Clinically Interpretable Computer Aided Diagnostic Tool for Fibromuscular Dysplasia Detection

Master thesis 2024
at the Faculty of Science
University of Geneva

for the obtention of the Master in Computer Science
by

Pannatier Yvan

under the supervision of :

Prof Fleuret François, thesis supervisor, UNIGE
Prof Anjos André, IDIAP
Dr. Oscar Jimenez-del-Toro, IDIAP

Geneva, UNIGE, 2024

Learn from yesterday, live for today, hope for tomorrow.
The important thing is not to stop questioning.
— Albert Einstein

To my parents, partner and friends, supporting me to follow my path...

Acknowledgements

I would like to express my gratitude to Prof. François Fleuret for his supervision and guidance.

I would also like to thank Prof. André Anjos and Dr. Oscar Jimenez-del-Toro for their supervision, advices, feedback.

My thanks also go to IDIAP institute for providing access to computational grid, without which this work would not have been possible.

Thanks to my family and friends for their support which has been a source of motivation all along my studies.

Lastly, I would like to thanks my partner, for her love, encouragement and constant reminder of what truly matters.

Once again, thanks to all of you for your contribution to this journey.

Geneva, December 3, 2024 Y. P.

Abstract

Fibromuscular dysplasia (FMD) is a rare vascular condition that remains challenging to diagnose accurately due to the non-specific nature of the most common symptoms. To the best of our knowledge, this study is the first investigation of the potential of convolutional neural networks as a tool to help detect FMD in medical imaging. Specifically, we explore four strategies (patch-based learning, self-supervised learning, and using two different sample resolutions) to train our model. We evaluate the performance of our proposed architecture using quantitative metrics and saliency maps for interpretability. Furthermore, we propose an extension of the ROAD score to evaluate the quality of the generated volumetric saliency maps. Our results indicate that while 3D CNNs show promise in identifying patterns associated with FMD, they face significant challenges, particularly in handling biases that may arise because the disease is more common in women than men.

Contents

List of Figures

List of Tables

Introduction

Fibromuscular Dysplasia (FMD) is an underdiagnosed vascular disease characterized by abnormal cell growth within the walls of medium-sized arteries. This growth gives the arteries a distinctive "beading" appearance in imaging studies. While the disease can affect a population of any gender, FMD primarily affects women, with only 10% of diagnosed patients being male. The mean age at diagnosis is 52 years, but patients have been diagnosed at any age[1].

The most frequently reported symptoms include hypertension, headaches, and pulsatile tinnitus, which can significantly impact the quality of life and may delay diagnosis due to their non-specific nature [2]. The renal arteries are the most commonly involved in FMD, as observed in a cohort of 447 patients, where 294 had renal artery involvement [2]. In particular, up to 90% of male patients show renal involvement [1].
While the exact cause of FMD is still unclear, several risk factors have been identified. Smoking, mechanical stress, and exposure to female hormones are among the most commonly associated factors that may contribute to the disease's onset and progression [3].

As explained by Van der Niepen and al.[4], the diagnostic pathway for FMD often begins with clinical suspicion based on symptoms and risk factors. The recommendation is to use computed tomography (CT) as the initial diagnostic modality. CT is a specialized non-invasive imaging modality for acquiring cross-sectional images relying on administering an intravenous contrast agent. The contrast agent enhances vascular structures, allowing for precise visualization of arterial walls and any abnormalities, such as stenosis or aneurysmal changes. The quality of the data acquired through CT is significantly influenced by the timing of contrast injection, image acquisition parameters, and technical specifications, which must be optimized to obtain clear and accurate imaging of the arteries [5]. As of today, radiologists must interpret CT scans manually, which can be time-consuming and costly.

Given that CT is the diagnostic modality for FMD, convolutional neural networks (CNNs) offer a promising approach to improve disease detection by leveraging automated image analysis techniques. CNNs, first introduced by Le Cun and al. [6] for image classification, have been widely used for medical image analysis. Leveraging CNNs for classification and segmentation has enabled researchers to improve disease detection and streamline data labeling processes. For instance, Waheed et al. [7] effectively employed CNNs to classify skin cancer images, while Hasan et al. [8] applied these models for detecting lung and colon cancers. CNNs have also

been used to identify other complex conditions, including Alzheimer's disease [9, 10] and various vascular abnormalities [11].

CNN-based architectures have also demonstrated remarkable success in medical image segmentation. Shu and al. [12] and Radiuk and al. [13] proposed architectures for advanced multi-organ segmentation by employing region-based CNNs and 3D U-Net models, respectively. In addition, CNNs are highly effective in isolating anatomical anomalies, such as tumors. Bhandari et al. [14] and Abdullah et al. [15] applied CNNs to segment brain tumors with significant accuracy, underscoring the potential of CNN-driven segmentation for detecting various abnormal structures across different imaging modalities.

However, FMD detection needs more research on the possibility of leveraging artificial intelligence to help clinicians diagnose the disease. Building upon the advancements in medical image analysis, **the main objective of this thesis is to investigate the potential usage of 3D CNNs to help detect FMD and determine the eventual challenges inherent to this task**.

To reach our objective, we propose the following contributions:

1. A first 3D CNN architecture in charge of detecting the disease by leveraging the volumetric information in CT scans.
2. The experimentation of four strategies (patch-based learning, self-supervised learning, and using two different sample resolutions) to train our model.
3. The evaluation and comparison of the model's performances across each experiment.
4. The extension of the ROAD score metric calculation from 2D to 3D to assess the quality of our saliency map, which is crucial for interpretability and explainability.

The rest of this work is divided into three parts and organized as follows. The first part, from Chapter 1 to Chapter 4, is dedicated to background knowledge that is helpful to understand our work. The second part, composed of Chapter 5, presents the state-of-the-art CNN models used in medical image analysis. The third part of this document, composed of Chapter 6 and Chapter 7, describes our methodology, our results, and their interpretation. Finally, the last part of this work, Chapter 8, summarizes and concludes this work.

Background Part I

1 Machine Learning

This chapter introduces foundational concepts in machine learning that are valuable for understanding the techniques applied in this research. We present essential elements such as loss functions, cross-validation, gradient accumulation, and evaluation metrics. These topics provide a basis for model training, validation, and performance assessment.

1.1 Loss Function

1.1.1 Binary classification and Binary Cross-Entropy Loss

Binary classification is a type of supervised learning that aims to classify instances into one of two possible classes. Formally, given a dataset $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{N}$, where $x_i \in \mathbb{R}^d$ represents the input features and $y_i \in \{0,1\}$ represents the binary labels, the task is to learn a model $f : \mathbb{R}^d \to \{0,1\}$ that maps input features to binary labels.

The likelihood function measures the probability of observing the provided data given a set of parameters. For a binary classification problem, the likelihood function for a single instance (x_i, y_i) is:

$$\mathbb{P}(y_i|x_i;\theta) = \hat{y}_i^{y_i}(1-\hat{y}_i)^{1-y_i}$$

where $\hat{y}_i = P(y_i = 1|x_i;\theta)$ is the predicted probability that $y_i = 1$ given x_i and parameters θ.

To facilitate the computations, we can define the log-likelihood function for the entire dataset:

$$\log L(\theta) = \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i) \right]$$

As we want to find the set of parameters to obtain the best estimation of the provided data, our objective is to maximize the likelihood function. In addition, as the logarithm is monotonic, maximizing the likelihood is the same as maximizing the log-likelihood. We are therefore

searching a set of parameters θ defined as:

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \log L(\theta)$$

Instead of searching to maximize the function, we will try to minimize the negative version of the log-likelihood, defined as negative log-likelihood (NLL). The NLL for the dataset can be defined as:

$$-\log L(\theta) = -\sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

This negative log-likelihood is also known as the binary cross-entropy loss, which measures the dissimilarity between the predicted probabilities and the actual binary labels. We usually normalize this result to make the loss independent of the number of samples. For the entire dataset, the binary cross-entropy loss is:

$$L(\mathscr{D}, \theta) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

We can then infer the binary cross-entropy loss for a single instance :

$$L(y_i, \hat{y}_i) = -\left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right]$$

1.2 Cross-Validation

Cross-validation is a robust model validation technique used to evaluate how well a machine learning model generalizes to an independent dataset. The most widely used form, k-fold cross-validation, involves partitioning the data into k subsets, named folds. The model is trained on $k-1$ folds and tested on the remaining fold. This process is repeated k times, with each fold serving as the test set exactly once. The final performance metric is typically the average of the results from each fold, providing a more reliable and unbiased estimate of model performance compared to a single train-test split. This method helps mitigate overfitting and ensures that the model's performance is consistent across different subsets of the data.

1.3 Gradient Accumulation

Gradient accumulation is a technique employed in training deep learning models to address memory limitations, particularly when dealing with large batch sizes that may not fit into GPU memory. Instead of updating the model weights after processing each mini-batch, gradient accumulation involves accumulating the gradients over several mini-batches before performing a single update. This approach allows for the effective utilization of large batch sizes without exceeding memory constraints, thereby enhancing model training dynamics

and promoting more stable convergence. By accumulating gradients, models can benefit from the advantages of large batch training, such as improved gradient estimates and smoother optimization paths, while maintaining feasible memory usage.

1.4 Evaluation Metrics

Evaluation metrics are crucial for assessing the performance of binary classification models. They provide insights into how well the model is performing and help identify areas for improvement. One of the fundamental tools for evaluating binary classifiers is the confusion matrix. From the confusion matrix, several metrics can be derived to evaluate the performance of a model. The confusion matrix and the metrics used in this study are described in the following subsections.

The confusion matrix is a table that summarizes the performance of a classification algorithm by displaying the counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions. This matrix provides detailed insights into the types of errors the model is making, which is crucial for improving its performance [16].

	Predicted Positive	Predicted Negative
Actual Positive	TP	FN
Actual Negative	FP	TN

From the confusion matrix, one can derive several metrics to evaluate the performances of a model. The metrics that we used are described in the following subsections.

1.4.1 Precision

Precision, also known as the positive predictive value, measures the accuracy of positive predictions. It is defined as the ratio of TP to the sum of TP and FP. High precision indicates that the model effectively identifies positive instances without including many false positives.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1.1}$$

1.4.2 Recall

Recall, or sensitivity, measures the proportion of actual positives that the model correctly identifies. It is the ratio of TP to the sum of TP and FN. A high recall value indicates that the model successfully captures the most favorable cases, which is essential in contexts such as disease detection, where identifying as many positive cases as possible is critical [17].

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{1.2}$$

1.4.3 F1-Score

The F1-score, introduced by Rijsbergen [18], is the harmonic mean of precision and recall. It balances the two metrics to provide a single score that accounts for both false positives and false negatives. This score is especially useful when there is an uneven class distribution or when both precision and recall are equally important. By taking the harmonic mean, the F1-score ensures that both metrics must be reasonably high for a high F1-score, making it a suitable metric for evaluating models that need a trade-off between precision and recall.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{1.3}$$

1.4.4 ROC and AUC

The Receiver Operating Characteristic (ROC) curve is a graphical representation used to evaluate the performance of a binary classifier. It plots the True Positive Rate (TPR) against the False Positive Rate (FPR) at various threshold settings. The TPR is the recall metric defined above and is also known as sensitivity. The FPR is defined as:

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{1.4}$$

A classifier that performs well will have an ROC curve that rises quickly towards the top-left corner of the plot, indicating a high TPR and a low FPR.

The Area Under the ROC Curve (AUC) is a single scalar value that summarizes the classifier's overall performance. It represents the probability that a randomly chosen positive sample is ranked higher than a randomly chosen negative sample by the classifier. An AUC of 0.5 indicates a model with no discriminative ability, equivalent to random guessing, while an AUC of 1.0 indicates a perfect classifier.

2 CNN Models

2.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are the standard architecture used to solve image processing problems. The first CNN, known as LeNet, was developed for handwritten digit recognition and is attributed to LeCun et al. [19]. Although the LeNet architecture was created in the 1990s, it paved the way for modern architectures based on the same principal components. These components are:

- **Convolutional Layers**: These layers apply a series of learnable filters to the input image to capture spatial hierarchies in the data, such as edges, textures, and patterns. Each filter slides over the image and creates a feature map, enabling the network to detect different visual patterns.
- **Pooling Layers**: These layers reduce the spatial dimensions of feature maps, which decreases computation, controls overfitting, and makes the network more robust to small shifts and distortions in the image. Pooling creates a hierarchical representation of features, from fine-grained to high-level patterns.
- **Fully Connected Layers**: These layers flatten the feature maps into a vector and connect every neuron in one layer to every neuron in the next. They are used to perform final classification tasks based on the features learned in the convolutional and pooling layers.

The core operation in a 2D CNN is the convolution. Mathematically, a 2D convolution operation can be expressed as:

$$(I * K)(x, y) = \sum_m \sum_n I(x - m, y - n) K(m, n)$$

where I is the input image, K is the kernel, and (x, y) are the coordinates of the output feature

map. This operation is repeated for each filter across the entire image, producing multiple feature maps that capture different aspects of the input.

LeNet also demonstrated the successful training of deep networks using backpropagation, which adjusts the network's weights to minimize classification error, typically through gradient descent optimization.

While LeNet set a foundation for CNN-based image recognition, its architecture was relatively simple, with limited depth and computational capacity. Researchers pushed CNN architectures further as computational power and data availability expanded, exploring deeper networks to capture increasingly complex patterns.

Alexnet, developed by Krizhevsky and al. [20], introduced ReLU as an activation function and proposed to mitigate overfitting by using dropout.

Simonyan and Zisserman [21] proposed VGGNet, an architecture leveraging small convolution filters to increase the number of layers. Their results emphasized the importance of depth in CNNs.

Szegedy et al. [22] introduced the concept of inception modules in their architecture called GoogLeNet. The key idea behind inception modules is to implement filters of different sizes inside the same layer, resulting in multi-level feature extraction.

He and al. [23] presented the Residual Network (ResNet), a new architecture introducing skip-connections to solve the vanishing gradient problem, leading to very deep networks.

2.2 3D Convolutional Neural Networks

While 2D CNNs excel at processing spatial information in images, they are limited when it comes to capturing temporal or volumetric data. This limitation led to the development of 3D CNNs, which extend the concept of convolutions to three dimensions, enabling the processing of volumetric data such as video frames and medical imaging. The components of a 3D CNN are very similar to their counterpart in 2D. The main change is that the convolution operation has one more dimension:

$$(I * K)(x, y, z) = \sum_i \sum_j \sum_k I(x - i, y - j, z - k) K(i, j, k)$$

where (I) is the input volume, (K) is the 3D kernel, and ((x, y, z)) are the coordinates of the output feature map. This operation is also repeated for each filter across the entire volume, producing multiple 3D feature maps that capture different aspects of the input.

