Assignment 4: Segmentation and Object Detection

Due date: TBA

Assignment by Anastasia Razdaibiedina

Submission: please submit a completed Google Colab notebook through Classroom. We provide you with a starter notebook that has code sections that you should fill in, and written questions that you should answer.

Google Colab notebook: https://drive.google.com/file/d/1T-ylxRFeeF1EbwcEOWPWvHDZpQxf-Zkc/view?usp=sharing

The Google Colab notebook contains all the detailed instructions, starter code and questions. **You should create a copy of the starter Google Colab notebook and finish all the tasks there**.

The assignments are individual work. You should attempt to answer all questions for this assignment. Most of the tasks can be completed at least partially even if you could not finish earlier questions. If you were unable to run the experiments, please discuss what outcomes you might hypothetically expect from the experiments. If you have any questions during completion of this assignment, please don't hesitate to contact TAs for help.

Introduction

This assignment consists of two parts - segmentation and object detection, these tasks have become popular computer vision applications in recent years. You will train a semantic segmentation model from scratch on a custom microscopy image dataset, and will experiment with pre-trained models for object detection from TensorFlow Hub. Before you start, let's briefly recap the main differences between object detection, semantic segmentation and instance segmentation (Fig. 1).

(1) **Object detection**

- Task: detect objects using rectangular bounding boxes and assign class probabilities;
- Instead of delineating a contour of every single object (like in instance segmentation algorithms), object detection algorithms draw a bounding box for every object;
- Example application: find different objects on the road for a self-driving car software;
- Algorithms: R-CNN, Fast R-CNN, YOLO [2, 1, 5].

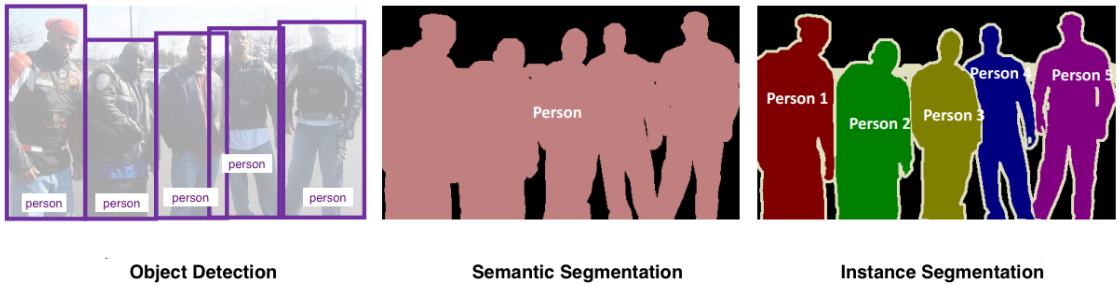

FIGURE 1. Differences between object detection, semantic segmentation and instance segmentation.

(2) **Semantic segmentation**
- Task: detect and delineate classes of objects (by assign labels to every pixel in the image);
- Multiple objects from the same class are considered a single entity;
- Example application: separate people from background on a photo;
- Algorithms: U-Net [6], FC-DenseNet103 [4].

(3) **Instance segmentation**
- Task: detect and delineate every single object of interest that appears on the image;
- Multiple objects from the same class are separated;
- Example application: delineate every healthy and cancerous cell on the input microscopy image;
- Algorithms: Mask R-CNN [3].

In your Google Colab notebook you will complete two tasks and learn about image semantic segmentation and object detection in practice.

Task 1: Segmentation

You will first learn how to perform image segmentation. The goal of **segmentation** is to **separate regions on the image corresponding to different objects and surfaces from each other**.

In this task you will focus on **semantic segmentation**, since our goal is separate cells from the background. You will work with the **dataset from Kaggle Competition** (Data Science Bowl 2018). The dataset contains pairs of microscopy images of cells of different types and their corresponding binary masks.

You will build and train from scratch **U-Net network**, which is one of the most popular and easy-to-implement semantic segmentation models (Fig. 2). The **idea behind U-Net** is simple: it consists of **contractive** and **expansive paths** (hence famous U-shape) [6]:

U-Net model overview:

