Identifying Cancerous Tumors using Convolutional Neural Networks

Ethan V. Sheridan-Smith: 19016074
Thomas E. Wilson: 19012042
Cameron Mackay Grant: 19013163
Lewis Clark: 19043679

Abstract

The identification of brain tumors is mostly conducted by neuropathologist and can be extremely challenging to detect within its initial stages, this can take decades of training. The various explicit features used by neuropathologist can be employed by Convolutional Neural Networks (CNNs) to classify brain scans within our dataset. The Brain Tumor dataset itself contains 3762 images, 1683 of which being positive for cancerous tumors and 2079 being negative. We will train our CNN on these images with the goal of successfully classifying them. Afterwards an ensemble will be created to test and compare to the CNN to see which results in more accurate predictions and fewer false negative predictions.

Our results show the power of CNNs in multiple applications and the importance they can have in the field of neuropathology.

1.Introduction

The task faced is to identify Brain Tumors within the brain scan given minimal information on the case itself. This being an important task in modern medicine with the world having a larger aging population and an increase in cancer rates globally. The dataset found was a collection of files, two being .CSV files consisting of brain information and the outcome of the scan (1 being cancerous 0 being tumor free). The final file being filled with the brain scans and each image name. The dataset was located on the site Kaggle a well-known, well used and vastly documented website for ML notebooks and Datasets such as ours. The website aims to make machine learning datasets and information more widely available to the growing userbase. (Bohaju, 2020)

Identifying the presence of a Brain Tumor is usually carried out by that of very well-trained neuropathologist with decades of experience both in industry and carried through generations of advancement and discovery in the field of neuropathology. The complexity of the brain can make it extremely hard to identify Tumors with the naked eye, CNNs can facilitate the early discovery of Tumors and reduce the load on neuropathologists enhancing accuracy and detection. Features that the CNN will be looking at are the visible differences in the brain matter and the tumor which on most cases is a fluorescent white color or a darker grey depending on the scan. Furthermore, looking at the shape of the brain and where matter may be irregular.

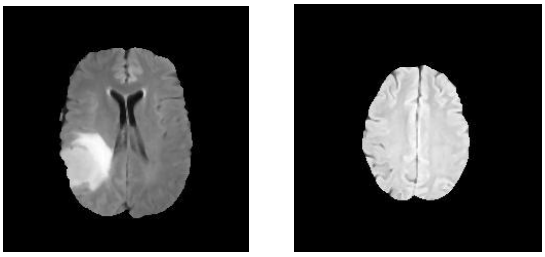

Figure 1. Both these brain scans can be found in the Brain Tumor folder of the dataset the left is positive for a cancerous brain tumor whereas the right is negative.

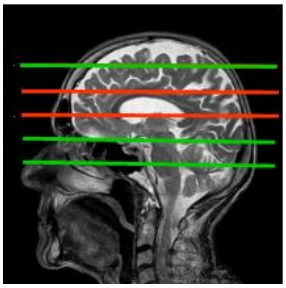

Figure 2 shows a horizontal view of each of the levels of our vertical scan.

Many of the brain scans we find in the file can vary in detail, color, and complexity (as seen in figure 1). The reason we see this happening within our dataset is the different MRI scan formats used by the NHS and whether the patient has had Gadolinium Contrast Medium (GCM) dye used in the procedure. GCM is used to highlight and contrast areas of the scan to more accurately display how the body is working. Within our Brain Tumor dataset multiple of our scans are of different levels depicted in figure 2. Most of the scans can be found within the middle two red lines of figure 2 however some of our scan can be found in anywhere within the green lines.

The area in green is still required within the dataset as tumors can still be observed in the green

boundaries. Images of the brain at these levels are very different to the images at red levels due to the "empty space" which is not part of the brain. Our classifier will have most trouble differentiating between images at these green levels, showing parts of the head not contained in the brain, and images at red levels containing brain tumors.

