# Deep Learning for Computer Vision HW1 Report

*Note. Only Assignment 1 provides a template. Please make sure you complete all the requirements for each question and respond in the order of the question numbers*

## P1.

(5%) Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (**including but not limited to** the name of the SSL method & data augmentation techniques you used, learning rate schedule, optimizer, and batch size setting for this pre-training phase. 150-300 words are enough)

**Answers (Your implementation details):**

I used the official DINO repository to perform self-supervised pre-training of a ResNet-50 backbone on the Mini-ImageNet dataset. DINO follows a student–teacher framework, where the student learns to match the teacher's outputs, and the teacher is updated by the exponential moving average (EMA) of the student.
For data augmentation, I applied the multi-crop strategy: two global crops (scale 0.14–1.0) and six local crops (scale 0.05–0.14), together with random horizontal flip, color jitter, random grayscale, Gaussian blur, and solarization. These augmentations encourage the model to learn more robust and invariant representations.
The pre-training lasted 200 epochs with batch size 50 per GPU, using AdamW optimizer with weight decay of 1e-4. The learning rate started at 0.03 and followed a cosine annealing schedule with 10 warm-up epochs and a minimum learning rate of 1e-6. The loss function was the DINO loss, essentially the cross-entropy between the student and teacher outputs.
Compared with the default DINO settings, I specifically adjusted several hyperparameters: the global crop scale (0.14–1.0), the local crop scale (0.05–0.14), the number of local crops (six), the batch size per GPU (50), and the training length (200 epochs). These changes made the pre-training fit the Mini-ImageNet dataset and my hardware constraints.

**Parameter:**

| Learning Rate | Batch Size | Optimizer | Scheduler | Epochs | Loss Function |
|---|---|---|---|---|---|
| 0.03 (cosine schedule, warmup=10 epochs, min_lr=1e-6) | 50 (per GPU) | AdamW | Cosine Annealing LR | 200 | DINO Loss |

(20%) Please conduct the Image classification on Office-Home dataset as the downstream task. Also, please complete the following Table, which contains different image classification settings, and **discuss/analyze the results**.

| Set | Accuracy |
|---|---|
| A | 0.5788 |
| B | 0.5542 |
| C | 0.6059 |
| D | 0.1552 |
| E | 0.4828 |

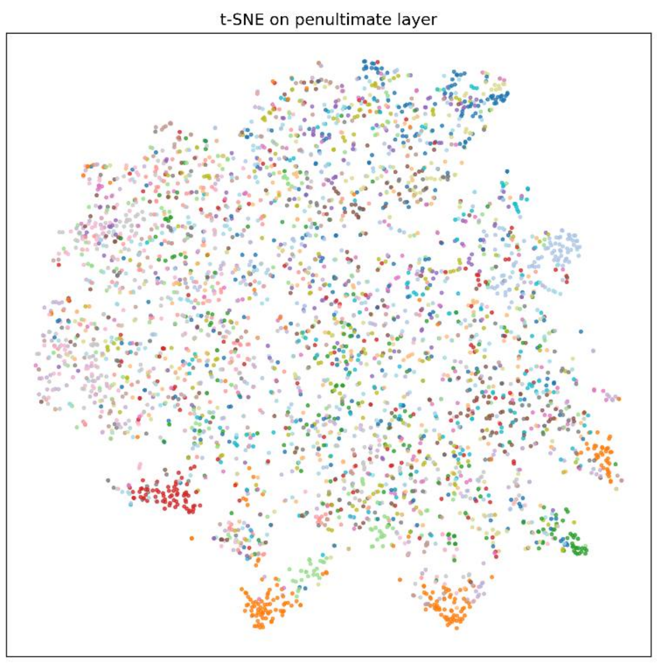
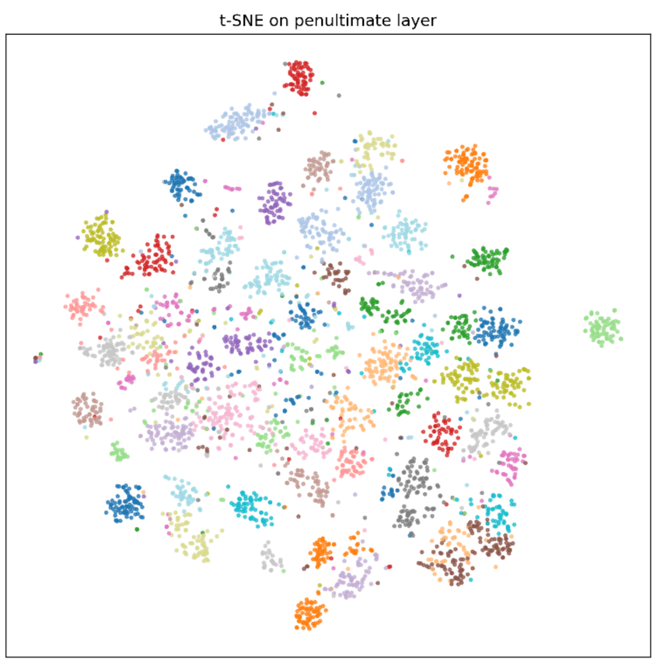
## Answers:

