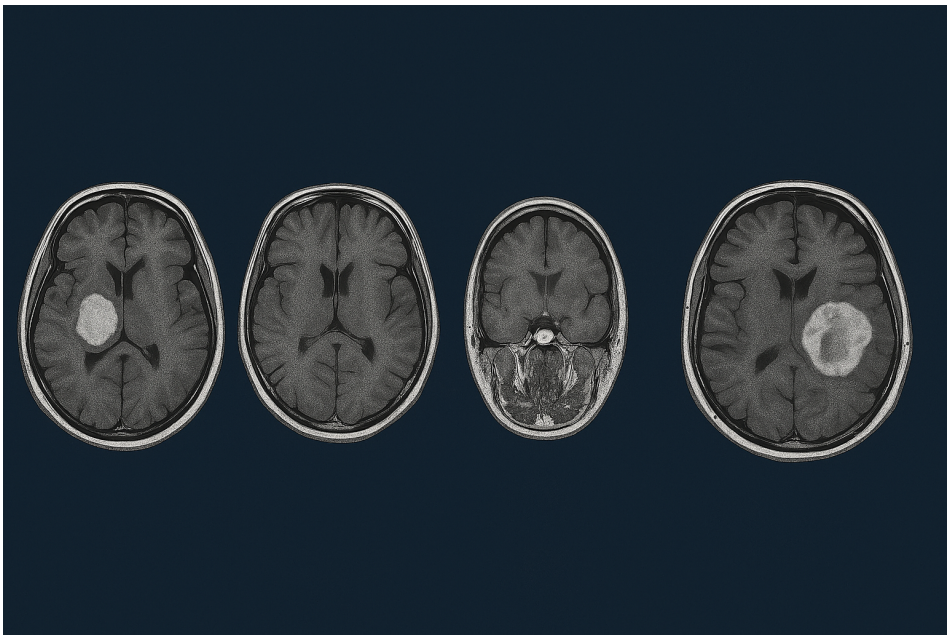
MRI Brain Tumor Image Multi-Class Classification

Vinith Kuruppu, Eduardo Jose Villasenor, Richard Yan, Jacqueline Yeung

August 8, 2025



Dataset: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/data
Github: https://github.com/Richard-Yan-UCB/datasci281_final_project

1 Introduction

Magnetic Resonance Imaging (MRI) is a critical tool in the early detection and diagnosis of brain tumors. While deep learning models have achieved state-of-the-art performance, they often demand large labeled datasets and extensive computational resources. In this work, we propose a hybrid computer vision pipeline that combines handcrafted and deep features for multi-class classification of 2D brain MRI images into four categories: no tumor, glioma, meningioma, and pituitary tumor. We extract edge and texture information using Canny Edge detection, tumor location and size information using Difference of Gaussian (DoG) and Determinant of Hessian (DoH), and high-level semantic representations from the penultimate layer of the pretrained DINOv2 Vision Transformer. We concatenate these features into different combinations, standardize them, and apply dimensionality reduction. We then train a diverse set of classical classifiers, including logistic regression, support vector machines, random forests, linear discriminant analysis, and quadratic discriminant analysis. We also evaluate an ensemble model combining the predictions of these classifiers as an attempt to address overfitting. Our results show that a logistic regression model paired with a complex feature extracted from a pre-trained model takes advantage of both low and high-level visual cues to achieve strong performance with an accuracy of 96.7% and does not require the overhead of end-to-end deep learning training, making it suitable for deployment in resource-constrained or low-data clinical settings.

2 Data

2.1 Dataset

The Brain Tumor MRI dataset is a curated collection of MRI images categorized into four classes: glioma, meningioma, pituitary, and no tumor.

- Glioma Tumor: Gliomas are aggressive brain tumors that often appear irregular in shape on MRI and may blend into surrounding brain tissue. They usually occur in the frontal or temporal lobes of the brain.
- Meningioma Tumor: Meningiomas originate from the meninges. They are typically well-defined, located near the outer surface of the brain, and show strong contrast enhancement on MRI.
- Pituitary Tumor: Pituitary tumors develop in the pituitary gland, a small hormone-producing gland located at the base of the brain, behind the eyes. On MRI, they appear in this central region, called the sellar region, and often cause noticeable structural changes.
- No Tumor (Healthy Brain): This class represents normal brain MRIs with no tumor present. Accurate classification helps reduce false positives.

The data is already pre-divided into training and test sets. The training set contains 5712 images with the following labeled class distribution (see Figure 2): glioma (1321), meningioma (1339), pituitary (1457), and no tumor (1595). The test set contains 1311 images, forming an approximate 80/20 split. The images exhibit differences in orientation (axial, coronal, and sagittal views), sequence type [1], contrast/brightness intensities, image size/resolution, and number of color channels (see Figure 1). Some images have additional noise factors such as grayscale inversion (GSI), text/watermarks, and aliasing due to JPEG compression, prompting the need for pre-processing to ensure consistent feature extraction (see Figure 3).

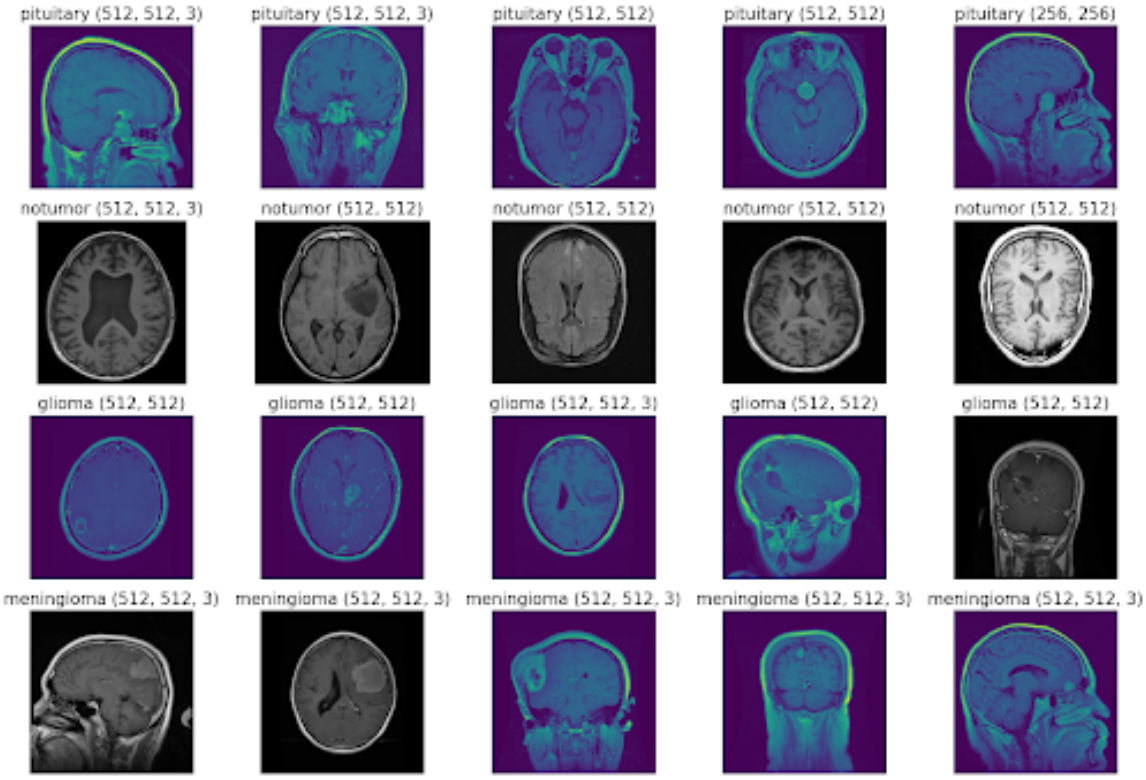


Figure 1: Example Unprocessed MRI Images

[1] MRI sequence types refer to the set of radiofrequency pulses and gradients that are used to generate the MRI images and influence how different tissues appear in the final scan. See https://radiopaedia.org/articles/mri-sequences-overview.

