

# Biologically Inspired Artificial Intelligence - Project report

Michał Bugdoł





September 15, 2024

# Contents

|          |                          |          |
|----------|--------------------------|----------|
| <b>1</b> | <b>Introduction</b>      | <b>3</b> |
| <b>2</b> | <b>Dataset</b>           | <b>3</b> |
| <b>3</b> | <b>Data augmentation</b> | <b>4</b> |
| <b>4</b> | <b>Model</b>             | <b>5</b> |
| <b>5</b> | <b>Training</b>          | <b>8</b> |
| <b>6</b> | <b>Summary</b>           | <b>9</b> |

# 1 Introduction

The goal of this project was to train an artificial neural network to recognize camouflaged animals in images. The network is a semantic segmentation model, which takes an image and classifies each pixel of the image into one of the following categories:

- **masking background** (green ): the sections made by pixels of this category constitute the object the animal is camouflaging in,
- **animal** (blue ): the sections made by these pixels represent the hidden animal,
- **attention** (white ): the sections made by pixels of this class draw attention (due to their size, shape, color...), even though they are neither part of the animal nor the camouflaging classes. An example would be a colorful flower in an image of a fox hiding in the snow,
- **background** (red ): the pixel doesn't belong to any other category.

The model output is then taken and processed to create a mask image according to the above color-codes.

# 2 Dataset

The dataset consists of 272 images of animals and matching images of the segmentation masks. The dataset was created by students of the SUT by finding relevant images online and tagging them by hand. The images are of various sizes and use different file extensions - mostly `.jpg` (226 files), but also `.png` (40 files) and `.jpeg` (6 files).

An example image-mask pair taken from the dataset can be seen on figure 1.

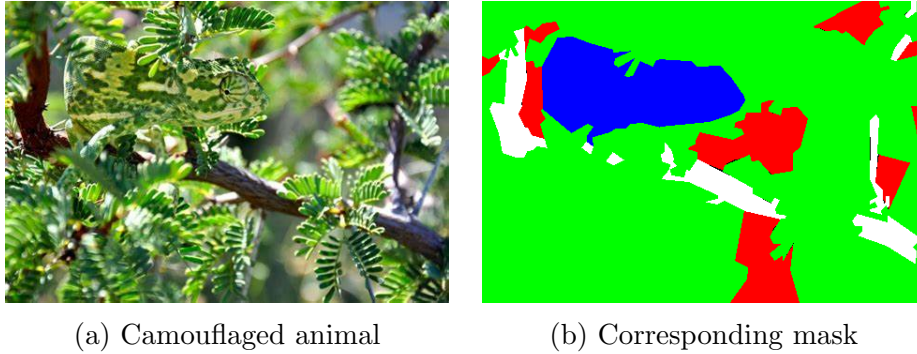


Figure 1: Example of an image-mask pair taken from the dataset

### 3 Data augmentation

Before feeding the dataset into the model, it was first augmented to artificially increase its size. Data augmentation is a process aiming to reduce overfitting and improve learning rates.

Data augmentation of the camouflaged animals dataset started by resizing all images and masks to the target size of  $512 \times 512$ . Then, some predefined transformations were applied at random. These transformations were split into two groups: common transformations and input transformations.

Common transformations are transformations, which may change the meaning of the image, and thus need to be applied to both the input and target image in the same way. Some examples of common transformations include:

- horizontal flip
- rotation
- translation
- scaling
- random dropout (remove some parts of the image)

Input transformations don't alter the meaning of the input image, and thus don't need to be applied to the target mask. These operations include:

- color jitter (adjust hue & brightness of image)

- equalization (equalizes the histogram of the image)
- grayscale
- blurring / sharpening
- posterization (reduce number of bits in a color channel)

From the list of above transformations, one was chosen at random each time using the `torchvision.v2.RandomChoice` transformation. An example of an image before and after augmentation can be seen in figure 2.

