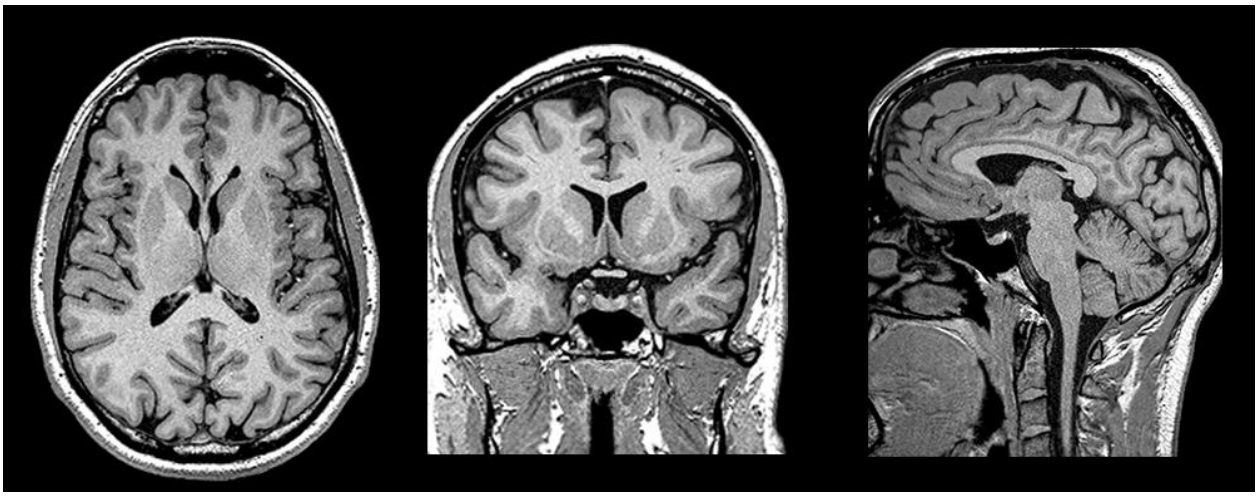
School of Engineering

Applied MSc in Data Science & Artificial Intelligence
Applied MSc in Data Engineering & Artificial Intelligence

A24 – Deep Learning

Project 1 – Real-World Computer Vision task

Intracranial Aneurysm Detection using deep-learning



Authors: *Marie-Caroline Bertheau*
Nathan Brunet
Claire Peyran
Anthony Reichen
Sourena Mohit Tabatabai

23 November 2025

Table des matières

Introduction

An aneurysm is a vascular pathology characterized by a localized dilation of the blood vessel wall, most commonly affecting arteries. It can appear in various shapes, ranging from a simple widening of an artery segment to a saccular outpouching connected to the parent vessel by a narrow neck. This dilation weakens the arterial wall, which may eventually fissure or rupture, leading to hemorrhage (Texakalidis et al., 2019).

Cerebral Aneurysms (CAs) are the most frequent type. The prevalence of unruptured CAs is estimated at around 3.2 % in the global population but is higher among females, older individuals, those with a strong family history, or people with predisposing behaviors (Din et al., 2023). Most CAs remain asymptomatic throughout the patient's life. However, up to 50% are only diagnosed after rupture, making them the most common cause of non-traumatic cerebral hemorrhage, responsible for approximately 85% of such cases. When rupture occurs, the resulting bleeding increases intracranial pressure and compresses adjacent brain structures, often leading to a poor prognosis: mortality can reach up to 44%, and among survivors, up to 20% remain functionally dependent (Din et al., 2023). Worldwide, intracranial aneurysms cause approximately 500,000 deaths each year, and roughly half of the victims are under 50 years old.

Early detection of CA is therefore essential to mitigate the risk of rupture and to guide timely and appropriate treatment decisions. In clinical practice, screening and diagnosis are primarily performed using two neuroimaging modalities, including Computed Tomography Angiography (CTA) and Magnetic Resonance Angiography (MRA) (Bizjak and Špiclin, 2023). Both techniques can employ contrast agents to enhance the visualization of blood vessels. During CTA, rapid X-ray-based images are acquired as the contrast dye flows through the vessels, allowing arteries and veins to be clearly delineated. In contrast, MRA is a specialized form of MRI focusing on visualizing blood vessels, which uses magnetic fields and radio waves. It offers a radiation-free alternative, though it may provide slightly lower spatial resolution and is more sensitive to patient movement. Currently, aneurysm detection is performed manually by radiologists who carefully analyze each image. This process is highly time-consuming and prone to human error. Even skilled radiologists achieve relatively low sensitivity when detecting small CA, ranging from 64 to 74% for CTA and from 70 to 92 % for MRA, and performance can even decrease further due to human factors such as fatigue, leading to an overall under-detection of aneurysms (Bizjak and Špiclin, 2023). To address these limitations, computer-assisted diagnosis approaches have been developed to support radiologists in this challenging task, for instance, 3D geometry modeling. Although such methods have demonstrated good performance, they often remain too complex or technically demanding to be routinely implemented in clinical practice (Dai et al., 2020).

In recent years, automated solutions using artificial intelligence and especially deep learning have shown remarkable results in the field of object detection and computer vision. Unlike traditional machine learning approaches, which rely on manually engineering features, deep learning models can automatically learn relevant features directly from raw data. This capacity makes them particularly well-suited for analyzing complex visual patterns characteristic of medical images (Burgos, 2023). Among deep-learning architectures, convolutional neural networks (CNNs) are the most widely used in computer vision tasks. CNNs process images by applying a series of layers of learnable filters that extract patterns in a hierarchical manner. Each layers typically capture low-level features such as edges, textures, and simple shapes, while deeper layers progressively learn more abstract and anatomically meaningful structures. This hierarchical feature learning enables CNNs to effectively distinguish normal anatomy from pathological patterns, such as aneurysms. Ultimately, the transformations performed across layers

culminate in a high-level representation from which the network can generate a prediction, making CNNs a powerful tool for medical image analysis.

The objective of this project is to develop a deep-learning-based method capable of automatically detecting CAs from neuroimaging data. The work primarily focuses on MRA, which is currently one of the most convenient and widely used techniques for non-invasive vascular assessment, offering high-quality, interpretable images without exposing patients to ionizing radiation. The proposed approach is a supervised medical-imaging computer vision task, using a binary classification framework. The model is designed to learn the spatial relationships present in volumetric data and extract discriminative features from textures, shapes, and local intensity variations. Based on these learned representations, the network aims to identify patches containing aneurysm signatures and ultimately infer their approximate coordinates within the brain volume.

Materials and Methods

1. Dataset description

The dataset used in this project consisted of expertly curated medical MRA examinations collected from multiple institutions, along with accompanying metadata. It was provided as part of a collaborative effort by the Radiological Society of North America, the American Society of Neuroradiology, the Society of Neurointerventional Surgery, and the European Society of Neuroradiology. The dataset is structured into one main folder and two metadata files, described below:

- *series/* **folder**: This directory contains the 1074 brain MRA series used in this project. Each patient examination, referred to as a series, is stored as a folder containing multiple images in DICOM format[1]. Each DICOM series represents a stack of 2D slices that together form a 3D volume of the brain. In addition to voxel (i.e., volumetric pixel) intensities, each DICOM file includes metadata (366 fields per file) describing patient information, scanner characteristics, and acquisition parameters.

- *train.csv* **dataframe**: This dataframe provides other metadata associated with each imaging series. Each row corresponds to one series and includes its unique identifier (SeriesInstancesUID) and basic demographic information such as patient age, sex, etc. Vascular locations were annotated with binary labels indicating the presence (1) or absence (0) of an aneurysm in each region. An additional column, AneurysmPresent, indicates whether at least one aneurysm is present anywhere in the entire scan, and serves as the primary **target variable** for the model training.

- *train_localizers.csv* **dataframe**: The file contains detailed localization information for annotated aneurysms in the training set. For each aneurysm, it specifies the series identifier (SeriesInstanceUID), the unique identifier for each 2D slice within a series (SOPInstanceUID), and the pixel coordinates (x, y) of the aneurysm center, along with a textual description of its anatomical location. This information enables accurate spatial mapping of aneurysms within the corresponding 3D brain volumes. Most aneurysms are located around the circle of Willis, a major circulatory blood structure at the base of the brain that supplies blood to the brain (Fig. 1).

[1] Digital Imaging and Communications in Medicine (DICOM) is the standardized format to store, transmit, and view medical images.

