Brain Tumor Classification: A comparison of ANN and CNN
FYS-STK4155 - Project 3

Navidah Amiri, Carolina Ceccacci, Lise Chevalier & Esther Zijerveld
University of Oslo
(Dated: December 18, 2025)

This study utilizes the openly available Brain Tumor Classification (MRI) dataset, accessible on GitHub [1]. The dataset comprises 3264 T1-weighted, contrast-enhanced MRI images that were augmented for analysis. MRI imaging is particularly suitable for brain tumor detection due to its high soft tissue resolution and non-reliance on ionizing radiation.

We employed both Artificial Neural Networks (ANNs) and Convolutional Neural Networks (CNNs) to classify tumors into four categories: no tumor, meningioma, glioma, and pituitary. For the ANN, the best validation performance was achieved using Leaky ReLU activation functions, a learning rate of 10^{-3}, and a hidden layer architecture of 128 and 64 nodes. This configuration yielded a validation F_1 score of 67.1% with an accuracy of 69.5%. However, performance on the test set dropped significantly, with an F_1 score of 37.1% and accuracy of 38.8%. Results for the CNN model showed higher performance. This was achieved for a CNN with architecture setup of 3 convolutional layers, 128 nodes, 2 dense layers with 512 channels in each dense layer, a dropout rate of 0.4 and learning rate 0.0001. This architecture gave a test accuracy of 77.16% and an F_1 score of 72.85%.

I. INTRODUCTION

Brain tumors, also known as intracranial tumors, represent an abnormal and unregulated proliferation of cells within the brain tissue. With over 100 distinct types [2], these growths pose severe, often life-threatening challenges due to their uncontrolled expansion and proximity to critical neural structures. A key difficulty in treatment is the body's inability to recognize these cells as foreign, as they possess the person's own DNA, effectively evading the immune system. Currently, the most effective technique for the detection and diagnosis of these lesions is Magnetic Resonance Imaging (MRI).

In clinical practice, the most widespread primary brain tumors include meningioma, glioma, and pituitary tumors. Gliomas are particularly challenging as malignant tumors with a high chance of recurrence [3]. Conversely, meningioma is recognized as the most common benign tumour. Pituitary tumours, while often benign, present a unique clinical dilemma as they are not graded on the World Health Organisation (WHO) system but lie close to the brain and can invade the Central Nervous System (CNS) [4]. Distinguishing between these diverse types is essential for establishing an appropriate prognosis and treatment plan.

Early detection of brain tumors is very important since they can grow rapidly. The identification and classification of brain tumors can be complicated and requires expert physicians and radiologists. The manual detection and identification can be time consuming, subject to inter- and intra-observer variability and can face logistic challenges in developing regions around the globe with limited access to medical expertise. Since early detection is paramount, there is a critical need for early and precise diagnosis.

While MRI provides high-resolution anatomical details, the manual analysis of these scans for tumour detection, segmentation, and classification is time-consuming and relies heavily on the expertise of the radiologist. Small or early-stage tumours can be easily overlooked, and the distinction between tumour subtypes (e.g., classifying a tumour as a meningioma versus a glioma) is often subtle and challenging, even for trained specialists. Consequently, there is a critical need for robust, efficient, and objective computational methods to assist clinicians in the rapid and accurate diagnosis of primary brain tumors from MRI scans.

To address the limitations of conventional methods, this report explores the application of Deep Learning (DL), a groundbreaking subset of Artificial Intelligence (AI), to the automated classification of brain tumors from medical images. At the core of most deep learning models are Artificial Neural Networks (ANNs), which form the basis for more advanced architectures. ANNs are structured in multiple layers—an input layer, several hidden layers, and an output layer. However, the number of parameters in ANNs can quickly become exponentially large. In addition, the ANN takes a 2D image and flattens it to a 1D array so spatial information is not preserved. CNNs address these limitations by employing convolutional filters and pooling layers to automatically perform feature selection, efficiently identifying the most discriminative visual patterns. CNNs utilise convolutional layers to detect patterns and spatial hierarchies. Despite their limitations in image-based tasks, ANNs remain computationally simpler and are still widely used as baseline classifiers, making them a useful point of comparison when evaluating more advanced architectures such as CNNs.

This report details the development and evaluation of ANN and CNN-based deep learning models designed to classify brain tumors, demonstrating their potential to provide a quick, accurate, and objective diagnostic aid, thereby enhancing clinical decision-making and improving patient outcomes globally. We based our research on Kadam, Bhuvaji and Deshpande's [4] dataset of MRIs. We first evaluated several ANN and CNN models separately under identical training and testing conditions, before comparing the best-performing ANN model with the best-performing CNN model. As we will see, when accuracy and precision are the priority in image analysis, CNNs are the best choice. While ANN models are included as a baseline for comparison, greater emphasis is placed on CNN architectures due to their suitability for image-based tasks and their central role in the experimental analysis.

Section II provides a comprehensive overview of the methodologies employed in this project, a description of the dataset, preprocessing steps, model architectures, and evaluation metrics used to support this comparative analysis. The outcomes of the experiments are presented in Section III, and analyzed in Section IV. Section V summarizes the key findings and conclusions derived from the study.

II. METHODS

A. Pre-processing the dataset

The dataset used for this analysis consists of Magnetic Resonance Imaging (MRI) brain scans from the Kaggle [1] open data website. The images are already divided into a training set and a test set containing 2870 and 394 images respectively. Each set contains four folders classifying the images into brain tumor types: no tumor, glioma, meningioma, and pituitary as shown in Figure 1. The images are two-dimensional T1-weighted contrast-enhanced scans, and the dataset includes three different views: axial, coronal and sagittal [4]. We used both Artificial Neural Networks (ANNs), Section II B, and Convolutional Neural Networks (CNNs), Section II C, for the classification task and compared the results.

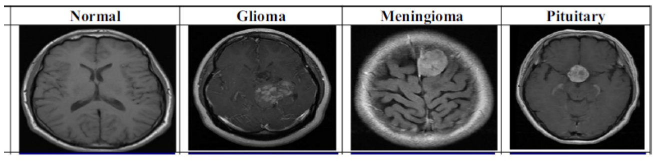

Figure 1. Illustration of the four primary brain tumor classes used in this study (Glioma, Meningioma, Pituitary Adenoma, and Normal/No Tumor), highlighting their distinct anatomical locations [5].

We started by resizing the images to the size $224 \times 224 \times 3$. This size was chosen so that most of the medically relevant information that the network needs to learn to recognize tumors would be preserved, while still decreasing the resolution enough to make sure that our neural networks could be trained efficiently.

Next, we decided to use the technique of data augmentation. Data augmentation is a technique used in machine learning to make the most out of the training data available by artificially increasing its size and diversity. This is done by creating modified copies of the dataset while preserving the class labels, by applying various transformations to the existing dataset. This technique is especially useful in the field of medical imaging, where the acquisition of quality data and its labelling is time-consuming and expensive [6]. The performance of CNNs and other deep-learning models relies heavily on the quantity and the diversity of the data used to train them, as CNNs for example can easily start overfitting when they are trained on small datasets [7]. Data augmentation adds data points for learning without needing to acquire new data, it helps the model learn invariant features and improves the model's ability to generalize to unseen data [8].

We used the transform modules from torchvision to augment our training data [9]. Deciding which transforms to use depends heavily on the kind of dataset we have, and which areas it is most important for the model to be robust in. When working with MRI scans specifically, there are several transforms that are relevant to this particular case to increase the robustness of the model. We can introduce rotational invariance in the model by rotating some images randomly between an angle (20 degrees in our model), orientational invariance by flipping the image horizontally and vertically, location invariance by translating the position of the image horizontally or vertically, occlusion invariance by removing random spots in the image, and many more. We can also make sure that the model will be able to recognize tumors even if the colours or brightness of the scan changes by randomly changing the brightness, contrast and saturation of the images in the dataset. The goal is to train the network to recognize a tumor even if the configurations of the scans change.

We limited the number of transforms used in our model and the size of the augmented dataset. We could not increase the size of our dataset and number of training epochs too much considering we mostly ran the code on Google Colab and Kaggle. We used the RandomRotation, RandomHorizontalFlip, and RandomVerticalFlip transforms. We then created 6 different versions of the transformed dataset which we then combined to form one larger dataset to use for training.

We also used the transforms ToTensor and Normalize to standardize the training and the test data [10]. This converts the image into a PyTorch tensor, rearranges dimensions so that it can be used with PyTorch, scales the pixel values from [0, 255] to [0.0, 1.0] and shifts the data from [0.0, 1.0] to [−1.0, 1.0]. This makes sure that the data can be used with PyTorch, speeds up the training, improves convergence, prevents exploding or vanishing gradients, and improves generalization [11].

