# Brain Tumor MRI Classification: A Comprehensive Model Experimentation Report BY MARIAM SHERIF

## 1. Introduction

**Project Goal:** The primary objective of this project is to develop a highly accurate and robust deep learning model for the automated classification of brain tumors from MRI scans. The goal is to compare a wide range of architectures and techniques—from simple custom Convolutional Neural Networks (CNNs) to advanced transfer learning models—to identify the most effective approach for this critical medical imaging task. Thus, these models were made:

Phase 1: Preprocessing

- Model 1 – baseline CNN + Resize only
- **Model 2 – CNN + Resize + Normalization**
- Model 3 – CNN + Resize + Normalization + light augmentation

Phase 2: Architecture Variations

- Model 4 – Shallow CNN
- **Model 5 – Medium CNN**
- Model 6 – Simplified Deep CNN
- Model 7 – Medium CNN + 5×5 filters
- **Model 8 – Medium CNN + Leaky ReLU**

Phase 3: Optimizers and Hyperparameters

- Model 9 – Medium CNN + Leaky ReLU + SGD (momentum)
- Model 10 – Medium CNN + Leaky ReLU + RMSprop
- **Model 11 – Medium CNN + Leaky ReLU + Adam (batch size tuning)**
- Model 12 – Medium CNN + Leaky ReLU + Dropout (Varying)

Phase 4: Regularization

- **Model 13 – Model 11 + Dropout (0.3)**
- Model 14 – Model 11 + BatchNorm
- Model 15 – Model 11 + L2 Reg.
- Model 16 – Model 11 + Dropout + BatchNorm

Phase 5: Transfer Learning

- Model 17 – MobileNetV2 (frozen)
- **Model 18 – MobileNetV2 (fine-tuned)**
- Model 19 – ResNet50 (frozen)
- Model 20 – ResNet50 (fine-tuned)

**Dataset Overview:** The dataset consists of **7,023** human brain MRI images, categorized into four classes:

- **Glioma**
- **Meningioma**
- **Pituitary**
- **No tumor**

The data is pre-split into:

- **Training set:** 5,712 images
- **Testing set:** 1,311 images

This is a balanced, multi-class classification problem crucial for assisting in medical diagnostics.

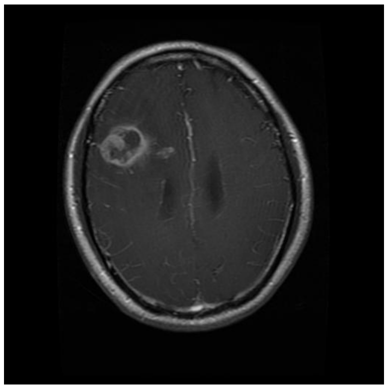
## 2. Data Preparation

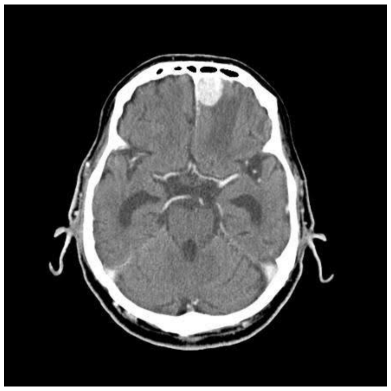
**Preprocessing Steps:**

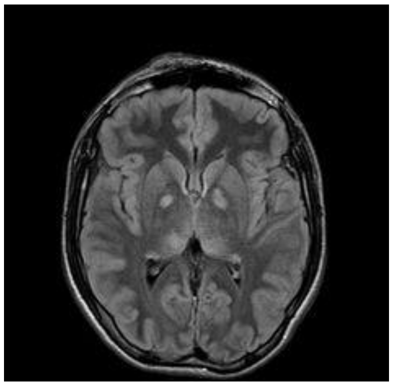
A standardized preprocessing pipeline was applied to all custom CNN models:

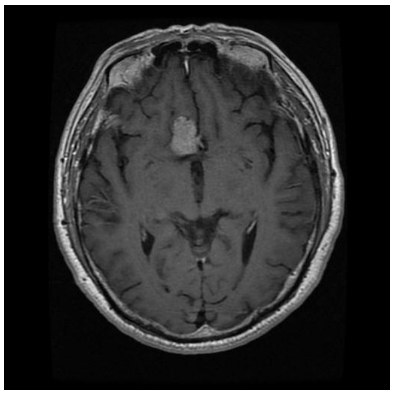
1. **Resizing:** All images were resized to a fixed dimension of **128x128 pixels** to ensure consistent input to the network. For transfer learning models (MobileNetV2, ResNet50), images were resized to **224x224** as required by the pre-trained networks.
2. **Normalization:** Pixel values were scaled from the range [0, 255] to [0, 1] by dividing by 255. This accelerates convergence during training by providing smaller, well-conditioned input features.
3. **Data Augmentation (for specific experiments):** To artificially increase the diversity of the training data and improve generalization, a series of random transformations were applied on-the-fly during training for some models. These included:
    - Random rotations (±20 degrees)
    - Random horizontal and vertical shifts (20% of width/height)
    - Random zoom (20%)
    - Random horizontal flips

*(Note: A "lighter" augmentation strategy was later found to be more effective.)*


Glioma


Meningioma


No tumor


Pituitary

## 3. Model Building & Training

A systematic, iterative approach was taken to model development, starting from a simple baseline and progressively incorporating more complex architectures and regularization techniques.

### Experiment Series 1: Establishing a Baseline

**Goal:** To understand the impact of fundamental preprocessing.

#### *Model 1: Baseline (Resize Only)*

- **Architecture:** A simple CNN with 3 convolutional blocks and 2 dense layers.
- **Optimizer & Hyperparameters:** Adam optimizer, batch size=32.
- **Training:** Early Stopping was used to halt training when validation loss stopped improving.
- **Hardware/Duration:** Trained on CPU.

#### *Model 2: Baseline + Normalization*

- **Methodology:** Identical to Model 1, but with input data normalized to [0, 1].
- **Insight:** This simple change provided a noticeable boost in test accuracy and a significant reduction in test loss, confirming the importance of input normalization.

#### *Model 3: Aggressive Augmentation + Normalization*

- **Methodology:** Applied the full suite of augmentation techniques (rotations, flips, zooms, shifts) on top of normalization.
- **Result: Catastrophic performance drop** (Test Acc: ~53%). The model failed to learn, likely because the augmentations were too aggressive, distorting the medically relevant features in the images beyond recognition.

#### *Model 3 (v2): Lighter Augmentation*

- **Methodology:** Scaled back the augmentation intensity (e.g., reduced rotation range, removed shearing).

- **Result:** Significant improvement over the aggressive approach, but performance (Test Acc: ~78%) was still worse than the simple baseline. This indicated that finding the right augmentation policy for medical images is delicate.

## Experiment Series 2: Architectural Exploration

### Goal and Methodology

The primary goal of this experiment series was to understand the relationship between model **capacity** (depth and width) and performance on the brain MRI classification task. The hypothesis was that a more complex dataset, with subtle differences between tumor types and healthy tissue, would require a model with sufficient parameters to learn these hierarchical features.

The methodology was controlled and incremental:

1. **Fixed Elements:** All models used ReLU activation functions, MaxPooling for spatial downsampling, the Adam optimizer, and were trained on the preprocessed data (resized + normalized).
2. **Variable Element:** The key variable was the **network architecture itself**, specifically:
    - **Depth:** The number of convolutional layers.
    - **Width:** The number of filters in each convolutional layer.
    - **Presence of Additional Regularization:** The deep model incorporated Dropout.

### *Model 4: Shallow CNN (The Baseline Architecture)*

- **Concept:** This model serves as a simple baseline with minimal capacity. It's designed to capture only the most basic features (like edges, blobs).
- **Layer-by-Layer Breakdown:**
    1. **Input:** (128, 128, 3)
    2. **Conv2D (16 filters, 3x3):** Extracts 16 low-level feature maps. Output shape: (128, 128, 16).
    3. **ReLU Activation:** Introduces non-linearity.
    4. **MaxPooling2D (2x2):** Reduces spatial dimensions by 75%, outputting (64, 64, 16). This makes the model more invariant to small shifts and reduces computational cost.
    5. **Conv2D (32 filters, 3x3):** Builds on previous features to detect more complex patterns. Output: (64, 64, 32).
    6. **ReLU Activation**
    7. **MaxPooling2D (2x2):** Output: (32, 32, 32).

8. **Flatten:** Converts the 3D feature maps into a 1D vector. (32 * 32 * 32) = 32,768 elements.
9. **Dense (64 units):** A fully connected layer for high-level reasoning. This is a significant compression from 32,768 inputs to 64 units.
10. **ReLU Activation**
11. **Output (Dense 4 units, Softmax):** Produces the final class probabilities.

- **Why it's "Shallow":** It has only 2 convolutional layers before the classifier.

