Classification and Segmentation of Stroke Regions

Luu Linh Ly[1], Do Thi Huong Tra[2], Vu Hoang Mai Nhi[3],Vu Hai Thien Long[4], Le Thuan Ninh[5]
[1]22BI13269, LyLL.22BI13269@usth.edu.vn
[2]BA12-174, TraDTH.BA12-174@st.usth.edu.vn
[3]22BI13352, NhiVHM.22BI13352@usth.edu.vn
[4]22BI13268, LongVHT.22BI13268@usth.edu.vn
[5]22BI13354, NinhLT.22BI13354@usth.edu.vn

***Abstract*—This paper presents a multi-task approach for stroke diagnosis and segmentation in medical images, using the "İNME VERİ SETİ" dataset. We developed and evaluated three different tasks to solve two main phase: stroke detection and stroke type classification, as well as accurate segmentation of brain lesions. In this paper, we experimented with a wide range of deep learning models and achieved the best results in recall with 96.43, 96.35 respectively for the task classify stroke and classify type of stroke. The two models demonstrated high performance in classification and segmentation, achieving significant accuracy and reliability. However, the third model showed slightly lower results than the previous two models with f1 score 65.03, demonstrating the complexity of optimizing models for diverse tasks. The research results not only demonstrate the potential of deep learning in medical imaging but also provide important insights into choosing the right model for multi-objective problems in the field of medicine.**

***Index Terms*—Introduction, Methodology, Results, Discussion.**

I. Introduction

Stroke, a critical medical illness resulting from interrupted blood flow to the brain, continues to be one of the world's top causes of death and disability, presenting serious problems for communities and healthcare systems. Comprehending the risk factors linked to stroke is essential for early identification and prevention. As part of its related challenge, we investigate the "İNME VERİ SETİ", a dataset created especially to address important facets of stroke study. We approach the problem by resolving two key issues: (1) determining if a stroke has occurred, and (2) segmenting the stroke region and classifying whether the stroke was ischemic or hemorrhagic. The main objective of these tasks is to enable improved clinical decision-making by improving early detection and offering accurate insights into stroke characteristics.

II. Methodology

A. Dataset Exploration

This report uses the "İNME VERİ SETİ", which was provided as part of the Artificial Intelligence in Health competition during TEKNOFEST 2021 [1]. It consists of CT images in DICOM and PNG formats, collected and processed by medical professionals to aid in the diagnosis and analysis of areas affected by stroke.

The training set has 6951 brain CT images, of which 4557 images have no signs of stroke or only signs of chronic stroke, and 2394 images have signs of acute stroke or cerebral hemorrhage.

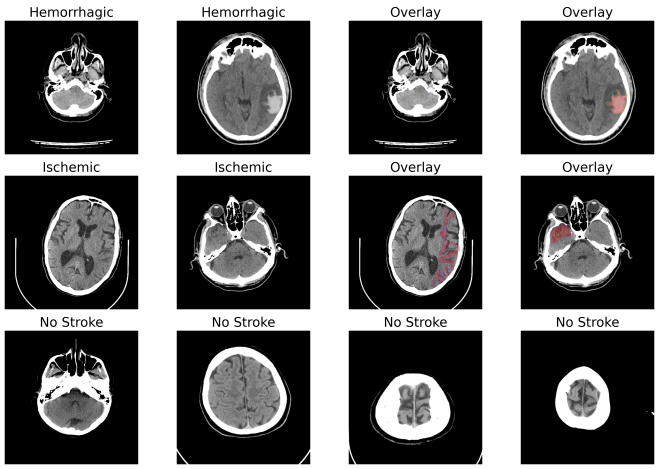


Fig. 1: Samples of training set for Hemorrhagic, Ischemic and No Stroke

After the competition ended, the test data for both phases was released. For phase 1, the data consisted of 100 images and had the same structure as the training data, including original and overlay images, along with the txt file store the labels stroke (1) and nonstroke (0).