At the end of our study, we also introduced the transforms RandomResizedCrop (crops a random portion of the image and resizes it to a given size), and ColorJitter (randomly change the brightness, contrast, saturation and hue of the image) [9].

B. Artificial Neural Network (ANN)

Artificial Neural Networks (ANNs) are multi-layer fully-connected neural nets (NNs). They consist of an input layer, multiple hidden layers, and an output layer. Every node in one layer is connected to every other node in the next layer. In this investigation we use a Feed Forward ANN (FFNN). We assume background knowledge on ANNs and FFNNs.

Table I summarises the hyperparameter combinations such as batch size, number of epochs for training and activation functions as well as the architectures (layers and number of nodes).

Table I. Overview of hyperparameters utilised in the ANN

Hyperparameter	Value
Batch Size	32
Epochs	40
Activation	Sigmoid, Relu, LeakyRelu
Layers	(64), (128), (256), (128, 64), (256, 128), (256, 128, 64)
Learning rates	1×10^{-2}, 1×10^{-3}, 1×10^{-4}

1. *Implementation*

The Artificial Neural Network (ANN) was implemented using the Keras high-level API within the TensorFlow framework. Due to its comparatively low computational complexity, the ANN model was trained and evaluated in a local CPU-based environment without the need for GPU acceleration. The implementation focused on a fully connected architecture operating on flattened image inputs, serving as a baseline model for comparison with the CNN-based approaches.

C. Convolutional Neural Network (CNN)

A convolutional neural network is a suitable choice for this dataset because MRI scans contains rich spatial patterns and features that CNNs are specially designed to capture, unlike ANNs which treat all inputs as independent features. CNNs exploit the two-dimensional layout of MRI scans through convolutional filters that learn local patterns such as edges, textures, and tumor boundaries. This not only improves classification performance but also greatly reduces the number of trainable parameters compared to an ANN with fully connected layers applied directly to raw images. As a result, CNNs are more efficient, less prone to overfitting, and better able to capture the complex visual features necessary for accurate brain tumor classification

1. *Activation Function*

We apply the Rectified Linear Unit (ReLU) activation function after each convolution and also to the fully connected layers (except to the last fully connected layer). The ReLU activation function is given by:

$$ReLU(z) = max(0, z) \tag{1}$$

This function helps the network learn non-linear relationships between the features in an image so that the network can better recognize different patterns [12]. Furthermore, using the ReLU function avoids suffering from the vanishing gradient problem [11].

2. *Convolutional Layers*

Each convolutional layer in the network performs a convolution, which means that it is applying a sliding window function (kernel or filter) to the input, which is a matrix of pixel values representing an image from the training set [12]. Several filters of equal size are applied in the layer. Each filter learns to detect features and recognize a specific pattern from the image. For example the first layer might learn edges, the second layer combines edges into shapes, and the third could learn more complex patterns [13]. Each node in the convolutional layer is connected only to a subset of the nodes in the previous layer. This is designed to reproduce the connectivity pattern in the animal visual cortex, where individual neurons only respond to stimuli from small sub-regions of the visual field. This is what allows neurons in the cortex, and convolutional networks, to exploit the strong local spatial correlations present in natural images [11].

3. *Pooling Layers*

Applying a pooling layer reduces the spatial dimensions of the feature map (input image). We used MaxPool2d for each convolutional layer. Each time this function is applied, it takes the maximum value in each 2×2 region. So if the image is of size 224×224, one application of a pooling layer will reduce its size to 112×112.

Applying it a second time will reduce the size of the image further and so on. This makes the network more efficient (as it reduces the memory used while training the network), and helps the network focus on the most important features in the image [13].

4. *Padding*

Padding an image prior to convolution involves adding extra rows and columns around the boundary of the image grid, filled with a chosen value, most commonly zeros (zero padding). The size of the padding is customizable, which means how many rows and columns the padding consists of. In this project, we will not focus on padding as a choice for our tuning parameters.

5. *Fully Connected Layers*

A fully connected layer is like a traditional neural network layer with ReLU as the activation function. After extracting the most important features with the convolutional layers, the output matrix of the last pooling layer is then flattened and passed through a traditional neural network for classification [13].

6. *Dropout*

With dropout, a regularisation technique, some neurons are randomly switched off during training. This forces the model to learn more robust patterns instead of relying on particular features too much [13].

7. *Training procedure*

For each epoch, the network goes through the same steps. First, it goes through the forward pass, where the images are fed into the network to get predictions. The class probabilities from multi-class classification are obtained from the soft-max function which is given by

$$\hat{y}_i = \frac{e^{z_i}}{\sum_{i=1}^{C} e^{z_i}} \tag{2}$$

where the z_i is the output of the network, i.e. prediction for a given class, and C is the number of classes in the classification problem.

Then, it calculates the loss with the cross-entropy function 3 and compares the predictions made by the network to the actual labels. Next, the calculated loss is backpropagated through the network. The gradients of the loss with respect to each of the network's parameters - the weights and biases - are calculated. These gradients indicate the direction and magnitude of change needed for each parameter to reduce the loss, and then the weights are updated [13].

8. *Implementation*

The models developed for this study were implemented and trained using several key libraries and cloud computing platforms. Google Colaboratory and Kaggle Notebooks were employed for the computational heavy lifting required by the CNN models. These environments provide access to GPUs essential for deep learning tasks. PyTorch [14] was used as the primary framework for constructing, training, and evaluating the CNN models. Seaborn [15] was utilized for generating statistical data visualizations, including performance plots, confusion matrices, and heatmaps, for both ANN and CNN models. The code was mostly based on code examples found on the class' github site [11].

D. Evaluation of performance

1. *Cost function*

The Categorical Cross-Entropy (CCE) loss function is used to measure the performance of the model, as we have a multi-class classification problem:

$$CCE = -\sum_{j=1}^{C} y_j log(\hat{y}_j) \tag{3}$$

where y_i is the indicator of which class it is and $\hat{y}_i$ is the probability of selecting that class which is often from the soft-max function 2. The CCE function measures how close the model's predictions are to the true labels in a classification problem. A lower loss means that the model makes relatively better predictions, while a higher loss indicates that there are relatively higher discrepancies between the predictions and the true labels. In a multi-class problem, a lower loss means that the model is assigning high probabilities to the correct classes and low probabilities to the wrong classes. This loss function is used to train neural networks to make better predictions by rewarding correct predictions and penalizing the model when it assigns low probability to the correct class [16].

2. *Test Accuracy*

When performing a classification task, accuracy is defined as the proportion of correct classifications in the test set. A high test accuracy would indicate that the model performs well on unseen data. Accuracy is defined as the number of correctly classified targets divided by the total number of targets and is given by the following equation:

$$\text{Accuracy} = \frac{\text{correct classifications}}{\text{total classifications}} = \frac{\sum_{i=1}^{n} I(t_i = y_i)}{n}, \tag{4}$$

where I is the indicator function, 1 if $t_i = y_i$ 0 otherwise, t_i represents the target while y_i represents the output from the network, and n is the number of targets.

While accuracy gives us a good indication of the performance of the network when we have a balanced dataset, it can be misleading in other cases. For example, if one class represents 90% of the data, and the network predicts that the outcome will be this class in any case, the network would obtain an accuracy score of 90%. In that case, even if the accuracy score was high, the network would not be making good predictions. There are therefore several other metrics that are commonly used and that will be implemented in this analysis to better evaluate the performance of a model.

3. *Test F1 Score*

The F1 score is the harmonic mean between the precision and the recall.

The precision measures the proportion of the model's positive predictions which were truly positive [17]. The precision is a meaningful metric to use when the number of true positive predictions is large enough. If that number is small, the precision score may be deceiving. The precision increases as the number of false positive predictions decreases. It can be illustrated in the following equation:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5}$$

The recall measures the proportion of the true positives that the model correctly identified [17]. The recall increases when the number of false negative predictions decreases:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{6}$$

In a multi-class classification problem like we have here, the precision and recall are computed for each class individually, and the overall score is computed from those using a weighted average (each class' metric will be weighed by the number of true instances in that class) [17].

The F1 score balances the precision and the recall scores into one number in the following way:

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

If the precision and the recall are both perfect (equal to one), the F1 score will also be equal to one. If the two metrics are equal, the F1 score will also be equal to their value. And if the two metrics are far apart, the F1 score will be closer to the worse one. When the classes in the dataset are imbalanced, the F1 score may provide a better insight into the model's performance than the accuracy. Depending on the type of dataset we have, and depending on what is most important that the model is able to do, we could favour using one metric over the other. For example, if it is most important that the number of false positives is minimized , we would favour using the precision score. But if it is more important to minimize the number of false negatives, for example in the case of cancer detection, using the recall score would be preferable. If both are equally important, the F1 score should be used. In the following analysis, we have decided to use the F1 score to balance the two metrics.

