RSNA intracranial Hemorrhage Detection

Jay Caceres Samuel Adeniyi-Ipadeola Ahmet Karagoz

Abstract

We use machine learning algorithms to detect if there exists an intracranial hemorrhage (ICH) in computed tomography (CT) scans and identify its subtypes. To achieve this, we implemented three machine learning models using the RSNA ICH dataset in hopes of finding a good model in achieving a high accuracy detection score. We divided the dataset into two categories. One batch of the dataset was used by using jpg files of the same dataset and the other was the original DICOM scan files. From here, two models were trained on jpg scans and the last was trained on the original scans. The first model was used as a baseline as the goal was to achieve binary intracranial hemorrhage detection. The second model was more of an experiment as the final iteration of the algorithm had 3 various augmentations on the data and a progressive resizing approach on the data. The last model handled the original Dicom. The training and testing of the data were done with the dicoms with a few augmentations to standardize the data We use multilabel accuracy and loss to evaluate our models and for the first two models, we achieved an accuracy of 90.6% and 89.7% respectively. Model 3 tough the training time was extremely long the result was with a loss function of .0838

1. Introduction

2. “Fifty-six patients (12%) were initially misdiagnosed, including 42 of 221 (19%) of those with normal mental status at first contact. Migraine or tension headache (36%) was the most common incorrect diagnosis, and failure to obtain a computed tomography (CT) scan was the most common diagnostic error (73%). Neurologic complications occurred in 22 patients (39%) before they were correctly diagnosed, including 12 patients (21%) who experienced rebleeding.”

Intracranial hemorrhage (ICH), bleeding that occurs inside of the skull or cranium, is a life-threatening emergency that can occur to anyone at any stage in their life. It requires rapid and active medical treatment and cares for the patient who has suffered an intracranial hemorrhage. Detecting intracranial hemorrhages is crucial and important when trying to diagnose a patient with this medical phenomenon and it is even more critical when trying to come up with a treatment. In the United States, around 20,000 people die from ICH every year and the 30-day mortality rate is 44%. Additionally, about 8-13% of all strokes are from intracranial hemorrhages[1], 40% of which are a risk of death, and causes may vary from patient to patient. Thus, It is important to identify the location and type of hemorrhage that the patient experienced since treating the patient is vital for their diagnosis, treatment, and recovery.

For this research, detecting acute intracranial hemorrhages is set to occur with several set categories: Intraparenchymal, intraventricular, subarachnoid, subdural, and epidural. Highlighting the different subtypes of ICH, an epidural hemorrhage occurs between the dura and the skull. A subdural hemorrhage occurs between the dura and the arachnoid membrane, and the source of bleeding is often from the bridging veins. Subarachnoid hemorrhage occurs in the subarachnoid space and is often the result of the cerebral artery. Intraventricular hemorrhage is bleeding that is anywhere inside the ventricle of the brain, and it is usually secondary following another hemorrhage. Intraparenchymal hemorrhage occurs anywhere inside the brain tissue

Classification among these subcategories is necessary to accurately determine precisely what type of ICH occurs with each patient. Since ten percent of stroke patients die due to intracranial hemorrhages how quickly Doctors may discern what subcategories the patient’s ICH falls under will result in a favorable outcome to the patients’ health. Traditionally discerning the subcategory of ICH has had different accuracy depending on the experience along with other interpretive variables of the physician. One example is of a study where the misdiagnosis of Subarachnoid hemorrhages of 56 patients resulted in

decreased quality of life at 3 months and an increased risk of death/ severe disability at the 12-month mark. Using Machine learning and Deep Learning algorithms, the Radiological Society of North America (RSNA) is looking to use some of the technology in the predictive analysis world of data science to reduce fatally in patients with intracranial hemorrhages.

