A Machine Learning Method for the Automatic Classification of Strokes

Henry Liu

Massachusetts Academy of Math and Science at Worcester Polytechnic Institute

Advanced STEM with Scientific and Technical Writing

Instructor: Kevin Crowthers, Ph.D.

Worcester, MA. 01605

Abstract

Strokes are often difficult to detect and classify quickly and require the presence of a doctor for the diagnostic process. As a result, many strokes are detected too late to administer effective treatment, leaving patients with long term disabilities. There are two main types of strokes: hemorrhagic and ischemic strokes. The goal of this project was to create a machine learning algorithm to automatically detect strokes and classify them as either ischemic or hemorrhagic using CT images. Datasets containing prelabeled brain CT scans of stroke patients as well as scans of healthy patients were collected. These images were used to create training and testing data as well as multiple image classification models. The optimal model was then determined using testing and validation accuracy as well as different measurements of loss. Using the testing data, the model was found to be highly effective with an accuracy of 90.26% and a loss value of 0.2210 calculated using sparse categorical cross entropy. The model was able to run efficiently, with an average processing time of 26 milliseconds per 32 image slices. The model could be used with the current process by a physician, speeding up diagnosis times and allowing patients earlier access to the treatment they need. Future additions to the model would be the inclusion of other brain diseases and expanding to other types of scans, such as MRI scans or microwave imaging. Future testing includes testing on more datasets and testing during the real-time diagnostic process.

Keywords: Image classification, stroke detection, machine learning

Acknowledgements

This project was made possible thanks to the family and friends who supported me throughout the profit, to Dr. Kevin Crowthers, who guided me along the much of the process, and to my school and the community at the Massachusetts Academy of Math Science and at Worcester Polytechnic Institute for giving me the motivation to see my project to completion.

A Machine Learning Model for the Automatic Classification of Strokes

Strokes are the leading cause of long-term disability worldwide, primarily a result of patients receiving treatment too long after the initial stroke onset (About Stroke, 2021). Stroke patients suffer from millions of brain cells dying every minute during the stroke before they receive treatment (Timeline of a Stroke, 2017). However, a patient cannot receive treatment until after the stroke has been detected and classified, a process that currently requires the attention and time of a physician. The time needed for the detection and classification is a flaw of the current diagnostic process, as it delays the administration of treatment to the patient.

Strokes and Current Methods of Stroke Detection

Strokes are the second leading cause of death and the leading cause of long-term disability, often causing patients limited mobility if not treated immediately (About Stroke, 2021). Strokes occur when the blood supply to the brain is either interrupted or blocked completely. The two main types of strokes are ischemic (also known as acute strokes) and hemorrhagic strokes. Hemorrhagic strokes occur when a blood vessel in the brain ruptures, resulting in excess blood scattered throughout the brain (Ireland & Bialkowski, 2011). Ischemic strokes are more common, making up about 80% of all stroke cases, and occur when the blood flow to the brain is blocked, typically the result of a blood clot (About Stroke, 2021). In both cases, the stroke causes the death of upwards of two million brain cells every minute after it begins (Timeline of a Stroke, 2017), making it essential for the patient to receive treatment as early as possible. However, since the treatment of a stroke varies depending on whether it is ischemic or hemorrhagic, the treatment process cannot begin until after the diagnosis and classification of the stroke.

Current methods of stroke detection rely on medical imaging, whether that be with Computed Tomography (CT) scans or Medical Resonance Imaging (MRI), and an interpretation of the scans by a physician. Of these two, CT scans are used more often due to its cheaper cost[1] and greater accessibility (Health Images, 2021). CT scans use X-rays from various angles around the patient to generate several cross-sectional images (slices) of the patient (CT scans, 2020). These slices contain information about the patient, such as the parameters of their bones and arteries, and are able to capture the presence of a stroke (Chawla et al., 2009). On such images, hemorrhagic strokes appear as bright white, or hyper dense, spots, while ischemic strokes appear as dark grey, or hypo dense, spots, with its contrast relative to the surroundings depending on the time since the stroke onset.

The primary disadvantage with this method is the amount of time needed for the physician to interpret the image and classify the stroke, delaying the administration of treatment to the patient by more time than necessary. As such, there have been several attempts to automate the process. A method developed by Ireland & Bialkowski (2011) sought to detect strokes automatically through the confocal algorithm[2] in microwave imaging. While this method yielded successful results with the head phantoms that were used for testing, it has yet to be tested on human patients. Methods aiming to detect strokes from CT images, such as those proposed by Chawla et al. (2009) and Chan (2007), typically detect strokes by using the contralateral symmetry of the brain and comparing the two hemispheres. Chawla et al. (2009) generated and compared histogram representations of each of the two hemispheres to determine the presence of a hemorrhagic stroke. Chan (2007) made use of similar methods, but specifically focused on the detection of hemorrhagic strokes. In both methods, the main

[1] CT scans cost an average of $1200 per scan while MRIs cost an average of $2000. However, MRIs produce more detailed images, although this extra detail is typically not necessary for the diagnosis of a stroke (Health Images, 2021).
[2] The confocal algorithm is a method of determining the parameters of a patient's body by sending and receiving echo signals, generating hypotheses about the origin of the signal, and summing the errors. This algorithm has been used primarily in cancer detection to determine the location and size of a tumor (Ireland & Bialkowski, 2011).

disadvantage was a large amount of false positive cases and difficulty detecting strokes spanning across both hemispheres.

Machine Learning and Image Classification

Machine learning is a branch of data analysis concerned with algorithms that learn through example. The field is split into supervised and unsupervised learning. In supervised learning, a model is given both example inputs and the expected output for each of the inputs as training data, allowing for the algorithm to draw inferences about the patterns in the inputs for a given output (Soofi & Awan, 2017). Supervised learning algorithms are either regression models, which have continuous outputs, or classification models, which have discrete outputs (classes). Image classification models sort image inputs, typically in the form of pixel arrays or vectors, into classes by extracting features and information about the image and detecting patterns in the different classes (Kanellopoulos & Wilkinson, 1997).

