**Final Project: Classifying Brain Tumor from the Data Set**

### Hyoung Chul Kim(김형철)

**(Important** reminder: Details of the codes I used is in the separate source code file. The report part deals more about explaining the model.)

**About the Data**

The data I used was downloaded from **Kaggle**.

Here is the link: <https://www.kaggle.com/preetviradiya/brian-tumor-dataset> . The data set consists of x-rays of the brains of patients(Total 4600 data). Some of these patients were diagnosed with brain tumor and some did not. Since we have both images of healthy brains and unhealthy brains, we can use this to classify the data with Deep-Learning technique. Applying Deep Learning to these types of data is very useful. This is because algorithm that differentiate and classify brain tumor can be used as a nice reference in hospitals. In fact, many hospitals nowadays are collaborating with universities to construct accurate deep-learning algorithm that can classify various diseases shown in x-rays and MRI. It is used as a useful reference when making a diagnosis. As human is not a machine, there are times when doctors make wrong judgments depending on their conditions. Deep-Learning has advantage in this situation because unlike humans, it never tires out. Also, med-schools use Deep-learning to train their pre-doc students. In that sense, classifying Brain Tumor with Deep Learning algorithm would be a nice topic for the final project as it is one of the most important usage of the Deep-Learning Theory.

To give a more detailed introduction to our data, I put the condensed images of the x-rays that were used in this project:

A close-up of the front and the back of a coin

Description automatically generated with medium confidence

**The method(algorithm) used in the Final Project**

In this final project, I used CNN to classify the brain tumor of the patients. The reason I used CNN for this task is because CNN is a commonly used Deep-Learning algorithm for image classification. Fully-Connected Neural Network is usually not efficient in image recognition. This is because image has enormous amount of inputs and data. As fully connected Neural Network does not have Convolutional layers and Pooling layers that reduce the dimension of the input data, it takes considerate time to perform the classification. As convolution neural network can extract the best features from the image using convolution layers, it is lot more efficient than fully connected neural network. Thus CNN is usually faster and more accurate than the fully connected neural network. This is why I used CNN to classify the brain tumor data.

**(Reminder**: I referred to the code used in homework 5 when performing CNN. I wrote HW5 as my reference in the end.)

For the application of CNN method, I referred to the basic code used in HW5. I used python to import tensorflow to construct CNN model(detailed codes are in source code file). Then I rescaled the data and allocated 20% of the data as testing(Validation) group and the rest as training group. Then I defined the keras model and did model.summary(details about the code to define keras model is in the source code file):

Table

Description automatically generated

In my first model, there are two convolutional layers and two max-pooling layers. After that, I added the fully-connected neural network that has one output. This is because the project is about classifying the image data into two groups. For the activation functions, I only used “ReLu”. This is because ReLu function is a widely used function for convolutional layer. It is also considered quite efficient compared to other activation functions. For the last output layer I used sigmoid function because this is classification algorithm.

After that, I fitted the model using "model.compile" and model.fit and got the accuracy around 0.9902.

Chart, line chart

Description automatically generated

It seems the accuracy of the model is quite high. In fact, it is around 99%. The graph above also shows well that the accuracy of the validation data(testing data) gradually converges to the accuracy of the testing data. This shows that model itself is a nice prediction algorithm for finding brain tumor.

But there are some problems we have to think of. First one is "overfitting" issue. It is possible that the model fits very well with the data that we have but does not work that well with the new data of brain tumor images. This is because the model is fitted too well to the testing data that we used that it does not fit well into new data out of the lab. There are various ways to solve the overfitting issue. One is simplifying the data and the other is using dropout. Overfitting can usually occurr due to the complexity of the model. So making it less complicated (less neurons, etc) can help avoid overfitting problem. Using dropout also helps because it randomly drops neurons from the network during the training process. So we can train different structure of neural network, which can help in reducing overfitting.

Although there is no indication of our model having overfitting issue (at least not explicit), we will still use the strategies mentioned above to make sure no overfitting is happening in the model. (This concern is happening because the accuracy of the model is a bit "too" high. This could mean the model is "too" well fitted to the current data. Thus, some issue of overfitting might occur in the future.)

Hence I added a dropout into the model as follows: Not only that, I greatly reduced the parameters of the model. The first model had total 429,921 parameters. But the second model now has only 207, 656 parameters. This means the number of parameters was reduced to more than half. This was done to simplify the model and add dropout.

Table

Description automatically generated

Then I ran "history=model.fit" (10 epochs) to see the difference. The code showed me that the accuracy of the new model was around 0.9891.

Chart, line chart

Description automatically generated

Surprisingly, the accuracy of the model is quite similar to the first model. It is around 98.9%. Since the first model had around 99% accuracy, we can say that the second model is almost as good as the first one. Not only that, the second one will have lower chance of overfitting as the structure of the model is more simple. It is also more efficient. This is because it uses less parameters. Thus, it will take less time to fit the model. In that sense, it seemed the second model greatly improved the first model.

I also wanted to check if changing other factors of the model improve the algorithm. So, I changed the activation functions in the model. I only used ReLu functions for the first two models(except at the output layer). But using other activation functions might perform better.

Therefore, I decided to mix other functions in the model. Firstly, I tried using sigmoid function as the activation function for my first convolutional layers. Also, I used tanh function in the fully connected layer. The result was not that satisfactory. In fact, the accuracy dropped from 98% to 55%.

After that, I put tanh functions for both of my convolutional layers and kept the ReLu function in the fully-connected neural network part. This is the code I used to define the model:

Graphical user interface, text, application, email

Description automatically generated

Interestingly, the accuracy of this third model is even higher than other two models. It was around 99.1%. It seemed that perhaps using functions other than ReLu could work better in some circumstances.

Now I expanded the third model by adding one more convolutional layer and one more layer in the fully connected neural network part. This was because I wanted to see if making this model(one with different activation functions) more complex leads to better accuracy. The summary of the fourth model is as follows:

Table

Description automatically generated

The accuracy of this fourth model was better than the third model. It was 99.24%. From this, it seemed that using third and fourth model might be better for the image classification of our data. But before using the model, we need to consider the fact that the fourth model has about 10,000 more parameters than the first, second model. That is, the model is quite heavy. As the difference in accuracy is not that large, it might be more efficient to use previous models. The criteria for “which model to use” would depend on the relative importance the researcher has on accuracy and efficiency.

(I also tried increasing the epochs from 10 to 15 when performing “model.fit” to see if there is any difference. But it did not lead to any significant improvement of the accuracy.)

### **Conclusion**

My project shows that CNN is a very powerful tool in image recognition and prediction. This is easily shown by the fact that the overall accuracy of the model I constructed was well above 95%. Also, my project shows that complexity is not always the answer when constructing a sound model. As mentioned above, simplifying the parameters and using dropout did not damage the performance of the model. Remembering the fact that I took out more than half of the parameters from the first model, it shows that it is possible to construct efficient model with less parameters. Since making CNN is quite an expensive work, this insight about efficiency is an important lesson. Hence it is crucial that one thinks not only about accuracy but also efficiency and soundness of the model.

### **References**

1. Dataset used: https://www.kaggle.com/preetviradiya/brian-tumor-dataset

2. CNN code adopted and revised from: Homework 5 from the "Deep Learning" class in Yonsei University Spring Semester 2021.