UNIVERSITAT DE BARCELONA

Interpretability of deep learning methods in carotid artery image classification and semantic segmentation

Authors
Christopher FADY
Susana CASTELA

Supervisor
Laura IGUAL

January 2020

Abstract

Medical imagery is arguably one of the clearest use cases for Deep Neural Networks: automatic detection of illnesses as an additional guidance tool could massively help doctors in their everyday work. However, the nature of the field makes errors extremely costly. That means that understanding the reasons for the decisions made by any Deep Learning model is absolutely crucial to its implementation and use.

Previous work has demonstrated the effectiveness of Deep Learning methods applied to the detection of atherosclerotic plaques in carotid arteries. These are a known factor in cardio-vascular diseases, and can be identified by measuring Intima Media Thickness (IMT).

To the best of our knowledge, these effective models, such as CNNs, VGG and Tiramisu (U-Net type), have not been studied under the lens of interpretability in this context. Our goal is to study the classification and segmentation decisions from the models in order to determine how they were taken, and if they make sense when compared to medical knowledge.

For this purpose, we propose to use previously studied interpretability techniques, mainly Grid Saliency and the well-documented SHAP values, both of which are adapted to Deep Learning models. Indeed, both of these methods attempt to study local decisions from the models, while having the advantage of being model agnostic and visual in the representation. This makes them easy to understand in general with minimal explanations, a clear advantage when one of the goals is to help medical personnel, as well as data scientists, in making better and faster decisions.

This study is applied to a dataset of ultrasound images, REGICOR. This work is framed within a larger research project within the UB, which has already spawned various works. This makes the dataset well suited for the purpose of interpreting the results from our models.

Acknowledgements

First and foremost, a huge thank you to our thesis supervisor, Dr Laura Igual, for her suggestion of this interesting topic, her guidance, her many insightful ideas, and most of all her patience. It was a pleasure to explore the topic of medical imagery with interpretability, and we hope our work will be useful for many others. We would also like to thank Maria del Mar Vila, Beatriz Remeseiro Lopez and Daniel Salvador Urgel for their availability and promptness in helping us understand the datasets.

In addition to that, we extend our gratitude to our many professors in the Foundations of Data Science Masters at the UB, who gave us the tools we needed for this work. In particular, Dr Jordi Vitrià and his Deep Learning course, with whom we worked on SHAP values in the context of Pneumonia chest Xrays, and which served as an inspiration for some of the work here.

Finally, we both would like to thank our current employers, for their flexibility and support during this masters.

Contents

1 Introduction

1.1 Problem Statement

Being able to humanly understand decision taken by Deep learning model ”black boxes” is ever more important as these models become more powerful. Indeed, we know the model took a correct decision, but did it look at the ”right” parameters to arrive at that decision? As such, can we trust that the model will take the right decision again when applied to a new dataset? This conundrum is even more critical in the context of medical imagery, when a correct decision from a model may mean a life or death situation.

With that context in mind, we focus on the effectiveness and interpretability of Deep Learning approaches for the detection of atherosclerosis. This specific disease consists of plaque building up inside the arteries (figure 1) and is a known symptom of cardio-vascular disease.

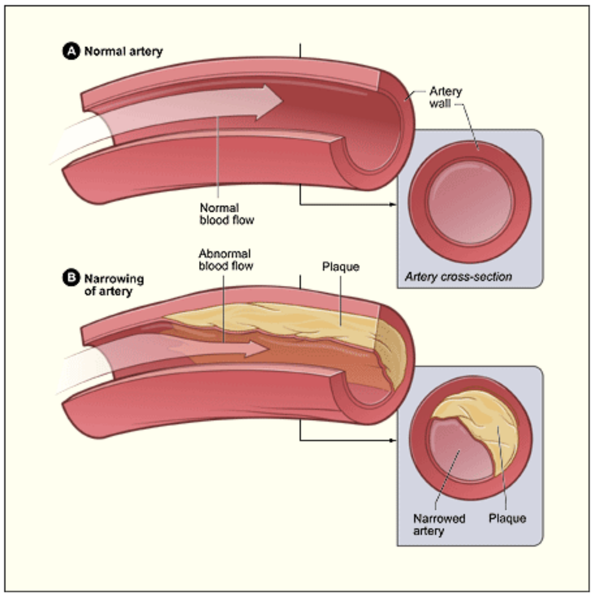


Figure A shows a normal artery with normal blood flow. The inset image shows a cross-section of a normal artery. Figure B shows an artery with plaque buildup. The inset image shows a cross-section of an artery with plaque buildup.

Figure 1: Atherosclerosis, Source [19]: US department of Health & Human Services

In a classic case, a qualified doctor will identify the risk by measuring the Intima-media thickness (IMT) on carotid artery ultrasound images, which will usually correlate with atherosclerosis (figure 2). If the maximum IMT is over 1.5mm [7], then the patient is considered to have plaque.

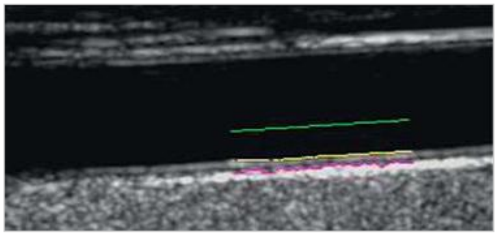


Figure 1. Intima-media thickness (IMT) definition - IMT is measured as the distance between lumen-intima (yellow line) and media-adventitia (pink line) interfaces.

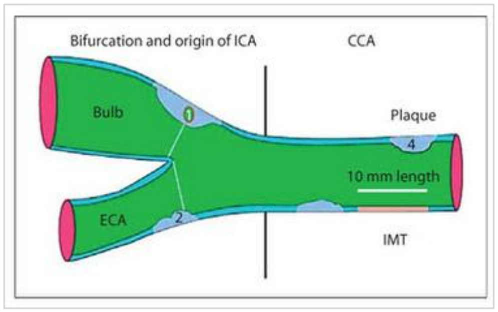


Figure 2. Proper location for IMT measurement

Figure 2: Measuring IMT, Source [20]: European Society of Cardiology

The question then becomes: are we able to automate this detection using Deep Learning methods? And if so, does the plaque detection actually rely on the right "parts of the image"?

1.2 Dataset and Previous Work

We build upon the previous work of our Universitat de Barcelona colleagues [1], who were able to generate segmentation labels using a Deep Learning (U-net) network, and did considerable work on automation of both plaque classification and segmentation. They also demonstrated the good performance of a CNN for plaque classification, and suggested that specific grey pixels may play a strong role in the classification. This opens up the possibility for interpretability of the work, which we will explore with both SHAP values (further defined in section 2.4.5) and Grid Saliency methods (see section 2.5).

We will use an extensive ultrasound image dataset from REGICOR [25], which includes:

- Ultrasound images of carotid arteries with IMT Ground Truth (GT), and their associated labels (see below in figure 3).
- The result of a previous semantic segmentation, with either 2 labels (IMT region and background), or 6 labels (Near Wall, Lumen-Bulb, Lumen-Center, IMT-Bulb, IMT-Center, Far Wall).

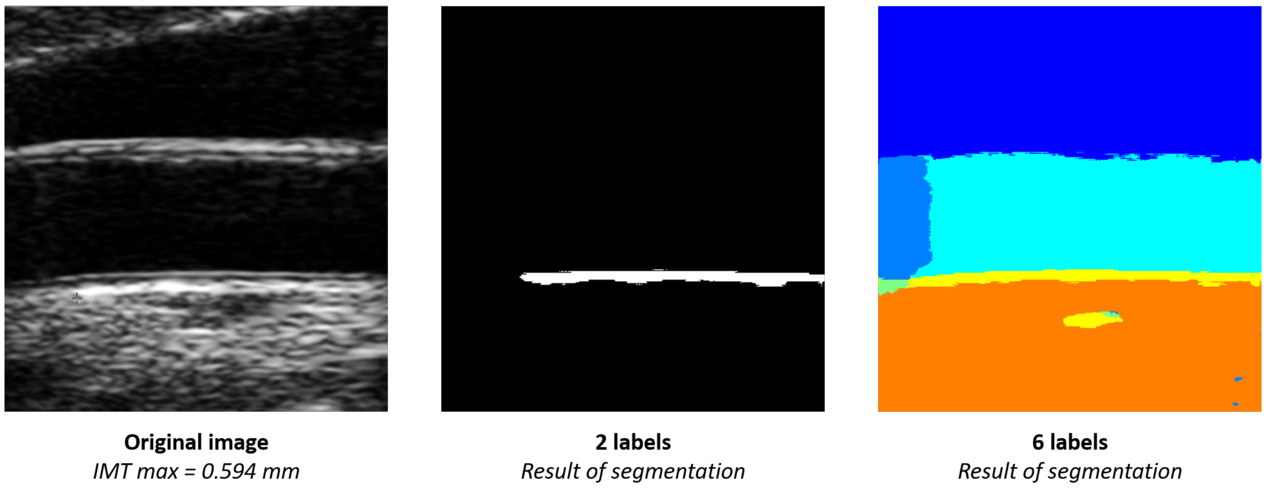


Figure 3: REGICOR Image dataset

Our goal is then to take this work into the field of interpretability, to understand the decisions, good or bad, taken by the networks, and to further validate the Deep Learning approach in the context of detecting atherosclerosis.

2 State of the art

2.1 Importance of interpretability

As the importance and human dependence on machine learning models and systems increases, it also increases the need to understand the reasoning behind the decisions taken by said systems. A large part of this field is summarized below, using as basis the book "Interpretable Machine Learning" [11].

The main reasons as to why interpretability is becoming so important are the following:

- **Human learning:** In contrast to black box models, interpretable machine learning models allow humans to understand why certain decisions are being made, satisfy their curiosity, and even expand their knowledge of the subject.
- **Safety and testing:** It is specially important to know why certain classifications or decisions are being made when the model has a delicate or important purpose. Knowing the reason behind its responses allows us to react and solve problems when we detect them.
- **Detecting bias:** Since machine learning models display the biases existing in the data used to train them, it is important to detect them and correct them to avoid having, for example, a system that discriminates against certain ethnic groups.

There are of course exceptional cases where interpretability is not so important or even not recommended.

We do not need interpretability when **the purpose of the model is simple or well studied**, like, as an example, a movie recommender system. Having a bad prediction in this case, will not cause any major problems or harm to society or individuals.

Another exception happens when we are **working with well studied problems**: we can argue that if a model has been thoroughly explored, studied and used, there is no need to add further interpretability methods: it should be already clear that they work and how they do so. An example of this would be optical character recognition (OCR systems).

Adding interpretability methods or explanations to a model, could cause users that have this information to take advantage of such a privileged information to **manipulate the results.** If we use proxies for causal features, users can "cheat" on these proxies in order to obtain a different outcome from the machine learning model. An example of this could be a bank system that helps decide if a customer should be granted a credit line or not based on the services he or she is using, such as credit cards. The customer could simply change the number of credit cards he or she has contracted, in order to obtain a different score.

2.2 Taxonomy of interpretability methods

We can arrange interpretability methods into four different classifications:

- **According to the characteristics of the algorithm**
  - Intrinsic: Methods that restrict the complexity of the machine learning model and that rely on its simplicity, such as decision trees and linear models.
  - Post hoc: Interpretability methods that are applied to the model after it is trained. They can also be applied to intrinsically interpretable models.
- **According to the results of the interpretation method**
  - Feature summary statistic and visualization: Methods that provide a value for each feature, which represents its importance or interaction with other features. The best way to convey their importance is via visualization of these statistics, for example in partial dependence plots.
  - Model internals: Model learned specifications that are sufficient to understand decisions (i. e. intrinsic interpretability methods)
  - Data point: This family of methods provides data points to make examples that help understand the results of the model, for example, exceptions or prototypes.

    - Intrinsically interpretable model: An interesting way to find explanations about black box models is to approximate them with an intrinsically interpretable model, and use the parameters of said intrinsic model as explanations for the black box one.

- **According to the target of the method**
    - Model specific: Model specific methods are only applicable to certain model classes (intrinsic methods fall into this category).
    - Model agnostic: This category of explanations can be applied to any kind of machine learning algorithm. It usually works by comparing input-output sets.

- **According to the scope of the method**
    - Local: Methods that explain individual predictions.
    - Global: The opposite, methods that explain the entire model behaviour.

We can see the taxonomy along with the main methods described and used in the following sections in figure 4.

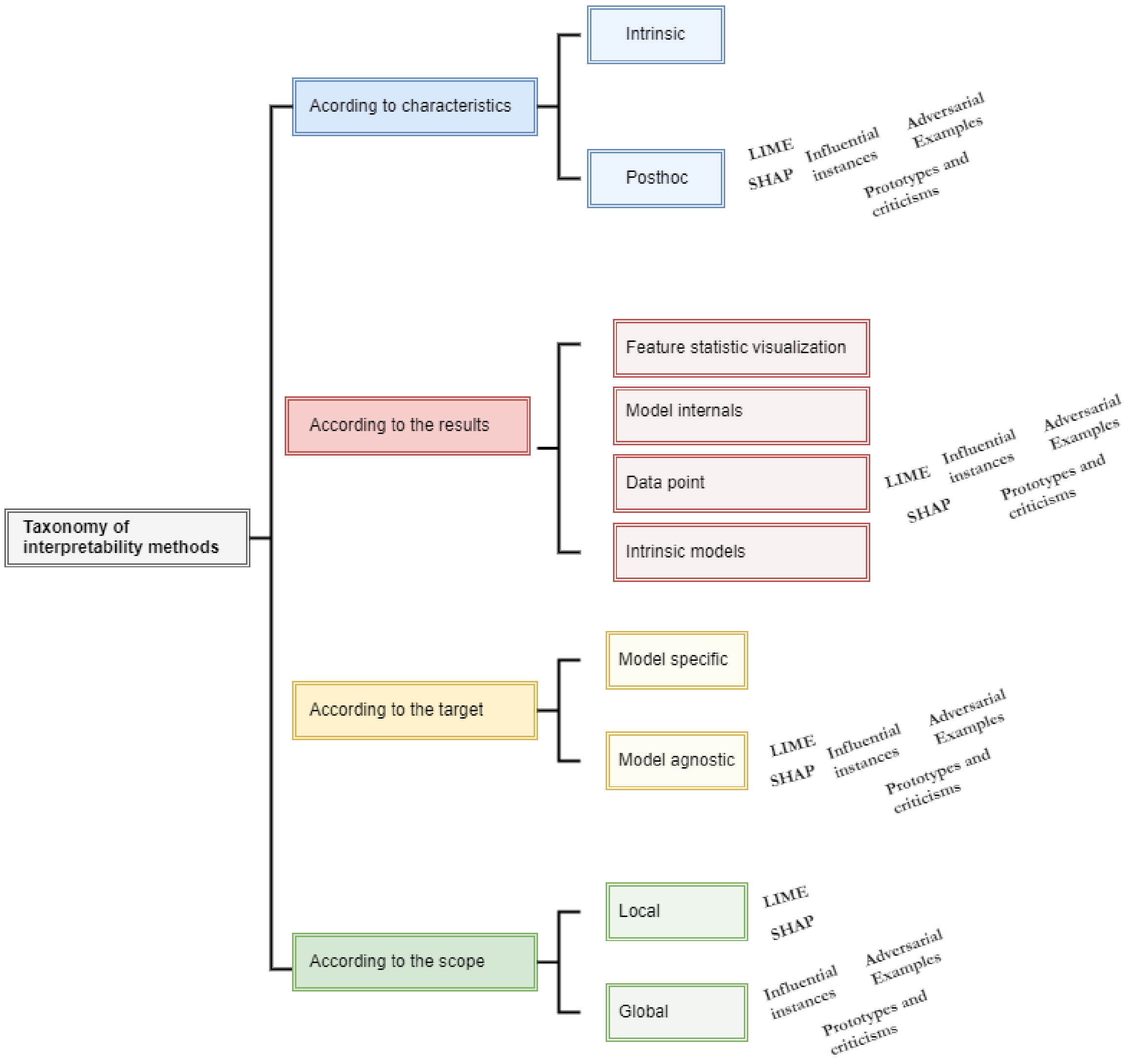


Figure 4: Taxonomy of interpretability methods, with some examples seen in the subsequent pages

2.3 Desired properties of explanations

When we are using interpretability methods to obtain explanations of model decisions, there are a set of properties we aim to obtain in order to have an interpretation method that is humanly understandable. These properties are the following:

- **Contrastiveness:** Contrastive explanations are those that explain decisions in contrast to other decisions, i.e. how would the output have changed if the input had been different. The key concept in contrastive explanations is that they do not have to be complete: this way they are more simple to understand.
- **Selectiveness:** Explanations that do not cover all the causes for a prediction: only 1 to 3 reasons is better than to have all the features involved.
- **Social:** It is important that explanations take into account their audience. Social science experts can help adapt the message extracted from interpretability methods.
- **Focus on the abnormal:** Rare values for features that contribute to a certain prediction, should by all means be included in the explanation, even if they contribute the same as other features with normal values. The usual occurrence is that the removal of an abnormal value would yield a completely different outcome.
- **Truthfulness:** Explanations are sustained in occurrences outside of the scope of the training of the machine learning model. Truthfulness is also referred as **fidelity**.
- **Consistence with prior knowledge:** As humans tend to disregard information that is not consistent with their prior beliefs, it is desirable that explanations conform to such previous knowledge of the matter.
- **General and probable:** Explanations that are able to give meaning to many events. They complement the cases where there are no abnormal reasons for the predictions, and avoid using probabilities of joint events, which are often misjudged.

