IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

A Co-Evolutionary Genetic Algorithm Approach to Optimizing Deep Learning for Brain Tumor Classification

ABDELMGEID A. ALI[1], MOHAMED T. HAMMAD[1,2], AND HASSAN S. HASSAN[1]

[1]Department of Computer Science, Faculty of Computers and Information, Minia University, Minia 61519, Egypt
[2]Department of Computer Science, Faculty of Computers and Artificial Intelligence, University of Sadat City, Sadat City 32897, Egypt

Corresponding author: Mohamed T. Hammad (e-mail: mohamed.tony@fcai.usc.edu.eg).

ABSTRACT Brain tumors are among the deadliest diseases, leading researchers to focus on improving the accuracy of tumor classification—a critical task for prompt diagnosis and effective treatment. Recent advancements in brain tumor diagnosis have significantly increased the use of deep learning techniques, particularly pre-trained models, for classification tasks. These models serve as feature extractors or can be fine-tuned for specific tasks, reducing both training time and data requirements. However, achieving high accuracy in multi-class brain tumor classification remains a major challenge, driving continued research in this area. Key obstacles include the need for expert interpretation of deep learning model outputs and the difficulty of developing highly accurate categorization systems. Optimizing the hyperparameters of Convolutional Neural Network (CNN) architectures, especially those based on pre-trained models, plays a crucial role in improving training efficiency. Manual hyperparameter adjustment is time-consuming and often results in suboptimal outcomes. To address these challenges, we propose an advanced approach that combines transfer learning with enhanced coevolutionary algorithms. Specifically, we utilize EfficientNetB3 and DenseNet121 pre-trained models in conjunction with the Co-Evolutionary Genetic Algorithm (CEGA) to classify brain tumors into four categories: gliomas, meningiomas, pituitary adenomas, and no tumors. CEGA optimizes the hyperparameters, improving both convergence speed and accuracy. Experiments conducted on a Kaggle dataset demonstrate that CEGA-EfficientNetB3 achieved the highest accuracy of 99.39%, while CEGA-DenseNet121 attained 99.01%, both without data augmentation. These results outperform cutting-edge methods, offering a rapid and reliable method for brain tumor classification. This approach has great potential to support radiologists and physicians in making timely and accurate diagnoses.

INDEX TERMS Brain tumors, Convolutional Neural Network (CNN), Deep Learning (DL), EfficientNetB3, DenseNet121, Transfer Learning (TL), CEGA, Hyperparameter Optimization, MRI images, Pre-trained models.

I. INTRODUCTION

The brain, an intricate and fundamental organ, governs many essential activities and comprises billions of nerve cells, synapses, and neurons. Nevertheless, similar to other organs, the brain is susceptible to the formation of irregularities called brain tumors. Brain tumors arise from the abnormal proliferation of cells in various regions of the brain, which can lead to the death of healthy cells and potentially be life-threatening [1].

According to WHO reports, cancer was a major cause of global death in 2020, resulting in nearly 10 million fatalities, which equates to roughly one in six deaths [2]. Over 5,250 people pass away each year from brain cancer or cerebral tumors, according to research by cancer research organizations published in the United Kingdom [3].

Brain tumors are classified and distinguished according to their dimensions, morphology, and position within the brain. A tumor can be categorized into two categories: benign or malignant [4]. Primary malignancies originate in the brain, while secondary tumors usually originate from cancers located outside the brain, with the majority of them being malignant [5]. The three most common types of primary brain tumors are gliomas, meningiomas, and pituitary adenomas. Meningiomas are slow-growing neoplasms that originate from the meninges, which are the protective layers of membranes surrounding the brain and spinal cord. The major-

ity of meningioma tumors are benign [6]. Pituitary adenomas are neoplasms that develop in the pituitary gland [7].

Various advanced Magnetic Resonance Imaging (MRI) visualization methodologies can be employed to categorize and detect brain tumors. MRI is the predominant non-invasive method that is widely used. It is frequently employed in medicine because it provides a range of high-resolution images of soft tissues by utilizing contrast-enhancing substances, all without exposing patients to ionizing radiation [8].

Rapid detection of tumors from MRI scans and automated categorization of tumor types are crucial for clinicians when strategizing treatment therapies [9]. Deep learning (DL) and transfer learning (TL) techniques can be utilized to enhance this rapid detection and classification process.

Deep learning models are trained from scratch through supervised learning to optimize data representation and achieve robust generalization on unseen test data. With progress in hardware systems, DL has become the preferred choice for intelligent systems in classification, detection, segmentation, and other tasks related to computer vision, making it a crucial component in the development of intelligent systems. DL achievements rely on large-scale training data, often requiring millions of samples due to data quality, expense, and time constraints in real-world scenarios [10]. Data scarcity poses significant obstacles to deep learning technology, and the pre-training method serves as a key technique to address this issue. The model acquires efficient representations from source datasets and transfers the acquired knowledge to target domains, hence expediting the learning process for tasks. Pre-training reduces the reliance of the model on extensive training data and enhances its capacity to perform well on smaller datasets with improved generalization [11].

TL is a commonly employed method that entails pre-training a model on a substantial dataset and then refining it on a smaller dataset to attain the best possible performance. By utilizing pre-trained parameters from other source data, TL creates a strong base model that helps prevent overfitting, accelerates training, and decreases the amount of data needed for retraining. This method accelerates the training process and improves accuracy by utilizing the weights of a pre-trained network, leading to a reduction in training time for the new task [12].

In transfer learning with CNNs for classification tasks, understanding hyperparameter tuning remains important, especially for the layers that are fine-tuned or newly added, as each layer may have distinct hyperparameters. Furthermore, few studies provide guidelines for hyperparameter adjustment across different models under specific conditions. Selecting hyperparameter values often relies on a combination of human expertise, experimentation, and grid search techniques [13]. Training CNNs is time-consuming due to their computational demands and the exponential increase in hyperparameter combinations. Grid search and manual optimization are impractical for CNNs, especially as their complexity grows. Researchers have explored various techniques for hyperparameter tuning with varying levels of success [14]. As a result, automating the optimization of CNN hyperparameters plays a vital role [15].

Existing research has demonstrated the effectiveness of CNNs in tumor classification. However, several challenges persist, including hyperparameter optimization, reliance on large-scale datasets, and achieving high accuracy without data augmentation. Additionally, many existing approaches lack comprehensive methods for accurately isolating brain regions from MRI scans, which can lead to classification ambiguities. To address these limitations, we introduce an enhanced genetic algorithm, the Co-Evolutionary Genetic Algorithm (CEGA). In this method, two genetic algorithms evolve within separate populations to optimize the hyperparameters of a CNN architecture tailored for brain tumor classification. The proposed methods, CEGA-EfficientNetB3 and CEGA-DenseNet121, leverage pre-trained CNN models—EfficientNetB3 and DenseNet121—to classify brain tumors from a Kaggle dataset through transfer learning (TL).

The critical research contributions of this study are outlined as follows:

- A new and reliable approach has been developed to automatically identify different types of brain tumors by effectively extracting features from MRI images. In this approach, the Co-evolutionary Genetic Algorithm (CEGA) is applied to optimize the hyperparameters of EfficientNetB3 and DenseNet121, improving both convergence speed and accuracy.
- We utilized Gaussian blur to reduce noise and smooth edges, along with Otsu's thresholding method to isolate the brain region from the background, effectively removing extraneous parts of the image to minimize training ambiguities and improve classification performance.
- Our models achieve high classification accuracy without using data augmentation, demonstrating their robustness and generalization capabilities.
- The fine-tuned CEGA-EfficientNetB3 model we propose demonstrates superior performance, surpassing the CEGA-DenseNet121 model and other methods reviewed in previous studies in model accuracy, precision, recall (sensitivity), specificity, and F1-score.
- In contrast to earlier studies that used the same or similar datasets, we achieved high classification accuracy without relying on data augmentation techniques.

The structure of the paper is outlined as follows: Section II presents a review of the relevant literature in this field. Section III outlines the methodologies that form the foundation of the proposed CEGA-EfficientNetB3 and CEGA-DenseNet121 models, including the core principles of the original CEGA, such as the limitations of the Genetic Algorithm and traditional optimization algorithms. It also provides an overview of the CEGA algorithm, along with a detailed explanation of its adaptation for hyperparameter optimization and the use of transfer learning. A comprehensive explanation of the proposed methodology is provided in Section IV. Section V covers the system and software requirements,

dataset, and hyperparameter settings. Section VI presents the experimental evaluation, covering CEGA's hyperparameter optimization, model training, performance evaluation for brain tumor classification, and a discussion of accuracy, loss curves, confusion matrix analysis, and AUC-ROC curve analysis. Section VII discusses the comparison with cutting-edge methods. Section VIII outlines the limitations of the CEGA algorithm and proposed models, highlighting challenges and suggesting future improvements in efficiency, scalability, and generalization. In conclusion, Section IX summarizes the key findings and outlines potential directions for future research.

II. RELATED WORK

This section reviews studies focused on classifying brain tumors in MRI scans, as shown in Table 1. Several machine and deep learning approaches have gained popularity for automating tumor detection, offering promising solutions with high diagnostic accuracy. Table 1 provides a comparative overview of various recent classification methods (binary and multiclass), datasets, techniques, and their achieved accuracies. It highlights the dominance of deep learning approaches, particularly CNN-based models and transfer learning architectures such as EfficientNet, ResNet, and DenseNet, with or without hyperparameter optimization. These methods have consistently demonstrated remarkable performance, often achieving accuracies exceeding 95%.

Recently, significant research has focused on the detection and classification of brain tumors to help save lives. We have analyzed several state-of-the-art methods that utilize both traditional machine learning and deep learning approaches. In [16], a machine learning-based method called "Extended Kalman Filter with Support Vector Machine (EKF-SVM)" was introduced to determine the existence of a tumor, automatically find tumors in medical images, outline their shape, and determine whether they are cancerous or non-cancerous. This method achieved an accuracy of 96.05%.

Ismael et al. [17] suggested an improved method for categorizing different forms of brain tumors using "Residual Networks (ResNet50)", achieving 99% accuracy. In addition, in [18], the authors presented a cutting-edge CNN model that incorporates hypercolumns, recursive feature elimination, support vector machines (SVMs), and pre-trained VGG-16 and AlexNet networks. The study attained a 96.77% binary classification accuracy.

The researchers in [19] proposed a hybrid deep CNN model that integrates U-Net as the core structure with ResNeXt-50. In this model, ResNext-50 serves as the encoding component for U-Net. This technique achieved a 99.7% classification accuracy. Furthermore, in [20], an approach for detecting brain tumors in their early stages was introduced, using "multi-level feature extraction and concatenation" with two pre-existing models: Inception-v3 and DenseNet201. Features extracted from Inception-v3 are classified using a softmax classifier, while features from DenseNet201 are also utilized. This method achieved testing accuracies of 99.34% and 99.51% for Inception-v3 and DenseNet201, respectively. In addition, in [21], the authors proposed a framework that utilizes three architectures (VGGNet, GoogLeNet, and AlexNet) for classifying brain malignancies, specifically gliomas, meningiomas, and pituitary adenomas. They reached a peak test accuracy of 98.69% among all the studies by using the fine-tuned VGG16 network.

In [22], the authors implemented a transfer learning approach to categorize various types of brain tumors, such as gliomas, meningiomas, and pituitary tumors from MR scans using DenseNet121, ResNet50, VGG16, and VGG19 architectures. These methods extract features with transfer learning, leveraging weights from previously trained networks, and use a dense neural network with three layers for classification. The highest classification accuracy achieved by the ResNet50 model was 99.02% when trained with the Adadelta optimizer. In [8], the enhanced ResNet50 model was employed to introduce a new deep-learning methodology utilizing transfer learning to assess the accuracy of brain cancer categorization. The brain tumor detection method utilizes four core strategies: data preprocessing, transfer learning techniques with the ResNet50 model, extracting features, and improvement. In the same context, in [23], the researchers introduced a CNN architecture for detecting pituitary, glioma, and meningioma brain tumors, aiming for high classification accuracy and efficiency. Their approach included selecting an appropriate dataset and applying processing in a three-step method to improve the quality of images. The model attained a total accuracy of 97.72%.

In [24], the authors implemented a framework to extract features and classify brain tumors using MRI. Initially, a CNN model is integrated to perform deep feature extraction from MRI images. The extracted features are subsequently input into a "Long Short-Term Memory (LSTM)" model for final classification. The LSTM model achieved accuracies of 95.93%, 97.80%, and 96.78% with the BMIDS, MBNDS, and BTDS augmented datasets, respectively. In addition, in [25], the researchers proposed a multiclass classification model called IVX16, which leverages TL by integrating the three top-performing TL models. They utilized a dataset containing 3,264 images in total. Through extensive experiments, they achieved peak accuracies of 94.5%, 93.88%, 95.11%, 93.88%,96.94%, 94.19%, and 93.58% for Xception, InceptionV3, VGG16, ResNet50, IVX16, VGG19, and Inception-ResNetV2, respectively.

