AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

Faculty of Computer Science

Brain tumor classification based on magnetic resonance imaging using convolutional neural networks (CNN)

Authors: *Milena Biernacka, Agata Dratwa, Patrycja Lewczuk, Kornelia Susz*
Field of study: Computer Science - Data Science
Lecturer: *Cemal Koba*

Cracow, 2024

Contents

1 Introduction

Brain tumor represents a complex and challenging medical condition. The timely and accurate diagnosis of brain tumors is crucial for effective treatment planning. In this project, we aim to leverage the capabilities of neural networks to contribute to classification of brain tumor based on magnetic resonance imaging (MRI) scans. Additionally, we tried to explore the impact of image quality on the accuracy of classification results by blurring some examples.

1.1 Problem

The use of Magnetic Resonance Imaging (MRI) in medical diagnostics has become a cornerstone for identifying and characterizing brain tumors. However, the quality of MRI scans can vary significantly due to factors such as differences in scanner hardware, patient movement, and imaging protocols. These variations present a substantial challenge for automated classification systems, as most machine learning models rely on consistent data quality to maintain high accuracy. When models trained on high-quality scans are presented with lower-quality images, their performance can degrade, leading to potential misclassifications. This issue is compounded when considering the need for robust algorithms capable of analyzing scans from diverse sources with varying levels of clarity and detail.

In clinical settings, the stakes are high, as incorrect tumor classification can lead to inappropriate treatment plans. Furthermore, the ability of a model to generalize across different qualities of scans is crucial for its deployment in real-world scenarios, where it must remain reliable despite the inherent variability in image quality. This project addresses the pressing need to develop machine learning models that are not only accurate in tumor classification but are also resilient to changes in image quality. By systematically modifying the resolution and introducing blur to simulate different quality levels, we investigate the robustness of convolutional neural networks (CNNs) in maintaining classification performance.

In our study, we implemented two distinct data modification techniques to assess the robustness of convolutional neural networks (CNNs) when classifying MRI brain scans of varying quality:

- Gaussian blur application:

We simulated the effect of lower image sharpness, often encountered in clinical practice due to factors such as suboptimal scanner settings or patient motion, by applying a Gaussian blur to the MRI images. This process reduces image detail and mimics the challenges a model might face when analyzing data from different-quality scanners.

- Resolution modification:

To emulate the varying resolutions of MRI scans that arise from different imaging equipment, we downsampled the images to a lower pixel density and then upsampled them back to their original size. This procedure inherently degrades the image quality by losing high-resolution details, testing the model's ability to identify and classify tumors under less-than-ideal imaging conditions.

These data manipulations are crucial for evaluating the model's diagnostic accuracy across a realistic spectrum of image qualities found in medical settings.

1.2 Dataset

We utilized the Brain Tumor Classification MRI Dataset obtained from the Kaggle platform. The dataset comprises MRI data of humans' brains that are categorized into following labels:

- Glioma tumor,
- Meningioma tumor,
- Pituitary tumor,
- No tumor.

The dataset is already split into training and test subsets. The training subset consists of 2870 MRI images while, the test subset consists of 394 MRI images. The distribution of each class in both subsets is illustrated in Figure 1 for training subset and Figure 2 for test subset. Notably, the distribution across classes is relatively balanced, with no dominating class, mitigating concerns of dealing with an imbalanced dataset.

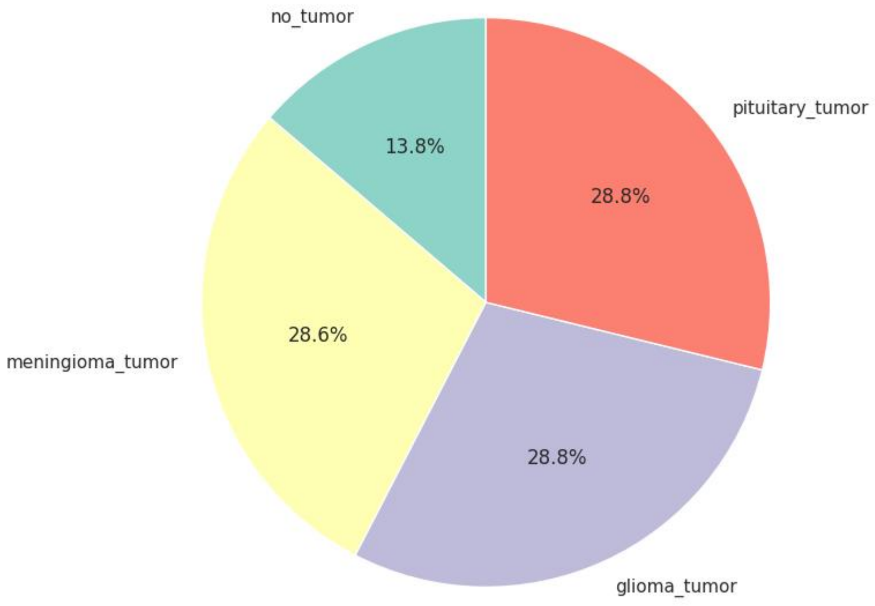


Figure 1 Distribution of classes in training subset

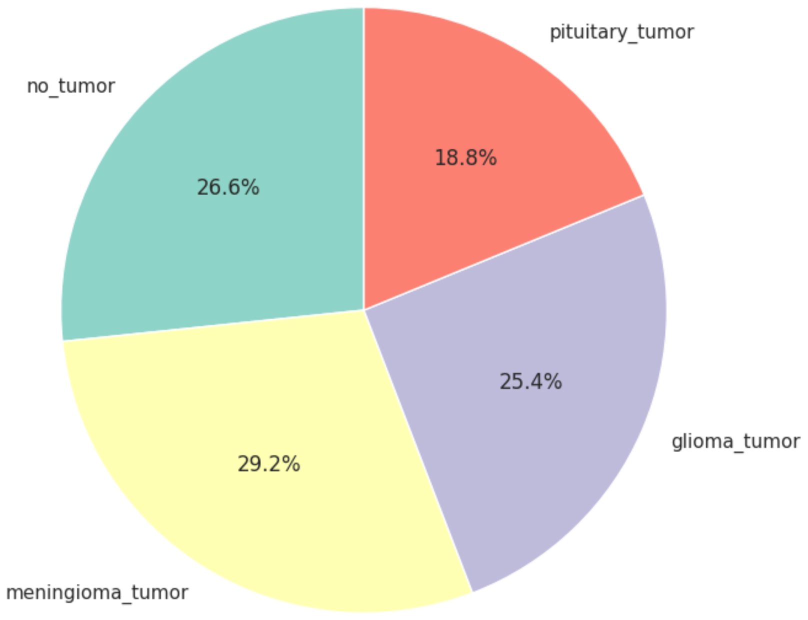


Figure 2 Distribution of classes in test subset

2 Methods

To perform classification, we used a Convolutional Neural Network (CNN) which is a specialized type of neural network designed especially for processing and analyzing visual data, specifically images. CNNs have proven highly effective in computer vision tasks like image recognition, object detection, and other visual perception problems. CNNs have three main types of layers:

