Liver Medical Image Semantic Segmentation

Anaïs Dubois and Adam Dzadon

November 20, 2024

Abstract

The use of Artificial Intelligence (AI) in the medical field is essential for improving current methods and rapidity. This study analyses the performance of a U-Net neural network for detecting the liver in human CT scans. For the training and testing, the mean Pixel Average (mPA) obtained is more than 0.99, and the mean Intersection over Union (mIoU) obtained is over 0.98. The loss value is lower than 0.2 on both validation and training. The U-Net performs very well in detecting the liver in the data set used here, and have better results than a simple Convolutional Neural Network (CNN).

1 Problem Description

The use of AI in the medical field is gaining ground. Nowadays, new challenges appear to improve AI efficiency, in terms of rapidity and accuracy. The purpose of this study is to evaluate the performance of a U-Net neural network, as defined by Ronneberger [RFB15], in an image semantic segmentation task. More precisely, the task of the neural network here is to find the liver in axial human CT scans.

The data can be accessed via this URL. The data set contains both inputs and corresponding annotated expected outputs (called labels or masks). In the masks, white pixels represent the liver. All Python files and results can be found on GitHub. PyTorch was used to build the neural network.

2 Approach to Solution

Our research task was divided into many steps:

1. Exploratory Data Analysis (EDA) and data preprocessing
2. Building models
3. Training models and adjusting parameters
4. Evaluating models.

The Exploratory Data Analysis (see section 2.1) allowed us to gain knowledge about the data provided, and interpret more accurately the results we obtained. Data preprocessing was done at the same time, by splitting the data set into training, validation, and testing data sets.

Then, we built the models. The U-Net model was adapted from the article by Ronneberger [RFB15]. The CNN baseline model was custom made but inspired by the same article. The idea was to keep some blocks from the U-Net, but with fewer and simpler layers. Thus, the baseline model contains convolutional layers, pooling layers and upsampling layers. As in the U-Net, ReLU was used.

We trained the models and adjusted them to improve performance metrics and reduce loss values (for training and validation). Performance metrics used here are *mean Pixel Accuracy (mPA)* and *mean Intersection over Union (mIoU)*. Since it is a binary task, Binary Cross Entropy (BCE) was used for the loss metric. We adjusted the number of epochs to obtain the best results possible.

The U-Net model was evaluated thanks to previous performance and loss metrics, and compared to the baseline model.

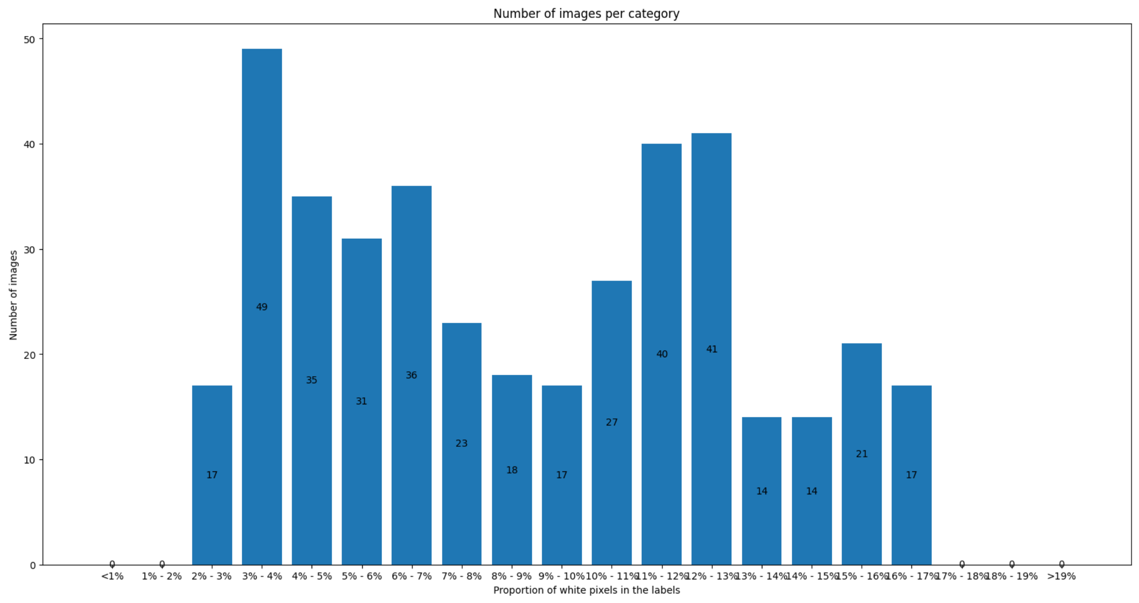


Figure 1: Number of white pixels in labels. The x-axis represents the percentage of white pixels in masks. The y-axis represents the number of masks. The chart gives a repartition of the number of masks per proportion of white pixels in it.

2.1 Exploratory Data Analysis

2.1.1 Analysis on the whole dataset

The data set contains 400 pairs of images and labels. The images and labels have size 512×512 pixels, with 1 color channel, thus, they are grayscale — each pixel represents the amount of light on a given position. Before the further processing, we normalized the values of each pixel to zero mean and unit variance.

The *images* contain many shades of gray, but the *masks* are in black and white. Figure 1 represents the number of masks depending on the proportion of white pixels in the masks (where white pixels represent the liver). The percentage of white pixels is between 2% and 17%, so a model predicting only "black" pixels (i.e., the liver is nowhere) would be expected to have at least 83% accuracy. The standard accuracy is then not really relevant here. Instead, mean Pixel Accuracy (mPA) and mean Intersection over Union (mIoU) are used as performance metrics.

