Introduction

The purpose of this report is to present AMI-Net, a Deep Learning model to detect brain regions affected by a stroke given medical imaging (mCTA) scans. We will outline AMI-Net's development, results on real data, limitations, future work and current capability to meet functional requirements outlined by our client, Andromeda Medical Imaging (AMI) Inc.

Background:

Strokes: Strokes are a leading cause of disability and the 3rd most common cause of death amongst developed countries [1]. During a stroke, a blood vessel that delivers oxygen and nutrients to the brain is blocked by a blood clot, causing parts of the patient's brain to be damaged or die. This results in a loss of motor and neural function, along with premature aging.
Importance of Imaging: Neurologists rely on medical imaging to find tissues affected by stroke. During diagnosis, these tissues are categorized as either **penumbral** (affected but not yet dead) or **core** (already dead). The more penumbral (saveable) tissue there is, the more likely it is that a surgical procedure would be worth performing at the risk of surrounding living tissue.
Existing Solutions: There are four common medical imaging techniques [1]. (1) Manual Diagnosis from CT imaging, while standard, is challenging for general physicians to interpret. (2) Contrast enhanced single phase CTA (sCTA) can show vessel blockages, but it cannot determine if the brain tissue supplied by that vessel is still alive and functioning (if it is viable) or if it has already been damaged due to a lack of blood supply. (3) Multiphase CT angiography (mCTA) lacks tissue viability and is hard to interpret. (4) CT perfusion (CTP) assesses tissue viability, but is time-consuming, resource-intensive, and costly.
Motivation: Large medical centers can use expert stroke neurologists and advanced imaging like CTP. We aim to identify penumbral tissue volumes using a fast, automated method that is more accessible to rural areas, which may lack advanced equipment and sufficient expertise.

Problem Statement:

Our client, Andromeda Medical Imaging (AMI) Inc, has taken their first step towards addressing this issue with the Simple Perfusion Reconstruction Algorithm (SPIRAL) [1]. This provides perfusion imaging from an mCTA scan, highlighting affected tissue areas (see Figure 1). However, the resulting image is still too coarse-grained for precise clot detection. We propose **AMI-Net**, a Deep Learning (DL) model to automate precise detection of affected brain regions from the raw mCTA scans or generated SPIRAL images. Our final solution must meet the following requirements, motivated by our client meetings:

Table 1: Our requirements, objectives and constraints, as informed by our client meeting.

Functional Requirements	(1) Must accept input data in the form of raw mCTA scans or generated SPIRAL images (2) Must determine zero or more of the 15 categorical sites as regions affected by clot.
Objectives	(1) Should identify affected brain regions more accurately than AMI's SPIRAL technique. (2) Should identify the stroke site in a short amount of time. (3) Should have a low memory footprint.
Constraints	(1) Must run in under 5 minutes for a single patient. (2) Must be able to run on a single CPU (use less than 8GB RAM)

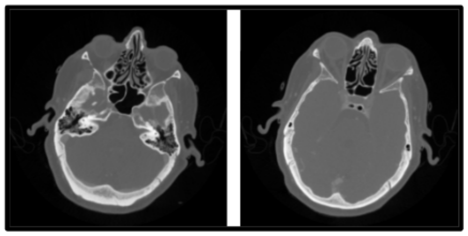
Data

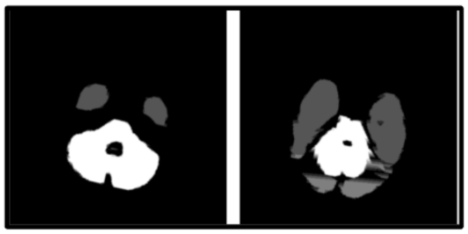
Overview of Dataset: Our dataset consists of 127 folders, one for each patient. In each folder, there are 3 raw mCTA scans (2 of which are image-aligned and thus usable as input to ML models), a clot location file, a SPIRAL map, and an ATLASMASK file. Each of these files were in .NII (nifty) format, were 512x512 pixels in dimension, and varied in depth (usually around 200 layers) across patients. We only consider the first image-aligned mCTA scan.

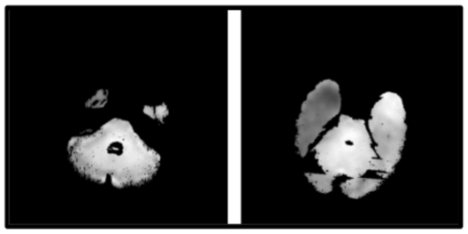
<u>Key features:</u> The **raw mCTA scans** are taken at the hospital and consist of stacks of 2D images of the brain at different depths. The **clot location file** indicates the pixel locations of the clot boundary. Each number in the 3d array is either 0, 1, or 2. For our purposes, 0 indicates pixels not affected, and 1, 2 are both treated as affected pixels. The **ATLASMASK** matrix represents the coverage of the different regions of the brain, and is patient-specific. Logically, each number is between 0 and 14, inclusive. The **SPIRAL map** is similar in structure to raw CTA scans. Each pixel value is within the range [0, 1], and represents the probability that it corresponds to damaged tissue (0 = healthy tissue, 1 = damaged tissue). The 4 files are visualized in Figure 1. Further visualization of each file for a single patient is conducted in Appendix A1.

<u>Label generation:</u> Each label is a 15-element binary vector, each index corresponding to one of 15 ATLAS brain segments. The value at each index is either 0 or 1, representing whether a clot is present in that region. To construct the label for a given image, we first identified where the clot was by looking at the locations of non-zero values in the clot file, and then identified what ATLAS map regions those locations mapped to in the corresponding ATLAS file.

<u>Data cleaning methods:</u> The input to our ML model - either raw CTA scans or SPIRAL maps were already aligned and similar in distribution (raw scan pixels between -1000 and 2000 approximately, SPIRAL map pixels between 0 and 1). To convert each patient's data to a format we could input to a model running on a single CPU, we decided to split the data in each file across the depth axis to obtain a set of ~257 individual 2D slices. This allowed us to avoid otherwise imminent CUDA Out Of Memory errors. To make training efficient, we further save each of these frames as 2D tensors (.pt file) offline, allowing us to load the data directly from storage rather than converting them from .nii to .pt online (i.e. during training). Notably, the ATLAS mask is aligned with the clot file; thus, label generation is straightforward, as we simply extract the union of brain regions that spatially (pixel-wise) correspond to clot locations. Finally, our data exploration showed that clot files are very sparse (only a few frames per patient actually have a clot pixel). For our deep learning approaches, we address this class imbalance in our objective function by weighting the loss term of clot pixels more than the loss term of non-clot pixels. This is crucial as the classifier can otherwise just predict no brain regions are affected for each frame, and get a high accuracy. Details are discussed in our Methodology section.


(a) Raw CTA scan samples


(b) ATLAS Map of Brain Regions [0-14]


(c) SPIRAL output - each pixel in [0,1]

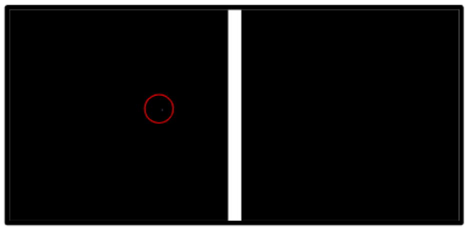

(d) Blood Clot - each pixel in [0,2]

Figure 1: Visualizations of the raw CTA scan, ATLAS map, SPIRAL output, and Blood Clot data for 2 frames. The frame on the right does not have any affected regions, but the left frame does.

Methods:

The clot location prediction problem is a ***multilabel classification*** task. Recent review papers [2,3] have highlighted the applicability of both traditional machine learning (ML) approaches (i.e. SVM, Random Forest), and deep learning methods like Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs) to brain stroke detection from CT scans. Notably, findings from [2] emphasize the efficiency of deep learning techniques in aiding clinicians due to their ability to automatically extract relevant features. In contrast, traditional ML methods often need the expertise of domain experts to hand-craft features, a process that is not only time consuming but also prone to errors [2]. Building on recent literature that have successfully employed CNN-based approaches for ischemic (blood-clot) stroke detection in CT and CTP scans [2], we wish to assess the efficacy of CNNs in this domain. Moreover, the works of [4] and [5] highlight the widespread adoption of ViT in medical imaging-based classification tasks. Thus, we wish to evaluate state-of-the-art (SOTA) CNN and ViT architectures, namely MobileNet and EfficientFormer, respectively. Architecture for these models are visualized in Appendix A3. We will compare these approaches to a baseline CNN and a non-neural approach.

The implementation of our deep learning based methods is done in PyTorch. In order to load the .NII format files into tensors, we utilized the Nibabel Python package [6]. Model training was done on a NVIDIA 3090 GPU with 24 GB VRAM, which allows us to use large batch sizes and parallelize model training with the RayTune hyperparameter tuning library.

In our analysis of each method, we will report the precision, recall, accuracy and F1 score after hyperparameter tuning. To determine the best hyperparameter setting for each method, as well as the best method overall, we will use the ***F1 score***. The F1 score is the harmonic mean of precision and recall; it suits our task as it aims to minimize both false positives (avoiding unnecessary surgery) and false negatives (not operating brain regions that require surgery).

1. SPIRAL Baseline, a Non-Neural Approach

In order to evaluate the success of our ML-based approaches, we implement a baseline that predicts clot locations directly from SPIRAL outputs provided by AMI. In this approach, for every frame, we identify *affected pixels as those with values greater than some specified threshold* (recall that larger SPIRAL values mean higher probability of damaged tissue). From the blood clot pixels, we can extract the brain regions (our final prediction) using the reference ATLASMASK. This association could be done easily because both the blood clot files and ATLASMASK had the same dimensions and are aligned. Thus, if a pixel corresponding to an affected brain region was located at index (x,y) in the SPIRAL map, the value at index (x,y) of the ATLASMASK would represent the brain region that this pixel belonged to. This method is visualized in Figure 2.

