# Brain Tumor Detection

**Nicholas Dziuba**

**2500101**

Seminararbeit

Lehrstuhl für Wirtschaftsinformatik
und Künstliche Intelligenz
Universität Würzburg

Betreuer: Prof. Dr. Gunther Gust
Assistent: Justus Ameling

Würzburg, den 08.12.2023
Bearbeitungszeit: 25.08.2023 - 08.12.2023

# Contents

## Abstract

With its success in natural language processing the Transformer architecture has garnered attention for its potential in computer vision tasks, including brain tumor detection. By leveraging their ability to capture complex patterns and relationships in Magnetic resonance imaging scans, these models have the potential to significantly improve the detection and classification of brain tumors, leading to earlier diagnosis and improved patient outcomes.

This seminar work will give a technological background on the architectures for Residual Networks and the Vision Transformer, Tranfer Learning and furthermore a practical implementation. Evaluating the results of each model on a given dataset, with a final conclusion.

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| **CNN** | Convolution Neural Network |
| **RNN** | Recurrent Neural Networks |
| **ViT** | Vision Transformer |
| **ResNet** | Residual Network |
| **MRI** | Magnetic resonance imaging |

# 1 Introduction

According to the yearly Cancer statistics report, based on the 2023 release, brain and other nervous system tumors are the most common tumor diagnostics, and of that represent 21%. This tumor can become cancerous in which brain cancer continues to be a leading cause of cancer-related deaths in the United States. Tumors can be non-cancerous, benign, or cancerous, being described as malignant, and are the growth of abnormal cells that can occur in any part of the body, with a brain tumor being inside the brain or central spine which can disrupt proper brain function. (Siegel et al., 2023)

To visually diagnose a brain tumor, different medical imaging techniques can be used, one of which are Magnetic resonance imaging (MRI) scans that use a magnetic field and radio frequency signals to produce images of anatomical structures of human tissue. It is a widely-used technique to analyze abnormalities inside the brain and because MRI systems can produce images equal to 65,535 different gray levels, which are difficult for humans to differentiate. (B. Panda and C. S. Panda, 2019)

This is supported by the difficulty of the accessibility and high cost of the otherwise needed human assessment for the detection and classification of the brain tumor and the needed experience and expertise of the radiologist that evaluates the MRI image. (Sailunaz et al., 2023) Various techniques have been proposed to support this process, one of which is training a Convolution Neural Network (CNN) to learn the complex features of MRI images, which resulted in raising the accuracy of diagnosis and tumor classification. (Seetha and Raja, 2018; Abiwinanda et al., 2019)

But with the current success of the Transformer architecture in natural language processing tasks, the possible application for image classification got explored and resulted in the creation of the Vision Transformer. Several studies have demonstrated its effectiveness for brain tumor detection. In a recent study, Asiri et al. (2023) evaluated the performance of a Transformer-based model on a dataset of 1311 MRI scans. And achieved an overall accuracy of 98%, outperforming traditional CNN-based models. Similarly, another study by Tummala et al. (2022) reported an overall testing accuracy of 98.7% for a Transformer-based model on a dataset of 3064 MRI scans. With these promising results, this seminar work will evaluate the performance of this new architecture and compare it against a CNN-based Residual Network (ResNet) model.

# 2 Scope and Limitations of this seminar work

This seminar work aims to provide a broad overview of the current state of Brain Tumor Classification using the ViT architecture, along with a practical implementation. Given the different variants I will limit the scope to the ViT models ViT-b16 and ViT-Bb2 and use a ResNet50V2 model as a baseline, given the prominence of ResNets in CNN-based research. The practical segment of the seminar will focus on applying these models to a specific brain tumor dataset available on Kaggle (Nickparvar, 2021) and evaluating their performance exclusively on this dataset.

It is important to note a key limitation of this approach as the assessment of model accuracy will be based solely on this single dataset. While this dataset provides a controlled environment for evaluation, it also presents potential limitations in terms of the generalizability of the models. The performance metrics obtained from this dataset may not fully represent the model's effectiveness to generalize to real-world scenarios. This is because the dataset may have specific characteristics or biases that do not fully illustrate the diversity and complexity of brain tumors encountered in a practical clinical setting. Representing only far-advanced stages or clearly distinguishable examples of brain tumors. Therefore, while the findings will offer valuable insights into the capabilities within the boundaries of the given dataset, caution should be exercised when evaluating these results to other datasets or clinical applications. The models might not generalize sufficiently across different datasets, which could lead to variations in accuracy and effectiveness.

# 3 Technological Background

In this section, we will look into the technological background of the technologies applied throughout this seminar work. This includes the model architectures implemented and the concept of Transfer Learning. Both of these elements play a crucial role in the understanding and efficient training of the classification models.

## 3.1 Residual Network

ResNet, short for Residual Network, is a CNN-based architecture that was first introduced in the paper "Deep Residual Learning for Image Recognition" by He et al. (2015). It emerged as a solution to a fundamental challenge in training exceptionally deep neural networks.

Before the introduction of this architecture, a common approach was to increase the depth of neural networks as it would consistently enhance their performance. However, as these networks grew deeper, they encountered a critical problem known as the vanishing gradient, which impeded their effective training. Because of that the ResNet architecture introduced a concept to tackle this: residual blocks.

Based on He et al. (2015) residual blocks incorporate skip connections, often referred to as shortcut connections or identity mappings. These skip connections allow the network to skip one or more layers, enabling the model to learn residual functions, the difference between the desired output and the current prediction. This innovative approach eased the training of extremely deep neural networks by mitigating the vanishing gradient problem and facilitating the training of networks with hundreds of layers. With that, they achieved state-of-the-art performance across various computer vision tasks, including image classification, object detection, and image segmentation.

Although novel models demonstrate theoretical efficiency, their effective scaling on modern hardware accelerators remains challenging due to their reliance on specialized attention patterns. As a result, for large-scale image recognition tasks, traditional ResNet-style architectures continued to be a state-of-the-art approach. (Dosovitskiy et al., 2021; Voon et al., 2022)

## 3.2 Visual Transformer

The ViT is a Transformer-based model, an architecture first introduced in the Paper 'Attention is all you need' by Vaswani et al. (2017). It introduced a new attention mechanism, the self-attention, to draw global dependencies between input and output, without using sequence-aligned Recurrent Neural Networks (RNN) or CNNs. This mechanism allows the model using it to be highly parallelizable and efficient, as it eliminates the need for recurrent units. (Vaswani et al., 2017)

Right now this architecture represents the de-facto standard approach in language processing as it is not only creating state-of-the-art results but also being much more efficient on modern accelerators than previous models. (Zhao et al., 2023)

The ViT brings this architecture into the realm of image recognition and provides a significant advancement in the field of computer vision, marking a departure from the CNN-based approaches that have been favored in the literature for years. The core concept for this model, first introduced in the paper "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" by Dosovitskiy et al. (2021), is to treat an image as a sequence of fixed-sized patches, as illustrated with Figure 1, and are subsequently flattened to generate linear embedding. Similar to how words are treated in language processing. Additionally, for these patches, a sequence of positional embeddings is provided, as transformers do not inherently understand spatial relationships or positional information within a sequence.

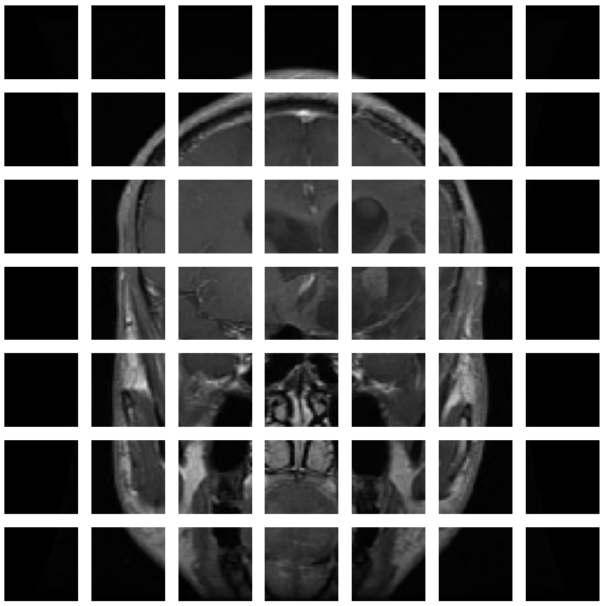


Figure 1: Example of splitting an image into fixed-size patches.

This approach allows the model to leverage the Transformer's self-attention mechanism, enabling it to weigh the importance of different patches of the image relative to each other. By doing so, ViT can capture complex relationships and dependencies between various image regions, offering a more nuanced understanding of the visual content. And doing so in a highly parallelizable manner with the ability to scale with larger datasets and compute resources.