2.1.2 Analysis on the training, validation, and testing datasets

The dataset has been split randomly to obtain training, validation, and testing data sets. The data sets used for training (training and validation data sets) contain 360 pairs of image and label i.e. 90% of the whole data set. The testing data set contains 40 pairs, i.e. 10% of the whole data set. The 360 pairs, were then split between actual training data set and validation data set, with 288 pairs of image and label in the training data set (80% of the 360 pairs) and 72 pairs in the validation data set (20% of the 360 pairs).

Figure 2 shows this repartition (number of masks depending on the proportion of white pixels in the masks) for the training dataset. Figure 3 shows it for the validation data set and Figure 4 for the testing data set.

The training data set fairly represents the whole data set, as the labels repartition in Figure 2 is similar to that shown in Figure 1. The validation data set (Figure 3) and the testing data set (Figure 4) are less representative in terms of the number of images per category. However, in the validation dataset, all categories are represented.

To conclude on this part, Figure 5 displays the average proportion of white pixels per image, in each data set. The mean is always in the same range, between 8% and 9%, and the standard deviation is very similar between the whole data set and the training data set. The standard deviation varies of 1% for validation and testing.

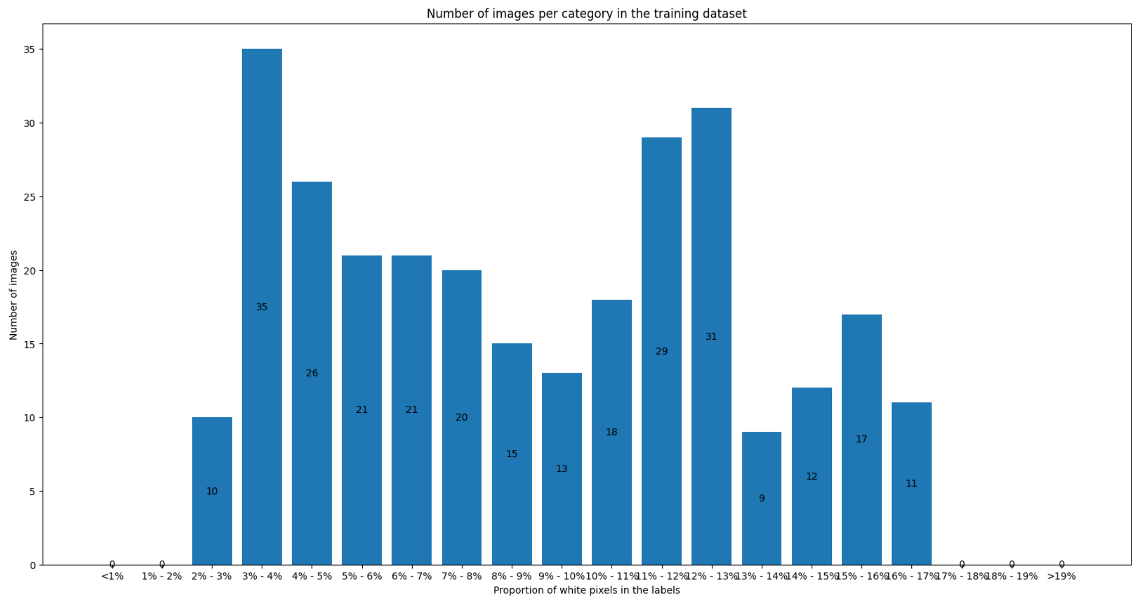


Figure 2: Number of white pixels in training labels. The x-axis represents the percentage of white pixels in masks. The y-axis represents the number of masks. The chart gives a repartition of the number of masks per proportion of white pixels in it.

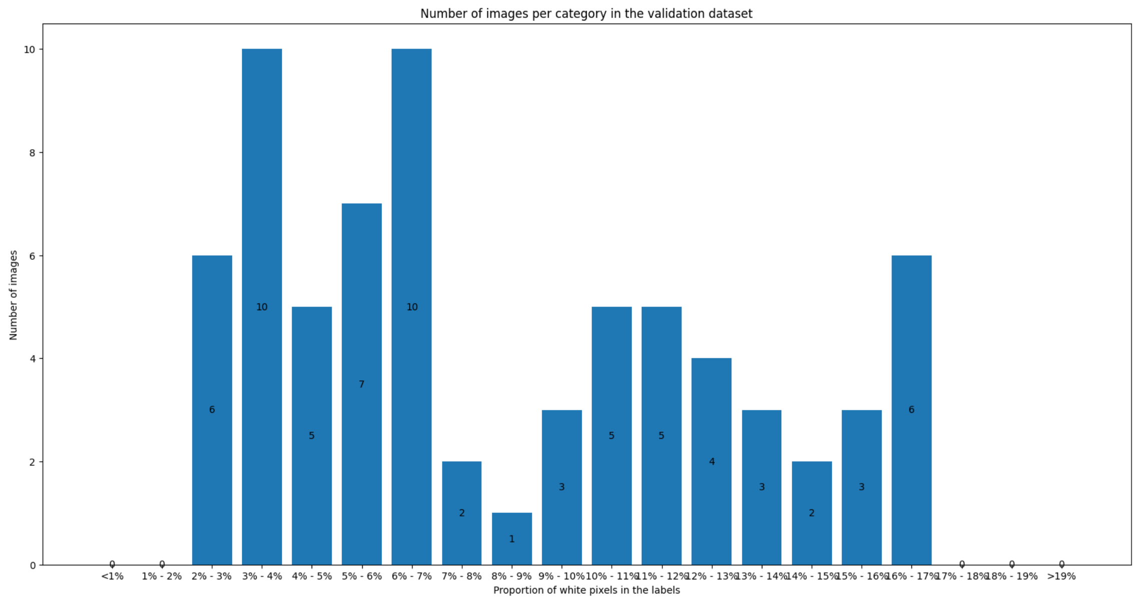


Figure 3: Number of white pixels in validation labels. The x-axis represents the percentage of white pixels in masks. The y-axis represents the number of masks. The chart gives a repartition of the number of masks per proportion of white pixels in it.