In [26], a novel EBT Deep Net DL model was introduced, which employs an advanced CNN technique for classifying three different types of brain cancers, including gliomas, meningeal tissue tumors, and tumors of the pituitary gland. The model utilizes pre-trained deep CNN architectures, including DenseNet201, MobileNetV3, VGG19, MobileNetV3, and ResNet152 for deep feature extraction. The accuracy of each model is as follows: MobileNetV3 achieved 97.15%, DenseNet201 achieved 95.44%, VGG19 achieved 97.46%, ResNet50V2 achieved 90.04%, and ResNet152 achieved 98%. Several methods have been developed to optimize the hyperparameters of machine learn-

ing and deep learning models. Kurdi et al. [27] introduced "the Harris Hawks Optimized Convolutional Neural Network (HHOCNN)," leveraging the Harris Hawks Optimization algorithm to minimize misclassification errors in tumor recognition. This approach addresses limitations in existing systems, including challenges in precise tumor localization, edge detail capture, and computational efficiency, achieving 98% accuracy on a Kaggle dataset. Celik et al. [1] introduced an innovative hybrid approach for accurate multiclass classification of brain tumors, combining a novel CNN model for feature extraction with machine learning (ML) algorithms for classification. Additionally, the study evaluates the performance of advanced hybrid models, including DenseNet201-SVM and EfficientNet0-SVM. Bayesian optimization is employed to fine-tune the hyperparameters of the ML algorithms, ensuring optimal performance. The results show that the mean accuracies for DenseNet201-SVM and EfficientNetB0-SVM are 96.87% and 97.01%, respectively.

In the same context, in [28], a novel approach to brain tumor detection is presented, employing a deep convolutional neural network (DCNN) based on EfficientNet-B4. By incorporating customized layers and optimizing hyperparameters with Bayesian Optimization, the resulting model achieved 99.33% accuracy on the "Brain Tumor Detection 2020 Kaggle dataset". In addition, the authors in [29] introduced a new CNN model for brain tumor diagnosis, with careful hyperparameter optimization to refine its performance. Using a grid search approach, the model was evaluated on a "publicly available brain tumor MRI dataset from Kaggle" achieving 96% accuracy. Additionally, the researchers in [30] presented an enhanced machine learning model for brain tumor classification, achieving a high testing accuracy of 98.21%. The model, designed to assist medical specialists, combines deep learning architectures (Inception V3 and DenseNet201) for feature extraction with radiomic features and uses a PSO-KELM classifier to distinguish between No Tumor, Gliomas, Meningiomas, and Pituitary Tumors.

Some traditional models overlook the critical role of hyperparameters or rely on manual hyperparameter selection in classification tasks, as discussed in this literature review. While recent models incorporate hyperparameters with transfer learning, their performance remains suboptimal. Additionally, the trial-and-error approach to hyperparameter tuning is often laborious and prone to errors, highlighting the advantages of employing metaheuristic algorithms for optimization. Our work advances these approaches by leveraging the Co-Evolutionary Genetic Algorithm (CEGA) for dynamic hyperparameter optimization, optimizing deep learning architectures like EfficientNetB3 and DenseNet121. This enables improved classification accuracy and robustness across diverse datasets, addressing limitations in traditional optimization techniques.

III. PRELIMINARIES

This section outlines the core methodologies employed in the suggested model, including the design of the co-evolutionary genetic algorithm, the key role of the CNN architecture, and the application of transfer learning.

A. CEGA: CO-EVOLUTIONARY GENETIC ALGORITHM

This subsection discusses the adaptation of the Co-Evolutionary Genetic Algorithm (CEGA) for hyperparameter optimization. It explains how the CEGA framework optimizes the balance between exploration and exploitation, accelerating the search for optimal hyperparameters while avoiding stagnation in local optima. The subsection also explores strategies for achieving rapid convergence during the tuning process. Furthermore, it begins by addressing the limitations of the original Genetic Algorithm (GA) and other traditional optimization algorithms in hyperparameter optimization.

1) Original Genetic Algorithm Limitations

Genetic Algorithms (GAs) [31], though robust and widely used for optimization, face several limitations that can hinder their effectiveness, particularly in complex tasks like hyperparameter optimization. A notable challenge is premature convergence, where GAs often get trapped in local optima, especially in multimodal or high-dimensional search spaces. This issue is compounded by their difficulty in balancing exploration and exploitation, which can result in either slow convergence or reduced solution quality. Additionally, GAs require significant computational resources, as evaluating the fitness of large populations across multiple generations is resource-intensive, particularly for problems with expensive objective functions.

These limitations become even more pronounced in hyperparameter optimization, which involves navigating two distinct search spaces—solutions (Ω) and scenarios (Ψ)—and addressing conflicting objectives of minimization and maximization. Traditional GAs struggle to effectively manage this complexity. The CEGA, however, provides a promising solution. By employing co-evolutionary strategies, CEGA can simultaneously explore multiple search spaces, address conflicting objectives, and dynamically adapt to the demands of the optimization process. This enhances its efficiency, enabling it to overcome the limitations of traditional GAs and perform more effectively in solving complex hyperparameter optimization challenges.

2) Limitations of Traditional Optimization Algorithms

Traditional optimization algorithms, such as Evolutionary Strategies (ES) [32], Bayesian Optimization (BO) [33], and Particle Swarm Optimization (PSO) [34], have demonstrated their effectiveness across various problem domains. However, these methods face notable challenges that limit their performance in certain complex optimization tasks.

PSO, though known for its simplicity and convergence speed, tends to lose diversity quickly and risks stagnation in dynamic or complex landscapes. BO is a powerful method for hyperparameter tuning, utilizing probabilistic models like

TABLE 1: Related Work in brain tumor classification.

Reference	Year	Classification Problem	Technique	Datasets used	Accuracy (%)
[16]	2021	Binary Classification	EKF-SVM	Tiantan Hospital dataset	96.05
[17]	2020	Multi Classification	ResNet50	Figshare	99
[18]	2020	Binary Classification	CNN-SVM	Kaggle	96.77
[19]	2021	Binary Classification	UnetResNext-50 based CNN	TCGA (The Cancer Genome Atlas)	99.7
[20]	2020	Multi Classification	(Inception-v3 and DensNet201) based CNN softmax classifier	Figshare	99.51
[21]	2020	Multi Classification	(AlexNet, GoogLeNet, and VGGNet) based CNN	Figshare	98.69
[22]	2021	Multi Classification	VGG16 VGG19 ResNet50 DenseNet121	Figshare	97.49 97.93 99.02 98.91
[8]	2023	Multi Classification	Modified ResNet50 based CNN	Kaggle	92
[23]	2022	Multi Classification	DCNN	Special	97.72
[24]	2022	Binary Classification	CNN-LSTM	BTDS MBNDS BMIDS	96.78 97.80 95.93
[25]	2024	Multi Classification	IVX16	Kaggle	96.94
[27]	2023	Binary Classification	HHOCNN	Kaggle	98
[17]	2024	Multi-Classification	VGG19 ResNet152 DenseNet201 ResNet50V2 MobileNetV3	Figshare (Multicancer Dataset)	97.46 98 95.44 90.04 97.15
[28]	2024	Binary Classification	EfficientNet-B4	Kaggle	99.33
[1]	2024	Multi-Classification	DenseNet201-SVM EfficientNet0-SVM	"Figshare, BrH35, and SARTAJ" combination	96.87 97.01
[29]	2024	Multi-Classification	HPCNN	"Figshare, BrH35, and SARTAJ" combination	96
[30]	2024	Multi-Classification	DenseNet201-PSO-KELM	Kaggle	98.21

Gaussian Processes (GPs) to conduct efficient searches, especially when evaluating the objective function, which is costly and limited. However, it faces challenges in complex tasks such as brain tumor classification. These challenges include scalability issues in high-dimensional spaces (leading to increased computational costs), the assumption of a smooth objective function (which may not hold for deep learning models with non-convex loss landscapes), difficulty adapting to changing search spaces, and potential suboptimal balances between exploration and exploitation in multimodal spaces. Additionally, BO can become computationally expensive with large datasets or many parallel evaluations.

Moreover, all these methods share a common reliance on static or manually tuned parameters, such as mutation rates, population sizes, or inertia weights. This lack of adaptivity reduces their efficiency, as these parameters are not well-suited for varying optimization phases. High computational costs and sensitivity to initial parameter settings further exacerbate these issues, making these traditional algorithms less effective for problems requiring dynamic adaptability and efficient exploration of large, complex search spaces.

These limitations highlight the need for more advanced optimization techniques, such as Co-Evolutionary Genetic Algorithms (CEGA), that can dynamically adapt to the demands of complex optimization tasks while addressing the shortcomings of traditional approaches. CEGA's adaptive co-evolutionary mechanisms make it well-suited for overcoming the challenges of high-dimensional, multimodal, and dynamic search spaces, ensuring robust and efficient hyperparameter optimization for complex models like EfficientNetB3 and DenseNet121.

B. OVERVIEW OF THE CEGA ALGORITHM

The Co-Evolutionary Genetic Algorithm (CEGA) [35] is an advanced form of the genetic algorithm [31] designed to optimize complex problems by simultaneously evolving multiple populations or search spaces. Unlike traditional GAs that focus on a single population, CEGA employs co-evolutionary strategies, allowing it to explore both solutions (Ω) and scenarios (Ψ) in tandem. This approach enables CEGA to han-

dle conflicting objectives, such as minimization and maximization, more effectively. By facilitating interaction between evolving populations, CEGA dynamically adapts to the changing demands of optimization tasks, making it particularly well-suited for high-dimensional, multimodal, and dynamic search spaces. These characteristics make CEGA a powerful tool for complex optimization challenges, such as hyperparameter optimization in machine learning models.

In this approach, two distinct GAs evolve independently within separate populations, with each population representing a different state space. The assessment of one population is performed while the other algorithm's population remains "frozen." Each individual's fitness in one population is affected by the entire group of individuals in the other population. During each generation, the populations share information to aid in convergence.

In the framework of brain tumor classification with hyperparameter optimization, the fitness function evaluates how well a particular hyperparameter configuration (individual X in state space Ω) performs. The fitness function for an individual X is defined as:

$$f_\Omega(X) = \max_{\psi \in \Psi} F(X, \psi) \tag{1}$$

This evaluates X by finding the best performance when paired with an individual ψ from the other state space Ψ. Similarly, the fitness function for an individual ψ in Ψ is:

$$f_\Psi(\psi) = \min_{X \in \Omega} F(X, \psi) \tag{2}$$

Here, $F(X, \psi)$ represents a weighted combination of multiple objectives related to classification and model performance, defined as:

$$\begin{aligned} F(X, \psi) = {} & \lambda_1 J_{\text{latency}}(X) + \lambda_2 J_{\text{c2c latency}}(X) \\ & + \lambda_3 I_{\text{imb}}(X, \psi) + \lambda_4 P_{\text{failure}}(X, \psi) \\ & + \gamma \sum_{\nu \in \Gamma} \max\{0, |\nu|^2\} \end{aligned} \tag{3}$$

Where:

- $J_{\text{latency}}(X)$ and $J_{\text{c2c latency}}(X)$ are latency terms related to the model's computational efficiency.
- $I_{\text{imb}}(X, \psi)$ measures the class imbalance impact during training.
- $P_{\text{failure}}(X, \psi)$ accounts for the probability of misclassification.
- Γ represents the set of constraints, while γ represents the penalty coefficient for violations of these constraints.

This multi-objective fitness function helps to optimize both classification accuracy and computational efficiency while ensuring that constraints are respected during hyperparameter optimization.

The pseudocode for the proposed co-evolutionary genetic algorithm (CEGA) [35] designed for hyperparameter optimization is presented in Algorithm 1. As explained, two random populations, popA and popB, are initialized during the first step. In this setup, popA represents possible hyperparameter configurations, including parameters like learning rate and optimizer choice, while popB represents model architectures, including the dropout layer and the number of neurons. Next, the algorithm proceeds to a loop (Step 5), where the populations evolve using two distinct GAs: GA_Ω (Step 6) hyperparameter optimization of the learning rate and optimizer, and GA_Ψ (Step 10) for architecture optimization of the dropout layer and number of neurons. Each genetic algorithm runs for a user-defined number of generations, n_g. After each cycle of n_g generations, each GA provides an updated population that influences the evolution of the other. This process continues until the stopping condition is met, which occurs when the difference in performance between the best solutions of the two populations becomes smaller than a given threshold ϵ.

