AI to Identify Spontaneous Intracranial Hypotension(SIH)

Yuyang Zhu(yz3691) and Tianle Zhu(tz2434)

Medical Image Analysis

03/06/2020

**Outline**

1. **Recap project**
2. **Implementation and progress of the project**

**2.1 Data Preprocessing**

**2.2 Segmentation - UNet**

**2.3 Classification - VGG16 and ResNet50**

**2.4 Object detection - Yolo**

1. **Challenges and state-of-the-art solutions**
2. **Exploration and discussion**

# 

# 1. Recap project

The data for SIH detection should contain 150 patients, and the patient’s brain MRIs with abnormality will also have the spinal MRI. Recently we obtained three patients’ brain and spinal MRIs. The partial data contains about 417 images for Brain MRI images and 584 images for Spinal MRI. Thus, the project can be divided into two parts.

In the first part, we need to find whether the brain MRI is abnormal or not. There are three signals to judge the abnormal. The first typical imaging findings of intracranial hypotension abnormal sign is cerebral vein dilation-related, such as venous distension and diffuse pachymeningeal enhancement. Moreover, for the brain descent-related abnormal signs, we can estimate it through midbrain pons angle and flattening of the pons. What’s more, we can judge the aberrant by some complications. For instance, subdural fluid collections and/or subdural hematoma. In the second part, the patient with abnormal MRI will do the spinal MRI further. Then we intend to develop a method that automatically identifies CSF leakages from spinal MRI pictures, and locate the place where the leakage happens with high accuracy.

# 2. Implementation and progress of the project

## 

Previously we practiced the deep learning model (UNet) with a Kaggle dataset that classifies flair abnormalities. It contains about 4000 brain MRI images for 110 patients. And carried out a moderate result. While that naive attempt has received some good results on the dataset with abundant data, we tried to formalize our approach by doing augmentations, preprocessing, picking up better loss functions, and hoping to get a better result.

Meanwhile, we also tried object detection, another approach in doing classification to solve this problem, we learned several models about it and are ready to apply.

## 2.1 Data Preprocessing:

Augmentation: We created one extra image for each image that is already in the dataset (due to the Kaggle brain MRI segmentation dataset is already pretty large and the limitation of our PC’s computing ability), which includes:

* Rotation up to 40 degrees.
* Width and height shift up to 20%
* A slight shearing
* A slight zooming
* Horizontal flip

In all, we created 6898 images for training purposes, 318 images for validation, and 162 images for testing results.

## 2.2 Segmentation:

We tried UNet, a convolutional neural network that is commonly used in medical image analysis to bring out the semantic segmentation. Here’s a brief pipeline for UNet[1].

First, it goes through several down block sessions, in each session it goes through two convolutional layers and one max-pooling layer, which lower the image size by one fourth in each session. Secondly, it follows two convolutional layers to extract more information from the image, this step is called a bottleneck. Finally, it goes through several up-sampling layers to retrieve the original sample size, in each session it first performs an up-sampling layer, then a concatenate process with the previous downsampling layer with the same size, and two convolutional layers. In the end, it does one more convolution to give a classification on every pixel.

We trained the neural network for 12 epochs, with about 800 steps in each epoch(choose batch size as 8), with two different loss functions:

* Cross-Entropy
* Intersection-over-Union, which computes the overlapping positive area divided by the union of positive areas in two images, and adds a negative sign to make it a loss function.
* Dice loss, which is computed by the overlapping positive area divided by the sum of the positive areas in both images, and also adds a negative sign to make it a loss function.

In the end, only Cross-Entropy works out, the networks trained by the other two-loss functions just tend to predict the whole picture to be negative, no matter what smoothing parameters we use. We believe it’s caused by the existence of many images that didn’t really have a positive mask (all pixels are actually classified as negative in the true data) that make the model dumb in training.

