Diyala Journal of Engineering Sciences

Journal homepage: https://djes.info/index.php/djes

ISSN: 1999-8716 (Print); 2616-6909 (Online)

Enhanced Feature Pyramid Network Using a Binary-Encoded Genetic Algorithm for Brain Tumor Segmentation

Bahman Kadhim Ibrahim*, Serwan Ali Mohammed, Sagvan Ali Saleh

Department of Electrical and Computer Engineering, University of Duhok, Duhok, Iraq

ARTICLE INFO

Article history:
Received: 24/01/2026
Revised: 20/08/2026.
Accepted: 05/09/2026
Available online: 15/09/2026

Keywords:
Brain Tumor Segmentation
Feature Pyramid Network
Genetic Algorithm
Hyperparameter Optimization
Multimodal MRI

ABSTRACT

Manually segmentation of brain tumors in magnetic resonance imaging (MRI) is time-consuming and requires years of professional experience and clinical knowledge. Automatic brain tumor segmentation is essential for diagnosis, tumor treatment planning, and patient monitoring using MRI data. For this task, the hyperparameters of deep learning models are often tuned manually, which makes the process less effective and more compute-intensive. This study proposed an Enhanced Feature Pyramid Network (FPN) optimized by Binary Encoding Genetic Algorithm (GA) for automatic brain tumor segmentation. The proposed framework combined a multi-scale FPN backbone and an evolutionary optimization method in which the hyperparameters of the architecture and training process are represented as binary chromosomes and evolved according to the segmentation performance. The genetic algorithm automatically tuned the parameters of the network like the depth, filter size, dropout rate, learning rate, optimizer type, activation function, and loss function by maximizing the average Dice coefficient. The experimental results on the BraTS 2020 dataset showed that the GA-optimized FPN outperforms the baseline FPN with Dice scores of 0.9703, 0.9594 and 0.9408 in the Whole Tumor, Tumor Core, and Enhancing Tumor regions, respectively, with the number of trainable parameters reduced by 93.7%. The findings demonstrated that binary-encoded evolutionary optimization can be used to boost segmentation accuracy while yielding models with substantially reduced trainable parameters and training time, showing improving parameter efficiency.

1. INTRODUCTION

A brain tumor can form when cells in the brain or skull grow abnormally or unevenly. The tumor can be either benign or malignant [1].

Brain tumors are among the most lethal cancers and pose a serious threat to human life. The Global Burden of Disease (GBD) Study classifies brain and central nervous system cancers as some of the biggest causes of Years of Life Lost (YLLs) globally [2].

In Iraq, brain and central nervous system cancer ranks sixth. In 2022, there were 2116 cases of brain and central nervous system cancer, which accounted for 5.4% of all cancer cases diagnosed in Iraq [3].

Due to the high soft-tissue contrast and non-invasive nature of MRI, it is widely used for diagnosis and treatment planning of brain tumors [1]. Multimodal MRI images (T1, T2, T1ce, and FLAIR) are able to complement each other for identifying various subregions of tumors, including tumors with enhancement (ET), non-enhancing (NET), and edema (ED) [4, 5]. However, manual tumor segmentation by radiologists is time-consuming and varies from person to person [6].

Automated brain tumor segmentation has gained a tremendous boost in recent years with the progress of deep learning [7]. Machine learning and deep learning methods, like convolutional neural networks (CNN) and vision transformers (ViT), can accurately find the type, size, and location of the tumor [8, 9]. CNNs are suitable for automatically learning hierarchical and task-adaptive features directly from raw data, so there is no need for manual feature engineering as in traditional machine learning approaches [10].

Early CNN-based architectures, such as the Fully Convolutional Network (FCN), marked a major step forward in semantic segmentation by substituting fully connected layers with convolutional ones. This architecture enabled processing of input images of any size and the production of pixel-level output maps [11]. The U-Net architecture is a foundational model that

*Corresponding author's E-mail: bahman.kadhim@gmail.com | ORCID: 0009-0008-4025-0844
DOI: 10.24237/djes.2026.19307

uses skip connections within an encoder-decoder framework to preserve spatial information during segmentation. These skip connections combine the high-resolution features of the encoder and upsampled outputs of the decoder [12, 13]. A number of variants of U-Nets are presented, including 3D-UNet and UNet++, but they are prone to losing fine-grained details in the spatial dimension, when dealing with tumors of different sizes. For this, Feature Pyramid Networks (FPNs) were suggested, using lateral connections as well as top-down pathways to improve multi-scale representations of features [11, 14].

Deep learning segmentation models are effective, but they are very sensitive to hyperparameter selection. For this reason, optimization methods such as Genetic Algorithms (GA) have been studied for the automatic determination of the optimum values of the architecture and of the training parameters [15]. GAs have been applied in deep learning to enhance the structure and learning algorithms of neural networks. But their applicability with more complex segmentation architectures, particularly in medical imaging applications, is still limited.

In this work, we introduce a Feature Pyramid Network (FPN) optimized by a Binary Encoding Genetic Algorithm (GA) for automated segmentation of brain tumors from multimodal MRI images. The proposed approach employs FPN as the foundational model for segmentation to obtain multi-scale tumor representations, while GA is utilized to autonomously determine both the architectural and training hyperparameters. By adjusting the network configuration based on segmentation performance, the proposed framework improves model performance across different tumor regions. This approach eliminates the need for manual hyperparameter testing. We test the suggested method on the BraTS 2020 dataset to assess its segmentation performance using the Dice coefficient and Intersection over Union (IoU) metrics.

The main contributions of this study are as follows:

- A GA-optimized FPN-based framework for multi-scale brain tumor segmentation from multimodal MRI images.
- A binary-encoded GA–based optimization framework that jointly optimizes both architectural parameters (network depth, base filter size, dropout rate) and training hyperparameters (optimizer, learning rate, activation function, loss function).
- A fully automated evolutionary search process that eliminates manual hyperparameter tuning and identifies high-performing model configurations.
- A parameter-efficient GA-optimized FPN framework for brain tumor segmentation, achieving a 93.7% reduction in trainable parameters compared to the baseline, evaluated on the BraTS 2020 dataset.

The rest of the paper is organized as follows: Section 2 presents related work and explores existing research on brain tumor segmentation, multi-scale architectures, and optimization algorithms. In Section 3, the proposed methodology is explored, such as the dataset, model design, and GA optimization process. Section 4 shows the experimental results and analysis, and Section 5 concludes the paper and outlines future directions.

2. RELATED WORK

The segmentation of brain tumors from multimodal MRI images has been an important problem in recent years due to its critical role in diagnosis, treatment planning, and patient monitoring. Currently, the tumor regions are manually segmented from MRI images by radiologists, which is time-consuming, tedious, and requires a high level of experience [16]. To address these challenges, computer-aided image segmentation methods based on conventional image processing were investigated.

The initial segmentation techniques included thresholding, watershed segmentation, and region-growing methods [1, 7, 17]. These methods were aimed at extracting tumor areas from pixel intensity distributions and spatial similarity. They were simple to compute and easy to implement, but were very sensitive to noise, intensity variations, fuzzy tumor boundaries, and low contrast in MRI images, making them unsuitable for complex medical imaging applications.

In the next phase of development, traditional machine learning (ML) techniques were used, with segmentation posed as a voxel classification problem using handcrafted features [10]. In the field of tumor classification and segmentation, many methods have been used, such as Support Vector Machines (SVM), Random Forests (RF), Artificial Neural Networks (ANN) and K-Nearest Neighbor (KNN) classifiers [17, 18]. While these methods improved segmentation accuracy over classical image processing methods, they relied on careful feature engineering and descriptor selection, and therefore had limited transferability to other datasets and tumor characteristics. With the emergence of large-scale medical image datasets with annotations and GPUs, deep learning techniques became popular in medical image segmentation, and specifically Convolutional Neural Networks (CNNs) came into the spotlight. CNN-based models can be trained automatically to learn hierarchical and task-specific features directly from raw image data without needing to manually extract features [10]. Long et al. proposed a Fully Convolutional Network (FCN) that achieved dense pixel-wise semantic segmentation by using convolutional layers instead of fully connected layers [19]. The FCN model is the source of numerous medical image segmentation architectures used today.

Ronneberger et al. incorporated the FCN framework and proposed the U-Net, which was among the most influential architectures in biomedical image segmentation [20]. The U-Net is an encoder–decoder network with skip connections that help improve

localization accuracy and boundary refinement by passing high-resolution spatial information from the encoder to the decoder. Subsequent architectures, such as U-Net++ [21] and 3D U-Net [22], leveraged dense skip connections and volumetric feature learning for 3D MR images, respectively, further enhancing segmentation.

One of the difficulties in segmenting brain tumors is the large variability in tumor size, shape, and subregions among MRI modalities [16]. To solve this problem, Feature Pyramid Networks (FPNs) were proposed to improve the multi-scale feature representation [23]. FPN is designed to fuse high-level semantic information with fine-grained spatial details across multiple feature scales via a top-down pathway with lateral connections. In medical image segmentation, shape- and boundary-complex tumors have been successfully captured using FPN-based approaches [11].

Although CNN- and FPN-based models perform well, hyperparameter tuning remains crucial for achieving strong segmentation performance. Manual tuning of the architectural and training parameters is costly, and usually leads to suboptimal parameters [13]. Hence, evolutionary optimization has recently gained interest for automatically finding hyperparameters for deep-learning systems [15, 24].

