HOHOHO: intracranial HemOrrHage detectiOn enHenced by asymmetric lOss with CNN-LSTM

Ming-Yang Ho
何明洋
r08945027
Graduate Institute of Biomedical Engineering and Bioinformatics
r08945027@ntu.edu.tw

Bin-Ray Wu
吳彬睿
r08942087
Graduate Institute of Communication Engineering
r08942087@ntu.edu.tw

Hsin-Yu Ho
何欣育
r08946005
Data Science Degree Program
r08946005@ntu.edu.tw

Si-Yang Jiang
蔣思陽
r08921098
Graduate Institute of Electrical Engineering
r08921098@ntu.edu.tw

1 Introduction

Cerebral hemorrhage, bleeding that occurs around or within the brains, is a serious health problem requiring rapid and often intensive medical treatment. The cerebral hemorrhage can be divided into 5 categories: Intracerebral hemorrhage (ICH), Intraventricular hemorrhage (IVH), Subarachnoid hemorrhage (SAH), Subdural hemorrhage (SDH), Epidural hemorrhage (EDH). While the diagnosis requires an urgent procedure, the process is complicated and often time-consuming. Herein, this problem is attempted to be solved by learning-based methods.

2 Methodology or Model Architecture

2.1 Preprocessing

Each CT image was firstly stacked with two aside ones in the preprocessing step to extract more information owing to the property of sequential CT scanning **(Figure 3 in appendix)**. Limited augmentation strategies, rotation with little color adjustment, were utilized to avoid interfering with the intrinsic CT data distribution.

2.2 Models

2.2.1 Big Dataset: CNN-LSTM

ResNet-18 was utilized as the features extraction backbone trained with asymmetric loss. Warmup steps were leveraged to avoid unstable model initialization during the previous training, and cosine-annealing learning rate scheduling was applied, which let the model have chance to jumpout, if it got stuck into local minimum.

Finally, a stacked LSTM architecture trained with BCE loss utilizing 512-dim embedding from ResNet-18 was leveraged to further enhance the overall performance. The effect of LSTM model is to get the temporal information, which thoroughly considers a patient at the same time. Shortcuts were made from every stage of LSTM output and than add to the final FC layer, which could solve

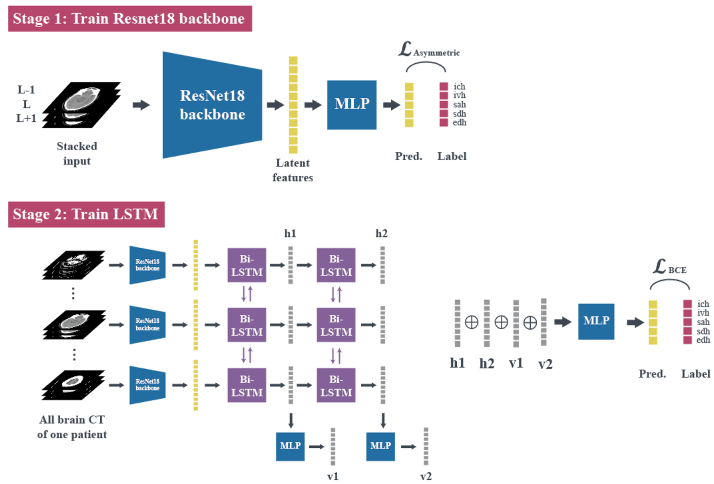


Figure 1: **CNN-LSTM:** ResNet18 backbone, with layer shortcut Bi-LSTM.

the gradient vanishing problems and make training converged faster **(Figure 4 in appendix)**.

2.2.2 Small Dataset: Self-supervised Learning with Backbone

SimCLR[1] is a method which uses data augmentations for distinguishing features in the latent space. Specifically, after the feature extractor, the contrast loss is defined, which help the positive features become more closely and the negative ones more far away from each other. Due to the label deficiency in the small dataset, this technique, which enables distinguishing the positive and negative samples without labels via appropriate data augmentations, was utilized to overcome this problem. Afterward, the pretrained weight from SimCLR was fine-tuned by the labels from the small dataset **(Figure 1)**.

2.3 Loss functions

Asymmetric Multi-label Loss (stage-1)

$$\begin{cases} \mathcal{L}_+ = (1-p)^{\gamma_+}\log(p), \\ \mathcal{L}_- = p^{\gamma_-}\log(1-p). \end{cases} \tag{1}$$

Asymmetric loss[2], an improved focal loss designed for unbalanced positive and negative samples training with a tunable parameter γ controlling the model to focus on positive samples, was utilized. It was found that the positive and negative portion of the dataset was 13:1, so $\gamma_+ = 0$ and $\gamma_- = 2$ were chosen. Furthermore, a clip design, which would prune the class if the confidence of the class was extremely excessive, would enable those uncertain classes having more opportunities to be trained and allow the model to get their features.