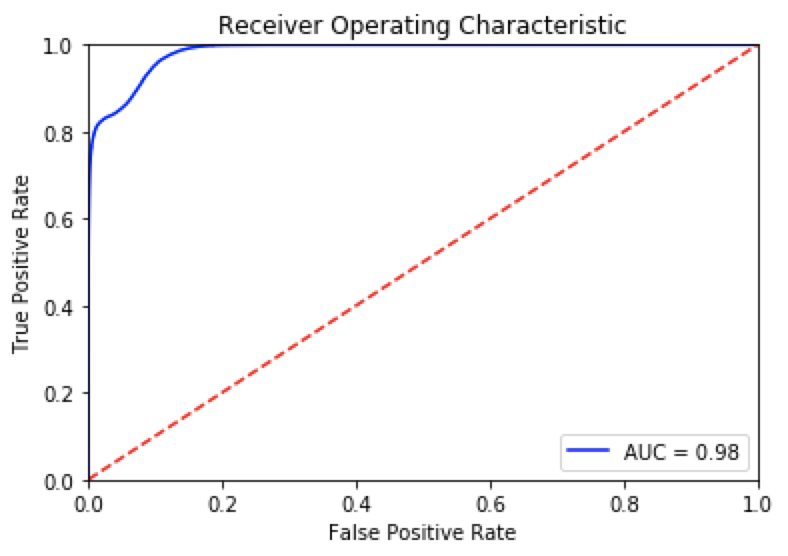
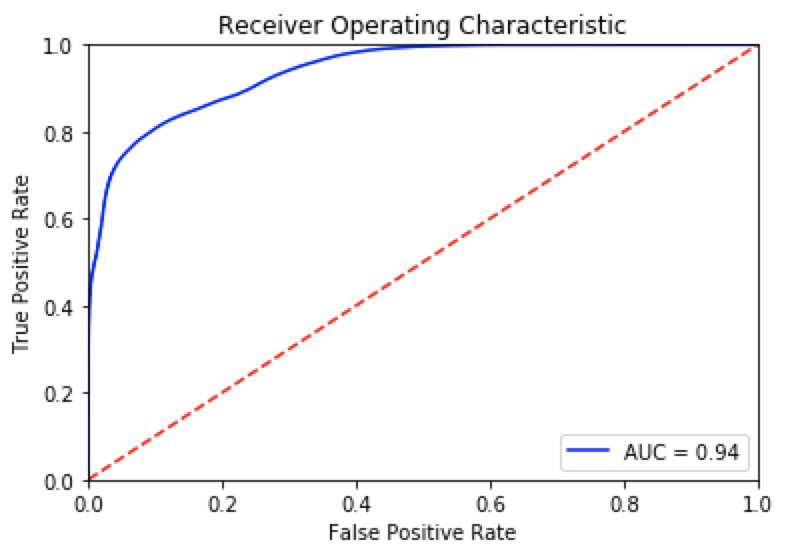
We used 0.3 as a threshold to classify the data (which is obtained from ROC curve in Figure 1, to make false-positive rate lower than 0.001 for each pixel) and plot the contour for the true results and predicted results for visualization, here are some sample outputs in Figure 2.

Figure 1(with crop(might be over crop) vs without crop the image)

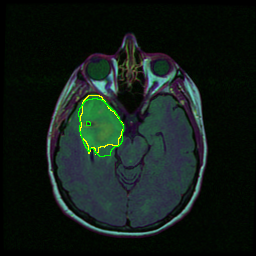
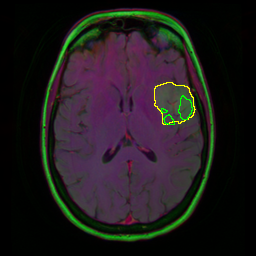
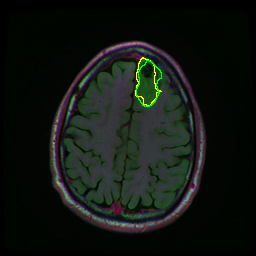


Figure 2(yellow line shows the true mask, the green line shows the predicted mask)

## 2.3 Classification

We implemented the classification by using VGG16 and ResNet on the Brain Tumor MRI dataset from Kaggle. For the Brain Tumor MRI data set, we did the data preprocessing.

