Brain Tumor Detection

Authors: Anna Aldrin and Madeline Follosco

Abstract

This report outlines supervised learning methods to detect brain tumors from MRI/CT medical imaging data. Logistic Regression, Random Forest, and YOLOv11 were used for binary classification (tumor vs no tumor) while CNNs and GradCAM were used to identify the most influential features in such a classification. The project pipeline includes the following: preprocessing, feature engineering, hyperparameter tuning, and comprehensive evaluation using quantitative and qualitative metrics. The study evaluates the advantages and constraints of each approach and discusses their implications for real-world application and future research.

1. Introduction

Brain tumors are dangerous and can cause serious health issues when left untreated and undiagnosed. Integrating machine learning can assist physicians in making timely diagnoses. AI assistance in early intervention and personalized treatment plans can truly make a difference in a patient's life. In this study we aim to identify the best models and features of an image that can provide the best results for a correct diagnosis.

2. Background/Related Work:

Prior work with medical imaging has shown that deep learning models like CNN's perform better with regards to simpler statistical models such as logistic regression and SVMs due to transfer learning capabilities. While such models have shown to be useful, strategic preprocessing especially, image segmentation and region of interest extraction, is important to implement for best results. Our work explores the comparison of traditional classifiers vs more complex deep learning, focusing on the effects of preprocessing on classification outcomes of a small detection-style dataset. Using a visualization tool such as batch Grad-CAM, we could compare performance across different CNN's, ranging from shallow to deep layers, allowing us to see which models perform best across our data.

3. Methods and Implementation

3.1 Dataset Description

We use a dataset from Kaggle containing 1,116 samples of MRI and CT scan images with corresponding labels. The dataset was modified to be compatible with YOLO11 object detection models and its labels contain information about bounding boxes. The dataset was presplit into a training set, consisting of 893 images, and a testing set, containing 223 images.

3.2 Binary image classification

Data Preprocessing

- Loading and Splitting: Brain tumor dataset with pre-split train/test folders. Each image paired with YOLO .txt label file (0=no tumor, 1=tumor). Images loaded as grayscale, resized to 224×224 pixels.
- Image Preprocessing: Applied Gaussian blur for denoising, normalized pixel values to [0,1] range, enhanced contrast using CLAHE (Contrast Limited Adaptive Histogram Equalization).

- Feature Extraction - Statistical: Computed first-order statistics from intensity histograms: mean, standard deviation, median, percentiles (25th, 75th), entropy, and energy.
- Feature Extraction - GLCM (Texture): Extracted Gray-Level Co-occurrence Matrix features at multiple scales (distances 1, 2, 3) and angles (0°, 45°, 90°, 135°). Computed contrast, dissimilarity, homogeneity, energy, and correlation at each scale.
- Feature Extraction - LBP (Local Patterns): Computed Local Binary Patterns at three scales (radius 1, 2, 3) to capture multi-resolution texture patterns. Extracted top 5 histogram bins per scale as features.
- Feature Extraction - Edge Features: Applied Sobel operators for edge detection. Computed edge magnitude statistics (mean, std, max) and edge density (ratio of edge pixels to total pixels).
- Feature Extraction - Interactions: Created interaction features via outer product of GLCM and LBP vectors to capture feature combinations. Applied log transformations to GLCM features for non-linear relationships. Total: ~70-80 features per image.
- Normalization: All extracted features standardized using z-score standardization (StandardScaler), fit only on training data and applied to both train and test sets.

Experimental Setup

Models tested:

- Logistic Regression
- Random Forest
- YOLOv11
- SimpleCNN
- ResNet18

K fold cross validation and grid search was used to tune hyperparameters across 5 folds. All code was implemented in Python using scikit-learn 1.4.2. Note that all feature extraction applied to Logistic Regression and Random Forest. YOLOv11 was used with its pretrained weights and no fine tuning to the MRI data was done.

3.3 CNN

Simple CNN and ResNet18

- Loading and Splitting: Brain tumor dataset with pre-split train/test folders. Each image paired with YOLO .txt label file (0=no tumor, 1=tumor). Images loaded as grayscale, resized to 224×224 pixels.
- Image Processing: Converted images to tensors using PyTorch
- Layers- the SimpleCNN was built to use 3 layers, and the ResNet was pretrained with 18 layers

Hyperparameters were optimized via grid search and validation set performance. All code was implemented in Python using scikit-learn 1.0 and XGBoost 1.6.

4. Experiments and Results

Binary Image Classification

For the Logistic Regression and Random Forest the following metrics were collected: accuracy (train and test), ROC-AUC, precision, recall, f1 score, and loss (cross entropy loss).

