Automatic Grading of Digital Subtraction Angiography (DSA) in Acute Stroke

Haoqing Dong
UID: 304622150
haoqingdong@ucla.edu

Shiqi Wang
UID: 304582601
sqwang97@ucla.edu

***Abstract*—Acute ischemic stroke occurs when a blood clot blocks an artery that carries blood to the brain [3]. The treatment of acute stroke requires a clear visualization of the blood vessels, which can be done through Digital Subtraction Angiography (DSA) imaging. The DSA images are graded by a scale called TICI score. This project is aimed to develop a model that could automatically assign a TICI score to a DSA image video. Convolutional neural network (CNN) will be used to train the automatic grading model based on a provided image set with TICI scores labeled by stroke neurologists for each image video.**

***Keywords*—Acute Stroke, TICI Score, DSA, CNN**

I. Introduction

Acute ischemic stroke is the most common stroke type, which occurs when a blood clot blocks an artery that carries blood to the brain [3]. Treatment of acute stroke requires clear visualization of the problem blood vessels. However, in ordinary X-ray projection images, the vessels are hardly visible due to the low contrast between blood vessels and the surrounding tissues. In order to get a clear view of blood vessels in the human body, digital subtraction angiography (DSA) will be used [1].

In DSA imaging, a radiopaque contrast medium (a iodinated solution) will be injected into the vessels to enhance the contrast. The radiopaque medium flows through the vessels and the entire process will be recorded as a sequence of images. However, the contrast between vessels and surrounding tissue is still significantly smaller than that between bone and surrounding tissues [2]. In order to remove this contrast distortion issue, each image will be subtracted from an image taken prior to the arrival of the contrast medium, thus making the processed images have a clear visualization of blood vessels.

To grade the DSA images, we use a scale called the Thrombolysis in Cerebral Infarction (TICI) score. TICI categories span from no perfusion (grade 0) to complete perfusion (grade 3) [4]. The "partial perfusion" category (grade 2) is defined as cases in which contrast passes the obstruction but with rates of entry and washout slower than normal and is subdivided into 2 subcategories, 2a and 2b [4]. Although there is a general scale for TICI score, it is not well quantified, especially for "partial perfusion" categories. Neurologists assign TICI score to DSA images based on their personal experience, which could potentially cause biases among the grading of TICI score from different neurologists [5].

In this paper, our goal is to build a model that could automatically assign a TICI score to a DSA image set. Upon given an image set, the tool will choose a single frame when the blood flow reaches the top of the brain as the frame of reference. Then it will apply a trained TICI score grading model on that frame to get an TICI score. The model will be trained from DSA image sets with TICI scores labeled by a number of stroke neurologists. Convolutional network, image processing and data augmentation will be used during the training process. With the automatic grading tool, the grading of TICI score could be quantified and becomes independent from neurologist's personal experience. Therefore, the tool could eliminate the bias among different neurologists and provide a uniform and standardized model for TICI score.

II. Review of State-Of-The-Art

Several researches have applied CNN to DSA imaging. Since DSA relies on consecutively acquired angiograms, the method is susceptible to inter scan motion, which could be very likely to happen. In order to solve this problem, Mathias Unberath and his team had devise a CNN in the well-known U-net architecture to regress image intensity value in coronary angiograms [6]. According to Unberath's team, they found promising results on both numerical phantom and real data, which encourages further research on deep learning-based virtual DSA [6]. Other researches such as Jin's[7] and Eulig's [8] also focus on improving the quality of DSA.

In addition, there is an previous research team that tried to implement a similar automatic grading tool to quantify TICI score [5]. Their research have shown a positive result for the automatic grading of TICI score, however, there is no open-source code that could be used right now. Therefore, building an automatic grading tool would be necessary and helpful for neurologists.

III. METHODS

a. Select the Frame of Reference

In this section, we introduce our method of how to select the correct frame from each image set for TICI score grading. Since the raw DSA images are grayscale images, it is hard to find the vessels in the image. For clear view, we convert the grayscale images into black-and-white (b&w) only images. The decision of whether a pixel is black or white is based on a threshold value: any value larger than the threshold value is white (i.e. 255 in grayscale value) and any value smaller than the threshold value is black (i.e. 0 in grayscale value). The threshold value is generated by skimage.filters package. For frames with threshold value 0, we remove those images from the dataset to reduce noise for future process. Since there is only two colors for processed images, we then mark white as 1 and black and 0, which resulted in a binary 2D matrix. With each grayscale image mapping to a binary 2D matrix, we are able to work on the binary matrices instead of the original images.

