CPSC 66 Final Report: Classifying brain tumors from MRI images

Ingrid Hsu IHSU1@SWARTHMORE.EDU
Moji Oludare MOLUDAR1@SWARTHMORE.EDU

Abstract

Because of its potential to support accurate and efficient decision-making, machine learning (ML) has become an increasingly important tool for automating complex pattern recognition tasks in the medical field. Our goal with this project is to apply deep learning techniques to detect brain tumors from MRI scans. In this paper, we compare the performance of traditional machine learning models– K-Nearest Neighbors, Decision Trees, Logistic Regression, and Random Forest classifiers– with Convolution Neural Networks (CNN), which are often considered the state-of-the-art models for image-based tasks. We then try to optimize our CNN using data augmentation and hyperparameter tuning with Optuna. Our results indicate that the hyperparameter-optimized CNN achieved the best performance overall, reaching 79.4% accuracy and an F1-score of 0.84. However, contrary to our initial hypotheses, both the baseline CNN and the CNN with data augmentation had lower performance than the traditional classifiers. These results highlight the challenges of applying CNNs to small, imbalanced datasets, a difficulty common in the medical imaging field. To address these limitations and move toward clinically-deployable reliability, we suggest future work should focus on expanding the dataset with alternative sources of data that are relevant and more readily available.

1. Introduction

Brain tumors are abnormal growths of cells in the brain that can be classified as either benign (noncancerous) or malignant (cancerous). Both types of tumors can have severe, even life-threatening complications if not diagnosed and treated promptly: malignant tumors spread aggressively, while benign tumors may press on surrounding structures. Therefore, early and accurate detection of tumors is critical. However, traditional diagnostic procedures rely on radiologists to manually analyze complex medical images, like Magnetic Resonance Imaging (MRI) scans. This process is time-consuming, and any delay in detection can be detrimental when considering the time-sensitive nature of the issue. Human classification may also be affected by subjectivity or variability between professionals, potentially leading to missed or inconsistent diagnoses.

Machine learning provides a promising solution to these limitations. An automated brain tumor recognition program could improve processing efficiency and possibly accuracy. This would not only benefit patients by increasing the opportunity for early intervention but also support doctors, radiologists, and researchers with insight into tumor recognition. In this work, we develop a supervised learning framework using open-source MRI data to efficiently and accurately detect brain tumors. We experimented with multiple traditional classifiers and convolutional neural networks (CNNs), which are especially effective for image processing tasks. We also applied data augmentation and hyperparameter tuning techniques to optimize performance, and evaluated model effectiveness with accuracy, precision, recall, and F1-score metrics to capture the varied costs of different error cases.

By utilizing machine learning methods, this study aims to improve the efficiency and accuracy of brain tumor detection from MRI scans, ultimately supporting medical professionals in their diagnostic process and reducing the risk of undetected or late-stage tumors.

2. Methodology

Our methodology was two-fold: we started by developing baseline models with traditional classifiers from the Scikit-Learn Python library (K-Nearest Neighbors, Decision Trees, Logistic Regression, and Random Forest) and a convolutional neural network based on standard design structures found online. We then applied data augmentation and hyperparameter tuning techniques to tailor the

CPSC 66 Machine Learning Proceedings, Swarthmore College, Spring 2021.

CNN to our problem and optimize results. This process was iterative, so we repeatedly evaluated the results of our experiments with accuracy and F1-scores then applied new techniques based on those findings.

2.1. Dataset, Exploratory Data Analysis, and Preprocessing

We developed our models with the publicly available *Brain MRI Images for Brain Tumor Detection* dataset from Kaggle (Navoneel, 2020). We specifically chose this dataset because its MRI scans were of a consistent cross-section of the brain, which we thought would help the model learn robust patterns. The dataset includes 253 images of MRI scans from individuals with brain tumors (n=155, positive class) and without brain tumors (n=98, negative class)(Figure 1). The images were in ".jpg," ".jpeg," and ".png," formats and separated by class. To preprocess the data before classification, we used the OpenCV library to convert each image to grayscale, resize to 224 x 224 pixels, and normalize brightness to a 0.0-1.0 scale. The main purpose of these steps was to ensure standardization across images and reduce features to a more manageable scale, as MRI scans are conventionally high resolution. For the traditional classifiers, the images were stored as flattened feature vectors in a Pandas dataframe with dimensions 253 x 50177, representing the number of images by the number of features for each image, with each feature corresponding to the intensity of a single pixel. For CNN classification, the images were stored in a four-dimensional tensor with dimensions 151 x 224 x 224 x 1, representing the number of images, image height, image width, and the number of channels respectively. Because the images were grayscale, we were only working with one channel.