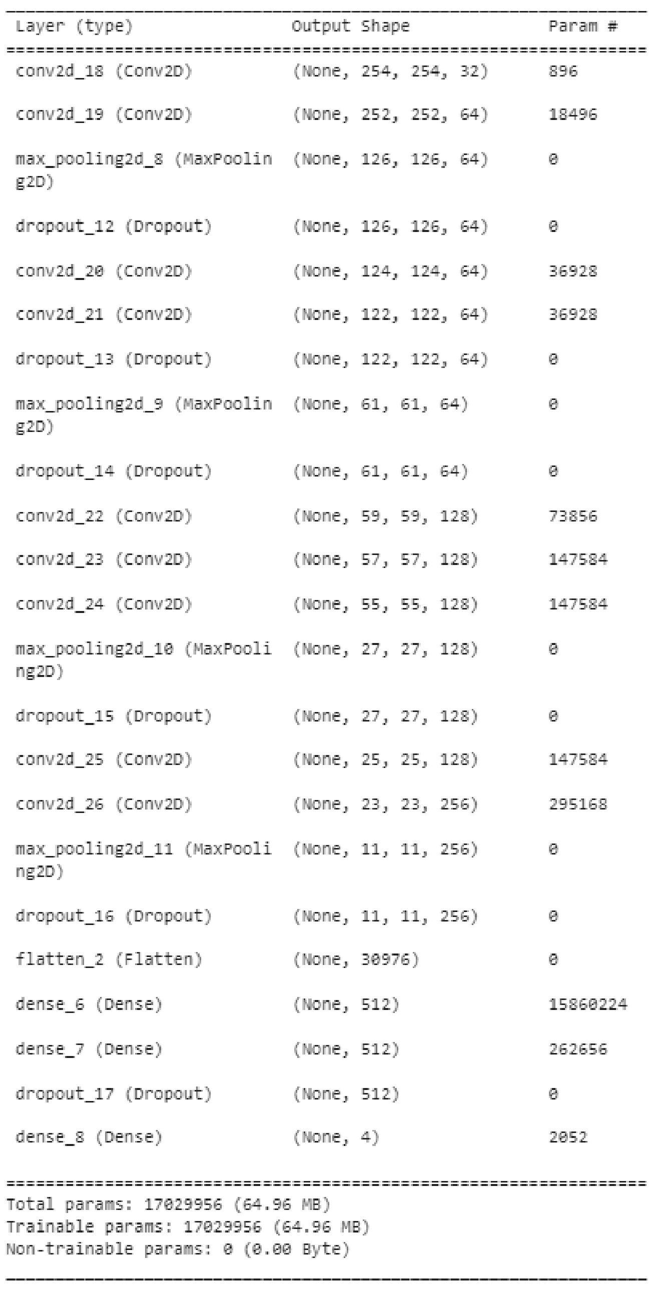
- Convolutional layer,
- Pooling layer,
- Fully-connected layer.

With each layer, CNN increases in its complexity, identifying larger portions of the image. The initial layers focus on simple futures, such as colors and edges. Subsequent layers start to recognize larger elements until the model is able to identify the intended object. The convolutional layer is the first layer, while the fully-connected layer is the final one. Between them there can be multiple convolutional layers or pooling layers.

In CNNs there can also be layers called Dropout. They are often integrated to mitigate overfitting. During training, random neurons are “dropped out” or deactivated, preventing the network from relying too heavily on specific features.

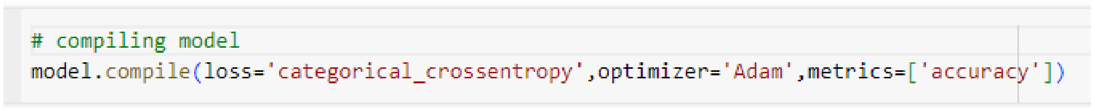
In our project each Convolutional layer employs the Rectified Linear Unit (ReLU) as an activation function. It is a commonly used activation function in CNNs due to its ability to introduce non-linearity into the model, allowing the network to learn complex relationships in the data. In the final layer we used the softmax activation function. It is a common choice because of its ability to produce probability distributions over multiple classes.

We employed four distinct approaches to the dataset in our project. Initially, we conducted multilabel classification on the original dataset. Subsequently, we compared these results with those obtained through binary classification. Following that, we investigated the impact of blurring on MRI images and assessed its effects on the results. Notably, the blurring process was individually applied to both the training and test dataset. Listing 1 illustrates the layers that were employed in our project using Keras Sequential API. The model is progressively extracting features through convolutional layers, reducing spatial dimensions with pooling, preventing overfitting with dropout, and making final predictions through fully connected layers. Listing 2 presents the code for compiling model.



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_18 (Conv2D)          (None, 254, 254, 32)      896

 conv2d_19 (Conv2D)          (None, 252, 252, 64)      18496

 max_pooling2d_8 (MaxPoolin  (None, 126, 126, 64)      0
 g2D)

 dropout_12 (Dropout)        (None, 126, 126, 64)      0

 conv2d_20 (Conv2D)          (None, 124, 124, 64)      36928

 conv2d_21 (Conv2D)          (None, 122, 122, 64)      36928

 dropout_13 (Dropout)        (None, 122, 122, 64)      0

 max_pooling2d_9 (MaxPoolin  (None, 61, 61, 64)        0
 g2D)

 dropout_14 (Dropout)        (None, 61, 61, 64)        0

 conv2d_22 (Conv2D)          (None, 59, 59, 128)       73856

 conv2d_23 (Conv2D)          (None, 57, 57, 128)       147584

 conv2d_24 (Conv2D)          (None, 55, 55, 128)       147584

 max_pooling2d_10 (MaxPooli  (None, 27, 27, 128)       0
 ng2D)

 dropout_15 (Dropout)        (None, 27, 27, 128)       0

 conv2d_25 (Conv2D)          (None, 25, 25, 128)       147584

 conv2d_26 (Conv2D)          (None, 23, 23, 256)       295168

 max_pooling2d_11 (MaxPooli  (None, 11, 11, 256)       0
 ng2D)

 dropout_16 (Dropout)        (None, 11, 11, 256)       0

 flatten_2 (Flatten)         (None, 30976)             0

 dense_6 (Dense)             (None, 512)               15860224

 dense_7 (Dense)             (None, 512)               262656

 dropout_17 (Dropout)        (None, 512)               0

 dense_8 (Dense)             (None, 4)                 2052

=================================================================
Total params: 17029956 (64.96 MB)
Trainable params: 17029956 (64.96 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

