

Final Project: Self-Supervised Representation Learning

Deep Learning (Fall 2025)

Instructor: [Your Name]

[Release: 2025-11-05] — [Milestone: 2025-11-21] — [Final Due: 2025-12-12]

Overview

In this project, you will design and implement a self-supervised learning (SSL) method to learn visual representations from a large unlabeled image corpus. You are encouraged to explore this as an open-ended research problem: propose algorithms, architectures, losses, optimization schemes, augmentations, or data pipelines. The final model must be fewer than **100M parameters**. All models must be **randomly initialized** (no pretraining or weights distilled from pretrained models).

We will provide:

- **Unlabeled Pretraining Set** (500,000 images) for SSL.
- **Public Downstream Evaluation Dataset** with a labeled split (train/val/test) for linear-probe evaluation only.
- **Held-out Private Test Set** used by staff for final evaluation.

Absolutely prohibited: training on any public test images in any way. Violations receive a grade of 0 and an academic dishonesty report.

Data and Splits

Unlabeled Pretraining Set (`pretrain/`). Approximately 500k images for learning representations without labels. Use this set exclusively for SSL pretraining of your backbone.

Public Downstream Evaluation (`eval_public/`). A labeled dataset for linear-probe evaluation of your learned features. Provided as:

- `train/`: labeled images for training a *linear* classifier on top of the *frozen* backbone.
- `val/`: labeled images for validation and hyperparameter tuning of the linear head only.
- `test/`: images with hidden labels for public reporting.¹

Private Held-out Test (`eval_private/`). A staff-only test set for final ranking and grading. You will submit your backbone and evaluation scripts; staff will run the linear probe and evaluation on this held-out set.

¹You may extract features and run inference, but you may not update weights using these images.

Rules and Restrictions

1. **No training on test images.** You may not use *any* `eval_public/test` or `eval_private` images for training, adaptation, distillation, feature bank construction, pseudo-labeling, or unsupervised updates. These images may only be used for forward-pass feature extraction and evaluation.
2. **Linear probing only on public labeled splits.** On `eval_public`, you may train *only a linear classification head* over a *frozen* backbone. Train on `train`, tune on `val`, and report results on `test`. Do not use labels from `val` or `test` to update the backbone.
3. **Random initialization.** Initialize all model weights randomly. Do not load any pretrained weights (ImageNet, CLIP, DINO, etc.). No model soups, no partial initialization from pretrained backbones or heads, no distillation from external pretrained teachers.
4. **Model size < 100M parameters.** Total parameters of the train-time backbone must be strictly below 100 million. We will verify programmatically.
5. **Data scope.** Use only the datasets provided. External data (labeled or unlabeled), external pseudo-labels, or metadata is not allowed. Synthetic data is allowed only if generated without pretrained vision models (e.g., pure procedural augmentations). Prior approval is required for any deviation.
6. **Test-time rules.** No test-time training/adaptation. Test-time augmentation (TTA) without learning is allowed. Feature whitening/normalization computed solely on training features is allowed; do not fit any statistics on test images.
7. **Teams.** Teams of 1–3 students. Larger teams must demonstrate proportional scope/rigor.

Suggested Directions (Non-Exhaustive)

- **Contrastive/Instance Discrimination:** SimCLR, MoCo-v2/v3, InfoNCE variants, memory banks, queue designs.
- **Redundancy Reduction:** BYOL, Barlow Twins, VICReg/VICRegL (EMA teacher allowed if randomly initialized).
- **Clustering/Prototypical:** SwAV, DeepCluster-v2, online assignments, balanced prototypes.
- **Masked Image Modeling:** MAE/SimMIM/iBOT-style masked prediction with lightweight decoders (respect parameter cap).
- **Architectures:** ResNet-50/101, ConvNeXt-T/S, MobileOne, ViT-T/S (ensure < 100M params). Explore normalization, activations, efficient blocks.
- **Optimization / Data:** Advanced augmentations, multi-crop, schedules, EMA, regularization, curriculum; custom optimizers (e.g., LARS, AdamW variants).

Deliverables

Project Proposal (1–2 pages).