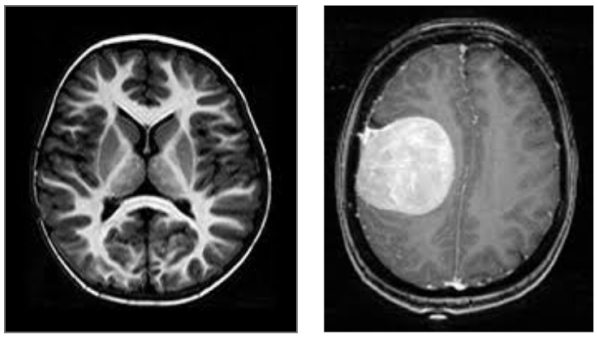

Figure 1. Example images from the *Brain MRI Images for Brain Tumor Detection* dataset showing negative (left)and positive (right) classes.

2.2. Baseline Model Implementation

2.2.1. Traditional Classifiers

For our baseline methods, we implemented K-Nearest Neighbors (KNN), Decision Tree (DT), Logistic Regression (LR), and Random Forest (RF) classifiers with Scikit-learn, an open-source Python library for machine learning. These traditional classifiers served as a reference point for evaluating whether Convolution Neural Networks provide meaningful improvements for image-based classification. We used default hyperparameters for most of the classifiers to maintain standard, reproducible baselines: KNN looked at 5 nearest neighbors with euclidean distance; the DT used a fixed random state of 0; LR was trained with stochastic gradient descent; and RF used entirely default Scikit-learn settings. We selected these four models because they provide a good representation of the content we covered in class, with a mix of linear, nonlinear, and ensemble approaches. Since we are using pixels as features, our data is high-dimensional, so we also tested Principal Component Analysis (PCA) with 100 components on the KNN classifier to see whether dimensionality reduction could improve performance by removing redundancy and noise.

2.2.2. Baseline CNN

Convolution Neural Networks are deep learning models that are especially well-suited for image classification tasks because they recognise spatial dependencies and can detect image features, such as edges and textures, through convolutional filters. These features are then compounded to provide information on more complex patterns. CNNs are typically made of convolutional layers, pooling layers, and dense layers. In convolutional layers, a set of learnable kernel weights is applied to the entire image in subsections to produce a feature map. Pooling layers decrease the dimensions of the produced feature maps while retaining the most important features to improve computational efficiency. Dense layers interpret complex information from feature maps to produce interpretable information such as class probabilities. This process makes them effective on complex image data, which is why they are the most commonly used technique for image processing in the machine learning field (Shervine, 2020); (GeeksforGeeks, 2021); (TheDeepHub, 2019). Therefore, CNNs were the main focus of this project.

We implemented our CNNs with TensorFlow/Keras, starting with a baseline CNN to serve as a reference point for more advanced optimization (TensorFlow Developers, 2025). The baseline model consists of two convolutional blocks followed by fully connected layers. Each convolutional block had two convolutional layers with a 3 x 3 kernel size, 32 filters, and ReLU activations (Figure 2). This model was then trained for 20 epochs with a learning rate

of 0.001, a batch size of 16, the Adam optimizer, and sparse categorical cross-entropy loss. The structure and hyperparameters of this network were modeled from common CNN examples found online, providing a standardized starting point for evaluating the performance of later optimization strategies .

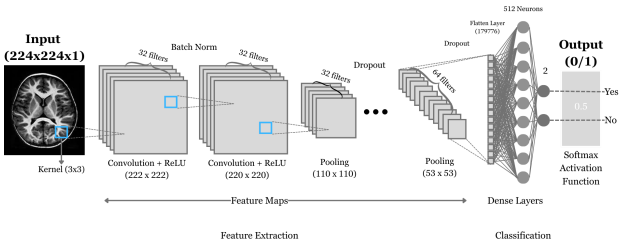

Figure 2. The baseline CNN architecture consists of two convolution blocks (only one is shown in the diagram) followed by one dense block. This architecture is based on common CNN structure found online, which were likely designed for larger datasets of natural images. We will use this as a reference point for CNN optimization techniques.