GA-based techniques are used to find the best possible configuration through an evolutionary process, in which selection, crossover, and mutation occur. But their use in brain tumor segmentation in the context of GA has been rather limited, especially for optimizing both the architectural and training hyperparameters in a single framework.

The optimization-based segmentation methods vary in terms of the optimizer, the backbone and the search space. Holland et al. [15] combined a GA with U-Net to find optimal learning rate, number of epochs, and batch size parameters, it improved upon standard U-Net, but it optimized only the training hyperparameters and their model was tested on a dataset of ultrasound images of nerve segments instead of MRI images of brain tumors. Saifullah and Dreżewski [13] introduced PSO-UNet, an approach that utilized Particle Swarm Optimization to tune filter size, kernel size and learning rate. It improves tumor segmentation accuracy but was limited to a small set of hyperparameters.

Recently, Saifullah et al. [25] proposed a hybrid Particle Swarm Optimization (PSO) Genetic Algorithm (GA) U-Net (PSO-GA-U-Net), in which PSO tunes continuous parameters and GA searches for discrete parameters while maintaining diversity. The hybrid PSO-GA configuration was found to be more consistent in terms of performance improvement on several brain-tumor datasets. However, it is evaluated only on whole-tumor segmentation rather than separate ET, TC and WT subregions. The proposed framework, on the other hand, employs binary-encoded GA with an FPN as a backbone model, and in contrast, it performs separate segmentation of the ET, TC, and WT regions and optimizes its configuration using the average Dice score across these three tumor subregions. Table 1 summarizes representative studies on brain tumor segmentation and their limitations.

Table 1: Summary of representative studies in brain tumor segmentation.

Method	Main Contribution	Limitation
FCN [19]	Introduced fully convolutional semantic segmentation for dense pixel-wise prediction	Limited recovery of fine spatial details
U-Net [20]	Encoder–decoder framework with skip connections for accurate medical image segmentation	Limited multi-scale feature integration
U-Net++ [26]	Dense skip pathways for improved feature fusion and localization	Increased model complexity and computation
FPN [11]	Multi-scale feature pyramid fusion for brain tumor segmentation	Requires manual hyperparameter tuning
MM-BiFPN [27]	Bidirectional feature pyramid fusion for multimodal MRI segmentation	Higher computational cost
ResUNet+ [28]	Residual and attention-based segmentation architecture for improved feature extraction	Large parameter counts and complex architecture
Efficient nnU-Net [29]	Improved segmentation efficiency with automated configuration strategies	Limited multi-scale feature integration
SwinBTS [30]	Transformer-based global contextual modeling for multimodal brain tumor segmentation	High training complexity and memory usage
PSO-Optimized U-Net [13]	Automated hyperparameter optimization using Particle Swarm Optimization	Optimization is limited mainly to the U-Net framework and small set of hyperparameters
Hybrid Genetic U-Net [15]	Genetic algorithm–based optimization for medical image segmentation	Optimization limited to training hyperparameters only
PSO-GA-U-Net [25]	Hybrid PSO–GA framework that optimizes architectural and training hyperparameters of U-Net	Uses binary/whole-tumor segmentation rather than separate ET, TC, and WT subregions

3. METHODOLOGY

3.1 Overview

In this study, we propose a method to address the challenge of brain tumor segmentation from multimodal magnetic resonance imaging (MRI) scans by integrating a Feature Pyramid Network (FPN) with an evolutionary optimization algorithm. Specifically, a Genetic Algorithm (GA) was used to optimize the hyperparameters in order to minimize the manual trial-and-error to enhance the performance of the model across tumor subregions. The suggested framework consists of two phases: the first stage is hyperparameter tuning using GA to identify the optimal model architecture and training parameters among the possible configurations, and the second phase is used to estimate the segmentation performance of the best hyperparameter set from the first stage.

The overall workflow consists of the following steps:

- **Data preprocessing:** In this step, multimodal MRI images from the BraTS 2020 dataset are normalized and resized, and then the mask is reformulated into three subregions (ET, TC, and WT).
- **Model Implementation:** The FPN model is implemented to effectively address the challenge of multi-scale variation in brain tumors.
- **Binary-encoded GA for hyperparameter optimization:** Automatic search over encoded architecture parameters such as number of layers, dropout rates, number of filters, and learning parameters including optimizers, learning rates, activation functions, and loss functions.
- **Model training:** Full training of the optimized FPN configuration using early stopping, checkpointing, and learning monitoring.
- **Evaluation:** Using the Dice coefficient and Intersection over Union (IoU) on the BraTS 2020 dataset to check the quality of the segmentation.

A schematic of the proposed workflow is shown in Figure 1.

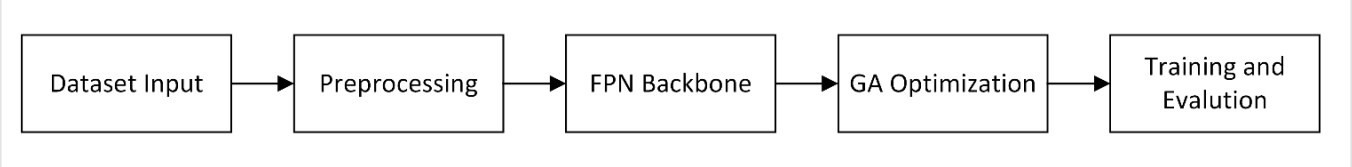

Figure 1. The proposed workflow

3.2 Dataset and Preprocessing

The BraTS 2020 dataset was used for this study. The BraTS 2020 Challenge focuses on evaluating state-of-the-art methods for multimodal MRI-based brain tumor segmentation. Its dataset was used in this study. BraTS 2020's main goal is to segment brain tumors that are naturally diverse (in terms of morphology, histology, and appearance), utilizing pre-operative MRI data gathered from different institutions. It has 369 volumes, and each case is a 3D MRI scan consisting of multiple slices, where each slice is a multi-channel MRI image and includes:

- **T1-weighted (T1):** provides anatomical detail
- **T1-contrast enhanced (T1ce or T1Gd):** tumor enhancement visualization
- **T2-weighted (T2):** highlights fluid and edema,
- **FLAIR:** suppresses cerebrospinal fluid for clearer edema visualization.

For each slice, voxel-wise ground-truth masks are annotated with three main tissue categories: enhancing, non-enhancing, and edema. The masks were manually annotated by expert raters. An example of the four MRI modalities and corresponding ground-truth segmentation is shown in Figure 2. The ground-truth masks for this study were reformulated into three binary channels, as shown in Figure 3.

To reduce computational cost, each MRI slice was resized to 128 × 128 pixels while maintaining four channels corresponding to the modalities. All pixel intensities were normalized to the [0, 1] range to ensure consistency across sequences and subjects.

A stratified data split was performed: 70% for training, 15% for validation, and 15% for testing. This ensured a balanced distribution of tumor sizes and types across subsets. The dataset was partitioned into training, validation, and testing subsets at the patient/volume level before extracting the 2D slices so that all the slices from a patient are in the same subset and cannot cross the subsets simultaneously.

3.3 Feature Pyramid Network Architecture

Feature Pyramid Network (FPN), is an effective method to combine multi-scale semantic information from the encoder. The main idea of FPN is to build a feature pyramid of several different resolutions of the input image and then fuse them together to form a completer and more detailed image of the input. FPN was originally designed for object identification; it can be modified to handle complex geometries and varying tumor sizes in medical imaging applications such as brain tumor segmentation. To segment brain tumors, we use an FPN-based encoder–decoder network, which incorporates an FPN into a U-Net-based encoder-decoder backbone. This approach collects multi-scale contextual information from a set of receptive fields, providing more accurate and effective tumor boundary segmentation than existing approaches based on U-net and FCN.

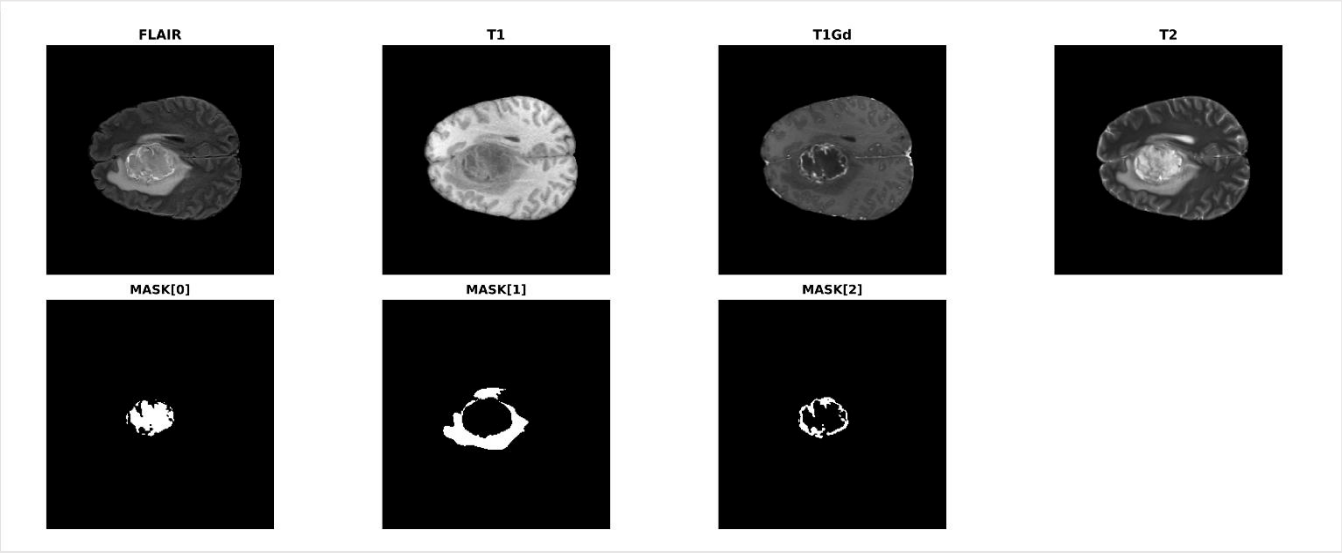

Figure 2. Example of multimodal MRI images from the BraTS 2020 dataset for a single subject, shows the FLAIR, T1, T1Gd and T2 modalities, along with the corresponding ground-truth segmentation mask, shows (from bottom left to right) the enhancing tumor, edema, and non-enhancing tumor.