3 Implementation Details

3.1 Hyperparameter Choices during preprocessing

All the CT images were stacked into three channels. Random rotate(40), random horizontal flip and random color jitter (brightness=0.1, contrast=0.1, saturation=0.1, hue=0) with apply probability=0.4 strategies were selected for augmentation while in the valid setting, images were simply transformed into tensor.

3.2 Hyperparameter Choices during model training

Regarding the full dataset training, cosine-annealing learning rate scheduling was set from 2×10^{-4} to 10^{-5} with Adam optimizer and batch size was set to 48. During the LSTM training stage, the LSTM unit was set at 64 with batch size at a value of 32. For the small dataset, ResNet18 was also selected as the backbone. BCE loss with posweight and Adam were used as loss function and optimizer, respectively. Learning rate was set at 10^{-3} and the batch size was 48. During the SimCLR, 0.5 was selected as the temperature parameter.

4 Experiments

4.1 Results

		ResNet 18 (rotation 20)		ResNet 18 (rotation 40)	
Batch size		4	64	64	128
LSTM units		64	64	64	64
Stacked	Train f2	0.8907	0.8943	0.8957	0.8988
	Val f2	0.7714	0.7714	0.7917	0.7911
	Test f2	0.7792	0.7787	0.7764	0.7726

Table 1: Comparison results of stage 2 LSTM with different rotation angles

Dataset		Full	Small
Best	Train f2	0.8907	0.6928
	Val f2	0.7714	0.5971
	Test f2	0.7792	0.6450

Table 2: Best results

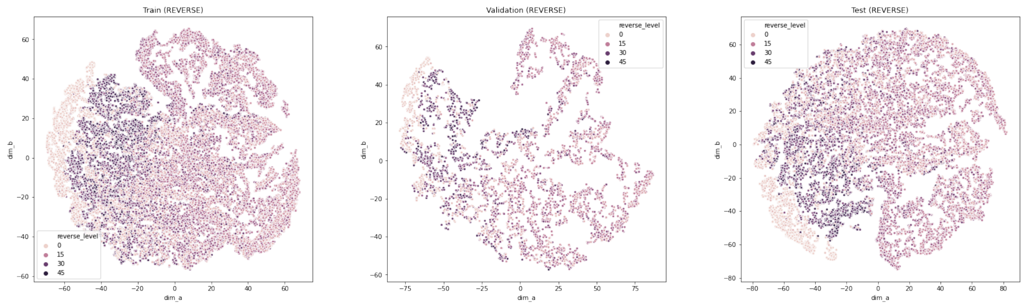
For the case of using full training dataset, multitudinous optimizers and augmentation parameters were firstly investigated with vanilla ResNet18 trained with BCE loss, which demonstrated the detrimental effect of either large batch size or color adjustment. However, model with BCE loss could only achieve at most 73.10% f2 score (**Table 3 in appendix**). If BCE loss was replaced with asymmetric loss, the model could achieved 2.50% f2 enhancement, compared to the former one. It showed that asymmetric loss is an effective strategy to further improve model performance (**Table 6 in appendix**). Moreover, two-stage training with stacked LSTM, what was mentioned in chapter 3, obtained the best performance with 78.25%. Comparing to the ResNet18 model which trained with asymmetric loss, it enabled approximate 2.75% f2 enhancement (**Table 1 and 2**). As for the case of using small training dataset, the self-supervised model obtained the best performance with 64.50% (**Table 2**).

Besides the designs of model architecture, multifarious strategies were also leveraged but the endeavor was in vain (**Table 4 and 5 in appendix**).

4.2 T-SNE

According to the close observation of raw CT data, it was found that all patients were scanned from the vertex, but had different end points, such as at mandibular and ocular level, which lead to different numbers of images for different patients.

To visualize and analyze the latent space of the ResNet18 backbone model, t-SNE technique was utilized. Embeddings of images were firstly extracted from different datasets, and reduced from 512 dimensions to 2 dimensions. Considering the original rules of position labeling, which labeled vertex to mandibular as 0 to 45, all the position levels were accordingly reversed for each patient (i.e. 0 denotes the vertex). It showed that the embeddings of vertex CT images and mandibular CT images were clustering together in the left side of the three t-SNE plots while the embeddings of middle CT images were in the right side. Therefore, it could be claimed that our ResNet18 backbone model had the ability to learn the position information and position information were not required to provided as the additional feature for training (**Figure 2**).