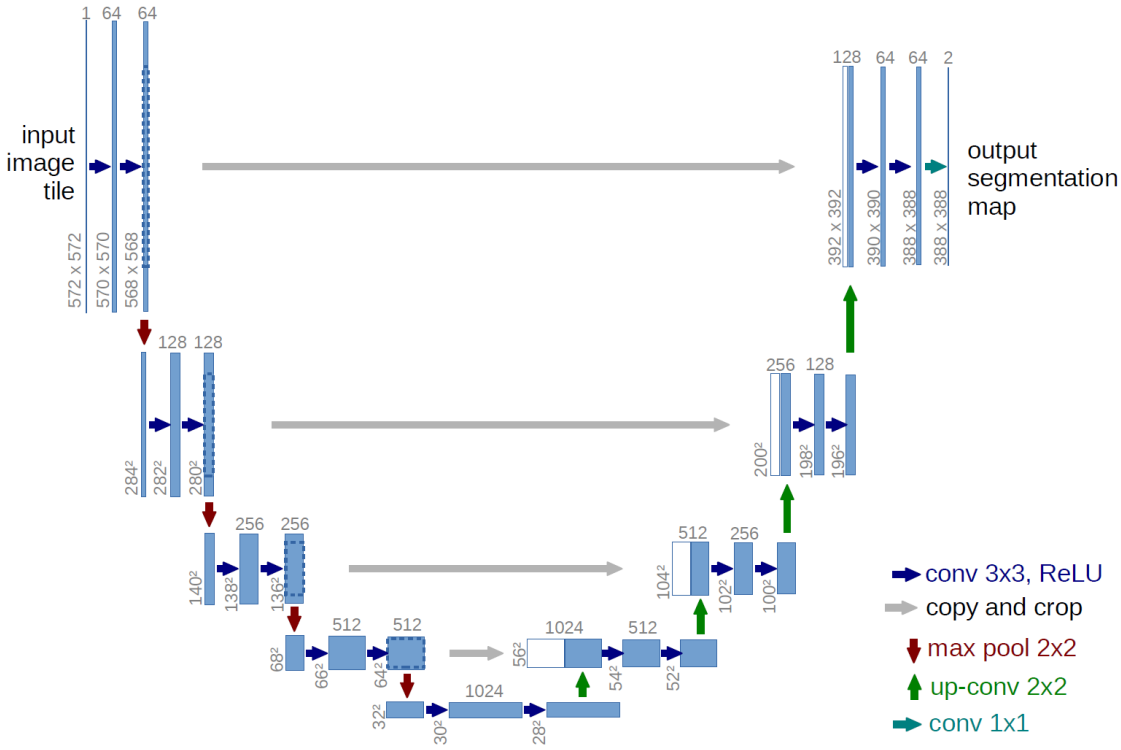

FIGURE 2. U-Net model architecture.

(1) First we perform contraction of the input image to 32x32 pixels, increasing its depth to 448 channels.
(2) Afterwards the image is expanded back into its original shape while lowering number of its channels.
(3) Important detail is stacking of the intermediate layers: you can see how layers of the same sizes are stacked together (grey arrows). This improves training by making the backpropagation happen faster along the shorter paths.

In the Google Colab notebook you have starter code for U-Net model and instructions on how to perform training. You can see segmentation results on test data in Fig. 3, you should achieve similar visual results after training.

Task 1 consists of the following exercises:

(1) Task 1a (code): complete one block of the U-Net model architecture that is commented out. You will use a Keras-style sequential model definition.
(2) Task 1b (theory): draw an illustration for the model you created in Task 1a using the code and *model summary* function. Use similar illustration style as U-Net model shown in Fig. 2. Attach your illustration in the assigned empty text cell in your Google Colab (you can draw it and attach a photo).
(3) Task 1c (code): after training is completed, run U-Net inference and plot illustrations on the test data.
(4) Task 1d (code): plot train and test loss changes during training.

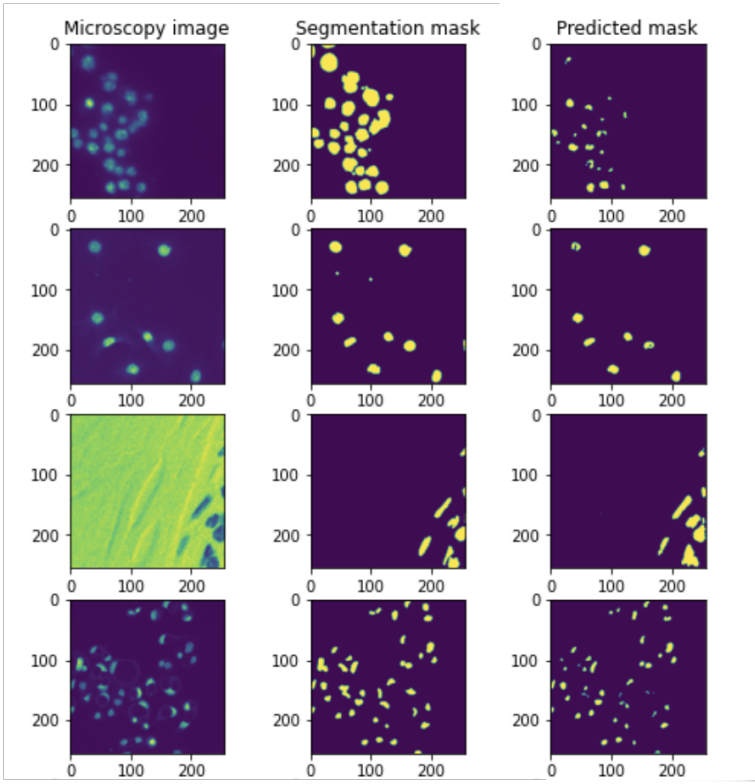

FIGURE 3. Original data, segmentation masks and predicted masks. All the shown images are from test data.