Firstly, split the data into training, validating and testing dataset, then we have 193 images for training, 50 images for validating, and 10 images for testing data. The data consists of MRI scans of positive (Tumor) and negative (No Tumor) two classes. Secondly, due to different image sizes in the data set, we want to normalize the image. So, we found the biggest contour of the image and the extreme points on the image, then we cropped the rectangular out of the brain. Thirdly, we changed the image sizes to be 224\*224. At last, we did data augmentation since we had a small data set. Data augmentation helped us to “increase” the size of the training set. Then, we did the transfer learning by using the VGG16 and ResNet50 model.

### 2.3.1 VGG16

VGG16 [1] is one of the earliest successes of Convolutional Neural Networks in image recognition tasks. It tried to increase network depth to improve model performance. The authors use very small convolution filters to reduce parameter space. The network generalizes extraordinary well to image recognition, classification and localization tasks. The VGG16 pre-trained weights were trained on ImageNet data.

Based on the original VGG16 model, we made some improvements in the structure. We added a drop of layer after the VGG16 model. With increasing complexity comes increasing overfitting. So, we regularize it by using a strategy dropout layer. On the Convolution layer, we used 512 filters with filter size 7\*7. On the dropout layer, we set the dropout rate to be 0.5. After that, we added the fatten layer and another dropout layer. The last layer would be the dense layer with a sigmoid activation function (structure of classifier: Conv layer → Drop layer→ Fatten layer→ Drop layer→ Fully Connected layer (Dense layer)).

### 2.3.2 ResNet:

ResNet [2] is a residual learning framework that can have substantially deeper neural network architecture than those used previously. The author explicitly reformulates the layers as learning residual functions by utilizing skip connections. Skip connections then become the standard computer vision modeling mechanism. The residual networks are easier to optimize and can gain accuracy from considerably increased depth. ResNet pre-trained models are trained on ImageNet data which has 1000 classes of objects.

Improved the ResNet50 model by adding the layers which are similar to the VGG16 model, but except for using 2048 filters with filter size 7\*7.

For both models, we trained the neural network for 120 epochs in this dataset, with 50 steps in each epoch. We had 30 steps for the validation dataset in each epoch. Our loss function is Binary Cross-Entropy. The optimizer we tried Adam. We calculated metrics to compare two methods. The accuracy is equal to the number of correctly predicted images divided by the total number of tested images. Finally, we obtained the accuracy for VGG16 is 90% while the accuracy for ResNet50 is 86%.

