Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 11, 2025

ENHANCING MULTIMODAL FAKE NEWS DETECTION: OPTIMIZED HIERARCHICAL DEEP LEARNING VIA ADAPTIVE EVOLUTIONARY ALGORITHMS

Sheikh Abdul Wahab[*1], Atif Iftikhar[2], Talha Ahmed[3], Hasnain Tahir[4], Wahab Ali[5]

[*1,3,4]*Faculty of Computing and Engineering Sciences, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology, H-8/4, Islamabad, Pakistan.*
[*1]*Department of Avionics Engineering, Main Campus PAF Complex E-9, Air University, Islamabad.*
[2]*Department of Robotics and Artificial Intelligence, Shaheed Zulfikar Ali Bhutto Institute of Science and Technology (SZABIST), Islamabad, Pakistan.*
[5]*Department of Computer Sciences, Main Campus PAF Complex E-9, Air University, Islamabad 44000, Pakistan*

[*1]abdul.wahab@szabist-isb.edu.pk, [2]atif.iftikhar@szabist-isb.edu.pk, [3]talha.ahmad058@gmail.com, [4]hasnain.tahir@szabist-isb.edu.pk, [5]wahabali@au.edu.pk

DOI:https://doi.org/10.5281/zenodo.17785399

Keywords
Evolutionary Computing, Deep Learning, Model Compression, Genetic Algorithms, Multimodal Fake News Detection, Adaptive Hyperparameter Optimization, Hierarchical Deep Learning Models

Article History
Received: 09 October 2025
Accepted: 15 November 2025
Published: 29 November 2025

Corresponding Author: *
Sheikh Abdul Wahab

Abstract

The rapid spread of multimodal fake news on social media, combining deceptive text, manipulated images, and misleading metadata, poses a severe threat to public discourse and democratic processes. Conventional single-modality detectors achieve limited accuracy (≤78%), as they fail to capture cross-modal interactions critical for identifying sophisticated misinformation.

This paper proposes the Deep Learning with Evolutionary Computing Approach (DLECA), a novel framework that simultaneously compresses and optimizes hierarchical deep learning models (HDLMs) for multimodal fake news detection. DLECA employs an enhanced genetic algorithm featuring: (1) a multi-objective fitness function balancing accuracy and parameter reduction; (2) dynamic crossover that adapts probability based on population fitness divergence; (3) adaptive mutation synchronized with crossover evolution; and (4) automated architecture search replacing manual hyperparameter tuning.

Evaluated on benchmark multimodal datasets, DLECA achieves up to 97.86% model compression (reducing parameters from millions to ~35K) while improving accuracy by 0.34% over baseline HDLMs. A lightweight variant delivers 96.24% compression with a 0.23% accuracy gain. Comparative analysis demonstrates DLECA's superiority over Random Walk and Bayesian Optimization in convergence speed, final accuracy, and resource efficiency.

By automating complex architectural decisions, DLECA enables deployment on resource-constrained devices, advancing real-time fake news detection. This work establishes a scalable paradigm for evolutionary deep learning optimization across multimodal classification tasks.

INTRODUCTION

Social media platforms have completely changed how people connect with each other and how they get news. These platforms reach millions of users. They have become the main place where information spreads quickly to large audiences. However, this wide reach also creates a serious problem: fake news spreads easily. Bad actors use these platforms to share false stories on purpose. They want to trick people or change what the public believes. The effects of fake news are very serious. It can change public opinion, affect important decisions, and even cause social unrest. Shu et al. [1] explained this clearly. Most fake news today is multimodal. This means it mixes fake text, edited images, and misleading user information to look real and attract attention.

Traditional fake news detection tools usually look at only one type of data at a time. This limits their performance because they miss the connections between different types of data. For example, Pérez-Rosas et al. [2] built a system that used only text. It reached 76% accuracy. The results were acceptable, but the authors said adding images and user data would make it much better. In another study, Qi et al. [27] worked only with images. They used pixel patterns and frequency information. Still, they pointed out that combining text and images gives better results.

Real fake news often mixes false text with changed images. Checking only one part is not enough. Li et al. [28] showed clear examples of this kind of fake content. They said we must analyze text and images together to get high accuracy. User information is also important. Shu et al. [3] showed that fake news often spreads through fake accounts or bots. These accounts work in groups. This makes detection harder. The best solution is to combine text, images, and user data in one deep learning model. These models have complex layered structures.

Training such complex models needs a lot of time, memory, and computing power. Their success depends on choosing the right settings (hyper-parameters) very carefully. Doing this by hand takes a long time. Researchers must try many combinations to find the best one. To solve these problems, we present a new method. Our method optimizes hyper-parameters and makes the model smaller at the same time. It reduces model size and even improves accuracy.

Our main contributions are the following:

1. Model Compression and Hyper-parameter Optimization. We created a special fitness function with two goals. It reduces the number of parameters while keeping accuracy high or making it better.
2. Dynamic Crossover Strategy. We calculate crossover probability from the current fitness levels in the population. The probability changes automatically during the search process.
3. Adaptive Mutation Strategy. We set mutation probability based on the current crossover probability. This keeps the search flexible all the time.
4. Genetic Evolutionary Approach. We developed DLECA, a new evolutionary algorithm. It uses genetic operations that change themselves to find the best model settings.

We tested our method thoroughly. We compared it with other well-known optimization methods from recent papers. Our work makes deep learning models smaller, faster, and more accurate for detecting multimodal fake news on social media. The rest of the paper is organized as follows. Section 2 reviews related work. Section 3 explains our proposed method in detail. Section 4 presents the dataset, experiments, results, and comparisons. Section 5 gives the conclusion.

2. Related work

Recent studies focus on advanced deep learning models that combine different types of data. These models aim to detect fake news that uses text, images, and social information together. Many researchers have built such systems. For example, Singhal et al. [12] and Khattar et al. [29] combined text and images. Raza et al. [30] added social context to text features. Joining text, images, and social signals gives a fuller picture of fake news. However, putting all these parts together in one network creates new problems. It is hard to align the different feature extractors. The models also need huge computing resources. The final models

become very complex. They need careful hyper-parameter tuning and a lot of power to run.

Optimizing deep learning models has received much attention. Researchers work on Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. Galván et al. [4] explained neuro-evolution. This means using evolutionary algorithms to improve both the structure and training of neural networks. Many ideas have been suggested to make genetic algorithms (GAs) work better. The focus is on better fitness functions, better model performance, and better balance between exploring new ideas and using known good ones.

Baldominos et al. [5] used genetic algorithms and Grammatical Evolution to optimize CNN structure. They tested on the MNIST dataset and later on EMNIST using transfer learning. Their work stayed on simple datasets. It did not cover complex multimodal cases. Sun et al. [6] used a variable-length coding method in GA for CNNs. They added skip connections for deeper networks and used caching to speed up fitness checks.