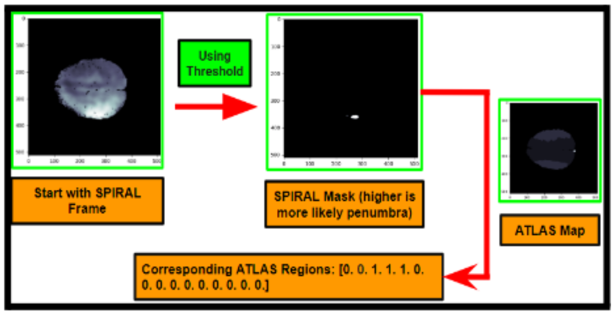

Start with SPIRAL Frame
Using Threshold
SPIRAL Mask (higher is more likely penumbra)
ATLAS Map
Corresponding ATLAS Regions: [0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

Figure 2: Pixels that represent blood clot regions are directly identified from the SPIRAL map

Hyperparameter Tuning: We split data into distinct train and test sets (80-20 split). We determine the optimal threshold by assessing performance on the train set, and evaluate performance by reporting metrics on the test set using this threshold.

Hyperparameter tuning results:

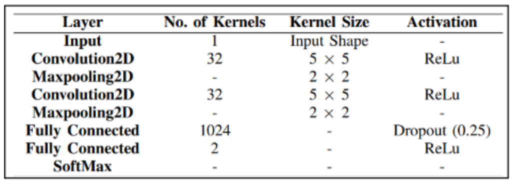
Table 2: Precision, Recall, F1 Score and Accuracy on various thresholds on the train set

Threshold	Precision	Recall	Accuracy	F1 Score
0.0	0.1577	0.7089	0.2139	0.2579
0.7	0.1659	0.6507	0.3023	0.2644
0.8	0.1879	0.5445	0.4587	0.2794
0.9	0.3344	0.3596	0.7389	0.3465
0.95	0.5443	0.1473	0.8119	0.2318

We found that the baseline had the best performance at a threshold value of 0.9, obtaining an F1 Score of 0.347. The benefit of this method is that it is guaranteed to satisfy our hardware constraints (< 8 GB RAM) and can perform inference on all slices for a single patient in the least amount of time. The F1 score on the test set is 0.3358, and full results are in Appendix A2a.

2. CNN Baseline

We implement a 4-layer CNN architecture proposed in literature regarding COVID-19 prediction from CT scans [7], shown in Figure 3. We experiment with using either raw CT scans or SPIRAL images as input. The model is applied to each frame from the depth dimension of the SPIRAL input. These 2D slices are 512 by 512 pixels by default but as discussed in the hyperparameter tuning section, we experiment with different input resizing dimensions. The softmax output has a dimension of 15 corresponding to each of the 15 brain region classes.



Layer	No. of Kernels	Kernel Size	Activation
Input	1	Input Shape	-
Convolution2D	32	5×5	ReLu
Maxpooling2D	-	2×2	-
Convolution2D	32	5×5	ReLu
Maxpooling2D	-	2×2	-
Fully Connected	1024	-	Dropout (0.25)
Fully Connected	2	-	ReLu
SoftMax	-	-	-

Figure 3: Baseline Convolution Neural Network Architecture

Hyperparameter Tuning: For all trials, we kept the train-val-test ratio at 64:16:20, the optimizer as Adam, and the loss function as binary cross entropy for multi-label classification. We kept the epochs for each trial at 40, as empirically we found this is enough to reach convergence for all tested hyperparameter settings. The tunable hyperparameters include the activation function (ReLU, sigmoid, tanh), dropout ratio (0.1 to 0.4), embedding size after convolution layers (128, 256, 512), pooling size (2x2, 4x4), class weights of clot vs non-clot regions (10-90 for clot regions), learning rate (0.0005 to 0.01), number of extra layers in the fully connected region (0 to 2), and the input image size (128x128, 256x256, 512x512). The batch size was kept at 32 for all experiments.

Explanation of class weights: This hyperparameter helps us address class imbalance in our dataset by adjusting the importance of each class in our loss function. Higher weights are assigned to the underrepresented class (in our case, this is the positive class as the proportion of pixels that correspond to a blood clot in any given image is tiny compared to non-clot pixels), so that the model can pay more attention to it and make better predictions for it.

Hyperparameter tuning results: The results using both SPIRAL and raw CTA scans are shown below. The optimal hyperparameters for best F1 score using each data input type are the values in the columns with green cells. After heavy tuning, using raw CTA scans as input gives a higher F1 score, indicating that using generated SPIRAL maps may result in a loss of information for downstream tasks. Full results are in Appendix A2b and A2c.

Table 3: Sample hyperparameter tuning runs using SPIRAL and raw CTA scans

Hyperparameter	Sample SPIRAL HParam Settings		Sample CTA HParam Settings	
pool_dim	4	4	4	4
feat_dim	512	512	512	512

activation	tanh	relu	tanh	tanh
learning rate	0.0013	0.00061	0.0011	0.0009
dropout_prob	0.32	0.39	0.25	0.27
pos_weight	17.72	15.26	12.91	15.65
extra_layers	0	2	2	2
resize_dims	128x128	128x128	256x256	128x128
iterations	8	20	20	20
total time (s)	79.80	106.68	325.90	132.72
Val_F1	0.33	0.60	0.62	0.67
Val_Precision	0.20	0.46	0.51	0.51
Val_Recall	0.91	0.86	0.78	0.95
Val_Accuracy	0.96	0.99	0.99	0.99

Further Training: After selecting the hyperparameters that give the highest F1 score on our validation set, we retrained the model for more epochs (40). The accuracy, precision, recall and F1 score over time are shown in Figure 4. These graphs match our expectations of multilabel classification of tiny blood clots - we achieve high recall early on as each frame has very few affected brain regions. We also get high accuracy as most regions will be unaffected within a single frame; so we can get a good accuracy just by outputting no region. Precision and F1 score, which capture how well we do in outputting affected regions take longer to converge. This is because the model must actually learn to minimize a possibly large amount of false positives while keeping the usually zero or one affected region.

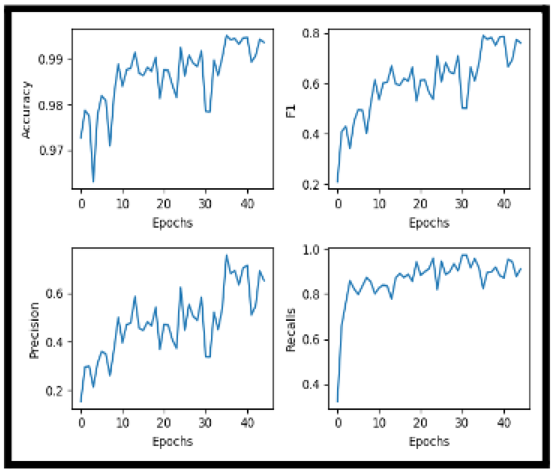


Figure 4: Evaluation metrics for CNN baseline with optimal hyperparameters throughout training

3. More Complex Deep Learning Architectures

We now aim to investigate more complex architectures, specifically MobileNetV2 and EfficientFormer. Since these models have a large number of parameters, training on our data alone may result in overfitting; we thus implement a transfer learning approach, whereby we take a frozen pretrained model (trained on ImageNet classification) and replace the final layer with a newly initialized one. We train the model for 5 epochs and then unfreeze *all* model weights. Subsequently, we fine-tune the model with a lower learning rate than before and train for 40 more epochs.

3a. MobileNetV2

We wish to investigate how much performance boost a more complex CNN architecture gives. The MobileNetV2 architecture is a CNN-based model that is SOTA amongst lightweight image classification models [8]. As such, it is commonly used in edge inference settings such as mobile devices, which makes it a suitable candidate for our exploration due to our compute constraints.

Hyperparameter Tuning: There is less architectural hyperparameter tuning involved as we are using a pretrained model. Following our earlier results, we experiment with CTA inputs only.

Input images are resized to 224 x 224 as that is the default size for this model. We tune the learning rate (0.0005 to 0.01) and class weights (10-90).

Hyperparameter tuning results:
The MobileNetV2 performs well in all metrics, with low learning rates and a positive class weight of ~15. The optimal hyperparameters are indicated by the green row.

Table 4: Sample hyperparameter tuning runs using raw CTA scans

lr (*10e-4)	Pos class weight	Epochs	Total time (s)	F1 on val set	Precision on val set	Recall on val set	Accuracy on val set
6.74	12.72	20	589.63	0.7700	0.6504	0.9435	0.9936
7.01	15.59	20	580.24	0.7899	0.6790	0.9555	0.9946
10.17	14.16	20	583.94	0.7315	0.6179	0.8963	0.9439
32.74	16.45	20	586.57	0.6930	0.5854	0.8492	0.9439
33.25	17.02	20	596.54	0.5390	0.4553	0.6605	0.8942

Further Training
As in the CNN baseline, we train the optimal model for 20 more epochs. The general patterns remain the same; we get very good accuracy and recall after just a few epochs, but precision and F1 score take more epochs to converge and don't converge to a value near 1. We also note a jump in performance as soon as fine-tuning begins at epoch 5. The final F1 and precision are better than their respective scores on the CNN baseline, which suggests that the vast knowledge stored in the pretrained model is useful for learning general features for image classification, and steering the model by fine tuning it to our dataset improves results for our specific task of blood clot detection.

3b. EfficientFormerV2
We now move from CNN to vision transformers. Vision transformers have recently emerged as a very strong computer vision model that makes use of attention to learn complex features across the entirety of images without being limited to a small receptive field in its earlier layers [9]. However, they tend to be data hungry since they do not have inductive biases like CNNs in the form of translation invariance and locality. Thus, we leverage pretrained weights of the EfficientFormerV2 model [10], which is a lightweight vision transformer that achieves state of the art performance on the ImageNet image classification benchmark.

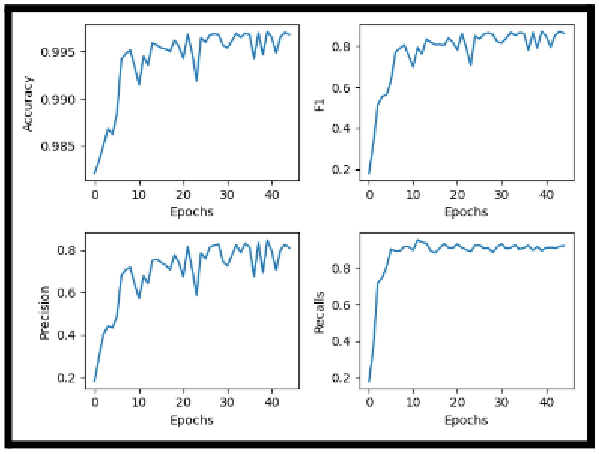


Figure 5: Evaluation metrics for MobileNetV2 fine tuning using optimal hyperparameters during training

Hyperparameter Tuning & Results: The hyperparameter tuning experiments follow the same procedure as outlined for MobileNetV2, whereby learning rate and positive class weight are tuned. We use CTA frames as input, which are resized to 224 x 224 pixels as required by default for the pretrained model. The best performing hyperparameters are shown below.

