Brain CT Image Hemorrhage Classification & Segmentation

Anupam Kumar, Wenfeng Wang, Tony Xiaochen Xiao, Abhilasha Jain, Rayan Ghanem

2021-12-07

Abstract

In this project, we constructed a classification model and a segmentation model for brain CT images based on CNN (Convolutional Neural Network) and U-Net architecture for CNN. Six classes are involved based on different categories of the hemorrhage in patient's brain, whereas the segmentation aims to provide the exact location of the hemorrhage for a given brain CT image. We cropped the 3D data of brain CT scan such that it compactly fits the ventricular region. We used 3D U-net as semantic segmentation, with weighted binary crossentropy loss. Finally, we reach 68% *accuracy to our CNN classification model and* 92% *accuracy to our U-net segmentation model for testing sets.*
Click on this link to access the github code repository for this project.

1 Introduction & Dataset

Brain surgery requires doctors to precisely locate the treatment part, since a tiny error may cause serious results, which is why it is so important to develop mathematical models to make better judgments for treatment. Given brain CT images taken from a patient, a common problem in brain surgery usually includes:
(1) Decide if there is a hemorrhage happening in patient's brain.
(2) If the answer to (1) is yes, then decide which types of the hemorrhage exist.
(3) If the answer to (1) is yes, then decide the exact location of the hemorrhage part.

Our project is about developing a mathematical model using machine learning, including deep learning techniques, to the given labeled dataset to let the computer automatically complete the all of the tasks (1), (2) and (3) above.

The dataset we used to train our model, in this project, is is provided by Zeta Surgical, a biomedical company which missions to democratize access to accurate, safe, and fast image guidance, to unlock the use of image guidance directly at the point of care, and to enable new treatments in cases such as emergencies and bedside procedures.

Here is a brief introduction to our dataset: We have totally 752803 instances of brain CT images which consist of all the categories as described in Table 1 below. Additionally, here is a file $hemorrhage - labels.csv$ which records the correct category for each image. In particular, here are images who have multiple hemorrhage. Besides, there are 6 files containing the information of correct locations for each image in the main directory, which is used to generate masks for segmentation.

Much research has been done on Brain CT images for hemorrhage studies, which shows how the relative ease of use of the ABC/2 method is neglecting the shortcomings of this method, and hence developing

Table 1: Instances of the Dataset

main directories\subdirectories	Brain Bone	Bone	Max Contrast	Subdural
epidural	1694	1694	1694	1694
intraparenchymal	15664	15664	15664	15664
intraventricular	9878	9878	9878	9878
subarachnoid	16423	16423	16423	16423
subdural	32200	32200	32200	32200
multiple hemorrhage	32074	32074	32074	32074

* Each instance is an image which is of size 512×512 for 3 channels (RGB).

automated models might give a proper diagnosis for the treatment. Some of the few articles on the topics can be found in Appendix A.

For the rest of the paper, we proceed as follows. The model of classification, involving the answer of tasks (1) and (2) above, is descripted in Section 2. In Section 3 we will show our analytic work and results about classification. The explanation of how we construct the model of Segmentation (task (3)) are explained in Section 4, and Section 5 gives the analytic work and results of segmentation. Section 6 concludes.

2 Classification of Hemorrhage types

2.1 Dataset

Due to the large size of the dataset, we randomly chose 1000 images from the Max Contrast subdirectory in every main directory for building our training set and testing set. Then, since there are 6 main directories, we have 6000 images in total that will be trained and tested.

After the images are read into the google cloud, we first resize them as well as merge for three channels by taking their mean value. Here is the formula for implementing such operations.

1. For each pixel (i, j) in an image, compute the gray scale X_{ij} by the formula below. Note that since $R_{ij}, G_{ij}, B_{ij} \in [0, 1]$, it is guaranteed that $X_{ij} \in [0, 1]$.

