# Brain Tumor Detection

**Evan Bosia** **Igor Solomatin**

## Abstract

This paper presents our attempts of using convolutional neural networks on MRI scans to detect brain cancer. The data sampling and experiments are included in the paper. The results of our experiments show that transfer learning on MobileNetV2 with l1 regularization produced the best accuracy for brain tumor detection at 99.7%.

## 1 Introduction

Approximately 700,000 Americans are living with a brain tumor, 70% of which are benign and 30% are malignant. It is estimated that 84,000 people will receive a brain tumor diagnosis in 2021 and 18,600 of those will be lethal [9]. Despite the seriousness of brain tumors and their impact on the quality of life of the patients, there are no widely utilized tests to screen for brain tumors. Most brain tumors are found when a person goes to a doctor because of the symptoms they are having [6].

Magnetic resonance imaging (MRI) and computed tomography (CT) scans are used most often to look for brain diseases. MRI scans are very good for looking at the brain and are considered the best way to look for tumors in this area. The images they provide are usually more detailed than those from CT scans [8]. An MRI with contrast dye is the best way to see brain tumors [7].

The detection of tumors from MR images is the primary responsibility of neuro-radiologists and neuro-oncologists. The accuracy of the outcome depends majorly on their experience but also on their current overall condition. Is the person looking at the MR image depressed, under influence, tired? Is their mind occupied by something else entirely? Traditional human limitations can be overcome with the use of computer-aided technology [4]. If a detection can be made with high accuracy and precision, brain tumors can be detected as early as possible and misdiagnosis can be avoided.

### 1.1 Background

We trained a convolutional neural network (CNN) for predicting brain tumors. Convolutional neural networks, which are specifically designed to deal with the variability of 2D shapes, are shown to perform well on image-based datasets [10]. The CNN structure consists of simple cells and complex cells. This structure is inspired by the receptive field structures found in the human primary visual cortex first discovered by Hubel and Wiesel (Hubel & Wiesel, 1962). The convolutional layer corresponds to simple cells and the pooling layer corresponds to complex cells in the primary visual cortex [3].

The image received by the retina of the human eye is not immediately processed in its entirety by the visual cortex. Each neuron processes only a limited region of the image. Regions of different neurons overlap to cover the entire input image. Similarly, a CNN applies a smaller filter (3x3 pixels or 5x5 pixels are typical values) to process the whole image [3].

A typical CNN consists of three layers: a convolutional layer (including ReLU), a pooling layer, and a fully connected layer.

A convolutional layer computes a dot product between two matrices, where one matrix is the kernel (learnable parameters) and the other matrix is the image region covered by this neuron. The kernel

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

is smaller but deeper than the image. If the image consists of three channels (RGB) then the kernel will have a depth of three for each of the channels. A convolutional layer is the main component of a CNN's architecture because it produces an output based on which parameters activate during a filter pass. In effect, it learns what is depicted in the image.

A pooling layer modifies the output of a previous layer by down-sampling the result using statistics of nearby slices of the output. The idea behind the pooling layer is that a relative position of a feature we are looking for is more important than its absolute position in the image. By down-sampling, we can reduce the image size which in turn leads to savings in memory and processing time.

A fully connected layer performs the final classification. It is connected to all neurons in the previous layer and all neurons in the next layer which gives us the ability to compute their activations using matrix multiplication and a bias offset.

## 2 Data Methods

Our model was trained and tested on Br35H Brain Tumor Detection 2020 dataset [1]. This dataset is available on Kaggle.com. It consists of 3060 MRI (Magnetic Resonance Imaging) brain images, 1500 of which have a brain tumor ("positive" or "yes" images), 1500 do not have a brain tumor ("negative" or "no" images) and 60 are left undetermined for the NN (neural network) to make a prediction on. For the purposes of our experiments, we removed the 60 undetermined images.

The dataset does not provide a test set to evaluate any models. For our experiments, we randomly chose 10% of the positive and negative images out of the training data.

### 2.1 Data Transformations

Before training the neural network, the data needed to be formatted into a uniform fixed size. One of the difficulties with the dataset were the images are not standardized: the sizes of the images were not uniform. Figure 1 is a scatter plot of the size of the images.