3. Related Work

Deep learning has been used greatly recently in the medical industry from trying to detect pneumonia in scans to tumor detection. In the case of intracranial hemorrhage detection, the use of convolutional neural networks has been extensively used. Because of the dire need to process head CT scans in an efficient and accurate manner, there have been many instances where research has been made in developing an efficient model to process images. One model made specifically for image classification is the Resnet 50 architecture[2]. Additionally, there are other papers such as Dipam Vasani et al. which provide advice as to how to improve a convolutional neural network such as data augmentations and resizing approaches in hopes to achieve an accurate model[3]. With all of this in mind, we decided to build 3 models following a similar approach. However, for our first two models, we decided to opt out of using pre-trained weights and just apply the corresponding architectures of Resnet 50. Because the models were trained on images and not medical scans, we decided to see if this would have an impact on the way the model learned. However, because of the lack of pre-trained weights, the advice from Dipam Vasani et al. of progressive resizing was implemented and the goal was to use progressive resizing as an alternative to pre-trained weights. Model 3 was trained on medical images using two models, The ResNet50 and Inceptionv3 our weights were pre-trained and adjusted accordingly the weights and learning rates selected had shown positive results. The main difference between model 3 and the first two models is, we added another model engine. Instead of the ResNet50, we include Inceptionv3. Also, model 3 was the only model trained with actual medical images.

3. Proposed Methods

In this project, we fit three different kinds of models in an attempt to find the best approach in detecting intracranial hemorrhages and their subtypes. The first model served as a baseline for all the other models. We started to build this model as a beginning point that would set a standard of comparison between the other models.

3.1 Model 1 Convolutional Neural Network

Convolutional neural networks are known for image classification through the use of feature extraction. Images that are run through a CNN algorithm are passed through various kernels where features that are relevant when being trained versus true values are extracted. Because of this, convolutional neural networks would work best in being able to successfully detect intracranial hemorrhages and specifically, the subtypes. Therefore, we trained the first model using a Resnet18 architecture with our metrics being error rate and accuracy. Since the goal of the first model is to accurately detect the presence of an intracranial hemorrhage, the first model was made with binary classification in mind. Additionally, the initial decision of not using pre-trained weights and just the Resnet18 architecture was done for the sole purpose of comparison between models that used pre-trained weights in the literature.

3.2 Model 2 Convolutional Neural Network

The second model was also done with a convolutional neural network. However, this model used the Resnet50 more robust architecture in hopes of achieving better accuracy. In addition, the objective of this model was to differentiate between the different subtypes of intracranial hemorrhages. Therefore, the metric we used was a multi-label accuracy metric from Fastai[4]. The first step is to decide which threshold to use for this metric since there would be various labels assigned to an image for classification. With this in mind, we would want to use a threshold of 0.5 to be conservative in the model's prediction and to be confident if there exists an intracerebral bleed. On top of this, the reason for not using pre-trained weights is the same as the other model.

3.3 Model 3 Convolutional Neural Network

Model 3 method was by Convolutional Neural Network with the Dicom files dataset The Resnet50 scores were somewhat close to the Inceptionv3. the Inceptionv3 only did slightly better than the ResNet function. Multiple label prediction was applied due to time constraints to try the model due to the size of the Dataset. This is something to carry on to continue to explore the modeling on this subject. The weights used were the imagenet values by Keras.

4. Experiments

The original dataset we used is The RSNA Intracranial Hemorrhage from the Radiology Society of America, the Dataset is a collection of Dicom, a medical image data that imprints CT scans onto Plastic surfaces. The Dicom file carries the highest amount of pixels being that it is the original Data. The CT scans also have a higher quality when compared with MR and XR files. [figure B]

