RSNA Intracranial Hemorrhage Detection

Iva Petkov*
Chhavi Singal*
ivpetkov@ucsc.edu
csingal@ucsc.edu
University of California, Santa Cruz
Santa Cruz, CA

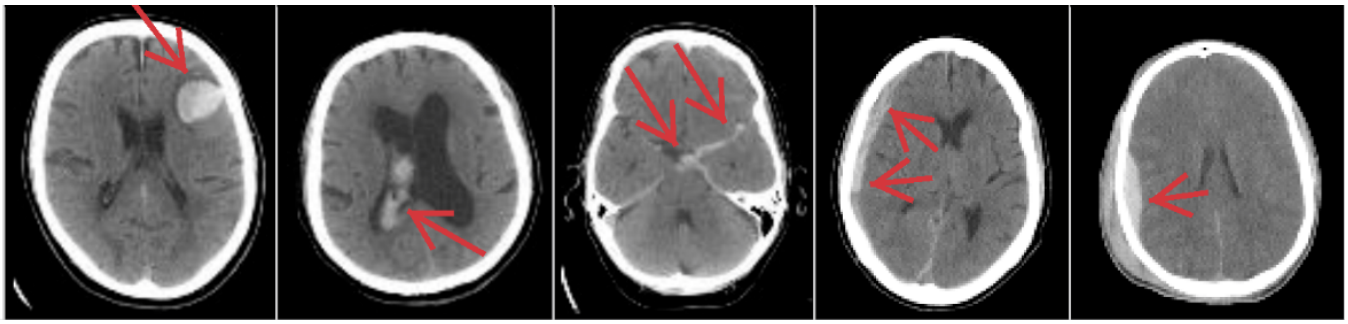

Figure 1: Example of intraparenchymal, intraventricular, subarachoid, subdural, and epidural intracranial hemorrhages.

ABSTRACT

The purpose of this project, proposed by the Radiological Society of North America (RSNA), is to build a classification model that can accurately predict the probability of both the existence and subtype of an intracranial hemorrhage. We built two models, a binary classification model to detect the presence of a hemorrhage and a multiclass classification model to predict the subtype. We used transfer learning using the pretrained ResNet50 model with ImageNet input weights and were able to achieve an 89% test accuracy on the binary model and a 54% test accuracy on the multiclass model.

KEYWORDS

datasets, neural networks, transfer learning, machine learning, classification, hemorrhages

1 MOTIVATION AND OBJECTIVE

Intracranial hemorrhages, or bleeding inside the brain, are a serious and sometimes deadly health problem that require rapid identification and treatment. Identifying the location and type of hemorrhage is a critical step before patient treatment can begin as consequences and treatment can vary extensively based on the size, type, and location of the hemorrhage. The role of a radiologist is to detect and characterize the hemorrhage and determine if its condition is serious enough to require immediate surgery. However, this process can require a lot of time, which for many patients is already limited. By creating a model that can identify the presence and type of a hemorrhage using a computed tomography (CT) scan, we can make this process more efficient. This will help the medical community treat patients more effectively.

The main identifying features of a hemorrhage on a CT scan are its density (new hemorrhages appear white), location, shape, and proximity to other cranial structures. To accurately identify these hemorrhages, our model needed to extract these features. There are various existing pretrained deep neural networks for such computer vision problems, such as VGG-19, Inception V3, Xception, and ResNet50. These networks are able to generalize images outside the ImageNet dataset via transfer learning, through methods such as feature extraction and fine-tuning. ImageNet is a visual database that categorizes over 14 million images into 22,000 separate object categories. These images have been hand-annotated to indicate the objects pictured. By using these input weights, we were able to extract features to use on our dataset, which decreased our model's training time. Furthermore, a pretrained model provides a useful starting point because the features learned while training on the old task can be useful for the new one.

The VGG-19 model is very simple; it uses only 3x3 convolutional layers stacked on top of each other. On the other hand, the ResNet50 model relies on micro-architecture modules instead of sequential network architectures, which allows the model to train using standard stochastic gradient descent without worrying about the vanishing gradient problem. Furthermore, even though this model contains 50 layers, the overall model size is still smaller than VGG-19 because of the use of Global Average Pooling layers rather than fully connected layers. Additionally, the ResNet50 model was more helpful for our problem since we were able to find many examples of it being used for transfer learning as compared to our other pretrained model options. Even though the project calls for one model, we determined, in order to fully encapsulate the scope of the classification problem, we needed to implement two models; this differentiates our solution from other similar approaches to this type of classification problem.