Since the specific frame we want is when the blood flows to the center top of the brain, we only consider the middle 1/4 of the upper 1/4 image (see the second image in Figure 1). We examine each row of this small matrix and count the black pixels in each row. This process collapses the 2D matrix of middle artery into a 1D array that counts the number of black pixels within each row. In Figure 1, the third image is an example of the 1D array generated from the image on the left. For graph purpose, we print each row's value as a line. If there is at least one black pixel in the row, the line is marked as black. Otherwise the entire row is white, then the line is marked as white.

After we get 1D pixel-count array for each frame within the given image set, we could begin to select the frame. The following algorithm is what we performed:

for each 1D pixel-count array:
initialize a flag to false
count the white elements in it
if number of white elements is larger than the half of the middle artery image height: turn flag on (set to true)
if the flag is on: choose the index of the array with number of white elements smaller than the half of the middle artery image height, return the image with that index
if all arrays are iterated and no image meets the condition, return None

Now we will explain why the algorithm works. Since each image data start from an empty brain image, like the one in Figure 2, the threshold of this image will be a low value due to the low contrast in this image. Thus, some pixels will be casted as black with a low threshold. However, the threshold of this frame is far lower than the threshold of other frames in the set where the blood vessels appear in the image (i.e. Figure 1). If we just choose frames that has number of black pixels greater than half of the image height, the first frame is what we will choose. This is definitely not what we want. Therefore, we set a flag to avoid selecting the empty brain image at the beginning. When the blood vessel appears, the top of b&w image will turns to white, and this is where we set the flag on to find the proper frame we want. The frames in Figure 3 record a common process of blood flow, where on the first frame we set the flag on and select the third frame as the frame of reference (black pixels greater than half).

For image set where our algorithm cannot find a frame of reference, we discard that image set from training data. In addition, we discard the image set that don't have corresponding TICI scores. The selected raw images are rescaled to 1/4 of their original width and height due to the RAM limitation, and are stored for further process.

b. Data Augmentation

In order to train an effective model, we need a large number of input training data, which could be generated by performing augmentation on selected raw images. We use tensorflow.keras library's ImageDataGenerator to perform random image augmentation and increase the dataset to 40x larger. The selection of this augmentation factor is mainly based on the RAM limit of Google Colaboratory, the platform where we run the code.

We first load the selected raw images and corresponding TICI scores from the previous step. Then, before we feed these data into the neural network model, we perform random data augmentation containing rotation, width shift, height shift and horizontal flip to each image in the training dataset. This step is applied both when we assess the model performance with cross validation and when we train the final model using the whole dataset, as we will discuss in the following section. To provide an example, we apply the augmentation to a sample image and make 9 new images as shown in Figure 4, with the original image on the top and 9 augmentation images on the bottom. Since we only define a range (i.e. the maximum value) for the rotation angle and shift distance, the actual value will be randomly generated. In this way, the

likelihood of having two exactly same augmented image in the training set is minimized.

The image augmentation is a very useful technique when the input data is limited. In our case, we are given only 201 image sets, and after discarding useless image sets, the number of available image sets decrease to 145. However, training a convolutional neural network usually need thousands of input data for the model to tune its parameter. Therefore, with augmented images, we increase the input data to 5800 images, and having this larger dataset increases the effectiveness of the model in the sense that the model is less likely to overfit.

c. Model Train

After preparing the input data, we begin to train the model. We accept the augmented data set as training input, and reformat the TICI score for each image before training the model. Since TICI scores are given as strings, we need to map them into integers. We map the strings 0, 1, 2a, 2b, 3 to integers 0-4 respectively. Then we convert each integer label to a vector of one-hot (aka one-of-K) encoded labels for better accuracy. For instance, the one-hot encoded label of 0 is <1,0,0,0,0> and the one-hot encoded label of 2b is <0,0,0,1,0>.

