Identification of Ischemic Stroke Origin Using Machine Learning Techniques

SSP - Computer and Communications Engineering

Faculty of Engineering, Alexandria University

Graduation Project 2023 - 2024

Abdelaziz Mohamed Khalil

Habiba Khaled ElMazahy

Logine Magdi Mohamed

Omar Salah AbdelKader

Yousef Yosry Mohamed

Aly Mohamed Aly

Under The Supervision Of:

Prof. Dr. Mohamed A. Ismail

Prof. Dr. Nagia M. Ghanem

Acknowledgment

Foremost, we would like to express our deepest gratitude to Allah for granting us the strength, wisdom, and perseverance to complete this project.

We wish to thank Prof. Dr. Mohamed Ismail and Prof. Dr. Nagia Ghanem for their invaluable supervision and guidance throughout this project. Their insightful comments and motivation have been instrumental in the completion of this work. We also extend our gratitude to Dr. Eman Sheta, whose expertise as a pathologist has been crucial in the analysis and interpretation of the data.

Lastly, we wish to express our sincerest appreciation to our families for their invaluable help, unwavering support, endless love, and constant encouragement throughout our efforts on this project.

Thank you.

Abstract

Strokes remain the second-leading cause of death globally, highlighting the need for effective stroke management. Ischemic strokes, being the most common type of stroke, encompass two main subtypes: thrombotic strokes, arising from the formation of a blood clot within a cerebral artery due to atherosclerosis, and embolic strokes, occurring when a clot or debris originating outside the brain travels to and obstructs a cerebral artery. The main issue concerning ischemic strokes is how quickly their subtype can be diagnosed, as time is of the essence when determining the best treatment for the patient. This report outlines a project that aims to classify the blood clot origins in ischemic strokes—using whole-slide digital pathology images provided by the Mayo Clinic, a nonprofit American academic medical center, as part of STRIP AI's Image Classification of Stroke Blood Clot Origin competition on Kaggle. The report focuses on two approaches: the first is leveraging multiple deep learning methods utilized in related work, such as different types of CNN models, to improve their results, and the second is applying feature extraction to aid with the classification process using classical machine learning algorithms, such as XGBoost. These proposed solutions will enable healthcare providers to better identify the origins of blood clots in deadly strokes, making it easier for physicians to prescribe the best post-stroke therapeutic management and reduce the likelihood of a second stroke.

Table of Contents:

List of Abbreviations

AI - Artificial Intelligence

AIS - Acute Ischemic Stroke

ANN - Artificial Neural Network

BDT - Binary Decision Tree

CE - Cardioembolic

CNN - Convolutional Neural Network

CSV - Comma Separated Values

CT - Computed Tomography

DL - Deep Learning

DNN - Deep Neural Network

FP - Fibrin Platelet

JPG - Joint Photographic Group

LAA - Large Artery Atherosclerosis

LIME - Local Interpretable Model-agnostic Explanations

MCLL - Multi-Class Logarithmic Loss

ML - Machine Learning

MRI - Magnetic Resonance Imaging

MSB - Martius Scarlet Blue

MT - Mechanical Thrombectomy

PNG - Portable Network Graphics

RBC - Red Blood Cell

ReLU - Rectified Linear Unit

ROC - Receiver Operating Characteristic

SOTA - State Of The Art

SVM - Support Vector Machine

TIFF - Tagged Image File Format

UE - Undetermined Etiology

WBC - White Blood Cell

WMCLL - Weighted Multi-Class Logarithmic Loss

WSI - Whole Slide Images

XAI - Explainable Artificial Intelligence

XGBoost - Extreme Gradient Boosting

YAD2K - Yet Another Darknet 2 Keras

YOLO - You Only Look Once

List of Figures/Tables

Chapter I: Introduction

1.1 Overview

Machine learning plays a transformative role in revolutionizing medical applications, offering innovative solutions to enhance diagnosis, treatment, and patient care. Its ability to analyze vast and complex datasets enables the identification of patterns, correlations, and insights that may be challenging for human professionals to discern. In medical imaging, Machine learning excels in tasks such as detecting anomalies in X-rays, MRIs, and CT scans, aiding radiologists in accurate and timely diagnosis.

Ischemic strokes are the most common type of stroke, accounting for about 85% of all stroke cases. An ischemic stroke is a type of stroke that occurs when there is a disruption of blood flow to a part of the brain, usually due to a blockage or narrowing of a blood vessel. The lack of blood flow to the affected brain area leads to a deprivation of oxygen and nutrients, causing brain cells to become damaged or die. The severity of the damage depends on factors such as the size of the blocked blood vessel and the speed at which medical treatment is provided.

Although image-based models have significantly facilitated rapid stroke diagnosis, accurate outcome predictions remain challenging. Our primary objective is the utilization of different machine learning techniques in categorizing the two main subtypes of ischemic strokes: CE and LAA. This endeavor aligns with the mission of the Mayo Clinic's Neurovascular Research Laboratory, which advocates for the development of AI-based stroke cause research and classification systems to support healthcare practitioners in making informed treatment decisions, ultimately striving to save the lives of stroke survivors.

1.2 Medical Background

1.2.1 Ischemic Strokes

A stroke is a sudden loss of brain function caused by a disruption of blood flow to the brain. Ischemic stroke is one of the two main types of strokes, hemorrhagic and ischemic, specifically occurring when there is a blockage or obstruction in the blood vessels supplying the brain, leading to a reduction or cessation of blood flow.[1] With a global prevalence of 25.7 million strokes per year, stroke is still one of the leading causes of death and sustained disability worldwide. The vast majority of strokes that occur are classified as ischemic.[2]

The composition of blood clots in AIS can vary depending on the cause of the stroke. Two types define an ischemic stroke, thrombotic (LAA) and embolic (CE). In thrombotic stroke, the blood clot (thrombus) forms in one of the arteries that supply blood to the brain. An embolic stroke happens when a blood clot forms away from the patient's brain usually in the patient's heart and travels through the patient's bloodstream to lodge in narrower brain arteries.[3]

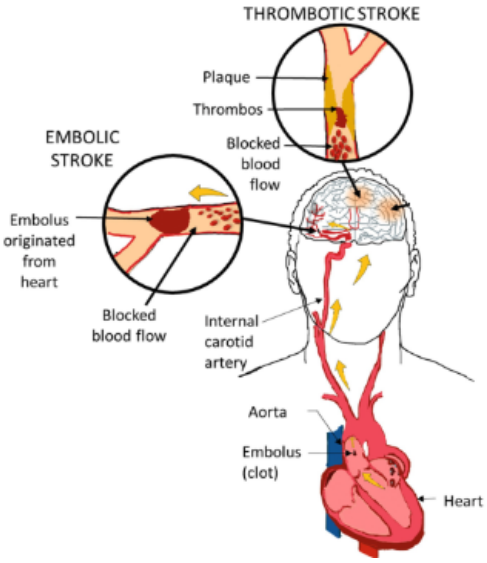


Figure 1.1: Types of ischemic stroke[4]

1.2.2 Thrombus Components

Normally, doctors perform an echocardiogram and a Doppler ultrasound for the patient to diagnose whether the stroke came originally from the large artery or if it is a cardioembolism. However, in the case of digital pathology images, pathologists depend on the components that form the thrombus and their respective percentages as the main differentiator between ischemic stroke subtypes.

A thrombus is a blood clot that forms within a blood vessel and remains attached at its site of origin. The components of a thrombus can vary depending on the circumstances of its formation and the specific conditions of the individual. Generally, a thrombus is composed of RBCs, platelets, fibrin, and WBCs.[1] Utilizing the thrombus components' measurements will come in handy as it is an important factor during classification.

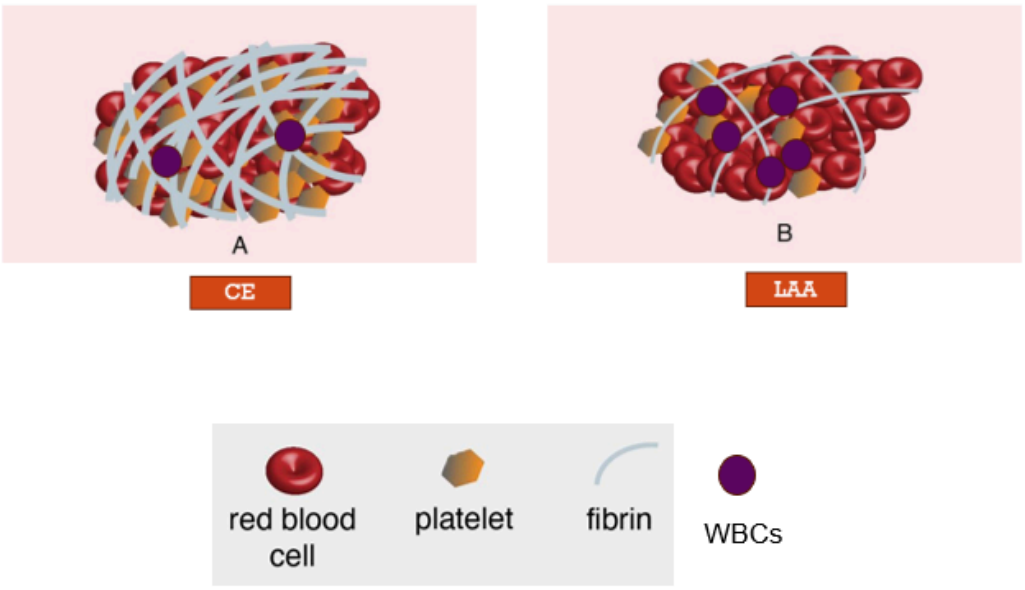


Figure 1.2: Thrombus components[5]

1.2.3 Acquiring Digital Pathology Images

Prior to creating the images, a blood clot sample is obtained from a patient, usually through a biopsy or surgical procedure. The sample may be collected from blood vessels, organs, or tissues where a clot has formed. The collected sample is immediately fixed to preserve its cellular structure. The sample is then cut into thin sections (typically 4-6 micrometers thick) using a microtome. These sections are then mounted onto glass slides. The sections are stained using various histological stains to highlight specific cellular components and structures. The pathologist or a technician captures images of the stained sections using a microscope equipped with a camera. These images can be stored digitally for further analysis, documentation, and communication.[6]

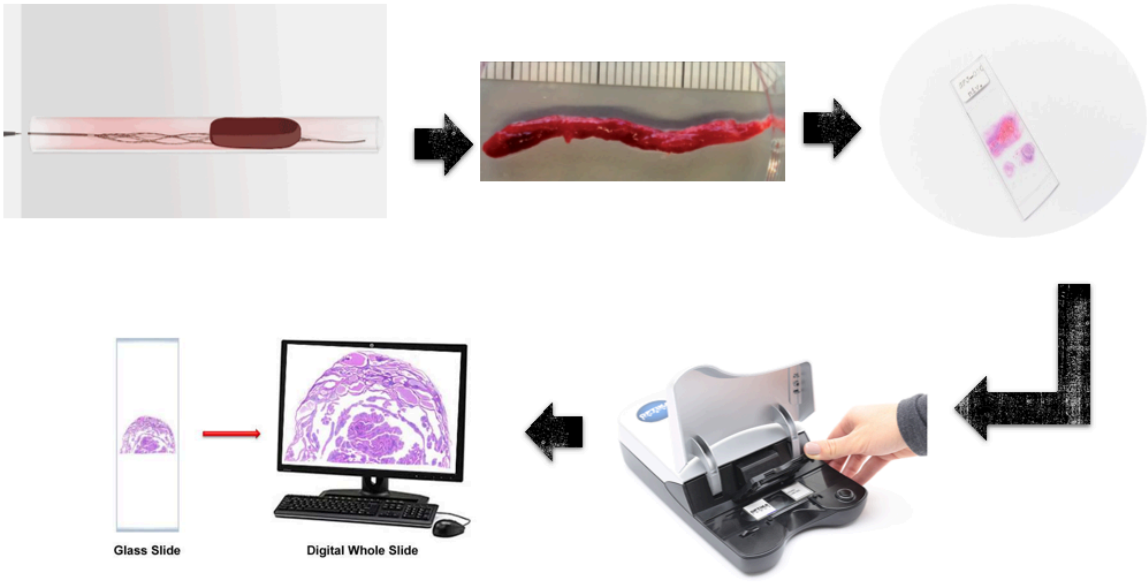


Figure 1.3: Obtaining thrombus digital pathology images[2]

With the help of histopathologic image expert Dr. Eman Shetta, it was identified that the MSB staining technique was conducted on our digital pathology image dataset. It was then easier to identify each of the components in the images based on their color after the staining.

MSB stain has been used for the characterization of clot composition a lot as it seems to be the optimal histology stain for the identification of the major components of AIS Clots. MSB staining allows for the identification of RBCs (yellow), F/Ps (reddish violet), and WBCs (purple). The major advantage of the MSB stain is that the distinctive color separation is much better than other types of staining techniques, therefore enabling more accurate quantification of clot components, as can be seen in Figure 1.4.[7]

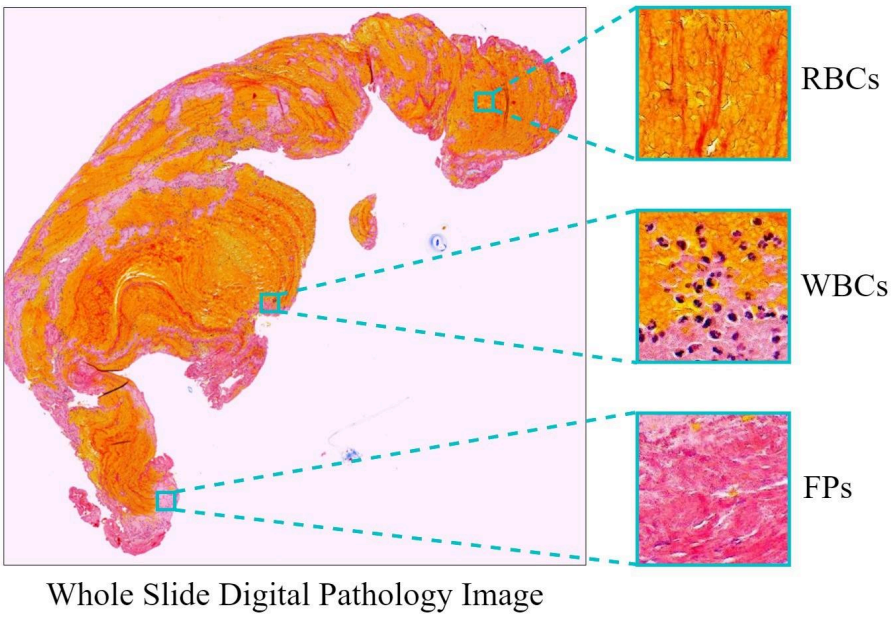


Figure 1.4: Zoomed-in thrombus image from dataset

1.3 Proposed Solutions

Based on our research, we realized that there is more than one way to tackle the problem. The first solution we proposed was to study the different preprocessing techniques, models, and results from related work comparatively, and by a series of trial and error, we tried to gradually increase the accuracy. The second solution was to use feature extraction on our data and utilize these features as parameters in classical ML. The team was divided into two groups to maximize our efforts and test both solutions.

In the first approach, we will extensively compare various models and architectures from different papers cited in Chapter 2. After careful consideration, we will select the models believed to be the best suited for our project. Each of these models comes with its own set of pros and cons. Through experimentation, we aim to evaluate their effectiveness and further improve their results.

For the second approach, we will start with image segmentation and employ medical information taken from section 1.2 combined with methods from the paper mentioned in section 2.4 as a basis for feature extraction. With the extracted features, we will easily be able to identify and select distinct characteristics from each ischemic stroke subtype. The selected characteristics will serve as an extra threshold for classification using classical ML.

We will also talk about some methods that we have read about but not actually applied or used, as we strive to ensure that we try every method that may possibly help us. At the end of this project, we will evaluate all approaches, state our results, and start planning how to move forward with them.

Chapter II: Literature Review

This chapter serves as a literature review, a comprehensive exploration of existing knowledge and papers relevant to our project. This review serves as the foundation for our conceptual framework, guiding our understanding of the project and paving the way for our own contributions and insights.

2.1 Image Classification of Stroke Blood Clot Origin Using Deep Convolutional Neural Networks and Visual Transformers

The paper[8] describes a particular approach to how to apply Artificial Intelligence to separate two major AIS etiology subtypes: cardiac (CE) and atheroma (LAA). Four DNN architectures and a simple ensemble method are used in the approach.

2.1.1 Dataset and Preprocessing

The training dataset includes 754 very high-resolution images in TIFF format and that is why it is associated with the following major disadvantages of using original images:

- Particular images are stored in files sized up to 2.5 GB, making it very hard to work with such files, especially if a lot of RAM is unavailable.
- A dataset with high-resolution images is not useful for classification tasks because even building a SOTA solution for a classification task requires images with resolution in the range from 224x224 pixels to 512x512 pixels.
- The original images can have vast ‘empty’ spaces that don't carry any useful information.
- Small training samples in DL tend to have a low ability for generalization and an increased risk of overfitting.

To avoid some of the disadvantages, data preparation included the following steps:

1. Pruning and rotation
2. Resizing
3. Augmentation

2.1.2 Models

The classification task of medical images with weak signals seems very challenging, notably due to the very small “train” dataset. Nevertheless, using a combination of several DL architectures helps to avoid their individual disadvantages and to extract signals to separate classes as right as possible. These are the DL architectures used to build the final solution:

- EfficientNet B4 with Noisy Student weights for resolution 384x384.
- EfficientNet B4 with Noisy Student weights for resolution 512x512.
- EfficientNet B5 with Noisy Student weights for resolution 512x512.
- Swin Large with window parameter 7 for resolution 224x224.
- Swin Large with window parameter 12 for resolution 384x384.