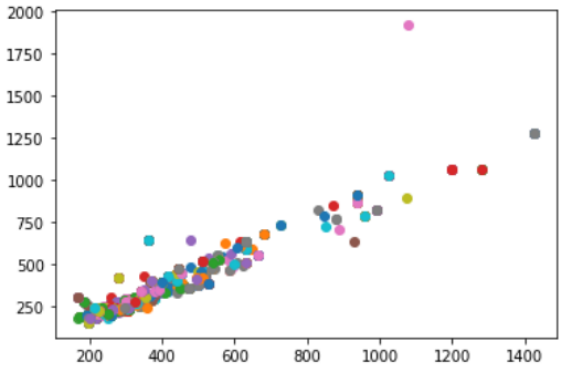


Figure 1: Aspect Ratios 1

The solution was to crop out the brains in the images because human brains should be roughly the same aspect ratio. Every image had an area of interest lighter than the background, so a threshold could be used to locate and segment the brain in the image.

Algorithm:

1. Apply Gaussian blur image to remove noise
2. Apply global threshold of 45
3. Find the largest contour in the image
4. Find the extreme-points of the contour
5. Crop the image on these extreme points
6. Scale cropped images to 224x224 (expected by model)

The cropped images are still not uniform in size, but they are very close. Before inputting into the model, these images are all scaled to 224x224.

## 3 Model Methods

We trained a few different neural networks to see if there would be any improvement through altering the model itself. We used Tensorflow [2] to develop and evaluate the models. These are the models trained (in general):

1. One convolutional layer
2. Two convolutional layers + dropout
3. Three convolutional layers + dropout
4. Data augmentation + three convolutional layers + dropout
5. Transfer learning off of MobileNetV2 [5]
6. Regularization Tests

All of the experiments except for the transfer learning experiment followed this high-level architecture:

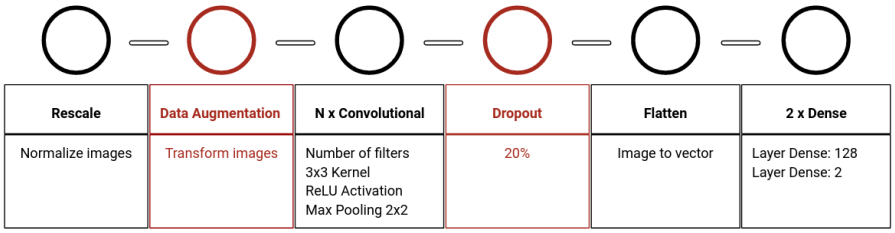


Figure 2: Model Architecture

The red blocks are optional depending on the experiment. The number of convolutional layers and the number of filters in each layer are also experiment dependent.

## 4 Experiments

We ran different experiments, each with the goal of improving the model. In all experiments, early stopping was used to make sure the models could make it to their best validation accuracy.

### 4.1 Experiment One

The first experiment had a single convolutional layer. This layer had 16 filters with a 3x3 kernel. Early stopping was used to prevent training the model for no gain.

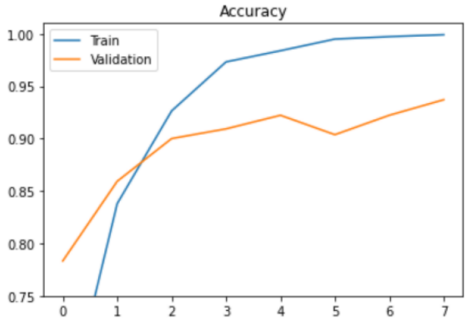


Figure 3: Experiment 1

Examining the accuracy curves between the training and validation data, it is clear the model is overfitted to the data. The best validation accuracy of this model was 93.7%.

### 4.2 Experiment Two

The second experiment had two convolutional layers. The first convolutional layer is the same as experiment one: 16 filters with a 3x3 kernel. The second convolutional layer also uses a 3x3 kernel but has 32 filters. We also added a dropout layer (0.2) before the dense layers at the end of the model.

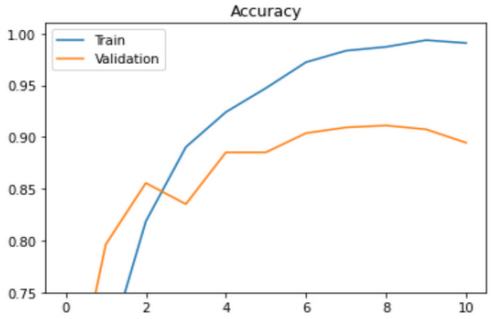


Figure 4: Experiment 2

The training and validation accuracy curves also signal that the model is overfitted to the data. The validation accuracy of this experiment was only 91.1%.