Initially introduced for action recognition in video data [24], 3D CNNs have become powerful tools in applications that require an understanding of both depth and motion, such as human

action recognition and video classification [25] [26] [27].

A significant milestone in this domain was the introduction of the C3D model by Tran et al. [28], which highlighted the unique advantages of 3D convolutions for extracting spatiotemporal features from video data. Unlike 2D CNNs, which process individual frames independently, the C3D model demonstrated how 3D convolutions could capture temporal dependencies, thereby outperforming 2D counterparts in tasks such as action recognition in videos. This work not only validated the potential of 3D CNNs but also set the stage for further innovations in the field.

Building upon this foundation, Carreira and Zisserman [29] introduced the Inflated 3D (I3D) model, which refined the application of 3D CNNs by inflating 2D convolutional filters into 3D. This approach allowed the I3D model to harness the benefits of pre-trained 2D CNNs, effectively combining spatial feature learning from 2D networks with the temporal modeling capabilities of 3D convolutions. The I3D model achieved state-of-the-art results on multiple video classification benchmarks.

3 Self Supervised Learning

Self-supervised learning is a type of unsupervised learning in which the data itself provides the supervision. The key idea is to create a pretext task from the input data, which generates labels automatically. The model is then trained to solve this pretext task, learning useful representations in the process. Pretext tasks are designed to work without prior labels and without human intervention. Some common pretext tasks include:

- **Predicting Rotations**: The model is trained to predict the rotation angle applied to an image (0, 90, 180, or 270 degrees).
- **Colorization**: The model learns to colorize a grayscale image.
- **Jigsaw Puzzles**: The model is trained to solve jigsaw puzzles created by dividing an image into patches and shuffling them.
- **Contrastive Learning**: The model learns to distinguish between different augmented views of the same data example.

Contrastive learning, presented by Chen and al. [30], is a self-supervised learning approach that learns representations by maximizing agreement between differently augmented views of the same sample.

To achieve this result, SimCLR consists of four main components: a data augmentation module, a neural network encoder, a projection head, and a contrastive loss function. The data augmentation module applies a variety of transformations (such as random cropping, color distortion, and Gaussian blur) to each image to generate two different views, or "positive pairs." These transformations create variation in the input, making the encoder invariant to certain visual changes and encouraging it to focus on more meaningful features.

The augmented images are then passed through a neural network encoder to extract feature representations. Following the encoder, a projection head maps the encoded representations to a latent space where the contrastive loss is applied. As shown by Chen and l. [30], The use

of a projection head improves performance by separating the feature space for the contrastive task from the downstream representation space, allowing for better separation of positive and negative pairs.

The contrastive loss is based on the idea of bringing representations of similar (positive) pairs closer while pushing representations of dissimilar (negative) pairs apart.

More formally, let's define an encoder $f(.)$, a projection head $g(.)$, and let's suppose a mini-batch X of N samples drawn from the set of all samples. For each sample $x_i \in X$, the data augmentation module will generate two views $\hat{x}_i, \hat{x}_j$ of the image x_i, resulting in $2N$ views. Then, we compute $z_i = g(f(\hat{x}_i))$ the representation of $\hat{x}_i$ is the projection space. The whole idea behind the training process proposed by Chal. and al [30] is that for each sample x_i, the pair of images $(\hat{x}_i, \hat{x}_j)$, denoted the positive pair, is more similar than any pairs because it is composed of augmentation of the same image. The representations of those images in the projection space should therefore be close. To push the model to generate representations that follow this idea, they compute, for each positive pairs, the contrastive loss. A common formulation of the loss is the normalized temperature-scaled cross-entropy loss (NT-Xent), which is defined as:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)}$$

where $\text{sim}(\mathbf{u}, \mathbf{v})$ denotes the cosine similarity between vectors $\mathbf{u}$ and $\mathbf{v}$, τ is a temperature parameter, and $\mathbb{1}_{[k \neq i]}$ is an indicator function that equals 1 if $k \neq i$ and 0 otherwise.

The total loss for a batch is computed as the average loss over all positive pairs:

$$\mathscr{L} = \frac{1}{2N} \sum_{k=1}^{N} [\ell_{2k-1,2k} + \ell_{2k,2k-1}]$$

While the encoder $f(.)$ is the most important part of the framework, the projection head $g(.)$ can easily be dropped after the training and replaced by a classifier head.

4 Explainability and Saliencies Analysis

Saliency maps are a powerful visualization tool used to highlight the regions of an image that are most important for CNN when making a prediction. These maps provide insights into the decision-making process of CNNs, making them a crucial component in the development of explainable artificial intelligence (XAI).

The concept of saliency maps originates from the study of human visual attention. In the context of computer vision, a saliency map is an image that reflects the importance of each pixel to the human visual system or a machine learning model. The idea was first introduced to highlight regions in an image that attract human attention, which was later adapted for use in CNNs to understand model predictions.

One of the first works in this area is by Simonyan et al. [31], who proposed a method to visualize the class-specific saliency maps by computing the gradient of the class score with respect to the input image. This approach allows for the identification of image regions that most influence the model's output.

Grad-CAM (Gradient-weighted Class Activation Mapping) is an advanced technique for producing visual explanations from CNNs. It was introduced by Selvaraju et al.[32] and provides a way to visualize the important regions in an image for a specific class prediction .

The method works by using the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map. The steps to compute Grad-CAM are as follows:

1. Compute the gradient of the score for class c, y^c, with respect to the feature map activations A^k of a convolutional layer:

$$\frac{\partial y^c}{\partial A^k}$$

2. Compute the weights α_k^c by averaging these gradients:

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{ij}^k}$$

where Z is the number of pixels in the feature map.

3. Compute the weighted combination of the feature maps, followed by a ReLU activation:

$$L^c_{\text{Grad-CAM}} = \text{ReLU}\left(\sum_k \alpha_k^c A^k \right)$$

This results in a heatmap $L^c_{\text{Grad-CAM}}$ that highlights the important regions in the image for predicting class c.

To ensure that saliency maps are a meaningful representation of the model decision process, one can evaluate the importance of highlighted regions by systematically perturbing the input image and observing the impact on the model's output. Common perturbation-based metrics include Deletion and Insertion metrics, which measure the change in model performance when important regions are removed or added back [33].

In their work, Samek and al. [34] propose to verify the quality of the saliency by removing the most important pixels from the original image and analyse the prediction of the model on the altered image.

However, Hooker and al. [35] argue that this method may modify the pixel distribution and therefore may not be trustable. To mitigate the effect of pixel perturbation, they proposed Remove and Retrain (ROAR) which consists of retraining a model after removing the part of the image that the saliency indicates as the most important on the full dataset. According to them, if one sees a drop of accuracy in the newly-trained model, then the saliency were meaningful. A big drawback of this method is that you need to fully retrain a model on the perturbated images, which can be costly.

In addition to the computational cost, Rong and al. [36] show that information leakage can occur in the image due to the perturbations and ROAR is also affected by this. The ROAD framework overcomes these issues, offering a more efficient and theoretically grounded approach. Feature removal strategies form the core of ROAD's evaluation methodology, employing two complementary techniques: Most Relevant First (MoRF) and Least Relevant First (LeRF). MoRF removes features in descending order of importance, expecting a rapid drop in model performance if the attribution method correctly identifies crucial features. Conversely, LeRF removes features in ascending order of importance, with model performance expected to remain stable if unimportant features are accurately identified. Together, these strategies assess attribution methods' ability to distinguish between important and negligible features. MoRF evaluates whether the most relevant features are indeed critical for model decisions, while LeRF verifies that least relevant features can be removed with minimal impact, providing

complementary insights into attribution quality. [36].The key component of the strategy is the Noisy Linear Imputation introduced by the author. While applying MoRF or LeRF, they replace the removed features by an average of the neighboring features. Direct neighbors are weighted with $w_d = \frac{1}{6}$, while indirect neighbors use $w_i = \frac{1}{12}$. This imputation strategy mitigates the information leakage due to removal of patterns, while avoiding the computationally expensive retraining step of prior methods such as ROAR.

State of The Art Part II

5 3D CNN models for medical image analysis

Over the past decade, 3D Convolutional Neural Networks (3D CNNs) have emerged as powerful tools for analyzing volumetric data, such as computed tomography (CT) and magnetic resonance imaging (MRI). These models enable the extraction of spatial dependencies across all three dimensions, making them highly effective for complex tasks, including tumor detection, organ segmentation, and anomaly localization. However, despite their versatility, to the best of our knowledge, no work has been conducted,either in 2D or 3D,toward the detection of fibromuscular dysplasia (FMD), leaving a significant gap in the application of machine learning to this rare vascular condition.

5.1 Advances in Segmentation Task

Among segmentation architectures, U-Net has established itself as a cornerstone in medical image analysis. By employing an encoder-decoder structure, U-Net effectively captures both local and global context in 2D and 3D images. Building on this foundation, Chen et al. [37] proposed the S3D-UNet, a novel adaptation that incorporates separable 3D convolutions, initialy used in video processing, to reduce computational complexity while maintaining high accuracy. Evaluated on a dataset of approximately 300 patients, S3D-UNet achieved Dice scores of 0.689, 0.839, and 0.783 for enhancing tumors, whole tumors, and tumor cores, respectively.

Similarly, DeepMedic, developed by Kamnitsas et al.[38], introduced an 11-layer deep 3D CNN tailored for brain lesion segmentation. Notably, it integrated a 3D Conditional Random Field (CRF) as a post-processing step, refining model outputs and boosting segmentation accuracy for complex lesions. DeepMedic's success in tackling high-variability cases highlights the utility of post-processing frameworks in conjunction with deep learning architectures, particularly for subtle and challenging abnormalities.

Dou and al. [39] present a two-stage framework for the automatic detection of cerebral microbleeds in susceptibility-weighted imaging volumes, leveraging 3D convolutional neural

networks. The first stage screens the entire volume to quickly retrieve potential positive candidates with high sensitivity, assisting radiologists in narrowing down the candidates for further inspection. The second stage uses a 3D CNN to accurately distinguish true positives from false positives, enhancing detection precision. The method significantly improves detection efficiency and accuracy, outperforming existing approaches in terms of sensitivity and false positive reduction.

5.2 Advances in Classification Task

For classification tasks, 3D CNNs such as ResNet and VGG have shown remarkable capabilities. Yang and al. [40] compared several 3D architectures, including 3D VGG and three variants of 3D ResNet, for the binary classification of Alzheimer's disease. Using a dataset of approximately 100 patients (50% with Alzheimer's disease), their models achieved accuracies ranging from 0.58 to 0.79, with basic 3D ResNet outperforming other architectures. Additionally, they explored explainability techniques, to provide visual insights into model predictions, emphasizing the growing importance of interpretability in clinical applications.

Further expanding on classification approaches, Kruthika and al. [41] combined 3D Capsule Networks, 3D CNNs, and pre-trained 3D autoencoders to classify Alzheimer's disease stages. With a dataset of around 1,800 samples, they demonstrated the superior effectiveness of Capsule Networks over traditional 3D CNNs, particularly in capturing spatial hierarchies within volumetric data. These findings highlight the potential of more advanced architectures for distinguishing subtle disease progression. .

In another domain, Noel and al. [42] applied 3D CNNs for gender classification based on human skull morphology. Using a balanced dataset of 97 skulls extracted from CT scans, they evaluated ResNet, MeshNet, and PointNet architectures, focusing on accuracy. This study underscores the flexibility of 3D architectures in adapting to various types of medical and anatomical data.

5.3 Challenges

The application of 3D Convolutional Neural Networks (3D CNNs) in medical imaging has shown considerable promise in tasks such as tumor detection, organ segmentation, and anomaly localization. However, several key challenges persist, particularly in their application to rare and underexplored conditions like *fibromuscular dysplasia* (FMD). 3D CNNs typically require large, annotated datasets for effective training, a requirement that poses significant barriers for rare conditions. While research has shown promising results with small datasets, the quality of the data is more impacting with hundreds of samples than with large scale datasets of several thousands of samples [43]. Additionally, datasets often suffer from class imbalance, with normal cases vastly outnumbering rare or subtle anomalies.

The “black box” nature of 3D CNNs limits their clinical applicability, as understanding the reasoning behind model decisions is critical in medical contexts. Although interpretability methods have been developed, their application to 3D models is not mature and has to be explored. [44]

The high dimensionality of 3D volumetric data leads to increased computational demands for training and inference. This creates challenges for widespread adoption, particularly for researchers and institutions with limited computational resources. [44] [43] [45]

Present Work Part III

6 Methodology

6.1 Extending ROAD score

In this section, we present the main idea behind our extension of the ROAD score for 3D tensors.

We extending the logic of Rong and al. [36] to find the weights to attribute to direct and indirect neighbors, to the third dimension. The cube on Figure 6.1 shows visually the distance from each voxels to the center. In addition, it also shows the equivalent 2D version proposed by Rong. We kept the same color code betwen 2D and 3D to better show the correspondance between both versions. Following the idea of the author of the metric, the direct neighbors (in red) of the central voxel should weight 2 times more than the indirect ones (in blue). Similarly the direct neighbor should weight 3 times more than the corner voxels at distance 3 (in green). Taking into account that, in the normal case, we have a total of 26 neighbors for a voxel (6 direct, 12 indirect and 8 corner) and that the weights should sum up to 1, it is possible to calculate the value of each weights: Direct neighbors are weighted with $w_d = \frac{3}{62}$, indirect neighbors $w_i = \frac{3}{124}$ and neighbors located in the corners $w_c = \frac{1}{62}$.

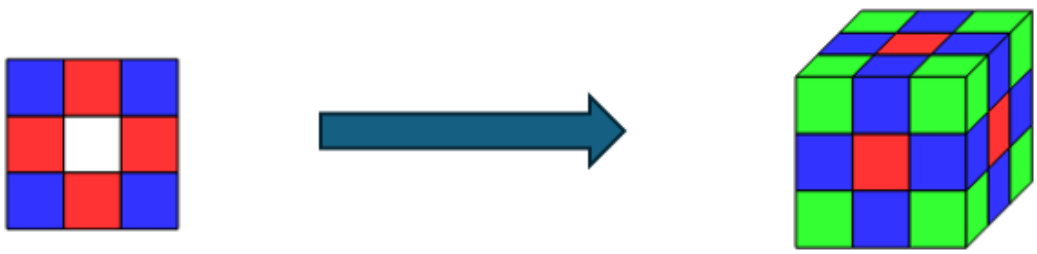


Figure 6.1: Illustration of distance between pixels used to obtain the weight matrix in 2D converted to 3D voxels. In red, pixels located at one unit distance, in blue pixels located at two units. Green represents the voxels located at 3 unit distance.

