# Deep Learning for Brain Tumour Detection

**Eliot Deacon**

ECM2423 – AI and Applications
March, 2025

# 1 Introduction

This report is organised into four sections, the introduction, the proposed method, the results and the summary respectively. It will ultimately describe how the deep learning technique has been successfully implemented achieving the goals listed below in 1.1.

**1.1 Aims of the project:**
This project aims to develop a deep learning model for accurate brain tumour classification, prioritizing low false-negative rates. Given the variations in MRI images and tumour structures, a key objective is improving the model's generalization. Additionally, to ensure efficient real-world deployment, the model must balance high accuracy with reduced computational cost.

**1.2 The Problem:**

This project aims to develop a deep learning model for brain tumour detection in MRIs using a dataset of 223 images (150 with tumours, 73 without). The NHS defines a brain tumour as *"a growth of cells in the brain that multiplies abnormally and uncontrollably"* [1]. Tumours can be benign or malignant and may originate in the brain or metastasize from other body parts. With over 130 known types, diagnosis and treatment are highly complex.

MRI scans provide high-resolution grayscale images that contrast soft tissues, making them valuable for tumour detection. Tumours typically appear as irregularly shaped bright regions due to differences in tissue composition. Developing a deep learning classifier for this task lays the groundwork for more advanced diagnostic tools that could assist medical professionals.

**1.3 Related Work & Suitability of Approaches:**

Deep learning has been widely applied to brain tumour classification, with research exploring custom CNNs, pre-trained models, and transfer learning. Studies suggest transfer learning outperforms CNNs, achieving higher accuracy across evaluation metrics [4].

For this study, I selected VGG-16 and EfficientNetB0. VGG-16 is among the best-performing transfer learning models for brain tumour classification [4,5] and other medical imaging tasks [3]. EfficientNetB0 was chosen to assess the trade-off between computational efficiency and classification performance.

# 2 Proposed Method

For this project I selected two pre-trained models, **VGG-16** and **EfficientNetB0**, to classify brain tumours using **transfer learning** to fine-tune to the dataset. Given the size of the dataset, training a deep CNN from scratch seemed infeasible, because of the risk of overfitting and long convergence times. This wouldn't have achieved the goals I defined in section 1. The proposed deep-learning technique avoids this by using pre-trained feature extractors while fine-tuning the final layers to adapt to MRIs. The model architecture is defined in detail in the sections below:

**2.1 Data Preprocessing & Augmentation:**

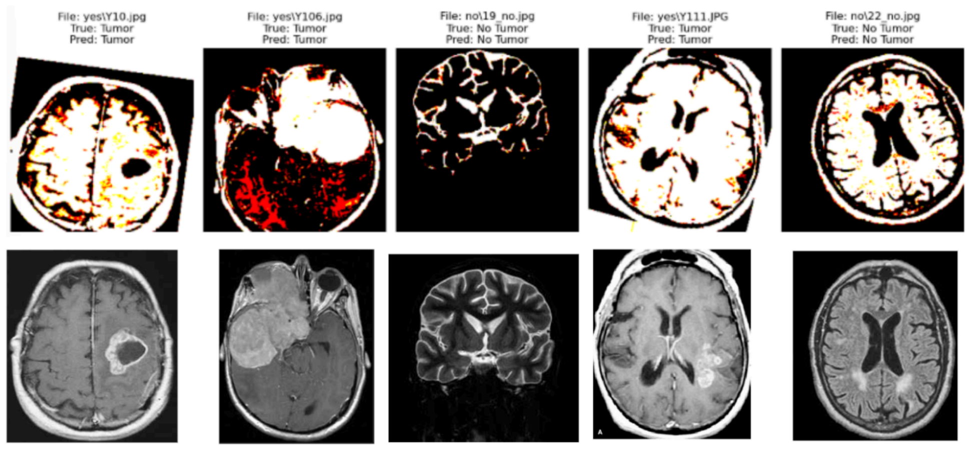


**Fig 1 - Preprocessed and Augmented Images vs Normal Images**

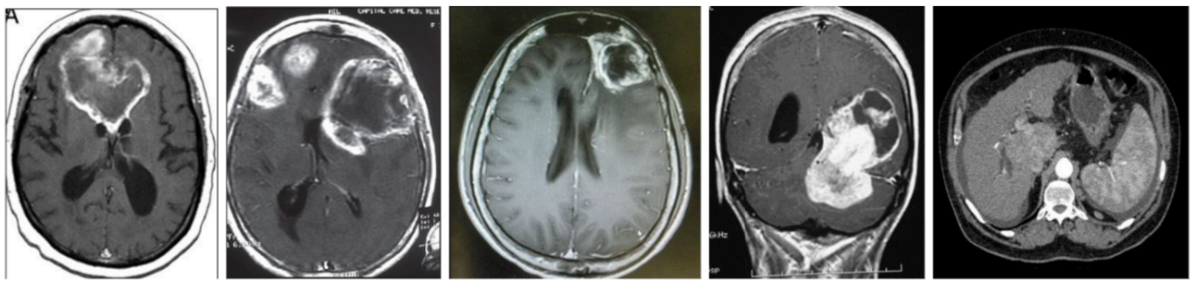
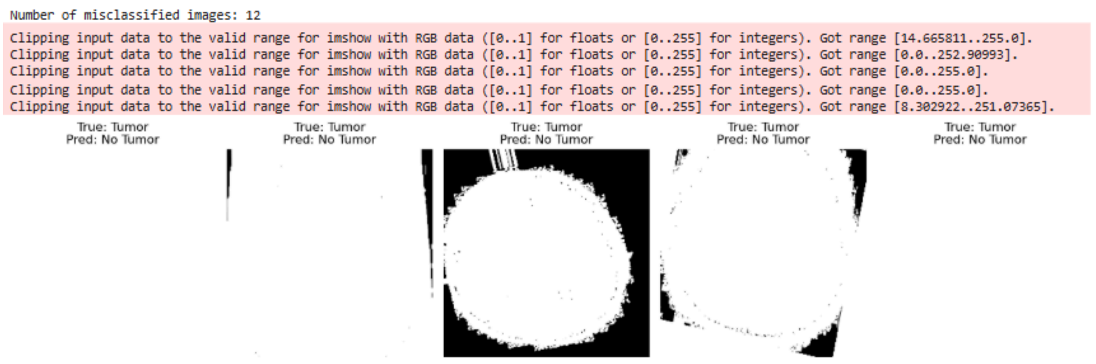


**Fig 2 - Image Variations**

After an initial look-through of the given dataset, I found significant variations in images, including images with writing on them, coloured images (MRIs are meant to be greyscale), variations in brightness, images within images, duplicate images and a scan of the abdomen. To ensure consistency, I removed the abdominal scan and a duplicate image from the 'no' folder, storing them separately in an 'outlier' folder.

**VGG-16 Preprocessing:**. I used the built-in vgg16 preprocessing method to give a standardised input to match the images the base model was trained on, improving the feature extraction and therefore the overall performance of the model.
**EfficientNetB0 Preprocessing:** The EfficientNet model internally normalises the input data. It seemed this method was leading to images with a complete loss of information, shown in Figure 3 below. This could be because the base model was trained on the ImageNet dataset which differs drastically from MRI scans. So I used the VGG-16 pre-processing method for this model and that showed significant improvement.



```
Number of misclassified images: 12
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [14.665811..255.0].
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [0.0..252.90993].
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [0.0..255.0].
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [0.0..255.0].
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers). Got range [8.302922..251.07365].
```

**Fig 3 - EfficientNet PreProcessed Images**

To prevent overfitting in the small dataset, I applied careful data augmentation to preserve tumour integrity. A 15° rotation introduced minor variations while maintaining anatomical structure. Width/height shifts and zoom (0.1) were limited to avoid cropping tumours, which often appear near the edges. Horizontal flipping was allowed, as brain symmetry ensures no impact on detection, while vertical flipping was disabled to maintain the natural orientation of the images, vertical flipping would introduce unrealistic changes that don't occur in MRI brain scans.