Table 5: Sample hyperparameter tuning runs using raw CTA scans

lr (*10e-3)	1.3	0.2	1.1	2.8
pos_weight	13.54	12.28	16.37	13.67
epochs	20	20	20	20
total time (s)	862.51	859.57	849.99	865.27
Val_F1	0.60	0.58	0.48	0.36
Val_Precision	0.44	0.42	0.35	0.26
Val_Recall	0.95	0.91	0.76	0.57
Val_Accuracy	0.99	0.95	0.96	0.97

Further Training: As discussed previously, due to the nature of the task, accuracy and recall reach near perfect in a few epochs. However, there are more fluctuations for the first few epochs- this can be credited to the large impact of initial gradient steps on the randomly initialized classification layer. The final validation F1 score is between the simple CNN and fine tuned MobileNetV2, suggesting that fine tuning large pretrained models is better than training small models from scratch, but also that frozen weights of a CNN are a better starting point than frozen weights of a ViT. Overall, the fluctuations in fine tuning an EfficientFormerV2 is more noticeable than fine tuning a CNN. Empirically, this suggests the CNN features are more suited to clot detection, as the classifier is more confident (less variance) in the weights of the final classification layer during training.

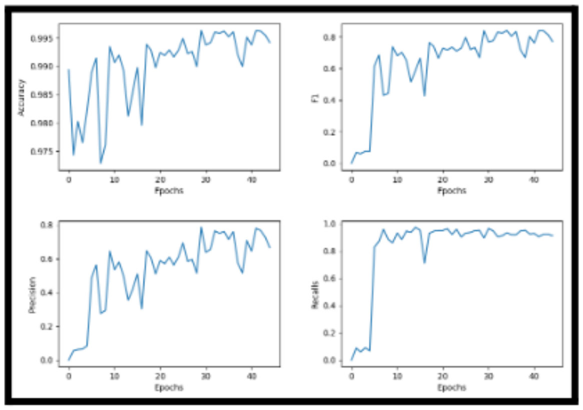


Figure 6: Evaluation metrics for EfficientFormerV2 fine tuning using optimal hyperparameters during training

Model Results and Comparisons:

Our final test results, obtained after training models with optimal hyperparameters on more epochs (40 total) are summarized in the table below.

Table 6: Comparison of the different methods on F1, Precision and Recall

Approach	Best Result
SPIRAL thresholding baseline	F1: 34%, Prec: 33%, Rec: 35%
Train a Simple CNN	F1: 79%, Prec: 73%, Rec: 86%
Fine-tuning MobileNetV2 as feature extractor	F1: 87%, Prec: 82%, Rec: 93%
Fine-tuning EfficientFormerV2 as feature extractor	F1: 84%, Prec: 77%, Rec: 92%

All 3 models outperform the SPIRAL map → brain regions baseline; we have (empirically) proven that neural methods are better than simple rule-based methods. The order of our model performances, from best to worst on F1 score is MobileNetV2 Fine-tuning, EfficientFormerV2 Fine-tuning, Simple CNN, SPIRAL Baseline. Notably, the best model on F1 is also best on Precision and Recall. This is expected as F1 is a holistic metric that ensures we both identify all regions that need surgery and not identify regions that do not require surgery. Note that we assume false positives and false negatives are equally as detrimental to the patient. If AMI decides one is more worrying, our metric can be adjusted by changing the alpha parameter of F1 to weight either precision or recall more.

It is natural to expect more complex architectures proven to work on medical imaging tasks will work better than simple CNNs, which will work better than simple rule-based methods. We speculate that MobileNetV2 outperforms EfficientFormerV2 because of a powerful pretrained base that captures intricate features that are useful for the downstream task of brain region classification. On the other hand, while EfficientFormerV2 also has a powerful vision transformer feature extractor, the fluctuations in the metric graphs suggest the extracted features are not immediately useful for our task. Thus, MobileNetV2 as a feature extractor captures the unique nuances present in brain CT scans more accurately, perhaps because a CNN-based model captures spatial relations in images and has inductive biases more suited to clot detection (for example, clot detection should be translation invariant), which are more powerful claims for our specific task than having an attention mechanism. Indeed, knowing that clots are concentrated in one region of the image makes CNNs a more intuitive choice due to convolution and pooling layers dealing with small windows of images. Moreover, clot detection in one region in the image is independent of clot detection in another region of the image, so attention is not as useful. Nonetheless, it is important to realize that the performance gap isn't too significant. In fact, while our hyperparameter search was very extensive, perhaps further searching the hypothesis space through hyperparameter tuning could push the results closer to one another. If our hypothesis that ViT features aren't as informative is valid, it may simply require more epochs after unfreezing all model layers.

Note that across all neural methods, the recall is much higher than the precision. This is because the output is the brain regions covered by a single frame; since the clots are very small, this is usually a small number of regions, if any. So, it is likely the model will predict the right region(s), achieving high recall, but will also predict incorrect regions, yielding low precision.

Discussion

<u>Usefulness:</u>

Our client, AMI, has two ultimate goals. The first goal is to output a categorical segmentation of the occlusion/clot with a relatively high F1 score. The second is to perform segmentation to show exactly which pixels belong to a clot. Due to the time restriction on the project, and on recommendation of our client, we have focused our efforts on delivering the best product to address the first goal alone.

AMI-Net meets the functional requirements of the project - to accept data as raw CTA or SPIRAL maps, and output zero or more categorical sites of occlusion, corresponding to regions of the brain served by different cranial blood vessels. All of our objectives are satisfied; as the union of affected brain regions for each frame defines which of the focused segments of blood vessels the stroke site is, and this is done while satisfying both constraints on time and memory. Regarding memory, AMI-Net uses very small compute power during inference - it is able to run inference on a single CPU (<8 GB RAM). Regarding time, our model performs inference for a single patient by looping through the ~257 frames and combining the results. The total time for this is well below 5 minutes; exact times are listed below. We are particularly pleased that the times are on the order of 10 seconds - since a stroke patient loses 7 million neurons each minute, it is vital to minimize this time.

Table 7: Time taken for each method to run inference on all 257 frames for a single patient

Model	CNN Baseline	EfficientFormerV2 Fine Tuning	MobileNetV2 Fine Tuning
Time for 1 Patient	5.92s	6.70s	8.41s

Furthermore, we achieve the primary objective of identifying affected regions of the brain to a standard acceptable by AMI-Net. The current rule-based approach that is available to AMI achieves a precision and recall of 33% and 35%, respectively. Our optimal method achieves 82% and 93% on the same metrics, respectively. Given that our client has instructed us to prioritize a high F1 score, we deem AMI-Net successful as it completely overwhelms the simple SPIRAL thresholding baseline (87% vs 34% on F1).

While we do not present AMI-Net as an end-to-end solution to determine which part of the brain to operate on in a stroke patient, we certainly deem it useful to help neurologists in rural areas (who reportedly don't have the same level of expertise as their counterparts in urban areas). The idea is that for a single patient, in under 10 seconds after obtaining CTA scans, the neurologist will have a set of recommended brain regions to investigate for clots. The neurologist then matches our output with the ATLAS map of the patient, and that way knows which regions of the CTA scan he/she should investigate further for signs of ischemic stroke. Given our precision and recall scores are both above 80%, and that most patients will only have one blood clot, our method will almost always give the neurologist an accurate general area to investigate.

Importantly, we present AMI-Net as an assistive tool; that way, we still place responsibility of misdiagnosis on the neurologist who must identify the best regions to operate in a limited amount of time. This is done to address the ethics behind medical diagnosis and treatment; a method that replaces a neurologist with an automated method would present uncertainties about who is at fault if the model performs poorly.

Limitations:

Our models are trained on a dataset of CT scans from 127 patients, which is not enough to learn a robust and deployable model. At least a few thousand patients' worth of data is required to develop models that can generalize and provide accurate predictions for new patients in a practical medical setting. It is essential to have data that originates from a large number of patients to capture the variety of patterns exhibited in ischemic stroke occurrences [1]. This prevents overfitting, by consequence helping deep neural networks generalize results to new patient data. Fortunately, our implementation can easily be reused by AMI for training on their proprietary 1000+ patient datasets with no modification.

Another limitation is that the area of the brain covered by our model output (predicted brain region(s)) is much larger than the size of a clot. This still leaves general physicians with much work to do in order to find the exact location of the blood clot. Nevertheless, this limitation can be addressed by developing a segmentation model that can output a region that is much closer to the actual size of the blood clot. Such a model will predict whether each pixel is part of the clot.

Effectiveness in comparison to similar products

Two competitors that have been identified include *Brainomix* [11] and *General Electric Health* [12]. Both companies have solutions that can identify the location of Large Vessel Occlusions (LVO) and hyperdense volumes which may indicate bleeding. Brainomix offers fully automated and standardized ASPECTS scores and can segment, outline and present its findings using graphical visualizations [13]. Additionally, General Electric Health has developed a solution, *ColorViz* [14], that elaborates the full set of images included in a CT stroke protocol (NeCT, mCTA) into one single color-coded map called ColorViz [14]. The vessels displayed on the map appear differently colored based on the arrival time of the contrast medium and on a per-person adaptive threshold technique [14, 15]. However, it must be noted that the SPIRAL scans provided by AMI provide information about both Large Vessel Occlusions (LVO) and Medium Vessel Occlusions [16]. Notably, Medium Vessel Occlusions account for 25%-40% of Acute Ischemic Stroke (AIS) [17]. The limitation in the scope of solutions offered by both

Brainomix and GE Health positions AMI-Net to have a significant advantage over both of its competitors. The additional precision provided via AMI-Net's segmentation would help save time, effort and cost in a clinical setting. Although it must be noted that AMI-Net does not provide all of these benefits currently due to the limitations discussed earlier, it is on the correct pathway to achieve these goals given more time and resources are put into its later development (specifically, the second ultimate goal of precise clot detection).