Using this self-attention, an attention map can be created to visualize how much attention the ViT model is paying to different parts of an image. Each attention head has an associated attention weight that serves as a metric for determining the model's focus on that specific region. Attention maps can be used to understand how a ViT model is making its predictions. By looking at an attention map, we can see which parts of the image the model is paying attention to, and we can make inferences about the features that the model is using to classify the image. (Dosovitskiy et al., 2021)

## 3.3 Transfer Learning

The in section 3.2 described ViT, when trained on a medium-sized dataset, was found to generally perform a few percentage points below ResNets of similar sizes. If however a larger dataset was provided, its performance notably improved, showing the significance of having a substantial corpus of data for training. (Dosovitskiy et al., 2021)

To combat this, Transfer learning is used for applications where available data is limited or computational resources are scarce. It is a technique where knowledge gained while solving one problem is applied to a different but similar one. Resulting in so-called pre-trained models that have already learned a rich set of features that can be effectively transferred to new tasks by fine-tuning the model, which adjusts the pre-trained model on a new, typically smaller, dataset to adapt its learned features for a specific task. Often leading to better generalization than models trained from scratch on limited data. In the domain of computer vision, many papers use models that are trained on the ImageNet datasets, which also enable better comparability between the results.

# 4 Practical Implementation

The setup was implemented using Python (2023) and trained on Kaggle using the P100 GPU environment. Tensorflow (Martín Abadi et al., 2015), Keras (Chollet et al., 2015), and vit-keras (Morales, 2023) were utilized to implement the models. For a baseline model, a ResNet50V2 was used, utilizing the default Keras implementation. For the ViT models the vit-keras library provided the vit-b16 and vit-b32 models as there was no implementation included with Keras.

## 4.1 Data Exploration

The brain tumor dataset from Kaggle (Nickparvar, 2021) comprises 5712 training and 1311 test images. Categorized into four different labels, notumor, and the three brain tumor types glioma, meningioma, and pituitary.

- **Glioma:** Gliomas are a type of brain tumor that originates from glial cells. These cells provide support and protection for neurons in the brain. Due to their origin from glial cells, gliomas are often found within the brain tissue itself. (American Cancer Society, 2023a; Asiri et al., 2023)
- **Meningioma:** The meningioma develops from the meninges, the membranes that envelop the brain and spinal cord. Most meningiomas are benign and grow slowly. These are mostly located in the outer part of the brain. (American Cancer Society, 2023a; Asiri et al., 2023)

- **Pituitary:** Pituitary tumors are often benign and occur in the pituitary gland, a small gland at the base of the brain. They can affect the gland's hormone production, leading to various hormonal imbalances. (American Cancer Society, 2023b; Asiri et al., 2023)

The following Figure 2 illustrates some exemplary images that are used to train the models. These images are already preprocessed and augmented, which will be further described in the next section.

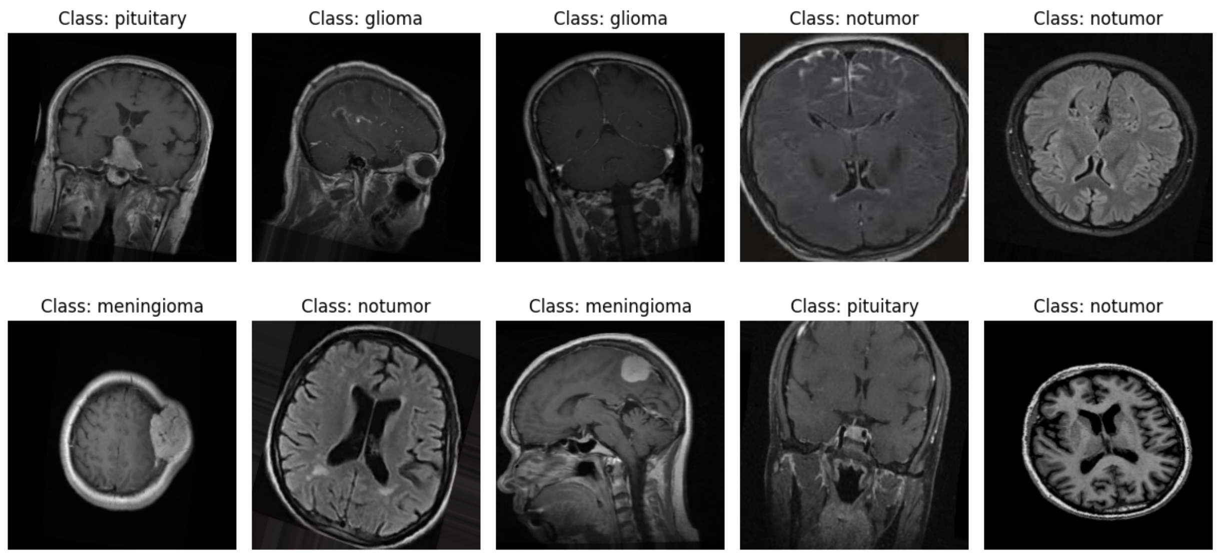


Figure 2: Example of training data for the models, after the preprocessing and data augmentation was applied.

To ensure that the training is representative and unbiased, no single classification class should be over-represented in the dataset. The distribution of classes in the dataset, ranging from 23.2% to 27.98%, indicates a balanced representation. This distribution of classification targets is further illustrated in Figure 3, confirming that no class significantly dominates the training data.

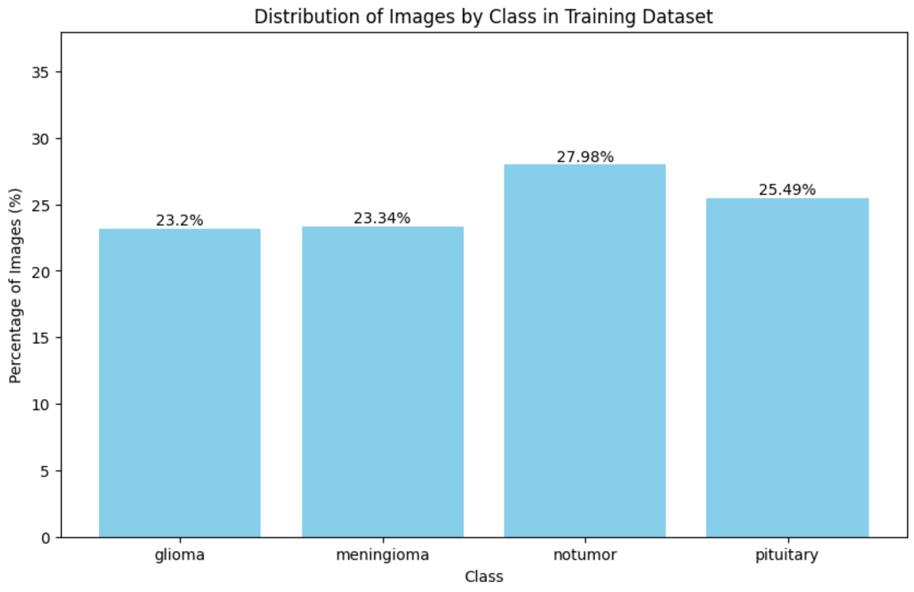


Figure 3: Distribution of images by class in the training dataset.

## 4.2 Data Preprocessing and Hyperparameters

For preprocessing the MRI images the `ImageDataGenerator` class from Keras (TensorFlow, 2023b) was used, as described by Sailunaz et al. (2023, p. 16) and Asiri et al. (2023, p. 5). It is part of the TensorFlow framework and performs real-time data preprocessing and augmentation on each batch of images as it is passed to the model during training. And since it generates data in real-time, it is very memory efficient.

Before beginning with the `ImageDataGenerator`, I set some variables and hyperparameters that are consistently applied throughout the code. The following Listing 1 shows the exact code that was used. Among these, the most noteworthy are the number of epochs, capped at 10, and the dense_neuron parameter, set at 32. This parameter represents the neuron count in the dense layer following the base model, which uses ReLU (Agarap, 2018) as its activation function.

For the training process, the Adam optimizer (Kingma and Ba, 2014) was utilized, with the learning rates set to 0.0001 and the Keras default of 0.001 when the base model was set to a frozen state, further described in section 4.3. The batch_size was configured at 56, and the image_size dimensions standardized the dimensions to 224x224, aligning with the Imagenet21k dataset to ensure the best compatibility with the imported pre-trained weights of each model. The code used for initializing, using these hyperparameters, can be found in the Appendix A for the ResNet50V2 model and the ViT models in the Appendix B.

```python
train_path = '/kaggle/input/brain-tumor-mri-dataset/Training'
test_path = '/kaggle/input/brain-tumor-mri-dataset/Testing'

class_name=['glioma','meningioma','notumor','pituitary']
num_classes=4
epochs = 10
learning_rate = 0.0001
dense_neuron = 32
batch_size = 56
activation = "relu"
adam_optimizer = Adam(learning_rate=learning_rate)
image_size = 224
```

Listing 1: Python code for setting up variables and hyperparameters.

Using the `ImageDataGenerator` for preprocessing and image augmentation the following arguments were set for the training data, only the preprocessing step of rescaling was applied to the test set, as shown with Listing 2.

- **Rescale:** Each pixel value in the image is rescaled by a factor of 1/255. This is a common practice in image processing to convert pixel values from a range of 0-255 to 0-1, this normalizes these pixel values to a range that is more suitable for processing.
- **Zoom Range:** Randomly zooms the image by 20% during training, which in terms of the ViT could train different patches as the same image can be split differently.
- **Brightness Range:** Adjusts the brightness of the image between 80% and 120% of the original image to represent different levels of contrast medium used with MRI scans.
- **Rotation Range:** Randomly rotates the image within a range of 20 degrees, which can again train different patches for the ViT.

