# CS583A: Course Project

Name: Xiangyao Deng

December 1, 2019

## 1 Summary

I participated in an active competition of classifying Intracranial hemorrhage, bleeding that occurs inside the cranium. The nal model we choose is multi-model, ResNet-50, and DPN, which take 224*224 and 112*112 images as input. The labels are five types of intracranial hemorrhage and one 'any' label meaning there is at least one type of intracranial hemorrhage. We implement the three models using Keras and running the code on two machines; one is Alienware M15 with one Intel i9 CPU and 16 GB memory and an NVIDIA GeForce 2080-MaxQ GPU with eight memory and a workstation from aws that have half of Tesla K80 with 7G memory. Submissions are evaluated using a weighted multi-label logarithmic loss. Each hemorrhage sub-type is its own row for every image, and predict a probability for that sub-type of hemorrhage. Besides, there is also an 'any' label, which indicates that a hemorrhage of any kind exists in the image. The 'any' label is weighted more highly than specific hemorrhage sub-types. The competition has two stages. I passed the preliminary round and entered the second stage. In the public leaderboard, my active final active score is 0.26559, ranking 128/1345, including stage1 and stage2, and the late submission score is 0.15971, ranking 90/1345. In the private leaderboard, my active score is 0.37444, ranking 401/1345, including stage1 and stage2, and the late submission score is 0.15971, ranking 367/1345. As for total ranking including stage-1 and stage-2, I rank 401/1345 as top 30%

## 2 Problem Description

### 2.1 Problem.

The problem is to classify Intracranial hemorrhage, bleeding that occurs inside the cranium, is a serious health problem requiring rapid and often intensive medical treatment. The competition is at `https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/overview`.

### 2.2 Data.

The data are 512×512 DICOM images. The number of training samples is $n = 60,0000$.The number of validation samples is $n = 15,2800$. The number of classes is 6(any, epidural, intraparenchymal, intraventricular, subarachnoid, subdural). The training set is unbalance: $n_{\text{Positive}} = 107933$ and $n_{\text{Negative}} = 644869$.In the positive samples, the partition of sub-types labels is also imbalance, the number of epidural n=3145, Intraparenchymal n=36118, Intraventricular n=26205, subarachnoid n=35675, subdural n=47166.

### 2.3 Challenges.

1. The first challenge is to pre-process the DICOM format. CT image values correspond to the Hounsfield unit(HU). But the values stored in CT Dicoms are not Hounsfield units, but instead a scaled version. To extract the HU, I need to apply the linear transformation, which can be obtained from the Dicom tags.

2. Because of the big size of data, over 400G, I cannot follow the way handling typical RGB images. A little bit of changes in data is hard to finish. Therefore, I implemented a custom Datagenerator with image Augmentation for feeding images into models.

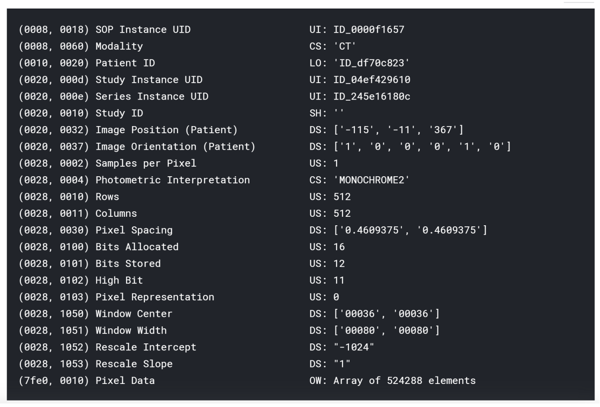
3. The data is an imbalance in positive and negative images, and also in types of lesions.

4. In this competition, I try three deep learning models and merge the results from two models, Resnet50 and DPN.

## 3 Solution

### 3.1 Pre-Process.

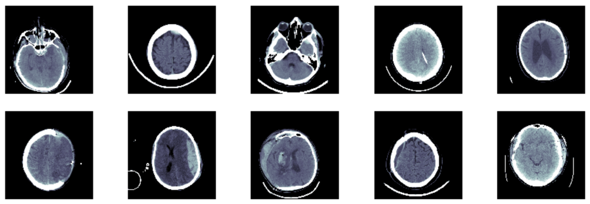
CT image values correspond to Hounsfield units. But the values stored in CT Dicoms are not Hounsfield units, but instead a scaled version[5]. I used pydicom to read DICOM format and extracted pixel array of image, slope, and intercept from meta of DICOM. Then, I apply a linear transformation: img*slope+intercept. Finally, I converted the image from DICOM format to .png format. The original size of the image is 512*512, but my GPU cannot run in this size of the image. Therefore, I transform the image into 224*224 and 112*112. Code: `https://github.com/TangBoo/CS583-Deep-Learning/blob/master/Project/Convert_dicom_png.ipynb`.



