CorrSeg

Technical report for the ChallengeData 2024 competition

Andrej Perković

May 2024

Contents

Chapter 1

Introduction

The problem of semantic segmentation of rust on ultrasound images was presented at this years *ChallengeData*. It is managed by the Data team (ENS Paris - PSL), in partnership with the Collège de France and the DataLab at Institut Louis Bachelier. It is supported by the CFM chair, the PRAIRIE Institute and IDRIS from CNRS. It is a platform when institutions post challenges on one side and student try at takling them on the other.

The challenge I decided to participate in presented by SLB SAS. SLB is a global technology company that was founded in France in 1927 by Conrad and Marcel Schlumberger to provide cabling services to the oil industry. Since 1929, strong investments in research to develop new logging tools and strategic acquisitions have positioned them as one of the leading companies in their field. Presently, operating in more than 100 countries with employees from nearly 200 nationalities.

Chapter 2

The Challenge

The problem at hand is the one of semantic segmentation. Corrosion-related failures represent serious hazardous events and account for more than 25% of overall well failures in the oil and gas industry. Corrosion of steel pipes throughout the years due to extreme conditions in the wellbore can have negative financial consequences, with potential for significant environmental impacts (groundwater contamination, gas leakage, and seepage at the surface). Pipe condition evaluation for well integrity workflows involves corrosion characterization, leak and structural deficiency monitoring, and detection of potential breaks in the pipe. Accurate corrosion detection is important for both operators and service companies to decrease interpretation time, reduce subjectivity in monitoring, and increase overall performance.

The goal of the hackathon is to produce a model that gives the highest possible score for groove defect segmentation.

This competition is evaluated on the intersection over union (IoU). The IoU can be used to compare the pixel-wise agreement between a predicted segmentation and its corresponding ground truth. The formula is given by:

$$\frac{|X \cap Y|}{|X \cup Y|}$$

where X is the predicted set of pixels and Y is the ground truth. The IoU is defined to be 1 when both X and Y are empty. The leaderboard score is the mean of the IoU coefficients for each image in the test set. As an approximation of the leadboard IoU score and for the purpose of monitoring the training of the model, I estimate the IoU on the validation data in a similar fashion

Chapter 3

Data Description

The pipes manufactured using a mold represent manufacturing patterns. Manufacturing patterns are homogeneous within small sections of the pipes called joints; they are organized in repetitive and visually coherent shapes and forms (circles and horizontal, vertical, or oblique lines with varying thickness). However, intra-joint feature distribution is high; two consecutive joints within the same well can have distinctly different manufacturing patterns. These patterns reflect loss and excess of metal due to the manufacturing process.