Flipping the image, in any way, was not applied as the structures of the brain are not symmetrical, and that could theoretically have a negative impact on the training. This would misrepresent the asymmetrical features, leading to potential confusion for a model that is trying to learn to recognize specific brain structures. However, it should be noted that the impact of image flipping was not further tested.

```python
#Data augmentation for training data
train_datagen = ImageDataGenerator(
    rescale=1./255,
    zoom_range=0.2,
    brightness_range=[0.8, 1.2],
    rotation_range=20
)

#No data augmentation for test data
test_datagen = ImageDataGenerator(rescale=1./255)
```

Listing 2: Using ImageDataGenerator to preprocess and augment the image data.

To complete the setup, I utilized the `flow_from_directory` method (TensorFlow, 2023b), the implementation is shown in Listing 3, to apply the generator configurations to the specified image directories. This involved setting the batch size and defining the target size of the images as 224x244 pixels. The classification was set to a categorical type and the shuffling option for a randomized image sequence was enabled for the training data and disabled for the test data. While implementing the ResNet50V2, the color mode of the images was set to grayscale but had to be changed to RGB as this was a requirement for the ViT models.

```
train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical',
    #color_mode='grayscale',
    color_mode='rgb', #needed for ViT
    shuffle=True
)

test_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=(image_size, image_size),
    batch_size=batch_size,
    class_mode='categorical',
    #color_mode='grayscale',
    color_mode='rgb', #needed for ViT
    shuffle=False
)

steps_per_epoch = train_generator.n // train_generator.batch_size
validation_steps = test_generator.n // test_generator.batch_size
```

Listing 3: Using flow_from_directory to apply preprocessing and variables to the source paths.

## 4.3 Training the models

All models were implemented using Keras and incorporated Transfer Learning by importing for each model the pre-trained weights of the ImageNet21k dataset. These models were then fine-tuned, on the in section 4.1 described and section 4.2 preprocessed and augmented training data. To ensure a close comparison among the models, they were all subjected to the same hyperparameters and identical image augmentation techniques.

As earlier described for the ResNet50V2 model, the default implementation available in Keras was used. However, for the ViT models, the vit-keras library (Morales, 2023) was installed into Kaggle to provide the vit-b16 and vit-b32 models, as Keras did not include a built-in implementation for those.

The ResNet50V2 was chosen because of the, as stated by He et al. (2016), easier training and faster generalization than ResNet50. Training a ResNet50 and ResNet50V2 using the same hyperparameters the Version 2 model converged much faster. While testing the ResNet50, it started to converge after epoch 8 as shown in Table 1, starting with a Validation Loss of 5.3063 in the first epoch, by comparison, the V2 model had a Validation Loss of 0.4214. Being fully converged by epoch 8, where the original model just started. And because all training runs were limited to 10 epochs the ResNet50V2 was the better choice as a CNN-based baseline.

| Epoch | ResNet50 | | | | ResNet50V2 | | | |
|---|---|---|---|---|---|---|---|---|
| | Train Loss | Train Acc | Val Loss | Val Acc | Train Loss | Train Acc | Val Loss | Val Acc |
| 1 | 0.2846 | 0.8960 | 5.3063 | 0.3144 | 0.3210 | 0.8790 | 0.4214 | 0.8602 |
| 2 | 0.0813 | 0.9715 | 5.3495 | 0.3144 | 0.0927 | 0.9688 | 0.3395 | 0.8843 |
| 3 | 0.0644 | 0.9779 | 5.2272 | 0.3144 | 0.0648 | 0.9802 | 0.1659 | 0.9425 |
| 4 | 0.0487 | 0.9835 | 6.7035 | 0.3144 | 0.0460 | 0.9832 | 0.0874 | 0.9689 |
| 5 | 0.0314 | 0.9895 | 5.9626 | 0.3183 | 0.0368 | 0.9870 | 0.1245 | 0.9697 |
| 6 | 0.0306 | 0.9895 | 8.3809 | 0.3354 | 0.0427 | 0.9841 | 0.0638 | 0.9829 |
| 7 | 0.0199 | 0.9947 | 5.8960 | 0.3602 | 0.0296 | 0.9891 | 0.1223 | 0.9674 |
| 8 | 0.0280 | 0.9905 | 3.3811 | 0.4301 | 0.0173 | 0.9951 | 0.0423 | 0.9852 |
| 9 | 0.0334 | 0.9900 | 1.4251 | 0.6250 | 0.0161 | 0.9949 | 0.0468 | 0.9798 |
| 10 | 0.0158 | 0.9951 | 0.1798 | 0.9449 | 0.0251 | 0.9923 | 0.0396 | 0.9852 |

Table 1: Comparison of training and validation performance between ResNet50 and ResNet50V2.

Utilizing the `Sequential` model (TensorFlow, 2023c) in TensorFlow, to every base model two additional dense layers were added. One is to learn the extracted features and the second one is to output the classification. The resulting models are shown in Table 2, which illustrates the significant size difference between the ResNet model being 90.14 MB and the ViT models around 330 MB.

| **Model** | **Layer (Type)** | **Output Shape** | **Param #** | **Size** |
|---|---|---|---|---|
| ResNet50V2 | resnet50v2 (Functional) | (None, 2048) | 23,564,800 | 90.14 MB |
| | dense_layer_32 (Dense) | (None, 32) | 65,568 | |
| | output_layer (Dense) | (None, 4) | 132 | |
| ViT-b16 | vit-b16 (Functional) | (None, 768) | 86,389,248 | 329.64 MB |
| | dense_layer_32 (Dense) | (None, 32) | 24,608 | |
| | output_layer (Dense) | (None, 4) | 132 | |
| ViT-b32 | vit-b32 (Functional) | (None, 768) | 88,045,824 | 335.96 MB |
| | dense_layer_32 (Dense) | (None, 32) | 24,608 | |
| | output_layer (Dense) | (None, 4) | 132 | |

Table 2: Architecture of ResNet50V2, ViT-b16, and ViT-b32 Models

For those three models, two distinct approaches were explored:

- **Freezing the base model:** In this approach, the base model was used solely as a pre-trained feature extractor, with only the dense layers in Table 2 being trained for the classification task. With a frozen base layer a learning rate of 0.001 was used to train the models.
- **Training all weights:** This involved making all the weights of the model trainable, which led to improved performance or faster convergence in every model evaluated.

While training each model, with the implementation in Appendix C, a `EarlyStopping` callback (TensorFlow, 2023a) for efficiency and to prevent further training into overfitting was applied. This callback monitors the validation loss, requiring a minimum improvement of 0.01 to consider the model improved. Training is halted if no significant improvement is observed for 5 epochs. Upon stopping early, the model reverts to the weights from its best-performing epoch.

## 4.4 Evaluating the models

In the subsequent sections, the effectiveness of each model is assessed using multiple metrics, such as a confusion matrix, accuracy and loss curves, and precision-recall statistics. The code for generating this evaluation can be found in the Appendix D. The confusion matrix provides a visual and quantitative depiction of the model's classification capabilities across the four categories: glioma, meningioma, pituitary, and notumor. The training and validation accuracy curves reveal the progress in the model's predictive accuracy over the 10 trained epochs, with a focus on reducing the gap between training and validation curves. The precision, recall, and F1 score support the detailed understanding of the model's performance, especially in terms of its ability to correctly identify each class while minimizing false positives and false negatives. These comprehensive metrics together provide a view of the model's strengths and weaknesses in classifying brain tumor types, guiding potential improvements and adjustments in future iterations of the model.

### 4.4.1 ResNet50V2

The training process for the ResNet50V2 had a total training duration of 10.31 minutes with a frozen base model and 11.42 minutes without. With the total model size being 90.14 MB.

#### Confusion Matrix Analysis

The confusion matrix, as shown in Figure 4 and 5 shows the unfrozen model's robust ability to discern the absence of tumors with a 100% success rate, as classified by notumor. And for the frozen base layer model with a high degree of accuracy, 399 true positives out of 405 cases. The inter-class misclassification is predominantly between glioma and meningioma, which may imply an inherent similarity in their imaging characteristics that the model occasionally misinterprets. Overall, the unfrozen model with being able to train the base model and with that its feature extraction, results in better performance across all classes.

#### Accuracy and Loss Curves

Based on the evaluation curves provided with Figure 6 and 7 the following observations can be made.

- The model with the frozen base model showed consistent improvement with both training and validation accuracy, but a dip in epochs 5 and 10. Also, the validation accuracy increases steadily, but more slowly than the training accuracy. This suggests that it is still generalizing but with some signs of overfitting at the end.

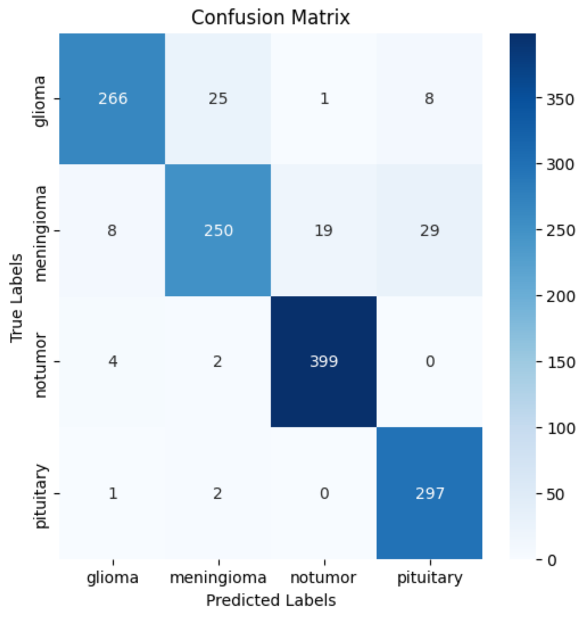


Figure 4: Confusion matrix for ResNet50V2 with a frozen base model.