Listing 1 CNN layers used in project

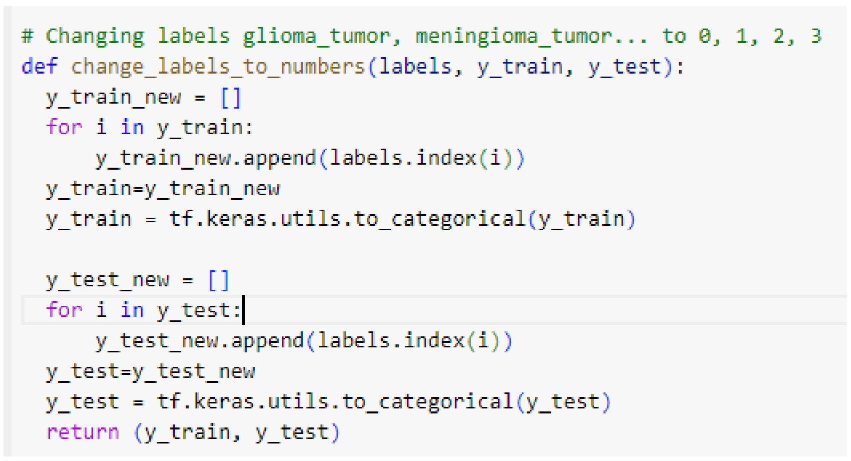


```
# compiling model
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

Listing 2 Code to compile previously defined model

2.1 Multilabel classification on original dataset

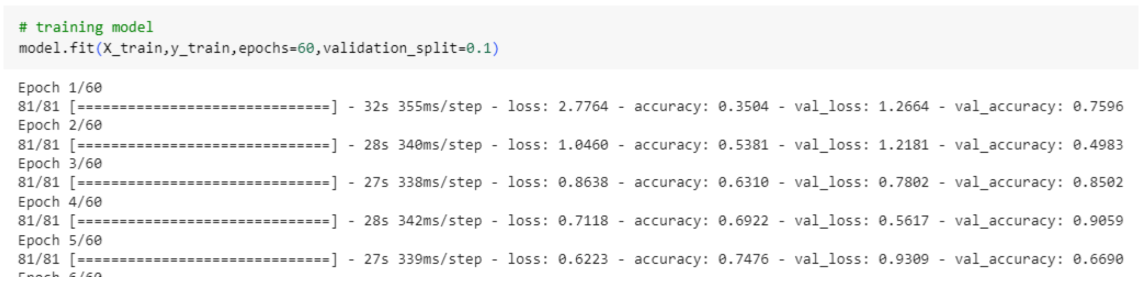
In this case, we trained the model on the original dataset. Prior to training, it was necessary to convert the categorical class labels into numeric representations as shown in Listing 3.



```
# Changing labels glioma_tumor, meningioma_tumor... to 0, 1, 2, 3
def change_labels_to_numbers(labels, y_train, y_test):
  y_train_new = []
  for i in y_train:
      y_train_new.append(labels.index(i))
  y_train=y_train_new
  y_train = tf.keras.utils.to_categorical(y_train)

  y_test_new = []
  for i in y_test:
      y_test_new.append(labels.index(i))
  y_test=y_test_new
  y_test = tf.keras.utils.to_categorical(y_test)
  return (y_train, y_test)
```

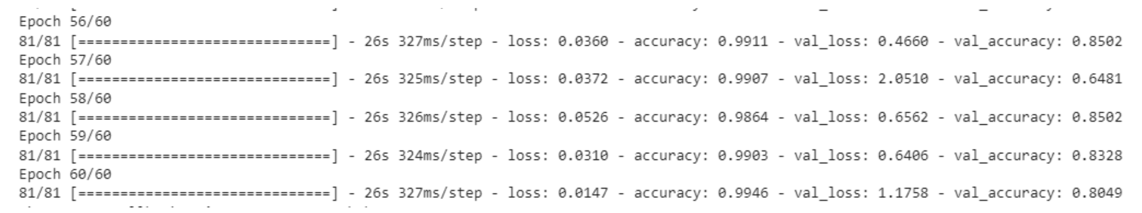
Listing 3 Function to convert the categorical class labels into numeric representation

Listings 4 and 5 illustrate the progression of the first and the last five epochs of training, respectively. It is evident that the accuracy on both training and validation subsets is consistently improving, so the final results should be satisfactory.



```
# training model
model.fit(X_train,y_train,epochs=60,validation_split=0.1)

Epoch 1/60
81/81 [==============================] - 32s 355ms/step - loss: 2.7764 - accuracy: 0.3504 - val_loss: 1.2664 - val_accuracy: 0.7596
Epoch 2/60
81/81 [==============================] - 28s 340ms/step - loss: 1.0460 - accuracy: 0.5381 - val_loss: 1.2181 - val_accuracy: 0.4983
Epoch 3/60
81/81 [==============================] - 27s 338ms/step - loss: 0.8638 - accuracy: 0.6310 - val_loss: 0.7802 - val_accuracy: 0.8502
Epoch 4/60
81/81 [==============================] - 28s 342ms/step - loss: 0.7118 - accuracy: 0.6922 - val_loss: 0.5617 - val_accuracy: 0.9059
Epoch 5/60
81/81 [==============================] - 27s 339ms/step - loss: 0.6223 - accuracy: 0.7476 - val_loss: 0.9309 - val_accuracy: 0.6690
```

Listing 4 First five epochs of training

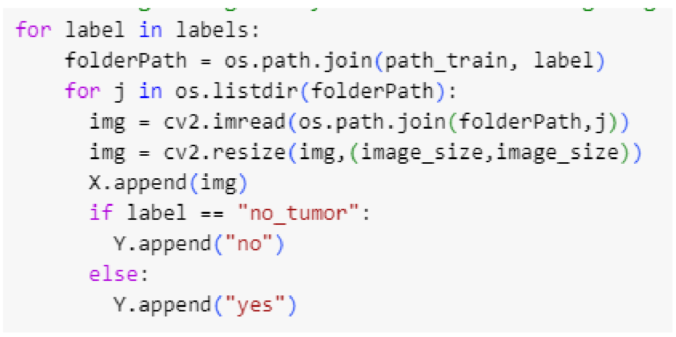


```
Epoch 56/60
81/81 [==============================] - 26s 327ms/step - loss: 0.0360 - accuracy: 0.9911 - val_loss: 0.4660 - val_accuracy: 0.8502
Epoch 57/60
81/81 [==============================] - 26s 325ms/step - loss: 0.0372 - accuracy: 0.9907 - val_loss: 2.0510 - val_accuracy: 0.6481
Epoch 58/60
81/81 [==============================] - 26s 326ms/step - loss: 0.0526 - accuracy: 0.9864 - val_loss: 0.6562 - val_accuracy: 0.8502
Epoch 59/60
81/81 [==============================] - 26s 324ms/step - loss: 0.0310 - accuracy: 0.9903 - val_loss: 0.6406 - val_accuracy: 0.8328
Epoch 60/60
81/81 [==============================] - 26s 327ms/step - loss: 0.0147 - accuracy: 0.9946 - val_loss: 1.1758 - val_accuracy: 0.8049
```

Listing 5 Last five epochs of training

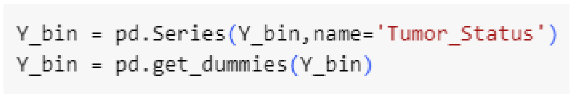
2.2 Binary classification on original dataset

To perform binary classification, we created the Y set while reading the data, assigning ‘yes’ for all tumor classes and ‘no’ for the ‘no_tumor’ class, as shown in Listing 6. Subsequently, we converted it into a pandas Series to utilize ‘get_dummies’ and obtain a dataframe with numeric values, as illustrated in listing 7.