Model	**Train Acc**	**Test Acc**	**ROC-AUC**	**Precision**	**Recall**	**F1**	**CE Loss**
Logistic Regression	0.7164	0.6816	0.7014	0.5658	0.5309	0.5478	11.4758
Random Forest	0.6777	0.6233	0.6173	0.4815	0.4815	0.4815	13.5770

Table 1: Model performance on test data

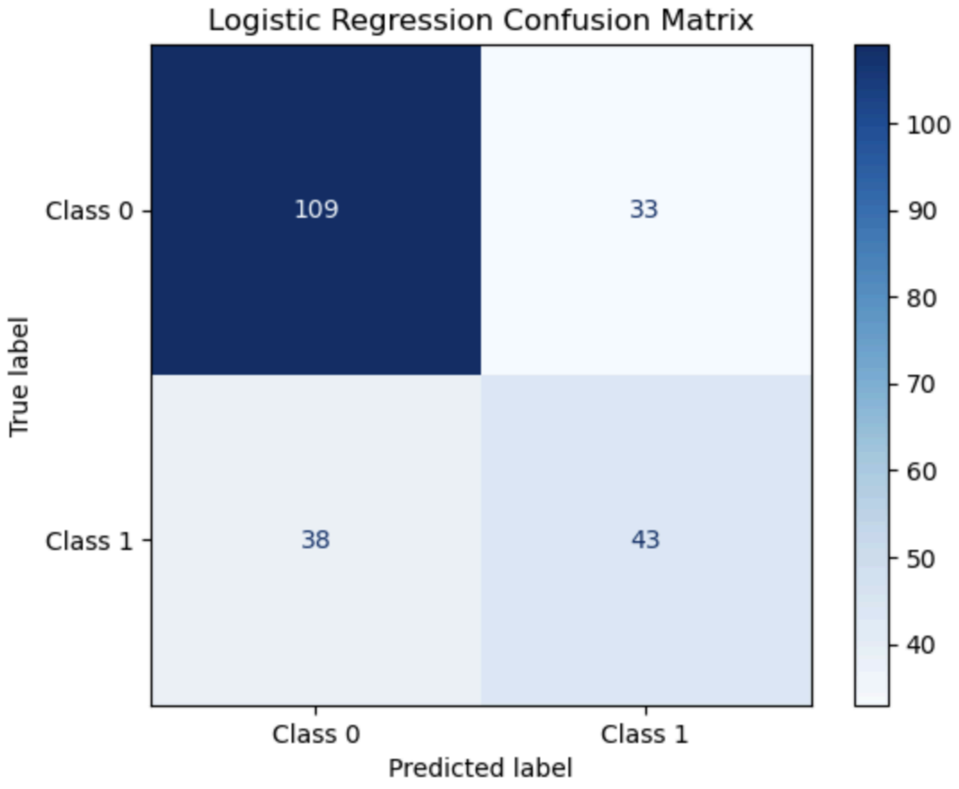


Figure 2: Confusion matrix from Logistic Regression results

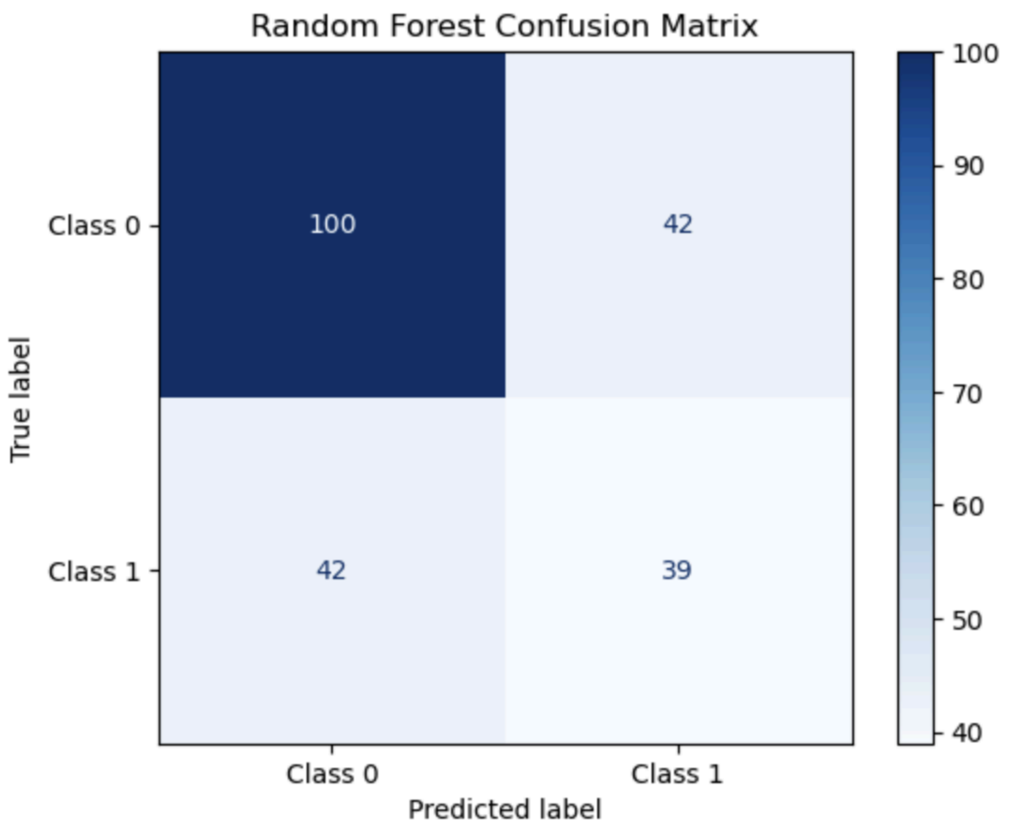
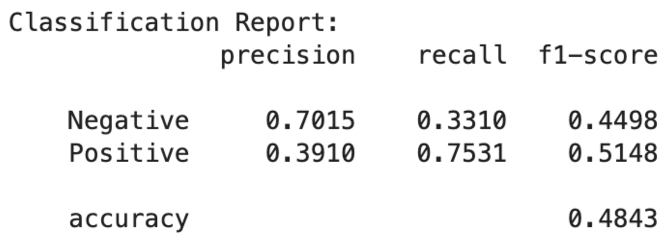