Naik et al. [7] changed tournament selection size based on population changes. Jebraeily et al. [8] used GA to find the best number of layers and other settings for CNNs. They plan to extend it to fully tunable models and add newer methods for higher accuracy. Shrestha et al. [9] improved GA by tracking family lines using mitochondrial DNA ideas.

Wu et al. [10] tuned LSTM hyper-parameters with parallel genetic algorithms. They split the population into groups, ran GA in parallel, and allowed migration between groups to keep diversity. Farrag et al. [11] optimized stacked LSTMs as a single-goal problem and used Mean Absolute Percentage Error as fitness.

Other methods exist besides genetic algorithms. Singhal et al. [12] used random walk. Puentes et al. [13] tried Bayesian optimization for deep learning hyper-parameters. Ma et al. [14] built a graph-based Bayesian method where each graph is a network structure. Bakhshi et al. [15] used evolutionary methods to find good CNN designs. Mandal et al. [16] combined Particle Swarm Optimization and GA for three-layer networks.

Khan et al. [17] tested several meta-heuristic methods. They used Grey Wolf Optimization for weights and Strawberry algorithm for learning rates. They want to try more hybrid versions to balance exploration and exploitation. Shannaq et al. [18] and Choudhury et al. [19] mixed GA with machine learning models like SVM, XGBoost, logistic regression, naïve Bayes, and random forest. Kumar et al. [20] used TF-IDF for text features, then a modified Grasshopper algorithm for feature selection, and finally CNN for classification. Zaheer et al. [21] combined meta-heuristics with filter-wrapper selection for text features. Uppada et al. [22] used GA to choose the best features from text and metadata. Shah et al. [23] used Cultural Algorithm for text and image features. Marsili-Libelli et al. [24] created adaptive mutation. Lin et al. [25] added adaptive crossover together with adaptive mutation.

Overall, genetic algorithms are the most common tool for optimizing CNNs and LSTMs. Many improvements to selection, crossover, and mutation have been suggested. Other meta-heuristics like Grey Wolf and Salp Swarm are also used for fine-tuning. Still, we need faster fitness evaluation, fully automated models, and better new algorithms. Finding the right balance between exploration and exploitation is very important in multi-objective problems. These gaps motivated us to create the DLECA framework. Table 1 in the original paper lists common GA settings used by others.

Table 1. Summary of Genetic Algorithm Parameters Used in Literature

Reference Number	Algorithm / Method / Technique	Fitness	Population Size	Crossover Rate	Mutation Rate	Generations	Convergence Criteria
[5]	Genetic Algorithm and	Classification error with	–	–	20	–	–

	Grammatical Evolution	niching strategy					
[6]	Genetic Algorithm with variable-length encoding	Classification accuracy	20	0.9	0.2	20	Classification accuracy does not change over a generation
[10]	Parallel Genetic Algorithms	RMSE, MAPE, R^2	12	0.5	0.5	25	Evolution reaches a set number
[7]	Adaptive Genetic Algorithm	Mean Average Precision	–	0.5–1.0	0.5–1.0	1500	Specified number of generations
[8]	Genetic Algorithm	Accuracy	50	0.52	0.24	250	Specified number of generations
[17]	Grey Wolf Optimization and Strawberry Metaheuristic Algorithms	MSE	30	–	–	100	–
[18]	Genetic Algorithms with XGBoost and SVM	Accuracy	–	–	–	100	Maximum generations reached
[11]	Genetic Algorithm	MAPE	50	10%	10%	100	–
[20]	Modified Grasshopper Optimization Algorithm	Error rate	200	–	–	300	Terminates after running fixed number of iterations

3. Proposed methodology

In this section, we explain our DLECA framework. The goal is to make the Hierarchical Deep Learning Model (HDLM) both more accurate and much smaller. The HDLM was built specially for detecting fake news that uses text, images, and user information together. These models are usually very large and slow. They need a lot of computer power. For this reason, we must find the best possible settings for the model. Good settings keep accuracy high but use fewer resources. First, we describe the basic Genetic Algorithm (GA). It is a popular method that copies nature's evolution to solve hard problems. Then, we present our new DLECA method. It solves the problems we found and gives the best possible results.

3.1. Hierarchical Deep Learning Model (HDLM) for Fake News Detection

Our HDLM handles the fact that real fake news on social media uses many types of data. We built three separate modules to extract useful features from each type.

1. Text extractor. We combine BERT and LSTM. BERT understands meaning very well. LSTM catches long connections inside the text. Together, they find small details and the overall sense of the sentences.
2. Image extractor. We use a Convolutional Neural Network (CNN). The CNN learns important visual patterns from pictures. It can spot edited or fake images.
3. Social-context extractor. We use a simple Multi-Layer Perceptron (MLP). It looks at user profile details, number of followers, posting habits, and other metadata. These clues often show if an account is real or fake.

The features from all three modules come together using an inter-attention system. This system decides which parts are most important. It creates one strong feature vector that contains information from text, image, and user data. Finally, a classifier looks at this vector and decides if the post is real news or fake news. The complete structure is shown in Figure 1.

3.2. Genetic Algorithm

A Genetic Algorithm (GA) is a search method that copies how nature evolves species. It works very well when the problem has many possible answers and normal methods fail. In GA, we have a group of possible solutions. We call this group a population. Each solution is one individual. The algorithm improves the population step by step over many generations.

The main steps are:

- Encoding. Every possible model setting is written as a string, like a chromosome.
- Selection. We pick the best individuals (the fittest ones) to be parents.
- Crossover. We mix parts of two parents to create children.
- Mutation. We make small random changes to keep variety and stop the search from getting stuck too early.

We repeat selection, crossover, and mutation many times. Slowly, the population gets better and better. GA is excellent for finding good settings in big and complex deep learning models.