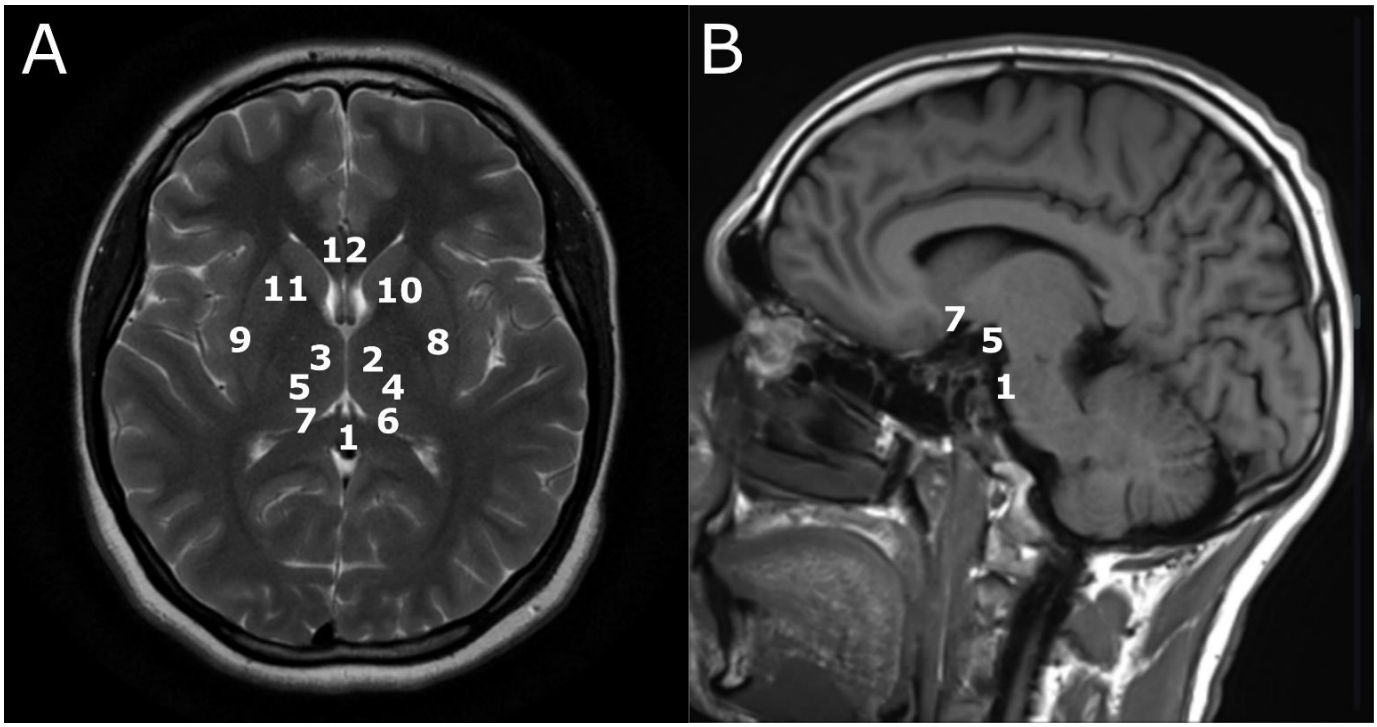


***Figure 1**: Approximate anatomical locations of the vascular sites investigated in this study, shown on MRI images. (A) Axial section acquired with a T2-weighted sequence. (B) Sagittal section acquired with a T1 post-contrast sequence. The numbered labels correspond to 1: Basilar Tip; 2: Right Posterior Communicating Artery; 3: Left Posterior Communicating Artery; 4: Right Infraclinoid Internal Carotid Artery; 5: Left Infraclinoid Internal Carotid Artery; 6: Right Supraclinoid Internal Carotid Artery; 7: Left Supraclinoid Internal Carotid Artery; 8: Right Middle Cerebral Artery; 9: Left Middle Cerebral Artery; 10: Right Anterior Cerebral Artery; 11: Left Anterior Cerebral Artery; and 12: Anterior Communicating Artery. Images adapted from Radiopedia.org.*

2. Preprocessing of the images

Due to the particularly large size of the dataset and the significant number of files, an offline preprocessing pipeline was implemented, which was executed once prior to the proper model training to transform raw medical imaging data into a standardized, model-ready format (Di Noto et al., 2023). The preprocessing pipeline consisted of six steps that must be carried out to obtain data ready to be ingested by the model.

Both to address the computational challenges of preprocessing and to ensure the need for reproducibility of the steps during the actual use of the model, a dedicated interface for processing DICOM series was developed. This interface was implemented in a dedicated class responsible for loading DICOM files, converting them into 3D volumes, and extracting and interpreting metadata. It also maps the aneurysm coordinates (from *train_localizer.csv* file) into voxel space so that it can be used later in the model to locate the aneurysm. The purpose of this abstraction was to encapsulate all low-level DICOM management operations to save time and resources, allowing us to focus on the development of the model. Without such a class, handling DICOM data would be prone to errors and inefficiencies, potentially resulting in inconsistencies during preprocessing and model input. Moreover, the interface was designed so that each series can be processed independently. This enabled parallelization of the preprocessing workflow and partially restarted it in case of an error, which is essential when dealing with datasets of such scale.

Conversion from DICOM to NIfTI format.

The first step of the preprocessing pipeline consisted of converting each series into a single 3D volume in NIfTI format. To achieve this, a differentiated treatment was applied depending on the metadata of each series in DICOM

format. Indeed, the structure may vary between a single multi-slice DICOM file (such as in Fig. 2A) and a series composed of as many DICOM files as there are slices. There can also be variations in the structure of DICOM files depending on the brand of the machine that performed the measurements (GE, Siemens, Philips, Hitachi) (Gorgolewski et al., 2016).

To ensure reliable and consistent handling of this heterogeneous format, the Python library `dicom2nifti` was used. This library supports a wide range of DICOM formats and automatically manages series reconstruction, slice sorting, etc., in order to convert them into NIfTI files. Once converted, each reformatted series was stored as a gzip-compressed NIfTI file (XXX.ni.gz).

Skull striping

To further refine the brain MRA dataset, the state-of-the-art literature recommends performing a skull-stripping step (Burgos, 2023), whose purpose is to remove non-brain tissues such as the skull, scalp, and background (Fig. 2B). For this step, the SynthStrip[2] model (Hoopes et al., 2022) was used. SynthStrip is a deep-learning-based skull-stripping tool that takes a NIfTI volume (compressed or not) as input and outputs two volumes in the same format: a clean volume containing only the brain where irrelevant areas were removed, and the corresponding binary mask indicating brain voxels (1) and background (0). Although relatively time- and resource-consuming, the SynthStrip model provides good performance in generating brain masks across a wide range of MRA contracts and acquisition conditions, making it perfectly suited for preprocessing in this study.

Bias Field Correction (N4)

MRI data are often affected by a low-frequency, smooth signal distortion caused by magnetic field heterogeneity (Burgos, 2023). This bias field could induce undesired variation in voxel intensities across the same tissue at different locations of the same image, which degrade the model performance as it may learn spurious features related to scanner artifacts rather than true anatomical or pathological variations. To mitigate this effect, a N4 bias field correction was applied to remove low-frequency intensity non-uniformities (Fig. 2C), using the SimpleITK library (Tustison and Gee, 2010).

Normalization

MRI intensities lack a standardized quantitative scale, meaning that voxel values can vary substantially across scanners, acquisition protocols, and patient anatomy. Therefore, it was necessary to perform a normalization step to ensure that all series have comparable intensity values. This normalization step aimed to place volume within a common intensity range while preserving meaningful anatomical contrast (Fig. 2D). Five different methods for normalization were implemented: z-score, robust z-score, min-max scaling, and percentile-based normalization. The min-max method was selected, as it preserves relative contrast and intensity ordering. The function takes a NIfTI volume as input and returns a normalized NIfTI volume.

Resample

MRI scans often exhibit different voxel spacing due to variation in acquisition parameters across scanners and protocols. To ensure spatial consistency throughout the dataset, all images were resampled to an isotropic 1x1x1 mm voxel grid using interpolation (Fig. 2E). Standardizing voxel dimensions is essential for downstream 3D convolutional networks, which rely on consistent spatial units to learn meaningful spatial features. This step also uses the `SimpleITK` library and implements three interpolation options: linear, nearest, and B-spline. Linear

[2] https://surfer.nmr.mgh.harvard.edu/docs/synthstrip/

interpolation was applied since it provides a good compromise between computational efficiency and interpolation accuracy.

Annotation coordinates

This step mapped 2D annotation points from the *train_localizers.csv* (DICOM slice coordinates) into voxel indices in the preprocessed NIfTI volumes. This transformation was essential to ensure that aneurysm annotations were expressed in the same voxel space used by the model for training patches extraction. The annotation conversion process consisted of three steps. First, it computed the physical 3D point in patient coordinates using DICOM metadata to retrieve the pixel spacing of the series. Then, it converted LPS (Left, Posterior, Superior) coordinates used in DICOM file format to RAS (Right, Anterior, Superior) coordinates used in NIFTI format. Finally, the resulting points were mapped from the computed voxel indices of the original NIFTI volume to the resampled one, producing coordinates fully aligned with the model input grid ($1mm^3$/voxel).