**2.2 Pre-Trained Base Model:**

This study employs VGG-16 and EfficientNetB0 models as pre-trained feature extractors to determine the most suitable for my deep-learning technique. This layer allows for general feature extraction and dimensionality reduction, without requiring extensive training. This approach is well-suited to the small dataset, and, as discussed in **section 1**, has been shown to be highly effective in applying deep learning to medical imaging.

The VGG-16 model uses a straightforward convolution-pooling architecture for feature extraction and dimensionality reduction, gradually reducing the spatial dimensions of the inputted 3D vector, while increasing the feature maps (224,224,3) → (7,7,512), to detect small features (edges, dots) to large ones (textures, shapes) as the spatial dimensions are slowly abstracted. It is easy to implement and tune, which was another reason for choosing it, but its 15 million parameters make it computationally demanding.

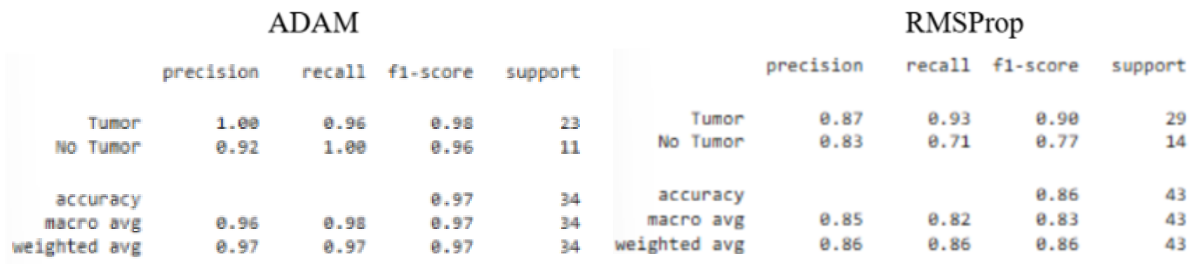
**EfficientNetB0** improves efficiency using squeeze and excitation blocks and the swish activation function, which improves efficiency while maintaining learning ability. The squeeze and excitation blocks enable the model to adjust the strength of different feature maps helping to focus on the most important features. The swish function allows small negative values instead of mapping them to zero, helping to improve learning while maintaining non-linearity. Despite having fewer parameters (~5 million),it produces richer feature maps for the custom classifier. However, its complex structure makes it much harder to understand and tune to specific datasets.

**2.3 Custom Classifier:**
On top of this is a custom classifier which learns to associate the extracted features to the presence/absence of tumours. The hyper-parameters have been tuned differently for the different base models.

**Global Average Pooling:** This reduces the spatial dimensions of the input (e.g., from (7,7,512) to (512)) without parameters, saving time and resources while improving generalization by collapsing the features into a single vector.
**Dense Layers (with ReLU Activation):** These layers help learn data-specific features, with ReLU introducing non-linearity to capture complex patterns.
**Batch Normalisation:** The batch normalisation layer prevents excessive shifting of activations by normalising the data to a mean of 0 and standard deviation of 1 improving convergence and therefore the time to train.
**Dropout:** I use dropout layers to randomly ignore some of the weights, thereby reducing the chances of overfitting to the small dataset. It does this by preventing the model from relying too heavily on specific weights.
**Final Dense Layer (with Sigmoid Activation):** The final dense layer produces the output as a probability of there being a tumour and therefore uses the sigmoid activation function.

**2.4 Loss Function, Optimisation & Backpropagation:**

This model uses **Binary Cross-Entropy Loss**, which is suited to the binary nature of this classification task. Gradients were optimized using ADAM, chosen for its adaptive learning rates and faster convergence compared to standard SGD. A study [2] ranked ADAM among the top optimizers, although RMSProp and NAG showed promise. Since NAG was unavailable in Keras, I compared ADAM and RMSProp, with ADAM performing better across all key metrics (Fig. 4 & 5).