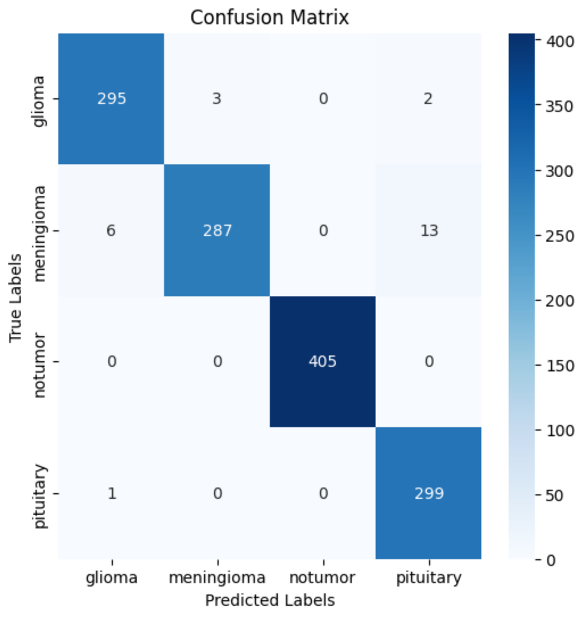


Figure 5: Confusion matrix for ResNet50V2.

- The model with the unfrozen base model showed a greater fluctuation between training and validation accuracy. The validation loss for this model also showed more fluctuation, which can be a sign of overfitting, although it did show a higher overall accuracy. This can imply that while the unfrozen model may have learned more detailed features from the data, it may not perform as well on unseen data without additional regularization or training techniques.

Overall, the learning curves suggest that the models are learning to classify brain tumors with reasonable accuracy. However, it is important to note that the models may be overfitting the training data to some extent.

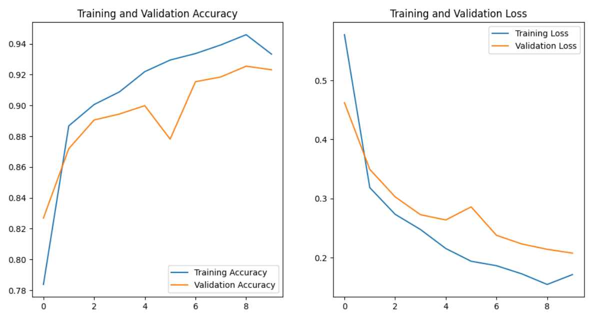


Figure 6: Accuracy and loss curves for ResNet50V2 with a frozen base model.

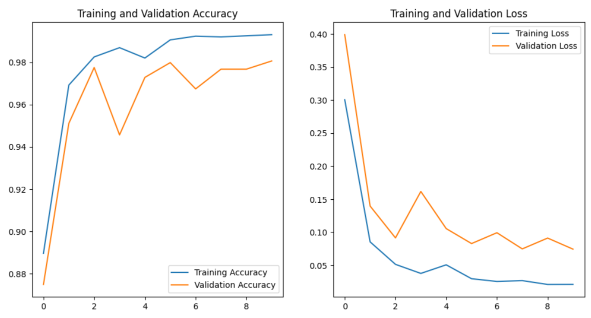


Figure 7: Accuracy and loss curves for ResNet50V2.

### Quantitative Metrics

The models achieve an overall accuracy of 93% and 99%, with high precision, recall, and f1-scores for each class, as demonstrated by Table 3 and 4. The f1-score, a balanced measure of precision and recall, remains consistently high across all classes, indicating a balanced classification performance.

As seen with the confusion matrix and now based on the performance metrics, the ResNet50V2 model with the unfrozen base model (Table 4) generally outperforms the one with the frozen base model (Table 3) across all reported metrics. The unfrozen model shows perfect precision and recall in the notumor class and very high scores in the remaining classes, leading to an overall accuracy of 98%. Resulting in a well-performing and well-balanced model across all classes. In comparison, the frozen model has lower precision, recall, and f1-scores especially for glioma and meningioma classes. And the overall accuracy of 93%, which, while still high, is less than the unfrozen model.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.95 | 0.89 | 0.92 | 300 |
| Meningioma | 0.90 | 0.82 | 0.85 | 306 |
| Notumor | 0.95 | 0.99 | 0.97 | 405 |
| Pituitary | 0.89 | 0.99 | 0.94 | 300 |
| **Accuracy** | | | 0.92 | 1311 |
| **Macro avg** | 0.92 | 0.92 | 0.92 | 1311 |
| **Weighted avg** | 0.92 | 0.92 | 0.92 | 1311 |

Table 3: Performance Metrics for ResNet50V2 with a frozen base model.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.98 | 0.98 | 0.98 | 300 |
| Meningioma | 0.99 | 0.94 | 0.96 | 306 |
| Notumor | 1.00 | 1.00 | 1.00 | 405 |
| Pituitary | 0.95 | 1.00 | 0.97 | 300 |
| **Accuracy** | | | 0.98 | 1311 |
| **Macro avg** | 0.98 | 0.98 | 0.98 | 1311 |
| **Weighted avg** | 0.98 | 0.98 | 0.98 | 1311 |

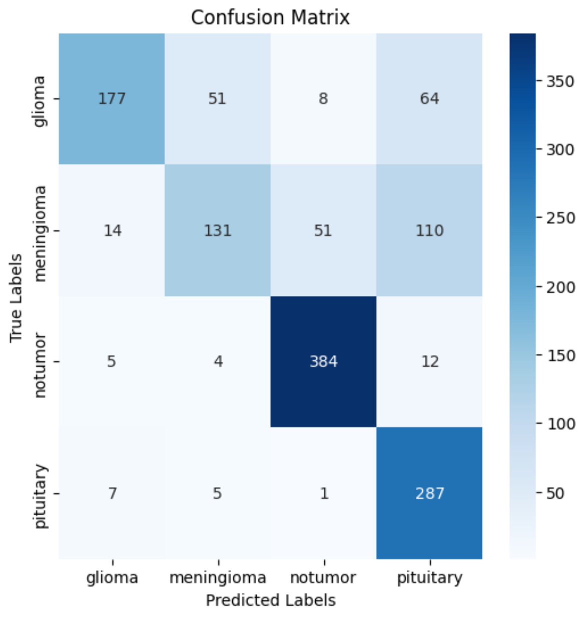
Table 4: Performance Metrics for ResNet50V2.

### 4.4.2 ViT-b16

The training process for the ViT-b16 had a total training duration of 12.06 minutes with a frozen base model. The unfrozen model trained for 20.33 minutes for 8 epochs, as it was stopped by the `EarlyStopping` implementation. Also restoring the model weights from the end of epoch 3. The ViT-b16 model has a total parameter size of 329.64 MB.

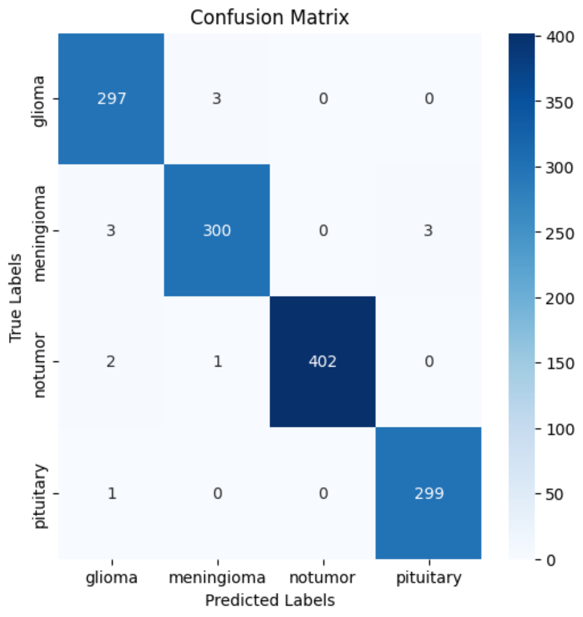
### Confusion Matrix Analysis

Figure 8 and 9 illustrate the confusion matrices for the ViT-b16 models, revealing a significant discrepancy in performance. The unfrozen model demonstrates a significantly better performing model, with a high number of true positives across all classes and minimal misclassifications. For the notumor class, six images were incorrectly identified as glioma or meningioma, but no false positive which are more important in a medical context. On the other hand, the frozen model faces notable difficulties, especially in accurately identifying glioma and meningioma. With 51 meningioma cases being wrongly labeled as notumor, resulting in false positive predictions, and 110 being predicted as pituitary, the model has more false than true positives for this class.



|  | glioma | meningioma | notumor | pituitary |
|---|---|---|---|---|
| glioma | 177 | 51 | 8 | 64 |
| meningioma | 14 | 131 | 51 | 110 |
| notumor | 5 | 4 | 384 | 12 |
| pituitary | 7 | 5 | 1 | 287 |

Figure 8: Confusion matrix for ViT-b16 with a frozen base model.



|  | glioma | meningioma | notumor | pituitary |
|---|---|---|---|---|
| glioma | 297 | 3 | 0 | 0 |
| meningioma | 3 | 300 | 0 | 3 |
| notumor | 2 | 1 | 402 | 0 |
| pituitary | 1 | 0 | 0 | 299 |

Figure 9: Confusion matrix for ViT-b16.