2.Related work

The problem we have chosen to take is widely documented and has been covered by many other teams. For example, Nan Zhang covers how feature selection can be used for brain tumor segmentation. Their approach to this problem involved learning the brain tumor and selecting the best features from the multi-spectral MRI scans which were done in Kernel space. Automatically segment the tumor into new data using an SVM with the existing feature selection & use a medical image analysis system to quantitatively watch the tumors evolution. The problem we have chosen to take is widely documented and has been covered by many other teams. For example, (Zhang, et al., 2011) covers how feature selection can be used for brain tumor segmentation. Their approach to this problem involved learning the brain tumor and selecting the best features from the multi-spectral MRI scans which were done in Kernel space. Automatically segment the tumor into new data using an SVM with the existing feature selection & use a medical image analysis system to quantitatively watch the tumors evolution. (Zhang, et al., 2011)

Another example was work done by Xiomei Zhao. In this approach they decided to use Fully Convolution Neural Networks with Conditional Random Fields in a unified framework to obtain optimal segmentation of the brain tumor. The steps taken involved training the FCNNs on image patches and trained the CRFs as Recurrent Neural Network (RNNs) with the parameters of the FCNNs on image slices. The finetuning of the models focused on the FCNNs hyperparameters and the Image slicing of the CRFs-RNNs particularly using coronal, axial, and sagittal views for later combination. (Zhao, et al., 2018)

The work implemented shares similarities from different areas of both projects. In the work covered by (Zhang, et al., 2011) we use feature selection also done in kernel space to extract the best features for the brain segmentation however instead of using a medical image analysis system to spot the tumors we use a CNN similar to (Zhao, et al., 2018) We also used image slicing however we only had axial slices as the dataset we used did not contain sagittal or coronal views, and thus we were unable to combine the image information from each plane. Furthermore, we did not observe the tumors evolution with later imaging as this system has not been developed with the intention of monitoring the tumors and only identifying them.

In future we would like to implement stages that can combine image slices from different axis as well as take advantage of the different neural networks finetuning advantages from using RNNs image tuning and CNNs hyper-parameter tuning to make the system more accurate.

3.Dataset

The Brain Tumor dataset was compiled on the website Kaggle but originated from the BRATS (Brain Tumor Image Segmentation Challenge) machine learning competition (BRATS, 2015) using the Cancer Imaging Archive to base its image selection (National Cancer Institute, 2020) and further so the Kistler visual skeleton database (kistler, 2013). The culmination of these datasets and journal articles gave us the Kaggle database we use today (Bohaju, 2020).

As prior mentioned in the introduction the data set contains 3762 images of brain scans from a range of levels of the brain depicted in figure 2. Each image is a slice of an MRI scan meaning multiple images of the same brain may be given. The dataset contains both brains with and without tumors. The dataset comes with an accompanying csv file containing metadata about each image. This file can be used to train classifiers when image recognition is not available, in our case it is used to train the ensemble.

After locating and implementing the data the next step is data preparation. To have a dataset that is prepped for implementation it must first go through some stages to ensure that the dataset can produce the most accurate models. For our dataset the main step used to ensure data preparedness was to check for

null values. Fortunately, due to the nature of the source, no null values were found. The only other data preparation that was deemed relevant was transforming each image into 224 by 224 pixels with three RGB channels so that the resulting images could be classified by the tested and trained CNN. The CSV is required to pass through the ensemble with the hope of minimizing variance in classifications and hopefully having zero false negatives.

4.Methods

4.1 Overview

We are using two main prediction methods to classify the images in our dataset. The first method is a Convolutional Neural Network (CNN) using the TensorFlow module. The second will be an ensemble using the Bagging method.

4.2 CNN Method

Our CNN takes 224x224 RGB pixels as its input, from this it creates a feature map which can then be used to train the CNN against each image's classification. The CNN will build up a set of rules based on these classifications which it can then use to make predictions on further images within the test set.