Although both GAs utilize standard genetic operators such as selection, crossover, and mutation, GA_Ω has been specifically optimized to handle hyperparameter constraints more efficiently. For example, when evolving hyperparameters such as learning rate, dropout rate, number of neurons, and optimizers, applying genetic operations can lead to configurations that explore diverse regions of the search space, potentially uncovering more effective solutions for model training.

C. DETAILED OF THE CEGA ALGORITHM

This section offers an in-depth overview of the Co-Evolutionary Genetic Algorithm (CEGA), presenting its step-by-step implementation. The outlined steps emphasize CEGA's innovative strategies for tackling complex optimization challenges.

1) Initialization

Two populations of agents are initialized, one representing architecture parameters and the other representing optimization parameters. Each agent is assigned initial values sampled randomly from a predefined hyperparameter search space, as illustrated in Table 3:

- **Architecture Agents**: These agents are initialized with random values for architecture parameters, such as the number of units in each dense layer and dropout rates. The number of units is selected from [128, 256, 512], while dropout rates are sampled from [0.2, 0.3, 0.4, 0.5].
- **Optimization Agents**: These agents are initialized with random values for optimization parameters, such as the learning rate and optimizer type. Learning rates are sampled from $[10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}]$, and optimizer types are selected from options such as SGD, Adam, RMSprop, Adadelta, Adamax, Nadam, or AdamW.

Each agent stores its architecture parameters and optimization parameters, along with a fitness score that will be updated during the evolutionary process.

Algorithm 1 CEGA - Co-Evolutionary Genetic Algorithm for Hyperparameter Optimization

Require:
M: Machine Learning model
K: Number of hyperparameters
$[h_{\min}, h_{\max}]$: Hyperparameter ranges
n_A, n_B: Population sizes
n_g: Number of generations
D_{val}: Validation dataset
popA : Population size for optimization agents
popB : Population size for architecture agents

Ensure:
BestH: Best hyperparameter configuration, **BestA**: Best architecture parameters

Initialization:
BestH $\leftarrow \emptyset$, BestA $\leftarrow \emptyset$
Generate initial random population popA for the hyperparameter space Ω:

$$\text{popA} = \{H_i \in \Omega \mid h_{\min} \leq H_i \leq h_{\max}, i = 1, \ldots, n_A\}$$

Generate initial random population popB for the model architecture space Ψ:

$$\text{popB} = \{A_i \in \Psi \mid i = 1, \ldots, n_B\}$$

repeat
  Hyperparameter Evolution:
  for $t = 1$ to n_g **do**
    Run genetic algorithm GA_Ω:

$$(\text{BestH}, \text{popA}) \leftarrow GA_\Omega(\text{popA}, \text{popB}, M, D_{\text{val}})$$

  end for
  Model Architecture Evolution:
  for $t = 1$ to n_g **do**
    Run genetic algorithm GA_Ψ:

$$(\text{BestA}, \text{popB}) \leftarrow GA_\Psi(\text{popB}, M, D_{\text{val}})$$

  end for
until Convergence is met:

$$|\text{Performance}(\text{BestH}) - \text{Performance}(\text{BestA})| \leq \epsilon$$

where ϵ is a predefined threshold, or after n_g generations.
Return: BestH, BestA

2) Fitness Evaluation

The fitness of each pair of agents (one architecture agent and one optimization agent) is evaluated by training a deep-learning model using the selected parameters. The fitness function returns the model's performance on the validation set, typically the minimum validation loss.

- **Model Training**: The model is trained using the selected architecture and optimization parameters. An early stopping mechanism is employed to halt training if validation performance does not improve for a predefined number of epochs, reducing computational overhead.
- **Fitness Metrics**: Fitness is assessed based on performance metrics such as the minimum validation loss or the maximum validation accuracy achieved during training. This ensures that agents contributing to models with superior generalization are favored in the evolutionary process.

3) Evolutionary Process

For each generation, the following steps are performed, as illustrated in Algorithm 2:

1) **Fitness Calculation**: The fitness of all agents is evaluated by training models using their respective parameters. The fitness score for each agent is updated.
2) **Selection**: The best agents based on fitness are selected to pass their parameters to the next generation.
3) **Mutation**: Each agent undergoes mutation with a given probability. For architecture agents, mutation can involve changes in the number of units in the dense layers or dropout rates. For optimization agents, mutation can modify the learning rate or optimizer type.
4) **Crossover**: Pairs of agents (either architecture agents or optimization agents) perform crossover with a probability of 50%. This involves exchanging parameters between two agents to create new combinations of hyperparameters.
5) **Diversity Maintenance**: To prevent premature convergence, diversity in the populations is maintained by ensuring that agents retain distinct parameter combinations and by periodically introducing random agents into the populations.
6) **Global Best Update**: The agent with the highest fitness in the population is tracked as the global best. This agent's parameters are considered the most optimal found so far.

4) Termination Criteria

The algorithm stops when one of the following conditions is met:

- **Predefined Number of Generations:** The algorithm executes for a fixed number of generations (e.g., 10 generations) unless an early stopping condition, such as meeting a satisfactory accuracy threshold, is met. This ensures the process explores a diverse range of hyperparameter configurations within a bounded computational budget.
- **Satisfactory Accuracy Threshold:** The evolutionary pro- cess terminates early if any agent achieves a validation accuracy equal to or greater than a predefined satisfactory threshold (e.g., 99%). This criterion prioritizes achieving high predictive performance over completing the full set of generations, saving computational resources when the target accuracy is reached.
- **Fitness-Based Evolution:** During each generation, agents are evaluated based on their validation loss. The algorithm primarily focuses on improving fitness by

reducing validation loss and continues the search until one of the specified stopping conditions is met. The validation loss is continuously monitored across all generations to guide the optimization process, and the global best model, selected based on the lowest validation loss, is updated iteratively.

Algorithm 2 Genetic Algorithm Details (GA_χ)

Initial population P_0, Fitness function f, Validation dataset D_{val}, Crossover rate p_c, Mutation rate p_m, Maximum generations G Best solution S_{best}

Steps:

1) **Selection:** Select parent solutions from the population P_t based on their fitness values $f(x)$.

$$P_{\text{selected}} = \text{Select}(P_t, f(x))$$

Common strategies include roulette wheel selection or tournament selection.

2) **Crossover:** Combine parent solutions to generate offspring solutions. For each pair of parents (p_1, p_2):

$$\text{offspring} = \text{Crossover}(p_1, p_2, p_c)$$

where p_c is the crossover probability.

3) **Mutation:** Introduce diversity by randomly altering genes (hyperparameters or architecture configurations) of offspring with a probability p_m:

$$\text{offspring} = \text{Mutate}(\text{offspring}, p_m)$$

4) **Evaluation:** Evaluate the fitness of each offspring on the validation dataset D_{val} using the fitness function f:

$$f(x) = \text{Evaluate}(x, D_{\text{val}})$$

5) **Replacement:** Update the population by selecting the best solutions from the union of parents and offspring:

$$P_{t+1} = \text{SelectBest}(P_t \cup \text{offspring})$$

Output: Track the best solution S_{best} across all generations:

$$S_{\text{best}} = \arg\max_{x \in P_t} f(x)$$

return S_{best}

5) Tracking the Best Solution

Throughout the evolutionary process, the best solution (S_{best}) is continuously tracked. This solution represents the agent (or pair of agents) that produced the model with the highest fitness score across all generations. At the end of the algorithm, S_{best} is returned as the optimal set of hyperparameters.

D. TRANSFER LEARNING

A transfer learning approach involves using a model that is pre-trained on a large dataset and transferring that training to a new dataset. Many applications can utilize ImageNet-trained models to extract features from smaller datasets, such as the brain's MRI dataset. The principle of transfer learning is illustrated in Fig. 1. For CNN to be effective, a large dataset must be used to train the model and avoid overfitting [12]. TL models offer dual functionality: they can serve as feature extractors or undergo fine-tuning for specific tasks. In feature extraction, the convolutional base of a pre-trained architecture is utilized, and a new classifier is trained on the features obtained from this process. Fine-tuning involves retraining selected layers of a pre-trained model to adapt it for a new specific task. This process updates the model's features to better suit the new classification task [36].

Transfer learning offers several significant advantages in machine learning. By utilizing pre-trained models, the training process becomes much faster compared to starting from scratch. This approach can also lead to higher accuracy, particularly when dealing with smaller target datasets or those with limited labeled data. Furthermore, transfer learning reduces data requirements, as the model benefits from knowledge gained from a larger, related dataset. Another key benefit is improved generalization, as knowledge from one domain can be effectively applied to enhance performance in another. This technique also optimizes the use of resources, minimizing the computational demands typically associated with training models from the ground up. Finally, adding fully connected layers to pre-trained models allows transfer learning to improve overall performance, making it an effective and powerful approach for a wide range of machine learning tasks.

A variety of transfer learning models can be found in the literature, including GoogleNet, ResNet, EfficientNet, DenseNet, Inception, and VGG, among others. After careful consideration of these architectures, this study uses the pre-trained models EfficientNetB3 and DenseNet121. EfficientNetB3 was selected for its optimal balance between computational efficiency and model depth, offering better accuracy-to-parameter ratio compared to ResNet and VGG architectures. While GoogleNet and Inception provide good performance, they lack the compound scaling benefits of EfficientNet that allows better feature extraction at multiple scales. DenseNet121 was chosen as our second model due to its feature reuse capability through dense connections, which is particularly valuable for detecting subtle tumor characteristics, while maintaining lower memory requirements compared to deeper variants like ResNet152 or Inception-v3.

E. DENSENET121

The DenseNet is a form of DL model that has been pre-trained and employs feedforward connections to link each layer with all the following layers [37]. DenseNet121 is a deep neural network used for computer vision tasks, particularly in extracting features from medical images like brain tumor MRI scans. It belongs to the Densely Connected Convolutional Networks (DenseNets) family, known for its efficiency in capturing intricate image features. Dense connectivity and multi-level feature extraction enhance brain tumor classification

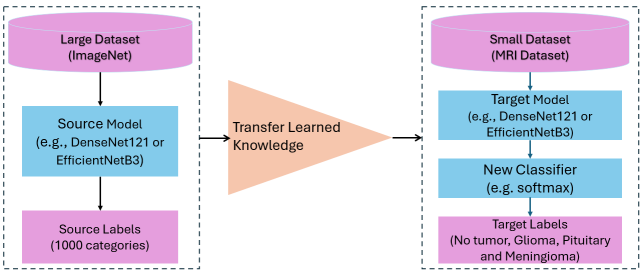


FIGURE 1: The principle of transfer learning.

accuracy by providing rich image representations. DenseNet was designed to address gradient vanishing challenges in deep neural networks, where longer paths between input and output layers increase the risk of data loss. In a composite function, the output from one layer becomes the input for the next. This process involves components like convolution, pooling, batch normalization, and non-linear activation layers working together to transform and process data. DenseNet consists of 121 layers, where each layer's output is concatenated with the next layer's input in a feed-forward manner. The direct connections of DenseNet121 have (Lx(L+1))/2 connections, while the L-layered traditional CNN has L connections. Each layer uses the feature maps from all previous layers as inputs, while its own feature maps serve as inputs for the subsequent layers. In this study, we chose this particular DenseNet variant because of its many advantages, such as its capacity to address gradient vanishing, minimize the number of parameters, improve feature propagation, and facilitate feature reuse. DenseNet reduces the network parameter count compared to regular CNNs by preventing redundant learning of feature maps. Additionally, to minimize the risk of overfitting, DenseNet incorporates regularization techniques, such as dropout layers, along with activation functions. The DenseNet121 architecture comprises four dense blocks, each of which contains six, twelve, twenty-four, and sixteen convolution blocks, respectively. Fig. 2 illustrates the revised DenseNet121 architecture, which this study ultimately used to achieve classification results.

F. EFFICIENTNETB3

The EfficientNet model family consists of eight variants labeled B0 to B7, all trained on the ImageNet dataset. EfficientNet-B3 utilizes a compound scaling approach to automatically optimize the model's architecture by proportionally increasing its depth (layers), width (filters per layer), and resolution of the input image [38]. This approach allows EfficientNetB3 to strike a balance between high performance and computational efficiency.