The following successive changes in each model architecture were made:

1. Linear layer with 128 outputs is added.
2. Dropout 0.1 is added.
3. Linear layer with 64 outputs is added.
4. Linear layer with 2 outputs is added.
5. Softmax function used to transform outputs of each model into probabilities.

2.1.3 Results

The WMCLL function was used as an evaluating metric:

$$WMCLL = -\frac{\sum_{i=1}^{M} w_i \times \sum_{j=1}^{N_i} \frac{y_{ij}}{N_i} \times \ln p_{ij}}{\sum_{i=1}^{M} w_i}$$

Where N is the number of images in the class set, M is the number of classes, W_i is the weight for class i (equality of weights has been given), ln is the natural logarithm, y_i is 1 if observation j belongs to class i and 0 otherwise, P_{ij} is the predicted probability that image j belongs to class i.

Since the evaluating metric is a loss it is used as a custom loss function in neural network training to ensure better convergence in the optimization process compared to the standard loss function applied for classification tasks. Therefore, training with minimizing loss function at the same time directly minimizes evaluation metric.

The probability of classes CE and LAA were calculated as the mean of probabilities of the five models. Despite simplicity, the ensemble approach among other things allows for increased prediction. Finally, WMCLL = 0.69682 on the public leaderboard and WMCLL = 0.67188 on the private leaderboard.

2.2 Identification of Ischemic Stroke Origin Using Machine Learning Techniques

The paper's[9] primary objective involves the extraction of images from the Mayo Clinic dataset and categorizing them into two main classes: CE and LAA. This approach employed CNNs and CatBoost in predicting stroke causes and enhancing treatment strategies.

2.2.1 Dataset and Preprocessing

The classification process initiates with extracting images from the dataset and retrieving corresponding labels from the dataset's CSV file. Subsequently, the image labels are renamed based on their respective classes (CE or LAA), leading to the division of the dataset into training and testing sets.

The TIF images, with diverse sizes and dimensions, are initially downsized to a standardized 200×200 pixel in JPG format. This downsampling not only simplifies computational complexity but also retains essential image information. The processed images are then tagged and undergo further processing.

To prevent overfitting issues, data augmentation methods are employed, involving diverse modifications to the training dataset, including horizontal and vertical flips and random cropping. These alterations are applied without changing the class labels, ultimately enhancing the overall performance of the model.

2.2.2 Models

CNN is a powerful neural model commonly used for object classification and identification. It excels at extracting complex features for categorization. In CNNs, neurons can learn weights and biases, and they process input data by applying these weights and biases through an activation function. CNNs use consecutive convolution layers and ReLU functions to extract important features with specific dimensions, followed by max-pooling to reduce feature map size. Fully connected layers connect every neuron to each other, and training involves gradient descent and backpropagation. The Softmax function is used for output probability normalization. Figure 2.1 depicts the proposed architecture.

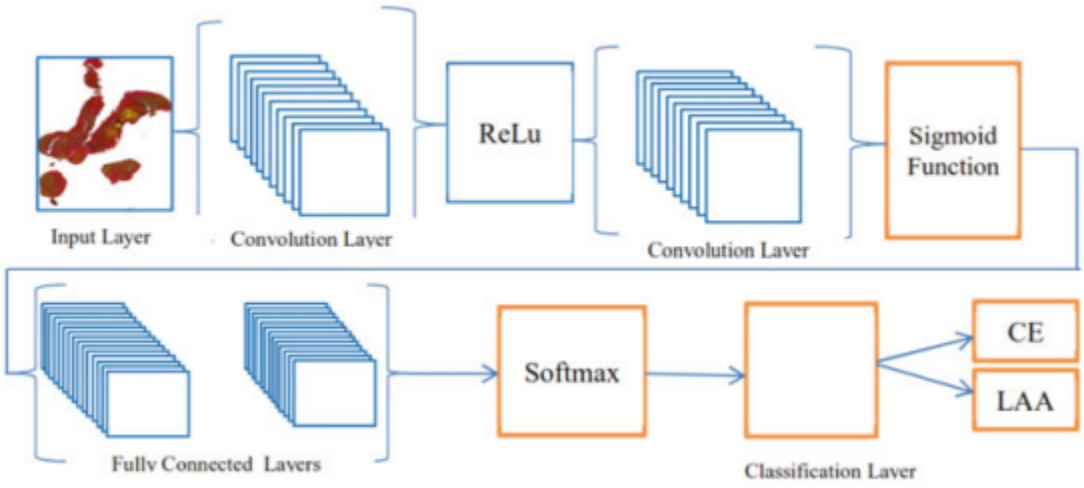


Figure 2.1: Proposed CNN architecture

The task of the input layer is similar to that in other types of ANN; it is used to hold the pixel value of the image input dataset. The convolution layer will compute the dot product of the individual weights and the region corresponding to the input volume. ReLU attempts to apply an element-wise activation function, and a sigmoid is used in the activation output from the previous layer. The pooling layer is used to sample down the spatial dimensions of the supplied input to lower the number of parameters inside that activation. The dense layers will function similarly to typical ANNs, trying to calculate the final scores for classification from the activation. The ReLU layer could be used between these levels to try to improve the performance.

Name	Type	Activations	Learnable parameters
Input layer	Input image	200 × 200 × 3	–
Convolution layer-1	Convolution	200 × 200 × 32	Weight 3 × 3 × 3 × 32 Bias 1 × 1 × 32
ReLU	ReLU	200 × 200 × 32	–
Convolution layer-2	Convolution	200 × 200 × 28	Weight 3 × 3 × 28 Bias 1 × 1 × 28
Sigmoid	Sigmoid	200 × 200 × 28	–
Dense layer-1	Dense	1 × 1 × 200	Weight 200 × 1,120,000 Bias 200 × 1
Dense layer-2	Dense	1 × 1 × 2	Weight 2 × 200 Bias 2 × 1
Softmax	Softmax	1 × 1 x 2	–
Classification layer	Classification output	0	0

Table 2.2: Parameter setting of the CNN model

The CatBoost classifier is another ML technique that is effective at predicting categorical features. It is a gradient-boosting method that uses BDTs as base predictors. Gradient boosting is an effective ML strategy for dealing with complex problems that involve heterogeneous features, inaccurate data, and complex dependencies.

For the experiment purpose, the hyper-parameters of the CatBoost model are indexed in Table 2. The depth of the decision tree is set to be 6, the value of step size is 0.01 describing the best learning rate and the model executed up to 100 iterations provides the better results.

Parameter name	Value
Depth	6
Learning rate	0.01
No. of iteration	100
Loss function	Binary class

Table 2.3: Hyper-parameters of CatBoost model

2.2.3 Results

The proposed study builds two classifiers: the CNN classifier and the CatBoost classifier. The dataset contains 754 images, and both classifiers were trained utilizing 80% (567) of the dataset and tested with the remaining 20% (207). It is ensured that both experiments follow the same division of the dataset for the results analysis. For the performance analysis, the authors used the confusion matrix and ROC to measure the accuracy, precision, and recall values of the proposed models.

The results from the experiment, as presented in Table 2.4, indicate that the CNN model exhibits superior accuracy compared to CatBoost in classifying blood clots using TIFF images. The CNN model demonstrates the highest values for recall, precision, and accuracy. Across all three parameters, CNN significantly outperforms the CatBoost classifier, establishing itself as the preferred choice for precise and reliable classification. The improved performance is evident in measures such as ROC, precision, accuracy, and recall values. Future studies could focus on enhancing CNN and CatBoost classifiers by selecting appropriate features and optimization techniques. Moreover, with a sufficient dataset, blood clot classification remains viable, and the exploration of hybrid classifiers could further enhance the accuracy of the analysis.

Model	Accuracy	Recall value	precision
CNN	0.9741246	0.75	0.85
Cat boost	0.731576	0.77	0.87
ANN	0.94325	0.87	0.79
SVM	0.863452	0.74	0.81

Table 2.4: Performance of CNN and CatBoost classifier

2.3 Advancing Ischemic Stroke Diagnosis: A Novel Two-Stage Approach for Blood Clot Origin Identification

The study[10] suggests a novel methodology for classifying the source of a blood clot into CE or LAA through the integration of data from whole-slide digital pathology images, which are utilized to fine-tune several cutting-edge computer vision models.

2.3.1 Dataset and Preprocessing

First, whole-slide digital pathology images of very large dimensions were obtained from the STRIP AI background clot dataset. These images were in the TIFF format, which is a standard format for digital pathology images. To extract patches from these images, the authors used the OpenSlide Python library. OpenSlide is an open-source library that provides a simple interface for reading digital pathology images. We extracted patches of size 600x600x3 dimensions from the whole-slide digital pathology images and saved them in PNG format.

Then Otsu's thresholding technique was used to filter out low-quality patches. The thresholding technique involves calculating the area of the slide contents in the image as well as the total content in the image. If the slide area consists of more than 30% of the total slide area, the image is kept, otherwise, it is discarded. This step helped us to remove images with low-quality content that could have affected our analysis negatively. Subsequently, the authors also used a previously trained background classifier to remove white images from the dataset. These steps ensured that the dataset we used for analysis was of high quality and relevance to the study.

Finally, the Albumentations library numerous powerful image augmentation techniques like HorizontalFlip, VerticalFlip, RandomAdjustSharpness, Rotate, and ColorJitter transformations were applied to balance the dataset. Also, the Resize transformation which resizes the 600x600x3 patches into 256x256x3 patches, and the Normalize transformation which uses the ImageNet mean and standard deviation to normalize the patches are applied on all splits of the dataset.

2.3.2 Models

Background Classifier: At first, the paper trained a background classifier model on the STRIP AI background clot dataset which contains 9999 images each for both classes. Then they used the transfer learning approach to fine-tune SOTA computer vision models for the task at hand. They wanted to create a lightweight and efficient background classifier, hence using the MobileNetV3 architecture, and replaced its classification head with a linear layer that produces the dataset-specific output. This background classifier was trained on images of dimension 128x128x3 to further reduce the inference time. Ultimately, the AdamW optimizer was used with an initial learning rate of 3e-4.

Blood Clot Classifier: The classification began with a transfer learning process that involved fine-tuning pre-trained models, using the Timm Python library, on a new dataset. In particular, several SOFA models including the swinv2 tiny window16 256 variant of the SwinTrans formerV2 architecture, poolformers36 variant of the PoolFormer architecture, convnext small variant of the ConvNeXt architecture, efficientnet b3 variant of the EfficientNet architecture, resnet50 variant of the ResNet architecture, and the inceptionv3 architecture, all of which have achieved impressive results in various computer vision tasks.

During training, the authors used two schedulers: EarlyStopping and StochasticWeightAveraging. The former computes the average of multiple weight values during training to achieve better generalization, while the latter stops training if the monitored metric does not improve. These schedulers help prevent overfitting and improve the models' generalization performance.

Split	Number patches with cell contents
Train	60489
Validation	12962
Test	12963
Total	**86414**

Figure 2.5: Blood clot dataset split

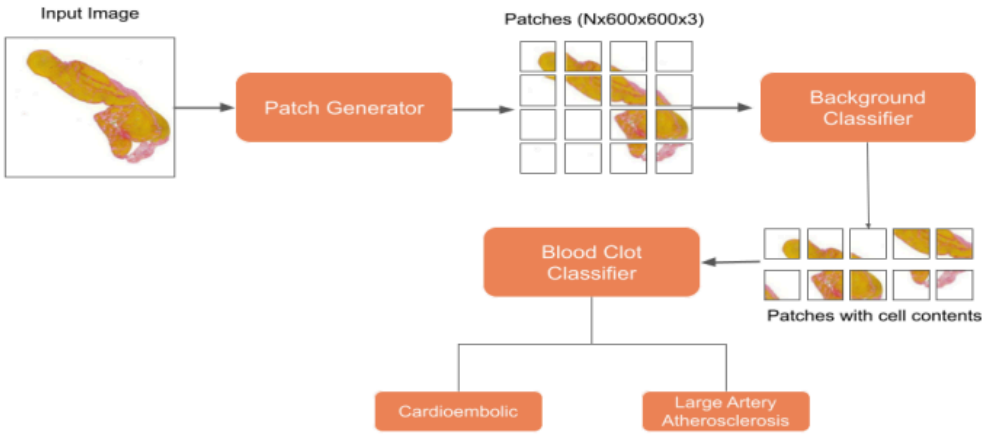


Figure 2.6: The overall architecture of the proposed system

2.3.3 Results

According to the results in Table 2.7, the SwinTransformerV2 model's swinv2 tiny window16 256 variant achieved the highest test accuracy, test precision, test recall, and test F1 score. The PoolFormerS36 model had the second-highest scores. The ResNet50 and InceptionV3 models had the lowest test F1 scores. The performance of the EfficientNetB3 model falls slightly short of the PoolFormerS36 model's performance. In the end, the SwinTransformerV2 model's swinv2 tiny window16 256 variant proved to be the top-performing model in classifying the source of blood clots in ischemic stroke patients and thus was chosen to develop the proposed system.

Model	Accuracy	Precision	Recall	F1-Score
ResNet50	0.592	0.6656	0.3839	0.4768
InceptionV3	0.5637	0.5822	0.4995	0.5294
EfficientNetB3	0.8773	0.867	0.8949	0.8775
ConvNeXt	0.7174	0.7213	0.7174	0.7193
PoolFormerS36	0.8871	0.8919	0.8871	0.887

Figure 2.7: Performance analysis

2.4 Image Classification of Ischemic Stroke Blood Clot Origin using Stacked EfficientNet-B0, VGG19, and ResNet-152

The study[11] addressed the critical issue of ischemic strokes aiming to classify the origin of blood clots, specifically between CE and LAA, using whole slide digital pathology images. The proposed approach involves a stacked deep learning model combining EfficientNetB0, ResNet-152, and VGG19 pre-trained on ImageNet.

2.4.1 Dataset and Preprocessing

The study utilizes the Mayo Clinic-STRIP AI dataset, which comprises 754 high-resolution WSIs in TIFF format, representing blood clots. This dataset exhibits a class imbalance, containing 547 images classified as CE and 207 images classified as LAA.

Preprocessing involves several steps to prepare the images for training. First, the WSIs are divided into patches using the openslide deep zoom generator, discarding tiles with white backgrounds or those not meeting the mean (<180) and standard deviation (>50) thresholds. Next, color normalization is performed using Marc Macenko's algorithm to address variations in color intensity and shading. Data augmentation techniques such as rotation, brightness adjustment, zooming, and flipping are applied, and images are rescaled by a factor of 255. To handle the class imbalance, up to 6 tiles from each CE image and up to 22 tiles from each LAA image are selected, resulting in a balanced training set of 2569 CE images and 2302 LAA images, all preprocessed and color normalized.

2.4.2 Models

The proposed model for classifying blood clot stroke origin is a stacked architecture combining EfficientNet-B0, VGG19, and ResNet-152. Preprocessed images are input into this model, which leverages the pre-trained backbones of these CNN architectures, initially trained on the ImageNet dataset, and fine-tunes them for this specific task. The model uses an ensemble technique without assigning weightage based on individual model performance. It accepts RGB images of 512x512 pixels, and the feature extractor outputs from EfficientNet-B0, ResNet-152, and VGG19 have shapes of 16x16x1280, 16x16x2048, and 16x16x512, respectively. These are passed through a global average pooling layer, flattened to shapes of 1x1280, 1x2048, and 1x512, and concatenated to form a single output of 1x3840.

Following the concatenation, a batch normalization layer is applied, and the data is passed through a hidden dense layer with 128 neurons using the ReLU activation function. The output layer consists of two neurons with a softmax activation function, providing the probabilities for each class label. The entire model comprises 82,952,165 parameters, with 499,586 being trainable. This architecture is fine-tuned using the Mayo Clinic-STRIP AI dataset to identify the optimal set of parameters.

2.4.3 Results

A total of 20 experiments were conducted, with experiments 1 to 10 using individual models and experiments 11 to 20 using stacked combinations of the three models. The experiments varied the input size and adjusted hyperparameters such as the number of epochs and batch sizes during training. Experiment 20, which employed a stacked model of EfficientNet-B0, ResNet-152, and VGG19 with an input size of 512, 40 epochs, and a batch size of 32, achieved the lowest weighted multi-class logarithmic loss value of 0.69312 among all the experiments. The results demonstrate that the proposed architecture effectively classifies the ischemic stroke blood clot origin between LAA and CE using WSI images.

