Deep learning for lesion detection on CHUV cardiology images, in collaboration with CIS

Hugo Thiallier, Paul Margain, Elodie Raes
Mentor: Dorina Thanou
Department of Computer Science, EPFL, Switzerland

***Abstract*—For the last decades, technology found its way into the medical field. Different machine learning algorithms are helping the medical staff on a daily basis. Earlier and more complete detection of culprit regions using Computer Aided Detection (CAD) algorithms can lead to a more specialised care plan and prevent life-threatening events in the future. In this study, multiple CAD algorithms are studied and developed to have an early detection of culprit regions in the arteries on cardiology images obtained by CHUV in Lausanne, Switzerland.**

I. Introduction

Computer aided detection is a technology that is designed to observe certain regions on medical images. By doing so, the detection rate is increasing and thus the false negative rate decreases. With strong algorithms, this will result in early detection of affected tissues and more complete analysis of the image. In this project, the focus is on cardiology images obtained by the 'Centre Hospitalier Universitaire Vaudios' (CHUV) in Lausanne, Switzerland. Those images visualise the arteries near the heart in patients that have cardiovascular diseases. The detection of more narrow arteries is very essential to predict if a myocardial infarction will occur in the future and to prevent a myocardial infarction by treating the patient prematurely.

In this project, the computer aided detection is studied by several different approaches. First of all, the narrowed regions were detected using regression models with Convolutional Neural Networks. Secondly, a UNet architecture is used for the medical image segmentation of the narrowed regions. Both of which will be analysed and compared on performance and limitations. Finally, future goals and methods and goals are discussed.

II. Data processing

A. Data visualisations

The data set contains two kind of images: the raw images from the different patients and the labelled images from the same patients. The labelled images are actually the raw images but provided with green and red dots on the regions where a lesion is located. The green dots represent a lesion that will not lead to a myocardial infarction while a red dot represents a culprit region that will lead to a myocardial infarction. Figure1 shows an example of the raw and labelled images used.

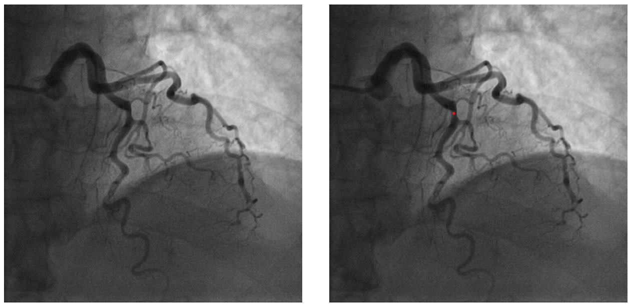
Fig. 1. Left: Raw image; Right: Labelled image

B. Input and target images

In the first part of the project, the respective colour of the dots were ignored and they were all considered as lesions in general. For the creation of a Neural Network, a target image has to be designed. When a raw image is sent into the Neural Network, a binary mask must be the output of the neural network. A binary mask is needed because it would be ideal to get a black output pixel when no lesion is detected for the input pixel and a white output pixel when a legion is detected in that region of the input image. This will lead to a binary output image, like shown in Figure 2. To obtain this, binary images must be created to feed the model during the training stage, because the strategy of detection is an example of supervised machine learning.

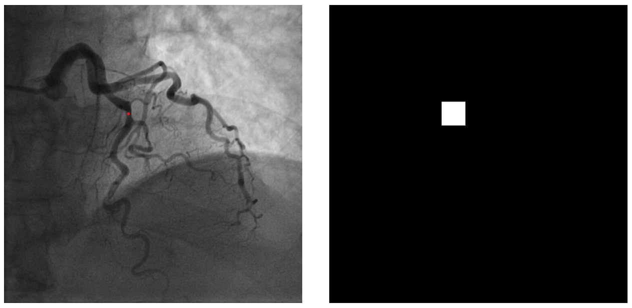
Fig. 2. Left: Labelled image; Right: Binary mask (target image)

C. Data augmentation

Because the original data set only contained images from 379 patients, data augmentation was needed to obtain good result using machine learning. First of all, the data set was enlarged by taking only 1 dot per image. As some of the images contained 4 dots, after the transformation this leaded to 4 different images instead of 1. Secondly, some basic data augmentation algorithms were applied. Some examples used are: Vertical flip, horizontal flip, image rotation, random RGB-values, elastic deformation, etc. Those data augmentations were added in the dataloaders function so that this leads to random data augmentations taken in the batch for training. This leads to an infinite amount of data, coming from the 369 useful initial images. The same data augmentations had to be applied to the raw images as to the labelled ones, so that the corresponding input and target images were describing the same augmented image. Figure 3 shows the augmented raw image and target image of the image shown in Figure 1. The augmentation shown is the horizontal flip and only 1 dot per image...