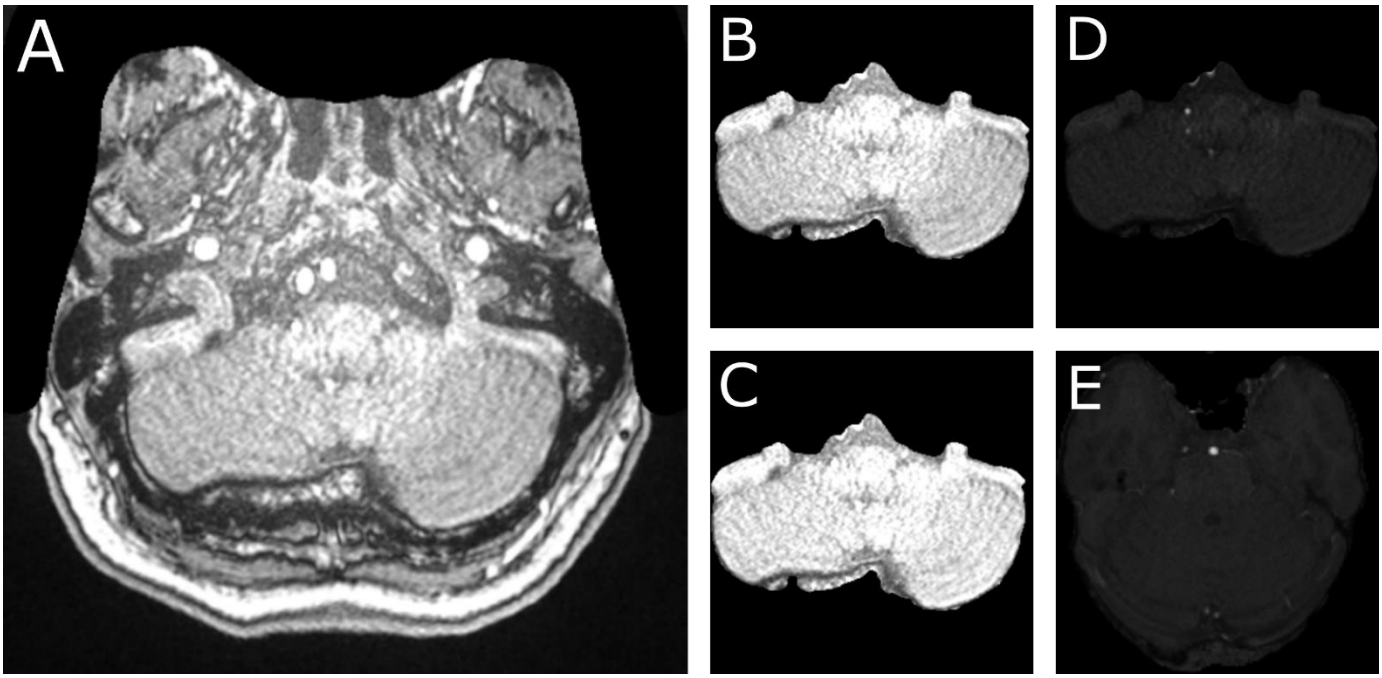


***Figure 2**: Impact of each processing step on an MRA slice. (A) Raw image. The irregular shape results from a black anonymization mask applied prior to image processing. (B) Image after skull stripping using the SynthStrip model. (C) Image after N4 bias-field correction. (D) Image after voxel-intensity normalization. (E) Image after resampling to isotropic resolution.*

At the end of the preprocessing pipeline, the data were exported into two structured outputs. The *resample* folder contained all preprocessed 3D brain volumes, each standardized to a clean, isotropic 1 x 1 x 1 mm voxels and homogenized in orientation, intensity, and spatial dimensions. The *mapping* folder stored the corresponding aneurysm coordinates expressed in the final voxel space, fully matching the resampled images. Together, these outputs formed a consistent and model-ready dataset that can be directly injected in the model pipeline.

3. Development of deep learning model

Overview

The project implemented a comprehensive, end-to-end deep learning pipeline for the automatic detection of CAs using the preprocessed 3D MRA volumes (described on Fig. 3). The entire workflow was developed using an object-oriented programming structure, which improves modularity, readability, and reproducibility. The pipeline

integrated all essential stages for a supervised binary classification model, including data loading and preparation, patch extraction, model training, evaluation, and whole-scan inference. A patch-based learning strategy was employed during training, where each full 3D scan was divided into smaller volumetric patches, which were then used to feed the model. This approach greatly reduced memory requirements by providing smaller inputs, increases the number of available training samples, and enabled explicit control over class balance by selecting an appropriate number of patches containing aneurysm versus healthy non-aneurysm patches. After training, the best-performing model, identified through patch-level evaluation, was applied to complete 3D scans using a sliding-window inference procedure. This step reconstructed patch-level predictions into a full-volume probability map, enabling the system to detect and spatially localize aneurysms across the entire brain volume. The evaluation, therefore, followed a two-step approach: a patch-level evaluation, used to select the most accurate and stable model during training, and a whole-scan evaluation, which assesses the model's ability to detect aneurysms under realistic clinical conditions. Each step of the pipeline is detailed in the following sections.

Implementation

Dataset and Annotation. The dataset was loaded and prepared to ensure the correct format for input into the neural network. This step was managed by the `DataManager` class, which links each brain scan to its corresponding aneurysm mapping file. It also assigned positive (presence of at least one aneurysm) or negative (no aneurysm) labels and stored the coordinates of aneurysms. Then, a simple threshold-based binary brain mask (0 = background; 1 = brain) was created and added for each volume (`VolumePreprocessor` class). This mask was essential for patch extraction, as it identified brain tissue and separated it from the background. The mask relied on voxel intensity: voxels with an intensity of zero were considered too dark to belong to the brain and were classified as background. Once the scans were efficiently loaded, the dataset was divided into a 70-20-10 split, meaning 751 scans for training, 215 for validation, and 108 reserved for scan-level testing.

Patch extraction strategy. For each volumetric scan of training and validation sets, fixed-size cubic patches were extracted to serve as input to the model (`PatchExtractor` and `AneurysmPatchDataset` classes). The patch size was configurable, as it can strongly influence both model performance and computational cost. In this project, patch sizes of 32 x 32 x 32 and 16 x 16 x 16 voxels were tested. **Negative samples** (patches without aneurysm) were randomly drawn uniformly from voxels located inside the brain mask, while excluding all annotated aneurysm regions. To ensure that negative patches contained meaningful anatomical information, only patches with at least 75% of brain tissue were retained. Patches falling below this threshold were discarded because they contained too much background or empty space to contribute a meaningful signal and would introduce bias in the training. For every annotated aneurysm location, a **positive sample** was generated by extracting a 3D crop centered at the aneurysm coordinates. If a scan reveals multiple aneurysms, several patches will be generated, one per aneurysm. Since CAs are relatively scarce and exhibit substantial variation in shape, size, and location, the dataset naturally becomes imbalanced in favor of negative samples. To mitigate this imbalance, an optional data augmentation strategy was implemented, which allowed for the generation of multiple positive patches per aneurysm by applying random spatial jitter to the patch center during extraction. This produces slightly shifted crops that still contain the aneurysm while enriching the positive class. The numbers of positive and negative patches extracted per scan were implemented as configurable parameters in the model pipeline, as well as the jittering range, allowing explicit control over class balance during training. Positive-to-negative ratios of 1:1 and 1:2 were specifically tested.

Finally, all extracted patches were converted into PyTorch tensors with a single-channel dimension (since MRA volumes are grayscale). Each dataset entry returned a pair consisting of the 3D patch and a binary label, where 1 indicates a positive (aneurysm-containing) patch and 0 indicates a negative patch.

Training procedure. The training procedure was implemented in the `AneurysmTrainer` class, which drove the complete learning workflow. During each iteration, the trainer loaded batches of patches, forwarded them through the model to obtain probability predictions, compared these predictions with the true labels using a selected loss function, and updated the model's weights using backpropagation and an optimizer. During training, the loss function measures how well the model's predictions match the true labels and guides weight updates by penalizing incorrect predictions. Two loss functions were tested in this project:

- *Binary Cross-Entropy (BCE) Loss*. It is standard loss for binary classification tasks (Terven et al., 2025) and was implemented using the `BCEWithLogitsLoss` function. BCE quantifies the difference between the true class labels (0 or 1) and the predicted logits[3]. Smaller BCE values indicate better predictive performance.

