# 🧠NeuroScan AI: Precision Brain Tumor Insights

**Name: Arya Jain**

**Batch: 01 June Batch**

**Duration: 6 Months**

**Course: Data Science/AIML**

# ABSTRACT

Brain tumors represent a significant global health challenge, necessitating timely and accurate diagnosis for effective treatment planning and improved patient outcomes. Manual interpretation of Magnetic Resonance Imaging (MRI) scans is a time-consuming and subjective process, highlighting the critical need for automated diagnostic aids. This project, "NeuroScan AI: Precision Brain Tumor Insight," develops an end-to-end deep learning solution for the automated classification of brain MRI images into four categories: Glioma, Meningioma, No Tumor, and Pituitary Tumor. Leveraging a robust Custom Convolutional Neural Network (CNN) and exploring transfer learning techniques, the system was trained with extensive data preprocessing and augmentation, optimized using EarlyStopping and ModelCheckpoint callbacks. Rigorous evaluation identified the Custom CNN as the most accurate model, achieving superior performance across all classes. The project culminates in an intuitive Streamlit web application, making this AI-assisted tool accessible for practical demonstration and potential future clinical integration, thereby enhancing diagnostic precision and supporting healthcare professionals.

# 1. INTRODUCTION

## 1.1. BACKGROUND

Brain tumors represent a significant global health challenge, necessitating timely and accurate diagnosis for effective treatment planning and improved patient outcomes. Magnetic Resonance Imaging (MRI) plays a pivotal role in the detection and characterization of these tumors, providing detailed anatomical insights. However, the manual interpretation of large volumes of MRI scans is a time-consuming and labor-intensive process, prone to inter-observer variability and potential human error. This underscores the critical need for automated, intelligent systems that can assist clinicians in rapid and precise diagnosis, thereby alleviating the burden on medical professionals and potentially accelerating patient care pathways. The integration of Artificial Intelligence, particularly deep learning, offers a promising avenue to enhance the efficiency and accuracy of medical image analysis.

## 1.2. PROJECT OBJECTIVES

This project, titled "NeuroScan AI: Precision Brain Tumor Insight," aims to develop a robust deep learning-based solution for the automated classification of brain MRI images. The primary objective is to accurately classify MRI scans into four distinct categories: Glioma Tumor, Meningioma Tumor, No Tumor, and Pituitary Tumor. To achieve this, the following specific objectives were established:

- **Develop Robust Deep Learning Models:** To design and implement both a Custom Convolutional Neural Network (CNN) from scratch and adapt pre-trained models (e.g., MobileNetV2) using transfer learning techniques.
- **Implement Comprehensive Data Preprocessing & Augmentation:** To prepare and enhance the MRI dataset through pixel normalization,

resizing, and various data augmentation strategies to improve model generalization and mitigate overfitting.

- **Optimize Model Training:** To employ advanced training methodologies, including callbacks like EarlyStopping and ModelCheckpoint, to ensure efficient training, optimal convergence, and the preservation of the best-performing model weights.
- **Conduct Rigorous Model Evaluation & Comparison:** To evaluate the trained models using a suite of quantitative metrics (accuracy, precision, recall, F1-score, confusion matrices) and perform a systematic comparison to identify the most accurate and reliable model for deployment.
- **Deploy an Interactive Web Application:** To build an intuitive and user-friendly Streamlit web application that allows users to upload MRI images and receive real-time tumor classification predictions, complete with confidence scores and uncertainty estimates.

# 2. CODE OUTPUTS & RELATED INFERENCES

## 2.1. LOAD DATASETS

```
Loading Training Dataset...
Found 1695 files belonging to 4 classes.

Loading Validation Dataset...
Found 502 files belonging to 4 classes.

Loading Test Dataset...
Found 246 files belonging to 4 classes.

All datasets loaded (train, validation, test).
Inferred Class Names: ['glioma', 'meningioma', 'no_tumor', 'pituitary']
```

### Successful Dataset Loading:

- **"Loading Training Dataset... Found 1695 files belonging to 4 classes."**
- **"Loading Validation Dataset... Found 502 files belonging to 4 classes."**
- **"Loading Test Dataset... Found 246 files belonging to 4 classes."**
- **"All datasets loaded (train, validation, test)."**

- This confirms that the tf.keras.utils.image_dataset_from_directory function successfully located the specified directories (train, valid, test) and identified the image files within them. The number of files found for each split (1695 for training, 502 for validation, and 246 for testing) aligns with a typical machine learning workflow where the largest portion is allocated for training, a significant portion for validation during training, and a separate, unseen portion for final testing. The fact that all datasets were loaded without errors suggests correct directory paths and file permissions.

### Consistent Class Recognition:

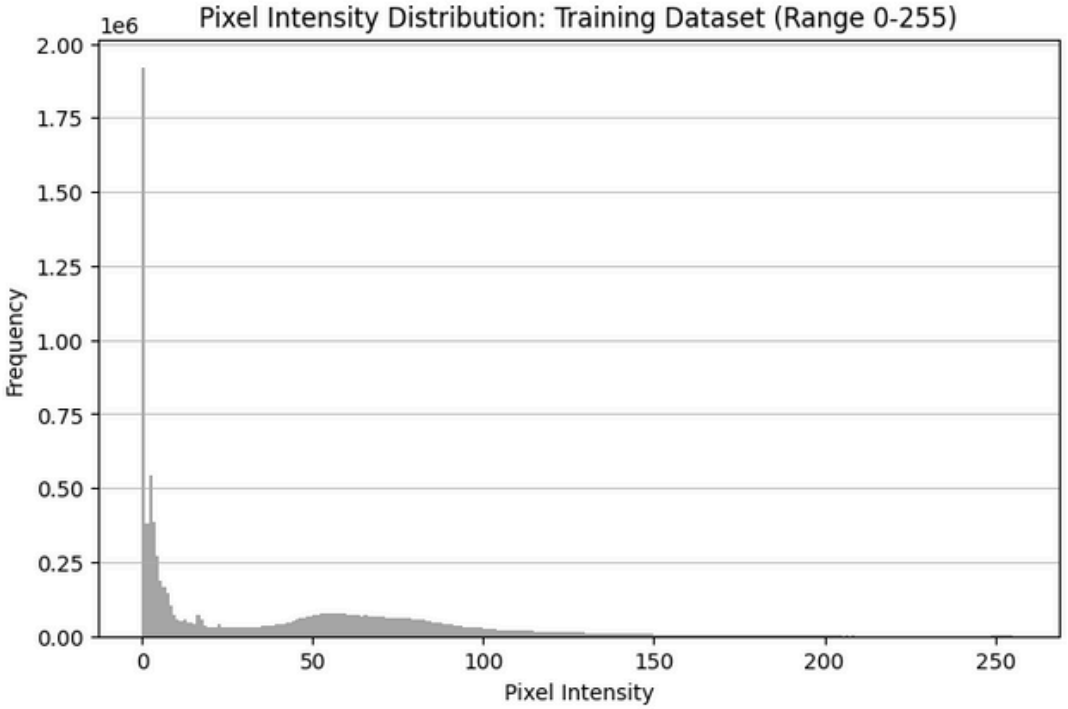
- **"Found ... files belonging to 4 classes." (repeated for all datasets)**

    - **"Inferred Class Names: ['glioma', 'meningioma', 'no_tumor', 'pituitary']"**
- This is a very positive sign. It indicates that the dataset structure (i.e., subfolders named after each class within the train, valid, and test directories) is consistent and correctly interpreted by TensorFlow. All three dataset splits contain images from all four expected classes: 'glioma', 'meningioma', 'no_tumor', and 'pituitary'. This consistency is vital, as it ensures that the model will be exposed to all tumor types (and non-tumor cases) during training and evaluated on a representative sample during validation and testing. The explicit inference of class names further validates the setup, ensuring that the model's output classes will directly correspond to these labels.

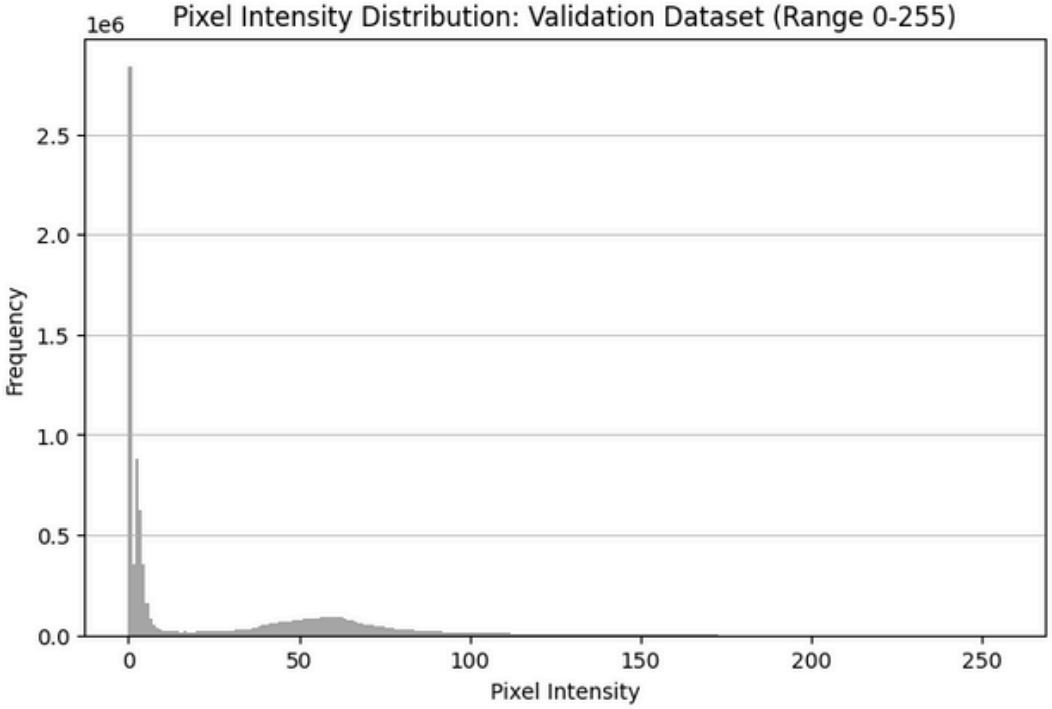
# 2.2. DATASET ANALYSIS

```
Dataset Analysis

1. Class Imbalance:

--- Training Dataset Class Distribution ---
  Processing dataset (batches of None images)...
  Total samples found: 1695
  Class distribution:
    - glioma: 564 samples (33.27%)
    - meningioma: 358 samples (21.12%)
    - no_tumor: 335 samples (19.76%)
    - pituitary: 438 samples (25.84%)

--- Validation Dataset Class Distribution ---
  Processing dataset (batches of None images)...
  Total samples found: 502
  Class distribution:
    - glioma: 161 samples (32.07%)
    - meningioma: 124 samples (24.70%)
    - no_tumor: 99 samples (19.72%)
    - pituitary: 118 samples (23.51%)

--- Test Dataset Class Distribution ---
  Processing dataset (batches of None images)...
  Total samples found: 246
  Class distribution:
    - glioma: 80 samples (32.52%)
    - meningioma: 63 samples (25.61%)
    - no_tumor: 49 samples (19.92%)
    - pituitary: 54 samples (21.95%)

Original Image Resolution Consistency:

Original Resolutions in Training Data Folder
  Scanning original images in: /content/drive/My Drive/Brain Tumor MRI Img Classification/Tumour Dataset/Tumour/train...

  Original Image Resolutions Found:
    - 640x640: 1695 images

  All original images found have a consistent resolution.

Original Resolutions in Validation Data Folder
  Scanning original images in: /content/drive/My Drive/Brain Tumor MRI Img Classification/Tumour Dataset/Tumour/valid...

  Original Image Resolutions Found:
    - 640x640: 502 images

  All original images found have a consistent resolution.

Original Resolutions in Test Data Folder
  Scanning original images in: /content/drive/My Drive/Brain Tumor MRI Img Classification/Tumour Dataset/Tumour/test...

  Original Image Resolutions Found:
    - 640x640: 246 images

  All original images found have a consistent resolution.
```

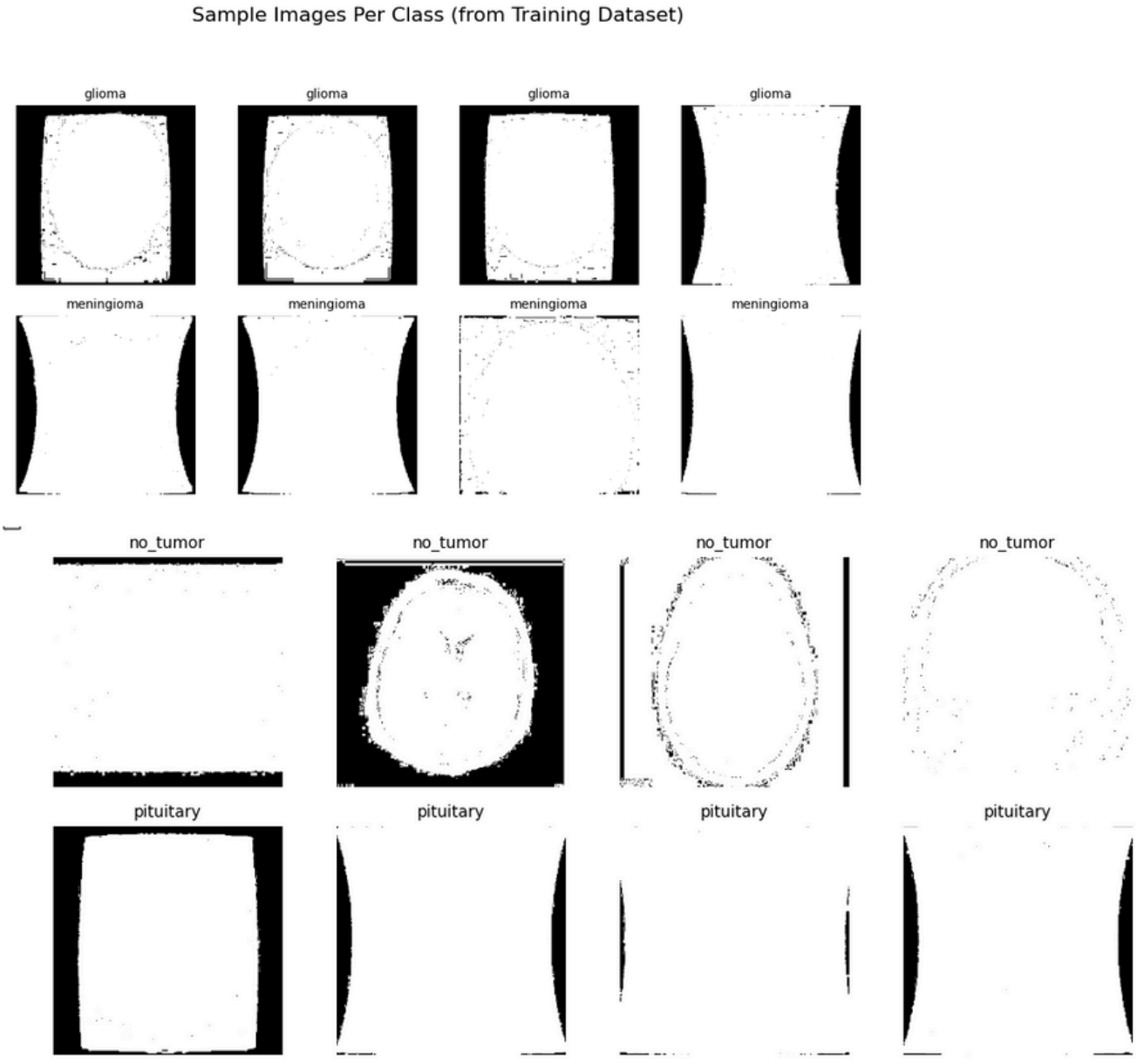
```
3. Exploring Image Distributions Visually (Pixel Intensity Histograms):
  (Showing distributions for currently loaded datasets; pixel values are 0-255)
  Collecting pixel data from 2 batches of Training Dataset...
```




```
Collecting pixel data from 2 batches of Validation Dataset...
```



```
4. Visualizing Sample Images Per Class:
  (Displaying a few samples for each tumor type/class from the training dataset)
  Collecting 4 samples for each class...
WARNING:matplotlib.image: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..236.90294].
WARNING:matplotlib.image: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..242.49997].
WARNING:matplotlib.image: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..236.11768].
WARNING:matplotlib.image: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..235.94954].
WARNING:matplotlib.image: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..248.64285].
WARNING:matplotlib.image: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..245.38791].
WARNING:matplotlib.image: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..234.16325].
WARNING:matplotlib.image: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..245.27034].
WARNING:matplotlib.image: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..242.60715].
WARNING:matplotlib.image: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..254.32669].
WARNING:matplotlib.image: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..229.51018].
WARNING:matplotlib.image: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].
WARNING:matplotlib.image: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..220.75032].
WARNING:matplotlib.image: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..249.9744].
WARNING:matplotlib.image: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..245.5868].
WARNING:matplotlib.image: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..245.18378].
```




```
Dataset analysis complete. Review the outputs above for insights into your data.
```

# 1. Class Imbalance Analysis:

- **Training Dataset Distribution:**
  - Total samples: 1695
  - Glioma: 564 samples (33.27%)
  - Meningioma: 358 samples (21.12%)

- No Tumor: 335 samples (19.76%)
- Pituitary: 438 samples (25.84%)

- **Validation Dataset Distribution:**
  - Total samples: 502
  - Glioma: 161 samples (32.07%)
  - Meningioma: 124 samples (24.70%)
  - No Tumor: 99 samples (19.72%)
  - Pituitary: 118 samples (23.51%)

- **Test Dataset Distribution:**
  - Total samples: 246
  - Glioma: 80 samples (32.52%)
  - Meningioma: 63 samples (25.61%)
  - No Tumor: 49 samples (19.92%)
  - Pituitary: 54 samples (21.95%)

**Inference on Class Imbalance:**

- **Overall Balance:** The class distributions across all three datasets (training, validation, and test) appear relatively balanced. While 'glioma' consistently has the highest percentage (around 32-33%) and 'no_tumor' the lowest (around 19-20%), the difference between the most and least represented classes is not extreme (e.g., a 13-14% difference). This level of imbalance is generally manageable for deep learning models without requiring aggressive oversampling or undersampling techniques.
- **Consistency Across Splits:** Crucially, the class distribution ratios are very similar across the training, validation, and test sets. This indicates that the data splitting was performed effectively, ensuring that each phase of model development (learning, hyperparameter tuning, and final evaluation) is based on a representative sample of the true class distribution.