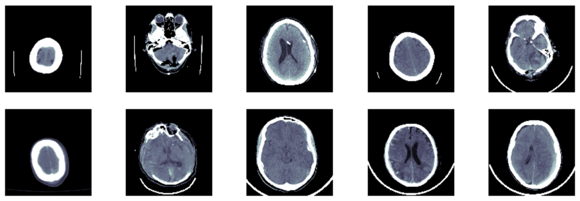
```
(0008, 0018) SOP Instance UID                    UI: ID_0000f1657
(0008, 0060) Modality                            CS: 'CT'
(0010, 0020) Patient ID                          LO: 'ID_df70c823'
(0020, 000d) Study Instance UID                  UI: ID_04ef429610
(0020, 000e) Series Instance UID                 UI: ID_245e16180c
(0020, 0010) Study ID                            SH: ''
(0020, 0032) Image Position (Patient)            DS: ['-115', '-11', '367']
(0020, 0037) Image Orientation (Patient)         DS: ['1', '0', '0', '0', '1', '0']
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 512
(0028, 0011) Columns                             US: 512
(0028, 0030) Pixel Spacing                       DS: ['0.4609375', '0.4609375']
(0028, 0100) Bits Allocated                      US: 16
(0028, 0101) Bits Stored                         US: 12
(0028, 0102) High Bit                            US: 11
(0028, 0103) Pixel Representation                US: 0
(0028, 1050) Window Center                       DS: ['00036', '00036']
(0028, 1051) Window Width                        DS: ['00080', '00080']
(0028, 1052) Rescale Intercept                   DS: "-1024"
(0028, 1053) Rescale Slope                       DS: "1"
(7fe0, 0010) Pixel Data                          OW: Array of 524288 elements
```

(a) Meta of DICOM

:

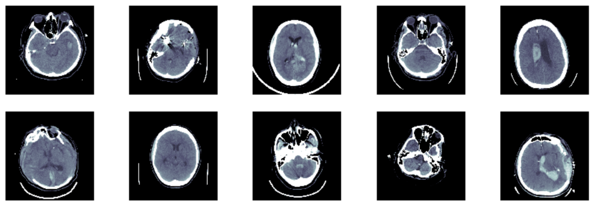


(b) epidural

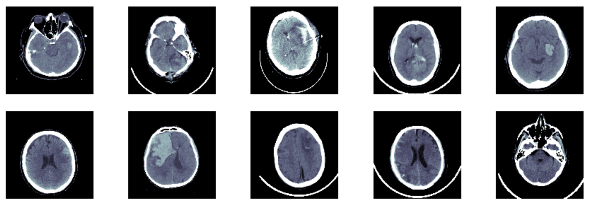


(c) subarachnoid

:



(d) Intraventricular

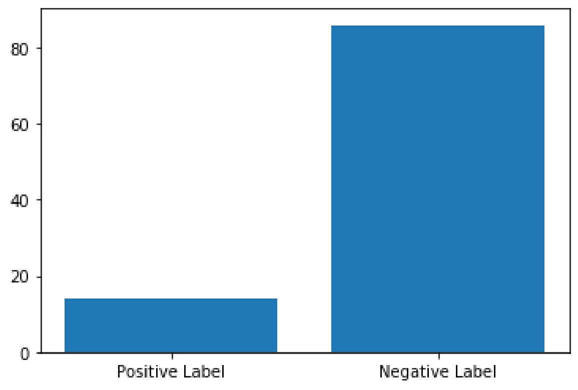


(e) Intraparenchymal

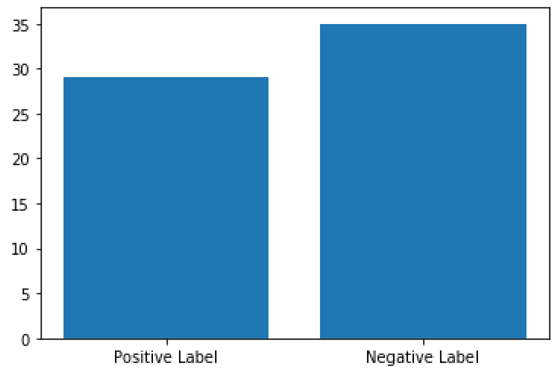
Figure 1: Pre_process

## 3.2 Imbalance Data.

To solve the imbalance between the positive and negative labels, I map positive and negative samples with different probability of extracting from the data set, such as map positive with 0.5, map negative with 0.35. Next, let the probability of samples compared with a random number 0-1, the samples their probability bigger than random number will be selected by data generator and put into batch[7].