*Both authors contributed equally to this research.

2 DATASET

The dataset provided by the RSNA contains images of CT scans in DICOM format. DICOM, or Digital Imaging and Communications in Medicine, is a medical standard for communication of images and related data. Before we could begin the training phase, we needed to convert these images to PNGs or JPGs. We were able to utilize a pre-converted dataset that maintained the quality of the images.

The PNG dataset is split up into two parts: stage_1_train.csv file and the stage_1_train directory. The stage_1_train.csv file has two columns: the ID for the image in the stage_1_train directory and the type of hemorrhage. The stage_1_train directory contains all the images that we will be using to train our model. We ran a script on these files in order to create a directory for each hemorrhage type, which we then uploaded to Google Drive in order to access this data in Google Colab. Since we were not presented with the test dataset, we performed an 80:20 split on each hemorrhage directory to create separate train and test directories. Each directory contains 1600 train images and 400 test images. Before we could input this data into our model, we had to preprocess it. First, we performed a 90:10 split on our train dataset to create train and validation sets, leaving 1440 images for train and 160 images for validation. Each time we loaded the data, the validation and train sets were randomly chosen. Then we loaded images as arrays of size 224x224 pixels because the ResNet50 model is trained on images of this size.

For the binary classification model we encoded the labels as 0 ('no') or 1 ('yes'). Then we created generators for each set - train, validation, and test - but for the train set we used Image Data Generator to perform transformations on each image, such as stretching, zooming, and more. This creates multiple batches of tensor image data not only allowing us to increase our train set, but also to apply real-time data augmentation. For the multiclass classification model we encoded each of the five labels as numeric values from 0 to 4 and then converted those to one hot encoded labels. This ensured that there was no bias towards higher values. Then we created the same type of generators as the binary model and performed the same data augmentation on the train set.

3 MODELS AND ALGORITHMS

For our problem, we decided to use two different models: binary classification and multiclass classification. Both models begin with the same structure: transfer learning using the pretrained ResNet50 model with ImageNet input weights. We used transfer learning because we did not have enough data to extract features from our images, so using this allowed us to transfer weights from a preexisting data pool to our model. Without transfer learning, we would have to create a new convolutional neural network (CNN) every time we wanted to identify a different object. This can be expensive as we have to recollect training data and rebuild models. To effectively apply this, we needed to transfer the basic features of an image such as shape and illumination. These are particularly helpful for our problem because these are the two main features we look at when identifying hemorrhages. For the binary model, the illumination feature was particularly important because hemorrhages appear lighter than the rest of the cranial structures in the CT scan. For the multiclass model, since we already knew a hemorrhage existed in the scan, the location and shape features were more important for the model to extract.

When we loaded the ResNet50 model, we froze all of its layers except the Batch Normalization layers. By making these layers trainable, we allow weight updates to happen during training. We also do not load the ResNet50 model's final fully connected layer because we add our own so that we can update the parameters of our dataset.

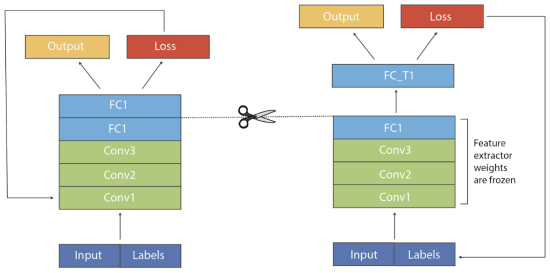

Figure 2: A visualization of our transfer learning technique. FC1 = fully connected layer 1, FC_T1 = fully connected transfer layer 1 (our added fully connected layer).

Before we added this final layer, we added a Global Average Pooling layer, which has no trainable parameters and calculates the average output of each feature map in the previous layer, the last fully connected layer of the ResNet50 model. This reduces the data significantly and prepares the model for the final classification layer. Then we add our final fully connected layer, which is a dense layer with 256 nodes and a relu activation function. Then, as a regularization method, we use a dropout layer with a fraction rate of 0.4; this makes the training process noisier which forces nodes within this layer to take on more or less responsibility for the inputs. Then we add a Batch Normalization layer which standardizes the inputs to a layer for each mini-batch. This not only stabilizes the learning process, but also reduces the number of training epochs required to train the networks. It also reduces overfitting because it adds noise to each hidden layer's activation.