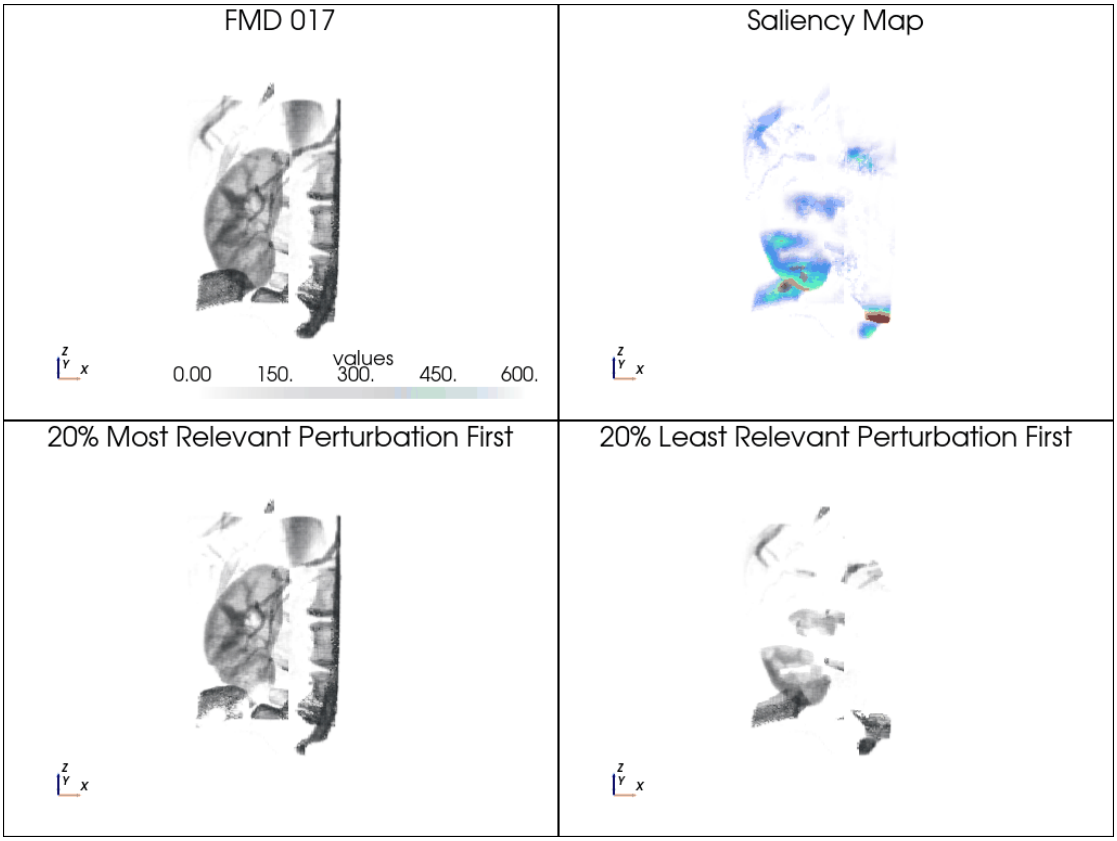


Figure 6.2: Visual representation of the effect of the perturbations applied to the input volume while computing ROAD score for patient FMD017. The top left corner shows the original volume, and the top right corner shows the saliency map. The bottom left image shows the volume after applying MoRF, and the bottom right shows the volume after applying LeRF.

For the technical implementation, we leverage the implementation of the ROAD score from the pytorch-grad-cam library [46] to extend it for our purpose. We essentially applied minor modifications to the road.py script to support 3D tensors and to update the matrix of weights to use $w_d = \frac{3}{62}$, $w_i = \frac{3}{124}$ and $w_c = \frac{1}{62}$. Figure 6.2 shows the effect of applying MoRF and LeRF on a 3D volume after we modify the original script. In the bottom left image, one can see that the part of the volume appearing bright red/yellow on the saliency map (bottom of the kidney and small part of the artery next to the spine) has been perturbated. On the opposite, on the bottom right image, the part of the volume not highlighted by the saliency map has been perturbated.

6.2 Datasets

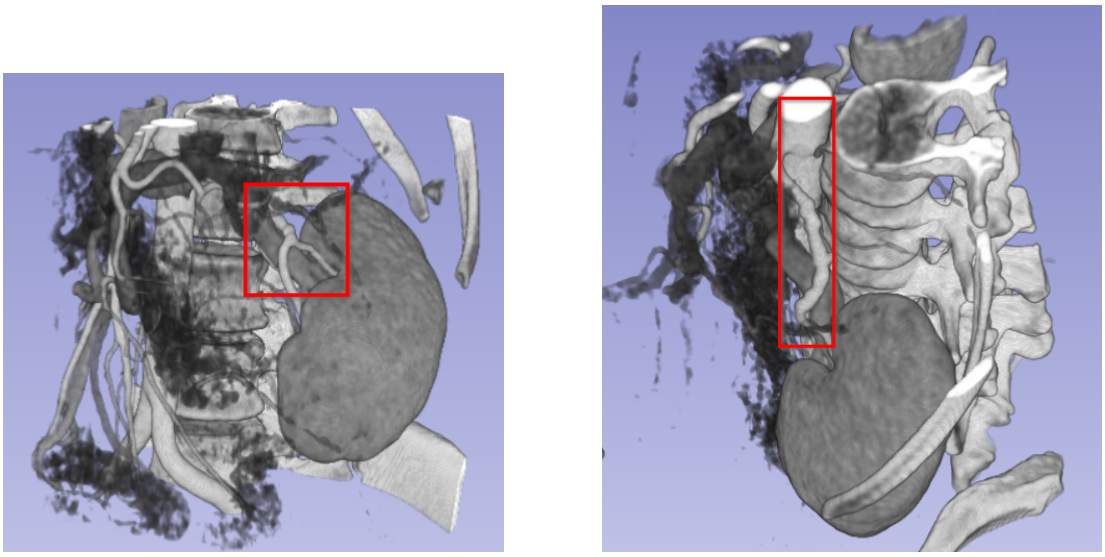
This section provides an overview of the datasets used in our experiments. The first dataset, called AIFMD, is the main data source we used to train and evaluate our model. Due to the small size of the dataset, we leverage two public datasets, namely C4KC-KiTS and CT-ORG,

initially created for studies on diseases different from FMD, to increase the number of available samples in some of our experiments.

Each of the following subsection describes the specific properties for each dataset.

6.2.1 AIFMD

The AIFMD dataset is composed of chest CT scans from 161 patients, including 63 cases diagnosed with the target condition and 98 healthy controls. The dataset is highly heterogeneous due to variations in imaging protocols, methodologies, and the timing between contrast agent injection and scan acquisition. The relatively small dataset size poses a challenge for training and evaluating deep learning models, especially in 3D. Figure 6.3 shows the kidney of a patient who has fibromuscular dysplasia. The beads on the arteria (inside the red square) are the signal of the disease that radiologist use for the diagnose. This is therefore one of the signal we expect our model to detect.



(a) Front View (b) Side View

Figure 6.3: CT scan image of the right kidney region of an AIFMD patient, highlighting the affected artery (marked with a red square). The beading pattern on the arteria is characteristic of fibromuscular dysplasia.

In addition to the scans, the dataset contains two masks per patient. Each mask delimits an area starting from one of the kidney and going towards the center of the body up to the vertebral column. Figure 6.5 shows a CT scan with such masks. Both green squares represent the bounding area around each kidney. The main reason to take the center of the body in the area is that, as shown on Figure 6.3, the arteries showing signs of FMD are going from the kidney to the aorta, which is located next to the vertebral column.

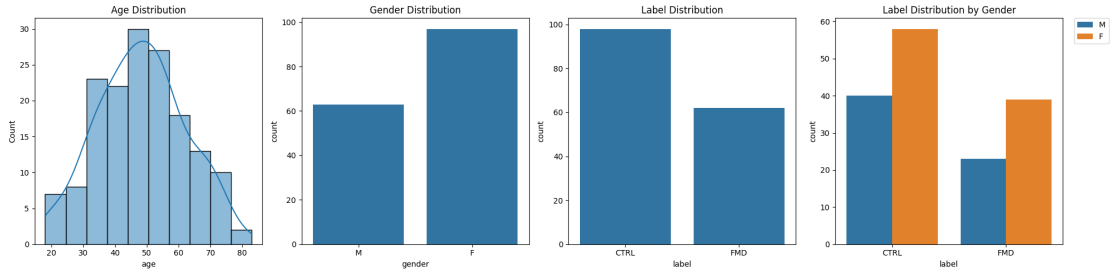


Figure 6.4: Metadata related to AIFMD dataset. From left to right: age distribution, gender distribution, label distribution and label distribution per gender.

Figure 6.4 shows the available metadata for the AIFMD dataset. In addition to the imbalance between the label that we already mentionned, the second plot shows that there are more female than male in the dataset. When looking at the distribution of label by gender, we can see that we have more control patient than positive patient for both genders. However, there is around 2 times more positive women than men. While it is not surprising because FMD is more frequent in women patient as explained in Section , we have to be careful to ensure that our model is effectively learning FMD patterns and not gender related patterns.

From a technical perspective, all images in the AIFMD dataset are full chest CT scans stored in the DICOM (Digital Imaging and Communications in Medicine) format, a standard widely adopted in medical imaging for preserving image quality and metadata. The heterogeneity of the dataset emphasizes the importance of preprocessing steps, such as normalization, to facilitate consistent analysis.

6.2.2 C4KC-KiTS

The C4KC-KiTS dataset is composed of CT scans from 210 patients. A part of the dataset originates from the 2019 Kidney and Kidney Tumor Segmentation Challenge [47]. These scans were acquired as part of clinical care for patients at the University of Minnesota Medical Center. The authors indicate that the dataset contains scans generated by different institutions. This implies that the scans vary according to the scanner manufacturers and acquisition protocols of the referring institutions and the dataset is therefore highly heterogenous. Thanks to this diversity, C4KC-KiTS is an ideal addition to our primary dataset, as it effectively expands the number of samples. In addition to the scans, the dataset contains masks defining the exact shape of the kidneys.

On a technical level, all images are stored in DICOM format, preserving both image quality and metadata crucial for medical imaging applications. The segmentations provided within the dataset were created manually by medical students, ensuring a baseline of accuracy. Although clinical data of the patients is also available in the C4KC-KiTS dataset, we are focusing on the imaging data and does not utilize the clinical information and metadata for this dataset.

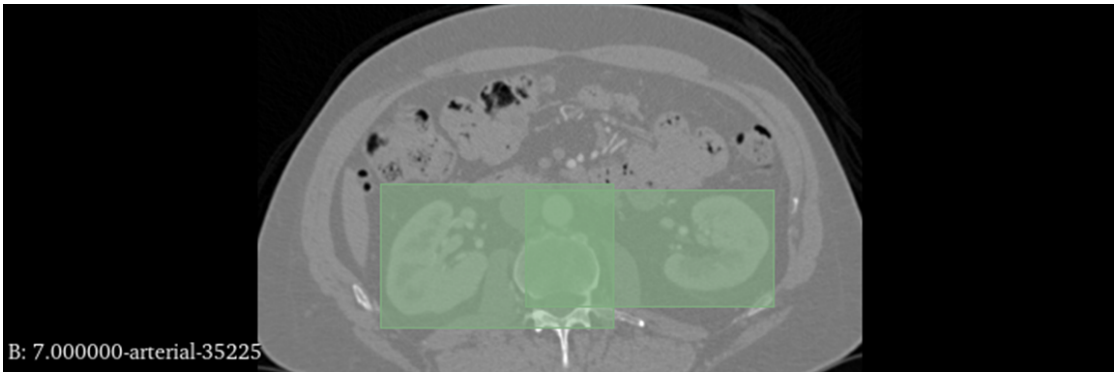


Figure 6.5: Example of area of interest drawn around each kidneys

6.2.3 CT-ORG

The CT-ORG dataset consists of 140 CT scans with labeled anatomical regions across five different organs [48]. Each patient in this dataset has a lesion in at least one of the five organs, with the liver being the most commonly affected. This dataset is highly heterogeneous, incorporating images from various sources with a mix of contrast-enhanced, non-contrast, low-dose, and high-dose scans. Once again, this diversity makes CT-ORG an ideal candidate to extend our primary dataset. The dataset contains one mask file per patient, defining the exact shape of the five different organs containing lesions.

On the technical side, the dataset is stored in the Nifti format, using 32-bit floating-point data to maintain high precision in voxel values. While some organs (such as lungs and bones) were segmented automatically, kidneys were manually segmented to ensure accuracy.

6.2.4 Preprocessing

In this section, we explain the different pre-processing steps that we applied to each dataset for all of our experiments. Some additional pre-processing techniques were applied individually for some specific experiments. Those additional steps are discussed later in the respective experiment sections.

We started by removing from the datasets samples with missing essential information, such as missing one or both kidney's masks. Additionally, samples where only one kidney was present were excluded to ensure consistency across all cases.

In order to reduce the computational load and the memory footprint, we decided to focus only on the kidneys area. We created an area of interest (AOI) around the kidneys for each scan. In addition to the technical constraints, this choice is motivated by the fact that given the limited dataset size, processing the entire scan to detect FMD would most likely result in searching a needle in a haystack. Instead, we leveraged prior knowledge as the kidneys region is known to be an area used by radiologists to diagnose the disease.

The generation of the AOI was different for each dataset. For the AIFMD dataset, the area of interest was defined by combining the masks of both kidneys to determine the bounding area around them. In the case of the C4KC-KiTS dataset, we calculated the minimum and maximum coordinates along each axis from each kidney's mask, defining an area that encompasses both kidneys accurately. For the CT-ORG dataset, we first isolated the label corresponding to the kidneys from the mask. We then split the mask along the spine axis to obtain separate masks for each kidney and applied the same bounding method as used in C4KC-KiTS to establish the area of interest.

As we have heterogenous datasets with scans coming from different brand and models of scanners, and generated following different methodologies, our final step was to normalize pixel spacing to 1mm along each axis for all samples across the datasets. This uniform voxel size ensures consistent spatial resolution. Normalizing pixel spacing allows for a more reliable comparison between scans originating from different imaging sources and acquisition protocols.

Table 6.1 shows the number of samples before and after preprocessing for each dataset. The total number of patients available is 503.