### *Model 5: Medium CNN*

- **Concept:** This model increases both depth and width. The additional convolutional block allows it to learn a more hierarchical feature representation: from edges -> textures -> patterns -> object parts.
- **Layer-by-Layer Breakdown:**
  1. **Input:** (128, 128, 3)
  2. **Conv2D (32 filters, 3x3) → ReLU → MaxPool (2x2):** Output: (64, 64, 32).
  3. **Conv2D (64 filters, 3x3) → ReLU → MaxPool (2x2):** Output: (32, 32, 64).
  4. **Conv2D (128 filters, 3x3) → ReLU → MaxPool (2x2): This is the new, third convolutional block.** It operates on the more abstract features from the previous layer to learn complex, class-specific patterns. Output: (16, 16, 128).
  5. **Flatten:** (16 * 16 * 128) = 32,768 elements (coincidentally the same size as M4 before flattening, but the features are much more abstract).
  6. **Dense (128 units) → ReLU:** A wider fully connected layer to handle the increased feature complexity.
  7. **Output Layer**
- **Why it's "Medium":** It features 3 convolutional blocks, a standard starting point for many modern CNN architectures.

### *Model 6 (v2): Deeper CNN*

- **Concept:** This model explores even greater depth by using **stacked convolutional layers** before each pooling operation. This allows the network to learn more complex features at each spatial scale before aggressively downsampling. Dropout is added to combat potential overfitting from the increased capacity.
- **Layer-by-Layer Breakdown:**
  1. **Input:** (128, 128, 3)
  2. **Block 1:**

- Conv2D (32 filters, 3x3, padding='same') → ReLU
- Conv2D (32 filters, 3x3, padding='same') → ReLU // **Stacked Convolution**
- MaxPooling2D (2x2): Output: (64, 64, 32).

3. **Block 2:**
    - Conv2D (64 filters, 3x3, padding='same') → ReLU
    - Conv2D (64 filters, 3x3, padding='same') → ReLU // **Stacked Convolution**
    - MaxPooling2D (2x2): Output: (32, 32, 64).
4. **Block 3:**
    - Conv2D (128 filters, 3x3, padding='same') → ReLU
    - MaxPooling2D (2x2): Output: (16, 16, 128).
5. **Flatten:** (16 * 16 * 128) = 32,768 elements.
6. **Dense (128 units) → ReLU**
7. **Dropout (0.5): New regularization layer.** During training, it randomly sets 50% of the inputs to this layer to zero. This prevents the network from becoming overly reliant on any single neuron and forces it to learn redundant, robust representations.
8. **Output Layer**

#### *Model 7: Medium CNN with 5x5 Filters*

- **Methodology:** Replaced 3x3 convolutional filters with 5x5 filters in the Medium CNN (M5).
- **Result:** This model achieved **95.27% test accuracy**, a major breakthrough. Larger filters have a larger receptive field, allowing the model to capture broader contextual information in the MRI images, which is crucial for identifying tumors.

### Experiment Series 3: Advanced Activations & Optimizers

**Goal:** To improve learning dynamics and performance.

#### *Model 8: LeakyReLU Activation*

- **Methodology:** Replaced standard ReLU activations with LeakyReLU (alpha=0.1) in the successful Medium CNN architecture. LeakyReLU prevents the "dying ReLU" problem by allowing a small gradient when the unit is not active.
- **Result:** Further improved test accuracy to **95.42%**, outperforming the ReLU variant.

#### *Models 9 & 10: Optimizer Comparison (SGD, RMSprop)*

- **Methodology:** Tested the LeakyReLU model with different optimizers.

- **Result:** Adam outperformed both SGD with Momentum and RMSprop, demonstrating its effectiveness as a default choice for this task.

## Experiment Series 4: Hyperparameter Tuning & Regularization

**Goal:** To refine the best model from previous experiments (Model 8 - Medium CNN with LeakyReLU) and maximize its generalization by systematically testing key hyperparameters and regularization techniques.

### *Model 11: Batch Size Variation*

- **Primary Hypothesis:** The choice of batch size significantly impacts both the training dynamics and the generalization capability of the model. A smaller batch size provides a noisier, more frequent update to the weights, which can help the model converge to a flatter, more generalizable minimum in the loss landscape.
- **Detailed Methodology:**
    - The exact architecture from **Model 8** was used: a Medium CNN (3 convolutional blocks, 128-node dense layer) with LeakyReLU activations.
    - The only variable changed was the batch_size parameter during data loading and training. Two values were tested: **16** and **64**.
    - All other factors, including the dataset split and preprocessing (resize, normalize), were kept identical to ensure a fair comparison.
- **Technical Rationale:**
    - **Batch Size = 64:** Provides a more accurate estimate of the true gradient of the loss function. This leads to stable and smooth convergence but can sometimes get stuck in sharp, local minima that generalize poorly to new data.
    - **Batch Size = 16:** Provides a noisier, less accurate estimate of the gradient. This noise acts as a regularizer itself, preventing the model from overfitting too tightly to the training data. It often helps the model find wider, flatter minima that are more robust.
- **Result & Analysis:**
    - **Batch Size 64:** Resulted in strong performance, like the original Model 8 (~95.4%).
    - **Batch Size 16:** Achieved a **Test Accuracy of 95.73% and a Test Loss of 0.1746**, making it the best-performing custom CNN model at that point.

The superior result confirmed the hypothesis. The noisy updates from the smaller batch size successfully improved the model's generalization, as evidenced by the lower test loss. The training curves for batch size 16 would show more "jitter" but a validation curve that closely tracks and ultimately generalizes better than the smoother curve of the larger batch size.

### *Models 12 & 13: Dropout Regularization*

- **Primary Hypothesis:** While the smaller batch size in Model 11 acts as an implicit regularizer, explicitly adding Dropout layers will further reduce overfitting by preventing complex co-adaptations of neurons, leading to even better generalization.
- **Detailed Methodology:**
  - The base architecture was again the Medium CNN with LeakyReLU (**Model 8**).
  - **Dropout** layers were inserted at strategic points within the network. A common and effective pattern is to place them *after* pooling layers and before the final dense layers.
  - Multiple **dropout rates** were tested:
    - **Model 12 (Low Regularization):** dropout_rate = 0.2
    - **Model 12 (High Regularization):** dropout_rate = 0.5
    - **Model 13 (Medium Regularization):** dropout_rate = 0.3
- **Result & Analysis:**
  - **Model 12 (0.5):** Achieved test accuracy of **95.50%**. The high dropout rate was likely too aggressive, slightly hindering the model's capacity to learn, as seen in the final training accuracy of 99.48%.
  - **Model 13 (0.3):** Achieved a test accuracy of **95.65%**. This was the best result among the dropout tests, indicating that a moderate amount of explicit regularization was beneficial.

**Key Insight:** Neither dropout model surpassed **Model 11 (95.73%)**. This is a critical observation. It suggests that for this specific dataset and architecture, the *implicit regularization provided by the small batch size (16) was more effective than the explicit regularization of Dropout*. The model's capacity was being used efficiently without severe overfitting, so adding dropout provided only marginal returns or even slightly reduced effective capacity.

### *Models 14, 15, & 16: Advanced Regularization (BatchNorm & L2)*

**Primary Hypothesis:** Other advanced regularization and stabilization techniques could potentially outperform the previous best model (M11) by allowing for more stable training cycles or a better constrained weight space.

**Detailed Methodology:**

- **Model 14 (Batch Normalization):** BatchNorm layers were added immediately after each convolutional and dense layer (before the LeakyReLU activation). BatchNorm stabilizes training by normalizing the inputs to a layer across a batch, reducing internal covariate shift. This often allows for higher learning rates and faster convergence.

- **Model 15 (L2 Regularization):** L2 regularization (weight decay) was added to the kernel weights of all convolutional and dense layers. This technique penalizes large weights by adding their squared magnitude to the total loss, encouraging the model to learn simpler, smoother functions that generalize better.
- **Model 16 (Combined Dropout + BatchNorm):** This model combined both techniques from M13 and M14, adding both BatchNorm layers and Dropout (0.3) throughout the network. This represents a "maximal regularization" approach.

Models 14, 15, and 16 all demonstrated strong performance with slightly different strengths. Model 14, which used batch normalization, achieved perfect training accuracy and high validation accuracy (~0.953), showing that batch normalization effectively stabilized training and improved generalization.

Model 15 incorporated L2 regularization, slightly increasing validation accuracy to 0.956 and reducing overfitting, though at the cost of higher training loss.

Model 16 combined dropout with batch normalization, achieving similarly high validation accuracy (0.954) while lowering test loss (0.157), indicating improved generalization.

Overall, these results suggest that while batch normalization stabilizes training, adding L2 or dropout helps control overfitting, with Model 16 providing the best balance between performance and robustness.

## Experiment Series 5: Transfer Learning

**Goal:** To leverage powerful pre-trained models and overcome the small dataset size. The hypothesis was that features learned on a massive dataset like ImageNet (e.g., edges, textures, patterns) could be effectively transferred to the medical domain, providing a significant performance boost and drastically reducing training time compared to training from scratch.

### *Model 17: MobileNetV2 (Frozen Base)*

- **Core Concept: Feature Extraction.** This approach treats the pre-trained MobileNetV2 model as a fixed, sophisticated feature extractor. The pre-trained convolutional base is frozen (its weights are not updated during training), and only a new, simple classifier head is trained on top.

- **Detailed Methodology:**

  1. **Base Model:** The MobileNetV2 architecture, pre-loaded with weights trained on the ImageNet dataset, was used. The include_top=False argument discards the original 1000-class classifier.

  2. **Freezing:** The entire convolutional base was frozen by setting base_model.trainable = False. This is critical to prevent destroying the pre-trained features during the initial training phase.