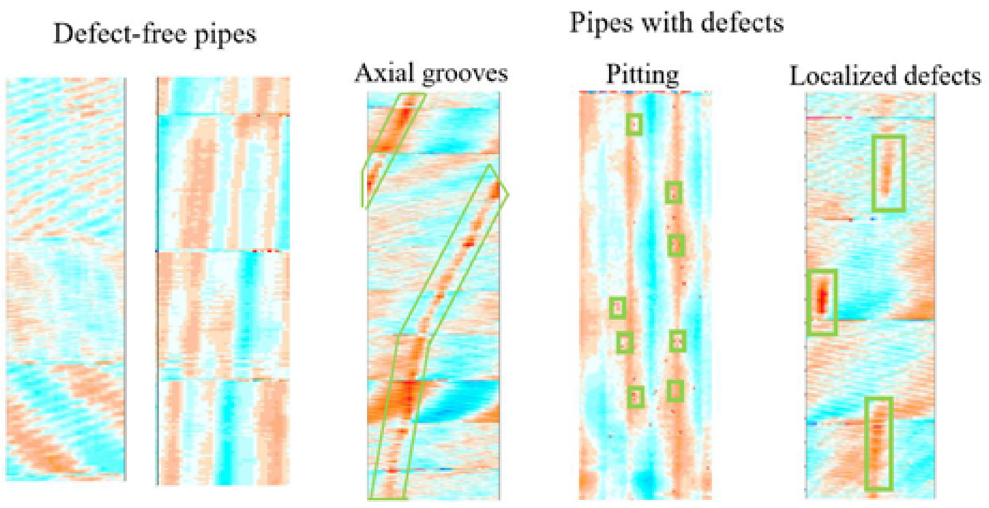


Figure 3.1: Corrosion and wear defects examples

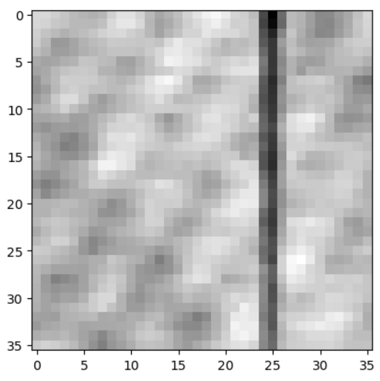
Collars are defined as the junction between two joints and are displayed in the radius and thickness map as a line.

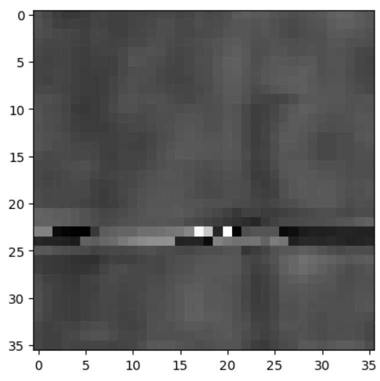
Manufacturing patterns and collars represent a background on which defects and anomalies are overlaid. Corrosion in the pipe always appears as metal loss (red patterns in figure 3.1); pipes lose their thickness in the inner or in the outer wall, sometimes simultaneously. (Corrosion is not an either-or process; it may happen on both the inner and outer wall at the same time). Those defects are random in size, metal penetration, and overlay on top of the manufacturing patterns.

Corrosion and wear defects on steel pipes can be classified into three categories:

- Pitting corrosion
- Localized defects
- Axial groove

As mentioned earlier, the database comprises ultrasonic images from **20 wells**. Each image within the database is paired with a binary mask of identical dimensions, acting as the corresponding label for that specific image. This gives a total of **9674** image-label pairs. Images are in grayscale, like the examples in Figure 3.2. Masks are made of 1's and 0's, classifying each pixel based on whether or not there is defect in it, like in the examples in Figure 3.3


(a) Well 2, patch 456


(b) Well 3, patch 1169

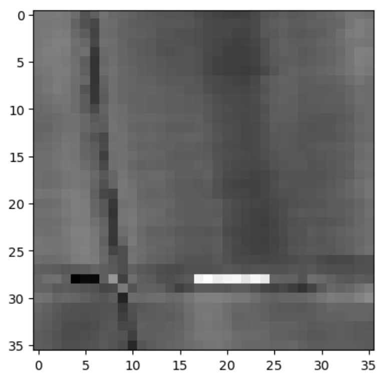
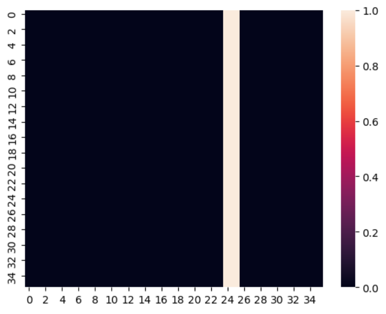
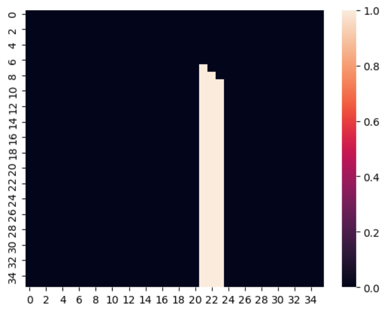

