Jordan Bruce

Brain Tumor Classification

Introduction

Convolutional Neural Networks take in a set of images and train around different classes to distinguish them. To do this, it slides a filter across the pixels in an image, looking for certain patterns and features within it that may relate to other images of the same class. The features can be used to recognize differences between classes in order to correctly distinguish between them.

For this project, I created a model to look at MRI scans of brains and classify whether a brain has a tumor and what kind it is. The types of tumors were glioma, meningioma, and pituitary.

Analysis

I imported this dataset from Kaggle, titled: “Brain Tumor MRI Dataset”. It includes 5712 images in the training set and 1311 in the testing set. These set were fairly evenly distributed between the four classes (glioma, meningioma, and pituitary, and no tumor) with each class having between 1300 to 1600 for the training set and 300 to 400 in the testing set.

A picture containing screenshot, text, plot, number

Description automatically generatedA picture containing screenshot, text, plot, number

Description automatically generated

The images themselves can come from different angles of the head: top (axial), side (sagittal), or the front (coronal). This could theoretically cause the model some difficulty. Upon looking at them it can be somewhat hard to distinguish between the different types by the human eye when first looking at them as we see below.

A collage of images of a brain

Description automatically generated with low confidence

Upon inspection the way I’ve identified the distinction between the different types is this. In images with no tumor, the “X” shape in the middle of the axial images is more distinguished. In pituitary tumors, the tumor appears as lighter, smaller, and right by the stem of the brain. For glioma and meningioma, there is no specific location for them. They both seem to be larger than pituitary tumors and can be located anywhere in the brain. The distinction between the two is that glioma tends to have a darker spot represent the tumor while meningioma has a lighter spot.

Methods

The architecture of my convolutional neural network is set up with the kb.Sequential framework from TensorFlow’s keras that allows you build a neural network by stacking layers in a sequential order. Before feeding the images through the model, I standardized the images to be 224 by 224 pixels and set this as the input shape which would then be scanned 3 filters (red, blue, and yellow). I then rescaled the pixel values (which are generally between 0 and 255) to between 0 and 1 as to prevent issues of vanishing gradient during adam optimization.

For the actual architecture of my model, I first had image augmentation to try to keep the model from overtraining. This involved the random flipping, zooming, and rotating of images sent through the model to try to make it better at not becoming overdependent on certain patterns in images. After that I had a cluster which repeated three times and consisted of two convolutional layers, a dropout layer, a densing layer, and a max pooling layer. These clusters would be increasing in the number of nodes as the model went along. It then had several densing and flattening layers slowly reducing the nodes back to its output of 4 for the number of classes.

I also created a second model that involved transfer learning by importing in the VGG16 pretrained model for image classification as basis for comparison to my model in seeing how it performs. The vgg model is pretrained on the ImageNet dataset consisting of over a million images in 1000 different classes. This model includes the vgg model followed by a flattening layer and two clusters including a densing layer, batch normalization, and dropout. And then finally a densing layer to get the same output of 4 nodes for the number of classes.

Results

I ran the models for 10 epochs each using adam for optimization and categorical crossentropy for loss, with the metrics of accuracy and mean square error. The constructed model did pretty well with a training accuracy of 79.67% and a testing accuracy of 68.12%. My model was a bit overfit with about a 12% difference between the training and testing accuracy. In comparison to the vgg model which had a training accuracy of 90.04% and a testing accuracy of 85.51%, my model didn’t do as well as the vgg pretrained model which also happened to be a bit overfit with a difference of 5%. This makes sense as the vgg model is much better at picking up patterns because it has been pretrained on a much larger dataset than my constructed model.

A graph with blue and orange lines

Description automatically generated with low confidenceA graph with blue and orange lines

Description automatically generated with low confidence

When looking at graphs that show how the different models scored over time in different metrics, the models look different. We can see that the graph for the accuracy of the constructed model shows a fairly steady rise in training accuracy, with a sporadic rise in testing accuracy that seemed to randomly hit valleys. For the VGG model, it gains high training and testing accuracy early, and for the training score, it grows a little, but caps out around 90, whereas the testing accuracy goes up and down in a non-consistent manner.

A picture containing text, screenshot, line, diagram

Description automatically generatedA picture containing text, screenshot, line, diagram

Description automatically generated

When comparing the loss and mean squared error, the same phenomena occurs, but inverse with the vgg model’s loss remaining consistently low and the constructed model lowers over time.

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

A picture containing text, screenshot, diagram, rectangle

Description automatically generated

When comparing the confusion matrices for both models, we can see that they both did fairly well with them sharing flaws in some of the same areas. They didn’t do great in classifying images as meningioma. The vgg model did fairly well with still the majority of true meningioma images being predicted correctly, but the constructed model didn’t do particularly well in identifying meningioma as less than half of the true values were predicted correctly. Meningioma makes sense for being the most mistaken tumor as it shares color pituitary and location with glioma.

Reflection

One of my main issues in this project was my limitation in computing power. For the majority of the time that I was trying to refine my model, I was working in a hotel out of town that had less than optimal internet. This caused models taking several hours to run for 5 epochs. Because of this, I struggled to find ways to tune my constructed model to make it less overfit. My original plan was to create that included both the vgg model and a complex architecture, but I kept getting issues with dimensions, and my inability to train at a fast pace led me to resort to making two different models for comparison.

I had an issue for a while where I was unable to perform metrics that used predicted values as whenever I tried, it would say that my model accuracy for either model was between 20-30%, which is the equivalent of randomly guessing the class of any given image, while the model itself said its accuracy was much higher. I eventually figured out that the method I was using to analyze results converted the data to numpy arrays which was not compatible with the model’s results for some reason.