```
for label in labels:
    folderPath = os.path.join(path_train, label)
    for j in os.listdir(folderPath):
      img = cv2.imread(os.path.join(folderPath,j))
      img = cv2.resize(img,(image_size,image_size))
      X.append(img)
      if label == "no_tumor":
        Y.append("no")
      else:
        Y.append("yes")
```

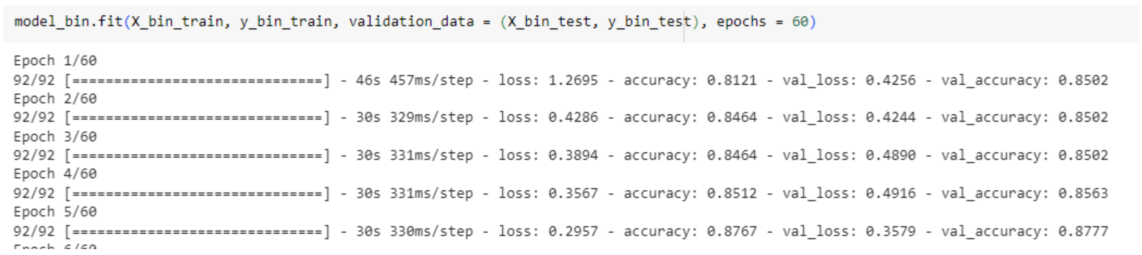
Listing 6 Converting categorical labels into binary labels



```
Y_bin = pd.Series(Y_bin,name='Tumor_Status')
Y_bin = pd.get_dummies(Y_bin)
```

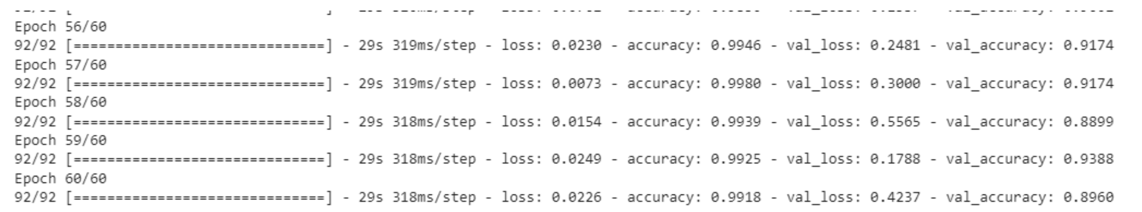
Listing 7 Utilization of 'get_dummies' to obtain dataframe

The fitting process was executed in the same manner as before. Listings 8 and 9 reveal that the accuracy was consistently high from the outset on both training and validation subsets. Notably, the accuracy achieved for binary classification surpassed that of multilabel classification.



```
model_bin.fit(X_bin_train, y_bin_train, validation_data = (X_bin_test, y_bin_test), epochs = 60)

Epoch 1/60
92/92 [==============================] - 46s 457ms/step - loss: 1.2695 - accuracy: 0.8121 - val_loss: 0.4256 - val_accuracy: 0.8502
Epoch 2/60
92/92 [==============================] - 30s 329ms/step - loss: 0.4286 - accuracy: 0.8464 - val_loss: 0.4244 - val_accuracy: 0.8502
Epoch 3/60
92/92 [==============================] - 30s 331ms/step - loss: 0.3894 - accuracy: 0.8464 - val_loss: 0.4890 - val_accuracy: 0.8502
Epoch 4/60
92/92 [==============================] - 30s 331ms/step - loss: 0.3567 - accuracy: 0.8512 - val_loss: 0.4916 - val_accuracy: 0.8563
Epoch 5/60
92/92 [==============================] - 30s 330ms/step - loss: 0.2957 - accuracy: 0.8767 - val_loss: 0.3579 - val_accuracy: 0.8777
```

Listing 8 First five epochs of training



```
Epoch 56/60
92/92 [==============================] - 29s 319ms/step - loss: 0.0230 - accuracy: 0.9946 - val_loss: 0.2481 - val_accuracy: 0.9174
Epoch 57/60
92/92 [==============================] - 29s 319ms/step - loss: 0.0073 - accuracy: 0.9980 - val_loss: 0.3000 - val_accuracy: 0.9174
Epoch 58/60
92/92 [==============================] - 29s 318ms/step - loss: 0.0154 - accuracy: 0.9939 - val_loss: 0.5565 - val_accuracy: 0.8899
Epoch 59/60
92/92 [==============================] - 29s 318ms/step - loss: 0.0249 - accuracy: 0.9925 - val_loss: 0.1788 - val_accuracy: 0.9388
Epoch 60/60
92/92 [==============================] - 29s 318ms/step - loss: 0.0226 - accuracy: 0.9918 - val_loss: 0.4237 - val_accuracy: 0.8960
```

Listing 9 Last five epochs of training

2.3 Modified dataset

The last two cases were performed on augmented data. Augmentation was performed in two ways – by modifying pixel resolution using the function shown in Listing 10 and by applying a blur effect with the function shown in Listing 11. The difference between modifying pixel resolution and applying blur effect is demonstrated in Figures 3, 4 and 5. Figure 3 illustrates non-modified MRI image, while Figure 4 and Figure 5 depict blurred and resolution-modified MRI images, respectively. We conducted two variants of training here – firstly, we trained the model on original dataset and tested it on the modified one, and secondly, we trained the model on modified dataset and tested it on the original one.

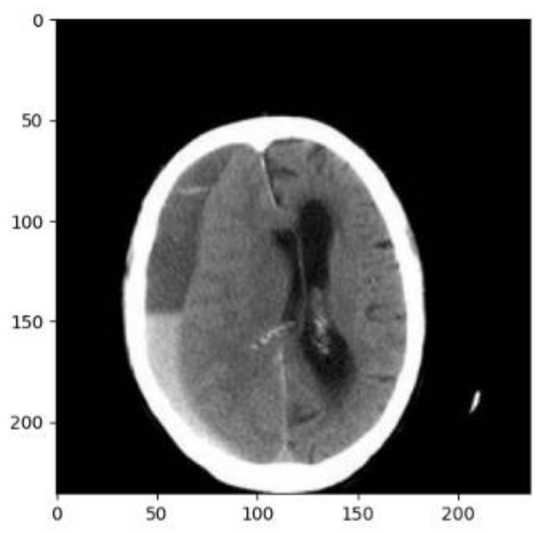
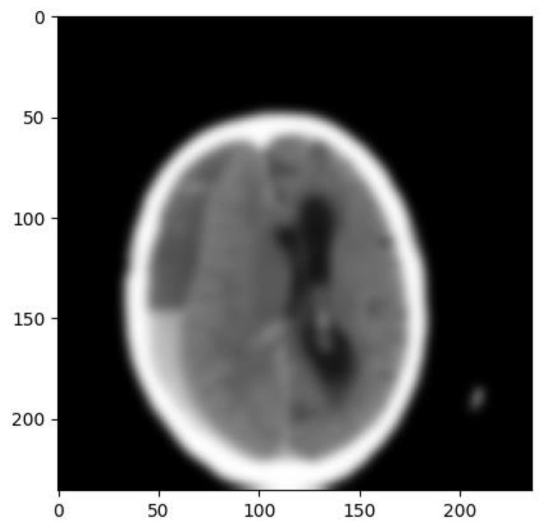


Figure 3 Original non-modified MRI image



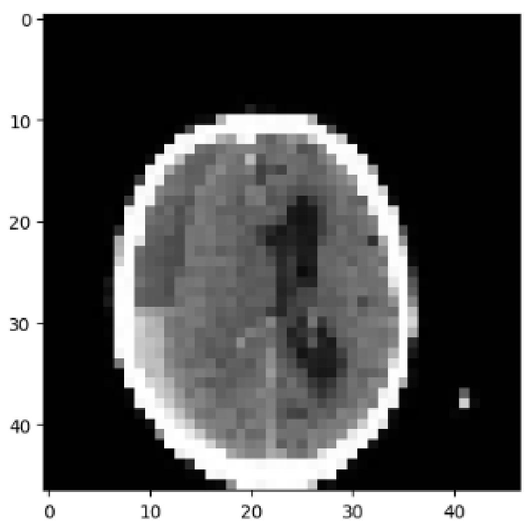
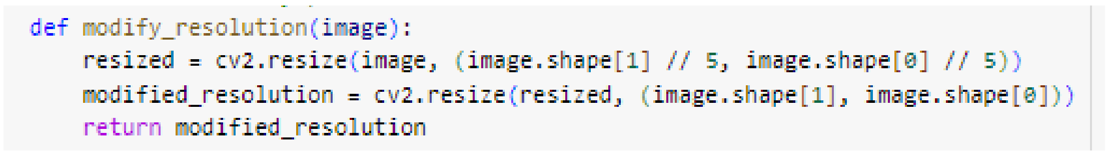


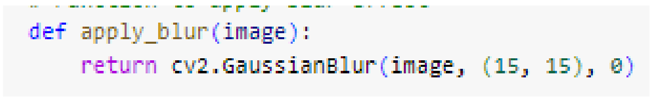
Figure 4 Blurred MRI image

Figure 5 MRI image with modified resolution



