CS 558: Homework Assignment 5
Due: December 13 (Friday), 6:00pm

Philippos Mordohai
Department of Computer Science
Stevens Institute of Technology
pmordoha@stevens.edu

Collaboration Policy. Homeworks will be done individually: each student must hand in their own answers. It is acceptable for students to collaborate in understanding the material but not in solving the problems. Use of the Internet is allowed, but should not include searching for previous solutions or answers to the specific questions of the assignment. I will assume that, as participants in a graduate course, you will be taking the responsibility of making sure that you personally understand the solution to any work arising from collaboration.

Late Policy. A total of **3 late-days** will be allowed in the **entire semester** without penalty. Days will be counted by rounding up, e.g. being 5 minutes late will be counted as one of the late-days. Assignments will no longer be accepted after all late-days have been used.

Deliverables. In a zip file include:

1. The complete **Jupyter notebook** with all cells executed. See below for details. **Make sure that the filename start with your Stevens username.**
2. The notebook in **pdf format**.
3. A **report in pdf format** explaining what you did and answering the questions below.

Please read the following list carefully:

- The objectives of this assignment include exposing you to **real-world challenges in training neural networks**. The images in the dataset are of different sizes, their sizes are not powers of 2, the classes are unbalanced, etc.
- You should not change the provided function specifications, but you can add your own functions as needed. Look for **TODO**s in the notebook.
- You are not allowed to use **any** library functions that generate network layers or train the network other than the ones in the pytorch tutorial we did in class and is listed under Modules in Canvas. Loading externally constructed network architectures is strictly prohibited. **If you are not certain, please email me.**
- If you develop using a different Python setup, you still have to submit a notebook, which must run on Google Colab.

Problem 1: Semantic Segmentation (100 points)

