Assignment 1: Computer Vision

María Alejandra Bravo and Yassine Marrakchi

Due 14.06.2022

1 Introduction

The codebase can be found at `https://lmb.informatik.uni-freiburg.de/lectures/dl_ss22/cv_assignment.zip`. The assignment is divided into 2 parts, which you will need to complete and run according to the instructions below. To complete the assignment you need to submit:

- A short (2-3 pages) report containing your results and analysis according to the instructions for each task.
- A copy of the (working) code you used to obtain those results.

In both experiments you will work with subsets of the COCO dataset[1], in particular with the segmentation annotations. Before starting, make sure to set up the working environment as described in the `README`.

2 Task 1. Self-supervised Pretraining

The goal of the first task is to train a feature representation using a self-supervised learning method (DINO) described in the lecture, and to use the learned representation to train a downstream network for semantic segmentation.

- Download and extract the dataset at:
  `https://lmb.informatik.uni-freiburg.de/lectures/`
  `dl_ss22/unlabelled_dataset.zip`
  which contains cropped instances of objects belonging to 5 different COCO classes. (Be aware of the "_" when copying the path) To extract the data use the password: "`dl_lab`".
- Since the teacher is not trained with backpropagation, disable the gradient computer for its parameters.

[1]http://cocodataset.org

- Implement the loss initialization.
- Implement the forward pass and the loss computation.
- Implement the exponential moving average update for the teacher.
- Implement the T-SNE embedding of feature maps to visualize the learned features
- Implement the centering and sharpening steps for the teacher output.
- Implement the batch center computation that is required to the exponential moving average update of the center
- Report the best validation results in terms of loss. Include plots of the loss for both training and validation[2].
- Check and comment the quality of the learned feature maps by comparing the t-SNE embeddings over the epochs.
- Complete the script `nearest_neighbors.py` to run NNs analysis as discussed in the lecture. Do you think that the network has learned meaningful feature representations? Motivate by including qualitative examples in the report.

3 Task 2. Attention-based Semantic Segmentation

The objective of the second task is to train and compare an attention based method with a classical pixel-wise classification method for semantic segmentation. Here you will have to deal with attention mechanism to get an intuition on how they work and where to apply them. To complete this task you will have to follow some steps. Record your loss and mIoU score for all training processes. If you like you can use tensorboard to plot the scores while training and to visualize examples.

3.0.1 Classical pixel-wise semantic segmentation

We will first train a classical method for semantic segmentation.

- Download and extract the segmentation dataset at:
  `https://lmb.informatik.uni-freiburg.de/lectures/dl_ss22/segmentation_dataset.zip`.

[2]NOTE: to log results and images you can use the `tensorboard` tool. You can install it as a python package in your virtual environment using `pip install tensorboard`, and use the class `SummaryWriter` from `torch.utils.tensorboard` to log the results. The logs can be visualized using a web browser, after having launched the tensorboard server with `python3 -m tensorboard.main --logdir=<path to log>`. More info at `https://pytorch.org/docs/stable/tensorboard.html`.

It contains images from the COCO dataset, and annotations for the segmentation masks.

- Download the weights initialization at:
  `https://lmb.informatik.uni-freiburg.de/lectures/dl_ss22/binary_segmentation.pth`.
  It contains a set of parameters that you can use to start your training from, so you don't have to run the networks for an excessively long time.
- Run the script `dt_multiclass_ss.py`. It contains the code for training and testing a network for semantic segmentation. The decoder network used is from the `DeepLabV3` architecture, it uses dilated convolutions to increase the decoder's receptive field. For the exercise we will only consider 5 classes: person, car, bicycle, cat, dog.
- Observe how the network trains, you will compare this performance with one with attention. Train the segmentation network starting from the best network snapshot from the pre-training task. The network should reach around 25% mIoU on validation by the 10th epoch.

3.0.2 Attention-based semantic segmentation

Now that you have run the classical way, we will train a semantic segmentation method conditioned to the class using the attention mechanism.

- Implement the `Scaled Dot-Product Based` attention mechanism by completing the corresponding function in `models/attention_layer.py`. Make sure it works by running the same script.
- Add the attention layer in your model so that when adding the class as a hot encoded vector your output segmentation corresponds to this class. For this, complete the script `models/att_segmentation.py`. Tips: place the attention layer at a point in the network where the semantic information is higher. Check that the dimensions correspond when calculating the attention. Follow the comments made on the file.
- Train the model with the attention mechanism by running `dt_single_ss.py`. During training, save the loss and mIoU for both sets, training and validation, and plot the scores. The network should reach around 20% mIoU on validation by the 10th epoch.
- Visualize the attention maps of the best model and compare to the predictions and labels. Draw conclusions about what the attention mechanism is doing.
- Compare the classical method of Section 3.0.1 with the Attention-based semantic segmentation method by running all the validation images in the dataset and calculating its mIoU. Compare quantitatively and qualitatively both models. Explain your results and show examples of the segmentations.