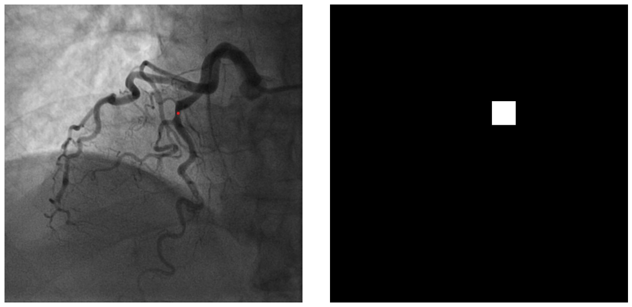
Fig. 3. Left: Labelled augmented image; Right: Augmented target

III. Model and Methods

A. UNet

For finding the culprit regions on the images, image segmentation methods are used. Different segmentation methods exist but they all have the same purpose: to highlight pixels that look like they might be part of a culprit region. They perform per-pixel labelling, which is exactly what is needed for our project. One of the known architectures for that, is the UNet architecture. It's name is derivated from the U-shape of the architecture, as seen in Figure 4.

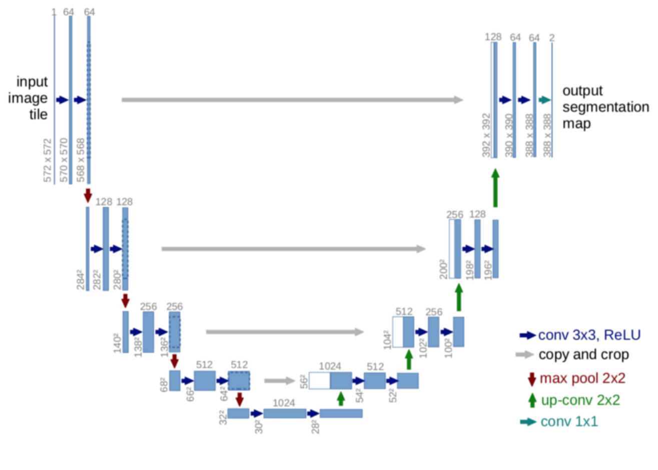

Fig. 4. The visual representation of the UNet architecture [1]

When detecting the culprit regions, also a lot of non-culprit pixels will be present in the image. This is why every pixel should be studied on its own. Since segmentation wants to have a pixel-wise detection, the input image must be go through a series of downscaling convolutions so that the network operations can be done for different image sections (sub-images with smaller dimensions). [2]

The output has the be of the same dimension as the input image. This is obtained by again going through a series of upscaling convolutions. Also, some padding must be done so that no pixels are lost off the edges of the image. After running through the UNet, the image in the upper left (input) must be of the same dimension as the image on the upper right (target/output).

Training the UNet has to be done by feeding a large amount of raw data and target images to the model. The target images are binary, with a while box where the culprit region is present and black pixels where the culprit region is not present. Those target images are constructed before, as described in Section II-B.

For training, the loss function used was the loss function based on the Dice coefficient. The Dice coefficient is a measure of overlap between two samples. The coefficient is one when there's complete overlap between the target image and the output image and between zero and one when there's partial overlap. This is very useful for binary data (like the target image used in the UNet). Due to the binary target mask, any pixels that are out of the prediction range are ”zeroed-out” which makes the calculation very efficient. Equation 1 shows how the Dice coefficient is calculated. [3]

$$Dice = \frac{2|A \cap B|}{|A| + |B|} \tag{1}$$

For updating the weights of the neural network, a stochastic gradient descent optimization algorithm was also implemented in the model. In that case, a single learning rate (α) is maintained for all weight updates and the learning rate doesn't change during the training.

Another approach used for updating the weights of the neural network, is the Adam optimization algorithm. This optimization algorithm doesn't have a constant learning rate, but instead, Adam also makes use of the average of the second moments of the gradients. Parameters of the Adam configuration are the learning rate α, β_1 as the exponential decay rate for the first moment (i.e. the mean) estimates, β_2 for the second-moment estimates (i.e. the uncentered variance) and ϵ to prevent division by zero. This optimizer is supposed to perform better than the SGD one for computer vision problems. [4]

B. Regression CNN

Second approach to detect lesions in each image was to train a regressive neural network with convolution layers and then linear ones, in order to get the pixel coordinates of the lesion in outputs. Regression is mainly used to detect box coordinate around a particular pattern in an image, but for this problem, only one point coordinate was needed (the center of the gaussian heatmap around pixels of a lesion) to point out the lesion directly and not an area around it, as it could be done by a specialist in real life. The overall design of this NN is displayed in Figure 5, with a first convolution sequence (one max pool and one conv2D layers with kernel of 3), then a bigger convolution sequence containing conv2D layers with ReLU activation functions, and finally (after flattening) linear layers sequence enables to get the coordinates as an output.

