LUT Engineering Science
Data-Centric Engineering
Tuomas Eerola, Joona Kareinen, Samuel Repka
Spring term 2025

BM40A0902 3D Computer Vision
Exercise 6
Object detection.
Deadline on 04.03.2025

1. Detection by color thresholding (1 point).

   In this task you will apply thresholding for segmenting color images.

   Download the color block image (blocks-col.png) and the color circle image (circles.png) to be used in the experiments, see Figure 1.

   Use thresholding to locate the red block in the image. Locate the red block both in RGB and HSI/HSV color spaces. You need to determine suitable thresholds, such that the red block is found, but most of the background is not detected.

   Next, apply the same thresholds to the circles image. Is the red circle properly segmented? Are the same thresholds OK for this task? Which color space (RGB vs. HSI/HSV) gives better results? Which other problems have you encountered with the segmentation?

   Note: Your segmentation does not need to be perfect, some noise is inevitable. As long as the majority of the cube pixels are properly segmented and the majority of background pixels are detected as background.

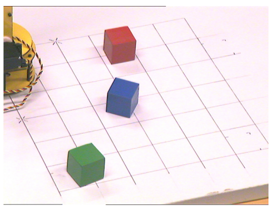
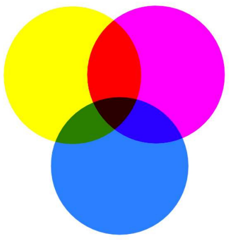

Figure 1: Images for task 1.

2. Object detection with YOLOv1 (4 points).

   One of the necessary skills for people who want to be successful in life is the ability to detect rabbits in the image, as shown in Figure 2. In this task, you will teach a neural network how to do that, so it can too be successful in life.

   For this task only Python template and codes are provided and only Python submissions are accepted.

   (a) Download the required files for this task from this Github repository.
   (b) Unzip the files and explore the code. Try to understand what the code is doing.

   In the template, most of the components are already set up, such as dataset loading, preprocessing, training, and validation.

Your task is to implement the detection part of the model and its forward pass. You will use a pre-trained feature extractor like AlexNet or ResNet, which reduces the required training time significantly. If you have never worked with PyTorch before, we recommend checking some of their tutorials, especially this one. If you encounter any issues, feel free to ask for help on Discord.

Note: The implemented model does not have to be large, and you should be able to train it in a reasonable time on a CPU. If you want to use a GPU, you can use your own or use for example Google Colab or Kaggle.

(c) Run the code until it breaks (at *1.5 Training the model*).

(d) Implement DetectionNet and its forward pass, as well as the forward pass for the main model (Fill all TODOs in the code). Design an architecture that works well but is not perfect, as some detection flaws are expected.

(e) Answer the following questions:

- Does the code apply further augmentation to the training dataset?
- What does the dataset return as a target? Explain the shape.
- How many parameters does your DetectionNet have?
- What is the shape of the output from the forward pass? Explain what each dimension represents.
- What is the average Intersection-over-Union (IoU) of the test set? Are there test images where IoU is poor? Why?
- How many epochs do you need to train to obtain good performance?
- Does the choice of backbone affect the performance? Train the network with one backbone, then try another, and compare results.
- How does grid size affect performance? In which types of tasks do we prefer smaller grid boxes, and when are larger grid boxes more suitable?
- Try to understand the loss function. Could it be improved? If so, how? Use your own head.

NOTE: We want to see what your answers are based on. If we ask how many parameters the network has, have a code that computes it. If you are asked about potential poor-performing images, show which. You do not have to create a separate document with answers (you may though), but make sure everything is shown and *explained* in the code, comments or Markdown cells. Mind that for this submission, you have much more time than in previous exercises.

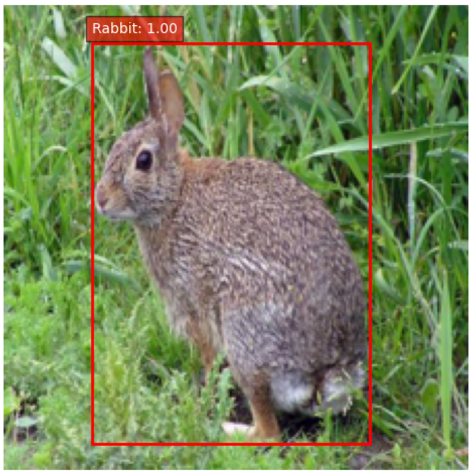

Figure 2: Rabbit detection example