- *Focal Loss.* A custom `FocalLoss` class was implemented to handle class imbalance more efficiently. Focal Loss is an enhancement of standard BCE that introduces a scaling factor, reducing the contribution of easily classified samples and forcing the model to focus more on difficult or underrepresented ones (such as aneurysm-containing patches). It relies on two parameters: γ, which controls how strongly the loss focuses on hard-to-classify samples. It was set to 2, making easy-to-classify samples contribute less to the gradient. Then, α adjusts class weighting. It was set to 0.8 to give greater importance to the positive (aneurysm) class.

Then, the model's parameters (weights and biases) were optimized during backpropagation using the ADAM optimizer. ADAM is an adaptive gradient descent optimization algorithm that combines two key parameters: the momentum, which stabilizes updates and accelerates convergence, and the adaptive learning rate via RMSPop-like variance scaling, allowing each parameter to have its own learning rate. Because of its fast convergence and robustness, ADAM is widely used for training convolutional neural networks, especially in medical imaging tasks.

After each epoch, model performance was evaluated using several evaluation metrics (described below). An early-stopping criterion was implemented to halt the training process if validation performance stopped improving. This prevents overfitting, reduces unnecessary computation, and ensures that the final model corresponds to the optimal validation epoch. The model checkpoint achieving the best validation performance was saved and later used for full-scan inference.

Scan-Level inference. While patch-level evaluation provides a good assessment of the model's ability to identify aneurysms within individual patches, it does not directly reflect performance at the whole-scan level, which is an essential requirement for real-world clinical applicability. To enable full-volume analysis, a dedicated `PatchGenerator` class was developed to systematically extract 3D patches from entire brain scans using configurable patch size and stride parameters, thereby supporting both overlapping and non-overlapping patches. Patch extraction can optionally be restricted to brain-mask regions to avoid processing irrelevant background voxels. In case where a patch partially extended beyond the scan boundaries, zero-padding (i.e, filling missing voxel regions with zeros) was applied to preserve consistent patch dimensions, ensuring uniform coverage and stable inputs to the model.

Then, patch-wise predictions were produced in batches by the `ScanInferenceEngine` class, which applied the best-trained model to each patch and converted the resulting raw logits into calibrated probabilities via a sigmoid activation function. These localized probabilities were then reassembled into a coherent 3D probability volume by

[3]The logit is the raw output from the neural network, before applying the activation function that will transform it into a probability. BCEWithLogitsLoss implicitly handles a sigmoid function, which is appropriate in the case of binary classification, like here.

mapping each patch back to its corresponding coordinates. Overlapping regions were merged by taking the voxel-wise maximum probability to preserve the strongest local evidence of aneurysm presence. The resulting probability map provided a dense spatial representation of aneurysm likelihood across the entire brain volume.

Scan-level evaluation was performed through the `ScanLevelEvaluator`, which aggregated the voxel-level probability map into a single score representing the likelihood that the scan contains at least one aneurysm. A decision threshold was then applied to convert this score into a binary prediction. Although a default threshold of 0.5 is commonly used in binary classification, it is not always optimal in a context characterized by class imbalance. To address this, a data-driven threshold calibration procedure was employed, using the validation dataset, to identify the optimal threshold based on validation performance. The final binary prediction was subsequently compared to the true scan label to assess scan-level performance.

Evaluation

Patch-level evaluation was performed to assess the model's ability to distinguish between patches containing an aneurysm and those that do not. After each training epoch, model performance was evaluated on the validation dataset using several evaluation metrics. Then, a complementary scan-level evaluation was conducted on the test dataset to assess the model's capability to localize aneurysms within full 3D brain volumes, providing a clinically more relevant evalutaion of its performance. In both evaluation settings, the following metrics were used:

- *Accuracy*: the proportion of correctly classified samples among all patches. While accuracy provides a general indication of performance, it can be misleading in the case of class imbalance. For example, if 95% of the dataset consists of negative samples, a classifier predicting all samples as negative would reach 95% accuracy while completely failing to detect the minority positive class.

- *Sensitivity (or Recall)*: the true positive rate. High sensitivity indicates the model rarely misses an aneurysm, which is essential in a clinical context where false negatives carry significant risk.

- *Specificity*: the true negative rate. High specificity reflects the model's ability to avoid false alarms.

- *AUC-ROC* (Area Under the Receiver Operating Characteristic Curve): measures the model's ability to discriminate between positive and negative classes. Higher AUC values indicate better separability, while a value of 0.5 corresponds to random guessing.

- *F2-score*: a generalized F1-score that places greater emphasis on sensitivity. While the standard F1-score balances sensitivity and precision equally, the F2-score weights sensitivity more heavily, reflecting the fact that missing a positive case is more costly. This makes the F2-score particularly suitable for aneurysm detection, where failing to identify an aneurysm is more dangerous than incorrectly flagging a non-aneurysm patch.

The F2-score was used as the primary validation parameter for early stopping and for selecting the best-performing model, which was subsequently used for scan-level inference.

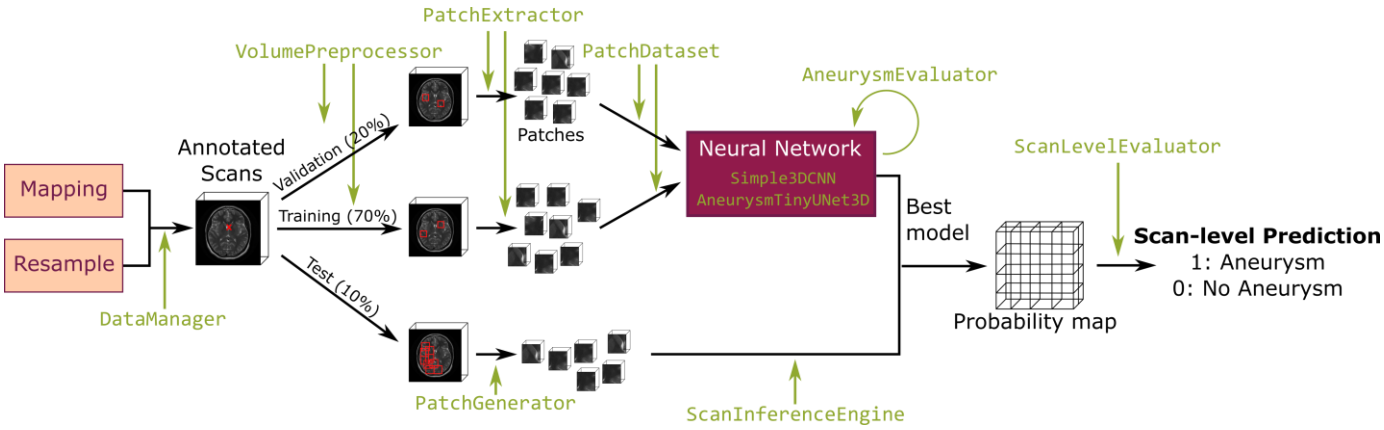


***Figure 3**: Schematic overview of the model pipeline implemented in this project following the preprocessing of raw MRA data. The custom classes responsible for each stage of the workflow are indicated in green.*

Network architecture.

Two architectures were tested: a classic simple 3D Convolutional Network (3D CNN) and a tiny 3D U-Net model inspired by Ham et al. (2023).

Simple 3D CNN (Fig. 4). A 3D Convolutional Neural Network (3D CNN) extends the principles of traditional 2D CNNs to volumetric data by introducing an additional depth dimension. Instead of processing an image of size height x width, a 3D CNN operates on height x width x depth inputs, enabling the network to learn spatial patterns and correlation across adjacent slices. This capability is particularly useful in medical imaging, where anatomical structures and pathologies such as aneurysms span multiple slices and exhibit complex 3D morphology. 3D CNN utilizes 3D convolutional kernels that slide across the volume to extract volumetric features.

In this project, an initial baseline architecture was implemented using a simple 3D CNN composed of three convolutional blocks followed by a fully connected linear layer. The input data consists of a single channel, as MRA images are grayscale. The first convolutional layer generates 16 feature maps, each representing a learned pattern or primitive structure in the 3D input. A ReLU activation function was applied to introduce non-linearity, followed by a 3D max-pooling operation that reduced the spatial dimensions by half. Pooling decreases computational cost and encourages the network to focus on increasingly abstract and semantically meaningful representations. Then, the second and third convolutional blocks followed the same pattern (3D convolution, ReLU activation, and pooling), gradually transforming the raw voxel intensities into higher-level volumetric features. Early layers typically learn simple structures such as edges and gradients, while deeper layers capture more complex anatomical shapes, such as tubular vessels or localized bulges indicative of aneurysms.