Several classification techniques have been used to classify images, including decision trees, instance-based learning (such as K-Nearest Neighbor), and support vector machines (Soofi & Awan, 2017). The most used methods of image classification are convolutional neural networks (CNNs), which aim to imitate biological neurons through layers of neurons[3] (IBM, n.d.). In a CNN, the features of an image are extracted through some number of convolution layers[4], each outputting a feature map to be used as the inputs for the next layer (Li et al., 2014). The final outputted feature maps are then flattened and pass through the classification layers, which are made up of layers of interconnected neurons, until going through the final output layer, after which the class of the image is outputted (Saha, 2021). CNNs

[3] Neurons in neural networks are mathematical functions that compute a value to pass to the next layer of a network, usually by applying weights to its inputs. Some commonly used functions are the sigmoid function and the ReLU activation function (IBM, n.d.).

[4] Convolution layers in a neural network extract features from an image by applying filters to detect features, such as edges, in a specific section of the input through the use of activation functions, the most common of which being the Rectified Linear Unit (ReLU). The resulting outputs, known as feature maps, are then subject to a pooling layer, which reduce the dimensions of the feature maps for the next convolutional layer (Saha, 2021).

are commonly used because the architectures of the models are very similar, even for different problems using different classes. As a result, CNNs are being used much more prominently for problems that require image classification and have been applied in the medical field as well, such as by Li et al. (2014) to classify lung patches and detect interstitial lung disease. The main difference between the CNNs used for different tasks are the types of functions used, the number of layers, and the weights used in the model[5] (Li et al., 2014).

Problem Statement

The primary disadvantage of current stroke detection methods is the amount of time needed to interpret the results of medical imaging, delaying the treatment of the patient and increasing the risk that the patient develops lasting effects, such as long-term disabilities.

Objective

The goal of this project was to create an image classification model able to accurately detect and classify strokes using images from CT scans. Medical professionals would be able to use to model to aid in the diagnostic process, decreasing the amount of time needed to classify a stroke and administer treatment to the patient.

The project addressed the problem statement by creating a model able to detect strokes with a high accuracy in a significantly shorter time span than current methods.

[5] The weights used by the neurons in a convolutional neural network are determined when the model is trained using the training images. The model updates the weights to achieve the highest accuracy on the training and validation data, typically through gradient descent or some variation, throughout the training process (Kanellopoulos & Wilkinson, 1997)

Section II: Methodology

Role of Student vs. Mentor

Over the past five months, I was responsible for the research and development of a model to classify strokes, including the design of the architecture, the construction of the model, and the gathering of data.

Equipment and Materials

Ischemic CT scans were obtained from the ISLES dataset in the NIfTI data format (Oskar et at., 2015; Kistler et al., 2013). The scans were taken of ischemic stroke patients at two different medical centers within eight hours of the stroke onset. Each scan contained between two and eight CT slices that the ischemic stroke appeared in, with a total of eighty scans from eighty different patients. The scans consisted of three dimensions each: the first two being the image information of the current slice and the last being the number of the slice. In total, there were 272 image slices from the scans in the dataset.

Hemorrhagic CT scans were obtained from the CQ500 dataset (Chilamkurthy et al., 2018) and a dataset provided by Rahman (2021), which also contained the normal CT scan images. The images provided by Rahman (2021) were in JPEG format and contained prelabeled slices of scans of both hemorrhages and healthy brains. From this dataset, 251 images labeled as hemorrhagic strokes and 251 images labeled as normal scans were used in the model. The data provided by the CQ500 dataset were of CT scan images (slices) in DICOM format. These images were obtained from the Centre for Advanced Research in Images, Neurosciences and Genomics (CARING) in New Delhi, India. All images were of slices containing a hemorrhagic stroke read by three radiologists with 8, 12, and 20 years in Cranial CT interpretation, respectively (Chilamkurthy et al., 2018), and 251 images were used from this dataset. In total, 1025 images were used to train and test the model, with 272 being of ischemic strokes, 502 of hemorrhagic strokes, and 251 of normal brain CT scans.

To construct the model used for the classification of these images, the Keras framework from the TensorFlow platform was used. Keras provided an API that was used for the construction of the neural network. For the analysis and compilation of data, NumPy, CV, and OS were used for array programming, image manipulation, and operating system interactions respectively. All programs written for the construction and testing of the model, as well as for all other aspects of the project, were written using Google Colaboratory, a Jupyter notebook environment run on the cloud that also provided access to GPUs to run the programs.

Data Analysis and Image Equalization

All CT images used were converted into the same data format, equalized, and normalized so that the model would classify them based on image features rather than image formatting. To begin this process, all images were converted into grayscale 256 by 256 two-dimensional pixel arrays with 8-bit data values, meaning the color values were between 0 and 255. The CT scans from the ISLES dataset contained three-dimensional arrays, as each scan contained information pertaining to multiple image slices from the same patient. For this dataset, each scan was split into individual two-dimensional arrays and resized to match the 256 by 256 pixel standard using CV2. In addition, the images contained 16-bit data values, meaning that pixel values ranged from -32,768 to 32,767. In order to convert these values to 8-bit data, the value of each pixel was divided by 128 and increased by 128. This resulted in each image containing pixel values between 0 and 255, meaning that the values in the array could then be converted to 8-bit integer values, which was done using NumPy.

The images provided by the CQ500 dataset were in DICOM format, meaning the images contained both pixel data as well as header information about the patient. Only the pixel data was used for the model, so the pixel array of the image was extracted and converted to 8-bit integer values using the same process as had been done for the images from the ISLES dataset. The dataset provided by Rahman (2021) contained images previously converted to JPEG format with 8-bit integer data values. These images were standardized by converting them to grayscale and resizing them.

