Computer Vision and Deep Learning 2021

LSTM Based Neural Network for Brain Stroke Detection

Artem Bakhanov and Anna Boronina

Innopolis University

December 12, 2021

1 Abstract

We present a neural network to classify brain MRI scans as "with brain stroke" and "without brain stroke". We use two datasets that contain images sequences with and without stroke. The model has Convolutional, Long Short-Term Memory (LSTM), and Feature Similarity Module (FSM) layers to classify sequences of images. The results of our model are close (and in some cases better) to the results of state-of-the art models such as XNet [1] and D-UNet [2].

2 Introduction

2.1 Background Information

Brain stroke is a condition when some brain parts do not receive enough blood. Every year, 795,000 Americans have a stroke, and it might be hard to catch first symptoms for two reasons: it requires doing an MRI scan and it might be hard to identify it from the scans, as it can be of any size. That is why researches start to offer Deep Learning and Computer Vision techniques to detect brain stroke and possible regions.
As Computer Vision and Deep Learning are gaining more trust by becoming more interpretable, World Health Organization are starting to accept the possibility to use Deep Learning models as a complementary tool in medicine. We can use powerful convolutional and recurrent neural networks to capture local and global features of any input.
Our projects aims to classify sequences of MRI scans from two different datasets. We compare our results to state-of-the-art architectures: XNet [1] and D-UNet [2].

2.2 Structure

First, we show a timeline of the process. Then we move on to methodology which includes data acquisition, analysis, preprocessing, augmentation steps, as well as architecture explanation. Everything we claimed is supported by tables and figures. Then we show the statistics of the results we obtained. In the next section we interpret the results and compare them to state-of-the-art models. Then we outline possible areas for improvement and change. Finally, we write a short conclusion that reflects our learning outcomes and acknowledge the help from our instructor.

3 Timeline

1. We had read about the topic and what kinds of brain representations there exist. MRI scans are considered the most informative ones, as it shows 3D model of a brain.

2. We decided to use a sequence of images for classification because we did not find a paper that aims to classify or detect brain stroke from a *sequence* of images.

3. We decided to use LSTM to process a sequence of images after using 3 convolutional layers to extract features. This way, LSTM gets to see the feature map instead of the original images.

4. We found two datasets: one from Kaggle and ATLAS. The first one is not trust-worthy, but we started with it because it contains .jpg images, whereas the second one contains .nii.gz files.

5. We tuned the following hyperparameteters:

   - sequence length: 5, 10, **15**, 25.
   - batch size: 2, 4, 8, **16**.
   - image size: **64**, 96, 128.
   - channels in the convolutional layers and number of convolutional layers.
   - others: learning rate, weight decay, etc.

6. After starting to get satisfying results we decided to try our model on the ATLAS dataset.

7. At the same time we discovered lightning-hydra-template and rewritten our code to match the template.

8. We saved the model and hyperparameters that worked best for Kaggle dataset and tried to apply it to the ATLAS dataset. It did not give good results.

9. Finally, we used W&B tool to track training, visualize dataset and save training and testing metrics.

4 Methodology

4.1 Data Acquisition, Augmentation, and Visualization

We used two datasets: Enhanced Kaggle dataset, ATLAS dataset.

Train/Test	Normal/Stroke	Num. of sequences	Sequence length
train	normal	51	avg: 27, min: 18, max: 40
train	stroke	31	avg: 26, min: 9, max: 35
test	normal	16	avg: 7, min: 1, max: 12
test	stroke	13	avg: 8, min: 3, max: 19