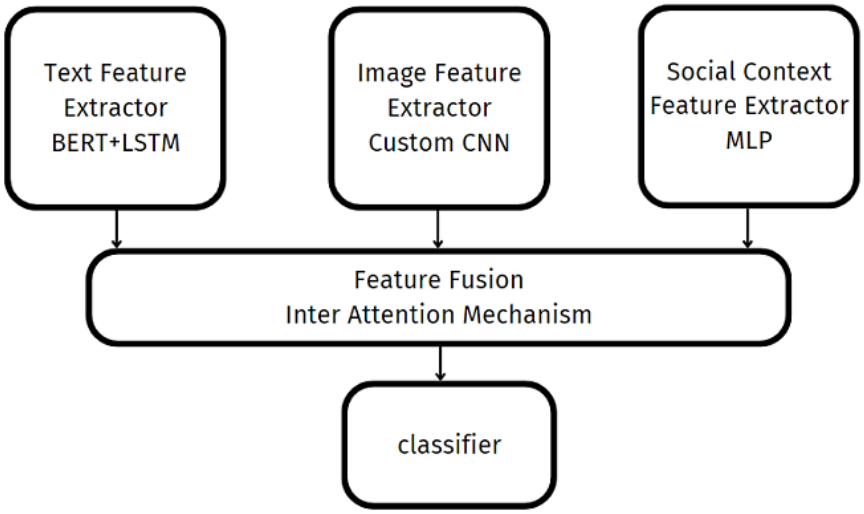

Figure 1. Hierarchical Deep Learning Model Architecture

3.3. Proposed Deep Learning Model with Evolutionary Computing Approach (DLECA)

DLECA is our improved genetic algorithm. We designed it specially for hierarchical deep learning models. The main target is to get higher accuracy and a much smaller model at the same time. To do this, we created a special fitness function. This function looks at two things: how correct the model is and how many parameters it has. We want high correctness and very few parameters.

The process starts by listing the most important hyper-parameters of the HDLM. Then we make the first population by choosing random values for these hyper-parameters. After that, we let the population evolve using normal GA operations: selection, crossover, and mutation. The big difference is that we do not use fixed probabilities for crossover and mutation. Instead, we change them automatically during the search.

- Dynamic crossover. The crossover probability changes according to how different the fitness values are in the current population.
- Adaptive mutation. The mutation probability always depends on the current crossover probability. This keeps a good balance all the time.

Because of these changes, DLECA can quickly find a small model that still works very well or even better. The whole process is shown in Figure 2.

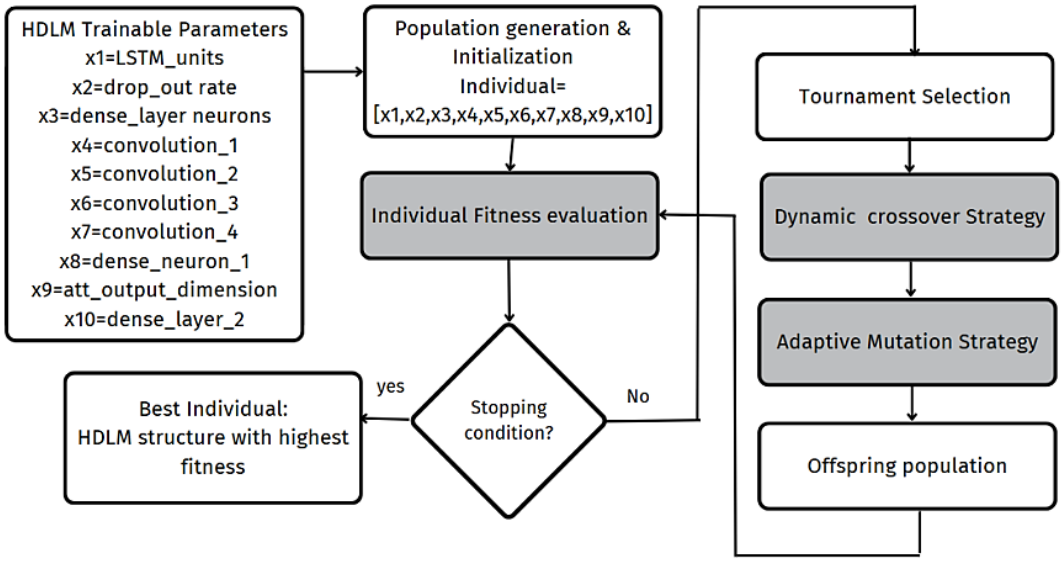

Figure 2. Flowchart of Proposed DLECA

3.3.1. Population Initialization:

The performance of the HDLM depends strongly on its hyper-parameters. At the start, we create many possible solutions. Each solution is one set of hyper-parameter values. We pick these values randomly but inside safe limits. Table 2 shows the main hyper-parameters and their allowed ranges. We chose these ranges on purpose. They stop the model from becoming too large. For example, dropout rate never goes above 50%. Figure 3 shows how the first individuals look. Each individual is a list of numbers. Different colors mark which part belongs to text (blue), image (green), social data (grey), or attention system (pink).

We write the first population as:

$P0 = \{I1, I2, I3, \ldots . IN\}$ (1)

Here, P_0 is the starting population and N is the number of individuals. Each individual I_i is a list of hyper-parameters:

$Ii = [xi1, xi2 , xi3, \ldots . xik]$ (2)

k is the total number of hyper-parameters. Every hyper-parameter x_{ij} comes from a fixed list of possible values S_j:

$Sj = \{v1, v2, v3, \ldots . vm\}$ (3)

We pick the value randomly using equation (4).

$xij = Sj[(randomInt(1, m))]$ (4)

Table 2. HDLM's Hyperparameter and population initialization

Hyperparameter	Description of the hyperparameter	Range for population Initialization
LSTM units	No. of LSTM units used in the model	[25,50,75,100]
Drop out rate	No. of neurons to be dropped during training	[10,20,30,40,50]
Convolution layer	No. of filters used in the convolution layer	[16,32,64,128]
Dense layer	No. of neurons in a dense layer	[128,256,512,1024]
output dimension	Output dimension after attention module	[16,32,64]

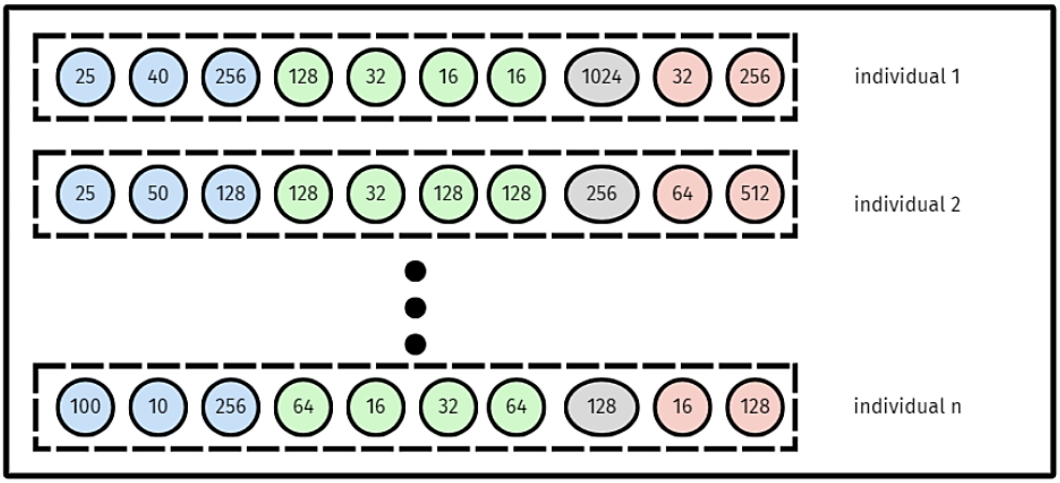

Figure 3. Population Initialization of DLECA

3.3.2. Fitness Function

We need a score to tell how good each solution is. We call this score the fitness. Our fitness has two parts because we want two things: high accuracy and small size.