## Accuracy and Loss Curves

Analyzing the Figures 10 and 11, the frozen model fit well to the training data with a steady increase in performance. However, it shows significant signs of overfitting, with a notable divergence between training and validation accuracy and an upswing in validation loss. In contrast, the model, with its base layers unfrozen, maintained a consistent performance throughout the training process. Keeping a close relationship between the training and validation curves, with some fluctuations. Where the implemented `EarlyStopping` halted further training, as there was no significant improvement. Consequently, the unfrozen model promises better generalization while also performing better.

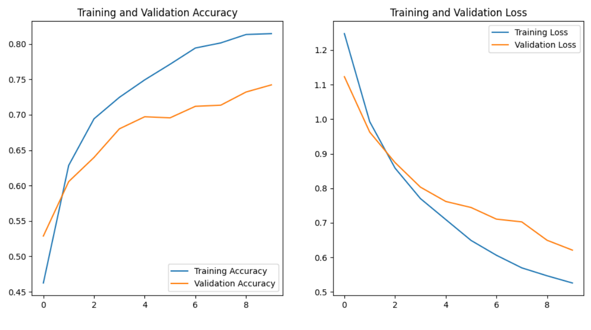


Figure 10: Accuracy and loss curves for ViT-b16 with a frozen base model.

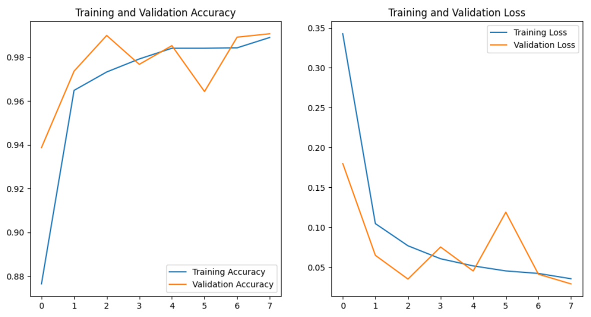


Figure 11: Accuracy and loss curves for ViT-b16.

### Quantitative Metrics

The unfrozen model, Table 6, outperforms the frozen model, Table 5, across all evaluated metrics. As found by the earlier evaluated metrics. High precision and recall suggest that the unfrozen model is both accurate and reliable in its predictions, showing a robust capacity to generalize across classes. The f1-scores close to 1.00 for all classes further confirm its balanced performance in terms of both precision and recall.

In contrast, when freezing the base layers the model struggles with lower precision and recall, particularly in distinguishing glioma and meningioma classes. This is reflected in the substantially lower f1-scores for these classes, indicating a less balanced performance and a potential weakness in either the model's feature extraction or classification capabilities. The overall accuracy and average scores are significantly lower than the unfrozen model, suggesting that freezing the base layers of the model has constrained its ability to learn discriminative features effectively. The data indicates that the unfrozen model is the better choice in this context, potentially due to its ability to fine-tune all layers of the network during training, allowing for more nuanced feature learning.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.87 | 0.59 | 0.70 | 300 |
| Meningioma | 0.69 | 0.43 | 0.53 | 306 |
| Notumor | 0.86 | 0.95 | 0.90 | 405 |
| Pituitary | 0.61 | 0.96 | 0.74 | 300 |
| **Accuracy** | | | 0.75 | 1311 |
| **Macro avg** | 0.76 | 0.73 | 0.72 | 1311 |
| **Weighted avg** | 0.77 | 0.75 | 0.73 | 1311 |

Table 5: Performance metrics for ViT-16, with a frozen base model.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.98 | 0.99 | 0.99 | 300 |
| Meningioma | 0.99 | 0.98 | 0.98 | 306 |
| Notumor | 1.00 | 0.99 | 1.00 | 405 |
| Pituitary | 0.99 | 1.00 | 0.99 | 300 |
| **Accuracy** | | | 0.99 | 1311 |
| **Macro avg** | 0.99 | 0.99 | 0.99 | 1311 |
| **Weighted avg** | 0.99 | 0.99 | 0.99 | 1311 |

Table 6: Performance metrics for ViT-b16.

### 4.4.3 ViT-b32

Training the ViT-b32 models, with a frozen and unfrozen base model, had a total training duration of 9.87 and 11.81 minutes respectively, and a model size of 335.96 MB.

### Confusion Matrix Analysis

As shown in Figure 12 and 13 the confusion matrices for the ViT-b32 models reveal a similar behavior as the ViT-b16. The unfrozen model significantly outperforms the frozen model. But with 100% true positives for the notumor class, it was able to always classify correctly if no tumor was present and with a high number of true positives across all additional classes, resulting in minimal misclassifications. And while the unfrozen ViT-b32 model performed better than the ViT-b16 counterpart, the same is not true for the frozen models. Here the frozen ViT-b32 showed the same or arguably slightly worst performance as it did have more misclassifications for the notumor class and confused more meningiomas for pituitary tumors. This further solidifies the need for the ViT model to train with unfrozen base layers.

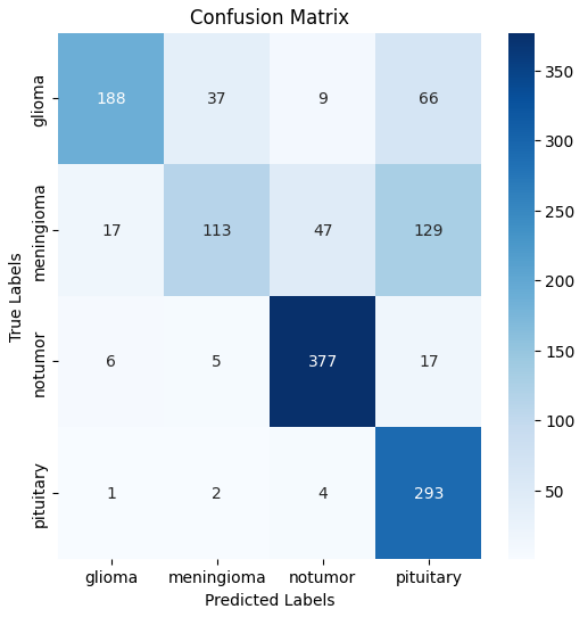


Figure 12: Confusion matrix for ViT-b32 with a frozen base model.

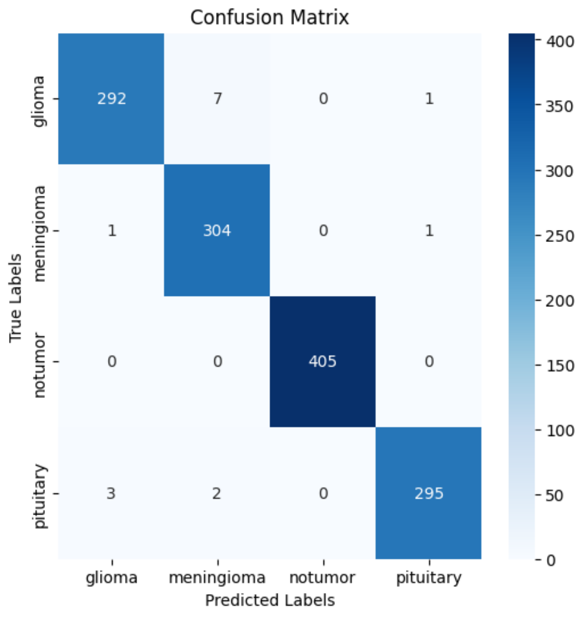


Figure 13: Confusion matrix for ViT-b32.

## Accuracy and Loss Curves

Like with the confusion matrices the accuracy and loss curves in Figure 14 and 15 showed a similar behavior as the ViT-b16 models. The frozen model fitting again well to the training data but showing significant signs of overfitting. And the unfrozen base model keeping a close relationship between the curves for training and validation. It converges closely with the validation accuracy, suggesting a good generalization of the model without significant overfitting. The validation accuracy remains consistent with the training accuracy after an initial climb and showing constant progress, indicating a stable model.

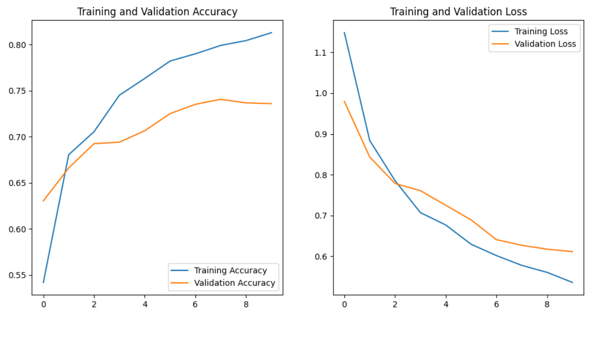


Figure 14: Accuracy and loss curves for ViT-b32 with a frozen base model.