All images had been converted to grayscale, two-dimensional pixel arrays with 8-bit data values. However, since the images were taken from different datasets and centers, the images still differed in some key characteristics, such as brightness and contrast levels. To combat this issue, all images were equalized, which adjusted the color values and contrast of each image so that the model would differentiate based on features rather than contrast or brightness. Figure 1 shows an example of an image used before and after equalization. Image equalization was performed on each of the pixel arrays using CV2. The pixel values of each image were then reduced by a factor of 255, normalizing each image to contain pixel values between 0 and 1. At this point, the images could be used for the model, so they were each appended to the set and split into training, testing, and validation data using a 70-15-15 train-test-validation split. Some non-equalized images were also added to the testing set in order to determine if the model is able to perform without the equalization of images.

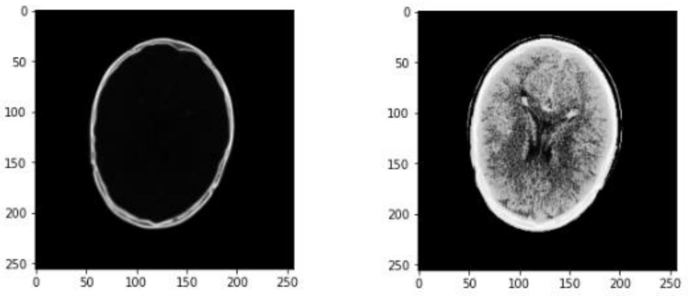


Figure 1: An example of a CT image of an ischemic stroke before histogram equalization (left) and after histogram equalization (right). The operation ensures that the contrast values of the image are evenly distributed among different bins.

Model Construction and Data Collection

Three different models were created, the first two used to test the use of certain layers and the latter as an improvement built based on the disadvantages of the other models. For testing, each model was evaluated on the same testing set containing 154 images, some equalized and others not. The accuracies of the models were then evaluated as well as the loss function value. For this project, the loss of a model was calculated using sparse categorical cross entropy, a function used to determine the performance of a model using the probabilities of each of the predictions (appendix C).

The first model contained three convolutional layers with a 9x9 kernel size and 32 filters, 5x5 kernel size and 64 filters, and a 3x3 kernel size and 128 filters, respectively. Between each layer, a 2x2 max-pooling layer was included to decrease the size of the input into the next layer. Each convolutional layer also used the ReLU activation function in order for the output of each layer to be non-linear. The output of the final convolutional layer was then flattened to a vector containing 15488 values. This flattened result was inputted into two dense layers, the first outputting 64 values using the ReLU activation function and the latter utilizing the SoftMax activation function and outputting the final output of the model, or the probability that the image belongs to each of the classes. The model was trained over 20 epochs using the Adam optimization algorithm (appendix C), during which the model was tested on validation data, as well as on the testing data following the completion of the model training.

The second model was created with a similar architecture to the first model, with the primary difference that max pooling layers were used rather than average pooling layers. The convolutional layers of the second model were the same as the layers of the first model, using the same kernel sizes across three layers, except 16, 32, and 64 filters were used across the three layers instead. The ReLU activation function was used on each of the layers and the SoftMax function was used on the last dense layer as well. This model was also compiled and trained using the Adam optimization algorithm and tested using the testing dataset.

The third and final model was created with the intention of combatting the problem with overfitting. As a result, an additional convolutional layer was incorporated into the model, as well as a dropout layer between the two dense layers, which would disable certain neurons during each of the training epochs. The model was created to have four convolutional layers, each followed by a max pooling layer, and then the flatten, dense, and dropout layers. The parameters of each of the layers were adjusted, optimized, and tested on the validation data during training and the testing data until the optimal parameters were determined. The final model began with 4 convolutional layers, the first applying 32 filters with a 9x9 kernel size, followed by a max pooling layer with a 3x3 window. The second and third layers had a 7x7 kernel and applied 64 filters and were each followed by a max pooling layer with a 3x3 window. The final convolutional layer had a 3x3 kernel and applied 128 filters and was followed by a max pooling layer with a 2x2 window. The output was then flattened and inputted into the first dense layer, which contained the dropout layer with a dropout rate of 50% and followed by the final dense layer which outputted the result of the model. This model was trained over 20 epochs using the Adam optimization algorithm and tested for accuracy and loss on the testing dataset. Details about the training and validation of each model, as well as the architecture of the models, can be found in appendix D.

Statistical Tests

To determine the significance of the third model, McNemar’s test was performed on the results of the testing data from the third model and the testing results of the first model. McNemar’s test was chosen as the output of each model was dichotomous.

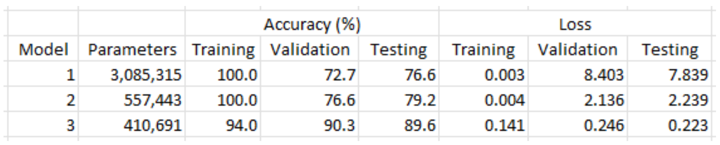
McNemar’s Test

Each image was either correctly classified by both models, neither model, or one of the two models. The values were then used in a contingency table to determine the p-value of the model, and if the p-value was found to be below 0.05, the final model would be determined to be significantly more accurate than the first model.

Section III: Results

The first model, when compiled, contained a total of 3,085,315 trainable parameters. With the training and validation data, the model was able to achieve an accuracy of 100% and 72.7% and loss values of 0.003 and 8.403 respectively. When evaluated using the testing data, the model performed with an accuracy of 79.2% and accumulated a loss value of 7.839. The second model contained 557,443 trainable parameters. The model achieved an accuracy of 100% and 76.6% on the training and validation data respectively, and it accumulated loss values of 0.004 and 2.136. On the testing data, the model performed with an accuracy of 79.2% and a loss value of 2.239. The confusion matrix for this model as well as the first model can be found in appendix E. The final model contained 410,691 parameters and achieved accuracies of 94.0% and 90.3% on the training and validation data with loss values of 0.141 and 0.246. The model performed similarly on the testing data, achieving an accuracy of 89.6% and a loss value of 0.223. The predicted and

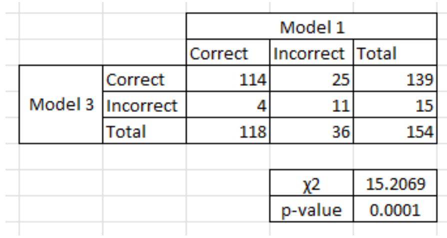
Table 1



		Accuracy (%)			Loss		
Model	Parameters	Training	Validation	Testing	Training	Validation	Testing
1	3,085,315	100.0	72.7	76.6	0.003	8.403	7.839
2	557,443	100.0	76.6	79.2	0.004	2.136	2.239
3	410,691	94.0	90.3	89.6	0.141	0.246	0.223

Table 1 shows the parameters, accuracy, and loss values of each of the models with the training, validation, and testing data. Training and validation data was gathered from the last epoch of the training process.