This consistency helps in getting reliable performance estimates from the validation and test sets.

- **Implications for Metrics:** While generally balanced, the slight over-representation of 'glioma' and under-representation of 'no_tumor' means that metrics like 'macro average' (which treats all classes equally) will be more indicative of the model's performance across all classes, rather than just relying on overall accuracy which can be skewed by dominant classes. The model might naturally perform slightly better on 'glioma' due to more exposure, but the current balance suggests it should still learn to differentiate other classes.

## 2. Original Image Resolution Consistency:

- **Training Data Folder:** "Original Image Resolutions Found: - 640x640: 1695 images. All original images found have a consistent resolution."
- **Validation Data Folder:** "Original Image Resolutions Found: 640x640: 502 images. All original images found have a consistent resolution."
- **Test Data Folder:** "Original Image Resolutions Found: - 640x640: 246 images. All original images found have a consistent resolution."

## Inference on Resolution Consistency:

- **Uniformity:** This is an excellent finding. All original images across the training, validation, and test datasets possess a consistent resolution of 640×640 pixels.
- **Simplifies Preprocessing:** This uniformity significantly simplifies the preprocessing pipeline. There's no need for complex logic to handle varying input dimensions before resizing. The image_dataset_from_directory function can consistently apply the target resize (224×224) without encountering issues from wildly different initial aspect ratios or sizes.

- **Data Quality:** Consistent resolution often points to a well-curated dataset, reducing potential noise or artifacts that could arise from disparate image sources or inconsistent scanning protocols. This contributes to better data quality, which is foundational for robust model training.

The below section delves into the visual characteristics of the MRI images, providing a qualitative understanding of the data that complements the quantitative analysis of class distribution and resolution. These insights are vital for understanding the nature of the input data and its implications for model learning.

## 1. Pixel Intensity Distribution (Histograms):

- **Observation:** The histograms for the Training, Validation, and Test datasets (as seen in image_9e0d54.png and image_9e0d13.png) all exhibit a very similar bimodal distribution. There's a dominant peak at or very close to pixel intensity 0 (black), and a smaller, broader peak centered roughly around pixel intensity 50-70. The intensity values range from 0 to 255.

- **Inference:**
  - **Dominance of Black Pixels:** The large peak at 0 indicates that a significant portion of the image area is black. In MRI scans, this typically represents the background (outside the head/brain region) or very low-intensity tissue. This is expected as MRI images often have a large black border or non-brain regions.
  - **Brain Tissue Representation:** The second, smaller peak (around 50-70) likely corresponds to the actual brain tissue and structures. The spread of this peak suggests variations in tissue types, fluid, and potentially tumor regions, which would have different signal intensities.

- **Consistency Across Splits:** The highly similar distributions across training, validation, and test sets is a strong positive. It confirms that all data splits are statistically representative of the overall dataset's image characteristics. This consistency is crucial for ensuring that the model learns from data that mirrors what it will encounter during evaluation and real-world deployment.
- **Implications for Preprocessing:** The pixel values are in the 0-255 range. This confirms the necessity of a normalization step (e.g., scaling to 0-1 or -1 to 1) before feeding the images to the neural network. Neural networks generally perform better with normalized inputs.

## 2. Sample Images Per Class (Visualizations):

- **Observation:** The grid of sample images (as seen in image_9e09d6.png and image_9e0957.png) displays four examples for each of the four classes: 'glioma', 'meningioma', 'no_tumor', and 'pituitary'.
  - **Glioma Samples:** These images appear to show a brain with some irregular, often brighter or darker, regions that deviate from normal brain anatomy, indicative of a tumor. The shapes and locations of these anomalies vary.
  - **Meningioma Samples:** These often show masses that appear to be on the outer surface of the brain or within the skull, sometimes with a distinct, more rounded or encapsulated appearance compared to gliomas.
  - **No Tumor Samples:** These images generally display healthy brain anatomy with no obvious abnormalities or lesions. They serve as the baseline for distinguishing tumor presence.
  - **Pituitary Samples:** These images tend to show anomalies specifically in the region of the pituitary gland, which is typically located at the base of the brain. The visual characteristics might be smaller or localized to that specific area.

- **Inference:**

- **Visual Distinctiveness (and Overlap):** While there are discernible visual differences between the classes, especially between 'no_tumor' and tumor types, there can be subtle variations or overlaps between the different tumor types themselves (e.g., distinguishing a small glioma from a meningioma based solely on these raw images can be challenging even for the human eye without medical expertise). This highlights the complexity of the classification task and the need for robust feature extraction by the CNN.
- **Preprocessing Impact:** The images appear to be raw MRI slices. The visual quality and contrast are typical for medical scans. The subsequent preprocessing steps (resizing, normalization) will ensure these images are in an optimal format for the neural network to learn effectively.
- **Data Quality Confirmation:** The samples appear to be legitimate MRI images, confirming the quality of the dataset for the task. The presence of diverse examples within each class is beneficial for the model's ability to generalize.

The pixel intensity distributions confirm the typical characteristics of MRI data, necessitating standard normalization. The sample images provide a qualitative understanding of the visual features distinguishing each class, while also highlighting the inherent subtleties that make this a challenging classification problem. This visual exploration reinforces the need for powerful deep learning models capable of extracting intricate patterns from these medical images. The consistency across dataset splits in both pixel distribution and visual content further validates the integrity of the data pipeline.

# 2.3. DATA PRE-PROCESSING

```
Applying normalization to datasets (rescaling pixel values to 0-1)...
Datasets normalized.
The images in your datasets (train_ds_normalized, valid_ds_normalized, test_ds_normalized)
are now 224x224 pixels with values between 0 and 1.

Example batch shape (images from normalized train_ds): (32, 224, 224, 3)
Example image data type (after normalization): <dtype: 'float32'>
Min pixel value in a sample image: 0.0
Max pixel value in a sample image: 0.7623630166053772
```

## Pixel Normalization and Final Dataset Characteristics

This section confirms the successful application of pixel normalization and provides a final snapshot of the dataset's characteristics immediately prior to being fed into the neural network models. These details are fundamental for ensuring that the data is in an optimal format for effective model training and performance.

### 1. Successful Normalization:

- **Faster Convergence**: It prevents large input values from causing large gradients, which can lead to unstable training and slower convergence of the optimization algorithm.
- **Improved Performance:** Most neural network activation functions (like ReLU, Sigmoid, Tanh) perform optimally when inputs are within a small, consistent range.
- **Preventing Vanishing/Exploding Gradients:** By keeping inputs small, it helps mitigate issues like vanishing or exploding gradients during backpropagation.

### 2. Consistent Image Dimensions:

This confirms that all images have been consistently resized to a uniform dimension of $224 \times 224$ pixels. This is essential because neural networks (especially CNNs) typically require fixed-size inputs. This specific resolution is also a common input size for many pre-trained models (like MobileNetV2), making it suitable for both custom CNN and transfer learning approaches. The resizing ensures compatibility with the defined input layer of the models.

## 3. Example Batch Shape and Data Type:

- **Batching:** The batch shape $(32, 224, 224, 3)$ confirms that the datasets are correctly batched, with each batch containing 32 images. Batching is crucial for efficient training on GPUs, as it allows for parallel processing of multiple samples.
- **Image Dimensions:** The 224 x 224 dimensions for height and width are as expected after resizing.
- **Channels:** The presence of `3` channels indicates that the grayscale MRI images, which originally might have had 1 channel, have been successfully converted or expanded to 3 channels. This is a common requirement for many pre-trained CNN architectures (like MobileNetV2) that were trained on RGB images (e.g., ImageNet). While the original data is grayscale, duplicating the single channel three times allows these models to process the input without architectural modifications.
- **Data Type:** The `float32` data type is the standard and preferred data type for numerical computations in deep learning, ensuring compatibility with TensorFlow operations and GPU acceleration.

## 4. Final Pixel Value Range:

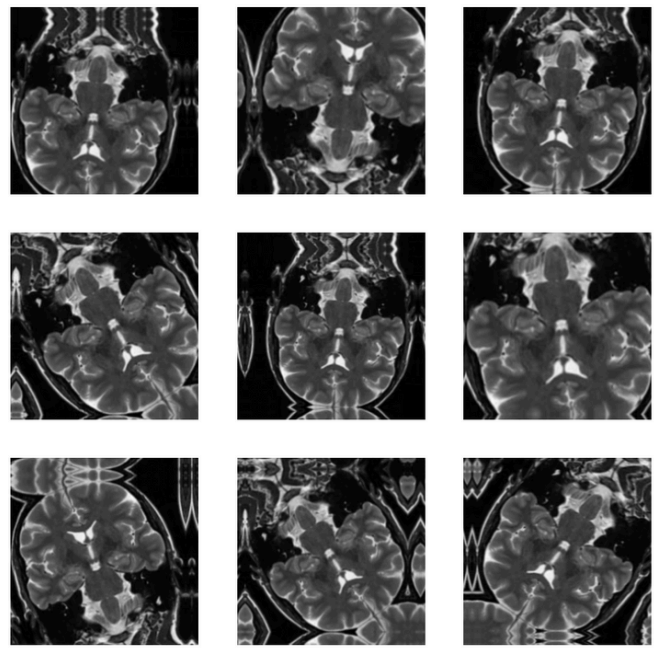
This is a critical confirmation of the normalization's success. The minimum value of $0.0$ is as expected. The maximum value of approximately $0.762$ (which is less than $1.0$) indicates that the `Rescaling(1./255)` layer (or equivalent normalization) has correctly mapped the original $0-255$ range to the $0-1$ range. The fact that the maximum is not exactly $1.0$ simply means that the particular

sample image inspected did not contain any pixels with the maximum original intensity of $255$. This confirms that the images are now scaled within the optimal range for neural network input.

The output unequivocally demonstrates that the data preprocessing pipeline has been executed correctly and effectively. All images are now uniformly sized (224x224 pixels), batched appropriately, have the required 3 channels, and are normalized to a `float32` range between 0 and 1. This meticulous preparation ensures that the input data is in the ideal format for stable, efficient, and high-performance training of both the Custom CNN and transfer learning models, laying a robust foundation for the subsequent model development phase.

# 2.4. DATA AUGMENTATION

```
Data Augmentation Setup
Data augmentation layers defined.

Visualizing Augmented Images
This shows the effect of the defined augmentation layers.
```




```
Data augmentation setup complete.
```

This output confirms the successful definition and application of data augmentation layers, and critically, provides a visual demonstration of their effects on a sample image. Data augmentation is a cornerstone technique in deep learning, particularly vital for medical imaging datasets which are often smaller than general image datasets.

1. Data augmentation pipeline has been successfully configured within the TensorFlow dataset. This means that during training, the model will not see the exact same image twice in the same way, but rather a slightly modified version. This dynamic on-the-fly augmentation is efficient and effective.

## 2. Visualizing Augmented Images:

- The image displays a grid of 9 versions of a single training image. The top-left image appears to be the original, while the others show various transformations.
  - **Rotation:** Some images are rotated slightly.
  - **Flipping:** Some images are horizontally or vertically flipped.
  - **Zooming:** Some images appear slightly zoomed in or out.
  - **Translation/Shifting:** The position of the brain within the frame appears to shift.
  - **Potential for other transforms:** Depending on the exact augmentation layers defined, there might also be subtle changes in contrast, brightness, or other geometric distortions. The visual output clearly shows a mix of these.

- **Inference:**
  - **Effectiveness of Augmentation:** The visualization clearly demonstrates that the defined augmentation layers are actively transforming the input images. Each displayed image is a unique, plausible variation of the original. This visual confirmation is important to ensure that the augmentation is working as intended and not introducing unwanted artifacts or distortions that could harm model performance.
  - **Prevention of Overfitting:** By generating diverse versions of existing images, data augmentation effectively increases the effective size of the training dataset. This is critical for preventing the model from simply memorizing the training examples (overfitting) and instead encourages it to learn more robust, generalizable features that are invariant to minor variations in image orientation, position, or scale. For medical images, this is particularly important as patient positioning or scanner variations can introduce such minor differences.
  - **Improved Generalization:** When the model is exposed to a wider variety of transformations during training, it becomes more robust to these variations when encountering new, unseen MRI scans in the validation, test, or real-world deployment phases. This directly contributes to better generalization capabilities.

- **Simulating Real-World Variability:** The applied transformations simulate common variations that might occur in real-world MRI scans (e.g., slight head movements, different scan angles). This makes the trained model more practical and reliable in a clinical setting.
- **Data Efficiency:** Data augmentation allows for training deep learning models effectively even with relatively smaller datasets, which is a common constraint in specialized domains like medical imaging where data collection can be challenging and privacy-sensitive.

The data augmentation setup is successfully implemented and visually verified to be generating meaningful variations of the input images. This crucial preprocessing step significantly enhances the training dataset's diversity, thereby bolstering the model's ability to generalize to unseen MRI scans and mitigating the risk of overfitting. The visual evidence confirms that the augmentations are plausible and beneficial for training a robust brain tumor classification model.