Problem framing, method sketch, architecture, compute plan, risk mitigation. Due: [Milestone: 2025-11-21].

Checkpoint Update.

2–3 slides or a 1-page memo: current results, ablations planned, blockers. Due: [2025-12-01].

Final Report (6–8 pages).

Method, experiments, ablations, analysis, compute accounting, limitations/ethics, references.
Due: [Final Due: 2025-12-12].

Code and Artifacts.

Reproducible training and evaluation code; configs; pretrained backbone weights; logs; script to run linear probe; environment file.

Evaluation and Grading

- **Performance (Public Linear Probe):** 30% Top-1 accuracy on `eval_public/test` using a linear head trained on `train` and tuned on `val`.
- **Performance (Private Held-out):** 30% Staff-run linear probe on `eval_private` with your backbone and scripts.
- **Scientific Rigor / Analysis:** 20% Ablations, design justifications, sensitivity studies, clarity of empirical conclusions.
- **Reproducibility / Code Quality:** 10% Clean code, deterministic runs, configuration management, documentation.
- **Report Quality:** 10% Clarity, writing, figures/tables, related work positioning, limitations/ethics.

Ties and Statistical Reporting. Report mean \pm std over at least 3 seeds for key results. Use fixed data splits. Staff may re-run to verify.

Submission Instructions

All submissions use the provided template repository structure (released with the dataset):

- `code/`: training and evaluation code (entry points: `train_ssl.py`, `eval_linear.py`).
- `configs/`: YAML or JSON configs for experiments (pretraining, linear probe).
- `scripts/`: `run_pretrain.sh`, `run_linear.sh` documented with arguments.
- `artifacts/`: saved backbone weights (`backbone.pth`), logs, tensorboard, metrics.
- `ENV.md` & `requirements.txt`: exact versions and setup steps.
- `REPORT.pdf`: final paper; `PROPOSAL.pdf`: proposal.

Upload a single archive or private repo link as instructed on the course platform.

Reproducibility Checklist

1. Fixed seeds, determinism flags where feasible; record seeds in logs.
2. Exact data preprocessing and augmentations documented.

3. Full configuration files checked in; command lines to reproduce results.
4. Parameter count report (automated script provided by staff). Must be $< 100\text{M}$.
5. Compute accounting: GPUs, hours, peak memory; any mixed precision usage.
6. Clear separation between SSL pretraining and linear probe stages.
7. No references to external data, models, or weights in code.

Compliance and Auditing

We will run automated checks to enforce rules:

- **Parameter cap.** Programmatic parameter counting on your backbone at train-time.
- **Data usage.** File hash audits to ensure no `eval_public/test` or `eval_private` images are accessed during training. Logs must include dataset roots and counts.
- **Backbone freezing.** Linear probe scripts must verify and assert that backbone gradients are disabled.
- **Initialization.** Code review for checkpoints and weight loading; any pretrained loading is prohibited.

Violations result in a grade of 0 and an academic dishonesty report.

Timeline (Tentative)

- Release (datasets, starter kit): [Release: 2025-11-05]
- Proposal due: [Milestone: 2025-11-21]
- Checkpoint update: 2025-12-01
- Final submission: [Final Due: 2025-12-12]
- Private evaluation and results posted: [2025-12-16]

FAQ (Abbreviated)

Can we use CLIP/DINO/vit pretrained weights? No. Random initialization only.

Can we use external unlabeled data? No, unless explicitly approved in writing before training.

Is EMA teacher allowed? Yes if all components start from random initialization and respect the parameter cap.

Can we tune the backbone with labels? No. Linear probe only; backbone must remain frozen during supervised evaluation.

Can we adapt on test images without labels? No. No learning of any kind on public/private test images.

Are we graded on speed? Not directly. Report compute fairly; efficiency is appreciated.

Academic Integrity

Follow the letter and spirit of the rules. Collaborate within your team; do not share code or weights across teams. Cite all external ideas and implementations you reference. Any attempt to train on test images, use pretrained weights, or otherwise circumvent the rules will be considered academic dishonesty and will result in a 0 for the project and a formal report.

We are excited to see creative and rigorous approaches to self-supervised learning. Treat this as a research problem: be bold, careful, and scientific.