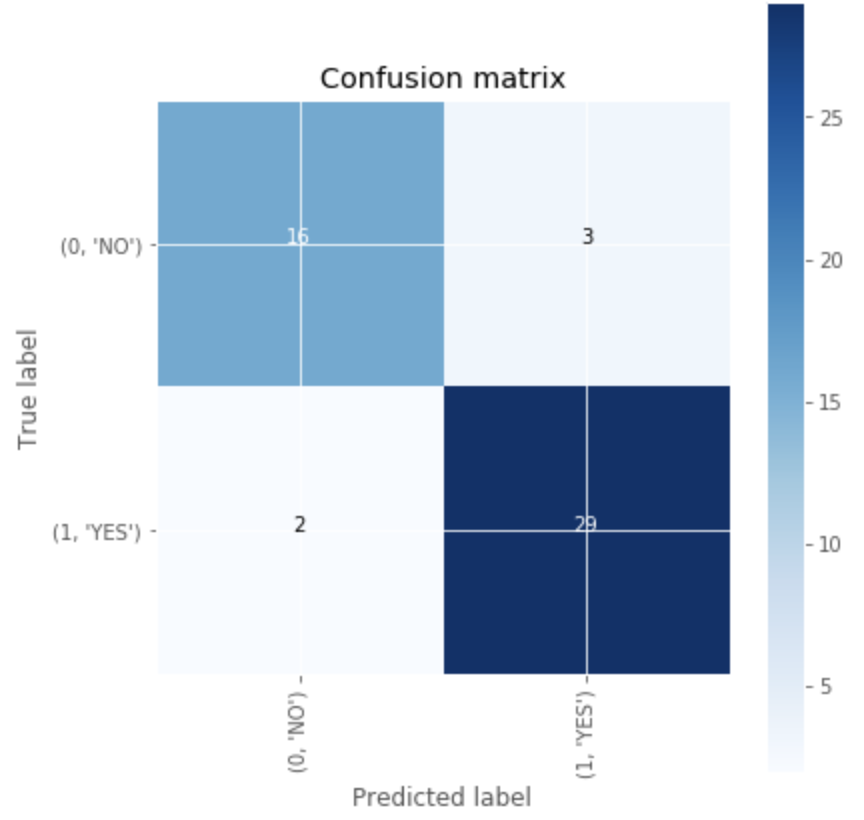
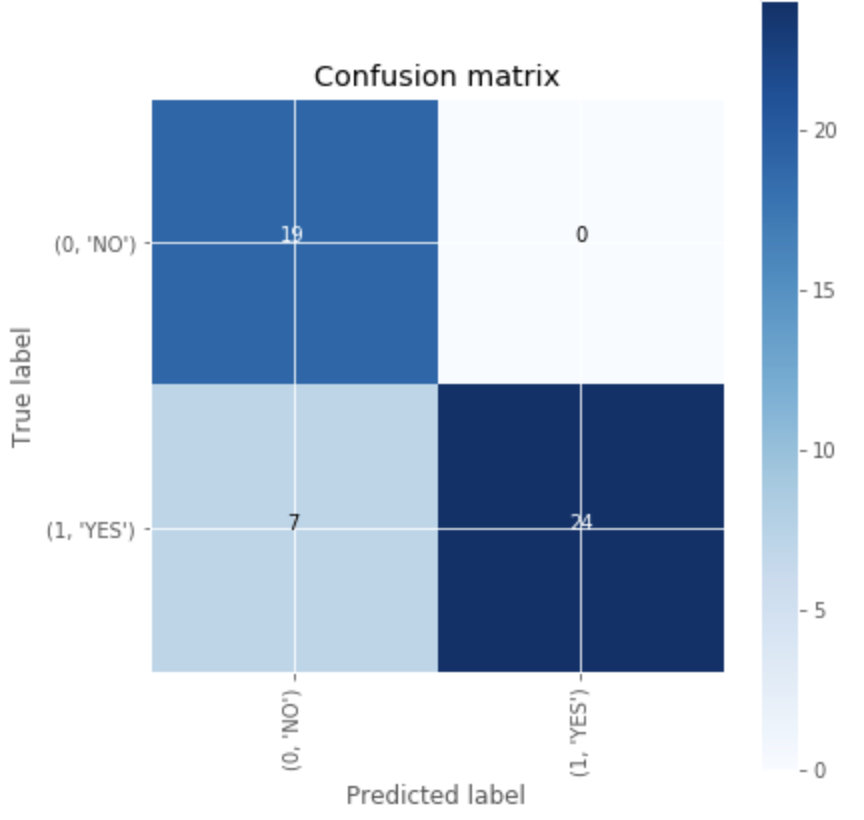
## VGG16 ResNet50

## 

## The following figures are the confusion matrix for VGG16 and ResNet50.

Type I Error (False Positive) and Type II Error (False Negative) help us to identify the accuracy of our Model which can be found with the help of the Confusion Matrix. On the confusion matrix, we can find the Type I (false positive) and Type II Error(False negative). For the confusion matrix on ResNet50, we can get that out of fifty patients the model has seven Type II error which means seven who were tumor but didn’t detect. Meanwhile, for the VGG16, We can conclude that out of fifty patients the model has two Type II errors and three Type I errors(which means three out of fifty patients who did not have a tumor but the model prediction says that they have a tumor). Comparing to ResNet50, the VGG16 model has a smaller total error that is sum the value of Type I and Type II Error. Thus, the Type II Error here is more dangerous than Type I Error.

## VGG16 ResNet50

## 

# 3. Challenges and Solutions

For the Brain segmentation dataset from Kaggle, we tried to give a reasonable outline when cropping the image, due to the large volume of the pictures with different outlooks, the thresholding method didn’t perform well on all of them, and sometimes the images all over-cropped as a result. While we can’t solve this by replacing the thresholds manually, we extend the outline a small percent for each image, which makes the cropped image a bit better, and other badly cropped images are discarded(We believe most of them doesn’t really have a valid mask)

And the initial results show a bit underfitting than expected, we suspected that there’s something to do with the layers and the data but we can’t confirm that yet. To improve the performance of the predictor, we looked at the ROC curve (derived by pixel-wise prediction) and tried to lower the threshold to increase the True Positive rate while controlling the number of False Positive predictions. And we choose 0.3 as a threshold in the end.