(c) Well 7, patch 1343

Figure 3.2: Examples of images


(a) Mask for well 2, patch 456


(b) Mask for well 3, patch 1169

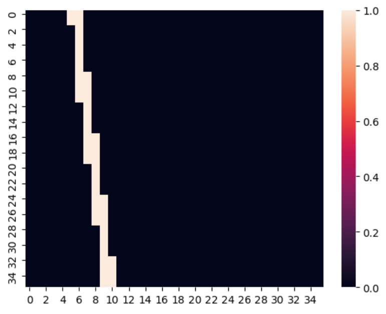

(c) Mask for well 7, patch 1343

Figure 3.3: Corresponding examples of masks

3.1 Train & Validation data:

The first 15 wells' dimensions are outlined as follows:

- Well 1: 3008 x 72 – vertical groove
- Well 2: 1088 x 36 – vertical discontinuous groove
- Well 3: 63892 x 36 – vertical discontinuous groove
- Well 4: 2136 x 36 – vertical groove
- Well 5: 1774 x 72 - vertical groove
- Well 6: 17364 x 72 – diagonal groove
- Well 7: 29800 x 72 - vertical and diagonal groove
- Well 8: 1424 x 72 – diagonal groove
- Well 9: 3016 x 72 – no groove
- Well 10: 1470 x 72 – no groove
- Well 11: 54457 x 36 – vertical and diagonal groove
- Well 12: 1593 x 36 - vertical groove
- Well 13: 68580 x 36 – vertical discontinuous groove
- Well 14: 12800 x 36 – diagonal groove
- Well 15: 7328 x 36 – diagonal groove

The images being really big, they were already cut them into smaller square patches of size 36×36 pixels. Each image is saved using the following name convention `well_<well_id>_patch_<patch_id>.npy`. The labels are binary images of size also 36×36, also are saved in a `csv` file.

Chapter 4

Methodology

In this chapter I will write about the approaches I used in tackling this challenge and the way I built my Convolutional Neural Network (CNN).

4.1 Preprocessing

I started off the project with some very simple preprocessing. I sticked to it, it was giving solid results without the need for more elaborate processing. I replaced the `nan` values with zeros.

Additionally, I standardized the data using `RobustScaler` from the `sklearn` library, since there were two of outlier values among the pixels.

4.1.1 Data Augmentation

Since I want my CNN to learn the patterns of defects and not the specificity of the instances in the dataset, I used data augmentation to enrich the dataset, namely the horizontal and vertical flip, like in the Figure 4.1. This is a proven way of making CNNs better.

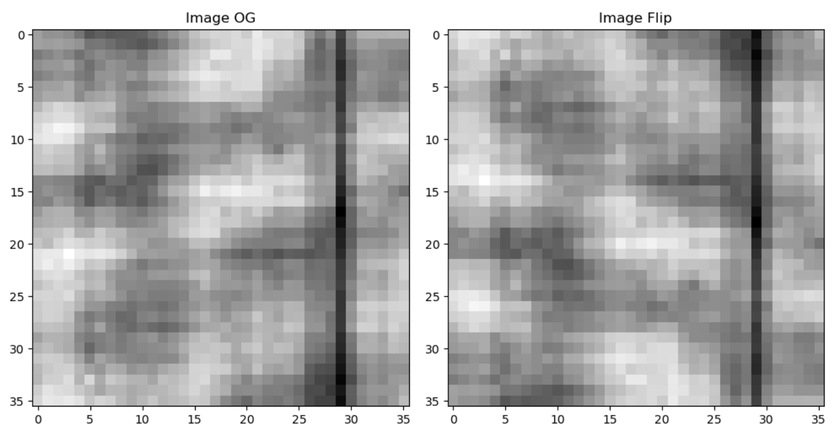


Figure 4.1: Example of a vertical flip of an image

Both augmentations were applied to all of the image-label pairs in the datasets, increasing the number of training examples to **29022**.

4.2 Network Architecture

The architecture of my CNN was inspired by the UNet network [1]. It was introduced by researches at University of Freiburg. They managed to segment brain tumors in MRI images with incredible accuracy marking an exceptional turning point in the field of semantic segmentation. You can see the diagram of their architecture in Figure 4.2. It consists of several down-sampling steps and corresponding number of up-sampling steps.