Table 2



		Model 1		
		Correct	Incorrect	Total
Model 3	Correct	114	25	139
	Incorrect	4	11	15
	Total	118	36	154

χ2	15.2069
p-value	0.0001

Table 2 shows the values used to perform McNemar’s test, as well as the results of the test. The values were obtained by evaluating both models on the same testing data simultaneously and recording the correct and incorrect

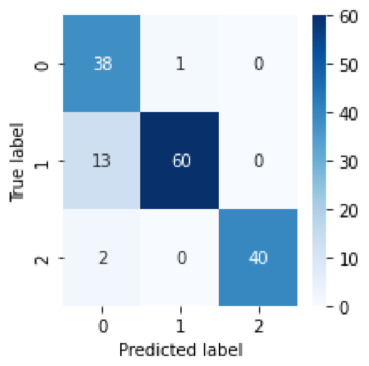


Figure 2: A confusion matrix displaying the predicted and actual classes of model 3 when evaluating the testing data. 0 pertains to normal scans, 1 to hemorrhagic strokes, and 2 to ischemic strokes

actual values of the training set when evaluated using the third model can be seen in figure 2. The output of the final model, if an image had been predicted to be of a normal CT scan, also included the probability that the scan contained some sort of cranial abnormality that may or may not be a stroke.

The comparison test between model 3 and model 1 yielded the results shown in table 2. Both models correctly predicted 114 images and incorrectly predicted 11 images. Model 3 correctly predicted 25 images that model 1 did not, and model 1 correctly predicted 4 images that model 3 did not. McNemar's test was applied on this data and found a chi-square value of 15.21. This value, with one degree of freedom, concluded a p-value of *****0.0001**.

Section IV: Discussion

The goal of the project to design a machine learning model capable of classifying CT scans as either healthy or containing hemorrhagic or ischemic strokes was achieved, with the final model able to classify the CT images with an 89.6% accuracy. The final model was also able to determine the probability of each class and output the probability that a normal scan contained an abnormality, even if it was not a stroke. The confusion matrix in figure 2 showed the predicted and actual labels of each of the 154 images in the testing data, as evaluated using the model. The class predicted with the highest accuracy was of the ischemic strokes, as 40 of the 42 images were correctly identified and none of the images in the other classes were mislabeled as an ischemic stroke. The class with the highest error was of the hemorrhagic strokes, the most common of which occurring when a hemorrhagic stroke was mislabeled as a healthy scan, which occurred in 13 of the 73 images. While only one healthy scan was mislabeled as containing a stroke, the most common error was a stroke being labeled as healthy, which occurred in 15 of the 16 mislabeled cases.

A potential confounding variable was the origin of the images, as the images for the different classes were from different centers, possibly resulting in differing factors such as the contrast and brightness of the images. To address this issue, the images were all equalized before they were used. The most difficult part of the project was processing the images, primarily a result of having to work with different file formats and normalize every image. The adjusting of models was also difficult, as the models had to fitted over the correct number of epochs with different parameters in order to determine the optimal model.

McNemar's test was chosen to determine statistical significance with a critical value of 0.05. McNemar's test was chosen due to the dichotomous nature of the outputs, as they could either be correct or incorrect, and the paired values. The test comparing the first and final model yielded a chi-square value of 15.21 and a p-value of 0.0001. This value was well below the critical value threshold and demonstrated that the final model was a significant improvement over the first model.

This project was meant to be an improvement on similar research, as conducted by Chawla et al. (2009) and Chan (2007). Both projects had high accuracy when used to detect hemorrhagic strokes but had low accuracy in the detection of ischemic strokes. Both methods also utilized the contralateral symmetry of the brain for the classification process, which is limited in the detection of strokes occurring in both hemispheres of the brain. This project also used convolutional neural networks as the method of classification, which was not used in the previous studies. This application may allow for further applications of neural networks in the medical field as a result.

Future Research

Future research and testing includes the improvement of the current modeling using more training and testing data. This data could be of more images containing strokes or images containing other cranial diseases, such as brain tumors or aneurysms. The latter would allow for the model to detect and classify an increased number of brain diseases, although doing so may be at the expense of model accuracy. In addition, future testing would include testing the model during the real-time diagnostic process in order to judge the effectiveness of the model outside of prelabeled images and assess the usefulness of the model to the physician responsible for the diagnosis. Finally, the model could be expanded to include the use of other types of medical imaging, such as MRI scans and microwave imaging, which would allow for the model to used in a wider range of scenarios.

Section V: Conclusion

The objective of the project was to develop an image classification model able to aid in the stroke diagnostic process by automatically detecting and classifying strokes as either ischemic or hemorrhagic, which was ultimately successful. The model was created following the collection of data and image processing and involved repeated testing and configuration to determine the optimal model. This project and the final model demonstrated the viability of the use of neural networks in stroke detection and classification and present the possibility of neural network usage in other medical fields as well.