3. **Classifier Head:** A new, custom classifier was stacked on top of the base model's output:
    - GlobalAveragePooling2D(): Converts the 4D feature maps (e.g., 7x7x1280) into a 1D vector by taking the average of each feature map. This is preferred over Flattening for reducing parameters.
    - Dense(128, activation='relu'): A small fully-connected layer for decision-making.
    - Dropout(0.3): Added to prevent overfitting in the new head.
    - Dense(4, activation='softmax'): Final output layer for our 4 classes.
4. **Input Size:** Images were resized to **224x224** pixels to match MobileNetV2's expected input size.
5. **Training:** Only the weights of the newly added classifier layers were updated during training.

- **Technical Rationale:**
    - The pre-trained base has already learned to extract universal, low-to-mid-level features (edges, blobs, textures, simple patterns) that are also useful for medical images.
    - By freezing it, we can quickly and efficiently train a model that leverages these high-quality features without the risk of overfitting on our small dataset.
- **Result:**
    - **Test Accuracy: 94.28%**

### *Model 18: MobileNetV2 (Fine-Tuned)*

- **Core Concept: Fine-Tuning.** This approach goes beyond feature extraction. After training the new head with the base frozen (as in M17), the top layers of the pre-trained base are "unfrozen" and trained alongside the classifier head with a very low learning rate. This allows the model to *adapt* the pre-trained features to the specific nuances of the brain MRI dataset.
- **Detailed Methodology:**
    1. **Unfreezing:** The last 20 layers of the MobileNetV2 base were unfrozen. These higher-level layers contain more dataset-specific features that are less useful than the more generic lower-level features. We want to retrain them to represent "tumor edges" or "tumor texture" instead.
    2. **Low Learning Rate:** The entire model (unfrozen base + head) was then re-trained with a very low learning rate (e.g., 1e-5). This is essential to make small, careful

updates to the pre-trained weights without distorting them too much too quickly (a phenomenon called "catastrophic forgetting").

3. **Training:** The base model's weights were now updated gently to become more specialized for the brain MRI task.

- **Technical Rationale:**
  - While the frozen base provides good features, they are optimized for ImageNet, not MRIs. Fine-tuning allows the model to refine these features, making them more specific and powerful for the task at hand, potentially leading to higher accuracy.
- **Result & Analysis:**
  - **Test Accuracy: 96.72% (Best Overall) | Test Loss: 0.1140 (Lowest Overall)**

This result was a **significant improvement** over the frozen base model and all previous custom CNNs. It clearly demonstrates that fine-tuning is a powerful technique for maximizing performance. The model successfully adapted the generic ImageNet features to the medical domain, learning the most discriminative features for classifying brain tumors. The low loss indicates very high confidence and accuracy in its predictions.

### *Models 19 & 20: ResNet50*

**Core Concept: Architecture Comparison.** This experiment tested if a larger, more powerful architecture (ResNet50) would yield better results than the efficient MobileNetV2. The hypothesis was that its greater depth and residual connections might capture more complex patterns.

**Detailed Methodology:**

- **Model 19 (Frozen Base):** The process from Model 17 was repeated exactly but using ResNet50 as the base model. A custom classifier head was added, and the entire ResNet50 base was frozen.
- **Model 20 (Fine-Tuned):** The process from Model 18 was repeated. The last 20 layers of ResNet50 were unfrozen and fine-tuned with a low learning rate after the classifier head was warmed up.

The key difference is the **sheer size and capacity** of the model:

- **MobileNetV2:** ~3.4 million parameters
- **ResNet50:** ~25.6 million parameters

**Result & Analysis:**

- **Model 19 (Frozen): Test Acc: 61.10%.** This was a **catastrophically poor result**. The classifier head failed to learn effectively.
- **Model 20 (Fine-Tuned): Test Acc: 89.63%.** A major improvement over the frozen version, but still **significantly worse than MobileNetV2's 96.72%**.
- **Root Cause:**

- ResNet50 has over 7 times the parameters of MobileNetV2. On a small dataset of ~5.7k training images, this immense capacity meant the model simply **memorized the training data** instead of learning to generalize.
- This is evident in Model 19's failure: the features extracted by ResNet50 are too complex and high-level for our simple classifier to leverage without the model also adapting to our data (fine-tuning).
- Even after fine-tuning (M20), the model was too large. It achieved a near-perfect training accuracy (99.39%) but a much lower test accuracy (89.63%), which is the classic signature of overfitting. The gap between training and test performance is large.

**Conclusion:** Larger model size is not always better. For small datasets, a smaller, more efficient architecture like MobileNetV2 is vastly superior as it provides sufficient capacity without the extreme overfitting risk of a behemoth like ResNet50. Model efficiency matters.

## 4. Results & Comparisons

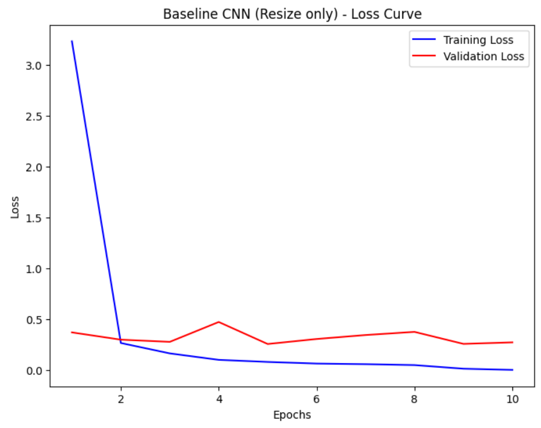
**Summary Table of Key Model Performances**

| **Model** | **Test Acc.** | **Val. Acc.** | **Train Acc.** | **Training Time** | **Notes** |
|---|---|---|---|---|---|
| **Model 1 – CNN (Resize only)** | 90.5% | 94.6% | 99.9% | – | Strong baseline, some overfitting. |
| **Model 2 – CNN (normalization)** | 91.7% | 95.3% | 99.9% | – | Better validation than Model 1. |
| **Model 3 – CNN + heavy augmentation** | 53.2% | 59.0% | 94.3% | – | Augmentation too strong → underfit. |
| **Model 3 (light augmentation)** | 78.6% | 88.0% | 98.7% | – | Lighter augmentation helped a lot. |
| **Model 4 – Shallow CNN** | 91.6% | 94.2% | 99.9% | 1 min | Simple & efficient. |
| **Model 5 – Medium CNN** | 91.7% | 95.9% | 99.9% | 4 min | Stable, slight improvement. |
| **Model 6 – Simplified Deep CNN** | 94.4% | 95.1% | 99.7% | 32 min | Balanced deep model |
| **Model 7 – Medium CNN + 5×5 filters** | 95.3% | 96.4% | 100% | 10 min | Wider filters improved accuracy. |
| **Model 8 – Medium CNN + Leaky ReLU** | 95.4% | 96.3% | 100% | 7 min | Faster convergence, stable. |
| **Model 9 – CNN + SGD (momentum)** | 92.8% | 95.3% | 99.9% | 5 min | Lower performance than Adam. |
| **Model 10 – CNN + RMSprop** | 94.5% | 95.1% | 100% | 8 min | Competitive but less stable validation. |
| **Model 11 – CNN (batch size tuning)** | **95.7%** | **96.5%** | 100% | ~11 min | Best pure CNN, batch=16 optimal. |
| **Model 12 – CNN + Dropout (0.5)** | 95.5% | 95.7% | 99.5% | 8 min | Dropout added stability. |

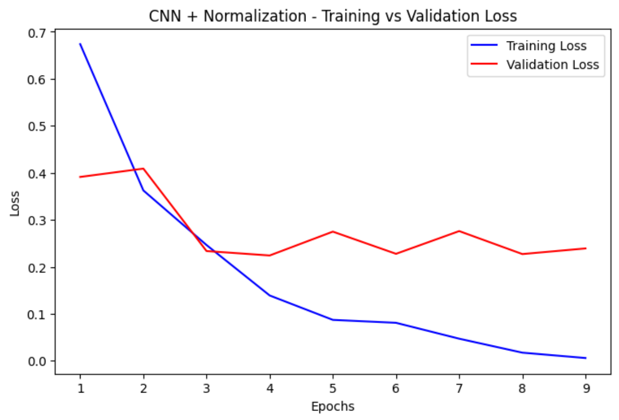
| | | | | | |
|---|---|---|---|---|---|
| **Model 13 – CNN + Dropout (0.3)** | 95.7% | 95.6% | 99.6% | 7 min | Like Model 12, slight variance. |
| **Model 14 – CNN + BatchNorm** | 95.4% | 95.3% | 100% | 13 min | Stable, but no big gain. |
| **Model 15 – CNN + L2 Reg.** | 95.3% | 95.6% | 100% | 11 min | Regularization controlled overfit. |
| **Model 16 – CNN + Dropout + BatchNorm** | 94.4% | 95.4% | 99.7% | 18 min | Extra regularization → slight drop. |
| **Model 17 – MobileNetV2 (frozen)** | 94.3% | 94.3% | 99.7% | 19 min | Transfer learning baseline. |
| **Model 18 – MobileNetV2 (fine-tuned)** | **96.7%** | 96.5% | 100% | 16 min | Best overall: accuracy & efficiency (~16 min). |
| **Model 19 – ResNet50 (frozen)** | 61.1% | 66.2% | 58.4% | 83 min | Severe underfitting, poor choice. |
| **Model 20 – ResNet50 (fine-tuned)** | 89.6% | 91.8% | 99.4% | 60 min | Better than frozen, but worse than MobileNet. |