$$X_{ij} = \frac{R_{ij} + G_{ij} + B_{ij}}{3}, \forall i, j = 0, 1, ..., 511$$

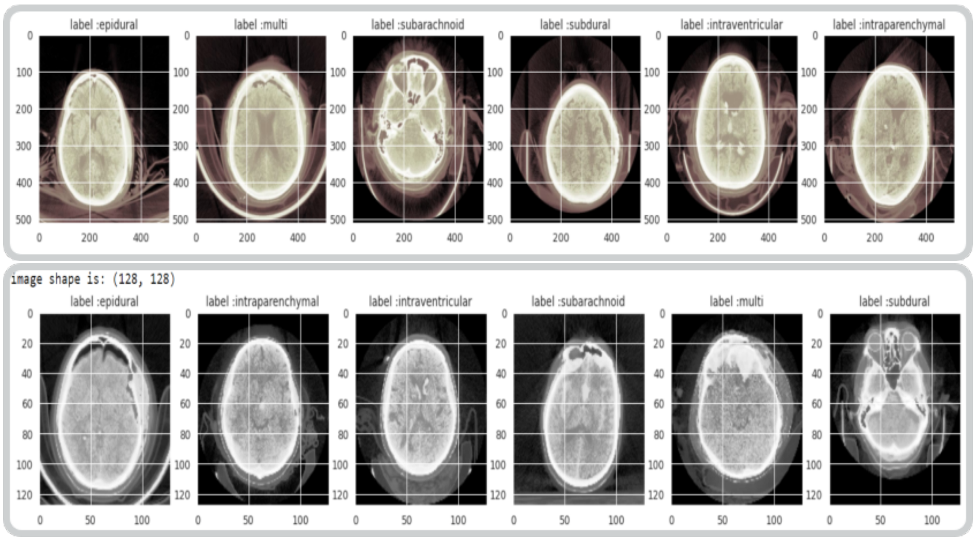
2. For each image X of size 512×512, the new image Y of size 128×128 is computed by

$$Y_{ij} = X_{4i,4j}, \forall i, j = 0, 1, ..., 127$$

We can see from the following diagram that after doing these steps, the number of entries is now 98,304,000 which is acceptable for the jupyter notebook to store without explosion.

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{DATASET}\\ \#\text{ entries} =\\ 752803 \times (512 \times 512 \times 3)\\ = 592,028,368,896\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Random Choose 6000}\\ \#\text{ entries} =\\ (1000 \times 6) \times (512 \times 512 \times 3)\\ = 4,718,592,000\end{array}} \arrow[r] & \boxed{\begin{array}{c}\text{Merge Channels}\\ \text{Resize the images}\\ \#\text{entries} =\\ (1000 \times 6) \times (128 \times 128)\\ = 98,304,000\end{array}}
\end{tikzcd}$$

Here is a group of pictures showing the comparison between original and processed images.



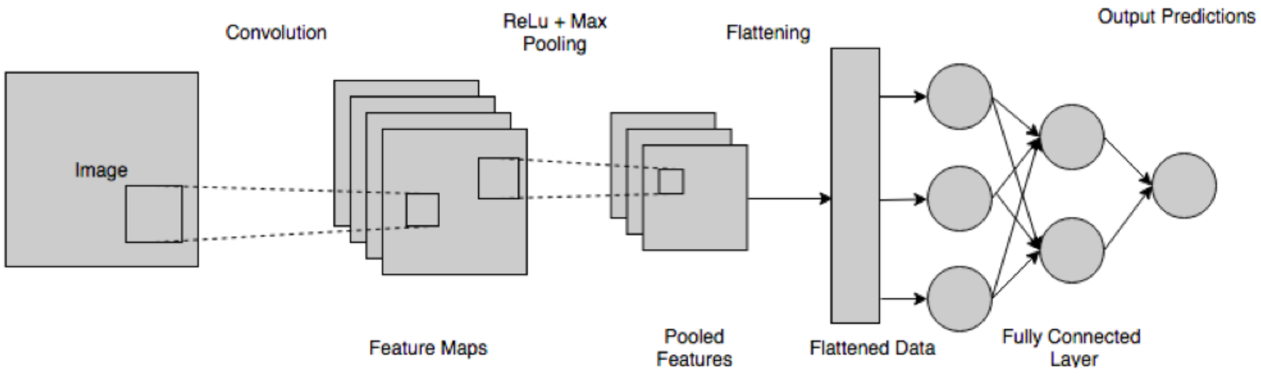
```
image shape is: (128, 128)
```

We store the matrix of training data by randomly choosing 80% out of all 6000 instances. The testing data is the remaining ones. So the sample sizes of training dataset and testing dataset are respectively

$$\#\text{Training Instances} = 4800; \#\text{Testing Instances} = 1200;$$

2.2 Classification CNN Architecture

Here is our model for classification: It is a CNN model with 3 convolution layers with ReLu and Max Pooling, 1 convolution layer with flattening layer, 2 fully-connected hidden layer. This architecture has 1.2 million parameters.