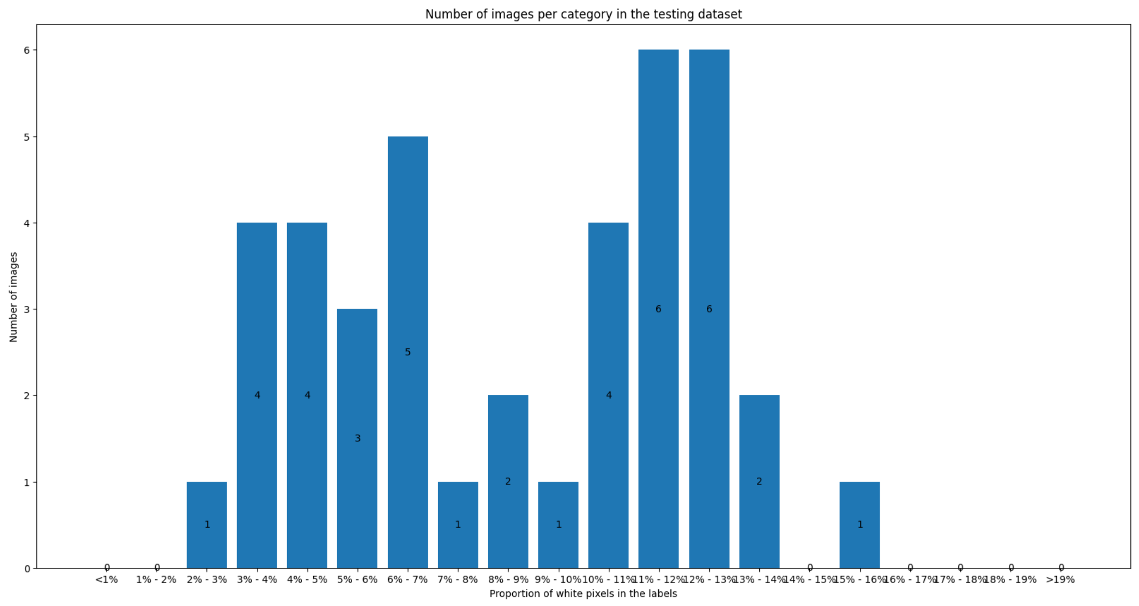


Figure 4: Number of white pixels in testing labels. The x-axis represents the percentage of white pixels in masks. The y-axis represents the number of masks. The chart gives a repartition of the number of masks per proportion of white pixels in it.

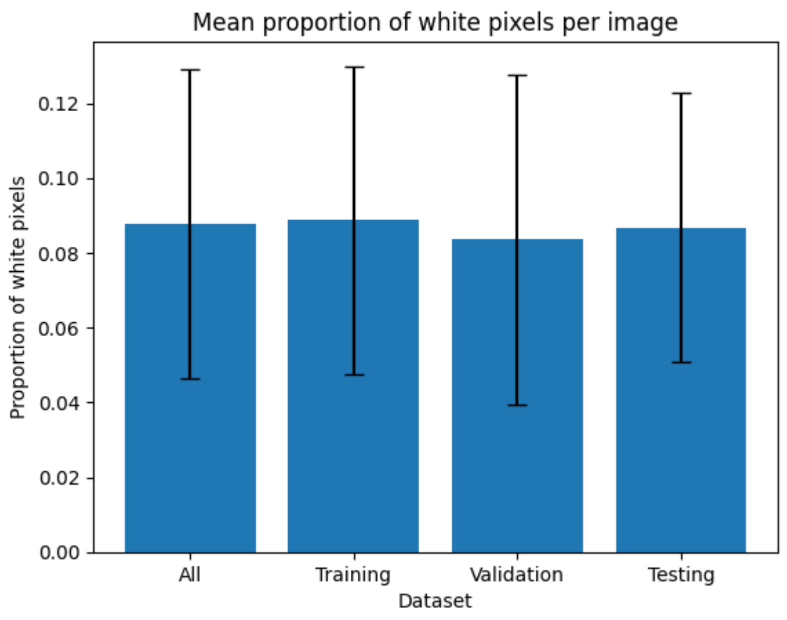


Figure 5: The average proportion of white pixels per image, in each data set: "All", "Training", "Validation" and "Testing". The mean is represented by the blue bars. The standard deviation is represented by the error bars.

3 Implementation

The U-Net model can be found here, and the baseline model is available here, in our GitHub repository.

The configurations used for training both models are as follows:

```
{
    "model"            : "unet",
    "in_dim"           : 1,
    "out_dim"          : 1,
    "out_activation"   : "sigmoid",
    "tdata_x_path"     : "data/train_x.csv",
    "tdata_y_path"     : "data/train_y.csv",
    "vdata_x_path"     : "data/val_x.csv",
    "vdata_y_path"     : "data/val_y.csv",
    "lr"               : 1e-5,
    "optim"            : "adam",
    "loss"             : "bce",
    "train_batch_size" : 1,
    "valid_batch_size" : 1,
    "epochs"           : 7,
    "val_metrics"      : ["mPA", "mIoU"],
    "save_model"       : true,
    "seed"             : 42,
    "num_workers"      : 0
}
```

Listing 1: Training settings. "unet" should be replaced by "baseline" for training the baseline model.