The selection criteria for EfficientNet-B3 are based on several factors, including dataset size, available resources for model training and evaluation, model depth, network parameters, and batch size. EfficientNet-B3 was chosen for this study due to its optimal trade-off between computational efficiency and classification accuracy. While smaller variants, such as

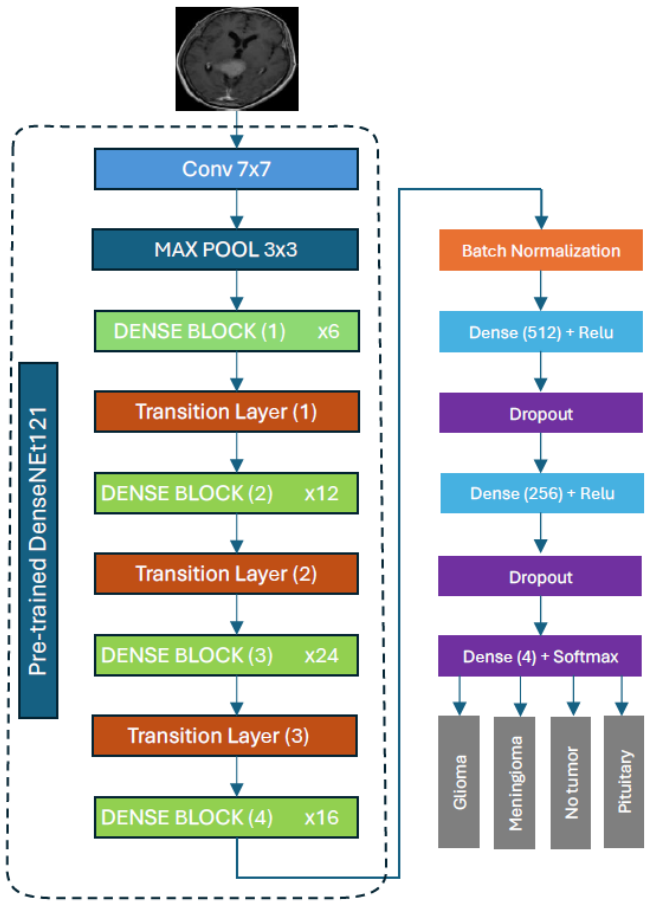


FIGURE 2: The modified DenseNet121 architecture is designed for multi-class classification.

B0 and B1, have less network depth and fewer parameters, offering lower computational costs, they may not provide the necessary performance for complex tasks like brain tumor detection.

On the other hand, larger variants, such as B5 and B7, have increased network depth and more parameters, making them prone to overfitting the training data and requiring higher computational resources (GPU and RAM). EfficientNet-B3 strikes a balance by offering high accuracy with reasonable computational demands, making it well-suited for brain tumor classification. This is achieved through its compound scaling method, which allocates additional resources to handle both small and large images, enhancing performance while minimizing the risk of overfitting and maintaining efficient resource utilization.

The compound scaling method enables computationally efficient deep learning models to achieve top accuracy. EfficientNet-B3 allows for the allocation of additional resources when handling both large and small images, resulting in enhanced performance and efficiency. The increased number of filters within each layer and the higher spatial resolution enable the model to capture finer details in input images [39]. EfficientNet-B3 is the third model in the

EfficientNet series, retaining the depth of EfficientNet-B0 while expanding its width by 1.3x and its resolution by 1.2x. EfficientNet-B3 features expanded layer filtering, enhanced spatial resolution, and 26 convolutional blocks with batch normalization and ReLU activations. Using depthwise separable convolutions reduces parameters while capturing complex patterns, achieving higher accuracy with less computational effort. Compared to B1, it is more efficient, with reduced model size, faster inference, and lower memory consumption. EfficientNet models are constructed using inverted residual convolutional blocks, which are derived from the MobileNetV2 architecture. These inverted residual convolutional blocks, also known as MBConv, utilize multiple kernel sizes, including 3x3 and 5x5 convolutions [40]. The EfficientNetB3 offers a convolution filter (Conv) with a kernel size of 3x3, MBConv1 with a kernel size of 3x3, and MBConv6 with both 3×3 and 5×5 kernel sizes. Certain MBConv6 blocks utilize inverted residual connections (IRC). A feature map is extracted from input images using an EfficientNetB3 model with kernel sizes of 3x3 and 5x5. The efficientNetB3 consists of 25 MBConv blocks that vary in feature map expansion ratios, resolutions, and output layer kernel sizes. The MBConv1 block uses a 3x3 convolutional kernel, while the MBConv6 blocks use both 3x3 and 5x5 convolutional kernels. These MBConv blocks rely on depthwise convolution, which is coupled with batch normalization and an activation function. Moreover, dropout layers and skip connections are incorporated in MBConv6 with 3×3 and 5×5 kernels, but these are not included in MBConv1. This approach allows the model to detect more intricate patterns in the input data while maintaining a relatively low number of parameters. Moreover, it allows us to allocate additional resources for handling both large and small images, leading to enhanced performance and efficiency. In summary, EfficientNetB3 was selected for this study because it provides a good trade-off between accuracy and efficiency, making it ideal for brain tumor classification tasks that require both high performance and reasonable computational demands. Fig. 3 shows the modified EfficientNetB3 architecture, which is designed for multi-classification.

IV. PROPOSED METHODOLOGY

In this section, we present the novel integration of CEGA with the pre-trained EfficientNetB3 and DenseNet121 models to optimize hyperparameters for brain tumor classification. The CEGA is applied to optimize the hyperparameters of EfficientNetB3 and DenseNet121, improving both convergence speed and accuracy. With the optimal hyperparameters identified, we train the final model on the complete dataset and evaluate its performance on a separate held-out test set to obtain the definitive performance metrics. Fig. 4 shows the procedure flow for brain tumor classification. The proposed methodology comprises four distinct stages:

A) Stage 1: Preprocessing data
B) Stage 2: Hyperparameter selection
C) Stage 3: Learning process
D) Stage 4: Performance Metrics

A detailed overview of each stage is provided in the following subsections.

A. PREPROCESSING DATA

In data preprocessing, data is cleaned and prepared so that ML models can perform better and be more effective.

The input images are resized to 240 x 240 x 3 to match the input tensor dimensions required by the pre-trained DenseNet and EfficientNet models. Fig. 5 demonstrates the process of cropping MR images by finding the borders of the brain contour and removing the leftover portions. The following approach is applied for extracting extreme points: First, the input image is converted to grayscale, reducing it to a single intensity channel to simplify processing, decrease computational complexity, and eliminate redundant color information. A Gaussian blur is then applied to the grayscale image. This technique smooths the image by reducing high-frequency noise and detail using a Gaussian function, which helps in creating a cleaner input for the subsequent segmentation step. By smoothing out small inconsistencies and sharp edges, Gaussian blur ensures that noise does not interfere with the detection of meaningful features, such as the brain's boundary. Subsequently, Otsu's thresholding method is employed to isolate the brain region from the background. This method automatically determines an optimal threshold value by minimizing the intra-class variance within the foreground (brain) and background. This property makes Otsu's method particularly suitable for segmenting MR images, which often exhibit distinct intensity differences between the brain tissue and surrounding areas. The thresholded image is then analyzed to detect contours, and the largest contour, assumed to represent the brain, is selected.

Lastly, the outermost points of the largest contour, representing the outer boundary of the brain region, are identified and marked with red stars on the original image. Then, the original image is cropped using these outermost points, eliminating any extraneous parts of the image that could potentially introduce ambiguity during training. This preprocessing step ensures that the input data is both focused on the region of interest and free from irrelevant details, enhancing the clarity and quality of the data provided to the models. After removing the undesired portions of the brain tumor MRI, the dataset is split, with 80% allocated for training and 20% reserved for testing.

B. HYPERPARAMETER SELECTION

As outlined in Section III-D, the TL approach leverages a pre-trained model with certain modifications. The primary modification involves replacing the existing classifier with a new one, which may involve introducing new hyperparameters or adjusting existing ones. The proposed CEGA-EfficientNetB3 and CEGA-DenseNet121 models incorporate four fine-tuned hyperparameters: the number of neurons , the dropout rate, the learning rate, and the optimizer algorithm.

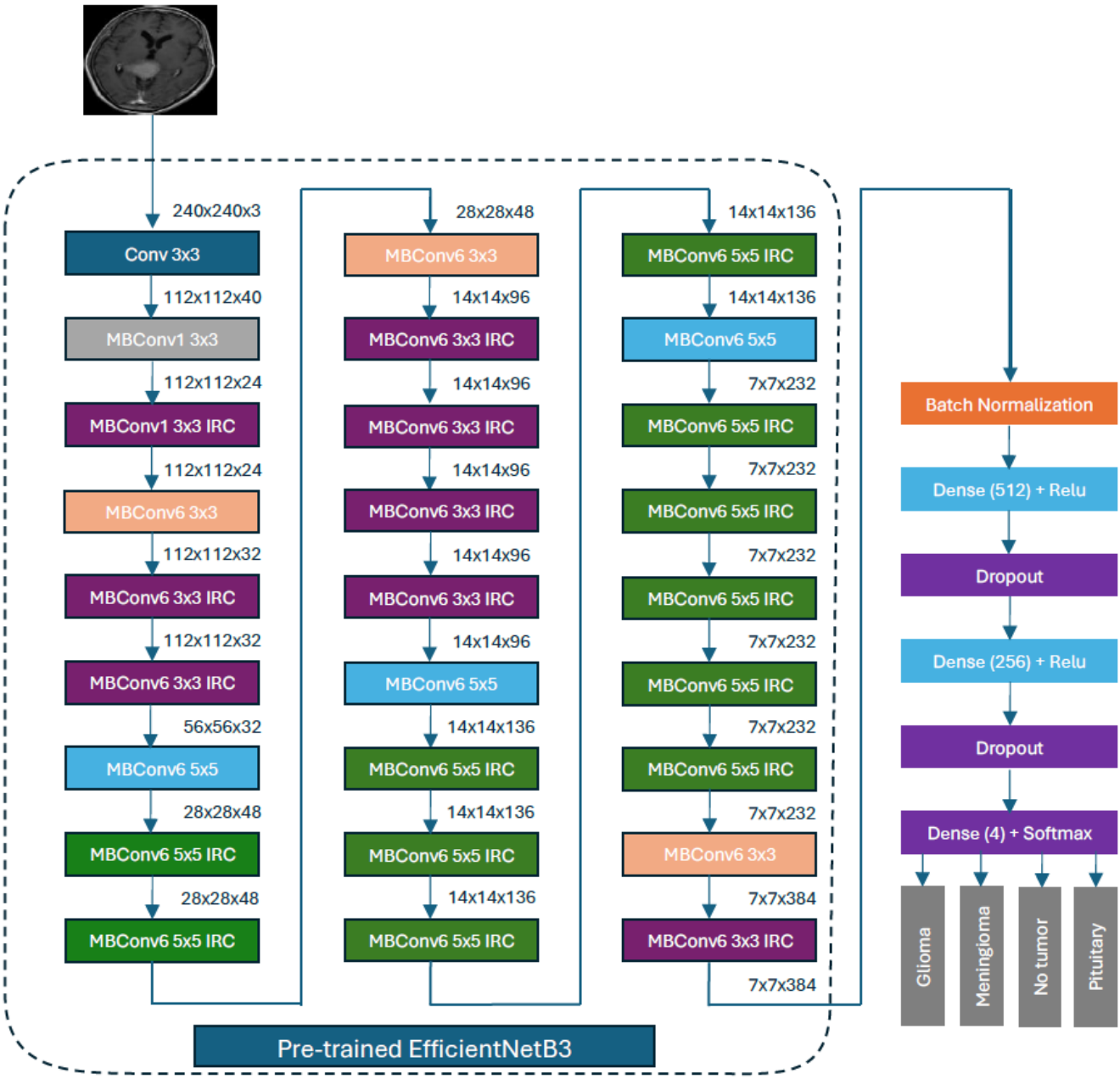


FIGURE 3: The modified EfficientNetB3 architecture is designed for multi-class classification.