```
def modify_resolution(image):
    resized = cv2.resize(image, (image.shape[1] // 5, image.shape[0] // 5))
    modified_resolution = cv2.resize(resized, (image.shape[1], image.shape[0]))
    return modified_resolution
```

Listing 10 Function to modify pixel resolution

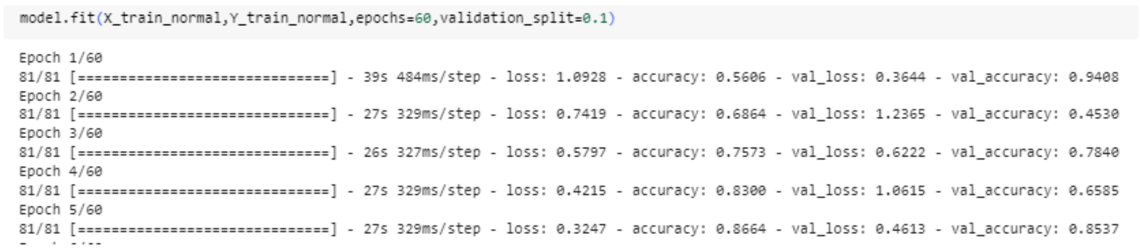


```
def apply_blur(image):
    return cv2.GaussianBlur(image, (15, 15), 0)
```

Listing 11 Function to apply blur effect

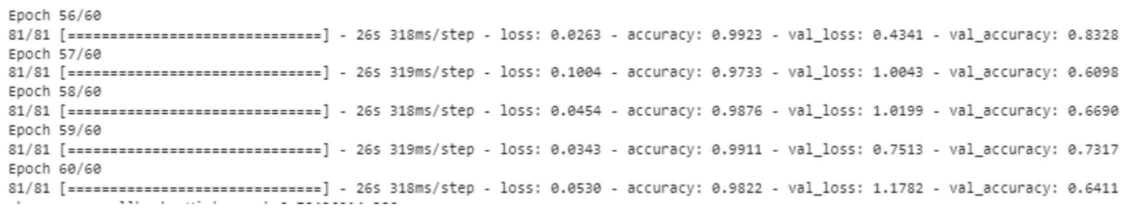
2.3.1 Training on original dataset, test on modified

Listings 12 and 13 display training process on the original dataset. The accuracy and val_accuracy are quite similar to those obtained in the first model, however, the val_accuracy in the final epochs is lower than in the previously mentioned model. This difference may be attributed to the random splitting of the training dataset into training and validation sets.



```
model.fit(X_train_normal,Y_train_normal,epochs=60,validation_split=0.1)

Epoch 1/60
81/81 [==============================] - 39s 484ms/step - loss: 1.0928 - accuracy: 0.5606 - val_loss: 0.3644 - val_accuracy: 0.9408
Epoch 2/60
81/81 [==============================] - 27s 329ms/step - loss: 0.7419 - accuracy: 0.6864 - val_loss: 1.2365 - val_accuracy: 0.4530
Epoch 3/60
81/81 [==============================] - 26s 327ms/step - loss: 0.5797 - accuracy: 0.7573 - val_loss: 0.6222 - val_accuracy: 0.7840
Epoch 4/60
81/81 [==============================] - 27s 329ms/step - loss: 0.4215 - accuracy: 0.8300 - val_loss: 1.0615 - val_accuracy: 0.6585
Epoch 5/60
81/81 [==============================] - 27s 329ms/step - loss: 0.3247 - accuracy: 0.8664 - val_loss: 0.4613 - val_accuracy: 0.8537
```

Listing 12 First five epochs of training



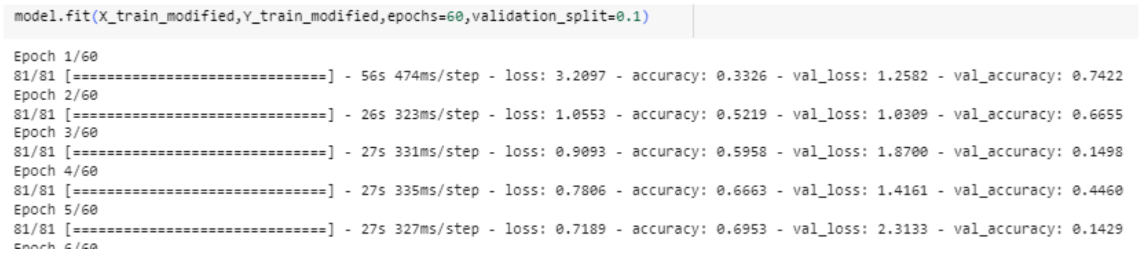
```
Epoch 56/60
81/81 [==============================] - 26s 318ms/step - loss: 0.0263 - accuracy: 0.9923 - val_loss: 0.4341 - val_accuracy: 0.8328
Epoch 57/60
81/81 [==============================] - 26s 319ms/step - loss: 0.1004 - accuracy: 0.9733 - val_loss: 1.0043 - val_accuracy: 0.6098
Epoch 58/60
81/81 [==============================] - 26s 318ms/step - loss: 0.0454 - accuracy: 0.9876 - val_loss: 1.0199 - val_accuracy: 0.6690
Epoch 59/60
81/81 [==============================] - 26s 319ms/step - loss: 0.0343 - accuracy: 0.9911 - val_loss: 0.7513 - val_accuracy: 0.7317
Epoch 60/60
81/81 [==============================] - 26s 318ms/step - loss: 0.0530 - accuracy: 0.9822 - val_loss: 1.1782 - val_accuracy: 0.6411
```

Listing 13 Last five epochs of training

2.3.2 Training on modified dataset, test on original

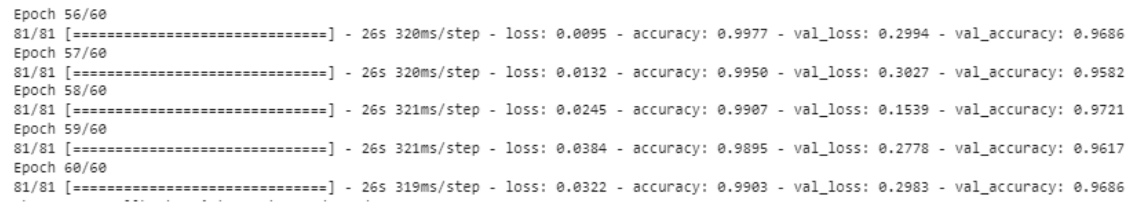
The training procedure here remained the same as before; the only difference was that the training dataset was augmented. Thus, one might expect the accuracy to be lower.

However, upon inspecting Listings 14 and 15, it is evident that unexpectedly, both accuracy and val_accuracy was high – even higher than in the previous case.