4. *Confusion matrix*

A confusion matrix is a tabular representation used to evaluate the performance of a classification model by comparing predicted labels with actual ground-truth labels. It summarizes outcomes into four key categories; true positives, true negatives, false positives, and false negatives—providing a detailed view of how well the model distinguishes between classes. Unlike aggregate metrics such as accuracy, the confusion matrix reveals class-specific strengths and weaknesses, making it especially useful in imbalanced datasets or contexts where different error types carry different costs. This structure allows researchers to compute a variety of diagnostic metrics, including precision, recall, and F1-score, thereby offers a comprehensive assessment of model behavior.

5. *ROC curves*

We use Receiver Operating Characteristics (ROC) curves because they show how well a classifier distinguishes between classes across all possible decision thresholds, not just at one fixed cutoff as the accuracy and F_1 do. ROC curves plot the True Positive Rate against the False Positive Rate across varying thresholds, with the Area Under the Curve (AUC) providing a scalar measure of discriminative ability (AUC = 0.5 indicates random guessing, while AUC = 1.0 indicates perfect classification).

6. *Heatmaps*

Hyperparameter performance was visualized using heatmaps, where color intensity encodes validation scores for different metrics (e.g. Accuracy, F_1) across parameter combinations. This allows rapid identification of optimal configurations and comparison between activation functions.

E. Use of AI tools

Generative AI tools were employed primarily to enhance the clarity of written text and support code development. Specifically, we utilized ChatGPT, GPT UiO, Google Gemini, and Microsoft Copilot. GPT UiO, which is based on OpenAI's models, was accessed through the institution's framework to ensure compliance with GDPR regulations. In addition, GitHub Copilot was leveraged to provide coding suggestions and assist in debugging Python code snippets.

III. RESULTS

We implemented both an Artificial Neural Network (ANN) and a Convolutional Neural Network (CNN) to address the classification task of brain tumors. The ANN was developed using the TensorFlow API [18], while the CNN was implemented in PyTorch [14]. Both models were trained on the Brain Tumor Classification (MRI) dataset, consisting of 3264 T1-weighted contrast-enhanced MRI images which were then augmented. Performance was evaluated using accuracy and F_1 score to capture both overall correctness and class balance.

Detailed results for each architecture, including validation and test scores, are presented in the following subsections.

A. ANN

Figure 2 presents the heatmap of validation F_1 score for the ANN with Leaky ReLU activation, which achieved the highest accuracy and F_1 score among all hyperparameter combinations explored via grid search. For comparison, heatmaps corresponding to the Sigmoid and ReLU activations are provided in Appendix A (Figures A3 and A4). All heatmaps were generated using a consistent color scale to ensure comparability.

Table II summarizes the optimal hyperparameters identified for each activation function, as derived from the heatmaps. The final two columns report the corresponding validation accuracy and F_1 scores, highlighting the superior performance of the Leaky ReLU configuration.

Figure 3 displays the confusion matrix for the Leaky ReLU activation. The confusion matrices for sigmoid and ReLU are given in Appendix A (Figures A5 and A6). The Leaky ReLU produced the most balanced predictions with the least false positive predictions. Across all models, pituitary tumors were generally classified correctly, reflecting their distinct imaging characteristics. However, the ReLU-based ANN frequently misclassified glioma and meningioma cases as "no tumor," underscoring its weaker discriminative ability (Figure A5, Appendix A). Furthermore, Figure A6 indicates that when the Sigmoid activation function is used, the model tends to classify most cases as "no tumor".

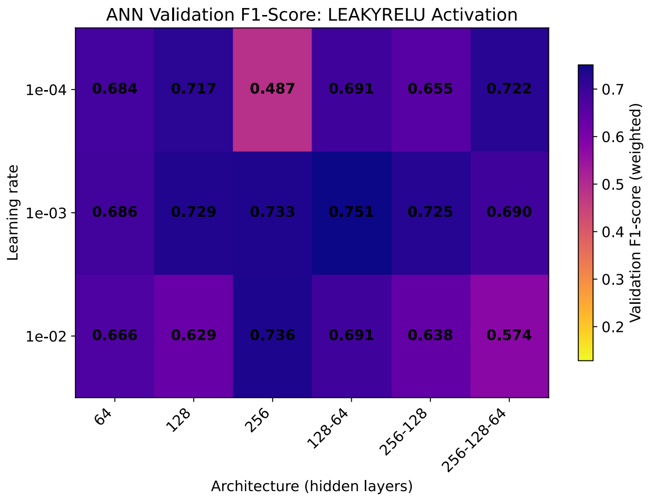

Figure 2. Heatmap illustrating the validation F_1 scores for Artificial Neural Networks (ANNs) using the Leaky ReLU activation function. The scores are presented across three learning rates (10^{-2}, 10^{-3}, 10^{-4}) and six network architectures listed in Table I. The color scale is consistent across all activation function heatmaps for direct comparison (Appendix A).

Table II. Summary of Top-Performing ANN Hyperparameter Configurations. This table lists the architecture and learning rate that achieved the highest validation F_1 score for each activation function, along with the corresponding accuracy score. "LR" stands for learning rate.

Architecture	Activation	LR	Accuracy	F1
128-64	ReLU	10^{-4}	0.669	0.659
128-64	Leaky ReLU	10^{-3}	0.695	0.671
256	Sigmoid	10^{-4}	0.139	0.037

Figure 4 presents the Receiver Operating Characteristic (ROC) curves for the Leaky ReLU activation. All curves lie above the random-guessing baseline, demonstrating improved sensitivity and specificity across the four tumor classes. In contrast, the ROC curves for Sigmoid and ReLU activations remain closer to the dashed line, reflecting weaker predictive performance. Notably, pituitary tumors consistently yield ROC curves furthest from the baseline across all activation functions, highlighting their relative ease of classification.

Figure A7 reports the macro Area Under the Curve (AUC), which is 0.624—only marginally better than ran-

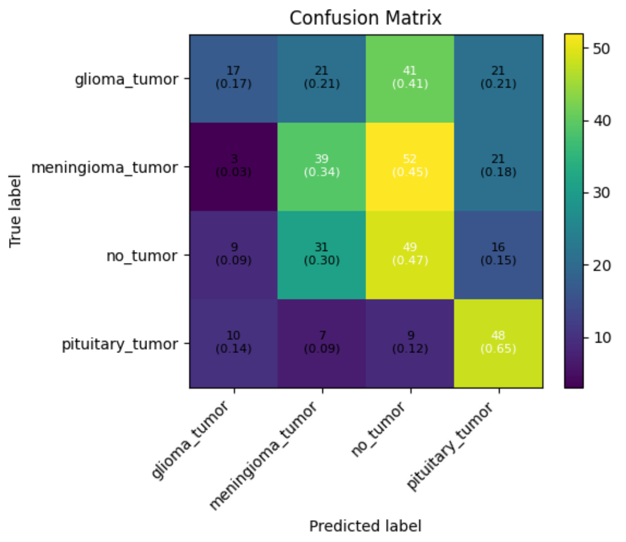

Figure 3. Confusion Matrix for the Artificial Neural Network (ANN) using the Leaky ReLU activation function. This matrix visualizes the classification results achieved with the optimal hyperparameters listed in Table II, which correspond to a Macro F_1 score of 0.671 on the validation set. Values are presented as raw counts with the corresponding class recall (True Positive Rate) in parentheses.

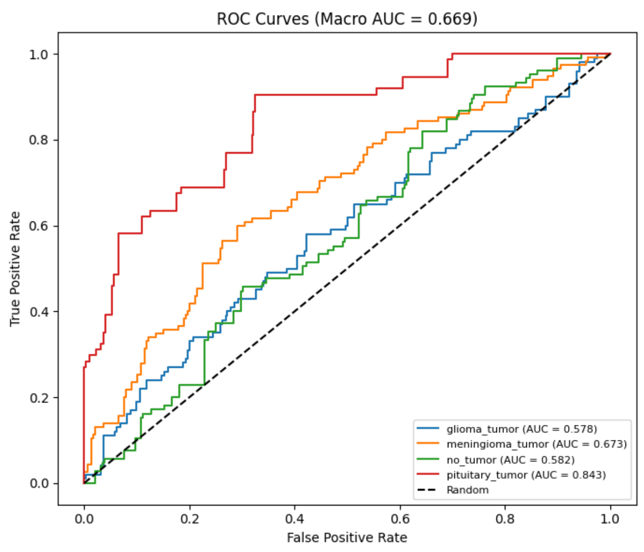

Figure 4. Receiver Operating Characteristic (ROC) Curves for the Leaky ReLU-activated ANN with learning rate and architecture that was presented in Table II. The plot illustrates the True Positive Rate (TPR) versus the False Positive Rate (FPR) for each of the four brain tumor classes. The Area Under the Curve (AUC) is provided for each class in the legend, showing pituitary tumor (AUC = 0.843) as the most separable class, and glioma tumor (AUC = 0.578) as the least separable. The overall model performance is summarized by the Macro AUC of 0.669. The dashed line corresponds to random guessing.

dom guessing. Class-specific AUC values further emphasize this limitation, with glioma (AUC = 0.566) and "no tumor" (AUC = 0.558) performing only slightly above chance level. When comparing the ROC curves we find that the Leaky ReLU gives a slightly better model compared to ReLU and both ReLU and Leaky ReLU are significantly better compared to Sigmoid.