Dataset	Total Samples (Before Preprocessing)	Total Samples (After Preprocessing)
AIFMD	161	161
C4KC-KiTS	202	167
CT-ORG	140	134

Table 6.1: Number of samples per dataset before and after preprocessing

6.2.5 Data Augmentation

Given the limited amount of data at our disposal, we took advantage of several well-known data augmentation techniques to virtually increase the number of samples in order to improve the generalization ability of the 3D CNN and prevent overfitting. Similarly to Krizhevsky and al. [49], data augmentation was applied on the fly during training, allowing the model to see different variations of the input data at each epoch. The augmentation methods were chosen to simulate real-world variations while maintaining the underlying anatomical structures in the 3D volumetric data.

The first method we used is intensity transformations. This method was applied to simulate variations in the contrast and brightness of the scans, which often appears due to differences in scanning equipment, protocols, or patient characteristics. Random adjustments to the intensity values of the 3D images were performed by scaling pixel intensities globally or locally. This ensures that the model remains robust to variations in contrast, brightness, and noise commonly found in real-world datasets.

We also used random affine transformation to increase the spatial variability of the data. The

transformations included translations, rotations, and scaling along the three spatial axes (x, y, and z). This method simulates changes in patient positioning or slight misalignments during data acquisition. By randomly rotating and shifting the 3D volumes, the network learns to recognize features regardless of their spatial orientation, improving its ability to generalize to unseen data with different orientations.

Finally, we applied random flipping along the three different axis. This type of augmentation introduces variations in the symmetry of the input data, which is especially important in cases where anatomical structures exhibit some degree of symmetry. This technique helps the model to learn invariant features that are robust to lateral changes in orientation.

6.3 Architectures

In this section, we explained the deep learning architecture employed in our experiment. We begin by introducing the 3D Convolutional Neural Network model. Following this, we present the modification applied to the network to obtain the Domain-Adversarial Neural Network (DANN) that we used to address domain shift issues by leveraging domain adaptation techniques. Finally, we present the modifications applied to our network to be able to train it with self-supervised learning techniques.

6.3.1 CNN

Our 3D Convolutional Neural Network (CNN) consists of two main components: the Feature Extractor and the Classifier.

The feature extractor is composed of five convolutional blocks. Each block contains two 3D convolutional layers, each followed by batch normalization and ReLU activation. The blocks progressively increase the number of channels from 16 to 256, while reducing the spatial dimensions through max-pooling layers. The feature extractor employs residual connections after each block, which allow the network to bypass certain layers and help to avoid the gradient vanishing problem. After the fifth block, global average pooling is applied to compress the feature maps into a 256-dimensional vector, representing the high-level features of the input data.

The output from the feature extractor is passed through the classifier head which is composed of two fully connected (FC) layers. The first FC layer reduces the dimensionality from 256 to 128, followed by a ReLU activation function. Dropout is applied to prevent overfitting. The second FC layer outputs the final classification result. As we are using the BCEWithLogitsLoss from PyTorch, we do not directly apply the sigmoid function to the output of the second FC layer.

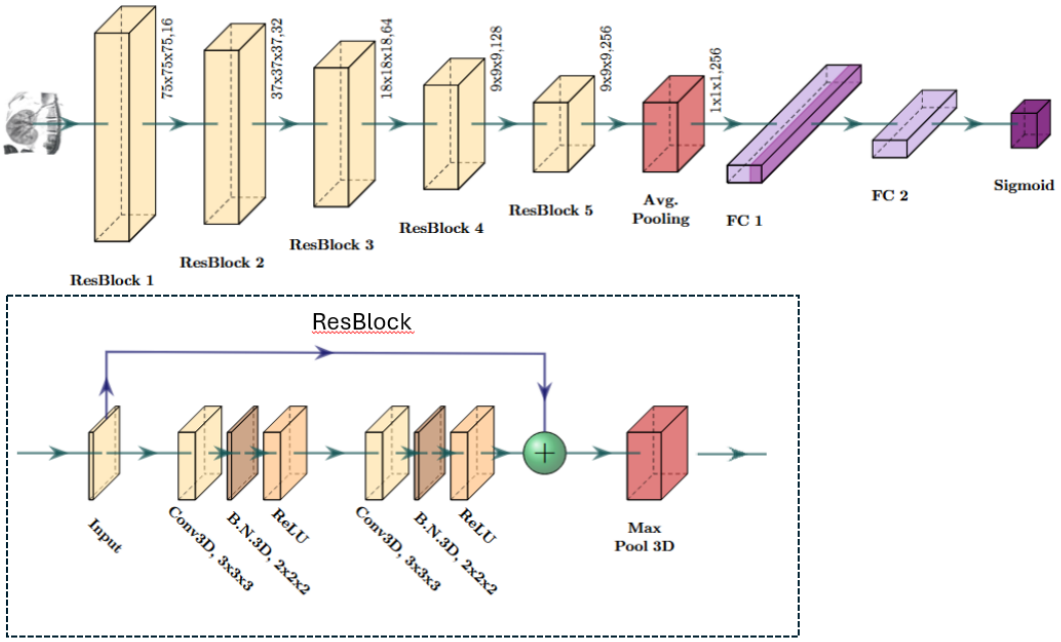


Figure 6.6: Architecture of our CNN

6.3.2 DANN

To use domain adversarial learning as presented by Ganin and al. [50], we needed to modify our architecture by adding two new modules. First, the gradient reversal layer (GRL) which is simply forwarding its input during the forward propagation and is multiplying the gradient by -1 during the backward propagation. The second module is a second head, in charge of classifying the samples in the correct dataset. This module is composed of two FC layers to reduce dimensionality from 256 to 128 and from 128 to 3. To implement the forward propagation in PyTorch, we had to start by calling the GRL, then calling the first FC layer followed by a relu. Finally, we called the second FC layer. Finally,we had to modify the output of the model. Instead of returning only the logit for the FMD classification, we also return the result for the classification in dataset.

6.3.3 SSL CNN

To adapt our model presented in section 6.3.1 for contrastive learning, we need to modify the model to output a representation of the input. The main change is that we added a projection head to the model. The projection head, composed of two FC layers reducing the dimensionality to 128, is used for the pre-training and is then replaced by the classifier head. The feature extractor remains unchanged.

6.4 Model Evaluation

We assessed the performances of the different models proposed in our experiments using a combination of quantitative metrics and interpretability tools to ensure both the accuracy and reliability of results.

To ensure the robustness of the models, we trained them until signs of overfitting emerged. A 10-fold cross-validation approach was employed to validate the model's consistency. The results across the folds were then averaged to provide a reliable estimate of the model's performance.

We used precision, recall, f1-score, and ROC-AUC metrics to quantify the model's predictive performance as defined in Section 1.4. The combination of those metrics allows us to understand better the model's strengths and weaknesses in FMD detection.

In addition to quantitative evaluation, we leverage saliency maps to get an idea of the model's decision-making process. Our objective was to 1) understand what a model with bad performances is looking at and 2) ensure that a model with good performances focuses on features aligned with human intuition and domain knowledge. In our case, we expect the model to focus either on the kidneys or the arteries leading to them. The quality of the generated saliency maps was assessed using the ROAD score. All saliency maps presented in this report combine a high prediction score and a high ROAD score computed.

6.5 Experimental Protocols

In this section, we describe the experiments we designed to train and evaluate the performance of our proposed model. We split our experiments into four different subsections, each representing a major specificity either in the data preparation or in the model architecture. For each set of experiments, we describe the specific data pre-processing steps done in addition to the general pre-processing described in Section 6.2.4. Then, we present the training procedure. As each set contains several experiments, we describe the parameters used for the training in Section 7.

6.5.1 Experiment Set 1: Patches

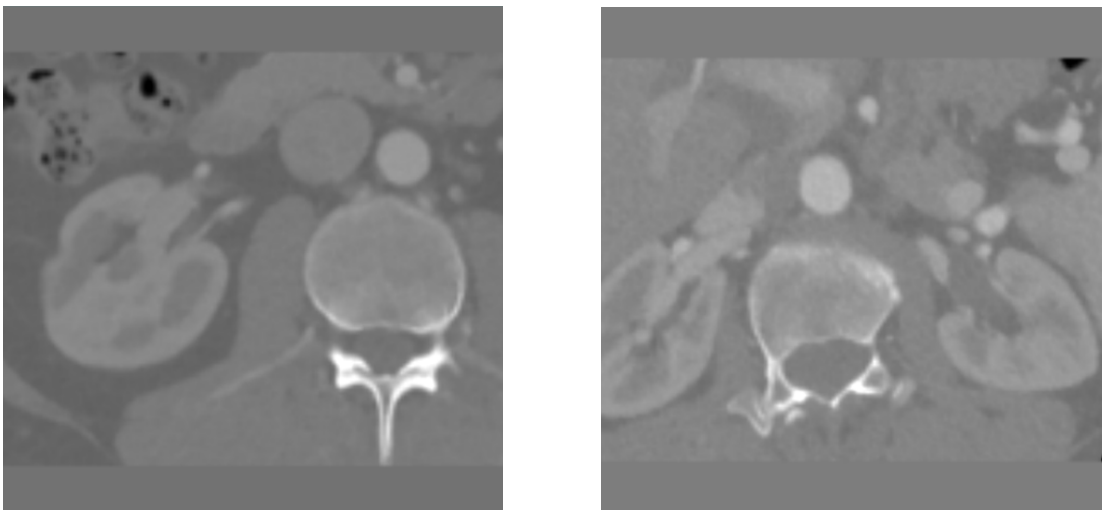
In this first set of experiments, we wanted to assess the effect of dividing each of our volume samples into non-overlapping smaller patches. While we fixed the patch dimension per experiment, we used 32x32x32, 64×64×64, and 110x110x100 patches. When the total volume dimension was not a multiple of the chosen dimensions, we added padding to complete the patch. To avoid presenting patches full of padding to the model, we dropped the patches containing more than 25% of padding. The dropped parts are usually located in corners of the area of interest, which, according to prior knowledge, do not relate to the disease. Each patch was assigned the patient's label from which it was taken. The idea behind this patch-based

approach is that 1) applying data augmentation on patches can improve and increase the model performances, and 2) we can leverage the patches to see which region gets higher scores. However, we foresee a potential risk to this strategy: the patches may not necessarily contain regions that are relevant to the disease. As a result, it may introduce difficulties for the model to learn meaningful signals of the FMD disease, which may impact the performance of our model.

6.5.2 Experiment 2: Full Volumes

In this second experiment, we considered an alternative approach that involves fitting the entire area of interest into a single volume of dimensions 150×150×150. This method ensures that for each patient, only one volume is generated, thereby eliminating any potential bias introduced by creating multiple patches per patient. However, a notable limitation of this approach is whether the chosen resolution is sufficient to capture the relevant disease signals. If critical features are lost or obscured due to resolution constraints, it could adversely affect the ability of the model to distinguish between classes effectively.

Figure 6.7b shows an example of 150x150x150 containing the pre-processed area of interest around both kidneys for the patients of the AIFMD dataset.



(a) One Kidney (b) Both Kidneys

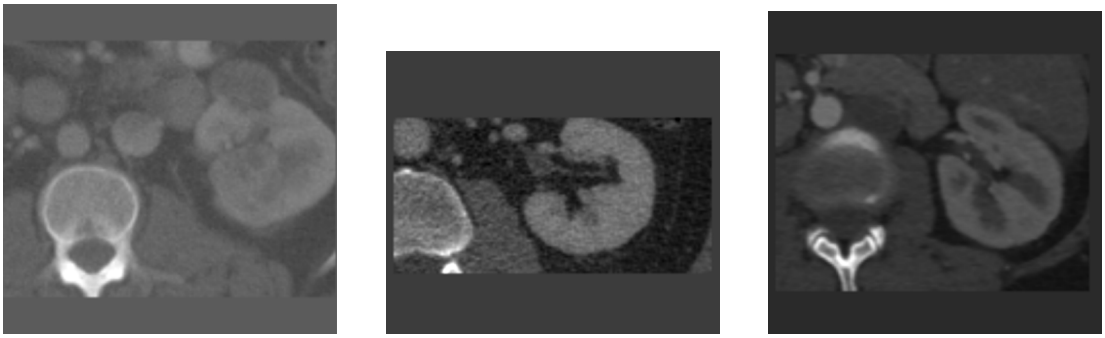
Figure 6.7: 150x150x150 volumes containing one kidney (left) and two Kidneys (right) after pre-processing for samples in the AIFMD dataset.

6.5.3 Experiment 3: Pretrained Model

In this experiment, we employed a contrastive learning approach to pretrain the model before fine-tuning it for the FMD detection task. The pretraining phase allows the model to learn

generalizable features from a large dataset, providing a strong foundation for subsequent fine-tuning on the specific task. This strategy offers two key advantages: first, the model benefits from the rich feature representations acquired during pretraining, potentially enhancing its performance on the target task; second, the training process is independent of biases (large class imbalance) introduced by the Kits and CT-ORG datasets, ensuring a more robust evaluation. However, this method also presents challenges, such as the potential misalignment of features learned during pretraining with those most relevant to the FMD detection task.

The dimension of input volumes during pre-training and fine-tuning is 150x150x150. Each sample corresponds to the AOI around a single kidney. If the original AOI is too small, some padding is added around the volume to match the expected dimension. If the original volume is too big, it is cropped. Figure 6.8 shows a random example coming from each dataset after fitting the data to the 150x150x150 volume.



(a) KiTS (b) CT-ORG (c) AIFMD

Figure 6.8: CT scan images of the right kidney area of a randomly chosen patient from each dataset. The images were taken after pre-processing to fit a 150x150x150 volume.

6.5.4 Experiment 4: One Kidney Per Volume

In this experiment, we divided the area of interest into two separate volumes, each covering one kidney, with dimensions 150×150×150.Figure 6.7a shows an example of the input that we feed to the model in this set of experiments. This approach results in two volumes per patient, one for each kidney. The primary advantage of this method is the improved resolution within each volume, which may enhance the model's ability to capture finer details relevant to disease detection. However, this setup introduces a potential drawback: the signs of the disease might only be present on one side, which could lead to incomplete or biased model predictions.

6.6 Hardware

We conducted our experiments on a high-performance computation grid managed by SLURM workload manager. Developped by Jette and al.[51], SLURM allows a flexible and efficient

allocation of resources across the multiple nodes of the grid. We ran all of our experiments using the same configuration. To handle the heavy computation, we requested an NVIDIA RTX 3090 GPU, 4 CPU cores were used, mainly to handle file system operations. Finally, 32 GB of RAM were allocated for our experiments.