```
model.fit(X_train_modified,Y_train_modified,epochs=60,validation_split=0.1)

Epoch 1/60
81/81 [==============================] - 56s 474ms/step - loss: 3.2097 - accuracy: 0.3326 - val_loss: 1.2582 - val_accuracy: 0.7422
Epoch 2/60
81/81 [==============================] - 26s 323ms/step - loss: 1.0553 - accuracy: 0.5219 - val_loss: 1.0309 - val_accuracy: 0.6655
Epoch 3/60
81/81 [==============================] - 27s 331ms/step - loss: 0.9093 - accuracy: 0.5958 - val_loss: 1.8700 - val_accuracy: 0.1498
Epoch 4/60
81/81 [==============================] - 27s 335ms/step - loss: 0.7806 - accuracy: 0.6663 - val_loss: 1.4161 - val_accuracy: 0.4460
Epoch 5/60
81/81 [==============================] - 27s 327ms/step - loss: 0.7189 - accuracy: 0.6953 - val_loss: 2.3133 - val_accuracy: 0.1429
Epoch 6/60
```

Listing 14 First five epochs of training

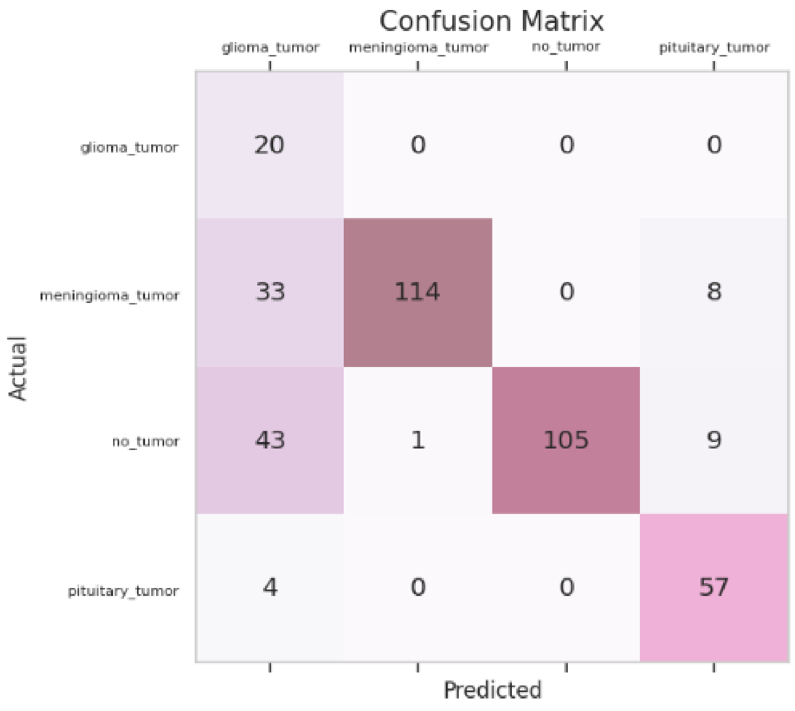


