Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

CONSISTENT FRUITFLY OPTIMIZATION-BASED GRAPH NEURAL NETWORK (CFO-GNN) FOR ANALYZING SENTIMENTS IN AUGMENTED REALITY-ENABLED ONLINE SHOPPING

[1]PRAGATHI ARAVABOOMI, [2]USHA S, [3]NELSONMANDELA S ,[4]DURGESH TRIPATHI, [5]BROSKHAN P

[1]PG Department of Journalism and Communication, Dwaraka Doss Goverdhan Doss Vaishnav College, Chennai, Tamil Nadu, India

[2]Department of Computer Applications, Faculty of Science and Humanities, SRM Institute of Science and Technology, Kattankulathur, Chennai

[3,5]Department of Animation and Virtual Reality, JAIN (Deemed-to-be University), Bengaluru, Karnataka, India

[4]University School of Mass Communication, Guru Gobind Singh Indraprastha University, Delhi, India.

Email-id: [1]pragathireddy.nlp@gmail.com, [2]ushas6@srmist.edu.in, [3]nelson.mandela@jainuniversity.ac.in, [4]drdurgeshtripathi@ipu.ac.in, [5]broskhan.p@jainuniversity.ac.in

ABSTRACT

Online shopping has evolved significantly with the integration of augmented reality (AR) technology, offering users the ability to visualize products in their physical space before making a purchase. However, sentiment analysis within AR-enabled platforms faces challenges due to sparse review data, unlike traditional e-commerce platforms. The Consistent Fruitfly Optimization-Based Graph Neural Network (CFO-GNN) proposed in this paper addresses this challenge by combining fruitfly optimization with graph neural networks. This innovative approach allows CFO-GNN to efficiently handle sparse data while capturing the intricate relationships present in AR shopping experiences. By leveraging these techniques, CFO-GNN enables more accurate sentiment analysis, empowering businesses to make informed decisions and enhance user satisfaction in the dynamic landscape of AR-enabled online shopping. Through comprehensive evaluation on a diverse dataset, CFO-GNN demonstrates its effectiveness in improving sentiment analysis within AR environments, highlighting its potential to drive advancements in user experience and competitiveness for businesses operating in AR-enabled online retail.

Keywords: *Augmented Reality, Analysis, Classification, Online Shopping, Sentiment, Sparse Data*

1. INTRODUCTION

Online shopping isn't just about convenience; it can also yield substantial cost savings. Online retailers frequently provide exclusive promotions and discounts you won't find in physical stores. Loyalty programs and email newsletters further reduce the cost of purchases [1]. The ability to easily compare prices across multiple websites ensures that consumers find the best deals. Online stores often have lower operating costs than brick-and-mortar shops, which translate to more affordable prices for customers. The elimination of commuting saves both time and money on transportation. Online shoppers can avoid impulse purchases that often occur during in-person shopping trips [2]. In summary, online shopping delivers convenience and significant cost-effectiveness, enabling consumers to maximize their budgets.

One of the standout features of AR-enabled online shopping is the ability to try before you buy. Whether shopping for clothing, accessories, or cosmetics, AR technology enables virtual try-ons that provide a realistic sense of how products will look or fit [3]. Imagine virtually trying on different outfits, experimenting with makeup looks, or even placing virtual furniture in your home to see if it suits your space. This level of interactivity empowers shoppers to make informed choices, reducing the uncertainty often accompanying online purchases. It's a game-changer for fashion enthusiasts and anyone looking to make confident online shopping decisions [4].

Sentiment analysis bridges businesses and their customers, transforming vast volumes of product reviews into meaningful insights. Quantifying the emotional tones in these reviews

converts qualitative data into quantitative data that companies can track and analyze over time [5]. This approach reveals trends in customer sentiment and offers opportunities to pinpoint areas of concern or excellence. Smartphone manufacturers might use sentiment analysis to identify consistent praise for battery life but frequent complaints about camera quality. Armed with this information, they can prioritize improvements in camera technology for their next product release [6].