(a) Imbalance Data



(b) Balance Data

Figure 2: Positive and Negative Label

## 3.3 Model.

Totally, I implemented three models, ResNet50[1], DensNet[2] and DPN[3]. The models I finally choose is ResNet50 and DPN. I merged the results from two models. I used Keras to implement the three models in Alienware M15 with GeForce 2080 Max-Q GPU and a workstation at AWS with half of Tesla K80. The ResNet took 60 hours(3 hours/Iteration), and DPN took 80 hours(4 hours/Iteration).

## 3.4 Implementation.

We implement the ResNet, DensNet, DPN model using Keras with TensorFlow as the backend. We run the code on an Alienware with one Intel i9 CPU and 16 GB memory and an Aws workstation with one half of Tesla K80 GPUs.)I spent 30 hours to train the ResNet and DensNet and spent 40 hours to train DPN. Our code is available at

DensNet Code. `https://github.com/TangBoo/CS583-Deep-Learning/blob/master/Project/DensNet.ipynb`

ResNet Code. `https://github.com/TangBoo/CS583-Deep-Learning/blob/master/Project/ResNet%20(1).ipynb`

DPN Code. `https://github.com/TangBoo/CS583-Deep-Learning/blob/master/Project/DPN.ipynb`

## 3.5 Settings.

Each image would have multi-label. For example, two and more types of bleeding may occur inside the cranium of one patient. Therefore, the loss function I choose is 'binary -crossentropy'. Besides, according to the requirements of the competition, I assigned 'any' labels with more weight. At first, I set the learning rate as lr=0.1 in SGD with momentum=0.9. Then, I used 'ReduceLROnPlateau' to monitor loss value and adjust the learning rate. After ten iterations, I used 'rmsprop' for reaching to converge point bette

### 3.6 Advanced tricks.

Adjust Optimizer: I used SGD as optimizer to train model 10 iterations at first, then I use 'rmsprop' as optimizer to train model 10 iterations.

Adjust Learning Rate: In training, I use 'ReduceLROnPlateau' to monitor the loss value for reducing the learning rate.

Recording Best Weight and Adjust Optimizer: We stored the weights with best accuracy of model using 'ModelCheckPoint'.Next time, I loaded the weight to train model with different optimizer.

### 3.7 Cross-validation.

Due to the size of the data set, I cannot do cross-validation in competition time. Therefore, for each iteration, I will shuffle all data set and split training data and validation data again. During training, I randomly select batch from all training samples for overcoming imbalance and set setps_per_epoch to 10000. In other words, I will randomly select 10000batch_size samples as the training set from 600000. Besides, I would randomly select 20% samples in the data set as a validation set.

## 4 Compared Methods

**DensNet:** I implemented DensNet by Keras. The training and validation accuracies are respective 0.8721 and 0.7621. Public score:0.27103, Private Score:0.37444

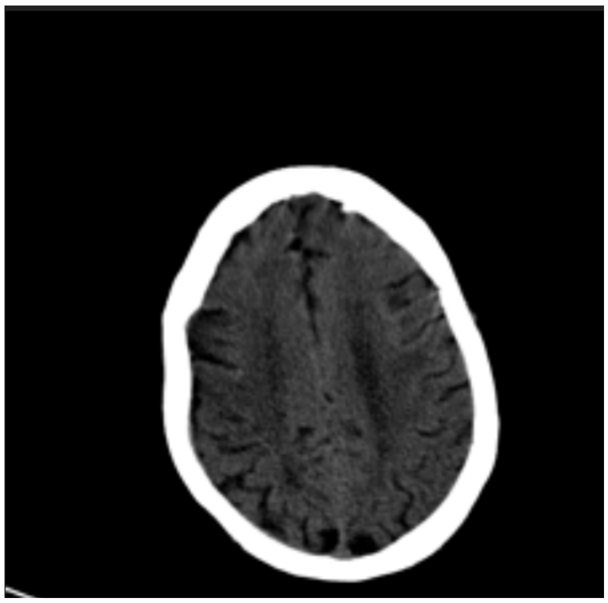
**DPN:** I implement DPN by Keras. The training and validation accuracies are respective 0.8858 and 0.8743. Public Score : 0.19962, Private score: 0.24898

**ResNet50:** I implemented RensNet-50 by Keras. The training and validation accuracies are respective 0.9074 and 0.8943. Public Score:0.09737, Private Score:0.21082

**Merge Results:** I merge the result of DPN and the result of Resnet50 Public Score:0.15971.Private Score:0.15750

**Advanced tricks.**

- Data augmentation. I build a generator with augmentation using imgaug. Each time, the augmenter would randomly flip and crop the image. Besides, the augmenter would randomly add one blur from GaussianBlur,AverageBlur,MediaBlur on the image. Finally, the augmenter would randomly add gaussian noise to the image.