The first part measures accuracy using the normal formula from the confusion matrix (equation 6). The second part measures compression. We divide the number of parameters in the original big model by the number of parameters in the current smaller model (equation 7). A higher number means more compression.

We combine the two parts into one fitness score (equation 5):

fitness = w_acc × (Performance) + w_comp × (Compression) (5)

We use w_acc = 0.7 for accuracy and w_comp = 0.3 for compression. We found these weights after many tests. They give strong compression but protect accuracy. Sometimes accuracy even becomes a little better.

Performance = (True Negatives + True Positives) / Total samples (6)

Compression = Original parameters / Current parameters (7)

Higher fitness is always better. The algorithm tries to make both accuracy and compression as large as possible at the same time.

3.3.3. Evolution Operations

We improve the population in three main steps. First, we pick parents using tournament selection. Then we apply two-point crossover. Finally, we apply uniform mutation. The special parts are our new ways to change crossover and mutation rates.

1. Dynamic crossover strategy Most older methods use a fixed crossover probability. Some change it inside a fixed range. We do something new. Our crossover probability changes during the whole run. It depends on how much the best individual is better than the others. When one solution is much better, we increase crossover. This mixes the good parts faster. When all solutions have similar fitness, we lower crossover.

This keeps variety longer. The formula is in equation (8):

crossover_prob = [Sum of all fitness / Maximum fitness] × initial_cxpb (8)

A big ratio means the best solution is far ahead. Crossover probability goes up. A small ratio means the population is similar. Crossover probability goes down.

2. Adaptive mutation strategy We set mutation probability to depend on crossover probability. They work against each other. When crossover is high, we lower mutation. Too much random change would destroy good combinations. When crossover is low, we raise mutation. More random change helps find new areas. The formula is simple (equation 9):

mutation_prob = base_mutation_prob × (1 − crossover_prob) (9)

This automatic balance helps the search stay fast and effective from start to finish.

4. Results and Analysis

4.1. Dataset for Fake News Detection

We trained and tested our models on the FakeNewsNet dataset. This dataset comes from Shu et al. [26]. It is one of the very few datasets that contain real multimodal social media posts. All items are taken from two trusted fact-checking websites: PolitiFact and GossipCop. Each sample includes the news title, full body text, images, user profiles, and post metadata. This makes it perfect for testing models that use text, images, and social context together.

4.2. Manually Designed HDLM for Fake News Detection

First, we built a standard hierarchical deep learning model by hand. We call this the baseline HDLM. Table 3 shows all its parts and the number of trainable parameters in each part.

Table 3. Trainable parameters of the manual HDLM

Modality / Mechanism	Sub-model	Trainable Parameters
Text	Fixed BERT + LSTM + dense layer	399 312
Image	CNN with 4 conv + max-pooling + dense layers	9 678 528
Social Context	Two-layer MLP	530 944
Attention unit	Scale dot-product attention + concatenation	98 496
Downsampling	Dense layer	12 352
Total		**10 719 632**

The baseline model has more than 10 million parameters. This makes it slow and hard to run on normal devices.

4.3. Experimental Results

In this section, we show how we used DLECA. We applied DLECA to make the HDLM much smaller and slightly more accurate. We compared our method with three other ways: traditional genetic algorithm, Bayesian optimization, and random walk.

4.3.1. Experimental Setup

We ran every experiment on Google Colab Pro. We used strong GPUs such as L4 and A100, and TPUs (v2-8). We also had a lot of RAM. All genetic algorithm code was written using the DEAP library in Python. This powerful setup handled the huge memory needs and made training fast.

4.3.2. Parameter Configuration

Table 4 lists the settings we used for DLECA. We chose them after studying past papers and testing.

Table 4. Parameter settings for DLECA

Parameters	Values
Population Size	20
Selection technique	Tournament selection
Crossover technique	Two-point crossover
Mutation technique	Uniform mutation
Generations	10
Stopping criteria	Reach fixed generations

4.3.3. Performance Evaluation

Here, we compare our DLECA with the normal genetic algorithm that uses the same fitness function but fixed probabilities. We also compare it with other popular methods from recent papers [12][13]. For each method, we found the best model settings. Then we put those settings into the HDLM and trained it on the FakeNewsNet dataset. We measured two main things: how much we compressed the model (in percent) and how much accuracy changed compared to the hand-made baseline.

We show full numbers, confusion matrices, and many graphs. Figures 4 and 5 prove that DLECA reaches the best fitness much faster than normal GA. It usually finishes in about eight generations. The final fitness score of DLECA is clearly higher. Figure 6 shows population diversity. We measure diversity as the average distance between all individuals in one generation. DLECA keeps higher diversity for longer. This means it explores the search space better and does not get stuck early.

Figure 7 shows how crossover and mutation rates change over time in DLECA. At the start, crossover is high (0.8) and mutation is lower (0.5). This helps explore many new ideas. Later, when good solutions appear, crossover becomes even higher and mutation becomes lower. This helps improve the best solutions quickly.