Section VI: References

About Stroke. (2021, August 02). Retrieved from https://www.cdc.gov/stroke/about.htm

Chan, T. (2007). Computer aided detection of small acute intracranial hemorrhage on computer tomography of brain. *Computerized Medical Imaging and Graphics*, *31*(4-5), 285-298. doi:10.1016/j.compmedimag.2007.02.010

Chawla, M., Sharma, S., Sivaswamy, J., & Kishore, L. (2009). A method for automatic detection and classification of stroke from brain CT images. *2009 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. doi:10.1109/iembs.2009.5335289

Chilamkurthy, S., Ghosh, R., Tanamala, S., Biviji, M., Campeau, N. G., Venugopal, V. K., Mahajan, V., Rao, P., & Warier, P. (2018, March 13). CQ500 [Dataset]. Qure.ai. http://headctstudy.qure.ai/dataset

CT scan. (2020, February 28). Retrieved from https://www.mayoclinic.org/tests-procedures/ct-scan/about/pac-20393675 #:~:text=A computerized tomography (CT) scan

Harris, C.R., Millman, K.J., van der Walt, S.J. et al. Array programming with NumPy. Nature 585, 357–362 (2020). DOI: 10.1038/s41586-020-2649-2

Health Images. (2021, August 30). *MRI vs. CT Scan*. https://www.healthimages.com/mri-vs-ct-scan

IBM Cloud Education. (n.d.). What are Neural Networks? Retrieved from https://www.ibm.com/cloud/learn/neural-networks

Ireland, D., & Bialkowski, M. E. (2011). Microwave Head Imaging for Stroke Detection. *Progress In Electromagnetics Research M*, *21*, 163-175. doi:10.2528/pierm11082907

Kanellopoulos, I., & Wilkinson, G. G. (1997). Strategies and best practice for neural network image classification. *International Journal of Remote Sensing, 18*(4), *711-725*.

Kistler et al. The virtual skeleton database: an open access repository for biomedical research and collaboration. *JMIR*, 2013 http://doi.org//10.2196/jmir.2930

Li, Q., Cai, W., Wang, X., Zhou, Y., Feng, D. D., & Chen, M. (2014). Medical image classification with convolutional neural network. *2014 13th International Conference on Control Automation Robotics & Vision (ICARCV)*. https://doi.org/10.1109/icarcv.2014.7064414

Oskar Maier et al. ISLES 2015 - A public evaluation benchmark for ischemic stroke lesion segmentation from multispectral MRI Medical Image Analysis, Available online 21 July 2016, ISSN 1361-8415 http://dx.doi.org/10.1016/j.media.2016.07.009.

Rahman, A. (2021, June 18). Brain Stroke CT Image Dataset (Version 1) [Dataset]. Afridi Rahman. https://www.kaggle.com/afridirahman/brain-stroke-ct-image-dataset

Saha, S. (2021, December 7). *A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way*. Medium. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

Soofi, A., & Awan, A. (2017). Classification Techniques in Machine Learning: Applications and Issues. *Journal of Basic & Applied Sciences*, *13*, 459-465. doi:10.6000/1927-5129.2017.13.76

Timeline of a Stroke. (2017, November 21). WebMD. https://www.webmd.com/stroke/stroke-symptoms-timeline

Section VII: Appendices

Appendix A: Limitations and Assumptions

Limitations:

- Size of datasets (accuracy of model could have been potentially increased had there been more data to use for testing and training)
- Image variation due to images originating from various facilities and sources
- Computational limitations, although this was mainly negated through the use of Google Colab

Assumptions:

- The models were developed with the assumption that classes were mutually exclusive, meaning no image could belong to two or more classes
- Images were assumed to have been preprocessed and normalized (not equalized) to match orientation before input into the model
- Earlier models assumed all CT scans that did not contain a stroke were healthy

Appendix B: ***Engineering design matrix comparing different methods of image classification***

Criteria	Max	Decision Tree	K-Nearest Neighbor	Support Vector Machines	Convolutional Neural Networks
Suitable with small datasets	6	3	2	5	4
High accuracy of output	10	6	7	7	9
High testing and training speed	9	6	4	8	8
High number of features without affecting speed and accuracy	8	3	6	5	7
Easily improved and configured	7	3	5	5	6
Total	40	21	25	30	34
Percent	100%	53%	63%	75%	85%

Advantages

- Decision tree
  - Simple to understand which decisions are made when by the model
  - Require little data preparation
- K-nearest Neighbor
  - Simple to implement and understand
- Support vector machines
  - Not biased by outliers
  - Does not overfit easily
- Convolutional neural networks
  - Able to detect large number of features
  - Fast training and testing time
  - Easily improved over time without much human intervention

Disadvantages

- Decision tree
  - Oftentimes over-complex trees are created, resulting in high overfitting
  - Small variations in data cause new trees to form (high memory)
  - Biased trees are created if classes do not have similar amounts of data
  - Performs poorly with low amounts of data
- K-nearest Neighbor
  - Significantly slows as number of samples increases
  - Requires large amounts of training data
  - Require manual changing of k-value
- Support vector machines
  - Not useful for non-linear problems
  - Not ideal for large number of features
- Convolutional neural networks
  - Difficulty classifying images with variances in angles and scaling
  - Often overfits to data

Appendix C: *Functions Used*

Sparse categorical cross entropy was used to categorize loss values, using the formula

$$\boldsymbol{L}(w) = -\sum_{i=1}^{N} y_i \cdot \log(\hat{y}_i)$$

where N is the size of the output, y_i is the value of the actual class, $\hat{y}_i$ is the predicted class, and w is the current configuration of the weights in the model.

The ReLu function was used as the activation function for each of the convolutional layers and one of the dense layers, using the formula