C. LEARNING PROCESS

Pre-trained models can serve as feature extractors or be fine-tuned for downstream tasks. In this work, these techniques are applied to prepare the EfficientNetB3 and DenseNet121 architectures for learning from the given dataset. During the feature extraction process, the convolutional base remains unchanged while a new classifier is implemented and additional top layers are added. Following the feature extraction process, the batch normalization layer is applied to normalize the extracted features based on standard and mean deviation. A three-layer dense neural network is then appended to the existing networks. The initial and subsequent fully connected layers are each followed by ReLU activations. The final dense layer is subsequently connected to a softmax activation function in order to classify the image. The dropout layer is employed after each fully connected layer. L1 and L2 regularization are applied to the first and second dense layers. Regularization and dropout techniques help combat overfitting. The CEGA selects the optimal dropout rates for the first and second dropout layers, as well as the number of neurons for the first and second dense layers. Using the optimal hyperparameters identified, we train the final model on the complete dataset and evaluate its performance on a separate held-out test set to obtain the definitive performance

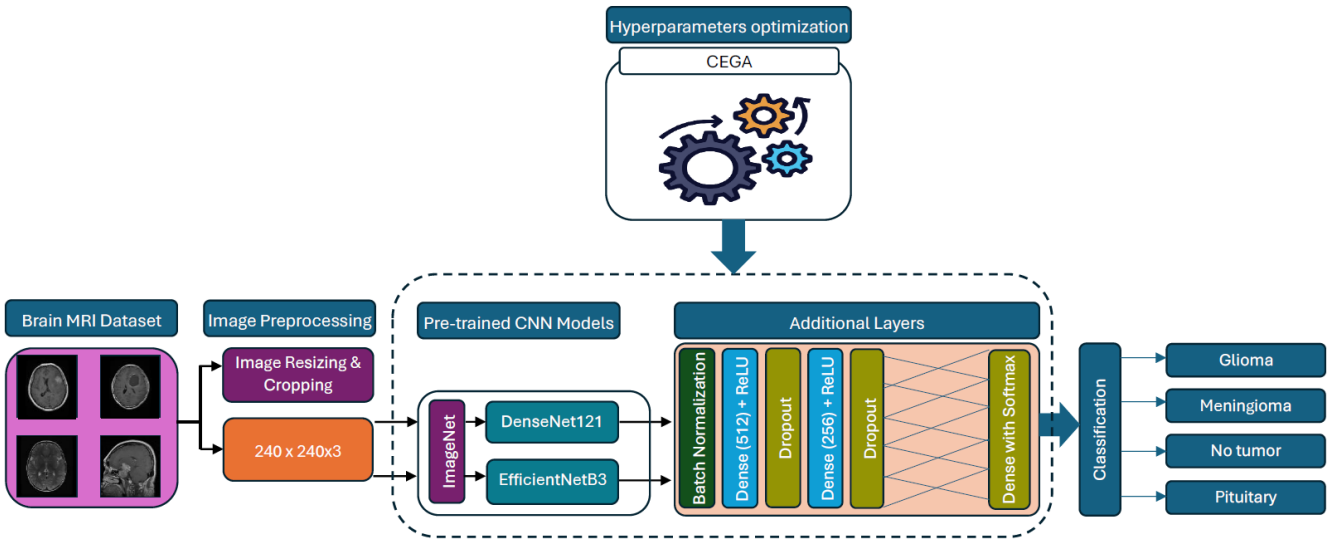


FIGURE 4: Procedural steps of the proposed approach.

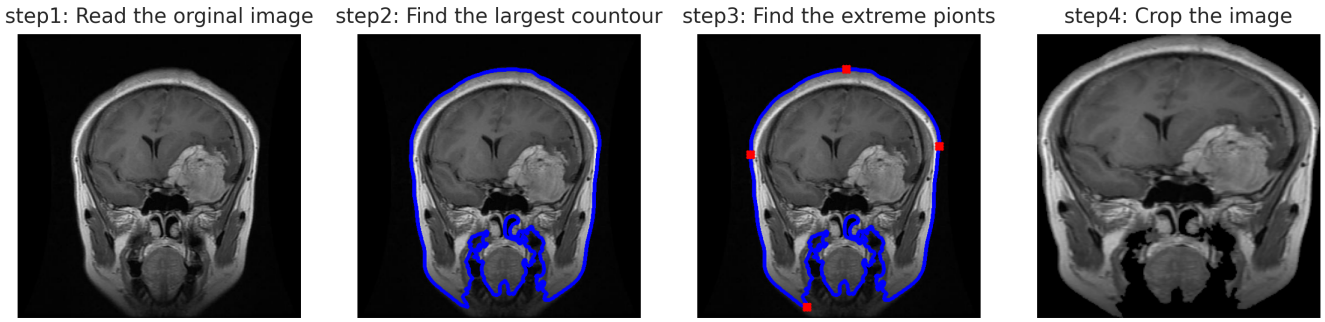


FIGURE 5: Steps used to remove undesired portions from a brain tumor MRI scan.

metrics.

D. PERFORMANCE METRICS

The performance of ML and DL models can be evaluated using a variety of metrics. A confusion matrix (CM) is a common method to formally present a model's performance. It is a table that summarizes how effectively a classification or prediction model performs. The CM can be used to calculate several metrics for analyzing classification performance. These metrics are employed to evaluate the predictive performance of the models CEGA-EfficientNetB3 and CEGA-DenseNet121, including Precision (Eq. (4)), Sensitivity (Eq. (5)), Accuracy (Eq. (6)), F1-score (Eq. (7)), and Specificity (Eq. (8)) [1], [41].

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4}$$

$$\text{Sensitivity} = \text{Recall} = \text{TPR} = \frac{TP}{TP + FN} \tag{5}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{7}$$

$$\text{Specificity} = \text{TNR} = \frac{TN}{TN + FP} \tag{8}$$

Here, 'TP' refers to "TRUE POSITIVE," representing the count of positive labels that have been accurately predicted. 'FP,' or "FALSE POSITIVE" represents the number of instances that were predicted as positive but are actually negative.

'FN' refers to "FALSE NEGATIVE," which represents the count of labels predicted as negative but are actually positive. It's essential to recognize that sensitivity and recall are equivalent; a higher sensitivity/recall value signifies a model's stronger ability to generalize effectively.

V. EXPERIMENTAL SETUP

We conduct our experiments on Google Colab, a publicly available notebook platform provided by Google, which offers both complimentary and subscription-based access to

TPU and GPU resources for academic and research purposes. The models are trained on an "NVIDIA T4 Tensor Core GPU with 16 GB of GDDR6 RAM" [42]. The system is configured with 25 GB of high RAM, enabling the handling of large datasets and complex models. The code is implemented in Python, utilizing TensorFlow as the backend and Keras as the frontend API for model development.

A. DATASET

We trained, validated, and evaluated our approaches using a publicly accessible Brain Tumor MRI dataset. The dataset is comprised of 3 distinct datasets: "Figshare, BrH35, and SARTAJ" [43]. As illustrated in Table 2, the dataset comprises 7,023 human brain MRI scans classified into four categories: gliomas, meningiomas, no tumors, and pituitary tumors. It is in JPG and grayscale format. The dataset is divided into four classes: for gliomas, the numbers of training and testing images are 1,321 and 300, respectively. The meningioma has 1,339 training images and 306 testing images. No tumor training and testing images are 1,595 and 405, respectively. The Pituitary class in the dataset contains 1,457 images for the training set and 300 images for the testing set. For training and evaluation, 80% of the images were used for training, while the remaining 20% were allocated for testing.

TABLE 2: Description of the dataset.

Class	Training set	Test set	Total
Glioma	1,321	300	1,621
Meningioma	1,339	306	1,645
Pituitary	1,457	300	1,757
No tumor	1,595	405	2,000
Total	5,712	1,311	7,023

B. HYPER-PARAMETERS TUNING

Optimizing hyperparameters is a crucial stage in developing effective machine-learning models. The objective is to identify the optimal hyperparameter configuration that improves the model's performance on a specific dataset. The parameters, such as optimizers, the number of neurons, dropout rates, learning rates, batch sizes, epochs, and loss functions, can affect model training and produce optimal results. These are referred to as hyperparameters. These parameters are adjusted through systematic experimentation until the ideal configuration is determined for training the model and achieving the intended outcomes; this process is called hyperparameter optimization. After several experiments using CEGA for hyperparameter optimization, the optimal values for the number of neurons, dropout rate, optimizer, and learning rate were determined. The fine-tuning of the proposed architectural models is accomplished using two pre-trained models: EfficientNetB3 and DenseNet121. Each model was tested for twenty epochs, utilizing a range of various optimizers, dropout rate values, learning rates, and the number of neurons, as detailed in Table 3.

1) Hyperparameter optimization for the modified versions EfficientNetB3 and DenseNet121:-

The CEGA algorithm is used to optimize four key hyperparameters for both models: Number Of Neurons, Learning Rate, Dropout Rate, and Optimizer. Dynamic hyperparameter tuning is essential for deep learning models. In this context, specific hyperparameters are continuously adjusted until an optimal value threshold is reached. Once the optimal settings are identified, the model is trained, and features are extracted for use in classification tasks. Hyperparameter optimization aims to enhance a machine learning algorithm's effectiveness by identifying and selecting the optimal configuration settings [44]. According to Equation 9, where f represents performance and x refers to a particular hyperparameter setting, the optimal choice is denoted as x_{opt}.

$$x_{\text{opt}} = \arg\max_{x \in X} f(x) \tag{9}$$

The hyperparameters chosen for optimization are the dropout rate, learning rate, optimizer algorithm, and number of neurons. The performance and effectiveness of this multiclass classification model heavily depend on these key hyperparameters. Table 3 outlines the acceptable value ranges for each of these hyperparameters. The optimizers considered include AdaDelta, RMSprop, Adamax, Adam, Nadam, SGD, and AdamW. The dropout rates are restricted to 0.2, 0.3, 0.4, and 0.5. The learning rates considered are 0.1, 0.01, 0.001, 0.0001, and 0.00001. The number of neurons for the dense layers is set to 128, 256, and 512.

The convergence of CEGA is governed by well-defined criteria to ensure a balance between optimization quality and computational efficiency. First, the algorithm operates for a predefined number of generations (e.g., 10 generations) unless an early stopping condition is triggered. One such condition is achieving a satisfactory accuracy threshold, where the process halts early if any agent attains a validation accuracy equal to or exceeding a predefined target (e.g., 99%), thereby prioritizing high performance and conserving computational resources. Additionally, the optimization process is driven by fitness-based evolution, with agents evaluated based on their validation loss during each generation. The search aims to minimize validation loss, updating the global best model iteratively as improvements are identified. The CEGA continues until one of these criteria—exhausting the allowed generations or meeting the accuracy threshold—ensures effective exploration while adhering to computational constraints.

VI. EXPERIMENTAL EVALUATION AND DISCUSSION

The outcomes of the proposed CEGA-EfficientNetB3 and CEGA-DenseNet121 models, as outlined in Section IV, are discussed and analyzed in this section. The results are organized into four main subsections to enhance the clarity of the research findings.

TABLE 3: Hyperparameters and their values used in the experiments.

Parameter name	Value
Population size	10
Mutation rate	0.1
No. of generations	10
Dense Units	[128, 256, 512]
Dropout Rate	[0.2, 0.3, 0.4, 0.5]
Optimizer	['SGD', 'Adam', 'RMSprop', 'Adadelta', 'Adamax', 'Nadam', 'AdamW']
Learning Rate	[0.00001, 0.0001, 0.001, 0.01, 0.1]
No. of epochs	20
Batch Size	32
Loss Function	Categorical cross-entropy

TABLE 4: Best hyperparameters obtained from the optimization process.

Hyperparameter	EfficientNetB3	DenseNet121
Optimizer	RMSprop	Adadelta
Learning Rate	0.0001	0.1
First-Layer Dropout Rate	0.5	0.5
Second-Layer Dropout Rate	0.2	0.5
First-Layer Dense Units	512	512
Second-Layer Dense Units	256	256

A. DEVELOPING THE CEGA FOR SELECTING HYPERPARAMETERS DURING THE OPTIMIZATION PHASE

This subsection outlines the search space boundaries for hyperparameters, with values determined by the proposed CEGA. Table 3 shows the parameter value configurations for the combination of the CEGA algorithm with the EfficientNetB3 and DenseNet121 models. The search space for the following hyperparameters is outlined in the table. The maximum number of generations is set to 10, limiting the duration of the optimization algorithm's operation. The population size hyperparameter sets how many solution candidates are evaluated per generation. With a value of 10, the optimization algorithm assesses 10 different candidates in each generation. The optimizer algorithm is responsible for adjusting the weights of the neural network throughout training to reduce the loss function and improve performance. This hyperparameter's search space consists of [AdaDelta, RMSprop, Adamax, Adam, Nadam, SGD, and AdamW]. The learning rate hyperparameter controls how much the optimization algorithm adjusts the weights of the neural network at each training step. The search space for the learning rate hyperparameter included values of 0.1, 0.01, 0.001, 0.0001, and 0.00001. The batch size hyperparameter defines the number of training samples processed per iteration. The value for this hyperparameter is 32. The dropout rate is a hyperparameter that regulates the neural network by randomly deactivating specific units during training to prevent overfitting. The search range for this hyperparameter is established at [0.2, 0.3, 0.4, 0.5]. The number of neurons is a hyperparameter that specifies how many neurons are used in the hidden layers of the EfficientNetB3 and DenseNet121 architectures. The search space for this hyperparameter is defined by the values [128, 256, 512]. Furthermore, the training process for the EfficientNetB3 and DenseNet121 models consists of 20 epochs.