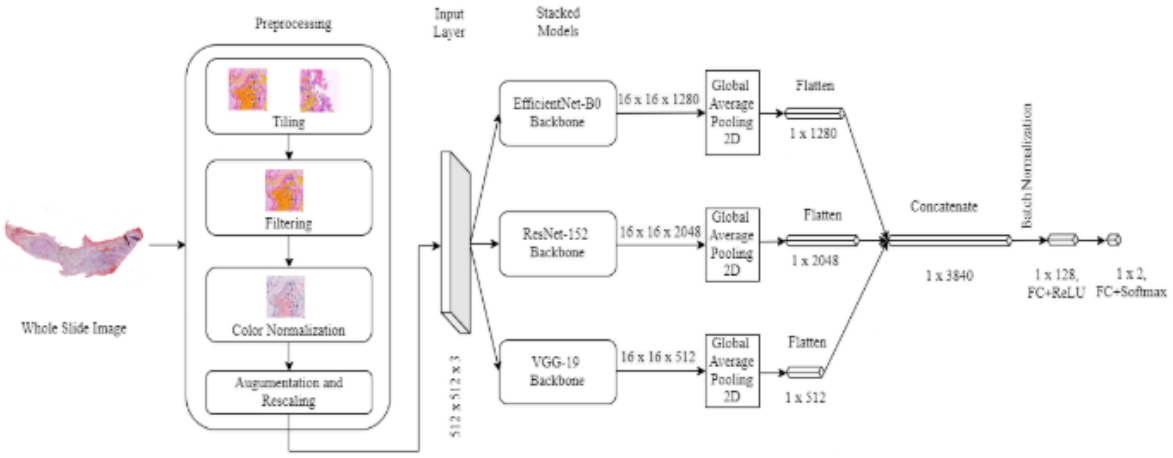


Figure 2.8: Architecture of stacked EfficientNet-B0, VGG19 and ResNet-152 model

2.5 Biologically Informed Clot Histomics Are Predictive of Acute Ischemic Stroke Etiology

The paper[12] presents a first-of-its-kind histomics pipeline to robustly quantify the complex structure and WBC heterogeneity in AIS clots and classify cryptogenic cases. This was possible by hypothesizing that histomic features of stroke blood clots retrieved by MT could be used to delineate stroke etiology.

2.5.1 Dataset and Feature Extraction

The dataset used in the paper consisted of 107 thrombi retrieved from patients with AIS who underwent MT at a single center between 2015 and 2018. The dataset included thrombi from patients with different stroke subtypes, including LAA, CE, and UE.

➢ Extracting RBC/FP Features:

The study focused in this step on extracting textural, spatial distribution, and geometric features to help in comprehensive analysis and understanding of the thrombi's characteristics. The RBC and FP components are segmented from low-resolution WSIs using Orbit Image Analysis. This step is crucial for isolating RBCs and FPs from the rest of the components in WSI. Images are then converted to grayscale using the standard formulation in scikit-image.

The open-source radiomics package called PyRadiomics is used to extract clot texture features. PyRadiomics extracts features from pixel intensity distributions, such as first-order statistics, gray-level run-length matrix, gray-level dependence matrix, gray-level co-occurrence matrix, and gray-level size zone matrix. Computational image analysis techniques are employed to engineer features descriptive of clot organization.

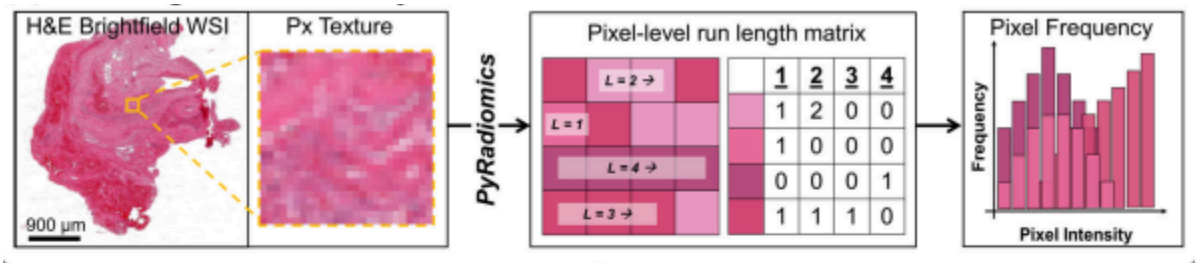


Figure 2.9: Extracting features with PyRadiomics

The distance transform, which represents the pixel distance from the outer edge of an image object, is calculated for the segmented clot region. The distance transform is then divided into 10 quantiles to capture increasing distances from the thrombus outer edge. This division ensures rotational invariance. The quantiles are used to determine the percentage composition of RBC, FP, or WBC classes within each quantile. RBC- and FP-enriched regions within each thrombus are identified by computing the thrombus area and selecting regions with an area greater than or equal to 5% of the clot area. This step aims to identify specific regions within the thrombus that are predominantly composed of RBC or FP components.

Geometric features such as area, perimeter, and extent are engineered from all the identified RBC and FP regions using the regionprops function. These features capture the size and shape characteristics of the enriched regions. Finally, the RBC/FP feature set consists of 227 features, which are likely a combination of features extracted from PyRadiomics, quantile-based analysis, and regionprops.

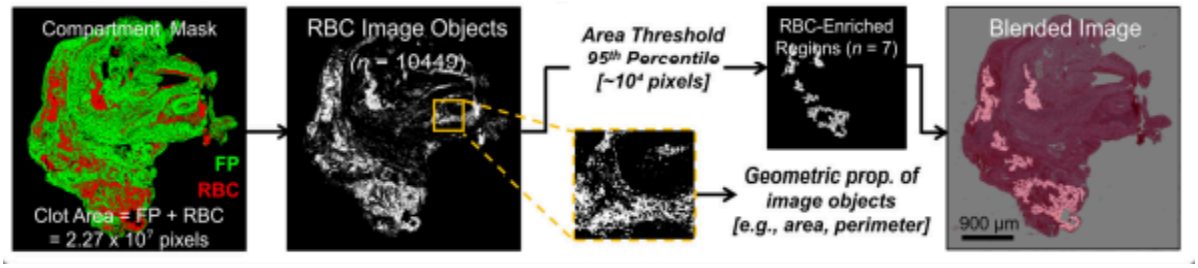


Figure 2.10: Engineering RBC and FP object features

➢ Extracting WBC Features:

Because the characterization of individual WBCs requires precise delineation of each nuclear instance, a high-resolution approach was taken for WBC segmentation. A whole-slide patching strategy is employed, breaking down WSIs into smaller, overlapping patches of fixed size. This facilitates targeted processing at the WBC level within each patch, enhancing computational efficiency.

Quality Control (QC) and Artifact Removal (AR) involve extracting features from patches using a pre-trained CNN. K-means clustering groups patches based on features, and expert review classifies them as "quality" or "artifact." QC Score Calculation computes the percentage of "quality" patches for each WSI, establishing a QC score. WSIs with scores below 40% are excluded, setting a threshold for data quality and ensuring robustness in subsequent analyses.

WBC detection accuracy is rigorously evaluated, demonstrating high agreement between automated segmentations and manual annotations using Krippendorff's alpha (0.8). Subsequently, textural features are extracted from both nuclear and extranuclear regions of WBC instances. Intranuclear texture is measured through the analysis of nuclear line profiles, and extranuclear regions (Zone-1 and Zone-2) are created. Texture quantification using PyRadiomics involves converting image patches to grayscale, and extracting relevant features, resulting in a comprehensive set of 645 features.

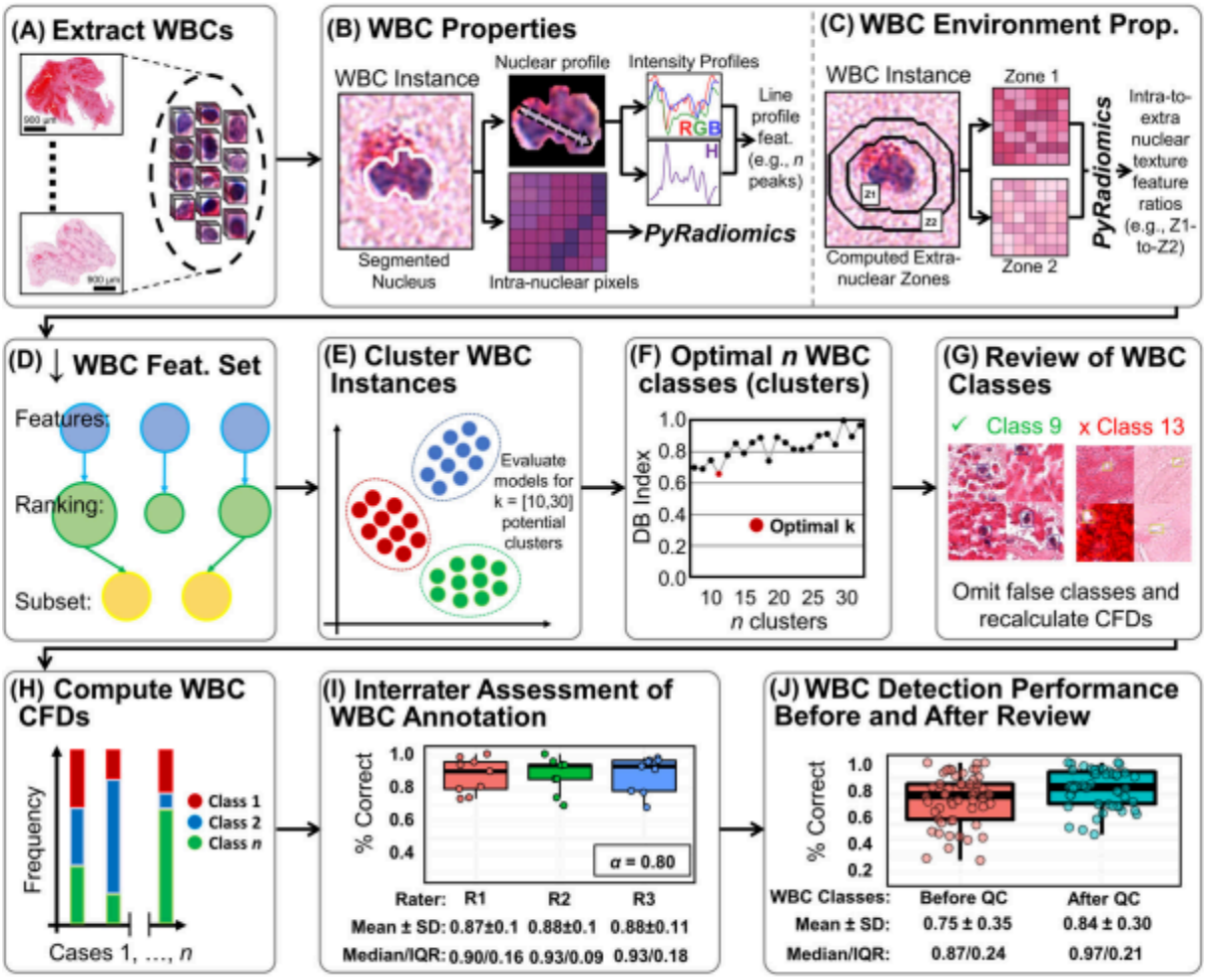


Figure 2.11: Extracting WBC features

2.5.2 Results

The study found that the top 3 selected features extracted from the RBC/FP regions of thrombi were able to differentiate between LAA and CE stroke etiologies. These RBC/FP features proved to be more discriminative compared to the currently used clot component metrics, offering the potential for improved differentiation between LAA and CE clots in stroke cases.

		Training				Testing			
Feature set	Fold	Accuracy	Sensitivity	Specificity	AUC	Accuracy	Sensitivity	Specificity	AUC
RBC/FP and WBC CFD features (n=6 features)	Fold 1	0.77	0.76	0.80	0.84	0.77	0.80	0.67	0.83
	Fold 2	0.77	0.76	0.80	0.92	0.69	0.80	0.67	0.87
	Fold 3	0.73	0.75	0.67	0.83	0.85	0.73	1.00	0.91
	Mean±SD	0.76±0.02	0.76±0.01	0.76±0.06	0.86±0.04	0.77±0.06	0.78±0.03	0.78±0.16	0.87±0.03

Table 2.12: Performance analysis

2.6 Correlation of Imaging and Histopathology of Thrombi in Acute Ischemic Stroke with Etiology and Outcome

The paper[13] reviewed the current literature on the histopathological composition of AIS clots. Numerous features of thromboemboli could be studied and characterized, including quantitative histomorphometry and diagnostic imaging characteristics. Accurately identifying the composition of the occlusive clot before intervention could significantly influence the success of the revascularization strategy used to treat them. The paper's main focus was on the correlation between clot composition and diagnostic imaging, stroke etiology, and revascularization outcomes. The authors also discussed types of pathology image stains and their effect on certain features in the thrombus images, allowing them to dive into imaging analysis and clot characterization.

2.7 Machine Learning in Action: Stroke Diagnosis and Outcome Prediction

The paper[14] provides an overview of ML technology and a tabulated review of pertinent ML studies related to stroke diagnosis and outcome prediction to evaluate the current state and application of the technology. Although image-based models have significantly facilitated rapid stroke diagnosis, stroke prognostication is dependent on a multitude of patient-specific factors. Also, oversight from clinical experts is still required to address specific aspects not accounted for in an automated algorithm. Hence, accurate outcome predictions remain challenging.

2.7.1 Stroke Diagnosis and Outcome Prediction

Stroke Diagnosis is time-sensitive due to the nature of stroke; stroke care underpins the need for accurate and rapid tools to assist in stroke diagnosis. Over the past decade, 13 different companies have developed automated and semi-automated commercially available software for acute stroke diagnostics, and their evaluation metrics appear to be excellent.

The authors reviewed a total of 54 studies utilizing ML for stroke diagnosis and prediction, each of these studies with a specific objective, different approaches, features, results, clinical implications, and limitations. The approaches of each one ranged from classical ML algorithms, both supervised and unsupervised, to deep CNN models with features ranging from medical records and clinical notes to MRI images and CT scans with various limitations to each of them.

2.7.2 Evaluation

Deep learning has significantly enhanced practical applications of ML and some newer algorithms are known to have comparable accuracy to humans. However, the diagnosis and prognosis of a disease, including stroke, is highly intricate and depends on various clinical and personal factors. Developing optimal ML programs requires comprehensive data collection and assimilation to improve diagnostic and prognostic accuracy.

Given the “black box” or cryptic nature of these algorithms, the end-user (i.e., clinicians) must understand the intended use and limitations of any ML algorithm to avoid inaccurate data interpretation. Although ML algorithms have improved stroke systems of care, blind dependence on such computerized technology may lead to misdiagnosis or inaccurate prediction of prognostic trajectories. At the current state, ML tools are best used as “aids” for clinical decision-making while still requiring oversight to address relevant clinical aspects overlooked.

2.8 Histological Stroke Clot Analysis After Thrombectomy: Technical Aspects and Recommendations

The paper[2] provides guiding information on thrombus handling, procedures, and analysis to facilitate and standardize this emerging research field. The recent advent of endovascular procedures has created the unique opportunity to collect and analyze thrombi removed from cerebral arteries, instigating a novel subfield in stroke research. Insights into thrombus characteristics and composition could play an important role in ongoing efforts to improve AIS therapy.

Chapter III: Materials and Methods

3.1 Dataset

The dataset consists of over a thousand high-resolution whole-slide digital pathology images, each portraying a blood clot from a patient who suffered an AIS. These images were in TIFF format, which is a standard format for digital pathology images. The slides, sourced from the Mayo Clinic as part of STRIP AI's competition[15], are divided into training and test sets, showcasing clots with known etiologies labeled as either CE or LAA. Specifically, the dataset includes 547 images associated with CE etiology and 207 images associated with LAA etiology. Additionally, CSV files are included in the dataset, containing annotations such as image_id, center_id, patient_id, image_num, and label for the images. More details on digital pathology images can be found in subsection 1.2.3.

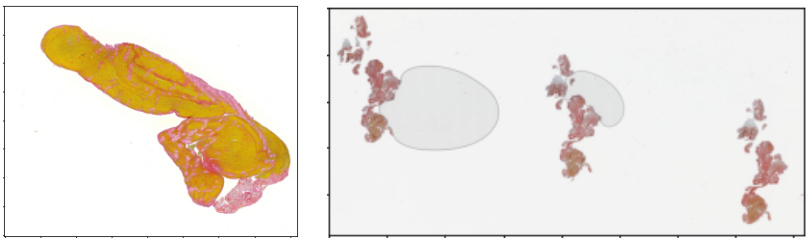


Figure 3.1: Whole-slide digital pathology images (CE and LAA)

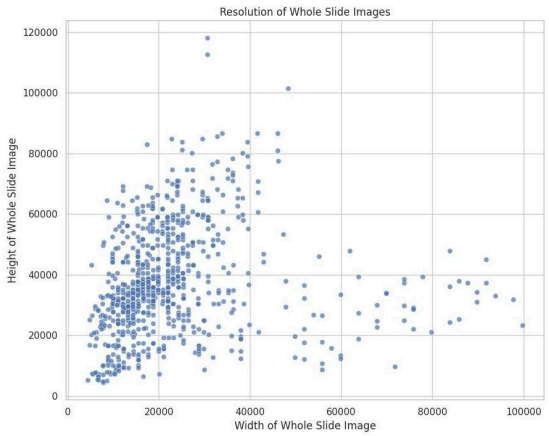


Figure 3.2: Graphical representation of dataset image resolution in pixels

3.2 Machine Learning Approach

3.2.1 Preprocessing

Preprocessing is a crucial step in data preparation that involves cleaning, organizing, and transforming raw data into a format suitable for ML tasks.

➢ Image Tiling:

The large size of the images (1-2 GB on average) presented a significant challenge in the preprocessing stage. Due to memory constraints, direct processing or prediction was not possible. To overcome this hurdle, a tiling approach was implemented where the image was subdivided into smaller tiles, however, memory limitations persisted even during the tiling process. Therefore, a two-step approach was implemented using the libvips[16],[17] library, which processes the images in chunks. First, a low-resolution thumbnail (20,000 x 20,000 pixels) of the image is created. Afterward, the tiling process is performed on the thumbnail, effectively alleviating memory constraints. Finally, the tiles are sorted based on the intensity of the tile, and only the 16 darkest tiles are picked, being the tiles with the most content.