```
Epoch 56/60
81/81 [==============================] - 26s 320ms/step - loss: 0.0095 - accuracy: 0.9977 - val_loss: 0.2994 - val_accuracy: 0.9686
Epoch 57/60
81/81 [==============================] - 26s 320ms/step - loss: 0.0132 - accuracy: 0.9950 - val_loss: 0.3027 - val_accuracy: 0.9582
Epoch 58/60
81/81 [==============================] - 26s 321ms/step - loss: 0.0245 - accuracy: 0.9907 - val_loss: 0.1539 - val_accuracy: 0.9721
Epoch 59/60
81/81 [==============================] - 26s 321ms/step - loss: 0.0384 - accuracy: 0.9895 - val_loss: 0.2778 - val_accuracy: 0.9617
Epoch 60/60
81/81 [==============================] - 26s 319ms/step - loss: 0.0322 - accuracy: 0.9903 - val_loss: 0.2983 - val_accuracy: 0.9686
```

Listing 15 Last five epochs of training

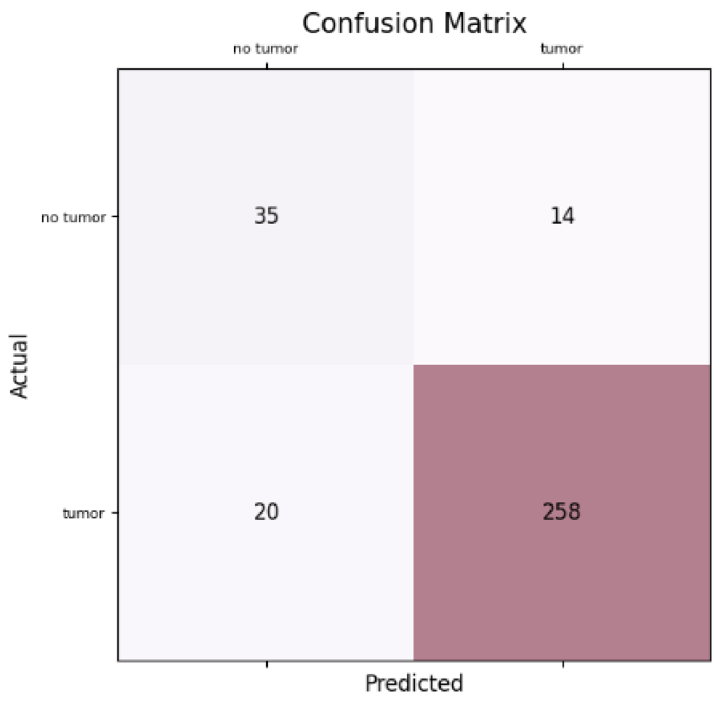
3 Results

3.1 Multilabel classification on original dataset



The multilabel classification model achieved an overall accuracy of 75.13%, indicating that 75.13% of instances were correctly classified. The precision of 74.04% signifies the proportion of correctly predicted positive instances among all instances predicted as positive. With a recall (sensitivity) of 83.36%, the model effectively identified 83.36% of actual positive instances. The F1-score, at 70.52%, represents a balanced measure between precision and recall. While the model demonstrates a generally good performance, further attention may be needed, especially regarding precision, particularly for the 'meningioma_tumor' class.

3.2 Binary classification on original dataset



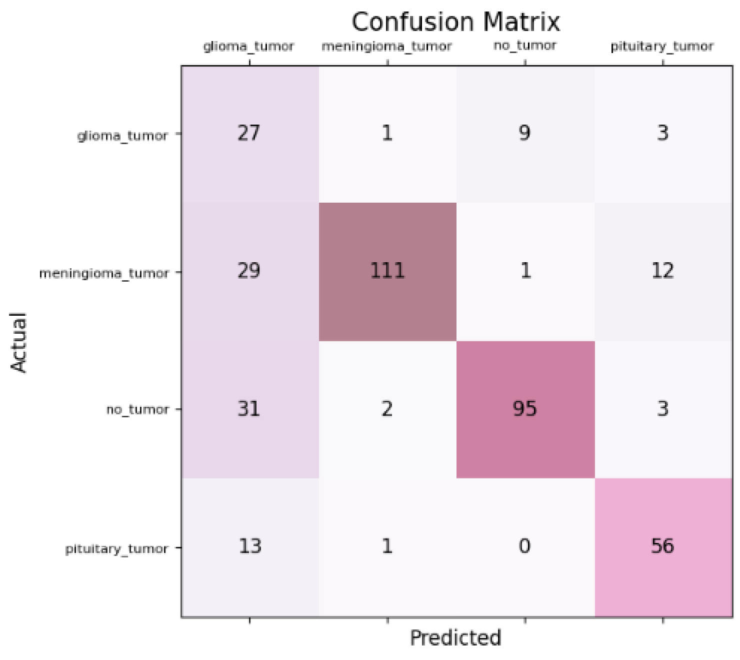
The model achieved an accuracy of 90%, indicating that 90% of predictions were correct overall. The precision of 79% signifies that, among instances predicted as "tumor," 79% were true positives. The recall, or sensitivity, is 82%, indicating that the model correctly identified 82% of actual "tumor" instances. The F1-score, a balanced measure of precision and recall, stands at 81%, highlighting a harmonious trade-off between these two metrics.

Overall, these results suggest a well-performing binary classification model with a good balance between correctness, precision, and recall.

Comparing to the 4 classes, the multilabel classification model achieved higher overall metrics, but it had to handle the complexity of distinguishing between four classes. The binary classification model, with only two classes, achieved high accuracy and balanced precision and recall for the specific task of distinguishing between "tumor" and "no_tumor."

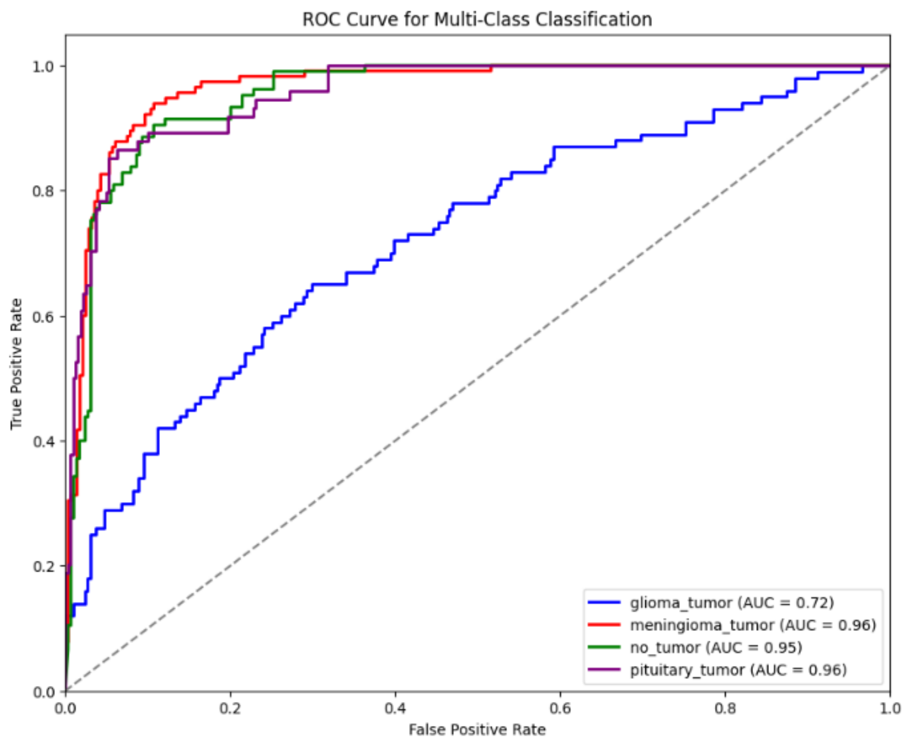
3.3 Modified dataset

3.3.1 Training on original dataset, test on modified



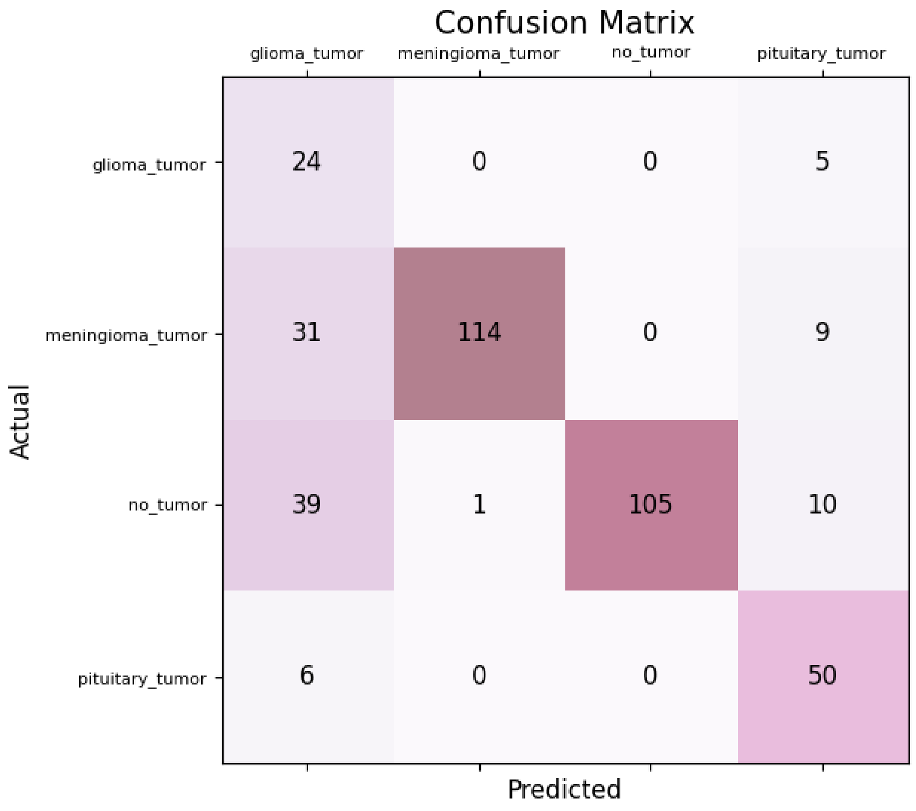
	glioma_tumor	meningioma_tumor	no_tumor	pituitary_tumor
glioma_tumor	27	1	9	3
meningioma_tumor	29	111	1	12
no_tumor	31	2	95	3
pituitary_tumor	13	1	0	56

The model achieved an accuracy of 73.35%, indicating that it correctly classified nearly three-quarters of the instances in the dataset. The precision of 72.42% signifies that, among instances predicted as positive for any class, 72.42% were correctly identified, providing insight into the model's ability to avoid false positives. The recall of 73.14% indicates that the model correctly identified 73.14% of all actual positive instances across all classes, demonstrating its effectiveness in capturing positive cases. The F1-score, a harmonic mean of precision and recall, is 69.92%. This balanced metric considers both false positives and false negatives, providing an overall measure of the model's performance.

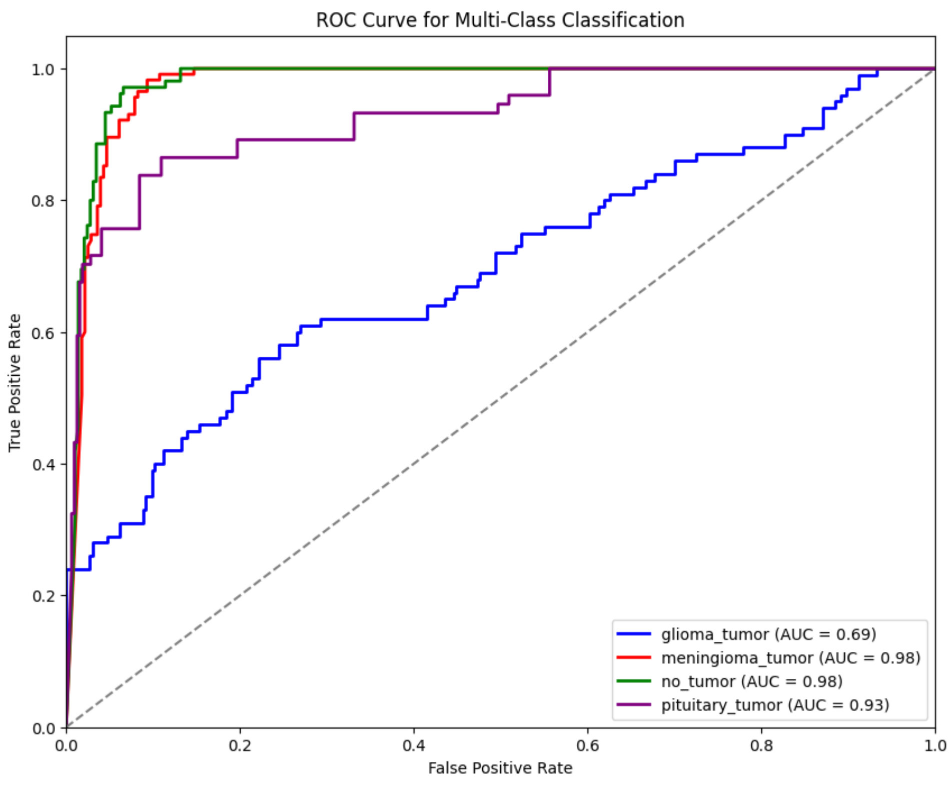


The model demonstrates overall good performance, with high accuracy, precision, and recall. 'glioma_tumor' shows a lower AUC compared to the other classes, suggesting that the model's performance in distinguishing this class from others is relatively weaker The ROC curves for all classes being in the upper-left triangle indicate good discrimination even though 'glioma_tumor' is slightly lower, meaning the model is effective in separating positive and negative instances.

3.3.2 Training on modified dataset, test on original



The model achieved an accuracy of 74.37%, indicating that it correctly classified approximately three-quarters of the instances in the dataset. The precision of 72.67% suggests that, among instances predicted as positive for any class, 72.67% were correctly identified, providing insight into the model's ability to avoid false positives. The recall of 78.45% indicates that the model correctly identified 78.45% of all actual positive instances across all classes, demonstrating its effectiveness in capturing positive cases. The F1-score, a harmonic mean of precision and recall, is 69.91%. This balanced metric considers both false positives and false negatives, providing an overall measure of the model's performance.