Implementation

As of yet, AMI has not proceeded with incorporating our proposed methods into their brain stroke patient diagnosis workflows. Nevertheless, we have taken steps to ensure a smooth deployment process. Our model inference scripts and trained model files are available in a private GitHub repository that has been shared with AMI employees. Inference on new patient data can be performed using aforementioned scripts on a system with at least 8 GB RAM and Python installed. Package dependency installation, needed for the scripts to import necessary modules, is also a one-time task that must be run on the machine prior to deployment. Since our model is only trained on 127 of AMI's proprietary patient dataset of 1000+ patients, we highly recommend training on the larger dataset to obtain a more robust model for deployment. Model training and data processing scripts are also provided so that AMI can perform larger training experiments with their proprietary datasets.

Conclusions and Future Directions

To conclude, we have successfully developed AMI-Net, a Deep Learning model that takes in input CT images and outputs a categorical site of occlusion as zero or more of 15 brain regions served by different cranial blood vessels. We explored 4 methods- a simple rule-based method, a simple 4-layer CNN architecture, fine tuning a pre-trained MobileNetV2 architecture, and fine-tuning a pre-trained EfficientFormerV2 architecture. With data from only 127 patients, and a train-val-test split of 64-16-20, we already achieve 82% and 93% on precision and recall, respectively, and only expect to increase once AMI uses all of its data (1000+ patients).

In general, fine tuning more complex architectures works better than training a simple architecture from scratch; the fine tuning methods achieve F1 scores of 87% and 84%, greater than 79%. Further, the best method, MobileNetV2, is significantly better than the non-neural rule-based system currently available to AMI (87% vs 34% on F1).

We are excited about future work that can improve results even further. The main next step is to *train on the full dataset* of 1000+ patients, so we can further steer weights of the pretrained model to fit our task of clot detection. We can also perform *more extensive hyperparameter tuning*, especially for the EfficientFormerV2 architecture, to determine whether EfficientFormer being worse than MobileNetV2 is truly due to limitations in the model architecture (attention vs convolution), or a limited search of the hyperparameter space.

Continuing, we wish to investigate models that can take *both raw CTA scans and SPIRAL maps as input*; this way we can investigate whether including both the pure/original data and the processed SPIRAL predictions can give better results than just training on raw scans. Since predictions on what regions to perform surgery on can be life-threatening, we could also *consider an ensemble approach*, using 3 methods to come up different region predictions, and performing majority voting to get the final predictions. This would give us more confidence in our results.

To improve the accuracy of our method while being able to run on a single CPU, we want to consider methods that evaluate affected regions by using a sliding window across the frames, rather than a frame-by-frame approach. This way, we will take more of the brain into consideration each time we run inference, giving us more data to predict off of and thus more confidence.

References

[1] C. d’Esterre, C. McDougall, and P. Barber, “SYSTEM AND METHOD FOR GENERATING PERFUSION FUNCTIONAL MAPS FROM TEMPORALLY RESOLVED HELICAL COMPUTED TOMOGRAPHIC IMAGES,” Mar. 24, 2022

[2] K. Ramamurthy, M. R, A. Johnson, and S. Anand, “Neuroimaging and deep learning for brain stroke detection - a review of recent advancements and future prospects,” Neuroimaging and deep learning for brain stroke detection - A review of recent advancements and future prospects - ScienceDirect, https://www.sciencedirect.com/science/article/pii/S0169260720315613?fr=RR-7 (accessed Dec. 20, 2023).

[3] M. S. Sirsat, E. Fermé, and J. Câmara, “Machine Learning for Brain Stroke: A Review,” Machine Learning for Brain Stroke: A Review - ScienceDirect, https://www.sciencedirect.com/science/article/abs/pii/S1052305720305802 (accessed Dec. 20, 2023).

[4] R. Raj, J. Mathew, S. K. Kannath, and J. Rajan, “StrokeViT with AutoML for brain stroke classification,” StrokeViT with AutoML for brain stroke classification - ScienceDirect, https://www.sciencedirect.com/science/article/pii/S095219762200762X (accessed Dec. 20, 2023).

[5] Y. Barhoumi and G. Rasool, “Scopeformer: N-CNN-VIT Hybrid Model for Intracranial Hemorrhage Classification,” arXiv.org, https://arxiv.org/abs/2107.04575 (accessed Dec. 20, 2023).

[6] Halchenko, Y., et al. NiBabel. Accessible: https://nipy.org/nibabel/

[7] Carvalho, E.D., et al. “Diagnosis of COVID-19 in CT image using CNN and XGBoost”. IEEE 2020.

[8] Sandler, M., et al. "MobileNetV2: Inverted Residuals and Linear Bottlenecks". CVPR 2018

[9] Dosovitskiy, A., et al. "An Image is Worth 16x16 Words. Transformers for Image Recognition at Scale." ICLR 2021.

[10] Li, Yanyu, et al. "Efficientformer: Vision transformers at mobilenet speed". NeurIPS 2022.

[11] AI-Powered Imaging Biomarkers for Better Treatment | Radiology AI, https://www.brainomix.com/. Accessed 20 December 2023.

[12] GE Healthcare, 24 June 2019, https://www.gehealthcare.com/en-sg/?utm_medium=cpc&utm_source=google&utm_campaign=USC-PS-REG-AlwaysOn&utm_term=&utm_content=12518725892&npclid=Cj0KCQiAyeWrBhDDARIsAGP1mWTw75tpBxjR9178bijCg4dvoqZboJG9CjM_sfOjAI899Wk6eZ4NGkcaAjKlEALw_wcB&gad_source=1&gclid=C. Accessed 20 December 2023.

[13] e-Stroke | Automated AI-powered Decision Support for Stroke Assessment." Brainomix, 2023, https://www.brainomix.com/stroke/. Accessed 20 December 2023. https://www.brainomix.com/stroke/

[14] Verdolotti, Tommaso, et al. "ColorViz, a New and Rapid Tool for Assessing Collateral Circulation during Stroke." NCBI, 20 November 2020, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7699692/. Accessed 20 December 2023.

[15] JM, Ospel, and Volny O. "Displaying Multiphase CT Angiography Using a Time-Variant Color Map: Practical Considerations and Potential Applications in Patients with Acute Stroke." *PubMed*, 9 January 2020, https://pubmed.ncbi.nlm.nih.gov/31919139/. Accessed 20 December 2023.

[16] Andromeda Medical Imaging, 2023, https://andromedamedicalimaging.com/. Accessed 20 December 2023.

[17] Ranta, Anna, et al. "Current challenges in the endovascular treatment of medium vessel occlusions." Frontiers, 24 July 2023, https://www.frontiersin.org/articles/10.3389/fstro.2023.1242961/full. Accessed 20 December 2023.

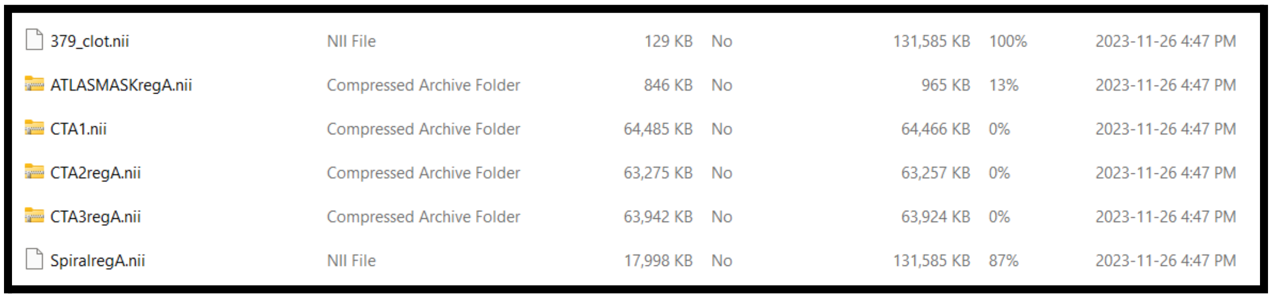
Appendix

A1: Data Exploration

In this section, we will walk the reader through a preliminary data exploration that will allow them to understand the intricacies behind the dataset we are dealing with. Please note that this analysis is meant to be extra information. The report should be self-contained; the data section should include all information required to understand our methodology

a. File Structure

Let us select a patient, say Patient #379. We observe the following files:

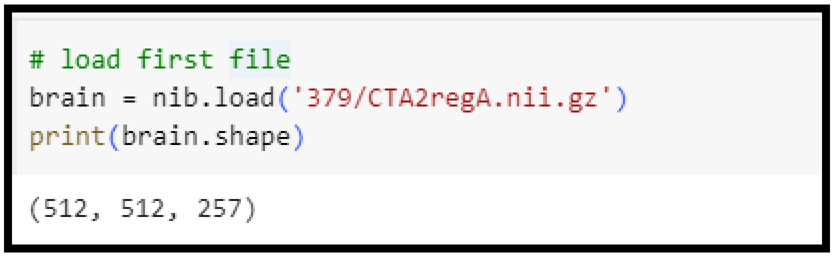


379_clot.nii	NII File	129 KB	No	131,585 KB	100%	2023-11-26 4:47 PM
ATLASMASKregA.nii	Compressed Archive Folder	846 KB	No	965 KB	13%	2023-11-26 4:47 PM
CTA1.nii	Compressed Archive Folder	64,485 KB	No	64,466 KB	0%	2023-11-26 4:47 PM
CTA2regA.nii	Compressed Archive Folder	63,275 KB	No	63,257 KB	0%	2023-11-26 4:47 PM
CTA3regA.nii	Compressed Archive Folder	63,942 KB	No	63,924 KB	0%	2023-11-26 4:47 PM
SpiralregA.nii	NII File	17,998 KB	No	131,585 KB	87%	2023-11-26 4:47 PM

Figure 7: All files for a single patient

b. CTA Scans

The 3 CTA files are meant to be 3 separate CTA scans. The regA means the individual frames of the scan are aligned. For input to our model, we wish to use these reg files (assuming input is raw CTA scan and not SPIRAL map), in order to simplify training. Let us investigate one of the CTA files.



```
# load first file
brain = nib.load('379/CTA2regA.nii.gz')
print(brain.shape)
```

```
(512, 512, 257)
```

Figure 8: Raw CTA data shape

The scan is 512x512x257. We visualize this scan below.