On the validation set, the five settings obtain: A 0.5788, B 0.5542, C 0.6059, D 0.1552, and E 0.4828. Overall, Setting C (DINO pre-training + full fine-tuning) performs the best. This is consistent with common observations in cross-domain transfer: DINO's student–teacher framework with strong multi-crop augmentations (global/local crops, color jitter, grayscale, blur, solarization) learns highly invariant, semantic features. When I fine-tune the whole network end-to-end on Office-Home, these generic representations adapt effectively to the target distribution, leading to superior accuracy over both training from scratch (A) and supervised pre-training (B).
Setting A (from scratch) reaches 0.5788, indicating that the dataset size, augmentations, and LR schedule are sufficient to learn a solid model directly on the target domain. However, it lacks the prior visual knowledge provided by Mini-ImageNet pre-training, so it trails C in generalization and stability. Setting B (ImageNet-supervised pretrain + fine-tune) achieves 0.5542, below A and C; a likely reason is a moderate domain shift between ImageNet and Office-Home. Supervised pre-training shapes decision boundaries around ImageNet labels; without careful adaptation (e.g., layer-wise LR decay, longer warm-up, smaller backbone LR), fine-tuning can settle at a sub-optimal solution.
Linear probing (frozen backbone) clarifies representation quality: E (DINO + frozen) = 0.4828 is far above D (supervised + frozen) = 0.1552. This shows DINO features are substantially more linearly separable on Office-Home even without label-driven adaptation, whereas supervised features remain tied to ImageNet semantics and transfer poorly under a fixed encoder. In short: (1) DINO yields more general, linearly usable features (E ≫ D); (2) peak accuracy still requires full-model adaptation, hence C > E, and end-to-end fine-tuning is key to converting SSL pre-training into the best downstream performance.

(5%) Visualize the learned visual representation of **setting C** on the **train set** by implementing **t-SNE** (t-distributed Stochastic Neighbor Embedding) on the output of **the second last layer**.

a. Depict your visualization from both **the first and the last epochs**.(3%)

b. Briefly explain the results.(2%)

| t-SNE the first epochs | t-SNE the last epochs |
|---|---|
|  |  |

**Answers(Briefly explain the results) :**

In the first-epoch t-SNE, points from different classes are widely intermingled with only a few loose blobs. This is expected under Setting C: although the ResNet-50 backbone starts from DINO features, the linear head is randomly initialized and the whole network has not yet adapted to the Office-Home label space. With a relatively high backbone LR at the beginning (before warm-up ends), features shift and neighborhood structure is unstable, so class manifolds are not yet separated. By the last epoch, clusters become compact and margins between groups widen. This indicates two things: (1) intra-class variance shrinks as the model aligns DINO features to category boundaries; (2) inter-class separation increases as later layers learn class-specific directions. Small islands around large clusters suggest sub-modes (e.g., different domains or viewpoints). Residual overlaps mainly happen among visually similar categories and can also reflect class imbalance or difficult samples

## P 2.

1. **(4%)** Do an ablation study. You need to modify the U-Net model. During training, randomly select one skip connection between encoder and decoder and drop it, so the decoder at that layer will not receive the encoder features. Report and discuss/analyze the performance difference with the standard U-Net (model A).

   **Ans:**

   **Training setup (concise):** 512×512 inputs; U-Net (ResNet-34, ImageNet-pretrained); num_classes=7; ignore_index=0; AdamW (lr=3e-4, wd=5e-4); batch size 16; 200 epochs; encoder frozen for the first 5 epochs.