Other challenges in Brain segmentation dataset mainly involve image preprocessing, we believe this has contributed to the failure when using dice coefficient and the underfitting issue. We only have done a little bit to this part yet (discarding some highly irrelevant images), and need more solid criteria for further improvement.

For the Brain Tumor dataset, there are two main challenges in using transfer learning to build the brain tumor classifier.

The first one is thatthe data of interest in this project is significantly different from the training data used by the original model. As we can see from the brain tumor dataset, the images are generated by medical instruments, while the datasets used by popular computer vision models are typically ImageNet or COCO. Therefore, it is safe to assume that the neural networks derived from the latter dataset would perform worse in detecting medical features. The pre-trained weights can only be used to initialize the weights. Fine-tuning is required to allow the network to learn how to detect medical features related to brain tumor classification and to improve model performance.

Therefore, we deal with it with unfreezing the weights initialized from ResNet50 and VGG16 with a small learning rate. The model would be able to slowly adjust the weight to detect new features which could perform better in classifying brain tumors.

The second challenge is that the architecture of the original model is designed for ImageNet data which has 1000 classes. The image recognition neural networks usually have two kinds of layers. The convolutional layers on top of the architecture and the linear layers at the bottom of it, which we usually call as the classifier. As we are only required to classify tumors and no tumor, in this case, the neural network should have a different classifier. As we are using this model on a novel and complicated task, we need to find the best hyperparameter space for this model. We need to customize the classification layers and tune hyperparameters to find the best parameter space.

Thus, we explored the hyper-parameter space (number of dense layers and dropout layers, dropout rate, number of neurons) of the classifier and selected the hyperparameter set with the best validation performance in terms of accuracy.

# 4. Exploration and Discussion

In our SIH dataset, we did data preprocessing such as cropped our dataset and did augmentation for the training set. However, recently, we only get a partial dataset without labels, it impedes our progress. The things we can do is explore more relative datasets, and find the similarities and useful methods while improving the methods we tried on the Kaggle dataset.

The Kaggle Brain segmentation dataset at most describes part of the features for CSF leakage but might describe little about features like midbrain pons-angle abnormality, or flattening of the pons. Currently, we only know it labels the abnormalities in the FLAIR sequence, it could be helpful to transfer this to recognize features like venous distension. We still need to find how to deal with midbrain pons-angle abnormality or flattening of the pons. Most importantly, we need to explore the way to capture the features in our final dataset. For instance, we should define those features that could indicate there’s a CSF leakage, and then think about how to do something like a transfer learning to finish the job. We will need to give masks to the no label data and go through some medical details.

For our Kaggle dataset in process, it seems there is more that can be done, like different ways of augmentation that make the most sense, and other preprocessing operations might also become helpful. In general, metrics like the dice coefficient should have worked out, but it failed in our training model in the brain segmentation dataset. Usually, the model ResNet50 would have better performance than the VGG16; however, in the brain tumor dataset, ResNet50 did not work well. Thus, we need to explore more on the hyperparameters, such as loss function and optimizer. So, I guess some improvements are still needed.

For our Brain and Spinal MRI dataset, we think object detection YOLO [3] ( You look only once) will useful. We want to explore this part and try to implement our dataset on YOLO. YOLO is an object detection approach proposed in. The author frames the object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. One single neural network predicts bounding boxes and class probabilities directly from full images. This one-stage methodology is significantly different from other two-stage methods like R-CNN. YOLO pre-trained models are trained on the COCO dataset with mAP as high as 60.6 as the author updated in their recently updated version YOLO v3.

# Reference:

@inproceedings{ronneberger2015u,

title={U-net: Convolutional networks for biomedical image segmentation},

author={Ronneberger, Olaf and Fischer, Philipp and Brox, Thomas},

booktitle={International Conference on Medical image computing and computer-assisted intervention},

pages={234--241},

year={2015},

organization={Springer}

}

[You Only Look Once: Unified, Real-Time Object Detection](https://arxiv.org/abs/1506.02640)

Deep Residual Learning for Image Recognition