The number of epochs was set at 7, since it seemed that fewer epochs did not give optimal results on the U-Net, and with more epochs the model was not improving significantly in terms of accuracy metrics (mPA and mIoU, see Figure 8). We trained the baseline model on the same number of epochs as determined to be optimal for the U-Net.

The optimizer used is *Adam*. Indeed, it combines properties of multiple optimizers, it adapts learning rate for each parameter, and proved to be very robust in most of the situations.

The learning rate was set to 1e-5. Bigger learning rates gave worst results.

The activation function chosen is *Sigmoid* for both models, since the case studied here is a binary classification task: white pixels (probability 1) stand for "the liver is here" and black pixels (probability 0) stand for "the liver is not here". During prediction, a threshold was set at 0.5 to transform probabilities into integers: 0 if the probability was above the threshold, or 1 if the probability was under the threshold.

Finally, batch sizes was set to 1 due to better performance compared to the higher values.

4 Discussion on Intelligence

4.1 Accuracy metrics and loss

Table 1 shows the evolution of metrics during the training process. These results are also displayed in the charts in Figure 6, 7, 8 and 9.

Even if loss is higher for U-Net than for the baseline model (Figure 7), the mPA and mIoU accuracy metrics are higher and closer to 1 for U-Net than for the baseline model (Figure 6). One can notice that the mPA and mIoU values are already very high at the beginning of U-Net training, which is not the case for the baseline model (mPA and mIoU around 0.5 at the beginning).

The learning curve for U-Net is displayed more precisely in Figure 8. As shown, the metrics are very high, and so since the beginning, with mIoU over 0.94 and mPA over 0.98. They both improve overall during training. In Figure 9, it is noticeable that loss decreases at each epoch. The validation loss is always under the training loss, which is surprising, since we would expect the validation loss to be higher than the training loss at the beginning (*underfitting*) and then lower than the training loss (*overfitting*). However, this phenomenon is also obeved with the baseline model (Figure 7), so it may be due to the data sets. Here, both curves tend to overlap at the end. Since the accuracy metrics were not improving significantly after 7 epochs, we decided to stop the training process at 7 epochs. The same number of epochs was used for the baseline model.