(a) Original Image

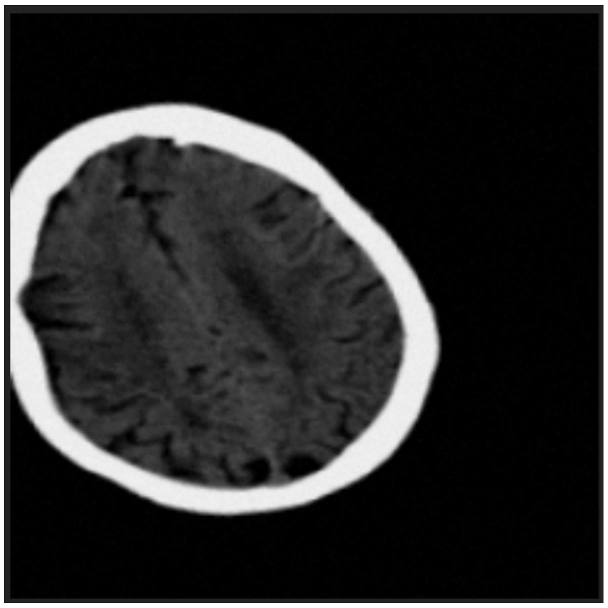
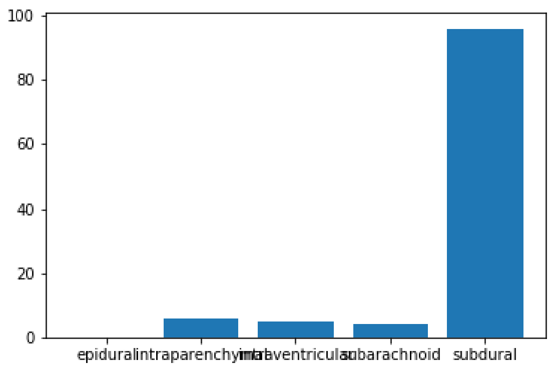
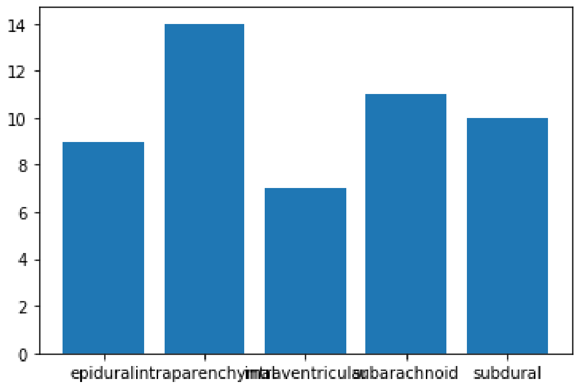

(b) Augmentation Image

Figure 3: Augmenter

- Over-sampling. The data is imbalance not only in positive and negative samples but also in sub-types in positive samples. Therefore, I need to do oversampling in each batch according to the proportion of sub-types in positive samples. I assigned the probability to each sub-type, which let sub-types with a small amount in data set have more chance to randomly selected by DataGenerator and sub-types with the big amount in data set to have less opportunity to randomly selected by DataGenerator[4]. Finally, I did the random pickup for batch size times, which let samples have an even distribution between sub-types samples



(a) Imbalance sub-types



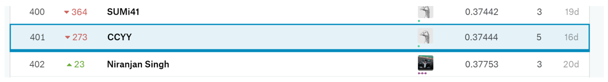
(b) Balance sub-types

Figure 4: Imbalance in sub-types

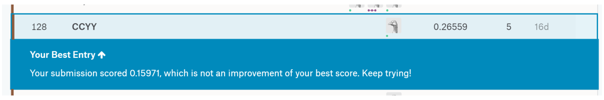
- Merge Results. I submit the results of the three models separately, but the performance of each model is not much different. Therefore, I merged the first two well-performance results according to their private score. The merged result exceeded the score of the best-performing model, ResNet-50, from 0.21 to 0.15.
- Data Generator. Due to the size of the data set, it is hard to operate data in the computer. I cannot directly use ImageDataGenerator from Keras. Therefore, I custom DataGenerator with Image augmentation using imgaug and Oversampling according to the distribution of classes in the dataset. Code: `https://github.com/TangBoo/CS583-Deep-Learning/blob/master/Project/augmentation.py`.

