Convolutional Neural Networks for Tumor Detection

Veda Chari
Department of Computer Science
Boston University
Boston, MA 02215
vechari@bu.edu

1 Introduction

A brain tumor is a cancerous or noncancerous mass or growth of abnormal cells in the brain, with the symptoms varying based on the tumor's location. Brain imaging techniques, such as magnetic resonance imaging (MRI) and computed tomography (CT) scans are needed in order to diagnose a tumor and formulate a recover plan. However, the detection of these tumors rely heavily on the doctor or radiologist's experience and abilities. This detection step can be aided through an image classification neural network. The aim of this project is to train two convolutional neural networks, one to determine if a MRI scan contains a tumor or doesn't, and one that aims to determine what type of tumor it is. The yes or no tumor classification model achieved an accuracy of 97.67%, and the type of tumor detection achieved an accuracy of 91.10%. The dataset the model is trained on contains 3653 brain MRI scans, either containing a tumor or not. The executable code can be found at `https://github.com/vechari/Brain-MRI-Tumor-Classification.git`.

2 Related Work

Currently, checking for tumors is something that is performed in a doctor's office. If signs of symptoms of a tumor exist, your doctor will perform tests to check reflexes, muscle strength, vision, eye movements, etc. If the results are abnormal, you are referred to a neurologist or neurosurgeon for a more detailed exam. It is very common to have one or more image tests. Once the scan is taken, a expert is needed to classify the tumor, and decide the next steps. There are many types of tumors in the brain, but the three most frequent tumors are gliomas (38% of primary brain tumors), meningiomas (27% of primary brain tumors), and pituitary tumors. [1] Glioma tumors originate from glial cells, which are supportive cells in the brain [5]. Meningioma tumors are from meninges, which are protective layers surrounding the brain and spinal cords [12]. Pituitary tumors are found in the pituitary gland, below the hypothalamus [7].

There are many popular image classification techniques such as SVM and CNN. SVM, or support vector machine, is a supervised ML algorithm for classification and regression. It is based on the statistical learning theory. It finds the correct hyperplane for classification through the help of support vectors. SVM is a linear classifier predicting each input's member class between two possible classifications. A CNN, or convolutional neural network, is a feedforward neural network that uses the convolution calculation of matrices and has a deep structure. A study conducted compared the two techniques on a large sample mnist dataset, and reached a 88% accuracy using SVM, and a 98% accuracy using CNN.[11] A CNN is also more adaptive to many types of data, and there are very good python packages such and Tensorflow and PyTorch.

3 Resources

The dataset contains 4 classes, glioma tumors, meningoma tumors, pituitary tumors, or no tumor. For training the network that only determines if there is a tumor or not, the glioma, meningioma,

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

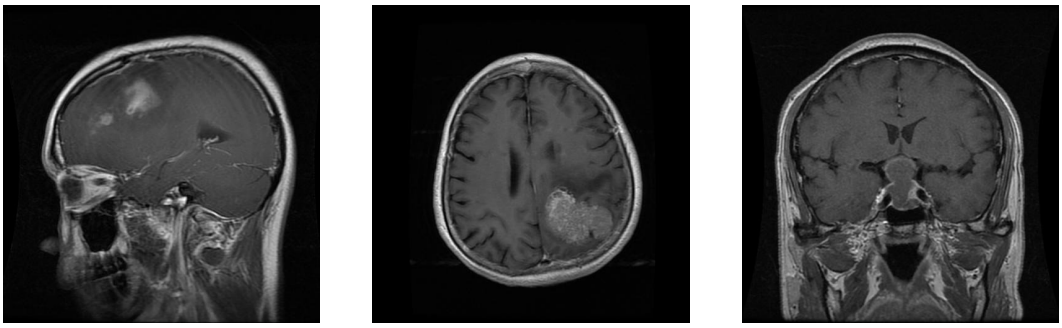
and pituitary tumor files were combined into a yes category. However, for both models, the number of no tumor images was inadequate for proper learning, having close to 400 images less than the other tumor types. In order to rectify this difference, I added images not containing a tumor from a different MRI scan dataset from Kaggle.
The main dataset can be found here: `https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri`. The added images can be found here:`https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection/data`, but the exact datasets used for training both models can be found at the github link.