(a) Training dataset (b) Validation dataset (c) Testing dataset

Figure 2: t-SNE for embeddings extracted by ResNet18 backbone model.

4.3 Saliency map

To elucidate what did the model learn, explainable saliency map was further utilized (**Figure 5 in appendix**). The left pair unequivocally showed that the model did focus on the region where the ICH occurred, while the saliency region distributed somehow evenly in the right pair, which, thus, was predicted erroneously.

4.4 Mislabeled

Some strange patterns, for example "...101..." and "10101", did exist in the provided ground-true labels. After thorough examination, several of them were actually mislabeled and would deteriorate the model performance (**Figure 6 in appendix**). However, our model could still predict the diagnosis correctly, which definitely demonstrate the advantages of deep learning model utilization in clinical scenario to reduce misdiagnosis.

5 Conclusion

The merit of utilizing stacked CT images, novel asymmetric loss, and Bi-LSTM to enhance vanilla CNN model in ICH multi-labels prediction is demonstrated in this work. Some unreasonable data augmentation or training strategies that would deteriorate model performance are also explicated in the aforementioned experiments. Besides, explainable saliency maps and distribution of embedding space enable insight into what actually the model learnt. Finally, the correct prediction of erroneously labeled data does manifest the unprecedented potential of leveraging deep learning technique in clinical diagnosis assistance.

References

[1] Chen, T. & Kornblith, S., Norouzi, M. & Hinton, G. (2020) A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*

[2] Ben-Baruch, E. & Ridnik, T. & Zamir, N. & Noy, A., Friedman, I. & Protter, M. & Zelnik-Manor, L. (2020). Asymmetric Loss For Multi-Label Classification. *arXiv preprint arXiv:2009.14119*

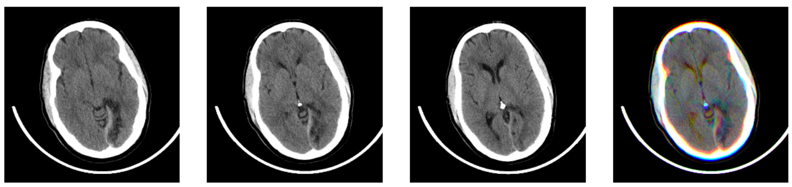
6 Appendix

Focal Loss

$$
\begin{cases}
\mathcal{L}_{+} = (1-p)^{\gamma} \log(p), \\
\mathcal{L}_{-} = p^{\gamma} \log(1-p).
\end{cases}
\tag{2}
$$

Binary Cross Entropy Loss (stage-2)

$$
\mathcal{L} = p \log(p) + (1-p) \log(1-p) \tag{3}
$$



(a) *l-1* (b) *l* (c) *l+1* (d) *3 Channels*

Figure 3: (a)(b)(c) are CT images. For a training sample, we stacked 3 consecutive CT images.

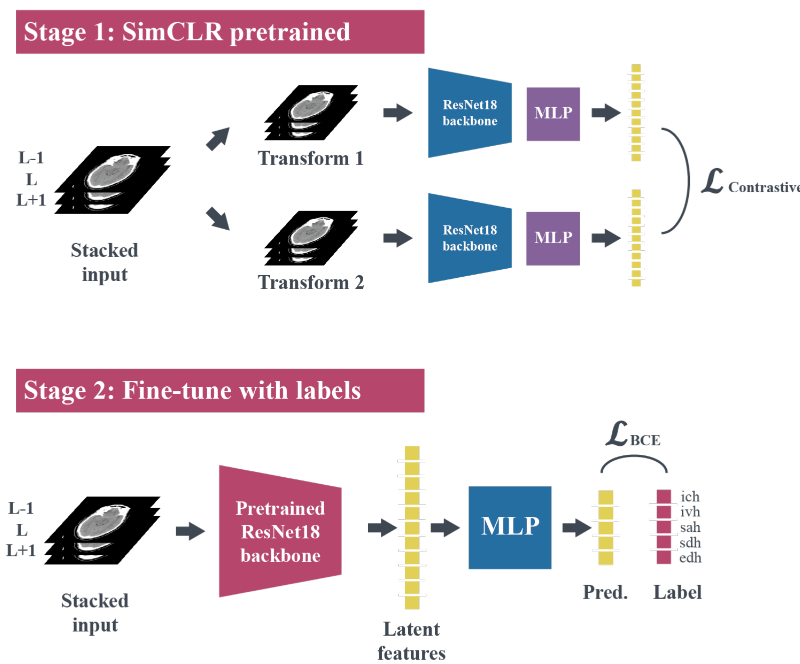
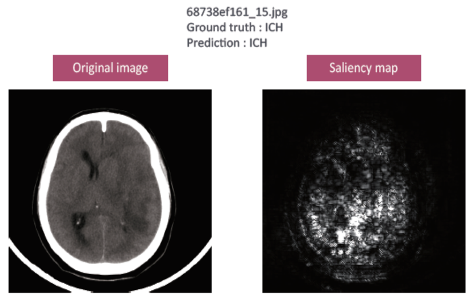
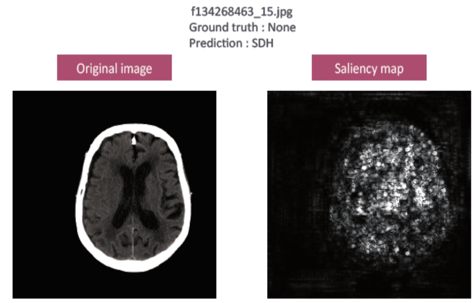