### 4.3 Experiment Three

The third experiment had three convolutional layers. The first two layers are the same as experiment two. The third layer is again a 3x3 kernel but with 64 filters. The dropout layer is also included.

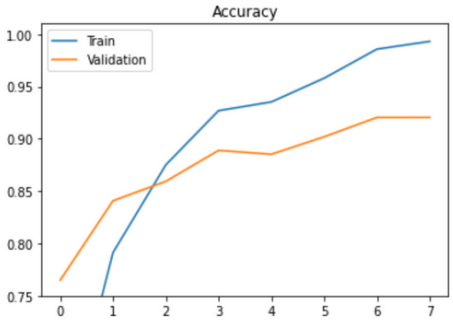


Figure 5: Experiment 3

There is still overfitting between the training and validation accuracy. The maximum validation accuracy of this step was still 91.0%. There was no improvement in validation accuracy from the second experiment.

### 4.4 Experiment Four

The fourth experiment takes the same model architecture as the third experiment but applies data-augmentation techniques to the input images. Data augmentation applies random transformations to the input images to synthetically give the model more data. This helps with overfitting because overfitting can be helped by increasing the amount of training data.

The data augmentation transformations applied in this experiment were:

- Random flips horizontal

- Random flips vertical
- Random rotations 10%
- Random zoom

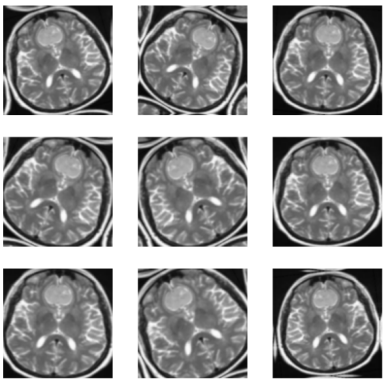

Figure 6: Image Augmentation Examples

Another possible data augmentation technique is random cropping. This was not appropriate for this dataset because the classification features in the image are localized to a part of the random brain. Random cropping could have removed the tumor from the image, which would cause misclassification.

The comparison of training and validation accuracy curves shows that data augmentation helps with overfitting. The validation accuracy of this model was 95.6%.

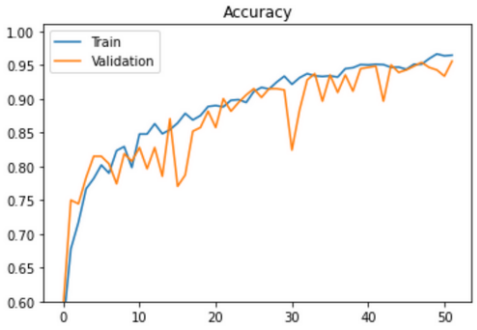


Figure 7: Experiment 4

### 4.5 Experiment Five

Experiment Five: Transfer learning allows a model to leverage the layers of a pre-trained model while training on the input data. This in theory allows for models to have robust outputs with less training data. For this experiment, we chose to train off of MobileNetV2 [5], which TensorFlow provides as a network available for transfer learning. We chose the pre-trained version of MobileNetV2 that was trained on ImageNet .

After 60 epochs, the model achieved a 94% validation accuracy. There was little to no overfitting here which opened the option of fine-tuning to improve performance.

To boost the performance of the model, a technique in transfer learning is to open some of the base model's layers to training. There is a risk of overfitting here because in the challenge there is a limited amount of training data, but the reward is increased performance. At this point, the model did not have overfitting, so it seemed like there could be benefits to fine-tuning the model.

The last 50 layers of the MobileNetV2 were opened to training. The red line in figure 8 shows the epoch that these layers were opened for training. By doing this, the model was able to be improved

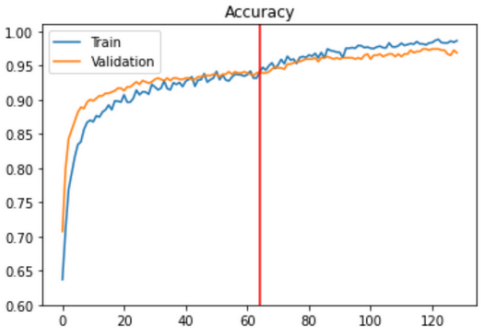


Figure 8: Experiment 5

from 94% validation accuracy to 97.4% validation accuracy. There was some amount of overfitting which was somewhat expected with the small training dataset, but the validation accuracy score was promising.