With the reformatted TICI scores, we build the convolutional neural network model with tensorflow.keras. The general idea is to first extract features with convolutions of filters, then reduce the size by max pooling. We could repeatedly extract more features by convolutions with an increasing number of filters. The choice of the actual neural network architecture is very tricky. In terms of training set accuracy, we want to extract enough features out of the inputs because any variation in the blood vessels may influence the TICI result. On the other hand, even with image augmentation, the number of training data we have is still too small compared to common datasets with hundreds of thousands of data. So we must keep the model very simple to avoid overfitting in a few training epochs. After trying several versions of the training process, our final version with the best performance consists of the following steps:

2d-convolution with 16 filters, 5x5 filter size, relu activation, l2 regularization
8x8 max pooling
perform a batch normalization for better accuracy
2d-convolution with 16 filters, 5x5 filter size, relu activation, l2 regularization
4x4 max pooling
perform a batch normalization for better accuracy
flatten the matrix to a vector
perform a dropout to reduce overfitting
densely-connected layer with softmax activation to generate output

The summary of our model is listed in Figure 5.

After building the model, we inject the previously processed DSA images together with their corresponding TICI scores to do the training. We assess the model performance using 5-fold cross validation. For each fold, the 145 usable images are split into 116 training data and 29 validation data. 40x image augmentation is then performed on the training set, while the validation set remains its size. Each fold is run with 30 epochs with a 32 batch size.

At the end, we train the same convolutional neural network making use of the whole dataset of 145 images with data augmentation, and save the trained model in the file model.h5.

IV. Results

As described in the previous section, the three major steps we have are the selection the feature frame, image augmentation, and training the model. The data is uploaded to Google Drive and the code in run on Google Colaboratory with the GPU hardware accelerator. The results of the feature images selection is stored under 'feature_images' folder as .mat files, which is the same file type as the given image set. Image augmentation is conducted at runtime so no results are stored. The trained model is stored in the file model.h5.

The overall runtime of the training the model is approximately 400 seconds. Consider the frame selecting time and TICI score reformatting time before the model train, the whole process would take approximately 15 minutes.

The result of the feature images selection is reasonable. Although we do not have supervised labels from professional neuroscientists telling us which is the key frame that should be used, the result of our selection meets our one expectation. The selection correctly deal with the noises in the images where the blood vessels doesn't exist. Also, images after the expected key frame where the blood vessels cover the whole brain is not mistakenly selected. In our algorithm, we convert the middle artery image to a 1D array by checking whether any black pixel exists in each row. We had some discussion about the potential vulnerability when random black pixel noises exist. It turned out that, in the context of our dataset, such noises only influence very few number of rows in each image, such that it in general does not change whether or not the number black rows exceeds a half. However, among 201

patients medical image data, 56 data are ignored with 32 of them are ignored because the algorithm does not find a valid image. The statistics imply that our algorithm does not perform very well for the image sets where each image, even those where the blood vessels appear, has a very narrow range of grayscale pixel values. The blood vessels parts thus look very close to the scull background, which means that it's difficult to find a filter threshold that bisects the vessels from the background. Another possible reason for the rate of ignorance is that some image sets are largely skewed or shifted, so that the middle 1/4 of the upper 1/4 image does not represent the section we want to look at. When we receive the dataset, we are informed to assume that the images are all centered and normalized, and it's difficult to perform normalization on ourselves due to the hard discrimination of the skewed images and the normalized images where the blood vessels look as if they are skewed.

The result of neural network performance is shown in Figure 6. The training set accuracy reaches about 80% after 30 epochs, but the validation set accuracy is around 20%, with a relatively higher validation loss than training loss. In the progress of training, the validation accuracy has reached as large as 50%, but fail to increase after that if not decrease. The result implies that slight overfitting still occurs in the model. We attempted to reduce the overfitting by doing more data augmentation and simpler model, but we either are limited by the Google Colaboratory RAM or observe very low increase in both training and validation accuracies, meaning that the model became too simple to learn anything from the data.

We do not implement a function to take in the trained model and a new input image set and predict a TICI score, because we do not have a test data set to evaluate the correctness of the final mode. But we implement a helper function that maps a scaler number, which we use as the model output, to a TICI score in the form of a string. For prediction, one can load the saved model and the input image set, perform the implemented choose_image function to select a key frame, rescale the frame to 256x256 (i.e., the input size of our model), then feed the rescaled image to the model, and finally use the helper function to covert the model output to a TICI string.