A loss function is required to measure how good the prediction model performs against expected outcomes. In our case we will be using a Binary Cross-Entropy loss function, this works by comparing prediction weights against actual class outputs. A negative score is calculated based on distance from expected values; the formula for this can be seen in figure 3.

$$-\frac{1}{N}\sum_{i=1}^{N}\left(\log\left(p_i\right)\right)$$

Figure 3 shows the formula for calculating binary cross-entropy or log loss, defined as: The negative average of the log of corrected predicted probabilities. (Saxena, 2021)

An optimizer is needed for the CNN which is used to modify the weights of the neural network. The optimizer focuses on minimizing the loss function across multiple epochs by following the gradient of said loss function. We have used the SDG optimizer for our CNN, this helps with generalization and lessens the negative effects of overfitting the data.

CNNs suffer from the fact that they are stochastic learning algorithms meaning they have high variance and can provide vastly different predictions across multiple runs. To combat this we can run an ensemble using the provided csv data to produce more accurate results.

4.3 Ensemble Method

The bagging classifier cannot derive image components from our data, we will be using the predefined features in our datasets accompanying CSV file.

The bagging classifier is an ensemble meta-estimator that combines the predictions of multiple different estimators to reduce variance in the classification results. It works best on datasets that cause a lot of variances. A visual representation of how this works can be seen in figure 4.

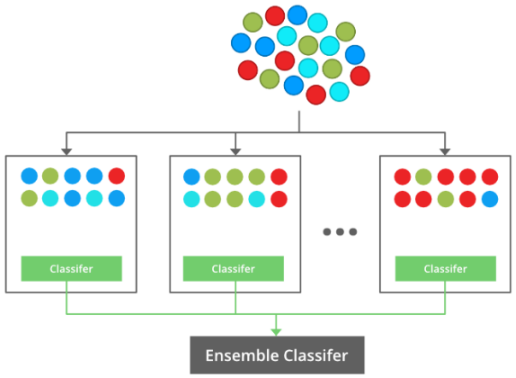

Figure 4 shows how a bagging classifier splits the data into multiple estimators and combines the results of these to create the ensembles classification. (Ranjan Rout, 2021)

In our case, the base estimator used by the bagging classifier will be the Decision Tree classifier from the sklearn.tree library. An example of how it does this can be seen in Figure 5.

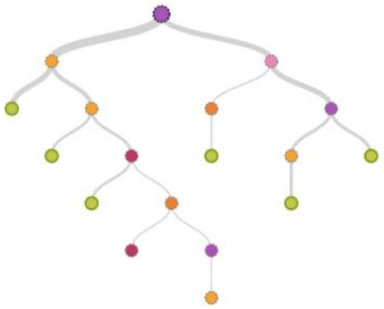

Figure 5 depicts the format of a standard decision tree classifier. (Saxena, 2017)

The decision tree classifier works by creating nodes based on a root decision rule. Nodes are continually created with subsequent decision rules.

5.Experiments

5.1 Setup

To successfully train our classifiers, we required various libraries. Numpy and Pandas are used extensively for data handling, matplotlib is used to plot and export graphs, sklearn is a very import library as it gives us access to metrics and model selection tools such as “confusion matrices” and “test_train_split” as well as ensembles. Finally, we are using tensorflow and keras to build a Convolutional Neural Network for image recognition.

Each image has 13 different data points deferred from it; these are all contained in the csv file. To train our ensemble we have removed the correlation and coarseness datatypes as these give no real insight into the existence of a brain tumor.

5.2.1 Experiment 1 - CNN Layer size

Based on 10 epochs we set out to find the optimal number of layers of a Convolutional Neural network (CNN) between 2-5. The accuracy was recorded after each iteration.

```
- loss: 0.2504 - accuracy: 0.8982
```

Layers = 2

```
- loss: 0.2497 - accuracy: 0.9020
```

Layers = 3

```
loss: 0.3196 - accuracy: 0.8640
```

Layers = 4

```
loss: 0.4538 - accuracy: 0.8078
```

Layers = 5

After experimenting with the CNN Layer size, the optimal number was 3 layers with an accuracy of 0.902 after 10 epochs.

5.2.3 Experiment 3 – CNN Evaluation Metrics

Predictions gathered from the CNN after training and testing can be easily visualized used a confusion matrix. This allows to gain a deeper understanding of how accurate the classifications were. Please reference figure 6.