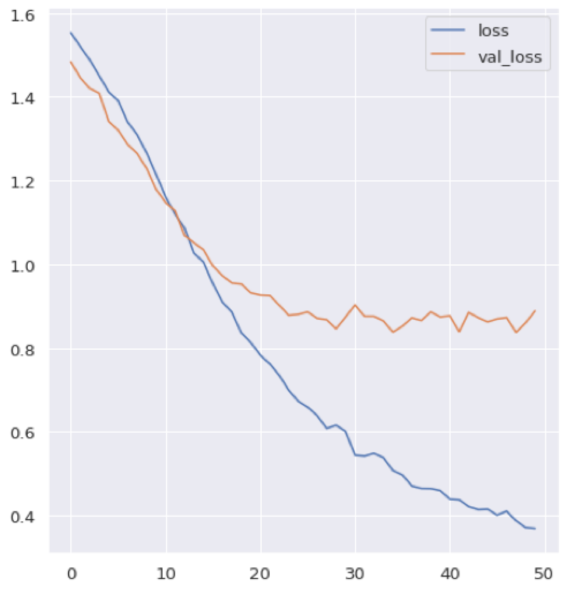
We found that there was a significant overfitting during training. The accuracy gap on training set (90% accuracy) and validation set (50% accuracy) was very high. Some ways to handle overfitting are adding more data, augmenting the data, change learning rate, reduce architecture size. We tried 1600 images of each type instead of 1000 images, but it did not improve the accuracy. We added *Dropout* layer, tried to tune the dropout parameter. We also changed the architecture shape and size to correct overfitting. We also tuned the learning rate. This results in achieving a maximum validation accuracy of approximately 70%.

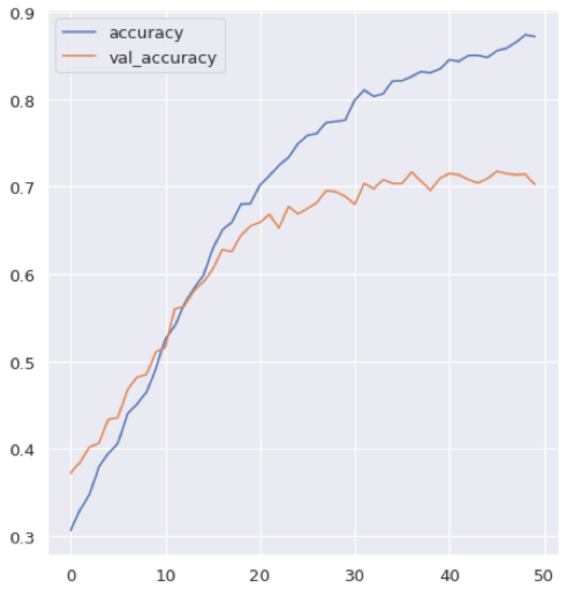
2.3 Classification Results

The accuracy and loss of first 50 epochs with regard to the training set and testing set of our CNN model are shown in the graphs below: Blue lines represent the accuracy and loss functions of the training set as epoch evolves, whereas orange lines represent that of the testing set. From the graphs, we can clearly see that the neural network reaches to be well-trained at approximately 25 epochs. After this time, there is no distinct improvement on testing dataset. Therefore, we decide to let our actual model stops training at epoch 25 to avoid overfitting.

Table 2: Results of Classification Model (At Epoch 25)

Testing Set	68%
Training Set	75%





We would like to report that the results for classification was not stable. For the same architecture and all the parameters, re-training the model was giving different outputs. This shows that the training process is depending a lot on random initialization. Some ways to overcome it are to use higher batch size, use of optimal learning rate, etc. We tried these approaches, but they did not work out.

3 Modeling for Segmentation

3.1 Dataset

We got our dataset from reference [4]. The dataset consits of 3D MRI images of ventricles, manually separated by software 3D slicer and converted to a python numpy format. There are a total of 87 3D brain images with masks for ventricles. The dimension of each 3D brain image is (176, 208, 176). Since the segmented ventricles are small parts in a particular location of the brain, to reduce the imbalance of background and mask, we sliced each image further to size (80, 120, 120) containing the ventricles compactly.

3.2 Model Architecture

For the segmentation part of our project, we choose to use the model named U-Net architecture for CNN since it is found to be the most appropriate method to implement. The U-Net architecture can localize the borders and areas by classifying every pixel and maintaining the same dimensionality as what the input provides.

The U-Net displayed in the graph below illustrates how our segmentation model works. The model is structured from multiple convolutions and pooling layers, and it does not have any dense layer. It consists of two paths symmetric to each other: the contraction path (marked in gray), and the expansion path (marked in black).