The code was run on an Apple Macbook Air with the Apple M1 chip with 8GB of memory. The apple M1 chip contains an 8-core CPU with 4 performance cores and 4 efficiency cores, a 7-core GPU, and a 16-core Neural Engine. The model was built using Tensorflow's Keras API. To improve runtime, Tensorflow has a Metal pluggin by Apple which gives direct access to the 7 core GPU. It doesn't give access to the neural engines because those are optimized for inference not training. The GPU cores are better suited due to memory bandwidth issues. [14]

4 Method

4.1 Brain Imaging

The images used for this are all Brain MRI cross sections. An MRI scan can be a cross section in 3 directions, horizontal (parallel to the ground), coronal (parallel to the face), and sagittal (parallel to the side of the head).



(a) Sagittal Cross Section (b) Horizontal Cross Section (c) Coronal Cross Section

Figure 1: Cross section cut examples from dataset

When looking at the brains in Figure 1, there are some key anatomical features to look for. First, the cerebral cortex, or outermost layer of the brain, is what outlines the brain. Secondly, there are ventricles in the center of the brain. These are cerebral fluid filled cavities. Figure 1 also shows the 3 types of tumors. Image 1a shows the saggital view of a glioma tumor, 1b shows the horizontal view of a meningioma tumor, and 1c shows the coronal view of a pituitary tumor.

4.2 Neuroscience Background for Neural Networks

Neural Networks aim to simulate the connections and flow of information in the brain. In brain networks, each neuron can receive inputs from numerous other neurons. The following equation can be using to map the input to a specific neuron

$$net_i(t) = \sum_{j=1}^{n} w_{ij}(t) o_i(t)$$

where to $net_i(t)$ represents the net input signal to the i^{th} unit in the network, $o_j(t)$ is the output from the j^{th} neuron from the previous layer, with the associated weight $w_{ij}(t)$ for that synaptic connection. [9] The output of an unit $o_i(t)$ is determined by some activation function based on the input value $net_i(t)$. Through the functions that simulate the connections in the brain, we can

simulate brain function and use computer models to act in a similar fashion. There are many famous models for synaptic connections, such as the Hodgkin Huxley, FitzHugh–Nagumo, Izhikevich, and Hindmarsh–Rose neuron models. [6] We can extrapolate these ideas of synaptic connections and weights to build a deep neural network.

4.3 Deep Neural Network

When referring to Deep Neural Networks, it usually refers to an Artificial Neural Network (ANN). [3]. Artificial Neural Networks are biologically inspired, aiming to simulate the way the human brain processes information through synaptic connections between neurons. ANNs detect patterns and relationships in the data to learn, or train, from the experience, through the use of hundreds of artificial neurons and connection weights. [2]. The neurons are structured in layers, where the weight determines the contribution that neuron has to the output. For image classification, a Convolutional Neural Network is best equipped for the task. It uses a series of layers to detect important features in the image, and trains based on those findings. A CNN has excellent performance of machine learning tasks such as image classification on a large dataset, computer vision, and in natural language processing. [3]

4.3.1 Convolutional Neural Network

A Convolution Neural Network (CNN) is best adaptable for classification and computer vision tasks. Using principles of linear algebra, such as matrix multiplications, a CNN model can identify patterns within an image. The neural network consists of many node layers. There is an input layer, one or more hidden layers, and an output layer. Each node in a layer connects to another with an associated weight and threshold. If the output of a node is above threshold, then that node is activated and sends data to the next layer. If the output is not above threshold, then no data is sent.

There are three main types of layers in a CNN model: convolutional layer, pooling layer, and full-connected (FC) layer. The convolutional layer is the first layer, followed by either additional convolutional layers or pooling layers, and finally goes to the fully-connected layer. The CNN increases in complexity with each layer, and identifies greater areas of the image. Earlier layers focus on color and line detection, and later layers focus on recognizing larger elements such as shapes and patterns.

