

Identifying Types of Intracranial Hemorrhage

Cristina Giraldo

Taisha Ferguson

George Washington University


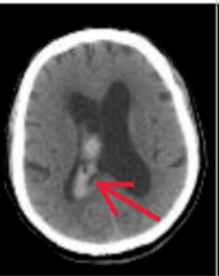
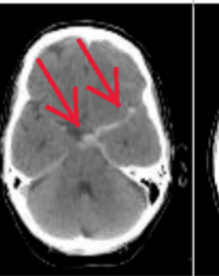
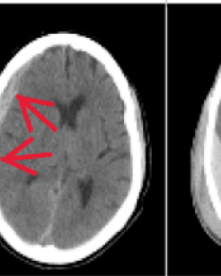

Table of Contents

<i>Introduction.....</i>	<i>3</i>
<i>Dataset.....</i>	<i>4</i>
<i>Network Training and Algorithm</i>	<i>6</i>
<i>Experimental Setup.....</i>	<i>8</i>
<i>Results</i>	<i>9</i>
<i>Summary and Conclusion</i>	<i>10</i>
<i>References</i>	<i>10</i>

Introduction

Intracranial hemorrhage, which is bleeding inside the skeletal structure of the brain, is responsible for approximately 10% of all strokes in the US. When a hemorrhage is suspected, medical specialists review images of the patient's cranium to determine if there is in fact a hemorrhage and the type of hemorrhage. The process of identifying hemorrhages can be complicated and time consuming[1] . The main objective of this project is to create an algorithm to correctly classify the presence of hemorrhage as well as the type of hemorrhage, if present. The five types of hemorrhages that we aim to correctly classify are intraparenchymal, intraventricular, subarachnoid, subdural, and epidural. The table below provides more details of each hemorrhage type. The table and the dataset were obtained from the Kaggle competition [RSNA Intracranial Hemorrhage Detection](#).

Table I: Hemorrhage Subtypes

	Intraparenchymal	Intraventricular	Subarachnoid	Subdural	Epidural
Location	Inside of the brain	Inside of the ventricle	Between the arachnoid and the pia mater	Between the Dura and the arachnoid	Between the dura and the skull
Imaging					
Mechanism	High blood pressure, trauma, arteriovenous malformation, tumor, etc	Can be associated with both intraparenchymal and subarachnoid hemorrhages	Rupture of aneurysms or arteriovenous malformations or trauma	Trauma	Trauma or after surgery
Source	Arterial or venous	Arterial or venous	Predominantly arterial	Venous (bridging veins)	Arterial
Shape	Typically rounded	Conforms to ventricular shape	Tracks along the sulci and fissures	Crescent	Lentiform
Presentation	Acute (sudden onset of headache, nausea, vomiting)	Acute (sudden onset of headache, nausea, vomiting)	Acute (worst headache of life)	May be insidious (worsening headache)	Acute (skull fracture and altered mental status)