$$\phi(x) = \max\,(0, x)$$

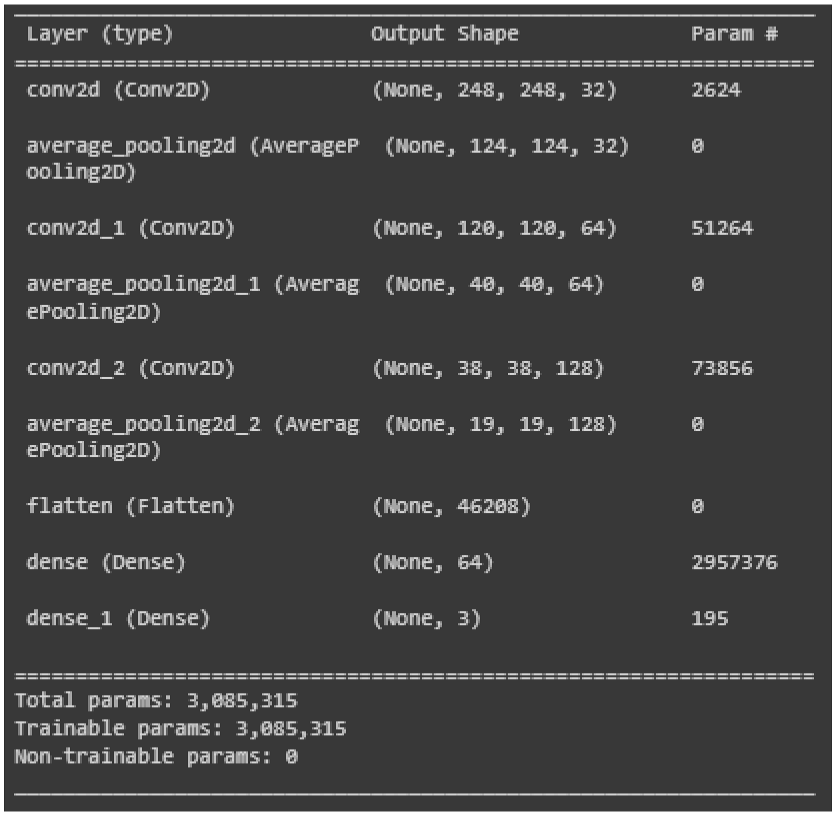
where x is the value of the input and ϕ is the weight that is applied onto the input.

The SoftMax function was used to condense the outputted 64 value vector from the first dense layer into 3 values, each showing the probability of its designated class. The SoftMax function is defined as

$$\sigma(\vec{z}) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

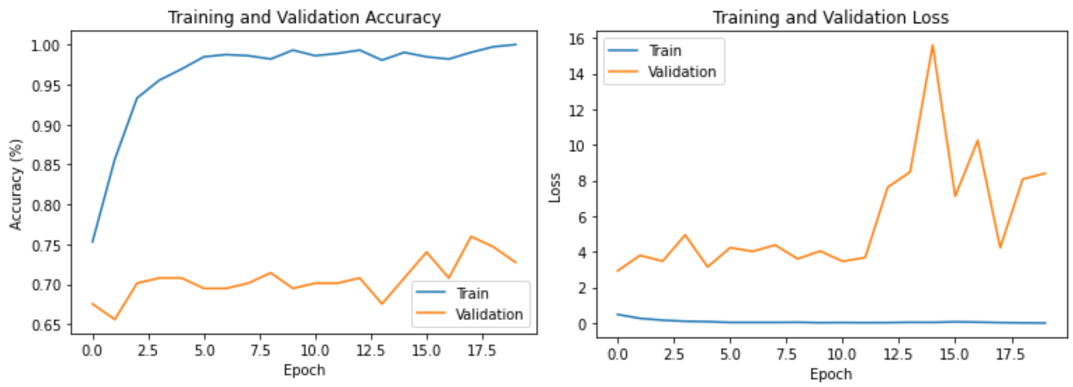
where $\vec{z}$ is the input vector and the outputted vector contains K values, each containing the probability of that the image belongs to a certain class.

Appendix D: ***Model Architecture and Training Data***

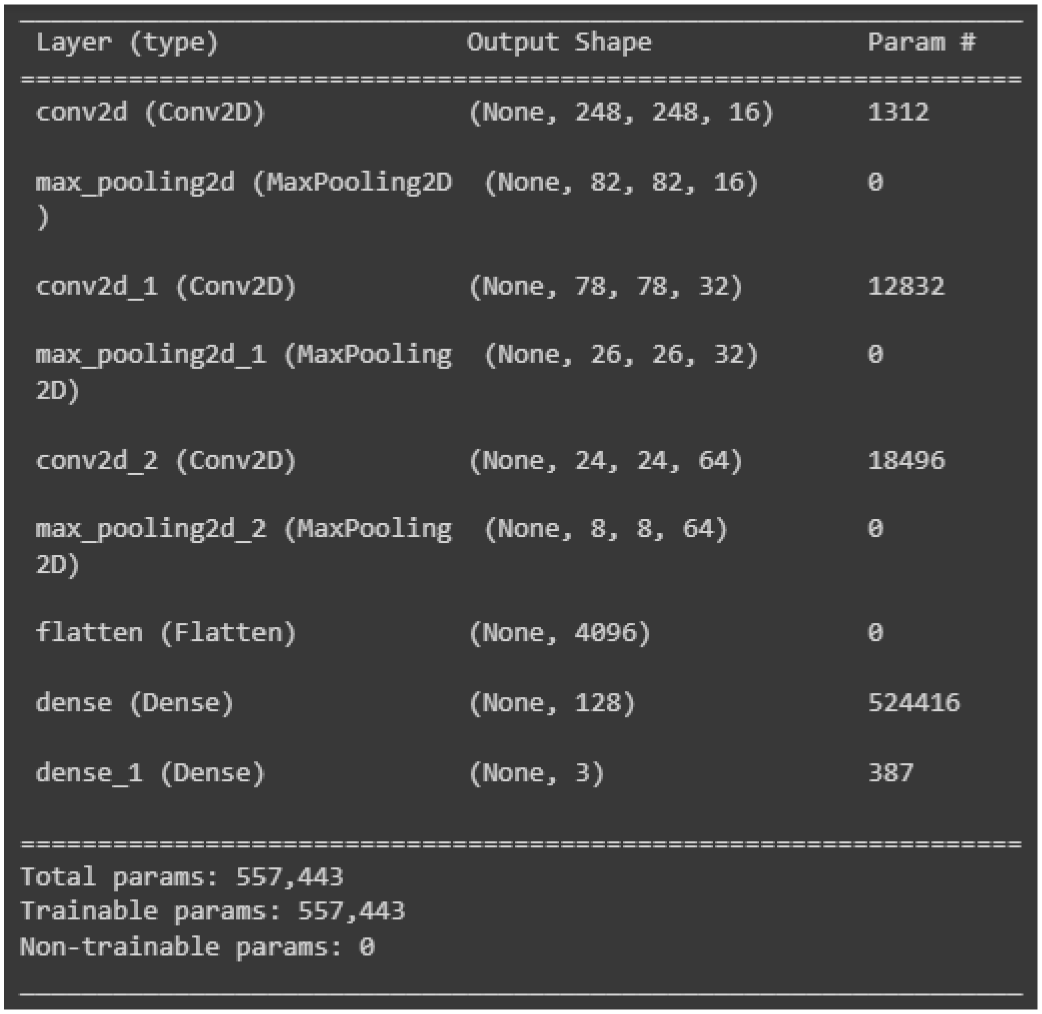


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 248, 248, 32)      2624

 average_pooling2d (AverageP  (None, 124, 124, 32)     0
 ooling2D)