4.3.2 Convolutional Layer

The convolutional layer does the majority of the computation on the image. The input data image will have 3 dimensions: a height, a width, and a depth which corresponds to that pixel's RGB values. We then use a filter, or feature detector, to move across the receptive field and check if the feature is present. The filter is usually a 2D array with weights, typically a 3x3 matrix. The size of the matrix determines the size of the receptive field. As the filter is applied to an area of the image, the dot product is calculated between the filter and the pixel values. This dot product is then loaded into a feature map, which is the output of the layer. Once the value is loaded, the filter is shifted by a stride and repeats the process.

There are many hyperparameters that can be set to affect the volume size of the output layer, such as number of filters, stride, and zero-padding. The number of filters affects the depth of the feature map. So, if you have 3 filters, you would create an output layer of depth 3 containing 3 distinct feature maps. The stride is the number of pixels the filter moves over the input matrix, and a larger stride creates a smaller input. Finally, zero-padding is used when the filter doesn't fit the input image, and sets all elements outside the input matrix to zero, producing either an equally sized or larger output matrix.

Once the convolutional layer is done, the image has been converted into numerical numbers. The convolutional layer can be followed by additional convolutional layers, or pooling layers.

4.3.3 Pooling Layer

The pooling layer is used for dimensionality reduction, or reducing the number of parameters in the input. The pooling layer also has a filter, but the filter contains no weights. The filter sweeps across

the entire input and applies an aggregation function to values within the receptive field to populate an output array.

There are two main types of pooling: max pooling and average pooling. Max pooling is when the filter moves over the receptive field and selects the pixels with the maximum value to send to the output array. Average pooling is when the filter calculated the average over the receptive field and populated the output array with the average. Once the pooling layer is done, it can be followed by additional pooling layers or the fully- connected layer.

4.3.4 Fully-Connected (FC) Layer

The fully connected layer performs classification based on the features the filters of previous layers extracted. Each node in the output layer is directly connected to a node in the previous layer. The layer uses the softmax activation function to classify inputs and produce a probability from 0 to 1.

4.3.5 Training

Training a convolutional neural network is highly dependent on its hyperparameters. Hyperparameters include the size of the convolutional kernels, number of kernels, the length of the strides, and the pooling size.[13] However, hyperparameter optimization is not easy to do, especially when there are a large number of hyperparameters. Studies show that a large number of hyperparameters and layer-specific hyperparameters are important for better performance of the model. [4]

The training is done based on the set parameters of the number of epochs and the batch size. Batch size is the number of samples processed before the model is updated. The number of epochs is the number of complete passes through the dataset the model performs.

During training, the data can get overfit if there isn't enough variability in the data. This means the model is memorizing the inputs instead of learning from them. There are two ways to prevent overfitting, data augmentation and a dropout layer. Data agumentation does random transformations on the data, such as horizontal flip, rotate, and zoom. A dropout layer sets certain nodes to the value zero during training, effectively leaving them out of the network. These nodes will have no influence over the prediction and backpropogation, creating a slightly modified network architecture on each run. The network is trying to learn from parts of the data on each run, and learns to make good predictions without certain inputs. [10]

4.4 Tensorflow

This project utilized the Tensorflow Python library to create a machine learning model. Tensorflow streamlines model construction, training, and export. Tensorflow has a high-level API called Keras which provides an easy to use interface for machine learning problems using modern deep learning.

The core data structures of Keras are layers, which are the simple input and output transformations, and a module, which is a directed acyclic graph of layers. Modules are groups of layers that can be trained on data. The model created is a Sequential model, which is a linear stack of layers. In the Sequential model, the fully connected layer, called Dense, expects a flattened input, so the Flatten layer turns the 2D or 3D input into a 1D vector.

The Keras Module class has many built in features, such as fit, which trains the model for a set number of epochs, predict, which creates an output prediction for an input, and evaluate, which gives the loss and other metrics values of the model.