2.4 Interpretability methods applicable to image recognition

The focus of this project is interpreting the results of black box neural networks for semantic segmentation and classification of images, so in this section we will have a look at the state of the art of interpretability methods that are applicable to this kind of task.

2.4.1 Adversarial Examples

Adversarial examples are instances with their features perturbed in a way that makes the machine learning model make a false prediction. Their main aim is precisely to deceive the model.

When working with images, adversarial examples are other images with intentionally perturbed pixels with the aim of obtaining a completely different prediction. In order to achieve this there are several methods. If we have access to the underlying model, we can manipulate the image adding a small perturbation based on the sign of the gradient in some of the pixels. These kind of "attacks" work with LSTMs, maxout networks, ReLU activated networks...

If we don't have access to the internals of the model, we can train a surrogate model to have the same outputs, find adversarial examples for the surrogate model and test if they also work for the original one.

The goal of finding these adversarial examples is to see if our model is classifying based on reasonable features or not, and also to re-introduce these instances into the model training, to avoid misclassification, as shown in figure 5.

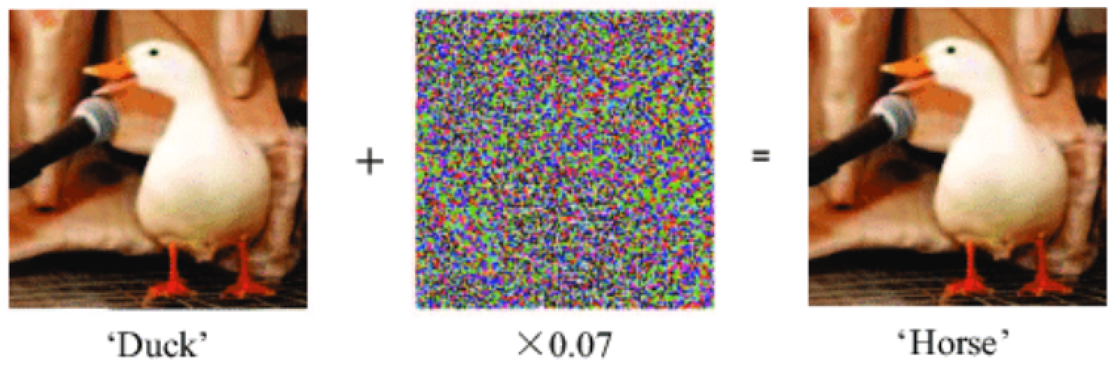


Figure 5: Introducing what looks like random noise, the result of the image classifier is completely different.

2.4.2 Prototypes and criticisms

Prototypes and criticisms are an example-based family of explanation methods. These methods aim to select, among all the data used by a black box system, some data instances representative of the data (prototypes) and on the opposite side, data points not represented by the set of prototypes (criticisms) .

They are selected among data instances to cover the center of the data distribution (prototypes) and points in clusters without a prototype (criticism). The selection can be performed manually or with the help of an algorithm such as Maximum Mean Discrepancy or MMD [13].

When working with image data, algorithms like MMD provides us with a set of images that represent the classes and a set of images that challenge the classification task, as we can see in figure 6.

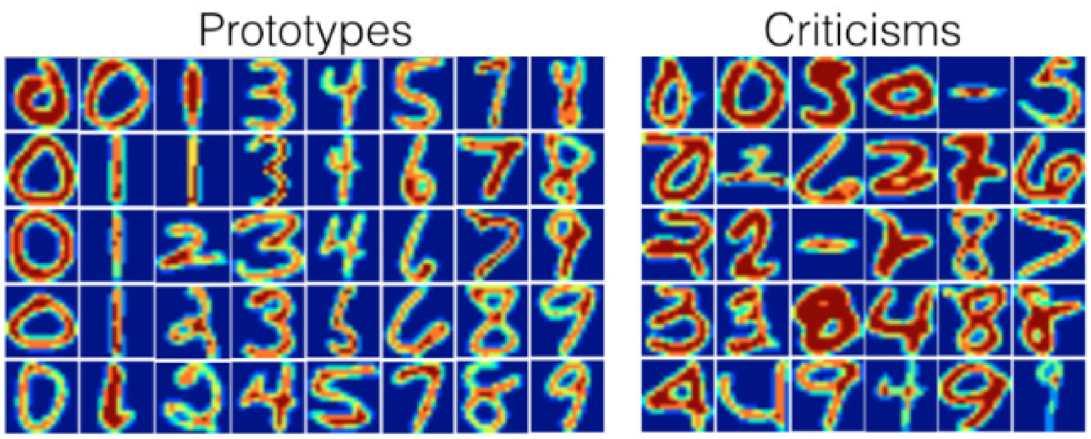


Figure 6: Prototypes and criticisms in an OCR system. Prototypes are the points that represent their classes while criticisms are far away from them.

2.4.3 Influential instances

Like prototypes and criticisms, influential instances are example based explanations. In this case, the goal is to search for the instances that are the most influential towards the black box system's result.

By influential instance, we mean a data sample whose removal from the training data would cause a significant change in the outcome of the model.

2.4.4 Local Surrogate (LIME)

LIME stands for local, interpretable model-agnostic explanations. This method aims to give explanations to a black box model based on individual predictions by means of a surrogate model [11].

LIME assumes we have an unlimited number of attempts to try the output of the model. In order to understand why certain predictions are made, a dataset of permutations of a certain input is created, and all the outputs for the permutations are obtained. Then, an intrinsically interpretable model is trained on this dataset. This surrogate model is a local approximation of the predictions of the original one.

When dealing with image data, it makes little sense to randomly permutate individual pixels. What is done instead is the following: the image gets segmented into "superpixels" that are turned "on" or "off" in each permutation. Superpixels are interconnected pixels with similar colors.

In figure 7 we can see an example of a LIME explanation: for a given image, groups of superpixels are turned on or off to give an explanation for 3 possible classifications of an image.

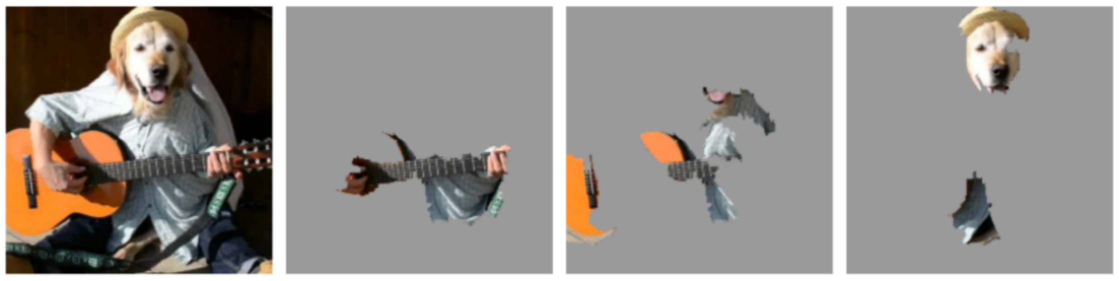

(a) Original Image (b) Explaining *Electric guitar* (c) Explaining *Acoustic guitar* (d) Explaining *Labrador*

Figure 7: LIME explanations for 3 different categories for a same image.

A difficulty when using LIME is to find the right region for the superpixels, which can only be inferred by testing and visualizing the explanations.

2.4.5 Shapley Values and SHAP

Shapley values have their origin in game theory. The idea behind them is to treat each feature as a player that has a contribution to the model prediction.

Players (or features) cooperate and receive a profit from it. In this case the game (profit) is "reaching a better prediction" and the gain is the difference between the actual prediction for an instance and the average prediction for all instances.

The Shapley value is defined as the marginal contribution of each feature across all possible coalitions of a feature. In order to compute the Shapley value for a feature, we take a coalition of the other features and sample for different values of the target feature. The difference between the predictions when changing the value of this feature, averaged for several samplings, will be the Shapley value for this feature, when taking into account all possible coalitions, as show below in figure 8.

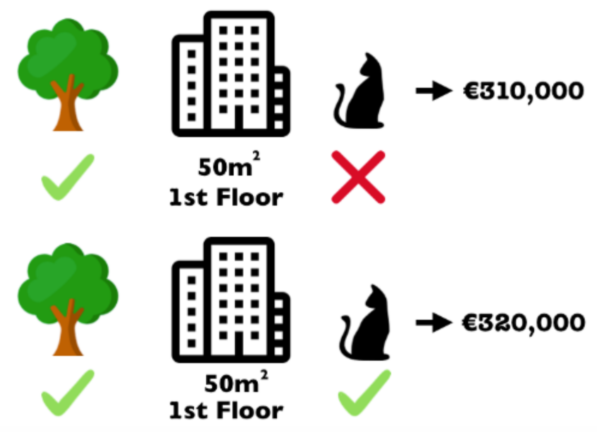


Figure 8: Contribution of Cat banned on the apartment price, by looking at one combination of features (area 50, 1st floor and park-nearby)

Features are in some cases grouped together. For images this means that pixels get grouped into superpixels, and Shapley values are estimated for those groups instead.

The issue with Shapley values is that as new features get added, the number of possible combinations increases exponentially. Indeed, in the vast majority of real world problems, only the approximate solution is feasible. To solve this, Strumbelj et al. [14] propose an approximation with Monte Carlo sampling $\hat{\phi}_j = \frac{1}{M}\sum_{m=1}^{M}(\hat{f}(x_{+j}^m) - \hat{f}(x_{-j}^m))$ at each feature j. The differences to the black box are then averaged and result in $\phi_j(x) = \frac{1}{M}\sum_{m=1}^{M}\phi_j^m$. This is then repeated for each feature to get Shapley values.

Going further, **SHAP (SHapley Additive exPlanations [6]), is a method based on Shapley values, which will be key to the current thesis document**. In SHAP the values are represented via a linear model, whereas the weight in the Shapley values is given by a coalition of features. This difference connects LIME and SHAP values.

It keeps the main properties of Shapley values, with the "value explanation" detailed as $g(z') = \phi_0 + \sum_{j=1}^{M}\phi_j z'_j$, where g is the explanation model, $z' \in \{0,1\}^M$ is the coalition vector or simplified features, M the max coalition size and $\phi_j \in \mathbb{R}$ is the feature attribution for a feature j. These "simplified features" are, in the context of images, superpixels, groups of "similar" pixels that have the same "explanation".

This is done in 5 steps [11], as the model agnostic KernalSHAP:

- Sample the coalition z'_k
- Get the prediction for each z'_k by applying the model f, with $f(h_x(z') = E_{\times C}[f(x)]$
- Use the SHAP kernel $\pi_x(z') = \dfrac{M-1}{(M\ choose\ |z'|)|z'|(M-|z'|)}$ to compute the weight of each z'_k.
- Fit the linear model g using the data, targets and weights and optimizing a sum of squared errors loss $L(f, g, \pi_x) = \sum_{z' \in Z}[f(h_x(z')) - g(z')]^2 \pi_x(z')$
- The coefficients of this model, ϕ_j, are the Shapley values.

The main difference VS LIME is in the weights of z': in SHAP, small coalitions (few 1s) and large ones (many 1s) get the highest weights, as they are the most "explainable" (as half 1s and half 0s doesn't help to explain the impact of an individual feature).

As a more specific solution to Deep Learning models, the GradientExplainer from SHAP (see figure 9), which we will use later, expands the SHAP theory explained previously and is described as follows [5]: *"Expected gradients is an extension of the integrated gradients method [15], a feature attribution method designed for differentiable models based on an extension of Shapley values to infinite player games (Aumann-Shapley values). As an adaptation to make them approximate SHAP values, expected gradients reformulate the integral as an expectation and combine that expectation with sampling reference values from the background dataset. This leads to a single combined expectation of gradients that converges to attributions that sum to the difference between the expected model output and the current output".* The model is approximated with a linear function between each background data sample and the current input to be explained, and input features are assumed to be independent.

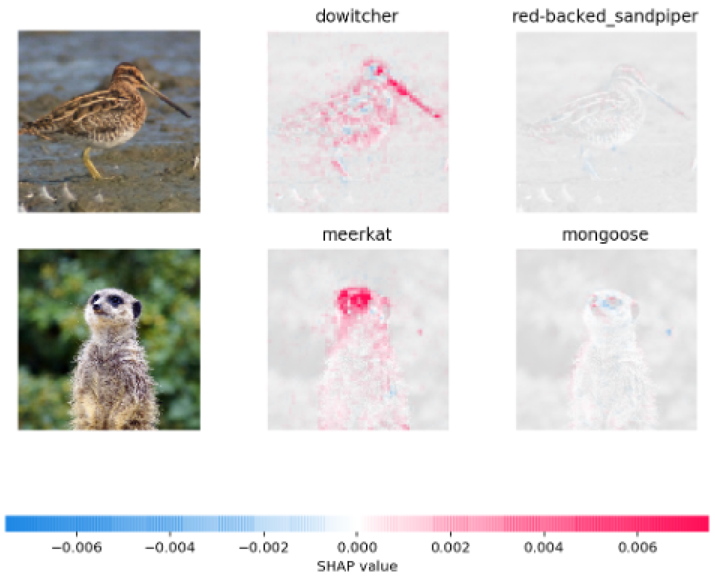


Figure 9: SHAP GradientExplainer example - How 7th intermediate layer of the VGG16 ImageNet model impacts the output probabilities

For a given class, the positive SHAP values (red pixels) increase the probability of that class, while negative SHAP values (blue pixels) reduce the probability of that class.

2.4.6 Deep Neural network (DNN) interpretation

When working with convolutional neural networks (CNN) and images, high level features are learned from the initial pixels in an image. At the end of the network, these high level features are passed through a highly connected layer to obtain a classification or a prediction.

One possible method for understanding why a DNN is making a certain classification is via **feature visualization** i.e. finding the input that maximizes the activation of a unit. Since looking at each neuron would take too long, usually channels are used for feature visualization.

Another approach is **network dissection** which quantifies the interpretability of a unit in a CNN by linking it with a human concept: objects, textures, colors, etc.

An approach for these types of decompositions suggested by literature is the **Deep Taylor decomposition**, as a means to explain non linear classifications [2]. It views each neuron of a DNN as a function that can be expanded and decomposed on its input variables, and are then aggregated and backpropagated: this results in a "Deep Taylor decomposition". The goal is to assign to each pixel a "relevance" score in $\mathbb{R}^+$ to explain a classification decision, via a heatmap R, with the following properties:

- *Conservative*, if the sum of assigned relevance to pixels corresponds to the total model relevance
- All values of the heatmap are *positive* (0 being the absence of relevance)
- Both of these definitions combine to describe the heatmap as *consistent*

The Deep Taylor is inspired by "Divide and Conquer", as the function learned by the DNN is decomposed into smaller subfunctions (either by connectivity or from training).

