Image Classification Of Stroke Blood Clot Origin Using Deep Learning Techniques

Ahmed Hassan[1], Habiba Fathallah[2], Rawan Mohamed[3], and Romaisaa Saad[4]

***Abstract*— To revolutionize stroke clot classification, this paper unveils a cutting-edge deep learning model. Built by merging powerful ResNet and SqueezeNet architectures, it thrives on a meticulously balanced dataset of 4313 high-resolution images. It tackles challenges like class imbalance and image size variations with a unique arsenal of transfer learning, precise hyperparameter tuning, and innovative data augmentation. This empowers the model to achieve a landmark Log Loss of 0.6036, paving the way for clinical implementation and empowering personalized stroke management. *Code available on* Clot-Classification**

Index Terms — Blood Clot, ResNet50, SqueezeNet

I. INTRODUCTION

Stroke, a devastating enemy of brain health, disrupts the delicate balance of blood flow in the brain. Often lurking unnoticed, the culprit is a blood clot, whose source determines the most appropriate defense. Identifying the clot's origin, either large artery atherosclerosis (LAA) or cardioembolic (CE), remains a huge challenge, shrouding treatment pathways in uncertainty.

That's where deep learning steps in. It's like a detective that can analyze digital images of blood clots and uncover hidden clues about their origins that even human eyes can't see. This breakthrough could lead to more personalized stroke prevention and treatment, and even pave the way for new diagnostic tools and therapies.

This paper explores how deep learning can be used to classify blood clots into LAA or CE, which is a huge step forward in stroke care. It details how these algorithms are trained and evaluated, and it proposes a new system that analyzes whole-slide digital pathology images of clots to accurately determine their origin.

Healthcare professionals are exploring deep learning to predict stroke causes and clot origins, but unique data formats, large images, and limited pathology slides pose challenges demanding innovative solutions. Despite these hurdles, the potential for more accurate clot classification is immense, allowing doctors to tailor treatment plans, optimize outcomes, and reduce future stroke risks for each patient.

[1]Ahmed Hassan is with the Biomedical Engineering Department, Cairo University, Giza, Egypt. ahmed.ali009@eng-st.cu.edu.eg

[2] Habiba Fathallah is with the Biomedical Engineering Department, Cairo University, Giza, Egypt. habiba.elshenofy01@eng-st.cu.edu.eg

[3]Rawan Mohamed is with the Biomedical Engineering Department, Cairo University, Giza, Egypt. rawan.mostafa01@eng-st.cu.edu.eg

[4]Romaisaa Saad is with the Biomedical Engineering Department, Cairo University, Giza, Egypt. romaisaa.elsaidy01@eng-st.cu.edu.eg

In this paper, we propose a system that solves all these challenges for simple classification task. This system aims to accurately classify them into CE and LAA, achieving performance on par with human pathologists in identifying the origin of stroke blood clots.

II. RELATED WORKS

Krishnan et al. (2023) [12] investigated the potential of deep learning in classifying blood clot images. Their work demonstrates promising results utilizing a combination of transfer learning and data augmentation techniques. The methodology involved extracting 600x600x3 image patches from whole-slide digital pathology images, followed by noise reduction via Otsu's thresholding. To address potential data limitations, (50%) of the training data underwent augmentation with different techniques. Leveraging transfer learning, the authors fine-tuned several pre-trained models. Hyperparameter optimization was achieved through the Optuna framework, and model performance was further enhanced by employing Stochastic Weight Averaging and Early Stopping schedulers. The Swin TransformerV2 model achieved the highest performance, demonstrating an accuracy of (94.24%), along with impressive precision, recall, and f1-score values. While the study primarily relied on accuracy and confusion matrix for evaluation, its findings highlight the substantial potential of deep learning in aiding stroke diagnosis. Furthermore, the successful application of data augmentation techniques underscores its value in mitigating the issue of limited datasets within the medical domain.

Narayana et al. [8] evaluated the performance of various image classification models (CNN, DenseNet, EfficientNet, ResNet) without data preprocessing (except resizing), comparing accuracy across different learning rates and optimizers. CNN with SGD achieved the highest accuracy (71.714%) at learning rates 0.001-0.005. DenseNet121 excelled with DLR variations (constant, exponential, step) reaching (68.21%), while EfficientNetB0 performed well with SGD and DLR variations (exponential, step), and surprisingly, with a constant learning rate (71.215%, 76.206%, respectively). ResNet121 with RMSprop yielded the highest training accuracy (72.546%) with a constant learning rate of 0.001. Notably, ResNet50 with RMSprop surpassed all models with an impressive (85.025%) accuracy, although exhibiting overfitting. Though lacking data preprocessing and potentially unsuitable for high-resolution images, this study offers a valuable overview of 36 model configurations, showcasing optimal performance with various hyperparameters.