All classes are in the upper-left triangle of the ROC space, indicating good discriminatory power. 'glioma_tumor' has a lower AUC (0.69) compared to the other three classes, suggesting that the model's ability to discriminate positive and negative instances for 'glioma_tumor' is somewhat lower. 'pituitary_tumor' is slightly lower than the other two, but they are all still in the same triangle, indicating reasonable performance in distinguishing positive and negative instances.

Both models demonstrate good discriminatory power with ROC curves in the upper-left triangle, indicating effective positive and negative instance separation. 'glioma_tumor' has a lower AUC in both models, but Model 2 has a slightly lower AUC compared to Model 1 (0.69 vs. 0.72), suggesting a potential challenge in distinguishing this class. While Model 2 trained on modified data and tested on normal data shows a slight improvement in some metrics, both models exhibit similar overall performance.

4 Discussion

4.1 Analysis of model performance and results

Our study's exploration into the application of convolutional neural networks for brain tumor classification via MRI has yielded significant insights, particularly in the context of model accuracy and efficiency. Notably, in binary classification tasks, our CNN models achieved an impressive 90% accuracy, indicating a high level of proficiency in distinguishing between tumor and non-tumor cases.

In multilabel classification scenarios, the models maintained a robust performance with an overall accuracy of 75.13%, precision of 74.04%, and a notable recall of 83.36%. These metrics are indicative of the models' capability to differentiate between various tumor types, an essential aspect of diagnostic imaging.

Furthermore, an interesting aspect of our study was the examination of model performance on blurred images. When the models were applied to images with reduced clarity, we observed a noticeable decline in accuracy. This finding is particularly relevant as it simulates real-world scenarios where image quality can vary. The decrease in performance with blurred images underlines the importance of high-quality imaging for effective CNN-based diagnostics and suggests areas for further improvement in model resilience to image quality variations.

This detailed analysis of our results demonstrates the potential and current limitations of CNNs in medical imaging, providing a foundation for future advancements in the field.

4.2 Strengths and limitations

The primary strength of this study is the effective application of convolutional neural networks (CNN) in the classification of brain tumors using MRI, showcasing the robustness of CNNs in handling complex image data. The use of a comprehensive and varied dataset further enhances the credibility and relevance of the findings.

Our research on CNN-based brain tumor classification aligns with existing studies but offers a distinct focus on image quality's impact on CNN performance. Previous research predominantly emphasized the general efficacy of CNNs in medical imaging, whereas our study sheds light on how varying image quality can influence diagnostic accuracy. Additionally, while many studies concentrated on distinguishing tumor types, our approach provides a broader evaluation across different tumor classifications. This contribution enriches the existing literature by highlighting critical aspects like image quality, which are essential for practical applications in clinical settings.

The potential of CNNs in enhancing diagnostic accuracy and efficiency in clinical settings is significant. Our study suggests that incorporating CNN-based tools in clinical practice could improve tumor classification accuracy. However, it also emphasizes the need for maintaining high-quality imaging standards and continuous model training with diverse datasets.

On the limitation front, the study encounters potential biases inherent in the dataset, which might affect the generalizability of the results. Additionally, the models' performance in real-world clinical settings remains to be tested, as the current study was conducted in a controlled environment. There's also the challenge of integrating such advanced AI tools into existing healthcare systems, which requires not only technological adaptation but also training for medical professionals.

4.3 Future directions

Future research should explore more diverse neural network architectures, including transfer learning and generative adversarial networks, to enhance model robustness.

Additionally, expanding dataset diversity and size will be crucial. Investigating the integration of CNNs with other diagnostic tools and electronic health records could offer a more holistic approach to patient care