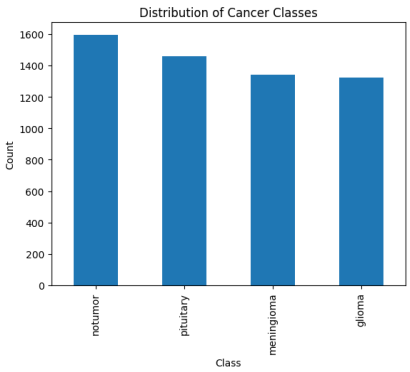


Figure 2: Class Distribution

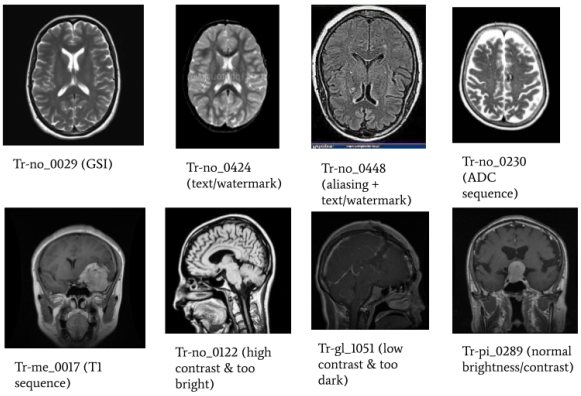


Figure 3: Example MRI Image Noise Factors

2.2 Image Pre-Processing

Before performing any feature extraction, an image pre-processing pipeline was run to standardize the images. The pipeline consisted of the following steps: convert image to grayscale, normalize pixel intensity to [0,1] range, and resize/rescale and center-crop images to the median training dimensions, 512 x 512 (see Figure 4).

Figure 4: Image pre-processing steps before feature extraction

3 Features

3.1 Canny Edges

Canny edge detection works by identifying points in an image where the brightness changes sharply. These points often correspond to the boundaries of objects or structural features. This process includes multiple steps such as noise reduction and gradient calculation. Canny edge detection allows important structures to stand out, regardless of the previously mentioned variations in orientation, contrast, or grayscale across the images. Using Canny edge detection, pixel-to-pixel contrast differences were emphasized, which helped unify the representation of both standard grayscale and inverted grayscale images. By focusing on these edges instead of the raw pixel values, the extracted features were cleaner and more consistent, as seen in Figure 5. The resulting binary edge maps (bottom row) showcase the most significant structural outlines in white and are then overlaid against the original MRI scans on the top row. As seen in the t-SNE visualization, Figure 6, most of the samples group closely with their respective class, indicating that the edge-based characteristics captured meaningful structural differences.

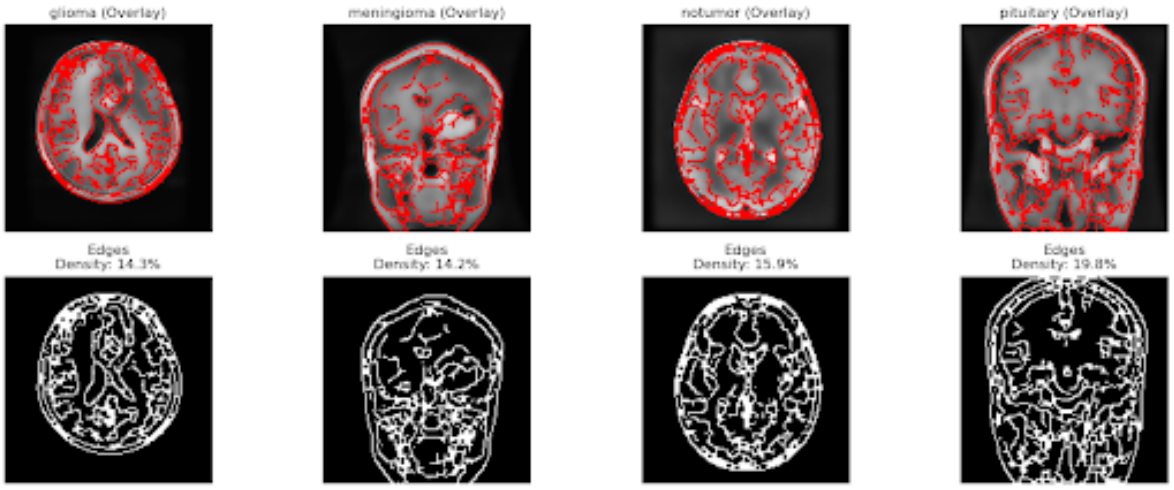
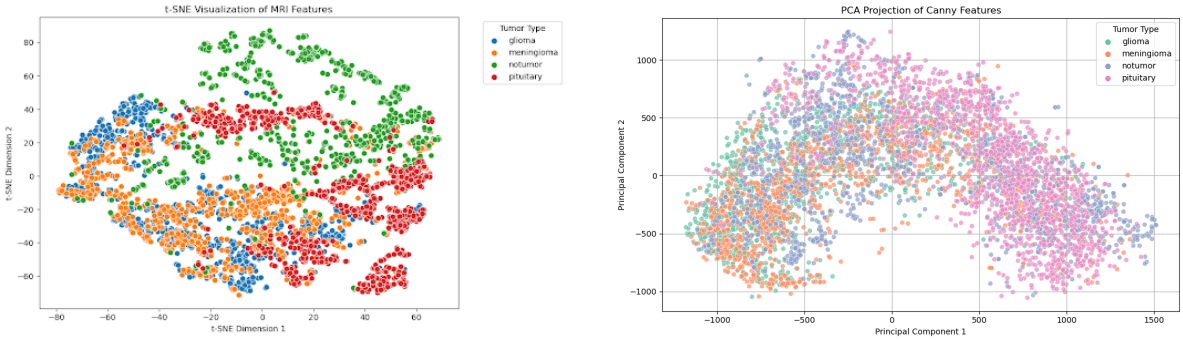


Figure 5: Canny Edge Results



Canny Edge t-SNE Plot

PCA Projection of Canny Features

Figure 6: t-SNE and PCA projections of Canny Edge features.

3.2 Blob Detection

We explored blob detection features to capture additional information about the visible tumors, such as their existence, location, and size. We implemented two features: Difference of Gaussian (DoG) and Determinant of Hessian (DoH). Before we generated these features, we applied additional pre-processing steps to the standardized images:

1. CLAHE (contrast limited adaptive histogram equalization): CLAHE is a histogram equalization [2] method that works by equalizing contrast in small local tiles, rather than globally. If any histogram bin exceeds a certain contrast limit, the pixels are clipped and distributed uniformly to other bins before applying histogram equalization. We chose this method to avoid information loss due to over-brightening/darkening that could potentially occur with global equalization.

2. Skull stripping: we applied vignettes to the images to de-emphasize the skull structures. We used steeper fall-offs for images we deemed too bright and shallower fall-offs for images we deemed too dark. We determined these categories by visually handpicking a set of too-bright and too-dark images and using the common mean pixel intensities observed in both groups as indicators. Appendix Figure 21 shows plots for a few examples.

We applied these pre-processing steps to both types of blob features using different hyperparameters and then performed the blob detection using these post-processed images.

[2]Histogram equalization is a pre-processing technique often used to improve the contrast of an image by taking the image's cumulative pixel intensity histogram and redistributing intensities to the full range.

3.2.1 Difference of Gaussian (DoG)

DoG works by taking the difference of two Gaussian-blurred versions of the image with increasing standard deviations (sigma), emphasizing the high-frequency signals in the image. Differences are taken for a range of sigmas and stacked in a cube. Blobs are identified as local maximas in this cube. We further tuned the blob detection granularity using a pixel intensity threshold. DoG is designed to detect bright blobs on dark backgrounds. We found DoG to do relatively well at identifying blobs more broadly than Laplacian of Gaussian, an alternative blob detection method which DoG is also a faster approximation for. We calculated DoG for blobs at different scales, specifying different min and max sigma values for "large" vs. "small" blobs. Figure 7 shows DoG blob visualizations for a pituitary sample.