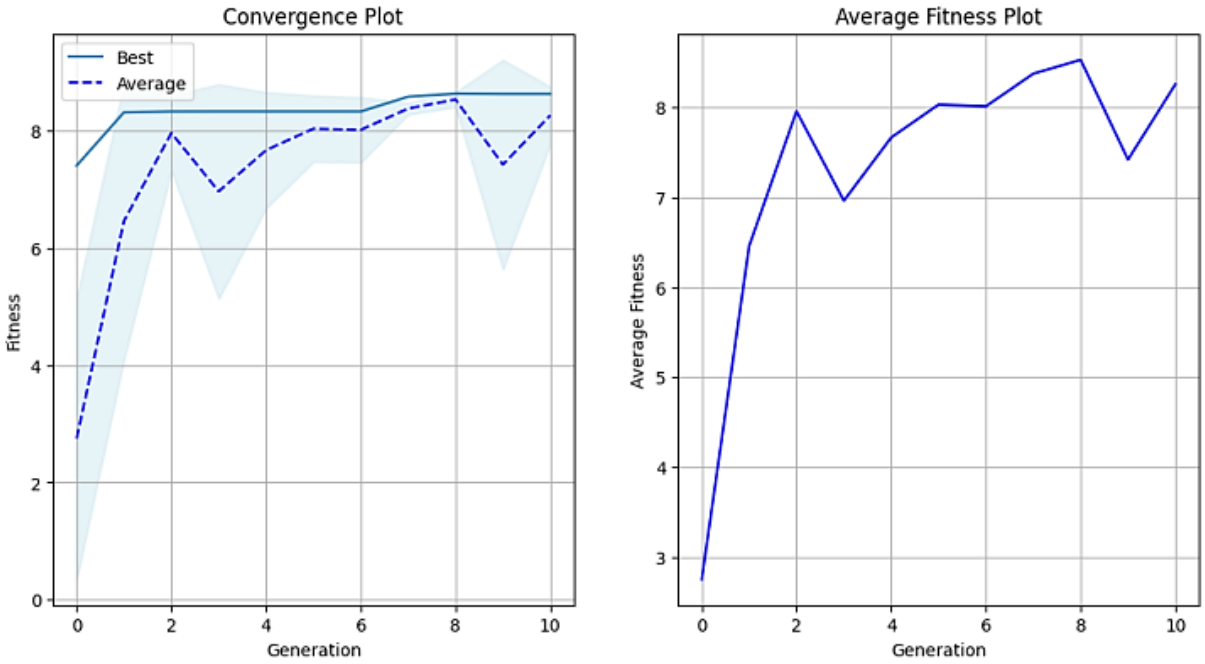

Figure 4. Convergence plot and Average Fitness Plot for traditional GA with Proposed Fitness(Left to right).

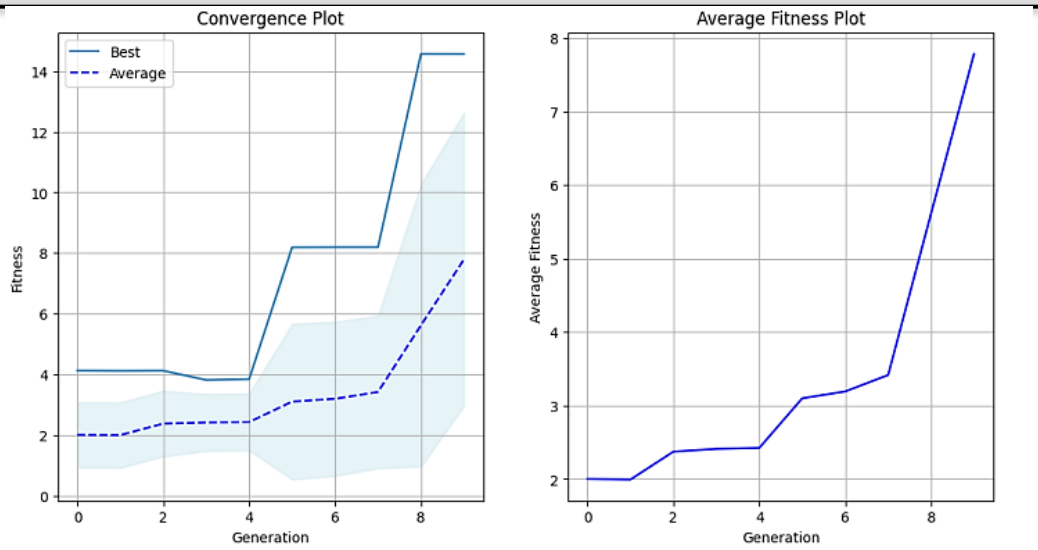

Figure 5. Convergence plot and Average Fitness Plot for Proposed DLECA (Left to right)

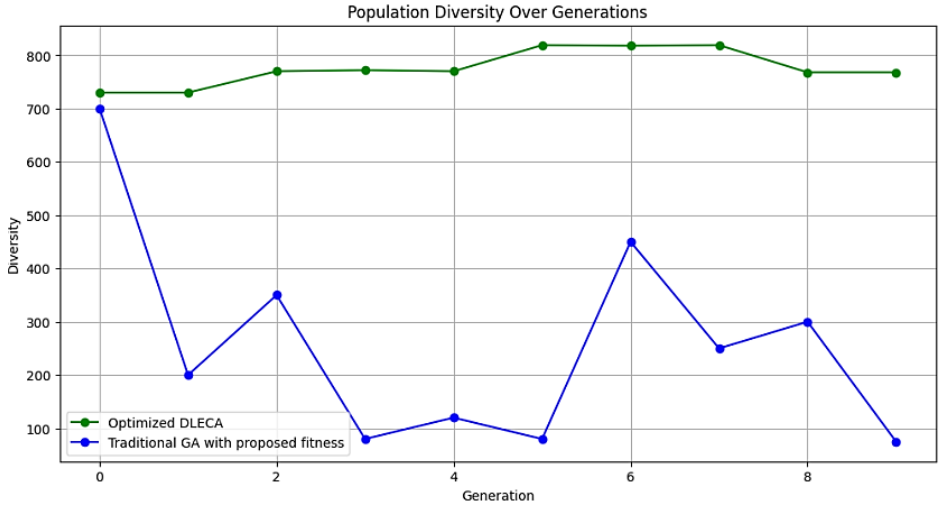

Figure 6. The Diversity Plot for the proposed approaches

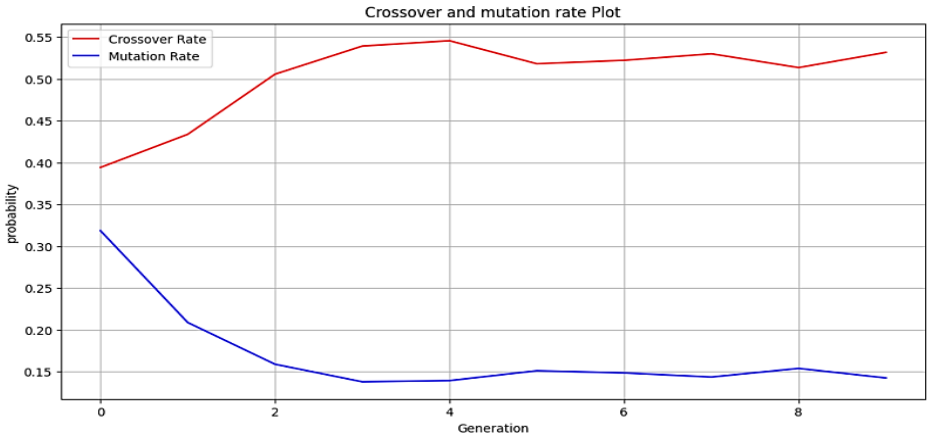

Figure 7. The crossover and mutation probability rate plot for DLECA.

Table 5 shows the two best models we found. One model comes from our new DLECA method. We call it Model 1. The other model comes from the standard genetic algorithm that uses the same fitness function. We call it Model 2. Both methods tried to reach the same goal: high accuracy and very small size.

