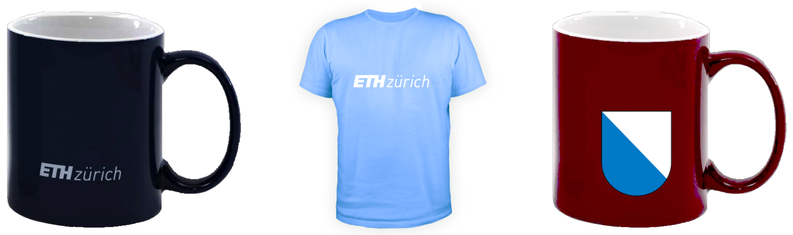
SML Project 2

SML 2024

1 Introduction

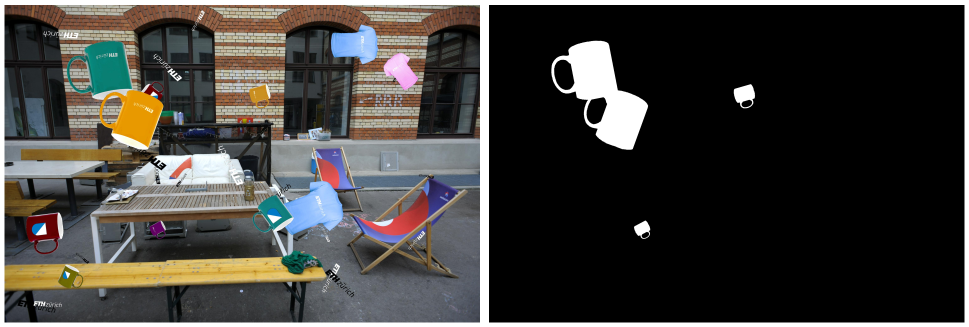
The tranquil campus of ETH Zurich has been hit by a storm, causing a sudden disappearance of cherished ETH and Zurich-themed merchandise. These items have seemingly got scattered around various locations at ETH. However, there are some customers who are waiting for their merchandise to be delivered. Among the lost items, there are some coffee mugs with the ETH logo (Fig. 1a), some t-shirts with the ETH logo (Fig. 1b), and some other coffee mugs with the Zurich cantonal flag (Fig. 1c). The most urgent task is the delivery of the **ETH mugs** - and your task is to find and identify the ETH mugs in images taken at ETH.



(a) ETH Mug (b) ETH T-shirt (c) Zurich Mug

Figure 1: Collection of ETH and Zurich-themed merchandise

Now that you learned the basics of deep learning, you can help us by training an image segmentation model that aims to identify the ETH mugs in a given picture. More specifically, the task we are targeting here is binary image segmentation. Given an RGB input image (Fig. 2a), the output should be a binary mask of the same dimensions (Fig. 2b), in which each image pixel that belongs to an ETH mug is assigned the value 1 in the segmentation mask, and the rest of the pixels are assigned the value 0.



(a) *Input:* RGB image (b) *Output:* Binary segmentation

Figure 2: Task: Given an RGB image as input (left), we want to identify the pixels that correspond to ETH mugs, and return a binary segmentation map (right) as output.

2 Data

The dataset is given in the folder **data** within the student template `.zip` file.

2.1 Data Sets

In the **train_images_378_252** folder, there are images (sub-folder **rgb**) and their corresponding ground truth (GT) segmentation maps (sub-folder **masks**). You can use these images while developing and training your model. In the **public_test_images_378_252** folder, we have a similar data structure, with RGB images and the GT binary segmentation masks necessary for you to evaluate whether your model passes the project baseline.

On the **27th of May, at 8am central European summer time (CEST)** we will release the private test set on Moodle. You will have to run your model on this dataset and submit the predicted masks as obtained from your model. In the **private_test_images_378_252** folder, you will find the RGB input images for testing your method, but not the GT segmentation maps, as we will use those for grading your approach. We recommend you save that file in the same directory as the other two datasets, and also unzip it there. When doing so, you can run the code from the student template to process the private test set without changing any of the paths.

