CS63 Spring 2022
Convolutional Neural Network for MRI Brain Tumor Detection

Brooke Coneeny and Sydney Levy

May 5th 2022

1 Introduction

In this project we are going to utilize a convolutional neural network (CNN) to classify brain tumors within MRI scans. The goal of the deep neural networks is to generalize well to novel examples, thus we do not want the network to memorize the MRI scans in the training set. Instead, we want it to learn key features of the training data which will generalize to new brain scans. We chose to apply a deep neural network to brain MRI images as brain tumors can be one of the most aggressive diseases. Thus, early detection is of the utmost importance. Additionally, in recent years, deep networks using convolutional layers have achieved high accuracy on classifying tumors within medical data and thus we thought that this would be a strong approach to use for detecting tumors within brain MRI scans. Brain tumors can occur in different areas of the brain and thus convolutional layer's ability to detect features anywhere in the images will be very important for this task. Further, tumors can come in different shapes and sizes, thus automation of this task can help aid human doctors in order to detect all kinds of tumors with higher accuracy.

To begin, we will provide an overview of the benefits of CNNs which will allow us to complete this task. These networks consist of many layers. The input layer receives the data as an input and sends output signals through hidden layers. After the hidden layers have processed the data, they send the final results to an output layer. CNNs are based on human's biological neural networks. The nodes in the network get inputs from many nodes and give outputs to many nodes. The weights between nodes represent the strength of the connection. Each node in the networks fires based on an activation function. If the weighted inputs to this activation function are above the threshold, it sends an output signal. A bias node connects to each node in the graph, except for nodes in the input layer, in order to make the activation threshold learnable.

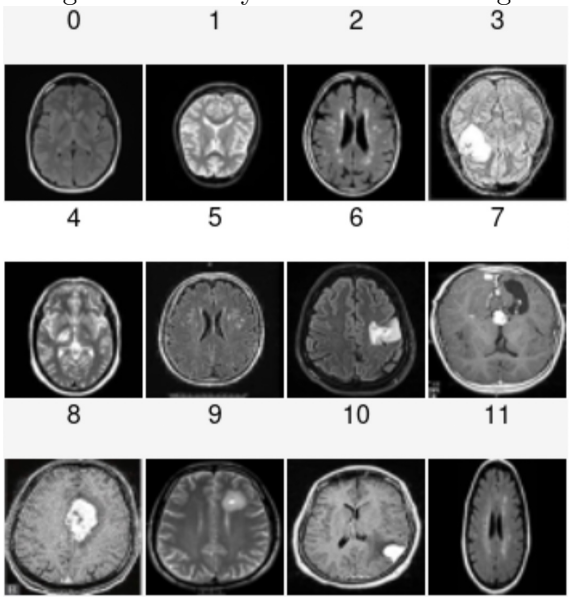
The network forward propagates data through the network layers until it determines an output. During forward propagation, the output of a node is determined by the activation function. We are using the ReLU (rectified linear unit) activation function to avoid the vanishing gradient problem. The ReLU activation function is $f(x) = max(0, x)$ and thus the derivative is always equal to 0 or 1, which avoids the vanishing gradient problem. This problem occurs when the partial derivatives of the activation function become so small that backpropagation no longer modifies the weights of the network. Thus, backpropagation plateaus before finding an optimal set of weights which would not be ideal. Within backpropagation, the loss function represents how to compute the error and the optimizer determines how to modify the weights of a network.

When using image data, flattening the data into one dimensional inputs can loose key spatial relationships within an image. Thus, for our project we use a two dimensional convolutional network to investigate the relationships within the images of the brain. Convolutional layers use feature detectors / kernels which have a learnable weight in order to identify key features within the sub region of the image they are looking at. These features could be something like a vertical or horizontal line in the image or in our case the outline of a tumor. One of the parameters of a convolutional layer is the window size which represents how large of a sub-region the feature detector is examining. The stride length is the number of pixels to slide the window. There are shared weights within sub-regions so that it can recognize the feature anywhere in the image. A two dimensional convolutional layer can learn different features at once by having multiple channels within the layer. Thus, the goal of our project is to effectively tune the hyperparameters of the CNN to detect brain tumors in novel MRI scans with the highest accuracy possible.

