ELEC 475

Lab 3 – Semantic Segmentation & Knowledge Distillation

Lab 3 – Semantic Segmentation & Knowledge Distillation

Naser Al-Obeidat and Michael Greenspan

Contents

1. Introduction

The goal of this lab is to explore the fundamentals of semantic segmentation and apply a knowledge distillation technique. Semantic segmentation involves generating a pixel-wise segmentation mask for an image, assigning classification labels to each pixel. Knowledge distillation, as introduced by Hinton et al., transfers knowledge from a larger model to a smaller one, enabling deployment on devices with limited computational resources (e.g., phones or smartwatches) while maintaining reasonable accuracy and processing speed. In this lab, you will design, implement, train, and evaluate a compact model that distills knowledge from an FCN-ResNet50 pre-trained on the PASCAL VOC 2012 Dataset.

You may use any large language model (e.g., ChatGPT, Claude) for this assignment. When prompting, be explicit and precise about what you want. Always test and fact-check every piece of code the model generates — do not accept code outputs at face value. Verify imports, function names, arguments, and tensor shapes, etc. - and of course make sure that it runs correctly!

2. Task

Your task comprises the following 4 steps:

2.1 Step One – Test Pretrained FCN-ResNet50 on PASCAL VOC 2012 Dataset

Create a test script that loads PyTorch's pretrained FCN-ResNet50 model, downloads the Pytorch segmentation version of PASCAL VOC 2012 dataset, and evaluates the model using the mean Intersection over Union (mIoU) metric provided.

Hint 1: Make the most of PyTorch's official documentation and tutorials—they provide valuable insights into the task, the mIoU metric, and can help streamline utility function setup, saving time and enhancing your understanding.

Hint 2: You may find it useful to prompt your LLM to draft a minimal example of loading and evaluating PyTorch's pretrained FCN-ResNet50 on the PASCAL VOC 2012 dataset. Ask for clear code showing dataset preparation, model evaluation, and mIoU calculation. Examine every import, tensor shape, and function call before running—treat the response as the starting point, not as the final code.

2.2 Step Two – Implement Your Own Model

Design a compact, small-scale model with a custom architecture, involving elements such as convolution kernels, paddings, strides, activation functions, and other standard network layers and hyperparameters. Apply techniques learned in class and from readings to optimize performance.

Focus on keeping your model lightweight, with a small number of trainable parameters, as this aligns with the lab objective. The evaluation will penalize models with larger scales, so efficiency is key.

Hint 3: When designing your compact segmentation network, try describing your intended architecture to the LLM in detail (e.g. "MobileNetV3-Small encoder with ASPP decoder and bilinear upsampling"). Request that it generate a clean nn.Module skeleton. This helps you confirm layer connections and parameter counts before you begin implementing and tuning your own version.

Example Architecture

Backbone (encoder):	MobileNetV3-Small feature extractor (torchvision.models.mobilenet_v3_small pretrained=True). Use features up to stride 16.
Context module:	ASPP (Atrous Spatial Pyramid Pooling) with dilation rates {1,6,12,18}.
Decoder:	skip connection from low-level features + upsampling + final 1×1 classifier to 21 classes (VOC 2012).
Upsampling:	bilinear.
Dropout:	0.5 before the final classifier head.
Feature taps:	
	low: stride≈4, channels=24
	mid: stride≈8, channels=40
	high: stride≈16, channels=576
	These three taps are used for feature-based KD ("low", "mid", "high").

2.3 Step Three – Train & Test Your Model

Implement the training and testing process for your custom model using the Pytorch segmentation version of PASCAL VOC 2012 dataset. Reuse code from previous labs to avoid redundant work and take advantage of PyTorch's dataset object for streamlined data prepration.

Hint 4: Image sizes vary in the PASCAL VOC 2012 dataset, so apply appropriate preprocessing to ensure efficient and accurate batching during training.

For the training and testing phases, you can ask the LLM to produce a template training loop that logs both loss and mIoU values. Review its handling of gradient updates, optimizer steps, and validation metrics carefully. Adapting and verifying this loop against a small subset of data is a quick way to catch issues early.