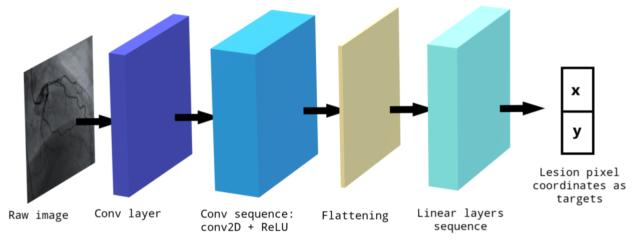
x
y
Lesion pixel coordinates as targets
Raw image
Conv layer
Conv sequence: conv2D + ReLU
Flattening
Linear layers sequence

Fig. 5. Regressive neural network structure

Through the convolution layers, the goal was to extract the features from the input image, so basically the first convolution layer will give the features such as edges or colours while the other layers will capture more high-level features. The pooling applied after first convolution enables to get the dominant features and reduce the computational power. When the features are detected, the results is first flattened, so a linear function can be applied to finally obtain the coordinates targeted.

Unlike the UNet method, the regression CNN requires to use a more geometrical loss function to assess the coordinate prediction, therefore the Mean Squared Error loss commonly used for such coordinates detection was implemented. However, the optimizers tested were the same as the one tested for UNet.

IV. Results

To begin with, we did not managed to obtain a successfully trained neural network. However, we tried and learned several different algorithms and implementations that are going to be explained below.

A. *UNet results*

At first the target were composed of the culprit regions extracted from the data-set augmented. This regions could have been multiple on each images as it can be on this typical target:

So we train our UNet with the raw images as input with the associated targets. With different loss functions (Dice Loss, CrossEntropyLoss, BCEWithLogitsLoss) and range of learning rate (1 to 10e-5) we could not achieve to have decrease of the loss function. The training took around 15 min per epoch so several hours in total for each. Because of this high time cost, we decided to simplify the training to get

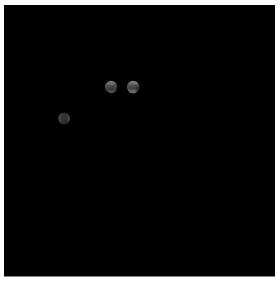

Fig. 6. Typical original Unet target (first version), with 3 culprit regions [1]

positive results and get back then to the original problem. The target was simplified with only one binarized culprit region on the target as we can see below:

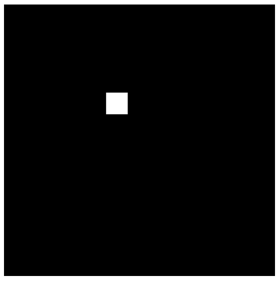

Fig. 7. Typical simplified Unet target (second version), with one culprit region [1]

Moreover this unique culprit region allow us to add images virtually to our dataset. With only one input-target couple left in our dataset we tried to over-fit our Unet with different learning rate.

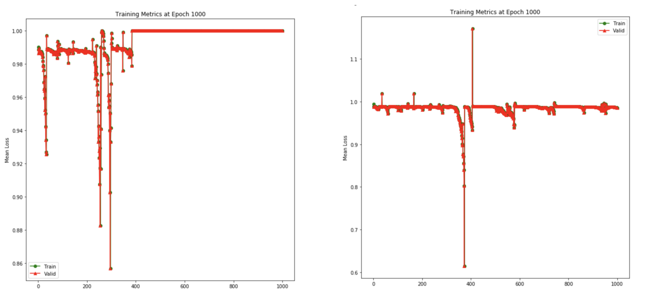

Fig. 8. Left: Training with DiceLoss; Right: Training with MSELoss

In Figure 8, the results for two different loss functions (DiceLoss and MSELoss, learning rate =10e-3) shows that the training is not operating correctly : after several iterations the loss function stays constant. The output corresponding is a black image instead of the target images we wanted. In order to correct that, we changed the UNet implementation and carefully changed some of the hyper-parameters of our code but without significant improvement.