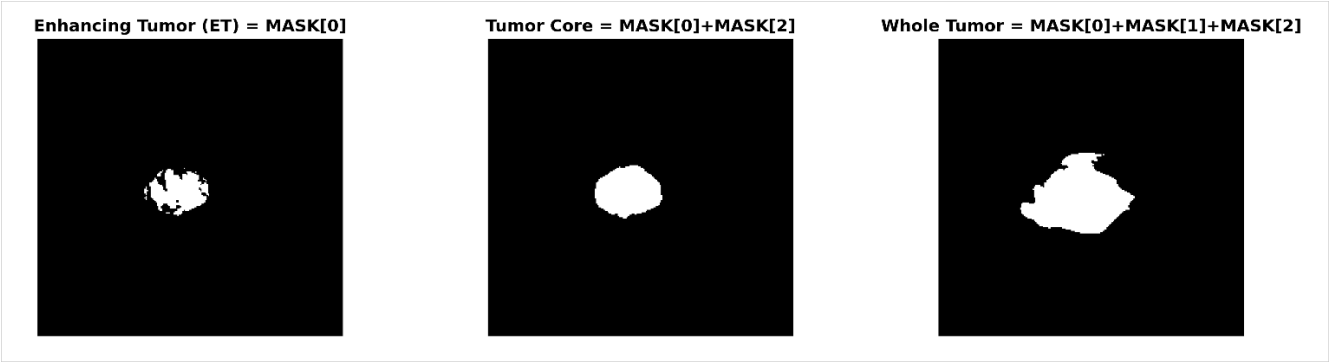

Figure 3. A sample of derived ground-truth segmentation masks provided for BraTS 2020 that shows the enhancing tumor (ET), tumor core (TC) and whole tumor (WT).

The FPN architecture for brain tumor segmentation is usually constructed upon a deep residual backbone (ResNet-50 or ResNet-101) and generates a four-level feature pyramid, denoted as P1, P2, P3 and P4. Each level of pyramid corresponds to a distinct receptive-field scale extracted from intermediate convolutional blocks.

This study uses the baseline architecture shown in Figure 4 as the segmentation backbone, which is based on FPN. The major novelty is the optimization process using a genetic algorithm, which is used to optimize both the architectural and training hyperparameters of the proposed segmentation model so that an efficient and lightweight model is obtained. Later, in Figure 7, the optimized configuration obtained from the evolutionary process is shown. As shown in Figure 4, the baseline FPN model adopted in this study is similar to the U-Net, employing an encoder–decoder structure with skip connections. However, unlike the U-Net, which directly concatenates encoder and decoder features at the same scale, FPN uses a top-down pyramid pathway with lateral connections to integrate multi-scale semantic information across different feature levels.

3.3.1 Encoder Path

The encoder is responsible for extracting meaningful features at different levels from the input MRI image gradually. It does this by convolving and downsampling it in several different stages. For each level, two convolutional layers are passed over the image and dropout is used to avoid overfitting with a slight rise in the dropout percentage along deeper layers. Then, max-pooling is used to decrease the feature map size while keeping critical information. Going deeper into the encoder, the number of filters grows (16, 32, 64, 128, 256) and the network gets more and more abstract and complex features. Finally, a bottleneck layer bridges the encoder and decoder by processing the most abstract feature representations before reconstruction.

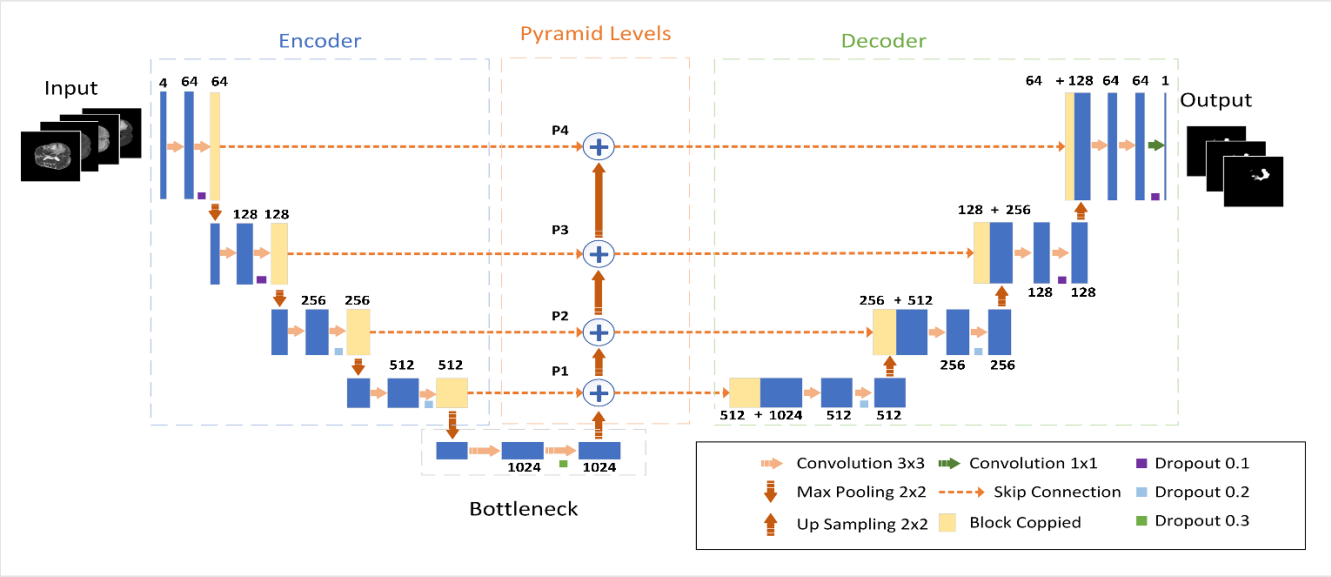

Figure 4. Baseline FPN-based architecture adopted in this study before GA optimization.

3.3.2 Feature Pyramid Integration

The FPN module is the most crucial component of the model architecture because it combines the semantic features of all the different stages in the encoder. This module is composed of several features from various levels of the encoder. Semantic features (from deeper layers) are upsampled and added to earlier layers of finer-grained and more detailed features. This produces a set of multi-scale features that include both the large-scale and small-scale details which are fed to the decoder.

3.3.3 Decoder Path

The decoder then upsamples the feature maps at each stage to maintain the original resolution and adds the skip connections with the encoder feature maps at that respective stage. This will help ensure fine detail, such as accurate tumor margins. Dropout and convolution layers refine the features at each step. Finally, a 1×1 convolution followed by a sigmoid function produces the output mask, yielding a probability for each pixel of belonging to the tumor or background.

3.4 Binary Encoding Genetic Algorithm Optimization

The choice of hyperparameters directly affects segmentation accuracy, generalization ability, and convergence speed, making it a crucial component of deep learning model performance. However, manually adjusting hyperparameters through trial and error is subjective and computationally costly. In order to solve this, encoded GA was implemented to perform automated hyperparameter optimization for both FPN architecture parameters, such as network depth (number of layers), dropout rates, and filter sizes, and training parameters, including learning rates, loss functions, activation functions, and the type of optimizers used in brain tumor segmentation. Natural selection provides the inspiration for the GA, an evolutionary computation technique. In order to optimize a fitness function, it evolves a population of candidate solutions, in this case, various training parameters and model configurations, over a number of generations. Here, the fitness function is computed as the mean Dice coefficient across the Whole Tumor (WT), Tumor Core (TC), and Enhancing Tumor (ET) regions following brief training sessions. The Dice coefficient was chosen as the main metric of performance for final segmentation across the ET, TC, and WT regions as it is the most commonly used metrics in this study to assess segmentation accuracy. Adopting the same metric as the GA fitness function in the optimization process and the final assessment ensures consistency between the metric used to select the chromosomes and the performance metric that is reported. Furthermore, the Dice scores in the three tumor regions are averaged to prevent picking a configuration that works well for one tumor region but poorly for the others. Thus, the fitness value of each chromosome was determined as the mean Dice score from the validation set. For the final evaluation of the model, IoU was used as an additional overlap measure. Dice and IoU are metrics that are closely related to regional overlap, while Hausdorff distance measures how far apart the boundaries are, which is an evaluation metric that is different. Since the aim of the GA was to maximise the overlap of regions, rather than minimise the distance between regions, the average Dice coefficient was taken as the primary fitness function for this study. This choice was also tested empirically against the mean IoU criterion under identical search conditions. Section 3.5.3 and reports the comparison. The mechanism and workflow of the proposed approach are shown in Figure 5. As the figure shows, it has two main phases: Genetic Algorithm Implementation and Evaluation of the model on the top-performance set of hyperparameters.