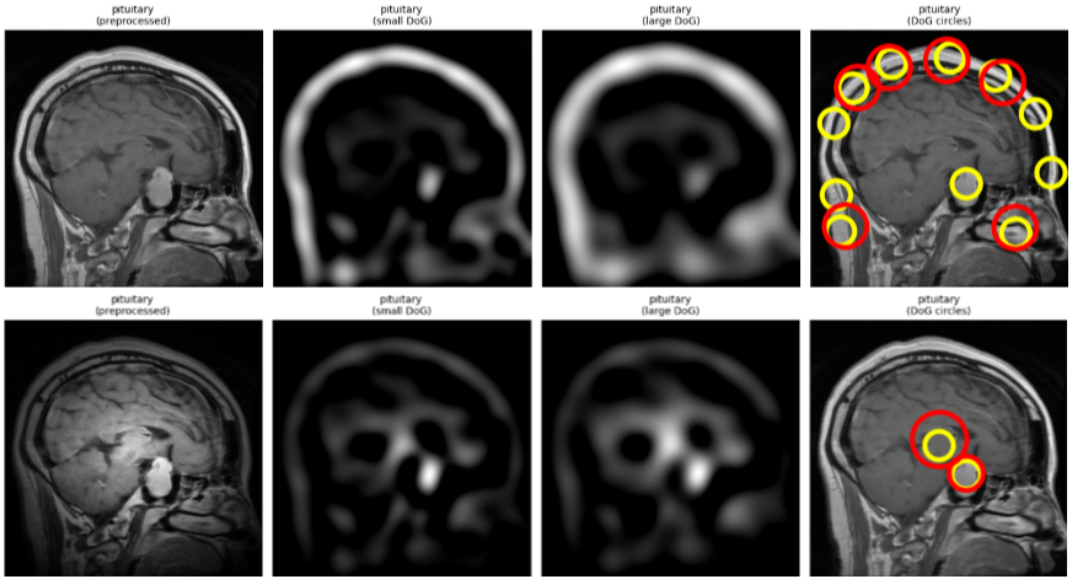


Figure 7: Difference of Gaussian (Pituitary): Each row shows the pre-processed image, followed by the "small" DoG raster, "large" DoG raster, and final detected blobs. The 2 rows represent before and after the additional pre-processing was applied. Red circles indicate "large" blob detection. Yellow circles indicate "small" blob detection.

3.2.2 Determinant of Hessian (DoH)

DoH works by calculating the matrix of the Determinant of Hessian, using Gaussian kernels with a range of standard deviations (sigma). The Hessian matrix, calculated via second-order partial derivatives, represents the local curvature at each point in the image. Blobs are identified as the local maximas in the Hessian matrix and are calculated using the determinant. We chose DoH to account for the limitation of DoG, which works best for bright on dark blobs. DoH is able to detect dark on bright blobs, which we observed to be the case for the glioma class. As a side effect, DoH may capture other brain structures such as ventricles or cavities, but we left it to the machine learning models to learn these patterns. Like DoG, we tuned the blob detection granularity using a pixel intensity threshold. Figure 8 shows DoH blob visualizations for the same pituitary sample shown in Figure 7.

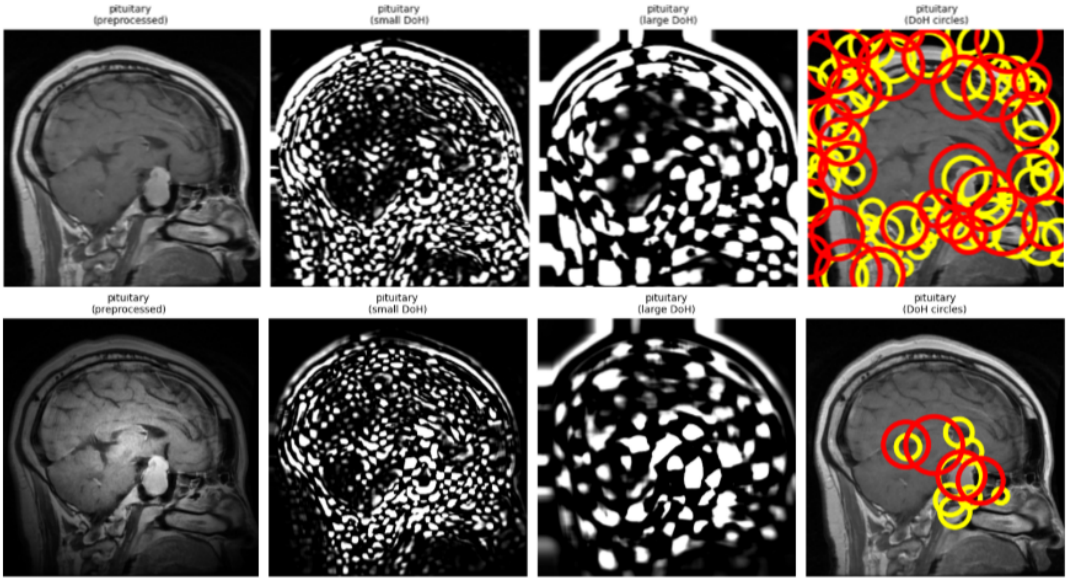


Figure 8: Determinant of Hessian (Pituitary): Each row shows the pre-processed image, followed by the "small" DoH raster, "large" DoH raster, and final detected blobs. The 2 rows represent before and after the additional pre-processing was applied. Red circles indicate "large" blob detection. Yellow circles indicate "small" blob detection.

The outputs of the blob detection are a collection of 3-length lists containing the y coordinate, x coordinate, and radius r of each blob. To construct the final feature vectors, we calculated 14 blob statistics and concatenated them together in the following format: [# blobs, blob density [3], average y, standard deviation of y, minimum y, maximum y, average x, standard deviation of x, minimum x, maximum x, average r, standard deviation of r, minimum r, maximum r]. We originally considered concatenating the raw 3-length lists for each image, but 1) this approach would treat blobs at potentially different locations across images with similar significance, and 2) ensuring all feature vectors were the same length was complicated. Hence, we chose the statistics approach. Figure 9 shows the PCA explained variance plot of the DoG and DoH features. Although the principal components don't explain much more of the total variance (i.e. some of the statistics are redundant), we decided that the feature vectors were small enough to forgo performing additional dimensionality reduction.

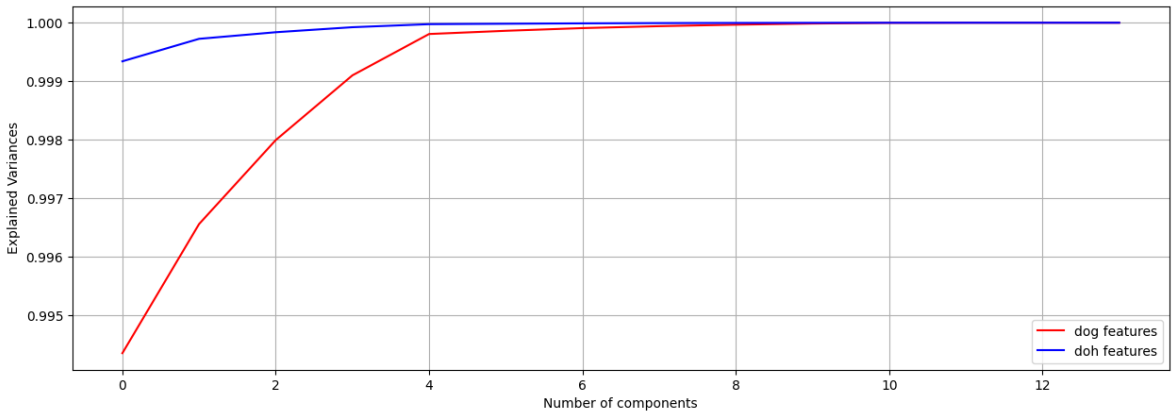
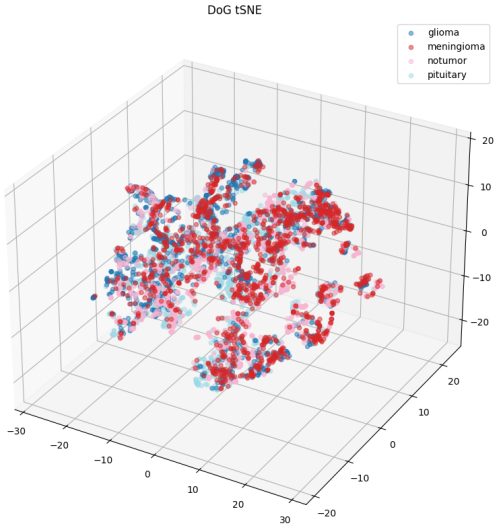