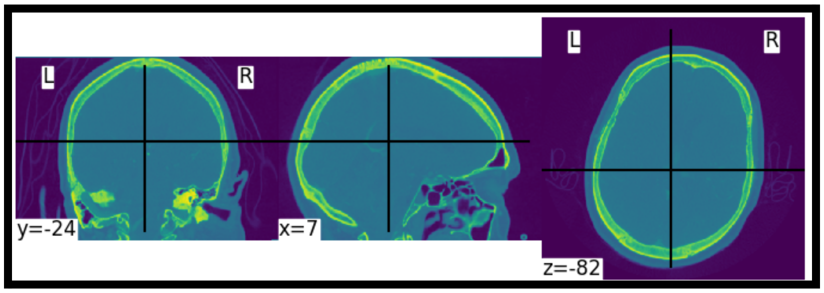


Figure 9: Visualization of raw CTA scan

The pixel values of the 3D brain scan are summarized by the histogram below:

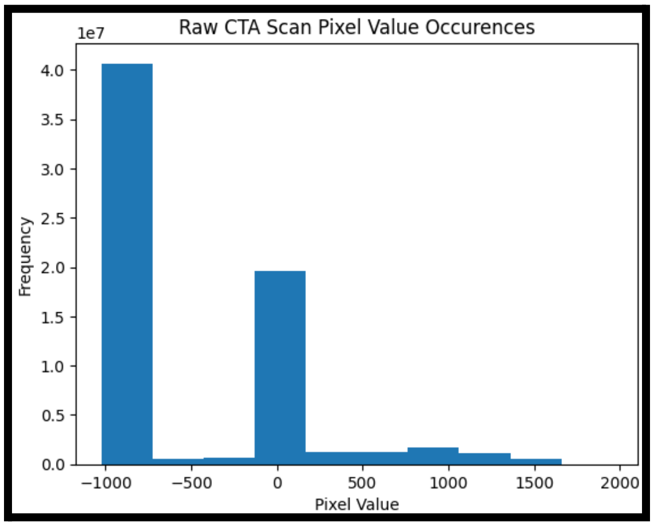
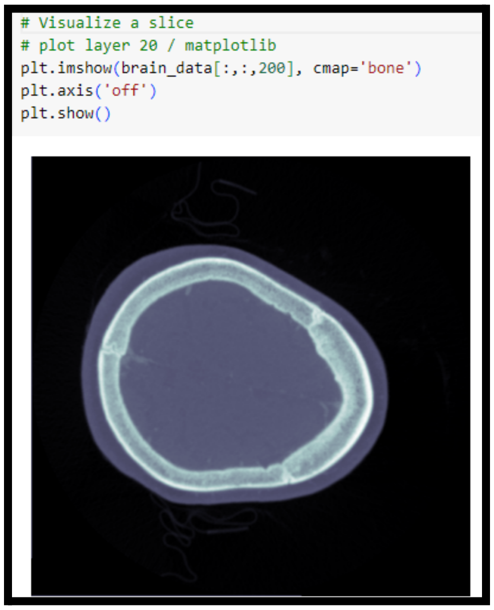


Figure 10: Distribution of pixel values in raw CTA scan

An individual slice is visualized below:



```
# Visualize a slice
# plot layer 20 / matplotlib
plt.imshow(brain_data[:,:,200], cmap='bone')
plt.axis('off')
plt.show()
```

Figure 11: Individual slice from CTA scans of a single patient

A series of slices, from the top of the brain to the bottom, is shown in Figure 12.

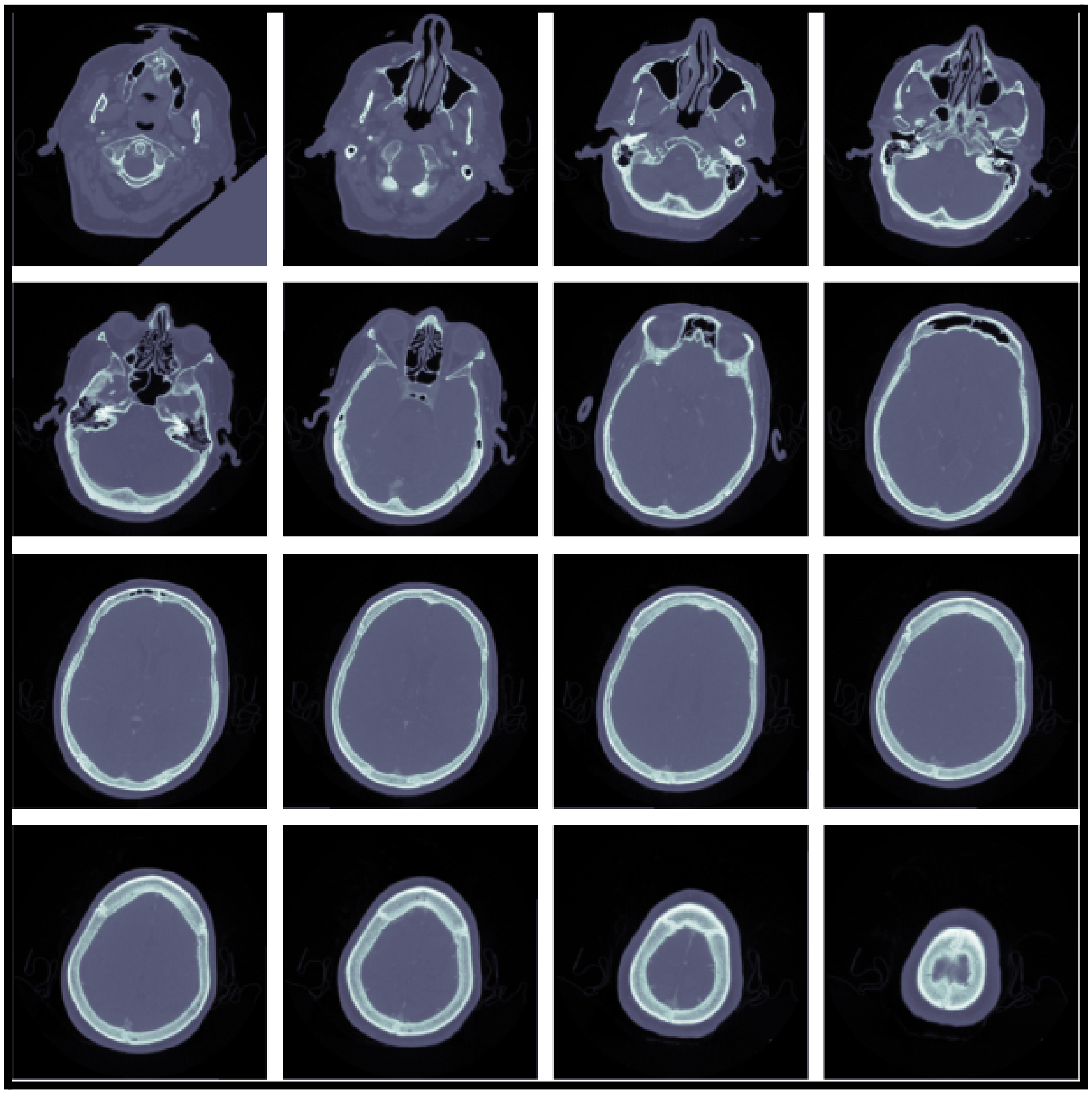
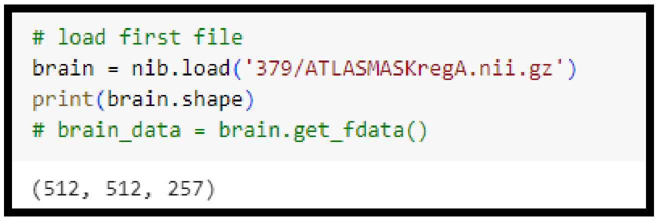


Figure 12: Series of CTA scans from top of head to bottom

c. ATLASMASK File

We continue with Patient 379, this time investigating the ATLASMASK. To begin, observe that the dimensions of the ATLASMASK and CTA scans are the same, i.e. they are aligned. This means if we identify regions of the brain, the neurologist will be able to easily find the corresponding regions in the scan.



```
# load first file
brain = nib.load('379/ATLASMASKregA.nii.gz')
print(brain.shape)
# brain_data = brain.get_fdata()
```

```
(512, 512, 257)
```

Figure 13: Shape of ATLASMASK data

We visualize the ATLASMASK below.

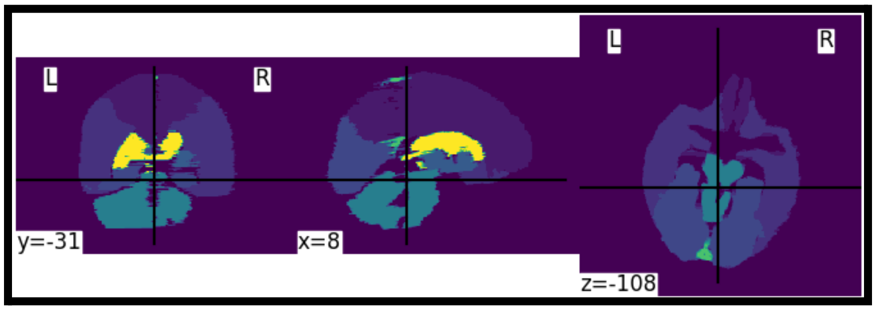


Figure 14: Visualization of ATLASMASK

The pixel values of the 3D ATLASMASK are summarized by the histogram below. For any patient, the pixel values can only be between 0 and 14, inclusive.