2.3. Model Optimization

2.3.1. Hyperparameter Tuning and Optimized CNN

After evaluating the traditional classifiers and baseline CNN, we used Optuna, an automated hyperparameter optimization framework, to improve our CNN performance, systematically testing different hyperparameter values (Optuna Contributors, 2019). With the Optuna "trail.suggest" functions, we experimented with the number of filters, kernel size, learning rate, activation function, number of dense units, and batch size. We integrated these tuning conditions into the baseline CNN architecture, and Optuna randomly sampled different hyperparameter combinations across 5 trials, optimizing for validation accuracy. We then used Optuna visualization methods to plot optimization history and hyperparameter importances to help understand which hyperparameters most influenced performance. Finally, we applied the best-performing configuration to train an optimized CNN for 50 epochs, then compared the results of this model against our baseline. This tuning process helps tailor the model to the specific characteristics of our dataset, and by analyzing the optimal hyperparameters, we may also gain some insight into how the model is learning.

Algorithm 1 CNN Hyperparameter Optimization with Optuna

Input: Training data (X_{train}, y_{train}), Test data (X_{test}, y_{test})
Create Optuna study to maximize accuracy
for each of 5 trials **do**
  Sample CNN hyperparameters
  Build CNN model
  Compile model with Adam optimizer
  Train model on training data
  Evaluate model on test data
  Return accuracy
end for
Output best hyperparameter combination

2.3.2. Data Augmentation

When testing our baseline CNN, we observed that it was consistently overfitting and classifying all images as positive cases, which we hypothesized was due to the small dataset size. To address this limitation, we applied data augmentation to artificially increase diversity in our training set and thereby reduce overfitting. This technique is commonly used for smaller image datasets because it encourages the model to learn more robust features of the different classes rather than memorizing specific training examples. Since the augmentation pipeline required three-channel inputs, we started off by replicating the original single-channel grayscale MRI scans across three channels with "numpy.repeat," creating a tensor of shape 151 x 224 x 224 x 3. We then prepended data augmentation layers to the optimized CNN, training and trained the model for 20 epochs. During training, these layers would apply random transformations like rotation, zoom, translation, flip, gaussian noise, and contrast (Figure 3). By switching up these features like position or angle, the model is exposed to a slightly different variation of the same image across epochs, effectively increasing dataset diversity without introducing new samples.

2.3.3. Metric Evaluation

After implementing each model and method, we evaluated and compared their performance using accuracy, precision, recall, and F1-scores as metrics. While accuracy captures what percent of samples the model correctly classifies, brain tumor detection, as with many medical applications of machine learning, has different costs for different types of errors. False positives could cause unnecessary mental, emotional, and financial stress as well as physical complications from excessive medical testing and treatment. On the other hand, false negatives could have life-threatening consequences from lack of or delayed treatment, a more costly result. To capture this distinction, we decided to cal-

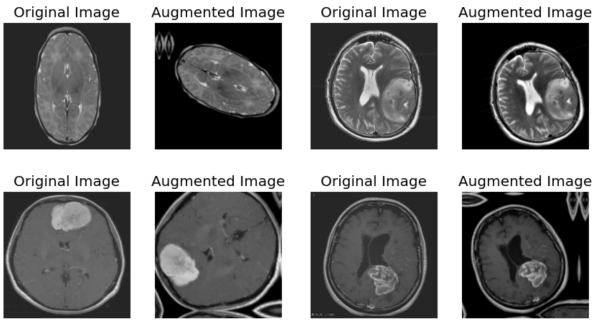

Figure 3. Data augmentation layers applied random transformations such as rotation,zoom, translation, flip, gaussian noise, and contrast to introduce diversity into the small dataset without new samples.

culate precision, recall, and F1-scores in addition to accuracy, focusing on F1 for evaluation since we are aiming for a balance between precision and recall. We also visualized the predictions and these error types with normalized confusion matrices.