For phase 2, in addition to the image and overlay, they provide additional information about the mask for 100 images. These mask images are divided into 2 types by the color of the mask part, blue for ischemic and green for hemorrhagic.

B. Phase 1: Stroke Detection

1) Data Preprocessing: Based on the test file, we organized our image data to indicate whether it showed a stroke or not. We had images classified as "Stroke" (which included both 'Kanama' - Hemorrhage and 'İskemi' - Ischemia types) and images classified as "No Stroke" ('İnme Yok'). We located all the PNG image files within these specific folders. We created a list of all image file paths and a corresponding list of labels, assigning the number 1 to all "Stroke" images and 0 to all "No Stroke" images.

We set aside 80% of the images for the training set and the remaining 20% for the validation set. **Importantly**, we used *stratify* during this split to make sure both the training and

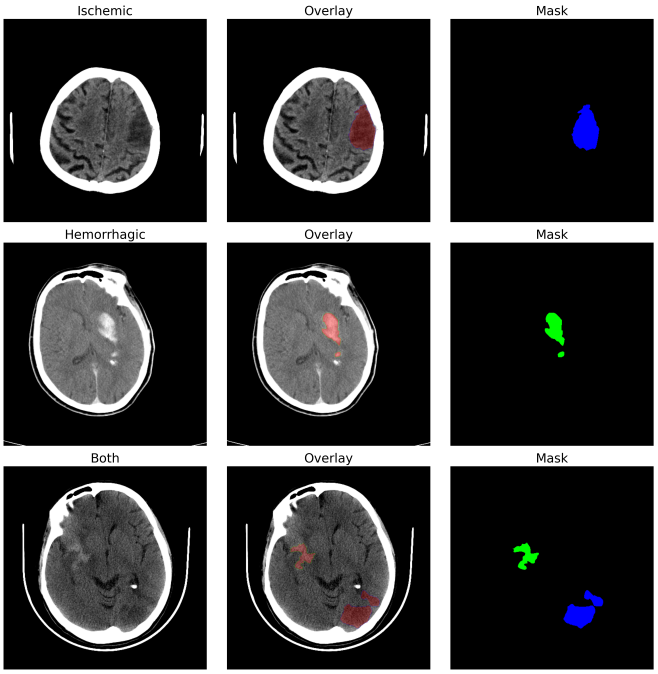


Fig. 2: Samples of phase 2's testing set

validation sets had roughly the same *percentage* of **"Stroke"** and **"No Stroke"** images as the original collection.

Each image was prepared for the model. We read the images in RGB format. They were then resized to the standard 224x224 pixels required by our model. After resizing, the pixel values were converted to a numerical format (float32).

For CNN model, we only do that, but for other models being used, another `preprocess_input` function was used to adjust pixel values (like scaling them) in the exact way the pre-trained model was originally trained, making it work much better.

Finally, to make training efficient, these processed images were grouped into batches (of 32), and techniques like prefetching (loading the next batch while the current one is being processed) were used to optimize training time, which is especially relevant for us when working with tf.data.

2) Model Training: We experimented with five models: `CNN`, `ResNet-18`, `MobileNetV2`, `EfficientNetB0`, and `EfficientNetV2B0`. Except for `CNN`, we used pre-trained models without their final classification layer (`include_top=False`) to adapt them for stroke detection. We fine-tuned their weights (`trainable=True`) and added custom layers:

- `GlobalAveragePooling2D` to condense extracted features
- `Dropout` ranges from 0.3 to 0.5 to prevent overfitting
- `Dense` layers with 128 units and `relu` activation for complex pattern learning
- Final `Dense` layer with `sigmoid` activation for binary classification problem

We compiled the model with the `Adam` optimizer, `binary_crossentropy` loss, and `accuracy` tracking, addressing class imbalance with class weights. Using `model.fit()`, we applied `ReduceLROnPlateau` and `EarlyStopping` callbacks during training. These parameters are tuned differently multiple times based on the performance of the models to achieve the best results.

The training process is completed in different epochs with ResNet2 being the smallest at 22 epochs and EfficientNetB0 being the largest at 72 epochs.