The training histories presented in Figures 5, A1, and A2 (Appendix A) demonstrate that the Leaky ReLU activation function resulted in the most stable convergence for both loss and accuracy metrics compared to both the Sigmoid and standard ReLU functions. However, the Leaky ReLU model (Figure 5) shows clear signs of overfitting beginning around Epoch 5. This is evidenced by the classic divergence pattern where the training accuracy continues to rise and fit the training data, while the validation accuracy simultaneously declines, indicating poor generalization. In comparison, the ReLU model (Figure 5) showed high volatility in its validation metrics, and the Sigmoid model (Figure A2) suffered from catastrophic failure to generalize, characterized by large spikes in validation loss.

Figure A1 shows the training history of the ANN with ReLU activation. The loss decreases from around 7 to approximately 1, while validation loss converges below the training loss, likely due to the small validation set or the application of regularization only to the training data. Accuracy rises to just under 70 %, with training accuracy increasing steadily from 40 % to 60 %. Validation accuracy fluctuates more strongly, which may be attributed to the relatively small batch size of 32.

Figure A2 illustrates the training history of the ANN with Sigmoid activation, using the configuration in Table II. The model exhibits instability and poor generalization: training loss remains nearly flat, suggesting vanishing gradients caused by the combination of Sigmoid activation and learning rate. Validation loss is volatile, indicating overfitting to the training dataset.

The test F_1 score for the configurations presented in Table II - that gave the highest validation F_1 score and accuracy for each activation function - are presented in Table III.

Table III. The test set performance for the models that were highest validation F_1 score as mentioned in table II.

Model	Loss	Accuracy	F1 score
128-64, relu, 10^{-4}	2.26	0.358	0.342
128-64, leaky relu, 10^{-3}	1.68	0.388	0.371
256, sigmoid, 10^{-4}	1.58	0.282	0.143

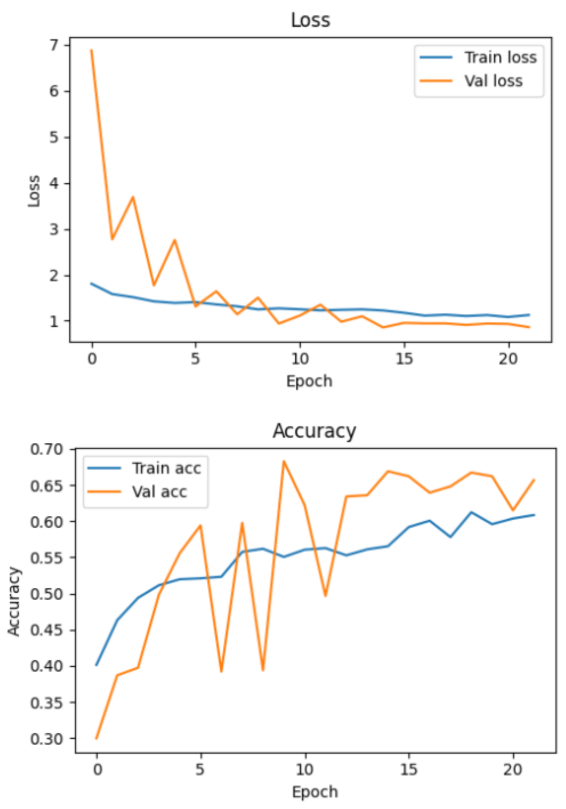

Figure 5. Training History for the Artificial Neural Network (ANN) using Leaky ReLU activation, a learning rate of 0.0001 and a two-layer 128-64 neuron layer. The 10-epoch training run is illustrated, where the left plot displays the Training (blue) and Validation (orange) loss, and the right plot shows the corresponding Training (blue) and Validation (orange) accuracy.

B. CNN

To study the performance of our CNN model systematically, we varied aspects of our CNN architecture, namely the number of convolutional layers, number of filters (nodes) in each layer, dropout regularization, the number of dense layer, and number of channels in dense layers, while training all models under a consistent set of conditions (e.g. fixed number of epochs and using Adam as optimizer). The objective of this analysis is twofold: first, to identify architectural configurations that produce the strongest classification performance, and second, to understand how model depth, capacity, and regularization affect learning dynamics in the context of MRI-based tumor classification.

We started by comparing the performance of the architectures for different numbers of convolutional layers and nodes in the CNN model, with a fixed dropout and number of channels in the dense layers. These results are shown in Table IV.

Table IV. Performance of different CNN models on the test set (number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32, number of channels in the dense layers = 128, dropout = 0.4)

Model	**Loss**	**Accuracy %**	**F1 score**
1-conv-32-nodes-0-dense	0.2785	62.94	0.5871
1-conv-64-nodes-0-dense	0.2888	61.68	0.5853
1-conv-128-nodes-0-dense	0.3135	55.33	0.5153
2-conv-32-nodes-0-dense	0.2339	65.48	0.6093
2-conv-64-nodes-0-dense	0.1236	70.05	0.6534
2-conv-128-nodes-0-dense	0.0894	71.57	0.6658
3-conv-32-nodes-0-dense	0.2896	63.71	0.5936
3-conv-64-nodes-0-dense	0.1551	71.07	0.6645
3-conv-128-nodes-0-dense	0.0750	70.56	0.6630
1-conv-32-nodes-1-dense	0.3582	66.24	0.6055
1-conv-64-nodes-1-dense	0.5323	60.41	0.5564
1-conv-128-nodes-1-dense	0.1500	70.30	0.6546
2-conv-32-nodes-1-dense	0.1564	72.59	0.6777
2-conv-64-nodes-1-dense	0.1205	72.59	0.6755
2-conv-128-nodes-1-dense	0.0798	74.11	0.6933
3-conv-32-nodes-1-dense	0.1934	69.80	0.6510
3-conv-64-nodes-1-dense	0.1073	72.59	0.6763
3-conv-128-nodes-1-dense	0.0293	73.60	0.6880
1-conv-32-nodes-2-dense	0.6403	52.03	0.4907
1-conv-64-nodes-2-dense	0.2509	63.45	0.5944
1-conv-128-nodes-2-dense	0.6100	56.85	0.5290
2-conv-32-nodes-2-dense	0.0739	71.32	0.6603
2-conv-64-nodes-2-dense	0.1179	74.37	0.6979
2-conv-128-nodes-2-dense	0.0183	71.57	0.6689
3-conv-32-nodes-2-dense	0.2438	68.02	0.6375
3-conv-64-nodes-2-dense	0.0351	73.10	0.6821
3-conv-128-nodes-2-dense	0.0790	73.10	0.6854

No clear trend can be identified from Table IV. However, we can notice that models that perform the best are in general the ones with a higher number convolutional layers, nodes and dense layers.

To see how the number of channels in the dense layer affected the performance of the CNN, we then chose the 5 best results, that is the models that gave the highest accuracy on the test set, and investigated the effect of the number of channels in the dense layers as shown in Table B1 in Appendix B. Again, there are no clear trends that can be identified. For each model, the best result is either obtained with 256, 512 or 1024 channels in the dense layers. The only thing we can say is that the best result for a given model is generally obtained with a number of channels in the dense layers higher than 128.

We also tested the effect of the dropout on those same 5 best models from table IV. The results are shown in table B2 in Appendix B. Again, no clear trend can be identified in this table.