Sentiment analysis also plays a vital role in enhancing customer engagement. Companies can engage with their audience more effectively by tracking sentiment trends and monitoring customer feedback channels. For example, a tech company might identify a recurring issue mentioned in customer reviews [7]. They can then proactively reach out to affected customers with solutions or assistance, demonstrating a commitment to customer satisfaction. Additionally, sentiment analysis can inform content creation and marketing strategies, ensuring companies deliver messages that resonate positively with their target audience [8]. This personalized approach fosters a sense of connection and strengthens brand loyalty. In summary, sentiment analysis is a powerful tool that helps companies and manufacturers make data-driven decisions, resulting in better customer products and services while enhancing customer engagement and satisfaction [9].

1.1. Problem Statement

In the realm of sentiment analysis within AR-enabled shopping apps, a central challenge revolves around the issue of sparsity. The inadequacy of available customer review data primarily characterizes this challenge. Unlike well-established e-commerce platforms, these apps often suffer from a shortage of user-generated sentiments, which has multifaceted implications. The sparse dataset hinders the development of accurate sentiment analysis models, as they typically require substantial, diverse data for training and validation. Moreover, the scarcity of sentiment insights impedes businesses' ability to effectively assess user satisfaction and make informed decisions about app improvements. This limitation could also introduce bias into performance evaluations, as it may not adequately capture the sentiments of the entire user base. Therefore, tackling the sparsity problem is of paramount importance in the context of sentiment analysis within AR-enabled shopping apps, as it is essential for advancing research in this area and enabling businesses to enhance user experiences and maintain competitiveness.

1.2. Motivation

Addressing the challenge of sparsity in sentiment analysis within AR-enabled shopping apps is driven by several compelling factors. These apps are at the forefront of reshaping the e-commerce landscape, offering innovative shopping experiences. To harness their full potential, accurate sentiment understanding is paramount. However, the lack of available sentiment data presents a formidable hurdle, hindering the development of robust sentiment analysis models and limiting informed decision-making for app enhancements. Solving this issue is crucial for enhancing user satisfaction, advancing AR-based shopping apps, and contributing valuable insights to natural language processing and machine learning. Moreover, overcoming sparsity in this context can inspire innovative techniques that transcend e-commerce, furthering the broader scope of AI research and its practical applications.

1.3. Objectives

The central research objective is to devise a custom-tailored sentiment classification algorithm specifically designed to address the pervasive data sparsity issue within AR-enabled shopping apps while benefiting end-users. This algorithm aims to substantially enhance the precision and relevance of sentiment analysis in augmented reality shopping, contributing to businesses, developers, and customers. By creating a robust model capable of effectively handling sparse sentiment data and accounting for the unique characteristics of AR shopping experiences, this research seeks to empower consumers with more informed purchasing decisions, improved user experiences, and increased satisfaction when navigating AR shopping platforms. Leveraging cutting-edge natural language processing techniques, the study aspires to provide a valuable tool for extracting precise sentiment insights from user reviews, ultimately elevating the overall quality and competitiveness of AR shopping apps while enhancing the customer journey.

2. LITERATURE REVIEW

"Enhanced Elman Spike Neural Network" [10] for sentiment analysis within online product recommendations represents a sophisticated fusion of neural network architecture and spike-based modelling, promising heightened accuracy and

effectiveness in understanding user sentiment dynamics. This method harnesses the power of the Elman spike neural network, renowned for its capacity to model temporal dependencies in sequential data. "Hybrid Recommendation Algorithm" [11] combines user comment sentiment analysis with matrix decomposition techniques and offers a powerful approach to enhance recommendation systems. The algorithm gains insights into users' emotional responses to products or services by employing sentiment analysis on user comments and reviews. "Negation in sentences" [12] plays a pivotal role in sentiment analysis and polarity detection. Neglecting its effect can lead to erroneous polarity assignments, potentially distorting the true sentiment conveyed in a text. It recognizes and handles negation appropriately, employing techniques like dependency parsing and context awareness. "Multilingual Transfer Learning for Bahraini Dialect Sentiment" [13] is a ground-breaking approach that harnesses the power of multilingual deep learning and transfer learning to perform sentiment analysis on sequential text data written in Bahraini dialects. This innovative methodology offers a solution to the challenge of limited labeled data for dialect-specific sentiment analysis.