## 5 Outcome

I participated in active competition. My score is 0.26559 in the public leaderboard and 0.37444 in the private leaderboard. We rank 128/1345 in public leaderboard and 401/1345 in the private leaderboard. However, My best score in late submission is 0.15750, ranking 367/1345 in private leaderboard and 0.15971 ranking 91/1345 in public leaderboard. The screenshots are in Figure 6.

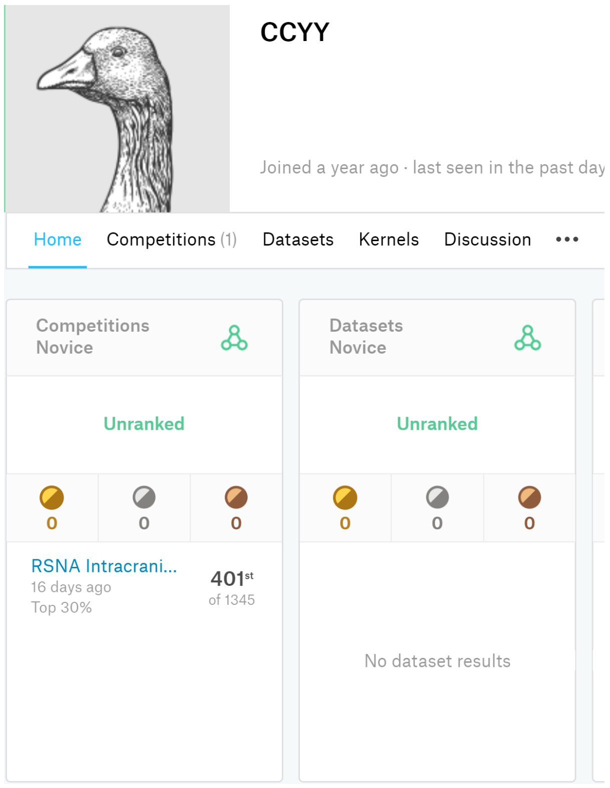

400 ▼364 SUMi41 0.37442 3 19d
401 ▼273 CCYY 0.37444 5 16d
402 ▲23 Niranjan Singh 0.37753 3 20d

(a) Private leaderboard.


128 CCYY 0.26559 5 16d
Your Best Entry ↑
Your submission scored 0.15971, which is not an improvement of your best score. Keep trying!

(b) Public leaderboard.

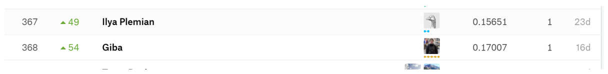
:


CCYY
Joined a year ago · last seen in the past day

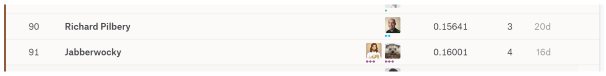
Competitions Novice
Unranked
0 0 0
RSNA Intracrani...
16 days ago
Top 30%
401st
of 1345
Datasets Novice
Unranked
0 0 0
No dataset results

(c) Final Ranking

Figure 5: I active rankings in the leaderboard.


367 ▲49 Ilya Plemian 0.15651 1 23d
368 ▲54 Giba 0.17007 1 16d

(a) Private leaderboard-late submission.


90 Richard Pilbery 0.15641 3 20d
91 Jabberwocky 0.16001 4 16d

(b) Public leaderboard-late submissionn.

Figure 6: I late submission rankings in the leaderboard.

## References

[1] KaimingHe,XiangyuZhang,ShaoqingRen,JianSun.,https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf

[2] Gao Huang,Zhuang Liu,Laurens van der Maaten https://arxiv.org/pdf/1608.06993.pdf

[3] Yunpeng Chen,Jianan Li,Huaxin Xiao,Xiaojie Jin ,Shuicheng Yan4,Jiashi Feng,https://arxiv.org/pdf/1707.01629.pdf

[4] Pavlos S.Efraimidis,Paul G.Spirakis,https://utopia.duth.gr/ pefraimi/research/-data/2007EncOfAlg.pdf

[5] Marco Vasquez E.,https://www.kaggle.com/marcovasquez/basic-eda-data-visualization

[6] Cheng Da,Haixian Zhang,Yongsheng Sang.,https://www.researchgate.net/publication/312769285_Brain_CT_I

[7] akensert,https://www.kaggle.com/akensert/inceptionv3-prev-resnet50-keras-baseline-model