2.2 Encodings

Each RGB image I has the resolution of 378×252, and can be loaded as an array of dimension $378 \times 252 \times 3$. Each GT segmentation mask M_{gt} is an array of dimension 378×252, and the value of each element in the array is in $\{0, 1\}$. The values are assigned in the following way:

$$\mathrm{M}_{\mathrm{gt}}[i,j] = \begin{cases} 0; & \text{if pixel } I[i,j] \text{ does not belong to an ETH mug} \\ 1; & \text{if pixel } I[i,j] \text{ belongs to an ETH mug} \end{cases} \tag{1}$$

3 Implementation

For the implementation of the model we will use the PyTorch deep learning framework in Python. The PyTorch library comes with simplified methods and classes for setting up a dataloader, setting up the model and training the model. You will not need to implement any of the lower level algorithms such as backpropagation. You can also use architectural building blocks such as convolutional layers and fully connected layers directly from PyTorch. Similar to the previous project, you will simply need to use the existing functions and models in a correct way, and adapt them to your data. This could mean modifying existing architectures, adding data augmentation steps, and tuning the parameters of your training pipeline.

3.1 Project Folder

The template we have provided you with comes with a number of files. Here is a short description for each:

`README.md` A file containing further instructions on technical details for setup and executing your code.

`Instructions_GoogleColab.ipynb` A file describing the specifics of running the project on Google colab.

`eth_mugs_dataset.py` In this file, you will need to define your own DataSet class which will load your data. You can also define any preprocessing in this class. We recommend you follow the given structure in this template. Check out the following link for more information: here.

`train.py` In this file, you will need to define your own neural network and design a training loop which makes this network learn how to produce the segmentation masks. This loop saves the parameters of the neural network after each epoch. This is called **checkpointing**. Please see the `README.md` for more information on how to run this file.

`test.py` This file loads your model parameters based on the checkpoints which are done in `train.py`, generates segmentation masks and evaluates these masks. See later sections for more on this. Please see the `README.md` for more information on how to run this file.

`utils.py` This file contains some helper functions.

3.2 Restrictions

You are free to use any network architecture - be it examples and ideas from the lecture, the exercise groups, or anything you find on the internet. However, **you are not allowed to use any pre-trained model. Only solutions from models that you have trained yourself using the provided datasets will be considered valid.**

3.3 Compute Infrastructure

You are free to run the project on any infrastructure. Note that depending on the model you design, JupyterHub might not have enough resources, or be very slow to train your model. Google colab is a free resource that offers a powerful GPU for training and inference. Depending on your what device you have, running the project locally might also be a good option (especially if your computer has a GPU).

4 Tips for solving this project

4.1 Creating Splits

When training models and evaluating different hyperparamters and model architectures, be careful not to do uncontrolled overfitting. It is common practice in machine learning to divide the data in train, validation, and test splits. The train split is used to learn the parameters of the model while the validation split is used to decide on the hyper-parameters of the model. Hyper-parameters are parameters that are not learned from the data during the training process but are set prior to training. They control various aspects of the learning algorithm like the learning rate and the image resizing factor. The test split is the data you use to evaluate your final model.

4.2 Preprocessing data

We now have our data which is divided in train and validation splits. The next step is to preprocess the data you load in the dataloader by applying different types of transformations to them.

1. You might need to consider whether or not you should normalize the color values of your image before passing it as an input to your model. In order for a machine learning model to perform well, it is important that the features are scaled in a common range. Therefore, you might need to use a scaling method.