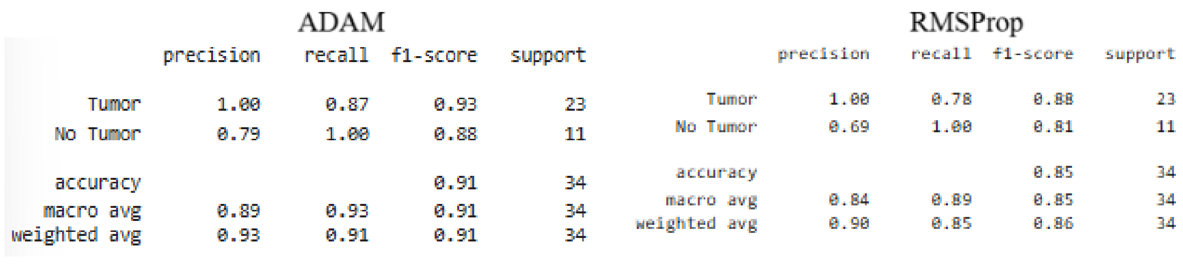
ADAM

```
              precision    recall  f1-score   support

       Tumor       1.00      0.96      0.98        23
    No Tumor       0.92      1.00      0.96        11

    accuracy                           0.97        34
   macro avg       0.96      0.98      0.97        34
weighted avg       0.97      0.97      0.97        34
```

RMSProp

```
              precision    recall  f1-score   support

       Tumor       0.87      0.93      0.90        29
    No Tumor       0.83      0.71      0.77        14

    accuracy                           0.86        43
   macro avg       0.85      0.82      0.83        43
weighted avg       0.86      0.86      0.86        43
```

**Fig 4 - ADAM vs RMSProp for VGG16**



ADAM

```
              precision    recall  f1-score   support

       Tumor       1.00      0.87      0.93        23
    No Tumor       0.79      1.00      0.88        11

    accuracy                           0.91        34
   macro avg       0.89      0.93      0.91        34
weighted avg       0.93      0.91      0.91        34
```

RMSProp

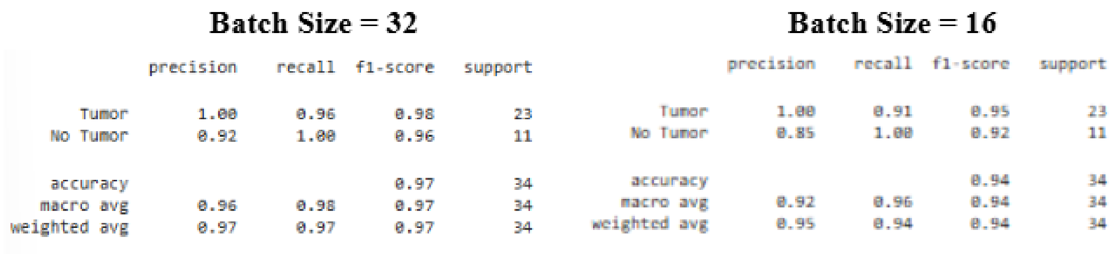
```
              precision    recall  f1-score   support

       Tumor       1.00      0.78      0.88        23
    No Tumor       0.69      1.00      0.81        11

    accuracy                           0.85        34
   macro avg       0.84      0.89      0.85        34
weighted avg       0.90      0.85      0.86        34
```

**Fig 5 - ADAM vs RMSProp for EfficientNetB0**

Initially the base models of VGG-16 and EfficientNetB0 are frozen, and only the custom classifiers are trained during backpropagation. Since the base layers have pre-trained weights already effective for feature extraction, freezing reduces the parameters to update, the time to train and prevents overfitting. The top layers are later gradually unfrozen with a reduced learning rate on the optimiser, to further fine tune to the given dataset.