Another key challenge associated with the dataset is the inherent class imbalance. The challenge lies in the significant class imbalance, with the majority class (CE) containing 73% of the data. To address this, a more selective tiling approach was adopted during preprocessing. This approach involves utilizing fewer tiles per image belonging to the overrepresented class. While this reduces the overall data volume for the majority class, it maintains a sufficient number of tiles per image for informative representation. This strategy is particularly suitable due to the utilization of a machine learning model rather than a deep learning one, where data quantity is less critical for robust performance.

➢ Tile Segmentation:

Classifying each pixel in the whole slide image as one of the three components of the blood clot (RBCs, WBCs, or FPs) is the goal throughout the segmentation phase. The Hue, Saturation, and Value (HSV[18]) color model, well-known for its ability to extract color information from an image pixel, was employed during this phase. As discussed in subsection 1.2.3, the blood clot color observations in the dataset images show RBCs as yellowish to brown hues, WBCs as purple hues, and fibrin/platelets as reddish violet hues. After pinpointing which pixel belongs to which component, this data is combined from all of the pixels ultimately segmenting the areas in the image to the components they belong to. Following the conversion of the tiles to HSV color space, a binary mask is created per component for each tile.

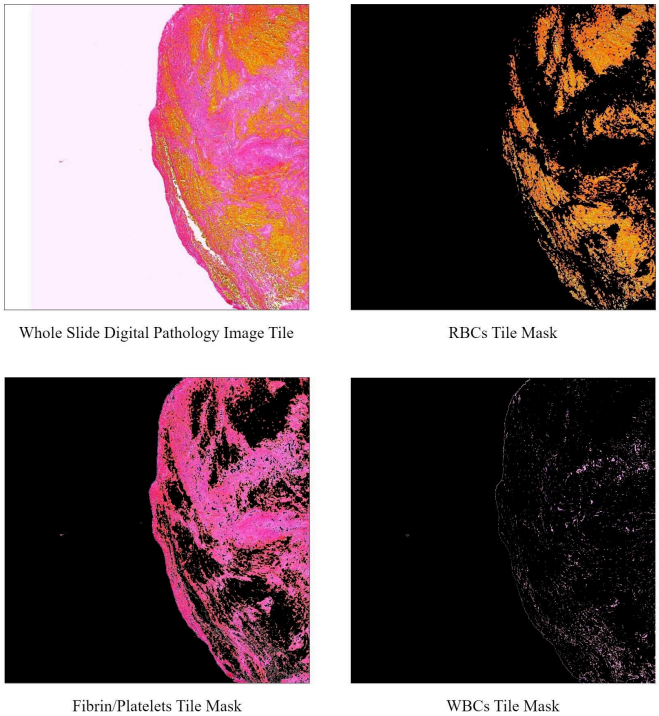


Figure 3.3: Whole slide digital pathology image tile and the binary masks for each component

3.2.2 Feature Extraction

After the segmentation process, the tiles and their masks are transformed into the SimpleITK[19]-[21] format–a user-friendly interface layered over the Insight Segmentation and Registration Toolkit (ITK)–to be then fed to the PyRadiomics[22] feature extractor where geometric distribution, spatial distribution, and textural data are extracted. PyRadiomics is an open-source Python package that provides a comprehensive set of tools for extracting quantitative features from medical images. The features that were extracted by PyRadiomics include First Order Statistics, Shape-based (3D), Shape-based (2D), Gray Level Co-occurrence Matrix, Gray Level Run Length Matrix, Gray Level Size Zone Matrix, Neighboring Gray Tone Difference Matrix, and Gray Level Dependence Matrix. Lastly, the features are merged and stored as a CSV file to be supplied to the machine learning model.

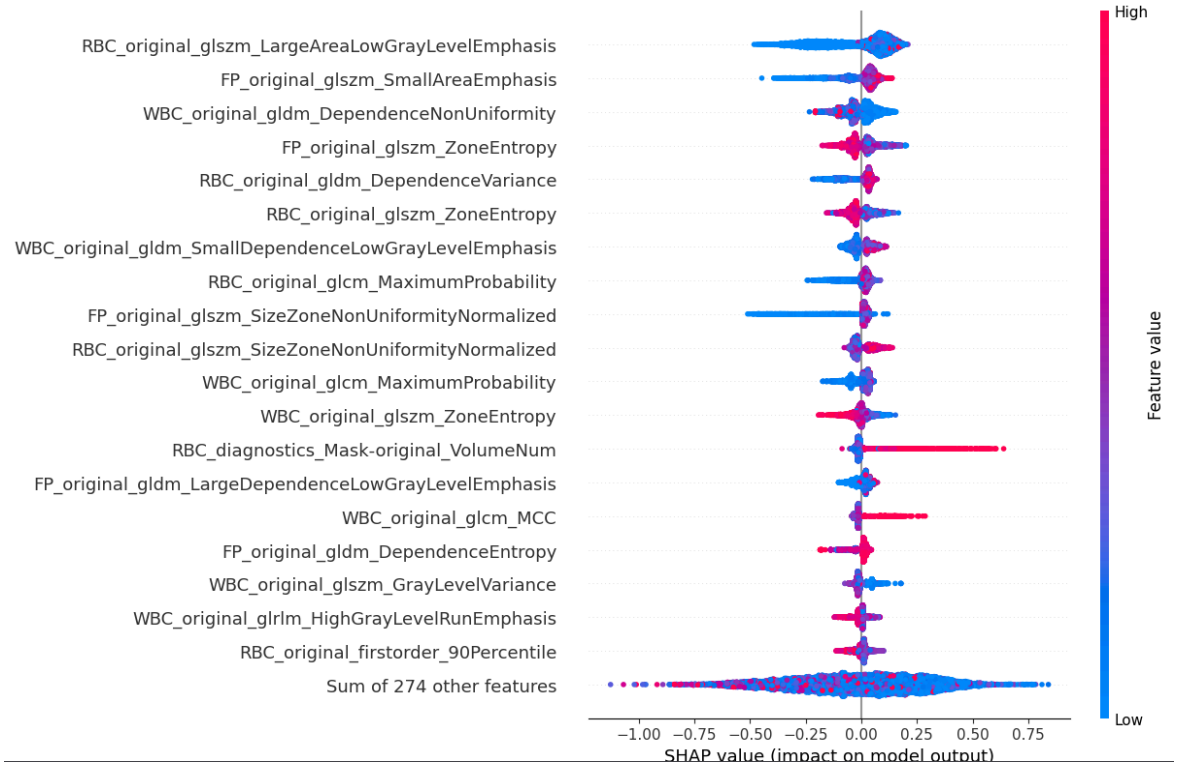


Figure 3.4: Graphical representation of the most significant features

3.2.3 Feature Selection

Feature selection provides a method to discard redundant data, which improves learning accuracy and reduces computation time. In this project, the Mann–Whitney U[23],[24] test for nonnormally distributed data is employed to select features based on a predefined P-value[25] threshold. The P-value is used to measure the statistical significance of a given feature. If the value is closer to 1, it suggests that the observed difference is likely due to chance. Conversely, if the value is closer to 0, it indicates that the observed difference is potentially significant. This method was used to set a maximum threshold P-value for the extracted features after calculating the value of each feature.

3.2.4 Model

Following a comprehensive review of relevant research and exploratory experimentation, the XGBoost model was identified as the most suitable choice for the project's ML objective. The XGBoost[26] model incorporated three key features that significantly impacted performance and contributed to a substantial decrease in loss. Firstly, XGBoost's feature normalization was utilized to ensure a positive impact on model performance. Secondly, the Minimal Cost-Complexity Pruning (MCCP[27]) feature in XGBoost was enabled to prevent overfitting by pruning the decision tree, yielding the most significant performance improvement. Lastly, the log loss function was selected and employed during both training and evaluation within XGBoost. While the exponential loss function led to a slight increase in accuracy, it also resulted in a considerably higher loss. Given the competition's focus on minimizing loss rather than maximizing accuracy, the log loss function in XGBoost was ultimately chosen as the preferred method.

During the training phase, the model was trained on individual tiles as if each tile was a separate image with its own extracted features. During testing, the model classifies each tile of a test image separately, and then the final classification is based on the average of the individual probabilities of the classified tiles. The architectural representation of this methodology is illustrated in Figure 3.3.

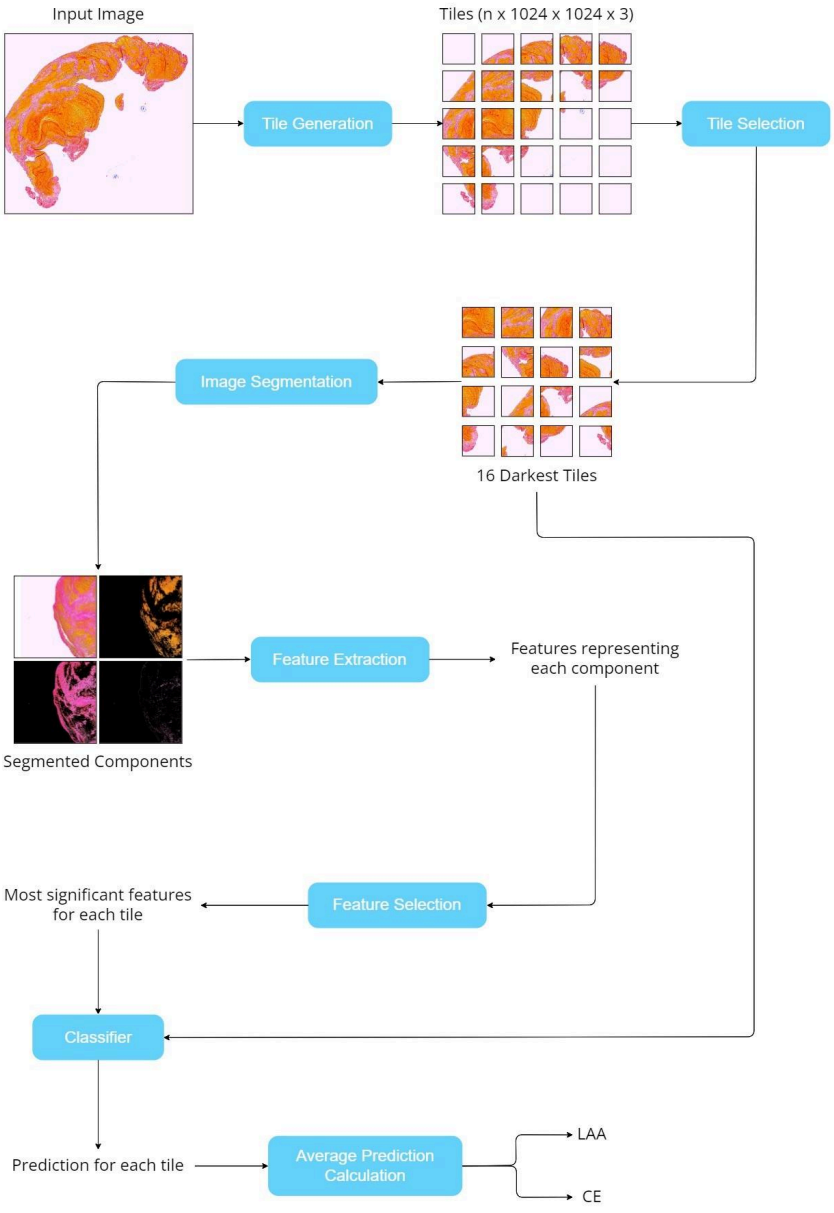


Figure 3.5: Architecture of proposed ML system

3.3 Deep Learning Approach

3.3.1 Preprocessing

In the context of high-resolution whole-slide digital pathology images, preprocessing is especially important to enhance the quality and usability of the data, particularly to be suitable for DL tasks.

➢ Seam Carving:

Seam carving is a technique for resizing medical images in a preprocessing pipeline. The script is carefully configured to ensure optimal image processing conditions, such as removing pixel limits and applying energy constraints. Each medical image undergoes a sequence of transformations. Initially, it is resized to a thumbnail size of (2048, 2048) pixels. Seam carving is then applied to further resize the images in a content-aware manner, refining them to a final size of (256, 256) pixels. This process optimizes image size and removes unnecessary white spaces or redundant features. The processed images are saved in the output directory in PNG format.

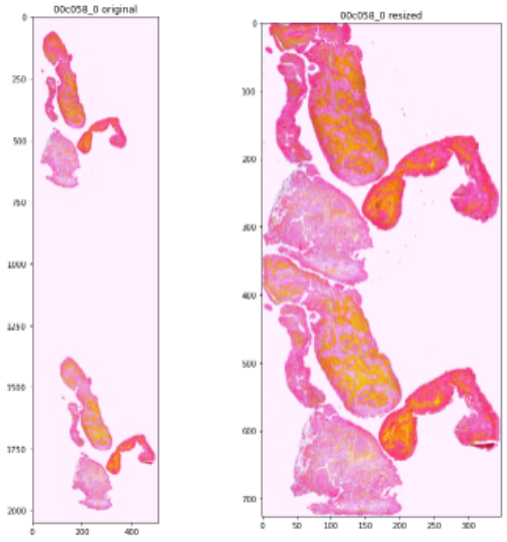


Figure 3.6: Image before and after seam carving

In our efforts to improve preprocessing, we leveraged the seam carving library to enhance the cropping process and eliminate unnecessary patches. To augment the results, we introduced a modified approach. Firstly, we divided the images into two parts and applied seam carving independently to each section. Subsequently, we incrementally augmented the smaller image with white areas until it matched the dimensions of the larger image. We then merged both parts to create a complete image. While this approach yielded a reduction in the number of patches, we encountered a drawback in the form of a discontinuity in the middle of the resulting images.

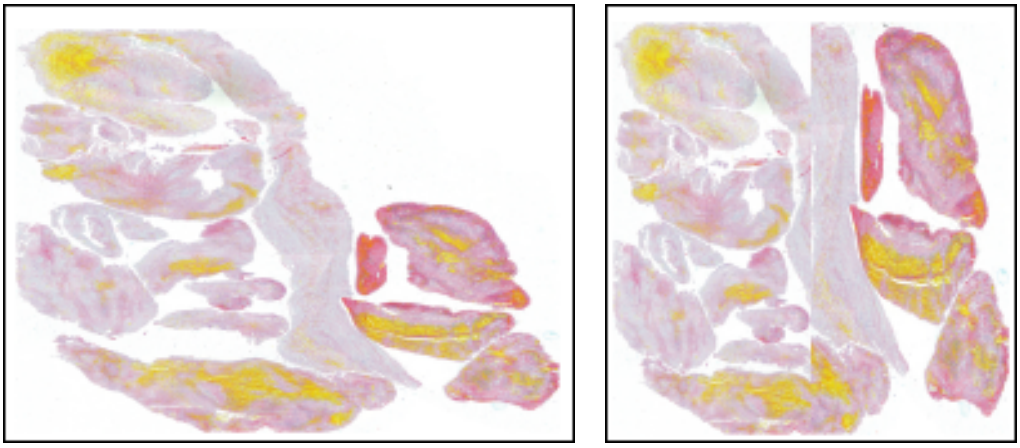


Figure 3.7: Image before using the approach (left) number of patches: 750,
Image after using the approach (right), number of patches: 650

While seam carving is great for resizing images, we have faced some limitations using it, particularly in preserving specific content or recognizing sophisticated details. These limitations cause distortions making the model unable to detect distinctive features. We also suffered from its slow running time due to the large size of our dataset, limiting its suitability for real-time applications. Also, Seam carving might lack user-guided control in certain implementations.

➢ Image Tiling:

Similar to the process in subsection 3.2.1, tiling is the process of dividing images into patches (tiles). We tried our best to keep the highest possible resolution to protect the images' characteristics since here we use resizing. We began by installing pyvips, a Python binding for the libvips[16],[17] image processing library. Libvips excels in the efficient and high-performance processing of large images. pyvips enables Python developers to leverage libvips for tasks such as resizing, cropping, and compositing in their applications.

We define a function named read_and_resize_image, which takes an image path as input, checks its size, calculates a scale factor depending on whether it exceeds the limit image size (178956970 Bytes) for the Kaggle notebook or not, and then uses the pyvips library to resize the image accordingly. This function is designed to handle images of varying sizes efficiently without losing a significant amount of resolution.

The main focus of the code is the function divide_tiff_into_tiles. This function reads the resized TIF image returned from the read_and_resize_image function and subsequently divides it into smaller tiles of size (512, 512). The tiles are then processed using the pyvips library, specifically employing its cropping functionality. Notably, the code includes a check to discard tiles with an average intensity greater than a certain threshold (185), we do this to filter out less informative regions, like backgrounds, from the image. The following image produced 962 useful tiles, some may go up to hundreds of thousands of tiles. Unlike the ML approach, here we keep all the generated tiles.