Azatyan et al. [2] employed a comprehensive approach for image classification using transfer learning and data augmentation. They leveraged state-of-the-art architectures like EfficientNet CNN and Swin Transformers. To address data limitations, they applied diverse preprocessing and augmentation techniques. Notably, they implemented an ensemble of 5 models with varying EfficientNet versions and Swin Transformer configurations, enhancing robustness through "noisy student" weights and different resolutions. Transfer learning was further refined by adding linear layers, dropout, and a softmax activation function. Optimization relied on Adam optimizer with specific parameters and label smoothing for regularization. WMCLL (weighted multi-class logarithmic loss) served as the primary evaluation metric, achieving a final score of 0.67188 with the ensemble model. While additional test-time augmentation or gradient-boosting machines built on neural network embeddings could further improve performance, Azatyan's study demonstrates the effectiveness of combining powerful architectures, data augmentation, and ensemble techniques for accurate image classification tasks.

III. DATASET AND FEATURES

A. Dataset

The dataset [5] comprises 754 high-resolution whole-slide digital pathology images in TIF format. Each image showcases a distinct blood clot, offering a window into the pathology of this devastating condition. Notably, these images stem from 632 patients across 11 medical centers, enriching the dataset with diverse clinical contexts. Interestingly, individual patients may contribute up to five images, providing valuable longitudinal insights into clot evolution. Each image carries a crucial label, categorizing its origin as either cardioembolic (CE) or large artery atherosclerosis (LAA), enabling the study of these distinct stroke etiologies. However, a noteworthy class imbalance characterizes the dataset, with CE clots constituting 72.5% and LAA clots representing 27.5%. Figure1 visually depicts this disparity, highlighting the need for careful consideration during model training and evaluation to mitigate potential biases.

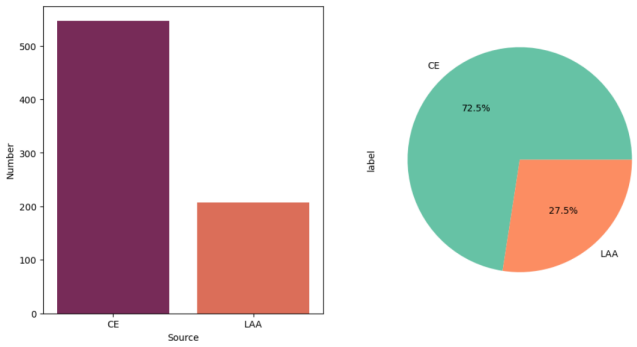


Fig. 1: Data distrbution

Furthermore. image sizes exhibit pronounced variability, necessitating robust preprocessing techniques to ensure consistent input dimensions for the deep-learning models Figure2. The dataset's images are in RGB color format.

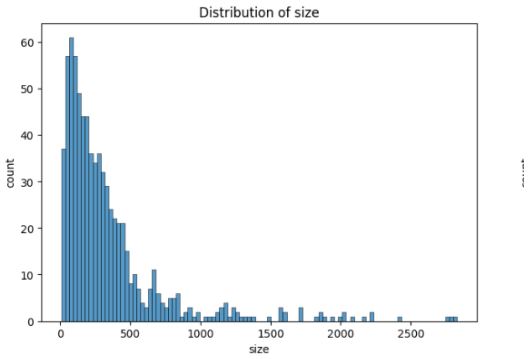


Fig. 2: Size distribution for images

To facilitate rigorous model evaluation, we meticulously crafted a balanced test set by extracting 20 images, ensuring a proportional representation of CE and LAA clots. The remaining images were employed for training and validation purposes, adhering to an 80:20 split, thereby optimizing model learning while reserving a sufficient portion for unbiased performance assessment.

B. Pre-Processing Approach 1: Images De-duplication

Pre-processing is an essential step in any project as it helps to improve the quality of an image, Which in return enhances the important features in the image to be suitable for the deep model to extract features from it clearly and if there's any noise or artifacts in the image, the model won't get affected with it. Since our data is a high-resolution whole slide digital pathology image, which is stored in 2.5 GB

Step 1, We start our pre-processing pipeline by using Pyvips [7] that utilizes libvip's multi-threaded and parallel pre-processing tasks to be efficient in handling lavage image files and minimize Ram usage, Normalize the image by a ratio Which is computed based on image's situation and color intensity Figure 3.