C. Phase 2: Type Classification and Segmentation

1) Data Preprocessing: We kept utilizing the "İNME VERİ SETİ" dataset and structured our image data based on whether the stroke was ischemic or hemorrhagic. We had images classified as **'Kanama'** - Hemorrhage (red overlay with **green border**) and images classified as **'İskemi'** - Ischemia types (red overlay with **blue border**). We located all the OVERLAY image files within these specific folders.

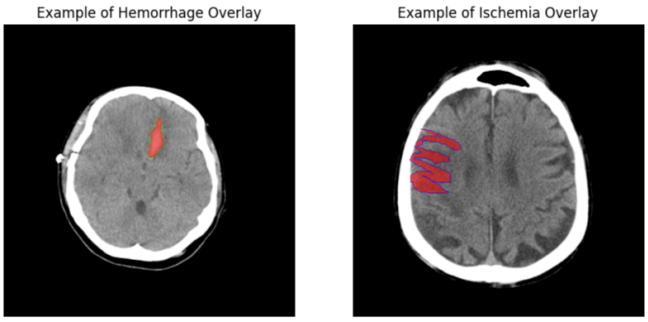


Fig. 3: Example Overlay of each Stroke types

We initially transformed all the OVERLAY images into binary masks by extracting the red regions in each images, which present stroke pathology areas. We use specific hue ranges (0-10 and 160-255 in PIL's HSV scale), along with saturation, and value thresholds for precise mask generation.

To handle large datasets efficiently, we also transformed the dataset into TFRecord format, a TensorFlow-specific format suited for large-scale machine learning operations. This step ensures efficient data loading and minimizes memory constraints.

We set aside 90% of the images for the training set and the remaining 10% for the validation set.

Each picture is scaled to 512x512 pixels to keep the homogeneity for model input.

2) Model Training: In the classification experiment, we trained three deep learning models to classify ischemic and hemorrhagic stroke cases using CT brain scan images from the INME VERI SETI dataset. We used only the PNG-format images from the ischemic and hemorrhagic folders, assigning label 0 for ischemic and label 1 for hemorrhagic cases. These images were preprocessed using standard ImageNet normalization and resized to 224×224 pixels to fit the input requirements of `ResNet-18` and `EfficientNet`. The dataset was randomly split into 80% training and 20% validation sets.

For the model architecture `ResNet-18`, we initial loaded without pretrained weights, and replaced its `final fully connected` layer to output two classes for binary classification. Additionally, we loaded custom pretrained weights into the network before `fine-tuning`.

For efficientNetV2B0, the base model was utilized without its top classification layer, initialized potentially without standard ImageNet weights by setting weights=None. Wr add thecCustom top layers including Global Average Pooling, Batch Normalization, Dropout, and a final Dense layer for the 2-class output. Crucially, specific preprocessing required by EfficientNetV2 was applied during prediction, differing from the PyTorch normalization used for ResNet-18 and the custom CNN.

For CNN, we set up the network with multiple Convolutional, ReLU and Max-Pooling layers, followed by Flattening, Fully Connected (with Dropout for regularization) layers and a final 2-unit output layer for more efficient classification. The input images receive the same preprocessing as ResNet-18.

The models was trained using the `Adam optimizer` and a batch size of 32 for different epochs. During training, we tracked both loss and classification accuracy. The entire training process was executed on GPU when available to speed up computation. Finally, a separate test set containing Phase 2 images (with `placeholder labels`) was prepared to evaluate the model's forward inference after training.