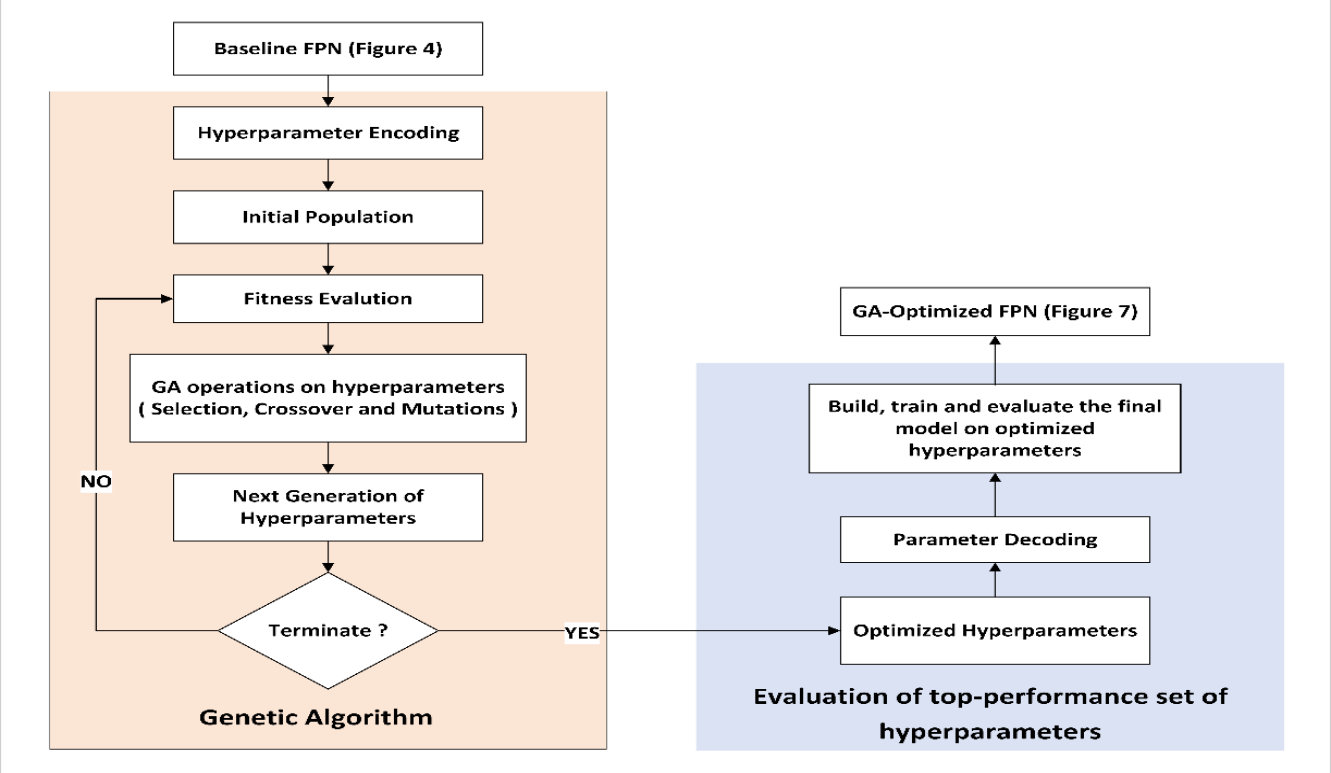

Figure 5. Workflow and optimization mechanism of the proposed binary-encoded GA–optimized FPN framework.

3.4.1 Genetic Algorithm Implementation

This phase is called the evolutionary search stage, in which the GA automatically tries different FPN architectural configurations and training setups to find the optimal set of hyperparameters within a reasonable amount of time. First, parameters are encoded in binary format, with each bit sequence representing a specific value for both the model structure and training hyperparameters. The complete chromosome structure used in the GA encoding scheme is illustrated in Figure 6, where each colored block represents a group of bits corresponding to a particular hyperparameter.

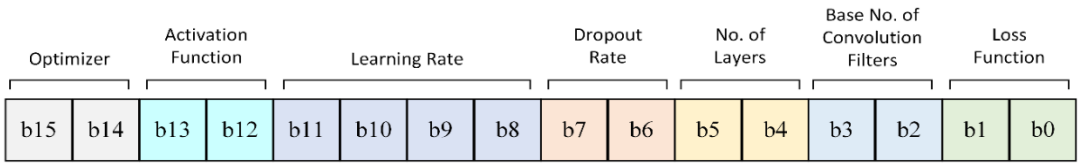

Figure 6. Binary chromosome structure used for genetic algorithm–based hyperparameter encoding

This graphical representation illustrates how the whole chromosome string is obtained by combining all the parameters listed in Table 2. Binary codes were duplicated, since some hyperparameters have fewer than four different values, but are encoded in two bits so that all of them have the same bit-length representation. This encoding scheme was adopted to preserve a fixed, uniform chromosome length across hyperparameter categories with unequal cardinality. We know that this duplication means that the duplicated values will be sampled more often than others in the initial population and mutation process (e.g., Adam, ReLU, depth = 3, and Binary Cross-Entropy). Subsequent generations, however, are determined mostly by selection, crossover, and elitism based on fitness (segmentation performance), instead of encoding frequency, so convergence to a final configuration is a consequence of fitness, not encoding frequency.

An initial population of chromosomes is randomly generated, with each chromosome corresponding to a unique configuration. For each individual in the population, the encoded values are decoded to construct an FPN model using the specific parameters. After that, each model is trained for a few epochs using the training set to reduce computational cost. After training, the model's performance is evaluated using the average Dice coefficient across the enhancing tumor (ET), tumor core (TC), and whole tumor (WT) regions.

After testing each configuration, a score is assigned to it, which is known as the fitness value. This fitness value indicates how each configuration performs in the

segmentation task. The population is then evolved toward better solutions by the GA operations such as selection, crossover, mutation, and elitism, which are described as follows:

- **Selection:** During the selection phase, individuals with higher fitness values corresponding to better segmentation performance are assigned a greater probability of being selected as parents for the next generation. This preferential selection strategy ensures that well-performing solutions have a stronger influence on the evolutionary process.
- **Crossover:** In crossover operation, a part of the parents' bit strings is swapped with a certain probability so that the offspring inherit characteristics of both parents.
- **Mutation:** At the final step, mutation randomly flips a bit at a specified mutation rate to maintain genetic diversity and exploration of the search space.
- **Elitism:** Elitism is also used in each generation to keep the best performing individuals from being lost during the crossover and mutation processes and pass them on to the next generation [31].

This approach gradually guides the population toward the most efficient configurations by continuously improving it over many generations of evolution. The best configuration is chosen from the individuals with the highest fitness values. Algorithm 1 shows the pseudocode for a GA for hyperparameter optimization.

Table 2: Binary encoding scheme of hyperparameters used in the GA for optimization process (i.e., a chromosome "0010111101101001" represents the configuration {optimizer: Adam, activation: Leaky ReLU, learning rate: 3×10^{-6}, dropout: 0.3, layers: 5, base filters: 64, loss function: Dice}).

Category	Hyperparametr	Encoding (Bits)	Search Space / Possible Values
Training	Optimizer	2	**00:** Adam, **01:** SGD, **10:** RMSprop, **11:** Adam
Training	Activation Function	2	**00:** ReLU, **01:** ELU, **10:** Leaky, **11:** ReLU
Training	Learning Rate	4	16 discrete values logarithmically spaced in the range from 3 × 10⁻3 to 3 × 10⁻6
Architectural	Dropout Rate	2	**00**: 0.0, **01**: 0.1, **10**: 0.3, **11**: 0.5
Architectural	Network Depth (No. of Layers)	2	**00:** 3, **01:** 4, **10:** 5, **11:** 3
Architectural	Base Filter Size	2	**00:** 16, **01:** 32, **10:** 64, **11:** 128
Training	Loss Function	2	**00:** Binary Cross-Entropy, **01:** Dice, **10:** combine (BCE + Dice), **11:** Binary Cross-Entropy

Algorithm 1: Genetic Algorithm for Hyperparameter Optimization

Input: Training **set** Dtrain, validation **set** Dval, generations G, population size N
Output: Optimized **configuration** C_{best}

```
Initialize population Pop ← {C_1, C_2, …, C_N} with random encoded hyperparameters
for generation = 1 to G do
    for each chromosome C in Pop do
        config ← decode chromosome C
        model ← build_FPN_model_from_config(config)
        Train model on Dtrain based on training parameters in config
        Evaluate model on Dval
        Assign fitness(C) ← Dice_avg
    end for
    select E best chromosomes (elitism)
    Initialize new_population with elite chromosomes
    while size(new_population) < N do
        select parent1 from population based on fitness
        select parent2 from population based on fitness
        Perform single-point crossover between paret1 and parent2 to produce offspring
        Apply bit-flip mutation on offspring with mutation rate
        Add offspring to new_population
    end while
    Pop ← updated population
end for
Evaluate the final population
C_best ← chromosome with highest fitness
return C_best as the optimized configuration for full training
```

3.4.2 Evaluation of the Best-performance Hyperparameters Set

In this phase, the corresponding optimized model is retrained from scratch using the entire training dataset for full epochs after the ideal configuration has been determined through the optimization process. Early stopping and checkpoint callbacks are applied to ensure ideal convergence. To assess the trained model on test data, Dice score and Intersection over Union (IoU), known as the Jaccard Index, are used as evaluation metrics. The results obtained, as shown in Section 4, indicate that the GA-based optimization significantly enhanced segmentation accuracy and model robustness. Thus, this optimization mechanism offers a data-driven approach to achieving high performance without manual intervention by automating the FPN's architectural design and training configuration.