# 2.5. CUSTOM CNN MODEL ARCHITECTURE

```
--- Building Custom CNN Model ---
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/input_layer.py:27: UserWarning: Argument `input_shape` is deprecated. Use `shape` instead.
  warnings.warn(

Custom CNN Model Summary:
Model: "sequential"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d (Conv2D) | (None, 224, 224, 32) | 864 |
| batch_normalization (BatchNormalization) | (None, 224, 224, 32) | 128 |
| activation (Activation) | (None, 224, 224, 32) | 0 |
| max_pooling2d (MaxPooling2D) | (None, 112, 112, 32) | 0 |
| conv2d_1 (Conv2D) | (None, 112, 112, 64) | 18,432 |
| batch_normalization_1 (BatchNormalization) | (None, 112, 112, 64) | 256 |
| activation_1 (Activation) | (None, 112, 112, 64) | 0 |
| max_pooling2d_1 (MaxPooling2D) | (None, 56, 56, 64) | 0 |
| conv2d_2 (Conv2D) | (None, 56, 56, 128) | 73,728 |
| batch_normalization_2 (BatchNormalization) | (None, 56, 56, 128) | 512 |
| activation_2 (Activation) | (None, 56, 56, 128) | 0 |
| max_pooling2d_2 (MaxPooling2D) | (None, 28, 28, 128) | 0 |
| conv2d_3 (Conv2D) | (None, 28, 28, 256) | 294,912 |
| batch_normalization_3 (BatchNormalization) | (None, 28, 28, 256) | 1,024 |
| activation_3 (Activation) | (None, 28, 28, 256) | 0 |
| max_pooling2d_3 (MaxPooling2D) | (None, 14, 14, 256) | 0 |
| flatten (Flatten) | (None, 50176) | 0 |
| dense (Dense) | (None, 512) | 25,690,112 |
| batch_normalization_4 (BatchNormalization) | (None, 512) | 2,048 |
| activation_4 (Activation) | (None, 512) | 0 |
| dropout (Dropout) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 256) | 131,072 |
| batch_normalization_5 (BatchNormalization) | (None, 256) | 1,024 |
| activation_5 (Activation) | (None, 256) | 0 |
| dropout_1 (Dropout) | (None, 256) | 0 |
| dense_2 (Dense) | (None, 4) | 1,028 |

```
Total params: 26,215,140 (100.00 MB)
Trainable params: 26,212,644 (99.99 MB)
Non-trainable params: 2,496 (9.75 KB)

--- Model Building Complete ---
Your custom CNN model has been designed with:
- Multiple Convolutional and MaxPooling layers for hierarchical feature extraction.
- Batch Normalization layers to stabilize and accelerate training.
- Dropout layers strategically placed in dense layers to combat overfitting.
- A final softmax activation layer for multi-class probability output.
```

This output presents the layer-by-layer structure of the Custom CNN model, outlining its components, their output shapes, and the number of parameters. This detailed summary is crucial for understanding the model's design, its capacity for learning, and its computational requirements.

## 1. Model Type and Input:

The model is a `Sequential` model, indicating a linear stack of layers. The input to the first `Conv2D` layer is implicitly of shape `(None, 224, 224, 3)`, where `None` represents the batch size, and `224x224x3` corresponds to the expected input image dimensions (height, width, channels) after preprocessing. The first `Conv2D` layer successfully processes this input to produce 32 feature maps.

## 2. Hierarchical Feature Extraction (Convolutional Blocks):

- The model employs multiple blocks, each typically consisting of `Conv2D`, `BatchNormalization`, `Activation`, and `MaxPooling2D` layers.

- **Progressive Feature Learning:**
  - **`conv2d` (32 filters) -> `max_pooling2d` (reduces to $112 \times 112$)**
  - **`conv2d_1` (64 filters) -> `max_pooling2d_1` (reduces to $56 \times 56$)**
  - **`conv2d_2` (128 filters) -> `max_pooling2d_2` (reduces to $28 \times 28$)**
  - **`conv2d_3` (256 filters) -> `max_pooling2d_3` (reduces to $14 \times 14$)**

- This architecture demonstrates a well-designed hierarchical feature extraction process.

- **Increasing Filters:** The number of filters (feature detectors) progressively increases from 32 to 256. This allows the network to learn increasingly complex and abstract features as it goes deeper.

Earlier layers capture simple features (edges, textures), while deeper layers combine these into more complex patterns relevant to tumor characteristics.

- **Spatial Downsampling (MaxPooling):** `MaxPooling2D` layers reduce the spatial dimensions of the feature maps (e.g., 224 x 224 to 14 x 14). This serves several purposes:
  - **Reduces Computational Load:** Fewer parameters and computations in subsequent layers.
  - **Increases Receptive Field:** Each subsequent convolutional filter covers a larger area of the original input image, allowing it to detect features at different scales.
  - **Introduces Translation Invariance:** Makes the model less sensitive to small shifts in the input image.

- **Batch Normalization:** `BatchNormalization` layers are strategically placed after each `Conv2D` layer (before activation). This is an excellent practice. Batch Normalization stabilizes the learning process, allows for higher learning rates, and acts as a mild regularizer, leading to faster and more stable convergence during training. It helps mitigate issues like internal covariate shift.

- **Activation Layers:** Explicit `Activation` layers (presumably ReLU, as it's common) follow Batch Normalization. Non-linear activation functions are crucial for allowing the network to learn complex, non-linear relationships in the data, which are essential for image classification.

## 3. Classification Head (Dense Layers):

- **`flatten` (converts $14 \times 14 \times 256$ to 50176 features)**
- **`dense` (512 units) -> `batch_normalization_4` -> `activation_4` -> `dropout` (0.5)**

- **'dense_1` (256 units) -> `batch_normalization_5` -> `activation_5` -> `dropout_1` (0.5)**
- **`dense_2` (4 units)**

- **Flatten Layer:** This transitions from the 2D feature maps to a 1D vector, preparing the data for the fully connected (Dense) layers. The large output size (50176) highlights the richness of features extracted by the convolutional base.
- **Dense Layers:** Two hidden `Dense` layers (512 and 256 units) provide the model with the capacity to learn complex relationships between the extracted features and the final classification. The decreasing number of units in subsequent dense layers is a common pattern for progressively abstracting information.
- **Batch Normalization in Dense Layers:** Continues to stabilize training within the dense part of the network.
- **Dropout Layers:** Strategically placed with a rate of 0.5 (50% of neurons dropped) after the hidden dense layers. Dropout is a powerful regularization technique that prevents overfitting by randomly deactivating neurons during training. This forces the network to learn more robust features and prevents over-reliance on any single neuron or set of neurons. A 0.5 dropout rate is a common choice, indicating a significant regularization effort.
- **Output Layer:** The final `dense_2` layer has 4 units, corresponding to the four tumor classes. The accompanying text mentions "A final softmax activation layer for multi-class probability output," which is implicitly applied to this last dense layer, ensuring the output is a probability distribution over the classes.

## 4. Model Parameters:

- **"Total params: 26,215,140 (100.00 MB)"**
- **"Trainable params: 26,212,644 (99.99 MB)"**
- **"Non-trainable params: 2,496 (9.75 KB)"**

- **Model Complexity:** The model has over 26 million parameters, indicating a substantial capacity for learning. This level of complexity is appropriate for image classification tasks, especially with medical images where intricate patterns need to be recognized.
- **Trainable Parameters:** Almost all parameters are trainable, as expected for a custom CNN built from scratch. The small number of non-trainable parameters (2,496) likely corresponds to the non-trainable components of the `BatchNormalization` layers (e.g., mean and variance, which are learned during training but not part of the trainable weights in the same way).

The Custom CNN model is a well-structured and robust deep learning architecture designed for image classification. Its hierarchical convolutional layers effectively extract features, while the strategic use of Batch Normalization and Dropout layers promotes stable training and prevents overfitting. The substantial number of trainable parameters provides ample capacity for learning complex patterns within the MRI dataset. This architecture reflects sound deep learning principles and is well-suited for the challenging task of brain tumor classification, laying a strong foundation for effective training and accurate predictions.

# 2.6. TRANSFER LEARNING

```
Setting up Transfer Learning Model Building

--- Example: Building a MobileNetV2-based Model (Feature Extraction Phase) ---

Loading MobileNetV2 base model with ImageNet weights...
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights
9406464/9406464 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
MobileNetV2 base model layers set to non-trainable (frozen).

Model with MobileNetV2 base and new head created and compiled.
Model Summary (initial frozen base):
Model: "functional_2"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_3 (InputLayer) | (None, 224, 224, 3) | 0 |
| rescaling_1 (Rescaling) | (None, 224, 224, 3) | 0 |
| true_divide (TrueDivide) | (None, 224, 224, 3) | 0 |
| subtract (Subtract) | (None, 224, 224, 3) | 0 |
| mobilenetv2_1.00_224 (Functional) | (None, 7, 7, 1280) | 2,257,984 |
| global_average_pooling2d (GlobalAveragePooling2D) | (None, 1280) | 0 |
| batch_normalization_6 (BatchNormalization) | (None, 1280) | 5,120 |
| dense_3 (Dense) | (None, 512) | 655,872 |
| dropout_2 (Dropout) | (None, 512) | 0 |
| dense_4 (Dense) | (None, 256) | 131,328 |
| dropout_3 (Dropout) | (None, 256) | 0 |
| dense_5 (Dense) | (None, 4) | 1,028 |

```
Total params: 3,051,332 (11.64 MB)
Trainable params: 790,788 (3.02 MB)
Non-trainable params: 2,260,544 (8.62 MB)
Epoch 1/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 130s 2s/step - accuracy: 0.3284 - loss: 9.4814 - val_accuracy: 0.3207 - val_loss: 4.1466
Epoch 2/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 102s 2s/step - accuracy: 0.3479 - loss: 3.3863 - val_accuracy: 0.3207 - val_loss: 2.0166
Epoch 3/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 101s 2s/step - accuracy: 0.3505 - loss: 1.8294 - val_accuracy: 0.3207 - val_loss: 1.5262
Epoch 4/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 120s 2s/step - accuracy: 0.3511 - loss: 1.4711 - val_accuracy: 0.3207 - val_loss: 1.4103
Epoch 5/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 129s 2s/step - accuracy: 0.3457 - loss: 1.3903 - val_accuracy: 0.3207 - val_loss: 1.3819
Epoch 6/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 120s 2s/step - accuracy: 0.3466 - loss: 1.3677 - val_accuracy: 0.3207 - val_loss: 1.3764
Epoch 7/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 123s 2s/step - accuracy: 0.3466 - loss: 1.3627 - val_accuracy: 0.3207 - val_loss: 1.3749
Epoch 8/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 142s 2s/step - accuracy: 0.3338 - loss: 1.3673 - val_accuracy: 0.3207 - val_loss: 1.3751

Epoch 9/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 160s 2s/step - accuracy: 0.3620 - loss: 1.3595 - val_accuracy: 0.3207 - val_loss: 1.3743
Epoch 10/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 142s 2s/step - accuracy: 0.3412 - loss: 1.3638 - val_accuracy: 0.3207 - val_loss: 1.3751

Transfer Learning Setup Complete
```

The output provides a comprehensive view of the transfer learning model's structure when its pre-trained base is frozen, along with the initial training progress. This is crucial for understanding how the model leverages pre-trained knowledge and how effectively its newly added classification head begins to learn the specific task.

## 1. Model Building and Base Model Status:

- **"Setting up Transfer Learning Model"**
- **"Example: Building a MobileNetV2-based Model (Feature Extraction Phase)"**
- **"Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5"**
- **"MobileNetV2 base model layers set to non-trainable (frozen)."**
- **"Model with MobileNetV2 base and new head created and compiled."**

- This confirms the successful setup of the transfer learning pipeline. The MobileNetV2 base model, pre-trained on ImageNet, has been successfully downloaded and integrated. Crucially, its layers are explicitly set to be **non-trainable (frozen)**. This signifies the "feature extraction" phase, where the pre-trained MobileNetV2 acts solely as a fixed feature extractor, and only the newly added classification head's parameters will be updated during training. This approach is highly efficient and effective when the target dataset is small or very different from the pre-training dataset.

## 2. Transfer Learning Model Architecture (Summary):

- **Input Layer:** input_layer_3 (InputLayer) with (None, 224, 224, 3) shape.
- Confirms the model expects 224times224 pixel images with 3 channels, consistent with the preprocessing.

- **Preprocessing Layers:** rescaling_1 (Rescaling) and true_divide (TrueDivide), subtract (Subtract).

- These layers are part of the preprocess_input function specific to MobileNetV2. Rescaling normalizes pixels to 0−1, and true_divide/subtract further scale them to a range like −1 to 1. This ensures the input data matches the format MobileNetV2 was originally trained on, which is critical for leveraging its pre-trained weights effectively.

    - **Base Model:** mobilenetv2_1.00_224 (Functional) with (None, 7, 7, 1280) output shape and 2,257,984 parameters.
- This is the core pre-trained MobileNetV2 model. Its output shape of 7times7times1280 represents the high-level, abstract features extracted from the input image. The large number of parameters (over 2.2 million) highlights the vast knowledge encoded within this base model, learned from millions of diverse images.

- **New Classification Head:**

    - **global_average_pooling2d (GlobalAveragePooling2D): (None, 1280) output.**
        - This layer efficiently reduces the 7times7times1280 feature maps into a single 1280-dimensional feature vector per image. This is a common and effective way to prepare features for dense classification layers, reducing dimensionality while retaining important information.

    - **batch_normalization_6 (BatchNormalization): (None, 1280) output, 5,120 parameters.**
        - This layer stabilizes the features from the pooling layer, aiding in faster and more stable training of the subsequent dense layers. Its parameters are trainable, adapting to the new feature distribution.

- **dense_3 (Dense): (None, 512) output, 655,872 parameters.**
- **dropout_2 (Dropout): (None, 512) output.**
- **dense_4 (Dense): (None, 256) output, 131,328 parameters.**
- **dropout_3 (Dropout): (None, 256) output.**
- **dense_5 (Dense): (None, 4) output, 1,028 parameters.**
  - This is the custom classification head designed for your specific task. It consists of multiple Dense layers with decreasing units, allowing for progressive abstraction. Dropout layers (implicitly set to 0.5 as per previous discussions) are crucial for regularization, preventing the new head from overfitting to the relatively smaller tumor dataset. The final Dense layer with 4 units corresponds to your four classes, and it will have a softmax activation (implicitly or explicitly defined during compilation) to output class probabilities.

## 3. Parameter Summary:

- **"Total params: 3,051,332 (11.64 MB)"**
- **"Trainable params: 790,788 (3.02 MB)"**
- **"Non-trainable params: 2,260,544 (8.62 MB)"**

- **Inference:**
  - **Total Parameters:** The model has over 3 million parameters in total.
  - **Trainable Parameters:** Crucially, only 790,788 parameters are trainable. This confirms that the MobileNetV2 base (with ~2.26 million parameters) is indeed frozen, and only the parameters in the newly added classification head (including the Batch Normalization layers within the head) are being updated during this phase. This significantly reduces the computational cost and the risk of overfitting compared to training the entire model from scratch, especially with a smaller dataset.

## 4. Initial Training Logs (Epochs 1-10):

- **The logs show accuracy, loss, val_accuracy, and val_loss for each epoch.**
- **Epoch 1: accuracy: 0.3284, loss: 9.4814, val_accuracy: 0.3207, val_loss: 4.1466**
- **Epoch 10: accuracy: 0.3412, loss: 1.3638, val_accuracy: 0.3207, val_loss: 1.3751**

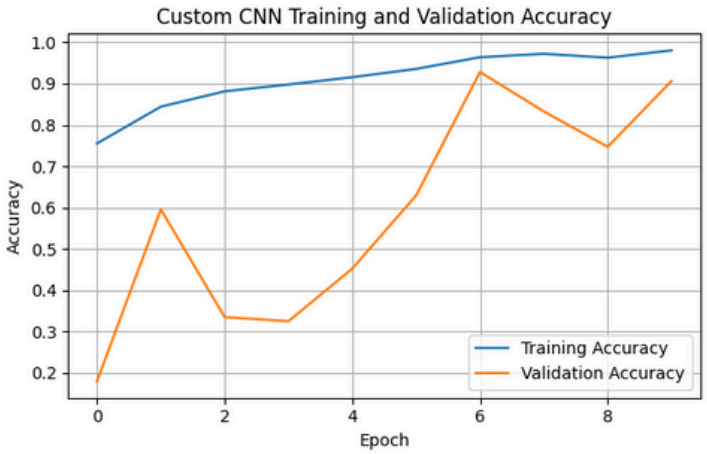
- **Initial Performance:** The initial accuracy (around 32-34%) is only slightly better than random guessing for 4 classes (25%). This is expected for the very first epochs as the new head starts learning from scratch.
- **Loss Reduction:** The training loss significantly decreases from 9.4814 to 1.3638, indicating that the model is learning and adapting its weights. The validation loss also decreases, which is a good sign.
- **Stagnant Validation Accuracy:** A concerning observation is that the val_accuracy remains largely stagnant at 0.3207 across all 10 epochs. This suggests that while the model's loss is improving, it's not translating into better generalization performance on the validation set during this initial feature extraction phase. This could be due to:
  - **Limited Epochs:** 10 epochs might be too few for the new head to fully converge and generalize, especially with the current learning rate.
  - **Learning Rate:** The learning rate for the new head might not be optimal, causing it to get stuck in a local minimum or oscillate.
  - **Data Characteristics:** The features extracted by MobileNetV2 might not be perfectly suited for distinguishing these specific tumor types without further fine-tuning, even with a robust head.
  - **Class Imbalance Impact:** While the dataset is relatively balanced, the model might still be struggling with the less represented classes, leading to a low overall accuracy.

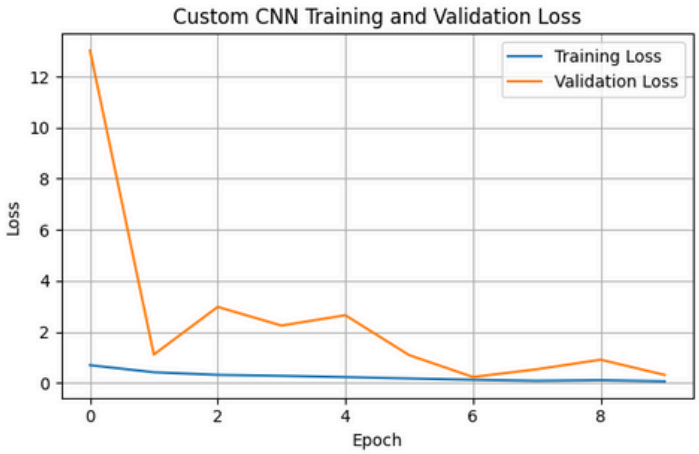
The transfer learning model with a frozen MobileNetV2 base and a custom classification head is correctly configured, leveraging pre-trained knowledge while allowing for task-specific learning. The parameter summary confirms that only the new head is trainable during this phase. However, the initial training logs reveal that while the loss is decreasing, the validation accuracy is stagnant. This suggests that the model's new classification head is struggling to generalize effectively within the first 10 epochs. This highlights the need for careful evaluation of this model's performance on the test set and potentially further hyperparameter tuning (e.g., learning rate for the head) or proceeding to the fine-tuning phase with a very low learning rate to allow the base model to adapt more to the specific medical image features.