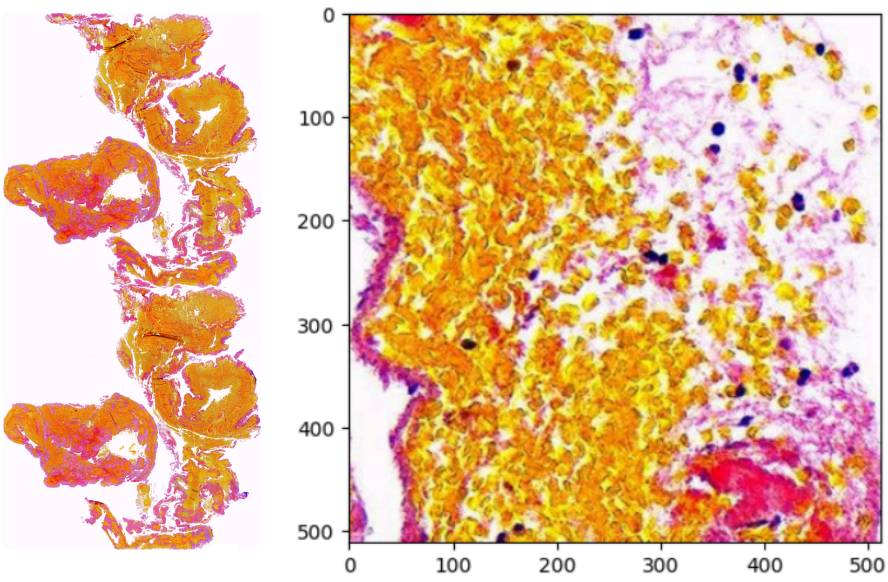


Figure 3.8: Original image (left) and Tile no. 500 (right)

➢ Data Augmentation:

We employed the Albumentations[28] library for image augmentation to enhance our training dataset by generating new samples from the existing data. Our objective was to achieve a more balanced dataset of 1400 images with 700 samples for each LAA and CE class. Initially, our dataset consisted of 754 total images: 547 CE images and 207 LAA images. By carefully fine-tuning the augmentation probabilities, we effectively expanded our dataset while maintaining a balanced distribution of samples across both classes.

To augment our data, we employed non-destructive functions that preserve essential image information. Our pipeline included Transpose() to swap rows and columns, VerticalFlip() and HorizontalFlip() to reverse the top-bottom and left-right sides, respectively, and RandomRotate90() for 90-degree rotations. We used ShiftScaleRotate() for random shifts, scaling, and rotation.

3.3.2 CNN Model

In the first phase of our project, we initially wanted to experiment and implement a CNN model architecture with a final sigmoid activation for binary classification similar to that of a related work mentioned in section 2.2. The dataset was split into 80% training and 20% validation and taken directly from the data frame containing the paths as well as the labels with no transformation or augmentation in between.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2  (None, 99, 99, 32)        0
 D)

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPoolin  (None, 48, 48, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 128)       73856

 max_pooling2d_2 (MaxPoolin  (None, 23, 23, 128)       0
 g2D)

 conv2d_3 (Conv2D)           (None, 21, 21, 128)       147584

 max_pooling2d_3 (MaxPoolin  (None, 10, 10, 128)       0
 g2D)

 dropout (Dropout)           (None, 10, 10, 128)       0

 flatten (Flatten)           (None, 12800)             0

 dense (Dense)               (None, 512)               6554112

 dense_1 (Dense)             (None, 1)                 513

=================================================================
Total params: 6795457 (25.92 MB)
Trainable params: 6795457 (25.92 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Figure 3.9: Implemented model,
validation accuracy = 71% and validation loss = 58%

After this experiment, we tried the same model on our augmented data, the augmentation pipeline mentioned in subsection 3.3.3. An unexpected observation emerged from our augmentation effort. Contrary to the initial expectations, increasing the dataset size led to a reduction in the model's validation accuracy percentage: validation accuracy = 48% and validation loss = 69%. This irregular outcome highlighted the complexity of the model performance and the need for further investigation to optimize our approach.

3.3.3 EfficientNet Model

➢ First Attempt:

EfficientNet-B2 is a variant of the EfficientNet[29] family of CNN models, known for their exceptional performance on both imagenet and common image classification transfer learning tasks. EfficientNet-B2 is an intermediate-sized model within the EfficientNet series, sitting between EfficientNet-B1 and EfficientNet-B3 in terms of depth and width. EfficientNet is among the most efficient models that reach SOTA accuracy.

To handle the dataset, we created The AIS_Dataset class which is a subclass of torch.utils.data. This class consists of an initialization function (init), a length function (len), and a getitem() function. The init() function takes data information as input and initializes the transforms() function, which resizes the image to (260, 260). The len() function returns the total number of rows in the dataset. The getitem() function retrieves a single sample from the dataset, reads and preprocesses the image, and returns the image along with its corresponding label. To facilitate training the EfficientNet model, PyTorch DataLoaders are created for training, validation, and testing, enabling the loading of data from the AIS_Dataset and the creation of mini-batches for training.

For our image classification task, The Pre-Trained EfficientNet-B2 model, initially trained on the ImageNet dataset, was loaded and subsequently fine-tuned using our own data. Since the original model classified images into 1000 classes, we modified the last layer to classify images into two classes. The training process involved iterating over the training data for multiple epochs and evaluating the model's performance on the validation data after each epoch. Within each epoch, the data was passed through the model, the loss was calculated, and the model parameters were updated accordingly. However, our observations indicated that the model suffered from overfitting, as the training accuracy reached 94% while the validation accuracy stagnated at 67%. To assess the model's performance, we employed the test set and it achieved an accuracy of 70%.

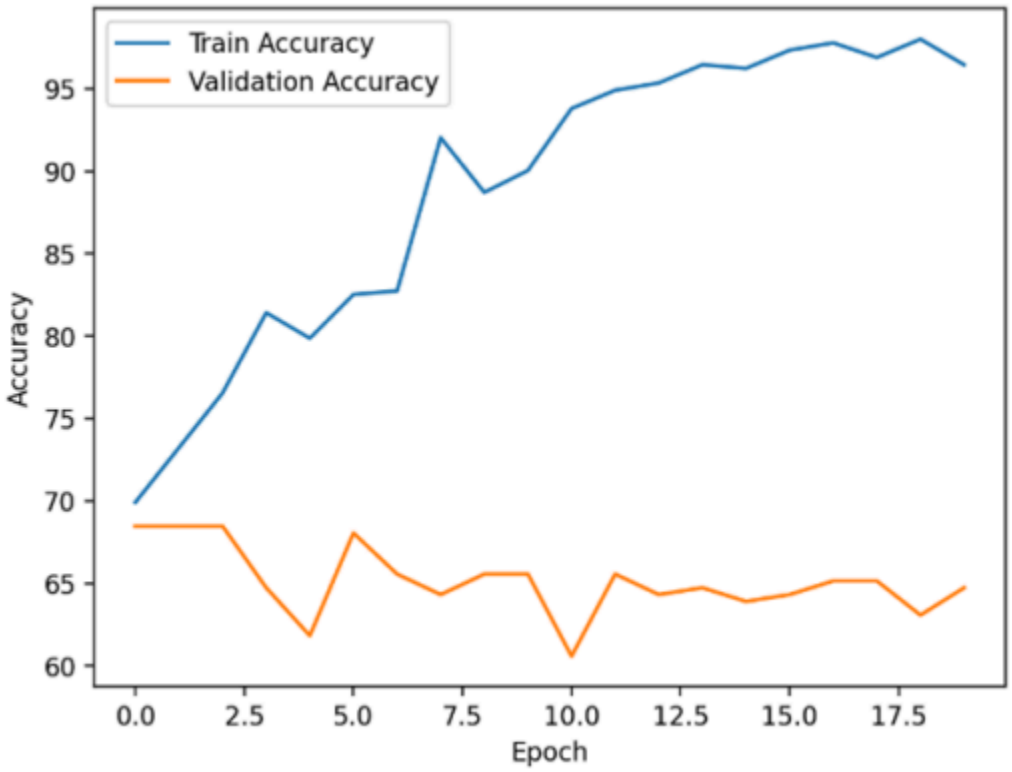


Figure 3.10: Train vs. Validation accuracy

➢ Second Attempt:

For the base model, we used EfficientNet-B0 which is the smallest, simplest, and fastest base model in the EfficientNet family. We chose the simplest EfficientNet version due to the small training size of our dataset to avoid the overfitting problems experienced in the EfficientNet-B2 and to bypass more computationally expensive versions.

Then we removed EfficientNet's classification head and added an adaptive average pooling layer for dimensionality reduction, followed by a dense layer and a linear layer, then finally a sigmoid layer as an activation function for binary classification. The EfficientNet base model was frozen and only the added layers were trainable due to the limited dataset.

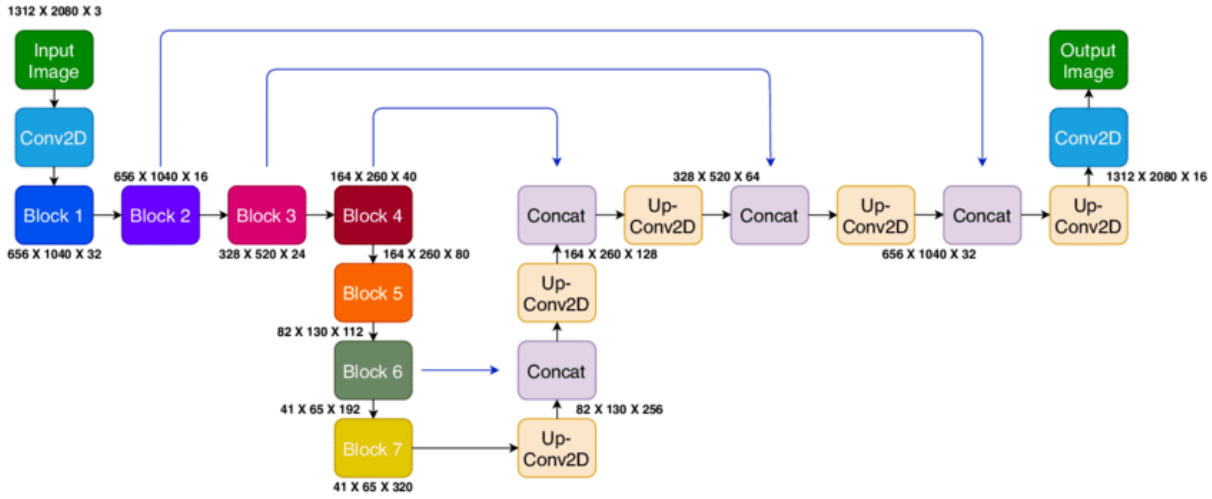


Figure 3.11: Architecture of EfficientNet with EfficientNet-B0 framework[30]

3.3.4 Ensemble Model

An ensemble[31] model is a ML technique that combines the predictions from multiple individual models to produce a more accurate and robust prediction than any of the individual models alone. In our ensemble model, we used three different pre-trained DL models: ResNet[32], GoogLeNet[33], and DenseNet[32], each of which offers distinct architectures and capabilities. The primary objective is to combine their predictions through a voting mechanism to enhance classification accuracy.

ResNet (Residual Network) is a popular DNN architecture known for its skip connections or residual connections. These connections enable the model to effectively learn and optimize deeper networks. The ResNet model is well-suited for image classification tasks due to its ability to handle a large number of layers.

GoogLeNet is characterized by its inception modules, which consist of multiple parallel convolutional layers with different filter sizes. This architecture helps capture features at different scales. GoogLeNet is known for its computational efficiency and performance in various computer vision tasks.

DenseNet (Densely Connected Convolutional Networks) introduces dense connections between layers. In this architecture, each layer receives input from all previous layers, which encourages feature reuse and gradient flow. DenseNet is efficient and often results in improved feature learning and model performance.

After training the three models individually on the same dataset, we combined their predictions through a voting mechanism. As shown in Table 3.12, we can see that the model doesn't predict the class LAA.

	Precision		Recall		F-score		Accuracy
Models	CE	LAA	CE	LAA	CE	LAA	%
ResNet	0.67	0.00	0.98	0.00	0.79	0.00	66
GoogLeNet	0.67	0.33	0.95	0.05	0.79	0.09	66
DenseNet	0.70	0.60	0.95	0.15	0.80	0.24	69
Combined	0.67	0.00	1.00	0.00	0.80	0.00	67

Table 3.12: Ensemble prediction results

3.3.5 Explainable Artificial Intelligence

XAI refers to the development of AI systems and models that can provide understandable and interpretable explanations for their actions and decisions. The goal of XAI is to make the decision-making processes of AI systems transparent and comprehensible to humans, allowing users to trust, validate, and understand the outputs generated by these systems.

➢ Grad-CAM:

An explainability technique we experimented on to better understand our model was Grad-CAM[34]. Grad-CAM utilizes the gradients of the classification score concerning the final convolutional feature map, to identify the parts of an input image that most impact the classification score. The places where this gradient is large are the places where the final score depends on the most.

Grad-CAM was used with the initial dataset and the implemented CNN model. It works by providing a heat map for the input to visualize with parts of the input image contributing more to the output. As we can observe in Figure 3.7, the heatmap response is high in some empty parts of the image. Although the model has almost 80% accuracy, it is not suitable for use outside its original dataset.

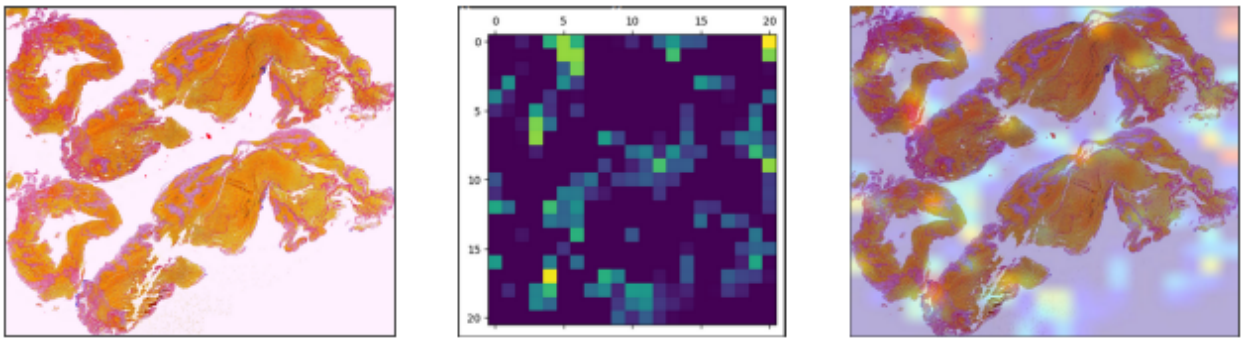


Figure 3.13: Heatmap response on image from our dataset

➢ LIME:

LIME[35] is an XAI technique designed to provide insights into the black-box behavior of ML models. LIME generates interpretable explanations for model predictions by creating a locally faithful approximation of the model's decision boundary. It does this by perturbing input data points and observing how the model's predictions change.

By comparing the model's behavior with and without perturbations, LIME identifies which features of the input are most influential in making a particular prediction, offering transparency and interpretability in complex AI systems. LIME is a valuable tool for improving model trustworthiness, aiding in model debugging, and ensuring that AI systems make decisions that align with human expectations and domain expertise.

As seen in Figure 3.14, the picture on the left is the original image, and the picture on the right is the image after the explanation with LIME. It can be observed that LIME puts a yellow border around the part that the model used to predict the output.

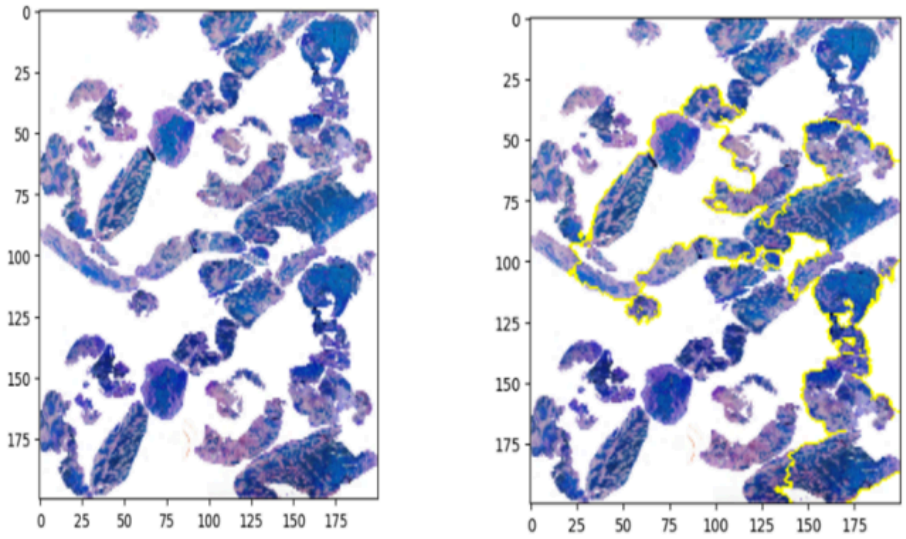


Figure 3.14: Image before and after LIME

3.4 Explored But Unimplemented Methods

In this section, we noted several innovative methods that had been documented but were not used in our project. While these methods showed promise in various studies, their integration into our project proved challenging due to compatibility issues, divergent project requirements, or time constraints. Furthermore, during our research journey, we discovered alternative and more tailored solutions that are better aligned with our project goals.

3.4.1 Swin Transformer

The Swin Transformer[36] is a DL architecture that combines the Transformer model with the concept of hierarchical patch-based processing. The following are the four stages of the Swin Transformer:

➢ Stage 1:

In the initial stage, the input image is divided into non-overlapping patches. Each patch is processed through a Swin Transformer block, capturing local information within each patch. The output of Stage 1 is a set of feature maps representing local details.

➢ Patch Merging:

After Stage 1, the patch tokens are reshaped to form a 2D grid, enabling global context modeling. Patch tokens are merged by rearranging them into a smaller number of larger patches. This reduces the spatial resolution while increasing the receptive field. The merged patches contain both local and global contextual information.