## 3 Experimental Results

**Hyperparameter Settings:**



**Batch Size = 32**

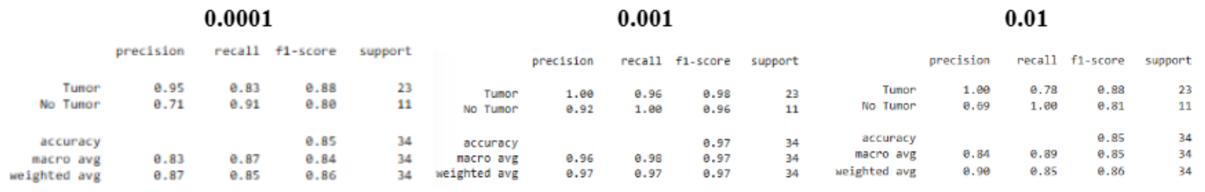
```
              precision    recall  f1-score   support

       Tumor       1.00      0.96      0.98        23
    No Tumor       0.92      1.00      0.96        11

    accuracy                           0.97        34
   macro avg       0.96      0.98      0.97        34
weighted avg       0.97      0.97      0.97        34
```

**Batch Size = 16**

```
              precision    recall  f1-score   support

       Tumor       1.00      0.91      0.95        23
    No Tumor       0.85      1.00      0.92        11

    accuracy                           0.94        34
   macro avg       0.92      0.96      0.94        34
weighted avg       0.95      0.94      0.94        34
```

**Fig 6 - Batch Size changes for VGG16**

**Batch Size:** Initially set to 16 to balance frequent weight updates and computational efficiency, improving generalization. However, 32 performed better for VGG-16 (Fig. 6). EfficientNetB0 crashed at 32, preventing direct comparison.
**Epochs:** During the initial training phase with the frozen base layers I used 20 epochs. After unfreezing I retrained over 10 epochs. Since only the custom classifier layers are being trained in the initial phase, a higher number of epochs allows for more effective learning without modifying pre-trained weights.



**0.0001**

```
              precision    recall  f1-score   support

       Tumor       0.95      0.83      0.88        23
    No Tumor       0.71      0.91      0.80        11

    accuracy                           0.85        34
   macro avg       0.83      0.87      0.84        34
weighted avg       0.87      0.85      0.86        34
```

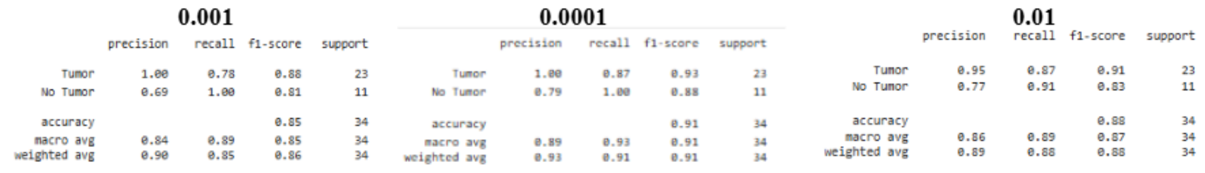
**0.001**

```
              precision    recall  f1-score   support

       Tumor       1.00      0.96      0.98        23
    No Tumor       0.92      1.00      0.96        11

    accuracy                           0.97        34
   macro avg       0.96      0.98      0.97        34
weighted avg       0.97      0.97      0.97        34
```

**0.01**

```
              precision    recall  f1-score   support

       Tumor       1.00      0.78      0.88        23
    No Tumor       0.69      1.00      0.81        11

    accuracy                           0.85        34
   macro avg       0.84      0.89      0.85        34
weighted avg       0.90      0.85      0.86        34
```

**Fig 7 - Initial Learning Rate changes for VGG16**



**0.001**

```
              precision    recall  f1-score   support

       Tumor       1.00      0.78      0.88        23
    No Tumor       0.69      1.00      0.81        11

    accuracy                           0.85        34
   macro avg       0.84      0.89      0.85        34
weighted avg       0.90      0.85      0.86        34
```

**0.0001**

```
              precision    recall  f1-score   support

       Tumor       1.00      0.87      0.93        23
    No Tumor       0.79      1.00      0.88        11

    accuracy                           0.91        34
   macro avg       0.89      0.93      0.91        34
weighted avg       0.93      0.91      0.91        34
```

**0.01**

```
              precision    recall  f1-score   support

       Tumor       0.95      0.87      0.91        23
    No Tumor       0.77      0.91      0.83        11

    accuracy                           0.88        34
   macro avg       0.86      0.89      0.87        34
weighted avg       0.89      0.88      0.88        34
```