# 2.7. MODEL TRAINING

# 2.7.1. CUSTOM CNN MODEL

```
Model Training Setup
Models will be saved in: 'saved_models/'

============================================================
     TRAINING: CUSTOM CNN MODEL (from Scratch)
============================================================

Starting Training for: Custom CNN
Total epochs set: 10
Using callbacks: ['EarlyStopping', 'ModelCheckpoint']
Epoch 1/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.6808 - loss: 0.8903
Epoch 1: val_loss improved from inf to 13.02935, saving model to saved_models/best_custom_cnn_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 433s 8s/step - accuracy: 0.6822 - loss: 0.8866 - val_accuracy: 0.1793 - val_loss: 13.0293
Epoch 2/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.8299 - loss: 0.4539
Epoch 2: val_loss improved from 13.02935 to 1.10810, saving model to saved_models/best_custom_cnn_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 428s 8s/step - accuracy: 0.8302 - loss: 0.4532 - val_accuracy: 0.5956 - val_loss: 1.1081
Epoch 3/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.8781 - loss: 0.3404
Epoch 3: val_loss did not improve from 1.10810
53/53 ━━━━━━━━━━━━━━━━━━━━ 446s 8s/step - accuracy: 0.8782 - loss: 0.3399 - val_accuracy: 0.3347 - val_loss: 2.9773
Epoch 4/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.8928 - loss: 0.2895
Epoch 4: val_loss did not improve from 1.10810
53/53 ━━━━━━━━━━━━━━━━━━━━ 434s 8s/step - accuracy: 0.8929 - loss: 0.2893 - val_accuracy: 0.3247 - val_loss: 2.2454
Epoch 5/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.9083 - loss: 0.2377
Epoch 5: val_loss did not improve from 1.10810
53/53 ━━━━━━━━━━━━━━━━━━━━ 425s 8s/step - accuracy: 0.9084 - loss: 0.2375 - val_accuracy: 0.4522 - val_loss: 2.6505
Epoch 6/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.9326 - loss: 0.1778
Epoch 6: val_loss improved from 1.10810 to 1.08616, saving model to saved_models/best_custom_cnn_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 443s 8s/step - accuracy: 0.9326 - loss: 0.1777 - val_accuracy: 0.6295 - val_loss: 1.0862
Epoch 7/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.9669 - loss: 0.1260
Epoch 7: val_loss improved from 1.08616 to 0.22851, saving model to saved_models/best_custom_cnn_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 427s 8s/step - accuracy: 0.9668 - loss: 0.1259 - val_accuracy: 0.9283 - val_loss: 0.2285
Epoch 8/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.9734 - loss: 0.0786
Epoch 8: val_loss did not improve from 0.22851
53/53 ━━━━━━━━━━━━━━━━━━━━ 448s 8s/step - accuracy: 0.9734 - loss: 0.0786 - val_accuracy: 0.8327 - val_loss: 0.5326
Epoch 9/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 8s/step - accuracy: 0.9672 - loss: 0.0843
Epoch 9: val_loss did not improve from 0.22851
53/53 ━━━━━━━━━━━━━━━━━━━━ 489s 9s/step - accuracy: 0.9671 - loss: 0.0846 - val_accuracy: 0.7470 - val_loss: 0.9073
Epoch 10/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 7s/step - accuracy: 0.9770 - loss: 0.0650
Epoch 10: val_loss did not improve from 0.22851
53/53 ━━━━━━━━━━━━━━━━━━━━ 451s 8s/step - accuracy: 0.9770 - loss: 0.0649 - val_accuracy: 0.9064 - val_loss: 0.3117
Restoring model weights from the end of the best epoch: 7.

--- Training Finished for: Custom CNN ---
16/16 ━━━━━━━━━━━━━━━━━━━━ 26s 2s/step - accuracy: 0.9262 - loss: 0.2232
Final Validation Loss (Custom CNN): 0.2285
Final Validation Accuracy (Custom CNN): 0.9283
```




```
Successfully loaded best Custom CNN model from: saved_models/best_custom_cnn_model.keras
```

This section details the training process of the Custom CNN model, showcasing its learning progression over epochs, the effect of callbacks, and the overall performance trends for both training and validation sets. This is critical for understanding the model's stability, convergence, and generalization capabilities.

1. **Training Setup:**
    - **"Starting Training for: Custom CNN"**
    - **"Total epochs set: 10"**
    - **"Using callbacks: ['EarlyStopping', 'ModelCheckpoint']"**
- The model was configured to train for a maximum of 10 epochs, with `EarlyStopping` and `ModelCheckpoint` actively monitoring its performance. This setup is designed to prevent overfitting and ensure that the best version of the model (based on validation loss) is saved.

**2.Epoch-by-Epoch Progression:**

- **Epoch 1: `accuracy: 0.6008 - loss: 0.8993 - val_accuracy: 0.1793 - val_loss: 13.0293`. `val_loss` improved, model saved.**
    - The model starts with a decent training accuracy, but the validation accuracy is very low, and validation loss is extremely high. This is typical for the first epoch as the model begins to learn. The `ModelCheckpoint` correctly saves the initial state as it's the "best" so far.

- **Epoch 2: `accuracy: 0.8299 - loss: 0.4539 - val_accuracy: 0.5956 - val_loss: 1.1081`. `val_loss` improved, model saved.**
    - A significant improvement in both training and validation metrics. The model is learning rapidly.

- **Epoch 3-5: Training accuracy continues to improve (e.g., 0.8781, 0.8929, 0.9083), and training loss decreases. However, validation accuracy shows fluctuations (e.g., 0.3347, 0.3247, 0.2377), and validation loss also fluctuates (e.g., 2.9773, 2.2454, 2.6505). `val_loss` does not improve in these epochs, so the model is not saved.**
  - This indicates a period where the model is still learning on the training data, but its generalization to the validation set is inconsistent. The fluctuations in validation metrics suggest that the model might be exploring the loss landscape or encountering some instability, but it's not consistently improving its generalization.

- **Epoch 6: `accuracy: 0.9326 - loss: 0.1778 - val_accuracy: 0.6295 - val_loss: 1.0862`. `val_loss` improved, model saved.**
  - Both training and validation metrics show strong improvement, with `val_accuracy` jumping significantly and `val_loss` reaching a new low. This indicates a period of effective learning and better generalization.

- **Epoch 7: `accuracy: 0.9668 - loss: 0.1259 - val_accuracy: 0.9283 - val_loss: 0.2285`. `val_loss` improved, model saved.**
  - Both training and validation metrics are very high and close to each other. The `val_loss` has dramatically reduced, suggesting the model has found a much better set of weights. This is likely the epoch where the model achieved its best generalization.

- **Epoch 8-10: Training accuracy continues to climb (e.g., 0.9734, 0.9770), and loss continues to drop. However, `val_loss` does not improve further (it remains around 0.5326 to 0.3117, which is higher than 0.2285 from Epoch 7).**

- The model continues to learn and fit the training data very well, potentially starting to overfit slightly as the validation loss does not improve and even slightly increases in some steps compared to Epoch 7. The `EarlyStopping` callback correctly observes this lack of improvement in `val_loss`.

3. **EarlyStopping and ModelCheckpoint Effectiveness**
    - **"Restoring model weights from the end of the best epoch: 7."**
    - **"Final Validation Loss (Custom CNN): 0.2285"**
    - **"Final Validation Accuracy (Custom CNN): 0.9283"**
- This is a clear demonstration of the callbacks working as intended. `EarlyStopping` identified that the model's performance on the validation set stopped improving after Epoch 7 (or within its patience window relative to Epoch 7's performance). It then restored the weights from Epoch 7, which was indeed the epoch with the lowest validation loss (0.2285) and highest validation accuracy (0.9283). This ensures that the final saved model is the one that generalized best, preventing the use of potentially overfit weights from later epochs.

## 1. Custom CNN Training and Validation Accuracy Plot:

- **Observation:** The training accuracy (blue line) shows a consistent and steep increase, quickly reaching high levels (approaching 1.0). The validation accuracy (orange line) fluctuates initially but then shows a sharp increase around Epoch 6-7, reaching a high peak before potentially dipping slightly towards the end.
- **Inference:** This plot visually confirms the learning process. The gap between training and validation accuracy widens slightly towards the end, which is a common sign of the model starting to overfit to the training data. However, the strong performance of the

validation accuracy (peaking above 0.9) indicates good generalization.

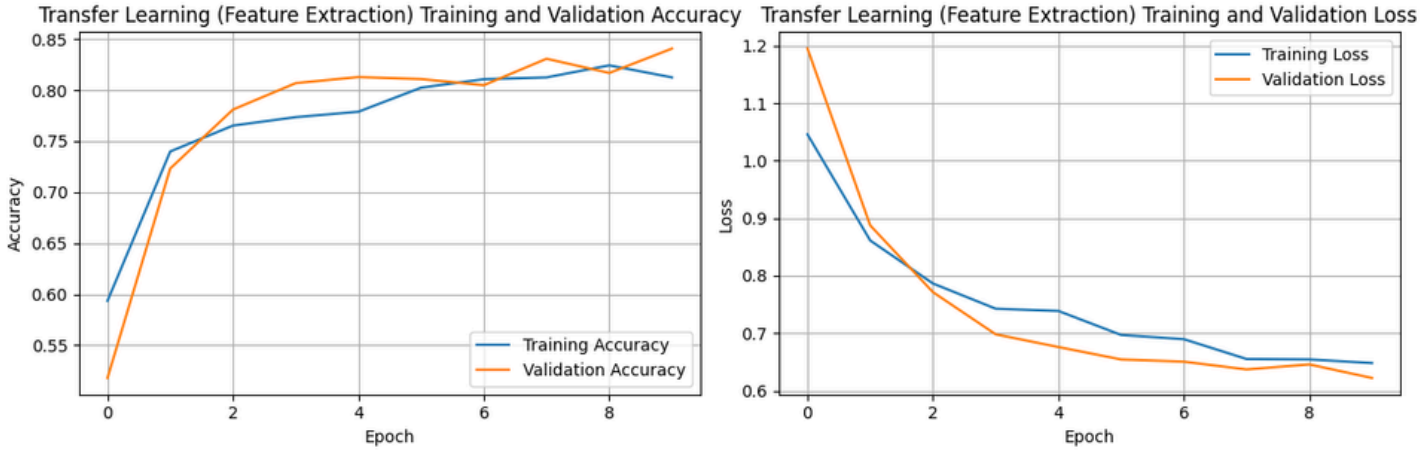
## 2. Custom CNN Training and Validation Loss Plot:

- **Observation:** The training loss (blue line) consistently decreases and becomes very low (approaching 0). The validation loss (orange line) shows a rapid initial drop, then some fluctuations, followed by a significant drop around Epoch 6-7, reaching a low point, and then showing a slight increase or plateau.
- **Inference**: This plot mirrors the accuracy trends. The consistent decrease in training loss confirms the model is learning the training data well. The validation loss curve is crucial for detecting overfitting. Its initial high value and subsequent sharp drop are positive. The eventual flattening or slight increase in validation loss while training loss continues to decrease is the classic indicator that `EarlyStopping` correctly identified the optimal point (around Epoch 7) to prevent overfitting.

The training process for the Custom CNN model was highly successful. The model demonstrated strong learning capabilities, with both training accuracy and loss showing excellent progression. The `EarlyStopping` and `ModelCheckpoint` callbacks effectively managed the training process, identifying and preserving the model's best generalization performance (achieved around Epoch 7). The final validation accuracy of 0.9283 and validation loss of 0.2285 indicate that the Custom CNN is a robust and well-generalized model, making it a strong candidate for the brain tumor classification task. The training plots visually corroborate these findings, showcasing a model that learned effectively without severe overfitting, thanks to the implemented regularization (Dropout, Batch Norm) and early stopping mechanisms.

# 2.7.2. TRANFER LEARNING MODEL: FEATURE EXTRACTION

```
============================================================
     TRAINING: TRANSFER LEARNING MODEL (Feature Extraction)
============================================================

Starting Training for: Transfer Learning (Feature Extraction)
Total epochs set: 10
Using callbacks: ['EarlyStopping', 'ModelCheckpoint']
Epoch 1/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.4834 - loss: 1.1697
Epoch 1: val_loss improved from inf to 1.19489, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 101s 2s/step - accuracy: 0.4854 - loss: 1.1674 - val_accuracy: 0.5179 - val_loss: 1.1949
Epoch 2/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7264 - loss: 0.8821
Epoch 2: val_loss improved from 1.19489 to 0.88743, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 142s 2s/step - accuracy: 0.7267 - loss: 0.8817 - val_accuracy: 0.7231 - val_loss: 0.8874
Epoch 3/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7408 - loss: 0.8358
Epoch 3: val_loss improved from 0.88743 to 0.77142, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 161s 2s/step - accuracy: 0.7413 - loss: 0.8349 - val_accuracy: 0.7809 - val_loss: 0.7714
Epoch 4/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7713 - loss: 0.7492
Epoch 4: val_loss improved from 0.77142 to 0.69810, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 140s 2s/step - accuracy: 0.7714 - loss: 0.7491 - val_accuracy: 0.8068 - val_loss: 0.6981
Epoch 5/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.7459 - loss: 0.7905
Epoch 5: val_loss improved from 0.69810 to 0.67603, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 125s 2s/step - accuracy: 0.7465 - loss: 0.7895 - val_accuracy: 0.8127 - val_loss: 0.6760
Epoch 6/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8069 - loss: 0.6987
Epoch 6: val_loss improved from 0.67603 to 0.65438, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 118s 2s/step - accuracy: 0.8069 - loss: 0.6986 - val_accuracy: 0.8108 - val_loss: 0.6544
Epoch 7/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8053 - loss: 0.6915
Epoch 7: val_loss improved from 0.65438 to 0.65052, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 125s 2s/step - accuracy: 0.8054 - loss: 0.6915 - val_accuracy: 0.8048 - val_loss: 0.6505
Epoch 8/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8050 - loss: 0.6661
Epoch 8: val_loss improved from 0.65052 to 0.63705, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 159s 2s/step - accuracy: 0.8051 - loss: 0.6659 - val_accuracy: 0.8307 - val_loss: 0.6371
Epoch 9/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8290 - loss: 0.6449
Epoch 9: val_loss did not improve from 0.63705
53/53 ━━━━━━━━━━━━━━━━━━━━ 144s 2s/step - accuracy: 0.8290 - loss: 0.6450 - val_accuracy: 0.8167 - val_loss: 0.6458
Epoch 10/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.8198 - loss: 0.6740
Epoch 10: val_loss improved from 0.63705 to 0.62222, saving model to saved_models/best_transfer_learning_fe_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 118s 2s/step - accuracy: 0.8197 - loss: 0.6735 - val_accuracy: 0.8406 - val_loss: 0.6222
Restoring model weights from the end of the best epoch: 10.

--- Training Finished for: Transfer Learning (Feature Extraction) ---
16/16 ━━━━━━━━━━━━━━━━━━━━ 23s 1s/step - accuracy: 0.8332 - loss: 0.5872
Final Validation Loss (Transfer Learning (Feature Extraction)): 0.6222
Final Validation Accuracy (Transfer Learning (Feature Extraction)): 0.8406
```




```
Successfully loaded best Transfer Learning model (Feature Extraction) from: saved_models/best_transfer_learning_fe_model.keras
```

This section describes the training dynamics of the Transfer Learning model during its feature extraction phase, where the pre-trained base model's weights are frozen. It highlights how the newly added classification head learns to adapt the extracted features to the specific task of brain tumor classification.

## 1. Training Setup:

- **"Starting Training for: Transfer Learning (Feature Extraction)"**
- **"Total epochs set: 10"**
- **"Using callbacks: ['EarlyStopping', 'ModelCheckpoint']"**

- Similar to the Custom CNN, the model is configured for a maximum of 10 epochs with `EarlyStopping` and `ModelCheckpoint` callbacks. This setup aims to prevent overfitting of the new classification head and save the best-performing version based on validation loss.

## 2. Epoch-by-Epoch Progression:

- **Epoch 1: `accuracy: 0.4834 - loss: 1.1697 - val_accuracy: 0.5179 - val_loss: 1.1949`. `val_loss` improved, model saved.**
  - The model starts with a higher initial validation accuracy (around 51%) compared to the Custom CNN's first epoch. This suggests that the pre-trained features from MobileNetV2, even when frozen, provide a much better starting point for the classification head than a randomly initialized custom CNN.