Table 1: Kaggle Dataset: original statistics

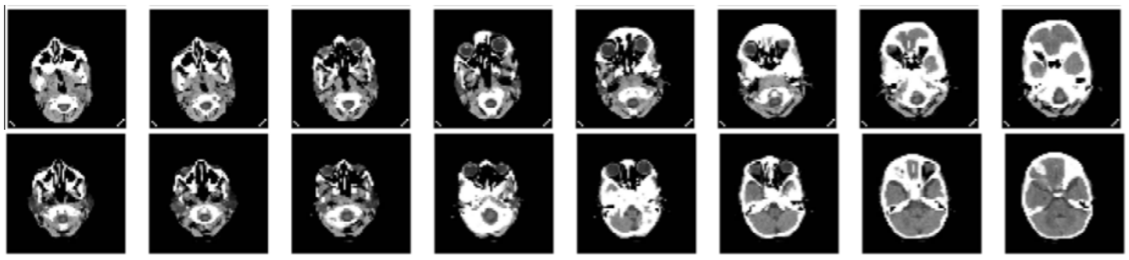

Figure 1: Kaggle dataset: original data

4.1.1 Train Augmentations

- Resize to 64x64;
- Gaussian Noise, filters from 1 to 7;
- Gaussian Blur, filters from 1 to 5;
- Grid Distortion, 3 steps, probability=0.3;
- Optical Distortion, probability=0.4;
- Coarse Dropout, probability=0.5;
- Normalization mean=0.5, std=0.3;

4.1.2 Test Augmentations

- Resize to 64x64;
- Normalization mean=0.5, std=0.3;

We also implemented rotation from -180 to $+180$ degrees. Nevertheless, training on such data was impossible: neither train nor test losses were not increasing. Possible fix would be to use a smaller angle, e.g., 30 degrees. After all, we found this augmentation unnecessary, because all MRI images are usually rotated the same way.

4.2 Dataset from Kaggle

We changed the order of some sequences, as it was obvious that they were shifted. We removed too short sequences from train set. Table 1 presents the statistics of the dataset.
Fig. 1 presents the original data visualization. The training set examples are presented in Fig. 2. Finally, Fig. 3 shows the test set examples.

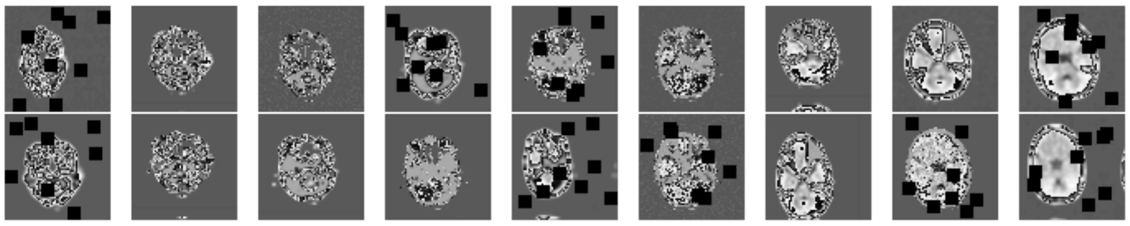
Figure 2: Kaggle dataset: train data

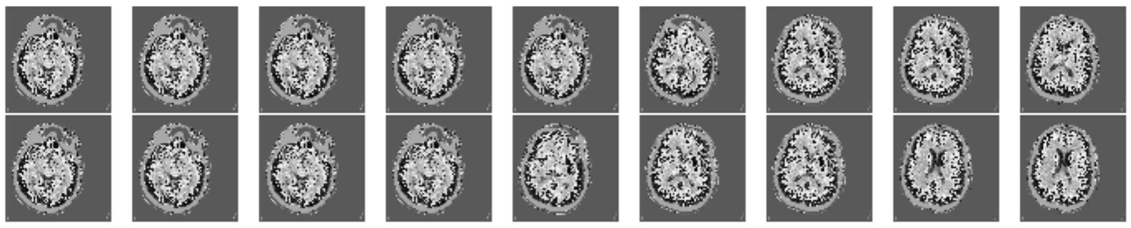
Figure 3: Kaggle dataset: test data

4.2.1 Preprocessing

After loading the dataset, we sampled sequences of length 15. When there was not enough images for 15, we replicated some of them. When there were too many images per patient, we created several sequences from that patient's data by sampling random images every time. Both repetition and extra sampling preserve the order of the images. Table 2 presents the statistics of the dataset after we performed the sampling. The table implies that the dataset is unbalanced which definitely had an effect on training and testing results.

4.3 ATLAS Dataset

ATLAS dataset [3] is a dataset of 220 3D brain models and 3D masks for brain stroke. Brain models are a reconstruction from MRI scans. One brain model can contain more than 1 region with brain stroke, and all the brain stroke regions have different sizes.
Fig. 4 shows the advantage of MRI scans and BrainViewer tool.