"Barnacles Mating Algorithm" [14] is an innovative approach for sentiment analysis in online product recommendation systems, albeit without emphasizing its positive aspects. The network leverages a graph-based modeling approach to represent user behaviour and preferences within a structured framework efficiently. "Efficient Multiple-Word Embedding" [15] describes a potent approach to sentiment analysis that underscores both efficiency and cross-domain adaptability. This method harnesses the power of multiple-word embeddings to extract comprehensive feature representations from text data across diverse domains, allowing for a nuanced understanding of language nuances and domain-specific vocabulary. "Improved Unified Domain Adversarial Category-Wise Alignment Network" [16] presents a sophisticated solution to the challenging problem of unsupervised cross-domain sentiment classification. When labeled data is scarce or unavailable in different domains, this methodology is a valuable tool. It accomplishes this skillfully by utilizing domain adversarial techniques to align the sentiment categories across various domains. "S3map" [17] is an advanced methodology that revolutionizes sentiment analysis by focusing on extracting sentiments associated with specific aspects within text data. This approach combines the strength of semi-supervised learning techniques with a masked aspect prediction model to significantly enhance the accuracy and granularity of aspect-based sentiment analysis.

"Topic Driven Adaptive Network (TDAN)" [18] introduces a state-of-the-art approach to tackle the complexities of sentiment analysis across different domains. This method employs adaptive networks driven by topic information to address the challenges of varying sentiment expressions in diverse contexts. By incorporating topic relevance, the model can discern sentiment nuances unique to each domain, resulting in more accurate and context-aware sentiment analysis. This innovation finds practical applications in market research, product sentiment tracking, and social media analysis, where understanding sentiment across domains is paramount for decision-making and insights. "Deep Ensemble Learning Technique (DELT)" [19] represents a significant advancement in recommendation systems. This innovative methodology recognizes that user sentiment is crucial in delivering personalized and relevant recommendations. SARWAS stands out by harnessing the power of deep ensemble learning, which combines the predictive capabilities of multiple models, ensuring that recommendations not only align with user preferences but also resonate with their sentiments. This approach finds applications in diverse sectors, including online content platforms, where sentiment-based recommendations can boost engagement, and in online retail, where it can drive purchasing decisions.

2.1 Comparative Analysis of Existing Techniques

The current state-of-the-art techniques for sentiment analysis exhibit varying strengths and limitations. TDAN effectively captures temporal dependencies in sequential data, making it suitable for time-sensitive sentiment prediction; however, its performance declines when dealing with sparse and graph-structured relationships. DELT enhances classification through ensemble learning and optimized feature engineering but often requires extensive preprocessing and computational effort. Traditional recommendation-based sentiment frameworks mainly rely on dense datasets and struggle to generalize in AR-enabled shopping environments where user feedback is relatively limited. In contrast, graph-based approaches are

increasingly recognized for their ability to preserve relational information among entities. Motivated by these observations, CFO-GNN integrates graph neural representations with Consistent Fruitfly Optimization to improve learning efficiency and sentiment classification under sparse review conditions.

2.2 Research Gap

Although several sentiment analysis approaches have been proposed for recommendation systems and online shopping environments, notable limitations remain in the context of augmented reality (AR)-enabled shopping. Existing methods largely focus on conventional e-commerce platforms and often depend on dense, large-scale review datasets for effective training. Models such as TDAN and DELT demonstrate promising classification capabilities; however, their ability to capture complex user interactions and contextual dependencies in sparse AR review environments remains limited. Furthermore, many existing studies do not adequately model the relational structure between user behaviour, product interactions, and sentiment dependencies. These limitations highlight the necessity for an intelligent framework capable of handling sparse sentiment information while preserving interconnected relationships in AR shopping ecosystems. Therefore, the proposed CFO-GNN model is designed to bridge these gaps through the integration of Consistent Fruitfly Optimization and Graph Neural Networks.

2.3 Bio-Inspired Optimization for Intelligent Sentiment Systems

Bio-inspired optimization methods have increasingly supported intelligent recommendation and sentiment systems by improving learning efficiency in sparse environments. Optimization-driven mechanisms including frog leap, cuckoo search, fish swarm, whale optimization, and wolf prey strategies demonstrated improved adaptability and pattern discovery under uncertain conditions [20], [21], [42], [45]–[47]. Adaptive routing and optimization frameworks further highlighted effective decision-making under incomplete information [22], [28], [41], [51], [55], [60]. Firefly optimization, particle swarm optimization, peregrine falcon-inspired approaches, and kingfisher-inspired routing also reported robust performance in extracting hidden behavioural relationships and improving adaptive intelligence [27], [33], [39], [43], [44], [52], [59]. Recent optimization studies involving energy-aware frameworks, delay minimization models, and performance modelling of bio-inspired systems emphasized improved adaptability and efficient learning under constrained environments [25], [26], [29]. These findings support the integration of Consistent Fruitfly Optimization in CFO-GNN for sparse AR shopping sentiment prediction.