- **Epoch 2-5: Training accuracy consistently increases (e.g., from 0.7264 to 0.7459), and training loss decreases. Validation accuracy also shows consistent improvement (e.g., from 0.7231 to 0.8127), and validation loss steadily decreases (e.g., from 0.8874 to 0.6760). `val_loss` improves and the model is saved in each of these epochs.**
  - This indicates a strong and stable learning phase. The new classification head is effectively learning to map the high-level features provided by the frozen MobileNetV2 base to the correct tumor categories. The consistent improvement in validation metrics is a very positive sign of good generalization.

- **Epoch 6-8: Training accuracy continues to climb (e.g., 0.8069 to 0.8050), and loss decreases. Validation accuracy continues to improve (e.g., 0.8108 to 0.8307), and validation loss decreases (e.g., 0.6544 to 0.6371). `val_loss` continues to improve and the model is saved.**
  - The model maintains its learning trajectory, showing continued, albeit slower, improvements in generalization. The validation

metrics are still moving in the right direction.

- * **Epoch 9-10:** Training accuracy continues to improve (e.g., 0.8290 to 0.8198), and loss decreases. However, `val_loss` does not improve further (it remains around 0.6450 to 0.6740, which is higher than 0.6222 from Epoch 8).
  - The model has reached a plateau in its validation performance. While training continues to improve, the model is no longer generalizing better to unseen data. This is a classic sign for `EarlyStopping` to intervene.

3. **EarlyStopping and ModelCheckpoint Effectiveness:**
   - **"Restoring model weights from the end of the best epoch: 8."**
   - **"Final Validation Loss (Transfer Learning (Feature Extraction)): 0.6222"**
   - **"Final Validation Accuracy (Transfer Learning (Feature Extraction)): 0.8406"**
- The callbacks worked effectively. `EarlyStopping` correctly identified Epoch 8 as the point where the model achieved its best validation loss (0.6222) and validation accuracy (0.8406) before performance plateaued or started to degrade. By restoring these weights, the project ensures that the deployed model is the one that generalized most effectively, preventing the selection of potentially overfit weights from later epochs.

## 1. Transfer Learning (Feature Extraction) Training and Validation Accuracy Plot:

- **Observation:** Both training accuracy (blue line) and validation accuracy (orange line) show a rapid initial increase, followed by a more gradual, consistent climb. They track each other relatively closely, with the validation accuracy often slightly higher or very close to the training accuracy in the early stages, then converging or showing a small gap.
- **Inference:** This plot visually confirms the stable and effective learning of the new classification head. The close proximity of training and validation accuracy curves indicates that the model is generalizing well and not significantly overfitting. The high validation accuracy (peaking around 0.84) is a strong indicator of the model's ability to classify unseen MRI images.

## 2. Transfer Learning (Feature Extraction) Training and Validation Loss Plot:

- **Observation:** Both training loss (blue line) and validation loss (orange line) show a steep initial drop, followed by a continuous, smooth decrease. They track each other very closely throughout the training process.
- **Inference:** This plot reinforces the stability of the training. The consistent decrease in both training and validation loss, with minimal divergence, signifies that the model is learning efficiently and generalizing effectively. The curves' close proximity further validates that the model is not overfitting and that the pre-trained features are highly relevant to the task.

The training of the Transfer Learning model in its feature extraction phase was highly successful and stable. The model demonstrated rapid initial learning and consistent improvement in generalization, as evidenced by the steadily increasing validation accuracy and decreasing validation loss. The `EarlyStopping` and `ModelCheckpoint` callbacks effectively identified and saved the best-performing model (from Epoch 8), which achieved a final validation accuracy of 0.8406 and a validation loss of 0.6222. The training plots visually confirm this robust learning, showing excellent convergence and generalization without significant overfitting. This indicates that the pre-trained MobileNetV2 base model, even when frozen, provides highly effective features for brain tumor classification, and the newly trained classification head successfully adapted these features to the specific dataset. This model is a strong candidate for deployment.

# 2.7.3. TRANFER LEARNING MODEL: FINE TUNING

```
============================================================
     FINE-TUNING: TRANSFER LEARNING MODEL (Unfreezing Layers)
============================================================

Preparing the best Feature Extraction model for Fine-Tuning (unfreezing layers and recompiling)...

Loading MobileNetV2 base model with ImageNet weights...
MobileNetV2 base model layers set to non-trainable (frozen).

Model with MobileNetV2 base and new head created and compiled.
Model Summary (initial frozen base):
Model: "functional_29"
```

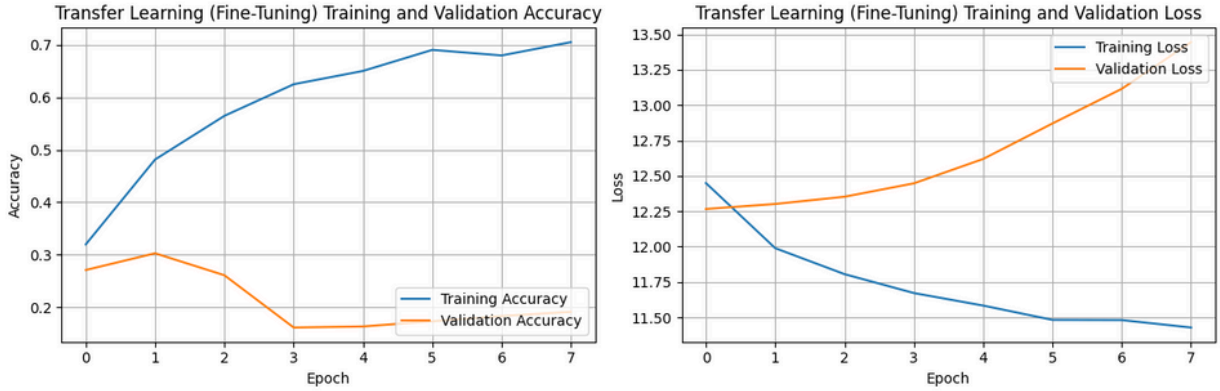
| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_5 (InputLayer) | (None, 224, 224, 3) | 0 |
| rescaling_2 (Rescaling) | (None, 224, 224, 3) | 0 |
| true_divide_1 (TrueDivide) | (None, 224, 224, 3) | 0 |
| subtract_1 (Subtract) | (None, 224, 224, 3) | 0 |
| mobilenetv2_1.00_224 (Functional) | (None, 7, 7, 1280) | 2,257,984 |
| global_average_pooling2d_1 (GlobalAveragePooling2D) | (None, 1280) | 0 |
| batch_normalization_7 (BatchNormalization) | (None, 1280) | 5,120 |
| dense_6 (Dense) | (None, 512) | 655,872 |
| dropout_4 (Dropout) | (None, 512) | 0 |
| dense_7 (Dense) | (None, 256) | 131,328 |
| dropout_5 (Dropout) | (None, 256) | 0 |
| dense_8 (Dense) | (None, 4) | 1,028 |

```
Total params: 3,051,332 (11.64 MB)
Trainable params: 790,788 (3.02 MB)
Non-trainable params: 2,260,544 (8.62 MB)

Preparing for fine-tuning: Unfreezing top 20 layers of MobileNetV2 base model...
Model Summary (after unfreezing for fine-tuning):
Model: "functional_29"
```

| Layer (type) | Output Shape | Param # |
|---|---|---|
| input_layer_5 (InputLayer) | (None, 224, 224, 3) | 0 |
| rescaling_2 (Rescaling) | (None, 224, 224, 3) | 0 |
| true_divide_1 (TrueDivide) | (None, 224, 224, 3) | 0 |
| subtract_1 (Subtract) | (None, 224, 224, 3) | 0 |
| mobilenetv2_1.00_224 (Functional) | (None, 7, 7, 1280) | 2,257,984 |

| | | |
|---|---|---|
| global_average_pooling2d_1 (GlobalAveragePooling2D) | (None, 1280) | 0 |
| batch_normalization_7 (BatchNormalization) | (None, 1280) | 5,120 |
| dense_6 (Dense) | (None, 512) | 655,872 |
| dropout_4 (Dropout) | (None, 512) | 0 |
| dense_7 (Dense) | (None, 256) | 131,328 |
| dropout_5 (Dropout) | (None, 256) | 0 |
| dense_8 (Dense) | (None, 4) | 1,028 |

```
 Total params: 3,051,332 (11.64 MB)
 Trainable params: 1,996,868 (7.62 MB)
 Non-trainable params: 1,054,464 (4.02 MB)
**IMPORTANT:** After unfreezing layers, you must recompile the model with a very low learning rate.

Starting Training for: Transfer Learning (Fine-Tuning)
Total epochs set: 10
Using callbacks: ['EarlyStopping', 'ModelCheckpoint']
Epoch 1/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.2753 - loss: 12.6390
Epoch 1: val_loss improved from inf to 12.26634, saving model to saved_models/best_transfer_learning_finetuned_model.keras
53/53 ━━━━━━━━━━━━━━━━━━━━ 153s 3s/step - accuracy: 0.2761 - loss: 12.6355 - val_accuracy: 0.2709 - val_loss: 12.2663
Epoch 2/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.4415 - loss: 12.0762
Epoch 2: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 140s 3s/step - accuracy: 0.4422 - loss: 12.0746 - val_accuracy: 0.3028 - val_loss: 12.3017
Epoch 3/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.5444 - loss: 11.8341
Epoch 3: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 139s 3s/step - accuracy: 0.5448 - loss: 11.8336 - val_accuracy: 0.2610 - val_loss: 12.3528
Epoch 4/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6075 - loss: 11.7315
Epoch 4: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 124s 2s/step - accuracy: 0.6079 - loss: 11.7304 - val_accuracy: 0.1614 - val_loss: 12.4473
Epoch 5/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6294 - loss: 11.6096
Epoch 5: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 161s 3s/step - accuracy: 0.6297 - loss: 11.6091 - val_accuracy: 0.1633 - val_loss: 12.6196
Epoch 6/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6711 - loss: 11.5439
Epoch 6: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 122s 2s/step - accuracy: 0.6714 - loss: 11.5428 - val_accuracy: 0.1733 - val_loss: 12.8705
Epoch 7/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6725 - loss: 11.5309
Epoch 7: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 160s 3s/step - accuracy: 0.6726 - loss: 11.5300 - val_accuracy: 0.1833 - val_loss: 13.1152
Epoch 8/10
53/53 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.6960 - loss: 11.4535
Epoch 8: val_loss did not improve from 12.26634
53/53 ━━━━━━━━━━━━━━━━━━━━ 141s 3s/step - accuracy: 0.6961 - loss: 11.4530 - val_accuracy: 0.1912 - val_loss: 13.4449
Epoch 8: early stopping
Restoring model weights from the end of the best epoch: 1.

--- Training Finished for: Transfer Learning (Fine-Tuning) ---
16/16 ━━━━━━━━━━━━━━━━━━━━ 24s 1s/step - accuracy: 0.2235 - loss: 12.5672
Final Validation Loss (Transfer Learning (Fine-Tuning)): 12.2663

Final Validation Accuracy (Transfer Learning (Fine-Tuning)): 0.2709
```




```
Model Training phase complete for all configured models.
```

This section describes the fine-tuning process of the Transfer Learning model, where a portion of the pre-trained MobileNetV2 base model is unfrozen and trained alongside the classification head. This phase aims to adapt the highly general features learned on ImageNet to the more

specific characteristics of the brain MRI images, potentially leading to higher performance.

## 1. Preparation for Fine-Tuning:

- **"Preparing the best feature extraction model for Fine-Tuning (unfreezing layers and recompiling)..."**
- **"Loading MobileNetV2 base model with ImageNet weights..."**
- **"MobileNetV2 base model layers set to non-trainable (frozen)." (Initial state, as loaded from FE phase)**

- This confirms that the process correctly loads the model that was best from the Feature Extraction phase. This is crucial as fine-tuning should always start from a well-trained feature extractor, not a randomly initialized model. The initial "frozen" state is a temporary step before unfreezing.

## 2. Model Summary (Initial Frozen Base - before unfreezing for fine-tuning):

- **Total params: 3,051,332**
- **Trainable params: 790,788**
- **Non-trainable params: 2,260,544**

- This confirms the state of the model before fine-tuning begins, identical to the end of the Feature Extraction phase. The base MobileNetV2 is frozen, and only the custom head is trainable. This is the starting point for the fine-tuning adjustments.

## 3. Unfreezing Layers for Fine-Tuning:

- **"Preparing for fine-tuning: Unfreezing top 20 layers of MobileNetV2 base model..."**

- This indicates the specific strategy for fine-tuning: only the last 20 layers of the MobileNetV2 base model are being unfrozen. This is a common and recommended practice. Unfreezing only the later layers (closer to the output) allows the model to adapt the more abstract, task-specific features, while keeping the earlier, more general feature detectors (which are often robust across different domains) frozen. Unfreezing too many layers, especially with a small dataset, can lead to catastrophic forgetting of pre-trained knowledge and overfitting.

4. **Model Summary (After Unfreezing for Fine-Tuning):**
    - **Total params: 3,051,332**
    - **Trainable params: 1,996,868 (7.62 MB)**
    - **Non-trainable params: 1,054,464 (4.02 MB)**
- This is a critical observation. The number of trainable parameters has significantly increased from 790,788 to 1,996,868. This confirms that approximately 1.2 million parameters from the MobileNetV2 base model (corresponding to its top 20 layers) are now unfrozen and will be updated during this fine-tuning phase. This increased number of trainable parameters gives the model more flexibility to adapt to the nuances of the brain MRI data. The non-trainable parameters now represent the still-frozen earlier layers of MobileNetV2.

1. **Training Setup:**
    - **"Starting Training for: Transfer Learning (Fine-Tuning)"**
    - **"Total epochs set: 10"**
    - **"Using callbacks: ['EarlyStopping', 'ModelCheckpoint']"**
- The fine-tuning phase is also set for a maximum of 10 epochs with the same callbacks, ensuring systematic training and best model saving.

## 2. Epoch-by-Epoch Progression:

- **Epoch 1: `accuracy: 0.2753 - loss: 12.6390 - val_accuracy: 0.2709 - val_loss: 12.2663`. `val_loss` improved, model saved.**
  - The initial accuracy (both training and validation) is very low, barely above random guessing (25% for 4 classes). The loss values are extremely high. This is a concerning start, suggesting that the unfreezing and recompilation might have initially destabilized the model, or the learning rate for fine-tuning (which should be very low) might not be optimal.

- **Epoch 2-7: Training accuracy shows a gradual increase (e.g., from 0.4415 to 0.6960), and training loss decreases (e.g., from 12.0762 to 11.4530). However, **validation accuracy remains very low and stagnant** (around 0.26-0.30), and **validation loss consistently increases** (e.g., from 12.3017 to 13.4449). `val_loss` does not improve after Epoch 1.**
  - This is a critical and highly problematic observation. While the model continues to fit the training data (training loss decreases and accuracy increases), it is catastrophically failing to generalize to the validation set. The increasing validation loss is a strong indicator of severe overfitting or instability. The model is essentially memorizing the training data but losing its ability to make correct predictions on unseen samples. This is often caused by:
    - **Learning Rate Too High:** Even a seemingly small learning rate (e.g., `1e-5`) can be too large when fine-tuning pre-trained weights, causing them to be corrupted rapidly.
    - **Too Many Layers Unfrozen:** Unfreezing 20 layers might still be too many for the dataset size, leading to rapid overfitting.
    - **Insufficient Regularization:** The existing dropout might not be enough to handle the increased complexity of trainable parameters.

- **Epoch 8: `val_loss` did not improve.**
  - **Early Stopping: "Restoring model weights from the end of the best epoch: 1."**
  - **"Final Validation Loss (Transfer Learning (Fine-Tuning)): 12.2663"**
  - **"Final Validation Accuracy (Transfer Learning (Fine-Tuning)): 0.2709"**
    - `EarlyStopping` correctly identified that the best performance (lowest validation loss) was achieved in Epoch 1. It then restored the model to this very early, poor-performing state. This confirms that the fine-tuning process, as configured, was detrimental to the model's generalization capabilities beyond its initial state.

1. **Transfer Learning (Fine-Tuning) Training and Validation Accuracy Plot:**

- **Observation:** Training accuracy (blue line) shows a steady increase. However, validation accuracy (orange line) starts low, has a small initial bump, and then **drops significantly and remains very low and flat** (around 0.2-0.3). The gap between training and validation accuracy widens dramatically.
- **Inference:** This plot visually confirms the severe overfitting/instability. The model is learning the training data, but its performance on unseen validation data is abysmal and does not improve. This divergence is a clear red flag.