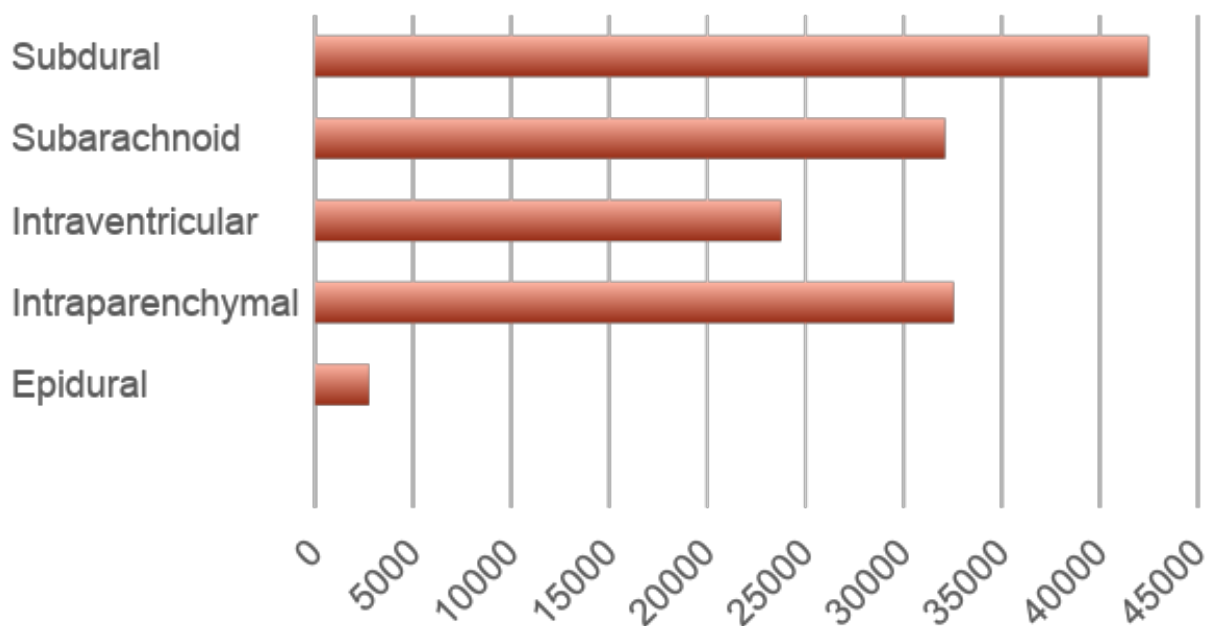
Dataset

As stated in the Introduction, our dataset for this project was obtained from the Kaggle competition “RSNA Intracranial Hemorrhage Detection”. The competition dataset is a compilation of brain exams obtained from Stanford University, Thomas Jefferson University, Unity Health Toronto and Universidade Federal de São Paulo (UNIFESP). The American Society of Neuroradiology (ASNR) organized and labeled the images for the purposes of the Kaggle competition. The images were organized into: 1) a csv file with file names and the corresponding hemorrhage labels and 2) a folder with dicom files labeled with patient ids. A dicom file is a medical file that contains the brain scan images as well as additional metadata about the images and the patient. The training csv contained 631,575 unique client records and the testing csv contained 121,232 unique client records. Because of the large amount of samples, we trained our models on subsample (200,000 images) which is about $\frac{1}{3}$ of the original dataset.

In addition to sampling our dataset we also applied image window of (level = 40, width = 80) which is a common practice for intracranial images[2].

There was a significant class imbalance in the data set. Out of the 631,575 patient images in the train set, 577,159 had no hemorrhage, 97,103 had some hemorrhage, 2,761 had hemorrhage type epidural, 32,564 had hemorrhage type intraparenchymal, 23,766 had hemorrhage type intraventricular, 32,122 had hemorrhage type subarachnoid, and 42,496 had hemorrhage type subdural. Table II below is a graph of the class distributions of the hemorrhage sub types.

TABLE II: Hemorrhage Sub-type Totals



Network Training and Algorithm

For this project we used Keras, which is an open source neural network library, to program and train our image classification algorithm. After reviewing some of the available coding examples on Kaggle's website, as well as several research articles related to image classification, we decided to build an ensemble model with the following components: a convolutional neural network (CNN) built from scratch, a pretrained model called Densenet, and another pretrained model called VGG16. From our research, CNN models, Densenet, and VGG16 models were very successful at classifying images individually. We thought the best strategy to improve the accuracy of the individual models would be to use a Machine Learning strategy called ensemble modeling. Ensemble modeling combines the predictions of two or more models into one prediction. It is generally accepted that ensemble models tend to be more accurate than the individual predictions of their component models[3,4].

Convolutional Neural Network

Convolutional Neural Networks (CNN) is a neural network architecture that uses one or more convolution layers in the network. A convolution layer uses the mathematical convolution operation to capture key elements from an image into feature maps. Another common component of CNN is the pooling layers. Pooling is the process of reducing the spatial size of the elements in the network. For our network, we used the following structure: three convolutional layers with relu as the activation function, one max pooling layer, one additional convolutional layer, one global max pooling, and final activation layer with the sigmoid function [5,6].

Densenet

The Keras program has several pretrained models with associated weights that can be downloaded and applied to your specific dataset. These pretrained models are based on specific network architectures that have been applied to large datasets such as Imagenet. Densenet is one of the pretrained models that is available through Keras. Densenet is a neural network architecture that is an extension of Resnet. One of the major contributions of the Densenet and Resnet architectures is the ability to solve the vanishing gradient problem by merging the outputs of previous layers into future layers. In our model we downloaded the weights from Densenet model applied to Imagenet dataset and added a global pooling layer as well as the final sigmoid activation layer[5,6].

VGG16

VGG16 is another pretrained model available in Keras. The major distinction of the VGG architecture is the depth of the network. VGG networks typically have at least 16 levels with 3X3 kernel sizes. Similar to the Densenet, we applied a global pooling layer followed by a sigmoid activation after downloading the pretrained weights[5,6].

Model Ensembling

Model Ensembling is a Machine Learning technique that combines the outputs of two or more models into one prediction. There are many different methods for model ensembling such as stacking, voting, concatenating, and averaging. For our model, we used the averaging method. In order to apply the averaging method, we added an additional layer to our network architecture that averaged the outputs of the CNN, Densenet, and VGG16 models[4].