## 5. Plots

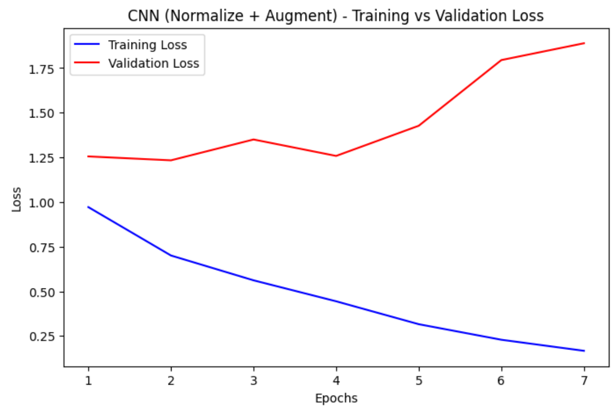
### 1. Loss curves



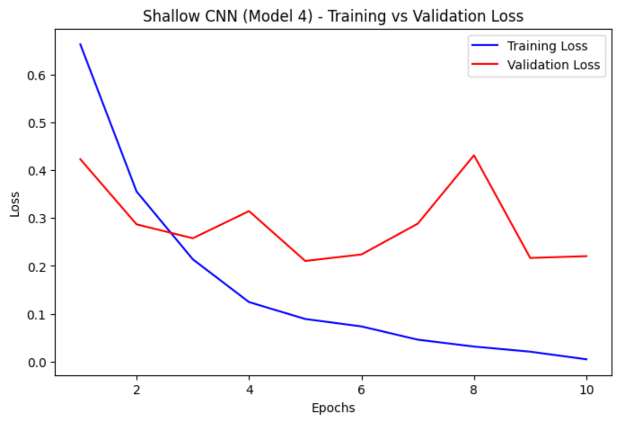
**Model 1 – CNN (baseline + early stopping)**