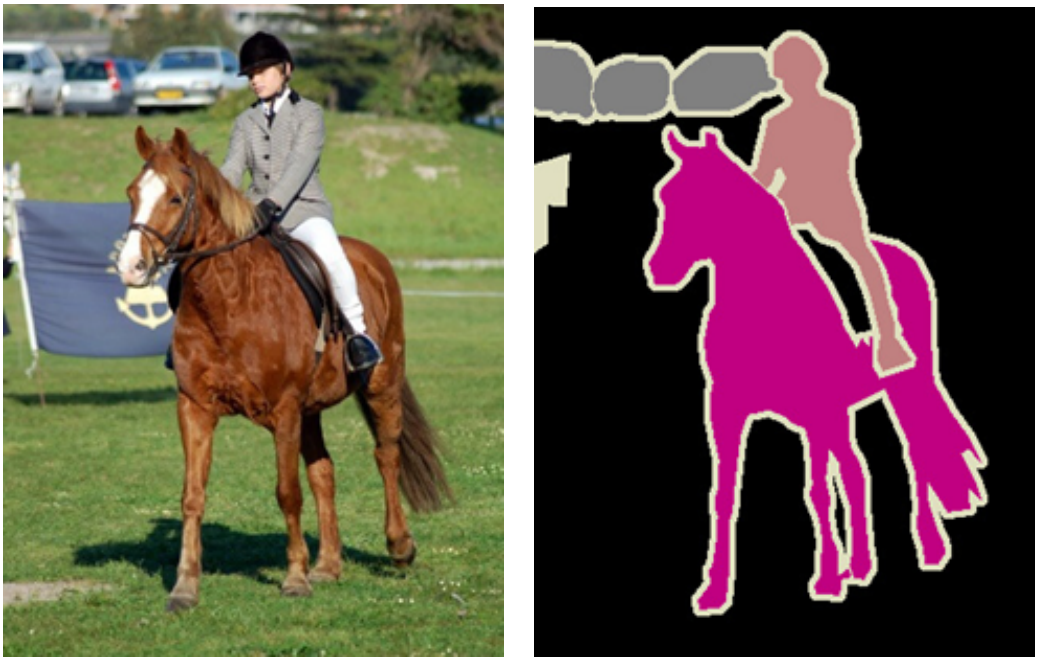
Input: `cs558_hw5.ipynb`

The objective of this problem is to perform semantic segmentation on the PASCAL VOC dataset (2007 version). See `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html` for an overview and `http://host.robots.ox.ac.uk/pascal/VOC/voc2007/htmldoc/voc.html#SECTION00060000000000000000` for details of the dataset. We are interested in the **segmentation task.** See Table 2 in the second URL for the distribution of images in the training and validation sets. (We will not use the test set.) Figure 1 shows an example from the dataset.

There are 20 labels corresponding to the types of objects in the table in alphabetical order. In addition, **label 0** indicates the background and **label 255** corresponds to boundaries which should not be considered in the evaluation. We will modify the problem so that there are two possible labels:

- **0:** corresponding to the original 0 in the VOC dataset, and
- **1:** encompassing all other labels, including 255.

We will use `torchvision` for data management. Links to all datasets, including VOCSegmentation can be found at `https://pytorch.org/vision/main/datasets.html`.



(a) Image (b) Ground truth labels

Figure 1: An example from the PASCAL VOC dataset. Notice that, besides the person and the horse, the cars in the background are also labeled.

Network Architecture

1. Ideally, you should specify a U-Net architecture detailed below.
2. If the above is impossible, use a fully-convolutional architecture with the same number of layers and set the stride always equal to 1. This will not receive full credit, but will allow you to complete the assignment.
3. If you face GPU availability issues, it is possible to train the network on the CPU, by commenting out all lines that transfer the network and data to the GPU. Consider an even smaller network if this happens.
4. The network should produce two outputs per pixel, since there are two possible labels.
5. You should be able to achieve accuracy in the order of 70% pretty easily, but see also below.

Architecture

1. The encoder has three blocks, operating on images with resolution: $H \times W$, $\frac{H}{2} \times \frac{W}{2}$ and $\frac{H}{4} \times \frac{W}{4}$. Each block consists of a few convolutional layers.
2. The decoder has three blocks operating on images with increasing resolutions, from $\frac{H}{4} \times \frac{W}{4}$ to $\frac{H}{2} \times \frac{W}{2}$ to $H \times W$.
3. The number of channels should be 64 at the highest resolution, 128 at the middle resolution and 256 at the lowest resolution.
4. **Each convolutional layer, except the very last one, should be followed by a ReLU nonlinearity** and no normalization. This is not shown in the description below to reduce clutter.
5. There should be two residual connections as shown in Figure 2.
6. Cross entropy loss should be used for training. The necessary tensor operations to facilitate it have been provided.

Network Specification

```
# Encoder block 1 - H*W resolution
kernel_size=3, in_channels=3, out_channels=c, padding=1
kernel_size=3, in_channels=c, out_channels=c, padding=1
Output: z0 [64, H, W]

# Encoder block 2 - (H/2)*(W/2) resolution
kernel_size=5, in_channels=c, out_channels=c*2, padding=2, stride=2
kernel_size=3, in_channels=c*2, out_channels=c*2, padding=1
```