   **a) Explain you skip which layer.**

   During training, at every iteration I randomly drop exactly one encoder–decoder skip connection in the U-Net. The encoder yields a multiscale pyramid of features at roughly 1/2, 1/4, 1/8, and 1/16 of the input resolution, and the bottleneck is not considered a skip. I uniformly sample one of these lateral connections and zero out the corresponding encoder feature map before it is concatenated in the decoder. At validation and test time no skip is dropped (drop probability is set to 0), so the network behaves exactly like a standard U-Net.

   b) Result for standard U-Net.

   Under the same data, optimizer, and schedule, the vanilla U-Net with all skips intact during both training and evaluation achieves a validation mIoU of 0.7372.

   **c) Result for your skip U-Net.**

   With the modification above—dropping one randomly chosen skip per iteration during training while disabling drops at evaluation—the model reaches a validation mIoU of 0.7362.

   **d) Explain the result.**

   The two scores are essentially identical (a difference of about 0.001), which falls well within normal variance due to random seeds and data augmentation. This indicates that "dropping one skip at a time" is a very mild regularizer in this setting: the U-Net decoder and the remaining multiscale skips provide enough redundancy to compensate for the missing lateral input, and the ImageNet-pretrained encoder further stabilizes the representations. To expose a clearer effect one could always drop the highest-resolution skip, drop multiple skips per iteration, or increase the drop probability over training as a curriculum, possibly in combination with stronger augmentation or boundary-aware losses.

2. **(4%)** Draw the network architecture of the improved model (model B) and explain it differs from your UNet model (model A).

   I use **SegNet (Badrinarayanan et al., TPAMI 2017)** with a VGG16-BN encoder and a decoder based on MaxUnpool with pooling indices, followed by 3×3 Conv-BN-ReLU and a 1×1 classifier. Inputs are 512×512 with ImageNet normalization; training applies scale jitter 0.5–2.0 with synchronized cropping and random flips; RGB (0,0,0) is Unknown (id=6) and excluded from loss/metrics. Loss: class-weighted CE + 0.05 label smoothing. Optimizer: AdamW (wd=1e-4) with a two-tier LR (encoder 0.5×, decoder 1.0×) and per-iteration polynomial decay; AMP enabled. Hyperparams: epochs=400, batch=16, lr=3e-4. Validation uses hflip-TTA and selects the best checkpoint by mIoU excluding Unknown. Compared to U-Net (no skips), SegNet's index-guided unpooling yields more precise localization and better boundaries/small objects without learnable upsampling.

   network architecture of the improved model (model B (SegNet)):

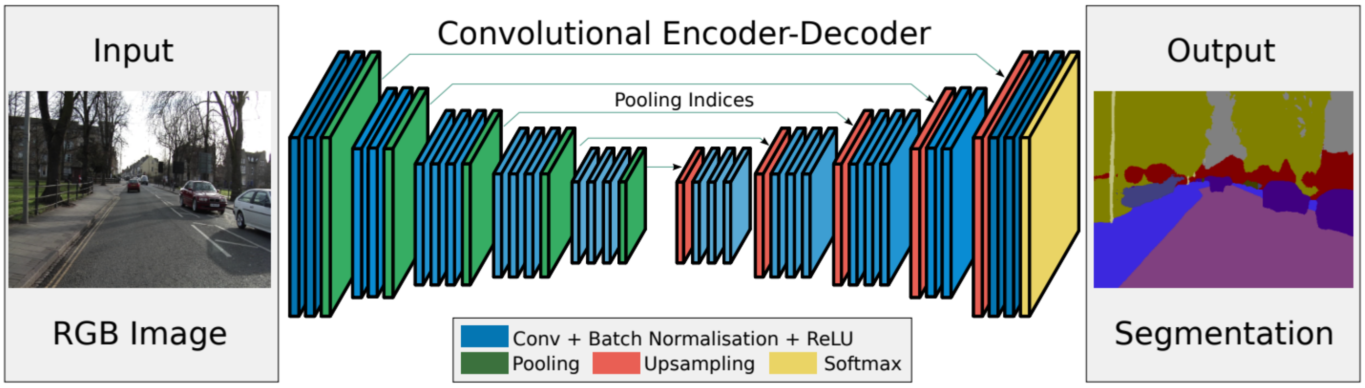


Figure 1. SegNet  architecture (source: Badrinarayanan et al., 2017, arXiv:1511.00561).