(5) Task 1e (theory): what can you say about the model looking at its accuracy and loss throughout training? Is it overfitting / underfitting? Does it need more or less training time? Write your answer in cell provided in the notebook by double-clicking on it (2-3 sentences is good).

Task 2: Object Detection

In this task we will focus on **object detection**. To avoid complications related to long training time and big dataset size, we will work with a pre-trained model. We will use TensorFlow Hub, a repositoty of trained machine learning models, to load the pre-trained model and use it for object detection.

All the starter code is provided in Google Colab. You will try to load models with two different architectures and perform inference on various RGB images, some of which you will select yourself. All the utility functions and starter code template are provided in the notebook. In task 2 you have to complete the following exercises:

(1) Task 2a (code): perform inference on the "New York" image provided in your notebook using InceptionResNet-v2 from TF Hub (link in Google Colab).

(2) Task 2b (code): select three more images available online and run object detection on them using the pipeline you just learned.

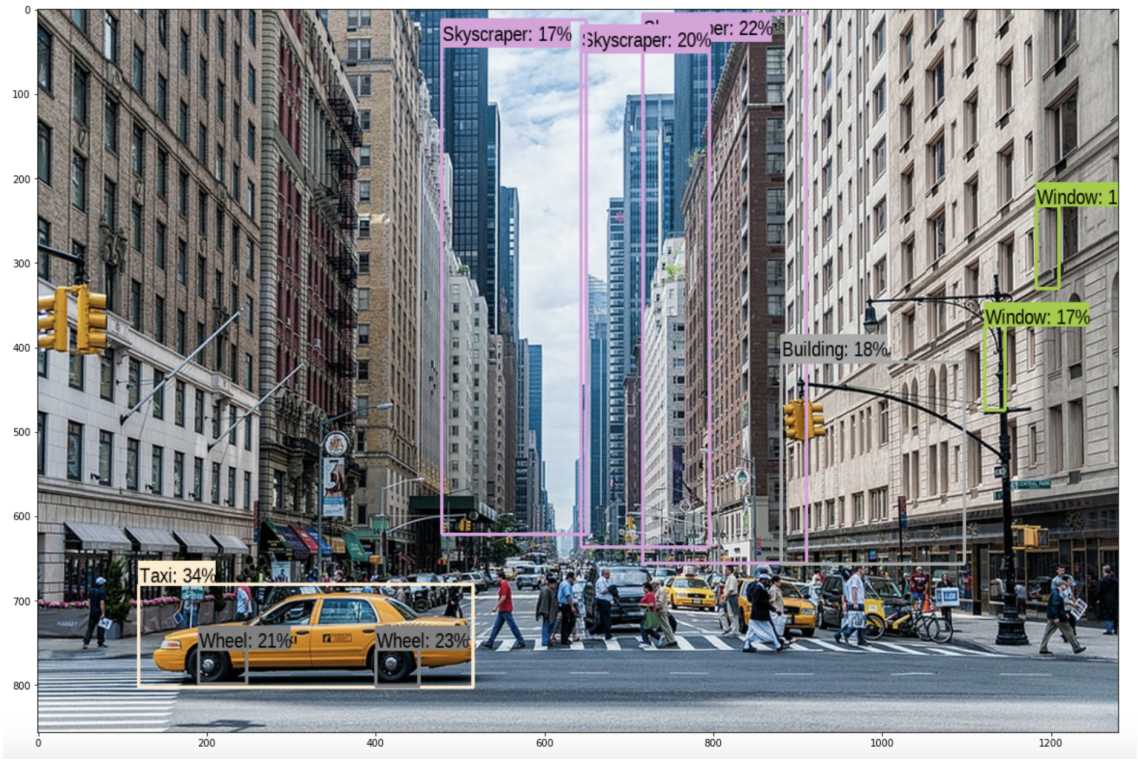

FIGURE 4. Results of running object detection algorithm with MobileNet-based architecture on "New York" image provided in the starter code.

REFERENCES

[1] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[2] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2015.

[3] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[4] Simon Jégou, Michal Drozdzal, David Vazquez, Adriana Romero, and Yoshua Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 11–19, 2017.

[5] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[6] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.