**Fig 8 - Initial Learning Rate changes for EfficientNetB0**

**Learning Rate:** For VGG-16 the learning rate on the optimiser is initially set to 0.001, on the first training phase. This allows it to make faster weight updates and more significant learning. When unfreezing I reduced this to 0.0001 to prevent drastic weight updates that would overwrite effective pre-trained knowledge. This aligns with current best practices in transfer learning, it also performs the best across the metrics shown in figure 7. For EfficientNetB0 a slower rate of 0.0001 had better results, as shown above.
**Dropout:** I experimented with a number of dropout values. Initially it was at 0.5. This was having a significant negative effect on my model, likely because a 50% dropout removes too much information. I experimented with dropout values of 0.4, 0.3, and 0.2, and found that 0.3 provided the best validation accuracy for VGG16, while 0.2 worked best for EfficientNetB0.
**Class Weights:** A significant problem with the dataset is the imbalance. Even with the architecture described above and data augmentation this remained a significant problem that essentially prevented my model from accurately making predictions. Initially with equal class weights the model would simply guess tumour most of the time. I initially tried to compute balanced weights but this didn't work either. I found that class weights of 1 for tumour and 5 for no-tumour had the best results.

## Model Training & Evaluation:

**Training Setup:** I initially attempted to use a train-validation-test split of 70:15:15, this left me with unimpressive results. Increasing the test size of the split led to a more accurate evaluation.

**Evaluation Metrics:** To evaluate my model I use a confusion matrix, I measure the precision, recall and f1-score of my models and I use an ROC curve (measuring the AUC as well).

**Results Analysis:**
The **best-performing model** used VGG-16, achieving a **precision, recall, and F1-score of 0.97**, with a tumour class recall of **0.96**. This satisfies the goal of **high accuracy while minimizing false negatives**, making it the most effective approach for brain tumour detection.

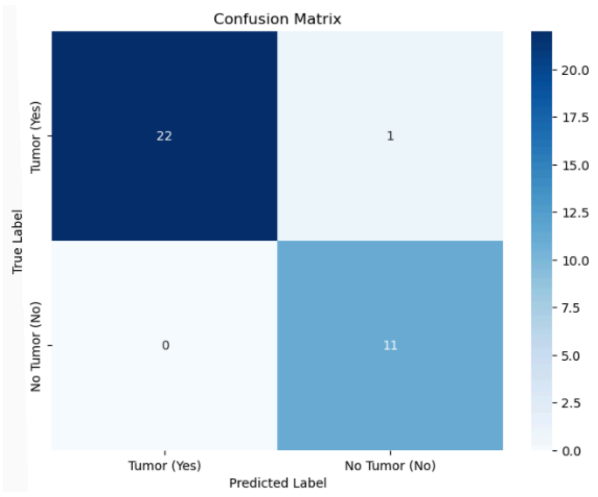
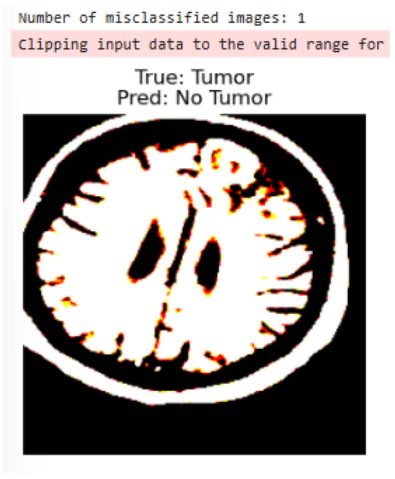


**Fig 9 - VGG-16 Confusion Matrix**



```
Number of misclassified images: 1
Clipping input data to the valid range for
```