Figure 3: Confusion matrix from Logistic Regression results

Between these two models it was clear that logistic regression outperformed random forest, only minimally however. Both models had fairly similar numbers of false positives and negatives. This minimal performance discrepancy may be due to the small size of the dataset, for which logistic regression tends to do better. These misclassifications, low accuracy, and high loss could be attributed to limits in data preprocessing that will be addressed later. However, both models did not have large train and test accuracy gaps suggesting that overfitting was not a problem for these particular models.

For YOLOv11 the following metrics were calculated using the model's confidence threshold of 0.7: accuracy, precision, recall, f1 score.



```
Classification Report:
              precision    recall  f1-score

    Negative     0.7015    0.3310    0.4498
    Positive     0.3910    0.7531    0.5148

    accuracy                         0.4843
```

Table 2: Metrics for YOLOv11

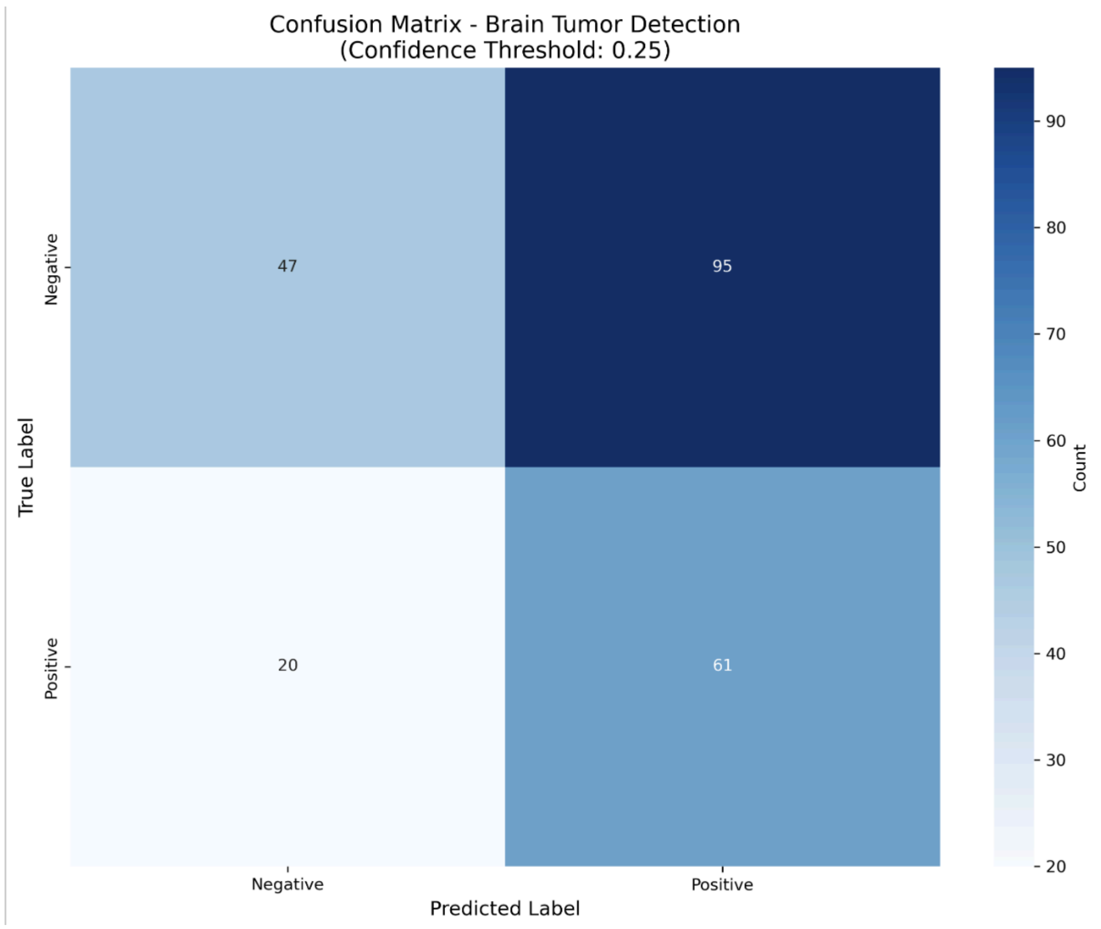


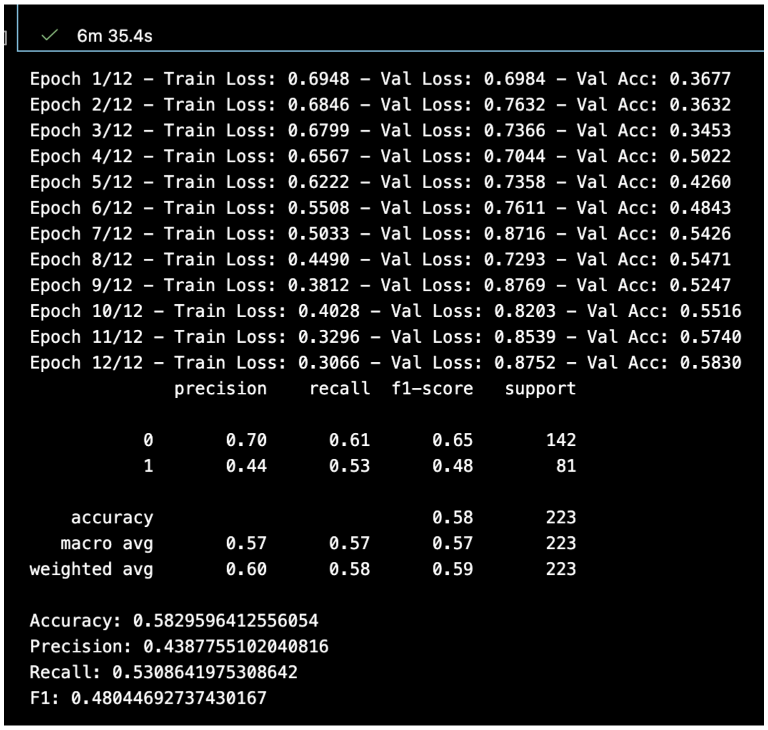
Figure 4: Confusion matrix for YOLOv11

While accuracy is not promising with this model, the high positive recall suggests that it is good at detecting the presence of a tumor, which makes sense given that the model is meant for object detection and not classification. The higher amount of false positives as opposed to false positives is promising in a medical setting given that a misdiagnosis of a tumor when there isn't one is preferred.

CNN

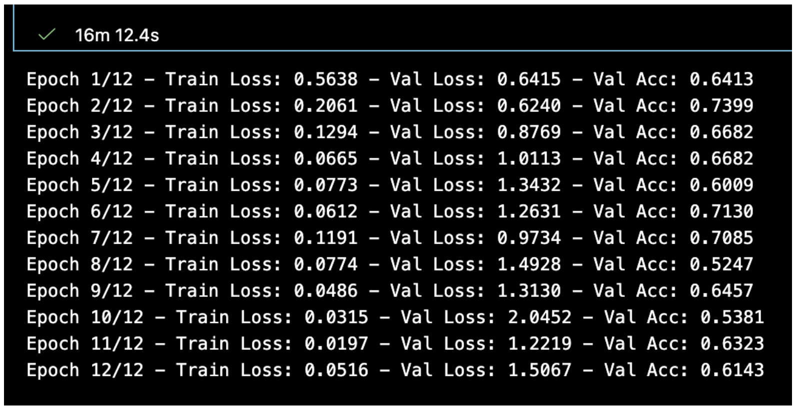
For our Simple CNN and Resnet18 models, they produced our quantitative results, including: loss, accuracy, precision, recall, and our f1 score.