Given that their inputs were 512×512 and the mask of 388×388, adaptation was necessary. The revolutionary part the paper introduced were skip connections. Outputs of earlier layers are concatenated to later layers. This reduce the problem of vanishing gradient while also "remembering" features extracted in up-sampling layers.

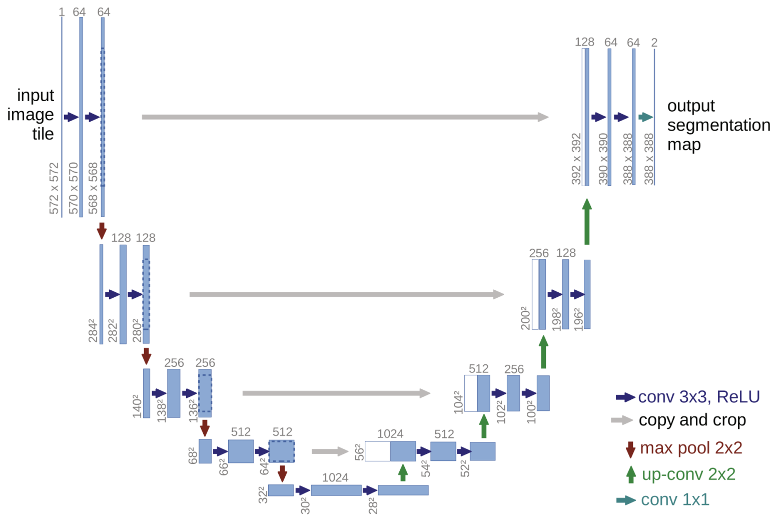


Figure 4.2: U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

First major difference between my network and UNet is that I dropped maxpooling in favor of a 3×3 convolution that halved the last two dimensions of the input. Since I was starting with a much smaller image then the researchers in Freiburg, I considered I was loosing valuable information by just simply dropping 3/4 of "pixels". Hence I combined dimension reduction with feature extraction into a convolution layer.

For the initialization of weights, I used the Kaiming normal distribution, which is shown to have the best performance when using ReLU as activation functions in the context of CNN.

The final architecture I decided on can be seen in Figure 4.3

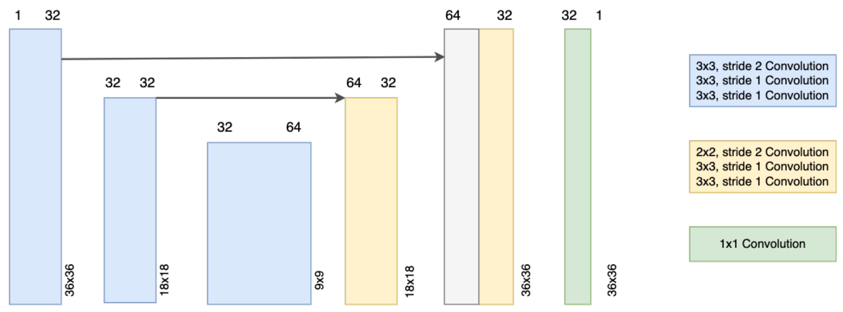

3x3, stride 2 Convolution
3x3, stride 1 Convolution
3x3, stride 1 Convolution
2x2, stride 2 Convolution
3x3, stride 1 Convolution
3x3, stride 1 Convolution
1x1 Convolution
36x36
18x18
9x9

Figure 4.3: The architecture of my CNN. Blue and green parts have three layers of convolution with ReLU activation after each, while the output layer (green) only has one convolution, compressing 32 channels into 1.

4.3 Chronology

Led by the idea of continuous incremental improvement, I started the project with a very basic network and built on top of that.