2 Methods

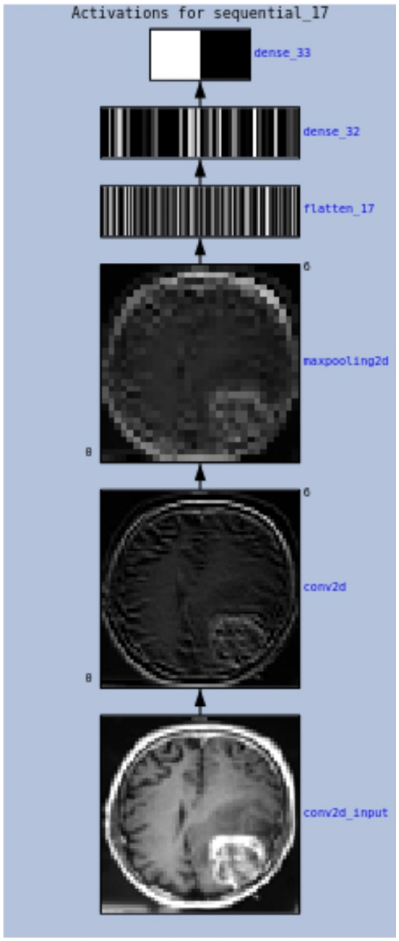
The data set we used for this project was found on Kaggle at the following url: `https://www.kaggle.com/datasets/ahmedhamada0/brain-tumor-detection`. It includes 3,000 brain MRI images. 1,500 of these images include a brain tumor and 1,500 have no tumor. We used the os.listdir() method in order to save the directory path for the different folders which contained MRI scans with and without tumors. We then went through all of the images in the no tumor folder and used cv2.imread() to load in each image array. After, we used the Image.fromarray() method to convert the arrays into images. From there, we resized each of the images to be (64,64). Finally, we added the images to our data list, and added a value of 0 to our label list to represent the images having no brain tumor. We followed a parallel process for the folder which contained MRI scans of brains with tumors. Figure 1 shows a sample of Brain MRI scans loaded from the dataset.

Figure 1: Gallery of Brain MRI Images



Once we had our list of images and labels, we used the scikit-learn function train_test_split() to split our data into 80% training and 20% testing data sets. Therefore, we ended up with 2,400 image-label pairs in the training set and 600 in the testing set. This function also shuffled the data so that we had a random assortment of tumor and non-tumor images in our sets.

We then trained a convolutional neural network based on the following architecture: a 2-dimensional convolutional layer, a max pooling layer, a flattening layer, and 2 dense layers. The convolutional layer includes 6 features and a $(3, 3)$ window size. We used the ReLu activation function and an input shape of $(64, 64, 3)$. The max pooling layer has a pool size of $(2, 2)$ followed by a flattening layer. The first dense layer has 64 units, and the second has 2 units, representing our two outputs of tumor or no tumor. A sample display of the different layers in our network architecture is shown in the image to the right.



When we compiled the model we used the adam optimizer function and sparse_categorical_crossentropy loss function. This loss function is the same as categorical_crossentropy with integer outputs.

We trained the model on our training set with 10 training epochs. The goal of our training was to achieve the highest accuracy as possible on the testing dataset. Thus, we tried to avoid over-fitting the model to the training set by only training the model for enough epochs to where accuracy began to plateau. We chose to train the model for 10 epochs because with any additional training epochs we found a lower testing accuracy as the model was over-fitting to the training set. Additionally, with fewer epochs we also achieved lower accuracy because the model did not have enough time to recognize important features in the input data.

While tuning the hyper parameters for our network we began by experimenting with different number of features. We found that because the dataset is fairly straightforward and the tumors are easy to detect in the images, we did not need a large number of features to achieve high accuracy.

Instead of focusing on a higher number of features in the convolutional layer, we found that the window size had a significant impact on the network's accuracy. It turned out that window sizes larger than (5,5) resulted in poor accuracy because the tumors are often quite small relative to the (64,64) image size. When examining the output of our channels with larger window sizes we found that they were not picking up on the tumors in the images, but instead on other spurious correlations such as the shape outline of the brain. Thus, we found that when we used a window