In the next experiment, we trained 2 models to perform semantic segmentation of stroke lesions in CT brain images, distinguishing between ischemic and hemorrhagic regions. The training and validation datasets were stored in `TFRecord` format with `ZLIB` compression, where each record contained a 512×512 RGB image and its corresponding segmentation mask with three classes: 0 for background, 1 for ischemic, and 2 for hemorrhagic. We used the U-Net [2] architecture with `EfficientNetB3` [3] and `ResNet50` [4] as the encoder backbone, provided by the segmentation models library. The model's final layer used a `Softmax` activation to support multi-class segmentation. The preprocessing involved normalizing the pixel values to [0, 1], and the datasets were efficiently loaded using the TensorFlow `tf.data` API. The model was compiled with the `Adam optimizer`, a predefined Dice loss function sm.losses.dice_loss from the segmentation_model [5] library, and evaluated using `IoU Score` and `F1-Score`. Training was conducted for 10 epochs with a batch size of 2, using `ReduceLROnPlateau` and `ModelCheckpoint` callbacks to manage learning rate scheduling and save the best model. All training logs were recorded using `CSVLogger` for further analysis. This setup aimed to enable pixel-level localization of stroke types and support clinical decision-making with fine-grained segmentation outputs.

III. Results

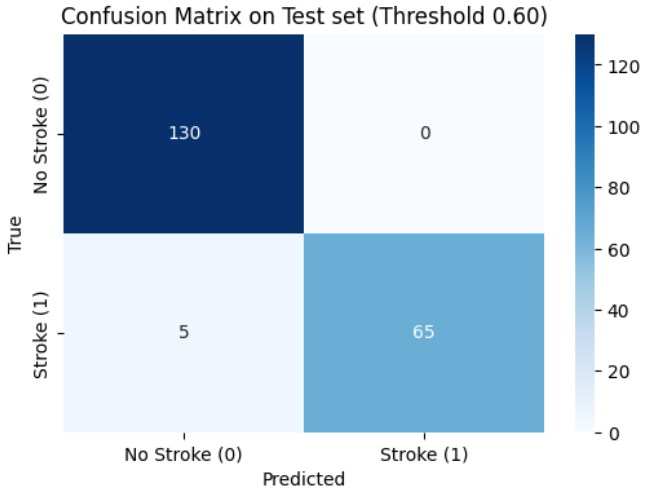
We will evaluate individual tasks and compare performance across different models.

A. Phase 1: Stroke Detection

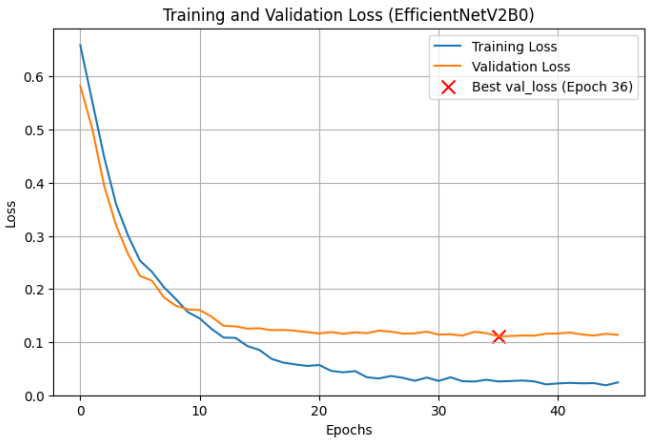
Since recall is considered the most important metric in disease detection problems, we changed the threshold to achieve a higher recall of 88% but with the condition of simultaneously controlling precision. Through experiments, the threshold level falls between 0.45 and 0.6 (very close to the default level of 0.5), demonstrating the stability and reliability of the models.

TABLE I: Comparison of Model Performance Metrics (Phase 1 Detection).

Model/Metrics	Precision	Recall	F1-score
EfficientNetV2B0	98.15	96.43	97.20
ResNet50	94.35	92.42	93.27
Custom CNN	94.90	91.04	92.56
EfficientNetB0	93.22	90.27	91.49
MobileNetV2	91.17	91.00	90.82



(a) Confusion matrix



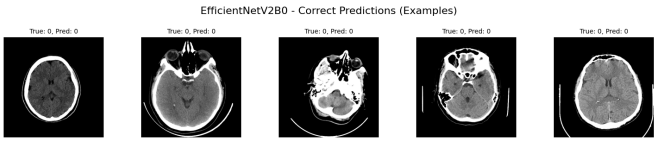
(b) Validation loss

Fig. 4: Performance of the best model EfficientNetV2B0 (Phase 1)