First I started with only:

- 5 layer CNN
- imputation of `NaN` values
- `RobustScaler`

The layers were in the sequence:

1. Input 3×3 convolution giving 36×36
2. 2×2 MaxPool giving 18×18
3. Downsample 3×3 convolution giving 18×18
4. Transpose 2×2 convolution giving 36×36
5. A 3×3 convolution giving 36×36 single channel output

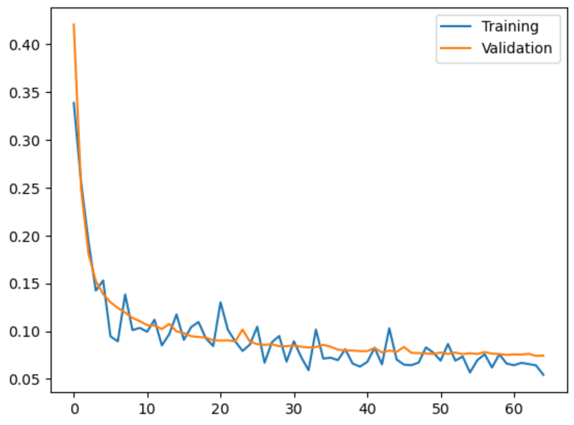
On the last layer's output, I applied a sigmoid activation, getting the final output. Layers 1 to 4 had ReLU activation. This setup resulted in the score of 0.399 on the ChallengeData platform leadboard.

Next major improvement was data augmentation, which increased the score to 0.416. Additional increase came from adding one more layer, resulting in the score to 0.454. After that, adding one more convolution layer at each level improved it to 0.562. Finally, the third layer at each level pushed the score to the highest achieved one. You can see all the submissions' score in the annex.

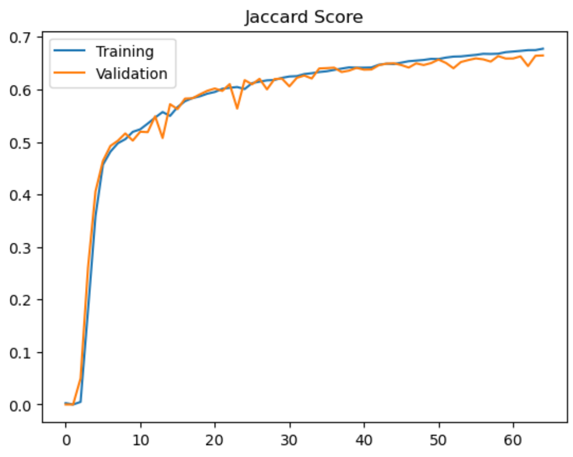
Chapter 5

Results

The score of the final version of the network was **0,669** granting me the fourth rank on the public leadboard at the time of writing of this report, present in the annex. On Figure 5.1 you can see the learning curves.