2.4 Step Four – Knowledge Distillation with a Teacher-Student Model

Distillation was a term that commonly refers to separating components of a liquid mixture. Hinton et al. introduced the term into Machine Learning, to allow a smaller network to learn from a pretrained larger network. It is also commonly known as a "Teacher-Student Model". The standard response-based knowledge distillation steps are as follows:

1. Forward a batch of images into the student model (i.e. the model you implemented in Step 2.2)

2. Forward a batch of images into the teacher model (i.e. the pre-trained FCN ResNet50 in inference mode, with frozen trainable weights)

3. Calculate the distillation loss based on the output of the student model z_s, the teacher model z_t, and groundtruth label y:

   $$\mathcal{L}(x; W) = \alpha * \mathcal{H}(y, \sigma(z_s; T = 1)) + \beta * \mathcal{H}(\sigma(z_t; T = \tau), \sigma(z_s, T = \tau))$$

   where x is the input, W is the trainable weight, α, β, τ are scalars, $\sigma(z; T)$ is the (modified, Hinton-style) softmax function, and $\mathcal{H}(\cdot,\cdot)$ is the cross-entropy loss.

4. Backpropagation of the joint loss

In addition to the response-based method, also implement the feature-based method (comparing intermediate feature maps) by using the cosine loss introduced in the lecture.

During knowledge distillation, an LLM can help explain or illustrate the difference between response-based and feature-based distillation losses. Ask it to outline the mathematical form and suggest a PyTorch implementation for each. Compare those ideas with the lecture material before finalizing your code to ensure consistency with the intended formulas.

3 Deliverables

The deliverables as described below comprise the following:

- The completed code with your trained model;
- A complete record of your LLM session(s);
- A report.

3.1 Completed Code

In addition to the code itself, provide the following two scripts to train and test your code (from the PyCharm terminal):

```
train.txt test.txt
```

3.2 LLM Sessions

Share the link to your conversation with the LLM of your choice (ChatGPT, Claude, etc.). If you used an IDE-embedded AI-assist tool, then you can share the link to your Git repository.

3.3 Report

You report should be relatively brief (4-6 pages), and include the following information:

1. What is your network architecture?
   You should describe the structure in sufficient detail so that it can be reimplemented solely from the textual description. You should also have proper citations to the techniques that you used.

2. How did you utilize the knowledge distillation?
   You should describe thoroughly step by step how your knowledge distillation works. You can use pseudo code or mathematical formulas to help illustrate the process.

3. Describe all hyperparameters used in training. How long did the training take? Include all loss plots.

4. Describe your experiments. Include a description of the hardware that you used, and the time performance for both training and testing (e.g. msec per image). Describe the mIoU statistics, as an accuracy measure. Include a table that shows a comparison of the different knowledge distillation methods, and how they impacted the quantitative results, e.g.:

Knowledge distillation	mIoU	# Parameters	Inference Speed
Without			
Response-based			
Feature-based			

   Lastly, include some qualitative results (i.e. images of segmentation masks overlayed) to support your claim. These can include both successful results, and some failure cases.

5. Discuss the performance of your system. Was it as expected? Discuss which (if any) of the knowledge distillation methods were beneficial. What challenges did you experience, and how did you overcome them?

6. Discuss your conversation with the LLM of your choice (ChatGPT, Claude, etc.). How did you make sure that the code or concepts generated by the LLM were not hallucinations?

4. Submission

The submission should include all your source code, and the report described in Section 3.

All deliverables should be compressed into a single `<zzz>.zip file,` where filename **<zzz>** is replaced with your student number. If you are in a team of 2, then concatenate both student numbers, separated by an underscore. The zipped directory should include your code and scripts, your trained parameter files, and all output plots and images.

The report should include a title (e.g. minimally ELEC 475 Project), your name and student number. If you are working in a team of two, then include the information for both partners in the report (but only make one submission in OnQ).

Your code should execute by entering the syntax provided in your two included scripts (i.e. `train.txt and test.txt)` on the PyCharm terminal.

The marking rubric is as follows:

Item	mark
Step 1	1
Step 2	1
Step 3	1
Step 4	2
Report	2.5
Correct submission format	0.5
Total:	**8**