CSC 375 Homework 3

TAs - Maria and Yun-Chun (csc375-staff@cs.toronto.edu)

March 20, 2022

Starter code for this assignment is available on Colab at `https://colab.research.google.com/github/pairlab/csc375-w22-assignments/blob/main/HW3/hw3.ipynb`. Please make a copy first. Kindly post any questions about the assignment on Piazza.

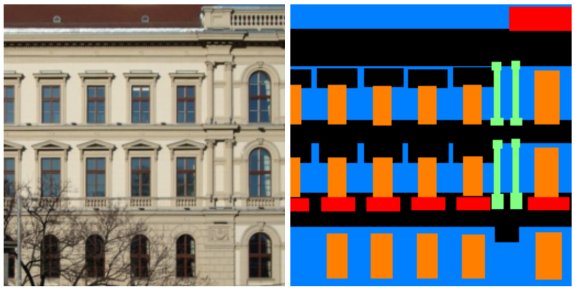
The instructions for each problem are summarized in this PDF file. Please read through them. Your submission will consist of a single merged PDF. In your PDF file, please include a link to your Colab. You will submit through Quercus.

Problem 1.

Semantic Segmentation (35 points)

Convolutional Neural Networks can generate dense predictions. A popular application is semantic segmentation. In this part, you will design and implement your Convolutional Neural Networks to perform semantic segmentation on the Mini Facade dataset.

Mini Facade dataset consists of images of different cities around the world and diverse architectural styles (in .jpg format), shown as the image on the left. It also contains semantic segmentation labels (in .png format) in 5 different classes: balcony, window, pillar, facade and others. Your task is to train a network to convert image on the left to the labels on the right.



Note: The label images are plotted using 8-bit indexed color with pixel value listed in Table 1. We have provided you a visualization function using a "jet" color map.

class	colour	pixel value
others	black	0
facade	blue	1
pillar	green	2
window	orange	3
balcony	red	4

Table 1: Label image consists of 5 classes, represented in [0, 4].

We have provided some starter code in the notebook which contains a dummy network. It uses a 1x1 convolution to convert 3 channels (RGB) to 5 channels, i.e. 5 heatmaps for each class, with cross-entropy loss. Your network should give an output of the same shape, but contains more layers of different types.

We will evaluate your model on its average precision (AP) on the test set (higher the better). We have provided you the code to evaluate AP and you can directly use it. An introduction to AP is available at `https://scikit-learn.org/stable/modules/model_evaluation.html#precision-recall-f-measure-metrics`.

Submission for this question. Your submission for this question will have three parts.

1. A screenshot of the code which contains your best combination of self.base module, optimizer and training parameters.
2. A brief report (a few sentences) detailing the architecture of your best model. Include information on hyperparameters chosen for training.
3. Report the average precision on the test set. You can use provided function to calculate AP on the test set. You should only evaluate your model on the test set once. All hyperparameter tuning should be done on the validation set.

Hints. You should still choose layers from classes under torch.nn. In addition to layers mentioned in the previous part, you might want to try nn.Upsample that upsamples the activation map by a simple interpolation, and nn.ConvTranspose2d that upsamples by performing transpose convolution.

- Jonathan Long, Evan Shelhamer, Trevor Darrell. *Fully Convolutional Networks for Semantic Segmentation*. CVPR 2015.
- Olaf Ronneberger, Philipp Fischer, Thomas Brox. *U-Net: Convolutional Networks for Biomedical Image Segmentation*. MICCAI 2015.
- Alejandro Newell, Kaiyu Yang, Jia Deng. *Stacked Hourglass Networks for Human Pose Estimation*. ECCV 2016.

You will easily get a high test AP if you use one of these models, but they can take a long time to train (hours with a CPU). Consider doing this part with the help of a GPU on Colab.

Problem 2.

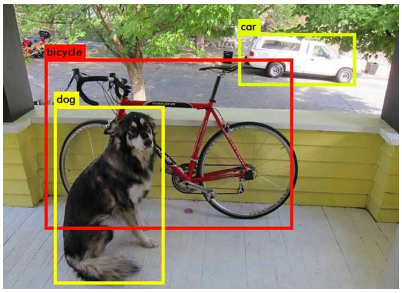
Object Detection (35 points)

In this part, we will perform object detection by fine-tuning a pre-trained model. Object detection is a task of detecting instances of semantic objects of a certain class in images (see image below). We will use the YOLO approach [a]. YOLO uses a single neural network to predict bounding boxes (4 coordinates describing the corners of the box bounding a particular object) and class probabilities (what object is in the bounding box) based on a single pass over an image. It first divides the image into a grid, and for each grid cell predicts bounding boxes, confidence for those boxes, and conditional class probabilities. Here, we take a pre-trained model, YOLOv3 [b] and fine-tune it to perform detection on the COCO dataset. You are encouraged to read through [a] and [b] to have a better understanding of how these two methods work.

Step 1: Fine-tuning from pre-trained models. A common practice in computer vision tasks is to take a pre-trained model trained on a large datset and finetune only parts of the model for a specific usecase. This can be helpful, for example, for preventing overfitting if the dataset we fine-tune on is small. To keep track of which weights we want to update, we use the PyTorch utility `Model.named_parameters()`[1], which returns an iterator over all the weight matrices of the model. Complete the model parameter freezing part in `train.py` by adding 3-4 lines of code where indicated. See also the colab for further instruction.

Step 2: Visualize the predictions. Visualize the predictions on 2 sample images by running the helper code provided in the colab.

[1]See examples at `https://pytorch.org/docs/stable/nn.html`



Submission for this question. Your submission for this question will have three parts.

1. A screenshot of code for freezing layers in `train.py` (step 1).
2. A screenshot of the visualization of predictions (step 2).

[a] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. *You Only Look Once: Unified, Real-Time Object Detection.*

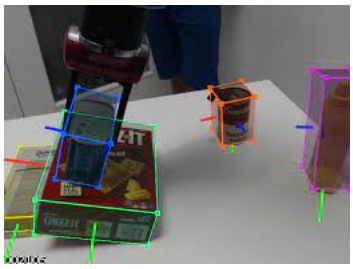
[b] Joseph Redmon, Ali Farhadi. *YOLOv3: An Incremental Improvement.*

Problem 3.

Object Pose Estimation (30 points)

In this part, we will train a model for object pose estimation. Object pose estimation is a task of detecting the 6D pose of an object, which includes its location and orientation (see image below). We will use a dataset collected in pybullet, where each image contains a single object and the ground truth object pose is provided. We will design and implement a convolutional neural network to perform object pose estimation on this dataset.

Note. Please search for **TODO** in colab. Those are the places where you will be implementing.



Submission for this question. Your submission for this question will have three parts.

1. A screenshot of code, which contains your best model architecture.
2. A brief report (a few sentences) detailing the architecture of your best model. Include information on hyperparameters chosen for training.
3. Report the best loss on the test set.