V. FIGURES

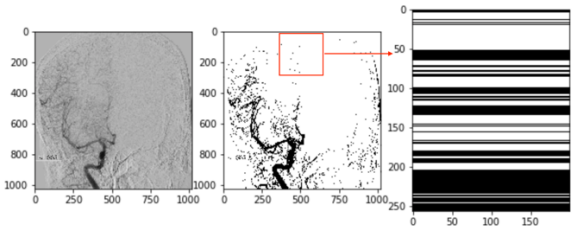

Figure 1. A mapping of one frame from a given image set. From left to right respectively, the first image is the raw DSA image, the second image is the black-and-white(b&w) image generated from the raw image, and the third image is the pixel count 1D array generated from the middle 1/4 of the upper 1/4 of the second image .

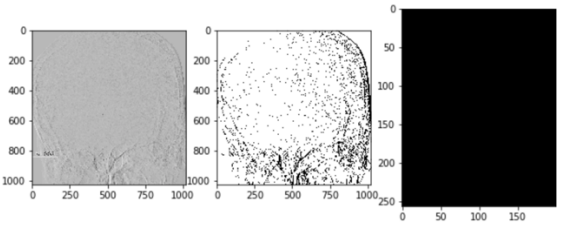

Figure 2. An empty brain frame, with lots of black pixels due to low threshold value.

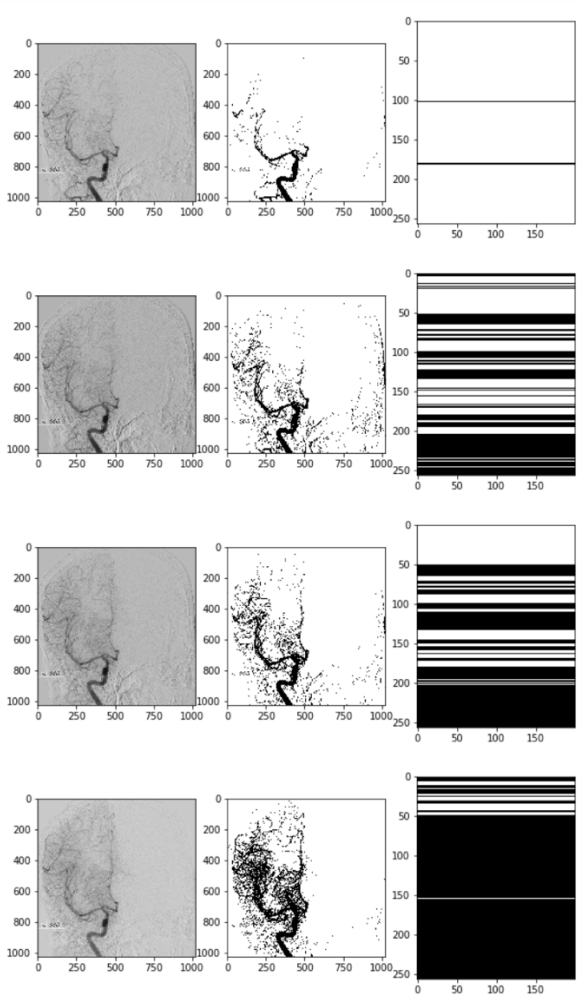

Figure 3. Some frames when blood flows through the vessels. The first frame is the beginning of the process and the third frame is what we what to select as the frame of reference for TICI score grading.