Then, we added the final output layer, which is where the differences between our two models begin. For the binary model, our output layer consisted of a dense layer with one node and a sigmoid activation function since we are classifying the images as either containing a hemorrhage or not. When we predicted the class of an image, this model outputted a value between 0 and 1, which can directly be translated to the probability of the CT scan containing a hemorrhage. Our metrics to evaluate this model were accuracy and binary cross entropy loss. For the multiclass model, our output layer consisted of a dense layer with five nodes and a softmax activation function since we had multiple classes. When we predict on this model, it outputs five probabilities, one for each hemorrhage subtype. Our metrics for this model were accuracy and categorical cross entropy loss.

Our binary model has a total of 24,113,537 parameters. Of these, 23,587,712 parameters come from the ResNet50 model in which 52,608 are trainable, due to the Batch Normalization layers, and the rest are not. From the layers we add at the end, we also add 525,825

more trainable parameters. Our multiclass model has a total of 24,114,565 parameters. Of these, 23,587,712 parameters come from the ResNet50 model in which 52,608 are trainable, due to the Batch Normalization layers, and the rest are not. From the layers we add at the end, we also add 526,853 more trainable parameters.

Considering all the classes we are provided with are not mutually exclusive, we explored creating a multilabel multiclass classification model. In this model, the two labels are the existence and absence of a hemorrhage and the classes are the subtypes, provided the hemorrhage exists. This would be the ideal model because it would take into account the correlation between the subtypes and the existence of a hemorrhage on a single input. With two different models, we can only detect one or the other. However, due to time constraints, we were unable to finish implementing this model.

4 RESULTS AND ANALYSIS

4.1 Binary Classification Model

If this model were to randomly guess 'yes' for each input, the validation accuracy would have been 83%. However, as shown in Figure 5, both the validation and test accuracies reach 89%. In Figure 3, we can see that the accuracies of both train and test improve over epochs. This model is biased towards the 'yes' label since we have five times the amount of 'yes' versus 'no' data and the labels are encoded numerically. It places a greater weight on higher values and, due to more data, learns the existence of a hemorrhage better than the absence.

However, to test this, we also ran our model on a small, evenly split dataset and were able to achieve an 83% validation accuracy. If this version of the model were guessing one label randomly, the accuracy would be 50%. Since it is not, the model is still able to extract features from the input samples, but using more data would help create a more reliable model. The ideal version of this model would contain large, even amounts of data for each label, but since we were restricted to the samples in our dataset, we used whatever was available to us, even if it meant a larger amount of data for the 'yes' label.

To evaluate this model, we used binary cross entropy log loss, which measures the performance of a classification model with only two outputs; as the predicted probability diverges from the actual label, the loss increases. As seen in Figure 4, the loss of our model decreases over epochs so it is getting better at classifying the existence of a hemorrhage. A loss value closer to zero implies a more accurate model and since our loss is still above 0.200, there is room for improvement in our model. Another evaluation metric we utilized was a confusion matrix to identify the amount of correctly classified inputs. In Figure 6, we can see that the majority of 'yes', or 1, labels are predicted accurately compared to 'no' labels.

4.2 Multiclass Classification Model

In this model, since we have five classes, each with equal amounts of data, if our model were to randomly guess one class, the accuracy would have been 20%. Since our accuracy for the train, validation, and test datasets are greater than that, as seen in Figure 9, this model is able to learn and distinguish between the features of each subtype. In Figure 7, the accuracy of our model improves over epochs, supporting that our model is learning. Although the value of our accuracy is low, 54% for the test set, we believe it is still valid because our model is not randomly guessing the labels. Furthermore, this classification problem is very difficult since the differences between the subtypes are minute and can be hard to distinguish, even for a human.