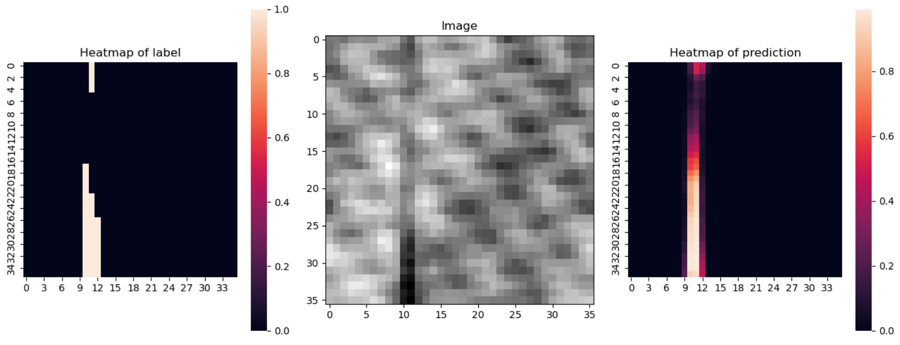
(a) The loss function



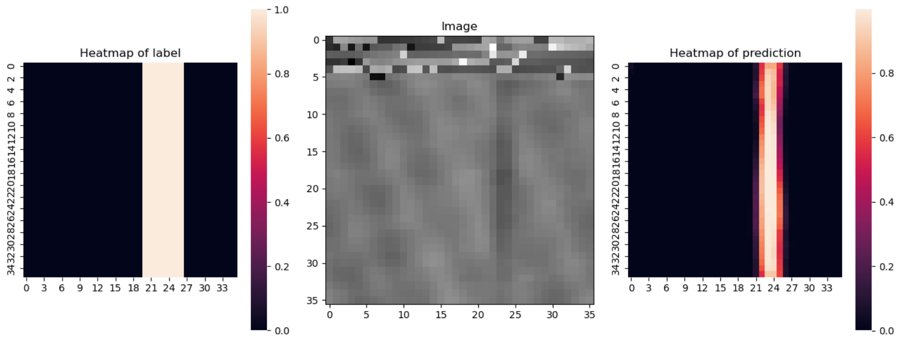
(b) The estimated Jaccard Index

Figure 5.1: Learning curves

On Figure 5.2 you can see two examples of input images, ground truth and the model prediction.



(a)



(b)

Figure 5.2: Examples of predictions

Chapter 6

Conclusion

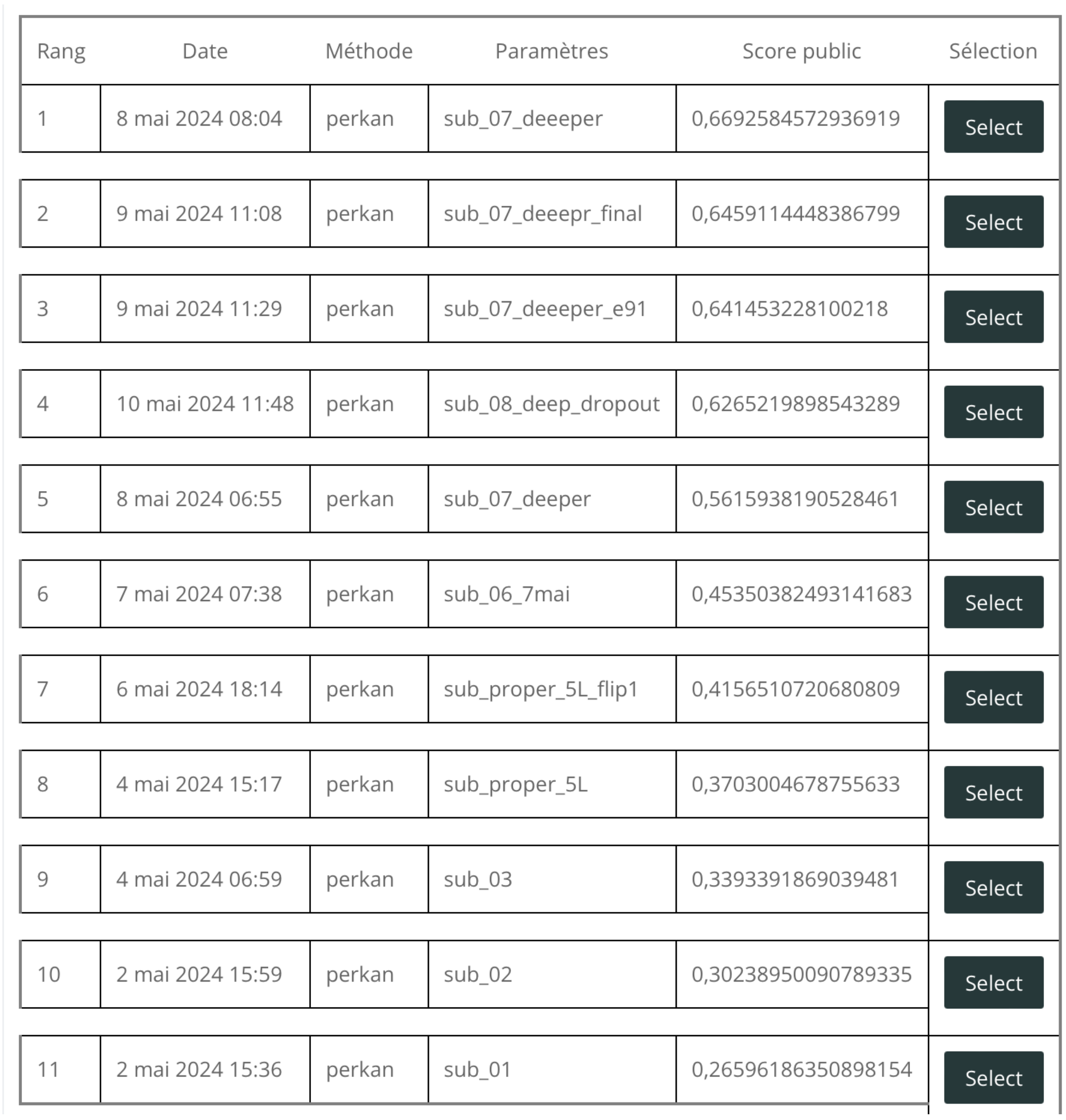
Comparing the score of the same model and data but with different number of epochs is consistent with what we know about when to stop training the model. As soon as the validation and training jaccard score start diverging, the performance starts dropping.