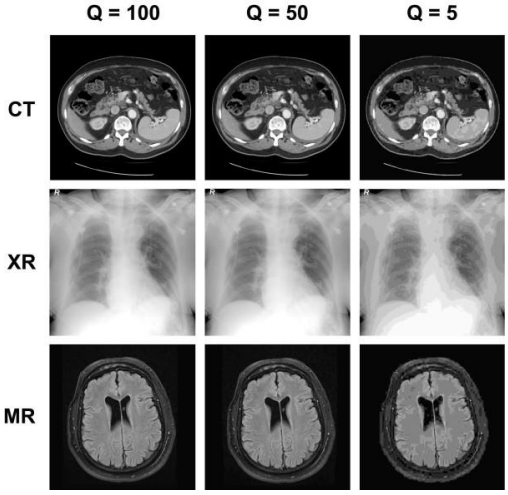
Table 2: Distribution of Hemorrhage Label Subtypes by Examination and Images for Both the Training and Test Sets

Subtype	Training Set Images	Test Set Images	Training Set Examinations	Test Set Examinations
Any hemorrhage type	107 933	15 902	8889	1243
Epidural	3145	208	354	23
Intraparenchymal	36 118	5468	5324	758
Intraventricular	26 205	4546	3692	616
Subarachnoid	35 675	4908	3936	528
Subdural	47 166	6555	3814	503
None	644 870	105 330	12 895	2285
Total	752 803	121 232	21 784	3528

Note.—The number of labels exceeds the actual number of examinations and images because more than one label may have been applied.

[figure A]

Another dataset that we used, specifically for models 1 and 2, is a subset of the same RSNA intracranial hemorrhage dataset. However, this specific subset of images is in jpeg form. This dataset has already been processed from the DICOM form and has been saved and turned into jpeg image files for training specifically. This dataset has 194,000 different jpeg images and is all in the size of 256x256.



[figure B]

4.1 Model 1

Since this model was used for a baseline and a means of comparison between the other models, there was really not much experimentation done for this model specifically.

We started out by loading the data into our corresponding workspace. The initial dataset provided the jpeg files and a CSV file with corresponding labels. After that, we split the image jpeg files into training and testing. To do this, we use a function called ImageDataLoaders.from_df which takes in several arguments one being the CSV file, the path to the jpeg files, batch size, which in this case we chose a batch size of 256, and seed number for reproducibility. We then used the function from Fastai called cnn_learner which takes in the split data from ImageDataLoaders.from_df[4], the architecture for the convolutional neural network which in this case is Resnet 18, the pre-trained weights boolean value to false, and our accuracy metric. We then let the learner train with our data and validate our data for 5 epochs with a learning rate of 3e-3.

4.2 Model 2

This multilabel classification model was the start of our experimentation in trying to build a model that would detect an intracranial bleed, specifically what type of bleed. There were a couple of iterations to this model all in hopes to achieve an accurate model.

The setup of this model was similar to model 1 except that the images were resized to 128x128 and the training and testing were split using Sklearn with a 70/30 split. However, the architecture of this model was initially a custom architecture using Keras. We initially also used the Adam optimizer with a learning rate of 1e-2 and a decay of 1e-6. However, we encountered that the model was learning too quickly and the loss was quickly crossing the zero mark. Because of this, we experimented with a learning rate of 1e-4. Binary cross entropy was used as the loss function.

```
Model: "sequential_9"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_44 (Conv2D)           (None, 128, 128, 32)      896

 conv2d_45 (Conv2D)           (None, 128, 128, 32)      9248

 max_pooling2d_22 (MaxPoolin  (None, 64, 64, 32)        0
 g2D)

 conv2d_46 (Conv2D)           (None, 64, 64, 64)        18496

 conv2d_47 (Conv2D)           (None, 64, 64, 64)        36928

 max_pooling2d_23 (MaxPoolin  (None, 32, 32, 64)        0
 g2D)

 conv2d_48 (Conv2D)           (None, 32, 32, 128)       73856

 conv2d_49 (Conv2D)           (None, 32, 32, 128)       147584

 max_pooling2d_24 (MaxPoolin  (None, 16, 16, 128)       0
 g2D)

 flatten_7 (Flatten)          (None, 32768)             0

 dense_6 (Dense)              (None, 128)               4194432

 dense_7 (Dense)              (None, 5)                 645

=================================================================
Total params: 4,482,085
Trainable params: 4,482,085
Non-trainable params: 0
_________________________________________________________________
```