4.5 Grad-CAM

Grad-CAM, or gradient-weighted class activation mapping, uses gradient information to localize important regions. First, there is a gradient calculation based on the feature maps of the last convolutional layer in the neural network, where the most features will be extracted due to the higher filter amount. The gradients represent how much each feature map contributes to the prediction of the target class. Once the gradients are computed, they are used to compute the importance weights for each feature maps. The gradients are then aggregated across spatial dimensions to create a weight for each feature map. Finally, the weighted combination of the feature maps is created to produce a heatmap, where the higher values indicate regions that are more relevant during learning and

prediction. A Grad-CAM can be used to better understand CNN based models, and visualize what features the model considers important. [8]

5 Experimental Results

Two models of different depths wewre trained on the dataset. The first model determines if there is a tumor in the supplied MRI scan. The second model aims to determine what type of tumor there is.

5.1 Tumor Detection

This model determines if there is a tumor in the MRI scan. The dataset was split into 2 folders: yes and no. The yes folder contains the MRI scans containing meningioma, glioma, and pituitary tumors. The photos per class is split in the following way

Table 1: Number of Photos per Class

File Name	Number of Photos
no tumor	890
yes tumor	2764

The sequential model used to determine if the MRI scan contains a tumor or doesn't is set up with 5 convolutional layers with max pooling layers in between. The number of filters doubles on each convolutional layer in order to extract a hierarchical order of features. At the higher filter numbers, a dropout layer was introduced to prevent overfitting, and the convolutional layers were doubled. There is also a dropout layer before the fully connected layer.

The batch size was set to 32 images. No data augmentation was needed due to how robust the dataset is, and because there are only 2 classes. Due to the shallowness of network, the model was able to run for 50 epochs without runtime issues.

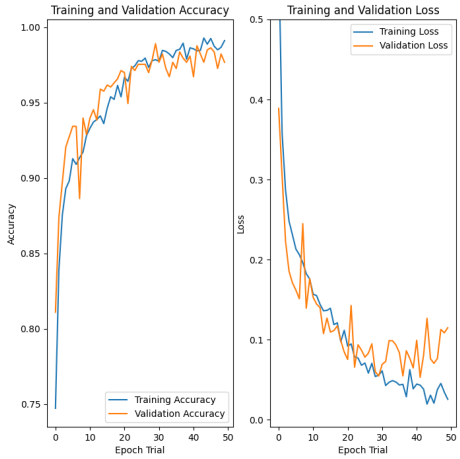
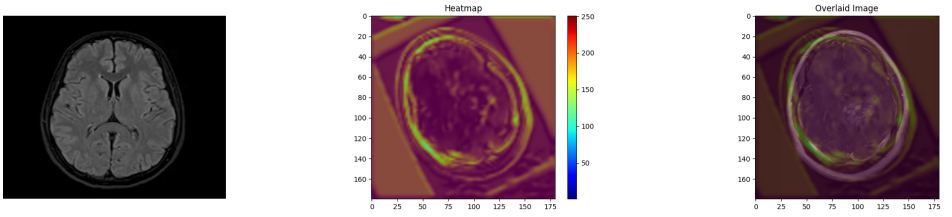


Figure 2: The training and validation accuracy and loss. The model reached a final validation accuracy of 97.67% and a validation loss of 0.1148

The model reached an accuracy of 97.67%, as seen in Figure 2. However, the loss function determines if the model performs well. The trends for the training loss and the validation loss are very similar, showing that the data is not over fit. If the data was overfit, the loss of the training set would be much higher than the testing, because the model is memorizing instead of learning. The model was tested on 4 images, 3 containing a tumor and 1 without. The Grad-CAM implementation was used to compute a heatmap based on the final convolutional layer's outputs, as shown in Figure 3.



(a) No Tumor (b) No Tumor Heatmap (c) No Tumor Overlay

(d) Tumor 1 (e) Tumor 1 Heatmap (f) Tumor 1 Overlay

Figure 3: Grad-CAM heatmaps for yes or no tumor detection.

The model predicted that there wasn't a tumor with 99.53% confidence, and that there was a tumor with 100% confidence. The Grad-CAM computes the gradients of the score of the target class with respect to the feature maps of the last convolutional layer. The gradients are used to compute the importance weights for each feature in the map, which is then turned into a heatmap. The areas of higher values indicate regions of the image with more relevant features. From the visualization, we can see that the tumor is highlighted in yellow. For the no tumor, there is highlighting around the cerebral cortex and slight coloration around the ventricles. The model seems to extract features that outline the brain, and features that are different from inside, which includes ventricle size. The heatmap is computed based on the model created, which has a data augmentation layer. Thus, the heatmap might be a little skewed compare to the input image when it is overlayed.