After the final pooling operation, the resulting feature volume was compressed into a compact representation by a global pooling, effectively producing a single value per feature channel. This vector was then flattened and passed through a fully connected linear layer, which outputs a single logit used for binary classification. A sigmoid function was subsequently applied during inference to convert the logit into a probability. ReLU was chosen as the activation function for its computational efficiency, simplicity, and strong empirical performance in convolutional architectures.

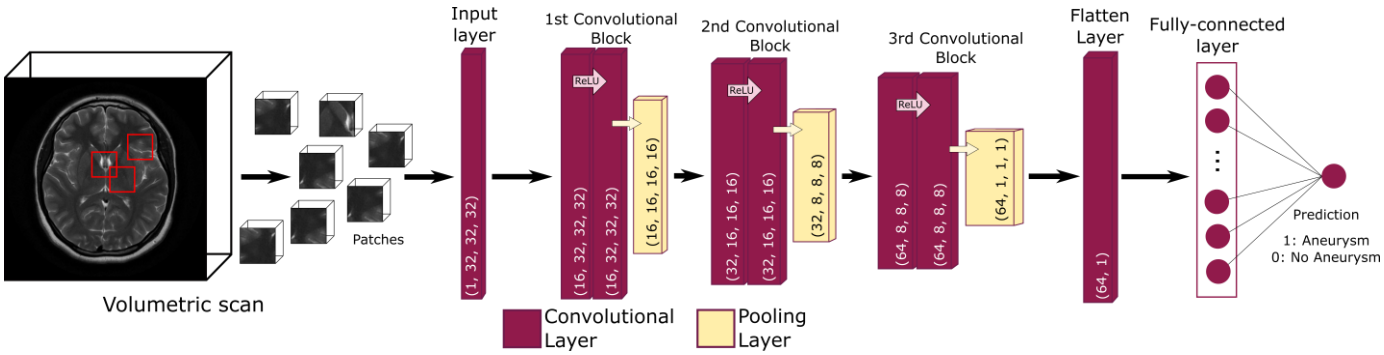


***Figure 4**: Schematic representation of the Simple 3D CNN architecture. The model here operates on input patches of size 32 x 32 x 32 voxels.*

Tiny 3D U-Net (Fig. 5). A 3D U-Net is a U-shaped encoder-decoder neural network. The first part, the encoder, works similarly to a 3D CNN: it progressively downsamples the input while extracting increasingly complex features. This is followed by the bottleneck, which is the deepest part of the network. At this stage, the spatial resolution is lowest and the number of channels is highest, allowing the network to learn the most global and abstract representations of the 3D structure. The decoder then upsamples the representation back to higher spatial resolution. 3D U-Nets also include skip connections that directly link encoder feature maps to their corresponding decoder layers. These skip connections allow the model to combine shallow features containing fine details (edges, textures) with deeper features carrying semantic meaning. This combination greatly improves performance. Classically, a 3D U-Net outputs a 3D segmentation mask, but in the present implementation, the network was modified for classification. Instead of producing a voxel-wise mask, the decoder output was fed into a global average pooling layer, which reduced the entire volume to a single feature vector, followed by a fully connected layer that produced one classification score. In practice, the encoder part in this tiny implementation contained two convolutional stages, each made of a two-layer convolutional block similar to one used in the 3D CNN. However, due to computational limitations, the U-Net network used fewer features (8 instead of 16). Max-pooling was applied after each encoder block. The encoder is then followed by the bottleneck block, composed only of convolutions and no pooling or upsampling. Finally, the decoder consisted of two upsampling stages that progressively reconstructed the spatial resolution of the input patch, using the skip connections from the encoder to recover fine details. The network ended with the classification head, made of a global pooling layer and a fully-connected layer producing a single output value for classification.

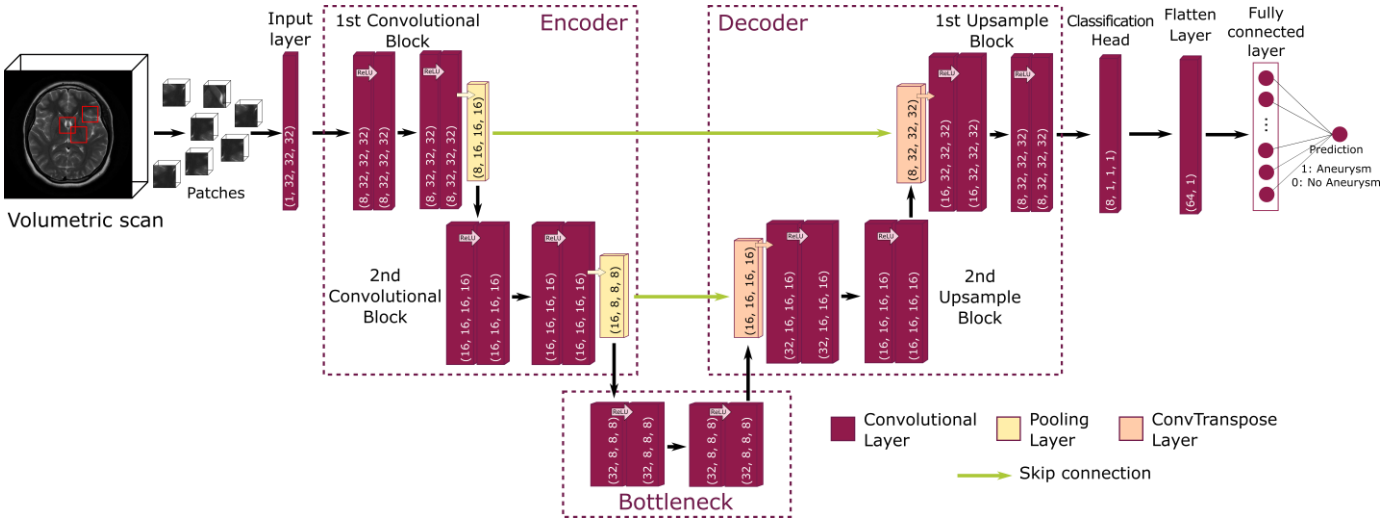


***Figure 5**: Schematic representation of the Tiny 3D U-Net architecture. The model here operates on input patches of size 32 x 32 x 32 voxels.*

Deployment

The project deployment repository was organized into three main modules, each corresponding to a key stage of the processing and modeling pipeline, as described below:

- *'preprocessing/'*: This module contains the full neuroimaging standardization pipeline used to transform raw DICOM series into model-ready NIfTI volumes. The workflow includes all the steps described in the preprocessing part:

1. DICOM to NIfTI conversion with geometry preservation,
2. Skull stripping using the SynthStrip model to isolate brain tissue
3. N4 bias field correction to reduce MRI intensity inhomogeneities,
4. Intensity normalization via z-score standardization
5. Isotropic resampling to 1mm³ voxel size.

In addition, this module converts 2D aneurysm annotations from DICOM slice coordinates into 3D voxel coordinates aligned with the final resampled NIfTI volumes. This involves LPS-to-RAS coordinate transformations, slice ordering verification, and propagation of coordinates to a standardized voxel grid.

- *'model/'*: This module implements the complete training, validation, and evaluation pipeline for patch-based 3D CNN. The two architectures described above are supported: the Simple 3D CNN '`AneurysmSimple3DCNN`', composed of a small encoder and global pooling, and the Tiny 3D U-Net encoder-decoder '`AneurysmTinyUNet3D`', adapted for binary classification. The module handles the extraction of 3D patches, mixed-precision training (AMP), configurable loss functions (including BCE and Focal Loss), and early stopping based on patch-level performance metrics (Accuracy, Sensitivity, F2-score, AUC). Scan-level inference is performed using a sliding-window approach combined with different probability-map aggregation strategies (max/mean/p95). Threshold calibration is applied using the Youden index, and additional localization-aware metrics (distance to ground truth) are computed to assess the spatial accuracy of predictions. All training configurations are automatically exported as JSON for reproducibility.

- *'app/'*: This module provides a Streamlit application for model deployment and result visualization. It includes the following interface:

- 'Inference': to run predictions on preprocessed volumes with configurable model/patch parameters,

- '2D Slice Viewer': to navigate axial/coronal/sagittal planes with predicted bounding boxes and ground truth overlays,

- '3D Volume Viewer': to generate interactive rendering with predicted aneurysm coordinates (cyan marker), ground truth annotations (green marker), focal region highlighting (red cube), and cropped 32^3 voxel detail view. The application also automatically displays the probability maps and metadata generated during inference, enabling an interactive interpretation of the model outputs.

Detailed deployment instructions are described in the *README.md* file at the root of the source code, including links to download the full raw dataset (200GB) as well as the preprocessed version. Interface visualizations are shown in Annex II.