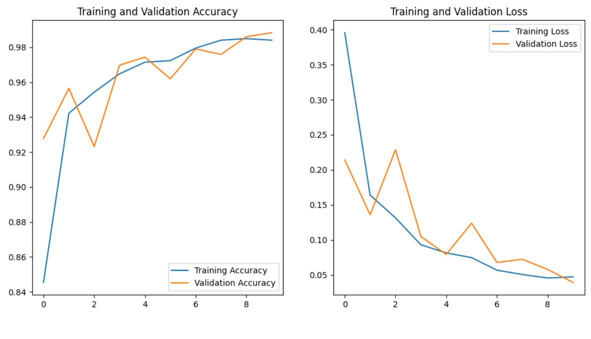


Figure 15: Accuracy and loss curves for ViT-b32.

### Quantitative Metrics

The performance metrics for ViT-b32 further demonstrate a contrast to a frozen base model implementation in classification effectiveness. For the frozen ViT-b32, precision and recall scores vary widely across classes, with particularly low recall for meningioma at 0.37 and a correspondingly low f1-score of 0.49. Conversely, the unfrozen model presents a much stronger performance across the board. It achieves perfect precision in notumor classification and nearly perfect scores in the remaining classes. The recall is also notably high, with pituitary tumors being identified correctly in all cases. This high recall translates into f1-scores that are close to 1.00 across all classes, underlining the model's balanced precision and recall. The overall accuracy of 99% stands against the 74% of the frozen variant. Again this suggests, like with the ViT-b16 model, that allowing the ViT to adjust its base layers is beneficial for this training with the in section 4.3 described architecture.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.89 | 0.63 | 0.73 | 300 |
| Meningioma | 0.72 | 0.37 | 0.49 | 306 |
| Notumor | 0.86 | 0.93 | 0.90 | 405 |
| Pituitary | 0.58 | 0.98 | 0.73 | 300 |
| **Accuracy** | | | 0.74 | 1311 |
| **Macro avg** | 0.76 | 0.73 | 0.71 | 1311 |
| **Weighted avg** | 0.77 | 0.74 | 0.73 | 1311 |

Table 7: Performance metrics for ViT-32, with a frozen base model.

| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.99 | 0.97 | 0.98 | 300 |
| Meningioma | 0.97 | 0.99 | 0.98 | 306 |
| Notumor | 1.00 | 1.00 | 1.00 | 405 |
| Pituitary | 0.99 | 0.98 | 0.99 | 300 |
| **Accuracy** | | | 0.99 | 1311 |
| **Macro avg** | 0.99 | 0.99 | 0.99 | 1311 |
| **Weighted avg** | 0.99 | 0.99 | 0.99 | 1311 |

Table 8: Performance metrics for ViT-b32.

### 4.4.4 ViT-b32 as described by Asiri et al. (2023)

To further evaluate the earlier trained models, with the in section 4.3 described two dense layers for classification, the architecture outlined by Asiri et al. (2023) was implemented as this model was trained on a similar brain tumor dataset. This setup, detailed in Appendix F, differs from the one in section 4.3 by incorporating a Flatten and BatchNormalization layer following the ViT-b32 model. The output is then passed through a ReLU-activated Dense layer with 11 neurons, followed by another BatchNormalization, and finally into a 4-neuron Dense layer for classification. As with the previously described models, this implementation, as described in the paper, was pre-trained on the ImageNet21k dataset and utilized a learning rate of 0.0001, which took 16.06 minutes to train for 10 epochs.

Because the paper described no frozen base layers, and based on the current findings, only an unfrozen model will be evaluated.

### Confusion Matrix Analysis

The confusion matrix depicted in Figure 16 displays a clear diagonal, indicating a high number of correct classifications. The relatively few misclassifications compared to the true positives imply effective performance across all classes. Notably, there are two false positives for the notumor class, which are problematic as stated earlier for a medical application. Also, as observed with the other models, this model shows the most difficulty in differentiating between glioma and meningioma classes.

### Accuracy and Loss Curves

The two graphs in Figure 17 show the training and validation accuracy and loss over the 10 trained epochs. With a strong correlation between the accuracy and loss curves. The curves resemble the unfrozen ViT-b32 model in section 4.4.3 with only the two dense layers, but with a less spiking behavior over the epochs for the validation curves.

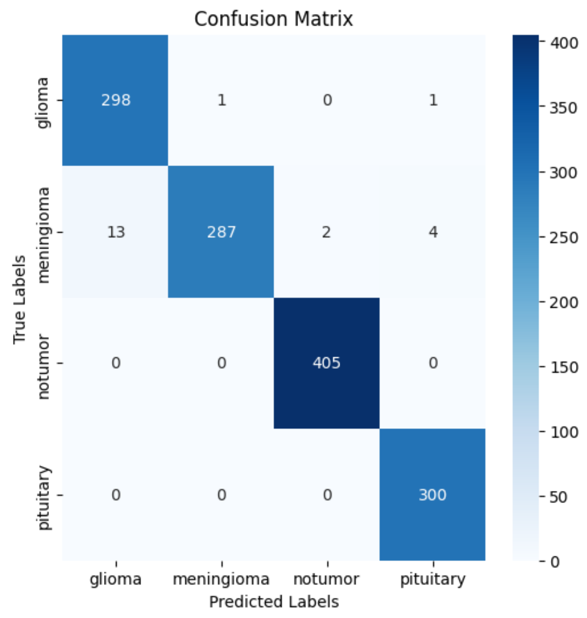


Figure 16: Confusion matrix for ViT-b32 implementation based on Asiri et al. (2023).

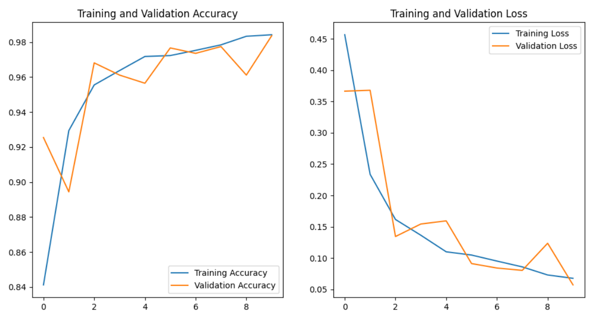


Figure 17: Accuracy and loss curves for ViT-b32 implementation based on Asiri et al. (2023).

### Quantitative Metrics

The reimplemented ViT-b32 model demonstrated comparable performance to the model in the paper of Asiri et al. (2023), achieving an overall accuracy of 98%. This is shown in Table 9. It also exhibited high precision, recall, and f1-score values across all four classes, consistently exceeding 94% for each metric. This indicates that the model can effectively identify and classify brain tumors with high confidence. But compared to the ViT-b32 model of section 4.4.3 it classified two meningioma tumors as notumor. This can also be observed in the paper suggesting a potential to optimize the architecture.

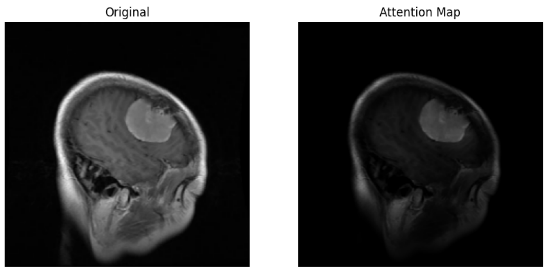
| Class | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Glioma | 0.96 | 0.99 | 0.98 | 300 |
| Meningioma | 1.00 | 0.94 | 0.97 | 306 |
| Notumor | 1.00 | 1.00 | 1.00 | 405 |
| Pituitary | 0.98 | 1.00 | 0.99 | 300 |
| **Accuracy** | | | 0.98 | 1311 |
| **Macro avg** | 0.98 | 0.98 | 0.98 | 1311 |
| **Weighted avg** | 0.98 | 0.98 | 0.98 | 1311 |

Table 9: Performance metrics for ViT-b32 implementation based on Asiri et al. (2023).

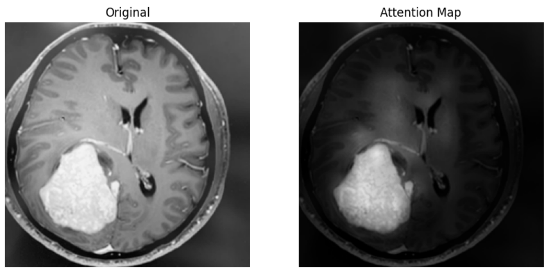
## 4.5 Attention Maps

To provide some explainability for the ViT models an attention map was generated, the code can be found in Appendix E. As illustrated by Figure 18 the models tend to focus on bright spots and areas in the image. This could be because tumors tend to be denser than the surrounding brain tissue, and they can reflect more of the magnetic field used in an MRI scan. As a result, brain tumors typically appear as white or bright areas on MRI images. This is often further enhanced by using a contrast medium. But as seen with image 18f which for example has a lot of bright spots the models are robust enough to correctly predict the notutor class.

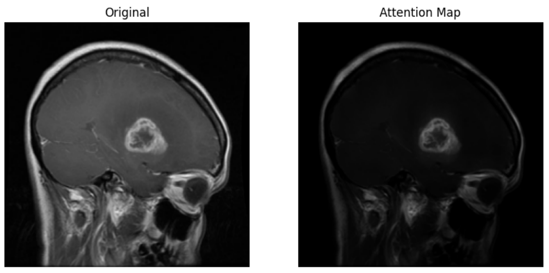
Additionally, it can be observed that the ViT models are focusing on the head structures which are also bright but can also suggest that the models are learning to detect the shape of a skull.