2.4 Optimization-Driven Neural Learning for Sentiment Classification

Optimization-assisted neural learning methods have shown promising outcomes in sentiment classification and predictive modelling. Kalman filtering-based neural frameworks improved sentiment understanding by reducing noisy information and improving contextual learning [30], while optimization-based extreme learning models and swarm-assisted classifiers enhanced classification reliability in large sentiment datasets [61], [66], [69]. Bio-inspired optimization combined with recurrent learning also improved prediction accuracy in complex classification tasks [32], [37], [48]. Chebyshev neural networks and multimodal deep learning approaches further highlighted improved capability for identifying hidden sentiment patterns from heterogeneous datasets [73], [74]. These developments strengthen the motivation for CFO-GNN in AR-enabled sentiment analysis.

2.5 Graph-Based Intelligence in AR Shopping Environments

Graph-based intelligence has become important for modelling user interactions and behavioural dependencies in digital ecosystems. Intelligent decision-making frameworks and bio-inspired computational systems demonstrated improved adaptability in interconnected environments [23], [58], [63], [70]. Graph-oriented learning integrated with reinforcement mechanisms also showed promising outcomes in understanding relational dependencies [54]. Since AR-enabled shopping involves dynamic interactions among users, products, and experiences, graph neural approaches provide improved contextual understanding. Emerging AR/VR communication ecosystems and intelligent connectivity models further reinforce the need for adaptive graph-aware learning for sentiment prediction [50], [62], [67].

2.6 AI and Behavioural Analytics in Digital Shopping

Artificial intelligence increasingly supports customer behaviour analysis and secure digital interactions across online platforms. Studies

discussing AI risks and intelligent automation emphasized reliable and adaptive decision-making for digitally connected systems [24], [64], [65], [71]. Blockchain-supported privacy mechanisms and secure communication frameworks further improved trust and reliability in online ecosystems [31], [34], [40]. AI-driven behavioural analytics, sustainability-oriented systems, cybercrime analysis, and intelligent user modelling highlighted the importance of extracting meaningful insights from large behavioural datasets [35], [36], [38], [49], [53], [68]. These perspectives align with CFO-GNN's objective of improving sentiment understanding in AR shopping systems.

2.7 Mathematical Modelling for Sparse Sentiment Analysis

Sparse sentiment information remains a major challenge in AR-enabled shopping environments due to inconsistent customer feedback. Mathematical modelling approaches such as plithogenic cubic sets and intelligent uncertainty frameworks demonstrated strong capability in representing incomplete information under complex decision environments [56], [57], [72]. Optimization-driven studies further emphasized adaptive reasoning for extracting reliable knowledge from limited and uncertain datasets [39], [43]. These findings strengthen the rationale behind CFO-GNN, where Consistent Fruitfly Optimization and Graph Neural Networks jointly address sparse sentiment prediction challenges in augmented reality-enabled online shopping systems.

3. CONSISTENT FRUITFLY OPTIMIZATION-BASED GRAPH NEURAL NETWORK

The Consistent Fruitfly Optimization-based Graph Neural Network (CFO-GNN) is a novel algorithm designed to address challenges in sentiment analysis, particularly in sparse data environments within augmented reality (AR) enabled online shopping platforms. By integrating fruitfly optimization principles with graph neural networks, CFO-GNN efficiently navigates complex data structures and captures subtle relationships inherent in AR shopping experiences. This innovative approach enhances the precision and relevance of sentiment analysis, empowering businesses to make informed decisions and improve user satisfaction. CFO-GNN represents a significant advancement in the field of natural language processing and machine learning, offering promise for applications beyond AR-enabled online retail.