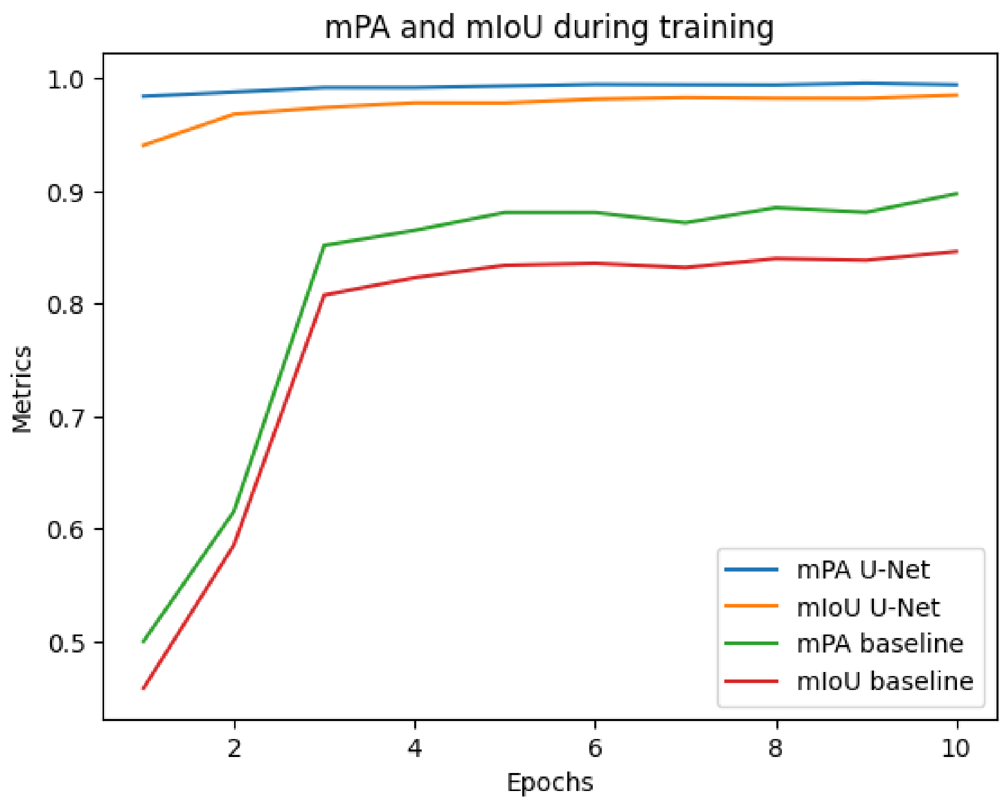


Figure 6: mPa and mIoU values during training, for both models

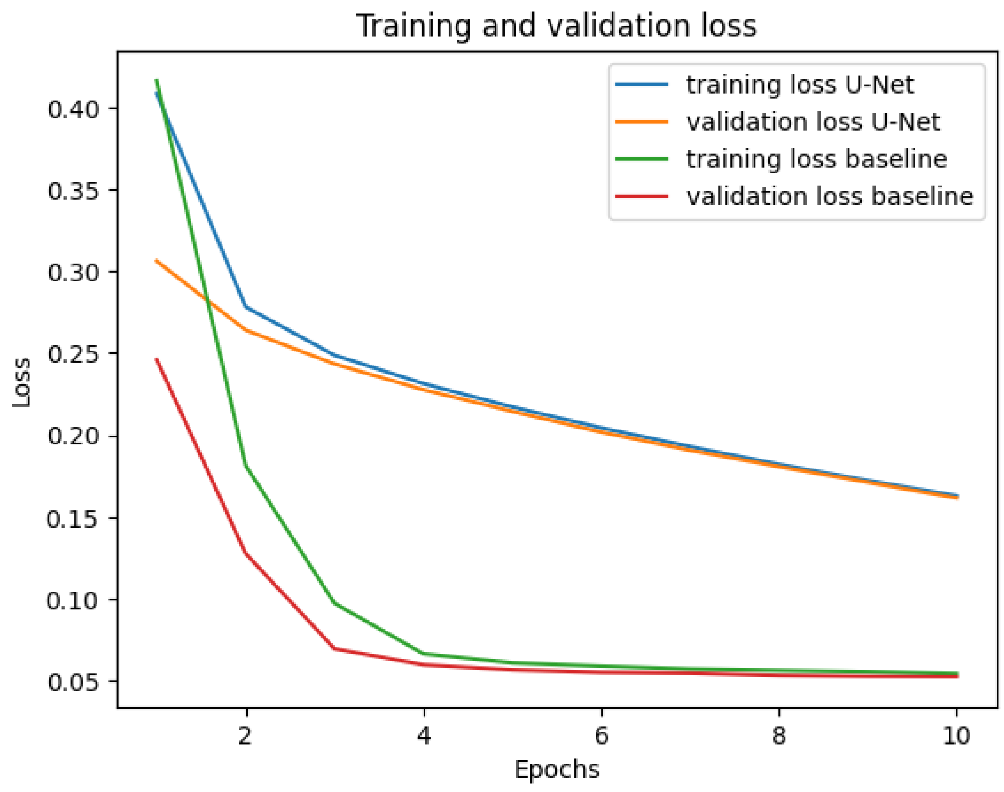


Figure 7: Training and validation loss during training, for both models

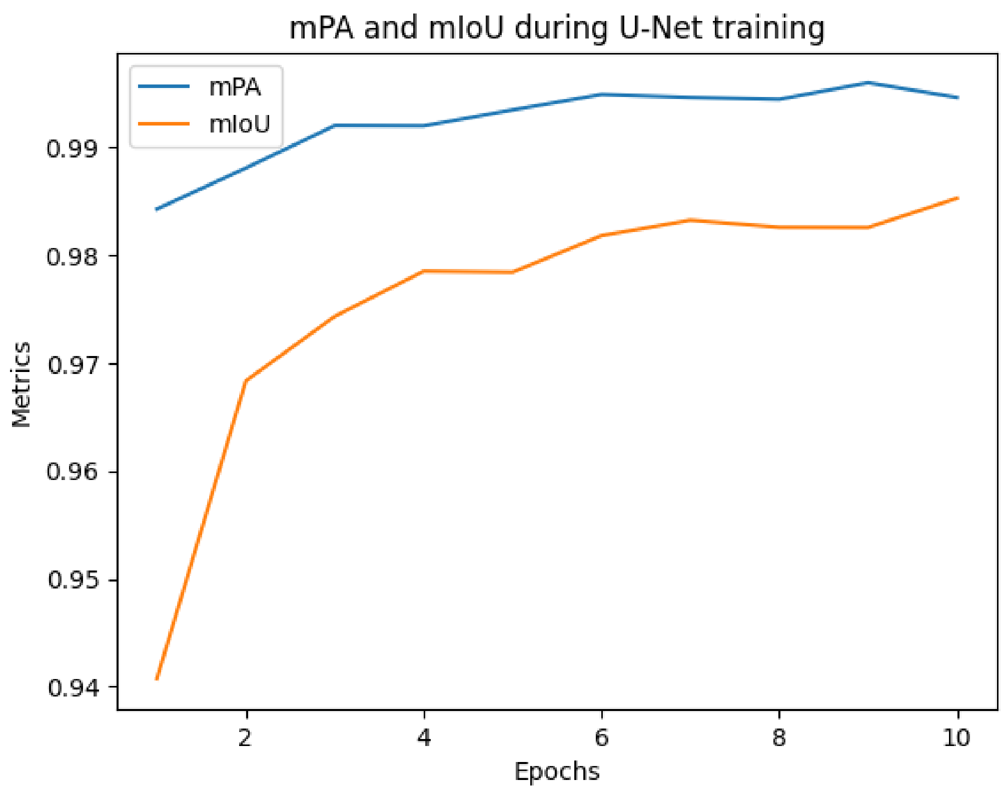


Figure 8: mPa and mIoU values during U-Net training

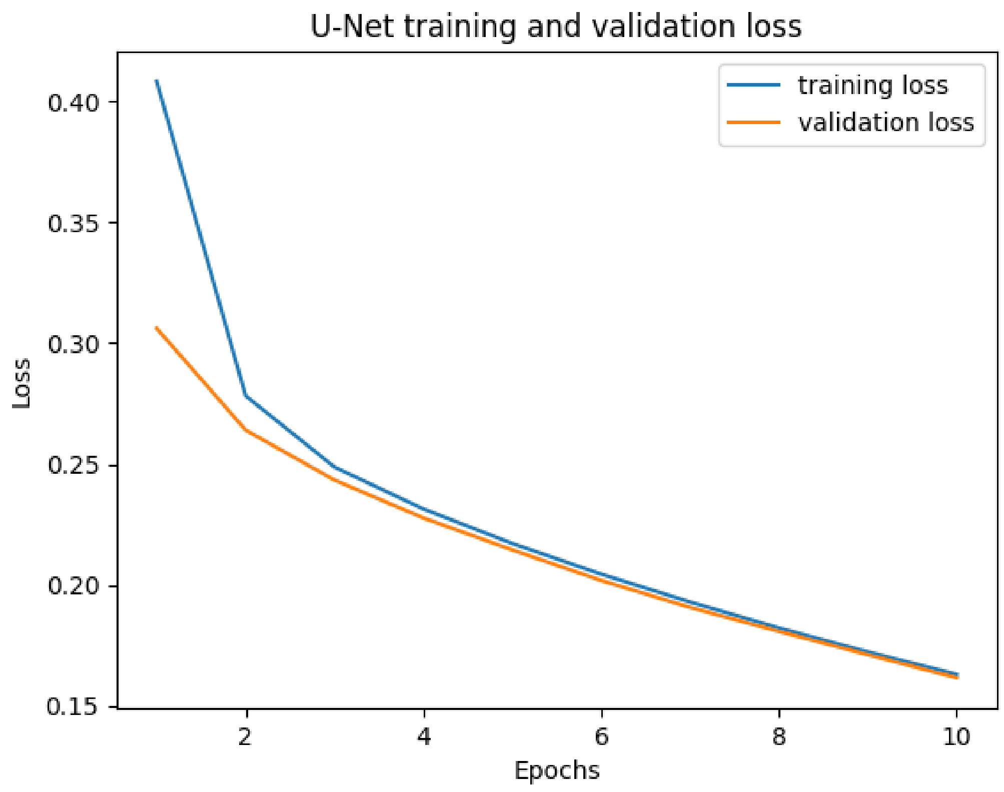


Figure 9: Training and validation loss during U-Net training

	Baseline				**U-Net**			
Epochs	mPA	mIoU	train loss	val loss	mPA	mIoU	train loss	val loss
1	0.5	0.4582	0.4163	0.2460	0.9843	0.9407	0.4084	0.3061
2	0.6153	0.5854	0.1814	0.1276	0.9881	0.9683	0.2782	0.2640
3	0.8518	0.8076	0.0972	0.0694	0.9920	0.9743	0.2487	0.2433
4	0.8650	0.8231	0.0663	0.0597	0.9920	0.9785	0.2314	0.2276
5	0.8810	0.8341	0.0608	0.0566	0.9934	0.9784	0.2171	0.2144
6	0.8809	0.8358	0.0588	0.0551	0.9949	0.9818	0.2044	0.2018
7	0.8718	0.8321	0.0571	0.0546	0.9946	0.9832	0.1929	0.1906
8	0.8852	0.8401	0.0562	0.0532	0.9944	0.9826	0.1821	0.1807
9	0.8811	0.8386	0.0554	0.0526	**0.9960**	0.9826	0.1722	0.1710
10	**0.8976**	**0.8464**	**0.0543**	**0.0524**	0.9946	**0.9853**	**0.1629**	**0.1615**

Table 1: mPA, mIoU, and loss values for baseline and U-Net models, on 10 epochs

Baseline		**U-Net**	
mPA	mIoU	mPA	mIoU
0.9081	0.8471	**0.9941**	**0.9854**

Table 2: mPA and mIoU for baseline and U-Net models during evaluation.