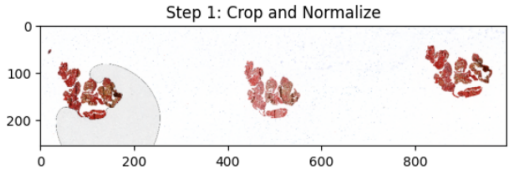


Fig. 3: Resulted Image after Step 1

Step 2, Resize the image to 256x256 pixels, Detect bubbles in the image, Remove them using a series of erosion and dilation processes, Convert to gray scale, Remove the background, and Apply a Gaussian filter. Figure 4.

Step 3, Convert the image to binary type using the gray-scale image, Apply the denoising function on the binary image [1], Label the connected components in the binary image using the label Function in Scikit-image to use the resulted labeled image in the rest of the pipeline Figure 5. Step 4, Use the labeled image to create a data frame that contains information about the labeled connected components as

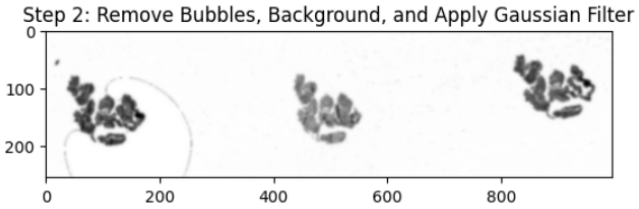


Fig. 4: Resulted Image after Step 2

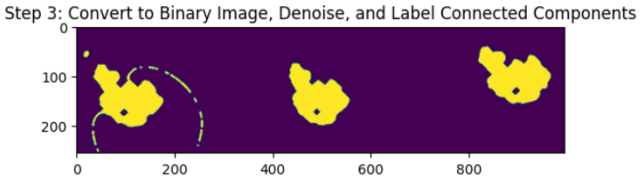


Fig. 5: Resulted Image after Step 3

area and bounding box coordinates, Measure the dissimilarity (Distance component) between the connected components, and Calculate two types of distances both Euclidean distance and shape distance and then Combine both distances to measure the overall dissimilarity between the sub-images, then based on these combined distances we set a threshold to identify the duplicated part in the image, Group the same objects of an image in clusters.

Step 5, based on the clusters created from the previous step, we define a region of interest (ROI) and draw bounding boxes around it Final step in the pipeline, We extract the ROI from the previous step and then normalize the resulting image and then it's our final image Figure 6 which we will work with it.

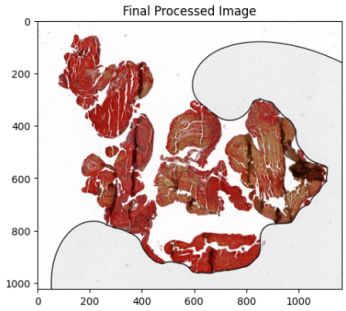


Fig. 6: Resulted Image

C. Pre-Processing Approach 2: Convert Image to Tiles

As the number of training dataset images is only 754 images and the distribution of the classes is not balanced as explained in the dataset part, we tried different approaches to deal with the dataset, and our main target was to increase the number of images taking advantage of repeated clots in one image and augmentation techniques.

Step 1, Extracted the image dimensions like width and height using OpenSlide [14] python library and detect which one of them is the bigger dimension to then calculate the maximum minimum ratio to determine the number of tiles (sub-images) that'll be split from the original image and then save all of this information in Data frame. to be able to retrieve it in the next step.

Step 2, Based on the number of splits found on the image, Determine the size of each tile in image 1, Crop it and then resize the resulted tile image to 224×224, if the resulted file has low contrast we discard this tile,and for the resulted tile image if the label is " CE" Perform one rotation degree and save it but if the label is " LAA" Perform three rotation operation with different degrees to try to balance data distrbution 7