The second iteration of this model was done with Fastai. The setup is almost identical to the first iteration of this model. However, we needed to use a delimiter since there was a semicolon separating the labels and the batch size was decreased to 64. We then decided to change the convolutional neural network architecture to a more robust one which was Resnet 50. On top of this, we wanted to augment the dataset. To achieve this, we added an argument to our cnn_learner that would add transformation on certain batches of our dataset. On this iteration, we introduced a flip and a rotation of -30 to 30 degrees to certain batches of the data since flipping the CT scan image would not really affect the integrity of the scan nor would it make the images lose any parts of the scan like normal image data augmentation such as a random crop would. The flip was applied randomly with a probability of a flip for the batch of 0.2 We then trained the model for 5 epochs with a learning rate of 3e-2

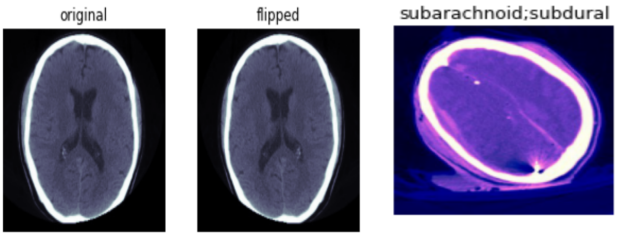


Figure 1. Flipped and rotated image

For the final iteration of this model, we wanted to see if we could improve the previous iteration's accuracy by introducing progressive resizing. We decided previously to use the Resnet 50 architecture but not the weights because these models were trained on random images and not medical scans. Because of this, we decided to use progressive resizing as an alternative way or version of using pre-trained weights. The idea was to familiarize the model with batches of the data in smaller sizes initially starting with 64x64 images and progressively increasing the size. To achieve this, we had to make a function that would take in the size and a batch size which would return the batched data in the appropriate size. We also introduced a new data augmentation that would increase the contrast of the scans. That way, we could hope that the important features would be more differentiated. Finally, we started to train the model for 5 epochs with a learning rate of 2e-2 and 64x64 image scans and a batch size of 512. We then trained the model again, increasing the image scan sizes to 128x128, a batch size of 256, and decreasing the learning rate to 1e-3 for 5 epochs. Lastly, we trained the model, increasing the image scan sizes to 256x256, a batch size of 64, and decreasing the learning rate to 1e-4 for 4 epochs.

4.3 Model 3

…

Model 3 was used using the Dicom file's original size 256x256. Due to some inconsistency in the Dicom data, the files had to be standardized before being trained. Some Dicom backgrounds were white while others were black. Before training, we changed the Dicom file to have a black background to increase our accuracy. The data was split to 90/10 with the Sklearn function. For our Engine, we used the Keras ResNet50 and Inceptionv3; these are two separate models However it seems their results were very similar; they've been grouped together. For both models, the batch size was set to 32 with the learning rate decreasing from 3e-3 to 5e-4. initially, the models were trained to 5 epochs with each epoch estimated to take over 2 hours. Due to training constraints, we reduced the epoch to 4 saving time.

```
# lets go for the first fold only
train_idx, valid_idx = next(ss)

# obtain model
model = MyDeepModel(engine=ResNet50, input_dims=(256, 256, 3), batch_size=32, learn
                    num_epochs=4, decay_rate=0.8, decay_steps=1, weights="imagenet

# obtain test + validation predictions (history.test_predictions, history.valid_pr
history = model.fit_and_predict(df.iloc[train_idx], df.iloc[valid_idx], test_df)
```

```
/opt/conda/lib/python3.6/site-packages/keras_applications/resnet50.py:265: UserWarning: The output
en changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resne
h5
94658560/94653016 [==============================] - 3s 0us/step
Epoch 1/4
 360/7871 [>.............................] - ETA: 2:03:55 - loss: 0.1835 - weighted_loss: 0.2048
```