3.1.1. Advanced Message Passing

The fundamental idea behind message passing is to iteratively update node representations by aggregating information from neighbouring nodes. Mathematically, the update rule for a node v_i at the l-th layer can be expressed as Eq.(1).

$$h_i^{(l+1)} = \sigma\left(\sum_{j\in N(v_i)} f^{(l)}\left(h_i^{(l)}, h_j^{(l)}, e_{ij}\right)\right) \quad (1)$$

where, $h_i^{(l)}$represents the hidden state of the node $v_i$at layer $l, N(v_i)$ denotes the set of neighbouring nodes, $f^{(l)}$is a message aggregation function, and e_{ij}represents the edge connecting nodes v_i and v_j. The aggregation function $f^{(l)}$ encapsulates the essence of message passing, capturing hidden states and edge information interplay.

Attention mechanisms are introduced to advance beyond conventional message passing. The attention mechanism assigns weights to neighbouring nodes, allowing the model to focus on more informative nodes during aggregation. The attention weight $a_{ij}^{(l)}$for the edge between nodes v_i and $v_j$is computed as Eq.(2).

$$a_{ij}^{(l)} = softmax(a^{(l)}\left(h_i^{(l)}, h_j^{(l)}\right)) \quad (2)$$

where, $a^{(l)}$ represents a learnable attention function. The final aggregated message is computed using Eq.(3) as a weighted sum.

$$m_i^{(l)} = \sum_{j\in N(v_i)} a_{ij}^{(l)} . f^{(l)}\left(h_i^{(l)}, h_j^{(l)}, e_{ij}\right) \quad (3)$$

Incorporating attention mechanisms allows the GNN to attend to relevant neighbours dynamically, enhancing its ability to capture essential graph structures. A recursive message-passing formulation is introduced to capture higher-order dependencies using Eq.(4).

$$m_i^{(l)} = \sum_{j\in N(v_i)} a_{ij}^{(l)} . f^{(l)}\left(h_i^{(l)}, h_j^{(l)}, e_{ij}\right) + \sum_{j\in N(v_i)} \sum_{k\in N(v_j)} a_{ij}^{(l)} . a_{jk}^{(l)} . f^{(l)}\left(h_i^{(l)}, h_j^{(l)}, h_k^{(l)}, e_{ij}, \quad (4)$$

This recursive formulation allows the GNN to capture interactions involving nodes beyond immediate neighbours, enabling a more comprehensive understanding of the graph's structural intricacies.

3.1.2. Graph Attention Mechanisms

Graph Attention Mechanisms represent a significant advancement in the field, introducing a principled way to assign varying levels of importance to different neighbours during the message aggregation process. This step not only refines the model's ability to capture nuanced relationships but also introduces a layer of adaptability by allowing the network to adjust its focus dynamically. The core mathematical formulation of attention mechanisms involves the calculation of attention weights for each neighbouring node. For a given node v_i, and its neighbor v_j at layer l, the attention weight $a_{ij}^{(l)}$ is computed using Eq.(5).

$$a_{ij}^{(l)} = \frac{exp\left(LeakyReLU\left((a^{(l)^T}[Wh_i^{(l)}, Wh_j^{(l)}]\right)\right)}{\sum_{k\in N(v_i)} expexp\left(LeakyReLU\left((a^{(l)^T}[Wh_i^{(l)}, Wh_k^{(l)}]\right)\right)} \quad (5)$$

where, $a^{(l)}$and W are learnable parameters, LeakyReLUis the leaky rectified linear unit activation function, and $[Wh_i^{(l)}, Wh_j^{(l)}]$denotes the concatenation of transformed node representations. The attention weight is computed based on the compatibility between the representations of the target node and its neighbours.

The overall message aggregation for a node v_i is then obtained by considering the weighted sum of the neighbour representations as expressed in Eq.(6).

$$m_i^{(l)} = \sum_{j\in N(v_i)} a_{ij}^{(l)}.Wh_j^{(l)} \quad (6)$$

Incorporating attention mechanisms into the message-passing process allows the GNN to focus on more relevant neighbours while downplaying the influence of less informative ones. This adaptability enhances the model's capacity to capture intricate graph structures. Multi-head attention is often employed to introduce further flexibility. The attention heads operate in parallel, each with its set of learnable parameters, and the results are concatenated using Eq.(7) to form the final aggregated message.

$$m_i^{(l)} = Contact\left(W_1 m_i^{(l)i}, \dots, W_h m_i^{(l)h}\right) \quad (7)$$

where, $W_1, \dots, W_h$are learnable parameters and $m_i^{(l)i}, \dots, m_i^{(l)h}$ represent the outputs of different attention heads.