7 Results and discussion

In this section, we present and discuss the results of the different sets of experiments. We present the best results obtained for each set of experiments. When relevant, we also provide results obtained with specific setups if they provide meaningful insights. The results are evaluated in terms of the model performance. By comparing the advantages and limitations of each strategy, we aim to provide a comprehensive understanding of how these techniques influence the detection of FMD signals and overall model robustness. Additionally, the discussion explores potential challenges and future directions for improving accuracy and reliability in similar applications.

7.1 Experiment 1: Patches

7.1.1 Results

In this set of experiments, we tried to use volumes of size 32x32x32, 64x64x64, 110x110x110. For all size of volumes, we obtained the best results by training the model over 250 epochs with a learning rate of 0.01 using Adam optimizer and a dropout of 0.3. and by applying data augmentation. In the 32x32x32 setting, we used a batch size of 16. For the other settings, we used a batch size of 5 with gradient accumulation over 3 epochs.

Table 7.1 shows the performances over the best fold for each of the subsets obtained by our best model trained with patches of size 32x32x32. While the recall seems to stay consistent with a value higher than 98% over all the sets, the other metrics are less stable. The model performed the worst for the specificity by obtaining less than 15% for each subset. F1-score reached 70% for two of the three subsets, due to the good recall of the model. The precision is indeed around 50% for all subsets.

Table 7.2 shows in detail the results obtained for each of the 10-folds for our model trained with patches of size 32x32x32. One can observe overall poor performance of the model and the variability of performances across the different folds is quiet high. For example, fold-8

barely reaches 42.8% accuracy while fold-9 reaches 55.4%. Despite the poor performances, the high Recall across folds shows the model's ability to detect true positives effectively.

Table 7.3 presents the averaged performances of our model trained with patches of size 32x32x32 over the 10-folds. The results shows that the model perform close to random for accuracy and precision but has high recall.

Table 7.4 shows the performances over the best fold for each of the subsets obtained by our best model trained with patches of size 110x110x110. This model is quiet stable. All subsets obtained an accuracy greater than 60%, a f1-score greater than 65% and a recall greater than 70%. The worst metric is once again the specificity a maximum value of 55% on the train set.

Table 7.5 shows in detail the results obtained for each of the 10-folds for our model trained with patches of size 110x110x110. While the precision of the model varies from 44% to 70% across the folds, its recall is stable across the folds with a minimum value of 64% for fold-1. This results in precision and recall leads to a f1-score geater or egal to 60% for all folds.

Table 7.6 presents the averaged performances of our model trained with patches of size 110x110x110 over the 10-folds. The results shows that the model obtained a score of 57+% for all metrics.

subset	num_samples	precision	recall	f1	specificity	roc_auc	accuracy
train	9988	0.535	0.981	0.692	0.131	0.716	0.560
validation	3239	0.411	0.980	0.579	0.048	0.562	0.425
test	3257	0.531	0.985	0.690	0.117	0.563	0.554

Table 7.1: Performance metrics for train, validation, and test subsets on our best model trained with patches of size 32x32x32

	Fold-0	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9
Precision	0.492	0.474	0.520	**0.534**	0.522	0.504	0.461	0.497	0.423	0.531
Recall	0.830	0.833	0.964	0.890	0.826	0.764	**0.987**	0.896	0.959	0.985
F1	0.618	0.604	0.676	0.668	0.640	0.607	0.628	0.639	0.587	**0.690**
Accuracy	0.505	0.506	0.547	0.533	0.521	0.524	0.468	0.486	0.428	**0.554**

Table 7.2: Performance metrics across 10 folds for model trained with patches of size 32x32x32. Bold values indicate the best scores for each metric.

	Min	Max	Avg.	Median
Accuracy	0.428	0.554	0.507 ± 0.032	0.515
Precision	0.423	0.534	0.496 ± 0.027	0.499
Recall	0.764	0.987	0.893 ± 0.06	0.893
F1	0.587	0.690	0.636 ± 0.025	0.634

Table 7.3: Summary statistics for performance metrics across folds for model trained with patches of size 32x32x32

subset	num_samples	precision	recall	f1	specificity	roc_auc	accuracy
train	1781	0.641	0.759	0.695	0.551	0.737	0.658
validation	277	0.572	0.828	0.677	0.420	0.645	0.617
test	485	0.704	0.725	0.714	0.483	0.651	0.635

Table 7.4: Performance metrics for train, validation, and test subsets on our best model trained with patches of size 110x110x110

	Fold-0	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9
Precision	0.546	0.566	0.481	0.581	0.537	0.475	0.523	0.513	**0.704**	0.441
Recall	0.914	0.637	0.996	0.819	**1.000**	0.983	0.760	0.842	0.725	**1.000**
F1	0.684	0.599	0.649	0.679	0.699	0.640	0.619	0.638	**0.714**	0.612
Accuracy	0.534	0.552	0.495	0.579	0.551	0.474	0.535	0.529	**0.635**	0.441

Table 7.5: Performance metrics across 10 folds for model trained with patches of size 110x110x110. Bold values indicate the best scores for each metric.

	Min	Max	Avg.	Median
Accuracy	0.441	0.635	0.577 ± 0.045	0.551
Precision	0.441	0.704	0.571 ± 0.046	0.581
Recall	0.637	1.000	0.716 ± 0.038	0.679
F1	0.599	0.714	0.653 ± 0.0381	0.637

Table 7.6: Summary statistics for performance metrics across folds for model trained with patches of size 110x110x110

7.1.2 Discussion

The results obtained show that the model is not able to learn any signal of FMD from the patches. While the model has an excellent recall when trained with small patches, the precision and accuracy are similar to a random model. When trained with bigger patches, the recall of the model stays relatively high and stable and the precision and accuracy are relatively higher. However, the results still don't show any significant learning.

We suspect that the difference of performances is due to the fact that with smaller patches, the total number of samples is bigger. Therefore, for positive patients, more samples containing no FMD signal are marked as positive and could therefore mislead the model.

7.2 Experiment 2: Full Volumes

7.2.1 Results

In this set of experiments, we tried different setups of datasets (aifmd only, aifmd + KiTS, aifmd + CT-ORG, aifmd + KiTS + CT-ORG) with different configuration of the hyper-parameters. The best results were obtained by training the model with the combination of all three datasets

over 500 epochs, with a batch-size of 7, with a learning rate of 0.01 and Adam optimizer. We used a dropout of 0.3 in the classifier and a dropout value of 0.4 in the feature extractor. To better understand the performance of the model, we used the three datasets in the train and validation sets. The test set is composed of 12 control, respectively 12 positive patients.

Table 7.7 shows the performances over the best fold for each of the subsets obtained by our best model trained on full volumes. While the model has a consistent recall higher than 90% across all subsets, other metrics such as specificity, roc_auc and accuracy decrease significantly on the unseen data from the test subset, droping respectively from 85%+ to 27%, from 90%+ to 63% and from 89% to 66%.

Table 7.8 shows in detail the results obtained for each of the 10-folds for our model trained on full volumes. Fold-0 shows the best Precision and Accuracy with 87%, respectively 69%. However, it also has the worst recall with a small 53%. In the opposite, Fold-5 obtained a 100% recall and a strong 76% F1-Score. In addition, its accuracy and precision are better than average.

Table 7.9 presents the averaged performances of our model trained on full volumes over the 10-folds. While the averaged metrics seems to be quiet stable, the consistent big gap between minimum and maximum values for each metric suggests instability across the folds.

subset	num_samples	precision	recall	f1	specificity	roc_auc	accuracy
train	255	0.578	0.974	0.725	0.876	0.952	0.890
validation	148	0.423	0.917	0.579	0.890	0.917	0.892
test	24	0.619	1.000	0.765	0.273	0.636	0.667

Table 7.7: Performance metrics for train, validation, and test subsets for best model trained on sample containing both kidneys.

	Fold-0	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9
Precision	**0.875**	0.632	0.579	0.526	0.545	0.619	0.667	0.360	0.423	0.643
Recall	0.538	0.923	0.846	0.769	0.923	**1.000**	0.769	0.692	0.846	0.692
F1	0.667	0.750	0.688	0.625	0.686	**0.765**	0.714	0.474	0.564	0.667
Accuracy	**0.696**	0.652	0.655	0.500	0.542	0.667	0.600	0.333	0.433	0.550

Table 7.8: Performance metrics across 10 folds for best models trained on sample containing both kidneys. Bold values indicate the best scores for each metric.

	Min	Max	Avg.	Median
Accuracy	0.333	0.696	0.563 ± 0.086	0.571
Precision	0.360	0.875	0.587 ± 0.113	0.587
Recall	0.538	1.000	0.800 ± 0.087	0.808
F1	0.474	0.765	0.660 ± 0.067	0.667

Table 7.9: Summary statistics for performance metrics across 10 folds for best model trained on sample containing both kidneys.

7.2.2 Discussion

By looking at Figure 7.7 we can see that the model has good accuracy, specificity and roc_auc on the train and validation sets. However, the results drop significantly on the test set. This indicate that while the model can detect all positive samples, it also outputs a lot of false positive. The high drop in specificity for the test set, which contains only aifmd samples, suggests that the model find a way to discriminate the datasets and is not learning patterns related to FMD. To tackle this problem, we tried to leverage the work of Ganin and al. [50] and trained our network using the architecture described in 6.3.2. While it seems to improve the ROC AUC of the model, it lowered other essential metrics. The results of this experiment can be found in Appendix C

7.3 Experiment 3: Pretrained Model

7.3.1 Results

In this section, we present the results we obtained with a model pre-trained using contrastive learning. We first trained the model using the projection head described in section 6.3.3 for 500 epochs with ADAM optimizer and a learning rate of 0.001. In this configuration, we reduced the batch-size to 3 to be able to fit the two augmentations of each sample of a batch in the 24Go of memory available.We used all three datasets for the pre-training. Once the model was pretrained, we switched back to the classification head and trained the model for another 250 epochs using ADAM optimizer,a learning rate of 0.001 and a batch size of 7. Dropout regularization was applied both within the classifier and between convolutional blocks, with a dropout rate of 0.4. For the fine-tuning, we only used the aifmd dataset.

Table 7.10 shows the performances over Fold-0 which is one of the best fold for each of the subsets obtained by our pretrained model. While the model shows stable accuracy and roc_auc, both higher than 60% across the different subsets, the rest of the metric are either stable but close random such as precision with about 52% on each subset, or unstable such as recall with 81% on the validation set and 45% on the test set.

Table 7.11 shows in detail the results obtained for each of the 10-folds by our pretrained model. Overall, the model performed poorly on all metrics. Even if the model obtained an accuracy higher than 60% on Fold-0 and maximum recall on Fold-1, Fold-5 and Fold-7, its best precision is only 52% on Fold-0 and Fold-8 obtained a recall of 18%.

Table 7.12 presents the averaged performances of our pretrained model. Even if the average recall is 62%, the model shows its instability by having a standard deviation of 0.193 on this metric and by obtaining recall score between 18% and 100%. The model does not even reach an average of 50% on the other metrics, the precision being the worst with an average of 40%.

subset	num_samples	precision	recall	f1	specificity	roc_auc	accuracy
train	224	0.523	0.511	0.517	0.699	0.642	0.625
validation	40	0.520	0.812	0.634	0.500	0.641	0.625
test	58	0.526	0.455	0.488	0.750	0.702	0.638

Table 7.10: Performance metrics for train, validation, and test subsets for best model pretrained using constrastive learning.

	Fold-0	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9
Precision	**0.526**	0.415	0.250	0.432	0.302	0.458	0.400	0.423	0.400	0.429
Recall	0.455	**1.000**	0.318	0.864	0.591	**1.000**	0.636	**1.000**	0.182	0.818
F1	0.488	0.587	0.280	0.576	0.400	**0.629**	0.491	0.595	0.250	0.562
Accuracy	**0.638**	0.466	0.379	0.517	0.328	0.552	0.500	0.483	0.586	0.517

Table 7.11: Performance metrics across 10 folds with model pretrained using constrastive learning. Bold values indicate the best scores for each metric.

	Min	Max	Avg.	Median
Accuracy	0.328	0.638	0.496 ± 0.059	0.508
Precision	0.250	0.526	0.4035 ± 0.0555	0.4165
Recall	0.182	1.000	0.6864 ± 0.193	0.727
F1	0.250	0.629	0.485 ± 0.082	0.489

Table 7.12: Summary statistics for performance metrics across folds with model pretrained using constrastive learning.

7.3.2 Discussion

The pretrained model is clearly unstable and does not seem to learn any FMD signal. Indeed, metrics such as recall ranges from 18% to 100%. In addition, the average accuracy, precision and f1-score don't even reach 50%. The saliency map on Figure 7.1 confirms that the model is not looking at FMD signal. Indeed, the figure shows that the model is focusing on the bottom right part of the volume which isis the location of the vertebral column.

7.4 Experiment 4: One Kidney Per Volume

7.4.1 Results

In this set of experiments, we used only the aifmd dataset and we tried different configuration of pre-processing, hyper-parameters. By observing the classification made by our model on the different folds and with different configuration, we identified 44 "hard samples" that have never been correctly classified. An example of the impact of the hard samples is shown in Appendix A.

The best results were obtained by removing those hard samples, by clamping the intensity of

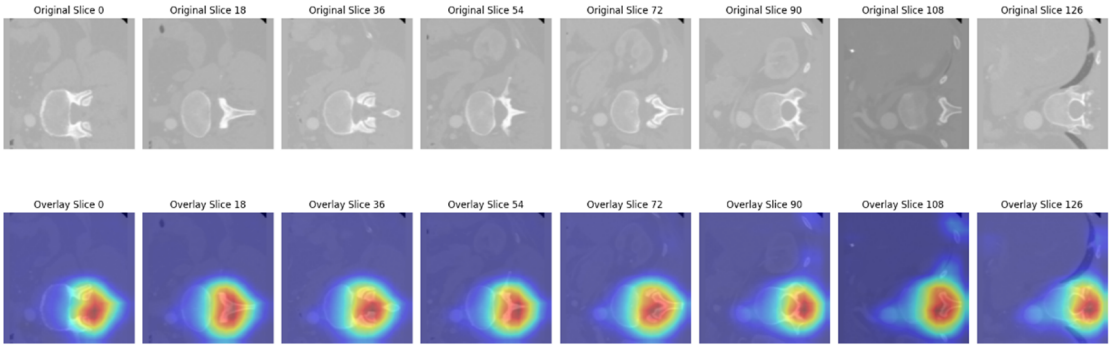


Figure 7.1: 2D saliency map for patient FMD30 on pretrained model. On the top, 8 slices taken from the full volumes. On the bottom, the same slice with the saliency map overlay.