Train/Test	Normal/Stroke	Num. of sequences
train	normal	661
train	stroke	381
test	normal	240
test	stroke	184

Table 2: Kaggle Dataset after getting sequences of 15 images

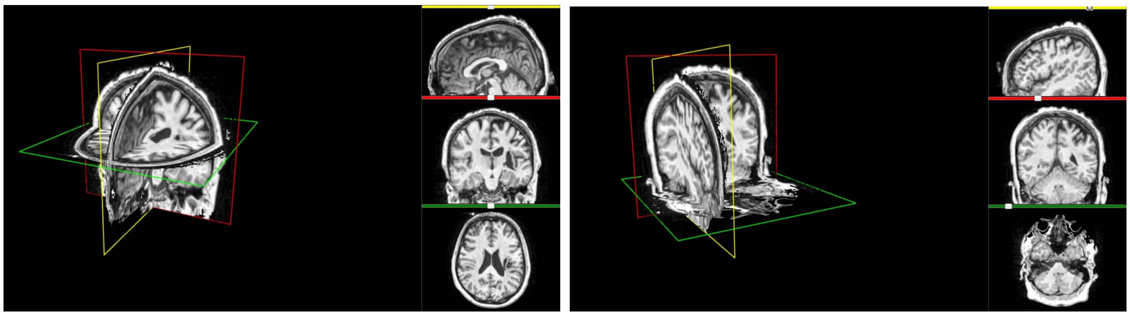
Figure 4: BrainViewer: ATLAS dataset

Train/Test	Normal/Stroke	Num. of sequences
train	normal	72
train	stroke	72
test	normal	18
test	stroke	18

Table 3: ATLAS dataset after getting sequences of 15 images

4.3.1 Sequences extraction

To extract sequences from 3D models, we did the following:

1. Read *.nii.gz* files with SimpleITK Python library and transform them to numpy arrays.
2. By analyzing 3D masks indicating the places of brain strokes, we extract two parts of brain slices: with and without brain stroke.
3. We sample from each chunk 15 2D slices twice. Therefore, from each patient we get 2 sequences, each with 15 images.
4. Some patients were excluded from sampling, since their stroke masks had less than 15 slices. Overall, we excluded 35 samples.

Table. 3 shows statistics for ATLAS dataset after we performed sequences extraction. Note that ATLAS has approximately 10 times less data, but this one is a balanced, unlike Kaggle dataset.
Fig. 5 shows an example of a sequence extracted from a 3D brain model.