We can define 3 separate propagation rules: the w^2 rule in unconstrained input space, and the z^+ and z^β rules in constrained input space.

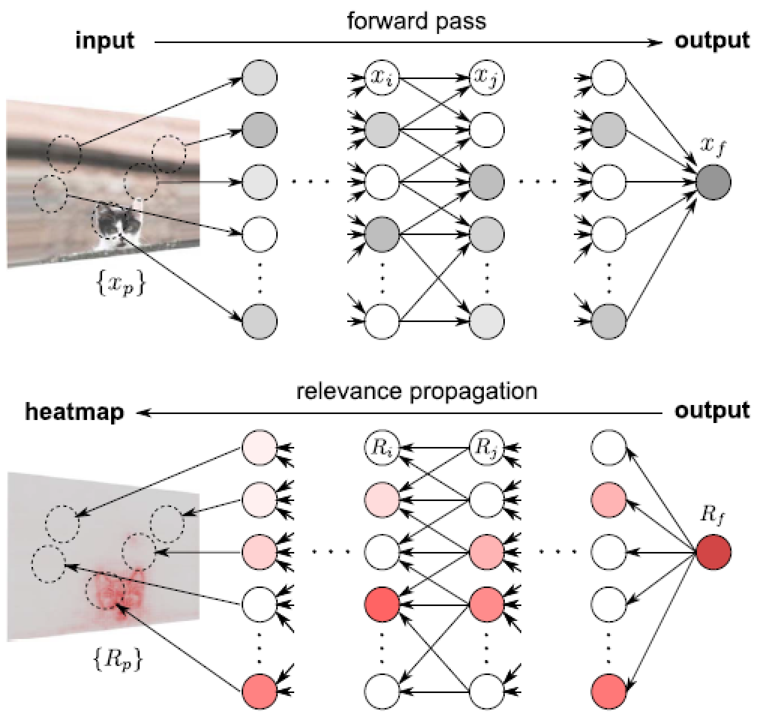


Figure 10: Computational flow of a Deep Taylor for the class "Cat"

The technique can be applied in 2 different models:

- *Min-Max relevance*: Enables to fully decompose the relevance of each input neuron, as a trainable relevance model (minimizing the mean square error).
- *Training-free relevance*: No need to train a relevance model for each neuron, the results are backpropagated onto the lower layers by modelling certain model parameters as constants. Suitable for very complex networks.

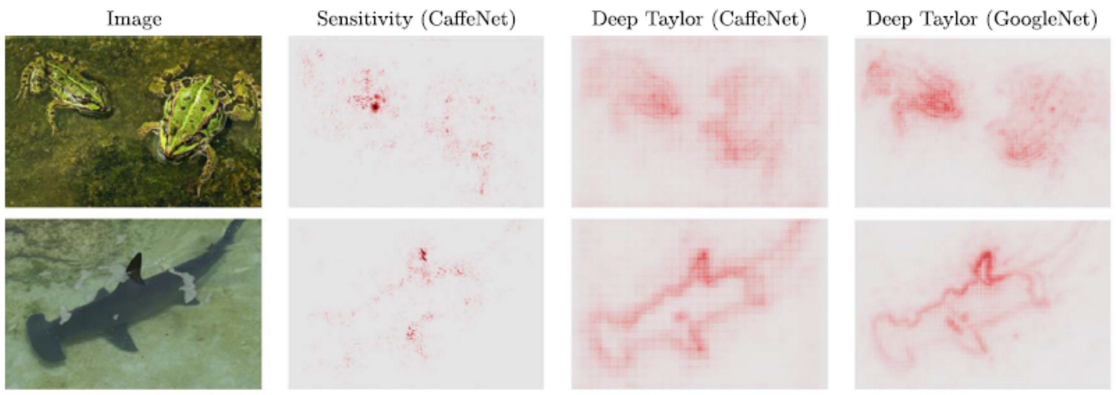


Figure 11: Results of training-free Deep Taylor using pre-existing models

The technique outperforms single Taylor and sensitivity analysis, and is also (relatively) model agnostic in producing relevant heatmaps. It helps interpretability of image recognition models, but should be applicable to a wide variety of DNNs.

This can be further expanded with Layer-wise relevance propagation (LRP), which focuses on post-hoc interpretability: understanding the results of a DNN rather than the algorithmic level [3]). Here, features that contribute to the decision can also be graded: in the case of pixels it can for example be a heatmap of pixels supporting the classification decision.

These backpropagation techniques, such as LRP, outperform Gradient or Single Taylor techniques as they rely on the structure of the network, progressively mapping the prediction to the lower layers until the input is reached (figure 12).

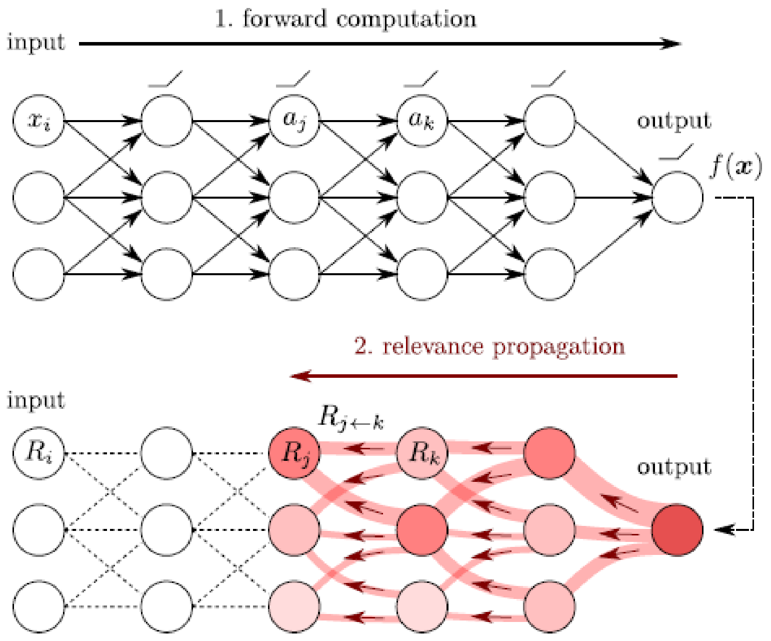


Figure 12: LRP procedure

LRP is deeply embedded in Deep Taylor decomposition, and has strong benchmark performance. The equation follows $\sum_{i=1}^{d} R_i = \cdots = \sum_j R_j = \sum_k R_k = \cdots = f(x)$, where R_k encodes the relevance of neuron k for the prediction $f(x)$. The propagation rule can be noted as $R_j = \sum_k (\alpha \frac{a_j W_{jk}^{+}}{\sum_j a_j w_{jk}^{+}} - \beta \frac{a_j W_{jk}^{-}}{\sum_j a_j w_{jk}^{-}}) R_k$. Setting $\alpha = 1, \beta = 0$ allows it to be interpreted as a Deep Taylor. This set up is proven to work well with images, and has strong benefits in terms of robustness.

Overall, to be explainable, a DNN is recommended to have the following characteristics (here in the context of a DNN having convolution layers and ReLU neurons):

- Low number of fully connected layers, and using dropouts
- Global sum pooling layer at the top of a DNN, and generally use multiple sum pooling layers
- Constrain biases to be 0 or negative

2.5 Grid Saliency

Grid saliency explanation methods aim to give explanations for image classification and image segmentation black boxes.

They are tools to explain predictions of a trained model by highlighting parts of the input that presumably have a high relevance for its predictions - in essence it identifies the image pixels that contribute the most to the network prediction.

Grid saliency methods were build and defined first for **image classification** [9] and on top of image classification, there have been implementations that extend image based saliency models towards **pixel-level dense prediction methods for semantic segmentation networks** [8].

In order to do so, these methods look for a mask that perturbs the images, blocking information from them. While trying to keep it as small as possible, the classification or segmentation score must be preserved for this new perturbed image.

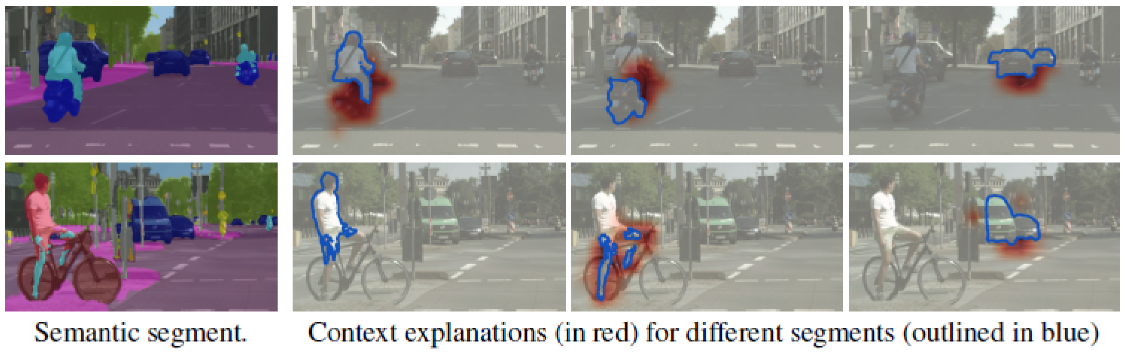

Semantic segment.
Context explanations (in red) for different segments (outlined in blue)

Figure 13: Context explanations for street scenes [8]

Overall, grid saliency is able to provide sensible and coherent explanations for network decision making, such as in the Cityscapes data, explaining bicycle mainly by sidewalk, car mainly by road, and rider by bicycle (figure 13). It is effective at detecting image bias, and identifying pre-labeled classes.

3 SHAP explanations for a plaque / no-plaque image classifier

Our goal will now be to use this research in the context of atherosclerosis: to validate Deep Learning approaches, including models previously tested, by the use of interpretability. Simply put, can a human understand the results, if we create the right networks and use the right techniques to showcase them? For this purpose, we first tackle the classification problem of our images.

Using SHAP, given we have a single binary class to explain (plaque / no plaque), the representation output is simple: the positive SHAP values (red pixels) increase the probability of plaque, while the negative ones (blue pixels) reduce it.

3.1 Methodology and data preparation

Goal: The goal of the experiment is to test the efficiency of a CNN classifier to obtain the final prediction - does the image indeed have plaque or not? In addition to that, we want to be sure that the classifier uses the information from the IMT region to make that prediction. For that purpose we will use SHAP, given it combines robust theory with simple to interpret visual results, which we can easily compare against our expectations.

Dataset: We use 4,731 images from the REGICOR database, and the result of the segmentation with 2 labels on those same images (see figure 3 for a reminder). These are the results of the segmentation work of the Tiramisu 67 U-net.

Data augmentation and preparation: Given the low amount of plaque images (exactly 60 of the 4,731 images), we need to run some data augmentation in order to train the classifier more reliably. The objective is to have at least 50% of plaque images out of the total.

For this, we combine 2 data augmentation strategies to which CNN networks are invariant:

1. Cropping and re-positioning of the data, going from (224,224) images to (205,205), with a randomized starting point

2. Vertical flips of the images, applied with a 50% chance in order to create noise

The cropping chosen is "small enough" to not affect the region where the IMT is calculated, while also providing enough variety to create new independent images. We apply the exact same transformations to the labels of the images to be able to use both image and label as inputs.

We also apply the same transformations (cropping and a random vertical flip) to the original images. If we didn't do that, the model would learn to recognize different image sizes and overfit, as shown in figure 14.

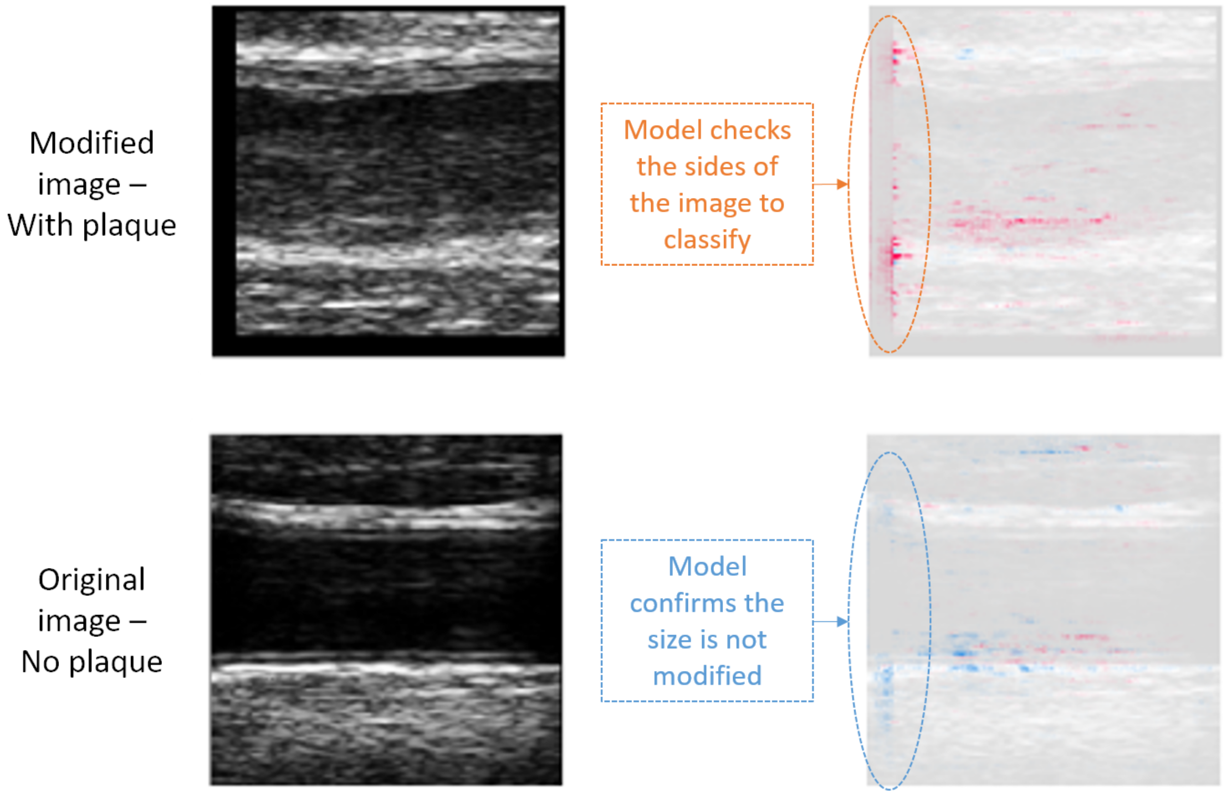

Modified image – With plaque
Model checks the sides of the image to classify
Original image – No plaque
Model confirms the size is not modified

Figure 14: The classifier will check the image sides to determine their sizes as if it were a parameter, clearly over-fitting

Finally, all the images are normalized to have pixel values between $[0, 1]$.

3.2 Network and training process

We conduct 2 experiments with different inputs:

- Experiment 1: Using as input the augmented and processed images
- Experiment 2: Using two images as input, the images and the label resulting from the segmentation, augmented and processed.

We create the network in figure 15 for Experiment 1, using a Keras model with Tensorflow backend. Experiment 2 uses exactly the same architecture: it runs the images and labels independently through the 2D convolutional and max pooling layers, then concatenates them, and finally runs the joint flattened image through the dense layers, as shown in figure 16.