The weights were set to imagnet which indicates that we want to use the pre-trained ImageNet weights for the respective model. on average the model was trained for over 8 hours ResNet50 had a loss value of .0825 and a weighted loss of .0908 while InceptionV3 had .0762 and a weighted loss of 0.0838

```
# train set (00%) and validation set (10%)
ss = ShuffleSplit(n_splits=10, test_size=0.1, random_state=42).split(df.index)

# lets go for the first fold only
train_idx, valid_idx = next(ss)

# obtain model
model = MyDeepModel(engine=InceptionV3, input_dims=(256, 256, 3), batch_size=32, learning_rate=5e-4,
                    num_epochs=4, decay_rate=0.8, decay_steps=1, weights="imagenet", verbose=1)

# obtain test + validation predictions (history.test_predictions, history.valid_predictions)
history = model.fit_and_predict(df.iloc[train_idx], df.iloc[valid_idx], test_df)
```

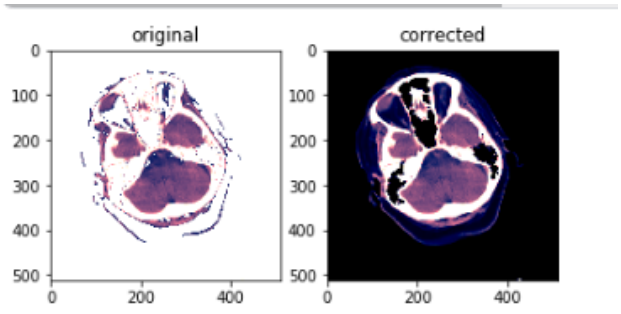
```
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.5/inception_v3_weights_tf_dim_ordering_tf_kernels_notop
87916544/87910968 [==============================] - 0s 0us/step
Epoch 1/4
7869/7869 [==============================] - 7439s 945ms/step - loss: 0.1193 - weighted_loss: 0.1322
Epoch 2/4
7877/7877 [==============================] - 4536s 576ms/step - loss: 0.0944 - weighted_loss: 0.1043
Epoch 3/4
7862/7862 [==============================] - 4451s 566ms/step - loss: 0.0845 - weighted_loss: 0.0934
Epoch 4/4
7875/7875 [==============================] - 4455s 566ms/step - loss: 0.0762 - weighted_loss: 0.0838
```

```
# train set (00%) and validation set (10%)
ss = ShuffleSplit(n_splits=10, test_size=0.1, random_state=42).split(df.index)

# lets go for the first fold only
train_idx, valid_idx = next(ss)

# obtain model
model = MyDeepModel(engine=ResNet50, input_dims=(256, 256, 3), batch_size=32, learning_rate=5e-4,
                    num_epochs=4, decay_rate=0.8, decay_steps=1, weights="imagenet", verbose=1)

# obtain test + validation predictions (history.test_predictions, history.valid_predictions)
history = model.fit_and_predict(df.iloc[train_idx], df.iloc[valid_idx], test_df)
```

```
/opt/conda/lib/python3.6/site-packages/keras_applications/resnet50.py:265: UserWarning: The output shape of `ResNet50(include_top
=False)` has been changed since Keras 2.2.0.
  warnings.warn('The output shape of `ResNet50(include_top=False)` '
Downloading data from https://github.com/fchollet/deep-learning-models/releases/download/v0.2/resnet50_weights_tf_dim_ordering_tf
_kernels_notop.h5
94658560/94653016 [==============================] - 0s 0us/step
Epoch 1/4
7881/7881 [==============================] - 4922s 625ms/step - loss: 0.1289 - weighted_loss: 0.1429
Epoch 2/4
7889/7889 [==============================] - 4532s 575ms/step - loss: 0.1021 - weighted_loss: 0.1127
Epoch 3/4
7861/7861 [==============================] - 4476s 569ms/step - loss: 0.0903 - weighted_loss: 0.0996
Epoch 4/4
7897/7897 [==============================] - 4573s 579ms/step - loss: 0.0825 - weighted_loss: 0.0908
```

5. Results

From all our experimentations and data collected, we determined that the model that performed best was determined by using the Dicom files as training rather than using the jpeg files. Additionally, the CNN with InceptionV3 was the superior approach in trying to detect Intracranial bleeds and the subtypes.