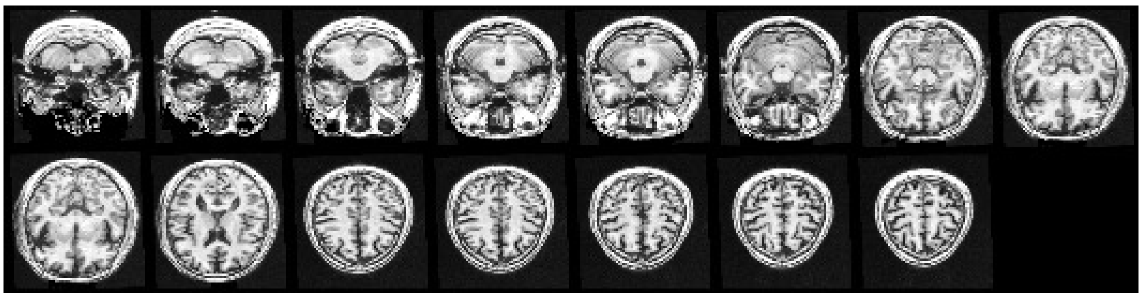
Figure 5: ATLAS dataset: one sequence

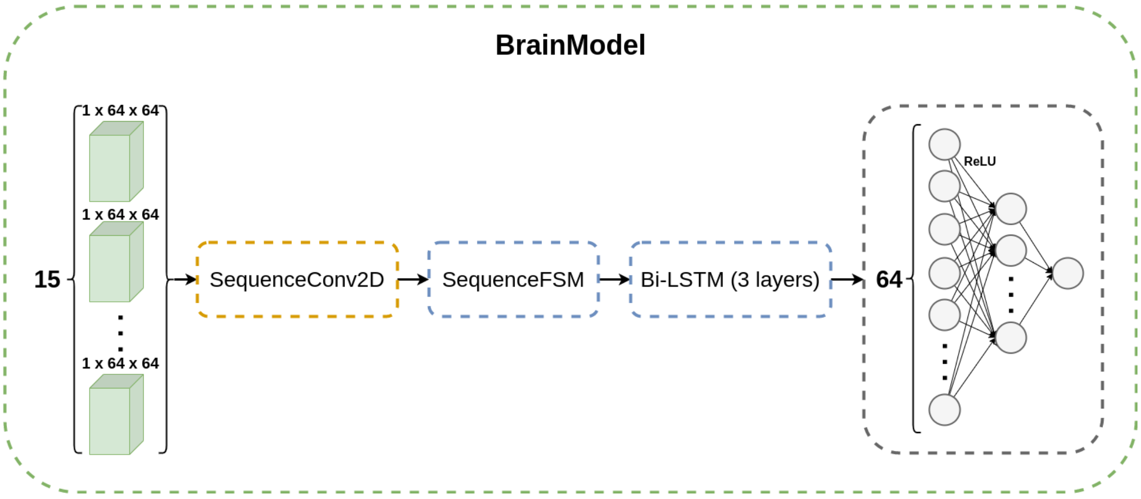

Figure 6: Architecture

Possible issue with the approach described above is that we do not have a sequence in which there are images of both types: with and without a stroke. So, by training on such dataset, the model would perform poorly on cases when the stroke is spanned across 5 out of 15 images.
The source code for the preprocessing of the dataset.

4.4 Architecture

Fig. 6 shows a full architecture of the model. The main idea is that three-layered Bi-LSTM receives feature maps instead of the original sequence. Hidden dimension contains 64 neurons that are mapped into 32 neurons and then to 1.
SequenceConv2D and SequenceFSM are two feature extraction blocks, and Fig. 7 shows their internal architecture. Finally, Fig. 8 shows two most basic blocks: FSM [1] and BasicModule. The output shape of FSM is the same as its input shape, therefore it can be plugged into any part of the model. BasicModule is a simple sequence of Convolution, BatchNormalization, and ReLU activation.
The architecture for FSM is taken from X-Net architecture [1]. The authors present this module as a visual attention module that can capture long-range dependencies by generating an attention matrix.

4.5 Hyperparameters

Optimizer
We use AdamW optimizer with learning rate 0.001 and first and second momenta $(0.9, 0.999)$. We tried Adam and Adamax, but we did not notice any difference.
Loss Function
For loss function we use Binary Cross-Entropy With Logits, that is why the model output is a single neuron without any final activation.
Hidden Dimension
The hidden dimension has 64 neurons. We tried 32, 48, 64, 96, and 128, yet 64 showed the best result when used after convolution where the last number of channels is 48.