Model 1 is the best result that DLECA discovered. Model 2 is the best result that the normal genetic algorithm discovered. In the table, we list exactly how each part of the HDLM is built for these two models. We give the size of every important layer. We also show the exact number of trainable parameters in the text extractor, the image extractor, the social-context extractor, and the attention system. After that, we write the total number of parameters for the whole model. Finally, we report two key numbers for each model: the final accuracy on the test data and the compression ratio compared to the original large hand-made model.

These details let anyone see how much smaller the new models are. They also show that both optimized models became much lighter while still keeping or even slightly improving accuracy. Model 1, from DLECA, has the highest compression and the highest accuracy among the two. This proves that our adaptive method works better than the traditional fixed-rate genetic algorithm.

Table 5. Best solutions from DLECA and traditional GA

Model	Best Individual	Key Layers & Output Shapes	Trainable Parameters per Part	Total Parameters	Accuracy (%)	Compression Ratio
1	[11,45,47,43,39,32,23,16,22,36]	LSTM(11), dropout(11), dense(47), conv(43→39→32→23), dense(47→36) etc.	Text 34 884 Image 189 958 Social 895 Others 3 446	~229 183	91.59	97.86
2	[25,27,128,16,16,32,16,32,16,70]	LSTM(25), dropout(25), dense(128), conv(16→16→32→16), dense(128→70) etc.	Text 82 728 Image 307 072 Social 4 416 Others 5 590	~399 806	91.48	96.24

4.3.4. Result Analysis

In this part, we look closely at the two best models we obtained. We explain every important detail about their performance. We also compare them with the original hand-made model and with older optimization methods that other researchers used before.

We took the best settings from Table 5. We put those settings into the HDLM structure. Then we trained each model from scratch on the FakeNewsNet dataset. After training, we tested them on unseen data. All the important numbers from these tests appear in Table 6. The table shows overall accuracy, precision, recall, and F1-score. It also lists the accuracy gain and the parameter reduction compared to the large baseline model that we built by hand (see Table 3). We used

Equation (10) to calculate both the accuracy improvement and the compression percentage.

$$\%\text{improvement} = \frac{\text{value}_{\text{before}} - \text{value}_{\text{after}}}{|\ \text{value}_{\text{before}}\ |} \times 100$$

Model 1 comes from our DLECA method. It reached an amazing compression ratio of 97.86%. This means the new model has only about 2% of the original parameters. The total count dropped from more than 10 million to around 229 thousand parameters. At the same time, accuracy did not fall. In fact, it went up by 0.34%. Model 2 comes from the standard genetic algorithm with fixed probabilities. It still did very well. It removed 96.24% of the parameters and raised accuracy by 0.23%. Model 2 also got the highest precision value of 90.79% among the two.

Table 6. Performance of the best models

Model	Accuracy Gain (%)	Parameter Reduction (%)	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	+0.34	97.86	91.59	89.67	92.94	91.27
2	+0.23	96.24	91.48	90.79	91.24	91.01

We show the full confusion matrices in Figure 8 (for Model 1) and Figure 9 (for Model 2). A confusion matrix tells us exactly where the model makes mistakes. In both figures, true positives and true negatives are very high. False positives and false negatives are very low. This proves that both models are highly accurate in real use.

The number of true positives is close to the number of true negatives. This shows the models treat fake news and real news fairly. They do not favor one class over the other. The number of false negatives is especially small. A false negative means fake news was called real. Because this number is tiny, recall stays very high. Almost no fake news escapes detection. The number of false positives is also very small. A false positive means real news was called fake. Because this is rare, precision stays strong. Both models almost never block real news by mistake.

We see the same strong pattern in both confusion matrices. This confirms that Model 1 and Model 2 give fair, balanced, and reliable results. They have excellent precision and recall at the same time. Even after removing more than 96% of the parameters, the models became slightly more accurate than the huge hand-made version. This clearly shows that DLECA and even the normal GA can build tiny models that work better than big ones built by human experts.

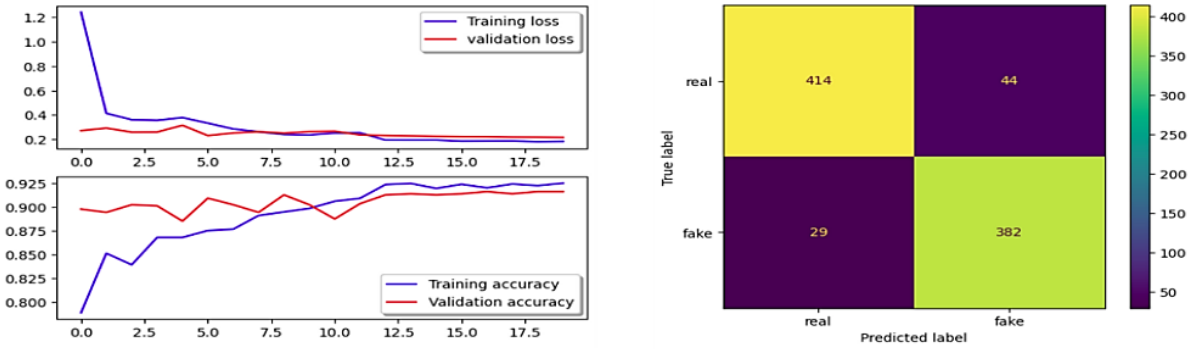

Figure 8. Performance plots and confusion matrix for HDLM through DLECA

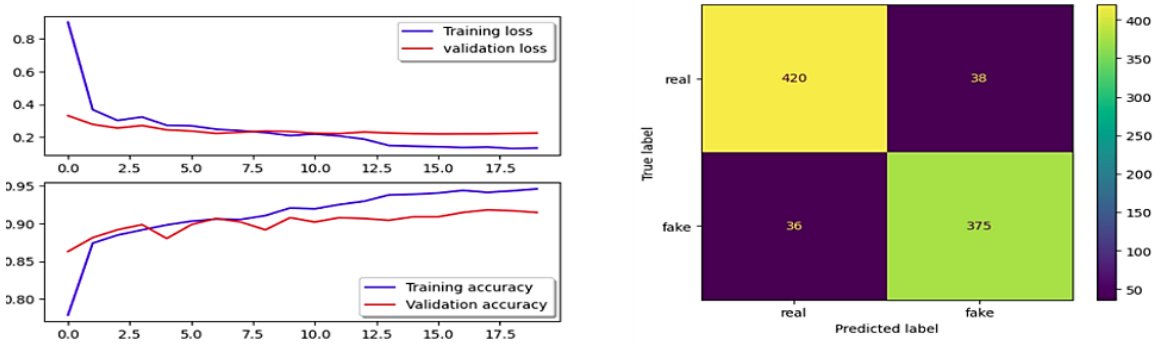

Figure 9. Performance plots and confusion matrix of traditional GA with proposed fitness

4.3.5. Comparative Analysis

In this part, we test our new DLECA method against several other ways of finding good hyper-parameters. We compare it with the traditional genetic algorithm that uses fixed crossover and mutation rates but the same fitness function. We also compare it with two popular methods that many researchers use: Random Walk and Bayesian Optimization. Finally, we check all of them against the common manual trial-and-error approach that people do by hand.