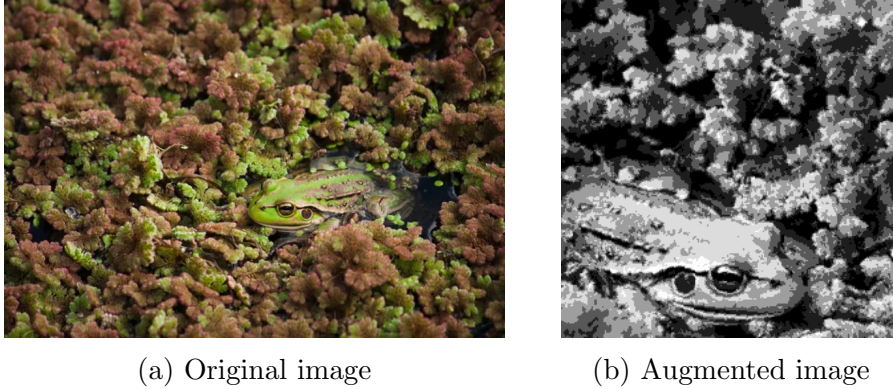


Figure 2: Example of an image before and after augmentation

## 4 Model

The U-Net was chosen to tackle this task. The U-Net is a convolutional neural network (CNN), first designed for medical image segmentation. The main characteristics of an U-Net are its ability to perform well with a relatively small amount of data and ability to train the upsampling process. The name comes from its shape - for each downsampling (convolution) operation, there is one upscaling operation, which when drawn out creates a 'U' shape.

For convoluting the image, a *depthwise separable-convolution* (DSC) approach was chosen. This approach is characterized by splitting the spatial correlation and the cross-channel correlation into two convolutions. The first convolution (depthwise) aims to reduce the dimensions of the input matrix,

whilst preserving the channel count. The second convolution (pointwise) reduces the channel count using a  $1 \times 1$  kernel without changing the spatial properties of the matrix. This approach allows to train the spatial and cross-channel reasoning separately. At the end of each DSC, a ReLU layer is added to introduce non-linear behavior.

The model consists of 3 downsample-upsample blocks. The downsample part is made using some count of DSC-ReLU iterations on the input image, while only the first iteration changes the channel count, followed by a single MaxPool operation. Upsampling is done using some number of DSC-ReLU iterations with the last operation changing the channel count, followed by a single MaxUnpool operation.

The model was implemented in `Python 3.10` using `PyTorch` and `TorchLightning` libraries. The full source code of the model (along with the dataset) can be found on the GitHub repository.<sup>1</sup>

The model summary can be seen in table 1, and the model graph can be seen in figure 3.

Table 1: U-Net summary (without batch dimension)

| Type       | Params | Input sizes     | Output sizes    |
|------------|--------|-----------------|-----------------|
| DownDSC    | 12.0 K | [ 3 , 512, 512] | [64 , 256, 256] |
| DownDSC    | 49.7 K | [64 , 256, 256] | [128, 128, 128] |
| DownDSC    | 253 K  | [128, 128, 128] | [256, 64 , 64 ] |
| UpDSC      | 256 K  | [256, 64 , 64 ] | [128, 128, 128] |
| UpDSC      | 51.2 K | [128, 128, 128] | [64 , 256, 256] |
| UpDSC      | 13.5 K | [64 , 256, 256] | [ 4 , 512, 512] |
| Full U-Net | 637k   | [ 3 , 512, 512] | [ 4 , 512, 512] |