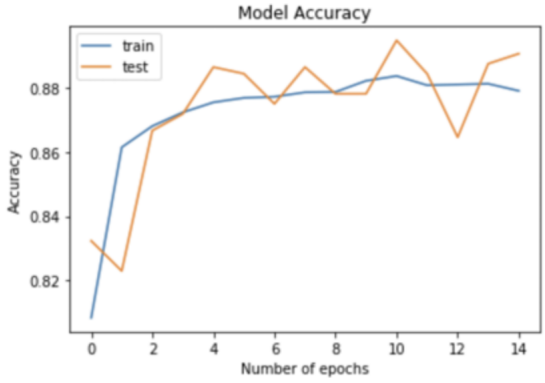

Figure 3: Accuracy versus Number of Epochs of the Binary Classification Model

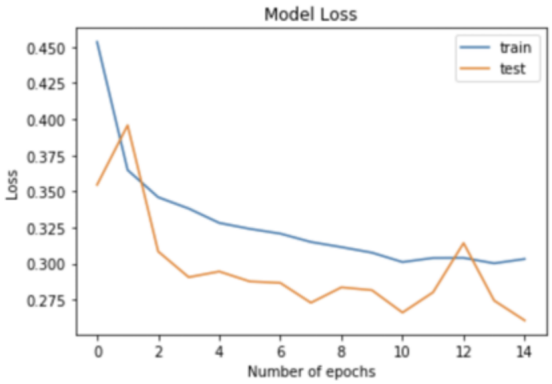

Figure 4: Loss versus Number of Epochs of the Binary Classification Model

Loss and Accuracy				
	Loss		Accuracy	
	Start	End	Start	End
Train	0.45	0.30	0.81	0.88
Validation	0.35	0.26	0.83	0.89
Test	0.27		0.89	

Figure 5: Loss and Accuracy of Train, Validation, and Test Sets for the Multiclass Classification Model

Confusion Matrix			
		Predicted labels	
		0	1
Actual labels	0	57	343
	0	257	1743

0 = no, 1 = yes

Figure 6: Confusion Matrix for the Multiclass Classification Model

The evaluation metric we used for this model is categorical cross entropy log loss, since we had more than two exclusive classes, a softmax activation function, and used one-hot encoded labels. The loss of our model is still quite high, but as seen in Figure 8, it is trending downward. We can see from the confusion matrix in Figure 10, the model is able to classify some inputs accurately. However, the low overall accuracy stems from how closely related the classes are. The model would ideally be able to learn the features for each class and predict the subtypes more accurately if we had more training data.

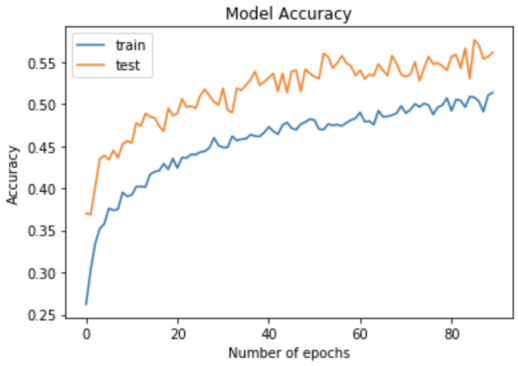

Figure 7: Accuracy versus Number of Epochs of the Binary Classification Model.

5 ISSUES

(1) Originally, the dataset we were provided with contained images in DICOM format, but the ResNet50 model takes in arrays of pixels, which we could only convert PNGs and JPGs to in Google Colab. To do so, we tried a few different methods. Various online conversion methods would not let us batch convert the images and doing so individually was not efficient. We tried to convert images through the command line using packages such as med2image, Imagemagick and various python scripts we found on Kaggle forums; however, these had their own range of issues such as not preserving the quality of the images. We also managed to find a possible solution that entailed taking a screenshot of the DICOM image and then converting it to a JPG; however, this method is time consuming. Ultimately, we were able to find a pre-converted dataset uploaded by a Kaggle user who was working on Stage 1 of the same project, which provided a solution for this issue. However, since this was the only pre-converted dataset we could find, we were limited to the amount of images in that set.

(2) Initially we planned to create a six class classification model, with the classes being the five subtypes and the sixth class would be the 'no hemorrhage' type. However, we realized that this represents each class as mutually exclusive, but

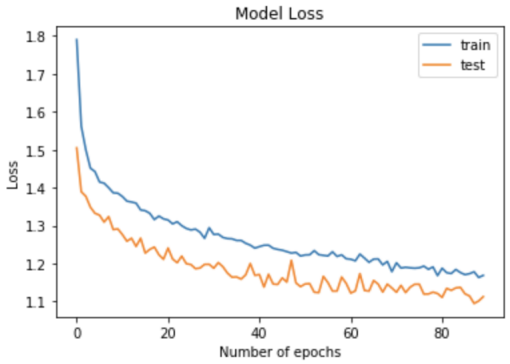

Figure 8: Loss versus Number of Epochs of the Multiclass Classification Model

Loss and Accuracy				
	Loss		Accuracy	
	Start	End	Start	End
Train	1.79	1.17	0.26	0.51
Validation	1.50	1.11	0.37	0.56
Test	1.14		0.54	