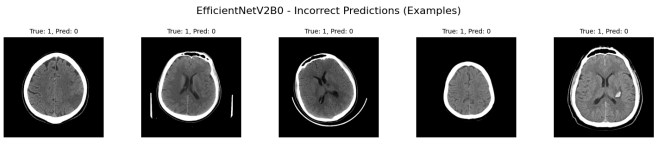
Observing the two figures, we see that only 5 stroke cases are actually missed (predicted as non-stroke). This is the only type of error that the model makes at this threshold. The model

has achieved perfect Precision, while still maintaining a very high Recall, showing that the model has a good ability to separate the two classes. On the other hand, we see that this strong model learns quite quickly, applying the Early Stopping mechanism is a necessary choice to choose the optimal point before overfitting becomes serious.

We also present some correct and incorrect prediction samples of the model below.



(a) Correct prediction



(b) Incorrect prediction

Fig. 5: Prediction sample of the best model EfficientNetV2B0 (Phase 1)

B. Phase 2: Type Classification and Segmentation

1) Classification: The classification task in this phase is evaluated similarly to the problem in phase 1. We measure important metrics such as precision, recall, f1 on validation set and show them in the table.

TABLE II: Comparison of Model Performance Metrics (Phase 2 Classification).

Model/Metrics	Precision	Recall	F1-score
Custom CNN	98.60	96.35	97.46
EfficientNetV2B0	97.20	94.98	96.07
ResNet-18 (pretrain)	98.45	90.52	94.32

The results show that the best performing model is custom CNN, we will show in this report the training process and some prediction samples as follows

2) Segmentation: The final performance of the `UNet` with `ResNet50` backbone segmentation model can be considered moderate higher than with `EfficientNetB3`.

TABLE III: Comparison of Model Performance Metrics (Phase 2 Segmentation).

Model/Metrics	F1-score	IOU-score
UNet-ResNet50	0.6554	0.55
Unet-EfficientNetB3	0.52	0.43

The training F1-score improved steadily, reaching 0.6554, while the validation F1-score peaked at 0.6503, indicating that the model was able to learn meaningful features. However, the validation metrics fluctuated across epochs, and the slight drop in performance during the final epoch suggests limited generalization capability. This may hint at potential overfitting

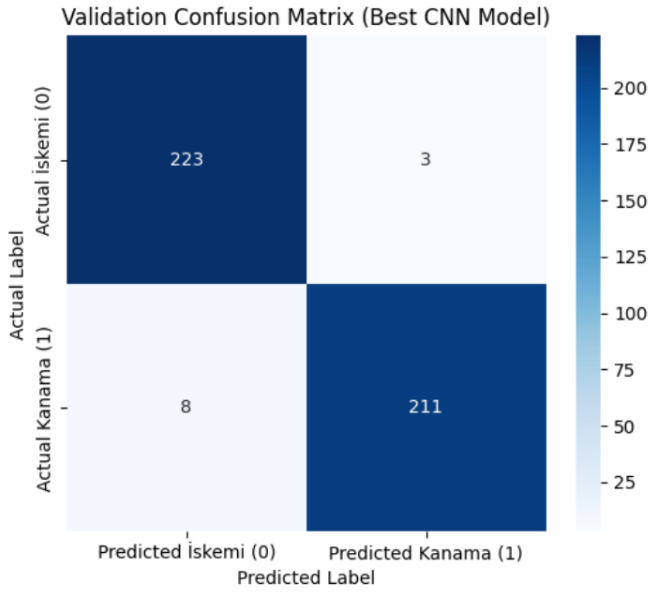


Fig. 6: Confusion matrix (Phase 2 Classification - Custom CNN)