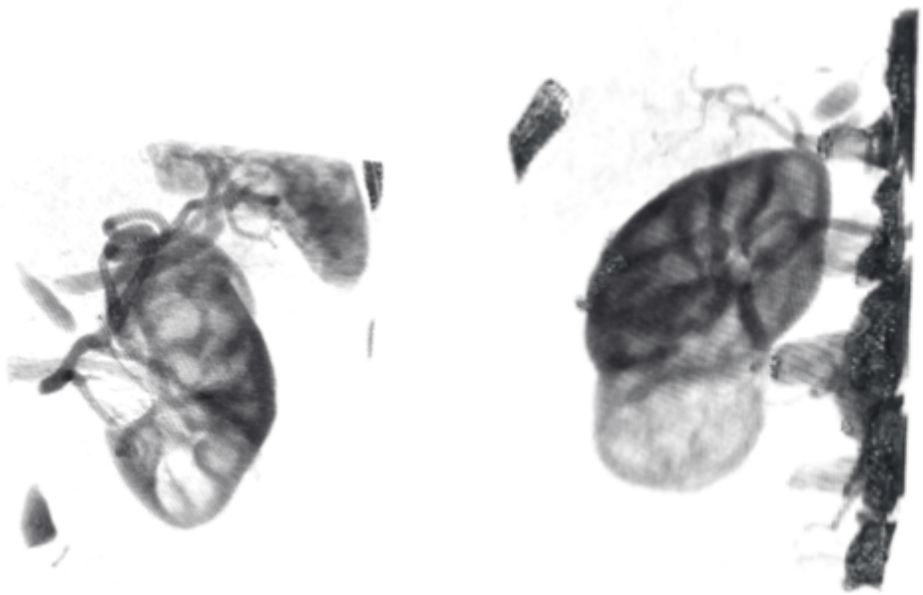
the samples in the range 150-500 and by cutting away the spine. For this last point, we applied a global threshold to remove the part of the volume containing the spine. This choice was guided by the clues obtained from saliencies on other experiments which show that the model is looking at the spine. While this method works well for the majority of samples, some of them still have remaining of the spine. Figure 7.2a show the effect of the threshold when it successfully cut the spine and Figure 7.2b shows one of the sample which still have a small part of the spine.

We then trained the model from scratch over 250 epochs, with a batch-size of 5, with a learning rate of 0.01 and Adam optimizer. We used a dropout of 0.4 for the classifier and the feature extractor.

Table 7.13 shows the performances over the best fold for each of the subsets obtained by our best model trained on one-kidney volumes. The results show that on its best fold, the model performed a score higher than 60% on each metric and for each subsets. The best results are achieved on the recall, which takes value from 74% for the training set to 87% on the validation set. In addition, the F1-score is higher than 65% for each subsets, which indicate consistent results for the precision and recall.

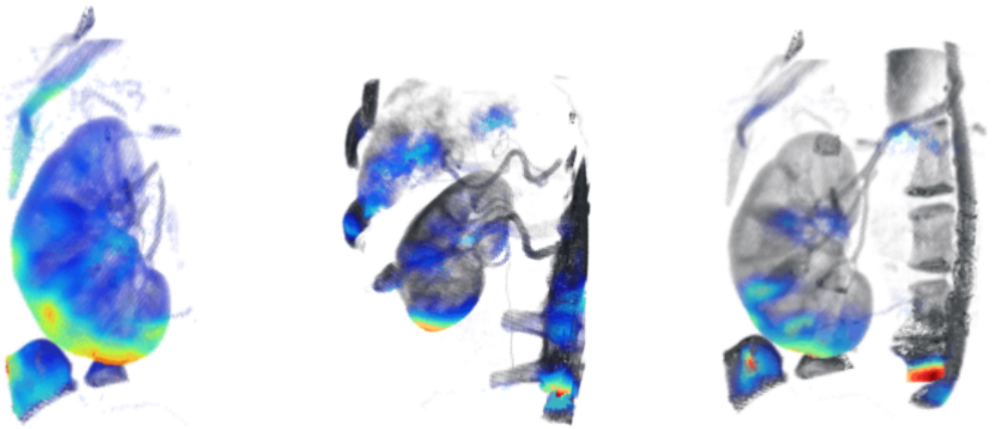
Table 7.14 shows in detail the results obtained for each of the 10-folds by our model trained on one-kidney volumes. While Fold-1, described in details in Table 7.13, obtained the best results on precision, F1-score and accuracy, Fold-9 obtained the worst overall results with a precision of 53% and a recall of 40%. Those results led to a F1-score of 46% whicih is the lowest result across the 10 folds.

Table 7.15 presents the averaged performances of our model trained on one-kidney volumes. In average, our model obtained results higher than 55% for all metrics and only precision stands below 60%. The median is close to the average and is higher than 55% for each metrics.



(a) (b)

Figure 7.2: Example of input volumes with the the cut of spine. On the left, the threshold cuts correctly the full spine. On the right, the threshold cuts most of the spine.



(a) (b) (c)

Figure 7.3: Saliency map for three different samples on the AIFMD dataset.

7.4.2 Discussion

The results presented in Figure 7.15 shows significative improvement after removing the hard samples. The model shows consistent results higher than 60% for all metrics except precision

subset	num_samples	precision	recall	f1	specificity	roc_auc	accuracy
train	193	0.613	0.739	0.670	0.610	0.757	0.668
validation	36	0.700	0.875	0.778	0.700	0.797	0.778
test	47	0.654	0.773	0.708	0.640	0.685	0.702

Table 7.13: Performance metrics for train, validation, and test subsets for best model trained with one-kidney volumes after removing hard samples.

	Fold-0	Fold-1	Fold-2	Fold-3	Fold-4	Fold-5	Fold-6	Fold-7	Fold-8	Fold-9
Precision	0.571	**0.654**	0.630	0.577	0.500	0.475	0.630	0.577	0.545	0.529
Recall	0.727	0.773	0.773	0.682	0.773	**0.864**	0.773	0.682	0.545	0.409
F1	0.640	**0.708**	0.694	0.625	0.607	0.613	0.694	0.625	0.545	0.462
Accuracy	0.660	**0.702**	0.681	0.600	0.551	0.520	0.681	0.647	0.600	0.596

Table 7.14: Performance metrics across 10 folds with model trained with one-kidney volumes after removing hard samples. Bold values indicate the best scores for each metric.

	Min	Max	Avg.	Median
Accuracy	0.520	0.702	0.6238	0.6235
Precision	0.475	0.654	0.5688	0.574
Recall	0.409	0.864	0.7001	0.750
F1	0.462	0.708	0.6213	0.625

Table 7.15: Summary statistics for performance metrics across folds with model trained with one-kidney volumes after removing hard samples.

which is slightly below. We wanted to make sure that this problem with "hard" samples was not due to a difficulty in learning some features because of the relatively small size of our model ($\tilde{4}$ million parameters). Therefore, we tried to make our feature extractor deeper and trained a model of $\tilde{1}5$ million parameters. However, the performance did not improve. The result of the experiment can be seen in Appendix B.

While there is still a lot of space for improvement, those results suggest that our model is learning some pattern from the data and is therefore slightly better than random guess. In addition, by looking at the saliency maps on Figure 7.3 we can see that the model is first looking at the bottom part of the kidney and a little to the spine. This shows that even if small portions of the spine remain on some samples, the cutting threshold helps to improve the results. However, two new questions emerged from those results:

- To which extend the learnt pattern is related to FMD ?
- How to explain that the hard samples are impossible to classify correctly ?

To answer the first question we decided to further analyze the results to search for potential correlations with patient metadata, especially the gender. We focused on the gender because

it has been shown that FMD is more frequently discovered in women patient[1]. Figure 7.4 shows the gender distribution of misclassified patients for control (7.4a), respectively fmd patients(7.4b) for each of the 10 folds of our experiments without hard samples. The plot shows the misclassified male patietn in blue, respectively the female patient in red. We observed on Figure 7.4a that there is slightly more control woman classified as positive. In the opposite, Figure 7.4b shows more male positive patient misclassified as control.

In order to confirm our visual assumption, we ran a Chi-Square statistic test to assess a possible correlation between the prediction done by the model and the gender of the patient. Figure 7.5 shows the result of the test over the 10 folds. The plot shows the output of the test(7.5a as well as the p-value(7.5b). To help interpretation, we draw the significance threshold (0.05) in red. With all values below the threshold, we clearly see that there is a strong correlation between the gender and the classifications of positive patient. For the control patient, we can see that the median is slightly above the threshold, which suggest a correlation, but weaker than for the positive patient.

The second question is however harder to answer. We did not find any relevant correlation between the hard samples and the metadata. In addition, we did not see any significant differences by inspecting them visually. We are suspecting that:

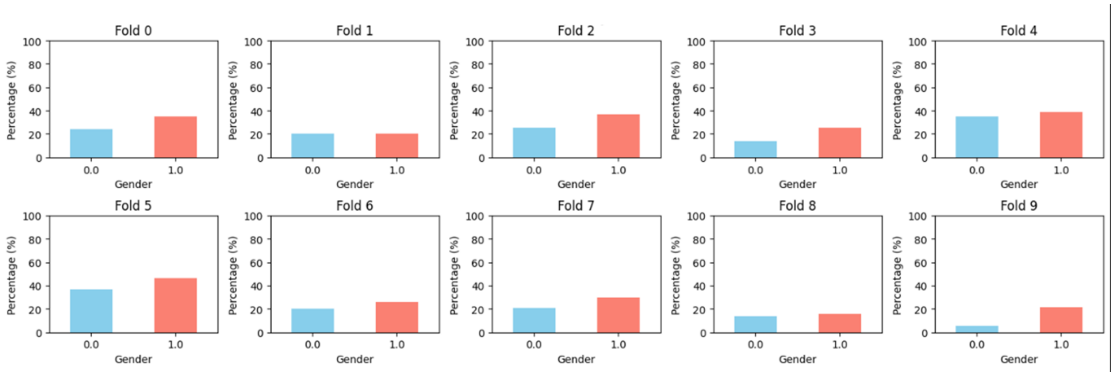
- those specific scans may have intensity range slightly different from the other samples which leads to removal of important features (ie: arteries / kidneys) when we apply our generic clamping.
- the model tries to classify the samples by gender and those specific samples have morphological features close to the other gender (ie: space between organs, size of kidney, etc.)

However, those are only assumptions and we need to run further analysis to confirm them.

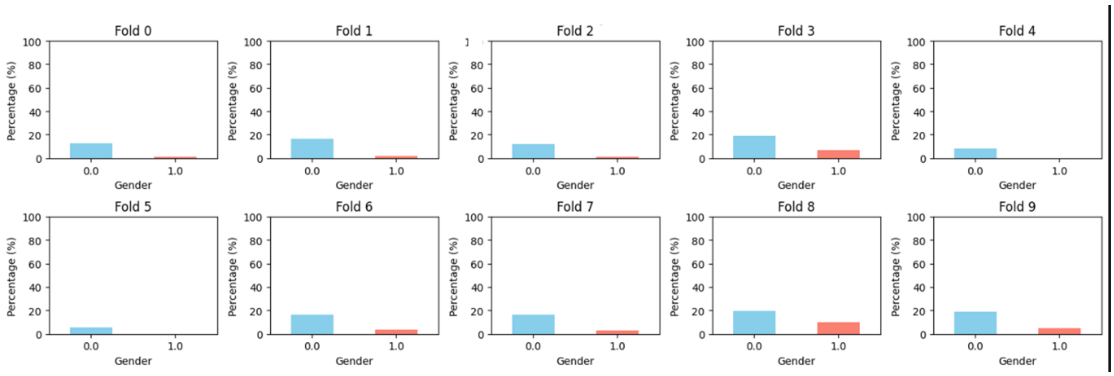
7.5 General Discussion

The results of our experiments highlight several key insights into the performance of our model.

One notable finding is that using one volume per kidney seems to be a good trade-off between the number of samples and the quality per sample. While data augmentation is mandatory to virtually increase the size of our dataset, leveraging prior knowledge to remove unessential features (ie: vertebral column) is as much important. With such pre-processing, the model consistently outperforms both splitting volumes into smaller patches and maintaining the full area of interest in a single volume. This aligns with our hypothesis that dividing the volumes into patches labeled with the global status of the patient may prevent the model from converging.



(a) CTRL Patients

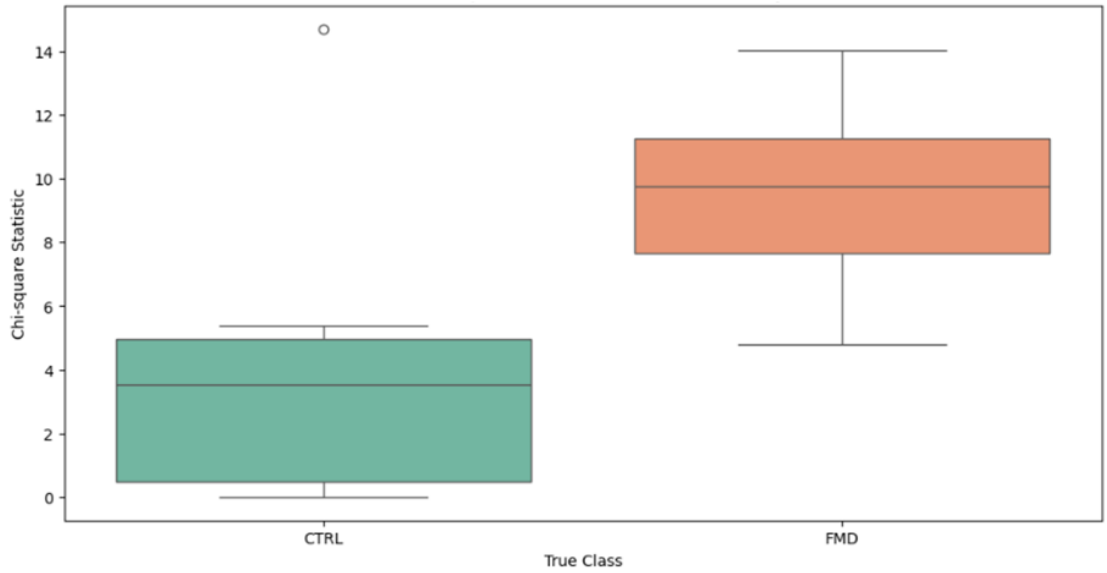


(b) FMD Patients

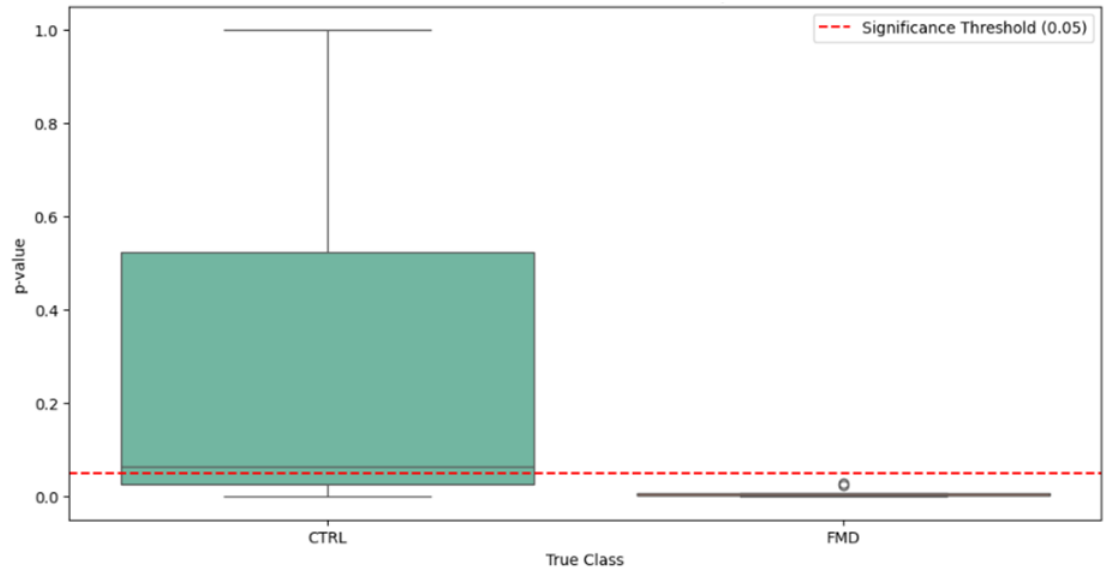
Figure 7.4: Distribution of misclassified patients by gender for each of the 10 folds. Subplots show control patients, respectively fmd patients.