What was somewhat surprising was that omitting max-pooling turned out to affect the performance positively. I was also shocked about the drop in the performance in the moment of introduction of batch normalization, but in retrospect, the model being simple considering the challenge at hand, normalizing made no sense. There wasn't any overfitting to cure in the first place. But, personally, the most surprising occurrence was the increase in the leadboard score after making same "level" layers have triple convolution.

There are several changes that should be tested that could lead to a better performing algorithm. I omitted processing outlier pixel and patches in my approach. Furthermore, experimenting more with data augmentation, which would require some more creativity and skill. For example, applying rotations. But, this would entail implications on the transformations of labels. Considering the data is only 36×36 pixels in resolution, rotations could make big changes to "edge" pixels. Hence, this way of augmentation is not as straightforward here as in some other types of problems. Next, one frontier is the dropout technique, i.e. setting some weight at certain iterations to 0 with a certain probability p. This forces the nodes of the network to better independently generalize the patterns instead of over-fitting the training set. Initial attempts with $p = 0.2$ showed subpar performance compared to the model without dropout, so further testing might be necessary. Finally, the post-processing step. I used a fixed threshold of 0.5. Testing different ones might have improved the Jaccard Index of the predictions.

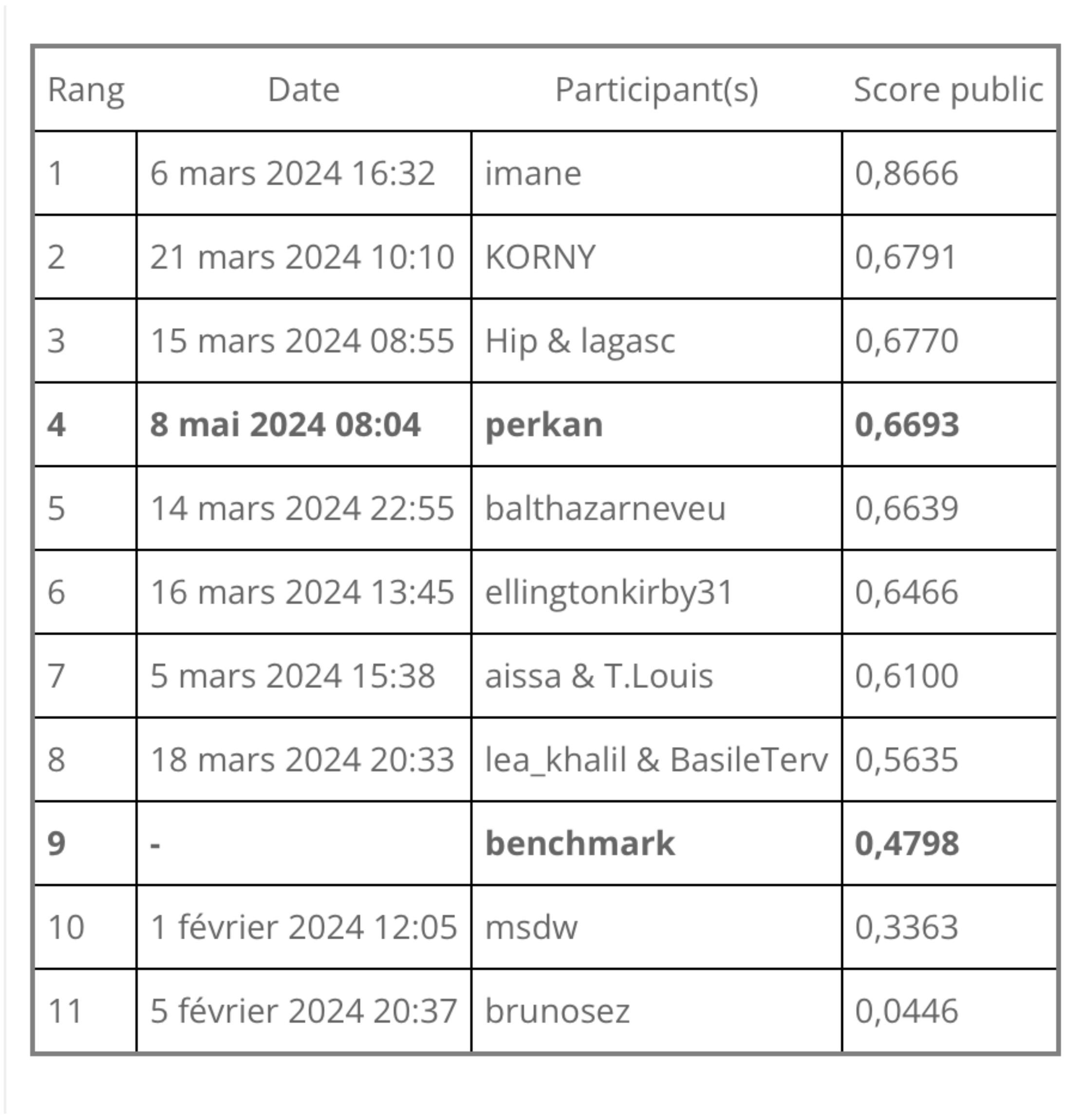
All in all, considering this was my first time modelling and training a neural network, I am very satisfied with the final result. This project has substantially increased my confidence in approaching AI in medicine as a possible future professional path.