5.1 Model 1 Results

The binary model that only detected if there was an intracranial bleed or not gave us a test accuracy of 90.67% and an error rate of 9.3%

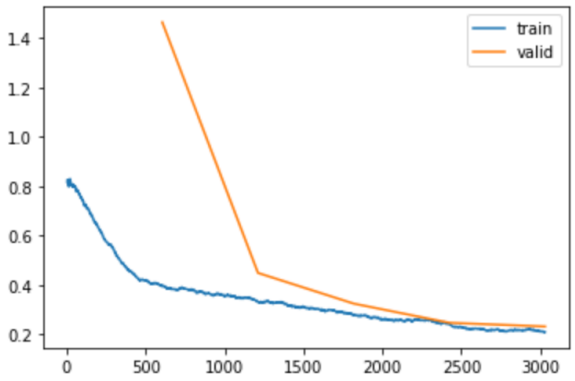


Figure 2. Train and validation loss function graph

5.2 Model 2 Results

The multilabel classification CNN that detected the subtypes of intracranial hemorrhages had an accuracy of 65.91% for the first iteration of the model. This model was with the custom architecture. From all the data that we gathered and by looking at the loss graphs, we concluded that this model was overfitting the training data thus having a mediocre validation accuracy score.

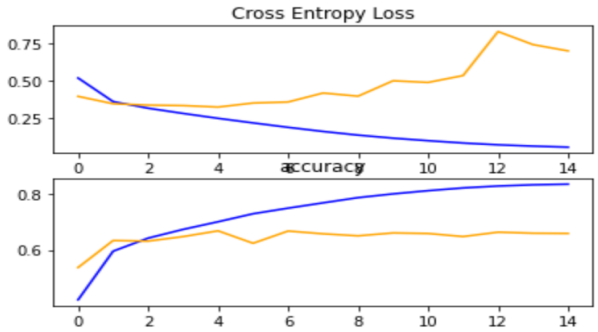


Figure 3. cross-entropy loss and accuracy function graph

For the second iteration of this model, we achieved an accuracy rate of 89.27% which is a vast improvement from the first iteration of the model.

On the other hand, in the third iteration of this model, the first training instance on 64x64 image scans, we achieved an accuracy rate of 78.05% which is still an improvement compared to the first iteration of this model.

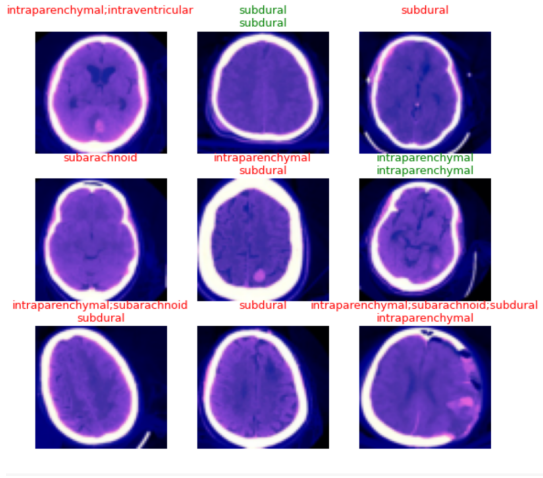


Figure 4. subset of a batch for the 64x64 training instance The upper line is prediction. The bottom line is the true value

By doing the 128x128 training instance for our model, we achieved an accuracy of 88.37%.