Out of all the results, the best model identified was the one with 3 convolutional layers, 128 nodes, 2 dense layers, and 512 channels in the dense layers with a test accuracy of 77.16% and an F1 score of 0.7285. The confusion matrix for this model is shown in Figure 6. This shows that all the test scans which showed not tumor were correctly

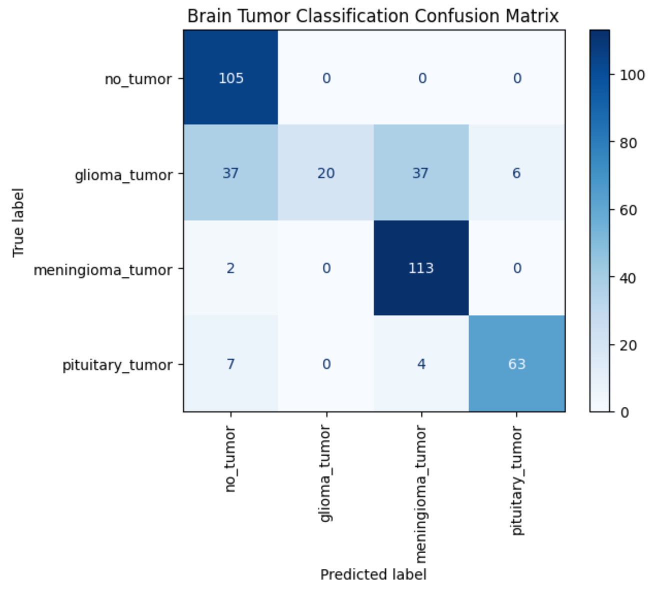

Figure 6. Confusion matrix for the overall best CNN model (3 convolutional layers, 128 nodes, 2 dense layers, 512 channels in the dense layers, number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32, dropout = 0.4

classified, while most of the test scans belonging to the glioma class were misclassified. Only a few of the scans belonging to the other classes were misclassified.

From all the results obtained, we chose the best performing model (highest accuracy and highest F1 score), and investigated the effect that data processing has on the end result. We first added two transforms to the data augmentation of the training set, and trained the best model with this data, and then removed all the transforms except ToTensor and Normalize and trained the model again. The results are shown in Table B3 in Appendix B. We can see that the more transforms we add in the training data augmentation, the higher accuracy the model obtains. However, this trend is not mirrored in the F1 score.

Lastly, we tested the effect of changing the training set size on the performance. The results are shown in table V. Increasing the size of the data set improves the accuracy and the F1 score, but only up to a certain point. We can also notice that the loss decreases as the training test size increases.

Table V. Investigating the effect of the size of the training dataset on the performance of the best model (3 convolutional layers, 128 nodes, 2 dense layers with 512 channels in each dense layer, dropout=0.4, number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32)

Training set size	**Loss**	**Accuracy %**	**F1 score**
×1	0.2558	64.97	0.6097
×6 (original setup)	0.0229	77.16	0.7285
×10	0.0147	75.63	0.7146
×15	0.0089	74.37	0.6976

IV. DISCUSSION

The ANN served as a baseline model, employing fully connected layers with varying activation functions, while the CNN leveraged convolutional and pooling layers to exploit spatial features in the MRI scans. As expected, the CNN demonstrated stronger generalization and higher test performance compared to the ANN, highlighting the importance of spatial feature extraction in medical imaging tasks.

A. ANN

The experimental results indicate that the choice of activation function and learning rate significantly impacts the model's performance. Across all configurations and learning rates, the Artificial Neural Network (ANN) utilizing the sigmoid activation function consistently yielded the lowest F_1 scores. In contrast, the Leaky ReLU activation function achieved the highest validation F_1 score of 75.1 % when configured with two hidden layers (128 and 64 nodes, respectively) and a learning rate of 10^{-3} (Table II). Models utilizing the ReLU function performed slightly lower, reaching a peak F_1 score of 73.7 % with the same layer configuration but a smaller learning rate of 10^{-4}.

A more granular analysis of the ReLU model's performance (Figure A5) reveals that it is most effective at classifying pituitary tumors, achieving a recall of 0.66. While the model showed reasonable performance for "no tumor" and "meningioma" classes, the non-diagonal elements of the confusion matrix highlight significant misclassifications.

Of particular concern is the model's performance regarding glioma tumors, where recall dropped to a mere 17 %. The majority of gliomas were incorrectly classified as "no tumor." This high rate of false negatives is a critical weakness; while meningiomas are generally benign, gliomas are malignant and require urgent intervention. Furthermore, the model exhibited "confusion" between meningiomas and both the "no tumor" and "pituitary" classes, suggesting that the feature representations learned by the ANN were not sufficiently discriminative for these categories.

As shown in Table III, the best-performing ANN (Leaky ReLU, 128-64 nodes, 10^{-3} learning rate) achieved a final test F_1 score of 0.371. This is substantially lower than the performance observed in the Convolutional Neural Network (CNN) models. The disparity in results suggests that a standard ANN is less suited for this image classification task. This outcome is expected, as ANNs require the input to be flattened, thereby failing to preserve the spatial hierarchies and local relationships inherent in medical imaging—features that CNNs are specifically designed to extract through convolutional layers.

B. CNN

The comparative evaluation of the different CNN architectures presented in Table IV reveals several important trends regarding how model depth, convolutional complexity, and dense layer configuration influence classification performance on the MRI dataset. Overall, the results show that increases in architectural depth, both in terms of convolutional layers and filter counts, tend to improve performance up to a certain point. Models with only a single convolutional layer generally produced the weakest results, with accuracy values often below 65% and F1 scores below 0.60. This suggests that a single convolutional block is insufficient for capturing the spatial and textural complexity typically present in MRI images. In contrast, architectures with two or three convolutional layers consistently achieved higher accuracy and F1 scores, indicating that deeper feature extraction is beneficial for distinguishing subtle MRI patterns. Similarly, increasing the number of filters from 32 to 64 or 128 had a clear positive effect. The models using 64 or 128 filters demonstrated the strongest performance, supporting the idea that richer feature maps improve representational capacity and consequently classification accuracy. The number of dense layers also influenced performance. Networks with one dense layer frequently outperformed those with zero or two dense layers. Adding a single dense layer appears to strike an effective balance, enabling the model to integrate high-level features without introducing excessive parameters that lead to overfitting, an issue that may explain the performance decline observed in some architectures with two dense layers.

Among all tested configurations, the best-performing models were those combining 2–3 convolutional layers with 64–128 filters, paired with one dense layer. These architectures achieved accuracies in the range of 73–74% and F1 scores close to 0.69, indicating robust performance relative to the other configurations. In particular, models such as 2-conv-64-nodes-2-dense, 2-conv-128-nodes-1-dense, and 3-conv-128-nodes-1-dense emerged as the strongest, suggesting that moderate increases in depth and number of filters yield meaningful gains without incurring substantial overfitting.

Overall, the results in Table IV demonstrate that both model depth and filter complexity are critical to effectively modeling MRI data, and that moderate network complexity leads to the best generalization in this setting. These findings align with broader trends reported in the literature, where deeper yet carefully regularized CNNs tend to perform best on medical imaging tasks [19]. In addition, the findings from changing the number of channels in the dense layer in Table B1 shows that increasing the channels to 512 gives an overall highest accuracy, with CNN architecture of 3 convolutional layers of 128 nodes and 2 dense layers. The effect of changing the number of channels in the dense layer depends on the model, there is not one consistent trend, but it seems that dense layers with very few channels (32–64) consistently leads to reduced accuracy and F1-scores, indicating underfitting. However, increasing the number of channels greatly (1024) does not result in consistent performance improvement. The best results were generally obtained with dense layer sizes of 256 and 512 channels. Both accuracy and F1-score remained within a narrow range across those models. These models seem to effectively combine convolutional features without introducing excessive model complexity, resulting in the best performances.

To explore how the dropout impacts the performance of the 5 best models we tuned the size in the range $0 - 0.6$. Table B2 shows the results of this experiment. The findings indicate that dropout has a nuanced effect on CNN performance. While extremely high dropout rates tend to hinder performance, moderate dropout (around 0.4) tend to give the best balance between preventing overfitting and maintaining high values in accuracy and F1 score. These results underscore that optimal dropout depends both on model depth and the number of filters.

The results in Table V illustrate the impact of increasing the size of the training dataset on the performance of the best CNN model identified in earlier experiments. A clear improvement is observed when moving from the smallest dataset ($\times 1$) to the dataset used in the original setup ($\times 6$). Specifically, accuracy rises from 64.97% to 77.16%, and the F1 score increases from 0.6097 to 0.7285, indicating that the model benefits substantially from additional training data and is better able to learn discriminative MRI features.

However, beyond this point, the performance gains begin to plateau. When the dataset size is increased further to $\times 10$ and $\times 15$, the accuracy remains high but shows slight fluctuations: 75.63% for $\times 10$ and 74.37% for $\times 15$. The F1 score follows a similar trend, decreasing slightly from its peak at $\times 6$. Although the training loss continues to decrease steadily, with the $\times 15$ dataset achieving the lowest loss (0.0089), this reduction does not translate into higher accuracy or F1 performance on the evaluation set. This suggests that once a sufficiently large dataset is reached, adding more samples yields diminishing returns, and may even lead to mild overfitting or over-smoothing of the learned feature representations.

Overall, the results in Table 5 demonstrate that expanding the dataset significantly enhances model performance up to a moderate scale, but excessive enlargement does not necessarily lead to further improvements. These findings align with common observations in deep learning for medical imaging: while adequate data quantity is crucial for model generalization, there exists a practical threshold beyond which additional data contributes less meaningfully to classification performance [20].