The convergence criteria for the CEGA are designed to balance exploration and exploitation while ensuring computational efficiency. The algorithm terminates based on one of three primary conditions: first, a predefined number of generations is reached, ensuring the process explores a diverse range of hyperparameter configurations within a bounded computational budget; second, a satisfactory accuracy threshold is met, where the evolutionary process halts early if any agent achieves a validation accuracy equal to or greater than a predefined target (e.g., 99%), prioritizing high predictive performance over completing all generations; or third, fitness-based evolution guides the process, with agents evaluated on validation loss, and the algorithm continues until the global best model, determined by the lowest validation loss, is identified and no further significant improvements are observed. These criteria collectively ensure that CEGA converges efficiently, either by exhausting the generational limit, achieving the desired accuracy, or optimizing fitness to a satisfactory level.

The objective of the CEGA algorithm is to maximize validation accuracy while minimizing validation loss as effectively as possible. The effectiveness of the proposed CEGA methods is measured by evaluating the validation loss achieved with the optimized parameters. Following training the proposed CEGA-EfficientNetB3 and CEGA-DenseNet121 models, optimal values for the learning rate, optimizer, dropout rate, and dense units were selected. Table 4 presents the optimal hyperparameters identified using the CEGA algorithm.

B. TRAINING THE EFFICIENTNETB3 AND DENSENET121 MODELS USING THE OPTIMIZED HYPERPARAMETERS

During this stage, the EfficientNetB3 and DenseNet121 models were trained using the training data and assessed on the test data, employing the optimal hyperparameters identified through the CEGA method, as shown in Table 4. EfficientNetB3 utilized the RMSprop optimizer with a learning rate of 0.0001, a first-layer dropout rate of 0.5, and a second-layer dropout rate of 0.2. Additionally, the first dense layer had 512 units, and the second dense layer had 256 units. DenseNet121 was trained with the Adadelta optimizer, with a learning rate of 0.1 and dropout rates of 0.5 for both layers. Similarly, DenseNet121's first dense layer contained 512 units, and the second dense layer contained 256 units. Both models were compiled using sparse categorical cross-entropy [45]. Experiments revealed the best test results at the 20th epoch, using a batch size of 32 on 5,712 training samples, with 20% reserved for validation.

C. EVALUATION OF CEGA-EFFICIENTNETB3 AND CEGA-DENSENET121 FOR BRAIN TUMOR CLASSIFICATION

This section introduces an evaluation of the performance of the CEGA-EfficientNetB3 and CEGA-DenseNet121 mod-

els using hyperparameter values derived from the proposed CEGA algorithm.

Firstly, the proposed CEGA-EfficientNetB3 model was tested for classifying MRI brain tumor scans, with optimal hyperparameters determined using CEGA. The classification results for the CEGA-EfficientNetB3 model, evaluated across various metrics, are summarized in Table 5. As shown in Table 5, the model achieved its highest classification performance using the RMSprop optimizer, with a test accuracy of 99.39%, 99.36% precision, 99.38% recall, 99.80% specificity, and an F1-score of 99.37%. The lowest classification performance using the Adadelta and Nadam optimizers, with a test accuracy of 99.01%, 98.99% precision, 98.97% recall, 99.67% specificity, and an F1-score of 98.98% for both optimizers.

Secondly, the classification outcomes related to CEGA-DenseNet121 are illustrated in Table 6. Based on Table 6, we have accomplished the best performance with 99.01% test accuracy, 98.95% precision, 98.97% recall, 99.67% specificity, and an F1-score of 98.96% using the Adadelta optimizer. The lowest performance was recorded with 98.32% test accuracy, 98.27% precision, 98.20% recall, 99.44% specificity, and an F1-score with 98.23% using the Adamax optimizer.

TABLE 5: Evaluation metrics for CEGA-EfficientNetB3 model performance with different optimizers.

Optimizers	Accurcy (%)	Precesion (%)	Recall (%)	Specificity (%)	F1-score (%)
ADAM	99.24	99.22	99.20	99.75	99.21
Nadam	99.01	98.99	98.97	99.67	98.98
Adamax	99.08	99.03	99.03	99.70	99.03
RMSprop	**99.39**	**99.36**	**99.38**	**99.80**	**99.37**
SGD	99.24	99.18	99.20	100	99.19
Adadelta	99.01	98.99	98.97	99.67	98.98
AdamW	99.24	99.22	99.22	99.75	99.22

TABLE 6: Evaluation metrics for CEGA-DenseNet121 model performance with different optimizers.

Optimizers	Accurcy (%)	Precesion (%)	Recall (%)	Specificity (%)	F1-score (%)
ADAM	98.86	98.85	98.76	99.62	98.79
Nadam	98.93	98.95	98.87	99.64	98.91
Adamax	98.32	98.27	98.20	99.44	98.23
RMSprop	98.70	98.62	98.64	99.58	98.63
SGD	98.93	98.97	98.87	99.64	98.92
Adadelta	**99.01**	**98.95**	**98.97**	**99.67**	**98.96**
AdamW	98.70	98.69	98.59	99.57	98.64

1) Class-Specific Performance Analysis

Table 7 shows the class-specific performance of the fine-tuned CEGA-EfficientNetB3 and CEGA-DenseNet121 on new test data in terms of sensitivity, accuracy, F1-score, specificity, and precision for each class. From the table, the CEGA-EfficientNetB3 model demonstrated excellent performance across all metrics using the RMSprop optimizer. Among all classes, pituitary tumors were classified with the highest accuracy, attaining 99.85% accuracy in testing, while the most misclassified is meningioma compared to pituitary, no tumor, and glioma, with an accuracy of 99.54%. CEGA-DenseNet121 correctly classified the No Tumor class using the Adadelta optimizer, achieving 99.77% test accuracy, but misclassified the meningioma class compared to no tumor, pituitary, and glioma, with a test accuracy of 99.16%.

TABLE 7: Evaluation metrics for each category of the proposed approaches.

Model	Tumor category	Precesion (%)	Recall (%)	Specificity (%)	F1-score (%)	Accuracy (%)
CEGA-DenseNet121	Glioma	99.33	98.33	99.80	98.83	99.47
	Meningioma	97.73	98.69	99.30	98.21	99.16
	Pituitary	99.00	99.33	99.70	99.17	99.62
	No Tumor	99.75	99.51	99.89	99.63	99.77
CEGA-EfficientNetB3	Glioma	98.68	99.67	99.60	99.17	99.62
	Meningioma	99.34	98.69	99.80	99.02	99.54
	Pituitary	99.67	99.67	99.90	99.67	99.85
	No Tumor	99.75	99.51	99.89	99.63	99.77

D. DISCUSSION

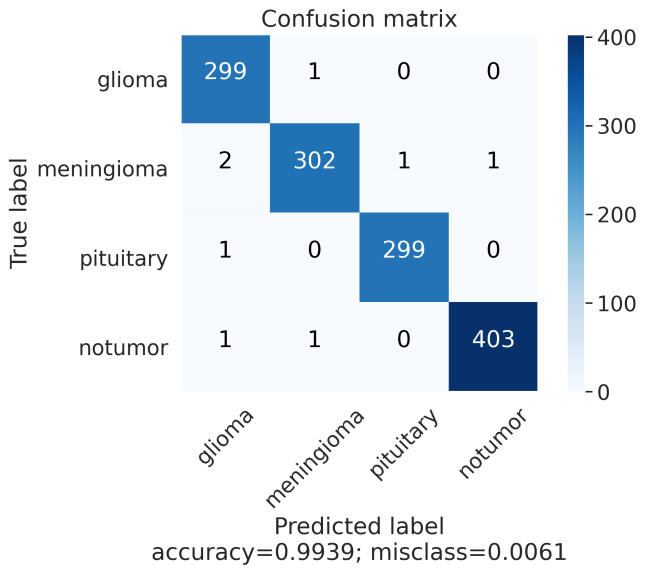
This article has effectively enhanced the accuracy of brain tumor classification. Utilizing Google Colab accelerated the computational process and improved the precision of the outcomes. The assessment of the suggested transfer learning approaches is illustrated by utilizing accuracy and loss curves.

1) Analysis of Training and Validation Accuracy and Loss Curves

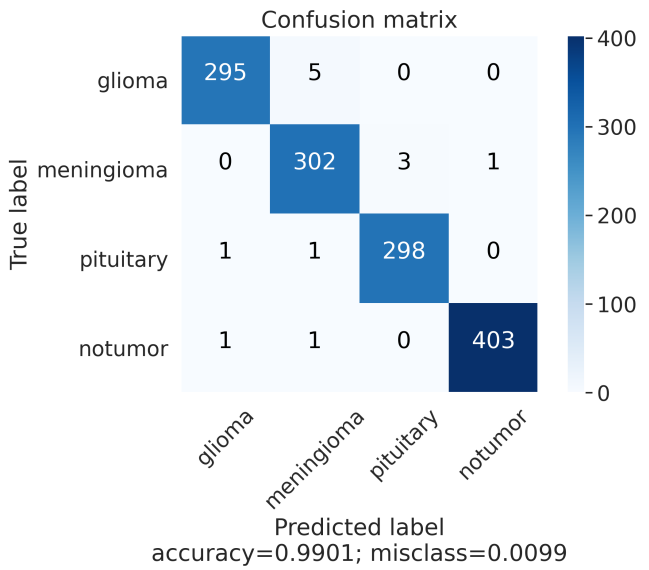
Figures 7 and 8 illustrate the training performance, highlighting metrics such as accuracy and loss for both training and validation of two distinct TL models, EfficientNetB3 and DenseNet121, using the RMSprop and Adadelta optimizers, respectively, across various epochs. The models achieved excellent convergence, attaining the highest possible accuracy with low training and validation losses. With increasing epochs, the models' training and validation accuracy improved while training and validation losses decreased. As learning iterations progressed, the proposed method demonstrated enhanced precision on both training and test data while simultaneously reducing error rates in both datasets.

2) Confusion Matrix Analysis for Model Performance

The performance of the optimized CEGA-EfficientNetB3 and CEGA-DenseNet121 models was evaluated by predicting their outcomes, as shown in the confusion matrix in Fig. 6. The CEGA-EfficientNetB3 model demonstrated superior performance over the other model. Figure 6(a) highlights the CEGA-EfficientNetB3 model's minimal classification errors across all categories, with only 1 misclassification for glioma, 4 for meningioma, 2 for non-tumor cases, and 1 for pituitary tumors. In comparison, Fig. 6(b) shows the misclassification errors of the CEGA-DenseNet121 model: 5 for glioma, 4 for meningioma, 2 for non-tumor cases, and 2 for pituitary tumors. The CEGA-EfficientNetB3 model achieved notably high accuracy, correctly identifying 900 out of 906 brain tumor cases and accurately classifying 403 out of 405 healthy individuals. In comparison, the CEGA-DenseNet121 model successfully detected 895 out of 906 brain tumor cases and accurately classified 403 out of 405 healthy individuals. Based on these results, the CEGA-EfficientNetB3 and CEGA-DenseNet121 models demonstrate high accuracy (99.39% and 99.01%, respectively), with minimal misclassifications, indicating robust performance in classifying glioma, meningioma, pituitary tumors, and non-tumor cases. Given these results, both models are suitable for real-time detection of brain tumors, providing reliable and accurate diagnostic capabilities.



(a) EfficientNetB3



(b) DenseNet121

FIGURE 6: Confusion matrix of the fine-tuned models.

3) AUC-ROC Curve Analysis for Model Evaluation

The AUC-ROC curve is a vital tool for evaluating the classification performance of the proposed optimized CEGA-EfficientNetB3 and CEGA-DenseNet121 models on the brain tumor dataset. Figure 9 illustrates the ROC curves for the two models, showcasing their ability to balance sensitivity (True Positive Rate) and specificity (False Positive Rate) across different thresholds. For each class, the AUC values were calculated, along with micro-averaged and macro-averaged scores, to summarize overall model performance. In CEGA-EfficientNetB3 (Figure 9(a)), the AUC values for individual classes were as follows: glioma: 0.9971, meningioma: 0.9992, pituitary: 1.0000, and no tumor: 0.9997, achieving a micro-average AUC of 0.9989 and a macro-average AUC of 0.9990. This model exhibited near-perfect classification performance, as its ROC curves closely approached the top-left corner, reflecting high sensitivity and specificity. Similarly, in CEGA-DenseNet121 (Figure 9(b)), the AUC values for individual classes were: glioma: 0.9968, meningioma: 0.9988, pituitary: 0.9999, and no tumor: 1.0000, with a micro-average AUC of 0.9990 and a macro-average AUC of 0.9989. The comparison highlights slight variations between the models in terms of class-specific performance. For instance, CEGA-DenseNet121 achieved an AUC of 1.0000 for Class 4 (no tumor), outperforming CEGA-EfficientNetB3, which achieved an AUC of 0.9997 for the same class. Meanwhile, CEGA-EfficientNetB3 demonstrated perfect performance (AUC: 1.0000) for Class 3 (pituitary) compared to CEGA-DenseNet121's AUC of 0.9999 for the same class. However, both models demonstrate exceptional generalization capabilities across all classes. These results affirm the robustness of the models while providing insights for fine-tuning and hyperparameter optimization to enhance performance further.