### 4.6 Regularization Experiments

Regularization can be used to fight overfitting and improve model performance. After completing the other tests, we tried adding the default l1 regulizer to the last layer of our model. We chose to try this technique on our best models: data augmentation 3 layer, and transfer learning on MobileNetV2, as these were the two best models from our previous experiments.

#### 4.6.1 Experiment 4 Regularization

Adding l1 regularization to the last layer of the data augmentation model from experiment 4 made a slight difference in the quality of the model.

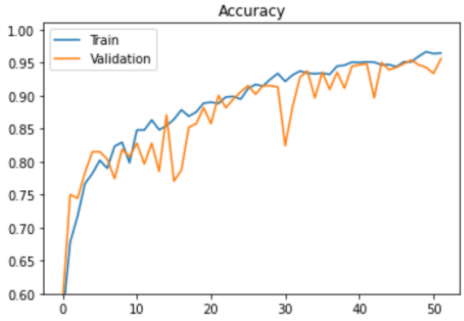


Figure 9: Experiment 4 Regularization

At the end of training, with regularization the model was able to achieve a validation score of 96.1%, which was a slight increase from the original experiment 4 result of 95.6%. The slight improvement is not too surprising, as the main benefit of using l1 regularization is fighting against overfitting. The experiment 4 model was not particularly overfit.

#### 4.6.2 Experiment 5 Regularization

Adding the l1 regularization term to the last layer of the transfer model (experiment 5) made a significant difference. At first, with the base model frozen, the model was only able to achieve a validation accuracy of 92.2%, which is worse than the experiment 5 results. This intuitively makes sense, as the benefits of regularization come with avoiding overfitting. The experiment 5 model before opening some of the base model layers was not overfit.

After opening the model to more training (red line in figure 10), there was a significant increase in validation accuracy to 99.1%. This was by far the best validation accuracy we were able to achieve.

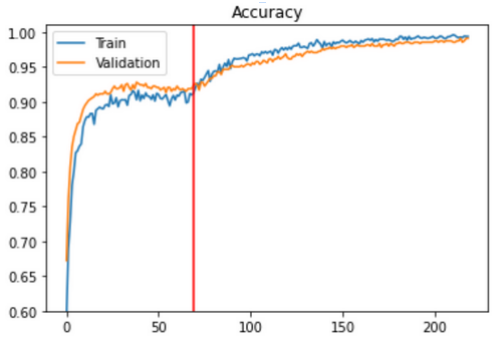


Figure 10: Experiment 5 Regularization

Adding the regularization term clearly helped the model avoid overfitting and continue to improve to the high accuracy.

## 5 Results

All of the experiments were run against the same test set of 150 positives and 150 negatives. The results are in the tables below.

| name | size (bytes) | layers | accuracy | true_pos | true_neg | false_pos | false_neg |
|---|---|---|---|---|---|---|---|
| experiment_001 | 137029 | 8 | 0.95 | 138 | 146 | 4 | 12 |
| experiment_002 | 148884 | 9 | 0.95 | 140 | 144 | 6 | 10 |
| experiment_003 | 177253 | 11 | 0.95 | 143 | 141 | 9 | 7 |
| experiment_004 | 483616 | 12 | 0.95 | 145 | 140 | 10 | 5 |
| experiment_004r | 494670 | 12 | 0.98 | 145 | 148 | 2 | 5 |
| experiment_005a | 4024888 | 7 | 0.96 | 144 | 143 | 7 | 6 |
| experiment_005b | 4165246 | 7 | 0.99 | 149 | 147 | 3 | 1 |
| experiment_005ra | 4035345 | 7 | 0.94 | 141 | 140 | 10 | 9 |
| experiment_005rb | 4180621 | 7 | 1.0 | 149 | 150 | 0 | 1 |

| name | pos_recall | neg_recall | pos_precision | neg_precision | pos_f1-score | neg_f1-score |
|---|---|---|---|---|---|---|
| experiment_001 | 0.92 | 0.97 | 0.97 | 0.92 | 0.95 | 0.95 |
| experiment_002 | 0.93 | 0.96 | 0.96 | 0.94 | 0.95 | 0.95 |
| experiment_003 | 0.95 | 0.94 | 0.94 | 0.95 | 0.95 | 0.95 |
| experiment_004 | 0.97 | 0.93 | 0.94 | 0.97 | 0.95 | 0.95 |
| experiment_004r | 0.97 | 0.99 | 0.99 | 0.97 | 0.98 | 0.98 |
| experiment_005a | 0.96 | 0.95 | 0.95 | 0.96 | 0.96 | 0.96 |
| experiment_005b | 0.99 | 0.98 | 0.98 | 0.99 | 0.99 | 0.99 |
| experiment_005ra | 0.94 | 0.93 | 0.93 | 0.94 | 0.94 | 0.94 |
| experiment_005rb | 0.99 | 1.0 | 1.0 | 0.99 | 1.0 | 1.0 |

Notably, the best performing model was the fine-tuned transfer model based on MobileNetV2 with regularization. The accuracy of this model was 99.6%, which was excellent. The only error this model made was a single false negative, so both the positive and negative precision values were very good.