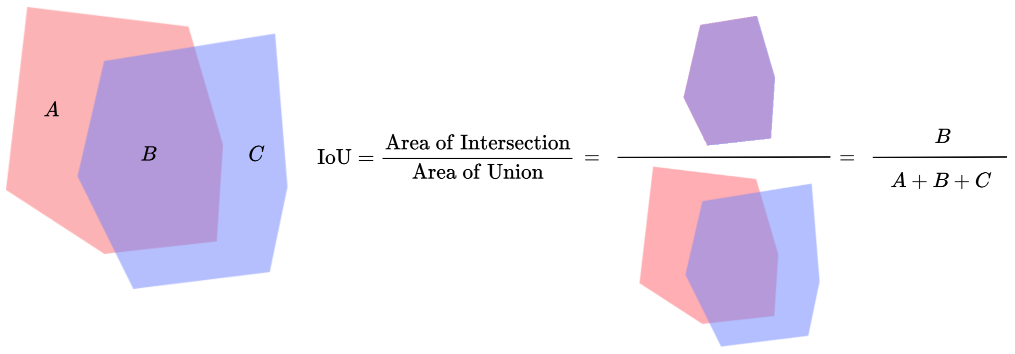


Figure 3: Illustration of the Intersection-over-Union (IoU) metric computation. Red area $(A + B)$ refers to the ground truth mask, and the blue area $(B + C)$ refers to the predicted mask. The area of intersection is the overlapping area, i.e. B.

2. You can consider applying data augmentation functions (you can take a look at this link) to your images. Don't forget that some augmentation functions will need you to apply the same transform to the GT mask as well. For instance, if you rotate or flip the RGB image, the positions of the ETH mugs in the new image will be different than their positions in the original segmentation mask. But slightly changing the color values (e.g. jittering) of the RGB image should not have any affect on the segmentation mask.

5 Evaluating your trained pipeline

Your trained pipeline should now be ready to take an RGB image as input, and predict which parts of the image belongs to an ETH mug. It is now the time to see if the model works well enough. Your model will be evaluated using the Intersection-over-Union (IoU) metric which is defined as the following:

$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}} \tag{1}$$

In Fig. 3, we illustrate the computation of the IoU metric. For our segmentation task, the IoU will be computed between the ground truth segmentation mask and your predicted segmentation task. In that case, pixels that are assigned the value 1 in your predicted mask will be one set, and the pixels that are assigned the value 1 in the ground truth mask will be another set. The **area of intersection** of these two sets is defined as the *number of pixels* (i, j) such that $\text{M}_{\text{gt}}[i, j] = 1$ *and* $\text{M}_{\text{pred}}[i, j] = 1$. The **area of union** of these two sets is defined as the *number of pixels* (i, j) such that $\text{M}_{\text{gt}}[i, j] = 1$ *or* $\text{M}_{\text{pred}}[i, j] = 1$.

To verify that you passed the project, you can use the script `evaluate.py` to evaluate the performance of your model. This script loads the predicted segmentation masks from the folder you saved your predictions, and evaluates the IoU value averaged over all examples. In other words, the script computes the IoU score between the predicted and GT segmentation masks for each frame, and returns the average of these per-frame IoU values.

6 Submitting your predictions

Remember your grade will be calculated on your model's performance on the *private test set*, whose GT segmentation masks will not be shared with you.

6.1 What to Submit

You must compute and submit the segmentation maps for each image in the private test set (at the exact same resolution). We will load your segmentation predictions and evaluate on the private test data with hidden ground truth segmentation masks.

You have to submit the following artifacts:

Predictions: Please submit a binary mask for every image in the private test set. Use the `test.py` script to run your model over the private test set and save the masks in the correct format.

Source code: We need your code for plagiarism checks.

Trained model parameters: Submit the checkpoint file that you use when you run `test.py` over the private test set.

Team names: A text file called `teammates.txt` with one line per team member. Each line contains the legi and the student's name separated by a comma. For example, `11-222-333,Max Muster`.

6.2 How to Submit

Create a `.zip` file that contains the following:

- a folder called **prediction**, containing a binary mask for every image in the private test set
- a folder called `code`, containing all program code (not including libraries) needed to run your model.
- the checkpoint file, containing all parameters of your model as used to generate the predictions
- the file `teammates.txt`, in the format as described above.

Submit this `.zip` file to Moodle. Only one person per team needs to submit.