3.1.3. GraphSAGE Variants

Building upon the foundation of advanced message passing and the incorporation of attention mechanisms, the third step explores GraphSAGE variants within Graph Neural Networks (GNNs). GraphSAGE, a well-established GNN architecture, forms the basis for this step, and its variants are introduced to enhance the model's ability to capture and generalize information from graph-structured data.

The initial formulation of GraphSAGE involves sampling a fixed-size neighborhood around each node and aggregating information from these neighbours to update the node's representation. Mathematically, for a node v_i and its sampled neighbours $N_i$at layer l, the aggregation is given by Eq.(8).

$$m_i^{(l)} = AGG\left(\left\{W.h_j^{(l-1)} | j \in N_i\right\}\right) \quad (8)$$

where AGG is an aggregation function, W is a learnable weight matrix, and $h_j^{(l-1)}$ represents the hidden state of the node $v_j$at the previous layer.

Variants are introduced to explore different sampling strategies and aggregation functions to extend the capabilities of GraphSAGE. One such variant involves inductive learning, where the model is trained to generalize to previously unseen nodes during inference. The aggregation step for inductive GraphSAGE can be represented as Eq.(9).

$$m_i^{(l)} = AGG\left(\left\{W.h_j^{(l-1)} | j \in N_i \cup [v_i]\right\}\right) \quad (9)$$

The addition of the target node v_i the aggregation allows the model to incorporate information about the node itself, making it more adept at handling new or unseen nodes. The different sampling strategies are explored to diversify the neighbours used in aggregation. Instead of a fixed-size neighbourhood, nodes can be sampled uniformly or based on importance. The aggregation for a node v_i with uniformly sampled neighbours is given by Eq.(10).

$$m_i^{(l)} = AGG\left(\left\{W.h_j^{(l-1)} | j \in UniformSample\ (N_i)\right\}\right) \quad (10)$$

where, UniformSample denotes the uniform sampling function.

The model can employ different aggregation functions to capture diverse neighbourhood structures. One such function is the

mean aggregation, which computes the mean of the sampled neighbours representations. Eq.(11) provides a different perspective on information propagation within the graph.

$$m_i^{(l)} = \frac{1}{|N_i|}\sum_{j\in N_i} W.h_j^{(l-1)} \quad (11)$$

3.1.4. Graph Pooling Techniques

The foundational concept of graph pooling involves aggregating information from group nodes to create a coarser graph representation. Mathematically, the aggregation process for a pooled node $v_i^{(l+1)}$at layer, $l+1$ is expressed as Eq.(12).

$$v_i^{(l+1)} = POOL\left(\left\{h_j^{(l)} | j \in N_i\right\}\right) \quad (12)$$

where, $POOL$ is the pooling function, and $h_j^{(l)}$represents the hidden state of the node v_j at layer l. The pooling function determines how information from the neighbours is aggregated to form the representation of the pooled node.

One prevalent graph pooling technique is the GraphSAGE-style pooling, expressed as Eq.(13), where a node's representation is obtained by concatenating its hidden state with its neighbors' mean or max-pooled representations.

$$v_i^{(l+1)} = Concat\left(h_i^{(l)}, POOL\left(\left\{h_j^{(l)} | j \in N_i\right\}\right)\right) \quad (13)$$

This technique allows the model to retain information about individual nodes while incorporating aggregated knowledge from the neighbourhood.

Differentiable pooling methods are introduced to explore hierarchical structures further. One such method involves assigning scores to nodes and selecting a subset of nodes based on these scores. The selection process is differentiable, allowing gradients to be backpropagated through the pooling layer. The selection scores $s_j^{(l)}$are computed as Eq.(14).

$$s_i^{(l)} = \sigma\left(u^T.tanh\left(W.h_j^{(l)}\right)\right) \quad (14)$$

where$u$ and W are learnable parameters, and σ is the sigmoid activation function.