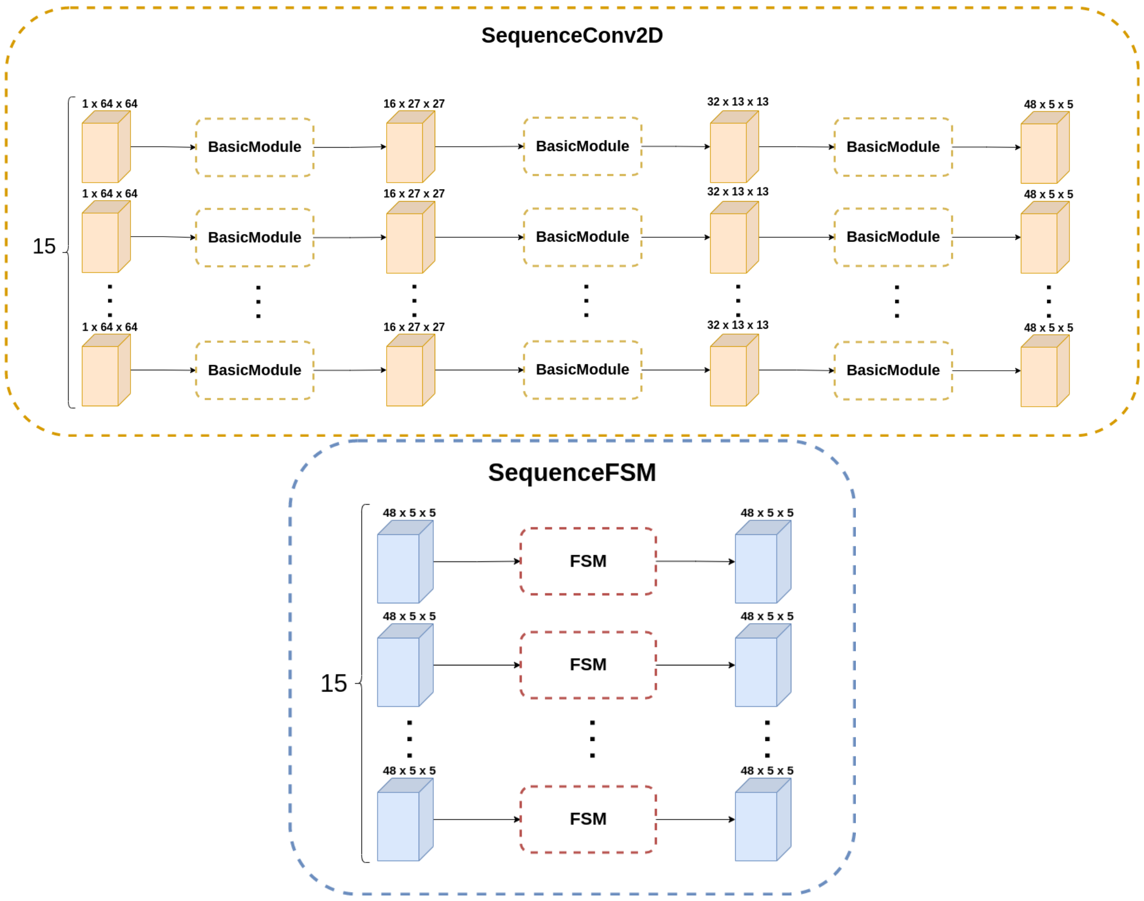

Figure 7: Sequential Blocks: Convolutional and FSM

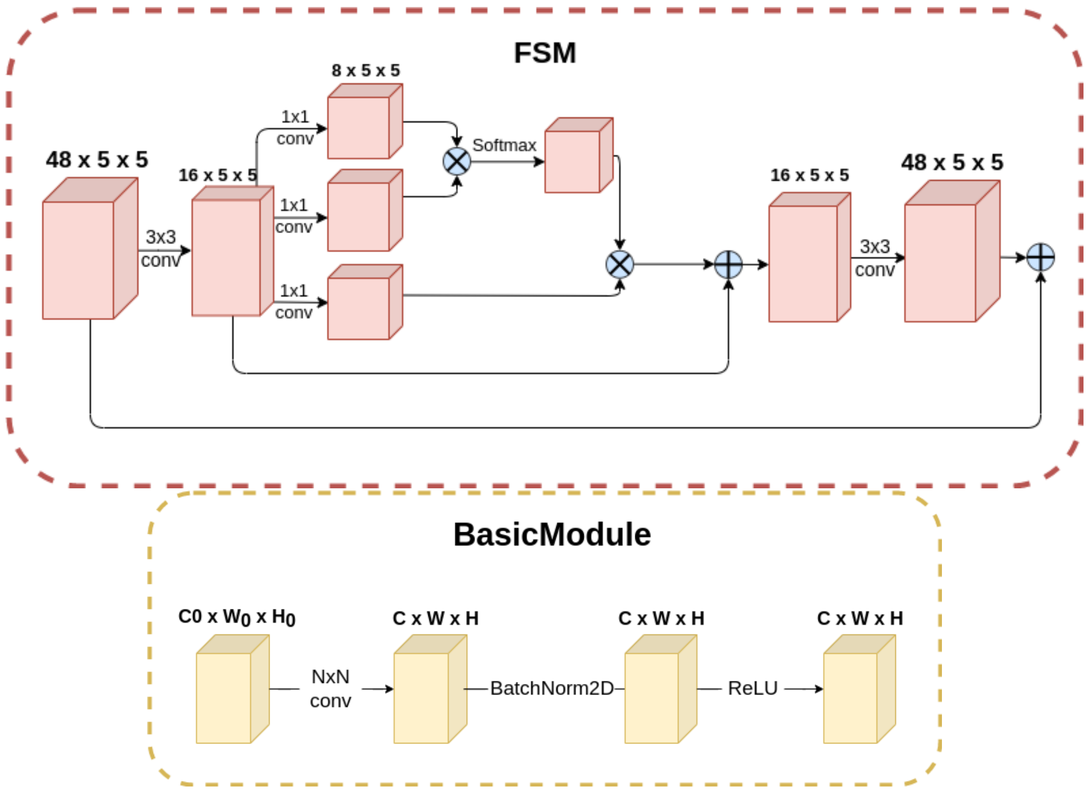

Figure 8: Basic Blocks: Convolutional and FSM

LSTM vs. GRU
We also tried to use Bi-GRU instead of Bi-LSTM, but it performed poorly. We also tried different number of layers for Bi-LSTM: 1, 2, 3, 5. The best results were obtained with 3 layers of Bi-LSTM.
Activation function
We tried SELU and ReLU, and ReLU showed better performance.
Sequence Length and Image Size
We tried different sequence lengths: 5, 10, 15, and 30; and different image sizes: 64, 96, 128. The best pair according to our heuristic search is (15, 64). Tuning the sequence length for Kaggle dataset is especially difficult, since all sequences there have different lengths.
Augmentation
We tried putting Coarse Dropout before and after Normalization. It was difficult to estimate which order gave better results, but the final version has Coarse Dropout after Normalization.
Number of epochs
We trained the model for 20 epochs. More epochs just led to the model overfitting.

4.6 Stack and Tools

- Google Colab - online Jupyter Notebook with GPU; we used it to tune initial hyperparameters; the link leads to our best model performance.
- pytorch lightning - high-performance wrapper for PyTorch allowing to reduce boilerplate.
- hydra - easy config manipulation.
- wandb - dashboard for model training and performance log visualization.
- lightning-hydra-template - template for project with pytorch-lightning and hydra.
- GitHub - project maintenance and collaboration.

4.7 Source Code

The link to GitHub
Google Colab Notebook (not as up-to-date as GitHub Repository)

5 Results

5.1 Kaggle Dataset

To view complete statistics of training and testing on Kaggle dataset, follow this link. Table 9 shows test metrics throughout 15 epochs (1.4k steps). Table. 4 shows best numbers for four metrics.