➢ Stage 2:

The merged patches from the previous stage are processed through another set of Swin Transformer blocks. This stage captures contextual information at a larger scale compared to Stage 1. The number of channels typically increases while the spatial dimensions decrease due to the merging operation

➢ Subsequent Stages:

Similar to Stage 2, each subsequent stage consists of Swin Transformer blocks that process the merged patches. Each stage further increases the receptive field and allows the model to capture contextual information at different scales. The number of channels typically increases while the spatial dimensions continue to decrease.

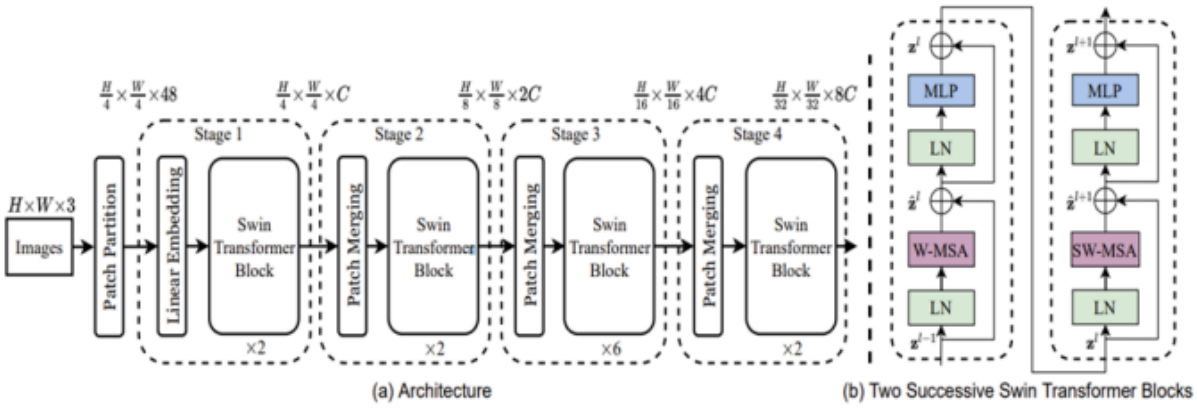


Figure 3.15: Swin transformer architecture diagram[37]

3.4.2 YOLO

We researched the YOLO[38] series of real-time object detection algorithms, which have gained widespread popularity in computer vision and DL communities. This series comprises multiple versions, including YOLOv4, YOLOv5, and several newer iterations like YOLOv8. Each version represents an enhancement over the previous one, offering improved performance, accuracy, features, accuracy, speed, and the capability to handle a broader range of object classes.

Initially, we explored the use of YAD2K, a tool designed primarily for object detection tasks, and the conversion of YOLO models trained in the Darknet framework to Keras format. Unfortunately, we encountered issues with the package, as it was missing essential Python files such as "get_colors_for_classes," "scale_boxes," "read_classes," "read_anchors," and "preprocess_image." Furthermore, YAD2K is not inherently designed to support instance segmentation, which is a more complex task involving both object detection and pixel-wise segmentation of object instances within an image.

After our issues with YAD2K, we encountered Roboflow, a comprehensive platform and toolset tailored for computer vision and ML practitioners, particularly those involved in image-related projects. Roboflow has recently become the official dataset management and annotation tool for YOLO, with a strong focus on enabling active learning. It offers a wide range of features and capabilities to streamline the data preparation, training, and deployment processes for computer vision models.

To demonstrate YOLO instance segmentation, we'll leverage a pre-trained model and then validate the model to show us the model's performance. we're training for x epochs. We're also starting training from the pre-trained weights. Larger datasets will likely benefit from longer training. Infer With Your Custom Model to show the resulting prediction overlayed on the input image.

Unfortunately, YOLO was detecting the gross or outer shapes of the thrombus fragments. This is inconsistent with our data as the model may detect LAA and CE presence in the same image which is impossible. We are not downplaying the capabilities of YOLO in object detection, but it seems that it is not suitable for our type of dataset.

3.4.3 Faster R-CNN

We have been investigating the application of the Faster R-CNN[39] model, commonly used for object detection tasks, to address our specific classification problem. The Faster R-CNN model is designed to detect various objects within an image, utilizing its inputs and outputs to identify the presence and location of objects by providing the coordinates (x_{min}, y_{min}, x_{max}, y_{max}) of bounding boxes around the detected objects.

To adapt this object detection model to our classification problem, we introduced a modification. We treated each image as a complete bounding box and assigned a label to it, which could be either CE or LAA. During testing, the model generates multiple labels for each input image, and the label with the highest score is considered the output of our model, even if there are other labels incorrectly detected by the model. It's important to note that in our problem, each image should only have one correct label. Similar to YOLO, Faster R-CNN identifies the overall or external forms of thrombus fragments.

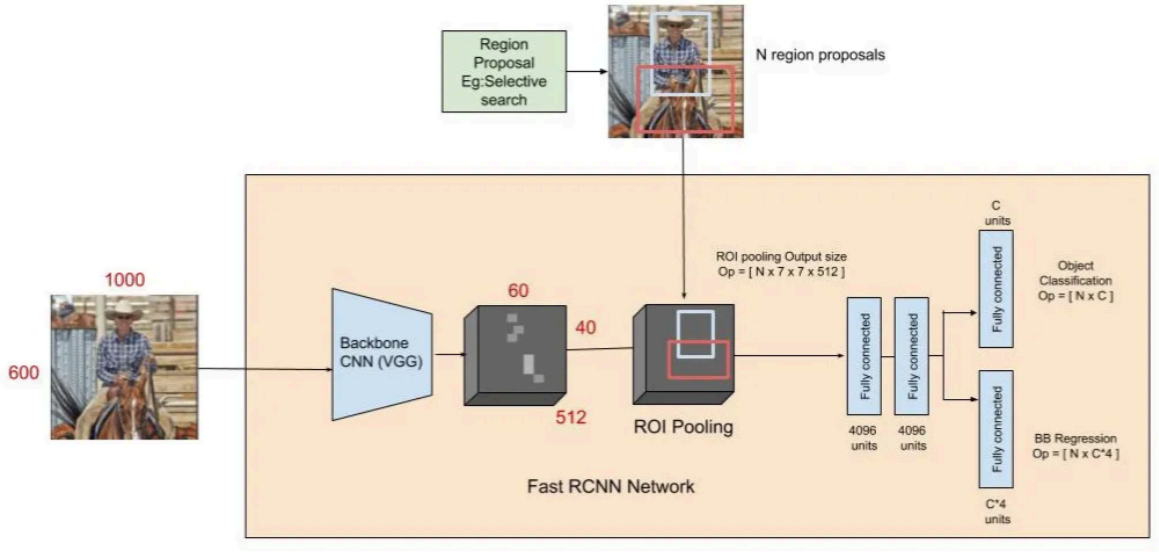


Figure 3.16: Faster R-CNN pipeline[40]

Chapter IV: Experimental Setup, Results, and Analysis

4.1 Experimental Setup

This project leveraged the Kaggle platform for computational reso[urces. Given the absence of GPU-intensive preprocessing or models, CPUs were exclusively employed to maximize available RAM. The computing environment provided 30 GB of RAM and 73.1 GB of disk space. To optimize workflow, image tiling and feature extraction were conducted in independent notebooks and stored as Kaggle datasets. Subsequently, the primary notebook served for training and testing. In particular, test image tiling and feature extraction were performed during the testing phase within the main notebook to ensure a realistic testing scenario.

4.2 Hyperparameter Settings

4.2.1 Machine Learning Hyperparameters

After a series of trial and error experiments, it was determined that the optimal hyperparameters for implementing the proposed ML methodology were 200 estimators, a maximum depth of 4, an MCCP[27] alpha of 0.0007, and a P-value[25] of 0.07 for feature selection. These values generated the best results during the study's evaluation process.

4.2.2 Deep Learning Hyperparameters

The proposed architecture employs the Adam optimization function[41] due to its minimal running time and dynamic adjustment of the learning rate for each neural network weight. The initial learning rate is set at 0.001. In this method, weights are updated based on a combination of the current and previous gradients, which helps in smoothing the updates.

4.3 Evaluation Criteria

Submissions to Kaggle's competition[15] are evaluated using a WMCLL function. The study's solution is then ranked by testing the model on a private dataset on Kaggle. Weights for each class have been added to improve this loss function, a variant of the conventional multi-class logarithmic loss. These weights highlight more significant classes by adjusting each class's contribution to the overall loss. Failed predictions are penalized by WMCLL, and the penalty is scaled based on the class weights that have been assigned. Because the loss is computed as the logarithm of predicted probabilities, accurate but confident predictions are penalized severely. The WMCLL formula is given as:

$$\text{Log Loss} = -\left(\frac{\sum_{i=1}^{M} w_i \cdot \sum_{j=1}^{N_i} \frac{y_{ij}}{N_i} \cdot \ln p_{ij}}{\sum_{i=1}^{M} w_i} \right)$$

where N is the number of images in the class set, M is the number of classes, **ln** is the natural logarithm, y_{ij} is 1 if observation i belongs to class j and 0 otherwise, p_{ij} is the predicted probability that image i belongs to class j.

4.4 Results and Analysis

As can be noticed in Table 4.1, the results established in this project successfully surpassed most previous approaches that aimed to solve the same problem with an identical dataset. The study in section 2.1 which relied on deep convolutional neural networks and visual transformers established a baseline WCMLL score of 0.67188. Another study, in section 2.4, that used Stacked EfficientNet-B0, VGG19, and ResNet-152, reached a WCMLL score of 0.69312. To also compare with studies that implemented ML methodologies, the study in section 2.5 that used a feature extraction and a classical machine learning model, Naïve Bayes Classifier, but on a different dataset established accuracy results around 77%. In contrast, the solutions introduced in this paper have shown a ML WCMLL score of 0.66588 and an accuracy of 79% and a DL WCMLL score of 0.67427 and an accuracy of 79%.

Models	WMCLL	Accuracy
Deep CNN and Visual Transformer, section 2.1	0.67188	–
Stacked EfficientNet-B0, VGG19, and ResNet-152, section 2.4	0.69312	–
Naïve Bayes Classifier, section 2.5	–	0.77±0.06
XGBoost (proposed ML approach)	0.66588	0.79
EfficientNet-B0 (proposed DL approach)	0.67427	0.62

Table 4.1: Comparative results

4.4.1 Machine Learning Approach Comparative Analysis

An interpretation of the degraded accuracy achieved by the Naïve Bayes Classifier study in section 2.5 might be due to weaknesses such as downsampling the whole slide images and the staining technique used in its dataset–hematoxylin and eosin (H&E) staining. On the other hand, this introduced a solution that exploits the advantage in Kaggle's dataset, which is the superior staining technique MSB. As mentioned in subsection 1.2.3, MSB staining allows for distinctive color separation assisting in the extraction of distinctive features.

Another reason for the higher baseline score achieved might be due to the segmentation process performed on high-resolution images. As previously explained in subsection 3.2.1, the images in the dataset are not downsampled to rely on them as low-resolution images, then using them for blood clot component segmentation as was observed in the study that used the Naïve Bayes classifier.

Despite not utilizing any deep learning methods, the suggested solution was evaluated through Kaggle's competition[15] private dataset, and it came in 5th place on the leaderboard. Among the top rankings, this study was the only one that used a traditional machine learning methodology, demonstrating its superiority over alternative deep learning methods.

4.4.2 Deep Learning Approach Comparative Analysis

In the paper discussed in section 2.4, the approach involves tiling each WSI where tiles with predominantly white backgrounds or lacking significant information are excluded. Tiles meeting specific mean and standard deviation thresholds are filtered and included in the training set. Additionally, color normalization is applied to standardize color intensity and shading. Data augmentation techniques such as rotation, brightness adjustment, zooming, and flipping are employed to enhance the dataset. On the other hand, our implementation adopts a different strategy. We perform a granular analysis of pixel variations within images at the block level, identifying regions with notable deviations from their surroundings. These identified blocks meeting defined thresholds are grouped into crops, ensuring non-overlapping selections through implemented measures. Our model is specifically designed to read images, ascertain crop positions, and select unique non-overlapping crops based on pre-defined criteria.

For model architecture, the paper utilizes a stacked ensemble of EfficientNet-B0, VGG19, and ResNet-152 backbones, fine-tuned from pre-training on ImageNet. A batch normalization layer is added after flattening and concatenating feature extraction outputs, with the final layer comprising two neurons using softmax activation for class probabilities. Contrastingly, during our efforts, our model employs transfer learning with models such as efficientnet_b0, resnet18, and regnet_x_1_6gf, freezing all layers except the last one, proving to be the better DL solution based on the WMCLL evaluation criteria. Compared to the competition scoreboard our deep learning approach achieved 12th place.

Chapter V: Conclusion and Utilities

5.1 Conclusion

The project proposes two solutions, ML and DL, and proves ML superiority, in the case of similar medical challenges, over the use of most DL approaches. This was done by using the XGBoost[26] model and training it on numerically extracted features characterizing blood clot textures, bypassing the need for direct image-based training included in DL.

Classical methods are more effective with limited datasets, like in this case, and obtaining large datasets can be difficult when dealing with limited resources. On the other hand, DL models are more resource-intensive and less interpretable than classical models. A limitation of classical machine learning might be its need for domain expertise, however, this was overcome by the consultation of an expert pathologist to help identify thrombus components, understand digital pathology images, and extract discriminative features that aided the study.

An ischemic stroke is a medical emergency that needs to be treated right away to reduce brain damage and other stroke complications. Thus, the project offers an ischemic stroke blood clot origin identification system that gives doctors a dependable and quick method of diagnosing the type of ischemic stroke and determining the best treatment approach. This type of system has the potential to save lives, decrease healthcare costs, and improve overall care quality.

5.2 Utilities

To further enhance the usability of our model, we developed both a desktop and a website application. The desktop application allows users to classify blood clot stroke origins directly from their computer, featuring a user-friendly interface for uploading images and receiving real-time classification results. The website provides an accessible platform for users to perform classifications online without the need for local installations. Together, these utilities aim to assist medical professionals in making quicker and more accurate diagnoses by leveraging the power of our advanced classification model.

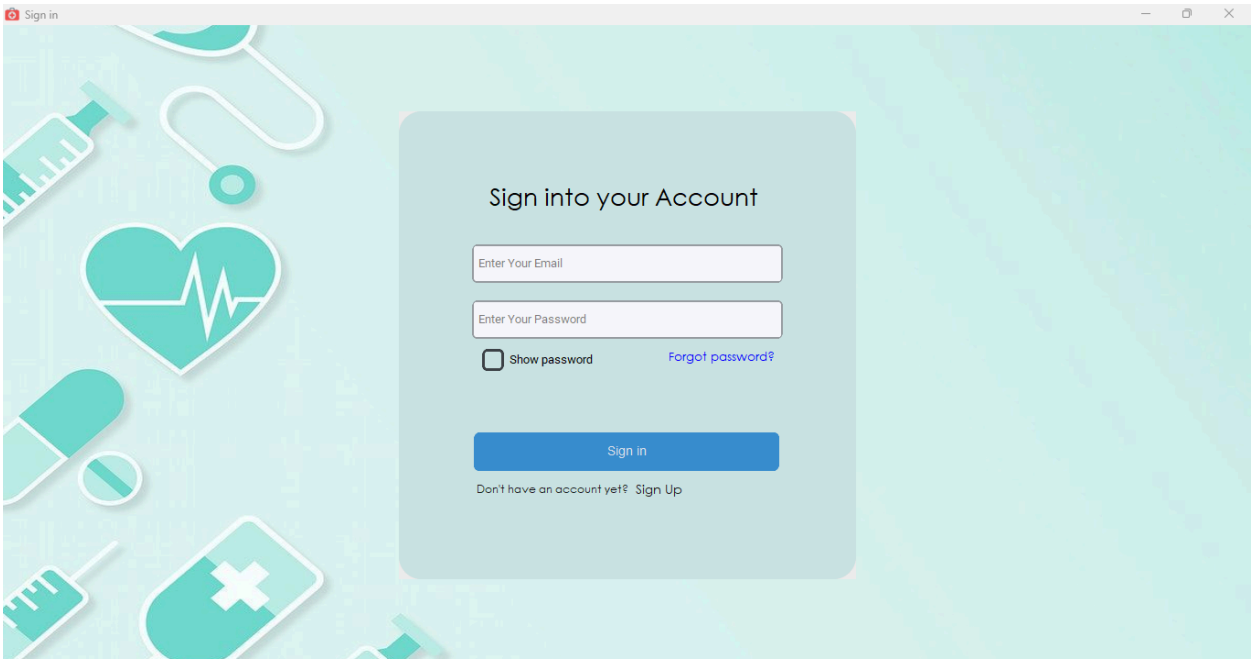
5.2.1 Desktop Application

The application is designed as a standalone, local desktop application, installed directly on hospital PC devices. It operates independently without the need for external servers but with minimal internet connectivity, ensuring data privacy and accessibility within the hospital's secure network.