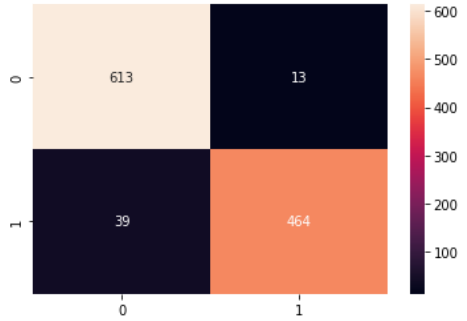

Figure 6 displays a confusion matrix. Predicted class is on the y-axis while true class is on the x-axis.

In this case, the CNN correctly predicted that 613 images did not have tumors and 464 showed a present tumor. The CNN incorrectly predicted 13 people to have tumors when they didn’t. We are trying to minimize the number of false negatives (39 made) as these are the most detrimental to the use case of the classifier; classifying an image as not having a tumor when one is present.

We are also able to visualize the CNNs training progress across multiple epochs. We tracked the loss and accuracy; graphs can be seen in figure 7 and 8 respectively.

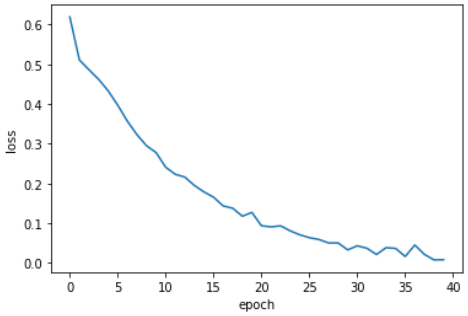

Figure 7 displays a graph of the loss value.

The loss calculated from the loss function essentially shows us the difference between predictions and expected results, thus making it a core metric we are aiming to minimize.

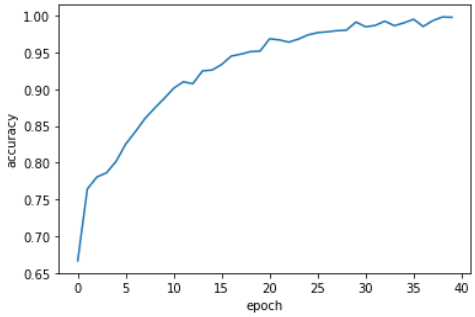

Figure 8 displays a graph of the accuracy value.

The accuracy value is calculated by:

$$accuracy = \frac{number\ of\ correct\ predictions}{number\ of\ total\ predictions}$$

It can be used to gauge an idea on how successfully the classifier is making predictions. In this instance the CNN reached an accuracy value of 97.8%.

5.3.1 Experiment 4 – Ensemble Hyper-Parameter tuning using Grid Search

We conducted a grid search on the decision tree classifier .

Criterion has 2 parameters – Gini & Entropy which measure the quality of a split. Gini measures the frequency of randomly labeled elements being mislabeled. Entropy measures the disorder of the features.

Max depth has a range 10-30 and is the maximum depth of the tree nodes.

Min samples split has a range of 2-100, it is the minimum number of samples required to split an internal node.

```
{'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 15}
```

The results showed that Entropy was more useful than Gini and that max depth and min samples split's hyperparameters were 10 & 15 respectively.

The bagging classifier has 3 hyper-parameters.

Max features has a range of 0.2-1.0 and is the number of features that are drawn from x to train the base estimators.

Max samples has a range of 0.2-1.0 and is the number of samples that are drawn from x to train the base estimators.

N estimators has a range of 100-500 The number of base estimators in the ensemble.

```
{'max_features': 1.0, 'max_samples': 1.0, 'n_estimators': 200}
```

The results of the bagging classifier showed that n estimators were the most effective at 200 and max features and max samples were most effective at 1.0 respectively.