Figure 4: **SimCLR:** ResNet18 backbone, with SimCLR pre-trained

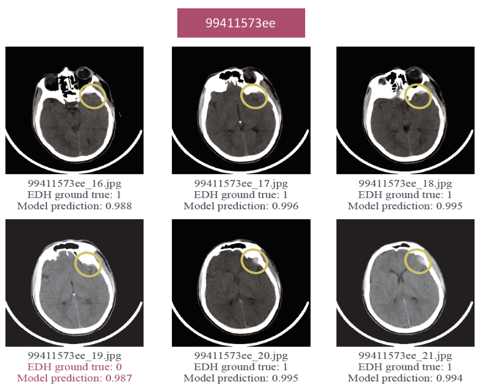


(a) Match

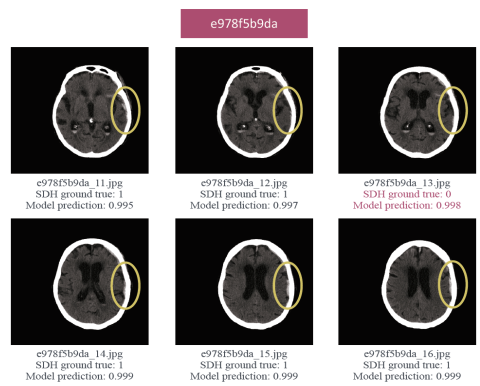


(b) Mismatch

Figure 5: Saliency map examples



(a) Case 1



(b) Case 2

Figure 6: Mislabeled examples

		Adam		**SGD**			**SGD No nesterov**
Batch size		64	40	64	40		40
Colorjitter		0.5	0.1	0.1	0.2		0.1
Rotation		20	20	20	30	30	30
Gray	Train f2	0.8496	0.7273	0.8612	0.8140	0.7480	-
	Val f2	0.6892	0.7045	0.7064	0.7143	0.7155	-
Stacked	Train f2	0.7208	0.7986	0.8140	0.8408	0.7726	0.7377
	Val f2	0.7011	0.7000	0.7143	0.7307*	0.7175	0.7153

*Test f2: 0.7310

Table 3: Comparison results of vanilla ResNet18 under different hyperparameters settings

Attempt		**ResNet-34 With Adam BCE Loss**	**ResNet-34 With SGD BCE Loss**	**ResNet-50 With Adam BCE Loss**	**ResNet-50 With SGD BCE Loss**	**DenseNet121 With Adam BCE Loss**	**DenseNet121 With Adam BCE Loss**
Stacked	Train f2	0.870	0.979	0.687	0.832	0.818	0.796
	Val f2	0.637	0.647	0.637	0.712	0.731	0.730
	Test f2	0.717	0.708	0.698	0.716	0.735	0.734

Table 4: Other attempts with vanilla ResNet and DenseNet

Attempt		ResNet-18 Asymmetric Loss Relabeled with 0.5	ResNet-18 Asymmetric Loss Relabeled with 1	ResNet-18 Asymmetric Loss Filtered data	ResNet-18 Asymmetric Loss Magic normalization	CNN-LSTM END2END
Stacked	Train f2	0.812	-	0.881	0.848	0.824
	Val f2	0.757	0.769	0.753	0.759	0.741
	Test f2	0.700	0.744	0.727	0.743	-

Table 5: Other attempts with multifarious tricks

		Adam with asymmetric loss					
Colorjitter		0	0.1	0.2	0.1	0.1	0.1
Rotation		20	20	20	30	40	50
Stacked	Train f2	0.843	0.842	0.829	0.865	0.843	0.854
	Val f2	0.741	0.743	0.729	0.781	0.773	0.771
	Test f2	0.743	0.750	0.735	0.736	0.749	0.755

Table 6: Comparison results of ResNet18 with asymmetric loss under different hyperparameters settings