**Fig 10 - The misclassified image**

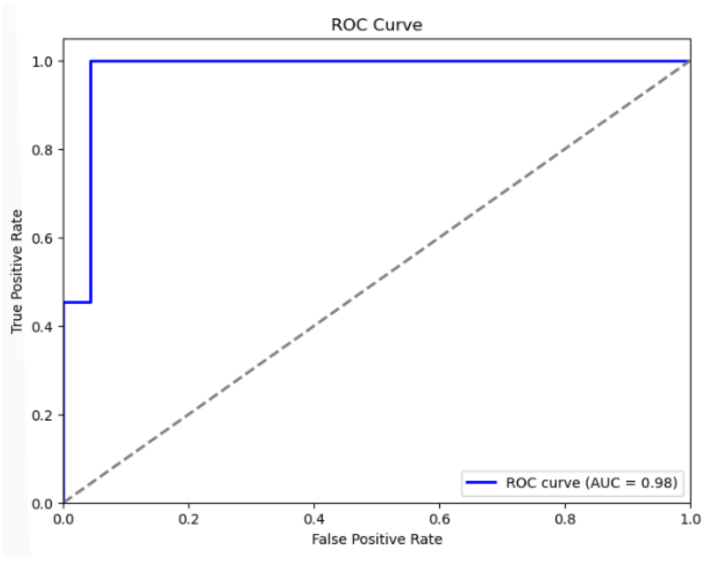
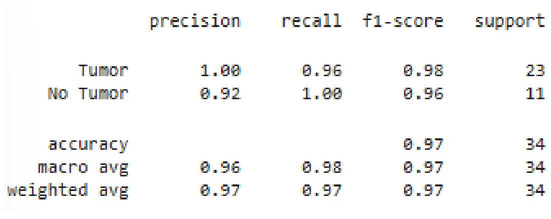


**Fig 11 - VGG-16 ROC/AUC**



```
              precision    recall  f1-score   support

       Tumor       1.00      0.96      0.98        23
    No Tumor       0.92      1.00      0.96        11

    accuracy                           0.97        34
   macro avg       0.96      0.98      0.97        34
weighted avg       0.97      0.97      0.97        34
```

**Fig 12 - Classification Report**

The best model with the EfficientNetB0 base layers achieved a precision of 0.93 , a recall of 0.91 and an f1-score of 0.91. The recall for the tumour class was significantly lower at 0.87, and the metrics overall were lower than VGG-16s. This means that for my deep learning technique a VGG-16 base model with the hyperparameters defined above is best suited for brain tumour classification.

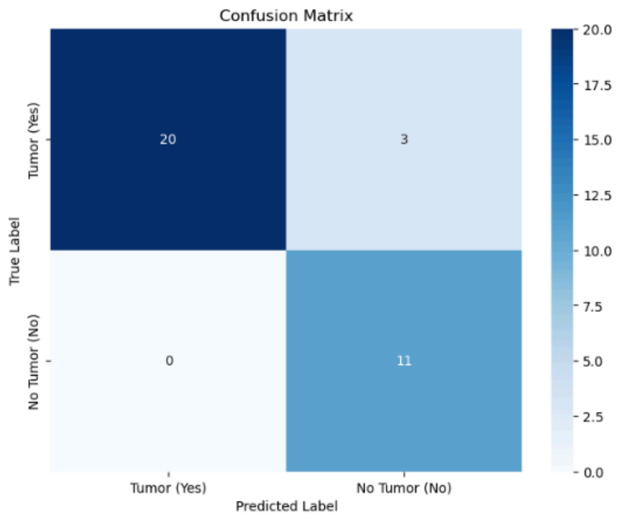
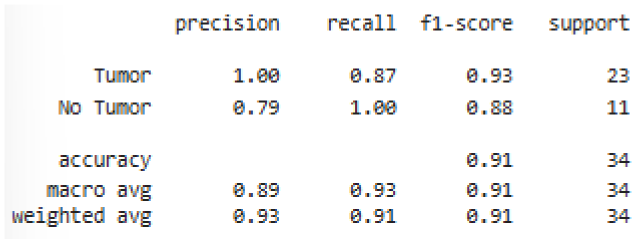