Another insight is that models pre-trained with constraint learning seem to have a tendency to focus on the vertebral column as shown on Figure 7.1 which leads to poor performance in FMD detection.

Finally, the gender bias inherent to the disease is also reflected in our dataset and has been corroborated by our analysis. A chi-square test confirmed a significant correlation between the model's predictions and gender, suggesting that the model may be leveraging gender-specific features as part of its decision-making process. Although this correlation highlights the model's sensitivity to anatomical differences, it also raises concerns about potential biases in the classification process. Addressing this issue will require careful consideration in future iterations.



(a) Chi-square value



(b) P-value

Figure 7.5: Result of Chi-Square test

Conclusion Part IV

8 Conclusion

In this document, we presented an approach to automated detection of fibromuscular dysplasia using 3D convolutional networks.

Our contribution is a first step towards FMD detection using deep learning by proposing a 3D CNN architecture to detect the disease. We also proposed to experiment with four strategies to handle the small dataset size inherent to rare diseases in medical imaging. In addition, we evaluate the performances of the model for each strategy. Finally, as explainability is crucial in medical imaging, we leverage GradCam++ to provide an explanation of the inferences made by our model by computing volumetric saliency mapping. Moreover, we proposed to extend the ROAD score metric calculation to 3D models to ensure that saliency maps are a good representation of the model decision process.

Our experiments show that FMD detection leveraging volumetric data from a dataset with less than 200 patients is a complex task. Our attempts to use patch-based learning, to use contrastive learning, and extract patterns from the entire area of interest (both kidneys per sample) did not lead to a converging model. We obtained promising signs of FMD detection by generating one sample per kidney and by refining the samples to remove as much noise as possible based on prior medical knowledge.

Future work can improve several areas of our research. In particular, our current pipeline, which removes noisy elements from the volume according to prior knowledge, must be refined and automated. Indeed, the current thresholds are set globally, which can lead to significant differences between volumes. Furthermore, we must run further analysis to understand if some specific samples are never classified correctly because of our current pre-processing pipeline, some common noisy pattern, or any other reason. Finally, we should tackle the gender bias problem to ensure that our model can effectively learn patterns related to FMD.

To address those issues, instead of focusing on the whole area of interest, a possible direction would be to train a model to isolate the arteries and leverage it to focus the training of our FMD classifier on this specific area. Another direction would be to extend the dataset with

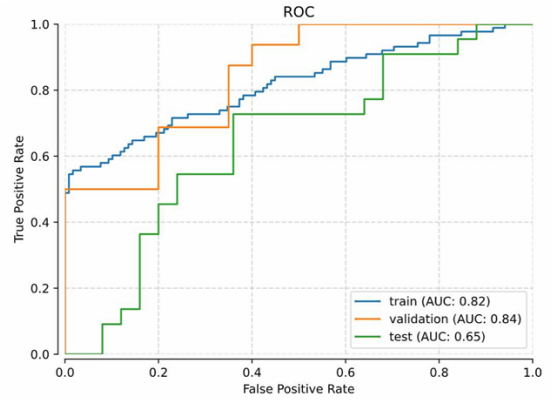
positive patients to help better capture the pattern of FMD. However, this solution may take time to implement because it requires manual work and is usually a lengthy and costly process. Finally, changing the representation of the data by using a spiral-spinning technique [52] could allow us to leverage transfer learning from 2D models to mitigate our current limitations.

A Effect of the "hard samples" on the performances of a model

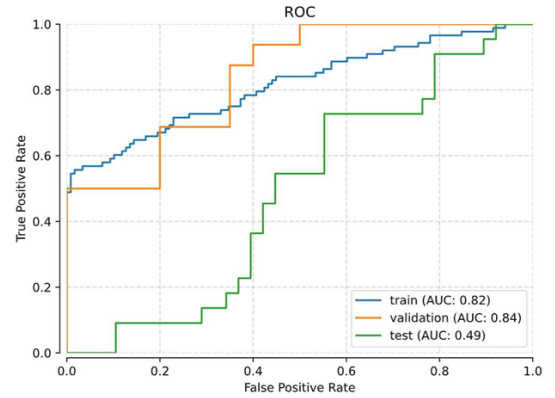
To show the effect of the control samples which are never correctly classify, we trained a model from scratch on AIFMD dataset by ensuring that there was no hard samples in the train and validation sets. We clamped the intensity of the samples in the range 150-500 and we trained the model over 250 epochs, with a batch-size of 5, with a learning rate of 0.01 and Adam optimizer. We used a dropout of 0.4 for the classifier and the feature extractor. After the training, we built two test sets. One composed of 47 samples without any hard samples and a second one with the exact same 47 samples and adding 13 hard samples. Figure A.1 shows the results we obtained with the model. One can see that the precision dropped from 59% to 40% and the accuracy dropped from 63% to 50%. The recall did not change because all of the hard samples are coming from control patients. Figure A.1 shows the ROC curve obtained during the experiment. While the ROC curve for the test set (in green) follows the trend of the other two curves in our experiment without hard samples, one can clearly see that the results get worse when we add the hard samples. In addition, the AUC of 0.49 indicate that the model is acting randomly when the hard samples are added.

subset	num_samples	precision	recall	f1	roc_auc	accuracy
train	206	0.565	0.841	0.676	0.816	0.655
validation	36	0.652	0.938	0.769	0.841	0.750
test no hard	47	0.593	0.727	0.653	0.645	0.638
test with hard	60	0.400	0.727	0.516	0.487	0.500

Table A.1: Comparison of the performance of a model on a test set with vs. without the hard samples. The subset "test no hard" does not contain hard samples. "test with hard" contains the hard samples.



(a)



(b)

Figure A.1: ROC curves. On the top, the test curve is obtained using the test set without hard samples. On the bottom, the curve is obtained using the hard samples in the test set. For both figures, the train and validation curve are identic as they are coming from the same configuration.

B Performance of deeper model on test set with "hard" samples

To understand if a deeper model would help to better classify the hard samples, we modified our model to make its feature extractor deeper by adding additional ResBlock to increase the number of parameter from 4 millions to 15 millions parameters. We then trained the model from scratch on AIFMD dataset. We clamped the intensity of the samples in the range 150-500 and we trained the model over 250 epochs, with a batch-size of 5, with a learning rate of 0.01 and Adam optimizer. We used a dropout of 0.4 for the classifier and the feature extractor. Figure B.1 shows the results we obtained with the model. One can see that while the training performance shows that the model is learning something from the data, the performance drops significantly for the validation and test set. Figure B.1 shows the ROC curve obtained during the experiment. The curves tend to confirm the results presented in the Table B.1. By comparing this model with the 4 million parameters version, one can see that a shallower model performs better than a deeper one for our task. This is certainly due to the fact that our dataset is pretty small and a deeper model requires more data for effective training.

subset	num_samples	precision	recall	f1	roc_auc	accuracy
train	193	0.613	0.739	0.670	0.748	0.668
validation	47	0.438	0.875	0.583	0.631	0.574
test	80	0.250	0.636	0.359	0.376	0.375

Table B.1: Performance of a 15 million parameters model on a test set with the hard samples. The subset.

Appendix B. Performance of deeper model on test set with "hard" samples

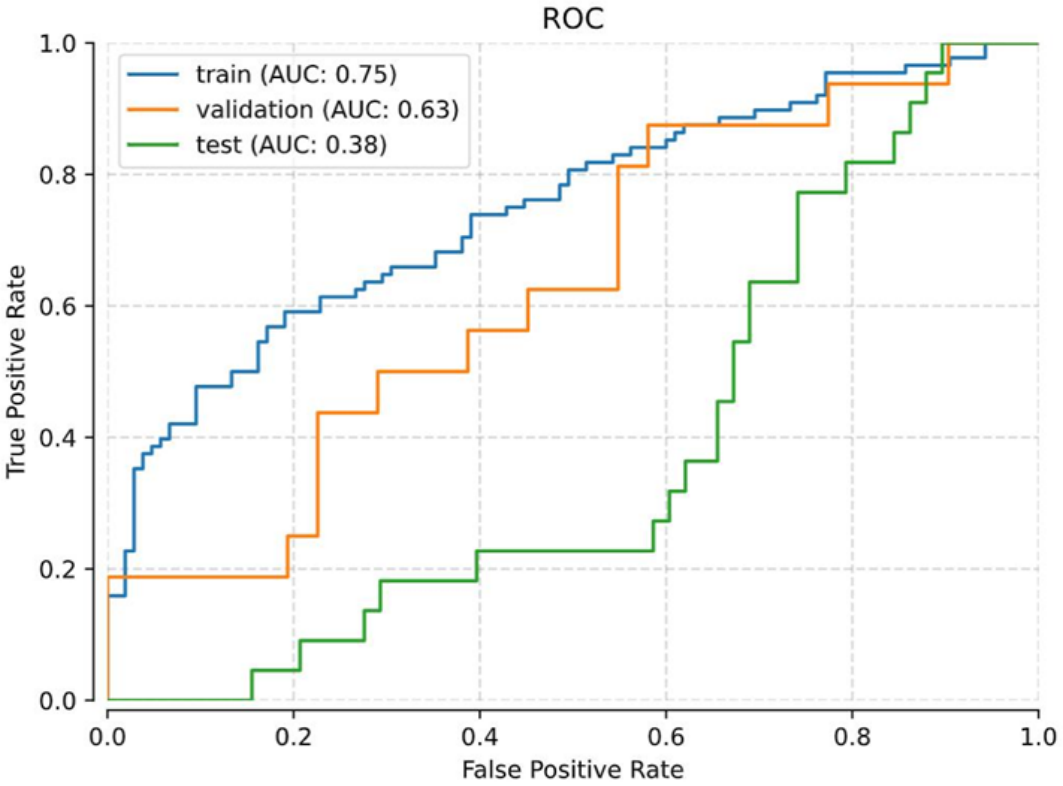


Figure B.1: ROC curve obtained from train, validation and test sets for a 15 million parameters model.

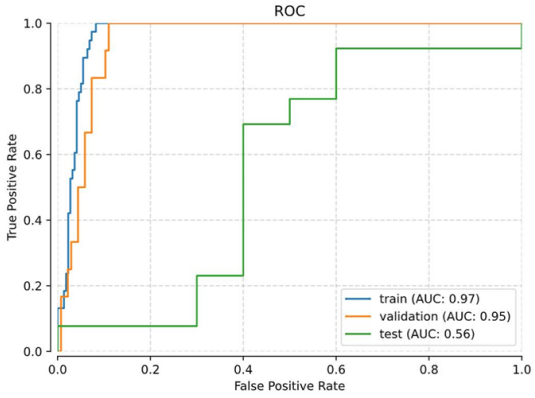
C Effect of Domain Adversarial Learning

In this experiment, we wanted to understand the effect of domain adverserial learning during the training of our model. Therefore, we started by training and evaluating our model from scratch . Then, we modified the model to add the modules required for domain adverserial learning. We trained this new model from scratch on the same data and with the same hyper parameters as the first model.

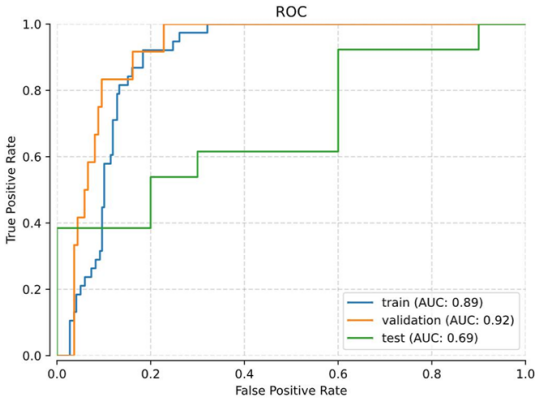
We used our dataset composed of 150x150x150 volumes containing both kidneys from AIFMD, KiTS and CT-ORG datasets. We selected randomly 11 control patients and 12 positive patients from AIFMD dataset to compose our test set. The rest of the samples were used to generate the training (256 samples) and validation set (148 samples). The models were trained over 500 epochs with a learning rate of 0.001 and ADAM optimizer. We used a batch-size of 7 with a gradient accumulation over 5 batches. In addition, we applied the three gradient accumulation techniques described in our methodology. Table C.1 and Figure C.1a show the performance, respectively the ROC curve of our model without domain adversarial learning. The ROC curve for the test set and the ROC AUC of 0.56 show poor generalization performance by comparison to the results for the train and validation sets, which both obtained ROC AUC higher than 0.95.