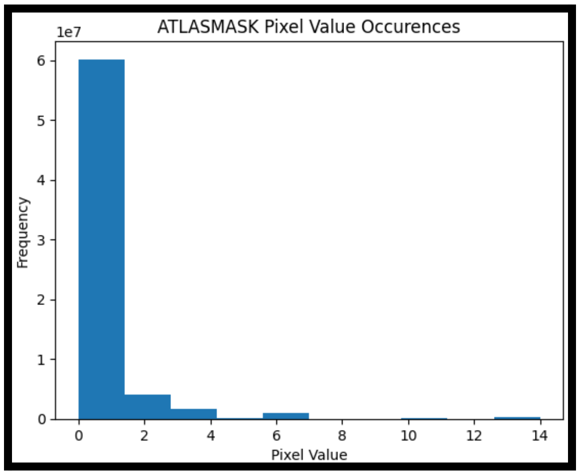
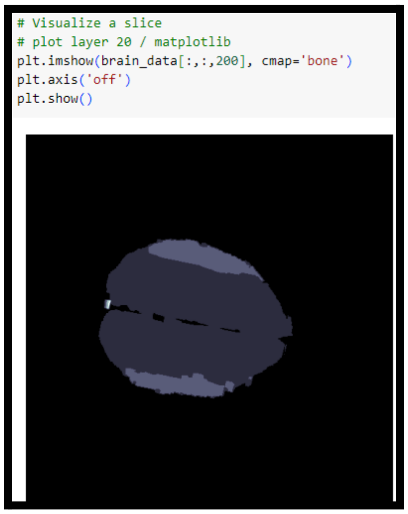


Figure 15: Distribution of pixel values in ATLASMASK

An individual slice is visualized below:



```
# Visualize a slice
# plot layer 20 / matplotlib
plt.imshow(brain_data[:,:,200], cmap='bone')
plt.axis('off')
plt.show()
```

Figure 16: Individual slice from ATLASMASK of a single patient

A series of slices, from the top of the brain to the bottom, is shown in Figure xyz

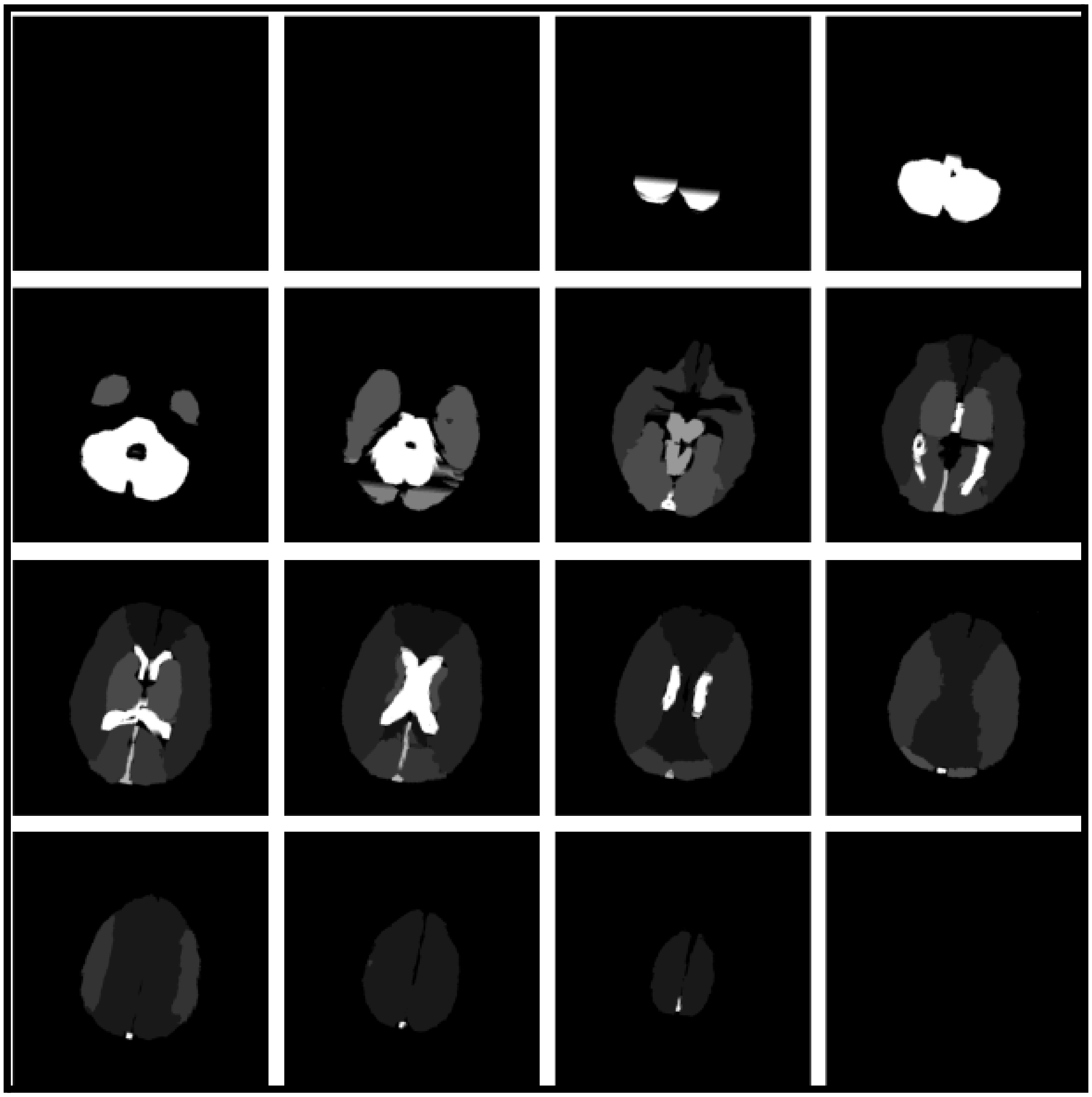
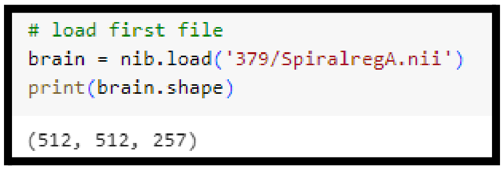


Figure 17: Series of ATLASMASK scans from top of head to bottom

d. SPIRAL Map File

We now investigate the SPIRAL file of the same patient. First, observe that we retain the consistent dimension 512x512x257. This has the same implications mentioned in part c, namely that we can use the ATLASMASK to identify which part of the SPIRAL map corresponds to affected regions. It also means we can train models that take SPIRAl and raw CTA scans concatenated as input.



```
# load first file
brain = nib.load('379/SpiralregA.nii')
print(brain.shape)

(512, 512, 257)
```

Figure 18: Shape of SPIRAL mask

We visualize the SPIRAL map below:

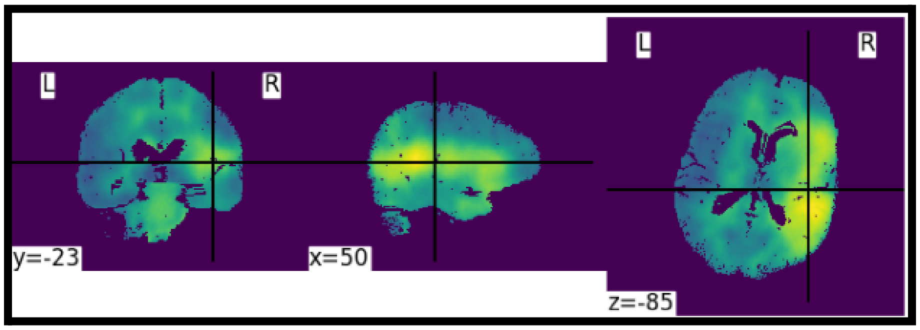


Figure 19: Visualization of SPIRAL map

The pixel values of the 3D SPIRAL map are summarized by the histogram below. For any patient, the pixel values are real numbers between 0 and 1.

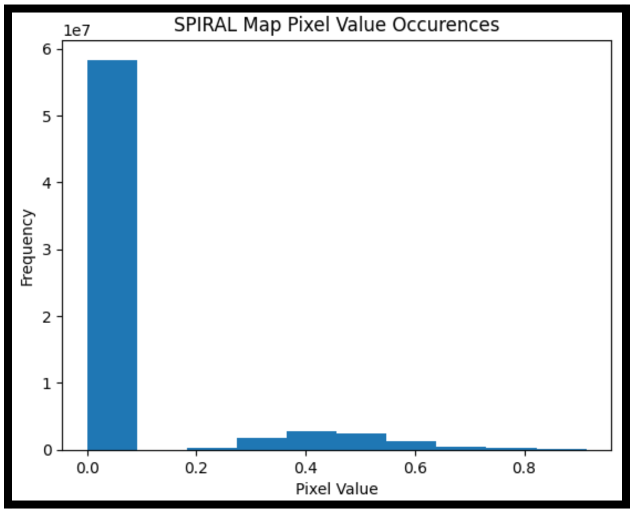
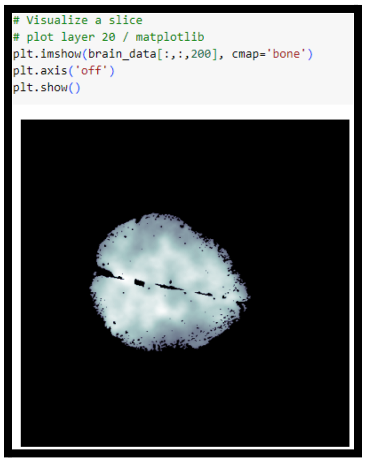


Figure 20: Distribution of pixel values in SPIRAL map

An individual SPIRAL slice is shown below:



```
# Visualize a slice
# plot layer 20 / matplotlib
plt.imshow(brain_data[:,:,200], cmap='bone')
plt.axis('off')
plt.show()
```

Figure 21: Individual slice from SPIRAL map of a single patient

A series of slices, from the top of the brain to the bottom, is shown in Figure xyz