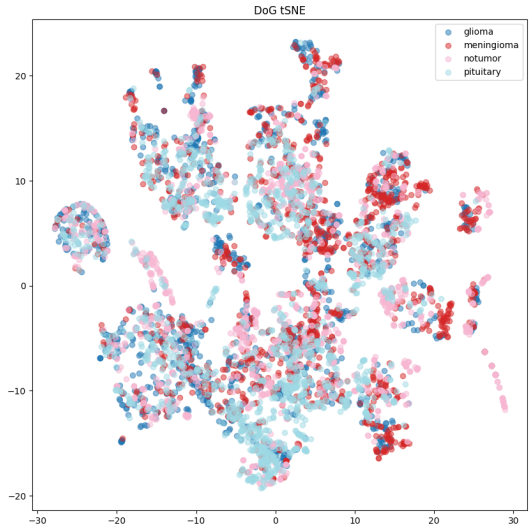
Figure 9: DoG and DoH PCA Explained Variance Plot

[3]calculated as $\#blobs/(height \times width)$

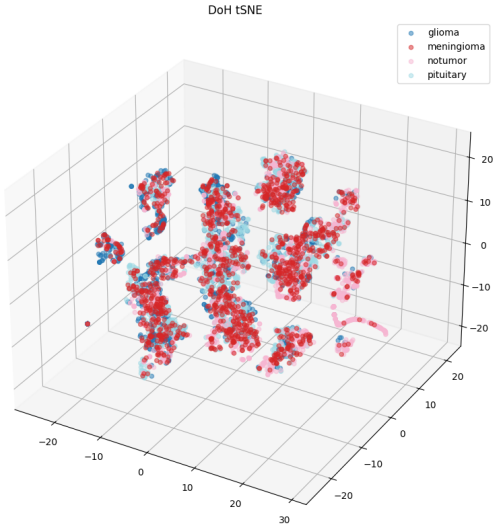
In terms of separability, the 3D and 2D t-SNE plots displayed in Figure 10 show great overlap for both feature types, but also subtle signs of separation. In particular, DoG seems to slightly differentiate pituitary/glioma with meningioma but may not differentiate between pituitary and glioma well. DoH seems to differentiate pituitary, glioma/meningioma, and notumor but may not differentiate between glioma and meningioma well.



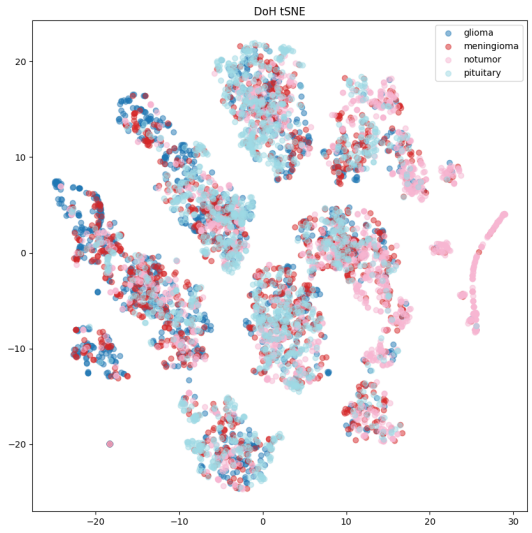
DoG 3D t-SNE Plot



DoG 2D t-SNE Plot



DoH 3D t-SNE Plot



DoH 2D t-SNE Plot

Figure 10: For DoG, pituitary and glioma appear to be pulled left, whereas meningioma is pulled right. For DoH, pituitary appears to be pulled to the left middle, glioma is pulled left, and no tumor is pulled right.

3.3 Complex Feature

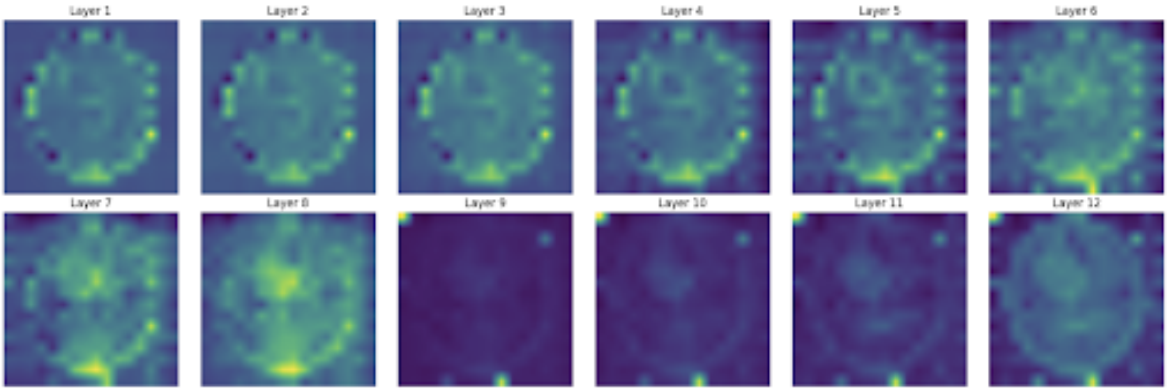


Figure 11: Layer-wise Activation Maps for a Glioma Sample: Visualization of intermediate feature activations across the 12 transformer layers of the DINOv2 model for a single glioma MRI slice. Layers 1–6 highlight low-level anatomical features such as edges and contrast boundaries. Mid-level layers (7–8) begin to localize more complex structures, including the tumor core. However, activations in the deeper layers (9–12) appear attenuated and spatially diffuse, suggesting a transition to more abstract, global representations with reduced spatial resolution. This trend is consistent with the transformer's hierarchical feature encoding behavior, where later layers prioritize semantic abstraction over local detail.

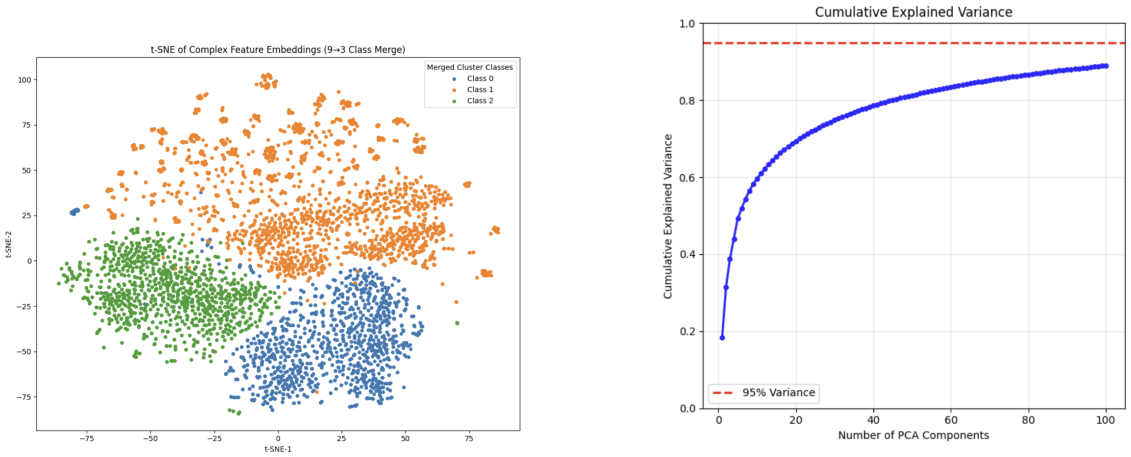
To address the challenge of tumor classification in the brain scans, we first confronted a critical confound: our dataset lacked imaging plane annotations (e.g., axial, sagittal, coronal), despite the fact that tumor appearance, shape, and anatomical context vary significantly across orientations. We hypothesized that explicitly modeling plane-specific structure could serve as a strong inductive bias for the downstream classification task.