Results

After exploring multiple architectures and parameter combinations, the Model 5, based on the Tiny 3D U-Net architecture trained with Focal Loss, emerged as the best-performing model. The three strongest models are summarized in Table 1, while additional details and experiments are reported in Annex I.

Model 5 was trained on patches of size 32x32x32 voxels. For each scan, four negative patches were extracted, and for each annotated aneurysm, three positive patches were generated using a jitter range of 4 voxels around the aneurysm center. This resulted in a training set of 1038 positive and 3004 negative patches, and a validation set of 300 positives and 860 negative patches, corresponding to an overall 1:3 positive-to-negative ratio. Although the training was scheduled for 100 epochs, the model began to plateau and stopped learning earlier. The best-performing checkpoint was reached at epoch 58, which was therefore selected for deployment.

***Table 1**: Top 3 of the best-performing models with the associated architecture and parameter configurations and evaluation metrics at patch and whole scan levels. The model with the best performance and selected for final implementation is displayed in bold.*

Name	Architecture	Ratio positive:negative	Nb Training patches	Validation sampling	Loss function	Parameters	Patch-level		Scan-level
							Best epoch	Scores	Scores
Model 5	**Tiny 3D U-Net**	**1:3**	**Positive = 1038 Negative = 3004**	**Positive = 300 Negative = 860**	**FocalLoss**	**Nb epoch = 100 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 4 Stride = 10**	**58**	**Accuracy = 0.91 Sensitivity = 0.94 Specificity = 0.89 AUC = 0.97 F2-score = 0.92**	**Accuracy = 0.62 Sensitivity = 0.68 Specificity = 0.58 AUC = 0.67**
Model 6	Simple 3D CNN	1:1	Positive = 692 Negative = 751	Positive = 200 Negative = 215	BCEWithLogitsLoss	Nb epoch = 100 Patch size = 32 Negative/patch = 1 Positive/patch = 2 Jitter range = 20 Stride = 16	15	Accuracy = 0.79 Sensitivity = 0.87 Specificity = 0.88 AU89C = 0.88 F2-score = 0.85	Accuracy = 0.65 Sensitivity = 0.36 Specificity = 0.82 AUC = 0.64
Model 7	Simple 3D CNN	1:1	Positive = 1730 Negative = 1502	Positive = 500 Negative = 430	BCEWithLogitsLoss	Nb epoch = 500 Patch size = 32 Negative/patch = 2 Positive/patch = 5 Jitter range = 20 Stride = 16	42	Accuracy = 0.82 Sensitivity = 0.89 Specificity = 0.74 AUC = 0.91 F2-score = 0.88	Accuracy = 0.67 Sensitivity = 0.39 Specificity = 0.83 AUC = 0.63

At the patch level, the model demonstrated strong predicted performance, with an accuracy of 0.91, indicating that the vast majority of samples are correctly classified. The AUC score of 0.97 further reflects excellent ability to discriminate between positive and negative patches. Sensitivity and specificity were both high, with 0.94 and 0.89, respectively, showing good performance on both positive and negative samples. The F2-score of 0.92 highlights the model's strong recall, which is crucial in a clinical context where missing aneurysms carries significantly more risk. Altogether, these results suggest that the model seems highly capable of correctly identifying local aneurysm patterns at the patch scale.

The learning curves for Model 5 (Fig. 6A) showed a smooth, steadily decreasing training loss, indicating stable optimization. While the validation loss curve was noisier, it followed a decreasing trend until approximately epoch 58, after which it began to rise slightly, signaling the onset of overfitting and justifying early stopping. Overall, the proximity of training and validation losses suggests that the model seemed to generalize well up to the optimal epoch.

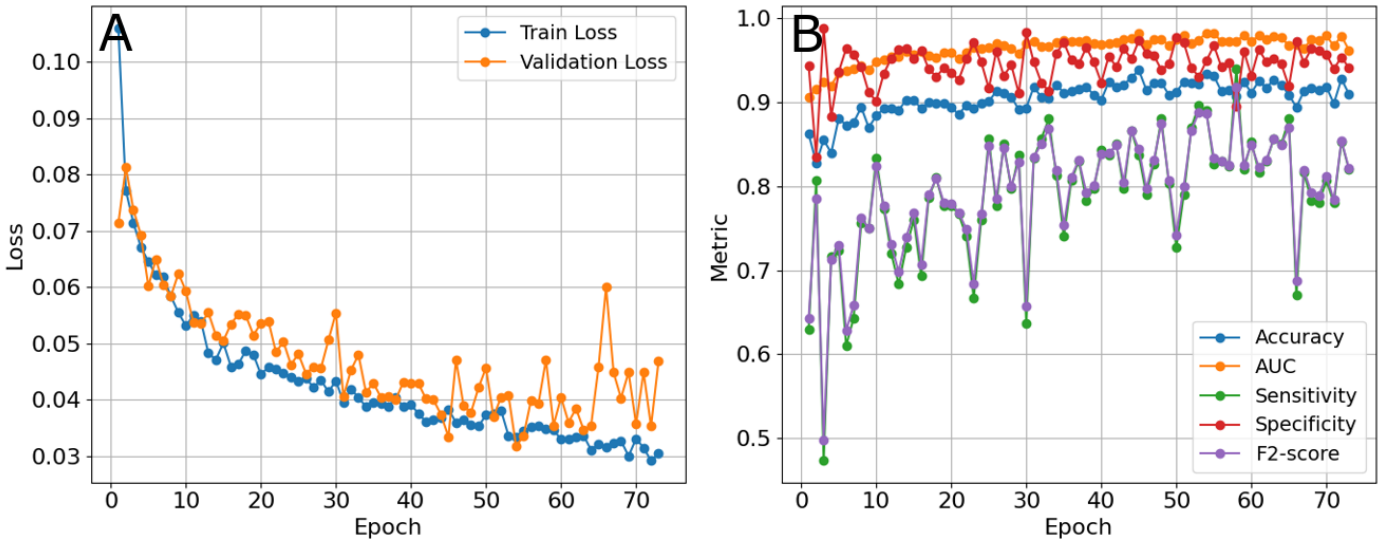


***Figure 6**: Evolution of Model 5 performance during training. (A) Training and validation loss curves and (B) patch-level evaluation metrics across epochs.*

In addition, patch-level validation metrics (Fig. 6B) exhibit a slight improvement in accuracy, specificity, and AUC over training. Sensitivity and F2-score were almost perfectly correlated, which is expected given their emphasis on positive samples, and exhibited higher variation with a zig-zag pattern. A slight decline after epoch 58 aligns with the rise in validation loss, confirming that this epoch is the best trade-off between learning and generalization, at the patch level.

However, when the model was evaluated at the scan level on the independent test set, performance decreased substantially, reaching an accuracy of 0.62 and an AUC score of 0.67 (Table 1). This drop suggests reduced discrimination ability at the whole-scan level and reflects a notable generalization gap. This pattern was consistent across all evaluated models.

Discussion

Although the best-performing model developed in this project (Model 5, based on a Tiny 3D U-Net, optimized with Focal Loss) achieved strong patch-level performance (AUC = 0.97, sensitivity = 0.94), this performance did not translate effectively to the scan-level evaluation, where the model reached only moderate performance (AUC = 0.67, sensitivity = 0.68). This discrepancy is noteworthy and highlights a significant generalization gap. From a clinical perspective, this gap is particularly concerning, as the final aim is to detect aneurysm at the full scan level.

These results were also somewhat disappointing given previously published studies that used conceptually similar patch-based methodologies (reviewed in Din et al. (2023) and Zhou et al. (2024)). For example, Ham et al. (2023) and Nakao et al. (2018) reported scan-level accuracies close to 90%, while Ueda et al. (2019) achieved up to 93% accuracy. However, one important difference lies in model complexity. Many of these studies used substantially deeper and more expressive architectures. Ham et al. (2023), for instance, employed a 23-layer 3D U-Net, whereas our model relied on a much smaller 4-layer Tiny 3D U-Net variant with far fewer parameters. This suggests that one primary avenue for improvement is to explore more powerful neural network architectures, potentially incorporating deeper encoders, or a convolution-transformer that better captures long-range 3D spatial context.

The project preprocessing implemented in the project followed standard state-of-the-art recommendations (Burgos, 2023; Zhou et al., 2024), including skull stripping, N4 bias correction, intensity normalization, and resampling to a uniform 1 mm^3 grid. While these steps ensure high-quality and standardized inputs for model training, one limitation is that the model was not provided with any form of anatomical prior or structural segmentation guideline. Several recent works demonstrate the benefit of “anatomically-informed” deep learning models. Indeed, Di Noto et al. (2023), for example, significantly improved their model sensitivity (from 65% to 83% of sensitivity) by incorporating anatomical contextual information into the model input. Integrating similar structural cues, such as vascular tree segmentations or a probability map of common aneurysm locations, may therefore substantially enhance performance.