## 6 Discussion

Despite the 99.6% accuracy result of the MobileNetV2 based model, there still exist some avenues of improvement. Some of these areas would have been interesting to explore if we had more time. Turning this model into a product would require additional adjustments as well. The model and methods presented here are experimentation and not necessarily something that would be used in a clinical setting.

### 6.1 Improvements

The data augmentation could be improved. The data augmentation step was chosen to have random flips, random zooms, and random rotations up to 10%. This decision was somewhat arbitrary. There is a chance this actually hurt the performance of the model because the supplied data is all in the same orientation. Maybe the tumor shape is orientation-dependent. If that is true, the model could be learning bad filters for finding the tumors when the input images are inverted.

### 6.2 Use Case

The model metric used for testing and comparison is accuracy, but this does not take into account the actual use-case of this tool. A doctor might use a model like this to filter out healthy MRI images from possible brain tumor images. In that case, the model should be biased to have fewer false negatives. Our best model had an excellent negative precision value of 99.3%. Ideally, the negative precision should be as close to 100% as possible, even at the expense of the positive precision. False positives can be reexamined by the doctor for further analysis, and there is no risk of clearing a patient with cancer.

The model is also missing interpretability. This is outside the scope of the particular Kaggle challenge, but a doctor could get more benefit from a segmented-tumor output with a positive classification. This could instead put the diagnosis responsibility on the doctor while providing a useful location output. In the ideal use case, the model would be weighted to have a very high negative precision, and on positives provide the doctor the segmented tumor. This would be the most efficient for the doctor, as the model would accurately filter out healthy brains, and provide easy diagnosis ability for the positives.

### 6.3 Future Work

Aside from tuning the model to fit the expected use case, there are other possibilities of extending the work on this problem. MRI images can be stacked to form a three-dimensional representation of the object scanned. Tumors have depth, so adding this extra dimension and taking a three-dimensional convolution could improve the model accuracy. Ideally, this would help with the collection of training data because more data should also help improve the model. These techniques could also be applied to other parts of the body, such as the lungs.

## 7 Conclusion

Convolutional neural networks seem to be a great solution for determining if patients have brain cancer. Through our experiments, we demonstrated the value of data augmentation in low training sample environments, and the power of transfer learning. Ultimately, there are a lot of avenues to expand upon this project, but for the task of binary classification the models performed well.

## References


[1] Br35H. Brain tumor detection 2020, 2020. URL `https://www.kaggle.com/ahmedhamada0/brain-tumor-detection`.

[2] Google. Tensorflow, 2021. URL `https://www.tensorflow.org/`.

[3] Lee M. Kim J., Kim S. Convolutional neural network with biologically inspired on/off relu. Technical report, ICONIP, 2015.

[4] Har Pal Thethi Nilesh Bhaskarrao Bahadure, Arun Kumar Ray. Image analysis for mri based brain tumor detection and feature extraction using biologically inspired bwt and svm. Technical report, International Journal of Biomedical Imaging, 2017.

[5] Mark Sandler, Andrew G. Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Inverted residuals and inverted bottlenecks: Mobile networks for classification, detection, and segmentation. Technical report, 2018.

[6] American Cancer Society. Can brain and spinal cord tumors in adults be found early?, 2021. URL https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/detection.html.

[7] American Cancer Society. Mri for cancer, 2021. URL https://www.cancer.org/treatment/understanding-your-diagnosis/tests/mri-for-cancer.html.

[8] American Cancer Society. Tests for brain and spinal cord tumors in adults, 2021. URL https://www.cancer.org/cancer/brain-spinal-cord-tumors-adults/detection-diagnosis-staging/how-diagnosed.html.

[9] National Brain Tumor Society. Brain tumor facts, 2021. URL https://braintumor.org/brain-tumor-information/brain-tumor-facts.

[10] Y. Bengio Y. Lecun, L. Bottou and P. Haffner. Gradient-based learning applied to document recognition. Technical report, IEEE, 1998.