Figure 9: Loss and Accuracy of Train, Validation, and Test Sets for the Multiclass Classification Model

Confusion Matrix						
		Predicted Labels				
		0	1	2	3	4
Actual labels	0	86	66	78	95	75
	1	87	56	73	94	90
	2	84	69	74	90	83
	3	91	63	81	94	71
	4	76	77	72	81	94

0 = epidural, 1 = intraparenchymal, 2 = intraventricular, 3 = subarachnoid, 4 = subdural

Figure 10: Confusion Matrix for the Multiclass Classification Model

the classes of subtypes are closely related to each other and are also dependent on the existence of a hemorrhage. To solve this, we split our model into two models, binary and multiclass, because in medical diagnoses, it is more important to at least know the existence of a hemorrhage even if you do not know the type, so that some form of treatment can begin. As mentioned earlier, we also explored creating a multilabel multiclass classification model, which would solve the mutual exclusion issue, but due to time constraints we were unable to finish its implementation.

(3) At first, the parameters we passed into Image Data Generator were just flipping our existing image or rotating it, but they were not actually zooming in into the intracranial portion of the image. This left the black background that surrounded the brain. Due to this, our model was just learning the shape of the brain and predicting that rather than actually looking at the hemorrhages. We had to find a way to have our Image Data Generator zoom into the brain so we could zoom into not only the image, but also smaller, more focused parts of the brain. Once we found the right parameters for the Image Data Generator, the accuracy of our model increased.
(4) During the training portion of the first version of our binary model, our training accuracy would increase, but the validation accuracy always fluctuated around 0.5. This happened regardless of the amount of data we put in, the type of hemorrhage we tried the model on, and other optimization techniques to reduce overfitting. We soon realized, the reason for this anomaly was that the ResNet50 model starts the weights from scratch for each epoch, so the model was not actually training. This issue happens when doing transfer learning on a dataset that the original model is not trained on, as was our case. Our solution was to make the Batch Normalization layers in the ResNet50 model trainable, so we could pass weights from one epoch to another.
(5) Throughout this project, we faced many general issues that slowed down our progress. For instance, loading our images and processing them took a long time and Colab would frequently max out the idle time during this portion of our code. Additionally, at some points, our RAM would max out and our entire runtime session would crash, forcing us to start over. This happened so frequently that we upgraded to Colab Pro; however, these issues continued to happen, albeit less frequently. Another difficulty we encountered was our third team member dropping the course a few weeks into our project. Without him, the remaining two members had to take up more work, which was an unexpected obstacle since we still had to implement a majority of our project. Regardless, we were still able to finish our model in the scope of one quarter.

6 CONTRIBUTION

Since we initially started out with three group members, the initial portion of our project was split equally between each of the three members. Before leaving the group, Christopher Gunter sorted the dataset into train and test directories and uploaded this to Google Drive for us to access. For the rest of the project - processing the data, building the model, training, evaluating, writing reports, and creating the presentation - the work was equally split between the two authors.

7 FUTURE WORK

If we were to continue this project in the future, one of the main improvements we would implement is creating the multilabel multiclass classification model. An additional feature that would be useful for doctors would be to place a border box around the hemorrhage, clearly highlighting its location and size. Since our model already outputs the likelihood of the subtype of hemorrhage, these are the only additional features a doctor would need in order to make a diagnosis. With our currently implemented models, some improvements would be to run the models on greater amounts of data, train them for longer, and implement fine tuning, all of which would help us achieve a higher accuracy. Ideally, our current models would also let you predict on one sample with the binary model and, if the final classification is a 'yes', run the same sample through the multiclass model.

A ONLINE RESOURCES

(1) RSNA Intracranial Hemorrhage Detection
(2) RSNA Intracranial Hemorrhage Detection Stage 1 PNG 128x128 Dataset
(3) Transfer Learning with ResNet
(4) Deep Learning Using Transfer Learning
(5) Deep Learning and Medical Image Analysis for Malaria Detection