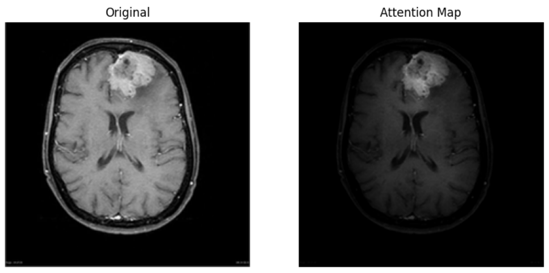
(a) Predicted Class: meningioma, Probability: 0.99 (ViT-b16)



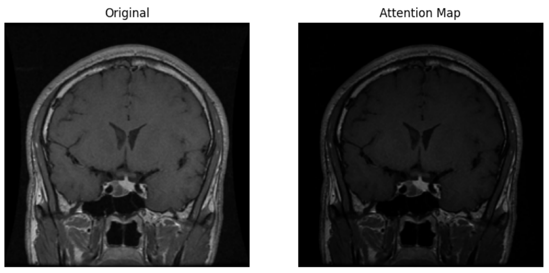
(b) Predicted Class: meningioma, Probability: 0.98 (ViT-b16)



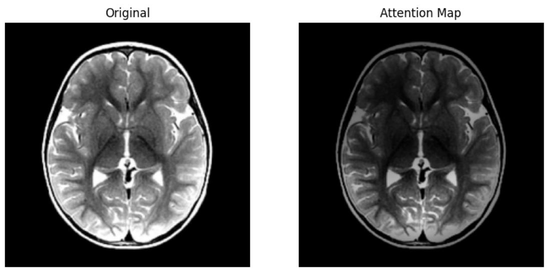
(c) Predicted Class: glioma, Probability: 0.99 (ViT-b16)



(d) Predicted Class: meningioma, Probability: 0.99 (ViT-b32)



(e) Predicted Class: pituitary, Probability: 0.99 (ViT-b32)



(f) Predicted Class: notumor, Probability: 0.99 (ViT-b32)

Figure 18: Attention Maps for ViT model predictions.

### 4.6 Results

Comparing the ResNet50V2 to the two ViT models ViT-b16 and ViT-b32, based on the architecture described in seciton 4.3, showed that all models exhibited good performance in classifying brain tumor types if the base layer where trainable. When training the models with frozen base layers the models performed significantly worse, suggesting that adjusting their base layers contributes to their effectiveness. This was especially true for the ViT models. A reason could be a to small size and depth of the dense layers following the respective frozen base model, but this was not further explored or the need to fine-tune the feature extraction for this specific task. Because of this, if not mentioned differently, the unfrozen variants are meant when mentioning a model.

The specific application and the relative costs of false positives and false negatives also need to be considered. For medical diagnoses, a false negative might be more severe than a false positive, in which case a model with higher sensitivity for critical classes might be preferred.
Based on this ResNet50V2, ViT-b16, and ViT-b32 all produce valid results, but the ViT-b32 based on the Asiri et al. (2023) architecture had two false positives which could be possibly resolved with further training. With a 1.00 f1-score, the ResNet50V2 and ViT-b32 managed to classify all 405 no-tumor test images correctly. Training speed was between those was also close with 11.42 and 11.81 minutes respectively. The ViT-b16 taking 20.33 minutes could be caused by the smaller patch size, resulting in more patches, and with that more attention weights needed to be calculated. Having double the patches than the ViT-b32, this would further be supported by the doubling of training time.

Considering accuracy and training time, the ViT-b32 model would be the suggested choice due to its highest accuracy. However, the ResNet50V2 model merits consideration when model size and complexity are factored in as its performance is competitive to the ViT model.

# 5 Conclusion

Brain tumor detection remains a challenging task with no single, clearly superior model. This seminar work explored the ResNet50V2 compared to two ViT architectures, all of which exhibited good performance in classifying brain tumor types. However, the observed overfitting for the ResNet50V2 and the fluctuating accuracy and loss curves suggest room for improvement. Future work could investigate techniques such as regularization, further data augmentation and hyperparameter tuning or the inclusion of additional datasets to enhance model generalizability. It is worth noting that all models exhibited difficulty differentiating between glioma and meningiomas, where more data could help in training the models.

## References

Abiwinanda, Nyoman, Muhammad Hanif, S Tafwida Hesaputra, Astri Handayani, and Tati Rajab Mengko (2019). “Brain tumor classification using convolutional neural network”. In: *World Congress on Medical Physics and Biomedical Engineering 2018: June 3-8, 2018, Prague, Czech Republic (Vol. 1)*. Springer, pp. 183–189.

Agarap, Abien Fred (2018). “Deep learning using rectified linear units (relu)”. In: *arXiv preprint arXiv:1803.08375*.

American Cancer Society (Dec. 2023a). *Types of Brain Tumors and Spinal Cord Tumors in Adults*. [Online; accessed 4. Dec. 2023]. URL: `https://www.cancer.org/cancer/types/brain-spinal-cord-tumors-adults/about/types-of-brain-tumors.html`.

— (Dec. 2023b). *What is a Pituitary Tumor? | American Cancer Society*. [Online; accessed 4. Dec. 2023]. URL: `https://www.cancer.org/cancer/types/pituitary-tumors/about/what-is-pituitary-tumor.html`.

Asiri, Abdullah A, Ahmad Shaf, Tariq Ali, Muhammad Ahmad Pasha, Muhammad Aamir, Muhammad Irfan, Saeed Alqahtani, Ahmad Joman Alghamdi, Ali H Alghamdi, Abdullah Fahad A Alshamrani, et al. (2023). “Advancing Brain Tumor Classification through Fine-Tuned Vision Transformers: A Comparative Study of Pre-Trained Models”. In: *Sensors* 23.18, p. 7913.

Chollet, François et al. (2015). *Keras*. `https://keras.io`.

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby (2021). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv: `2010.11929 [cs.CV]`.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2015). *Deep Residual Learning for Image Recognition*. arXiv: `1512.03385 [cs.CV]`.

— (2016). *Identity Mappings in Deep Residual Networks*. arXiv: `1603.05027 [cs.CV]`.

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

Martín Abadi et al. (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. URL: `https://www.tensorflow.org/`.

Morales, Fausto (2023). *vit-keras*. URL: `https://github.com/faustomorales/vit-keras`.

Nickparvar, Msoud (2021). *Brain Tumor MRI Dataset*. DOI: `10.34740/KAGGLE/DSV/2645886`. URL: `https://www.kaggle.com/dsv/2645886`.

Panda, Bichitrananda and Chandra Sekhar Panda (2019). “A Review on Brain Tumor Classification Methodologies”. In: *International Journal of Scientific Research in Science and Technology* 6, pp. 346–359. URL: `https://api.semanticscholar.org/CorpusID:213420293`.

Python (2023). *Python Software Foundation*. URL: `https://www.python.org/`.

Sailunaz, Kashfia, Deniz Bestepe, Sleiman Alhajj, Tansel Özyer, Jon Rokne, and Reda Alhajj (2023). “Brain tumor detection and segmentation: Interactive framework with a visual interface and feedback facility for dynamically improved accuracy and trust”. In: *Plos one* 18.4, e0284418.

Seetha, J and S Selvakumar Raja (2018). “Brain tumor classification using convolutional neural networks”. In: *Biomedical & Pharmacology Journal* 11.3, p. 1457.

Siegel, Rebecca L., Kimberly D. Miller, Nikita Sandeep Wagle, and Ahmedin Jemal (2023). “Cancer statistics, 2023”. In: *CA: A Cancer Journal for Clinicians* 73.1, pp. 17–48. DOI: https://doi.org/10.3322/caac.21763. eprint: https://acsjournals.onlinelibrary.wiley.com/doi/pdf/10.3322/caac.21763. URL: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21763.

TensorFlow (Sept. 2023a). *tf.keras.callbacks.EarlyStopping | TensorFlow v2.14.0*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/callbacks/EarlyStopping?hl=en.

— (2023b). *tf.keras.preprocessing.image.ImageDataGenerator*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/preprocessing/image/ImageDataGenerator.

— (2023c). *tf.keras.preprocessing.image.ImageDataGenerator*. URL: https://www.tensorflow.org/api_docs/python/tf/keras/Sequential.

Tummala, Sudhakar, Seifedine Kadry, Syed Ahmad Chan Bukhari, and Hafiz Tayyab Rauf (Oct. 2022). “Classification of Brain Tumor from Magnetic Resonance Imaging Using Vision Transformers Ensembling”. In: *Current Oncology* 29.10, p. 7498. DOI: 10.3390/curroncol29100590.

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). “Attention is all you need”. In: *Advances in neural information processing systems* 30.

Voon, Wingates, Yan Hum, Yee Tee, Wun-She Yap, Maheza Salim, Tian Tan, Hamam Mokayed, and Khin Wee Lai (Nov. 2022). “Performance analysis of seven Convolutional Neural Networks (CNNs) with transfer learning for Invasive Ductal Carcinoma (IDC) grading in breast histopathological images”. In: *Scientific Reports* 12, p. 19200. DOI: 10.1038/s41598-022-21848-3.