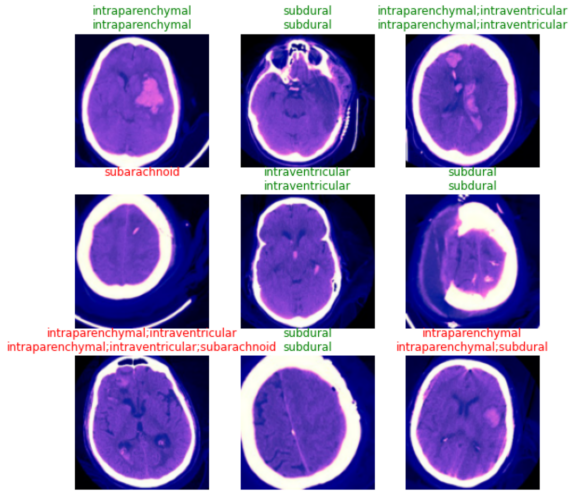


Figure 5. A subset of a batch for the 128x128 training instance

The final training instance for our model was done on 256x256 image scan size and we achieved an accuracy of 89.71% which is a small but nevertheless an improvement from the second iteration of this CNN model.

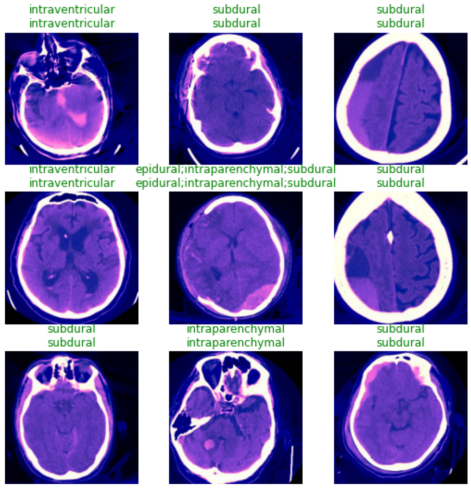
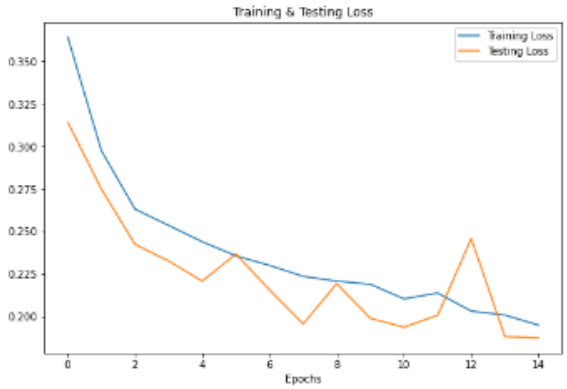


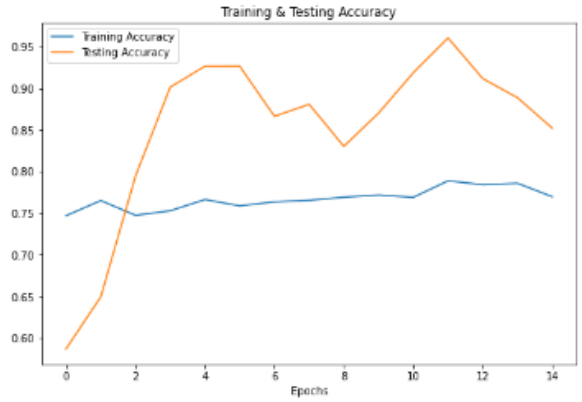
Figure 6. A subset of a batch for the 256x256 training instance

5.3 Model 3 Results

Models InceptionV3 had a metric score of .1043 as weighted loss trained to 2 epochs. When we increased the epoch to 4 the weighted loss decreased to .0838. For the ResNet50, it had a metric score of .1127 as a weighted loss trained to 2 epochs. When we increased the epoch to 4 the weighted loss decreased to .908

This is the final model created with Dicom files. it had an average accuracy of 78.87% due to the limited time and resources this model was not explored any further than this. The Potential to improve on the model is





6. Results Comparison

The sensitivity and positive predictive value of principal diagnosis coding for the source of hemorrhage were typically 80% to 95%. The first model had the best score of 90.67%. This was with 256x256. model 2 For the size 128x128 the score was 88.37% With the size 64x64 the score was 78.05%. model 2 first iteration had a score of 65.91% the second iteration had a score of 78.05%. Model 3 had an accuracy score of 78.87% whereas other models were built with weighted loss as low as .0838. Some other projects found results of 72.3% and 99.4%, they did their experiment with upper endoscopy, they got close results to ours.