B. *Regression results*

As explained in method part the targets are coordinates of lesion pixels while the inputs are the raw image of our data set. First in order to check the validity of our neural net architecture we tried to perform overfitting by training on a single image and its corresponding target. This gave confident result since we obtained a converging MSE loss, tending to 0, as displayed in Figure 9. Conventionally a validation metrics as well as the training metrics should be displayed to assess the validity of the model, but it appears that despite using the same image for validation no good loss were obtained, hence it has not been taken into account for plotting.

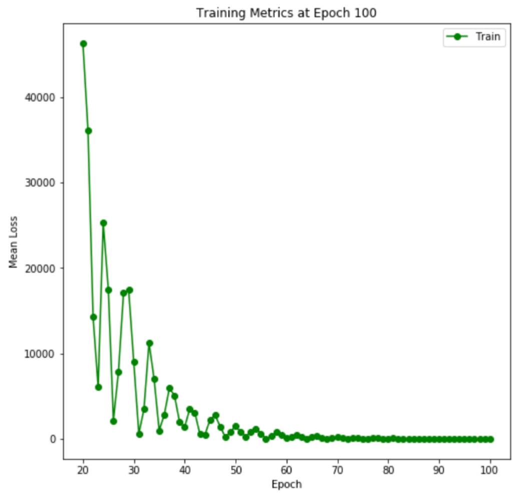

Fig. 9. MSE mean loss obtained for each epoch

If the over-fitting test was promising, it appeared that when using all images and their targets (711 each) the loss tends to decrease, but after few epochs it stabilises at a quite high value (at least 15000). Some hyper-parameters tuning were done like decreasing the learning rate or using different settings in the convolution layers and linear layers, but it still gave too high losses.

As a final attempt we tried to use 11000 images/coordinates combination from the augmented data, but once again it gave a very high training loss, decreasing at first, then stabilising around 15000. Moreover when printing the outputs from this trained model applied either on training or testing images, it seemed that the predicted coordinates were always the same no matter the inputs. Therefore the model is likely returning a mean or median value across all possible outputs.

V. Discussion

For the UNet implementation, after further discussion with our teaching assistant and supervisor, we may explain our unsuccessful training results because of the possible inadequacy of the UNet with this problem. UNet architectures are typically used in the biomedical imaging, however it is more about segmentation task that object localisation.

Concerning the regression implementation it is clear that some improvements could be done by using bigger data sets or by using coordinates of pixel boxes around the culprit regions of all images instead of only the ones of the pixels in the middle of these regions. This last method could be really convenient as it is commonly used and especially for the model described here, that mostly comes from a work on localising pixel boxes coordinates around a particular object [5], even if the main challenge and novelty of this implementation was to localise a small pixel region and directly get coordinates of the pixel in the middle of this region.

Another and final assumption that could be interesting is about the data set that was used for this work and its viability to undergo such training tasks. The fact is that our images have a grey hue and the culprit region seems quite hard to identify especially for our neural nets. Then it could be great to work on the contrast of each image in order to get more perceptible edges of the vessels and so the culprit regions. With such contrasted images it could be easier for the model to perform segmentation and the convolutions will be much more efficient. The vessel detection could also be achieved with Convolutional Neural Networks.

VI. Summary

The aim of this project was to study different methods to obtain an algorithm to detect culprit regions in the vessels of cardiographical images. There for, pytorch implementations of UNet and CNN with regression were created and trained. After multiple hyper-parameters tunings and different choices of loss functions and optimizers, we could conclude that the UNet architecture is not ideal for our quite complex problem. More pre-processing should be done or different architectures should be implemented like ResNet.

Even if the outcome of this project is not completely successful, it has been a positive experience for us. Indeed, with our multiple tries and thanks to the helpful discussions with our supervisor Dr Dorina Thanou and Teaching Assistant Guillermo Ortiz Jimenez we have learned about different Deep Learning techniques which complete well the theoretical knowledge of the Machine Learning EPFL course.

References

[1] "UNet — Line by Line Explanation. Example UNet Implementation — by Jeremy Zhang — Towards Data Science." [Online]. Available: https://towardsdatascience.com/unet-line-by-line-explanation-9b191c76baf5
[2] E. Stevens, L. Antiga, and T. Viehmann, "Deep Learning with PyTorch," Tech. Rep.
[3] "An overview of semantic image segmentation." [Online]. Available: https://www.jeremyjordan.me/semantic-segmentation/
[4] P. L. Lagari, L. H. Tsoukalas, and I. E. Lagaris, "Variance Counterbalancing for Stochastic Large-scale Learning," *International Journal on Artificial Intelligence Tools*, vol. 29, no. 5, aug 2020.
[5] "Object Detection and Localization with Deep Networks." [Online]. Available: https://engineering.purdue.edu/DeepLearn/pdf-kak/week8.pdf