Training duration was also investigated as a potential factor limiting performance. Multiple models were trained up to 500 epochs, but none showed meaningful improvement beyond approximately epoch 40. This plateau suggests that the models tend to learn quickly from the available data and that longer training leads primarily to overfitting, rather than better generalization. Implementing an adaptive learning rate scheduler, such as cosine annealing, could help stabilize optimization and potentially improve convergence toward better minima.

Then, a common hypothesis for limited model generalization is class imbalance, particularly in medical classification tasks where positive samples are scarce. However, our results suggest that imbalance was not a major factor in this project. Model 5, trained with a 1:3 positive-to-negative ratio, outperformed more balanced configurations. This indicates that the combination of targeted positive sampling (jitter patch extraction around aneurysm centers) and Focal Loss, which is explicitly designed to reduce the influence of easy negatives, likely mitigated the effects of imbalance. Therefore, imbalance alone does not explain the poor generalization at the scan level.

Another factor that likely contributed to the limited whole-scan performance is the lack of variability in aneurysm examples. Aneurysms exhibit substantial heterogeneity in shape, size, orientation, and anatomical location across patients. The training and validation datasets used in this project consisted of 751 and 215 scans, respectively, which may not sufficiently capture the full diversity of aneurysm presentations. This limited variability can lead the model

to overfit specific patterns seen during training while failing to generalize to unseen cases. Additional data augmentation, such as rotations, zooming, and flips, as used in Ueda et al. (2019), could help increase apparent variability and improve robustness. In addition, more systematic parameter optimization, for example, using GridSearch and cross-validation, could likely improve model performance and stability. However, due to the computational cost associated with 3D CNN training, such optimization was not feasible within the constraints of this project.

Finally, another factor that may have contributed to the modest scan-level performance is the calibration of the decision threshold and the way patch-level probabilities were aggregated into scan-level predictions. The model outputs a continuous probability for each extracted patch, but the final aneurysm detection at the whole level requires converting these local predictions into a global binary decision. In our pipeline, this was achieved through a simple threshold applied to the interpolated 3D probability map reconstructed from patch predictions. However, the threshold selected during validation may not have been optimal for whole-scan inference, especially in cases where multiple aneurysms may be present. Several issues can arise from this approach. First, patch-level probabilities are not necessarily well calibrated, meaning that a predicted probability of 0.9 does not always correspond to an actual 90% likelihood of aneurysm presence. Poor calibration can strongly degrade performance when aggregating predictions across large 3D volumes. Second, the method used to generate the probability map, typically interpolation followed by merging overlapping patch predictions, can introduce smoothing artefacts, produce blurred probability distributions, or dilute highly localized aneurysm signals. Such effects can cause small aneurysms to appear as diffuse regions in the probability map, making them more likely to fall below the detection threshold. Lastly, relying on a single global decision threshold ignores scan-level variability in noise, anatomical shape, or local intensity patterns, which may require a more adaptive decision rule, like per-scan calibration. All together, these limitations suggest that both probability-map reconstruction and threshold calibration may have contributed significantly to the observed generalization gap and represent important targets for future methodological improvements.

Conclusion

The objective of this project was to establish a deep-learning model capable of automatically detecting cerebral aneurysms from MRA images. After evaluating several architectural and training parameter configurations, a tiny encoder-decoder architecture, namely Tiny 3D U-Net, trained with Focal Loss, emerged as the most suitable model for this binary classification task. A comprehensive preprocessing pipeline was implemented to address the inherent challenges of 3D medical imaging, and a patch-based learning strategy was developed to enable the model to focus on localized anatomical patterns associated with aneurysms. While the model achieved strong performance at the patch-level, with high sensitivity and discriminative power, these results did not fully translate to the whole-scan evaluation, where performance stagnated around 60% accuracy. This gap highlights the need for further refinement of the scan-level inference strategy, such as improved probability aggregation, better threshold calibration, or more advanced pre-processing.

Future improvements could also include increasing the number and diversity of aneurysm cases in the training set, as well as exploring deeper or more expressive neural architectures to capture subtle vascular patterns. The project initially aimed to incorporate multiple neuroimaging modalities, such as CTA and various MRI sequences (T1-weighted and T2 post-contrast), but these were excluded due to the computational cost of preprocessing, especially the deep-learning-based skull-striping step. Extending the model to multi-modal imaging remains an important direction, as incorporating heterogeneous clinical data could substantially enhance robustness and generalizability in real-world diagnosis settings.

Overall, although the final model unfortunately did not achieve state-of-the-art performance at the scan level, the work conducted in this project lays the technical foundations for an automated aneurysm detection pipeline. Considering the increasing workload faced by medical staff, such deep-learning tools hold significant potential to support radiologists, accelerate diagnosis workflows, and ultimately contribute to improved patient care.

Bibliography

Bizjak, Ž., Špiclin, Ž., 2023. A Systematic Review of Deep-Learning Methods for Intracranial Aneurysm Detection in CT Angiography. Biomedicines 11, 2921. https://doi.org/10.3390/biomedicines11112921

Burgos, N., 2023. Neuroimaging in Machine Learning for Brain Disorders, in: Neuromethods. pp. 253–284. https://doi.org/10.1007/978-1-0716-3195-9_8

Dai, X., Huang, L., Qian, Y., Xia, S., Chong, W., Liu, J., Di Ieva, A., Hou, X., Ou, C., 2020. Deep learning for automated cerebral aneurysm detection on computed tomography images. Int. J. Comput. Assist. Radiol. Surg. 15, 715–723. https://doi.org/10.1007/s11548-020-02121-2

Di Noto, T., Marie, G., Tourbier, S., Alemán-Gómez, Y., Esteban, O., Saliou, G., Cuadra, M.B., Hagmann, P., Richiardi, J., 2023. Towards Automated Brain Aneurysm Detection in TOF-MRA: Open Data, Weak Labels, and Anatomical Knowledge. Neuroinformatics 21, 21–34. https://doi.org/10.1007/s12021-022-09597-0

Din, M., Agarwal, S., Grzeda, M., Wood, D.A., Modat, M., Booth, T.C., 2023. Detection of cerebral aneurysms using artificial intelligence: a systematic review and meta-analysis. J. Neurointerv. Surg. 15, 262–271. https://doi.org/10.1136/jnis-2022-019456

Gorgolewski, K.J., Auer, T., Calhoun, V.D., Craddock, R.C., Das, S., Duff, E.P., Flandin, G., Ghosh, S.S., Glatard, T., Halchenko, Y.O., Handwerker, D.A., Hanke, M., Keator, D., Li, X., Michael, Z., Maumet, C., Nichols, B.N., Nichols, T.E., Pellman, J., Poline, J.-B., Rokem, A., Schaefer, G., Sochat, V., Triplett, W., Turner, J.A., Varoquaux, G., Poldrack, R.A., 2016. The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments. Sci. Data 3, 160044. https://doi.org/10.1038/sdata.2016.44

Ham, S., Seo, J., Yun, J., Bae, Y.J., Kim, T., Sunwoo, L., Yoo, S., Jung, S.C., Kim, J.-W., Kim, N., 2023. Automated detection of intracranial aneurysms using skeleton-based 3D patches, semantic segmentation, and auxiliary classification for overcoming data imbalance in brain TOF-MRA. Sci. Rep. 13, 12018. https://doi.org/10.1038/s41598-023-38586-9

Hoopes, A., Mora, J.S., Dalca, A. V., Fischl, B., Hoffmann, M., 2022. SynthStrip: skull-stripping for any brain image. Neuroimage 260, 119474. https://doi.org/10.1016/j.neuroimage.2022.119474

Nakao, T., Hanaoka, S., Nomura, Y., Sato, I., Nemoto, M., Miki, S., Maeda, E., Yoshikawa, T., Hayashi, N., Abe, O., 2018. Deep neural network-based computer-assisted detection of cerebral aneurysms in MR angiography. J. Magn. Reson. Imaging 47, 948–953. https://doi.org/10.1002/jmri.25842

Terven, J., Cordova-Esparza, D.-M., Romero-González, J.-A., Ramírez-Pedraza, A., Chávez-Urbiola, E.A., 2025. A comprehensive survey of loss functions and metrics in deep learning. Artif. Intell. Rev. 58, 195. https://doi.org/10.1007/s10462-025-11198-7