3. Experimental Results

3.1. Traditional Classifiers

Our standard classifiers achieved accuracy and F1 scores of between 71.6%-76.4% and 0.74-0.81 respectively, indicating that traditional machine learning models are able to capture some meaningful patterns from the raw MRI pixel data (Table 1). Of the four classifiers tested, Random Forest had the best overall performance (accuracy=76.4%, F1=0.81), likely because ensemble methods can model nonlinear relationships that are present in image data while being more robust against noise. KNN had the worst performance (accuracy=71.6%, F1=0.74), likely due to the curse of dimensionality from the large number of features: in such high-dimensional image spaces, all points appear similarly distant, reducing the effectiveness of the distance-based comparisons used by KNNs. Surprisingly, Principal Component Analysis further reduced KNN accuracy from 71.6% to 62.8%, suggesting that the drastic dimensionality reduction may have lost relevant information. This poor performance may also be because our dataset had slight misalignments and scaling differences across images, so PCA on the pixel intensity values was less effective.

3.2. Baseline CNN

Our baseline CNN performed worse overall than all of our traditional classifiers, reaching an accuracy of 59.80% and an F1 score of 75% (Table 1). From the confusion matrix,

Table 1. Accuracy and F1-Score metrics from traditional classifiers and baseline, optimized, and augmented CNN models.

MODEL	ACCURACY	F1-SCORE
K-NEAREST NEIGHBORS	71.6	0.74
DECISION TREE	73.5	0.77
LOGISTIC REGRESSION	76.4	0.80
RANDOM FOREST	76.4	0.81
BASELINE CNN	59.8	0.75
AUGMENTATION CNN	59.8	0.75
OPTIMIZED CNN	79.4	0.84

we can observe that the model predicted every test sample as positive, resulting in perfect recall (1.0) but low precision (0.6) (Figure 4). These predictions explain why the accuracy is effectively capped at 59.80%, the proportion of positive samples in the test set for this train-test split. They also explain why accuracy changes noticeably between runs with different train-test splits. This behavior is an example of overfitting. The model reaches 100% accuracy when training, which indicates that it is learning, but because of the small size of the dataset, it is likely unable to generalize beyond the training data. In general, CNNs perform best with at least a few thousand observations, with popular image datasets like MNIST or CIFAR-10 containing upwards of 60,000 samples (LeCun et al., 1998); (Krizhevsky et al., 2009). With limited training data, our model may be memorizing examples instead of learning more robust features, and the imbalance of the dataset, leaning more towards positive cases, may be why it favors positive classification. This CNN's architecture and hyperparameters were also adapted from generic CNN examples found online, so they are likely better suited for large sets of natural images rather than the small-scale, relatively homogeneous images that we are working with.

3.3. Data Augmentation

Applying data augmentation did not improve performance over the baseline CNN: it produced the same results, classifying all samples as positive and returning a 59.80% accuracy (Table 1)(Figure 4). This was surprising, as we expected data augmentation to address the data scarcity limitation that we hypothesized was impeding the baseline CNN. Data augmentation is typically good at increasing dataset diversity and reducing overfitting, but perhaps the MRI scans in this dataset are similar enough that the transformations we applied, like rotation, translation, and flipping, did not introduce enough variation. Still, looking at Figure 3, these augmented images appear meaningfully altered, suggesting there should be at least some marginal improvement. Instead, it is possible that our model was al-

ready capturing all the low-level patterns the augmentation could introduce, meaning these transforms could not produce the same effect as actual new data. As a result, our model continued to overfit and collapse to predicting the majority class.

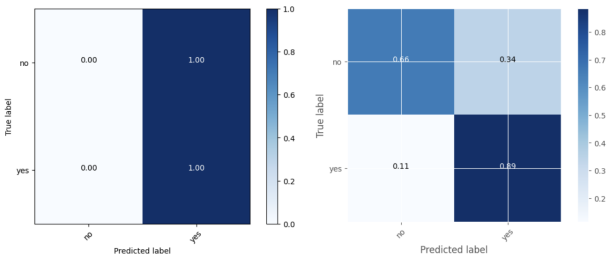

Figure 4. The baseline CNN and the data augmentation CNN achieve the same confusion matrix (left). Both models predicted the majority class, positive, for all test images, resulting in a capped 59.80% accuracy. In comparison, the confusion matrix from the optimized CNN shows more diversity in prediction, although still leaning more towards positive predictions, resulting in a lower number of false negatives.