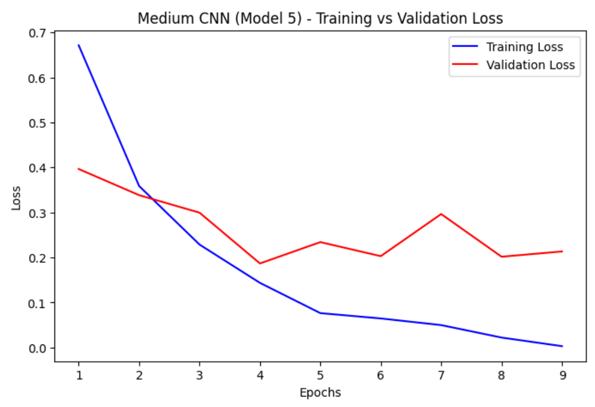
**Model 2 – CNN (Normalization)**



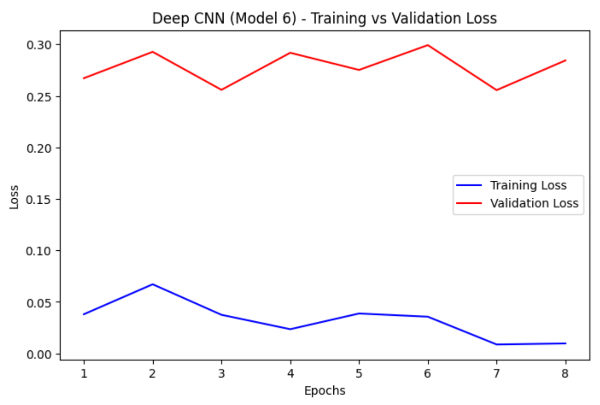
**Model 3 (light augmentation)**



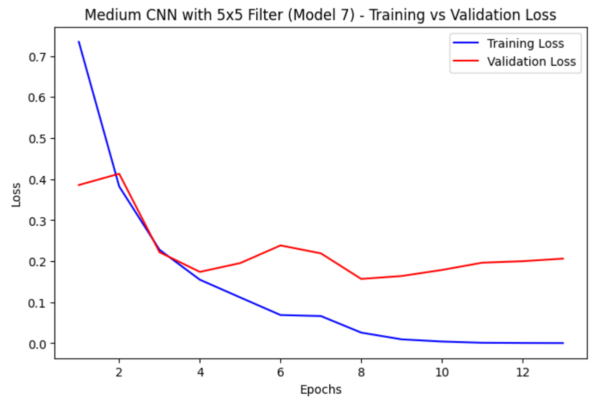
**Model 4 – Shallow CNN**



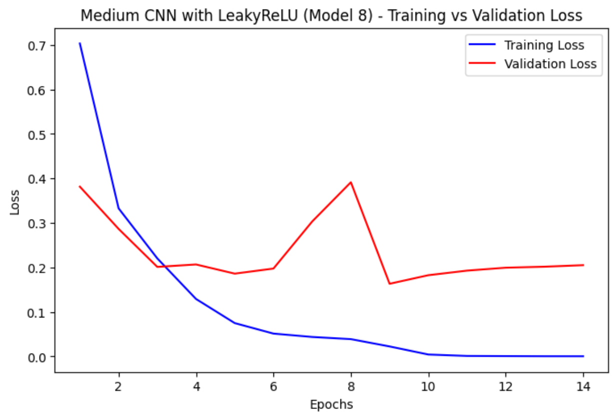
**Model 5 – Medium CNN**



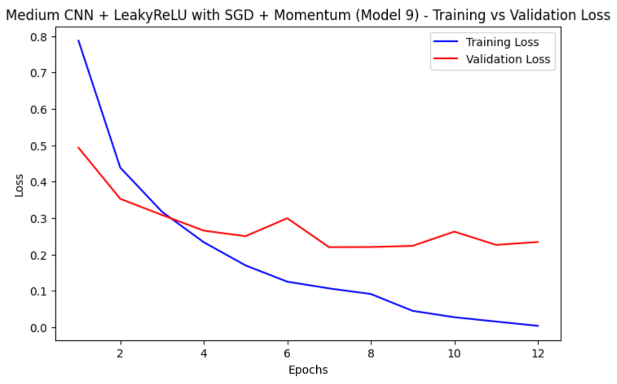
**Model 6 – Simplified Deep CNN**



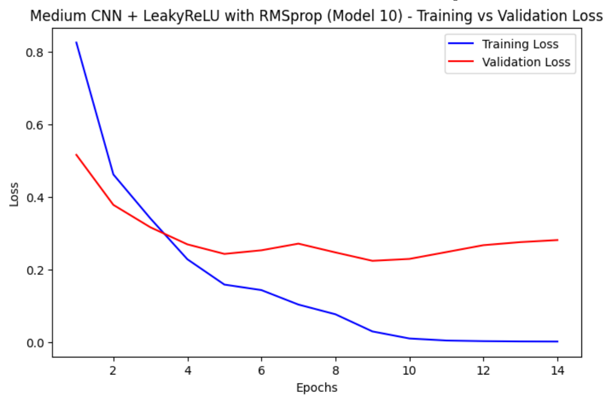
**Model 7 – Medium CNN + 5×5 filters**



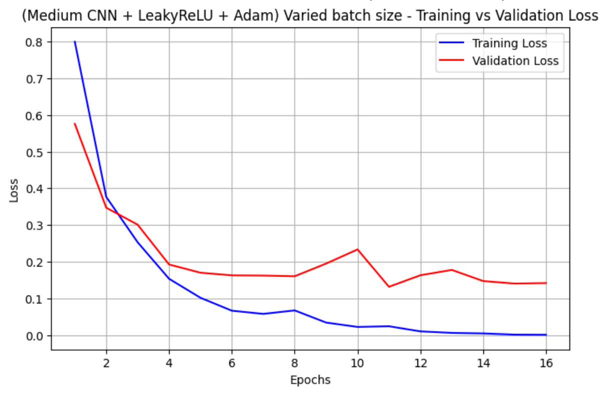
**Model 8 – Medium CNN + Leaky ReLU**



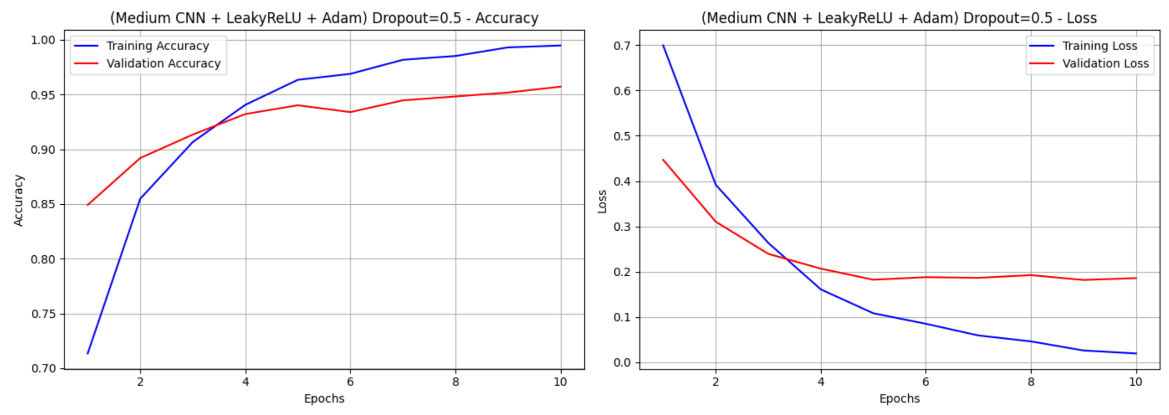
**Model 9 – CNN + SGD (momentum)**



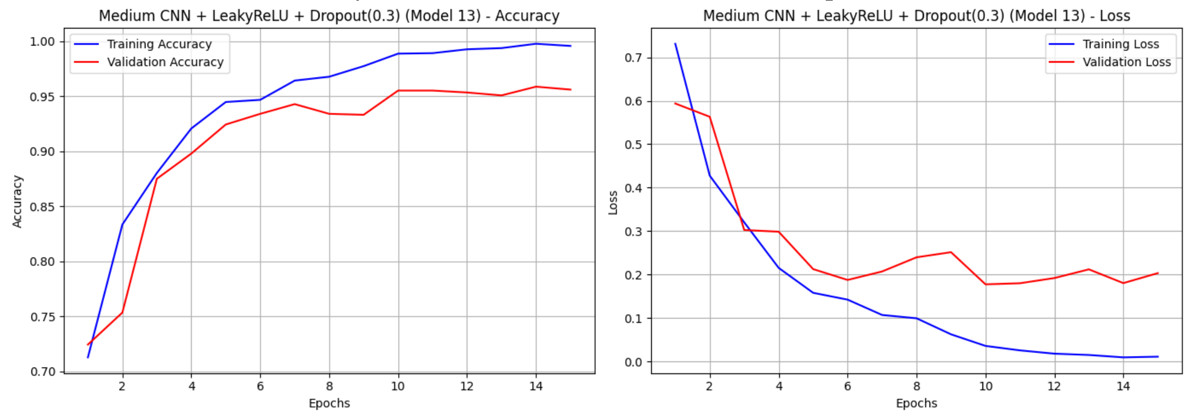
**Model 10 – CNN + RMSprop**



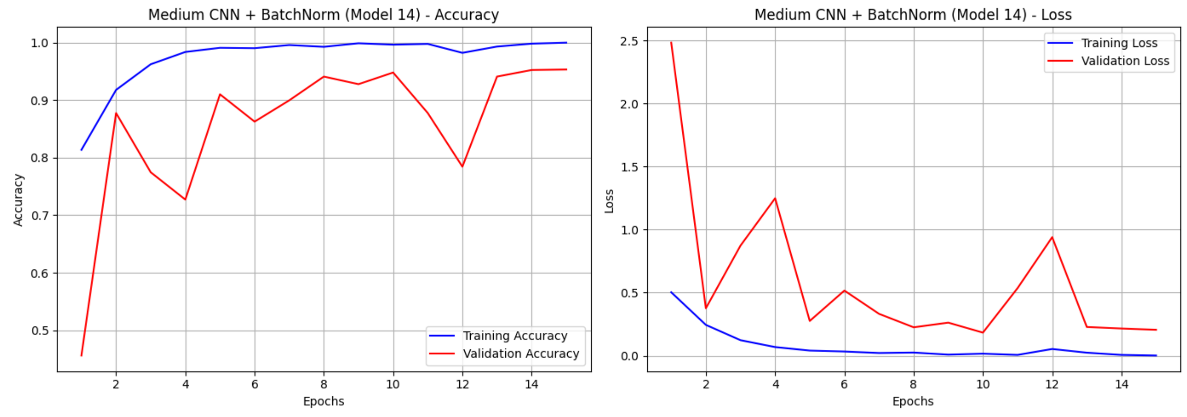
**Model 11 – CNN (batch size tuning)**



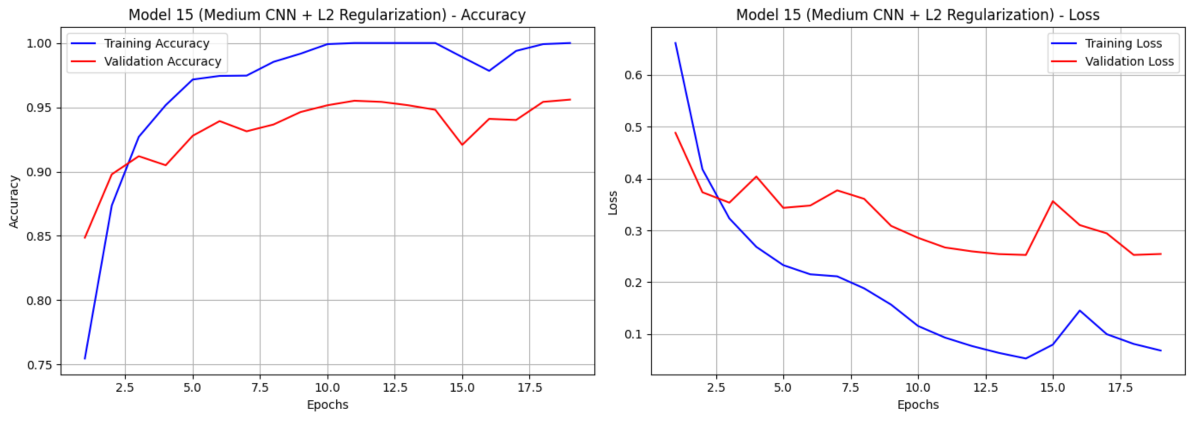
**Accuracy and loss curves: Model 12 – CNN + Dropout (0.5)**



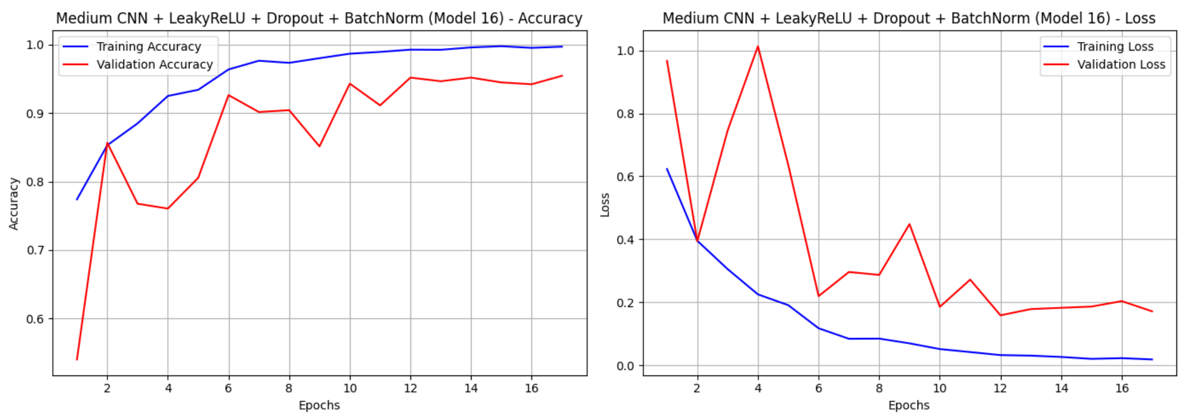
**Accuracy and loss curves: Model 13 – CNN + Dropout (0.3)**



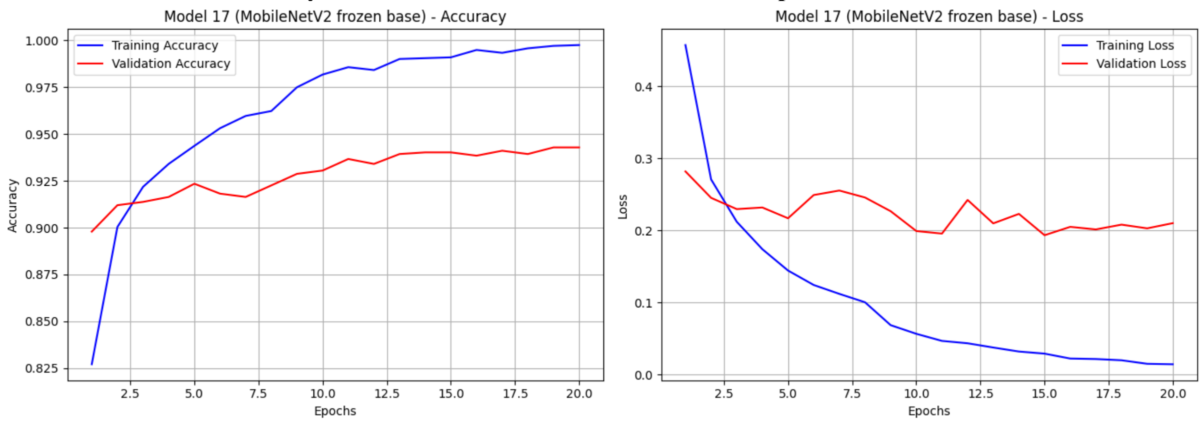
**Accuracy and loss curves: Model 14 – CNN + BatchNorm**