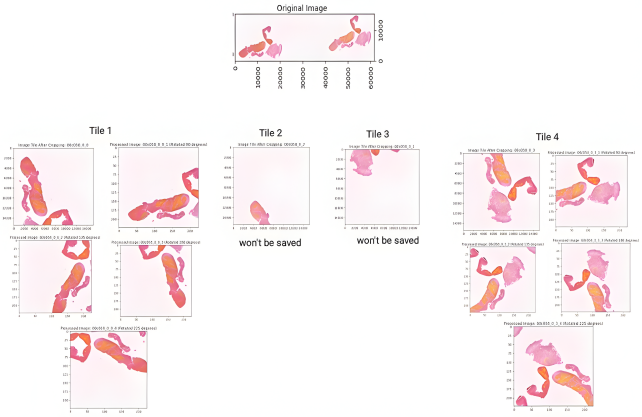


Fig. 7: Example of "LAA" Clot Type With Different Rotation Angles

This technique effectively balanced the dataset and boosted training data volume into 4313 used in model training. Figure8 Shows the final data distribution

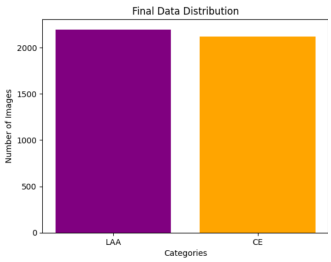


Fig. 8: Final data distrbution

IV. METHODS

In this section, we detail the final approach employed for achieving good performance: ensemble learning between CNN Architecture as EfficientNetB2, SqueezeNet, and ResNet, leveraging the aforementioned second approach for data preprocessing. The detailed architecture for each model is as follows:

A. Main Architectures

EfficientNet [15] boasts a clever design that prioritizes both accuracy and efficiency with limited computing resources, it's architecture as in figure 9consists of main

blocks: Compound Scaling, which adjusts depth, width, and resolution using pre-determined coefficients to uniformly scale dimensions simultaneously. Mobile Inverted Bottleneck (MBConv) block, combines depthwise separable convolutions, squeeze-and-excitation optimization, and linear bottlenecks. Progressive Inverted Bottleneck (PIB) block, used in later stages to add inverted residual connections and to increase kernel size progressively. Squeeze-and-Excitation (SE) optimization, This attention mechanism dynamically adjusts channel weights within MBConv blocks.

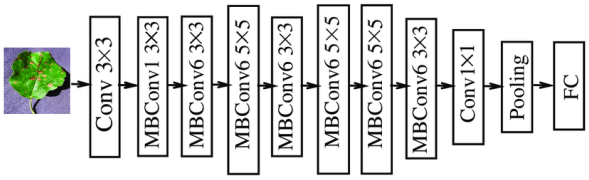


Fig. 9: EfficientNet Block Diagram

SqueezeNet[11], an architectural marvel in the realm of convolutional neural networks (CNNs) that is an improvement for AlexNet, the architecture depends mainly on Fire Modules which is the core building blocks of SqueezeNet. Each fire module shown in Figure 10 consists of two layers: a squeeze layer and an expand layer. The squeeze layer employs a 1x1 convolution filter to reduce the input channels, creating a "bottleneck" that drastically cuts down on computational cost. This is followed by the expanding layer, which utilizes a mix of 1x1 and 3x3 convolution filters to introduce new features and expand the channels back to a desirable level.

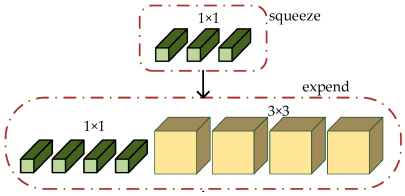


Fig. 10: Fire Module Block Diagram

The final combination in our model between fire module, batch normalization, and pooling is shown in Figure 11

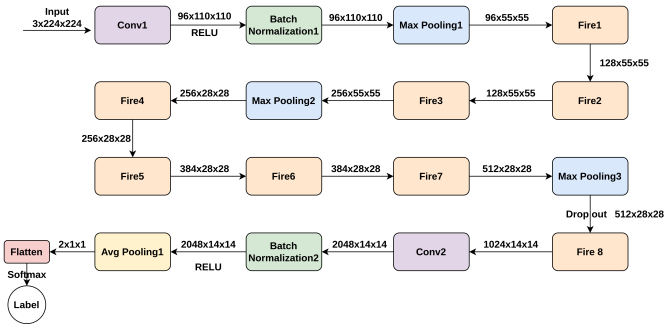


Fig. 11: SqueezeNet Block Diagram

ResNet50 [10] To address the vanishing gradient problem, this convolutional neural network architecture (CNN) leverages the innovative concept of residual connections. The network comprises 50 meticulously arranged layers distributed across five stages as in Figure 12 These residual connections act as information superhighways, directly adding the original input to the transformed output of each convolutional block, akin to preserving vital details in a cascading waterfall. This bypass mechanism ensures vital information persists throughout the network's hierarchical processing, ultimately facilitating the training of deeper architectures and boosting feature extraction capabilities.