Like in the ANN investigation, the CNN study showed that the glioma tumor class consistently showed a much higher number of misclassified results compared to the other classes. We are not medical experts, so we cannot know if the dataset was flawed. If it was not, it may be that the glioma tumor is harder to distinguish than the other tumor types. It is particularly important to be able to classify this type of tumor correctly, as it is a very dangerous malignant form of tumor.

A scan with no tumor will be visually very distinct from the other classes, so that is why the CNN performs really well in classifying the no tumor scans. Furthermore, meningioma and pituitary tumors are relatively well-localised in the brain and show a strong contrast with the rest of the brain, as shown in Figure 1. However, glioma tumors can occur anywhere in the brain and have unclear, hard to distinguish borders. This inconsistency makes it more difficult to train a CNN to recognize this type of tumor.

C. Limitations and future work

A limitation of the current study is the use of a fixed training duration of 15 epochs across all CNN architectures. While this approach ensures consistency when comparing model configurations, it does not account for differences in convergence behavior between architectures. Some models may require more epochs to fully learn relevant MRI features, while others may reach optimal performance earlier and begin to overfit if training continues unnecessarily. Implementing early stopping in our code, where training stops when the performance stops improving could provide a more efficient training process.

Despite the insights gained from this study, several limitations remain that offer avenues for future research. While the ANN achieved peak performance with specific learning rates, the static training approach may have led to sub-optimal generalization. Future work should incorporate early stopping or similar adaptive training strategies to monitor validation loss and terminate training at the point of peak generalization, thereby preventing overfitting.

Another limitation of this study is the limited use of data augmentation techniques for the MRI images. Although data augmentation was applied during training, the augmentation strategy was limited to basic geometric transformations, including random rotations and horizontal and vertical flips. These techniques increase data variability and help reduce overfitting; however, they do not fully capture the range of variations present in real-world MRI data. More advanced and domain-specific augmentation methods were not explored in this study. For example, intensity-based transformations, elastic deformations, bias field simulation, or noise injection, commonly used in medical imaging, could further improve model robustness. In addition, the impact of individual augmentation techniques was not evaluated separately, as the primary focus of this work was on comparing CNN architectures under consistent training conditions. Future work could include a more systematic investigation of augmentation strategies, as well as the use of synthetic data generation techniques, to further enhance generalization performance on unseen MRI images.

Beyond augmentation, the study is also limited by its reliance on unimodal MRI data. Further work could make use of T2 enhanced images too. This may improve the accuracy of the models. While the current model achieved [mention performance] using a single MRI sequence, clinical diagnosis often leverages multimodality imaging—combining T1, T1-contrast enhanced (T1ce), T2, and Fluid Attenuated Inversion Recovery (FLAIR) scans. Each modality reveals distinct pathological features: for instance, T1ce is superior for identifying the tumor core, while FLAIR is essential for detecting surrounding edema. Future research should explore the integration of multimodal inputs, as the fusion of these complementary spatial features would likely mitigate the current model's confusion between tumor classes and significantly improve the detection of malignant gliomas.

V. CONCLUSION

In this report we showed that CNNs outperform ANNs for classifying brain tumors into glioma tumor, meningioma tumor, no tumor and pituitary tumor.

The ANN model achieved an accuracy of 69.5 % and F_1 score of 67.1 % on the validation set. On the test set the performance was lower with an accuracy of 38.8 % and an F_1 score of 37.1 %. The CNN model with 3 convolutional layers, 128 nodes, 2 dense layers, and 512 channels in the dense layers achieved both the highest accuracy and the highest F1 score (with batch size 32, Adam optimizer, a learning rate of 0.0001, dropout 0.4 and 15 epochs). The accuracy was 77.16 % and the F1 score was 0.7285.

The findings in this report help highlight the potential of AI in the biomedical field, however, the models need to be improved a lot before they can truly be performant enough to be used in the field. Ways to improve would be to train with a larger training set, more transforms in the data augmentation, and be able to test a larger number of hyperparameters combinations.

[1] S. Bhuvaji, A. Kadam, P. Bhumkar, and et. al., "Brain tumor classification (MRI)," 2025. [Online]. Available: https://www.kaggle.com/dsv/12745533

[2] Brain Tumor Organization, "Brain Tumor Facts," https://braintumor.org/brain-tumors/about-brain-tumors/brain-tumor-facts/, n.d., accessed: [26.12.2025].

[3] D. N. Louis, A. Perry, P. Wesseling, D. J. Brat, I. A. Cree, P. Figarella-Branger, C. Hawkins, H.-K. Ng, S. M. Pfister, G. Reifenberger *et al.*, "The 2021 WHO classification of tumors of the central nervous system: a summary," *Neuro-oncology*, vol. 23, no. 8, pp. 1231–1251, 2021.

[4] A. Kadam, S. Bhuvaji, and S. Deshpande, "Brain tumor classification using deep learning algorithms," *International Journal for Research in Applied Science Engineering Technology (IJRASET)*, 2021. [Online]. Available: https://www.ijraset.com/best-journal/brain-tumor-classification-using-deep-learning-algorithms

[5] M. Yildirim, E. Cengil, Y. Eroglu, and A. Cinar, "Detection and classification of glioma, meningioma, pituitary tumor, and normal in brain magnetic resonance imaging using deep learning-based hybrid model," *Iran Journal of Computer Science*, vol. 6, no. 4, pp. 455–464, Dec. 2023. [Online]. Available: https://link.springer.com/10.1007/s42044-023-00139-8

[6] A. A. Awan, "A complete guide to data augmentation," *Datacamp*, 2024. [Online]. Available: https://www.datacamp.com/tutorial/complete-guide-data-augmentation

[7] S. Kumar, U. Esiowu, and P. Asiamah, "Enhancing image classification with augmentation: Data augmentation techniques for improved image classification," *University of Calgary*, 2025. [Online]. Available: https://arxiv.org/html/2502.18691v1#abstract

[8] P. D. Rosero-Montalvo, "Data augmentation with tf and pytorch." [Online]. Available: https://www.paulrosero-montalvo.com/Tutorial_DataAugmentation_TF_PyTorch/

[9] T. maintainers and contributors, "Torchvision: Pytorch's computer vision library," https://github.com/pytorch/vision, 2016.

[10] G. for geeks, "Data preprocessing in pytorch," 2024. [Online]. Available: https://www.geeksforgeeks.org/deep-learning/data-preprocessing-in-pytorch/

[11] M. Hjorth-Jensen, *Computational Physics Lecture Notes 2025.* University of Oslo, Norway: Department of Physics, 2025. [Online]. Available: https://github.com/CompPhysics/MachineLearning/blob/master/doc/LectureNotes/week44.ipynb

[12] J. C. Luna, "Pytorch cnn tutorial: Build and train convolutional neural networks in python," *Datacamp*, 2025. [Online]. Available: https://www.datacamp.com/tutorial/pytorch-cnn-tutorial

[13] N. Fatima, "Building your first convolutional neural network with pytorch: A beginner's guide," *Medium*, 2025. [Online]. Available: https://medium.com/@noorfatimaafzalbutt/building-your-first-convolutional-neural-network-with-pytorch-a-beginners-guide-12d341473ecb

[14] J. Ansel, E. Yang, H. He, and et. al., "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24).* ACM, 2024. [Online]. Available: https://docs.pytorch.org/assets/pytorch2-2.pdf

[15] M. L. Waskom, "seaborn: statistical data visualization," *Journal of Open Source Software*, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://doi.org/10.21105/joss.03021

[16] G. for geeks, "What is cross-entropy loss function?" 2025. [Online]. Available: https://www.geeksforgeeks.org/machine-learning/what-is-cross-entropy-loss-function/

[17] P. Kashyap, "Understanding precision, recall, and f1 score metrics," *Medium*, 2024. [Online]. Available: https://medium.com/@piyushkashyap045/understanding-precision-recall-and-f1-score-metrics-ea219b908093

[18] M. Abadi, A. Agarwal, P. Barham, and et. al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[19] R. B. Z. A. Md Ashik Khan, "Comparative analysis of resource-efficient cnn architectures for brain tumor classification," *arXiv preprint aarXiv:2411.15596*, 2024. [Online]. Available: https://arxiv.org/abs/2411.15596?utm_source=chatgpt.com

[20] J. Hestness, S. Narang, N. Ardalani, and et. al., "Deep learning scaling is predictable, empirically," *CoRR*, vol. abs/1712.00409, 2017. [Online]. Available: http://arxiv.org/abs/1712.00409