3.5 Experimental Setup

3.5.1 Genetic Algorithm Configuration

The GA was configured to automatically explore a wide search space for both architectural and training hyperparameters of the FPN model.

During genetic optimization, each candidate model was trained using a short proxy schedule of six epochs, with approximately 15% of the available training batches processed per epoch, and subsequently evaluated on the validation subset to compute its fitness value. This reduced training regime significantly lowered the computational cost of exploring diverse hyperparameter configurations while preserving a reliable relative estimation of model quality. To ensure scalability and fairness for evaluations, the number of training and validation steps per epoch was set to a fixed fraction of the total number of batches in the datasets. Table 3 summarizes the details of the GA process configuration.

Table 3: GA configuration used for hyperparameter optimization

Parameter	Value / Setting	Description
Population size	6 Individuals	Number of model configurations per generation
Generations	20	Number of evolutionary iterations
Total candidate evaluations	120	Total candidate-evaluation runs across 20 generations (6 × 20)
Average candidate evaluation time	4.48 min	Average time required for model construction, proxy training, proxy validation and fitness calculation for one candidate.
Total GA execution time	568 min (9.47 h)	Total wall clock time; 538 min of candidate evaluations and 30 min of overhead for selecting, crossing over, mutating, managing generations and searching other functionalities.
Chromosome length	16 bits	Total bits representing all encoded hyperparameters
Selection method	Roulette wheel selection	Probabilistic selection favors higher fitness
Crossover type	Single-point crossover	Recombination of parent chromosomes
Crossover rate	0.7	Probability of crossover between parents
Mutation type	Bit flip	Random alteration of bit values
Mutation rate	0.0625 per bit	Probability of flipping each bit, defined as (1/L), where (L=16) is the chromosome length
Fitness function	Average Dice coefficient (ET, TC, WT)	Measures how well each configuration performs
Termination criteria	Fixed number of generations (20)	Evolution stops after 20 iterations
Solution found	Highest average Dice score	Defines the best optimum configuration

3.5.2 Training and Evaluation Settings

All models were built and trained in Google Colab (cloud service) to accelerate training. The input data included 128×128 image patches and four MRI channels of T1, T1ce, T2 and FLAIR. The number of batch size used to train the models is 48 which gave a good balance between memory usage and stable gradient updates. In order to ensure consistency and reproducibility in the experiments, the same fixed random seed (42) was applied to all runs and the dataset was split uniformly for all experiments:

- Training set: 70%
- Validation set: 15%
- Testing set: 15%

The model's performance was measured using two main evaluation metrics: the Dice coefficient and the Intersection over Union (IoU). Both were computed separately for the three tumor regions — the enhancing tumor (ET), tumor core (TC), and whole tumor (WT). The two metrics are defined as:

$$Dice = \frac{2|A \cap B|}{|A| + |B|} \quad (1)$$

$$IoU = \frac{|A \cap B|}{|A \cup B|} \quad (2)$$

where A is the predicted segmentation mask, and B is the ground-truth mask. These metrics were selected

because they are well-known in medical image segmentation and provide a good sense of the overlap between the predicted regions and the true tumor regions.

3.5.3 Fitness Function Selection and Validation

For ranking candidate chromosomes during the genetic search, we used the average Dice coefficient in the three regions (ET, TC, and WT) as shown in Table 3. The entire genetic search was conducted again, keeping the mean Intersection over Union (IoU) as the fitness function in support of this choice. For this purpose, all other settings that reported in Table 3 remain the same, including population size, number of generations, encoding scheme, selection method, crossover and mutation operators, data splitting, random seed and the training budget allocated to each solution candidate. Two searches were performed in the same computational configurations, as outlined in Section 3.5.4. Equations (1) and (2) suggest that the two measures are deterministically related for a given binary segmentation:

$$Dice = \frac{2.IoU}{1 + IoU}, \qquad IoU = \frac{Dice}{2 - Dice} \qquad (3)$$

Because this mapping is monotonic, IoU is always numerically smaller than Dice for the same segmentation, the two fitness values cannot be compared in their natural units. Both were then transformed using Equation (3) to a common scale. The result is given in Table 4.

After both types of searches were brought to the same scale, the Dice-driven search found a better individual than the IoU-driven search. The best chromosome value of it is 0.6954, compared to 0.6561 for the IoU-driven search. The difference in Dice terms is 0.0393 and in IoU terms is 0.0448, which is approximately 6% difference in relative terms. The average population fitness value was almost identical under both criteria, with a Dice value of 0.6407 and 0.6414 respectively. With their monotonic relationship, they therefore exert comparable selection pressure on the population as a whole, but the Dice criterion found a better best chromosome in the same number of evaluations.

There are two caveats with the comparison. First, the fitness value is the mean of three regional scores as compared to Equation (3) which is exactly for a single segmentation. The mapping is concave: This means that mapping an aggregate value result in a slightly higher value than mapping each region individually and then averaging the results. The IoU driven search value is thus an optimistic value reported and the comparison is conservative with respect to Dice. Second, each criterion was optimized separately in a single genetic search and evolutionary optimization is random. The result is corroborating the hypothesis that Dice is at the same level or better than IoU at the current budget, instead of a blanket statement that one of the measures is better than the other.

Table 4: Comparison of candidate fitness criteria used to drive the genetic search. Native values are given in the units of the criterion used. Values converted in accordance with Equation (3) can be compared on a like-for-like basis.

Fitness criterion	Best Fitness (native)	Average Fitness (native)	Best fitness as Dice	Best fitness as IoU	Average fitness as Dice
Average Dice	0.6954	0.6407	0.6954	0.5330	0.6407
Average IoU	0.4882	0.4721	0.6561	0.4882	0.6414

3.5.4 Hardware and Computational Environment

All experiments were conducted on Google Colab Pro with T4 GPU. For the GA, baseline FPN, and the final optimized model the same environment was used. So, all the results reported were measured under identical hardware and software configuration settings. The GPU greatly shortened the training time and helped process large MRI datasets and computationally heavy segmentation models. The deep learning frameworks for developing the models were TensorFlow 2.15 and Keras 2.14, while the data preprocessing and evaluation were performed using Python 3.10. To have standardized dimensions for all experiments, the BraTS 2020 dataset was resized to 128 × 128 pixels across four MRI modalities. The hardware and software setting of this study are summarized in Table 5.

Table 5: Hardware and software configuration used in this study

Component	Specification
Hardware Accelerator	Google T4-GPU
GPU RAM	15 GB
System RAM	51 GB
Disk Storage	235.7 GB
Software Framework	TensorFlow 2.15 / Keras 2.14
Programming Language	Python 3.10
Operating Environment	Google Colab Pro
Dataset	BraTS 2020
Image Resolution	128 × 128 pixels (4 modalities)

4. RESULTS AND DISCUSSION

This section demonstrates and discusses the results of the proposed Genetic Algorithm–Optimized Feature Pyramid Network (GA-FPN). It starts with a description of the optimized configuration found automatically by the GA, and continues with a quantitative evaluation of the performance of the final model on the BraTS 2020 dataset. Comparisons with the other methods are provided, highlighting the improvements in the segmentation accuracy and parameter-based model compactness. Finally, results are discussed based on model behaviour, efficiency and clinical relevance.

4.1 GA-Optimized Configuration

After running the GA for 20 generations, the setup that yielded the best performance, based on the average Dice score on the validation set, was selected as the final training and architectural hyperparameters. The full list of parameters found by the GA is shown in Table 6. This setup achieved the highest fitness value among all the tested configurations. Figure 7 illustrates the optimized FPN architecture obtained after applying the GA optimization process to the baseline FPN framework shown in Figure 4. The figure shows the three-level pyramid design, where each stage has been tuned for the number of filters and dropout level found by the GA.

Table 6: Optimized training and architectural configuration selected by the GA

Parameter	Value / Setting	Description
Optimizer	Adam	Adaptive moment estimation optimizer
Activation Function	ELU	Nonlinear activation improves gradient flow
Learning Rate	0.0008	Step size controlling weight updates
Dropout Rate	0.3	Fraction of neurons randomly dropped during training
Number of Layers	3	Depth of the encoder-decoder pyramid
Base Filter Size	64	Number of filters in the first convolutional block
Loss Function	Combined (Dice Loss + Binary Cross Entropy)	Balances region overlap and voxel-wise accuracy

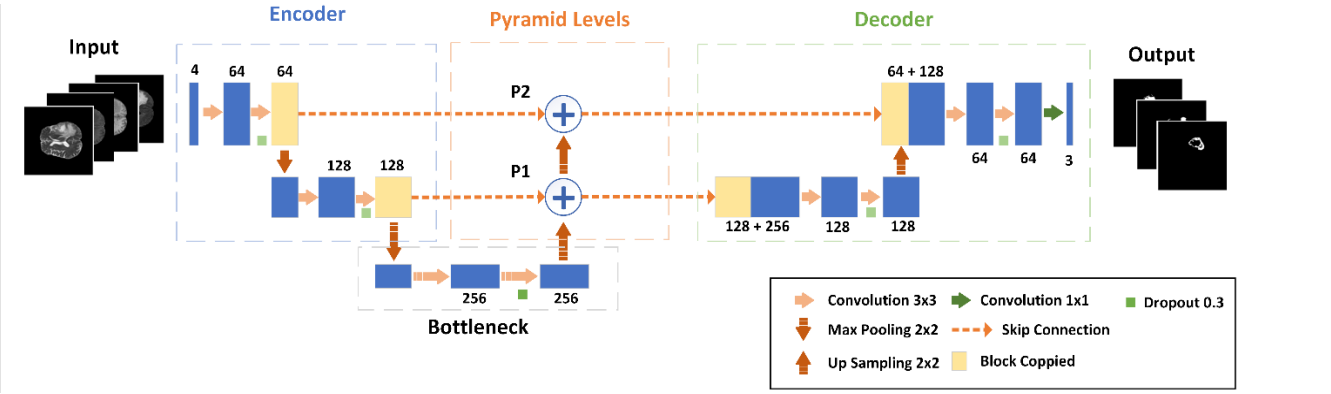

Figure 7. Network architecture of GA-Optimized FPN.