3.4. Hyperparameter Tuning and Optimized CNN

Hyperparameter tuning had a substantial impact on CNN performance. Results ranged widely across trials, demonstrating that the baseline CNN was highly responsive to model configuration. Across multiple tuning runs, we found batch size was consistently considered the most important hyperparameter, which may be reduction of overfitting due to noise (Figure 5).

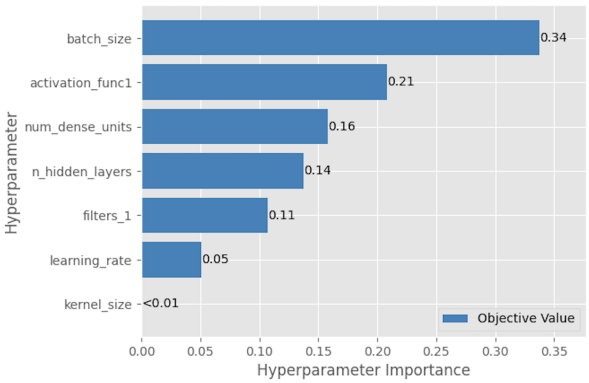

Figure 5. Optuna visualization showing which of hyperparameter importances during the tuning process. Batch size was consistently the most important hyper parameter across runs, while the next most important hyperparameter switched between learning rate, number of filters, and activation function.

Table 2. Best values for CNN hyperparameters after 5 trials of tuning with Optuna.

HYPERPARAMETER	BEST VALUE
FILTERS (CONV1)	192
KERNEL SIZE	10
LEARNING RATE	1.12×10^{-5}
ACTIVATION FUNCTION	RELU
HIDDEN LAYERS (CONV)	20
DENSE UNITS	256
BATCH SIZE	36

With a larger batch size, the model encounters more of the dataset before updating model parameters, thereby ensuring that gradient updates closely estimate the true gradient. After adjusting the batch size range, the relative importance of other hyperparameters–such as learning rate, number of filters, and activation function–varied across runs, suggesting there was no next most important hyperparameter. Ultimately, the best-performing trial used a single convolutional block with a large 10x10 kernel, ReLU activation, a low learning rate, and a high batch size, achieving a test accuracy of 79.41% and an F1-score of 0.84 (Table 1)(Table 2). This simpler architecture, with only one convolutional block, further demonstrates that the model is highly prone to overfitting, so limiting model complexity improves results. The larger kernel size captured a broader spatial context, which may be relevant for identifying tumor regions specifically. The optimized CNN outperformed all traditional classifiers and the earlier CNN baseline (Table 1). However, it still exhibited overfitting, reaching perfect training accuracy and still leaning towards positive classifications, suggesting that while hyperparameter tuning resolved some issues, data scarcity and class imbalance were still fundamental limitations (Figure 4).

4. Social Implications

The primary stakeholders for this project include patients and medical professionals, such as clinicians or radiologists. Secondary stakeholders include patients' loved ones, hospitals, and tumor researchers. If effective, all stakeholders stand to benefit from this work. Patients would have more immediate testing results and lower chances of misdiagnosis, reducing unnecessary stress and increasing chances of treatment efficacy if positive. Medical professionals would be empowered with decision support tools providing an additional perspective, increasing diagnosis confidence and reducing cognitive load. Importantly, our system is not intended to replace doctors. In practice, we see our model serving as a second opinion for diagnoses or

as preliminary screening for high-risk patients, particularly in areas with limited access to radiologists. It could also be used to help train medical students in tumor identification. In all cases, however, it should be used with a human in the loop. Otherwise, if misused, there would be significant ethical questions about algorithmic autonomy and responsibility for inaccuracies, which, as discussed previously, are highly consequential in our problem space.