5.2 Type of Tumor Detection

This model determines if there is a tumor, and what type of tumor, in the MRI scan.

Table 2: Number of Photos per Class

File Name	Number of Photos
no tumor	890
meningioma tumor	938
glioma tumor	927
pituitary tumor	902

The dataset has 4 classes: no tumor, meningioma tumor, glioma tumor, and pituitary tumor. The photos in each folder is shown in Table 2.The validation split was set to 0.2, meaning 80% of the images were for training, and 20% for validation. This gave 3653 files in total, 2923 files for training and 730 files for validation.

The model was set up with 6 convolutional layers in groups of 2, which a max pooling and dropout layer in between them. The number of filters increased on each level to form a hierarchical order of feature detection. A data agumentation layer was introduced in the beginning to prevent overfitting. The data augmentation does two random transformations, a rotation of up 10 degrees, or a zoom of up to 10%. The batch size was set to 32 images per batch. Due to the depth of the neural network, it was only able to run for 25 epochs without encountering major runtime setbacks.

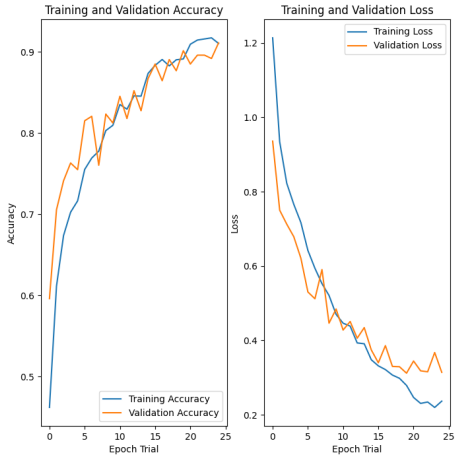


Figure 4: The training and validation accuracy and loss. The model reached a final validation accuracy of 91.10% and a validation loss of 0.314

As with the model that detects if there is a tumor, the trends for the training and validation accuracy and loss follow similar patterns, as seen in Figure 4. Based on the graphs, the model doesn't overfit the data, as training and validation are very similar. The loss on the model is relatively high with a validation loss of 0.31. However, the validation loss is less than 0.05 away from the training loss. The Grad-CAM can be used to determine what is affecting the loss.

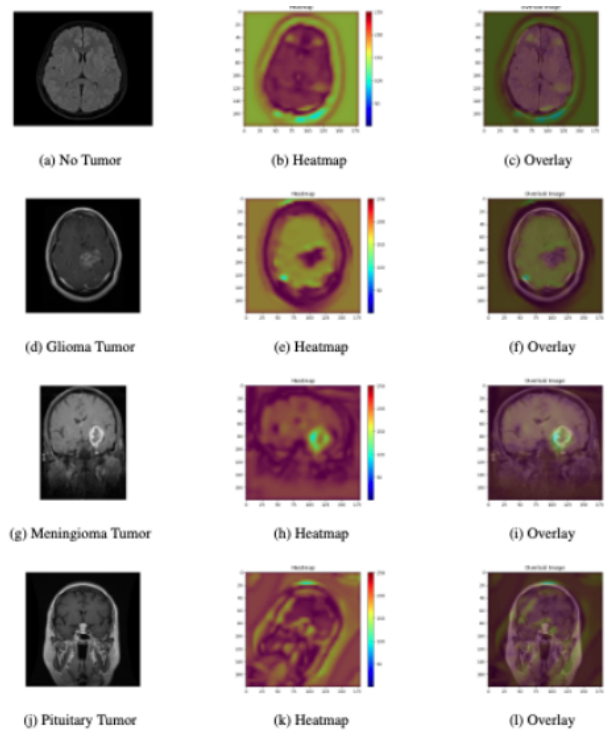


Figure 5: Heatmaps and overlayed images for no tumor, glioma, tumor, meningioma tumor, and pituitary tumor.