The contraction path follows the usual CNN process. To begin with, an image, which in our case is of size is 512×512, is inputed at the red arrow shown in the graph. Then, it passes two convolutions operators marked in blue arrows to reach step $C1$. Then, it gets to $P1$ by max-pooling, which intrinsically is a maximal function:

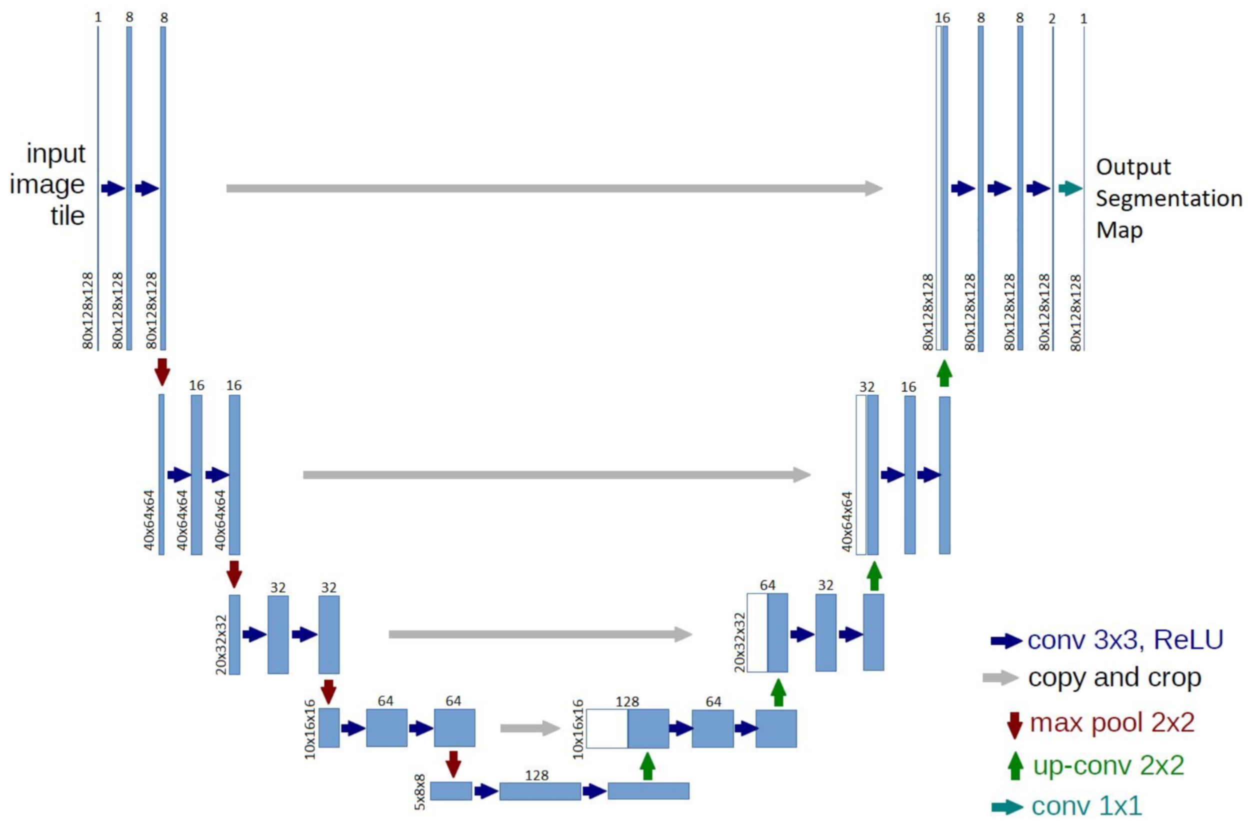
$$P1_{ij} = \max\{C1_{2i,2j}, C1_{2i+1,2j}, C1_{2i,2j+1}, C1_{2i+1,2j+1}\}, \forall i,j = 0,1,...,511$$

followed by another two convolution operators, and another max-poolings, etc.

The architecture of 3D U-net is very similar to 2D U-net with added dimension. We used Conv3D layer with 3x3x3 filters. However, the input shape needs to account for the feature channels within the upcoming convolution layers, hence the 4-dimensional input shape. Within each convolution layer, the 3-dimensional filter applies padded convolution and outputs a value for each voxel. After two convolution layers, 3D max-pooling is applied to take the largest element from the rectified feature map within a specified cuboid. We followed the standard structure of the original paper and applied max-pooling with a $2 \times 2 \times 2$ cube, which cuts the input dimension by half after the operation. The downsampling steps are then repeated multiple times with each convolution layer having double the amount of feature channels as the previous repetition. This reduced the input dimension down to $10 \times 15 \times 15$. This is then accompanied by two additional convoluted layers without max pooling. We then enter the expansive path.
The total number of parameters of the architecture is 1.2 million. We used Adam optimizer with learning rate $2e-4$, a custom weighted binary crossentropy loss with weight [0.2, 0.8] for background and mask, and accuracy as metrics.