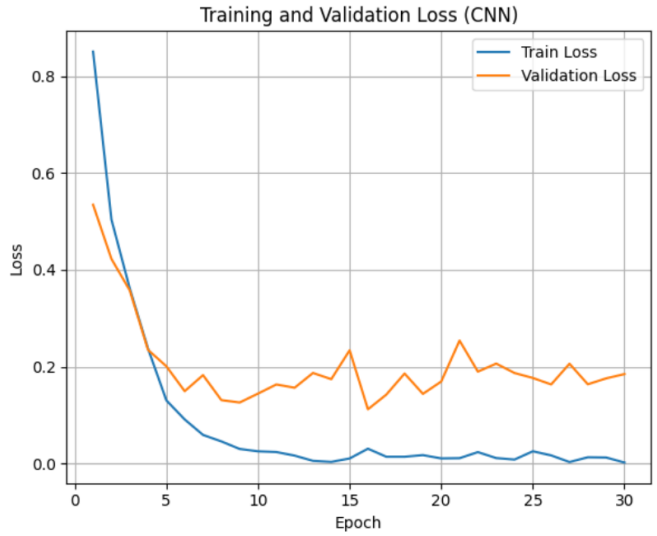
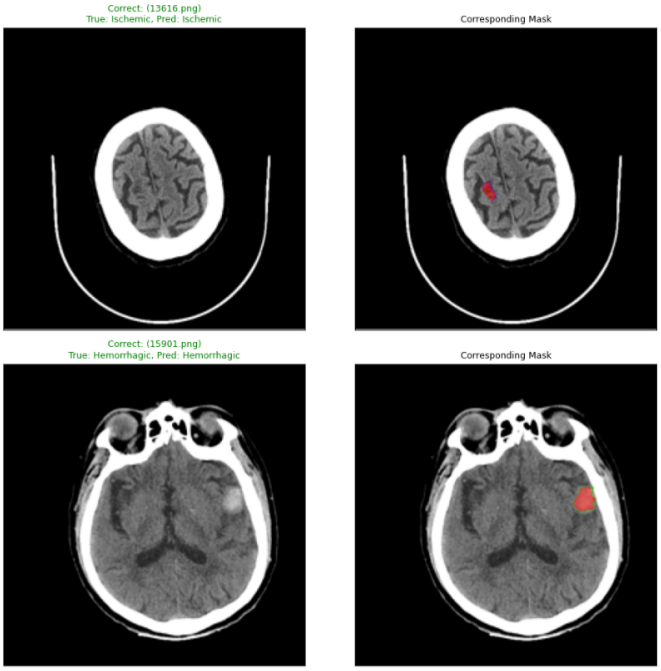
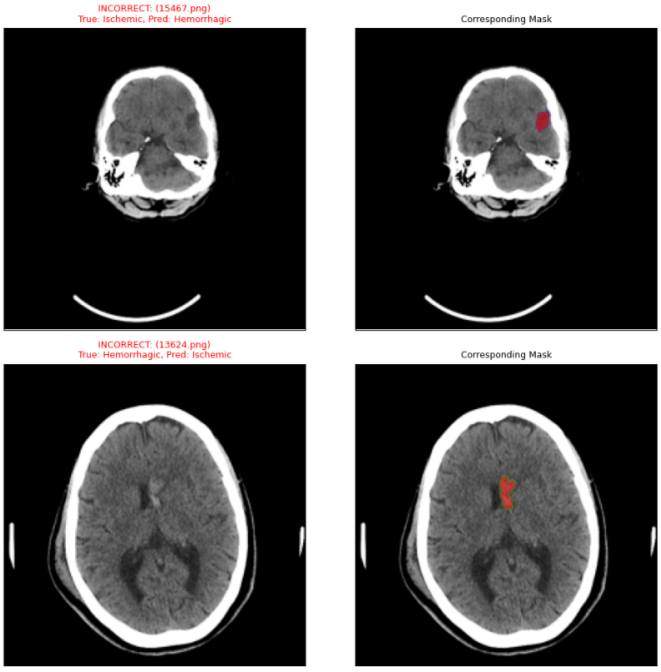


Fig. 7: Train and validation loss (Phase 2 Classification - Custom CNN)

or sensitivity to variations in the validation data. Overall, while the model has demonstrated a reasonable ability to segment stroke lesions, its robustness and performance on unseen cases remain limited.