Appendix A: ANN

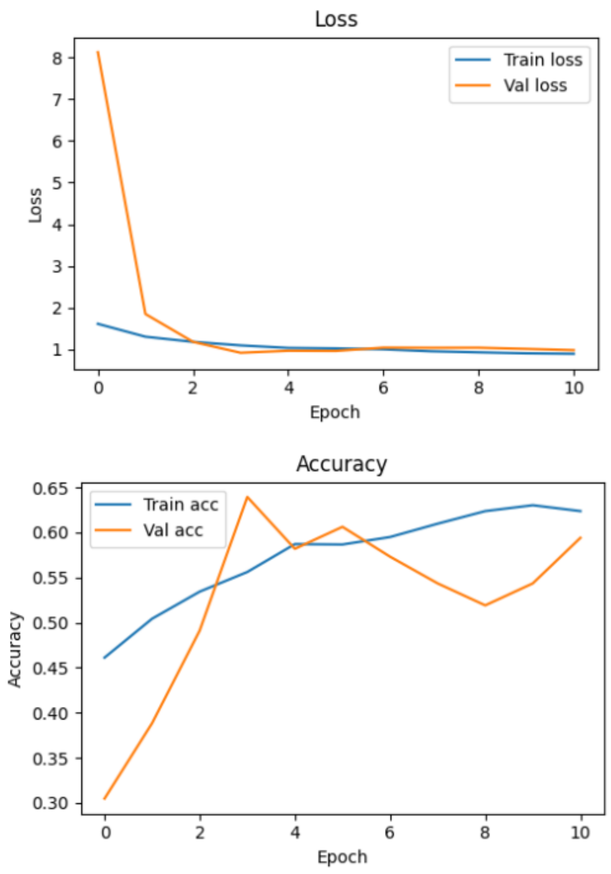

Figure A1. Training History for the ReLU-activated Artificial Neural Network (ANN) using a learning rate of 0.001 and a two-layer 128-64 neuron layer. The first 10 epochs of the training is illustrated, where the top plot displays the Training (blue) and Validation (orange) loss, and the bottom plot shows the corresponding Training (blue) and Validation (orange) accuracy.

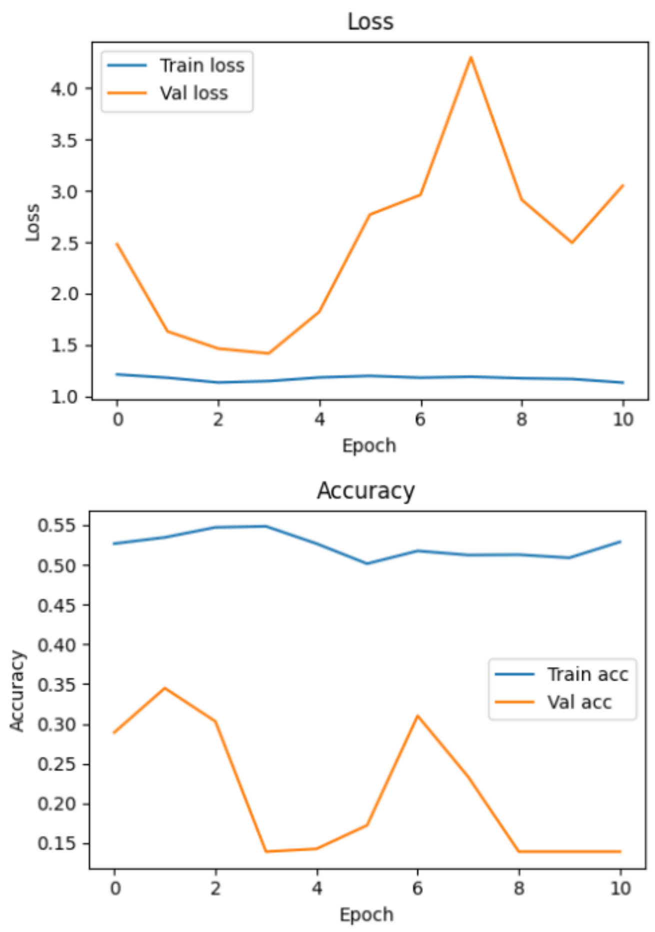

Figure A2. Training History for the Sigmoid-activated Artificial Neural Network (ANN) using a learning rate of 0.0001 and a single 256-neuron layer. The 10-epoch training run is illustrated, where the left plot displays the Training (blue) and Validation (orange) loss, and the right plot shows the corresponding Training (blue) and Validation (orange) accuracy.

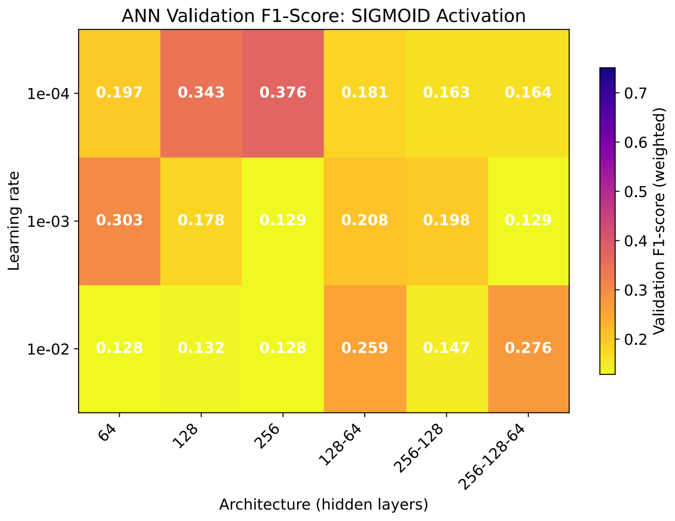

Figure A3. Heatmap illustrating the validation F_1 scores for Artificial Neural Networks (ANNs) using the Sigmoid activation function. The scores are presented across three learning rates (10^{-2}, 10^{-3}, 10^{-4}) and the six network architectures listed in Table I. The color scale is consistent across all activation function heatmaps for direct comparison.

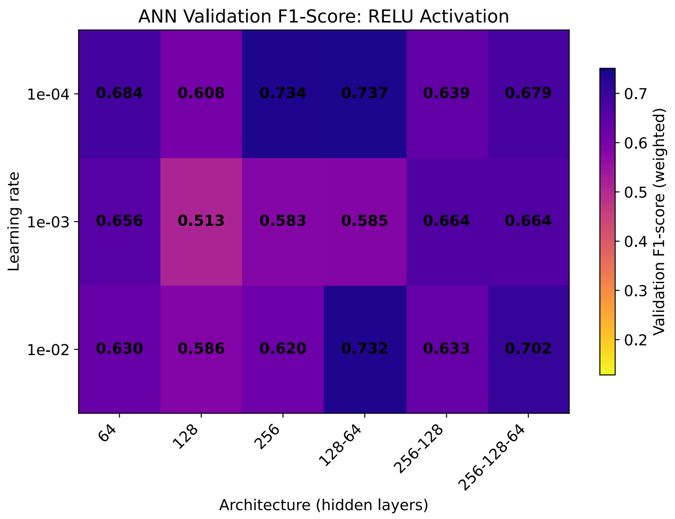

Figure A4. Heatmap illustrating the validation F_1 scores for Artificial Neural Networks (ANNs) using the ReLU activation function. The scores are presented across three learning rates (10^{-2}, 10^{-3}, 10^{-4}) and the six network architectures listed in Table I. The color scale is consistent across all activation function heatmaps for direct comparison.

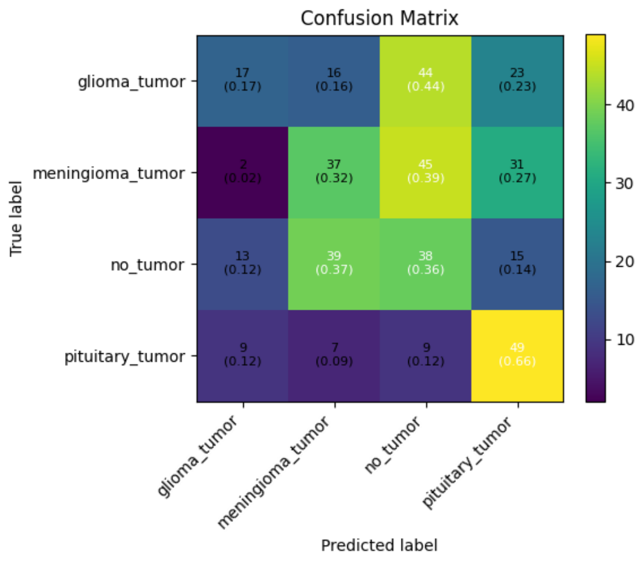

Figure A5. Confusion Matrix for the Artificial Neural Network (ANN) using the ReLU activation function. This matrix visualizes the classification results achieved with the optimal hyperparameters listed in Table II, which correspond to a Macro F_1 score of 0.659 on the validation set. Values are presented as raw counts with the corresponding class recall (True Positive Rate) in parentheses.