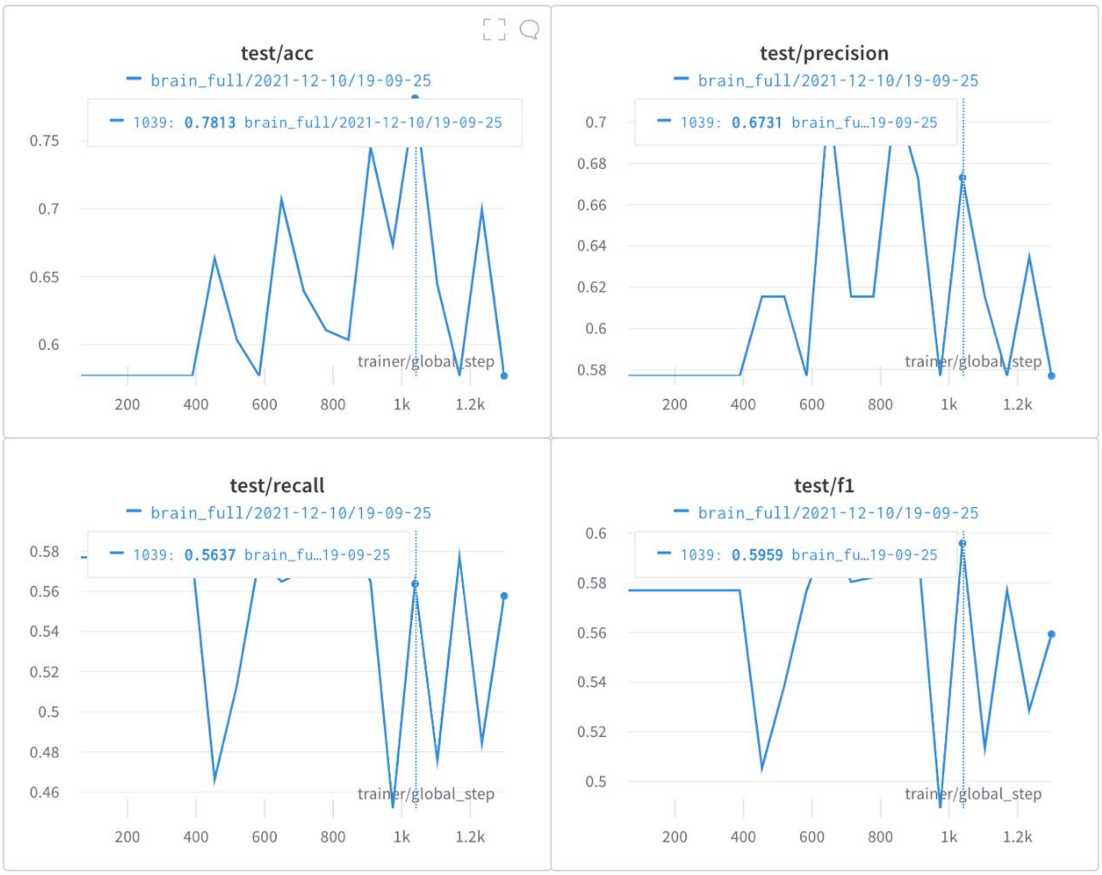

Figure 9: Results of training on Kaggle dataset

Epoch	Accuracy	Recall	Precision	F1-score
9	0.71	0.56	**0.71**	**0.60**
12	0.60	**0.59**	**0.71**	**0.60**
15	**0.78**	0.56	0.67	0.6

Table 4: Kaggle dataset: test statistics

Epoch	Accuracy	Recall	Precision	F1-score
1	0.56	0.41	**0.56**	0.44
13	**0.72**	0.49	0.51	0.46
14	0.56	**0.75**	0.53	**0.56**