As an initial exploratory step, we sought to determine whether image plane structure could be recovered from the raw data in an unsupervised fashion. We began by applying PCA followed by t-SNE to the flattened pixel values of the images. The resulting projection exhibited minimal separation across views, suggesting that low-level pixel statistics were insufficient to capture global anatomical structure. Repeating the same procedure with feature embeddings from MobileNetV2, a lightweight convolutional architecture pretrained on ImageNet, led to marginal improvement in separability but still failed to recover consistent clusters corresponding to imaging planes (see Appendix Figure 20 for a comparison of raw and CNN feature spaces).

In contrast, features extracted from the penultimate layer of DINOv2, a self-supervised Vision Transformer trained with the DINo-style distillation objective on large-scale natural image corpora, yielded substantially more structured embeddings. Without any supervision, the t-SNE projections of DINOv2 features revealed distinct clusters aligned with imaging planes. Notably, images exhibited three separable subclusters, which we empirically confirmed to correspond to different depth slices within the same orientation. Increasing the number of clusters from three (corresponding to the expected axial, sagittal, and coronal views) to nine - capturing three distinct depth levels within each view, yielded markedly cleaner separability in the t-SNE space. This refined clustering pattern revealed that DINOv2 embeddings not only captured view-level distinctions but also preserved fine-grained spatial variation across depth slices. This emergent structure, recovered without any explicit supervision, indicates that DINOv2 encodes a rich hierarchy of semantic and anatomical features that are highly transferable to downstream tasks in the medical imaging domain (see Figure 12 for visualization of the DINOv2 feature space).

Motivated by these observations, we adopted DINOv2 as a frozen feature extractor. Each 2D MRI scan was passed through the network, and we extracted the CLS token from the penultimate layer - a 768-dimensional embedding designed to summarize the global content of the image. Each 2D MRI scan was passed through the network, and we extracted the CLS token from the penultimate layer, a 768-dimensional embedding designed to summarize the global content of the image. We chose to use the CLS token instead of the full sequence of patch embeddings, as it offered a compact and semantically

rich representation that preserved high-level spatial and anatomical information without requiring dimensionality reduction techniques like PCA. This compact representation was then used as input to a downstream classifier trained to predict tumor type. Our assumption is that DINOv2's ability to encode both anatomical orientation and depth, critical factors in radiological interpretation, would translate into improved generalization for tumor classification. Crucially, this modular design avoids the need for end-to-end fine-tuning on small medical datasets while leveraging the strong inductive biases learned by large-scale self-supervised transformers.



(a) t-SNE of Complex Features

(b) Cumulative Variance of Complex Features

Figure 12: Visualization of DINOv2 Feature Structure: (a) t-SNE visualization of DINOv2 [CLS] token embeddings. Clusters correspond to merged imaging planes and depth slices (3 views × 3 depths = 9 total clusters, merged into 3 superclasses). The structure reveals separability across anatomical orientation and intra-view depth, indicating that the model captures semantically meaningful features despite the absence of supervision. (b) Cumulative explained variance curve for PCA applied to DINOv2 embeddings. The first 70 principal components account for over 95% of the total variance, suggesting a compact representation of the original high-dimensional feature space.

4 Model Development

4.1 Model Training Strategy

There were a variety of classifiers available for this multi-classification task. The classifiers that we tested as part of the model experimentation include logistic regression, support vector machines (using linear and RBF kernels), random forest, and linear and quadratic discriminant analysis. Each of these models have pros and cons. The linear models, such as logistic regression, linear SVM, and LDA, are interpretable but will not work for non-linear boundaries. SVM, random forest, and QDA can capture non-linear boundaries and may be robust to class imbalance, although they are less scalable. From our previous t-SNE analysis, it's not obvious whether all of our features are linearly separable, but the complex feature shows some promise. For these reasons, we tried a variety of modeling approaches.

Initially, a full set of experiments was performed with all combinations of independent features and model types. Several feature combination experiments were also done to see if the information from multiple features concatenated together would help to further distinguish between each of the classes. For hyperparameter tuning, cross-validation using grid search was performed on all model types with the default 5 data folds. In terms of performance metrics, the accuracy percentage was chosen due to the minimal class imbalance seen among the training data. The results are displayed in Figure 13.

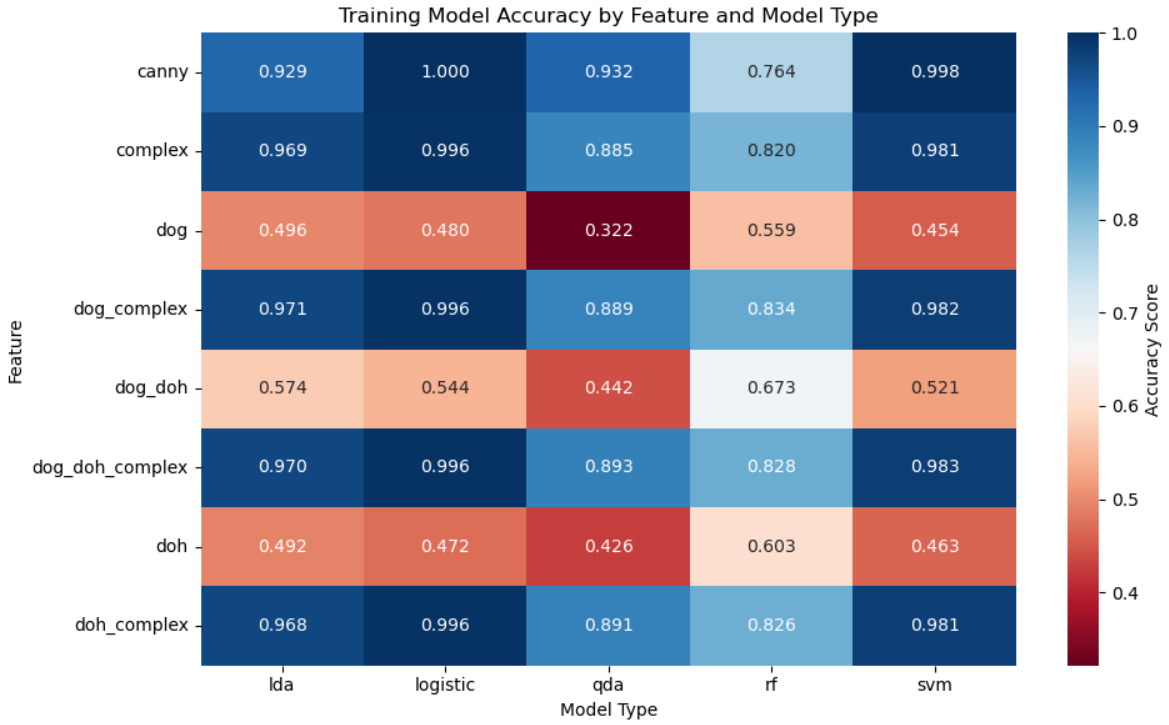


Figure 13: Training Accuracy by Feature and Model Type

4.2 Training Accuracy

Based on the experimental results, we found that the combination of Canny Edges and a logistic regression model performed the best on the training data and was able to fit a perfect model with 100% precision as shown in the confusion matrix in Figure 14. This indicates that the logistic regression model was able to learn the distinctions between all the classes within the training data even after dimensionality reduction of the Canny Edge feature vector from 262,144 to 1,000 using Principal Component Analysis (PCA). It also indicates that the Canny Edge feature space is approximately linearly separable. Linear models are simpler to implement and easier to understand than non-linear models, so this is a good sign. The primary reason for stopping at 1000 components for Canny Edges was due to training time. A longer feature vector was one of the biggest determining factors in training time. Similarly, the complex feature performed quite well in combination with linear models, indicating its linear separability. In addition, when using the complex feature in combination with DoG and DoH, there is slight improvement with some of the model types. This indicates that while the blob features are not effective alone, they are potentially effective when used in combination with the complex feature and the high-level information it captures.