Fig 13 - EfficientNetB0 Confusion Matrix



```
              precision    recall  f1-score   support

       Tumor       1.00      0.87      0.93        23
    No Tumor       0.79      1.00      0.88        11

    accuracy                           0.91        34
   macro avg       0.89      0.93      0.91        34
weighted avg       0.93      0.91      0.91        34
```

Fig 14 - EfficientNetB0 Classification Report

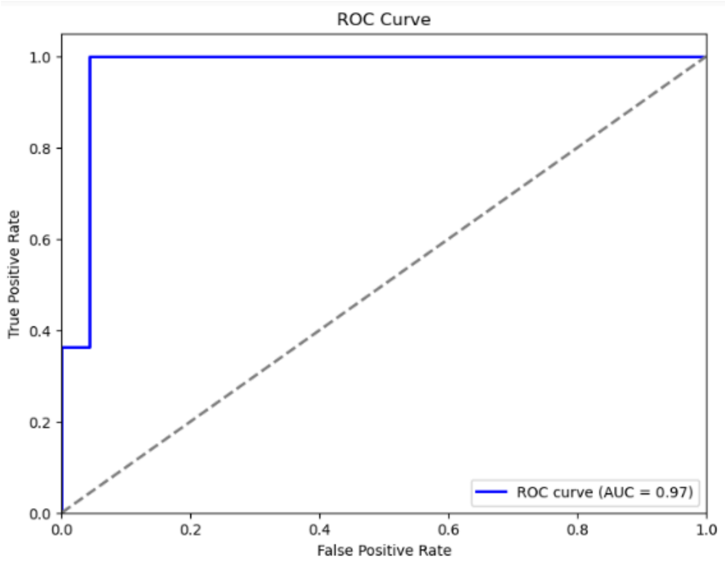


Fig 15 - EfficientNetB0 ROC/AUC

## 4 Summary

This report explored deep learning for brain tumour detection using transfer learning with VGG-16 and EfficientNetB0. Despite challenges from a small, imbalanced dataset, AI techniques minimise their impact. VGG-16 outperformed EfficientNetB0, achieving higher accuracy and fewer false negatives. The final model demonstrates a successful approach to MRI-based tumour classification.

## 5 References

[1] NHS, "Brain tumours," NHS, Mar. 20, 2025. [Online]. Available: https://www.nhs.uk/conditions/brain-tumours/. [Accessed: Mar. 20, 2025].

[2] R. M. Schmidt, F. Schneider, and P. Hennig, “Descending through a crowded valley—Benchmarking deep learning optimizers,” *Int. Conf. Machine Learn.*, pp. 9367–9376, 2021.

[3] M. Abu, N. A. H. Zahri, A. Amir, M. I. Ismail, A. Yaakub, S. A. Anwar, and M. I. Ahmad, “A comprehensive performance analysis of transfer learning optimization in visual field defect classification,” *Diagnostics*, vol. 12, no. 5, p. 1258, 2022.

[4] M. Z. Khaliki and M. S. Baçarslan, “Brain tumor detection from images and comparison with transfer learning methods and 3-layer CNN,” *Sci. Rep.*, vol. 14, no. 1, p. 2664, 2024.

[5] S. Deepak and P. M. Ameer, “Brain tumor classification using deep CNN features via transfer learning,” *Comput. Biol. Med.*, vol. 111, p. 103345, 2019.

When submitting your assessment, you must include the following declaration, ticking all that apply:

☒ I have used GenAI tools for developing ideas.

☒ *I have used GenAI tools to assist with research or gathering information.*

☒ I have used GenAI tools to help me understand key theories and concepts.

☐ I have used GenAI tools to identify trends and themes as part of my data analysis.

☒ I have used GenAI tools to suggest a plan or structure for my assessment.

☒ I have used GenAI tools to give me feedback on a draft.

☐ I have used GenAI tool to generate image, figures or diagrams.

☒ I have used GenAI tools to proofread and correct grammar or spelling errors.

☒ I have used GenAI tools to generate citations or references.

☐ *Other [please specify]*

☐ I have not used any GenAI tools in preparing this assessment.

*I declare that I have referenced use of GenAI outputs within my assessment in line with the University referencing guidelines*