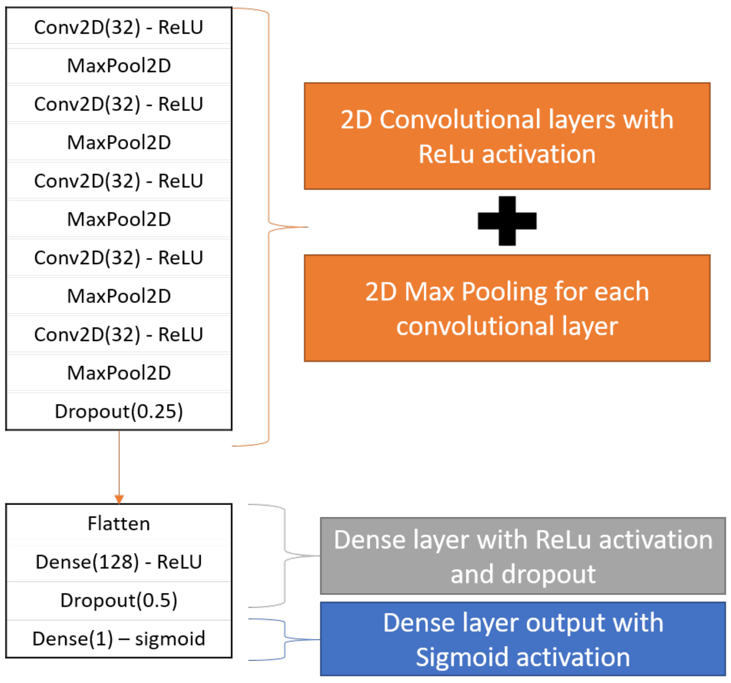


Figure 15: Classifier used for Experiment 1 - All Conv2D layers have (3x3) kernels with 32 filters, and the Max Pooling layers use a (2x2) stride

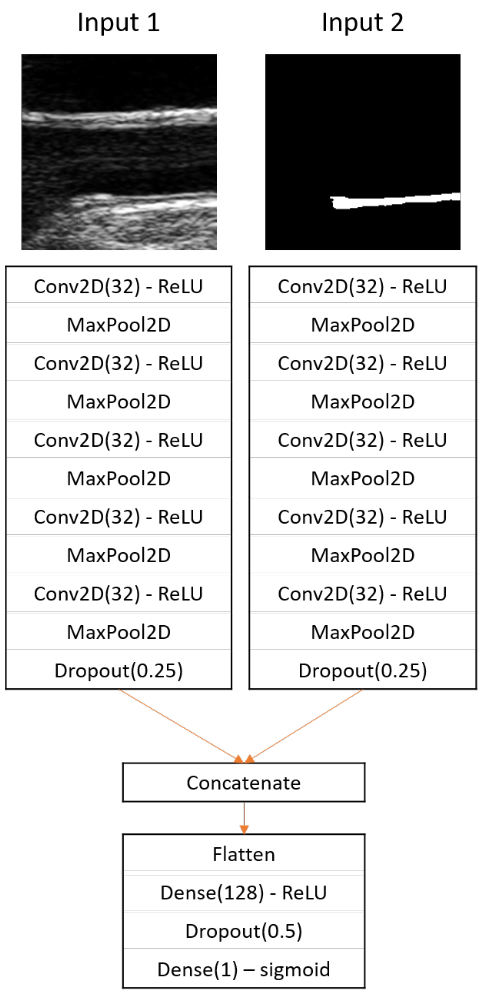


Figure 16: Classifier used for Experiment 2

Why did we make these choices for the model?

- We know that convolutional neural networks are particularly suited for Deep Learning applied to images [12].
- We have also confirmed in literature that multiple pooling layers are beneficial to interpretability: we use max pooling layers, both for ease of implementation in Keras and for their propensity to recognize patterns (which is our objective).

- Similarly, we use 2 fully connected layers (one of which being the output), and combine them with dropouts (after the convolutions / pooling).
- We use ReLU activations on all layers (convolutional or fully connected), as the most efficient function seen in literature for DNN applied to images [12].
- We are outputting the probability of a binary class, hence the use of a sigmoid activation on the last layer.

With that in mind, we can now set out to train our model on the augmented image dataset, with the following model training parameters:

- Accuracy to judge the model performance (given we are in a 0 - 1 prediction case)
- Adam optimizer, another classic for DNN optimization [17]
- Binary cross-entropy loss (as we are dealing with a binary class prediction, the probability of having 0 or 1): $-(ylog(p) + (1-y)log(1-p))$

We isolate 20% of the dataset for validation and test, and then 40% of this isolated set for pure test. We will then train over 15 epochs with mini-batches of 128. The training time is between 1 to 2 hours on a standard computer: the goal, again, is to test an "accessible" approach.

3.3 Classification results

We expect very bad results, and that we should try with another method, typically focusing on the segmentation.

The results are summarized in table 1:

Dataset	Model	Accuracy	Sensitivity	Specificity	Precision	F1 score
Augmented REGICOR 4000 9551 Images	Experiment 1	97.12%	100.0%	94.15%	94.65%	97.25%
Augmented REGICOR 4000 with labels 9371 Images / 9371 Labels	Experiment 2	98.13%	100.0%	96.26%	96.41%	98.17%

Table 1: Results of the models used - Using 1 or 2 images as input

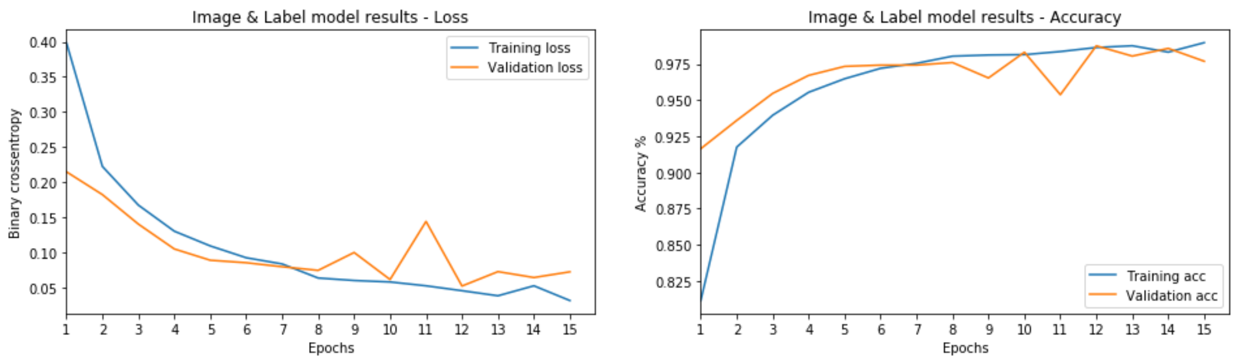


Figure 17: Training and validation loss for the model in Experiment 2

As we can see, the accuracy of the network is very strong, even in Experiment 1 that doesn't use the results of the segmentation. The results in Experiment 2 are slightly better, and it can be assumed that they would be more robust if applied to a different dataset.

Interestingly, the model will much more often misclassify an image as positive (i.e. "has plaque") than the opposite, as shown by the Sensitivity value. This is a good sign in a medical context, where the penalty for a FN (interpreting as no sickness when there is) is much higher than for a FP (sickness when there is none). In the current context of this work, the accuracy obtained is more than enough to explore the interpretability of results.

Indeed our main objective of this work is to determine if the network is using the "correct" sections of the images to take decisions, more than simply correctly classifying.

For this we display the SHAP values on the images to explain the classification decisions, as detailed previously (beginning of section 3). Our objective is to further validate our DNN approach for the interpretation of atherosclerosis detection in carotid medical imagery:

- For this, we use the "*GradientExplainer*" function (explained in section 2.4.5) from the Python SHAP library [5]: the results are then plotted, using a random sample of 100 images as background.
- This enables us to visualize, on test set images, groups of pixels that push to a

"positive decision" (there is plaque, in red) and those that push for a negative decision (there is no plaque, in blue).

- To know which images are misclassified, the image titles in the following examples are in the format *[Correct class].[Predicted class]*, with 0 meaning "no plaque" and 1 meaning "plaque".
- Note that since we applied a vertical flip randomly in the data augmentation process, in some cases the IMT region will be in the top part of the image (i.e. the image was flipped), and in other cases in the lower part.

We show both correct and incorrect classifications in figure 18 (Experiment 1) and 19 (Experiment 2).

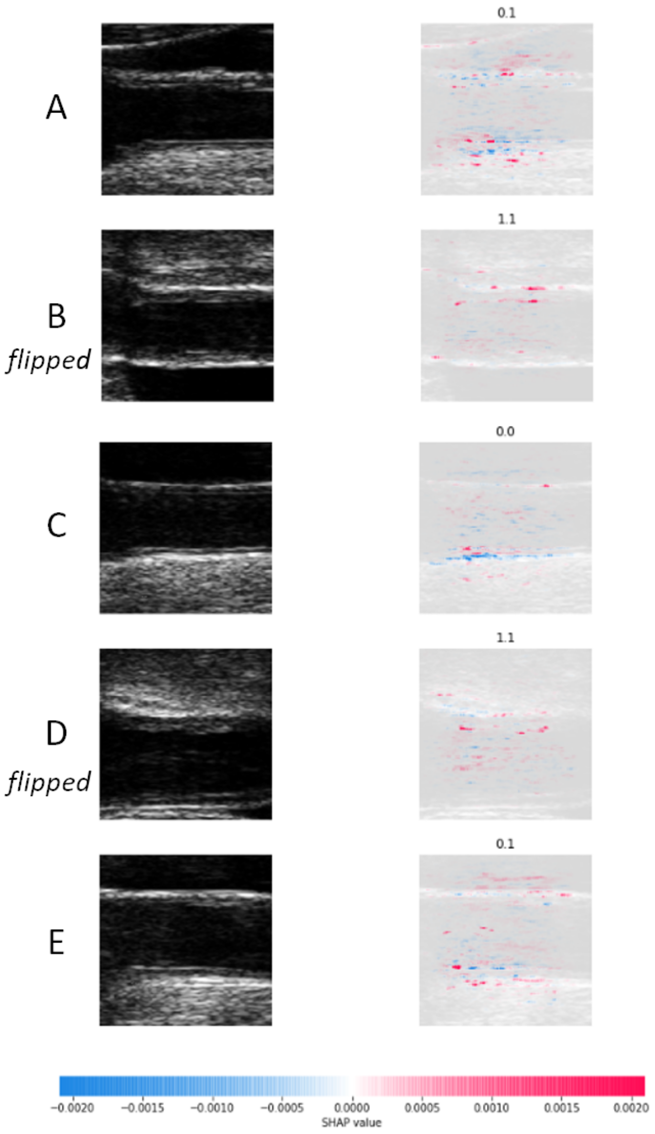


Figure 18: SHAP values in Experiment 1, including 2 misclassifications (images A and E): the interpretation focuses mainly on the inner layer of the carotid

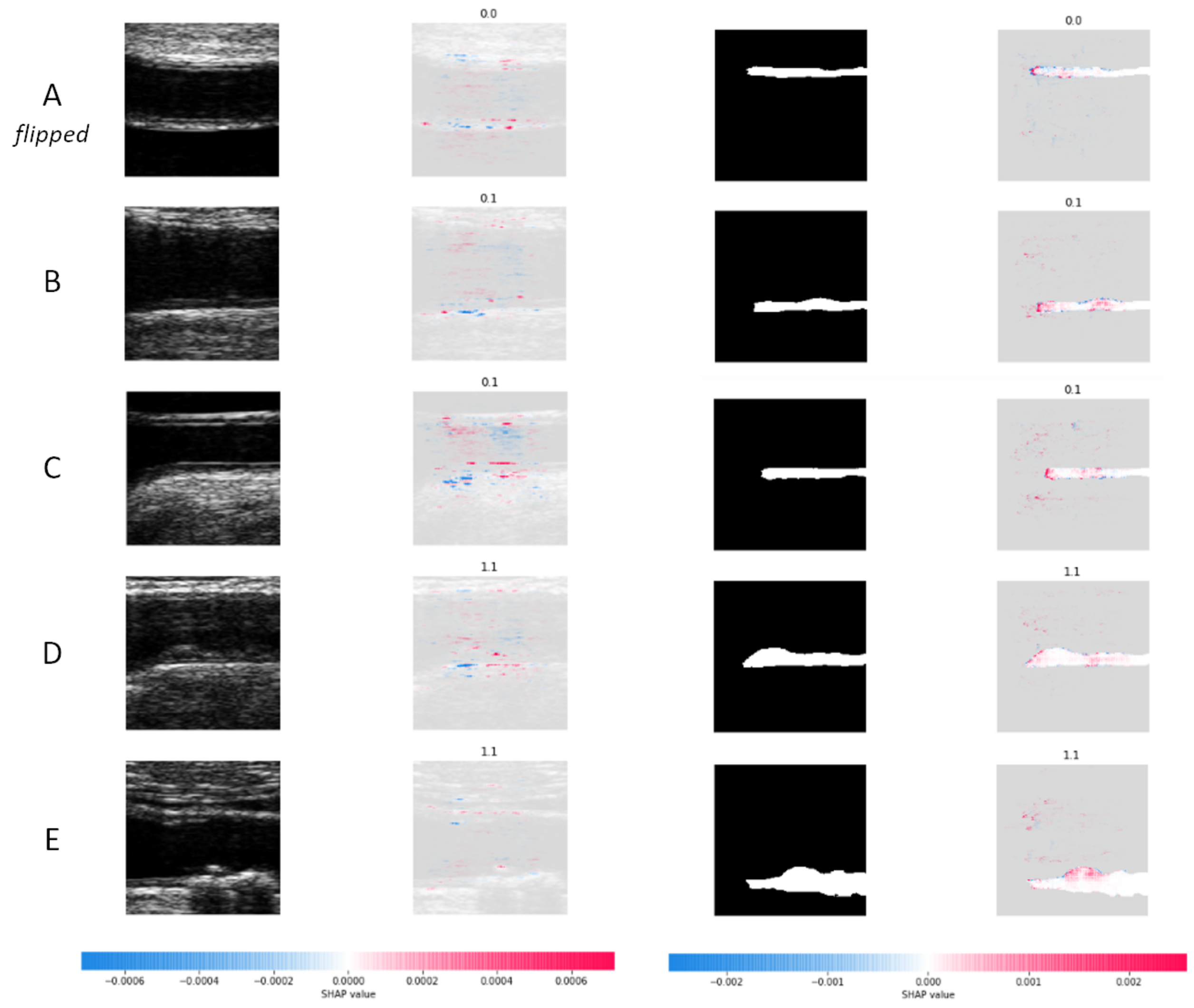


Figure 19: SHAP values in Experiment 2, images B and C misclassified

As we can see, the model will generally check the "correct" sections of the image. A human interpretation is that:

- With the labels, the model focuses on the width, and the bumps, which constitute a larger max IMT, the main criteria for plaque / no plaque.
- It checks the areas around the labels, which also influence the width.

- With the images, it focuses mainly the sides of the arteries where the IMT region is (it could be top or bottom), with some noise coming from the center.

This can be examined in further detail by focusing on single examples. In figure 20, the model misclassifies, and we can easily see that the sparse SHAP values don't enable a clear decision. As a human we would not trust this decision either.