4.2 Final Training and Performance Evaluation

Using the selected parameters, the final model was retrained from scratch on the full training dataset. This stage was run for 50 epochs, with early stopping (patience = 10) and model checkpointing enabled to save weights whenever validation loss improved. These settings helped prevent overfitting and ensured that the model converged toward a stable and reliable solution. To verify the training stability of the optimized FPN model, the training and validation loss curves were monitored, as shown in Figure 8. The steady decline of both losses demonstrates good convergence without overfitting, confirming that the optimized hyperparameters and regularization settings were effective. The final model, therefore, represents the best combination of architectural and training settings discovered automatically by the evolutionary search, and it was later used for the complete performance evaluation on the BraTS 2020 dataset.

The proposed framework was assessed by comparing the ground truth of various tumor classes (WT, TC, and ET) with the segmentation mask predicted by the model. Dice Similarity Coefficient (DSC) and Intersection over Union (IoU) are key metrics for evaluating segmentation accuracy. The results achieved by the trained model are shown in Table 7. The high Dice and IoU scores indicate that the proposed model achieved accurate segmentation across all tumor subregions.

The framework yielded the highest Dice coefficient (97.03%) in the Whole Tumor (WT) region; followed by 95.94% in the Tumor Core (TC) region; and 94.08% in the Enhancing Tumor (ET) region and as shown in Table 7, the IoU values were 94.23%, 92.21% and 88.89% respectively. The ordering is more informative

when expressed as residual error. The Dice error increases from 2.97% for WT to 4.06% for TC to 5.92% for ET; the Dice error almost doubles from the largest to the smallest subregion with only 2.95 points difference between them. The impact is greater under IoU with a gap between WT and ET scores increasing from 2.95 points to 5.34 points, an increase in around 81%. This widening is inherent to the metrics rather than the model: IoU penalises the same boundary disagreement more heavily than Dice, and small structures have a higher ratio of boundary to interior voxels.

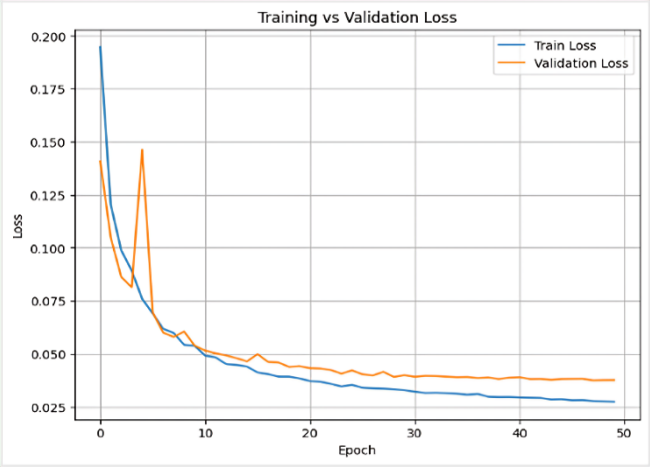

Figure 8. Training and validation loss curves of the GA-optimized FPN.

This is confirmed by the results published in Table 9. Across the compared methods, reported ET Dice ranges from 62.11% to 91.90%, a spread of 29.79 points with a standard deviation of 8.23, against a spread of 14.01 points and a standard deviation of 4.54 for TC. But, given its smaller size, the irregular borders, and class imbalance that occurs when the enhancement region is a small portion of the image, it is perhaps unsurprising that methods vary most for ET. The regional ordering here does not represent one specific ordering across this framework, but something common throughout the literature.

The proposed model shows a WT Dice of 97.03% and IoU of 94.23% whereas the literature gives a baseline FPN, with a Dice of 92% and an IoU of 86%. This literature-reported baseline is assessed using a different split of the dataset, a different preprocessing pipeline and a different computational environment, which is why this comparison is provided as indicative context. This is like the others in the literature base from Section 4.4 and not a direct quantitative improvement. In contrast, a direct comparison of the number of trainable parameters is provided in Table 8 for our re-implementation of the baseline FPN, which had 30.71 million trainable parameters, versus 1.93 million trainable parameters for the GA-optimized model (6.28% of the number of parameters in the baseline model). The number of parameters was reduced in this case without any specific penalty in the GA fitness function, suggesting that the increased depth of the base architecture was not crucial for accurate segmentation on this dataset. Since the GA was driven by mean validation Dice alone and carried no penalty on model size, the fact that the highest-scoring chromosome is also the smallest indicates that the additional depth of the baseline was not being converted into accuracy on this dataset.

The combined Dice and binary cross-entropy loss combines a region-level overlap term with a voxel-level term that is most significant for the imbalanced ET region; the three-level pyramid retains the multi-scale pathway of the large WT region in addition to the smaller TC and ET regions; the ELU activation function, Adam optimization and learning rate together stabilise convergence as shown in Figure 8; dropout rate of 0.3 provides stability for the convergence. These parameters were selected jointly as a single chromosome, so their individual contributions cannot be separated from the present results. A controlled single-parameter ablation would be required, and this is identified as a limitation in Section 4.4.

Table 7: Segmentation performance of the optimized model on BraTS 2020 dataset.

Metric	Whole Tumor (WT)	Tumor Core (TC)	Enhancing Tumor (ET)
Dice Index (DSC)	97.03%	95.94%	94.08%
Intersection over Union (IoU)	94.23%	92.21%	88.89%

4.3 Parameter-Based Complexity and Training-Time Analysis

Besides segmentation accuracy, the computational properties of the proposed framework were also assessed by comparing the baseline FPN with GA-optimized FPN. The comparison examines the number of parameters to be trained as an indicator of compactness of the model and the total training time measured in the same experimental conditions. As shown in Table 8, the number of trainable parameters decreased from 30.71 million to 1.93 million, or about 93.7% reduction, by using GA optimization. The training time also decreased from 1590 minutes for the baseline FPN to 1,019 minutes for the optimized FPN, which represent a 36% reduction. This improvement is largely due to the optimized model's architecture of shallower depth and its significantly smaller number of model parameters, thus requiring less computational power for training. As reported in Table 3, the GA performed 120 candidate-evaluation runs, with an average evaluation time of 4.48 min and a total execution time of 568 min (9.47 h). The 1,019 min

reported for the optimized FPN is only for the last model training and not for the GA search which is performed once. An inference-time efficiency was not measured in this study and is considered a future direction of research, as is the inference latency under controlled hardware conditions. The present comparison is limited to trainable-parameter count and training-time reduction, which are the dimensions evaluated under matched conditions in this study.

Table 8: Trainable-parameters and final-model training comparison between baseline and GA-optimized FPN.

Model	Role	Number of parameters (million)	Training Time (Min)
Baseline FPN	**Before GA optimization**	30.71	1,590
Optimized FPN	**After GA Optimization**	**1.93**	**1,019**

The total optimization and training time was 1,587 min, which is similar to the total training time for the baseline FPN of 1,590 min. Therefore, the total end-to-end computational cost was comparable to that of the baseline, with a reduction of only 0.18%. But, once the selected architecture is trained, the training does not need the GA search anymore, and it takes 35.9% less time than the baseline.

The parameter reduction in optimized model is mainly due to the architecture used by the GA. Therefore, the optimized network represents a three-level network without the deeper, high channel layers, and extra encoder–decoder operations in the baseline architecture. This means that there are fewer connections between convolutional filters, fewer feature transformations and fewer lateral connections.

The fitness function was average validation Dice, so the number of parameters was not included as a direct fitness term. For this reason, the reduction in the size of the model came out as a consequence of the choice of shallow and simple architecture. This finding suggests that the additional complexity of the baseline FPN was unnecessary for achieving effective segmentation on this dataset.

The total training times given in Table 8 directly compare the computational effort needed by the two models. However, the number of parameters does not alone determine training time or inference time. Other factors that affect runtime include feature map dimensions, operation type, memory access, batch size, software implementation and hardware utilization. Therefore, although a reduction in final-model training time was measured, no claim is made regarding inference-time improvement because inference latency was not evaluated under controlled conditions.

These two axes of comparison are opposite to the standard trade-off when put together. Optimized model requires only 64.1% of the final model training time and 6.28% of the trainable parameters of the baseline model, and achieves 5.03 points of improvement in WT Dice compared to the baseline model reported in [11]. Accuracies gained from pruning or by narrowing the track are generally forfeited for the compression, but in this case a narrowing just for compression brings accuracy. This reinforces the hypothesis that the baseline was not adjusted to meet this segmentation task size, but that it was raised above the required baseline for this task.