When predicting on the no tumor image, the model gave a false prediction that the scan contained a "pituitary tumor with 99.97% confidence." When observing the no tumor Grad-CAM in Figure 5, it highlights area around the ventricles, which is usually where a pituitary tumor can be seen on a horizontal MRI scan. Even with data augmentation, there was no way to fix this error. However, it detected the glioma tumor with 99.91% confidence, the meningioma tumor with 83.55% confidence, and the pituitary tumor with 99.95% confidence. The only class the model had a hard time predicting was the no tumor class, due to the location of the pituitary tumor.

6 Conclusion

The main challenge faced during this project was that the models kept overfitting the data, especially the model that determined what type of tumor the scan contained. I was able to get around these issues by adding more convolutional layer, due to the complexity of the features being extracted, adding drop out layers, and using data augmentation. However, I ran into a problem while computing and overlaying the heatmap due to the augmentation. When the image is passed for prediction, it is randomly augmented in the model. However, I had no way to retrieve that augmented image to overlay the heatmap with that, so the augmented heatmap is overlayed with the original image. Originally I had a random horizontal flip layer during data augmentation, however that skewed the prediction heatmap so much that the Grad-CAM gave no useful information. For example, if the tumor was on the right hemisphere, it would shwo in the left hemisphere of the heatmap. I also realized that the location of the tumor greatly effects the type of tumor it is, and thus a horizontal flip is not a good data augmentation step to perform. Now, the heatmap doesn't fully align, but the highlighted areas are in the general area of the tumor, and a relatively good marker of the area. These models are good at predicting if a MRI scan contains a tumor or not, or if it is a glioma or meningioma tumor. More training needs to be done to correctly differentiate a pituitary tumor from normal ventricles. Or these models can be used in tandem, with the first model determining if there is a tumor, and then the second model only determining what type of tumor it is by removing the no class.

References

[1] Adult central nervous system tumors treatment. National Cancer Institute Physician Data Query (PDQ). Accessed: April 29, 2024.

[2] S Agatonovic-Kustrin and R Beresford. Basic concepts of artificial neural network (ann) modeling and its application in pharmaceutical research. *Journal of Pharmaceutical and Biomedical Analysis*, 22(5):717–727, 2000.

[3] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[4] Tirana Noor Fatyanosa and Masayoshi Aritsugi. Effects of the number of hyperparameters on the performance of ga-cnn. In *2020 IEEE/ACM International Conference on Big Data Computing, Applications and Technologies (BDCAT)*, pages 144–153, 2020.

[5] Eric C Holland. Progenitor cells and glioma formation. In *Current Opinion in Neurology*, volume 14, pages 683–688, 2001.

[6] NİMET KORKMAZ, İSMAİL ÖZTÜRK, and RECAİ KILIÇ. Multiple perspectives on the hardware implementations of biological neuron models and programmable design aspects. In *Turkish Journal of Electrical Engineering and Computer Sciences*, volume 24, pages 1729–1746, 2016.

[7] S Melmed. Pathogenesis of pituitary tumors. In *Nat Rev Endocrinol*, volume 7, pages 257–266, 2011.

[8] Ramprasaath R Selvaraju, Abhishek Das, Ramakrishna Vedantam, Michael Cogswell, Devi Parikh, and Dhruv Batra. Grad-cam: Why did you say that?, 2017.

[9] David M. Skapura. *Building Neural Networks*. ACM Press, 1996.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[11] Pin Wang, En Fan, and Peng Wang. Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, 141:61–67, 2021.

[12] J Wiemels, M Wrensch, and E.B. Clause. Epidemiology and etiology of meningioma. In *J Neurooncol*, volume 99, pages 307–314, 2010.

[13] Lu Y, Huo Y, Yang Z, Niu Y, Zhao M, Bosiakov S, and Li L. Influence of the parameters of the convolutional neural network model in predicting the effective compressive modulus of porous structure. In *Front. Bioeng. Biotechnol.*, volume 10, 2022.

[14] Christian Zibreg. What is apple's neural engine and how does it work? Make Use Of, February 2023. Accessed: April 28, 2024.