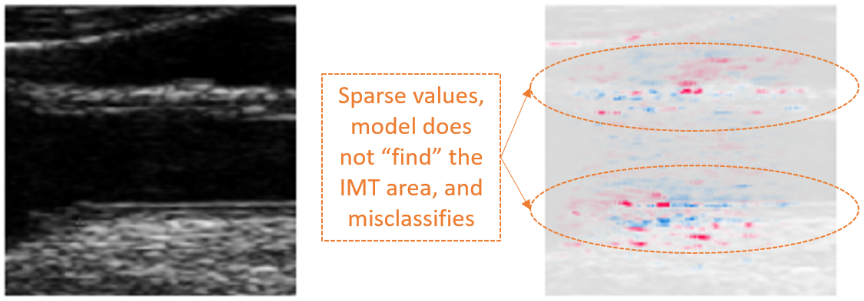

Sparse values, model does not "find" the IMT area, and misclassifies

Figure 20: Detailed results of figure 18 Image A, a misclassification: we see that the model is not capable of finding the IMT area

On the other hand, in figure 21, we see that the model will check both sides of the artery for the IMT calculation (given some images are flipped, it can be on either side), and will focus on the "bumps" parts of the label:

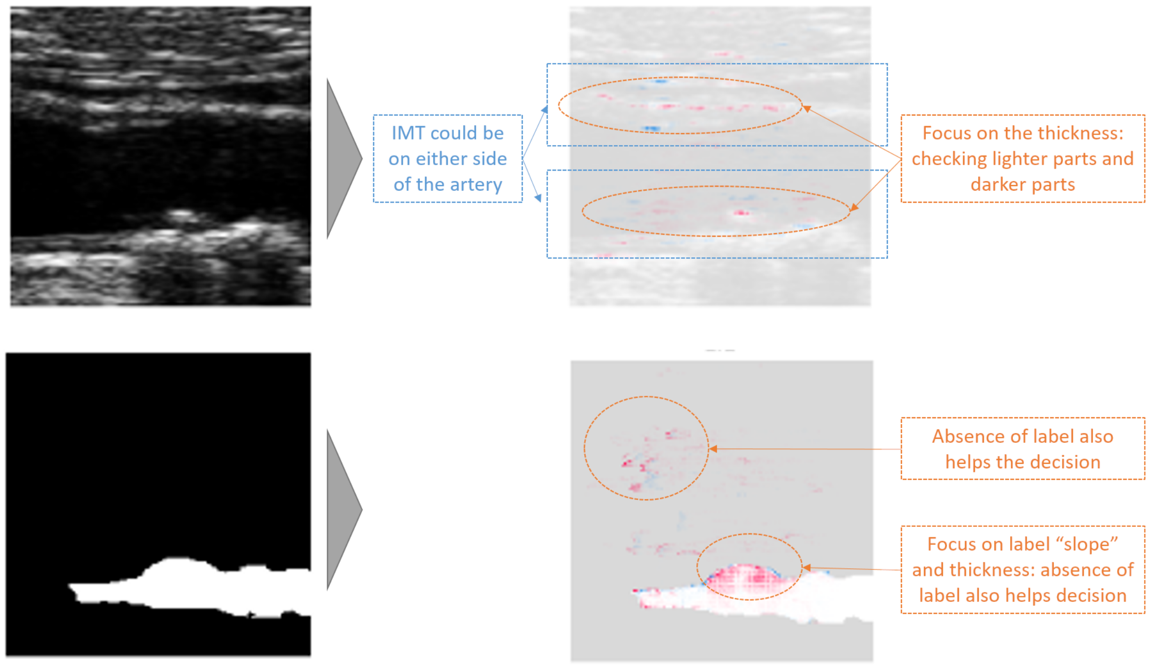

IMT could be on either side of the artery
Focus on the thickness: checking lighter parts and darker parts
Absence of label also helps the decision
Focus on label "slope" and thickness: absence of label also helps decision

Figure 21: Detailed results of figure 19 Image E, a correct prediction of plaque: the classifier checks both sides of the image for IMT, and the "bump" in the label.

3.4 Expansion to regression

Given the good results, we also try expanding our approach to the regression problem of estimating the IMT values, using a similar model.

We use the original set of images as we cannot run data augmentation (all the IMT values would be the same in the augmented set, and it is unnecessary since we no longer have an unbalanced dataset). We use a similar architecture, switching to a mean squared error loss (MSE), a ReLU activation unit on the final dense layer (since all the values we want are positive), and optimizing for Mean absolute error. The preliminary results for the loss values are shown in figure 22:

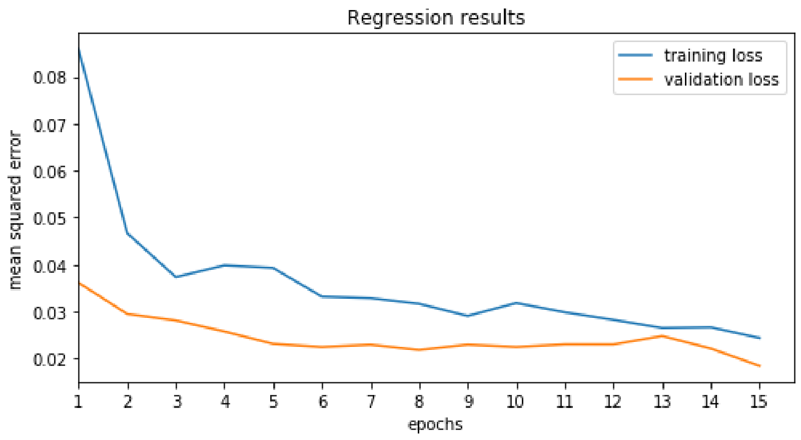


Figure 22: Experiment 2 loss values for the regression on IMT

As we can see, the validation MSE tends to stagnate overtime, meaning we could improve our model further. Moreover, as we are in a context of plaque/no plaque classification where we use IMT > 1.5 as the key metric, even slight variations could generate misclassifications as a consequence. The results are detailed in table 2:

Dataset	Model	MSE	MAE
REGICOR 4000 4,731 Images	Experiment 1	0.037	0.1034
REGICOR 4000 with labels 4,731 Images / 4,731 Labels	Experiment 2	0.025	0.0971

Table 2: Results of the Regression - with 1 or 2 images as input

However, when running the images through the SHAP framework as previously, we notice that the interpretability of our results is again strong (and similarly to classification, higher sparsity of SHAP values gives a lower accuracy). This implies that with some improvements to the model, the results could help automate further the IMT calculation and identification process of plaque. The main difference that we can see in figure 23 compared to the classification case is the higher concentration of values in the IMT area. This is logical given the model is now attempting to calculate an IMT rather than classify an image, and for this, it needs a higher amount of data in that specific area.

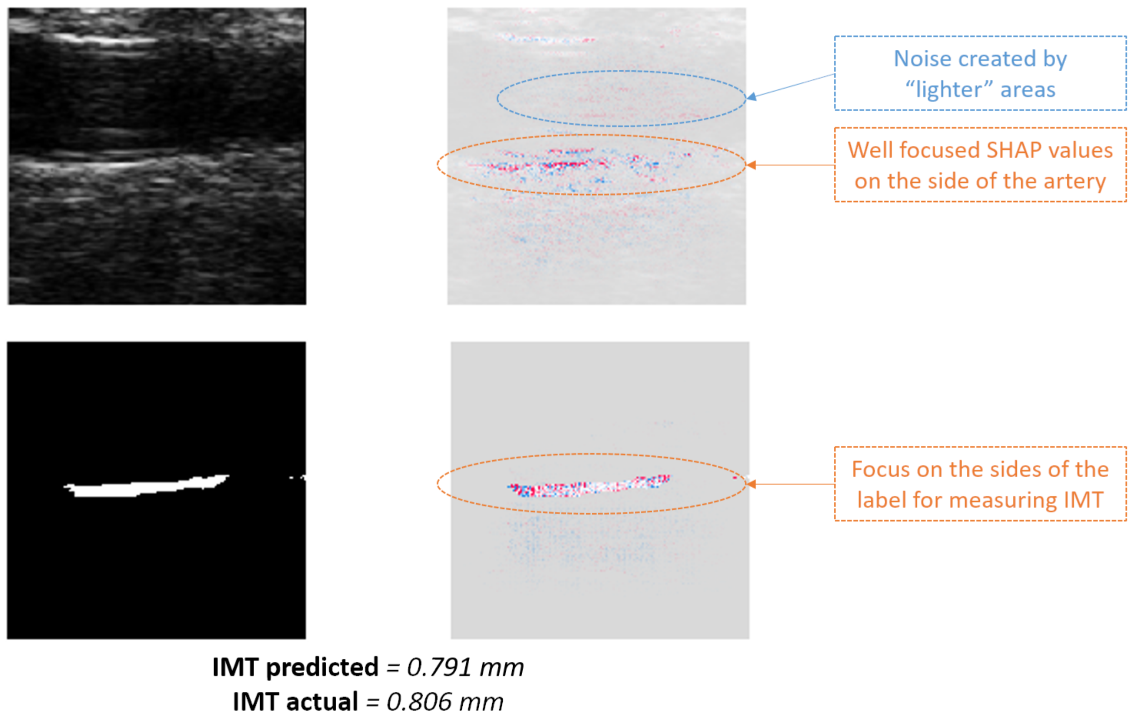

Noise created by "lighter" areas
Well focused SHAP values on the side of the artery
Focus on the sides of the label for measuring IMT
IMT predicted = *0.791 mm*
IMT actual = *0.806 mm*

Figure 23: Example SHAP values for regression Experiment 2 - the network correctly focuses on the full label area to estimate the IMT value.

This conclusion holds even if we run the model with the Experiment 1 setup: even if the regression underperforms compared to Experiment 2, as expected, the results are still highly interpretable. The model results are shown in table 2. We can see that in the case of a lower accuracy, as in figure 24, the SHAP values have high sparsity, whereas in a more accurate one in figure 25, the SHAP values are more focused on the IMT area.

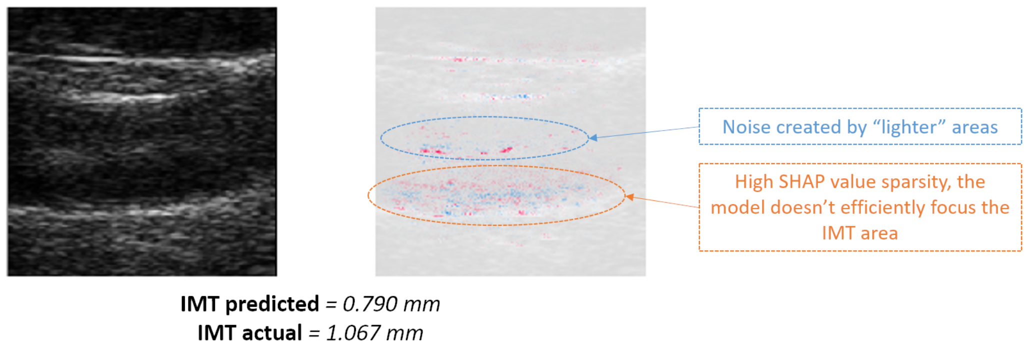

Noise created by "lighter" areas
High SHAP value sparsity, the model doesn't efficiently focus the IMT area
IMT predicted = *0.790 mm*
IMT actual = *1.067 mm*

Figure 24: Example of a lower accuracy result for Experiment 1 - the model correctly focuses on the IMT area, but with high sparsity.

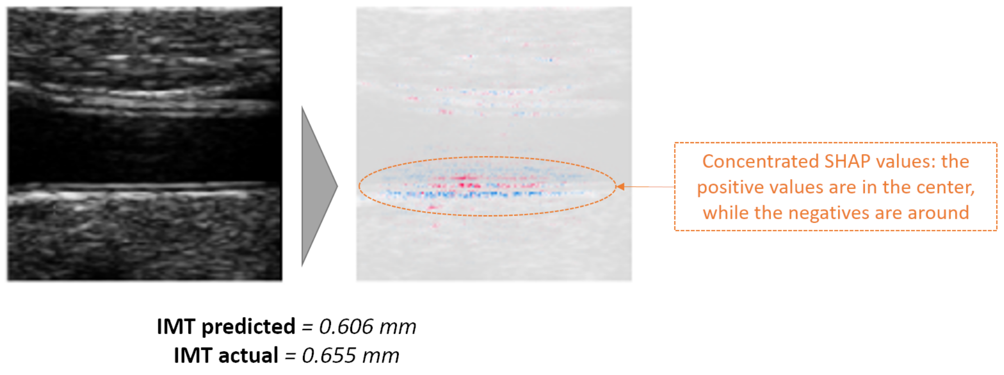

Concentrated SHAP values: the positive values are in the center, while the negatives are around
IMT predicted = *0.606 mm*
IMT actual = *0.655 mm*

Figure 25: Example of a higher accuracy result for Experiment 1 - the model is well focused on the IMT area.

This is a very good sign for the future. Labels are either manual or long to generate via the models we currently have: being able to use only the images to estimate IMT would be a huge time saver.

3.5 SHAP values test conclusion

As a conclusion, we see that our method, using a CNN and SHAP interpretability shows a lot of promise. The network classifies well, and this classification decision has a strongly understandable basis for a human, as we defined in our review of interpretability. In addition to that, it can run on a normal computer.

We will now attempt to further these promising and interpretable results with work on the segmentation output, namely the 6 labels. And here we will require more computational power, and new models!

4 Grid saliency methods implementation and experimentation

We will in the following steps implement explanations for the two types of Grid Saliency methods based on the published works from section 2.5, with the final goal of obtaining explanations for a semantic segmentation network of images of the carotid artery.

4.1 Grid saliency methods for image classification

The grid saliency explanations for semantic segmentation that we are going to explore are based on black box explanations for image classification via meaningful perturbation. These methods are based on the same ideas as LIME: perturbing groups of super pixels as much as possible, and observing how the black box system behaves under such modification of the input.

The idea lies in figuring out which pixels in an image are the most responsible for obtaining a given result and highlighting them as the classification explanations, as we can see in figure 26.

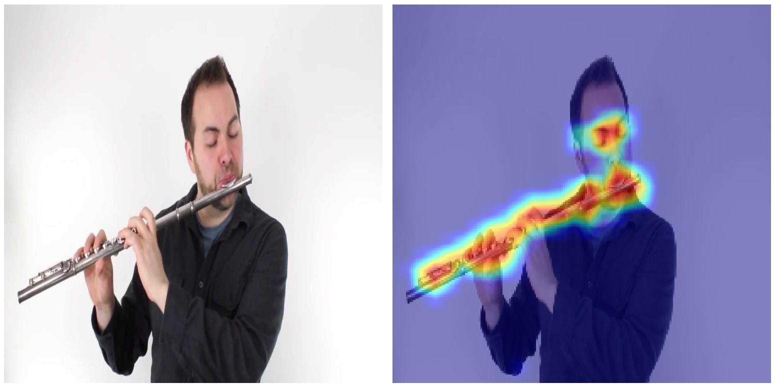


Figure 26: A flute image and an example heatmap of the parts of the image most responsible for obtaining the "flute" classification, Source: GitHub [10]

In order to achieve this, we formulate a meta learning problem in which we will use the following elements:

- **The original image (I)**: The original classified image, with its classification and class probability.
- **The black box classification score ($f_c(x)$)**: The score for classifying the image x as class c.
- **An empty image (E)**: An image devoid of meaning, either random noise, a solid color or a blurred version of the original image.
- **A mask (M)**: A lower dimensional mask with values in [0,1].
- **A perturbed image (P)**: The image resulting from the combination of the original image with the empty image following the mask: $P = M * I + (1 - M) * E$, where * denotes the element-wise multiplication. In the areas where M approaches 1, we will keep the original image information, and in those that approach 0, the image will be closer to the empty one.

The goal is to learn the smallest mask M (in terms of L1 norm, so with the maximum number of ”turned off” pixels) such that P achieves at least the same classification and score as the original image.

We can formulate this objective as an optimization problem:

$$M = argmin_M \lambda \|M\|_1 + |f_c(I) - f_c(P)|$$

Where the first term accounts for obtaining the smallest mask and the second one serves as a penalization for misclassifying the perturbed image, and λ controls the sparsity of M.

4.1.1 Implementation details

We used Pytorch which allows for easy customized loss function writing. The image classifier we used is VGG19, a convolutional neural network which is pretrained and ready to use with torchvision.

We used as a starting source the code available in the Github repository **PyTorch implementation of Interpretable Explanations of Black Boxes by Meaningful Perturbation**[1].

We optimize the mask M over 500 iterations, following the process shown in figure 27:

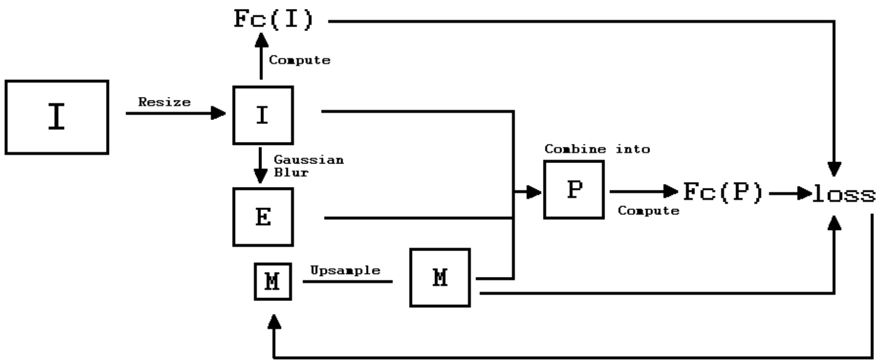


Figure 27: Diagram of the iterative process to obtain the grid saliency map for image classification.

