduction. Together these tweaks should nudge the score below the current 6.03 plateau.