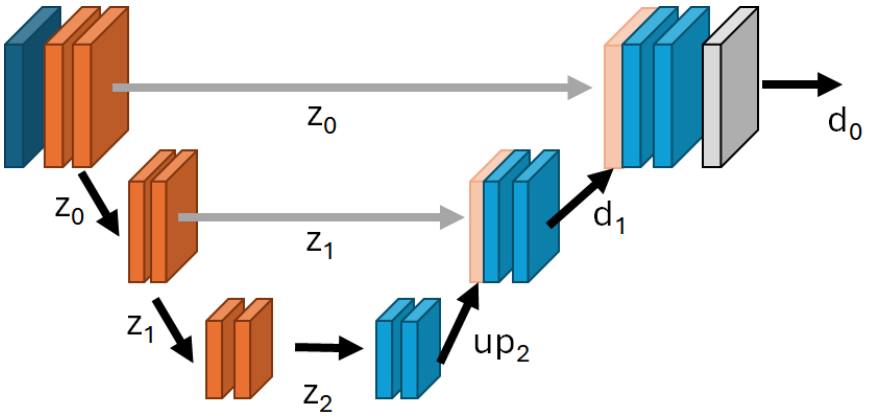


Figure 2: Illustration of the U-Net architecture. The precise details are in pseudo-code.

```
kernel_size=3, in_channels=c*2, out_channels=c*2, padding=1
Output: z1 [128, H/2, W/2]

# Encoder block 3 - (H/4)*(W/4) resolution
kernel_size=5, in_channels=c*2, out_channels=c*4, padding=2, stride=2
kernel_size=3, in_channels=c*4, out_channels=c*4, padding=1
kernel_size=3, in_channels=c*4, out_channels=c*4, padding=1
Output: z2 [256, H/4, W/4]

# Decoder
up2 = F.interpolate(z2, scale_factor=2, mode="bilinear")
c2 = kernel_size=3, in_channels=c*2, out_channels=c*4, padding=1,
        input=z1
d1 = kernel_size=1, in_channels=c*4, out_channels=c*2, padding=0,
        input = up2 + c2
Output: d1 [128, H/2, W/2]

up1 = F.interpolate(d1, scale_factor=2, mode="bilinear")
c1 = kernel_size=3, in_channels=c, out_channels=c*2, padding=1,
        input=z0
d0 = kernel_size=1, in_channels=c*2, out_channels=2, padding=0,
        nonlinearity="none", input = up1 + c1
Output: d0 [2, H, W]
```

The relevant documentation pages are the following:

- `https://pytorch.org/docs/stable/generated/torch.nn.functional.conv2d.html`

- `https://pytorch.org/docs/stable/generated/torch.nn.functional.relu.html`
- `https://pytorch.org/docs/stable/generated/torch.nn.functional.interpolate.html`
- `https://pytorch.org/docs/stable/generated/torch.nn.functional.cross_entropy.html`

Be mindful of:

1. There are TODO annotations in the starter code that require your attention, since the code will fail as is.
2. The dataset contains images of different sizes, therefore they cannot be stacked in a tensor. Your batch size must always be set to 1.
3. You must manipulate the labels before they are used in the training and evaluation functions. 0 should map to 0, while all other labels should be mapped to 1.
4. Upsampling certain tensors by a factor of 2, and adding them to tensors that have not been downsampled as much, may lead to resolution mismatch. You are allowed and encouraged to crop (trim) the input images as needed to guarantee that these steps succeed.

Report - Part 1

Describe how you addressed items 3 and 4 above in your implementation. You may paste relevant code, if it helps.

Performance Analysis

Modify the evaluation function so that it collects the data needed for the confusion matrix. After training the network for a few epochs, around 10 should be easily sufficient, collect the data, make the confusion matrix and include it in your report. It should look like the following:

	True 0	True 1
Pred. 0	40	20
Pred. 1	10	30

According to this example, 40 of the 100 exemplars in the dataset have true label equal to 0 and also predicted label equal to 0. Therefore, they were classified correctly. 30 more exemplars were classified correctly as having label 1, while 20 had true label 1 but were misclassified. The total accuracy was $(40 + 30)/(40 + 20 + 10 + 30) = 70\%$.

Your confusion matrix should look similar but there is a much larger total number of pixels. In addition, to raw numbers, show also percentages by diving the four cells by the total number of pixels.

Report - Part 2

Also, in your report:

1. Include the confusion matrix as detailed above.
2. Describe the differences between the two labels.
3. Discuss whether you are satisfied with the overall accuracy of your network.
4. In either case, justify the above answer. In retrospect, are you surprised by what happened?