4.1.2 Experiments

We tested this approach on two images, a dog and a person, and the saliency mask obtained for them is shown in figures 28 and 29:

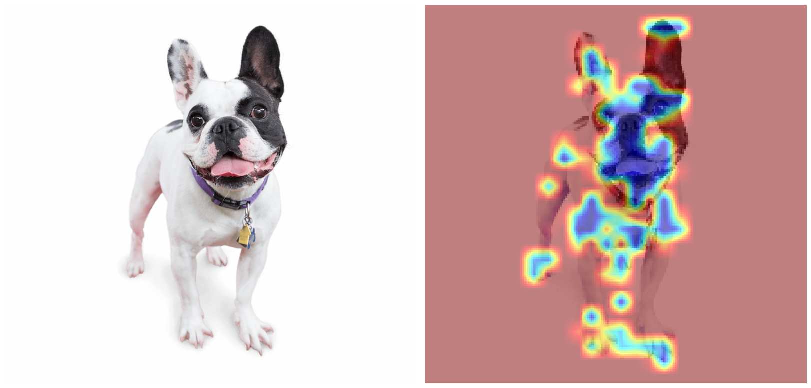


Figure 28: Saliency heatmap obtained for a dog classification. Image source: [21]

[1]https://github.com/jacobgil/pytorch-explain-black-box

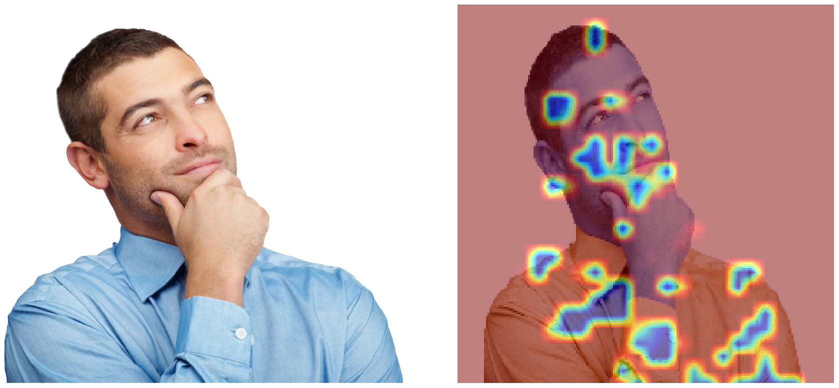


Figure 29: Saliency heatmap obtained for a person classification. Image source: [21]

We see that for the dog, the main focus is on the face, legs and ears, and for the person the main salient areas are located on the face, the hand and the shirt.

4.2 Grid saliency explanation methods for image classification outside of a region of interest

In our journey to obtain saliency explanations for semantic segmentation, it is interesting to modify our image classifier explainer, and make it focus on the outside of a given area region.

In order to do so, we introduce a new parameter to the optimization problem stated in grid saliency methods for image classification.

- **A static mask (SM)**: A mask {0,1} that is not optimized, determined at the very beginning, used in combination with M to obtain the perturbed image.

Now we must force our algorithm to always use the information of the original image I inside the area defined by SM, so P will be now computed as:

$$IM = (1 - SM) * M$$

$$P = IM * I + (1 - IM) * E$$

Where once again * stands for element-wise multiplication and IM is an intermediate mask computed using M and SM. A schematic overview of the resulting architecture can be seen in figure 30:

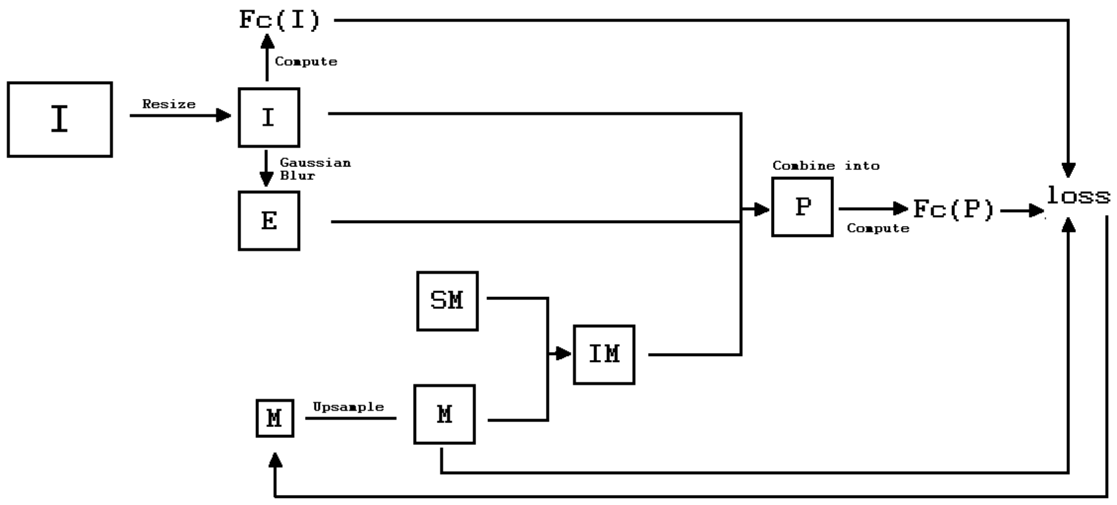


Figure 30: Iterative process to obtain a grid saliency map outside of a static region.

4.2.1 Experiments

We tested this new formulation on the same images we classified in the previous section, with two different static masks: one focused on the faces, and one containing the full figure. We obtained the results shown in figures 31 and 32:

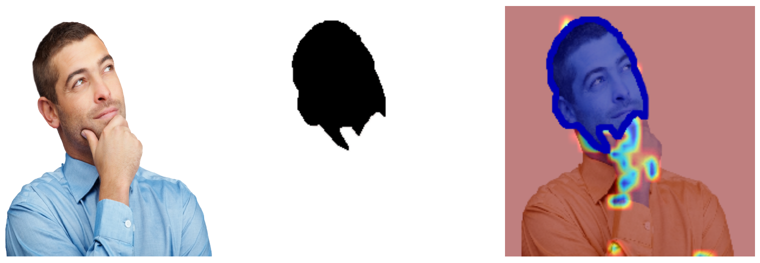


Figure 31: Original image (I), static mask (SM) and saliency heatmap for obtaining a person classification, freezing the face area.

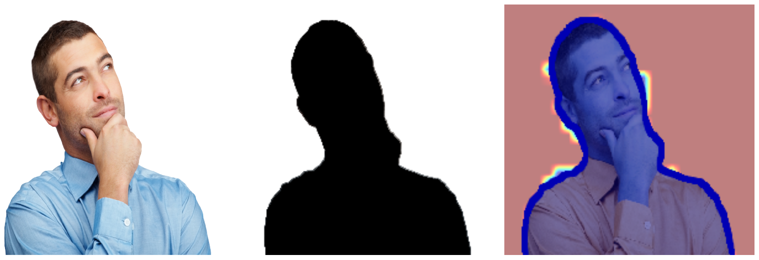


Figure 32: Original image (I), static mask (SM) and saliency heatmap for obtaining a dog classification, freezing the full person area.

We see that in the person image from figure 31, when freezing the face area, the classifier network only relies on the hand to maintain the classification score. When we freeze the full area of the person in figure 32, almost no other pixels get highlighted as explanations for the classification outside that area.

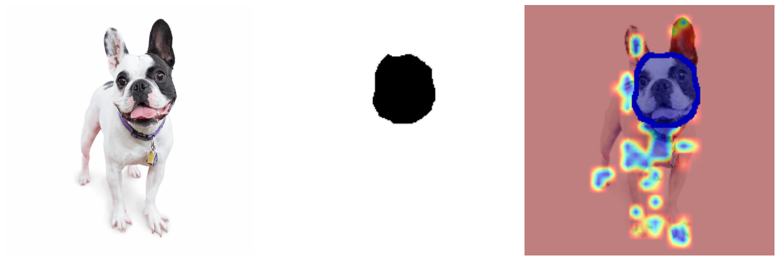


Figure 33: Original image (I), static mask (SM) and saliency heatmap for obtaining a dog classification, freezing the face area.

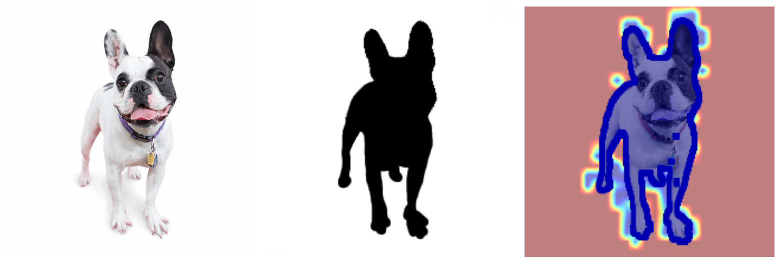


Figure 34: Original image (I), static mask (SM) and saliency heatmap for obtaining a dog classification, freezing the full dog area.

In the case of the dog in figure 33, if we freeze the face area, it still heavily relies on the ears and legs to maintain the classification score. Even when freezing the full dog area in figure 34, we still see some pixels highlighted outside of it, mainly in the shadows cast by the dog (with a dog-ish shape).

4.3 Grid saliency explanation methods for semantic segmentation

Recall that our final goal is to implement grid saliency explanations for semantic segmentation.

The main difference when working with semantic segmentation compared to image classification, is that instead of obtaining a score for a category on the full image, we will now obtain a score for each pixel. The segmentation class of each pixel will be the class with a higher score.

We will introduce yet another parameter to our vocabulary:

- **Region of interest (R)**: The segmented region for which we want to obtain context explanations (the request object).

We will then reformulate the goal problem [8] as learning the smallest mask M outside of the region of interest R (determined by the segmentation) such that we preserve the same segmentation inside of that region of interest:

$$M = argmin_M \lambda \|M\|_1 + \frac{\|R * |(f_c(I) - f_c(P))|\|_1}{\|R\|_1}$$

4.3.1 Implementation details

Since we now integrate R into the loss function, we will use an implementation similar to that of figure 27.

Since we now need to test in a semantic segmentation network, we used DeepLabV3-ResNet101, that similarly to VGG19, comes pretrained and able to segment for 21 classes in torchvision.

4.3.2 Experiments

Sadly, since we are changing the black box model we are obtaining explanations from, and we need to focus on context, we had to drop our man and dog image examples.

Instead, we tested our grid saliency "explainer" in the set of images shown in figures 35, 36, 37 and 38:

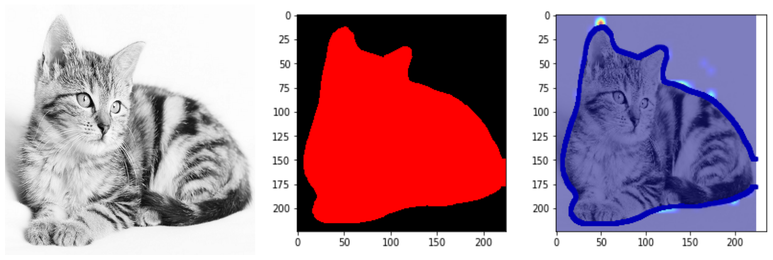


Figure 35: Original image (I), segmentation result with R in red, and context explanations for a cat segmentation.

As we see in figure 35 we seek context explanations in an image with only one element, we observe that barely any pixels get highlighted as explanations for the cat segmentation.

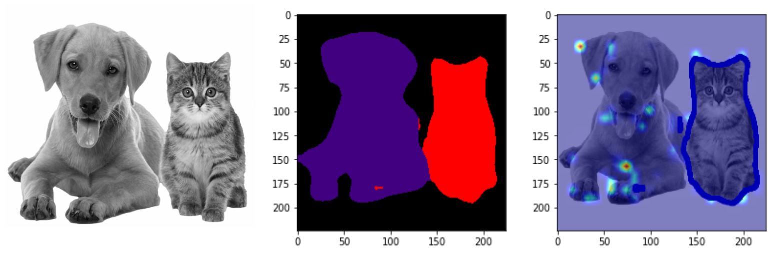


Figure 36: Original image (I), segmentation result with R in red, and context explanations for a cat and dog segmentation.

If instead of having only a cat in the image, we put the cat besides a dog (figure 36), our grid saliency explainer highlights some dog areas: mainly the paws and the ears, meaning that being next to a dog is part of what contributes to the segmentation of the cat as a cat.

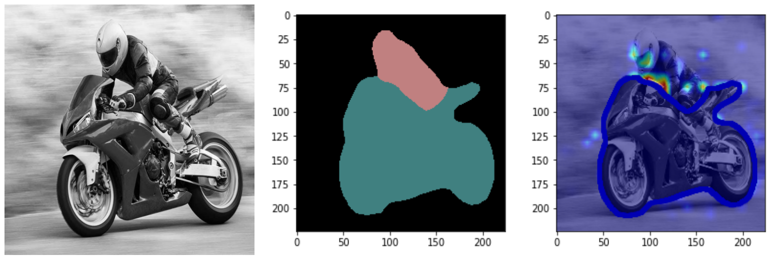


Figure 37: Original image (I), segmentation result with R in green, and context explanations for a motorbike and rider segmentation.

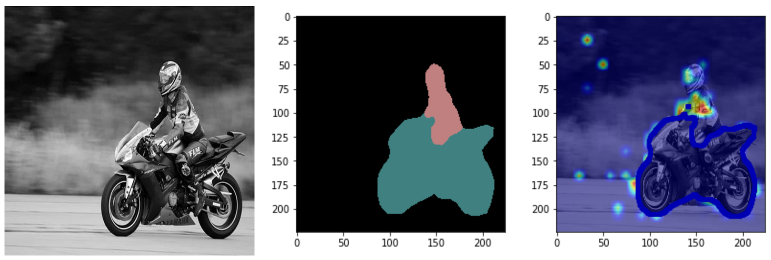


Figure 38: Original image (I), segmentation result with R in green, and context explanations for a motorbike and rider segmentation.

In figures 37 and 38 we ditch the white background and try to determine why the motorbike is being classified as such. We see that the main pixels responsible for this segmentation are located on the biker (helmet and hands), with the road having some relevance as well, especially in figure 38.

5 Grid saliency for semantic segmentation applied to carotid artery images

In the following section we will use the grid saliency explanation methods we have developed to analyse the results of two semantic segmentation neural networks trained on the REGICOR dataset.

5.1 DeepLabV3 ResNet101 fine tuned

ResNet101 is a residual neural network model characterized by its skip connections. Skip connections allow fitting the input from a previous layer to the next without needing to modify the input. Based on this architecture, Google constructed DeepLabV3, one of the best image segmentation networks to date, with 85,7% IoU (Intersection over Union) score on the PASCAL VOC 2012 dataset. There is a Deeplabv3-ResNet101 pre-trained with a subset of COCO train2017 in pytorch.

In order to use this network to semantically segment our carotid artery images, we fine tuned this model. The process of fine tuning consists of adapting an already trained network to work with your target task.

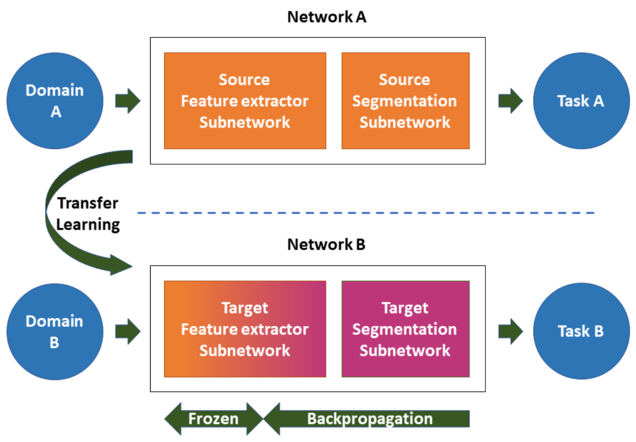

Network A
Domain A
Source Feature extractor Subnetwork
Source Segmentation Subnetwork
Task A
Transfer Learning
Network B
Domain B
Target Feature extractor Subnetwork
Target Segmentation Subnetwork
Task B
Frozen
Backpropagation

Figure 39: Transfer learning diagram (Source: Transfer learning for segmentation [22])

A schema of how this fine tuning process works can bee seen in figure 39. We change the target segmentation sub-network and adapt it to detect the IMT CCA area in the carotid artery images that are our target problem, training with a lower learning rate than in the normal training process.