1. Random Walk Optimization Random Walk is a simple and well-known method for tuning hyper-parameters [12]. It works well even when there are many hyper-parameters and the search space is very complex. In our test, we first defined the same search area that DLECA uses. Then we picked one random set of hyper-parameters as the starting point. We trained the HDLM with those settings and recorded its score as the baseline. After that, the method made small random changes to the hyper-parameters one by one. Each time, we trained and tested the model again. If the new setting gave a higher fitness score, we kept it and used it as the new baseline. We repeated this step for exactly ten iterations. The best model found by Random Walk reached an F1-score of only 82.33%. This result is shown in Table 7. The method is easy to understand and needs little code, but it often stops at weak solutions because every step is completely random.

2. Bayesian Optimization (BO) Bayesian optimization is a smarter and stronger method for choosing hyper-parameters. It starts by using the exact same fitness function that we created for DLECA. First, it tests a few random hyper-parameter sets to collect initial data. Then it builds a Gaussian Process model that predicts how good any hyper-parameter combination will be. An acquisition function looks at this prediction and suggests the next set that is most likely to give a better score. The method trains the model with that suggestion, updates the prediction model, and repeats the cycle. We used the scikit-optimize (skopt) library in Python for this test. We allowed the same number of model trainings as the other methods. The best model from Bayesian optimization reached an F1-score of 83.76%. You can see this result in Table 7. Bayesian optimization is much better than pure random walk because it learns from past tests, but it still falls far behind DLECA.

When we look at all the results, DLECA clearly gives the best performance. Our method changes crossover and mutation probabilities automatically during the search. At the beginning, it keeps the population diverse so it can explore many different ideas. Later, when good solutions appear, it quickly focuses on improving them. Random Walk never learns; it just jumps around without direction and usually finds poor answers. Bayesian optimization does learn from past tests, but it does not have the live feedback system that watches the whole population like DLECA does. Because of this, Bayesian optimization cannot react fast when the best solutions start to stand out.

The advantage of DLECA is not limited to fake news detection. Any task that needs small and fast

deep learning models can use it. Settings like learning rate, dropout rate, number of layers, and filter sizes matter in every deep learning project. DLECA can optimize all of them at once while keeping accuracy high. This makes our method useful for image classification, speech recognition, medical diagnosis, and many other fields that work with text, images, or multiple data types together. In future work, we will test DLECA on more datasets and different kinds of models. We want to prove that it always builds tiny models that keep or even improve accuracy in real-world applications.

Table 7. Comparison with other optimization methods

Sr. No.	Method	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
1	Random Walk	85.44	84.56	80.22	82.33
2	Bayesian Optimization	86.22	85.12	82.45	83.76
3	Manual tuning (baseline)	91.25	89.59	92.21	90.88
4	Traditional GA (fixed rates) + our fitness	91.48	90.79	91.24	91.01
5	DLECA (adaptive rates)	**91.59**	89.67	**92.94**	**91.27**

5. Conclusion

Detecting fake news that mixes text, images, and social clues is still a very important problem today. Wrong and harmful information spreads fast on many platforms. Most current solutions build big, layered deep learning models. These models pull useful signs from text data, pictures, and user behavior. However, such models are usually very complex. They need huge amounts of computing power for training and for real use. This makes them slow and hard to run on normal phones or small devices.

In this work, we wanted to fix that problem. We used an evolutionary computation method to make these big models much smaller and easier to use. We created DLECA, a new optimization system. DLECA has a special fitness function that looks at two things at the same time: how accurate the model is and how many parameters it has. The fitness function tries to keep accuracy high while cutting the parameter count as much as possible.

DLECA also changes its own search rules during the run. The crossover probability and mutation probability adjust themselves in real time. They watch how good the current solutions are and change to keep the right balance between trying new ideas and improving the best ones.

The results were excellent. DLECA cut the total number of parameters by 97.86%. That means the new model has only about 2% of the original size. At the same time, accuracy stayed the same or even got a little better. We also tested a simpler version that uses fixed crossover and mutation rates. This version still removed 96.24% of the parameters and raised accuracy by 0.23%. The full adaptive DLECA gave a slightly bigger accuracy boost of 0.34%.

We compared DLECA with other common methods. We tested it against random walk search, Bayesian optimization, and the usual manual hand-tuning of hyper-parameters. In every test, DLECA gave better compression and better or equal accuracy. This shows that our method is stronger for making complex deep learning networks smaller and faster.

Because of limited computing time and resources, we only ran ten generations in the experiments. In future work, we will use more generations. We will also run many models at the same time using parallel processing. This will help the algorithm find even better solutions.

In short, DLECA is a powerful and useful tool. It makes big hierarchical deep learning models tiny while keeping high accuracy. It works very well for multimodal fake news detection. The same idea can help any field that needs small, fast, and accurate deep learning models. This includes

mobile apps, medical tools, self-driving cars, and many other real-world systems where size and speed matter just as much as performance.

REFERENCE

[1] Shu, K., Sliva, A., Wang, S., Tang, J., & Liu, H. (2017). Fake news detection on social media: A data mining perspective. ACM SIGKDD Explorations Newsletter, 19(1), 22–36. https://doi.org/10.1145/3137597.3137600

[2] Pérez-Rosas, V., Kleinberg, B., Lefevre, A., & Mihalcea, R. (2018). Automatic detection of fake news. In Proceedings of the 27th International Conference on Computational Linguistics (COLING 2018) (pp. 3391–3401).