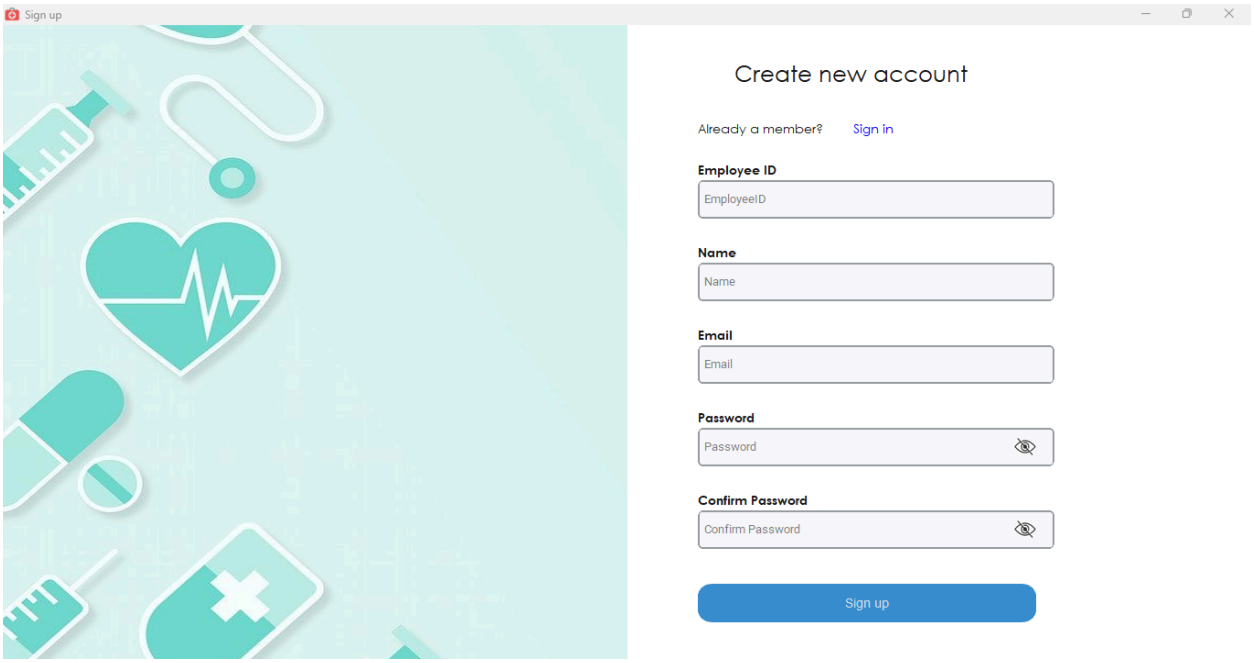
Python serves as the cornerstone of the Ischemic Stroke Diagnosis Application, offering a versatile and robust programming language perfectly suited for developing desktop applications. The user interface was built using CustomTkinter alongside the standard Tkinter library in Python. SQLite was employed as the local database management system for storing patient records securely within the application.

The application contains top-level safety measures, where doctors need email verification before signing in and password verification for every step within the application to ensure patient data confidentiality. Overall, the application was a success allowing hospitals to locally utilize complex models easily.

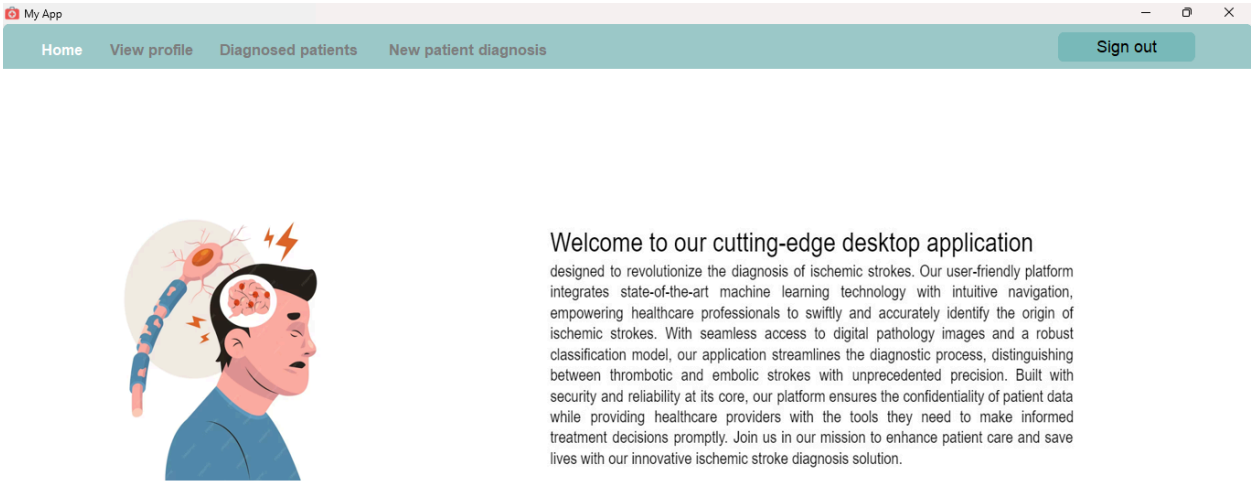
➢ Sample Screenshots:



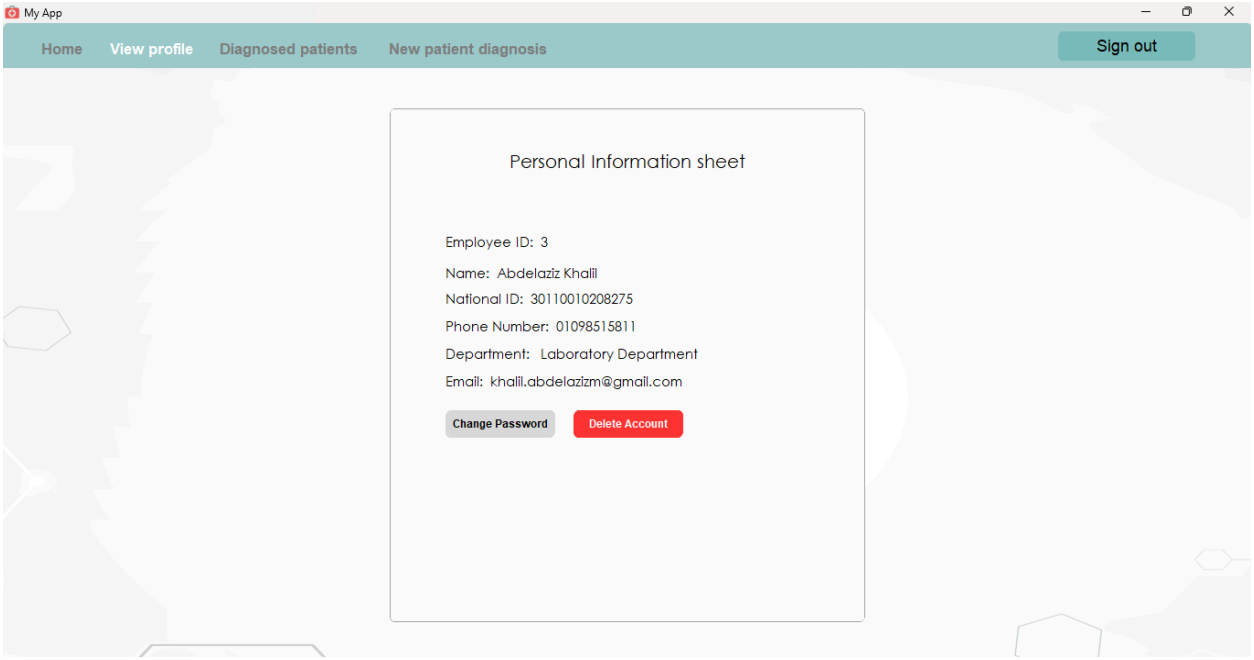
Sign In Page



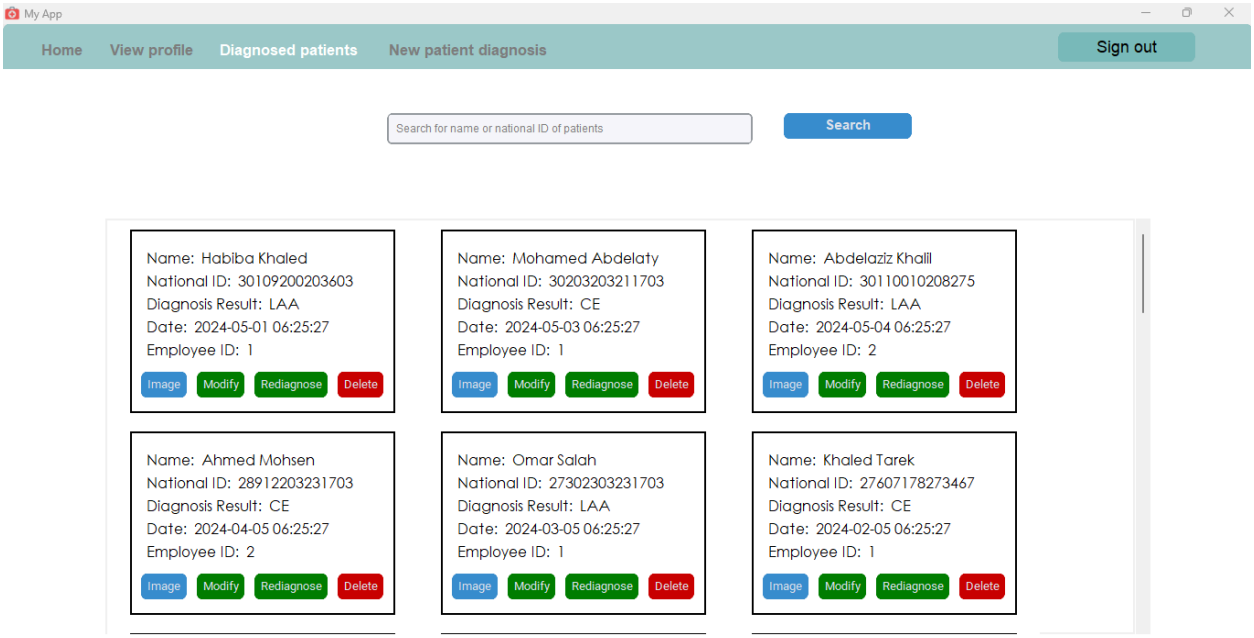
Sign Up Page



Home Page



Profile Page



Diagnosed Patients Page

New Patient Diagnosis Page

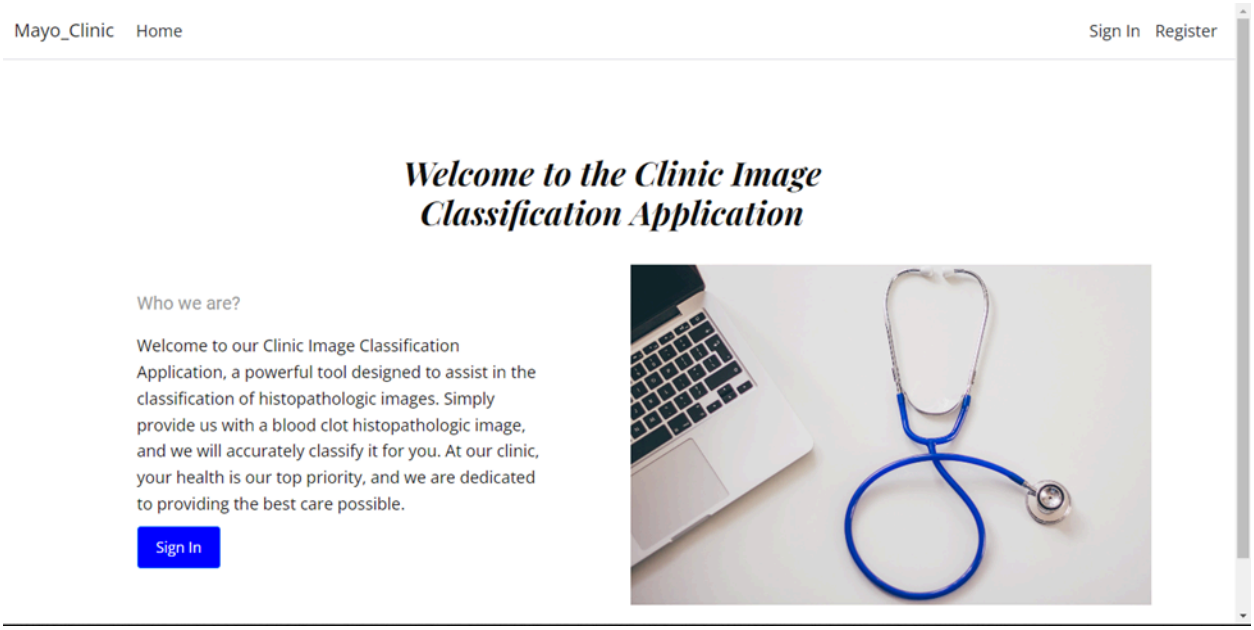
5.2.2 Web Application

We have developed a website using ASP .NET Core designed to help doctors efficiently manage patient information, including uploading blood clot images for classification using our advanced model. Doctors can sign up to access the system, allowing them to add and edit patient details, classify images, and request explanations using XAI. Unauthorized users are restricted from accessing these functionalities.

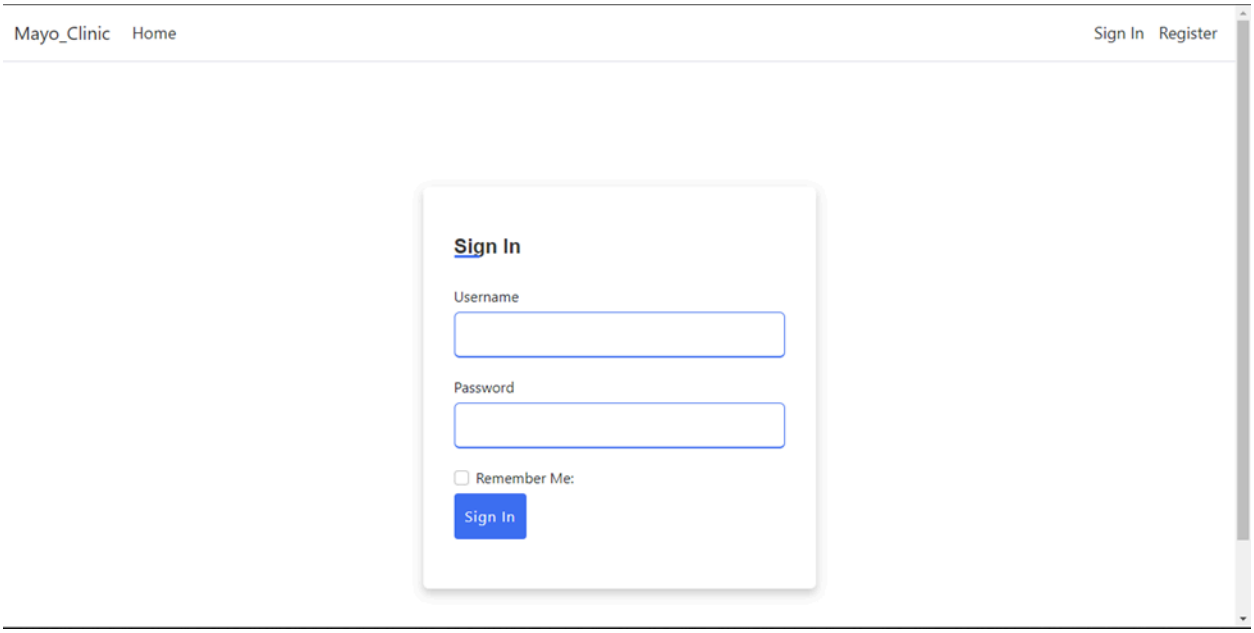
The welcome page invites doctors to sign up or log in to access the system. The clean and professional design ensures an intuitive user experience, guiding users through the authentication process seamlessly. The login screen provides a straightforward interface for doctors to enter their credentials securely.

The patient management dashboard is the central hub for doctors, allowing them to view and manage patient records efficiently. Each patient entry includes details such as name, age, medical history, and an uploaded blood clot image. The interface is designed to be user-friendly, with clear navigation and quick access to patient data.