In comparison, Table C.2 and Figure C.1b show the performance, respectively the ROC curve obtained after the training with domain adversarial learning. By comparing the results for the different sets, the model also shows difficulty to generalize with significant gap between the test set and the two others. While the precision increase from 43% to 70%, the accuracy and the ROC AUC dropped from 80+% to 60%, respectively from 89+% to 69%.

by comparing the two models, the version trained with domain adverserial learning has worse performance on all metrics for the train and validations sets. However, on the test set, its ROC curve look show better performance and its ROC AUC increased from 56% to 69%. Those results tend to confirm that the assumption that the model is learning to discriminate the datasets was correct. Indeed, with domain adverserial learning, the model performed worse on the same data and the same training setup because we forced it to not learn the features allowing to discriminate the datasets. If the model was not looking at those features, the



(a)



(b)

Figure C.1: ROC curves. On the top, the model has been trained without domain adversarial learning. On the bottom, the model has been trained with domain adversarial learning.

domain adverserial learning would have a less significant impact.

subset	num_samples	precision	recall	f1	roc_auc	accuracy
train	256	0.763	0.763	0.763	0.967	0.930
validation	148	0.500	0.833	0.625	0.947	0.919
test	23	0.692	0.692	0.692	0.562	0.652

Table C.1: Performance of our model trained without domain adversarial learning

subset	num_samples	precision	recall	f1	roc_auc	accuracy
train	256	0.476	0.526	0.500	0.890	0.844
validation	148	0.435	0.833	0.571	0.919	0.899
test	23	0.700	0.538	0.609	0.692	0.609

Table C.2: Performance of our model trained with domain adversarial learning

Bibliography

[1] Marianne H Khoury, Sims Hershey, and Rebecca M LeLeiko. Sex and gender differences in fibromuscular dysplasia. *US Cardiol. Rev.*, 18:e08, July 2024.

[2] Jeffrey W Olin, James Froehlich, Xiaokui Gu, J Michael Bacharach, Kim Eagle, Bruce H Gray, Michael R Jaff, Esther S H Kim, Pam Mace, Alan H Matsumoto, Robert D McBane, Eva Kline-Rogers, Christopher J White, and Heather L Gornik. The united states registry for fibromuscular dysplasia: results in the first 447 patients. *Circulation*, 125(25):3182–3190, June 2012.

[3] Taylor Petropoulos, Anita Shah, Andrew Dueck, Christine Hawkes, Sheldon W. Tobe, William Kingston, and Mina Madan. Fibromuscular dysplasia: A focused review for the cardiologist. *CJC Open*, 6(11):1274–1288, 2024. ISSN 2589-790X. doi: https://doi.org/10.1016/j.cjco.2024.07.014. URL https://www.sciencedirect.com/science/article/pii/S2589790X24003202.

[4] Patricia Van der Niepen, Tom Robberechts, Hannes Devos, Frank van Tussenbroek, Andrzej Januszewicz, and Alexandre Persu. Fibromuscular dysplasia: its various phenotypes in everyday practice in 2021. *Polish Heart Journal (Kardiologia Polska)*, 79(7-8):733 – 744, 2021. ISSN 1897-4279. doi: {}. URL https://journals.viamedica.pl/polish_heart_journal/article/view/KP.a2021.0040.

[5] W Dennis Foley and Musturay Karcaaltincaba. Computed tomography angiography: principles and clinical applications. *Journal of computer assisted tomography*, 27:S23–S30, 2003.

[6] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. *Handwritten digit recognition with a back-propagation network*, page 396–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990. ISBN 1558601007.

[7] Safa Riyadh Waheed, Saadi Mohammed Saadi, Mohd Shafry Mohd Rahim, Norhaida Mohd Suaib, Fallah H Najjar, Myasar Mundher Adnan, and Ali Aqeel Salim. Melanoma skin cancer classification based on cnn deep learning algorithms. *Malaysian Journal of Fundamental and Applied Sciences*, 19(3):299–305, May 2023. ISSN 2289-5981. doi: 10.11113/mjfas.v19n3.2900. URL http://dx.doi.org/10.11113/mjfas.v19n3.2900.

[8] Mohammad Asif Hasan, Fariha Haque, Saifur Rahman Sabuj, Hasan Sarker, Md. Omaer Faruq Goni, Fahmida Rahman, and Md Mamunur Rashid. An end-to-end lightweight multi-scale cnn for the classification of lung and colon cancer with xai integration. *Technologies*, 12(4), 2024. ISSN 2227-7080. doi: 10.3390/technologies12040056. URL https://www.mdpi.com/2227-7080/12/4/56.

[9] A. M. El-Assy, Hanan M. Amer, H. M. Ibrahim, and M. A. Mohamed. A novel cnn architecture for accurate early detection and classification of alzheimer's disease using mri data. *Scientific Reports*, 14(1), February 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-53733-6. URL http://dx.doi.org/10.1038/s41598-024-53733-6.

[10] Safa Jraba, Mohamed Elleuch, Hela Ltifi, and Monji Kherallah. Alzheimer disease classification using deep cnn methods based on transfer learning and data augmentation. *International Journal of Computer Information Systems and Industrial Management Applications*, 16(3):17–17, 2024.

[11] Wei Wu, Jingyang Zhang, Hongzhi Xie, Yu Zhao, Shuyang Zhang, and Lixu Gu. Automatic detection of coronary artery stenosis by convolutional neural network with temporal constraint. *Computers in Biology and Medicine*, 118:103657, 2020. ISSN 0010-4825. doi: https://doi.org/10.1016/j.compbiomed.2020.103657. URL https://www.sciencedirect.com/science/article/pii/S0010482520300512.

[12] Jian-Hua Shu, Fu-Dong Nian, Ming-Hui Yu, and Xu Li. An improved mask r-cnn model for multiorgan segmentation. *Mathematical Problems in Engineering*, 2020:1–11, July 2020. ISSN 1563-5147. doi: 10.1155/2020/8351725. URL http://dx.doi.org/10.1155/2020/8351725.

[13] Pavlo Radiuk. Applying 3d u-net architecture to the task of multi-organ segmentation in computed tomography. *Applied Computer Systems*, 25(1):43–50, 2020.

[14] Abhishta Bhandari, Jarrad Koppen, and Marc Agzarian. Convolutional neural networks for brain tumour segmentation. *Insights into Imaging*, 11(1), June 2020. ISSN 1869-4101. doi: 10.1186/s13244-020-00869-4. URL http://dx.doi.org/10.1186/s13244-020-00869-4.

[15] Mohammed A. M. Abdullah, Sinan Alkassar, Bilal Jebur, and Jonathon Chambers. Lbts-net: A fast and accurate cnn model for brain tumour segmentation. *Healthcare Technology Letters*, 8(2):31–36, March 2021. ISSN 2053-3713. doi: 10.1049/htl2.12005. URL http://dx.doi.org/10.1049/htl2.12005.

[16] Stephen V. Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote Sensing of Environment*, 62(1):77–89, 1997. ISSN 0034-4257. doi: https://doi.org/10.1016/S0034-4257(97)00083-7. URL https://www.sciencedirect.com/science/article/pii/S0034425797000837.

[17] Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

ISSN 0306-4573. doi: https://doi.org/10.1016/j.ipm.2009.03.002. URL https://www.sciencedirect.com/science/article/pii/S0306457309000259.

[18] C. J. Van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, 2nd edition, 1979.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665.

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton. Imagenet classification with deep convolutional neural networks. *Neural Information Processing Systems*, 25, 01 2012. doi: 10.1145/3065386.

[21] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015. URL https://arxiv.org/abs/1409.1556.

[22] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions, 2014. URL https://arxiv.org/abs/1409.4842.

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015. URL https://arxiv.org/abs/1512.03385.

[24] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35 (1):221–231, 2013. doi: 10.1109/TPAMI.2012.59.

[25] Du Tran, Lubomir D. Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. C3D: generic features for video analysis. *CoRR*, abs/1412.0767, 2014. URL http://arxiv.org/abs/1412.0767.

[26] Rui Hou, Chen Chen, Rahul Sukthankar, and Mubarak Shah. An efficient 3d CNN for action/object segmentation in video. *CoRR*, abs/1907.08895, 2019. URL http://arxiv.org/abs/1907.08895.

[27] J. Arunnehru, G. Chamundeeswari, and S. Prasanna Bharathi. Human action recognition using 3d convolutional neural networks with 3d motion cuboids in surveillance videos. *Procedia Computer Science*, 133:471–477, 2018. ISSN 1877-0509. doi: https://doi.org/10.1016/j.procs.2018.07.059. URL https://www.sciencedirect.com/science/article/pii/S1877050918310044. International Conference on Robotics and Smart Manufacturing (RoSMa2018).

[28] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks, 2015. URL https://arxiv.org/abs/1412.0767.

[29] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset, 2018. URL https://arxiv.org/abs/1705.07750.

[30] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020. URL https://arxiv.org/abs/2002.05709.

[31] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2014. URL https://arxiv.org/abs/1312.6034.

[32] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, 128(2):336–359, October 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7. URL http://dx.doi.org/10.1007/s11263-019-01228-7.

[33] Vitali Petsiuk, Abir Das, and Kate Saenko. Rise: Randomized input sampling for explanation of black-box models, 2018. URL https://arxiv.org/abs/1806.07421.

[34] Wojciech Samek, Alexander Binder, Grégoire Montavon, Sebastian Bach, and Klaus-Robert Müller. Evaluating the visualization of what a deep neural network has learned. *CoRR*, abs/1509.06321, 2015. URL http://arxiv.org/abs/1509.06321.

[35] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. Evaluating feature importance estimates. *CoRR*, abs/1806.10758, 2018. URL http://arxiv.org/abs/1806.10758.

[36] Yao Rong, Tobias Leemann, Vadim Borisov, Gjergji Kasneci, and Enkelejda Kasneci. A consistent and efficient evaluation strategy for attribution methods, 2022. URL https://arxiv.org/abs/2202.00449.

[37] Wei Chen, Boqiang Liu, Suting Peng, Jiawei Sun, and Xu Qiao. S3d-unet: Separable 3d u-net for brain tumor segmentation. In Alessandro Crimi, Spyridon Bakas, Hugo Kuijf, Farahani Keyvan, Mauricio Reyes, and Theo van Walsum, editors, *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 358–368, Cham, 2019. Springer International Publishing. ISBN 978-3-030-11726-9.

[38] Konstantinos Kamnitsas, Christian Ledig, Virginia F. J. Newcombe, Joanna P. Simpson, Andrew D. Kane, David K. Menon, Daniel Rueckert, and Ben Glocker. Efficient multi-scale 3d CNN with fully connected CRF for accurate brain lesion segmentation. *CoRR*, abs/1603.05959, 2016. URL http://arxiv.org/abs/1603.05959.

[39] Qi Dou, Hao Chen, Lequan Yu, Lei Zhao, Jing Qin, Defeng Wang, Vincent CT Mok, Lin Shi, and Pheng-Ann Heng. Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks. *IEEE Transactions on Medical Imaging*, 35(5):1182–1195, 2016. doi: 10.1109/TMI.2016.2528129.

[40] Chengliang Yang, Anand Rangarajan, and Sanjay Ranka. Visual explanations from deep 3d convolutional neural networks for alzheimer's disease classification. *CoRR*, abs/1803.02544, 2018. URL http://arxiv.org/abs/1803.02544.

[41] K.R. Kruthika, Rajeswari, and H.D. Maheshappa. Cbir system using capsule networks and 3d cnn for alzheimer's disease diagnosis. *Informatics in Medicine Unlocked*, 14: 59–68, 2019. ISSN 2352-9148. doi: https://doi.org/10.1016/j.imu.2018.12.001. URL https://www.sciencedirect.com/science/article/pii/S235291481830176X.

[42] Lake Noel, Shelby Chun Fat, Jason L. Causey, Wei Dong, Jonathan Stubblefield, Kathryn Szymanski, Jui-Hsuan Chang, Paul Zhiping Wang, Jason H. Moore, Edward Ray, and Xiuzhen Huang. Sex classification of 3d skull images using deep neural networks. *Scientific Reports*, 14(1):13707, Jun 2024. ISSN 2045-2322. doi: 10.1038/s41598-024-61879-6. URL https://doi.org/10.1038/s41598-024-61879-6.

[43] Junghwan Lee, Cong Liu, Junyoung Kim, Zhehuan Chen, Yingcheng Sun, James R. Rogers, Wendy K. Chung, and Chunhua Weng. Deep learning for rare disease: A scoping review. *Journal of Biomedical Informatics*, 135:104227, 2022. ISSN 1532-0464. doi: https://doi.org/10.1016/j.jbi.2022.104227. URL https://www.sciencedirect.com/science/article/pii/S1532046422002325.

[44] Satyam Tiwari, Goutam Jain, Dasharathraj K. Shetty, Manu Sudhi, Jayaraj Mymbilly Balakrishnan, and Shreepathy Ranga Bhatta. A comprehensive review on the application of 3d convolutional neural networks in medical imaging. *Engineering Proceedings*, 59(1), 2023. ISSN 2673-4591. doi: 10.3390/engproc2023059003. URL https://www.mdpi.com/2673-4591/59/1/3.

[45] Satya P. Singh, Lipo Wang, Sukrit Gupta, Haveesh Goli, Parasuraman Padmanabhan, and Balázs Gulyás. 3d deep learning on medical images: A review, 2020. URL https://arxiv.org/abs/2004.00218.

[46] Jacob Gildenblat and contributors. Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam, 2021.

[47] Nicholas Heller, Niranjan Sathianathen, Arveen Kalapara, Edward Walczak, Keenan Moore, Heather Kaluzniak, Joel Rosenberg, Paul Blake, Zachary Rengel, Makinna Oestreich, Joshua Dean, Michael Tradewell, Aneri Shah, Resha Tejpaul, Zachary Edgerton, Matthew Peterson, Shaneabbas Raza, Subodh Regmi, Nikolaos Papanikolopoulos, and Christopher Weight. C4KC KiTS challenge kidney tumor segmentation dataset, 2019.

[48] Blaine Rister, Kaushik Shivakumar, Tomomi Nobashi, and Daniel L Rubin. CT-ORG: A dataset of CT volumes with multiple organ segmentations, 2019.

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran

Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[50] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks, 2016. URL https://arxiv.org/abs/1505.07818.

[51] Morris Jette, Andy Yoo, and Mark Grondona. Slurm: Simple linux utility for resource management. 07 2003. ISBN 978-3-540-20405-3. doi: 10.1007/10968987_3.

[52] Xin Wang, Ruisheng Su, Weiyi Xie, Wenjin Wang, Yi Xu, Ritse Mann, Jungong Han, and Tao Tan. 2.75d: Boosting learning by representing 3d medical imaging to 2d features for small data. *Biomedical Signal Processing and Control*, 84:104858, July 2023. ISSN 1746-8094. doi: 10.1016/j.bspc.2023.104858. URL http://dx.doi.org/10.1016/j.bspc.2023.104858.