CNN

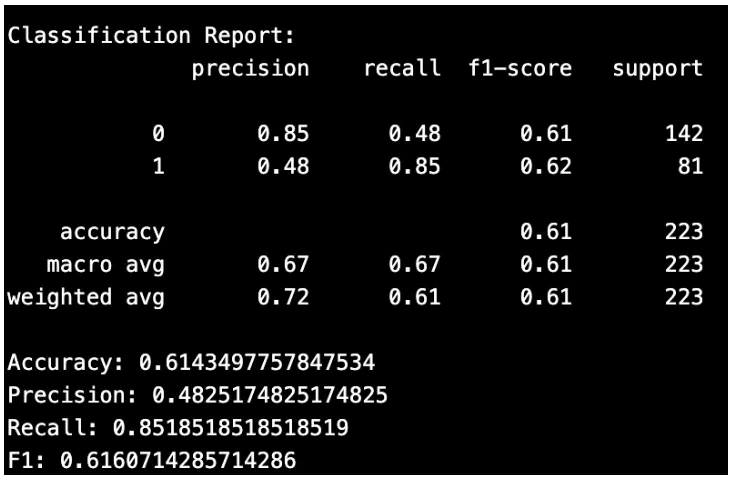


```
✓ 6m 35.4s

Epoch 1/12 - Train Loss: 0.6948 - Val Loss: 0.6984 - Val Acc: 0.3677
Epoch 2/12 - Train Loss: 0.6846 - Val Loss: 0.7632 - Val Acc: 0.3632
Epoch 3/12 - Train Loss: 0.6799 - Val Loss: 0.7366 - Val Acc: 0.3453
Epoch 4/12 - Train Loss: 0.6567 - Val Loss: 0.7044 - Val Acc: 0.5022
Epoch 5/12 - Train Loss: 0.6222 - Val Loss: 0.7358 - Val Acc: 0.4260
Epoch 6/12 - Train Loss: 0.5508 - Val Loss: 0.7611 - Val Acc: 0.4843
Epoch 7/12 - Train Loss: 0.5033 - Val Loss: 0.8716 - Val Acc: 0.5426
Epoch 8/12 - Train Loss: 0.4490 - Val Loss: 0.7293 - Val Acc: 0.5471
Epoch 9/12 - Train Loss: 0.3812 - Val Loss: 0.8769 - Val Acc: 0.5247
Epoch 10/12 - Train Loss: 0.4028 - Val Loss: 0.8203 - Val Acc: 0.5516
Epoch 11/12 - Train Loss: 0.3296 - Val Loss: 0.8539 - Val Acc: 0.5740
Epoch 12/12 - Train Loss: 0.3066 - Val Loss: 0.8752 - Val Acc: 0.5830
              precision    recall  f1-score   support

           0       0.70      0.61      0.65       142
           1       0.44      0.53      0.48        81

    accuracy                           0.58       223
   macro avg       0.57      0.57      0.57       223
weighted avg       0.60      0.58      0.59       223

Accuracy: 0.5829596412556054
Precision: 0.4387755102040816
Recall: 0.5308641975308642
F1: 0.48044692737430167
```

ResNet



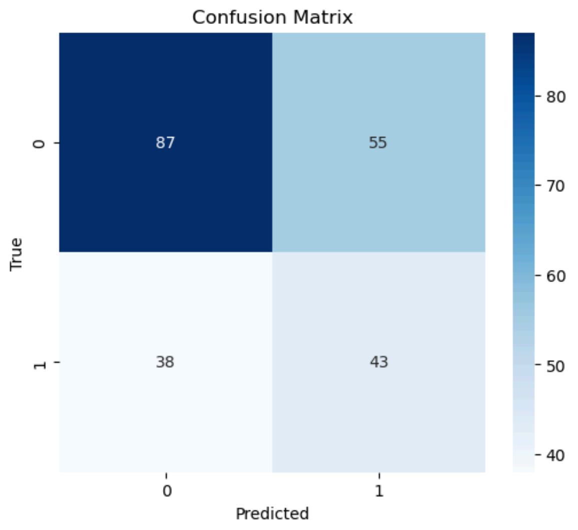
```
✓ 16m 12.4s

Epoch 1/12 - Train Loss: 0.5638 - Val Loss: 0.6415 - Val Acc: 0.6413
Epoch 2/12 - Train Loss: 0.2061 - Val Loss: 0.6240 - Val Acc: 0.7399
Epoch 3/12 - Train Loss: 0.1294 - Val Loss: 0.8769 - Val Acc: 0.6682
Epoch 4/12 - Train Loss: 0.0665 - Val Loss: 1.0113 - Val Acc: 0.6682
Epoch 5/12 - Train Loss: 0.0773 - Val Loss: 1.3432 - Val Acc: 0.6009
Epoch 6/12 - Train Loss: 0.0612 - Val Loss: 1.2631 - Val Acc: 0.7130
Epoch 7/12 - Train Loss: 0.1191 - Val Loss: 0.9734 - Val Acc: 0.7085
Epoch 8/12 - Train Loss: 0.0774 - Val Loss: 1.4928 - Val Acc: 0.5247
Epoch 9/12 - Train Loss: 0.0486 - Val Loss: 1.3130 - Val Acc: 0.6457
Epoch 10/12 - Train Loss: 0.0315 - Val Loss: 2.0452 - Val Acc: 0.5381
Epoch 11/12 - Train Loss: 0.0197 - Val Loss: 1.2219 - Val Acc: 0.6323
Epoch 12/12 - Train Loss: 0.0516 - Val Loss: 1.5067 - Val Acc: 0.6143
```