5.1.1 Training the segmentation model

Using the code in the Github repository **DeepLabv3FineTuning** [22], we fine tuned the pytorch implementation of Deeplabv3-ResNet101 targeting our carotid artery images. The code changes the classifier module of the model for a new one with a new number of output channels: two in this case, IMT area and background.

We split the images in the REGICOR dataset (157 used in this case) into a train set with 90% of the images and a test set with 10% of them, and trained the network, during 25 epochs and batches of 3 images. The training lasted about 8 hours. This code was trained on a Universitat de Barcelona server, with an Intel Core i7-6800K CPU (3.40GHz).

The semantic segmentation model reaches an average IoU for the images in the test set of **0.791**. We will now use the grid saliency method to obtain explanations of this model's performance.

In figure 40 we can see the qualitative result of our fine tuned network over the test set:

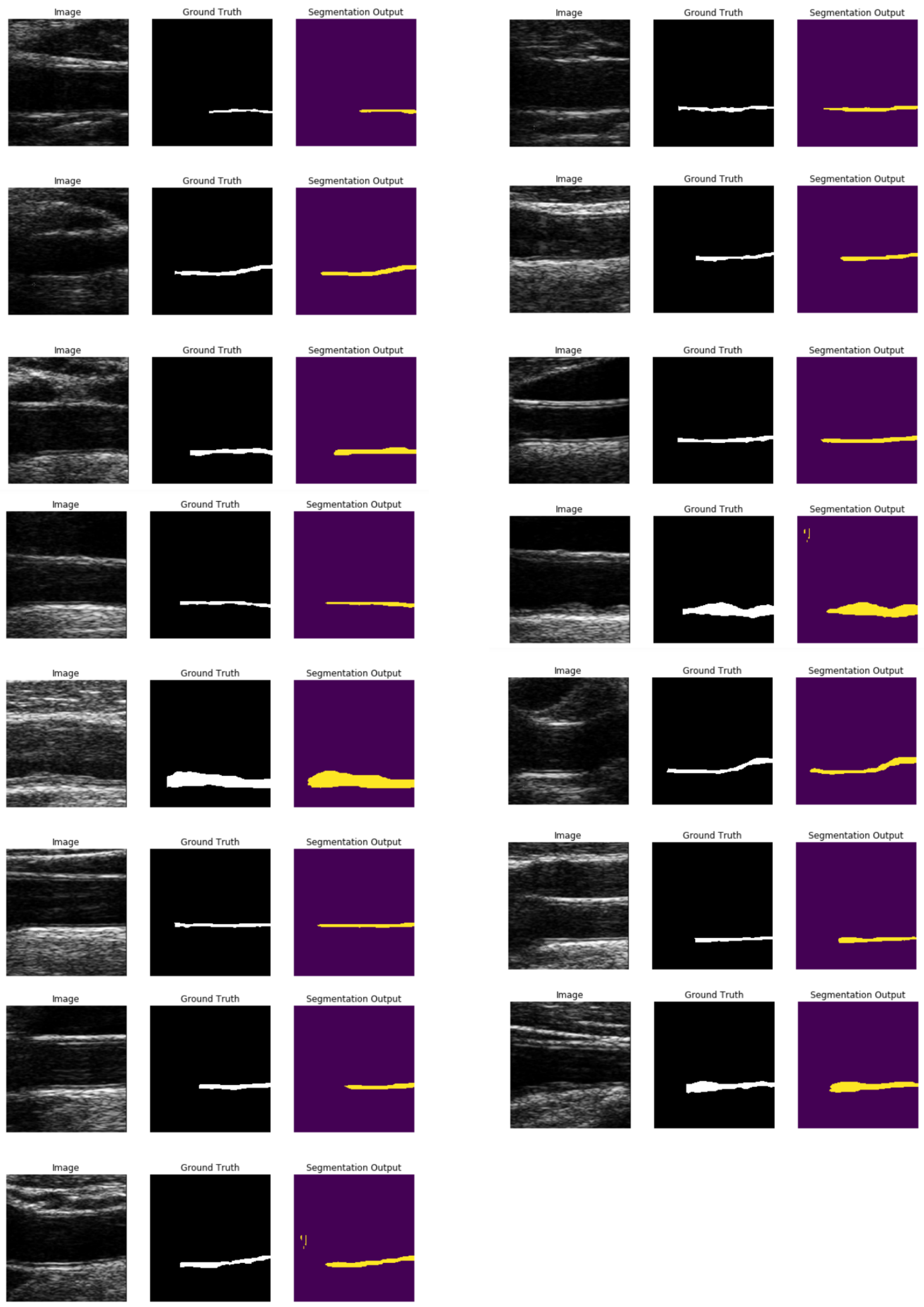


Figure 40: Test set original images, ground truth of the IMT CCA area and segmentation result using Deeplabv3-ResNet101 fine tuned.

5.1.2 Grid saliency results

We can see below in figure 41 the results of the grid saliency for semantic segmentation explanations applied to our test set.

In general, the areas highlighted by the algorithm refer to:

- **The IMT area**: In some images like 4, 6 and 7, areas next to the IMT CCA are barely highlighted, implying there is enough information inside the area of interest to perform the semantic segmentation. The outside areas that get slightly highlighted in these cases are mostly located on the far wall.
- **The far wall**: In other cases, like in images 11 and 13, some areas of the far wall are strongly highlighted, with a smaller appearance of the lumen bulb. This is one of the adjacent areas and as opposed to the lumen, it contains less dark pixels overall, so it is a very reasonable part of the image to look at for information.
- **The far wall and the bulb areas**: Another area that gets highlighted occasionally is the one corresponding to the lumen bulb and the IMT bulb. We can see it in images like 2, 5 8 and 9.

Overall we can argue that our fine tuned DeepLabV3 network, besides the area corresponding to the IMT, uses information corresponding to the areas nearest to it to perform the segmentation, specially the neighbouring areas.

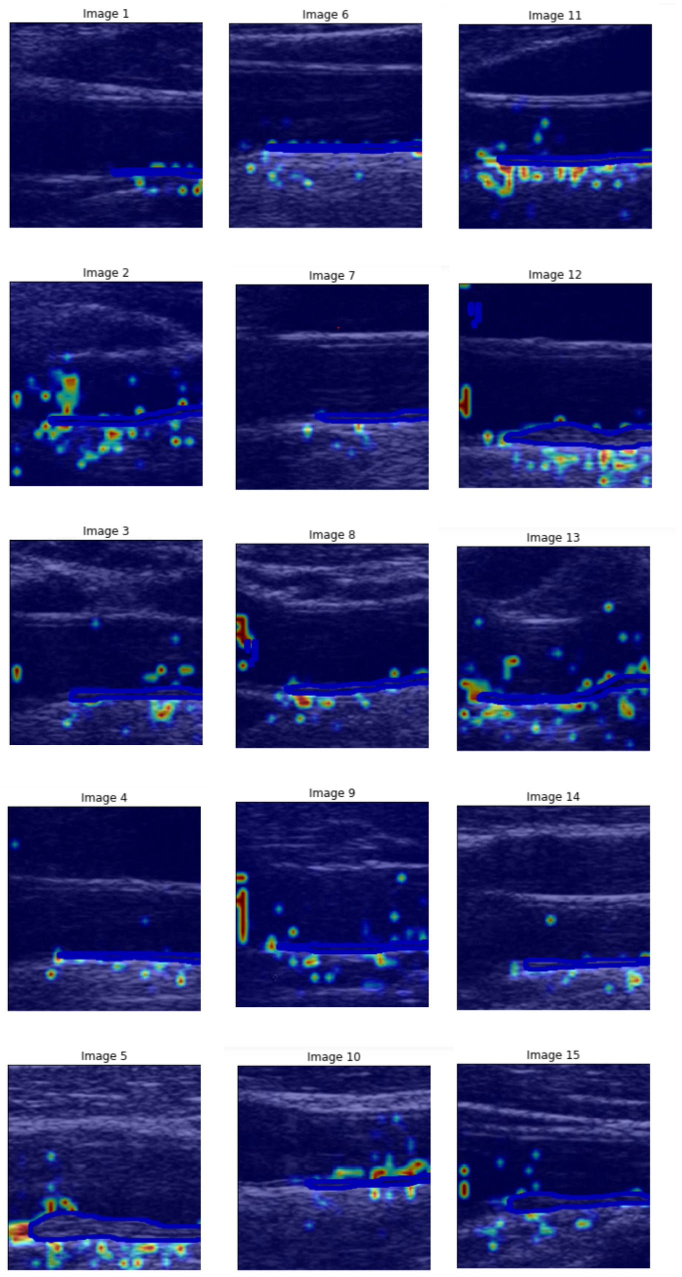
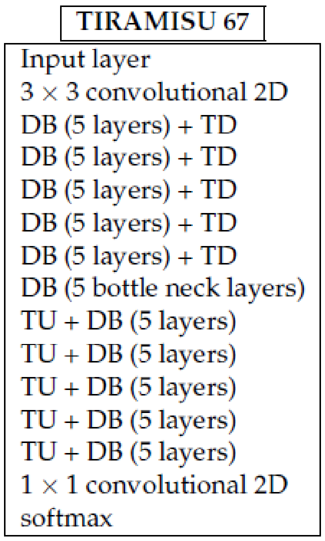


Figure 41: Grid saliency result over the test set.

5.2 Tiramisu67

In the work by Escapa et al. [1] a Tiramisu67 network was used for semantic segmentation of the carotid artery images.



TIRAMISU 67
Input layer
3 × 3 convolutional 2D
DB (5 layers) + TD
DB (5 layers) + TD
DB (5 layers) + TD
DB (5 layers) + TD
DB (5 layers) + TD
DB (5 bottle neck layers)
TU + DB (5 layers)
TU + DB (5 layers)
TU + DB (5 layers)
TU + DB (5 layers)
TU + DB (5 layers)
1 × 1 convolutional 2D
softmax

Figure 42: Architecture of Tiramisu67 [1]

Tiramisu is an extension of Dense Net architecture to Fully Connected Networks. This network can be constructed with different configurations and in this case 67 layers are used (hence the name of the network). In figure 42 we can see the network's architecture.

In this case, the aim of the semantic segmentation is to obtain a segmentation for the six areas that can be seen in the carotid artery images: the near wall, the lumen bulb, lumen CCA,the IMT bulb, the IMT CCA and the far wall.

5.2.1 Training the segmentation model

Using the code in the Github repository **Carotid Artery**[2] that implements Tiramisu67 in Keras, we trained this network with the REGICOR dataset, using a train set of 141 images (90%) and a test set of 16 images (10%).

In order to train this network, we set up a docker image on a server located in Universitat de Barcelona. We used nvidia-docker to make the nvidia CUDA drivers available in our code in order to be able to train with the GPU unit. The server had a powerful GPU installed: the network was trained on a GeForce GTX 1080 (12GB).

This process was split into 2 phases:

- **Phase 1**: Training with a learning rate of 0.001, using data augmentation with random cropping and adding 1% of Gaussian noise. Images are introduced in batches of 3.
- **Phase 2**: Training with a learning rate of 0.0001 and without data augmentation. Images are introduced in batches of 1.

The trained lasted for about 15 hours during a total of 263 epochs. The average IoU for the images in the test set is **0.659**.

[2]https://github.com/DaniSalva/CarotidArtery-DomainAdaptation

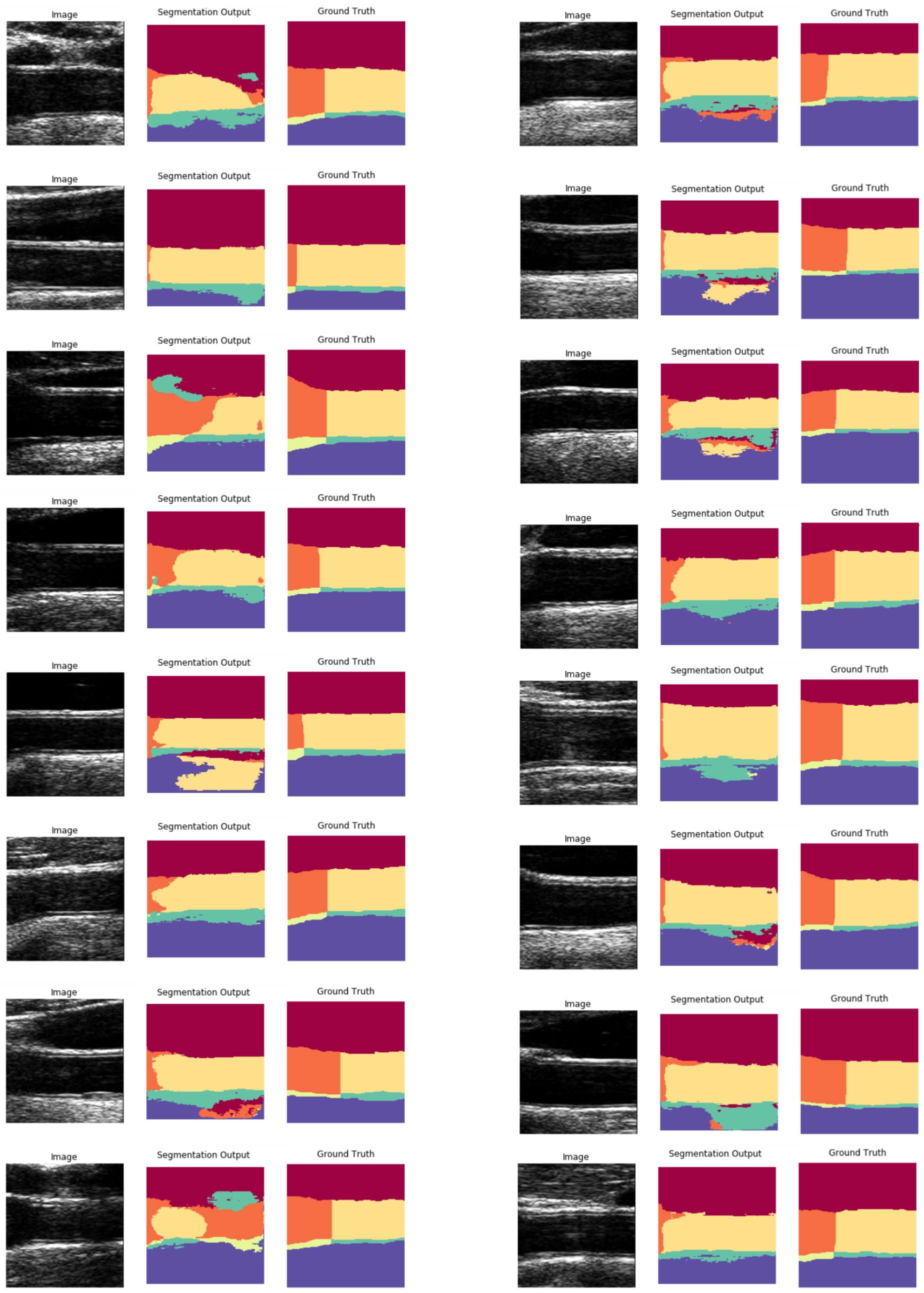


Figure 43: Test set original images, ground truth of the carotid artery, 6 class segmentation result using Tiramisu67: Near wall (granate), IMT CCA (green), Far wall (blue), Lumen bulb (orange), Lumen CCA (yellow), IMT Bulb (fluorescent yellow).

In figure 43 we can see the qualitative result of the Tiramisu67 network over the test set.

5.2.2 Grid saliency results

This model was created using Keras, and has the particularity of being constructed using a fork of Keras and a wrapper. The wrapper deals with the way the images are passed through the network and also with getting the final segmentation result.

These particularities did not allow us to run our grid saliency algorithm in order to obtain explanations. Instead, we created an approximation of it, based on pre-constructed masks.

For each image segmentation we created several masks, and then computed the value of the loss stated in the grid saliency method for each one of them, in order to see which one was optimal.