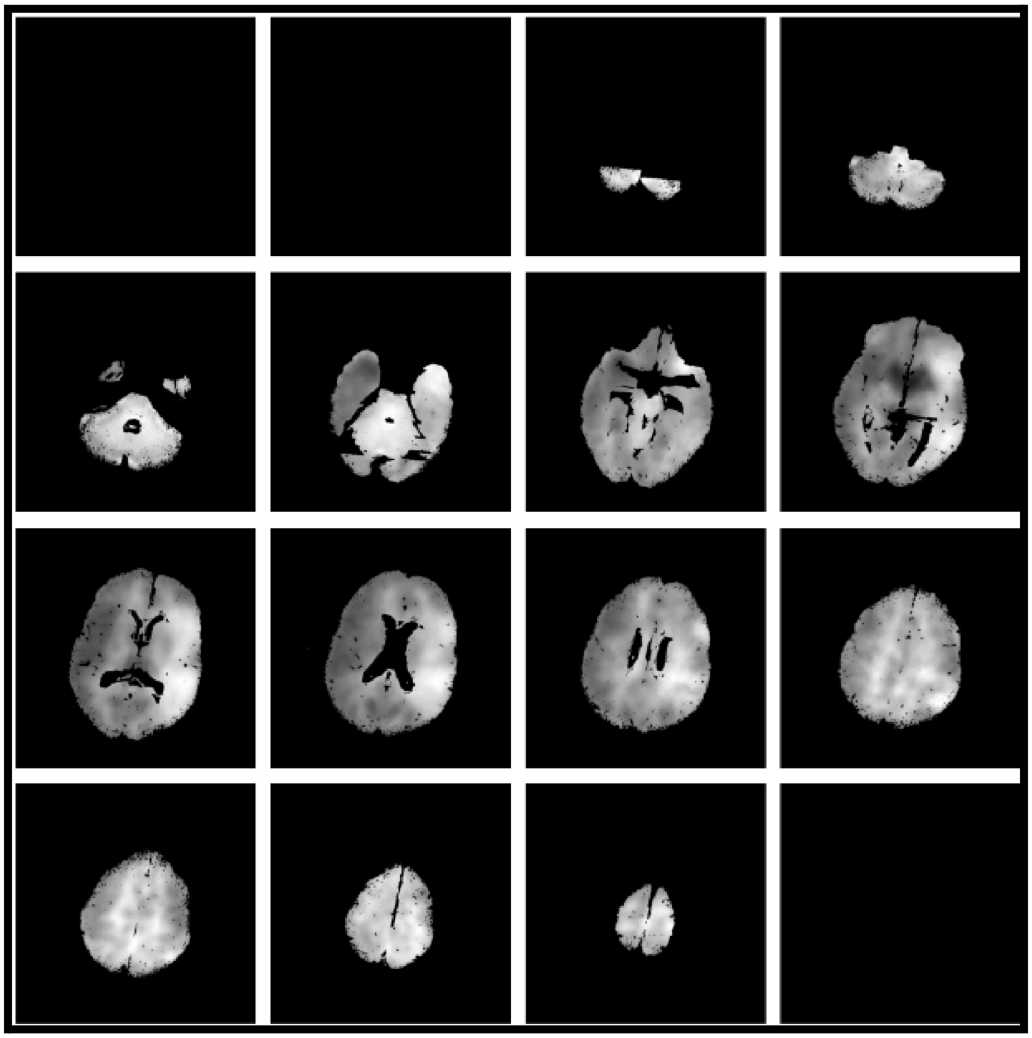
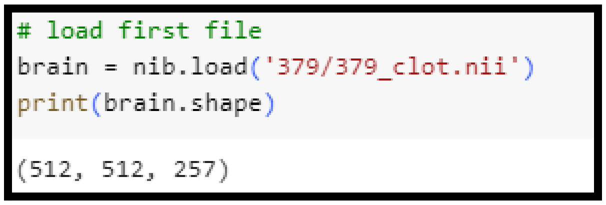


Figure 22: Series of SPIRAL map scans from top of head to bottom

e. Clot File

The trend continues. The clot file has the same dimensions. This is especially important as it makes data labeling easy. We simply find the brain regions corresponding to blood clot pixels.



```
# load first file
brain = nib.load('379/379_clot.nii')
print(brain.shape)

(512, 512, 257)
```

Figure 23: Shape of Clot file

We visualize the clot file below. Observe how these clot files are very sparse. The pixel(s) corresponding to clots are circled.

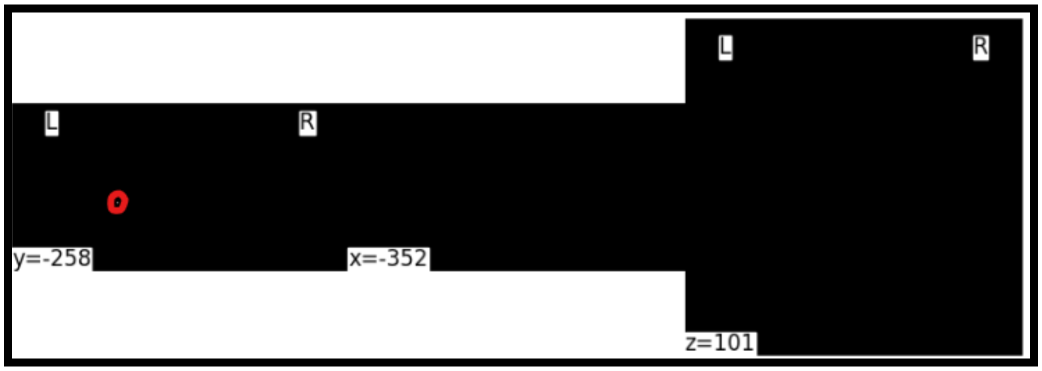


Figure 24: Visualization of Clot file

The pixel values of the 3D clot files are summarized by the histogram below. For any patient, the pixel values are either 0, 1, or 2. For our purpose, we treat 0 as pixels not affected by the stroke and 1,2 as pixels affected by the stroke. As can be seen in the bar graph below, the clot file is very sparse - almost all pixels are 0!

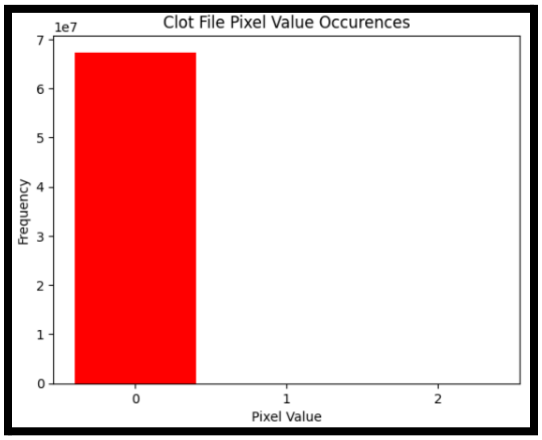


Figure 25: Data Imbalance in Clot File

To show that there *are* non-zero clot pixels, we can choose to ignore 0 in the clot. This gives:

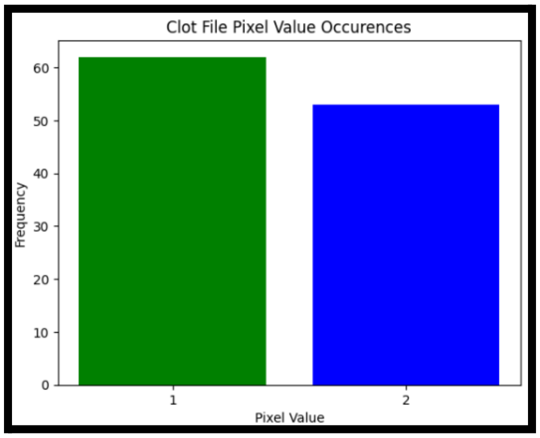


Figure 26: Clot Pixel Values (only non-zero values)

An individual clot slice is shown below:

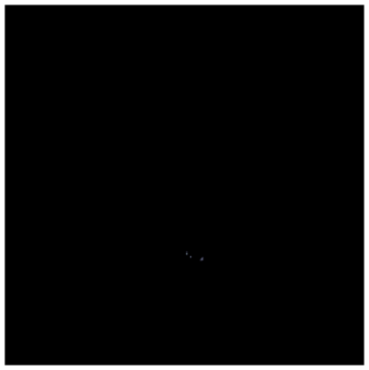


Figure 27: Clot Visualization for a single 2D frame

A series of clot slices, from the top of the brain to the bottom, is shown in Figure 27

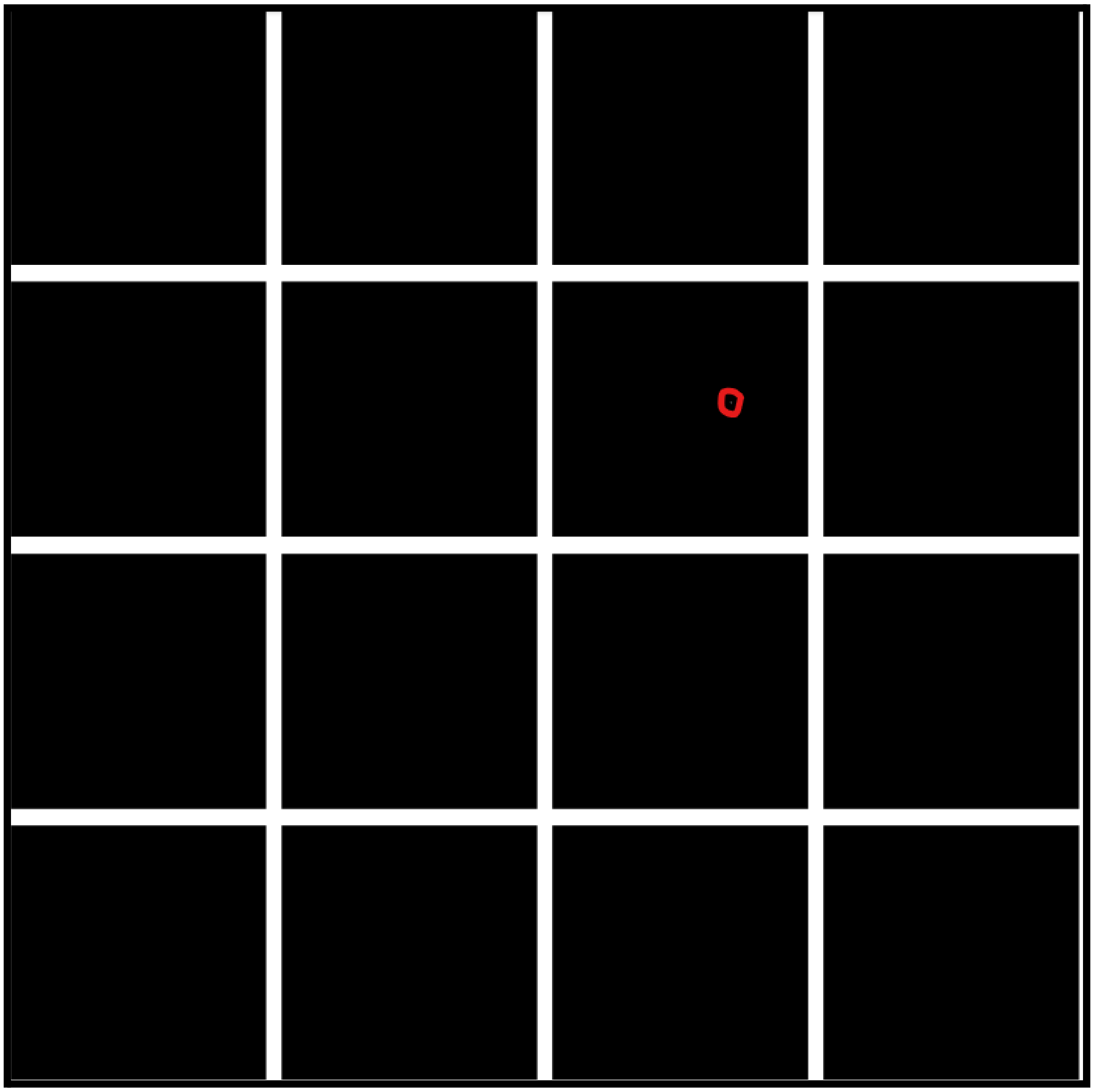


Figure 28: Series of Clot file scans from top of head to bottom

A2: Results

a. SPIRAL Rule-Based Baseline:

The results of our rule-based SPIRAL map to affected brain regions across all 4 metrics are shown in Figure xyz. The optimal threshold is highlighted in green. For this non-neural approach we do not highlight the best precision, accuracy or recall. It is intuitive that the higher the threshold, the better the precision and the lower the threshold, the better the recall.

Threshold	Precision	Recall	Accuracy	F1 Score
0.0	0.1577	0.7089	0.2139	0.2579
0.1	0.1577	0.7089	0.2139	0.2579
0.2	0.1577	0.7089	0.2139	0.2579
0.3	0.1577	0.7089	0.2139	0.2579
0.4	0.1577	0.7089	0.2139	0.2579
0.5	0.1577	0.7089	0.2139	0.2579
0.6	0.1559	0.6747	0.2330	0.2532
0.7	0.1659	0.6507	0.3023	0.2644
0.8	0.1879	0.5445	0.4587	0.2794
0.9	0.3344	0.3596	0.7386	0.3465
0.95	0.5443	0.1473	0.8119	0.2318

Table 1: Evaluation metrics across various thresholds for our rule-based method

b. CNN Baseline:

i) Using SPIRAL as Input

The results of our CNN baseline across different hyperparameter settings are shown in Figure xyz. The best results are highlighted in green. Note that the setting that optimizes the F1 score also optimizes precision, recall, and accuracy.

pool_dim	4	4	4	4	4
feat_dim	512	512	512	512	512
activation	tanh	relu	sigmoid	relu	sigmoid
learning rate	0.0013	0.00061	0.0031	0.0048	0.00068
dropout_prob	0.32	0.39	0.20	0.13	0.17
pos_weight	17.72	15.26	15.58	12.29	13.91
extra_layers	0	2	2	4	2
resize_dims	128x128	128x128	512x512	512x512	128x128
iterations	8	20	4	8	4
total time (s)	79.80	106.68	288.84	478.25	67.42
Val_F1	0.33	0.60	0.18	0.40	0.17
Val_Precision	0.20	0.46	0.35	0.27	0.11
Val_Recall	0.91	0.86	0.13	0.77	0.41
Val_Accuracy	0.96	0.99	0.99	0.98	0.96

Table 2: Evaluation metrics across different hyperparameter settings for simple CNN using SPIRAL map as input

ii) Using Raw CTA Scan as Input:

We tabulate the same hyperparameters and evaluation metrics as above, but train and test using the raw CTA scan as input. Once again, the setting that optimizes F1 score also optimizes the other 3 metrics.

pool_dim	4	4	4	4	4
feat_dim	512	512	512	512	512
activation	tanh	relu	sigmoid	tanh	sigmoid
learning rate	0.0011	0.0019	0.0007	0.0009	0.0014
dropout_prob	0.25	0.31	0.29	0.27	0.32
pos_weight	12.91	13.41	14.97	15.65	14.28
extra_layers	2	2	2	2	2
resize_dims	256x256	256x256	512x512	128x128	128x128
iterations	20	16	4	20	4
total time (s)	325.90	298.29	252.01	132.72	52.29
Val_F1	0.62	0.47	0.19	0.67	0.36
Val_Precision	0.51	0.32	0.16	0.51	0.23
Val_Recall	0.78	0.89	0.24	0.95	0.78
Val_Accuracy	0.99	0.98	0.98	0.99	0.97

Table 3: Evaluation metrics across different hyperparameter settings for simple CNN using raw CTA scan as input

A3. Model Architectures

In our complex model fine tuning approaches, we consider 2 main architectures: MobileNetV2 and EfficientFormerV2. These architectures are visualized in Figures 29 and 30, respectively.

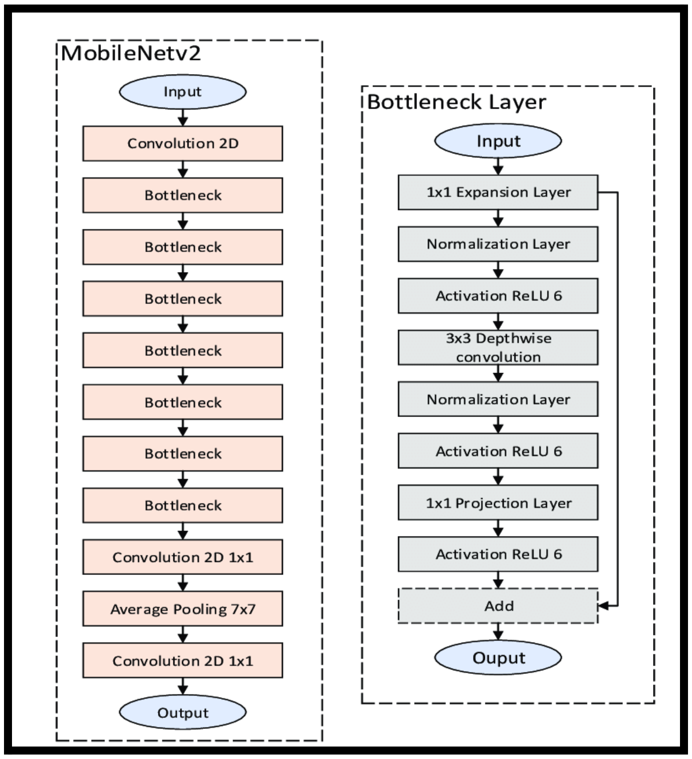


Figure 29: MobileNetV2 Architecture

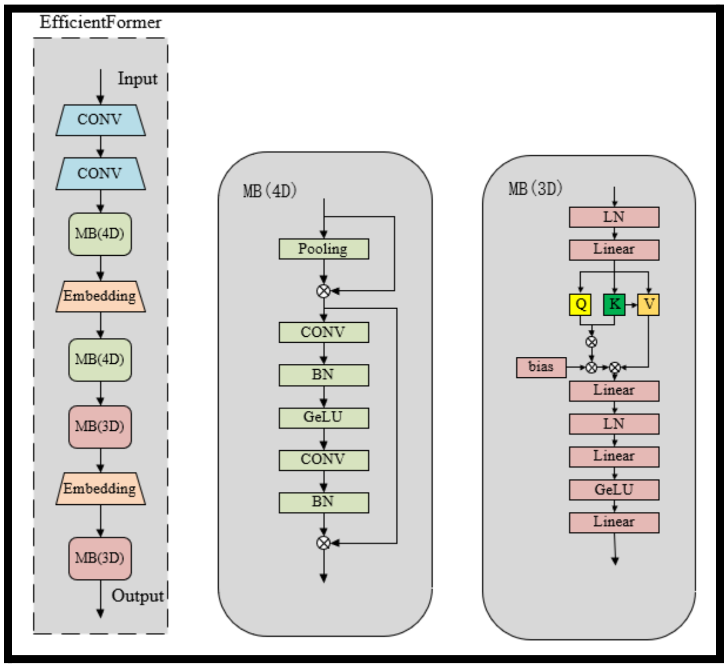


Figure 30: EfficientFormerV2 Architecture

We use these pretrained architectures via *transfer learning* - we initially train only the classifier weights, then unfreeze the base model and finetune all weights at a smaller learning rate.

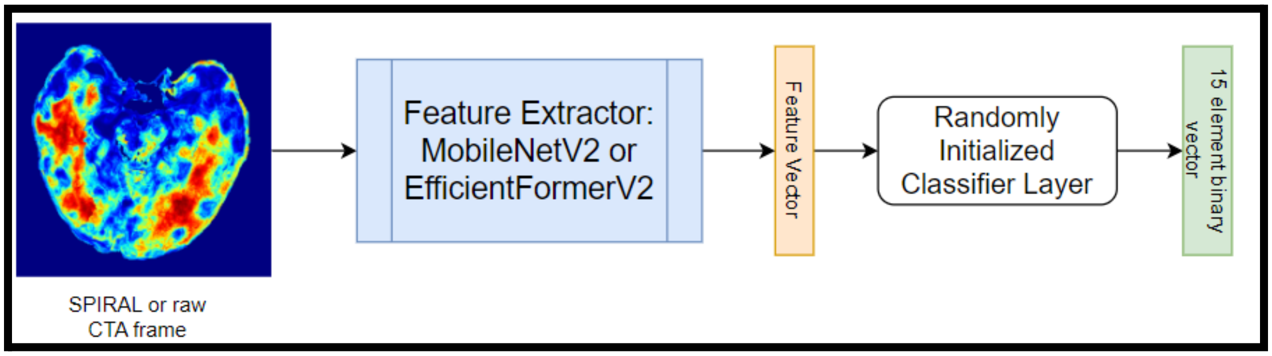

SPIRAL or raw CTA frame
Feature Extractor: MobileNetV2 or EfficientFormerV2
Feature Vector
Randomly Initialized Classifier Layer
15 element binary vector

Figure 31: Transfer Learning Approach for Complex Neural Architectures