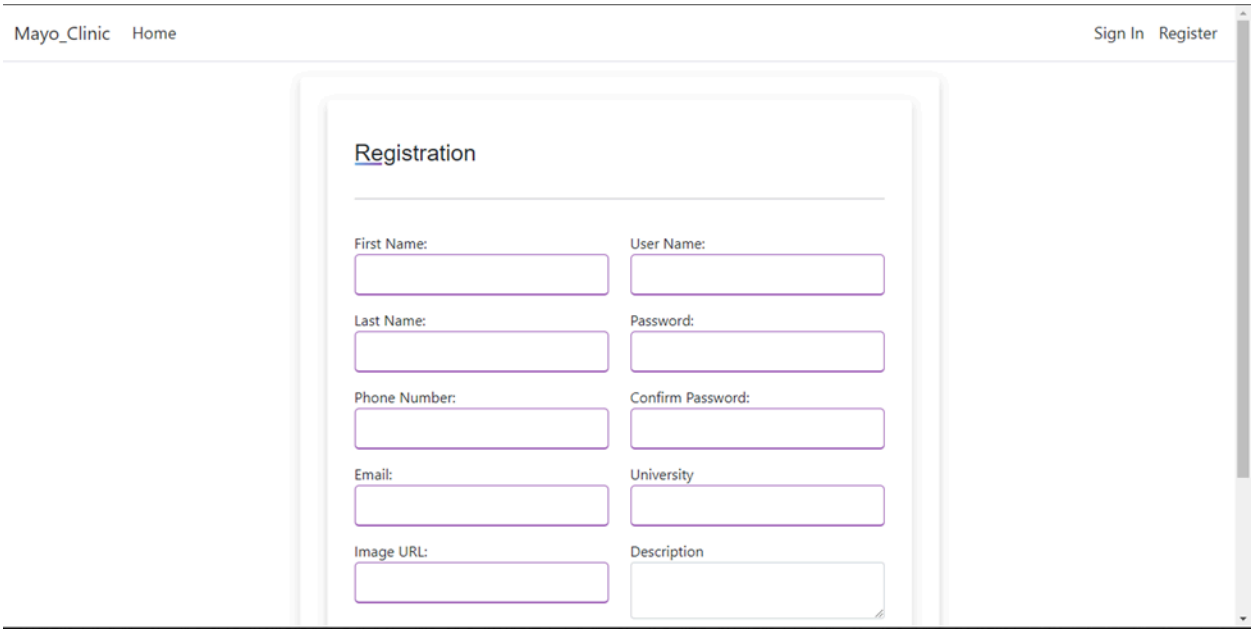
➢ Sample Screenshots:



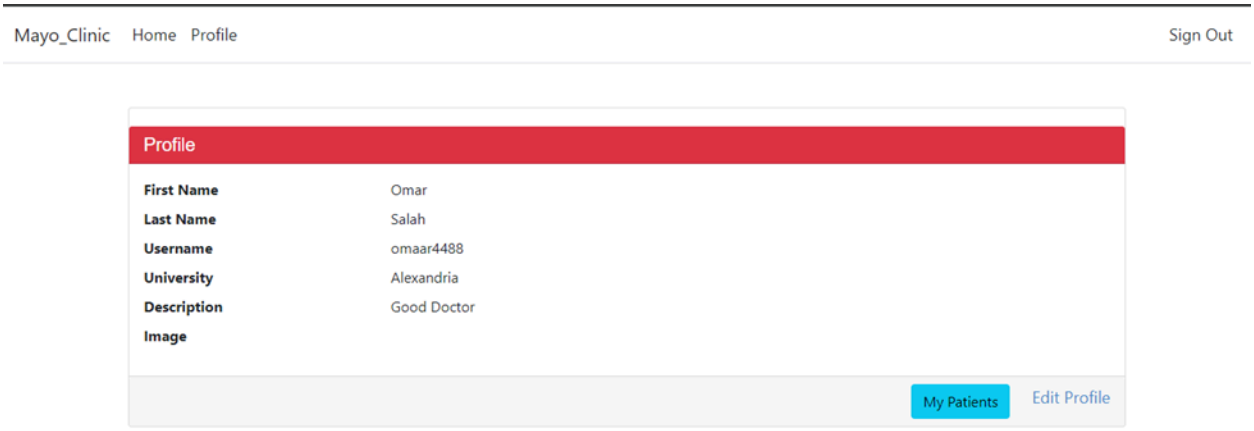
Welcome Page



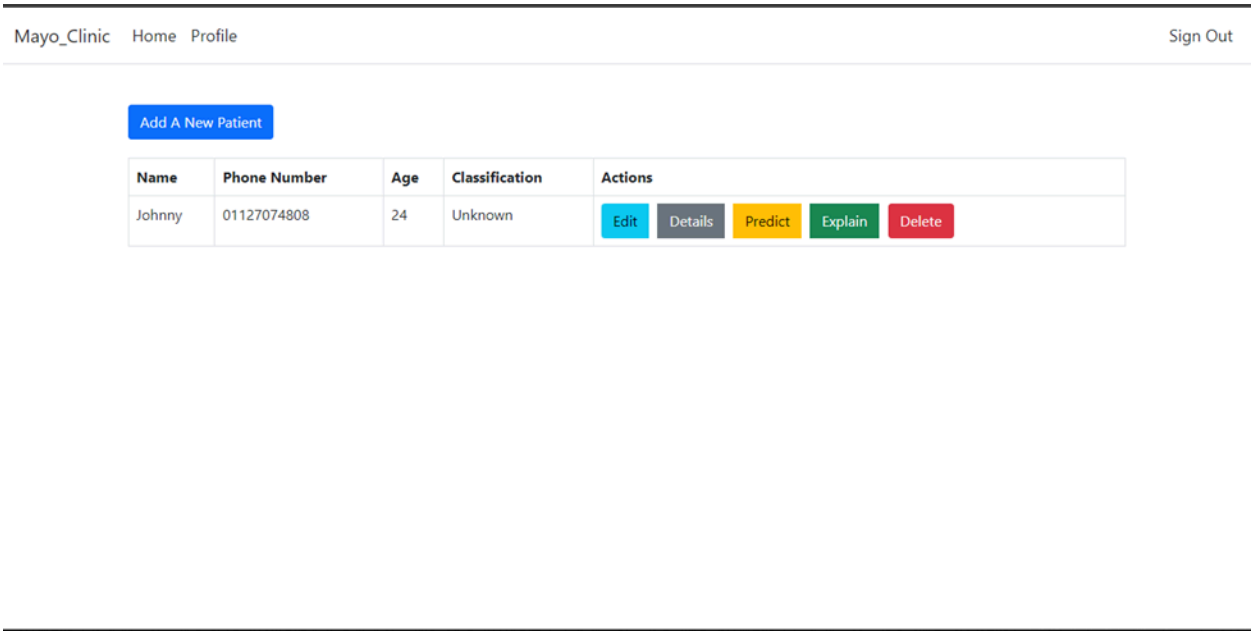
Sign In Page



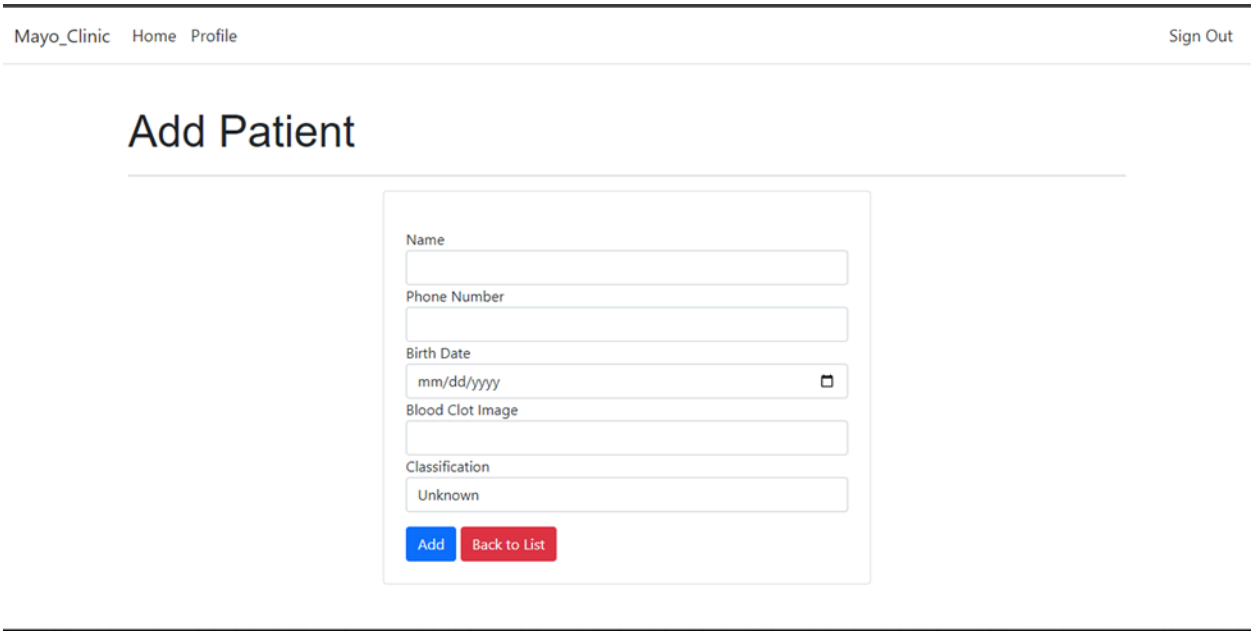
Sign Up Page



Profile Page



Patient Management Page



Add Patient Page

Bibliography

[1] Mayo Clinic. (2022). Stroke - Symptoms and Causes. Mayo Clinic. https://www.mayoclinic.org/diseases-conditions/stroke/symptoms-causes/syc-20350113#:~:text=An%20ischemic%20stroke%20occurs%20when

[2] Senna Staessens, Fitzgerald, S., Andersson, T., Frédéric Clarençon, Frederik Denorme, Gounis, M. J., Hacke, W., Liebeskind, D. S., István Szikora, Acgm van Es, Waleed Brinjikji, Doyle, K. M., and De, S. F. (2019). Histological Stroke Clot Analysis After Thrombectomy: Technical Aspects and Recommendations. International Journal of Stroke, 15(5), 467–476. https://doi.org/10.1177/1747493019884527

[3] Jauch, E. C., Saver, J. L., Adams, H. P., Bruno, A., Connors, J. J., Demaerschalk, B. M., Khatri, P., and McMullan, P. W. (2013). Guidelines for the Early Management of Patients With Acute Ischemic Stroke: A Guideline for Healthcare Professionals From the American Heart Association/American Stroke Association. Stroke, 44(3), 870–947. https://doi.org/10.1161/str.0b013e318284056a

[4] Farah Amna Othman, Asmaa' Mohd Satar, and Suat Cheng Tan. (2023). Roles of Sustainable Biomaterials in Biomedical Engineering for Ischemic Stroke Therapy. Sustainable Material for Biomedical Engineering Application, 415–433. https://doi.org/10.1007/978-981-99-2267-3_19

[5] Jolugbo, P., and Ariëns, R. A. S. (2021). Thrombus Composition and Efficacy of Thrombolysis and Thrombectomy in Acute Ischemic Stroke. Stroke, 52(3), 1131–1142. https://doi.org/10.1161/strokeaha.120.032810

[6] Chen, X., Wang, L., Jiang, M., Lin, L., Ba, Z., Tian, H., Li, G., Chen, L., Liu, Q., Hou, X., Wu, M., Liu, L., Ju, W., Zeng, W., and Zhou, Z. (2022). Leukocytes in Cerebral Thrombus Respond to Large-Vessel Occlusion in a Time-Dependent Manner and the Association of NETs With Collateral Flow. Frontiers in Immunology, 13. https://doi.org/10.3389/fimmu.2022.834562

[7] Brinjikji, W., Duffy, S., Burrows, A., Hacke, W., Liebeskind, D., Majoie, C. B. L. M., Dippel, D. W. J., Siddiqui, A. H., Khatri, P., Baxter, B., Nogeuira, R., Gounis, M., Jovin, T., and Kallmes, D. F. (2016). Correlation of Imaging and Histopathology of Thrombi in Acute Ischemic Stroke with Etiology and Outcome. Journal of NeuroInterventional Surgery, 9(6), 529–534. https://doi.org/10.1136/neurintsurg-2016-012391

[8] Azatyan, D. (2023). Image Classification of Stroke Blood Clot Origin using Deep Convolutional Neural Networks and Visual Transformers. ArXiv. https://doi.org/10.48550/arXiv.2305.16492

[9] Prasad, R., and Praveen Kumar Shukla. (2023). Identification of Ischemic Stroke Origin Using Machine Learning Techniques. Advances in Data-driven Computing and Intelligent Systems, 253–265. https://doi.org/10.1007/978-981-99-0981-0_20

[10] Krishnan, K. S., John, J. N. P., Gnanasekar, S., and Krishnan, K. S. (2024). Advancing Ischemic Stroke Diagnosis: A Novel Two-stage Approach for Blood Clot Origin Identification. https://arxiv.org/pdf/2304.13775.pdf

[11] Rao, M. V. S. R., Puligundla, S., Ekkala, N. S., and Chebrolu, S. (2023). Image Classification of Ischemic Stroke Blood Clot Origin using Stacked EfficientNet-B0, VGG19 and ResNet-152. In 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC). 2023 Third International Conference on Secure Cyber Computing and Communication (ICSCCC). IEEE. https://doi.org/10.1109/icsccc58608.2023.10176805

[12] Patel, T. R., Santo, B. A., Jenkins, T. D., Waqas, M., Monteiro, A., Baig, A. A., Levy, E. I., Davies, J. M., Snyder, K. V., Siddiqui, A. H., Kolega, J., Tomaszewski, J. E., and Tutino, V. M. (2023). Biologically Informed Clot Histomics Are Predictive of Acute Ischemic Stroke Etiology. Stroke: Vascular and Interventional Neurology, 3(2). https://doi.org/10.1161/svin.122.000536

[13] Fitzgerald, S., Mereuta, O. M., Doyle, K. M., Dai, D., Kadirvel, R., Kallmes, D. F., and Brinjikji, W. (2019). Correlation of imaging and histopathology of thrombi in acute ischemic stroke with etiology and outcome. Journal of Neurosurgical Sciences, 63(3), 292–300. https://doi.org/10.23736/s0390-5616.18.04629-5

[14] Mainali, S., Darsie, M. E., and Smetana, K. S. (2021). Machine Learning in Action: Stroke Diagnosis and Outcome Prediction. Frontiers in Neurology. https://doi.org/10.3389/fneur.2021.734345

[15] Mayo Clinic - STRIP AI. (2022). Image Classification of Stroke Blood Clot Origin. Kaggle. https://www.kaggle.com/competitions/mayo-clinic-strip-ai

[16] Martinez, K., and Cupitt, J. (2005). VIPS – A highly tuned image processing software architecture. In Proceedings of IEEE International Conference on Image Processing (Vol. 2, pp. 574-577). Genova.

[17] Cupitt, J., and Martinez, K. (1996). VIPS: An image processing system for large images. Proceedings of SPIE (Vol. 2663, pp. 19–28).

[18] Gonzalez, R. C., and Woods, R. E. (2008). Digital Image Processing (3rd ed., pp. 408-409). Upper Saddle River, NJ: Prentice-Hall.

[19] Beare, R., Lowekamp, B., and Yaniv, Z. (2018). Image segmentation, registration, and characterization in R with SimpleITK. Journal of Statistical Software, 86(8). Foundation for Open Access Statistics. https://doi.org/10.18637/jss.v086.i08

[20] Yaniv, Z., Lowekamp, B. C., Johnson, H. J., and Beare, R. (2017). SimpleITK image-analysis notebooks: A collaborative environment for education and reproducible research. Journal of Digital Imaging, 31(3), 290–303. https://doi.org/10.1007/s10278-017-0037-8

[21] Lowekamp, B. C., Chen, D. T., Ibáñez, L., and Blezek, D. (2013). The design of SimpleITK. Frontiers in Neuroinformatics, 7. Frontiers Media SA. https://doi.org/10.3389/fninf.2013.00045

[22] van Griethuysen, J. J. M., et al. (2017). Computational radiomics system to decode the radiographic phenotype. Cancer Research, 77(21), e104–e107. https://doi.org/10.1158/0008-5472.can-17-0339

[23] Mann, H. B., and Whitney, D. R. (1947). On a test of whether one of two random variables is stochastically larger than the other. Annals of Mathematical Statistics, 18, 50–60

[24] Wilcoxon, F. (1945). Individual comparisons by ranking methods. Biometrics Bulletin, 1, 80–83.

[25] Dahiru, T. (2008). P-value, a true test of statistical significance? A cautionary note. Annals of Ibadan Postgraduate Medicine, 6(1), 21-26. https://doi.org/10.4314/aipm.v6i1.64038

[26] Chen, T., and Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785–794). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939785

[27] Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). Classification and Regression Trees. Belmont, CA, USA: Wadsworth International Group.

[28] Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., and Kalinin, A. A. (2020). Albumentations: Fast and flexible image augmentations. Information, 11(2), 125. https://doi.org/10.3390/info11020125

[29] Tan, M., and Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. arXiv. https://doi.org/10.48550/ARXIV.1905.11946

[30] Ahmed, T., and Sabab, N. H. N. (2021). Classification and understanding of cloud structures via satellite images with EfficientUNet. Wiley. https://doi.org/10.1002/essoar.10507423.1

[31] Harrison, P. W., Amode, M. R., Austine-Orimoloye, O., Azov, A. G., Barba, M., Barnes, I., Becker, A., Bennett, R., Berry, A., Bhai, J., Bhurji, S. K., Boddu, S., Branco Lins, P. R., Brooks, L., Ramaraju, S. B., Campbell, L. I., Martinez, M. C., Charkhchi, M., Chougule, K., … Yates, A. D. (2023). Ensembl 2024. In Nucleic Acids Research (Vol. 52, Issue D1, pp. D891–D899). Oxford University Press (OUP). https://doi.org/10.1093/nar/gkad1049

[32] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep Residual Learning for Image Recognition. arXiv. https://doi.org/10.48550/ARXIV.1512.03385

[33] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2014). Going Deeper with Convolutions. arXiv. https://doi.org/10.48550/ARXIV.1409.4842

[34] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2016). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. arXiv. https://doi.org/10.48550/ARXIV.1610.02391

[35] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "Why Should I Trust You?": Explaining the Predictions of Any Classifier. arXiv. https://doi.org/10.48550/ARXIV.1602.04938

[36] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., and Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. arXiv. https://doi.org/10.48550/ARXIV.2103.14030

[37] Krishnan, K. S. (2021). Swin Transformer is all you need for Computer Vision. Medium. https://koushik0901.medium.com/swin-transformer-is-all-you-need-for-computer-vision-f2763a4f3fed

[38] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. arXiv. https://doi.org/10.48550/ARXIV.1506.02640

[39] Girshick, R. (2015). Fast R-CNN. In 2015 IEEE International Conference on Computer Vision (ICCV). 2015 IEEE International Conference on Computer Vision (ICCV). IEEE. https://doi.org/10.1109/iccv.2015.169

[40] Ananth, S. (2019) Fast R-CNN for object detection, Medium. Available at: https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022

[41] Kingma, D. P., and Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv. https://doi.org/10.48550/ARXIV.1412.6980