CSE164 Final Project Report

Semi-Supervised Image Classification & Segmentation

Adan Shafei

June 2025

Overview

In this project, I implemented a semi-supervised pipeline that performs both image classification and pixel-wise segmentation on a subset of ImageNet. By combining a pretrained backbone, pseudo-label augmentation, multi-loss training, and test-time enhancements, I achieved a public leaderboard score of **0.45947**, securing the top rank and comfortably surpassing the instructor's baseline of 0.19856.

1. Running the Code

To reproduce the results:

1. Clone or download the repository and navigate to its root directory.
2. Unzip the provided model weights:
```
unzip model_weights.zip -d weights/
```
3. Open `CSE164_Final_Project.ipynb` in Jupyter or Google Colab.
4. In Colab, enable GPU acceleration: `Runtime` → `Change runtime type` → `GPU`.
5. (Optional) Create a virtual environment:
```
conda create -n cse164 python=3.10
conda activate cse164
pip install pandas scikit-learn opencv-python sympy
pip install torch torchvision torchaudio --index-url https://
    download.pytorch.org/whl/cu118
```
6. Execute all notebook cells in order; the final cell will generate `submission.csv`.
7. In Canvas, submit a single ZIP containing:
    - This report (PDF or Markdown).

- All code files (.ipynb or .py).
- A link to the Google Drive file:
  `https://drive.google.com/file/d/13gU55GIEpT6hr-RjB30bOV8CPb4h5BzL/view?usp=drive_link`
- `submission.csv` produced by the notebook.

2. Experimental Design

2.1 Data Preparation

The dataset is organized as follows:

- `train-semi/`: 50 class-labeled folders of images.
- `train-semi-segmentation/`: corresponding ground-truth masks.
- `unlabeled/`: additional images used for pseudo-label generation.
- `test/`: 752 images for final predictions.

Images and masks are resized to 224×224. Training transforms include random crops, flips, rotations, and color jitter; inference uses center crop and normalization.

2.2 Model Architecture

My network uses a frozen ResNet-18 backbone (up to `layer4`), followed by two specialized heads:

- **Classification head:** Adaptive average pooling into a 512-d vector, dropout (0.5), then a linear layer mapping to 50 classes.
- **Segmentation head:** a 1×1 convolution projecting to 50 channels, then bilinear upsampling to the original 224×224.

2.3 Training Strategy

Training was divided into four key phases:

1. *Warm-up (5 epochs):* train classification head only (cross-entropy loss).
2. *Joint training (10 epochs):* optimize both heads with cross-entropy for classification and combined cross-entropy + Dice loss for segmentation (segmentation loss weighted by 0.5).
3. *Pseudo-labeling:* after epoch 5, generate high-confidence class labels (threshold 0.8) on the unlabeled pool, append these samples to the training set, and rebuild the data loader.

4. *Segmentation fine-tune (20 epochs):* freeze backbone and classification head, train only the segmentation head with cross-entropy to refine mask accuracy.

All optimization used AdamW with weight decay 1×10^{-4}, learning rates ranging from 5×10^{-4} to 1×10^{-3}, and a cosine-annealing scheduler.

3. Results

Stage	Epochs	Public IoU	Comment
Baseline (random)	0	0.020	No training
Cls-only warm-up	5	0.116	CE loss
Seg-only fine-tune + TTA	20	0.125	Mask CE
Pretrained joint (CE+Dice)	10	0.185	Weighted seg + Dice
Full pipeline + pseudo labels	20+	0.215	+ pseudo-labeling
Final submission	–	**0.45947**	Ranked #1

Table 1: Public leaderboard scores as the pipeline evolved.

The final submission more than doubled the baseline, demonstrating the strength of combining semi-supervision and test-time augmentation.

4. Insights from the Kaggle Competition

Throughout the competition, I observed:

- extbfPretrained features accelerate convergence and improve generalization in both classification and segmentation tasks.
- extbfPseudo-label augmentation effectively leverages unlabeled data to boost class prediction accuracy.
- extbfDice loss adds robustness to segmentation boundaries, especially when combined with cross-entropy.
- extbfTest-time augmentation and simple post-processing (median filtering) yield consistent gains without extra training.
- Rapid iterations on Colab's GPU environment enable efficient experimentation under tight deadlines.

5. Conclusion

By integrating a pretrained backbone, multi-head multi-loss training, pseudo-labeling, and test-time techniques, I delivered a top-ranked solution achieving an IoU of 0.45947. The full code, model weights, and submission file are provided for replication and further study.