In general and from the perspective of features, any model that used the summarizing features of Difference of Gaussian and Determinant of Hessian, alone or combination, did not perform very well (<60% accuracy in most cases) compared to complex and Canny Edge features in all model types. One possible reason for this could be that the summaries of the DoG/DoH blob data itself can have a lot of overlap between different classes especially considering all of the noise factors that the images came with, making it difficult for any model to learn the nuances that would enable strong classification performance. In contrast, Canny Edges and complex features seemed to have better captured those nuances in images by accentuating the anatomical structures and important features within the MRI scan, leading to stronger performing models.

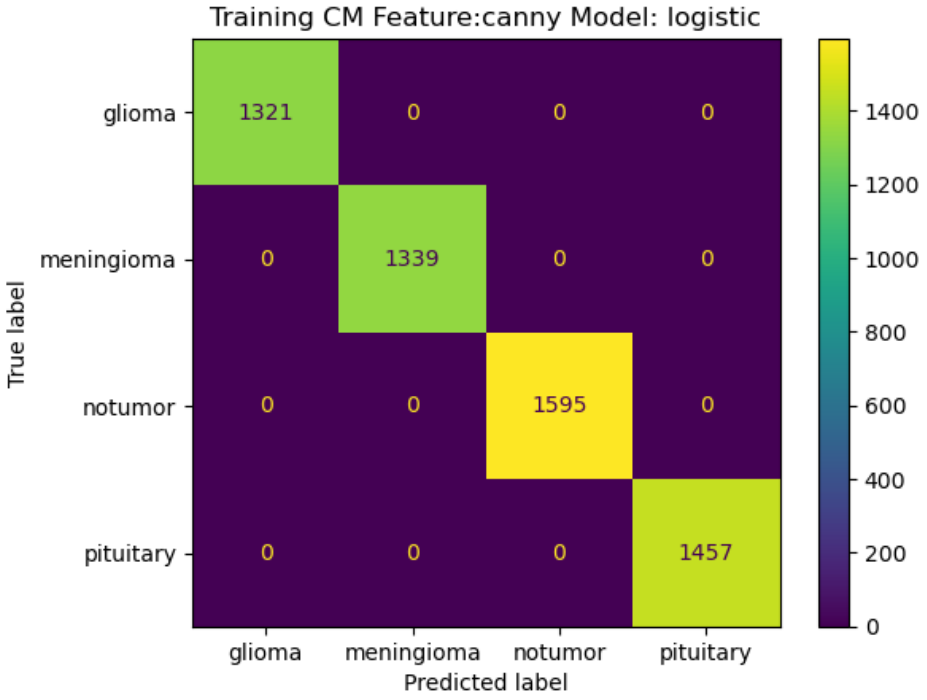


Figure 14: Canny Edge and Logistic Regression Training Confusion Matrix

4.3 Training Time

From a training time perspective, the model that trained the fastest was a linear discriminant analysis model paired with the Determinant of Hessian feature with a training time of 0.077 seconds (Figure 15). It is not surprising that a feature with only 14 components would not take very long to train compared to a feature with 1000 or 768 features that Canny Edges and the complex feature had, respectively. However, this combination of feature and model type only had an accuracy of 49.2% which would not be acceptable in a production environment, especially not a medical one. If we filter to models that only had a 90% accuracy and above, the fastest model would be logistic regression paired with Canny Edges at a training time of 3.028 seconds. This combination showed a lot of promise in terms of both accuracy and training time and could be a good candidate to watch for when testing for generalization.

4.4 Maximized Efficiency

From an efficiency perspective, we can create a custom metric that takes into account both accuracy percentage and training time. By dividing the accuracy by the training time, this custom metric gives us the biggest accuracy percentage gain per second of training time. The combination that gives the highest score in terms of this custom metric would be the linear discriminant analysis model paired with DoH with a score of 6.41, seconded by quadratic discriminant analysis paired with DoH (4.86), and then linear discriminant analysis with the combined DoG and DoH feature vector (4.59). While all of these models did not have the best accuracies ($< 60\%$), from an efficiency standpoint, they performed the best.

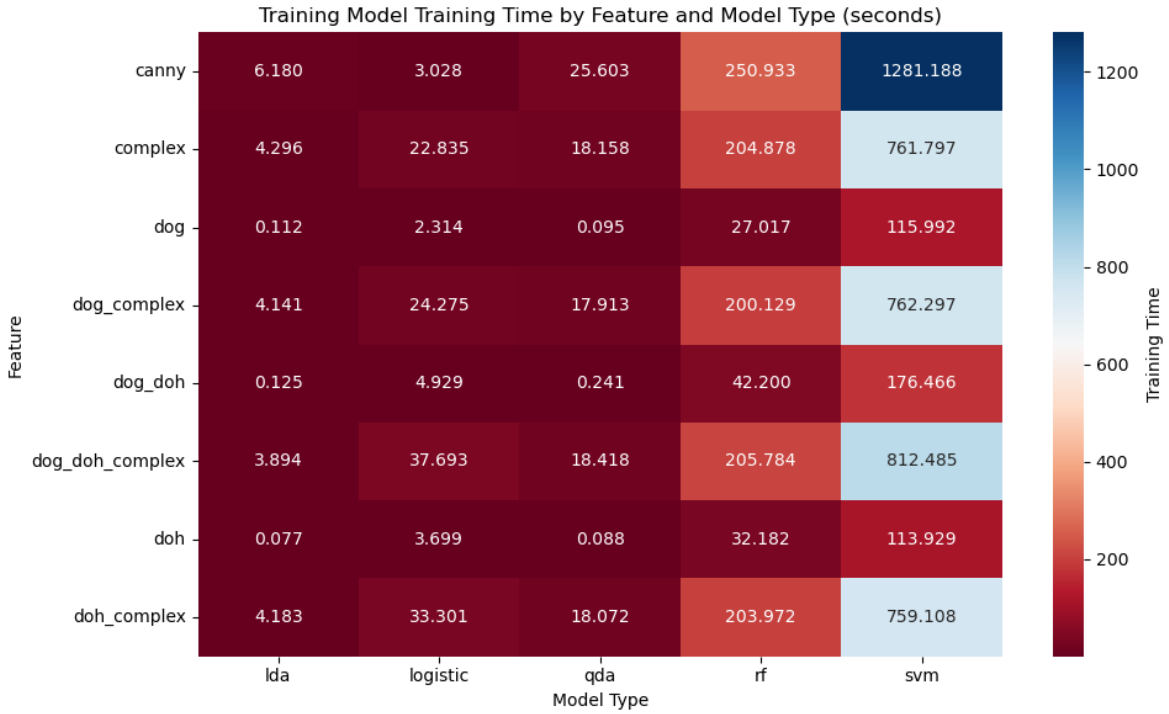


Figure 15: Training Time in Seconds by Model Type and Features

4.5 Ensemble Attempt

An additional strategy to try and maximize performance was to use a model ensemble. We took the 3 best model and feature combinations to create an ensemble model to see if it would perform just as well as logistic regression paired with Canny Edges on the training data. The thought here was that, even though logistic regression and Canny Edges performed perfectly on the training set, there is a possibility that the model was overfitted. This ensemble was a proactive attempt to maximize both training performance and generalization on the test set. The top 3 combinations were logistic regression and Canny Edges, support vector machines (linear kernel) and Canny Edges, as well as logistic regression and the complex features. Looking at the confusion matrix in Figure 16, the ensemble model comes pretty close to 100% accuracy, only classifying one candidate incorrectly in the training set.