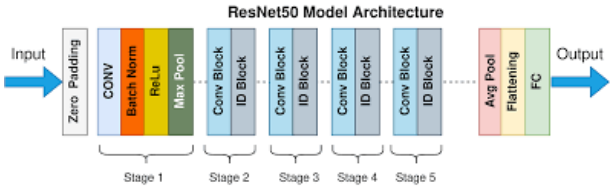


Fig. 12: ResNet50 Block Diagram

B. Loss Function

Binary Cross Entropy used as a loss function for all models

C. Final Model and Test

The following diagram describes how the final architecture works for testing

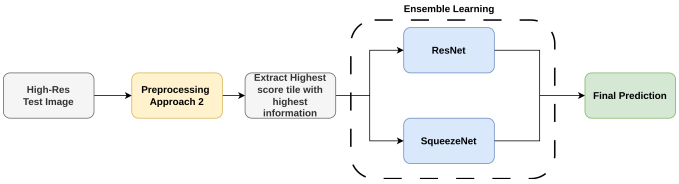


Fig. 13: Test Image Prediction Pipeline

V. Experiments

A. Experiments

Through final model experiments, initial learning rate = 0.001, Adam optimizer, batch size = 16, and using reduce on plateau technique for learning rate every 5 epoch check with reducing factor = 0.1

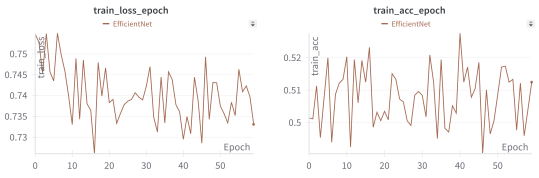
EfficientNet B2, with 60 epochs in Figure14

SqueezeNet, with 200 epochs in Figure 15

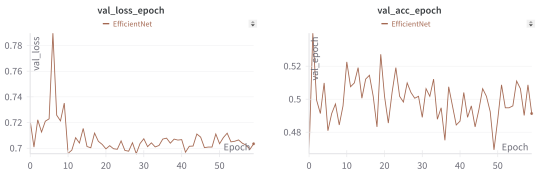
ResNet50, with 60 epochs and learning rate = $1 \cdot 10^{-7}$. in Figure16

B. Results

In this section, we run models on 20 test images with the distribution of original data (14 CE and 6 LAA). To ensure a comprehensive assessment of our model's performance, we employed a multi-faceted evaluation approach. This involved the utilization of confusion matrices shown in figure 17, both for individual models and the ensembled model, as well as additional evaluation techniques.

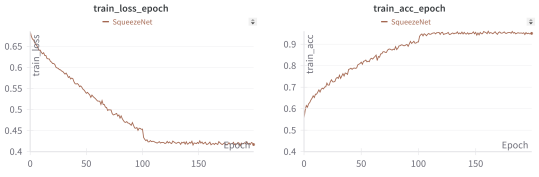


(a) Training Loss (b) Training Accuracy

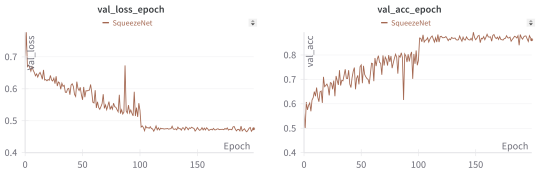


(c) Validation Loss (d) Validation Accuracy

Fig. 14: Training insights for EfficientNet



(a) Training Loss (b) Training Accuracy



(c) Validation Loss (d) Validation Accuracy

Fig. 15: Training insights for SqueezeNet

For competition evaluation, they use a weighted multi-class logarithmic loss equation

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

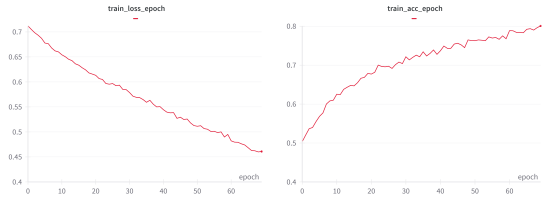
where **N** is the number of images in the class set, **M** is the number of classes, ln is the natural logarithm, $\mathbf{y}_{ij}$ is 1 if observation **i** belongs to class **j** and 0 otherwise, $\mathbf{p}_{ij}$ is the predicted probability that image **i** belongs to class **j**.