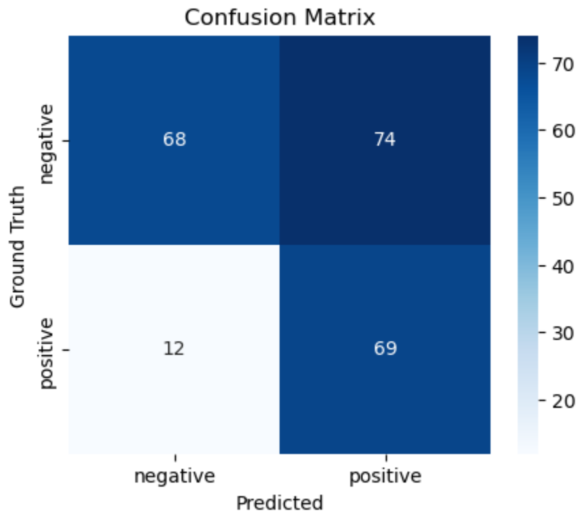
```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.48      0.61       142
           1       0.48      0.85      0.62        81

    accuracy                           0.61       223
   macro avg       0.67      0.67      0.61       223
weighted avg       0.72      0.61      0.61       223

Accuracy: 0.6143497757847534
Precision: 0.4825174825174825
Recall: 0.8518518518518519
F1: 0.6160714285714286
```

Based on these metrics, we generated the following confusion matrices.

CNN

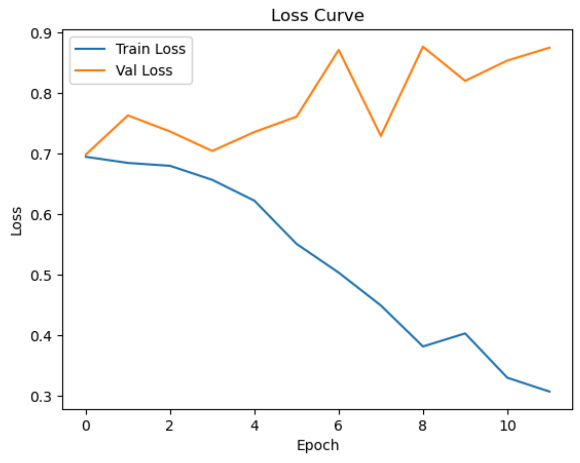


ResNet

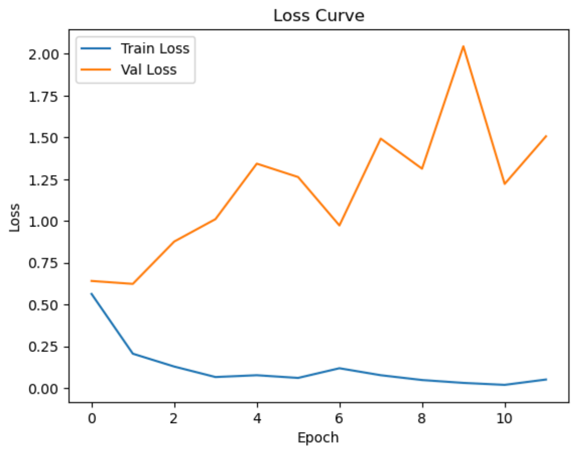


Our simple CNN seemed to provide many true values, whereas the ResNet produced many true values but also many false positives, indicating that our model was prone to Type I errors, thus overfitting our data. This was also seen through our loss plots as our train loss began to increase over time, also indicating that there was possible overfitting from the model and that it memorized the noise from the data.