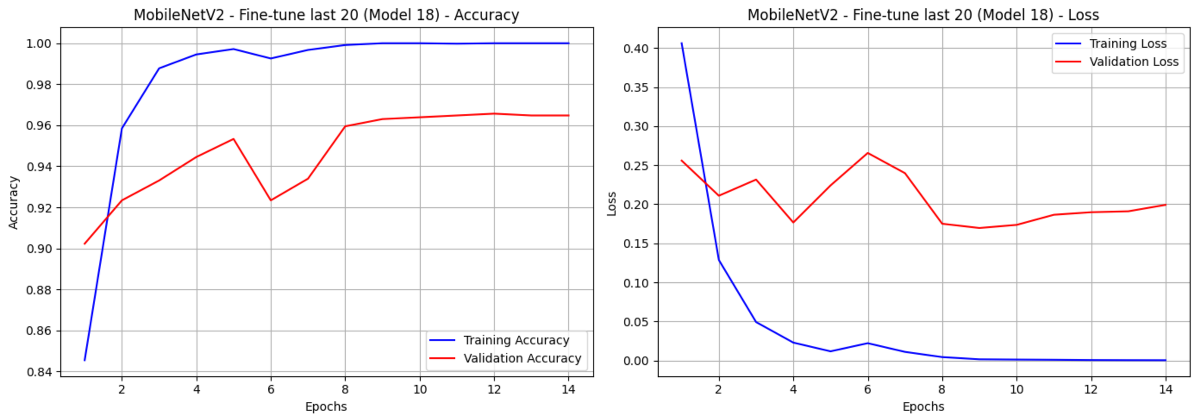
**Accuracy and loss curves: Model 15 – CNN + L2 Reg.**



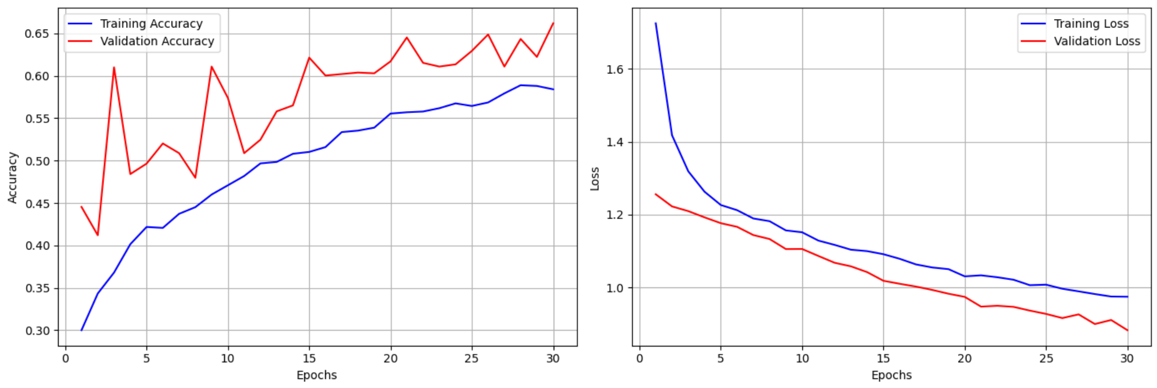
**Accuracy and loss curves: Model 16 – CNN + Dropout + BatchNorm**



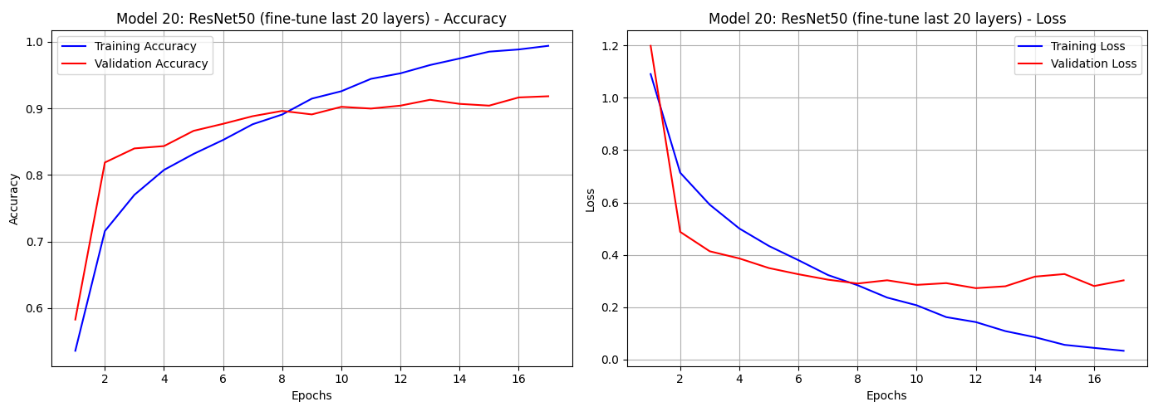
**Accuracy and loss curves: Model 17 – MobileNetV2 (frozen)**



**Accuracy and loss curves: Model 18 – MobileNetV2 (fine-tuned)**

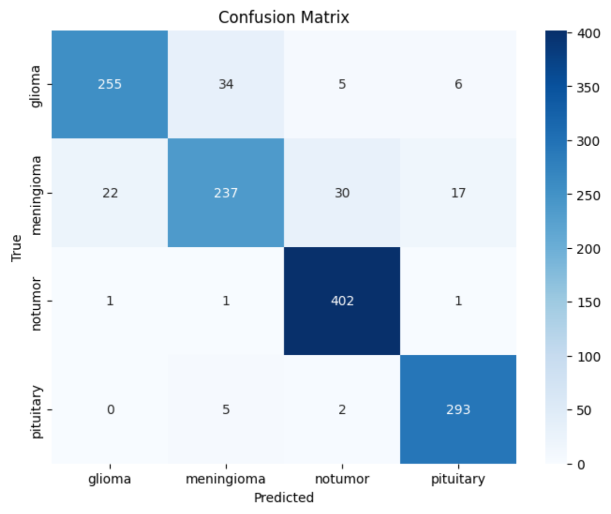


**Accuracy and loss curves: Model 19 – ResNet50 (frozen)**

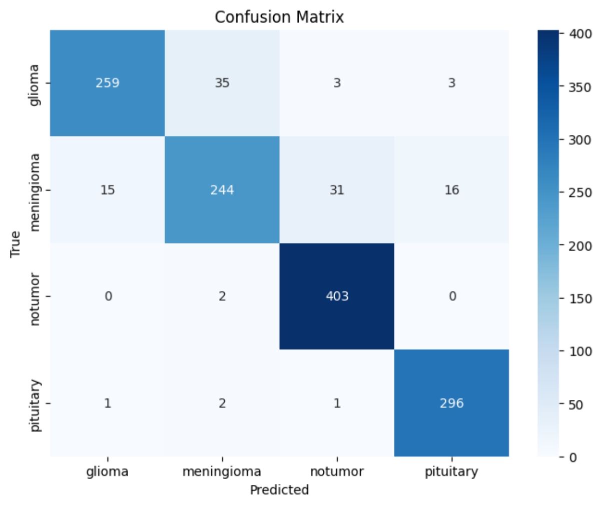


**Accuracy and loss curves: Model 20 – ResNet50 (fine-tuned)**

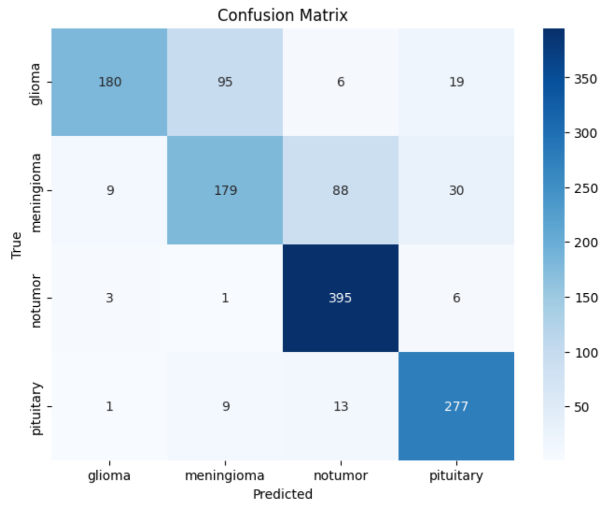
## 2. Confusion Matrices



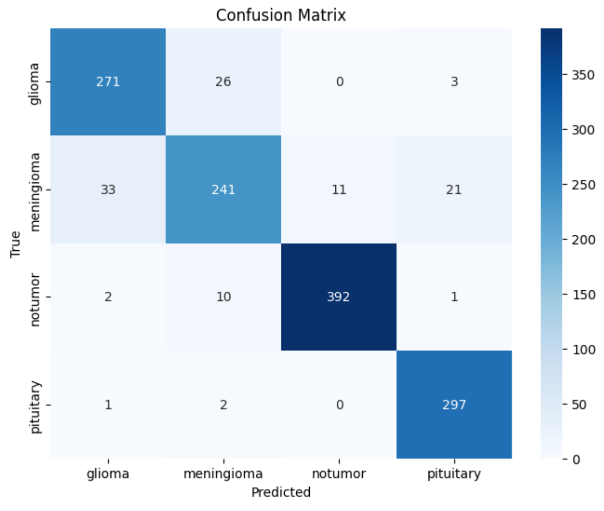
**Model 1 – CNN (baseline + early stopping)**



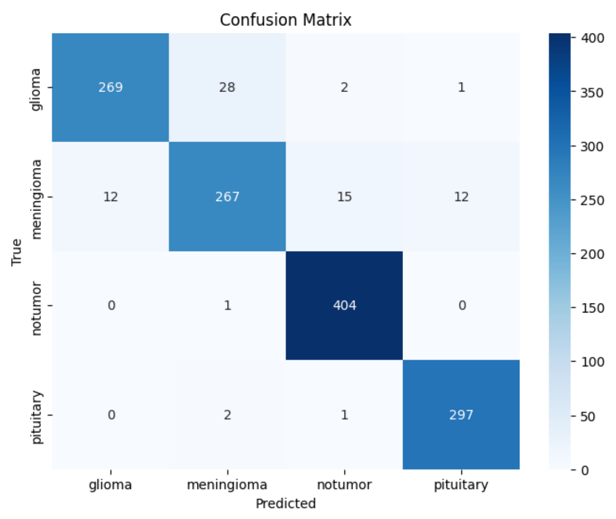
**Model 2 – CNN (improved, early stopping)**