Table 5: ATLAS dataset: test statistics

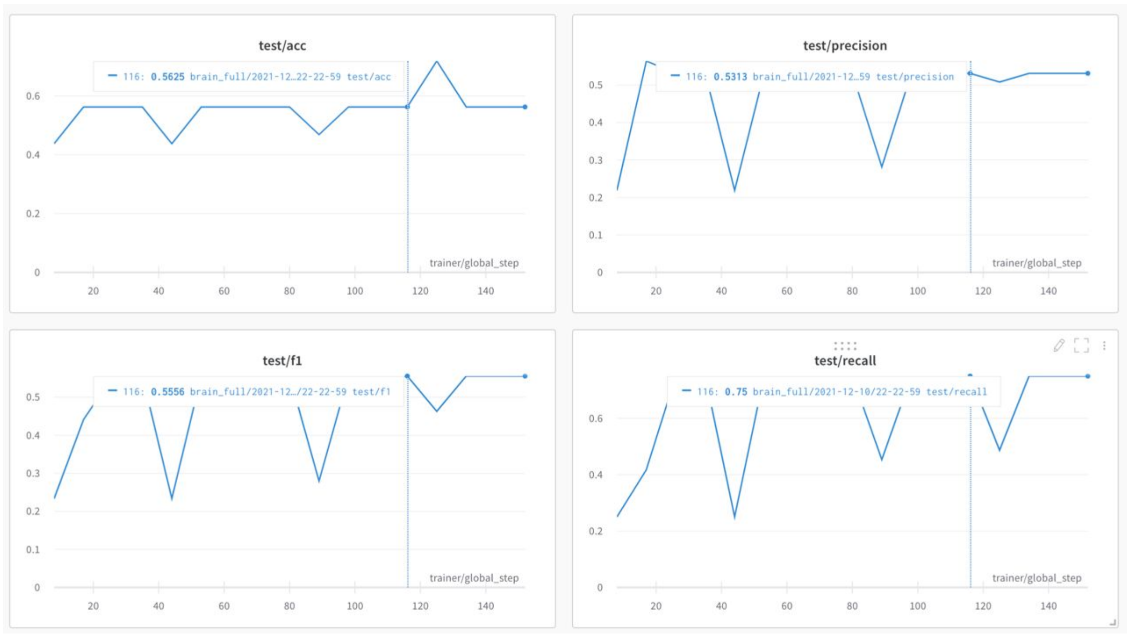

Figure 10: Results of training on ATLAS dataset

5.2 ATLAS Dataset

To view complete statistics of training and testing on ATLAS dataset, follow this link. By following it, you will see the training graphs as well which prove that the model overfitted after the second epoch.
Table. 10 shows test metrics throughout 15 epochs (1.4k steps). Table 5 shows the best numbers for four metrics. Nevertheless, since the model overfitted, we do not consider test statistics meaningful.
In the next section we discuss possible reasons for these results.

6 Discussion and Further Work

6.1 Results Interpretation

For medical images, recall is the most important metric, since it represents a number of elements that were successfully retrieved. In other words, we want to maximize recall, so that our model does not label a patient with a stroke as a stroke-free patient.
The highest recall for Kaggle dataset is 0.59, whereas the highest recall for ATLAS dataset is 0.75, which is higher that state-of-the-art models' recall for this dataset. Note that recall of 0.75 stays the same until the end of training. It might mean that it's the highest recall score the model could get and that this number did not happen by pure luck.
Table 6 shows comparison between our model performance on ATLAS Dataset and other state-of-the-art models [1] [4]. Note that these models performed detection of brain stroke not on sequences but on single images.

Model	Recall	Precision	Num. of parameters
X-Net	0.48	0.60	15.1M
U-Net	0.44	0.60	34.5M
D-Unet	0.52	**0.63**	-
Ours	**0.75**	0.53	**1.9M**

Table 6: ATLAS dataset benchmark

6.2 Further Work

We outline several possible areas to work on to improve the performance below. Note that all of them concern the ATLAS dataset, as we believe that Kaggle dataset is not trust-worthy.

- Increase the size of ATLAS dataset by sampling 10-15 sequences from each brain model;
- Combine slices with and without stroke in one sequence because it is closer to real-world scenarios;
- Sample slices that are close to each other according to some threshold;
- Tune the sequence length parameter; try to run the model with sequence length equals 1;
- Check if the ideas above reduce overfitting of the model;
- Create an environment for automatic hyperparameters tuning, e.g., Hydra;
- Run an interpretation algorithm;

Conclusions

The amount of time we had was not enough to go through all the areas of improvement and try all datasets permutations. Nevertheless, we are satisfied with the results and with the fact that we can explain them.
Besides that, we have learned new tools such as hydra, pytorch-lightning, and wandb.

Acknowledgements

We would like to acknowledge the contribution of our lecturer, Imad Eddine Ibrahim BEKKOUCH. He supported the project and suggested ideas on how to process a sequence of images.

References

[1] K. Qi, H. Yang, C. Li, *et al.*, "X-net: Brain stroke lesion segmentation based on depthwise separable convolution and long-range dependencies", *Medical Image Computing and Computer Assisted Intervention – MICCAI 2019*, pp. 247–255, 2019. DOI: 10.1007/978-3-030-32248-9_28. [Online]. Available: http://dx.doi.org/10.1007/978-3-030-32248-9_28.

[2] Y. Zhou, W. Huang, P. Dong, Y. Xia, and S. Wang, "D-unet: A dimension-fusion u shape network for chronic stroke lesion segmentation", *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. PP, pp. 1–1, Sep. 2019. DOI: 10.1109/TCBB.2019.2939522.

[3] S.-L. Liew, *The anatomical tracings of lesions after stroke (atlas) dataset - release 2.0, 2021*, 2021. DOI: 10.3886/ICPSR36684.v4. [Online]. Available: https://doi.org/10.3886/ICPSR36684.v4.

[4] *Papers with code - anatomical tracings of lesions after stroke (ATLAS) benchmark (lesion segmentation)*, https://paperswithcode.com/sota/lesion-segmentation-on-anatomical-tracings-of?metric=Recall., Accessed: 2021-12-11.