VII. COMPARISON WITH PREVIOUS STUDIES

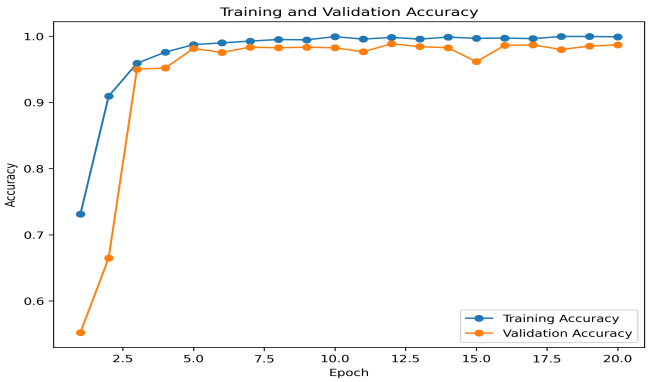
This section presents a comparative evaluation of the performance of the fine-tuned CEGA-EfficientNetB3 and CEGA-DenseNet121 models against leading, cutting-edge approaches developed between 2020 and 2024, using a combined dataset from 'Figshare, BrH35, and SARTAJ,' along with additional datasets, as shown in Table 8. The efficacy of the proposed models is compared based on accuracy, the most commonly used metric across studies. As indicated in Table 8, the results obtained from experiments using the proposed CEGA-EfficientNetB3 and CEGA-DenseNet121 models are significantly higher compared to other cutting-edge methods. These models are capable of extracting more robust and distinct deep features, resulting in superior classification. The features extracted from the final fully connected layer are then fed into the softmax classifier for classification. Table 8 highlights that the CEGA-EfficientNetB3 model achieved the highest classification accuracy on the dataset, with the CEGA-DenseNet121 model following as the second-highest classification accuracy. These outcomes indicate that the proposed models demonstrate high accuracy in classifying brain MRI images. In conclusion, the evaluation metrics demonstrate that the CEGA-EfficientNetB3 model outperformed the other models on the same dataset.

VIII. LIMITATIONS AND FUTURE DIRECTIONS

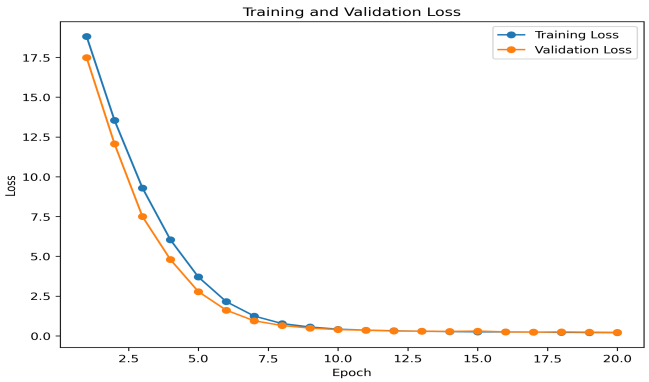
The Co-evolutionary Genetic Algorithm (CEGA) offers a promising approach for optimizing both the architecture and hyperparameters of deep learning models. The proposed CEGA-EfficientNetB3 model demonstrates high classification accuracy in brain tumor detection using MRI. Notably, the integration of CEGA with pretrained models has enhanced convergence speed and improved the accuracy of optimal hyperparameter selection. Additionally, it has strengthened the exploration capability of the Genetic Algorithm (GA) and optimized both the exploration and exploitation phases. Furthermore, the CEGA approach achieved impressive accuracy in determining the optimal hyperparameter values for the EfficientNetB3 and DenseNet121 models, with results of 99.39% and 99.01%, respectively. Despite these successes, there are still areas for improvement that need to be addressed in order to further refine the algorithm and enhance its scalability and generalization.

The limitations of the CEGA algorithm, as well as those of the proposed CEGA-EfficientNetB3 model, are outlined as follows:

- The computational cost of the CEGA algorithm increases with more agents and generations, but this added time enhances performance.
- The CEGA-EfficientNetB3 model was trained and evaluated only on the "Figshare, BrH35, and SARTAJ" combined dataset, which restricts its generalizability to other datasets.
- The CEGA algorithm has been shown to effectively tune hyperparameters for EfficientNetB3 and DenseNet121, which are distinct architectures and do not belong to the same variant.

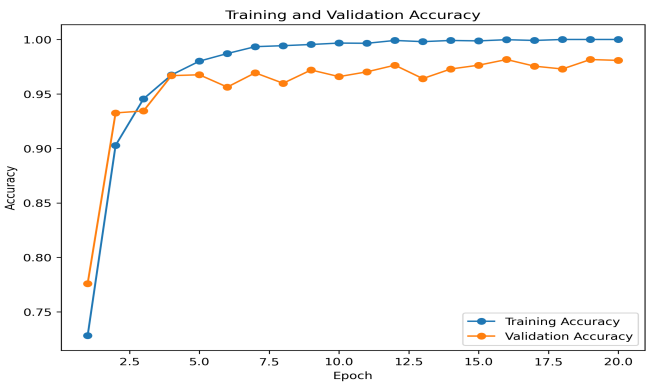


(a) Training vs. Validation Accuracy

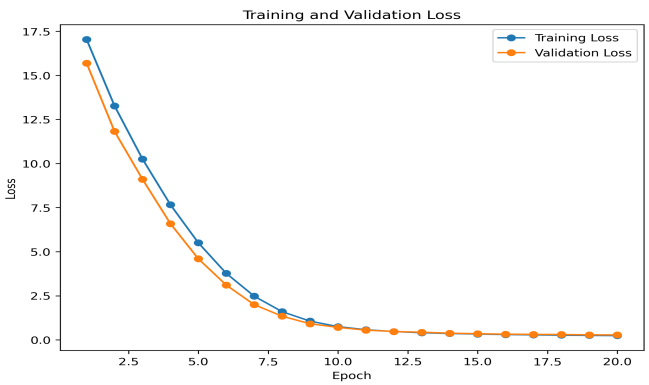


(b) Training vs. Validation Loss

FIGURE 7: Accuracy and loss curves for the proposed fine-tuned CEGA-EfficientNetB3 model.



(a) Training vs. Validation Accuracy



(b) Training vs. Validation Loss

FIGURE 8: Accuracy and loss curves for the proposed fine-tuned CEGA-DenseNet121 model.

However, further research is needed to determine whether this approach is also effective for other pre-trained CNN architectures.

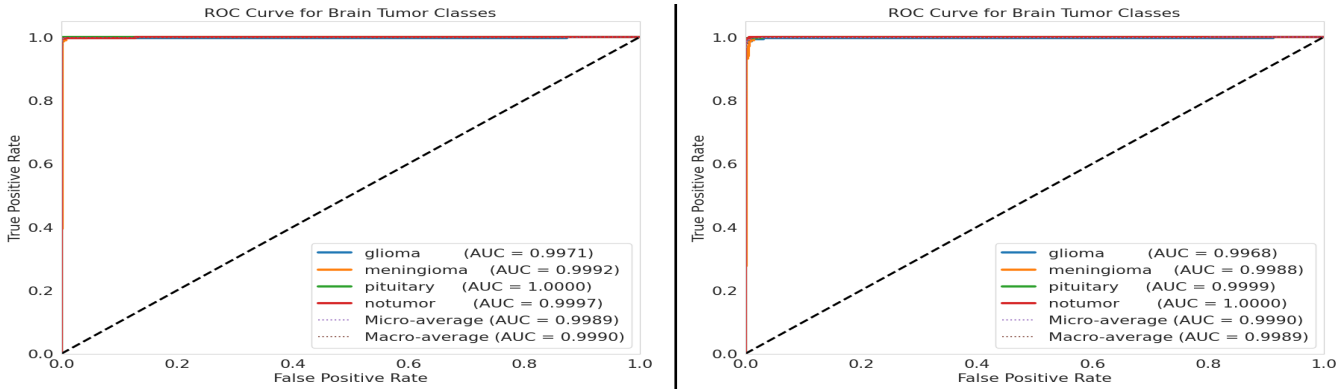
- The CEGA algorithm's performance is highly sensitive to the defined hyperparameter search space. Inadequate ranges can prevent finding optimal solutions or lead to inefficient exploration of irrelevant configurations.

Despite these limitations, the CEGA algorithm consistently outperforms several well-established and state-of-the-art algorithms in terms of both efficiency and optimization quality.

IX. CONCLUSION AND FUTURE WORK

Our study leverages transfer learning to fine-tune a reliable model that automatically classifies MRI scans into four distinct classes: gliomas, meningiomas, pituitary adenomas, and no tumors. The pre-trained EfficientNetB3 and DenseNet121 models, optimized with the CEGA algorithm, were effectively used to classify features extracted from brain images. The CEGA algorithm, a co-evolutionary variant of the Genetic Algorithm, addresses the original GA's drawbacks by enabling the parallel evolution of distinct populations, thereby increasing diversity, enhancing solution quality, and reducing the risk of premature convergence. We employed the CEGA algorithm to address global optimization challenges and to fine-tune the hyperparameters of the EfficientNetB3 and DenseNet121 models for brain tumor classification. The proposed CEGA-EfficientNetB3 model demonstrated excellent performance across all evaluation metrics when using the RMSprop optimizer, while the CEGA-DenseNet121 model showed strong performance with the Adadelta optimizer. The top-performing model, CEGA-EfficientNetB3, achieved the highest test accuracy among all models in this study. Our findings demonstrate the promise and efficiency of our technique, showcasing how transfer learning can expedite brain tumor diagnosis via MRI analysis. Furthermore, our study highlights the critical role of hyperparameter optimization in CNN models for medical diagnostic applications. Through precise tuning of the model's hyperparameters, we achieved substantial improvements in accuracy.

In future work, we plan to test additional pre-trained models alongside the CEGA algorithm for brain tumor classification. Additionally, we will focus on implementing advanced deep learning techniques, particularly Vision Transformers (ViTs). Adapting ViTs for 3D MRI data and exploring multi-modal fusion could potentially enhance classification accuracy, especially for rare tumor types.



(a) EfficientNetB3 (b) DenseNet121

FIGURE 9: ROC curves for the proposed models.

TABLE 8: Comparison of the optimized CEGA-EfficientNetB3 and CEGA-DenseNet121 models with leading-edge models based on classification accuracy.

Reference	Model	Dataset	Classification Type	Data Augmentation	Accuracy (%)
[46]	CNN-SVM	Figshare Harvard	Multi-classification (3 classes)	No	95.82 98.7
[47]	hidden Markov- model (HMM)	CE-MRI	Multi-classification (3 classes)	No	96.88
[48]	DenseNet-121 EfficientNetV2-M	Kaggle	Multi-classification (4 classes)	Yes	97.88 98.01
[49]	MobileNetV3	"SBE-SMU, BTC, and NINS 2022" combination	Multi-classification (4 classes)	Yes	91
[30]	PSO-KELM with Inception V3 and DenseNet201	Kaggle	Multi-classification (4 classes)	No	98.21
[50]	Proposed TL-CNN	"Figshare, BrH35, and SARTAJ" combination	Multi-classification (3 classes)	Yes	95.75
[1]	EfficientNetB0-SVM CNN-KNN	"Figshare, BrH35, and SARTAJ" combination	Multi-classification (4 classes)	No No	97.93 97.15
[51]	Generic CNN	"Figshare, BrH35, and SARTAJ" combination	Multi-classification (4 classes)	Yes	81.05
[52]	ResNet50	"Figshare, BrH35, and SARTAJ" combination	Multi-classification (4 classes)	Yes	99.00
Our proposed model	**CEGA- EfficientNetB3 CEGA- DenseNet121**	"Figshare, BrH35, and SARTAJ" combination	Multi-classification (4 classes)	**No**	**99.39 99.01**

REFERENCES

[1] M. Celik and O. Inik, "Development of hybrid models based on deep learning and optimized machine learning algorithms for brain tumor multi-classification," *Expert Systems with Applications*, vol. 238, p. 122159, 2024.

[2] World Health Organization, "Cancer," 2020, accessed: Dec. 13, 2023. [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/cancer

[3] Cancer Research UK, "Cancer research uk," 2024. [Online]. Available: https://www.cancerresearchuk.org/

[4] N. A. O. Bush, S. M. Chang, and M. S. Berger, "Current and future strategies for treatment of glioma," *Neurosurgical review*, vol. 40, pp. 1–14, 2017.