7. Conclusion

RSNA Challenge to the Data Science community to export the predictive value of Intracranial Hemorrhage has proven to be a success. In our research, we discovered the potential for the outcome of our models in medical applications for instance images having an MRI machine built with modeling and training capability. Instead of waiting on a radiologist to determine if a patient has an intracranial Hemorrhage. The machine uses pre-trained models from millions of Dataset trained by other devices which share a server to predict each patient's possibility of Hemorrhages. Not only will this process save time but it reduces the Challenge Hospital faces with limited radiologists. There are many ways in which this technology can be extremely beneficial. The model we used for the Convolutional Neural Network was The ResNet 50 and InceptionV3. The Best result he had was with the jpeg file with an accuracy score of 90.67%
The best score for the Dicom file was an Accuracy of 78.87 on the first try. The Model used where also ResNet50 and InceptionV3

8. Contribution

Jay Caceres: Explored jpeg dataset, performed data augmentations on the datasets, and constructed model 1. For model 2, Jay experimented on various approaches to improve the multilabel model where then he applied the progressive resizing approach on top of augmenting the dataset on batches for that specific model.

Temiloluwa Adeniyi-Ipadeola: Explored the Dicom Dataset, Standardized the dicom file. The dicom files were plotted for viewing. Explored different model options and batch sizes. used different optimizers for the models. Model 3 was tested with Inceptionv3 and ResNet50.

Ahmet Karagoz: helped construct of model 1 for base comparison, comparison of all models and similar work results, comparison to real world results

9. Future Works

For Future work, I would recommend taking on the task of model type comparison and for each model Increase the epoch to see how accurate the models can be. I would recommend training on the model on a cloud service. The amount of storage needed to save the dicom files is about 500Gb. To train at a higher epoch, GPU processing is required. The other models built with Dicom files need to be explored further. The result comes from just a few runs and appears to be a limiting factor in drawing insight into the accuracy of our predictive analysis. One way to also take this research further is to start with the best models and shift the weights, learning rate, and epoch values to see if there are ways to increase the accuracy even further. One could even try with a different optimizer. Getting a result of 90% accuracy is greater but how could that value be increased?

References

[1] Liebeskind, David S., and Helmi L. Lutsep. 2018. “intracerebral Hemorrhage: Background, Pathophysiology, Epidemiology.” Medscape Reference. https://emedicine.medscape.com/article/1163977-overview.

[2]K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90.

[3]Dipam Vasani, Prajakta Phadke, Siddhesh Mirjankar, Biraj Parikh, “Detection of Intracranial hemorrhage and its subtypes”

[4] https://www.fast.ai/

[5]Robert G. Kowalski, BS; Jan Claassen, MD; Kurt T. Kreiter, PhD; et al. Initial Misdiagnosis and Outcome after Subarachnoid Hemorrhage, February 18, 2004, *JAMA.* 2004;291(7):866-869. doi:10.1001/jama.291.7.866 https://jamanetwork.com/journals/jama/fullarticle/198199

[6]Cooper, G. S., Chak, A., Lloyd, L. E., Yurchick, P. J., Harper, D. L., & Rosenthal, G. E. (2000). The accuracy of diagnosis and procedural codes for patients with upper GI hemorrhage. Gastrointestinal endoscopy, 51(4 Pt 1), 423–426. https://doi.org/10.1016/s0016-5107(00)70442-1