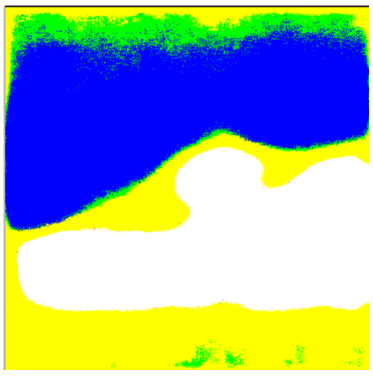
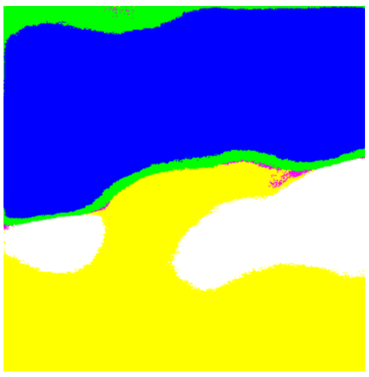
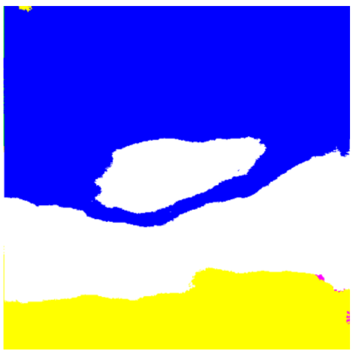
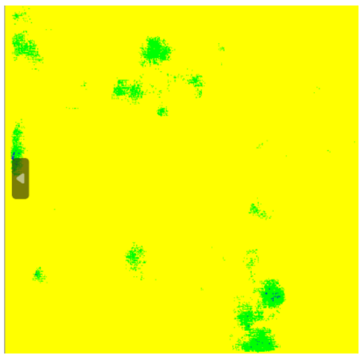
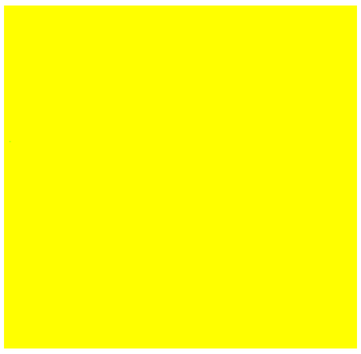
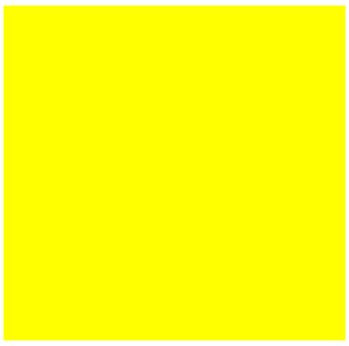
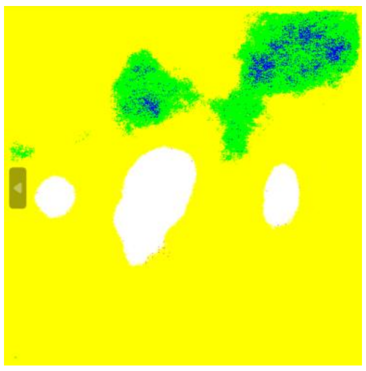
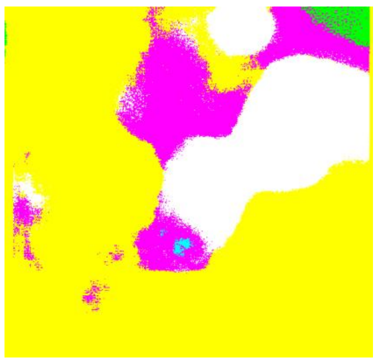
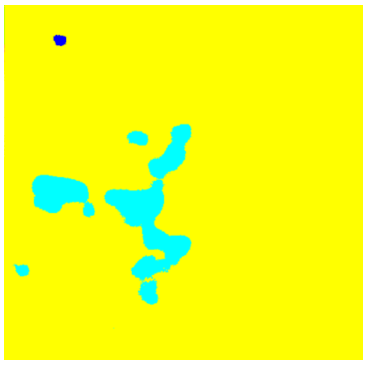
   Differentiation:

| Aspect | Model A: U-Net (no skip connections) | Model B: SegNet |
|---|---|---|
| Upsampling | Bilinear/transpose conv | MaxUnpool + pooling indices |
| Spatial cues | Weaker (no encoder→decoder concat) | Precise locations restored by indices |
| Detail recovery | Harder to recover fine details | Better boundaries & small objects |
| Parameters / memory | Params depend on upsampling; no skip tensors | Few params; store indices (small) |
| Pretraining compatibility | Encoder can use ImageNet, benefits don't pass to decoder | VGG16-BN encoder fits ImageNet cleanly |

3. **(2%)** Report mIoUs of two models on the validation set.