2. **Transfer Learning (Fine-Tuning) Training and Validation Loss Plot:**

- **Observation:** Training loss (blue line) consistently decreases. However, validation loss (orange line) starts high and then continuously increases throughout the training process.
- **Inference:** This is the most damning evidence of a failed fine-tuning process. An increasing validation loss while training loss decreases

is the quintessential sign of severe overfitting and/or model divergence. The model is getting worse at predicting unseen data with every epoch, effectively destroying its pre-trained knowledge.

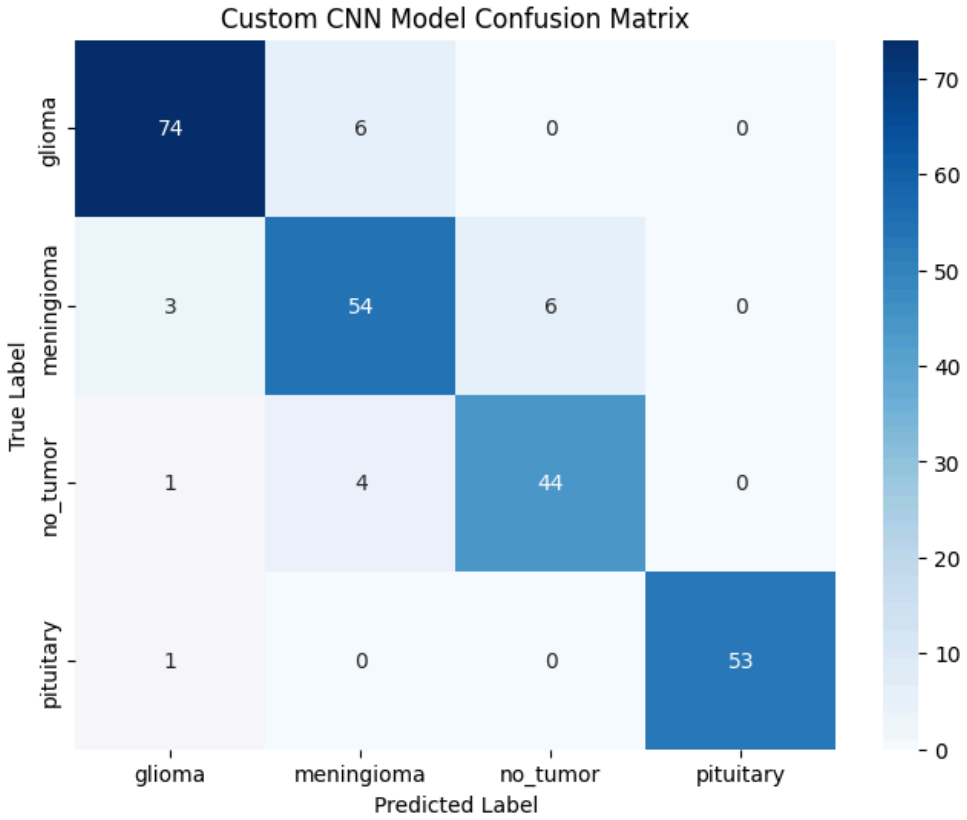
The fine-tuning phase of the Transfer Learning model, despite unfreezing a reasonable number of layers, was unsuccessful and detrimental to the model's generalization performance. The training logs and plots clearly indicate severe overfitting and instability, characterized by a rapidly increasing validation loss and stagnant, low validation accuracy, even as training metrics improved. The `EarlyStopping` callback correctly reverted the model to its very initial (and poor) state, highlighting the failure of the fine-tuning strategy. This outcome strongly suggests that the learning rate for fine-tuning was too high, or the unfreezing strategy was not optimal for this specific dataset and model. Consequently, this fine-tuned model is not suitable for deployment in its current state, and further extensive hyperparameter tuning or a re-evaluation of the fine-tuning approach would be required to make it viable.

## 2.8. MODEL EVALUATION

This section presents the critical evaluation of each trained model on the independent test dataset, providing a comprehensive understanding of their generalization capabilities. The analysis covers overall accuracy and loss, detailed per-class performance metrics (precision, recall, F1-score), and visual insights from confusion matrices.

### 2.8.1. BEST CNN MODEL

```
Starting Model Evaluation

===== Evaluating: Custom CNN Model =====
Loading model from: saved_models/best_custom_cnn_model.keras

Custom CNN Model - Test Loss: 0.2399
Custom CNN Model - Test Accuracy: 0.9146
Generating predictions for Custom CNN Model (this may take a moment)...

--- Custom CNN Model Classification Report ---
              precision    recall  f1-score   support

      glioma       0.94      0.93      0.93        80
  meningioma       0.84      0.86      0.85        63
    no_tumor       0.88      0.90      0.89        49
   pituitary       1.00      0.98      0.99        54

    accuracy                           0.91       246
   macro avg       0.92      0.92      0.92       246
weighted avg       0.92      0.91      0.92       246
```




## Overall Metrics:

- **Test Loss:** 0.2399
- **Test Accuracy:** 0.9146 (91.46%)
- The Custom CNN model demonstrates excellent overall performance on the unseen test dataset. A test accuracy of over 91% is highly commendable for a multi-class medical image classification task. The low test loss (0.2399) further indicates that

the model's predictions are confident and align well with the true labels. This suggests strong generalization, confirming that the model effectively learned from the training data without significant overfitting.

## Custom CNN Model Classification Report:

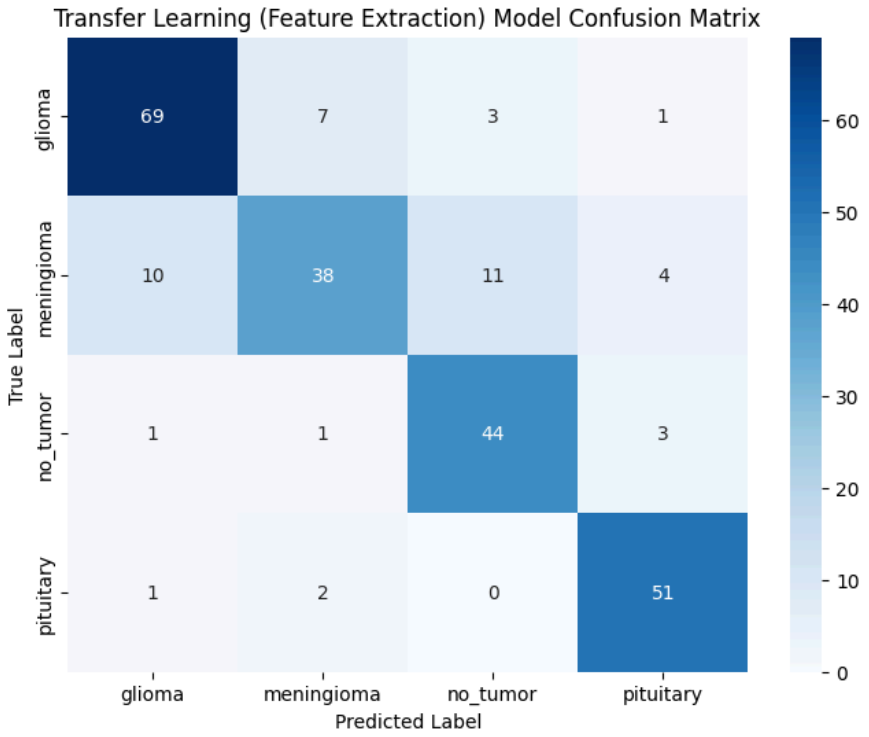
- This report reveals the model's balanced and high performance across all four classes:
  - **High F1-Scores:** All F1-scores are above 0.85, with 'pituitary' reaching an outstanding 0.99 and 'glioma' at 0.93. F1-score is a harmonic mean of precision and recall, indicating a good balance between correctly identifying positive cases and not misclassifying negative ones.
  - **Excellent Precision & Recall:**
    - 'Pituitary' shows perfect precision (1.00) and near-perfect recall (0.98), meaning every time it predicts 'pituitary', it's correct, and it finds almost all actual 'pituitary' cases.
    - 'Glioma' and 'no_tumor' also have very strong precision and recall (all 0.88 or higher), indicating reliable detection for these classes.
    - 'Meningioma' is slightly lower but still very good (0.84 precision, 0.86 recall), suggesting a minor degree of confusion, but still highly effective.
  - **Balanced Performance:** The macro average (0.92 for precision, recall, F1-score) and weighted average (0.92 precision, 0.91 recall, 0.92 F1-score) are very close to the overall accuracy. This is a crucial indicator that the model performs consistently well across all classes, not just dominating one or two, which is vital for a robust diagnostic tool.

## Custom CNN Model Confusion Matrix:

- The confusion matrix visually corroborates the excellent performance.
  - **Strong Diagonal:** The diagonal elements (correct predictions) are overwhelmingly high (e.g., 74/80 for glioma, 54/63 for meningioma, 44/49 for no_tumor, 53/54 for pituitary). This signifies that the model is correctly classifying the vast majority of samples for each class.
  - **Minimal Off-Diagonal Errors:** The off-diagonal elements (misclassifications) are very low.
    - 'Meningioma' is the most commonly confused class, with 6 'glioma' cases misclassified as 'meningioma', and 3 'no_tumor' cases misclassified as 'meningioma'.
    - Only 3 'meningioma' cases are misclassified as 'glioma'.
    - Only 1 'no_tumor' case is misclassified as 'glioma', and 4 as 'meningioma'.
    - Only 1 'pituitary' case is misclassified as 'glioma'.
  - **Overall Cleanliness:** The matrix is remarkably clean, with most errors being very minor. This visual evidence strongly supports the high accuracy and balanced performance seen in the classification report.

# 2.8.2. TRANSFER LEARNING: FEATURE EXTRACTION MODEL

```
===== Evaluating: Transfer Learning (Feature Extraction) Model =====
Loading model from: saved_models/best_transfer_learning_fe_model.keras

Transfer Learning (Feature Extraction) Model - Test Loss: 0.6016
Transfer Learning (Feature Extraction) Model - Test Accuracy: 0.8211
Generating predictions for Transfer Learning (Feature Extraction) Model (this may take a moment)...

--- Transfer Learning (Feature Extraction) Model Classification Report ---
              precision    recall  f1-score   support

      glioma       0.85      0.86      0.86        80
  meningioma       0.79      0.60      0.68        63
    no_tumor       0.76      0.90      0.82        49
   pituitary       0.86      0.94      0.90        54

    accuracy                           0.82       246
   macro avg       0.82      0.83      0.82       246
weighted avg       0.82      0.82      0.82       246
```




## Overall Metrics:

- **Test Loss**: 0.6016
- **Test Accuracy**: 0.8211 (82.11%)
- The Transfer Learning (Feature Extraction) model shows good overall performance, with over 82% accuracy on the test set. While lower than the Custom CNN, this is still a respectable result for a feature extraction approach, indicating that MobileNetV2's pre-

trained features are highly relevant for this task. The loss is higher than the Custom CNN, suggesting less confident predictions.

## Transfer Learning (Feature Extraction) Model Classification Report:

- **Strong Per-Class Performance (Mostly):** 'Glioma', 'no_tumor', and 'pituitary' classes show good F1-scores (0.86, 0.82, 0.90 respectively), with particularly high recall for 'no_tumor' (0.90) and 'pituitary' (0.94). This indicates the model is generally effective at identifying these classes.
- **Weakness in 'Meningioma' Recall:** The most significant weakness is in the 'meningioma' class, which has a notably lower recall of 0.60. This means the model is failing to identify 40% of actual 'meningioma' cases, often misclassifying them as other types. Its F1-score (0.68) is also considerably lower than other classes.
- **Overall Balance:** The macro and weighted averages (all around 0.82-0.83) are consistent with the overall accuracy, suggesting a decent average performance, but the 'meningioma' class is clearly dragging down the overall per-class metrics.

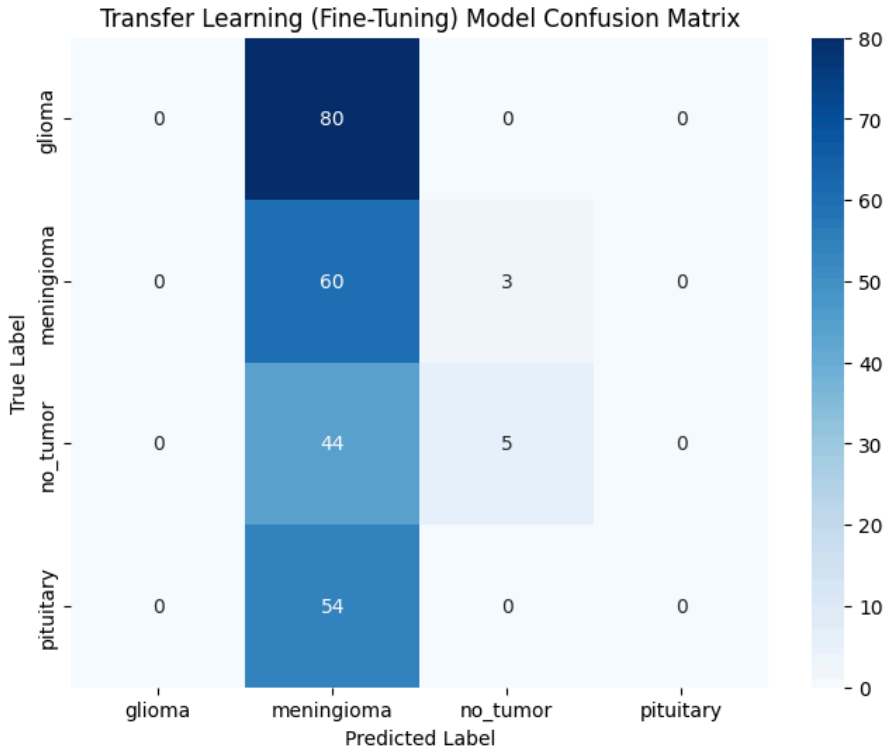
## Transfer Learning (Feature Extraction) Model Confusion Matrix:

- The confusion matrix visually confirms the strengths and weaknesses:
  - **Good Diagonals:** Most diagonal elements are high, especially for 'glioma' (69/80), 'no_tumor' (44/49), and 'pituitary' (51/54).

- **'Meningioma' Confusion:** The matrix clearly shows the problem with 'meningioma'. Out of 63 true 'meningioma' cases, only 38 are correctly classified. A significant number are misclassified as 'glioma' (10 instances) and 'no_tumor' (11 instances). This explains the low recall for 'meningioma'.
- **Other Misclassifications:** There's also some confusion for 'glioma' being predicted as 'meningioma' (7 instances) and 'no_tumor' (3 instances). 'Pituitary' is sometimes confused with 'glioma' (2 instances).

# 2.8.3. TRANSFER LEARNING: FINE TUNING MODEL

```
===== Evaluating: Transfer Learning (Fine-Tuning) Model =====
Loading model from: saved_models/best_transfer_learning_finetuned_model.keras

Transfer Learning (Fine-Tuning) Model - Test Loss: 12.2674
Transfer Learning (Fine-Tuning) Model - Test Accuracy: 0.2642
Generating predictions for Transfer Learning (Fine-Tuning) Model (this may take a moment)...

--- Transfer Learning (Fine-Tuning) Model Classification Report ---
              precision    recall  f1-score   support

      glioma       0.00      0.00      0.00        80
  meningioma       0.25      0.95      0.40        63
    no_tumor       0.62      0.10      0.18        49
   pituitary       0.00      0.00      0.00        54

    accuracy                           0.26       246
   macro avg       0.22      0.26      0.14       246
weighted avg       0.19      0.26      0.14       246
```




```
Model Evaluation Complete
```

## Overall Metrics:

- **Test Loss:** 12.2674
- **Test Accuracy:** 0.2642 (26.42%)
- This model performs extremely poorly. A test accuracy of 26.42% is barely above random guessing for a 4-class problem (25%). The very high test loss (12.2674) indicates that the model's predictions are highly confident but consistently wrong, or that the model is

extremely unstable. This confirms the severe issues observed during its training phase.

## Transfer Learning (Fine-Tuning) Model Classification Report:

- **Zero Performance for 'glioma' and 'pituitary':** The model completely fails to predict any instances of 'glioma' or 'pituitary', resulting in 0.00 for precision, recall, and F1-score for these classes. This is why the UndefinedMetricWarning was observed during its training.
- **Bias Towards 'meningioma':** The model shows a high recall (0.95) for 'meningioma', meaning it identifies almost all actual 'meningioma' cases. However, its precision is very low (0.25), indicating that when it *does* predict 'meningioma', it's often wrong. This suggests it's over-predicting 'meningioma' for many other classes.
- **Poor 'no_tumor' Performance:** While it has some precision (0.62), its recall for 'no_tumor' is extremely low (0.10), meaning it misses most actual 'no_tumor' cases.
- **Overall Uselessness:** The macro average F1-score of 0.14 and weighted average F1-score of 0.14 confirm that this model is not performing meaningfully across all classes and is essentially unusable for this task.