4.2 Evaluating models

4.2.1 Accuracy metrics during testing

The models, as described in Listing 1, are evaluated on the test data set. The mPA and mIoU results are displayed in Table 2. U-Net has better results than the baseline model: +9% for mPA and +14% for mIoU.

4.2.2 Results on three images

The models were tested more precisely on three images, No. 33, No. 158 and No. 209 (Figure 10).

The corresponding masks, as well as baseline and U-Net predictions, are shown in Figures 11, 12 and 13. Figures 14, 15 and 16 add colors to improve understanding. Blue color corresponds to "false positive", i.e. the model predicted "liver" (white pixel) when there was "no liver" ; red color corresponds to "false negative", i.e. the model predicted "no liver" (black pixel) when there was "liver". This shows the high performance of the U-Net, especially compared to the baseline model. The baseline model predicts shapes that barely fit the ground-truth, while the U-Net has nearly no error.

5 Quotations

The U-Net architecture was based on the paper from Ronneberger [RFB15]. For the baseline CNN, ChatGPT was used for solving problems during code edition, especially about dimensions

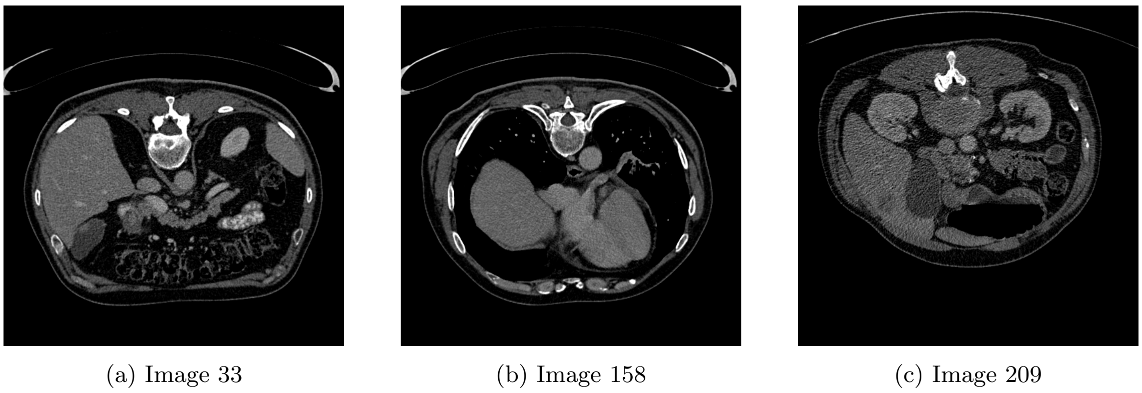
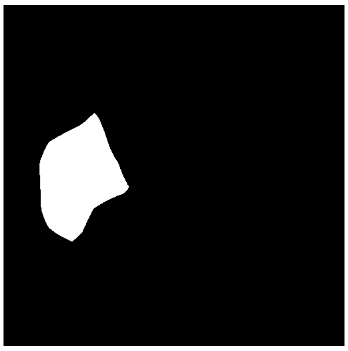
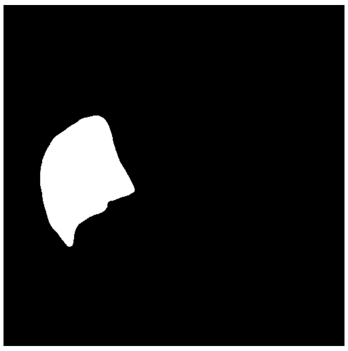