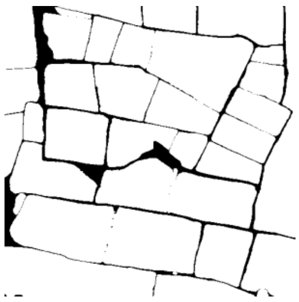
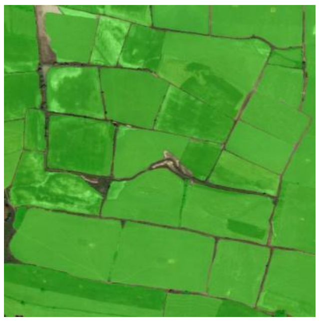
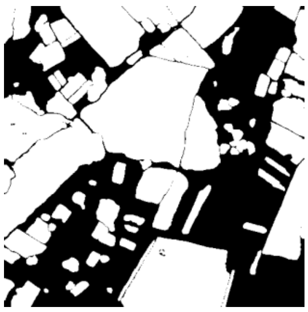
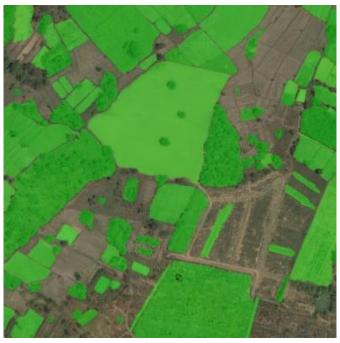
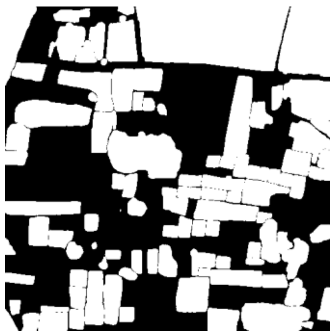
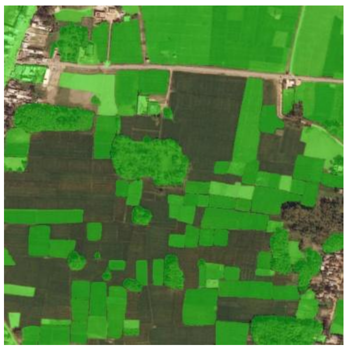
| A | B |
|---|---|
| 0.7372 | 0.7441 |

4. **(3%)** Show the predicted segmentation mask of "validation/0013_sat.jpg", "validation/0065_sat.jpg", "validation/0104_sat.jpg" during the early, middle, and the final stage during the training process of the improved model.

| | Epoch 1 | Epoch 100 | Epoch Best |
|---|---|---|---|
| 0013_sat.jpg |  |  |  |
| 0065_sat.jpg |  |  |  |
| 0104_sat.jpg |  |  |  |

5. (7%) Use segment anything model (SAM) to segment three of the images in the validation dataset, report the result images and the method you use.

**Your Answer:**

| | Mask visuals | Overlay visuals |
|---|---|---|
| 0000_sat.jpg |  |  |
| 0002_sat.jpg |  |  |
| 0003_sat.jpg |  |  |

I applied the Segment-Anything Model (SAM) with the **ViT-H** backbone to automatically segment three validation images (*0000_sat.jpg*, *0002_sat.jpg*, *0003_sat.jpg*). Using the **SamAutomaticMaskGenerator**, I produced binary masks without manual prompts of any kind (no bounding boxes or point inputs). I then overlayed each mask onto its original RGB image using a semi-transparent green color and alpha = 0.4, so that predicted mask areas are visible while preserving underlying texture. From inspection, the resulting masks effectively cover most of the large field regions in each image; for example, in *0000_sat.jpg* the major contiguous crop areas align well, whereas in *0002_sat.jpg* and *0003_sat.jpg* some small or narrow gaps (e.g. between fields, roads, or non-crop background) are either missed or over-included. These overlay visualizations reveal that SAM works quite well for gross segmentation of homogeneous large areas but struggles with fine boundary delineation and small isolated patches