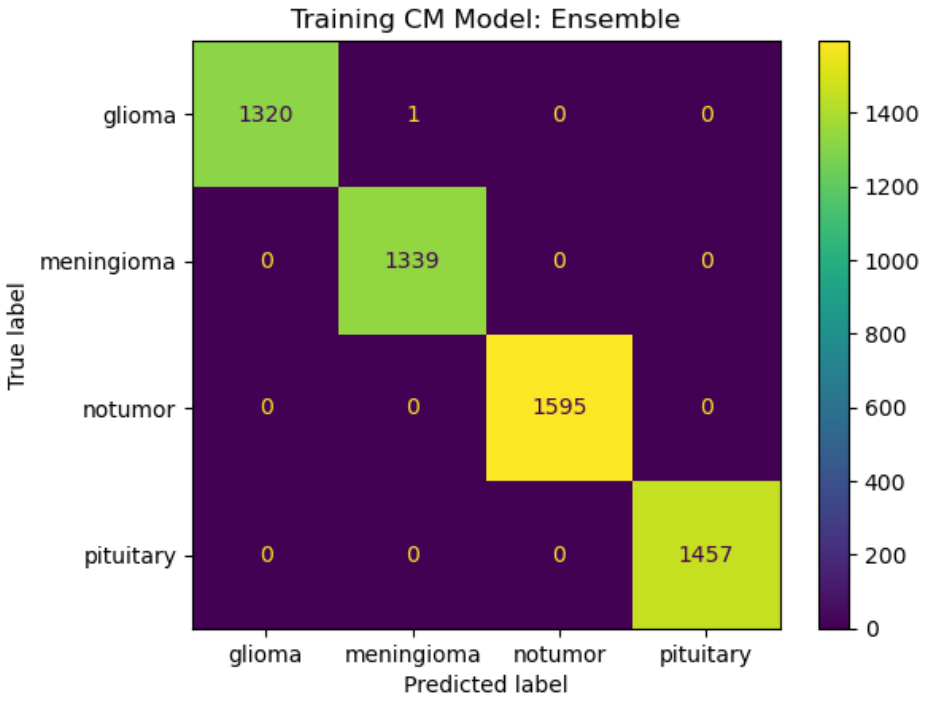


Figure 16: 3-Model Ensemble Training Confusion Matrix

5 Test Results

5.1 Test Accuracy

Finally, after training 40 different models along with 1 ensemble, we assessed how well each of the models generalized to the test set. The full accuracy matrix can be seen below in Figure 17. We can see that overall, the model that generalized the best was the logistic regression model paired with the complex feature with an accuracy of 96.7%. Again, we see high performance with a linear model. The most misclassifications came from glioma getting predicted as meningioma (Figure 18), indicating possible similarities in their semantic and anatomical features. The best training combination, logistic regression and Canny Edges, had a final test accuracy of 92.9% which is still good, but indicates slight overfitting occurred. DoG/DoH features used in combination with the complex feature showed close performance with the complex feature used alone, indicating that they were ultimately not too effective. For the model ensemble using the top 3 models from the training experiments, the accuracy was 94.2% which comes in second place, indicating that it had some effect on mitigating the overfitting. Overall, there were many combinations of features and models that performed relatively well with accuracies well into the 90% range while also maintaining very reasonable training times.

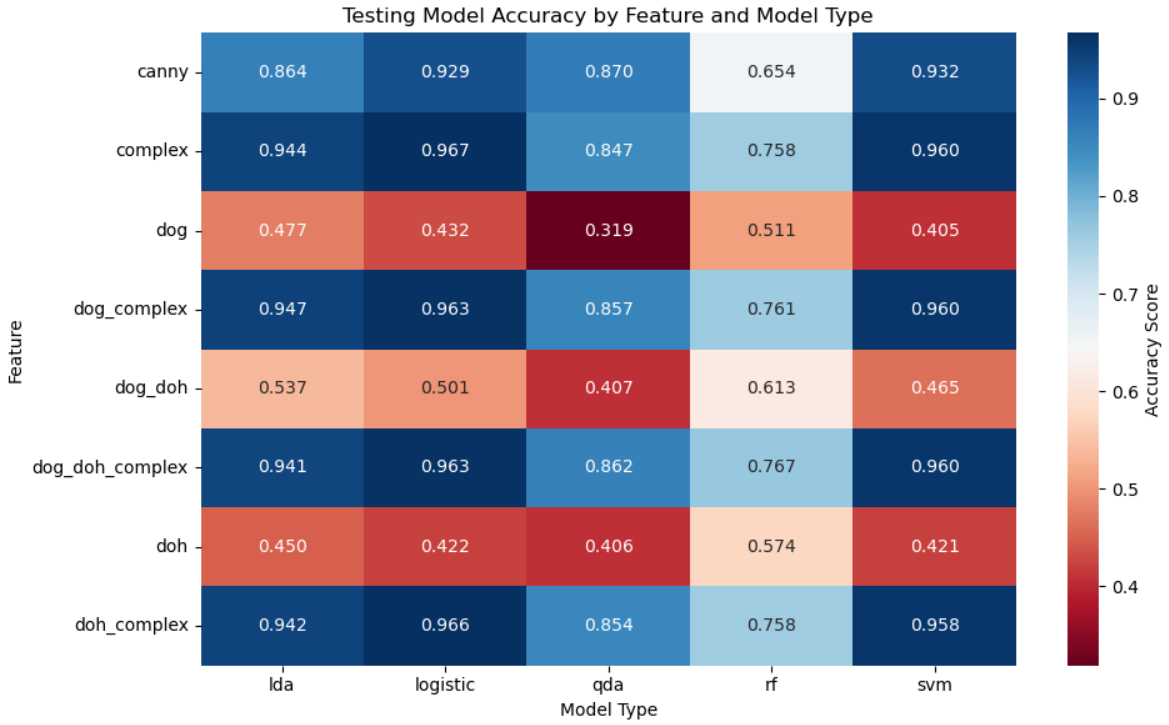


Figure 17: Testing Accuracy by Feature and Model Type

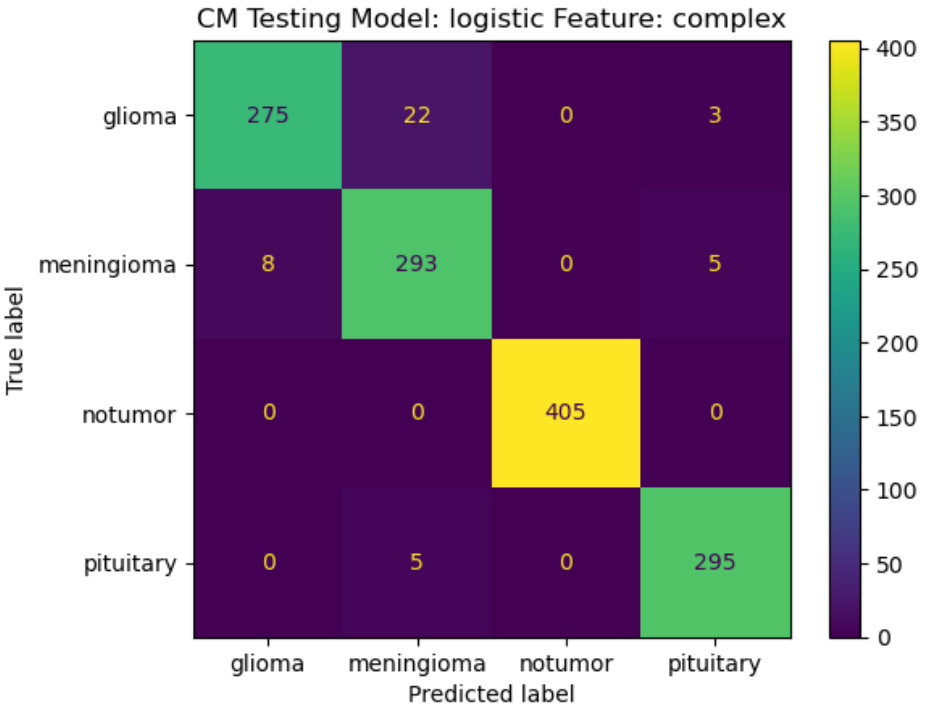


Figure 18: Complex Feature and Logistic Regression Testing Confusion Matrix

5.2 Inference Time

Another consideration for implementing a model in a production environment is the inference time. Although most of the model inference times are fairly inconsequential when compared to the model

training times, it is still important especially when performing inference on a larger volume of images (e.g. millions). Looking at Figure 19, we can see that logistic and linear discriminant analysis models are the fastest, followed by quadratic discriminant analysis, and lastly support vector machines. Support vector machines were much slower than the rest of the models. As expected, the inference time also scaled with the feature length with the fastest inference times coming from the summarized DoG/DoH feature vectors with only a length of 14 each per image. On the other end, Canny Edges had a 1000 feature length which had longer inference times on average assuming the same model type.