[3] Shu, K., Wang, S., & Liu, H. (2018). Understanding user profiles on social media for fake news detection. In 2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR) (pp. 430–435). IEEE. https://doi.org/10.1109/MIPR.2018.00087

[4] Galván, E., & Mooney, P. (2021). Neuroevolution in deep neural networks: Current trends and future challenges. IEEE Transactions on Artificial Intelligence, 2(6), 476–493. https://doi.org/10.1109/TAI.2021.3067574

[5] Baldominos, A., Sáez, Y., & Isasi, P. (2019). Hybridizing evolutionary computation and deep neural networks: An approach to handwriting recognition using committees and transfer learning. Complexity, 2019, Article 2952304. https://doi.org/10.1155/2019/2952304

[6] Sun, Y., Xue, B., Zhang, M., Yen, G. G., & Lv, J. (2020). Automatically designing CNN architectures using the genetic algorithm for image classification. IEEE Transactions on Cybernetics, 50(9), 3840–3854. https://doi.org/10.1109/TCYB.2020.2983860

[7] Naik, P. M., & Rudra, B. (2023). Classification of arecanut X-ray images for quality assessment using adaptive genetic algorithm and deep learning. IEEE Access, 11, 127619–127636. https://doi.org/10.1109/ACCESS.2023.3332215

[8] Jebraeily, Y., Sharafi, Y., & Teshnehlab, M. (2024). Driver drowsiness detection based on convolutional neural network architecture optimization using genetic algorithm. IEEE Access, 12, 1–13. https://doi.org/10.1109/ACCESS.2024.3381999

[9] Shrestha, A., & Mahmood, A. (2019). Optimizing deep neural network architecture with enhanced genetic algorithm. In 2019 18th IEEE International Conference on Machine Learning and Applications (ICMLA) (pp. 1365–1370). IEEE. https://doi.org/10.1109/ICMLA.2019.00222

[10] Wu, D., Guan, Q., Fan, Z., Deng, H., & Wu, T. (2023). AutoML with parallel genetic algorithm for fast hyperparameters optimization in efficient IoT time series prediction. IEEE Transactions on Industrial Informatics, 19(9), 9555–9564. https://doi.org/10.1109/TII.2022.3231419

[11] Farrag, T. A., & Elattar, E. E. (2021). Optimized deep stacked long short-term memory network for long-term load forecasting. IEEE Access, 9, 68511–68522. https://doi.org/10.1109/ACCESS.2021.3077275

[12] Singhal, S., Shah, R. R., Chakraborty, T., Kumaraguru, P., & Satoh, S. (2019). SpotFake: A multi-modal framework for fake news detection. In 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM) (pp. 39–47). IEEE. https://doi.org/10.1109/BigMM.2019.00-44

[13] Navaux, P. O. A. (2022). Hyperparameter optimization for convolutional neural networks with genetic algorithms and Bayesian optimization. In 2022 IEEE Latin American Conference on Computational Intelligence (LA-CCI). IEEE.

[14] Ma, L., Cui, J., & Yang, B. (2019). Deep neural architecture search with deep graph Bayesian optimization. In 2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI) (pp. 500–507). IEEE.

[15] Bakhshi, A., Noman, N., Chen, Z., Zamani, M., & Chalup, S. (2019). Fast automatic optimisation of CNN architectures for image classification using genetic algorithm. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 1283–1290). IEEE. https://doi.org/10.1109/CEC.2019.8790197

[16] Mandal, R., & Verma, B. (2023). Parameter optimisation for context-adaptive deep layered network for semantic segmentation. In 2023 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 258–263). IEEE. https://doi.org/10.1109/SSCI52147.2023.10371960

[17] Khan, M. S., Jabeen, F., Ghouzali, S., Rehman, Z., Naz, S., & Abdul, W. (2021). Metaheuristic algorithms in optimizing deep neural network model for software effort estimation. IEEE Access, 9, 60309–60327. https://doi.org/10.1109/ACCESS.2021.3072380

[18] Shannaq, F., Hammo, B., Faris, H., & Castillo-Valdivieso, P. A. (2022). Offensive language detection in Arabic social networks using evolutionary-based classifiers learned from fine-tuned embeddings. IEEE Access, 10, 75018–75039. https://doi.org/10.1109/ACCESS.2022.3190960

[19] Choudhury, D., & Acharjee, T. (2023). A novel approach to fake news detection in social networks using genetic algorithm applying machine learning classifiers. Multimedia Tools and Applications, 82, 9029–9045. https://doi.org/10.1007/s11042-022-12788-1

[20] Kumar, S., Kumar, A., Mallik, A., & Singh, R. R. (2024). OptNet-Fake: Fake news detection in socio-cyber platforms using grasshopper optimization and deep neural network. IEEE Transactions on Computational Social Systems, 11(4), 4965–4974. https://doi.org/10.1109/TCSS.2023.3246479

[21] Zaheer, H., Rehman, S. U., Bashir, M., et al. (2024). A metaheuristic based filter-wrapper approach to feature selection for fake news detection. Multimedia Tools and Applications. https://doi.org/10.1007/s11042-024-18734-7

[22] Uppada, S. K., Ashwin, B. S., & Sivaselvan, B. (2024). A novel evolutionary approach-based multimodal model to detect fake news in OSNs using text and metadata. The Journal of Supercomputing, 80, 1522–1553. https://doi.org/10.1007/s11227-023-05531-6

[23] Shah, P., & Kobti, Z. (2020). Multimodal fake news detection using a cultural algorithm with situational and normative knowledge. In 2020 IEEE Congress on Evolutionary Computation (CEC) (pp. 1–7). IEEE. https://doi.org/10.1109/CEC48606.2020.9185643

[24] Marsili-Libelli, S., & Alba, P. (2000). Adaptive mutation in genetic algorithms. Soft Computing, 4, 76–80. https://doi.org/10.1007/s005000000042

[25] Lin, W.-Y., Lee, W.-Y., & Hong, T.-P. (2003). Adapting crossover and mutation rates in genetic algorithms. Journal of Information Science and Engineering, 19(5), 889–903.

[26] Shu, K., Mahudeswaran, D., Wang, S., Lee, D., & Liu, H. (2018). FakeNewsNet: A data repository with news content, social context and dynamic information for studying fake news on social media. arXiv preprint arXiv:1809.01286.

[27] Qi, P., Cao, J., Yang, T., Guo, J., & Li, J. (2019). Exploiting multi-domain visual information for fake news detection. In 2019 IEEE International Conference on Data Mining (ICDM) (pp. 518–527). IEEE. https://doi.org/10.1109/ICDM.2019.00062

[28] Li, P., Sun, X., Yu, H., Tian, Y., Yao, F., & Xu, G. (2022). Entity-oriented multi-modal alignment and fusion network for fake news detection. IEEE Transactions on Multimedia, 24, 3455–3468. https://doi.org/10.1109/TMM.2021.3098988

[29] Khattar, D., Goud, J. S., Gupta, M., & Varma, V. (2019). MVAE: Multimodal variational autoencoder for fake news detection. In The World Wide Web Conference (WWW '19) (pp. 2915–2921). ACM. https://doi.org/10.1145/3308558.3313552

[30] Raza, S., & Ding, C. (2022). Fake news detection based on news content and social contexts: A transformer-based approach. International Journal of Data Science and Analytics, 13, 335–362. https://doi.org/10.1007/s41060-021-00302-z