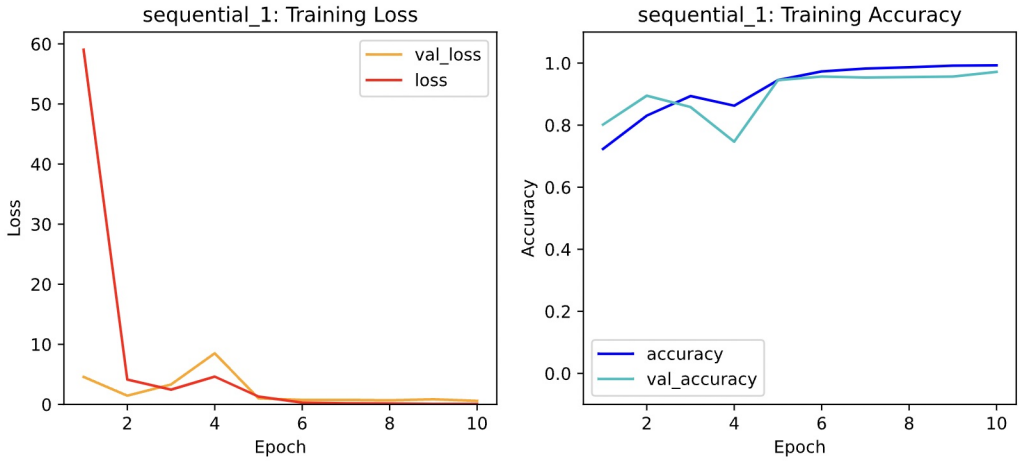
size of (3,3) we achieved the highest accuracy and our channels actually picked up on the brain tumors.

Finally, we tested our model with only one dense layer versus two dense layers. We found that the model needed both dense layers in order to extract all of the important features from the convolutional and max-pooling layers. Also, we landed on 64 as the size of our first dense layer as this allowed for more information to be passed from the flattening layer to the output layer and thus resulted in higher accuracy of the model (when compared to dense layers with fewer units).

3 Results

After 10 training epochs, our convolutional network achieved 97.2% accuracy at classifying whether the brain MRI scans contained a tumor or not (as shown in Figure 2). As can be seen in the graph below, after 2 training epochs, the loss of our network dropped steeply. However, after 5 training epochs the loss seems to plateu. We believe this occurs because the tumors are fairly easy for the network to detect and thus the network can learn relatively quickly. The accuracy of the network begins at roughly 70% for the training set and gradually climbs over the 10 epochs. The accuracy for the testing set also grows steadily over the 10 epochs and starts to pleateu after 6 epochs.

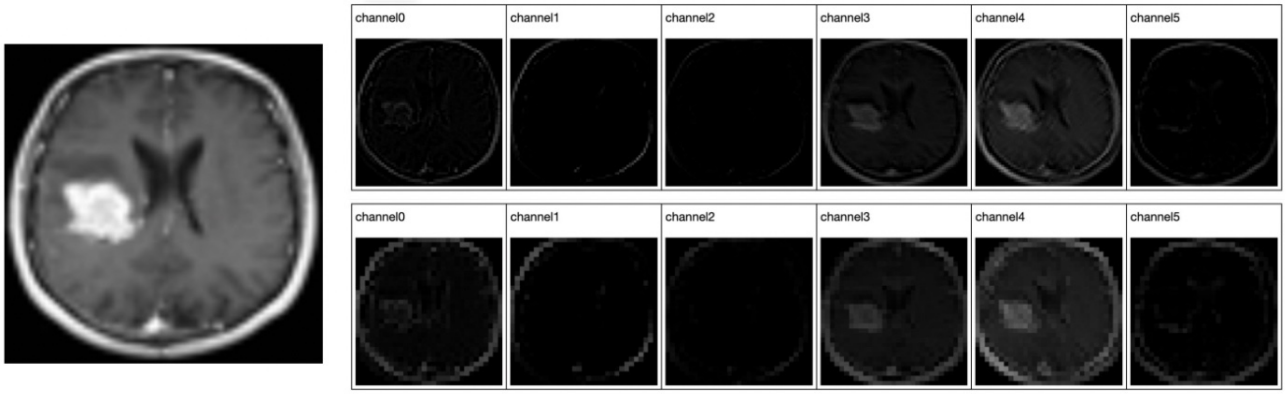
Figure 2: Loss and Accuracy of Network over 10 Epochs




```
Epoch 10/10 loss: 0.0690028965473175 - accuracy: 0.9929166436195374 - val_loss: 0.5924692749977112 - val_accuracy: 0.971666693
687439
```