5.3.2 Experiment 5 – Ensemble Evaluation Metrics

An ensemble was also used to see if the result The confusion matrix generated from the decision tree classifier and bagging classifier ensemble resulted in:

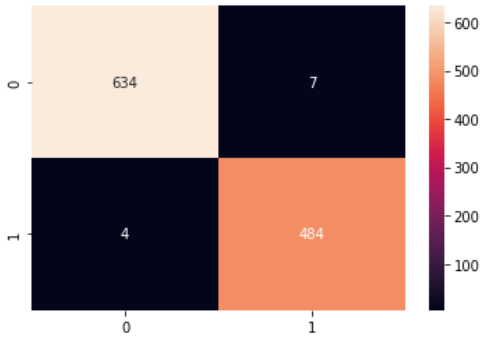

Figure 9 displays a confusion matrix from the ensemble.

As it can be clearly seen the number of false negatives has dropped heavily from the previous 39 down to 4.

6.Results and discussion

6.1 Qualitive Analysis

The primary aim of our approaches was to minimize false negatives, this is because in a physical setting false negatives would prove to be the most detrimental as they would be tumors that would not be flagged and therefore not looked at by doctors putting a patient's life at risk. Vice versa it is important to also reduce false positives however they are considerably less impactful as a doctor would then look at the flagged scan and can tell that there is no tumor present in the scan.

6.2 Method Comparisons

The CNN method proved to be slightly less accurate that the ensemble method with an average 2% difference between accuracy ratings however was much worse than ensemble at reducing false negatives in a microcosmic sense. This can be seen from the confusion matrices in figures 8 and 9 with the CNN producing 39 false negatives while the ensemble only produced 4.

6.3. Conclusion

In this problem, we introduced an approach of feature selection with a CNN to identify the presence of brain tumors. Our focus for the system was prioritizing the bias of identifying a tumor, even when this proved false, due to the severity of a lack of identification. Our system successfully identifies brain tumors in image slices from a dataset of around 3,760 images, using a CNN. Further, the system also identifies brain tumors on CSV data using an ensemble method with 13 columns of different numerical data on the same image slices. In a real-world medical setting, the image slices and CNN identification proves to be more applicable due to its faster processing of images and simpler input requirements. However, in a research background, the CSV data ensemble method was far more accurate but required longer processing of the data making it a less flexible system to use when time is a constraint. The accuracy of the image identification was around 97% making the results very accurate, however, the ensemble method had an accuracy of around 99% which made it the favorable option with regards to such a fatal condition.

In future work we would like to build multiple different CNNs each using different optimizers such as Adam or SDG and different loss functions Binary and Categorical Cross-Entropy. We could then compare the different performances or accuracies of these models to implement a better prediction mechanism.

7. References

Bohaju, J., 2020. *Brain Tumor dataset.* [Online]
Available at:
https://www.kaggle.com/datasets/jakeshbohaju/brain-tumor

BRATS, 2015. *BRATS2015.* [Online]
Available at: https://www.smir.ch/BRATS/Start2015

kistler, m., 2013. The Virtual Skeleton Database: An Open Access Repository for Biomedical Research and Collaboration. *journal of medical internet research,* november.15(11).

National Cancer Institute, 2020. *cancer imaging archive.* [Online]
Available at:
https://www.cancerimagingarchive.net/collections/

Ranjan Rout, A., 2021. *GeeksForGeeks.* [Online]
Available at: https://www.geeksforgeeks.org/bagging-vs-boosting-in-machine-learning/
[Accessed 23rd April 2022].

Saxena, R., 2017. *DataAspirant.* [Online]
Available at: https://dataaspirant.com/decision-tree-algorithm-python-with-scikit-learn/
[Accessed 24 April 2022].

Saxena, S., 2021. *Binary Cross Entropy/Log Loss for Binary Classification.* [Online]
Available at:
https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/

Zhang, N. et al., 2011. Kernel Feature Selection To Fuse Multi-Spectral MRI Images For Brain Tumor Segmentation. *Computer Vision And Image Understanding,* 115(2), pp. 256-269.

Zhao, X. et al., 2018. A Deep Learning Model Integrating FCNNs and CRFs For Brain Tumor Segmentation. *Medical Image Analysis,* Volume 43, pp. 98-111.