We also explored the difference between using 3D semantic segmentation of 2D semantic segmentation for segmentation of 2D slices of 3D image. We found the reference [5] interesting. We used a big chunk of codes from repository [6].

In the expansive path, we're bringing the image back up to the original input size. It begins by upsampling the output of the previous layer. Here, we upsampled by the same size of what was used in the max-pooling layers earlier, which simply doubled each voxel to double up the size of each dimension. After upsampling, the 3D-image is concatenated with the corresponding 3D-image from the contracting path to combine the information from the previous layers to get a more precise prediction. This causes the feature channels to double again from the concatenation. However, it is then followed by two convolution layers with half of what the previous convolution layer has. Similar to the contracting path, the upsampling steps are then repeated the same amount of times as what is done in the contracting path with concatenation of each respective 3D-image in the contracting path. Finally, a convolution layer with two feature channels followed by another layer with one feature channel to produce the final trained output.



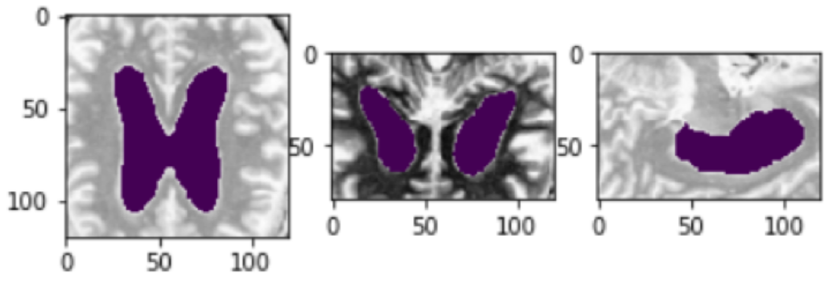
3.3 Segmentation Results

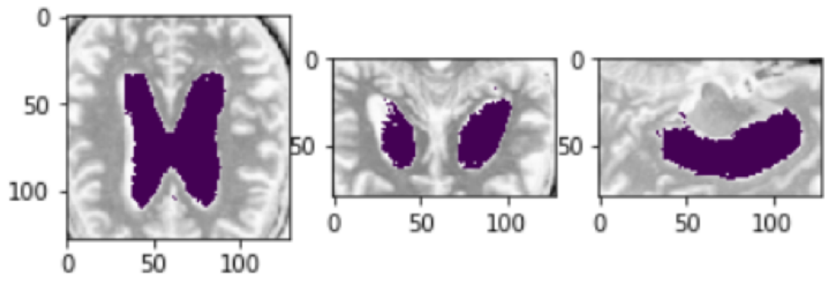
By the U-Net model described above, we finally reach the 92% accuracy on the testing set and 98% on the training set.

Table 3: Results of Segmentation Model

Testing Set	92%
Training Set	98%

Here are some intuitive comparisons between the images labeled with correct masks and the segmentation results. Please note that the three images on the first row are labeled previously, whereas the ones on the second row are predicted by the computer.





4 Conclusion and Future Works

We learned that classification was a much difficult task. We tried two different approaches to read data and keep getting low accuracy it took a lot of trial and error to get good accuracy on classification. 3D segmentation training task was more stable got good accuracy, but not good F1 score as there is high imbalance in two classes. We can use different weights for weighted binary crossentropy loss and see its effect on the confusion matrix. We can perform data augmentation to add more data if given further time. The model can be fine tuned by changing several hyperparameters of the architecture. Our architecture have 1.4M parameters. A bigger architecture can be used.

5 Acknowledgements

We hereby especially thank Professor He Wang for providing the opportunity for us to work on this project and to visit the company. We are also grateful for Zeta Surgical, for giving us such an opportunity to work on live industrial project by generously offering us its first-hand dataset. Additionally, we want to thank Raahil Sha, the Co-Founder and CTO at Zeta Surgical, with his team, for welcoming our visit in person and for the offering of their computers with higher power.

References

[1] Intracerebral Haemorrhage Segmentation in Non-Contrast CT
[2] Automatic Segmentation of Intracerebral Hemorrhage from Brain CT Images
[3] Segmentation and quantification of intra-ventricular/cerebral hemorrhage in CT scans
[4] Segmentation part from MATH7243-2020 course
[5] Performance evaluation of 2D and 3D segmentation approaches on CT images
[6] MRI-Image-Segmentation