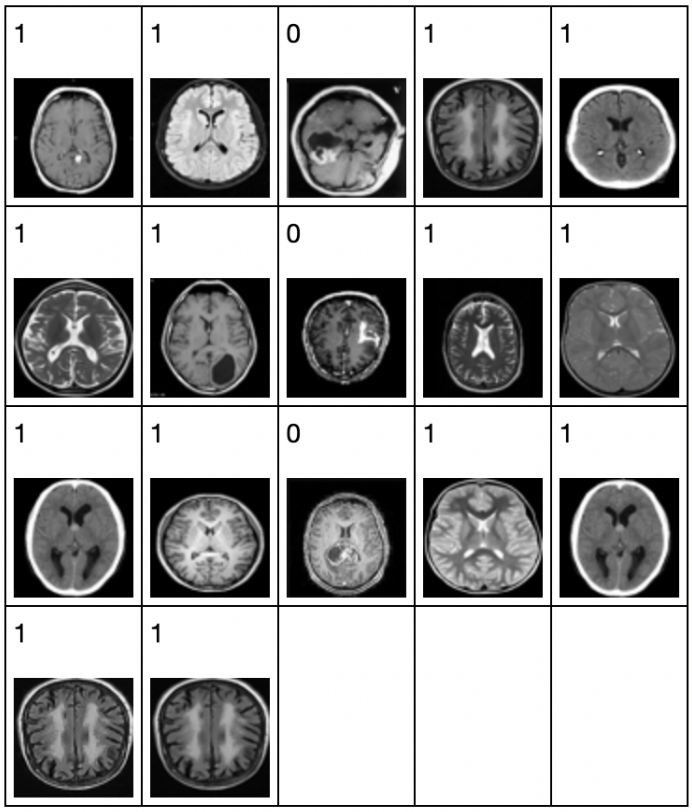
When examining the channels of our convolutional layer, it is clear that channel3 and channel4 are picking up on the presence of tumors in the MRI scans, as seen in Figure 3. Below we show an image of a brain with a large tumor in the left hemisphere which is also present in the accompanying convolutional (top) and max-pooling (bottom) channels. Both channel3 and channel4 show brighter spots where the tumor is located. Thus, we believe that our convolutional neural network is picking up on the presence of tumors in the brain MRIs as opposed to any other spurious correlations in the data.

Figure 3: Example of Convolutional and Max Pooling Channels for MRI with a Tumor



To better understand the images that our network is misclassifying we have displayed in Figure 4 all of the scans that the network incorrectly predicted. The number above the images represents the incorrect output produced the network. Of the images incorrectly predicted by our network, 14/17 are scans that do not contain a tumor but the network predicted that they do have tumors. These images are shown in Figure 4 as they have a classifier output of 1 but no tumors are located within the image. These images all seem to have brighter sub-regions within the brain that could present as similar to a tumor to the network.

Figure 4: Images Misclassified by Network



It is important to analyze the type of error the network is making as this will be important for understanding the medical application of our network (as shown in Figure 5). The majority of the errors our network made are Type 1 errors (or false positives). Type 2 errors occur when the brain has a tumor but the network outputs no tumor as can be seen by the images in Figure 4 that the network outputted 0 but the image contains a tumor. The type of error most often made by the network is especially important within a medical setting because it is important for doctors to avoid misinforming a patient. Currently, this model runs the risk of telling patients that they have a brain tumor when they do not. Obviously, this poses major ethical concerns. On the flip side, we also would not want a network that repeatedly makes Type 2 errors and fails to detect tumors in patients that do have one.

Figure 5: Types of Error by Presence of Tumor and Classifier Output

Correct: Brain has a tumor & Network outputs tumor	**Type 1 Error (False Positive):** Brain does not have a tumor & Network outputs tumor
Type 2 error (False Negative): Brain has a tumor & Network outputs no tumor	**Correct:** Brain does not have a tumor & Network outputs no tumor

4 Conclusions

Our project accurately trains a convolutional neural network to identify tumors in brain MRI scans. Our network included a 2D convolution layer with 6 features, a (3,3) window size, and ReLU activation function. The max pooling layer has a pool size of (2,2) and is followed by a flattening layer. Lastly, we had to dense layers, the first containing 64 units and the last having 2 units.

We compiled this model using the Adam optimizer function and the sparse_categorical_crossentropy loss function. The model was then trained for 10 epochs. In the end our network achieved 97.2% accuracy at classifying whether novel brain MRI scans contained a tumor or not. Additionally, the channels within the convolutional layer and max pooling layer successfully detected the tumors within the scans as opposed to making predictions based off of spurious correlations within the data images.

As for the images which were misclassified by the network, the majority represented Type 1 errors (false positives). We believe this was caused mostly due to the fact that these scans had more highlighted sub-regions within the brain which confused the network. Going forward we would like to try and train this network to not only detect the presence of tumors but to also report which type of tumor has been found.

We believe our project could represent an important step for utilizing CNNs to aid doctors in identifying tumors within brain MRI scans.