---

<sup>1</sup><https://www.github.com/MBugdol/u-net-impl>

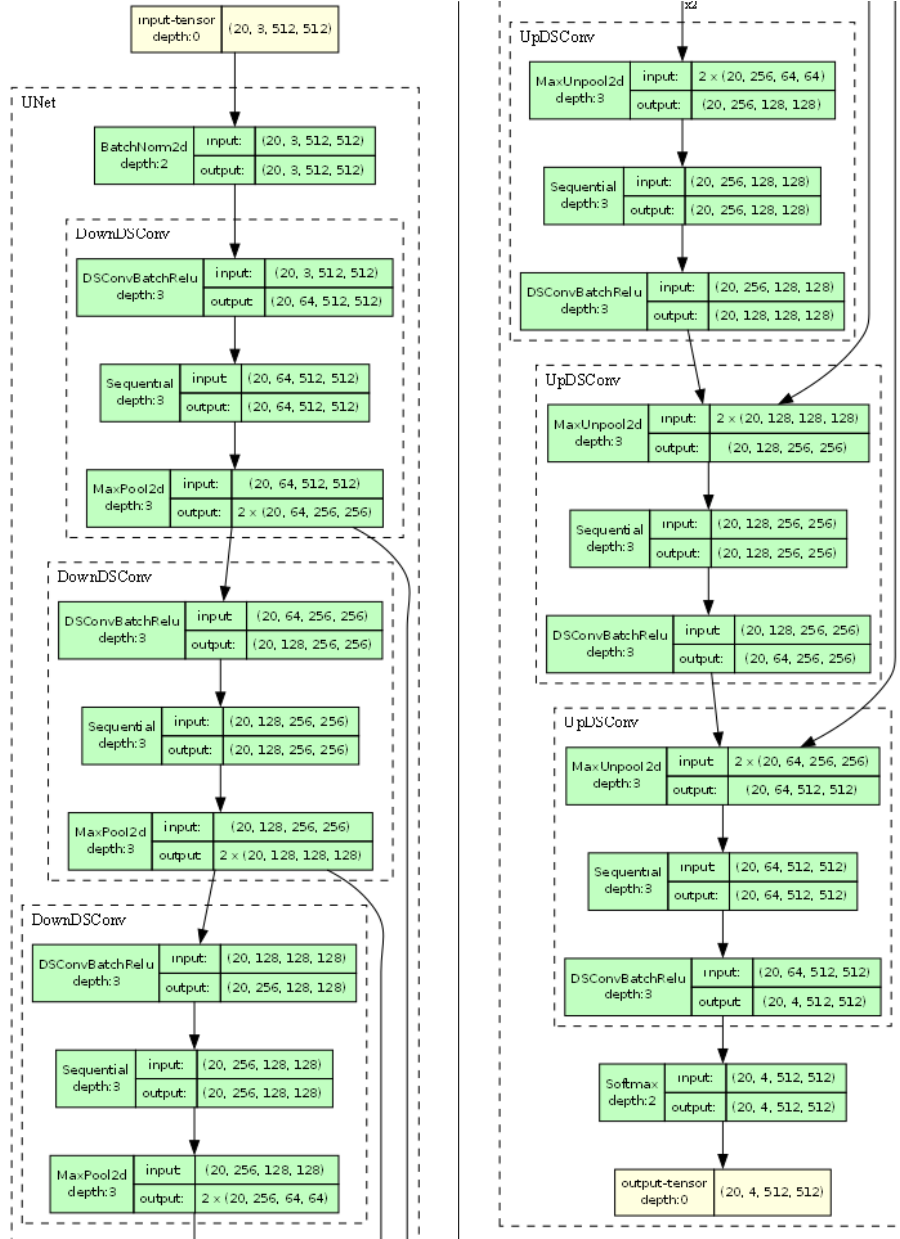


Figure 3: U-Net visualisation as a graph

## 5 Training

The model was trained on a NVIDIA RTX A500 laptop GPU, in batches of 5 images. The model was left to train for 2 days (50 epochs). The resulting loss and accuracy graphs can be seen on figures 4 and 5.

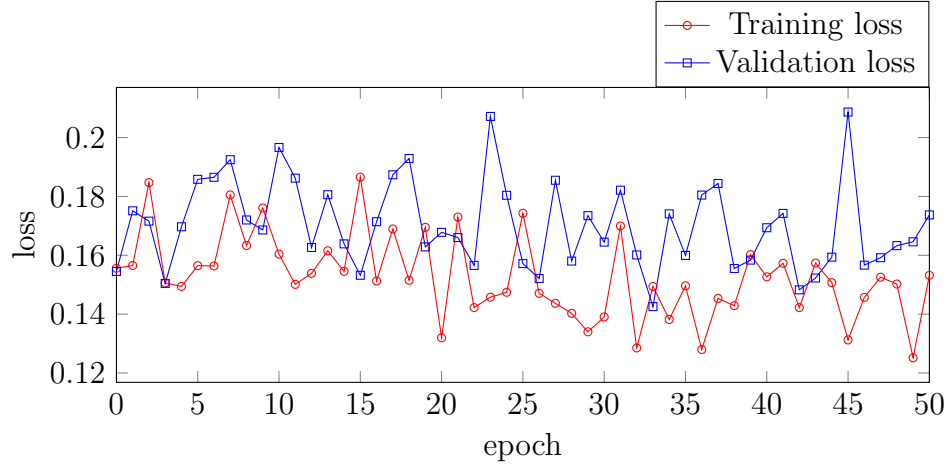


Figure 4: Loss graph (training & accuracy)