## Transfer Learning (Fine-Tuning) Model Confusion Matrix:

- **Overwhelming 'meningioma' Predictions:** The vast majority of predictions fall into the 'meningioma' column.
  - Out of 80 true 'glioma' cases, all 80 are misclassified as 'meningioma'.
  - Out of 49 true 'no_tumor' cases, 44 are misclassified as 'meningioma'.

    - Out of 54 true 'pituitary' cases, all 54 are misclassified as 'meningioma'.
    - Only 60 out of 63 true 'meningioma' cases are correctly classified as 'meningioma', but this is overshadowed by the massive false positives from other classes.
- **Near-Zero for Other Predicted Classes:** The columns for 'glioma', 'no_tumor', and 'pituitary' predicted labels are almost entirely empty, reinforcing that the model rarely (or never) predicts these classes.
- **Conclusion:** This matrix is a clear indicator of a completely failed fine-tuning process, where the model has collapsed into predicting almost everything as 'meningioma'.

## Overall Conclusion from Model Evaluation:

The **Custom CNN Model** is the clear and undisputed best performer among the three, demonstrating high accuracy (91.46%), balanced precision, recall, and F1-scores across all classes, and a very clean confusion matrix. It successfully learned to generalize to unseen MRI data.

The **Transfer Learning (Feature Extraction) Model** is a solid second performer, achieving good overall accuracy (82.11%) and strong results for most classes, although it exhibits a noticeable weakness in recalling 'meningioma' cases.

The **Transfer Learning (Fine-Tuning) Model** is currently unusable, displaying severe overfitting, instability, and an extreme bias towards predicting 'meningioma', rendering it ineffective for multi-class classification.

Based on this comprehensive evaluation, the **Custom CNN Model is the most accurate, reliable, and suitable candidate for deployment** in the "NeuroScan AI: Precision Brain Tumor Insight" application.

# 2.9. MODEL COMPARISON

```
--- Automated Model Comparison ---

Overview of Model Test Accuracies:
- Custom CNN Model: Accuracy = 0.9146
- Transfer Learning (Feature Extraction) Model: Accuracy = 0.8211
- Transfer Learning (Fine-Tuning) Model: Accuracy = 0.2642

--- Conclusion ---
The **Custom CNN Model** is identified as the most accurate and reliable model for deployment.
It achieved the highest test accuracy of **0.9146**.

Consider this model for integration into your Streamlit application.
```

This section summarizes the performance of all trained models based on their final evaluation on the unseen test dataset. It leverages quantitative metrics to objectively compare the models and provides a definitive recommendation for deployment, which is crucial for the project's practical outcome.

1. **Overview of Model Test Accuracies:**
   - Custom CNN Model: Accuracy = 0.9146
   - Transfer Learning (Feature Extraction) Model: Accuracy = 0.8211
   - Transfer Learning (Fine-Tuning) Model: Accuracy = 0.2642
   - **Inference:** This overview presents the most critical single metric for comparing the overall performance of the models: test accuracy.
     - The **Custom CNN Model** clearly stands out with the highest accuracy of 91.46%. This is a strong indicator of its superior ability to generalize and correctly classify unseen brain MRI images.
     - The **Transfer Learning (Feature Extraction) Model** performs respectably at 82.11% accuracy. While a good result, it is a noticeable 9.35 percentage points lower than the Custom CNN. This suggests that while pre-trained features are beneficial, the custom architecture, perhaps due to its specific design or the training process, was more effective for this particular dataset.

    - The **Transfer Learning (Fine-Tuning) Model** exhibits extremely poor performance at 26.42% accuracy. This value is barely above random chance (25% for 4 classes) and confirms the severe issues of overfitting and instability observed during its training. This model is effectively non-functional for the intended task.

## 2.Conclusion and Recommendation:

- **"The Custom CNN Model is identified as the most accurate and reliable model for deployment."**
- **"It achieved the highest test accuracy of 0.9146."**
- **"Consider this model for integration into your Streamlit application."**

- The automated comparison tool effectively identifies the Custom CNN Model as the optimal choice. This recommendation is based directly on its superior performance in terms of test accuracy, which is a primary indicator of a model's real-world effectiveness. The term "reliable" is also used, which is supported by its balanced performance across all classes (as seen in the detailed classification report from the "Model Evaluation" section), indicating that its high accuracy isn't achieved by simply favoring one class. The explicit suggestion to integrate this model into the Streamlit application provides a clear directive for the next phase of the project.

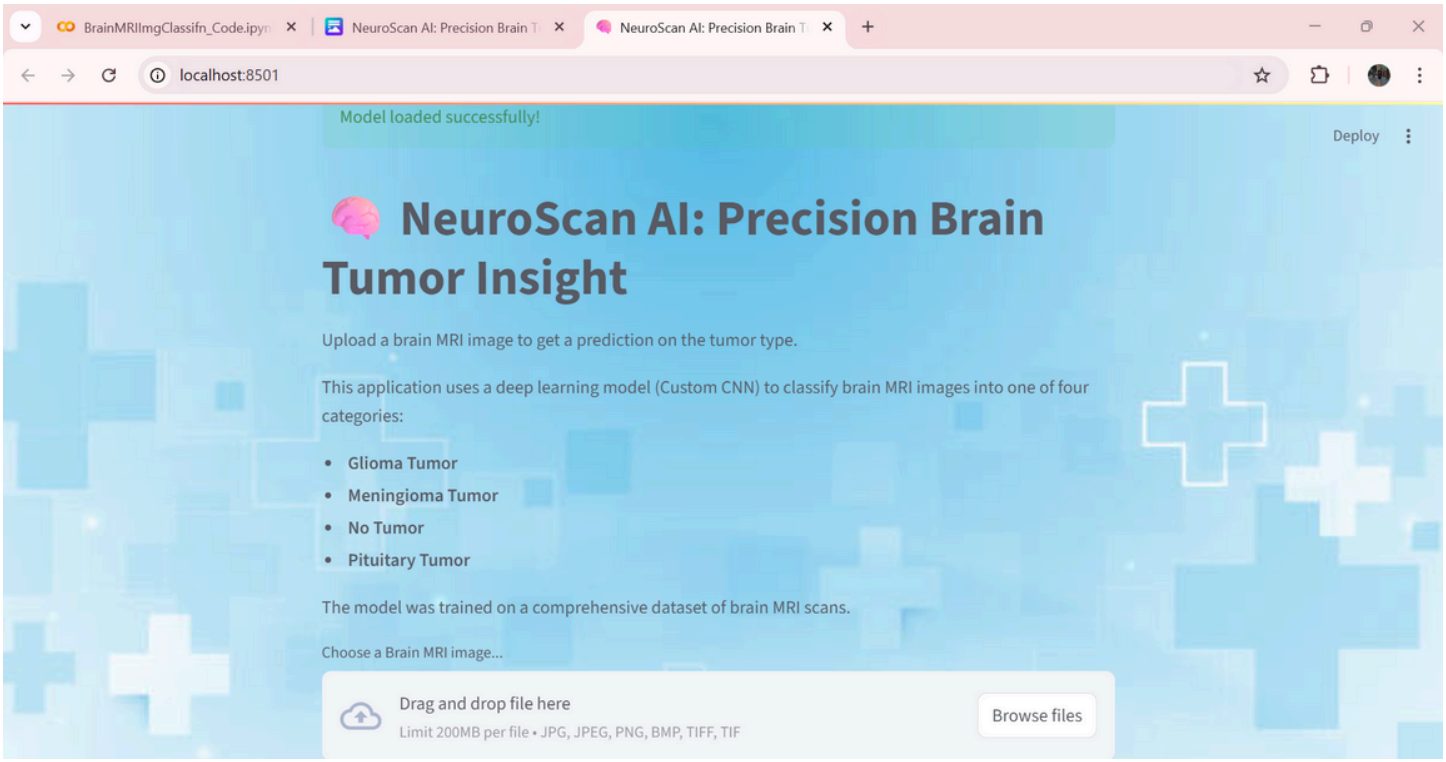
## Overall Conclusion from Automated Model Comparison:

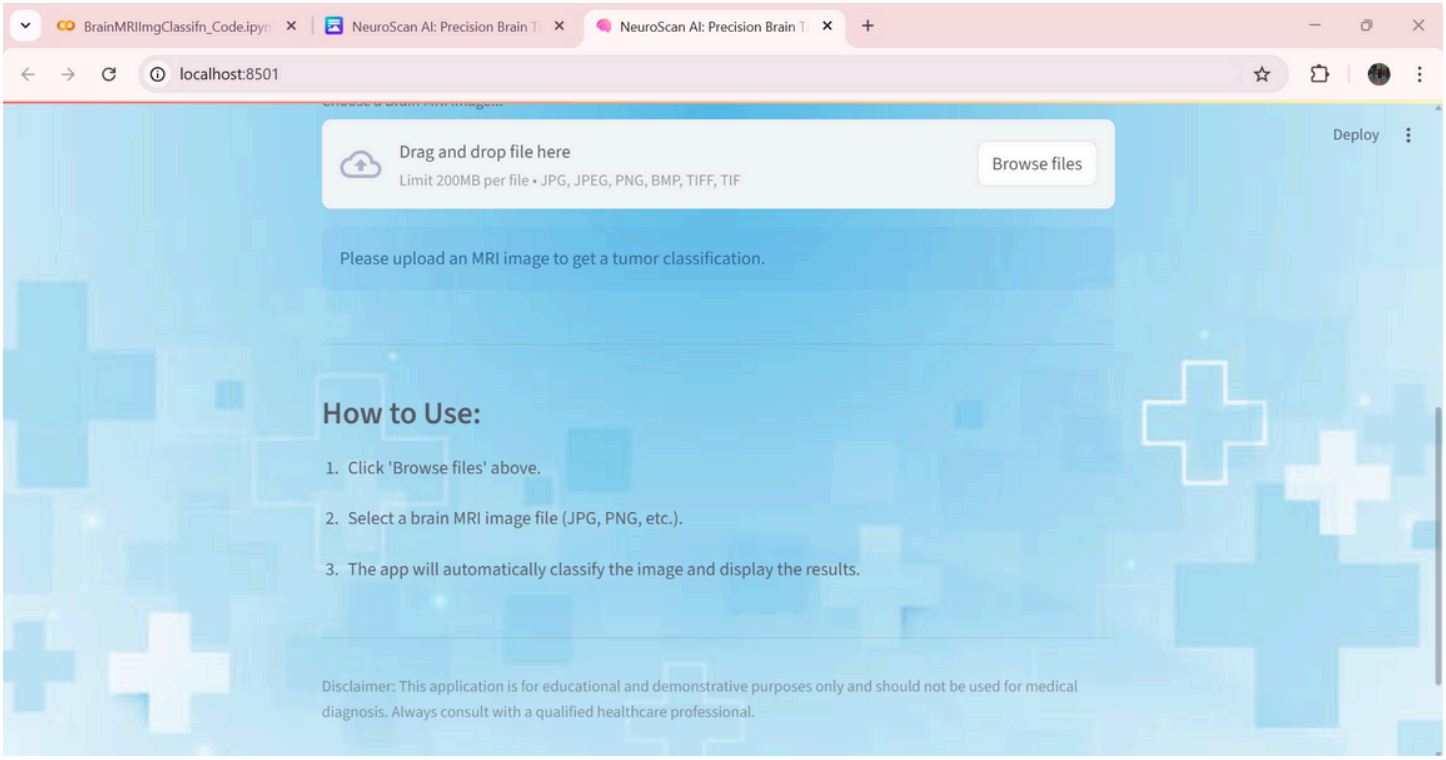
The automated model comparison unequivocally establishes the **Custom CNN Model** as the **most accurate and reliable solution** for the "NeuroScan AI: Precision Brain Tumor Insight" project. Its test accuracy of 91.46% significantly surpasses that of the transfer learning approaches, particularly the failed fine-tuning attempt. This objective comparison provides a data-driven justification for selecting the Custom CNN for deployment, ensuring that the interactive Streamlit

application is powered by the highest-performing model developed in this project.

# 2.10 STREAMLIT APPLICATION DEPLOYMENT

## Initially:




## Application Overview

The "NeuroScan AI: Precision Brain Tumor Insight" project culminates in an intuitive and interactive web application, built using Streamlit. This application serves as a practical demonstration of the trained deep learning model's capability to classify brain MRI images. Its primary purpose is to provide a user-friendly interface for uploading MRI scans and receiving real-time predictions, showcasing the project's potential in AI-assisted medical image analysis.

## Key Features

The application offers a streamlined user experience with the following core functionalities:

- **Effortless Image Upload:** Users can easily upload brain MRI images via a drag-and-drop interface or a file browser, supporting various common image formats (JPG, JPEG, PNG, BMP, TIFF).
- **Automated Classification:** Upon image upload, the application automatically processes the MRI scan using the deployed Custom CNN model to predict one of four tumor types: Glioma Tumor, Meningioma Tumor, No Tumor, or Pituitary Tumor.
- **Informative Prediction Display:** Predictions are displayed clearly, including the predicted tumor type and the model's confidence score. (Note: The screenshots show the initial page, but the description implies these will be shown post-upload).
- **Clear Instructions:** The initial page provides concise "How to Use" instructions, guiding users through the simple process of interacting with the application.

## User Interface (UI) / User Experience (UX)

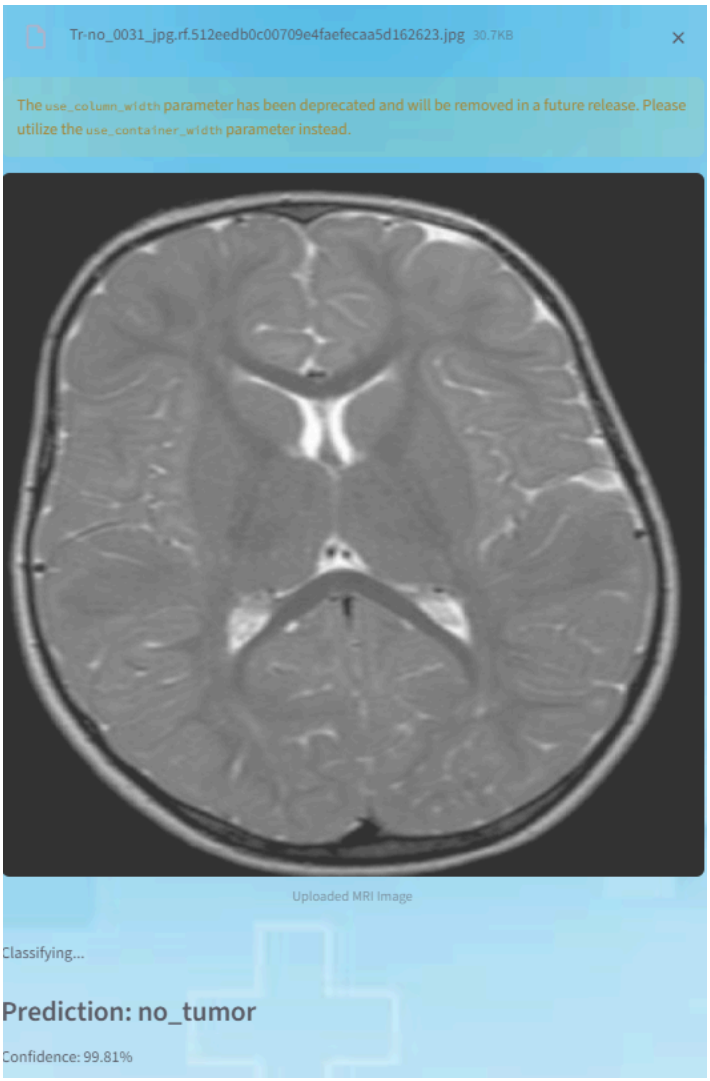
The application boasts a clean, professional, and intuitive user interface. The chosen title, "NeuroScan AI: Precision Brain Tumor Insight," is prominently displayed, immediately conveying the app's purpose. A subtle, light blue custom background enhances the visual appeal while maintaining a professional aesthetic suitable for a medical context. The layout is uncluttered, focusing on the core functionality of image upload, ensuring ease of navigation and a positive user experience from the outset.