The result for our test dataset was = **0.6036** which is good compared to the leaderboard.

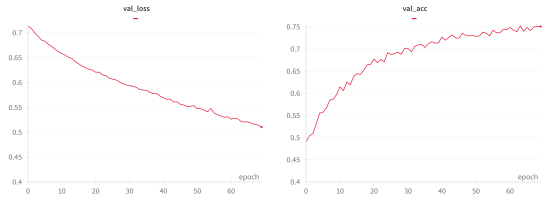
C. Discussion

Through experiments and trying to change hyperparams, the hyperparameters for both SqueezeNet and ResNet showed a better performance than EfficientNet. So, EfficientNet was excluded from the final architecture and testing steps.

Multiple initial learning rates were experimented with for a low number of epochs until observed that the learning rate is needed to be changed based on loss reduction per 3 epochs.

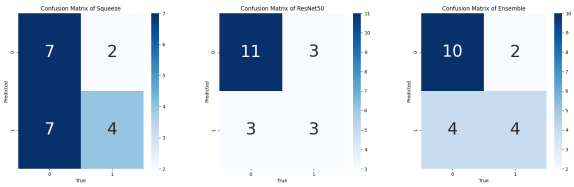


(a) Training Loss (b) Training Accuracy



(c) Validation Loss (d) Validation Accuracy

Fig. 16: Training insights for ResNet50



(a) SqueezeNet (b) ResNet50 (c) Ensemble

Fig. 17: Confusion Matrices for models

Starting with the lowest possible learning rate showed very long time training.

Batch sizes of 8, 16, and 32. were experiments. batch size of 16 showed better performance during training time

VI. Conclusion and Future Work

A. Conclusion

Interestingly, for our smaller dataset, which has similarities to ImageNet, SqueezeNet outperformed EfficientNet. This suggests that SqueezeNet's architecture, with its focus on parameter efficiency and fewer layers, might be better suited for scenarios where data scarcity could lead to overfitting. Compared to EfficientNet, ResNet-50 boasts a significantly deeper architecture, characterized by a greater number of convolutional layers. This enhanced depth empowers ResNet-50 to extract more intricate and nuanced feature representations from the data. As our data contains very important small details so ResNet50 was better with our data than EfficientNet

B. Furure Work

While our system excels in pinpointing blood clot origins, there's room to grow. Future research will integrate patient-specific data, refine hyperparameters for better results, and enrich training data. We'll explore alternative ensemble learning approaches and work to train more architectures.

References

[1] Erosion and dilation in image processing. See "Morphological Operations" section.

[2] David Azatyan. Image classification of stroke blood clot origin using deep convolutional neural networks and visual transformers, 2023.

[3] David Azatyan. Image classification of stroke blood clot origin using deep convolutional neural networks and visual transformers, 05 2023.

[4] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[5] Ashley Chow, Barbaros, Ryan Holbrook, Sobhi Jabal, and Vikash Gupta. Mayo clinic - strip ai. In *Kaggle*, 2022.

[6] Alex Clark. Pillow (pil fork) documentation, 2015.

[7] John Cupitt. *Pyvips: a fast image processing library with low memory needs*. GitHub, 2015.

[8] Narayana Darapaneni, B.G Sudha, Anwesh Reddy, Ab Abdul Karim, Dhanalakshmi Marothu, Shirish Kulkarni, and Deepak Das Menon. Image classification of stroke blood clot origin. In *2023 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)*, pages 1–6, 2023.

[9] William Falcon and The PyTorch Lightning team. Pytorch lightning.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[11] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016.

[12] Koushik Sivarama Krishnan, P. J. Joe Nikesh, M. Logeshwaran, G. Senthilkumar, and D. Elangovan. Automated classification of stroke blood clot origin using whole-slide digital pathology images, 2023.

[13] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019.

[14] OpenSlide Project. Openslide: A c library for reading whole slide images, 2023.

[15] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 6105–6114. PMLR, 09–15 Jun 2019.

VII. Contribution

Ahmed Hassan, responsible for building the ResNet Model, choosing its hyperparameters, and applying the results part(test data).

Habiba Fathallah, responsible for implementing different preprocessing techniques.

Rawan Mohamed, responsible for data exploring and related work part.

Romaisaa Saad, responsible for building EfficientNet(expriement B0, B2, B4 versions) and SqueezeNet(Fire Module with batch normalization)Models, and choosing their hyperparameters.