Experimental Setup

After subsampling our data into 200,000 images, we split that sample into a training, testing and validation set. The training and validation sets were used to train the model and the testing set was used to evaluate our model after training. We used Keras Image Generator and Data Generator packages to format the image and labels. The Image Generator package was used to apply a 0.1 zoom, random horizontal and vertical flips, as well as apply a constant filter to each image. Once the images were formatted, the Data Generator was used to combine the images and labels in a suitable format for modeling.

During the training processes, we used Adam as our optimizer with a learning rate of 0.001 and Binary-Cross Entropy as our loss function. The best weights were saved based on the validation score of each epoch. As explained in the previous section, our final model was an ensemble of three different neural network models, created by averaging their outputs.

To evaluate our model, our testing set was used to generate predictions. From there, a weighted multi-label logarithmic loss formula was computed based on the correct labels. For logarithmic loss functions lower scores indicate a better model. Below is a formula for weighted multi-label logarithmic loss formula which was obtained from the RSNA Kaggle competition website and “Understanding binary cross-entropy / log loss: a visual explanation” article from Towards Data Science[1,7].

$$\text{Predicted Label}(p) = \max(\min(p, 1-10^{-15}), 10^{-15})$$

$$H_p(q) = -\frac{1}{N} \sum_{i=1}^N y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

Binary Cross-Entropy / Log Loss

Results

Table III below shows the results of the 3 individual and ensemble model combinations that we tested. As shown, Densenet performed the best individually compared to our custom CNN model and the pretrained VGG16 model, with a training accuracy of 95% and weighted multi-label log loss score of 0.1743. The model that performed the best overall was the ensemble model with our custom CNN and Densenet, with a weighted multilabel log loss of 0.14620. This was an interesting result because it outperformed our 3 individual models as expected but also better than all three models combined. We believe that this is due to the fact the VGG model was trained on 1 epoch as opposed to 5 epochs as with the other 2 models. We were unable to run the VGG model for more epochs due to time constraints.

TABLE III: Model Performance

Model	Accuracy Train Set	Binary Loss	Log Loss
Custom Model	94.28%	0.1998	0.1878
Densenet	95.25%	0.138	0.1743
VGG16	95.41%	0.1795	0.183
Custom Model & Dense net			0.1462
Custom Model, VGG16 & Densenet			0.1474

Summary and Conclusion

Our results confirmed that our ensemble models outperformed their individual components but not have a great overall performance when compared to other models in the Kaggle competition. With a final log loss score of 0.14620 our model was in the 80th percentile of the competition's final leaderboard. Our place on the leaderboard suggests that there is a lot of room for improvement. In future work, we would like to explore many model enhancements such as: creating a custom loss function, training on more images from the data set, using pretrained models with image data more closely related to our data such as Nifty Net, trying different methods for ensembling such as voting or stacking, and combining different neural network architectures like Multi-layer Perceptron or Long Short-term Memory (LSTM).

References

1. RSNA Intracranial Hemorrhage Detection. (n.d.). Retrieved from <https://www.kaggle.com/c/rsna-intracranial-hemorrhage-detection/overview>
2. Yogananthem, G. (2018, July 23). CT numbers, window width and window level. Retrieved from <https://www.slideshare.net/ganesahyogananthem/ct-numbers-window-width-and-window-level>.
3. Das, R., Turkoglu, I., & Sengur, A. (2009). Effective diagnosis of heart disease through Neural networks ensembles. *Expert Systems with Applications*, 36(4), 7675–7680. doi: 10.1016/j.eswa.2008.09.013
4. Mikhaylov, M. (2019, February 21). Ensembling ConvNets using Keras. Retrieved from <https://towardsdatascience.com/ensembling-convnets-using-keras-237d429157eb>.
5. Huang, G., Liu, Z., Maaten, L. V. D., & Weinberger, K. Q. (2017). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. doi: 10.1109/cvpr.2017.243
6. CIABURRO, G. I. U. S. E. P. P. E. (2018). *Keras 2.X Projects: 9 projects demonstrating faster experimentation of neural network and... deep learning applications using keras*. S.l.: PACKT PUBLISHING LIMITED.

7. Godoy, D. (2019, February 7). Understanding binary cross-entropy / log loss: a visual explanation. Retrieved from <https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a>.
8. Das, R., Turkoglu, I., & Sengur, A. (2009). Effective diagnosis of heart disease through Neural networks ensembles. *Expert Systems with Applications*, 36(4), 7675–7680. doi: 10.1016/j.eswa.2008.09.013
9. Gulli, A., & Pal, S. (2017). *Deep Learning with Keras*. Birmingham: Packt Publishing.
10. Jesucristo. (2019, September 20). RSNA Introduction, EDA, Models. Retrieved from <https://www.kaggle.com/jesucristo/rsna-introduction-eda-models>.
11. Mathormad. (2019, November 14). [5th place solution] stacking pipeline. Retrieved from <https://www.kaggle.com/mathormad/5th-place-solution-stacking-pipeline>.