[5] D. N. Louis, A. Perry, G. Reifenberger, A. Von Deimling, D. Figarella-Branger, W. K. Cavenee, H. Ohgaki, O. D. Wiestler, P. Kleihues, and D. W. Ellison, "The 2016 world health organization classification of tumors of the central nervous system: a summary," *Acta neuropathologica*, vol. 131, pp. 803–820, 2016.

[6] A. Behin, K. Hoang-Xuan, A. F. Carpentier, and J.-Y. Delattre, "Primary brain tumours in adults," *The Lancet*, vol. 361, no. 9354, pp. 323–331, 2003.

[7] T. Rahman and M. S. Islam, "Mri brain tumor detection and classification using parallel deep convolutional neural networks," *Measurement: Sensors*, vol. 26, p. 100694, 2023.

[8] A. K. Sharma, A. Nandal, A. Dhaka, L. Zhou, A. Alhudhaif, F. Alenezi, and K. Polat, "Brain tumor classification using the modified resnet50 model

based on transfer learning," *Biomedical Signal Processing and Control*, vol. 86, p. 105299, 2023.

[9] D. Stoyanov, Z. Taylor, S. M. Kia, I. Oguz, M. Reyes, A. Martel, L. Maier-Hein, A. F. Marquand, E. Duchesnay, T. Löfstedt *et al.*, *Understanding and Interpreting Machine Learning in Medical Image Computing Applications: First International Workshops, MLCN 2018, DLF 2018, and IMIMIC 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 16-20, 2018, Proceedings*. Springer, 2018, vol. 11038.

[10] I. H. Sarker, "Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions," *SN computer science*, vol. 2, no. 6, p. 420, 2021.

[11] J. Jiang, Y. Shu, J. Wang, and M. Long, "Transferability in deep learning: A survey," *arXiv preprint arXiv:2201.05867*, 2022.

[12] Z. Zhao, L. Alzubaidi, J. Zhang, Y. Duan, and Y. Gu, "A comparison review of transfer learning and self-supervised learning: Definitions, applications, advantages and limitations," *Expert Systems with Applications*, p. 122807, 2023.

[13] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, pp. 281–305, 2012.

[14] R. Mohakud and R. Dash, "Survey on hyperparameter optimization using nature-inspired algorithm of deep convolution neural network," in *Intelligent and Cloud Computing: Proceedings of ICICC 2019, Volume 1*. Springer, 2021, pp. 737–744.

[15] D. Ezzat, A. E. Hassanien, and H. A. Ella, "An optimized deep learning architecture for the diagnosis of covid-19 disease based on gravitational search optimization," *Applied Soft Computing*, vol. 98, p. 106742, 2021.

[16] B. Chen, L. Zhang, H. Chen, K. Liang, and X. Chen, "A novel extended kalman filter with support vector machine based method for the automatic diagnosis and segmentation of brain tumors," *Computer Methods and Programs in Biomedicine*, vol. 200, p. 105797, 2021.

[17] S. A. A. Ismael, A. Mohammed, and H. Hefny, "An enhanced deep learning approach for brain cancer mri images classification using residual networks," *Artificial intelligence in medicine*, vol. 102, p. 101779, 2020.

[18] M. Toğaçar, Z. Cömert, and B. Ergen, "Classification of brain mri using hyper column technique with convolutional neural network and feature selection method," *Expert Systems with Applications*, vol. 149, p. 113274, 2020.

[19] H. M. Rai, K. Chatterjee, and S. Dashkevich, "Automatic and accurate abnormality detection from brain mr images using a novel hybrid unetresnext-50 deep cnn model," *Biomedical Signal Processing and Control*, vol. 66, p. 102477, 2021.

[20] N. Noreen, S. Palaniappan, A. Qayyum, I. Ahmad, M. Imran, and M. Shoaib, "A deep learning model based on concatenation approach for the diagnosis of brain tumor," *IEEE access*, vol. 8, pp. 55 135–55 144, 2020.

[21] A. Rehman, S. Naz, M. I. Razzak, F. Akram, and M. Imran, "A deep learning-based framework for automatic brain tumors classification using transfer learning," *Circuits, Systems, and Signal Processing*, vol. 39, no. 2, pp. 757–775, 2020.

[22] Ö. Polat and C. Güngen, "Classification of brain tumors from mr images using deep transfer learning," *The Journal of Supercomputing*, vol. 77, no. 7, pp. 7236–7252, 2021.

[23] A. S. Musallam, A. S. Sherif, and M. K. Hussein, "A new convolutional neural network architecture for automatic detection of brain tumors in magnetic resonance imaging images," *IEEE access*, vol. 10, pp. 2775–2782, 2022.

[24] A. U. Haq, J. P. Li, B. L. Y. Agbley, A. Khan, I. Khan, M. I. Uddin, and S. Khan, "Iimfcbm: Intelligent integrated model for feature extraction and classification of brain tumors using mri clinical imaging data in iot-healthcare," *IEEE Journal of Biomedical and Health Informatics*, vol. 26, no. 10, pp. 5004–5012, 2022.

[25] S. Hossain, A. Chakrabarty, T. R. Gadekallu, M. Alazab, and M. J. Piran, "Vision transformers, ensemble model, and transfer learning leveraging explainable ai for brain tumor detection and classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 28, no. 3, pp. 1261–1272, 2023.

[26] P. Tejashwini, J. Thriveni, and K. Venugopal, "Ebt deep net: Ensemble brain tumor deep net for multi-classification of brain tumor in mr images," *Biomedical Signal Processing and Control*, vol. 95, p. 106312, 2024.

[27] S. Z. Kurdi, M. H. Ali, M. M. Jaber, T. Saba, A. Rehman, and R. Damaševičius, "Brain tumor classification using meta-heuristic optimized convolutional neural networks," *Journal of Personalized Medicine*, vol. 13, no. 2, p. 181, 2023.

[28] R. Preetha, M. J. P. Priyadarsini, and J. S. Nisha, "Automated brain tumor detection from magnetic resonance images using fine-tuned efficientnet-b4 convolutional neural network," *IEEE Access*, vol. 12, p. 112181–112195, 2024.

[29] A. A. Asiri, A. Shaf, T. Ali, M. Aamir, M. Irfan, and S. Alqahtani, "Enhancing brain tumor diagnosis: an optimized cnn hyperparameter model for improved accuracy and reliability," *PeerJ Computer Science*, vol. 10, p. e1878, 2024.

[30] B. Sandhiya and S. K. S. Raja, "Deep learning and optimized learning machine for brain tumor classification," *Biomedical Signal Processing and Control*, vol. 89, p. 105778, 2024.

[31] G. D'Angelo, M. Ficco, and F. Palmieri, "Association rule-based malware classification using common subsequences of api calls," *Applied Soft Computing*, vol. 105, p. 107234, 2021.

[32] A. M. Vincent and P. Jidesh, "An improved hyperparameter optimization framework for automl systems using evolutionary algorithms," *Scientific Reports*, vol. 13, no. 1, p. 4737, 2023.

[33] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee, "Basic enhancement strategies when using bayesian optimization for hyperparameter tuning of deep neural networks," *IEEE access*, vol. 8, pp. 52 588–52 608, 2020.

[34] P. R. Lorenzo, J. Nalepa, M. Kawulok, L. S. Ramos, and J. R. Pastor, "Particle swarm optimization for hyper-parameter selection in deep neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '17. ACM, Jul. 2017, p. 481–488.

[35] G. D'Angelo and F. Palmieri, "A co-evolutionary genetic algorithm for robust and balanced controller placement in software-defined networks," *Journal of Network and Computer Applications*, vol. 212, p. 103583, 2023.

[36] E. H. Houssein, M. M. Emam, and A. A. Ali, "An optimized deep learning architecture for breast cancer diagnosis based on improved marine predators algorithm," *Neural Computing and Applications*, vol. 34, no. 20, pp. 18 015–18 033, 2022.

[37] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[38] M. Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[39] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *arXiv preprint arXiv:1905.11946*, 2019.

[40] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[41] S. Mohsen, A. M. Ali, E.-S. M. El-Rabaie, A. ElKaseer, S. G. Scholz, and A. M. A. Hassan, "Brain tumor classification using hybrid single image super-resolution technique with resnext101_32× 8d and vgg19 pre-trained models," *IEEE Access*, vol. 11, pp. 55 582–55 595, 2023.

[42] Google, "Google colab," 2024, [Online; accessed September 15, 2024]. [Online]. Available: https://colab.research.google.com/#

[43] N. Masoud, "Brain tumor mri dataset," 2023, [Online; accessed September 15, 2024]. [Online]. Available: https://www.kaggle.com/datasets/masoudnickparvar/brain-tumor-mri-dataset/data

[44] V. Nguyen, "Bayesian optimization for accelerating hyper-parameter tuning," in *2019 IEEE second international conference on artificial intelligence and knowledge engineering (AIKE)*. IEEE, 2019, pp. 302–305.

[45] P. Singh, A. Manure, P. Singh, and A. Manure, "Neural networks and deep learning with tensorflow," *Learn TensorFlow 2.0: Implement Machine Learning and Deep Learning Models with Python*, pp. 53–74, 2020.

[46] S. Deepak and P. Ameer, "Automated categorization of brain tumor from mri using cnn features and svm," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 8, pp. 8357–8369, 2021.

[47] G. Li, J. Sun, Y. Song, J. Qu, Z. Zhu, and M. R. Khosravi, "Real-time classification of brain tumors in mri images with a convolutional operator-based hidden markov model," *Journal of Real-Time Image Processing*, vol. 18, no. 4, pp. 1–13, 2021.

[48] R. İncir and F. Bozkurt, "Improving brain tumor classification with combined convolutional neural networks and transfer learning," *Knowledge-Based Systems*, p. 111981, 2024.

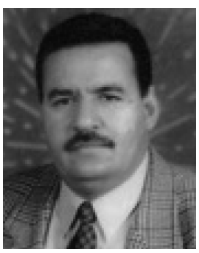
[49] A. F. Majeed, P. Salehpour, L. Farzinvash, and S. Pashazadeh, "Multi-class brain lesion classification using deep transfer learning with mobilenetv3," *IEEE Access*, vol. 12, p. 155295–155308, 2024.

[50] M. F. Alanazi, M. U. Ali, S. J. Hussain, A. Zafar, M. Mohatram, M. Irfan, R. AlRuwaili, M. Alruwaili, N. H. Ali, and A. M. Albarrak, "Brain tumor/mass classification framework using magnetic-resonance-imaging-

based isolated and developed transfer deep-learning model," *Sensors*, vol. 22, no. 1, p. 372, 2022.

[51] M. A. Gómez-Guzmán, L. Jiménez-Beristaín, E. E. García-Guerrero, O. R. López-Bonilla, U. J. Tamayo-Perez, J. J. Esqueda-Elizondo, K. Palomino-Vizcaino, and E. Inzunza-González, "Classifying brain tumors on magnetic resonance imaging by using convolutional neural networks," *Electronics*, vol. 12, no. 4, p. 955, 2023.

[52] A. Younis, L. Qiang, Z. Afzal, M. J. Adamu, H. B. Kawuwa, F. Hussain, and H. Hussain, "Abnormal brain tumors classification using resnet50 and its comprehensive evaluation," *IEEE Access*, vol. 12, p. 78843–78853, 2024.



ABDELMGEID A. ALI is currently a Professor in the Computer Science Department, Minia University, Minya, Egypt. He has published over 90 research papers in prestigious international journals and conference proceedings. He has supervised over 60 Ph.D. and M.Sc. students. His research interests include information retrieval, software engineering, image processing, data security, metaheuristics, the IoT, digital image steganography, data warehousing, computer vision, machine learning, and deep learning. He is a member of the International Journal of Information Theories and Applications (ITA).

MOHAMED T. HAMMAD received his B.Sc. degree in Computer Science from the Faculty of Computers and Information, Minia University, in 2017. He is currently a Teaching Assistant in the Department of Computer Science, Faculty of Computers and Artificial Intelligence, at the University of Sadat City, Egypt. His research interests include image processing, artificial intelligence, computer vision, machine learning, and deep learning.

HASSAN S. HASSAN received the Ph.D. degree in computer science. He currently works as a Lecturer with the Computer Science Department, Faculty of Computers and Information, Minia University, Egypt. He has published more than 40 scientific research articles in prestigious international journals, covering topics such as optimization, machine learning, image processing, and IoT applications. His research interests include wireless sensor networks, security, optimization, metaheuristics, computer vision, machine learning, and deep learning.