CNN

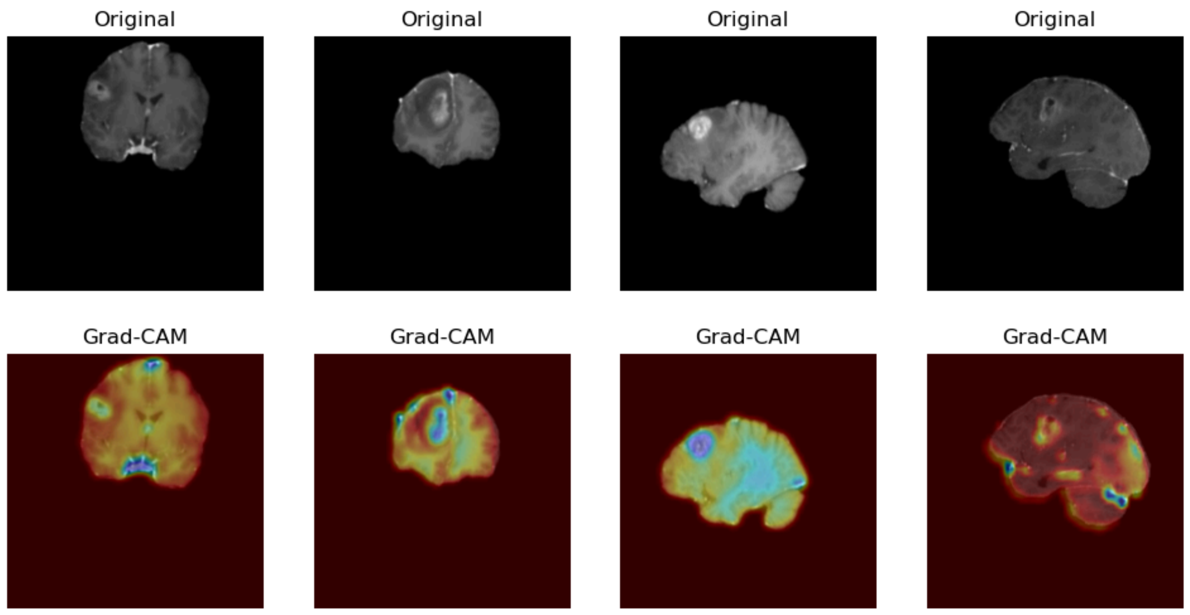


ResNet

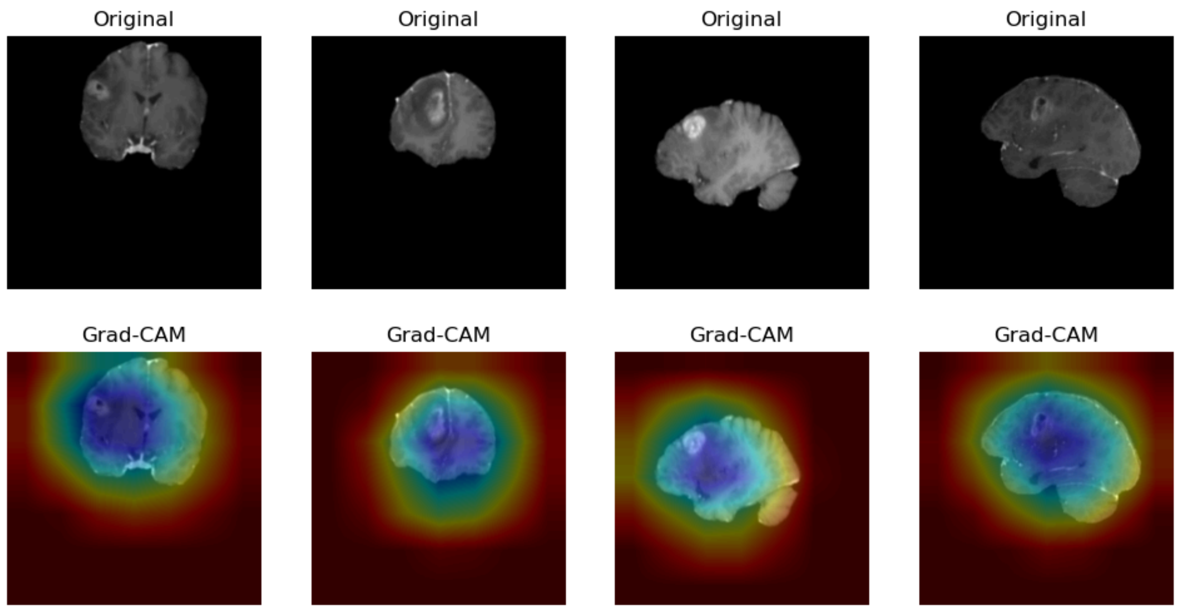


Lastly, for our qualitative results, we were able to see how the model performed in detecting the tumor from the given data and could see just how much of the heat maps were blue and green in our Grad-CAM implementation.

CNN



ResNet



In our code for the SimpleCNN and ResNet18 models, we also generated the first 9 misclassified images from the dataset for a better understanding of the Type I errors being found from our models. This produced images of our false positives and false negatives, confirming the models' tendencies to overfit.

Discussion

What worked, what didn't, model comparison, recommendations, error analysis

Binary Image Classification

Both Logistic Regression and Random Forest struggled to generalize while YOLO struggled to classify, but did well with tumor detection. Related works that found success with more simple classifiers highlighted the importance of the preprocessing phase. Region of interest extraction and segmentation were crucial in isolating relevant features. Given the YOLO model was successful in detecting tumor presence, this should be made one of the first layers in the pipeline. The model can be used to highlight regions of interest with its bounding boxes. By restricting analysis to these regions, simpler classifiers are less influenced by background noise and irrelevant image content, enabling more accurate classification.