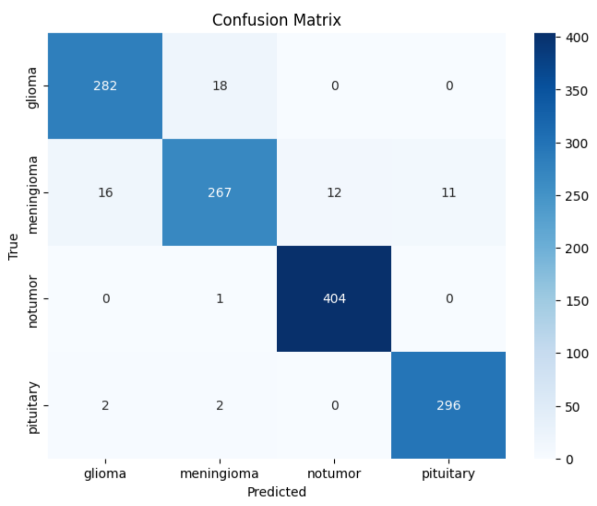
**Model 3 (light augmentation)**



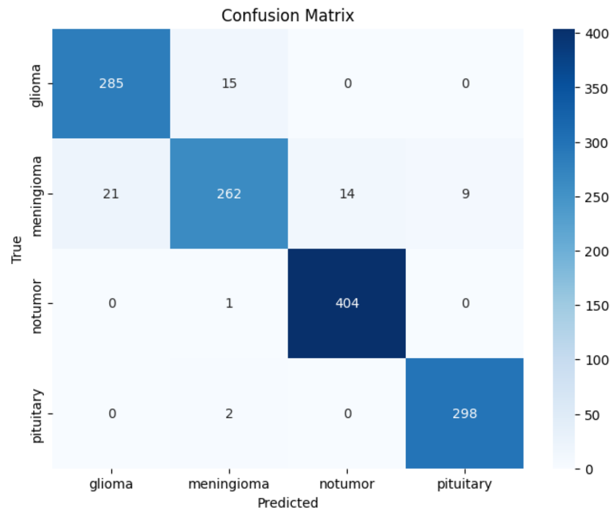
**Model 4 – Shallow CNN**



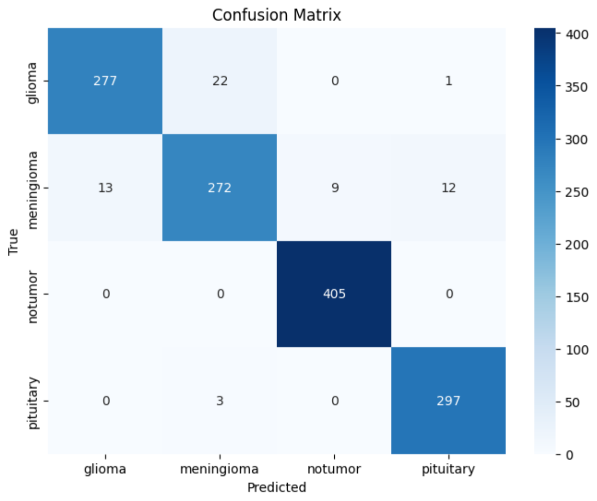
**Model 5 – Medium CNN**



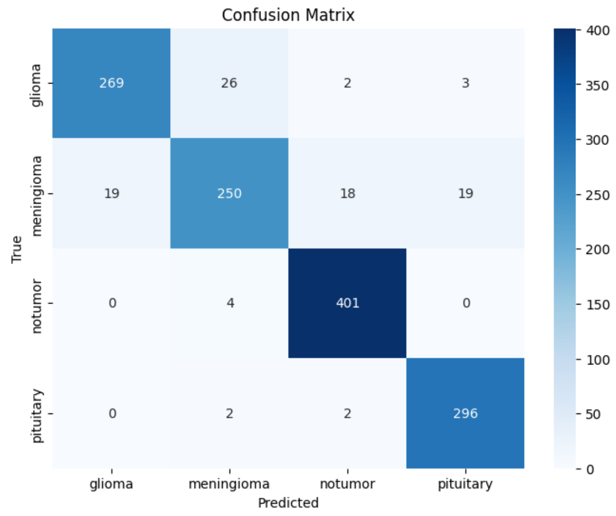
**Model 6 – Simplified Deep CNN**



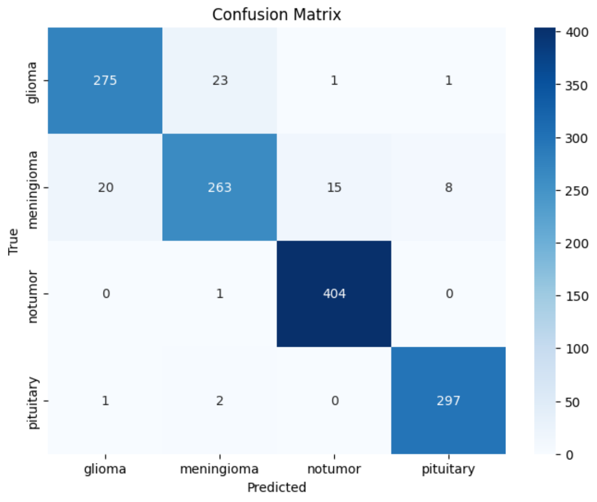
**Model 7 – Medium CNN + 5×5 filters**



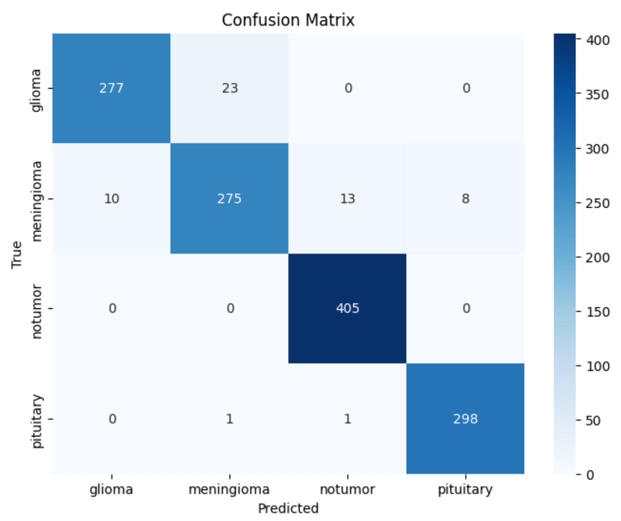
**Model 8 – Medium CNN + Leaky ReLU**



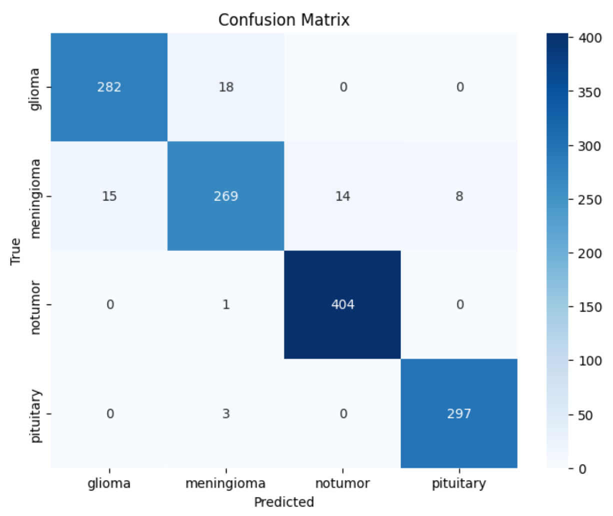
**Model 9 – CNN + SGD (momentum)**



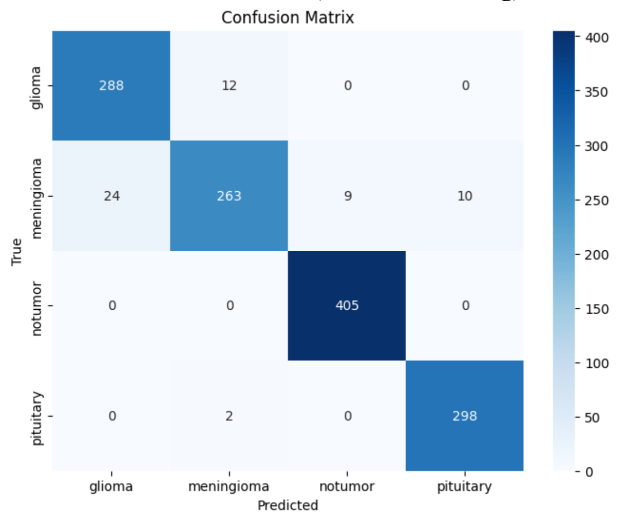
**Model 10 – CNN + RMSprop**



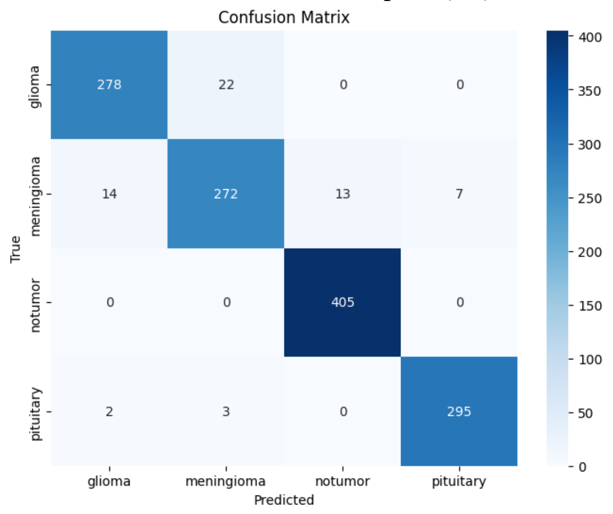
**Model 11 – CNN (batch size tuning)**



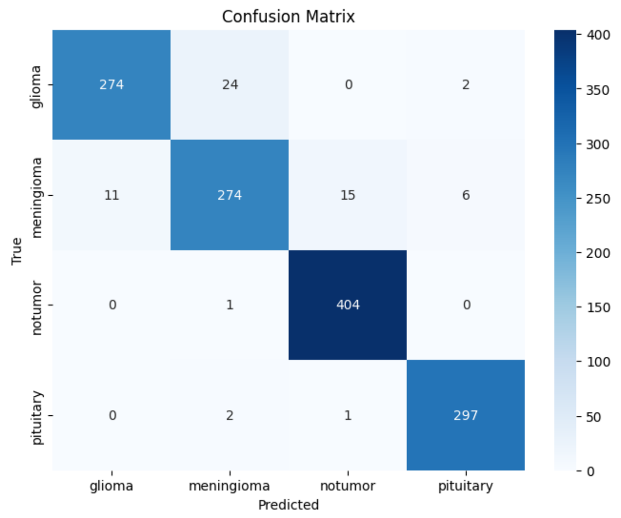
**Model 12 – CNN + Dropout (0.5)**



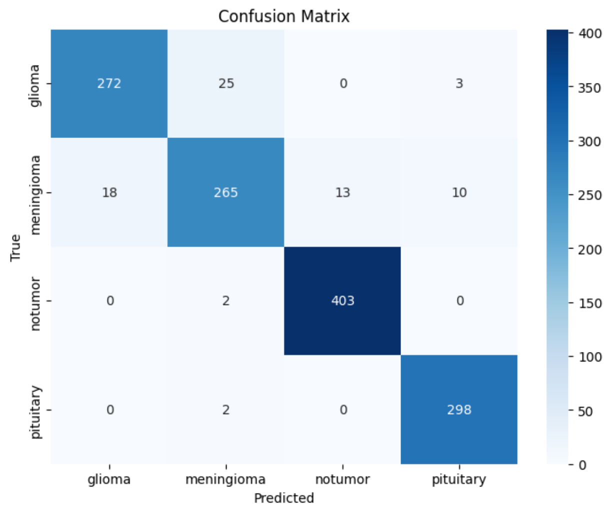
**Model 13 – CNN + Dropout (0.3)**



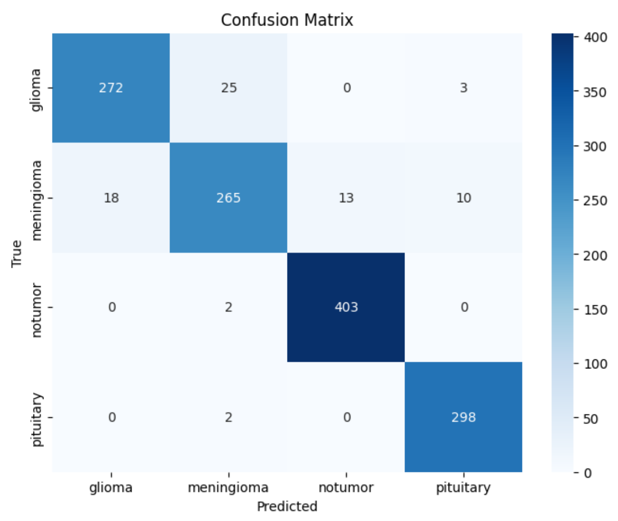
**Model 14 – CNN + BatchNorm**



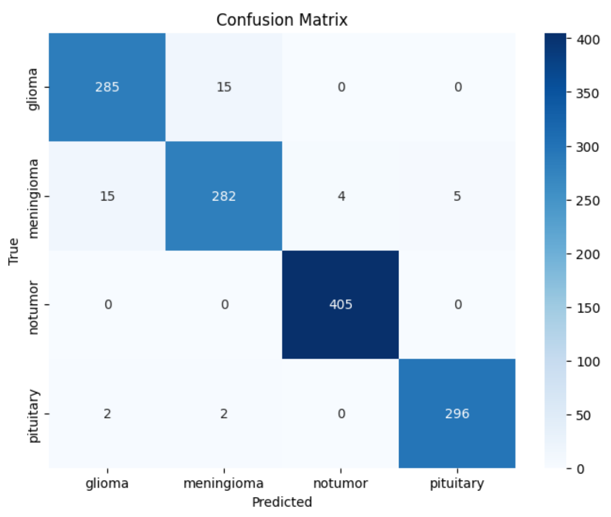
**Model 15 – CNN + L2 Reg.**



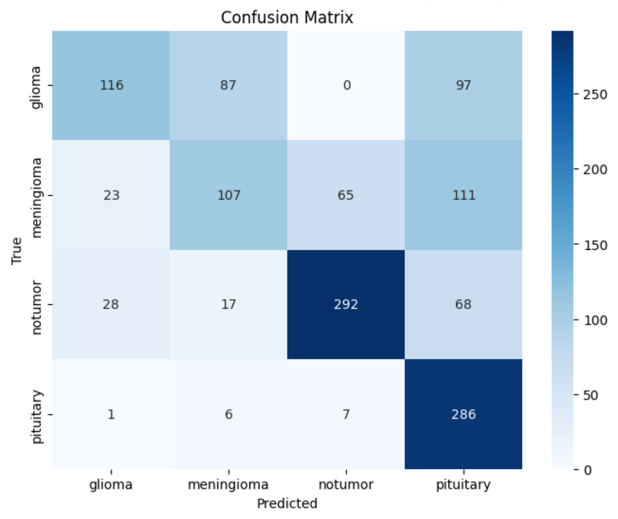
**Model 16 – CNN + Dropout + BatchNorm**



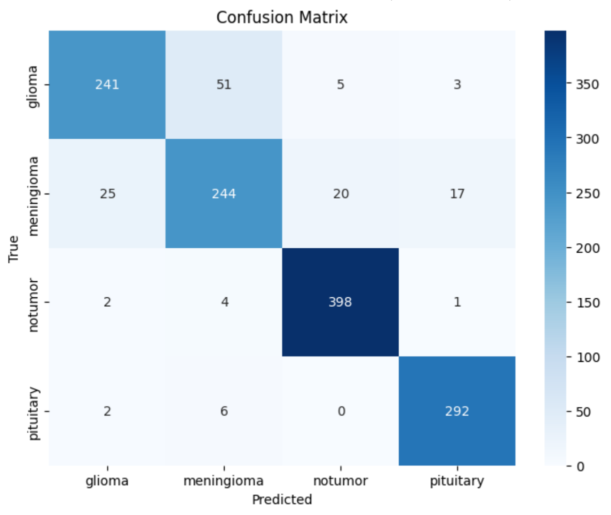
**Model 17 – MobileNetV2 (frozen)**



**Model 18 – MobileNetV2 (fine-tuned)**



**Model 19 – ResNet50 (frozen)**



**Model 20 – ResNet50 (fine-tuned)**

## 6. Conclusion

This project successfully explored a wide range of deep-learning techniques for brain tumor classification from MRI scans. The initial baseline CNN achieved a respectable 91.7% accuracy. Through careful architectural design, regularization, and hyperparameter tuning, a custom CNN was developed that achieved a 95.7% test accuracy.

The most significant finding was the superior performance of **transfer learning**. By fine-tuning a pre-trained MobileNetV2 model, a state-of-the-art test accuracy of **96.7%** was achieved. This model not only performed best but also trained faster than the deepest custom CNNs, making it the most efficient and effective solution.

## 7. Suggestions for Future Improvements:

a. **Advanced Data Augmentation:** Explore more sophisticated augmentation techniques tailored for medical images, such as elastic deformations or using generative models (e.g., GANs) to create synthetic data.

b. **Model Ensembling:** Combine predictions from the top-performing custom CNN (Model 11) and the fine-tuned MobileNetV2 (Model 18) to potentially achieve even higher and more robust accuracy.

c. **Explainable AI (XAI):** Implement Grad-CAM or similar techniques on the best model to visualize which regions of the MRI scan most influenced the decision, building trust and providing potential insights for medical professionals.

d. **Hyperparameter Optimization:** Use automated tools like Optuna or Bayesian Optimization to more thoroughly search the hyperparameter space for the top models.

## 8. Code & Implementation Reference

The code for this project, including all model architectures, training scripts, and data preprocessing pipelines, is available on GitHub.

**GitHub Repository Link:** https://github.com/mariamsherif04/Brain_Tumor_CNN_Models.git