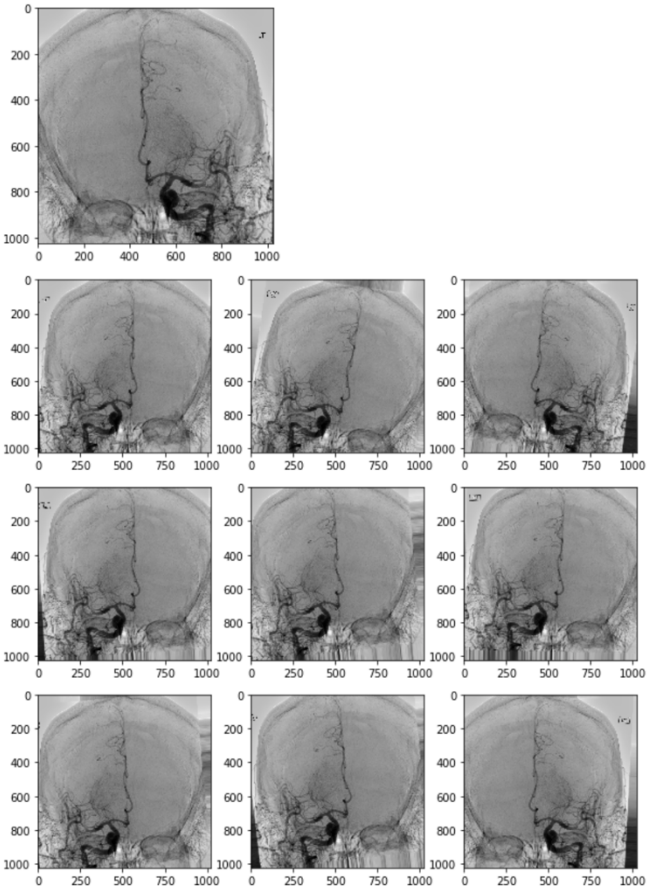

Figure 4. The augmentation of one selected raw image. On the top is the original image and the bottom 9 images are all generated by applying random augmentation method (width shift, height shift, horizontal flip and rotation) to the original image.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 256, 256, 16)	416
max_pooling2d (MaxPooling2D)	(None, 32, 32, 16)	0
batch_normalization_v1 (Batc	(None, 32, 32, 16)	64
conv2d_1 (Conv2D)	(None, 32, 32, 16)	6416
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 16)	0
batch_normalization_v1_1 (Ba	(None, 8, 8, 16)	64
flatten (Flatten)	(None, 1024)	0
dropout (Dropout)	(None, 1024)	0
dense (Dense)	(None, 5)	5125

Total params: 12,085
Trainable params: 12,021
Non-trainable params: 64

Figure 5. Summary of the Model

Model performance on the 5-fold cross validation:

	train_loss	train_acc	validation_loss	validation_acc
1	0.7404790076716193	0.8379310369491577	4.052580833435059	0.2068965584039688
2	0.49166037399193335	0.9168103337287903	1.7136542797088623	0.4137931168079376
3	0.4243195379602498	0.9295258522033691	3.469365119934082	0.3448275923728943
4	1.8362427645716173	0.5012931227684021	3.6599748134613037	0.3103448152542114
5	0.7093753039836883	0.7851293087005615	2.7695038318634033	0.48275861144065857

Figure 6. Model Performance on the 5-fold cross validation

REFERENCES

[1] E. H. W. Meijering, W. J. Niessen, and M. A. Viergever, "Retrospective motion correction in digital subtraction angiography: a review", IEEE Transactions on Medical Imaging, vol. 18, no. 1, Jan. 1999.

[2] E. H. W. Meijering, K. J. Zuiderveld, and M. A. Viergever, "Image Registration for Digital Subtraction Angiography", International Journal of Computer Vision, vol. 31, no. 2/3, April 1999, pp. 227–246.

[3] Society of Neurointerventional Surgery, "Acute Stroke", Society of Neurointerventional Surgery, [http://snisonline.org/stroke]

[4] J.E. Fugate, A.M. Klunder, and D.F. Kallmes, "What Is Meant by 'TICI'?", American Journal of Neuroradiology, December 2013.

[5] A. Sattar, K. Asif, and M. Teleb, "E-011 TICI Quantified: Automated Cerebral Revascularization Grading in Acute Ischemic Stroke", Journal of Neurointerventional Surgery, vol. 6, no. suppl 1, 2014.

[6] M. Unberath, J. Hajek, and T. Geimer, "Deep Learning-based Inpainting for Virtual DSA", IEEE Nuclear Science Symposium and Medical Imaging Conference, Atlanta, November 2017.

[7] H. Jin, Y. Yin, and M. Hu, "Fully automated unruptured intracranial aneurysm detection and segmentation from digital subtraction angiography series using an end-to-end spatiotemporal deep neural network", SPIE Medical Imaging, March 2019.

[8] E. Eulig, J. Maier, and M. Knaup, "Deep DSA (DDSA): learning mask-free digital subtraction angiography for static and dynamic acquisition protocols using a deep convolutional neural network", ECR, 2019