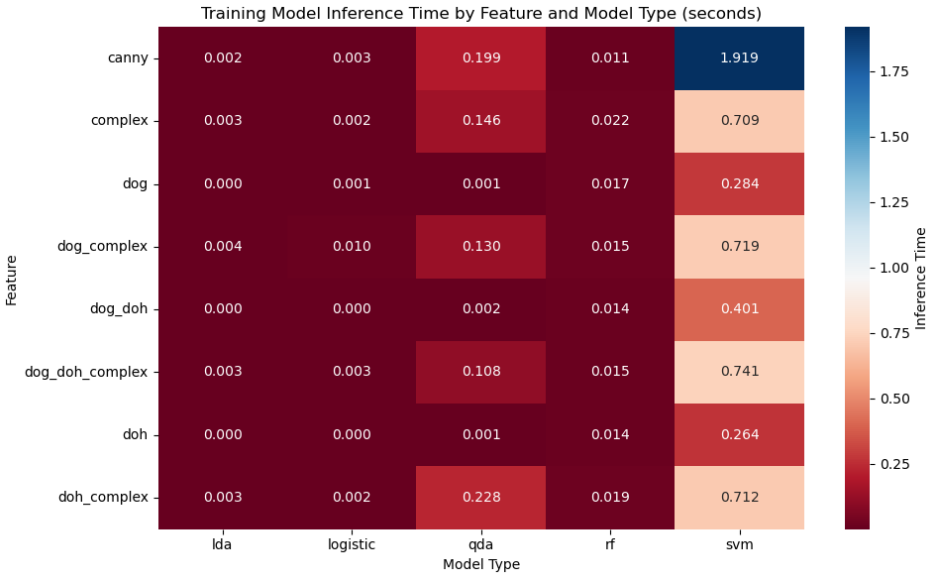


Figure 19: Model Inference Times

5.3 Areas for Improvement

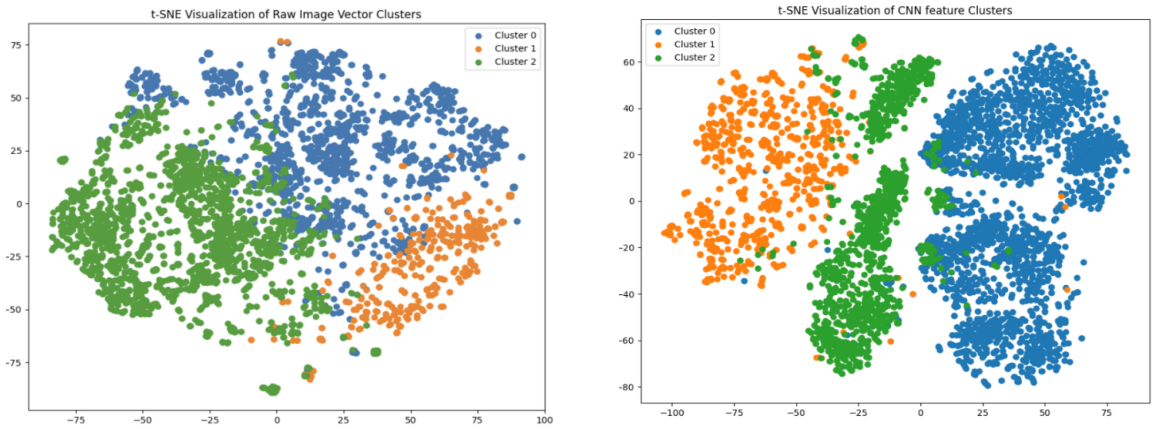
While our best model performed pretty well at 96.7% accuracy on the test set, there are certainly areas for improvement, especially in the training time and computational efficiency. The best model utilized the full length of the complex feature (768) which is still less than the Canny Edge feature length at 1000; however, we see through the principal component analysis (PCA) that we only need around 70 components of the complex feature to obtain around 95% of the explained variance in the feature. The reduction of from 768 components to around 70 is around a 90% decrease in the overall length of the feature, which might be able to drastically decrease the training time of the model while still maintaining relatively high performance.

6 Conclusion

It is interesting to see that even with an image dataset that contained a variety of noise factors, utilizing a proper image pre-processing pipeline, simple and complex computer vision features, and traditional classifiers can yield fairly good multi-classification performance without the need to train a deep CNN which requires higher compute power and training times. Moreover, through this exercise, we learned that much of the time and effort into creating a good model is placed in the image pre-processing and feature extraction space rather than the model training space.

A Appendix

A.1 Complex Feature Experimentation



(a) t-SNE plot of raw image vectors

(b) t-SNE plot for CNN features

Figure 20: Supplementary visualizations of complex feature space structure: (a) t-SNE plot of raw image vectors: Visualization of pixel-level representations without any learned feature extraction. The clusters show poor separability, and upon visual inspection, many samples were assigned to incorrect clusters, indicating that raw pixel values do not encode sufficient semantic structure for distinguishing between anatomical views or depth slices. (b) t-SNE plot of CNN features: Visualization of MobileNetV2 feature embeddings. Compared to raw vectors, CNN features yield improved clustering structure, but still show notable overlap across clusters and frequent misassignments, suggesting limited ability to capture global anatomical relationships.

A.2 Blob Feature Experimentation

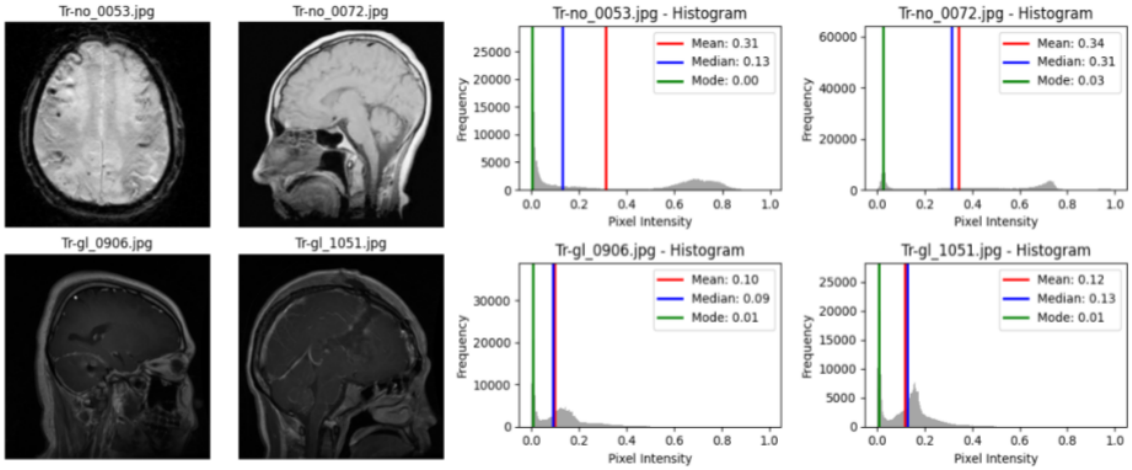


Figure 21: Sample Contrast Variated Images and Their Pixel Intensity Histograms: Images that were too bright tended to have higher mean pixel intensities (≈ 0.3) than those that were too dark (≈ 0.15).