 conv2d_1 (Conv2D)           (None, 120, 120, 64)      51264

 average_pooling2d_1 (Averag  (None, 40, 40, 64)       0
 ePooling2D)

 conv2d_2 (Conv2D)           (None, 38, 38, 128)       73856

 average_pooling2d_2 (Averag  (None, 19, 19, 128)      0
 ePooling2D)

 flatten (Flatten)           (None, 46208)             0

 dense (Dense)               (None, 64)                2957376

 dense_1 (Dense)             (None, 3)                 195

=================================================================
Total params: 3,085,315
Trainable params: 3,085,315
Non-trainable params: 0
_________________________________________________________________
```

The architecture of model 1, containing three convolutional layers and two dense layers. Average pooling was used in between each convolutional layer and the model had a total of 3,085,315 parameters (weights)

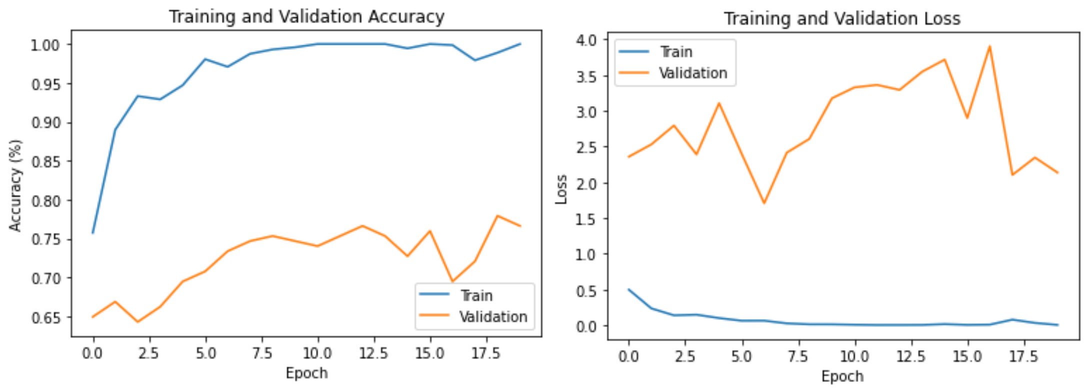


The values of training and validation loss across the 20-epoch training stage for model 1.

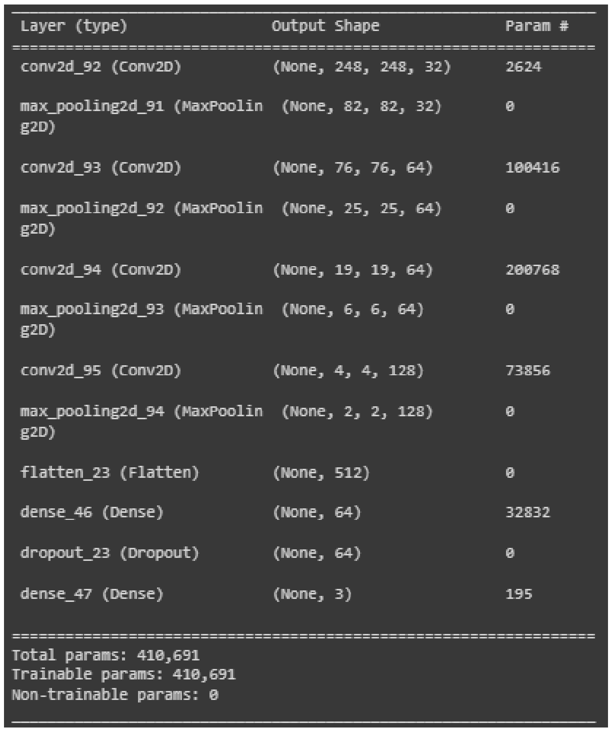


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 248, 248, 16)      1312

 max_pooling2d (MaxPooling2D  (None, 82, 82, 16)       0
 )

 conv2d_1 (Conv2D)           (None, 78, 78, 32)        12832

 max_pooling2d_1 (MaxPooling  (None, 26, 26, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 24, 24, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 8, 8, 64)         0
 2D)

 flatten (Flatten)           (None, 4096)              0

 dense (Dense)               (None, 128)               524416

 dense_1 (Dense)             (None, 3)                 387

=================================================================
Total params: 557,443
Trainable params: 557,443
Non-trainable params: 0
_________________________________________________________________
```

The architecture of model 2, with a very similar architecture to model 1. The main difference was the use of max pooling layers rather than average pooling layers. The model contained a total of 557,443 parameters