The 10 pre-constructed masks were the following:

- **IMT CCA Only**: Only the region of interest, the IMT CCA.
- **IMT CCA and Near Wall**: The IMT CCA and the Near Wall area, which are opposite areas of the image.
- **IMT CCA and Far Wall**: The IMT CCA and the Near Wall area, which are connected areas (the Far wall lies below the IMT).
- **IMT CCA and Lumen Bulb**: The IMT CCA and the Lumen Bulb area.
- **IMT CCA and IMT Bulb**: The IMT CCA and the IMT Bulb area (once again, connected areas, the IMT Bulb lies left of the IMT CCA).
- **IMT CCA and Lumen CCA**: The IMT CCA and the Lumen CCA area (connected elements, with the Lumen CCA above the IMT).
- **IMT CCA Expanded 5** The IMT CCA area, expanded using a Gaussian blur of sigma 5.
- **IMT CCA Expanded 10** The IMT CCA area expanded using a Gaussian blur of sigma 10.

- **IMT CCA Expanded 15** The IMT CCA area expanded using a Gaussian blur of sigma 15.
- **IMT CCA Expanded 20** The IMT CCA area expanded using a Gaussian blur of sigma 20.

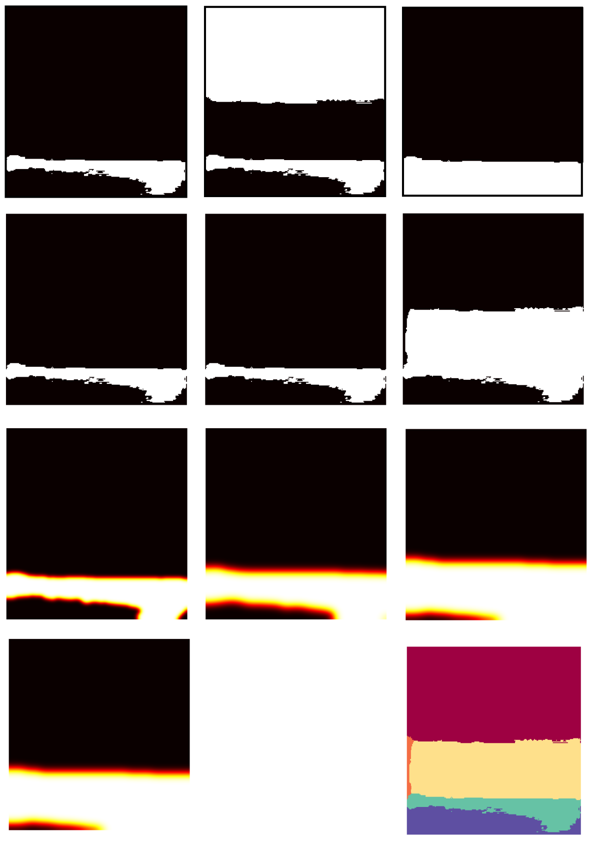


Figure 44: Example of the set of 10 masks tested for a given semantic segmentation result (bottom right).

In figure 44 we can see an example of the set of 10 masks we will test for a given segmentation.

In all cases, the mask that achieved the lower loss value was the one that contained only the IMT CCA area. In figure 45 we can see some of the results.

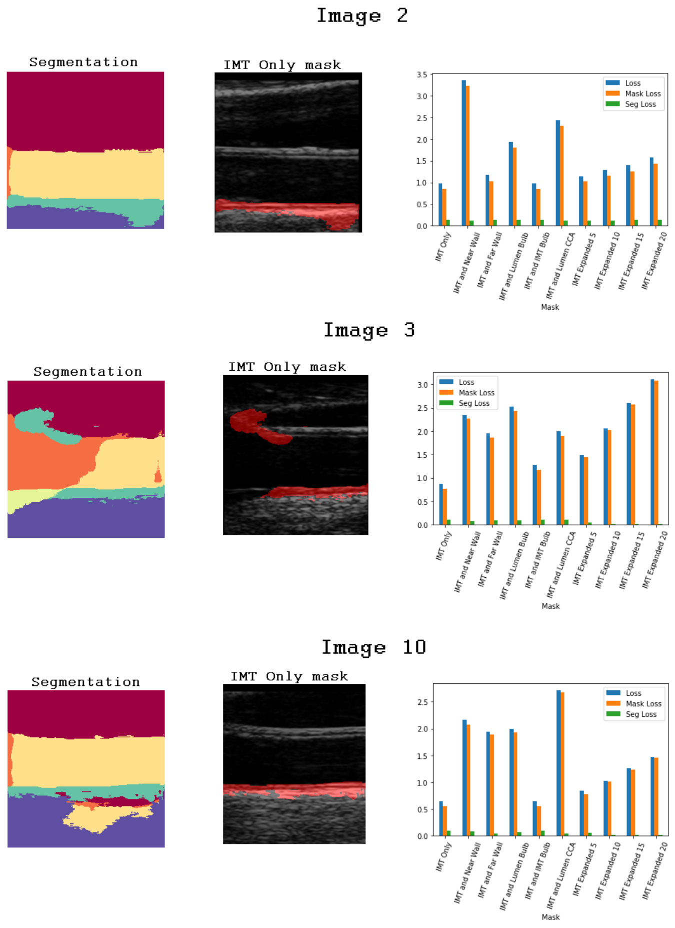


Figure 45: Segmentation result, mask with minimum loss value and loss for all the pre-constructed masks (mask loss accounts for the sparsity of the matrix and seg loss for the ”correctness” of the segmentation. Loss is the sum of both).

We can say that among all these pre-constructed masks, the one that better explains the segmentation is the one that contains the IMT CCA area only. Looking at the loss values, the component that accounts for the loss of segmentation (how the segmentation worsens by perturbating the outside area) remains more or less stable. So, the lower loss attributed to the IMT CCA means that the IMT CCA is the smallest area that gets a decent score. This approximation tells us that this model is using the information inside this area the most when determining which region is the IMT CCA. That could be the reason why we obtain wrong segmentation in some cases, (like image 3, where there's IMT CCA segmented at the top of the lumen) because instead of looking in the neighbouring areas, the network is focusing on the pixels inside the IMT CCA mask.

We must take into account that these properties are not as complete as with the mask we would obtain by applying the grid saliency method, but only a mere approximation.

6 Conclusions

The purpose of this work was to study the viability of Deep Learning methods in the context of atherosclerosis detection, by examining the model ”decisions” through the lens of interpretability.

We successfully demonstrated, through the use of SHAP values, that our CNN models were strong candidates for the automation of plaque detection, and were also good candidates for measuring the IMT values on an image. The main fact that makes them well suited to the tasks is that the classification and regression decisions seem to focus on the correct image locations. This means they would ”reasonably” classify the ultrasound images in the eyes of a professional, and could be expanded more datasets.

We also explored grid saliency methods for image segmentation explanations. We found out where a pair of neural networks get the main information in order to perform a prediction. These explanations can be used in the future to improve the performance of the black box models explained, and, since we are working in the medical field, help humans make sure that the predictions are not being made based on arbitrary image features, but on the same features a human would use to take a decision.

Generally speaking, we confirmed the interest of Deep Learning models for plaque detection, and would strongly encourage others to continue this research. If these approaches can be run at scale, in a ”real” environment, they could be a massive help to medical professionals.

7 Future work

We strongly believe that the interpretability of DNN in the context of plaque detection, classification and segmentation makes them candidates for further use: having a trained network analyse and display results faster than a human, but always with understandable underlying decisions.

Further expansion of this work would be in refining the precision and interpretability on both classification and segmentation. We saw that the model will generally focus on the correct locations in the context of plaque detection, and that the differences obtained by adding the labels are small. This means that by refining the training on the original images, we could apply the CNN to more ultrasound images for plaque detection. In addition to that, we noticed that the CNN could in itself be a good candidate for calculating IMT directly through regression, but would need a larger refinement. We could use the explanations obtained with SHAP to improve the IMT detection.

Concerning the grid saliency, it would be interesting to confirm our findings with human experts in the field and apply what has been discovered to make neural networks perform better in their prediction tasks. More precisely, it would be interesting to get grid saliency explanations for the Tiramisu67 network (in contrast to the approximation proposed) and see if the explanations for a 6 label segmentation network coincide with the 2 label one.

Another idea to train the classification network (from section 3) further, without the costly process of obtaining and cleaning new images in our unbalanced data set, would be to generate images through a Generative Adversarial Network (GAN [24]). As a reminder, a GAN consists basically of 2 models, the generator, and the discriminator, that try to ”outdo” one another based on an original set of images. The generator tries to create a fake, while the discriminator tries to identify the created image as real or fake. Over time, the generated images should increasingly resemble those of the original set.

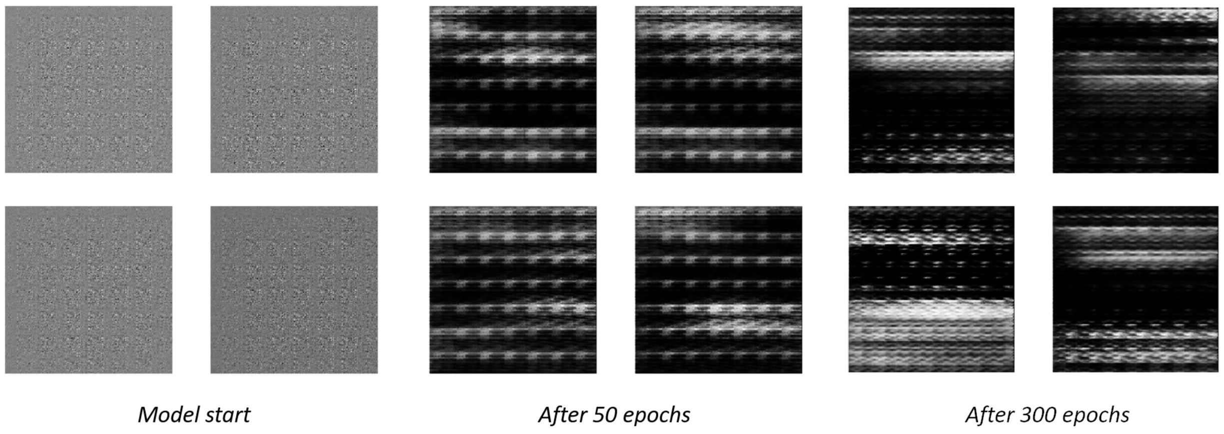


Figure 46: Results of GAN training per epoch

We tried to train a GAN network (Tensorflow docs [18]), feeding the 4,731 REGICOR original images as the source, over as many epochs as possible for the computer. The results aren't good yet, but with a longer training process it is possible that we would be able to create much better fakes, in order to continuously train the model.

As always, we let those with the right computational power generate the new batch of training images! We know that it can only improve the accuracy of our interpretable Deep Learning models, as we provide new examples to train from, including from patients generated by the computer that have not been found yet by local doctors. The only issue will be the need to calculate the IMT for these generated images, but we consider the idea worth exploring nonetheless.

8 Source code

All the source code used to develop this project is available on GitHub: Link

Members contribution

It should be noted that the thesis was done as a group work, and no task was done in isolation. The distribution of tasks has been as follows:

- Susana Castela: Started with a review of the state of the art, in particular the theory of interpretability methods. After that, worked on the training of the networks for semantic segmentation and focused on the implementation of grid saliency methods.
- Christopher Fady: Started with a review of the state of the art, in particular the articles on the application of interpretability. After that, working on the application of SHAP values to the REGICOR images and labels.

Both members contributed equally to the project. Christopher contributed more to the writing of this document and Susana dedicated more time to server configuration, docker set up and large model training.

References

[1] **Deep Learning for the Detection and Characterization of the Carotid Artery in Ultrasound Imaging, 2018**
Arnau Escapa, Jonatan Piñol, Enric Sarlé,
https://github.com/escapa12/carotidsTFM

[2] **Explaining nonlinear classification decisions with deep Taylor decomposition, 2015**
Grégoire Montavona, Sebastian Lapuschkinb, Alexander Binderc, Wojciech Samekb, Klaus-Robert Müllera

[3] **Methods for interpreting and understanding deep neural networks, 2017**
Grégoire Montavon, Wojciech Samek, Klaus-Robert Müller

[4] **Protecting Voice Controlled Systems Using Sound Source Identification Based on Acoustic Cues, 2018**
Yuan Gong, Christian Poellabauer
https://www.researchgate.net/figure/An-illustration-of-machine-learning-adversarial-examples-Studies-have-shown-that-by_fig1_325370539

[5] **A game theoretic approach to explain the output of any machine learning model**
Scott M. Lundberg et al.
https://github.com/slundberg/shap,

[6] **A Unified Approach to Interpreting Model Predictions**, NIPS 2017
Scott M. Lundberg, Su-In Lee,
http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions

[7] **Mannheim Carotid Intima-Media Thickness and Plaque Consensus**
Touboul, PJ et al. (2012): An Update on Behalf of the Advisory Board of the 3rd and 4th Watching the Risk Symposium 13th and 15th European Stroke Conferences, Mannheim, Germany, 2004, and Brussels, Belgium, 2006. In: Cerebrovascular diseases (Basel, Switzerland) 34.4, pp. 290–296.
http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3760791/.

[8] **Grid Saliency for Context Explanations of Semantic Segmentation**, 2019

[9] **Interpretable Explanations of Black Boxes by Meaningful Perturbation**, 2018,
Ruth C. Fong, Andrea Vedaldi

[10] **PyTorch implementation of Interpretable Explanations of Black Boxes by Meaningful Perturbation, 2018**
https://github.com/jacobgil/pytorch-explain-black-box

[11] **Interpretable Machine Learning: A Guide for Making Black Box Models Explainable**
2019, *Christoph Molnar*
https://christophm.github.io/interpretable-ml-book/taxonomy-of-interpretability-methods.html

[12] **ImageNet classification with deep convolutional neural networks**
In NIPS, 2012, pp. 1106–1114, 2012. *Krizhevsky, A., Sutskever, I., and Hinton, G. E.*

[13] **Examples are not Enough, Learn to Criticize! Criticism for Interpretability**,
2016, *Been Kim, Rajiv Khanna, Oluwasanmi Koyejo*

[14] **Explaining prediction models and individual predictions with feature contributions,2014**
Štrumbelj, Erik, and Igor Kononenko, Knowledge and information systems 41.3: 647-665

[15] **Axiomatic Attribution for Deep Networks, 2017**
Mukund Sundararajan, Ankur Taly, Qiqi Yan
https://arxiv.org/abs/1703.01365

[16] **SmoothGrad: removing noise by adding noise**,
2017, *Daniel Smilkov, Nikhil Thorat, Been Kim, Fernanda Viégas, Martin Wattenberg*,
https://arxiv.org/abs/1706.03825

[17] **Adam: A Method for Stochastic Optimization, 2015-2017** *Diederik P. Kingma, Jimmy Ba*,
https://arxiv.org/abs/1412.6980

[18] **Deep Convolutional Generative Adversarial Network, Tensorflow tutorials**
https://www.tensorflow.org/tutorials/generative/dcgan

[19] **Atherosclerosis definition**, US department of Health & Human Services,
https://www.nhlbi.nih.gov/health-topics/atherosclerosis

[20] **IMT measurement**,
European Society of Cardiology,
https://www.escardio.org/Journals/E-Journal-of-Cardiology-Practice/Volume-13/Intima-media-thickness-Appropriate-evaluation-and-proper-measurement-described

[21] **Images used for grid saliency experiments**:
https://enviragallery.com/wp-content/uploads/2018/10/background.png
https://papermilkdesign.com/images/cat-png-dogs-19.png
https://i.ibb.co/BrNP4bc/motobiker.png
https://i.ibb.co/TW24DM0/motorbiker2.png

[22] **Finetuning DeepLabV3**, 2019
https://expoundai.wordpress.com/2019/08/30/ transfer-learning-for-segmentation-using-deeplabv3-in-pytorch/
https://github.com/msminhas93/DeepLabv3FineTuning/tree/79c4c1cd5a345033a9fc2e5f555ac6bd64d89166

[23] **Carotid Artery Domain Adaptation**
https://github.com/DaniSalva/CarotidArtery-DomainAdaptation

[24] **Generative Adversarial Nets**
Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio
https://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[25] **Dataset** provided by: https://regicor.cat/