(a) Image 33 (b) Image 158 (c) Image 209

Figure 10: Images provided to the models for testing


(a) Baseline


(b) U-Net

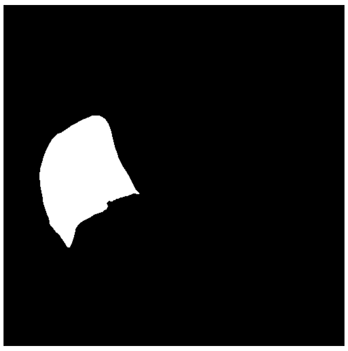
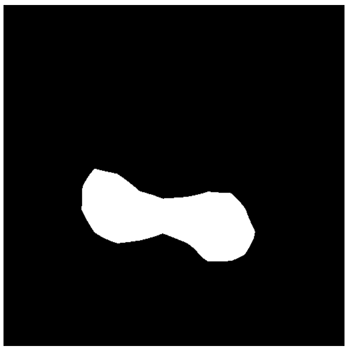
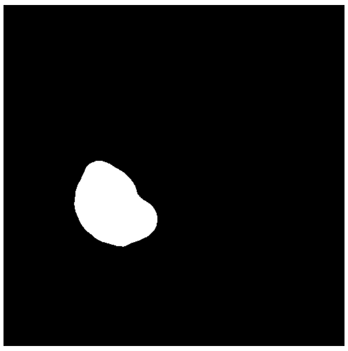

(c) Ground-truth (masks)

Figure 11: Baseline and U-Net predictions on image No. 33, and ground-truth


(a) Baseline


(b) U-Net

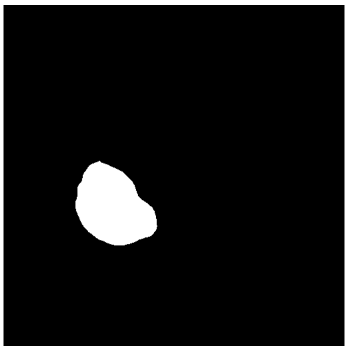
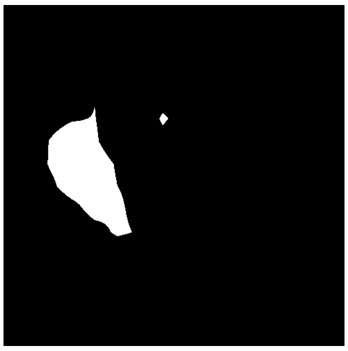
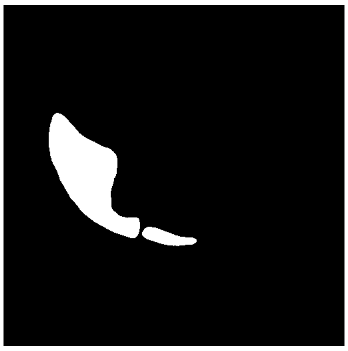

(c) Ground-truth (masks)

Figure 12: Baseline and U-Net predictions on image No. 158, and ground-truth


(a) Baseline


(b) U-Net

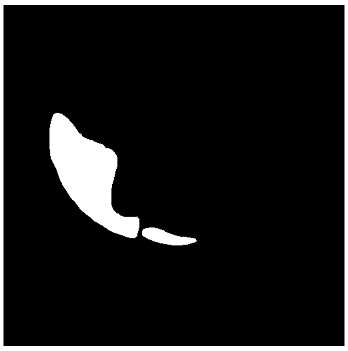

(c) Ground-truth (masks)