To assist with this human-AI collaboration, such a system would also need to promote transparency for appropriate trust calibration. If a medical professional relied on the system too much, the same issues of autonomy and responsibility would arise. This is especially important with a model like ours, which still has a relatively high error rate and is prone to overfitting. Conversely, if a medical professional could not tell what patterns the model is learning from, they may not trust the system enough to use it to its full potential. To address these issues, model interpretability techniques like visualizing convolutional filters or analyzing influential hyperparameters could be incorporated to show what MRI regions most influence the model's prediction, promoting better context and trust of model outputs. Developers and deployers of the model should also consider the type of data they are training on; as with all machine learning, the choice of dataset can introduce risk of bias.

As with all biomedical data, neural data is also extremely personal, so privacy and data security are major relevant considerations. When training machine learning models, it is critical that sensitive data is stored securely, anonymized if possible, and used only with informed patient consent. Neural data privacy regulations are an ongoing policy concern, and studies have shown unique risks from the misuse of neural data. In 2013, researchers designed a video game where certain subliminal cues prompted a specific type of brain signal (P300 waves), and by collecting this data they were able to extract confidential information, like bank PIN numbers, from players without their knowledge or consent (Rainey et al., 2020). While such covert information extraction is less likely from MRI data, these potential dangers raise questions about what type of information can be extracted from neural data without explicit consent, and how such data collection can be protected and regulated in a secure way.

5. Conclusions

Overall, we found CNNs were not generally better than traditional classifiers at detecting tumors based on our MRI dataset, with the baseline and data augmented CNNs performing worse than all of the traditional classifiers. These results were surprising considering our initial hypothesis that CNNs would improve performance by better capturing spatial and hierarchical features present in the image data. However, we were able to achieve significant improvements in the performance of the CNN through hyperparameter tuning, particularly of batch size, reaching 79.41% accuracy and an F1-score of 0.84. Through this project, we confronted two challenges common with medical datasets: 1) the relatively rare circumstances that lead to MRI scans and the sensitive nature of biomedical information result in limited publicly available data, and 2) because people are unlikely to get an MRI scan for no reason, available data is skewed towards the positive class. We attempted to use data augmentation to reduce the impact of this data scarcity, but observed no improvement. Without solving these issues, our model's performance may not be accurate enough for our high-stakes use case. Still, it could potentially serve as an efficient preliminary screening tool or second opinion for medical professionals. In the future, perhaps it would help to incorporate brain scans from different orientations, which could increase dataset size and diversity but may also require more preprocessing and image standardization. It would also be interesting to analyze the difference between malignant and benign tumors by extending our binary classification to a multiclass setting, or to include images of tumors from other parts of the body, where larger datasets may be more readily available.

Acknowledgments

We would like to thank Professor Gabriel Hope for his constant support and feedback throughout this project, and our fellow students in CS 066 for their insightful questions during our presentation. We would also like to acknowledge Navoneel Chakrabarty for providing the dataset through Kaggle.

References

GeeksforGeeks. Introduction to convolutional neural network, 2021. URL https://www.geeksforgeeks.org/machine-learning/introduction-convolution-neural-network/.

Krizhevsky, A., Nair, V., and Hinton, G. Cifar-10 and cifar-100 datasets, 2009. URL https://www.cs.toronto.edu/~kriz/cifar.html.

LeCun, Y., Cortes, C., and Burges, C. J. Mnist handwritten digit database, 1998. URL http://yann.lecun.com/exdb/mnist/.

Navoneel. Brain mri images for brain tumor detection, 2020. URL https://www.kaggle.com/datasets/navoneel/brain-mri-images-for-brain-tumor-detection/data.

Optuna Contributors. Optuna: A next-generation hyperparameter optimization framework, 2019. URL https://optuna.org/.

Rainey, S., Martin, S., Christen, A., Mégevand, P., and Fourneret, E. Brain recording, mind-reading, and neurotechnology: Ethical issues from consumer devices to brain-based speech decoding. *Science and engineering ethics*, 26(4):2295–2311, 2020.

Shervine, A. Cs230 cheatsheet: Convolutional neural networks, 2020. URL https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

TensorFlow Developers. Tensorflow and keras documentation, 2025. URL https://www.tensorflow.org/.

TheDeepHub. Convolutional neural networks: A comprehensive guide, 2019. URL https://medium.com/thedeephub/convolutional-neural-networks-a-comprehensive-guide-5cc0b5eae175.