Zhao, Wayne Xin et al. (2023). *A Survey of Large Language Models*. arXiv: 2303.18223 [cs.CL].

## A Appendix Initializing ResNet50V2

```python
model_resnet50 = Sequential(name='ResNet50V2')

resnet_init = tf.keras.applications.ResNet50V2(
    include_top=False,
    weights='imagenet',
    input_shape=(image_size, image_size, 3), #Has to be 3 for RGB images after changing to ViT
    pooling='avg',  #Add global average pooling to reduce the dimensions and because ViT models typically use a form of global average pooling to aggregate the information from the sequence of transformer blocks into a single vector representation (often achieved through attention mechanisms).
    classes=num_classes
)
#resnet_init.trainable = False

model_resnet50.add(resnet_init)
model_resnet50.add(Dense(dense_neuron, activation = activation, name=f'dense_layer_{dense_neuron}'))
model_resnet50.add(Dense(num_classes, activation="softmax", name='output_layer'))
model_resnet50.compile(optimizer=adam_optimizer, loss="categorical_crossentropy", metrics=['accuracy'])
model_resnet50.summary()
```

## B Appendix build_vit_model function

```python
def build_vit_model(vit_function, image_size, num_classes, dense_neuron, rate, activation, freeze_base_model=False):
    """
    Build and compile the Vision Transformer model.

    Args:
    - vit_function: Use a base model from vit_keras (example: vit.vit_b32 or vit.vit_b16).
    - image_size: Size of the input images.
    - num_classes: Number of classes for classification.
    - rate: Learning rate for the optimizer.
    - activation: Activation function for the dense layer.

    Returns:
    - Compiled Keras model.
    """
    vit_model = Sequential()

    #Initialize the ViT model
    vit_init = vit_function(
        image_size=image_size,
        activation='softmax',
        pretrained=True,
        include_top=False,
        pretrained_top=False,
        classes=num_classes,
        weights="imagenet21k"
    )

    vit_init.trainable = not freeze_base_model

    #Layers for the classification
    vit_model.add(vit_init)
    vit_model.add(Dense(dense_neuron, activation=activation, name=f'dense_layer_{dense_neuron}'))
    vit_model.add(Dense(num_classes, activation="softmax", name='output_layer'))


    adam_optimizer = Adam(learning_rate=rate)
    vit_model.compile(optimizer=adam_optimizer, loss="categorical_crossentropy", metrics=['accuracy'])
    vit_model.summary()

    return vit_model
```

## C Appendix train_model function

```python
def train_model(model, train_generator, validation_generator, steps_per_epoch,
    validation_steps, epochs, gpu_device='/gpu:0'):
    """
    Train a model.

    Args:
    - model: Compiled model to be trained.
    - train_generator: Training data generator.
    - validation_generator: Validation data generator.
    - steps_per_epoch: Number of steps per epoch for training.
    - validation_steps: Number of steps per epoch for validation.
    - epochs: Number of epochs to train the model for.
    - gpu_device: GPU device to be used for training (default is '/gpu:0').

    Returns:
    - Training history object.
    """
    callback = tf.keras.callbacks.EarlyStopping(
        monitor="val_loss",
        min_delta=0.01,
        patience = 5,
        restore_best_weights = True,
        verbose = 1
    )

    start = time.time()
    with tf.device(gpu_device):
        history = model.fit(
            train_generator,
            steps_per_epoch=steps_per_epoch,
            epochs=epochs,
            validation_data=validation_generator,
            validation_steps=validation_steps,
            callbacks=callback,
            use_multiprocessing=True,
            workers=4  #Virutal Workers on Kaggle
        )
    stop = time.time()
    print(f'Training on GPU took: {(stop - start) / 60} minutes')

    return history
```

## D Appendix evaluate_and_plot function

```python
def evaluate_and_plot(model, validation_generator, history):
    """
    Evaluate the model, plot the confusion matrix and training curves.

    Args:
    - model: Trained Keras model.
    - test_generator: Generator to extract the classes and use a random test image for
      classification.
    - history: Training history object.
    """
    #Predictions and true classes
    predicted_classes = np.argmax(model.predict(validation_generator, steps=
    validation_generator.n // validation_generator.batch_size + 1), axis=1)
    true_classes = validation_generator.classes

    #Reverse mapping from index to class name
    index_to_class = {v: k for k, v in validation_generator.class_indices.items()}
    # Sort class labels to match the order of the class indices
    sorted_class_labels = [index_to_class[idx] for idx in sorted(index_to_class)]

    #Confusion Matrix
    confusionmatrix = confusion_matrix(true_classes, predicted_classes)
    plt.figure(figsize=(6, 6))
    sns.heatmap(confusionmatrix, cmap='Blues', annot=True, fmt='d', cbar=True,
    xticklabels=sorted_class_labels, yticklabels=sorted_class_labels)
    plt.xlabel('Predicted Labels')
    plt.ylabel('True Labels')
    plt.title('Confusion Matrix')
    plt.show()

    #Classification Report
    print(classification_report(true_classes, predicted_classes, target_names=
    sorted_class_labels))

    #Training Curves
    epochs = len(history.epoch)
    accuracy = history.history['accuracy']
    val_acc = history.history['val_accuracy']
    loss = history.history['loss']
    val_loss = history.history['val_loss']
    epochs_range = range(epochs)

    #Plotting
    plt.figure(figsize=(12, 6))
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, accuracy, label='Training Accuracy')
    plt.plot(epochs_range, val_acc, label='Validation Accuracy')
    plt.legend(loc='lower right')
    plt.title('Training and Validation Accuracy')

    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, loss, label='Training Loss')
    plt.plot(epochs_range, val_loss, label='Validation Loss')
    plt.legend(loc='upper right')
    plt.title('Training and Validation Loss')
    plt.show()
```

## E Appendix attention_map function

```python
def attention_map(model, img_path, generator):
    """
    Extract the attention map for a ViT model run with a random image.

    Args:
    - model: A ViT model from vit_keras.
    - img_path: Path to the images to be tested, includes subfolders.
    - generator: Generator to extract the classes for classification.
    """

    #Extract ViT model out of the Sequential for the attention_map function
    base_model = model.get_layer(model.layers[0].name)
    predict_model = model

    #Get class indices and create a mapping from indices to class names
    class_indices = generator.class_indices
    index_to_class = {v: k for k, v in class_indices.items()}

    all_images = []
    for root, dirs, files in os.walk(img_path):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg')):
                all_images.append(os.path.join(root, file))

    #Select a random image from the img_path and convert it to the right format
    random_image_path = random.choice(all_images)
    print(f"Selected Image: {random_image_path}")

    #Preprocess the random image
    image = Image.open(random_image_path)
    image = image.resize((224, 224))
    if image.mode != 'RGB':
        image = image.convert('RGB')
    image_array = np.array(image)
    image_array = image_array * (1./255)

    #Predict the class
    prediction = predict_model.predict(np.expand_dims(image_array, axis=0))
    predicted_class_name = index_to_class[np.argmax(prediction, axis=1)[0]]
    predicted_probability = np.max(prediction)
    class_indices = train_generator.class_indices

    print(f"Predicted Class: {predicted_class_name}, Probability: {
    predicted_probability}")

    #Generate the attention map
    attention_map = visualize.attention_map(model=base_model, image=np.array(image))

    #Power Law (gamma) transformation
    gamma = 1.5  #Value to control the enhancement
    attention_map = np.array(255 * (attention_map / 255) ** gamma, dtype='uint8')

    #Plotting
    fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(10, 10))
    ax1.axis('off')
    ax2.axis('off')
    ax1.set_title('Original')
    ax2.set_title('Attention Map')
    _ = ax1.imshow(image)
    _ = ax2.imshow(attention_map, cmap='hot')

    plt.show()
```

## F Appendix Implementation of Asiri et al. (2023)

```
vit_b32_paper_model = Sequential()

vit_b32_paper_init = vit.vit_b32(
        image_size = image_size,
        activation = 'softmax',
        pretrained = True,
        include_top = False,
        pretrained_top = False,
        classes = num_classes,
        weights = "imagenet21k"
)

vit_b32_paper_model.add(vit_b32_paper_init)
vit_b32_paper_model.add(Flatten())
vit_b32_paper_model.add(BatchNormalization())
vit_b32_paper_model.add(Dense(11, activation = activation))
vit_b32_paper_model.add(BatchNormalization())
vit_b32_paper_model.add(Dense(num_classes, activation="softmax", name="output_layer"))
adam_optimizer = Adam(learning_rate=0.0001)
vit_b32_paper_model.compile(optimizer=adam_optimizer, loss="categorical_crossentropy",
     metrics=['accuracy'])

start = time.time()
with tf.device('/gpu:0'):
    history = vit_b32_paper_model.fit(
    train_generator,
    steps_per_epoch=steps_per_epoch,
    epochs=epochs,
    validation_data=test_generator,
    validation_steps=validation_steps
)
stop = time.time()
print(f'Training on GPU took: {(stop-start)/60} minutes')
```

## Eidesstattliche Erklärung

Hiermit versichere ich, die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie die Zitate deutlich kenntlich gemacht zu haben.
Ich erkläre weiterhin, dass die vorliegende Arbeit in gleicher oder ähnlicher Form noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Würzburg, den 8. Dezember 2023 Nicholas Dziuba