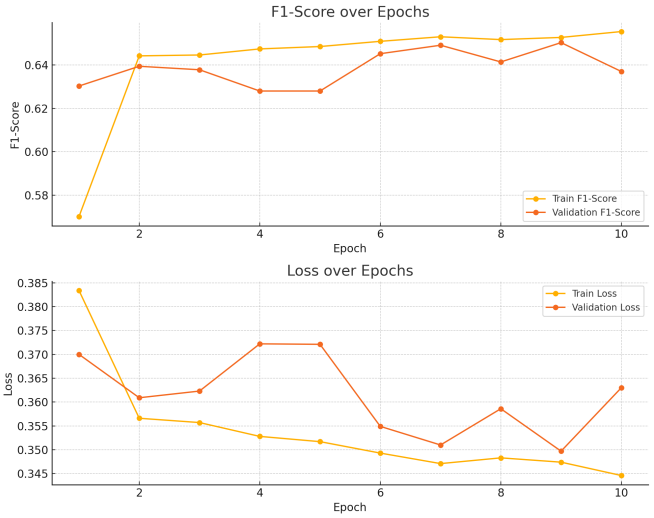


(a) Correct prediction

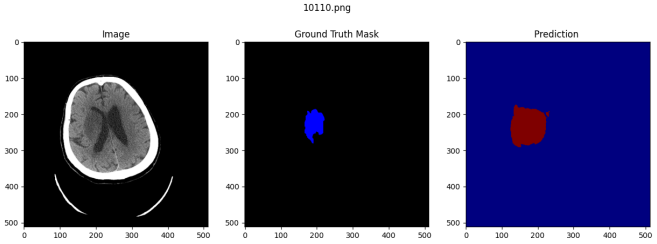


(b) Incorrect prediction

Fig. 8: Prediction sample of the best model Custom CNN (Phase 2 Classification)



(a) Graph of loss over epochs



(b) Segmentation Demo

Fig. 9: Performance of the best segmentation model (Phase 2 Segmentation - UNet-ResNet50)

IV. Discussion

In both detection and classification tasks, our results demonstrated strong results in important metrics such as recall and f1 all greater than 90. This result shows that we have succeeded in identifying the stroke factor, the low recall index shows promise in that the model will miss fewer cases with stroke signs, which is a prerequisite for putting the model into practical use.

However, analysis of the loss curves revealed concerning signs of overfitting despite many techniques being applied. Most notably, our augmentation approach had surprising outcomes: the more augmentation is applied, the earlier overfitting occurs, and the worse the performance on the test set becomes. This indicates that our augment method is not suitable for the model and needs to be adjusted in future work.

As for the segmentation task, our work achieved decent but not outstanding results. This may be because we implemented two separate models for phase 2 to reduce complexity. In the future, we will combine these two models into one to learn features simultaneously and expect better results.

References

[1] U. Koç, E. Akçapınar Sezer, Y. A. Özkaya, Y. Yarbay, O. Taydaş, V. A. Ayyıldız, H. A. Kızıloğlu, U. Kesimal, Çankaya, M. S. Beşler, E. Karakaş, F. Karademir, N. B. Sebik, M. Bahadır, Sezer, B. Yeşilyurt,

S. Varlı, E. Akdoğan, M. M. Ülgü, and Birinci, "Artificial intelligence in healthcare competition (teknofest-2021): Stroke data set," *The Eurasian Journal of Medicine*, vol. 54, no. 3, pp. 248–258, 2022. [Online]. Available: https://doi.org/10.5152/eurasianjmed.2022.22096

[2] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *MICCAI*, 2015.

[3] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *ICML*, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CVPR*, 2016.

[5] P. Yakubovskiy, "Segmentation models," https://github.com/qubvel/segmentation_models, 2019.