4.4 Comparison with Recent Methods

To evaluate the effectiveness of the proposed optimized FPN, its results were compared with several recent deep-learning-based methods for brain-tumor segmentation that employed the BraTS 2020 dataset or comparable multimodal MRI benchmarks. These methods include classical convolutional architectures, enhanced U-Net variants, feature-pyramid networks, transformer-based models, and optimization-based approaches based on particle swarm optimization and genetic algorithm.

Dice and IoU values reported in these studies were collected for the available tumor subregions (ET, TC, WT) to enable a broad and representative comparison of segmentation capability. Table 9 summarizes the quantitative results across these approaches. The results presented in the compared studies can only be treated as an indicative cross-study comparison because of differences in dataset versions, data partitioning, preprocessing, and evaluation procedures.

The reported results show that the GA-optimized FPN consistently outperforms the baseline FPN architecture in segmentation performance across three regions of interest (ROIs): WT, TC, and ET. The improvement is attributed to the GA's ability to automatically discover effective hyperparameter configurations for the architecture and training parameters, thereby improving multi-scale feature representation. The proposed framework showed excellent performance in the ET region, which is one of the most difficult tumor subregions due to its small size and irregular boundaries.

The margin of the proposed framework is narrower than the overall range of the reported results as compared to the other methods shown in Table 9. The difference is 1.25 Dice points (for WT vs. PSO-UNet), 2.84 points (for TC vs. ResUNet+), and 2.18 points (for ET vs. ResUNet+) when compared to the strongest comparator in each subregion respectively. The larger differences in Table 9 are to transformer-based methods, which do the poorest job in the ET region, giving IMS2Trans at 62.11% and SwinBTS at 77.36%. Transformer

architectures rely on large training sets to learn the spatial relationships that convolutional inductive bias provides directly, and ET is the subregion which provides the least number per case of positive voxels in which this is the case. The comparison with transformer models is not just about optimization, it is about the architectural families too; the proposed architecture is convolutional, with the same inductive bias.

The proposed framework is also compared with the optimization-based approaches to further illustrate its contribution. A filter size, kernel size and learning rate tuning method based on PSO-UNet [13] with a WT Dice score of 95.78% and a WT IoU of 91.94% was reported. PSO-GA-U-Net [25] reported the best WT Dice score of 94.06% and the best WT IoU of 88.81% using PSO and GA for continuous hyperparameters and discrete hyperparameters, respectively. The proposed GA-FPN, however, obtained the WT Dice score of 97.03% and WT IoU of 94.23% in comparison. Furthermore, these methods evaluated only whole-tumor segmentation performance, whereas the proposed framework separately considers three subregions of the tumor: ET, TC and WT, and optimizes the framework configuration based on the average Dice score from all three subregions.

These comparisons are meant to be used as indicative and not as direct comparisons. The studies compared differ in dataset version, in data partitioning, in data preprocessing, and in the evaluation procedure. Evaluation used in the present work is performed at the 2D slice level, which results in systematically higher overlap values than in some of the compared studies where the evaluation is performed at the 3D (volumetric) level, per patient. The differences mentioned above are not a controlled comparison, and therefore the position of the proposed framework in Table 9 should be interpreted accordingly. Within the present study, the evidence supports a narrower and better-founded claim. Optimization of the architectural and training hyperparameters together as a single chromosome yields a configuration that is 62.9% better at minimizing the residual WT Dice error compared to the baseline configuration with 6.28% of the trainable parameters; the regional ordering of the performance of this configuration mimics the difficulty trend observed across the published methods in Table 9. The results cannot be separated into the contributions of each of the individual hyperparameters, because the GA evaluates chromosomes as whole configurations, and it would be appropriate to use a controlled single parameter ablation around the selected chromosome, which is left for further research.

Table 9: Quantitative comparison of the proposed optimized FPN with recent brain-tumor segmentation methods.

Method	Dice Similarity Coefficient (DSC)			Intersection over Union (IoU)		
	ET	TC	WT	ET	TC	WT
Proposed GA-FPN	**94.08%**	**95.94%**	**97.03%**	88.89%	**92.21%**	**94.23%**
Baseline FPN [11]	-	-	92%	-	-	86%
MM-BiFPN [27]	77.95%	81.47%	83.58%	-	-	-
UNet [32]	-	-	71.39%	-	-	62.09%
Efficient nnU-Net [29]	79.20%	84.80%	91.20%	-	-	-
ResUNet+ [28]	91.90%	93.10%	92.80%	**91.24%**	92.12%	90.01%
RMU-Net [33]	83.26%	88.13%	91.35%	86.79%	88.17%	92.43%
DResU-Net [4]	80.04%	83.57%	86.60%	-	-	-
IMS2Trans [34]	62.11%	79.09%	87.33%	-	-	-
SwinBTS [30]	77.36%	80.30%	89.06%	-	-	-
PSO-UNet [13]	-	-	95.78%	-	-	91.94%
PSO-GA-U-Net [25]	-	-	94.06%	-	-	88.81%

4.5 Qualitative Results and Visualization

In addition to quantitative evaluation, qualitative visualization provides an intuitive understanding of the segmentation performance of the proposed optimized model. Figure 9 presents representative examples comparing the ground-truth tumor masks with the predicted segmentation maps for the three tumor subregions: Whole Tumor (WT), Tumor Core (TC), and Enhancing Tumor (ET). As shown in Figure 9, the optimized model accurately delineates tumor boundaries and subregions with high consistency across patients. The model effectively captures small ET regions and complex irregular boundaries of the TC without over-segmentation or leakage into healthy tissue. The predicted masks exhibit close spatial overlap with ground truth annotations, consistent with the high Dice and IoU values reported in Table 7. These qualitative observations further confirm the robustness and reliability of the proposed model.

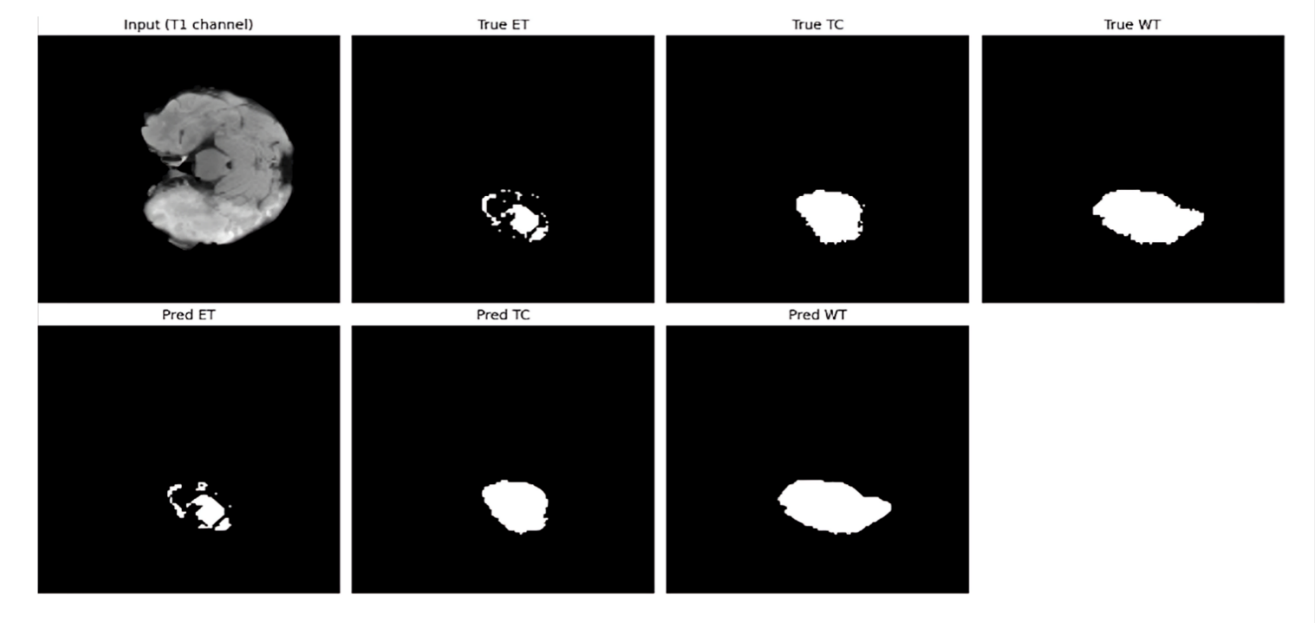

Figure 9. Visual comparison between ground truth and the optimized FPN segmentation results on sample BraTS 2020 cases. Each row shows (from left to right) the input FLAIR image, ground-truth segmentation, and the predicted output masks for WT, TC, and ET, respectively

5. CONCLUSION AND FUTURE DIRECTIONS

The aim of this study was to develop a GA-optimized framework for FPN-based brain tumor segmentation in multimodal MRI images. The proposed framework combines multi-scale feature extraction with automatic hyperparameter optimization using the GA to obtain optimal architecture and training parameters without manual tuning. Experimental results on the BraTS 2020 dataset showed that the proposed method provides accurate segmentation across the three regions of WT, TC, and ET with substantially lower model complexity than the FPN baseline.

The results show that evolutionary hyperparameter optimization is an effective approach for enhancing segmentation performance and generating models that are substantially more compact in trainable parameters and faster to train than the baseline. In addition, the qualitative outcomes showed good agreement between the predicted masks and the expert annotations across various tumor types and boundary complexities.