CNN

Both models did well to classify the tumors in terms of visualization, however, both CNN models struggled with stronger accuracy and avoiding overfitting due to type 1 errors.
From our heatmaps, it was evident that the SimpleCNN could pick up on the smaller details and hone in on the areas where it looked like a tumor was present. And for the ResNet, there were much larger spots of areas where it indicated a tumor could be present. After research and analysis, we could see that this was because the SimpleCNN model was much shallower with fewer layers, allowing it to pick up on prominent features in images and hone in on the differences. Since the ResNet had more layers, it focused more on representing the distribution of the data to help with better generalization– the models differed between focusing on isolated features versus more of the global context.
Even though both worked with different amounts of layers, each struggled in different areas. Where ResNet18 may have had a higher accuracy, SimpleCNN had higher false positives. Where ResNet had more layers for extraction to focus on generalization, SimpleCNN had shallower layers, but it allowed it to focus on prominent and distinct features. Both had their pros and cons, but as models that used layers and residual networks to prioritize efficiency and cost, they performed as well as could be expected.

Future Work

Binary Classification

Due to the analysis in the previous section, the next approach would be to combine all model strengths by moving YOLO earlier in the preprocessing stage of the pipeline. This, in addition to fine tuning the YOLO model may provide more successful results.

CNN

To improve our accuracy for the SimpleCNN and Resnet 18 models, it would have been interesting and most likely helpful to regularize our data using a method such as ridge or lasso regression to see if this would have helped increase the accuracy for each model. Especially with the number of epochs used to train the models, combined with the shallow layers, increasing the number of layers and/or our number of epochs may have helped to increase the accuracy as well. Such extensions may have provided better insight into which models were best representative and truthful to our data despite not needing so many layers to model and classify the data. Using regression to regularize our data with these different models, compared to logistic regression and random forest, prove a viable path to help train our models efficiently and correctly.

References

1. Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M. A., Al-Amidie, M., & Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. Journal of Big Data, 8(1), Article 53. https://doi.org/10.1186/s40537-021-00444-8
2. Awad, F. H., Hamad, M. M., & Alzubaidi, L. (2023). Robust classification and detection of big medical data using advanced parallel K-means clustering, YOLOv4, and logistic regression. Life, 13(3), Article 691. https://doi.org/10.3390/life13030691
3. Disci, R., Gurcan, F., & Soylu, A. (2025). Advanced brain tumor classification in MR images using transfer learning and pre-trained deep CNN models. Cancers, 17(1), Article 121. https://doi.org/10.3390/cancers17010121
4. Fonseca, L. (2024). A practical comparison between CNN and ResNet architectures: A focus on attention mechanisms. Medium. https://medium.com/@leonardofonseca.r/a-practical-comparison-between-cnn-and-resnet-architectures-a-focus-on-attention-mechanisms-cee7ec8eca55
5. GeeksforGeeks. (n.d.). Residual networks (ResNet) in deep learning. https://www.geeksforgeeks.org/deep-learning/residual-networks-resnet-deep-learning/
6. Hewlett Packard Enterprise. (n.d.). What is a convolutional neural network (CNN)? https://www.hpe.com/us/en/what-is/convolutional-neural-network.html
7. Llourenc. (2022, March 18). The importance of logistic regression in image classification. Medium. https://medium.com/@MudSnail/the-importance-of-logistic-regression-in-image-classification-1966d07e7a0c
8. MathWorks. (n.d.). Grad-CAM reveals the why behind deep learning decisions. https://www.mathworks.com/help/deeplearning/ug/gradcam-explains-why.html
9. Naseer, A., Yasir, T., Azhar, A., Shakeel, T., & Zafar, K. (2021). Computer-aided brain tumor diagnosis: Performance evaluation of deep learner CNN using augmented brain MRI. International Journal of Biomedical Imaging, 2021, Article 5513500. https://doi.org/10.1155/2021/5513500
10. Palaniappan, D., et al. (2025). YOLO in healthcare: A comprehensive review of detection architectures, domain applications, and future innovations. IEEE Access, 13, 145714–145735. https://doi.org/10.1109/ACCESS.2025.3599358
11. Srivaishnavi, K. R., Pramananda Perumal, T., & Anishiya, P. (2025). Feature extraction from segmented brain MRI images using traditional methods and classification analysis of brain tumor using machine learning classifiers. Indian Journal of Science and Technology, 18(17), 1309–1320. https://doi.org/10.17485/IJST/v18i17.1902

Other Acknowledgements

AI Tools

- Google Gemini - Explanation of conceptual information
- Claude - Helpful for feature extraction with Graycom and interpretation of YOLO results
- Open AI ChatGPT - Python and tool specific information

Dataset

- Jocher, G., Munawar, M. R., Ultralytics. “Brain Tumor Dataset.” Kaggle, 2023, https://www.kaggle.com/datasets/ultralytics/brain-tumor/data. Accessed 8 Dec. 2025.

Primary Libraries:

- sklearn

- Numpy
- matplotlib
- utralytics