Texakalidis, P., Sweid, A., Mouchtouris, N., Peterson, E.C., Sioka, C., Rangel-Castilla, L., Reavey-Cantwell, J., Jabbour, P., 2019. Aneurysm Formation, Growth, and Rupture: The Biology and Physics of Cerebral Aneurysms. World Neurosurg. 130, 277–284. https://doi.org/10.1016/j.wneu.2019.07.093

Tustison, N.J., Gee, J., 2010. N4ITK: Nick's N3 ITK Implementation For MRI Bias Field Correction. Insight J. https://doi.org/10.54294/jculxw

Ueda, D., Yamamoto, A., Nishimori, M., Shimono, T., Doishita, S., Shimazaki, A., Katayama, Y., Fukumoto, S., Choppin, A., Shimahara, Y., Miki, Y., 2019. Deep Learning for MR Angiography: Automated Detection of Cerebral Aneurysms. Radiology 290, 187–194. https://doi.org/10.1148/radiol.2018180901

Zhou, Z., Jin, Y., Ye, H., Zhang, X., Liu, J., Zhang, W., 2024. Classification, detection, and segmentation performance of image-based AI in intracranial aneurysm: a systematic review. BMC Med. Imaging 24, 164. https://doi.org/10.1186/s12880-024-01347-9

Annexes

Annex I: Details and results of model tested during this project

Name	Architecture	Ratio positive:negative	Nb Training patches	Validation sampling	Activation function	Loss function	Optimizer	Parameters	Patch-level		Scan-level	
									Best epoch	Score		Score
Model1	Simple 3D CNN	1:1	Positive = 1730 Negative = 1502	Positive = 500 Negative = 430	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 100 Patch size = 16 Negative/patch = 2 Positive/patch = 5 Jitter range = 10 Stride = 16	21	Accuracy = 0.76 Sensitivity = 0.93 Specificity = 0.57 AUC = 0.89 F2-score = 0.90	TP = 466 TN = 244 FP = 186 FN = 34	Accuracy = 0.63 Sensitivity = 0.41 Specificity = 0.76 AUC = 0.59
Model2	Tiny 3D U-Net	1:2	Positive = 346 Negative = 751	Positive = 100 Negative = 215	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 100 Patch size = 16 Negative/patch = 1 Positive/patch = 1 Jitter range = 8 Stride = 16	15	Accuracy = 0.82 Sensitivity = 0.71 Specificity = 0.87 AUC = 0.89 F2-score = 0.71	TP = 71 TN = 188 FP = 27 FN = 29	Accuracy = 0.57 Sensitivity = 0.22 Specificity = 0.79 AUC = 0.49
Model3	Tiny 3D U-Net	3:2	Positive = 1038 Negative = 751	Positive = 300 Negative = 215	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 100 Patch size = 32 Negative/patch = 1 Positive/patch = 3 Jitter range = 10 Stride = 16	1	Accuracy = 0.64 Sensitivity = 1 Specificity = 0.14 AUC = 0.81 F2-score = 0.94	TP = 300 TN = 31 FP = 184 FN = 0	Accuracy = 0.62 Sensitivity = 0 Specificity = 1 AUC = 0.38
Model4	Tiny 3D U-Net	1:3	Positive = 1038 Negative = 3004	Positive = 300 Negative = 860	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 100 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 4 Stride = 10	16	Accuracy = 0.89 Sensitivity = 0.79 Specificity = 0.93 AUC = 0.95 F2-score = 0.80	TP = 239 TN = 799 FP = 61 FN = 61	Accuracy = 0.39 Sensitivity = 1 Specificity = 0.01 AUC = 0.49
Model5	**Tiny 3D U-Net**	**1:3**	**Positive = 1038 Negative = 3004**	**Positive = 300 Negative = 860**	**ReLU**	**FocalLoss**	**Adam**	**Nb epoch = 100 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 4 Stride = 10**	**58**	**Accuracy = 0.91 Sensitivity = 0.94 Specificity = 0.89 AUC = 0.97 F2-score = 0.92**	**TP = 282 TN = 770 FP = 90 FN = 18**	**Accuracy = 0.62 Sensitivity = 0.68 Specificity = 0.58 AUC = 0.67**
Model6	**Simple 3D CNN**	**1:1**	**Positive = 692 Negative = 751**	**Positive = 200 Negative = 215**	**ReLU**	**BCEWithLogitsLoss**	**Adam**	**Nb epoch = 100 Patch size = 32 Negative/patch = 1 Positive/patch = 2 Jitter range = 20 Stride = 16**	**15**	**Accuracy = 0.79 Sensitivity = 0.87 Specificity = 0.88 AUC = 0.88 F2-score = 0.85**	**TP = 174 TN = 152 FP = 63 FN = 26**	**Accuracy = 0.65 Sensitivity = 0.36 Specificity = 0.82 AUC = 0.64**

Model7	**Simple 3D CNN**	**1:1**	**Positive = 1730 Negative = 1502**	**Positive = 500 Negative = 430**	**ReLU**	**BCEWithLogitsLoss**	**Adam**	**Nb epoch = 500 Patch size = 32 Negative/patch = 2 Positive/patch = 5 Jitter range = 20 Stride = 16**	**42**	**Accuracy = 0.82 Sensitivity = 0.89 Specificity = 0.74 AUC = 0.91 F2-score = 0.88**	**TP = 447 TN = 318 FP = 112 FN = 53**	**Accuracy = 0.67 Sensitivity = 0.39 Specificity = 0.83 AUC = 0.63**
Model8	Tiny 3D U-Net	1:3	Positive = 1038 Negative = 3004	Positive = 300 Negative = 860	ReLU	FocalLoss	Adam	Nb epoch = 100 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 4 Stride = 16	7	Accuracy = 0.86 Sensitivity = 0.88 Specificity = 0.85 AUC = 0.93 F2-score = 0.86	TP = 265 TN = 734 FP = 126 FN = 35	Accuracy = 0.54 Sensitivity = 0.56 Specificity = 0.52 AUC = 0.59
Model9	Tiny 3D U-Net with base_filters=16 instead of 8	1:3	Positive = 1038 Negative = 3004	Positive = 300 Negative = 860	ReLU	FocalLoss	Adam	Nb epoch = 100 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 10 Stride = 16	5	Accuracy = 0.80 Sensitivity = 0.92 Specificity = 0.76 AUC = 0.93 F2-score = 0.86	TP = 276 TN = 655 FP = 205 FN = 24	Accuracy = 0.54 Sensitivity = 0.49 Specificity = 0.56 AUC = 0.61
Model10	Simple 3D CNN	1:2	Positive = 3460 Negative = 7510	Positive = 1000 Negative = 2150	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 500 Patch size = 32 Negative/patch = 10 Positive/patch = 10 Jitter range = 20 Stride = 16	58	Accuracy = 0.83 Sensitivity = 0.81 Specificity = 0.84 AUC = 0.91 F2-score = 0.80	TP = 811 TN = 1815 FP = 335 FN = 189	Accuracy = 0.39 Sensitivity = 0.97 Specificity = 0.03 AUC = 0.57
Model11	Tiny 3D U-Net with base_filters=16 instead of 8	1:3	Positive = 1038 Negative = 3004	Positive = 300 Negative = 860	ReLU	FocalLoss	Adam	Nb epoch = 200 Patch size = 32 Negative/patch = 4 Positive/patch = 3 Jitter range = 5 Stride = 10		Accuracy = 0.92 Sensitivity = 0.89 Specificity = 0.93 AUC = 0.97 F2-score = 0.88	TP = 244 TN = 826 FP = 34 FN = 56	Accuracy = 0.54 Sensitivity = 0.65 Specificity = 0.46 AUC = 0.61
Model12	Tiny 3D U-Net	1:3	Positive = 1038 Negative = 3004	Positive = 300 Negative = 860	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 500 Patch size = 20 Negative/patch = 2 Positive/patch = 2 Jitter range = 20 Stride = 16	133	Accuracy = 0.79 Sensitivity = 0.79 Specificity = 0.79 AUC = 0.88 F2-score = 0.77	TP = 158 TN = 339 FP = 91 FN = 42	Accuracy = 0.59 Sensitivity = 0.41 Specificity = 0.70 AUC = 0.59
Model13	Tiny 3D U-Net	1:2	Positive = 3460 Negative = 7510	Positive = 1000 Negative = 2150	ReLU	BCEWithLogitsLoss	Adam	Nb epoch = 500 Patch size = 32 Negative/patch = 10 Positive/patch = 10 Jitter range = 20 Stride = 16	150	Accuracy = 0.85 Sensitivity = 0.90 Specificity = 0.83 AUC = 0.94 F2-score = 0.88	TP = 902 TN = 1789 FP = 361 FN = 98	Accuracy = 0.36 Sensitivity = 0.95 Specificity = 0.0 AUC = 0.53

Annex II: Streamlit app screenshots