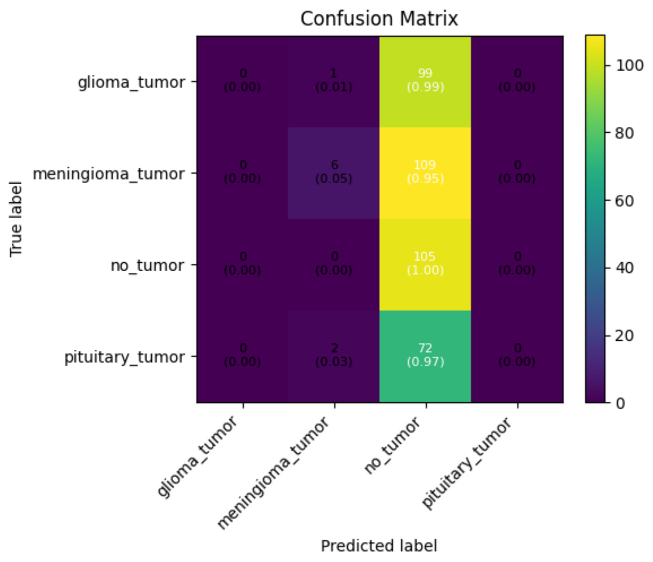

Figure A6. Confusion Matrix for the Artificial Neural Network (ANN) using the Sigmoid activation function. This matrix visualizes the classification results achieved with the optimal hyperparameters listed in Table II, which correspond to a Macro F_1 score of 0.0368 on the validation set. Values are presented as raw counts with the corresponding class recall (True Positive Rate) in parentheses.

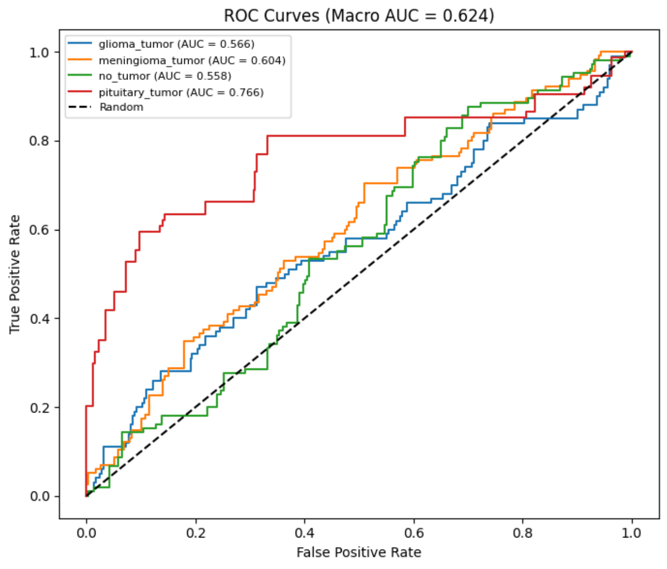

Figure A7. Receiver Operating Characteristic (ROC) Curves for the ReLU-activated ANN with learning rate and architecture that was presented in Table II. The Area Under the Curve (AUC) is provided for each class in the legend, showing pituitary tumor (AUC = 0.766) as the most separable class, and no tumor (AUC = 0.558) as the least separable. The overall model performance is summarized by the Macro AUC of 0.624. The dashed line corresponds to random guessing.

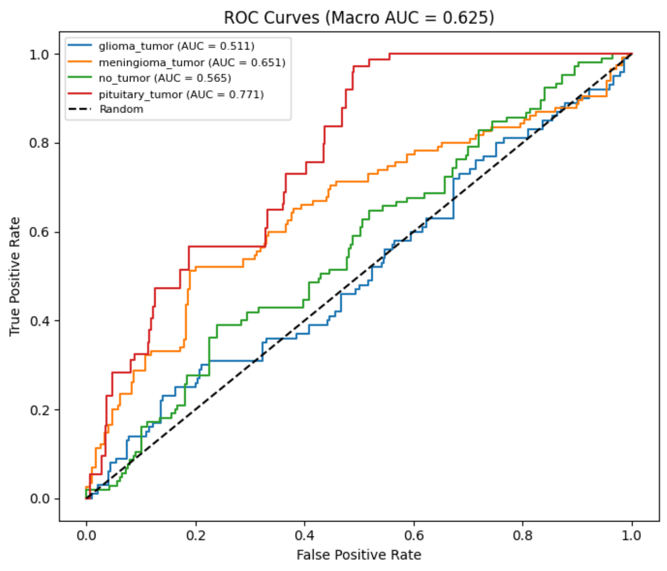

Figure A8. Receiver Operating Characteristic (ROC) Curves for the Sigmoid-activated ANN with learning rate and architecture that was presented in Table II. The plot illustrates the True Positive Rate (TPR) versus the False Positive Rate (FPR) for each of the four brain tumor classes. The Area Under the Curve (AUC) is provided for each class in the legend, showing pituitary tumor (AUC = 0.771) as the most separable class, and glioma tumor (AUC = 0.511) as the least separable. The overall model performance is summarized by the Macro AUC of 0.625.

Appendix B: CNN

Table B1. Investigation of the effect of the number of channels in the dense layer on the performance of the previously best models (number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32, dropout=0.4)

Model	**Channels in dense layers**	**Loss**	**Accuracy %**	**F1 score**
2-conv-128-nodes -1-dense	1024	0.0203	73.10	0.6824
	512	0.0337	75.38	0.7053
	256	0.0440	74.37	0.6937
	128	0.0798	74.11	0.6933
	64	0.1425	72.08	0.6743
	32	0.1990	69.80	0.6512
3-conv-128-nodes -1-dense	1024	0.0182	74.37	0.6972
	512	0.0247	74.11	0.6921
	256	0.0391	74.37	0.6938
	128	0.0293	73.60	0.6880
	64	0.1675	71.32	0.6636
	32	0.2327	65.74	0.6098
2-conv-64-nodes -2-dense	1024	0.0220	74.11	0.6937
	512	0.0269	73.35	0.6888
	256	0.0600	75.38	0.7063
	128	0.1179	74.37	0.6979
	64	0.1711	71.32	0.6642
	32	0.2842	65.48	0.6064
3-conv-64-nodes -2-dense	1024	0.0210	74.62	0.6972
	512	0.0288	74.37	0.6956
	256	0.0611	73.86	0.6909
	128	0.0351	73.10	0.6821
	64	0.1956	69.04	0.6488
	32	0.3482	63.96	0.5974
3-conv-128-nodes -2-dense	1024	0.0159	74.62	0.6964
	512	0.0229	77.16	0.7285
	256	0.0352	74.62	0.6997
	128	0.0790	73.10	0.6854
	64	0.1175	70.81	0.6575
	32	0.2032	63.45	0.5968

Table B2. Investigation of the effect of the dropout on the performance of the previously best models on the test set(number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32, number of channels in the dense layers = 128)

Model	**Dropout**	**Loss**	**Accuracy %**	**F1 score**
2-conv-128-nodes -1-dense	0.6	0.0324	73.86	0.6936
	0.4	0.0798	74.11	0.6933
	0.2	0.0351	75.38	0.7088
	0.0	0.0201	74.87	0.7057
3-conv-128-nodes -1-dense	0.6	0.0072	74.87	0.7049
	0.4	0.0293	73.60	0.6880
	0.2	0.0043	72.59	0.6765
	0.0	0.0167	73.35	0.6883
2-conv-64-nodes -2-dense	0.6	0.0155	74.62	0.6980
	0.4	0.1179	74.37	0.6979
	0.2	0.0075	71.57	0.6693
	0.0	0.0182	72.59	0.6823
3-conv-64-nodes -2-dense	0.6	0.1895	70.56	0.6578
	0.4	0.0351	73.10	0.6821
	0.2	0.0623	70.81	0.6590
	0.0	0.0300	73.10	0.6829
3-conv-128-nodes -2-dense	0.6	0.1193	71.57	0.6672
	0.4	0.0790	73.10	0.6854
	0.2	0.0409	72.84	0.6799
	0.0	0.0215	73.60	0.6858

Using the best model (3 convolutional layers, 128 nodes, 2 dense layers with 512 channels in each dense layer, dropout=0.4) and modifying the training dataset is shown in Table B3.

Table B3. Investigation of the effect of pre-processing of the training data on the performance of the best model on the test set(3 convolutional layers, 128 nodes, 2 dense layers with 512 channels in each dense layer, dropout=0.4, number of epochs = 15, optimizer = Adam, learning rate = 10^{-4}, batch size = 32, size of dataset = $\times 6$)

Training set	**Loss**	**Accuracy %**	**F1 score**
Original setup	0.0229	77.16	0.7285
Added transforms RandomResizedCrop and ColorJitter	0.0500	77.92	0.7283
Removed transforms except ToTensor and Normalize	0.0075	75.63	0.7179