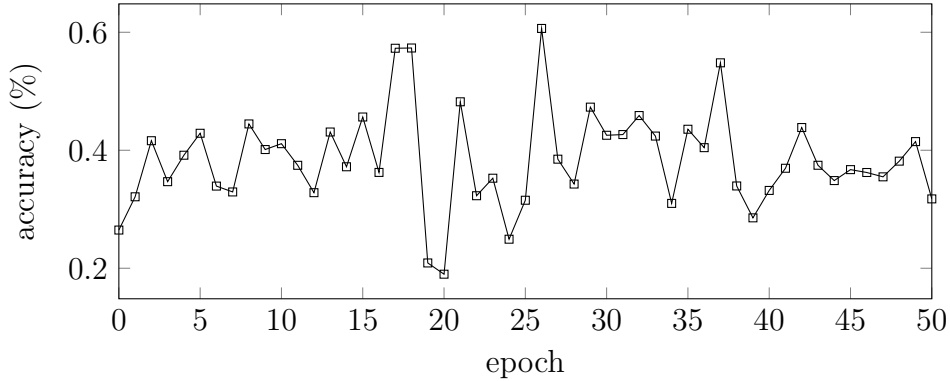


Figure 5: Validation accuracy graph

Looking at the loss graph, whilst a decreasing trend can be seen for training loss, no obvious trend is present in validation loss data. Validation accuracy also shows no progress over epochs 0-50.



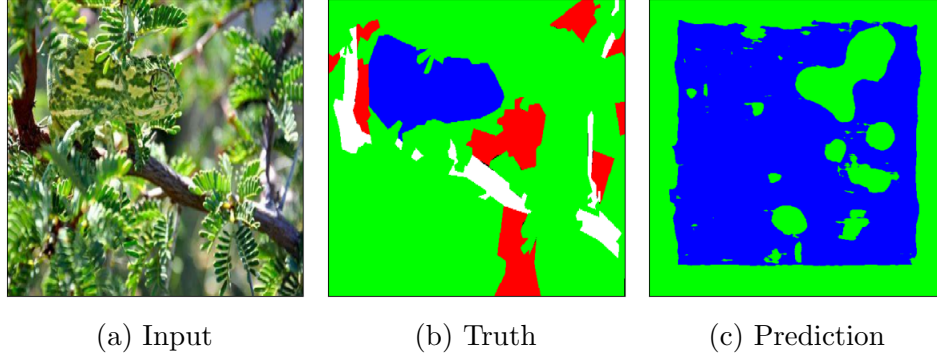


Figure 6: Input, truth mask and predicted mask after 50 epochs of training

An example image from the dataset was passed to the model to see how well it performs. Input, truth mask and the predicted mask images can be seen on figure 6

As can be seen, the model failed to achieve its task of finding camouflaged animals on an image. The resulting prediction preserves no hint of the original image, and shows that the model has a preference to assigning pixels to the camouflaged animal group.

## 6 Summary

The model didn't achieve what it was aiming to. The accuracy and loss metrics show that the model isn't accurate at finding camouflaged animals in images, and, judging by the trends present in the resulting data, this could hint to overfitting on the training set. The unsatisfactory result could be attributed to several factors described below.

### Parameter count

The model was relatively big - it had 637'000 trainable parameters. This, combined with the relatively small dataset made it impossible to sufficiently train the model to fulfill its role. This issue could possibly be solved with using a simpler model or fewer layers.

## **Dataset size**

The dataset just had not enough data for the task it was aiming for, even when adding data using augmentation. Options to mitigate this problem include expanding the dataset or using a pre-trained segmentation model and specializing it using the given set of images.

## **Task complexity**

Recognizing animals in images is a hard task in itself - with camouflaging added to the problem, it could be possible that the model created for this task just wasn't complex enough for the job. Finding details hinting at a presence of camouflaged animals is difficult for even the most advanced image recognition model, i.e. humans. The model may have not been able to find the intricate patterns revealing the presence of an animal, moreso with presence of noise (the attention class), that would have been a major confusing factor in this task.