Figure 13: Baseline and U-Net predictions on image No. 209, and ground-truth

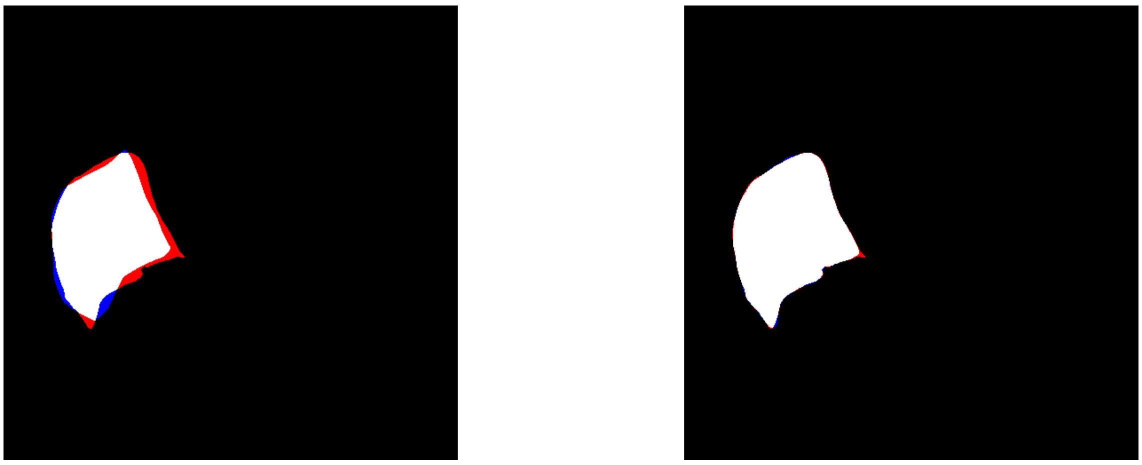

(a) Baseline (b) U-Net

Figure 14: Baseline and U-Net predictions on image No. 33, with colors. Blue color corresponds to "false positive", i.e. the model predicted "liver" (white pixel) when there was "no liver" ; red color corresponds to "false negative", i.e. the model predicted "no liver" (black pixel) when there was "liver".

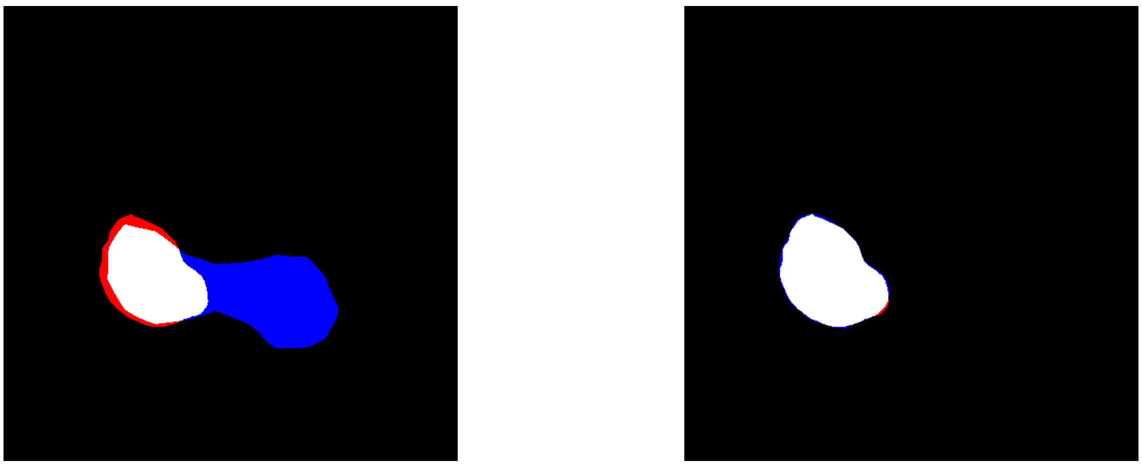

(a) Baseline (b) U-Net

Figure 15: Baseline and U-Net predictions on image No. 158, with colors.

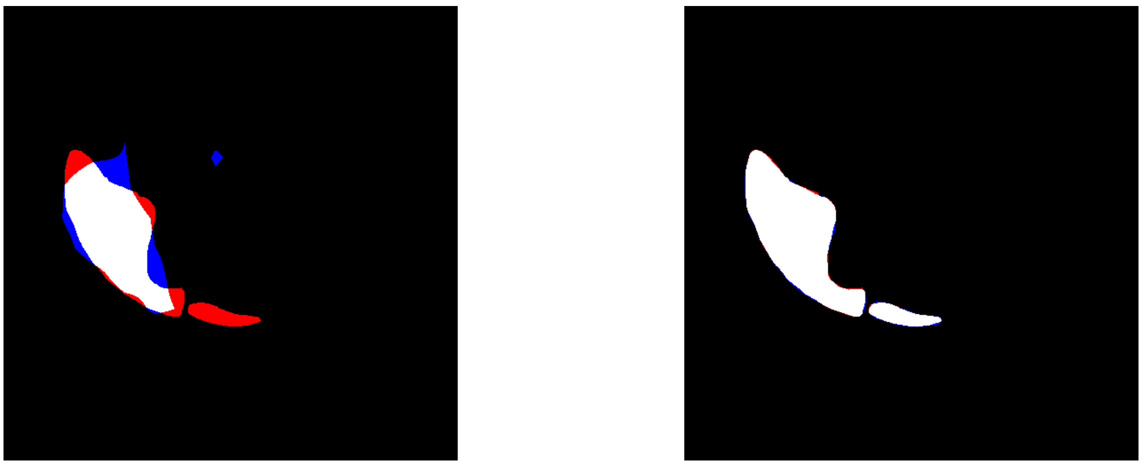

(a) Baseline (b) U-Net

Figure 16: Baseline and U-Net predictions on image No. 209, with colors.

matching. Documentations from PyTorch, NumPy, Matplotlib, Pandas, Scikit-learn, TQDM, Pillow and CUDA were used.

For layout in this report, LaTeXdocumentation in Overleaf was used.

6 Contributions

The implementation of the U-Net architecture was handled by Adam Dzadon. The baseline model was implemented by Anaïs Dubois. Since they both worked on different model, they developed training and testing for their model. The final and unified tools to train, save and test models (that appears in the GitHub repository) were created by Adam Dzadon. Anaïs Dubois did the exploratory data analysis, charts about metrics, and images colorization.

This report was written by Anaïs Dubois.

7 Reflections

The U-Net model performed particularly well on the data set provided, with the parameters set. It outperformed the simple CNN baseline model, ans highly proved its efficiency. However, one question is still pending, that is to know if the model would perform that well on other data sets. Further work should be needed to evaluate U-Net model performance on new data. Nevertheless, the current results, compared to the baseline model, let us think that it would still perform better than average neural networks.

References

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.