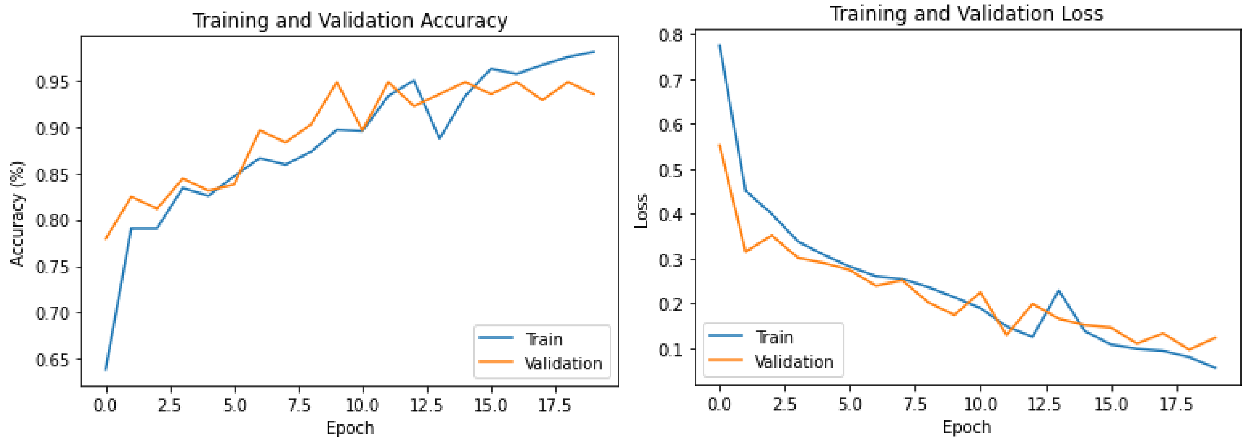


The training and validation accuracies and loss values across the 20-epoch training stage for model 2.



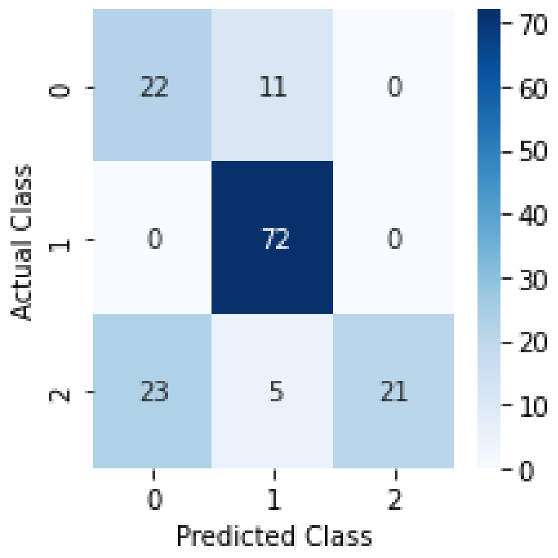
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_92 (Conv2D)          (None, 248, 248, 32)      2624      
                                                                 
 max_pooling2d_91 (MaxPoolin  (None, 82, 82, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_93 (Conv2D)          (None, 76, 76, 64)        100416    
                                                                 
 max_pooling2d_92 (MaxPoolin  (None, 25, 25, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_94 (Conv2D)          (None, 19, 19, 64)        200768    
                                                                 
 max_pooling2d_93 (MaxPoolin  (None, 6, 6, 64)         0         
 g2D)                                                            
                                                                 
 conv2d_95 (Conv2D)          (None, 4, 4, 128)         73856     
                                                                 
 max_pooling2d_94 (MaxPoolin  (None, 2, 2, 128)        0         
 g2D)                                                            
                                                                 
 flatten_23 (Flatten)        (None, 512)               0         
                                                                 
 dense_46 (Dense)            (None, 64)                32832     
                                                                 
 dropout_23 (Dropout)        (None, 64)                0         
                                                                 
 dense_47 (Dense)            (None, 3)                 195       
                                                                 
=================================================================
Total params: 410,691
Trainable params: 410,691
Non-trainable params: 0
_________________________________________________________________
```

The architecture of model 3 contained four convolutional layers as well as the addition of a dropout layer between the two dense layers. The model only contained 410,691 parameters.

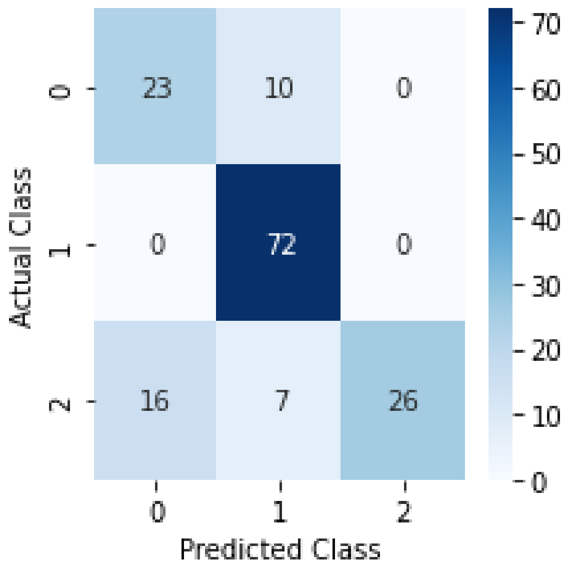


The training and validation values during the training of model 3 across 20 epochs.

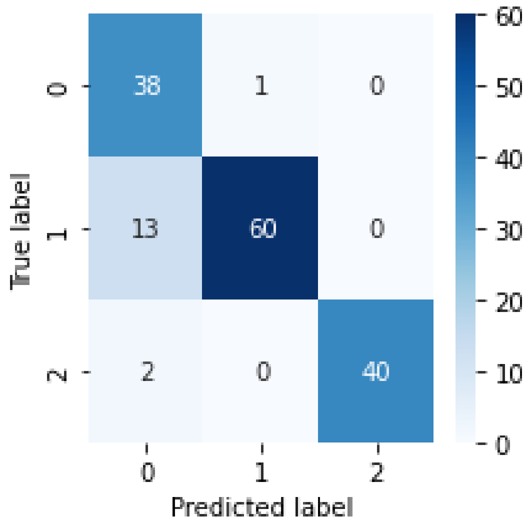
Appendix E: ***Confusion matrices for each of the models on the testing data***



The confusion matrix for model 1, which demonstrated low accuracy for ischemic strokes and high accuracy for hemorrhagic strokes



The confusion matrix for model 2, which demonstrated similar results to model 1 with slight improvements in accuracy and loss



The confusion matrix for the final model, which demonstrated the highest overall accuracy across the three classes and the lowest loss on the testing data