Annexe 1



Rang	Date	Méthode	Paramètres	Score public	Sélection
1	8 mai 2024 08:04	perkan	sub_07_deeeper	0,6692584572936919	Select
2	9 mai 2024 11:08	perkan	sub_07_deeepr_final	0,6459114448386799	Select
3	9 mai 2024 11:29	perkan	sub_07_deeeper_e91	0,641453228100218	Select
4	10 mai 2024 11:48	perkan	sub_08_deep_dropout	0,6265219898543289	Select
5	8 mai 2024 06:55	perkan	sub_07_deeper	0,5615938190528461	Select
6	7 mai 2024 07:38	perkan	sub_06_7mai	0,45350382493141683	Select
7	6 mai 2024 18:14	perkan	sub_proper_5L_flip1	0,4156510720680809	Select
8	4 mai 2024 15:17	perkan	sub_proper_5L	0,3703004678755633	Select
9	4 mai 2024 06:59	perkan	sub_03	0,3393391869039481	Select
10	2 mai 2024 15:59	perkan	sub_02	0,30238950090789335	Select
11	2 mai 2024 15:36	perkan	sub_01	0,26596186350898154	Select

Figure 6.1: Submissions



Rang	Date	Participant(s)	Score public
1	6 mars 2024 16:32	imane	0,8666
2	21 mars 2024 10:10	KORNY	0,6791
3	15 mars 2024 08:55	Hip & lagasc	0,6770
4	**8 mai 2024 08:04**	**perkan**	**0,6693**
5	14 mars 2024 22:55	balthazarneveu	0,6639
6	16 mars 2024 13:45	ellingtonkirby31	0,6466
7	5 mars 2024 15:38	aissa & T.Louis	0,6100
8	18 mars 2024 20:33	lea_khalil & BasileTerv	0,5635
9	**-**	**benchmark**	**0,4798**
10	1 février 2024 12:05	msdw	0,3363
11	5 février 2024 20:37	brunosez	0,0446

Figure 6.2: Leadboard

Bibliography

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