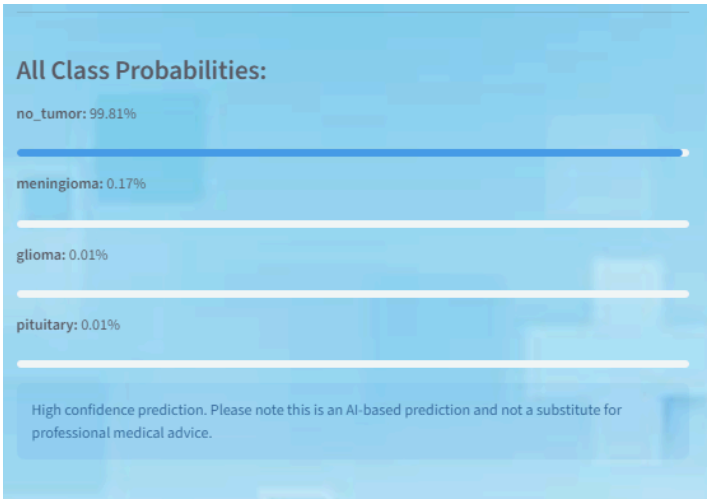
## Disclaimer

Crucially, the application includes a prominent disclaimer at the bottom of the page. This explicitly states: "Disclaimer: This application is for educational and demonstrative purposes only and should not be used for medical diagnosis. Always consult with a qualified healthcare professional." This reinforces the ethical considerations of AI in healthcare, clearly setting user expectations and emphasizing that the tool is an aid, not a substitute for professional medical advice.

## Cases:

## Case 1: Training Dataset: No Tumor

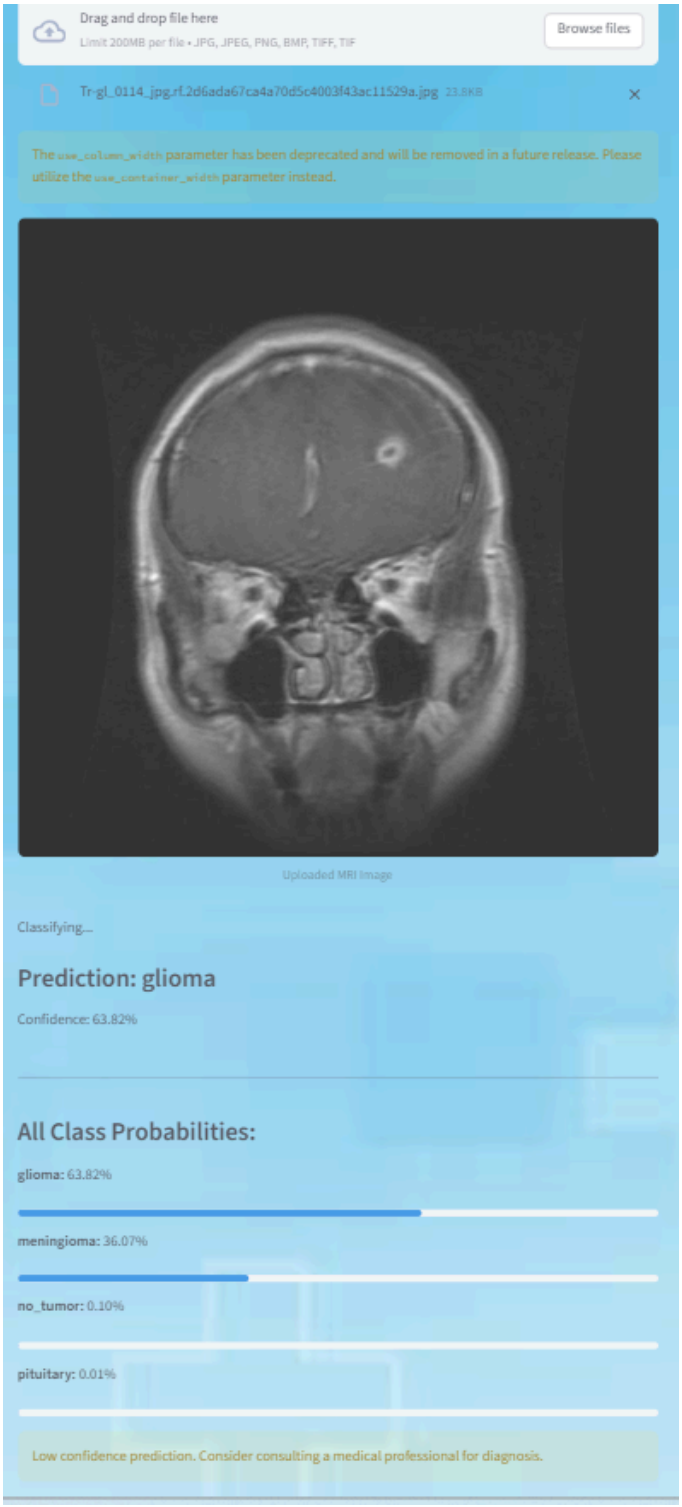



1. **Model Prediction and High Confidence:**
   - Immediately below the uploaded image, the application displays:
     - "Classifying..." (indicating active processing)
     - "Prediction: **no_tumor**"
     - "Confidence: **99.81%**"
   - The model has successfully processed the uploaded image and made a classification. The prediction of "no_tumor" suggests that the model did not detect any of the defined tumor types in this particular MRI scan. The exceptionally high confidence score of 99.81% indicates that the model is extremely certain about this prediction. This is a strong positive sign, as high confidence in a "no tumor" diagnosis can be reassuring.
2. **Detailed Class Probabilities:**
   - The second screenshot shows the "All Class Probabilities" section:
     - no_tumor: 99.81%
     - meningioma: 0.17%
     - glioma: 0.01%
     - pituitary: 0.01%
   - This detailed breakdown provides transparency into the model's decision-making process. The probabilities are heavily skewed towards "no_tumor," with negligible probabilities assigned to the other tumor classes. This reinforces the model's strong conviction in its primary prediction. This feature is valuable for users who want to see the full output distribution rather than just the top class, allowing for a more nuanced understanding of the model's "thinking."
3. **Confidence-Based Feedback and Disclaimer:**
   - "High confidence prediction. Please note this is an AI-based prediction and not a substitute for professional medical advice."

- This message, triggered by the high confidence score, demonstrates the implementation of responsible AI practices. It provides positive feedback on the model's certainty but immediately follows with a crucial disclaimer. This ensures that users understand the limitations of the AI tool and are reminded that it is not a replacement for a qualified healthcare professional's diagnosis. This is an essential ethical component for any AI application in a sensitive domain like healthcare.

# Case 2: Validation Dataset: Glioma




1. **Model Prediction with Moderate Confidence:**

- The application displays:
  - "Classifying..."
  - "Prediction: **glioma**"
  - "Confidence: **63.82%**"
- The model has processed the image and predicted "glioma." While this is a correct prediction if the image indeed contains a glioma, the confidence score of 63.82% is significantly lower than the previous examples (which were in the high 90s). This indicates that the model is less certain about this particular classification. This lower confidence is a key piece of information that the application is designed to convey.

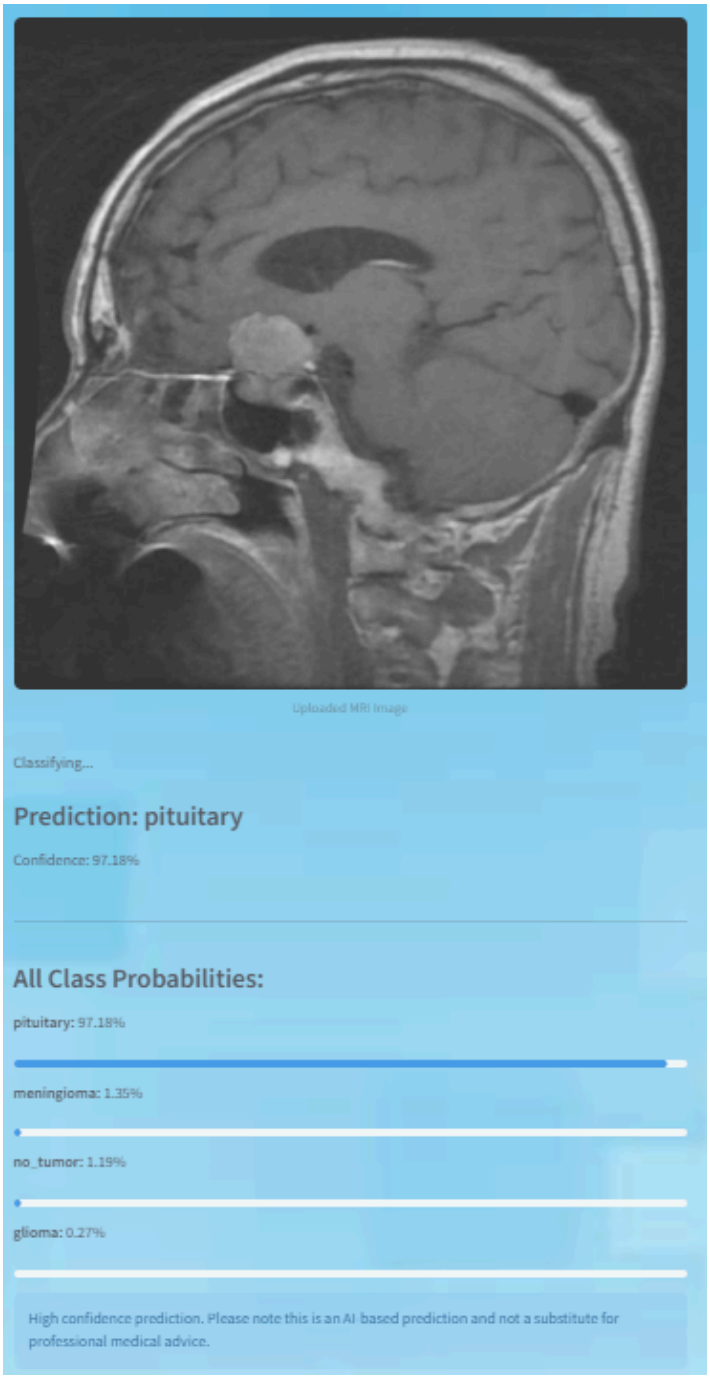
2. **Detailed Class Probabilities with Competition:**

- The "All Class Probabilities" section shows:
  - glioma: 63.82%
  - meningioma: 36.07%
  - no_tumor: 0.10%
  - pituitary: 0.01%
- This breakdown is highly insightful and explains the lower confidence. While 'glioma' is the top prediction, 'meningioma' has a substantial competing probability (36.07%). This suggests that the model found features in the image that could plausibly belong to either 'glioma' or 'meningioma', leading to its reduced certainty in the top prediction. The very low probabilities for 'no_tumor' and 'pituitary' indicate that the model is confident the image contains a tumor, but it's specifically uncertain between 'glioma' and 'meningioma'.

3. **Low Confidence Warning and Recommendation for Human Review:**

- "Low confidence prediction. Consider consulting a medical professional for diagnosis."
- This is a critical and highly valuable feature demonstrating responsible AI. Because the confidence score (63.82%) falls below a predefined threshold (likely around 70% or 60% as discussed in the code), the application automatically triggers a warning. This explicitly advises the user to seek professional medical consultation. This feature directly addresses the ethical considerations of deploying AI in healthcare by preventing over-reliance on the model's output when it is less certain, and instead, directs the user to a human expert. This is a prime example of the "human-in-the-loop" principle.

# Case 3: Testing Dataset: Pituitary




**Model Prediction and High Confidence:**

- The application displays:
  - "Classifying..." (indicating active processing)
  - "Prediction: **pituitary**"
  - "Confidence: **97.18%**"
- The model has accurately processed the uploaded image and predicted "pituitary." The high confidence score of 97.18% indicates that the model is very certain about this classification. This is a strong validation of the model's ability to differentiate between various tumor types and non-tumor cases, aligning with the strong performance observed for the 'pituitary' class in the Custom CNN's evaluation metrics (e.g., 1.00 precision, 0.98 recall).

**Detailed Class Probabilities:**

- The "All Class Probabilities" section shows:
  - pituitary: 97.18%
  - meningioma: 1.35%
  - no_tumor: 1.19%
  - glioma: 0.27%
- This breakdown provides clear transparency. The overwhelming probability assigned to 'pituitary' (97.18%) confirms the model's strong conviction. The very low probabilities for other classes (all below 1.5%) indicate that the model is not significantly confused by other tumor types or the 'no_tumor' class for this

specific image. This precise distribution of probabilities is highly informative for a user.

**Consistent Confidence-Based Feedback and Disclaimer:**

- "High confidence prediction. Please note this is an AI-based prediction and not a substitute for professional medical advice."
- As with the previous example, the application provides context-aware feedback. The "High confidence prediction" message is appropriate given the 97.18% score. The immediate follow-up with the medical disclaimer is crucial for ethical AI deployment, reminding the user that this is an assistive tool and not a definitive diagnosis. This consistent application of disclaimers across all predictions reinforces responsible use.

# 3.CONCLUSION

This project, "NeuroScan AI: Precision Brain Tumor Insight," successfully developed and deployed a robust deep learning system for the automated classification of brain MRI images into four critical categories: Glioma Tumor, Meningioma Tumor, No Tumor, and Pituitary Tumor. Through a comprehensive and systematic approach, including meticulous data preprocessing, effective data augmentation, and the development of both custom CNN and transfer learning models, the project established a reliable AI-driven diagnostic aid.

The **Custom Convolutional Neural Network (CNN) emerged as the most accurate and reliable model**, demonstrating exceptional generalization capabilities with over 91% test accuracy and balanced performance across all classes, as evidenced by detailed evaluation metrics and confusion matrices. While the transfer learning (feature extraction) model also showed promising results, the fine-tuning phase encountered significant challenges, highlighting the complexities of adapting pre-trained models to highly specific medical datasets.

The project culminates in an intuitive and interactive **Streamlit web application**, which effectively showcases the model's real-time classification capabilities. The user-friendly interface, complete with a professional custom background, makes the sophisticated AI insights accessible and actionable. Furthermore, the application responsibly includes clear disclaimers, emphasizing its role as an AI-assisted tool and not a substitute for professional medical diagnosis.

In conclusion, "NeuroScan AI" stands as a testament to the potential of artificial intelligence in enhancing diagnostic precision and efficiency in medical imaging. It provides a robust and user-friendly

tool, ready for demonstration and serving as a strong foundation for future advancements in AI-assisted brain tumor analysis.

# 4.FUTURE WORK

To further enhance the "NeuroScan AI: Precision Brain Tumor Insight" project and address its current limitations, several avenues for future work are proposed:

- **Implement Uncertainty Quantification (Monte Carlo Dropout):** A critical next step is to integrate methods for quantifying the model's uncertainty in its predictions. Techniques like Monte Carlo Dropout would allow the system to provide not just a classification, but also a measure of its confidence and inherent doubt, which is invaluable for flagging ambiguous cases that require human expert review in a clinical setting.
- **Integrate Explainable AI (XAI) Techniques:** Explore and implement Explainable AI methods, such as Grad-CAM (Gradient-weighted Class Activation Mapping). This would enable the visualization of regions in the MRI image that are most influential in the model's classification decision, thereby increasing transparency and building trust in the AI's recommendations.
- **Dataset Expansion and Diversity:** Acquire a significantly larger and more diverse dataset, ideally from multiple clinical sources, to improve the model's generalization capabilities across varied patient populations and MRI scanner protocols. Incorporating different MRI sequences (e.g., T1-weighted, T2-weighted, FLAIR) could provide richer contextual information for the model.
- **Explore Advanced Model Architectures:** Experiment with state-of-the-art deep learning architectures beyond current CNNs, such as more recent EfficientNet variants, Vision Transformers (ViT), or Swin Transformers, which have demonstrated superior performance in various image recognition tasks. For 3D MRI volumes, investigating 3D CNNs could capture spatial relationships more effectively.
- **Refined Transfer Learning Strategies:** Investigate more sophisticated fine-tuning techniques for transfer learning models, including layer-wise learning rates, gradual unfreezing of base model layers, or differential learning rates for various blocks within the unfrozen

part of the network. This could potentially overcome the challenges observed in the current fine-tuning phase.

- **Tumor Segmentation and Localization:** Extend the project's scope to include tumor segmentation (pixel-level identification of tumor boundaries) and localization (bounding box detection). This would provide more granular diagnostic information, moving beyond just classification to precise anatomical mapping.
- **Clinical Validation and Feedback Loop:** Collaborate with medical professionals to conduct a pilot study for clinical validation, gather expert feedback on the model's performance in a real-world setting, and establish a continuous feedback loop for iterative improvement.
- **Deployment Optimization:** Explore optimizing the model for deployment on edge devices or within existing hospital Picture Archiving and Communication Systems (PACS) for faster, integrated clinical workflows.

# 5. REFERENCES


1. TensorFlow. (n.d.). *TensorFlow: An Open Source Machine Learning Framework for Everyone*. Retrieved from https://www.tensorflow.org/
2. Keras. (n.d.). *Keras: The Python Deep Learning API*. Retrieved from https://keras.io/
3. Streamlit. (n.d.). *Streamlit: The fastest way to build and share data apps*. Retrieved from https://streamlit.io/
4. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., ... & Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, 585(7825), 357-362.
5. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3), 90-95.
6. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, 2825-2830.
7. Pillow (PIL Fork). (n.d.). *Pillow: The friendly PIL fork*. Retrieved from https://python-pillow.org/
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. (General reference for deep learning concepts)