Although the proposed framework achieved good results, it relied solely on the BraTS 2020 dataset. While BraTS 2020 is a popular benchmark dataset for brain tumor segmentation, using additional open-source datasets, such as BraTS 2021, as well as multicenter clinical datasets with different MRI acquisition settings in future work, would help assess the robustness and generalization capability of the proposed approach. A possible extension is to incorporate attention mechanisms or adaptive feature selection within the feature pyramid to segment low-contrast and/or small tumor regions. Further research could also include optimizing the genetic algorithm to perform multi-objective optimization problems, with additional consideration of inference speed, memory usage, and segmentation accuracy. In addition, larger multi-center datasets and the addition of 3D modalities (or 2.5D–3D) to the proposed framework may further enhance robustness and generalization capability. Finally, evolutionary optimization, when used in conjunction with semi-supervised or self-supervised learning strategies, can potentially reduce the need for fully annotated medical datasets.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no competing interests to disclose.

DATA AVAILABILITY STATEMENT

The data used in this study is publicly available from the BraTS 2020 Challenge dataset: https://www.med.upenn.edu/cbica/brats2020/data.html

REFERENCES

[1] S. Krishnapriya and Y. Karuna, "A survey of deep learning for MRI brain tumor segmentation methods: Trends, challenges, and future directions," *Health and Technology,* vol. 13, no. 2, pp. 181-201, 2023, doi: 10.1007/s12553-023-00737-3.

[2] I. Ilic and M. Ilic, "International patterns and trends in the brain cancer incidence and mortality: An observational study based on the global burden of disease," *Heliyon,* vol. 9, no. 7, 2023, doi: 10.1016/j.heliyon.2023.e18222.

[3] M. o. H. Iraqi Cancer Board, Republic of Iraq, "CANCER REGISTRY OF IRAQ, ANNUAL REPORT 2023," 2023. [Online]. Available: https://storage.moh.gov.iq/2024/11/24/2024_11_24_12127028949_4299728097670824.pdf

[4] R. Raza, U. I. Bajwa, Y. Mehmood, M. W. Anwar, and M. H. Jamal, "dResU-Net: 3D deep residual U-Net based brain tumor segmentation from multimodal MRI," *Biomedical Signal Processing and Control,* vol. 79, p. 103861, 2023, doi: 10.1016/j.bspc.2022.103861.

[5] M. A. Abid and K. Munir, "A systematic review on deep learning implementation in brain tumor segmentation,

classification and prediction," *Multimedia Tools and Applications,* vol. 84, no. 31, pp. 38203-38242, 2025, doi: 10.1007/s11042-025-20706-4.

[6] S. Hashmi *et al.*, "Optimizing brain tumor segmentation with mednext: Brats 2024 ssa and pediatrics," *arXiv preprint arXiv:2411.15872,* 2024, doi: 10.48550/arXiv.2411.15872.

[7] J. Nodirov, A. B. Abdusalomov, and T. K. Whangbo, "Attention 3D U-Net with Multiple Skip Connections for Segmentation of Brain Tumor Images," *Sensors,* vol. 22, no. 17, p. 6501, 2022, doi: 10.3390/s22176501.

[8] K. Bibi *et al.*, "Artificial Intelligence–Based Approaches for Brain Tumor Segmentation in MRI: A Review," *NMR in Biomedicine,* vol. 38, no. 11, p. e70141, 2025, doi: 10.1002/nbm.70141.

[9] A. Hamza and R. Damaševičius, "Deep learning for brain tumor segmentation and classification: a systematic review of methods and trends," *Computers, materials and continua.,* vol. 86, no. 1, pp. 1-41, 2025, doi: 10.32604/cmc.2025.069721.

[10] T. Magadza and S. Viriri, "Deep Learning for Brain Tumor Segmentation: A Survey of State-of-the-Art," (in eng), *J Imaging,* vol. 7, no. 2, Jan 29 2021, doi: 10.3390/jimaging7020019.

[11] H. Sun *et al.*, "Brain tumor image segmentation based on improved FPN," *BMC Medical Imaging,* vol. 23, no. 1, p. 172, 2023, doi: 10.1186/s12880-023-01131-1.

[12] D. E. Cahall, G. Rasool, N. C. Bouaynaya, and H. M. Fathallah-Shaykh, "Dilated inception U-net (DIU-net) for brain tumor segmentation," *arXiv preprint arXiv:2108.06772,* 2021, doi: 10.48550/arXiv.2108.06772.

[13] S. Saifullah and R. Dreżewski, "Particle Swarm-Optimized U-Net Framework for Precise Multimodal Brain Tumor Segmentation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2025, pp. 323-326, doi: 10.1145/3712255.3726561.

[14] M. Hu, Y. Li, L. Fang, and S. Wang, "A2-FPN: Attention aggregation based feature pyramid network for instance segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 15343-15352, doi: 10.1109/CVPR46437.2021.01509.

[15] J.-O. Holland, Y. Djenouri, R. Laidi, and A. Yazidi, "Hybrid Genetic U-Net Algorithm for Medical Segmentation," in *ICAART (3)*, 2023, pp. 558-564, doi: 10.5220/0011703700003393.

[16] P. Wang and A. C. Chung, "Relax and focus on brain tumor segmentation," *Medical image analysis,* vol. 75, p. 102259, 2022, doi: 10.1016/j.media.2021.102259.

[17] Y. M. Mohammed, S. El Garouani, and I. Jellouli, "A survey of methods for brain tumor segmentation-based MRI images," *Journal of Computational Design and Engineering,* vol. 10, no. 1, pp. 266-293, 2023, doi: 10.1093/jcde/qwac141.

[18] F. Prinzi, T. Currieri, S. Gaglio, and S. Vitabile, "Shallow and deep learning classifiers in medical image analysis," *European radiology experimental,* vol. 8, no. 1, p. 26, 2024, doi: 10.1186/s41747-024-00428-2.

[19] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431-3440, doi: 10.1109/CVPR.2015.7298965.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, 2015: Springer, pp. 234-241, doi: 10.1007/978-3-319-24574-4_28.

[21] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE access,* vol. 9, pp. 82031-82057, 2021, doi: 10.1109/ACCESS.2021.3086020.

[22] U. Baid *et al.*, "A novel approach for fully automatic intra-tumor segmentation with 3D U-Net architecture for gliomas," *Frontiers in computational neuroscience,* vol. 14, p. 10, 2020, doi: 10.3389/fncom.2020.00010.

[23] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117-2125, doi: 10.1109/CVPR.2017.106.

[24] M. A. K. Raiaan *et al.*, "A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks," *Decision Analytics Journal,* vol. 11, p. 100470, 2024, doi: 10.1016/j.dajour.2024.100470.

[25] S. Saifullah, R. Dreżewski, A. Yudhana, R. Tanone, and A. P. Suryotomo, "A Hybrid Particle Swarm–Genetic Algorithm Framework for U-Net Hyperparameter Optimization in High-Precision Brain Tumor MRI Segmentation," *Applied Sciences,* vol. 16, no. 6, p. 3041, 2026, doi: 10.3390/app16063041.

[26] Z. Zhou, M. M. Rahman Siddiquee, N. Tajbakhsh, and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," in *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support: 4th International Workshop, DLMIA 2018, and 8th International Workshop, ML-CDS 2018, Held in Conjunction with MICCAI 2018, Granada, Spain, September 20, 2018, Proceedings 4*, 2018: Springer, pp. 3-11, doi: 10.1007/978-3-030-00889-5_1.

[27] N. S. Syazwany, J.-H. Nam, and S.-C. Lee, "MM-BiFPN: multi-modality fusion network with Bi-FPN for MRI brain tumor segmentation," *IEEE Access,* vol. 9, pp. 160708-160720, 2021, doi: 10.1109/ACCESS.2021.3132050.

[28] S. Metlek and H. Çetıner, "ResUNet+: A new convolutional and attention block-based approach for brain tumor segmentation," *IEEE Access,* vol. 11, pp. 69884-69902, 2023, doi: 10.1109/ACCESS.2023.3294179.

[29] T. Magadza and S. Viriri, "Efficient nnU-net for brain tumor segmentation," *IEEE Access,* vol. 11, pp. 126386-126397, 2023, doi: 10.1109/ACCESS.2023.3329517.

[30] Y. Jiang, Y. Zhang, X. Lin, J. Dong, T. Cheng, and J. Liang, "SwinBTS: A method for 3D multimodal brain tumor segmentation using swin transformer," *Brain sciences,* vol. 12, no. 6, p. 797, 2022, doi: 10.3390/brainsci12060797.

[31] R. Landa, D. Tovias-Alanis, and G. Toscano, "Optimization of Deep Neural Networks Using a Micro Genetic Algorithm," *AI,* vol. 5, no. 4, pp. 2651-2679, 2024, doi: 10.3390/ai5040127.

[32] F. D. Hernandez-Gutierrez *et al.*, "Brain Tumor Segmentation from Optimal MRI Slices Using a Lightweight U-Net," *Technologies,* vol. 12, no. 10, p. 183, 2024, doi: 10.3390/technologies12100183.

[33] M. U. Saeed *et al.*, "RMU-net: a novel residual mobile U-net model for brain tumor segmentation from MR images," *Electronics,* vol. 10, no. 16, p. 1962, 2021, doi: 10.3390/electronics10161962.

[34] D. Zhang, C. Wang, T. Chen, W. Chen, and Y. Shen, "Scalable Swin Transformer network for brain tumor segmentation from incomplete MRI modalities," *Artificial Intelligence in Medicine,* vol. 149, p. 102788, 2024, doi: 10.1016/j.artmed.2024.102788.