The pooled representation is then computed as a weighted sum of the selected nodes' by applying Eq.(15). This differentiable pooling approach allows the model to focus on informative nodes while downscaling the graph.

$$v_i^{(l+1)} = \sum_{j\in N_i} s_j^{(l)} h_j^{(l)} \quad (15)$$

Adaptive pooling strategies are explored, where the model learns the pooling operation during training. The pooling weights are parameterized and optimized as part of the learning process using Eq.(16).

$$v_i^{(l+1)} = \sum_{j\in N_i} \alpha_j^{(l)} h_j^{(l)} \quad (16)$$

where, $\alpha_j^{(l)}$represents the learnable pooling weight associated with the node $v_j$at layer l.

3.1.5. Graph Convolutional Networks (GCNs)

The essence of a graph convolutional layer in a GCN lies in the propagation of information from neighbouring nodes to update the representation of a target node. Mathematically, the update rule for a node $v_i$at layer, l is expressed as Eq.(17).

$$h_i^{(l+1)} = \sigma\left(W.\left(\sum_{j\in N_i} \frac{1}{\sqrt{|N_i|.|N_i|}}.h_j^{(l)}\right)\right) \quad (17)$$

where$W$ is a learnable weight matrix, σ represents the activation function, and $h_j^{(l)}$ denotes the hidden state of the neighbouring node $v_j$at layer l. The normalization factor $\frac{1}{\sqrt{|N_i|.|N_i|}}$ensures stable training by addressing the diminishing gradient problem.

Various variations are introduced to enhance the expressive power of GCNs. One such variant involves ChebNet[73], where graph convolution is formulated as a polynomial function of the graph Laplacian. The Chebyshev polynomial of order K is applied to approximate spectral filters using Eq.(18).

$$h_i^{(l+1)} = \sigma\left(W.\left(\sum_{k=0}^{k-1} \theta_k.T_k\left(\hat{L}\right).h_i^{(l)}\right)\right) \quad (18)$$

where, $T_k\left(\hat{L}\right)$represents the k-th Chebyshev polynomial of the normalized Laplacian $\hat{L}$, and $\theta_k$denotes the filter coefficients to be learned.

Another variant introduces ARMA-GCN, incorporating autoregressive moving average filters to capture long-range dependencies in the graph, which is expressed as Eq.(19).

$$h_i^{(l+1)} = \sigma\left(W.ARMA\left(\hat{L}\right).h_i^{(l)}\right) \quad (19)$$

The ARMA filter $ARMA\left(\hat{L}\right)$is parameterized and optimized during training, allowing the model to learn the filter characteristics adaptively. Depth-wise separable convolutions are explored to reduce model complexity while maintaining expressive power. The convolution operation is decomposed into depth-wise and point-wise convolutions, where it uses Eq.(20)

$$h_i^{(l+1)} = \sigma\left(DepthwiseConv\left(\hat{L}\right).PointwiseConv\left(W, h_i^{(}\right. \quad (20)$$

where, $DepthwiseConv\left(\hat{L}\right)$represents the depth-wise convolution operation, and $PointwiseConv\left(W, h_j^{(l)}\right)$is the point-wise convolution.

3.1.6. Graph Isomorphism Networks (GIN)

The fundamental objective of GINs lies in fostering the invariance to the order of node neighbours, a key aspect in graph-based learning tasks. The GIN aggregation operation is expressed mathematically as Eq.(21).

$$h_i^{(l+1)} = MLP\left(\left(1+\epsilon^{(l)}\right).h_i^{(l)} + \sum_{j\in N_i} h_j^{(l)}\right) \quad (21)$$

where MLP denotes a multi-layer perceptron, $\epsilon^{(l)}$is a learnable parameter, and $h_i^{(l)}$and $h_j^{(l)}$ are the hidden states of nodes v_i and $v_j$at layer l, respectively. The use of $\epsilon^{(l)}$introduces a self-loop term, ensuring that each node contributes to its aggregation.

To augment GINs and facilitate greater model expressiveness, various enhancements are explored. One such enhancement involves introducing a non-linearity within the aggregation process. The revised aggregation operation incorporates a non-linear function, typically a simple activation like a ReLU function expressed as Eq.(22)

$$h_i^{(l+1)} = ReLU\left(MLP\left(\left(1+\epsilon^{(l)}\right).h_i^{(l)} + \sum_{j\in N_i} h_j^{(l)}\right)\right) \quad (22)$$

This non-linear transformation enables GINs to capture more intricate relationships within the graph. An additional aggregation step is introduced using Eq.(23) to refine the expressive power further, incorporating an aggregation function that captures the set union of node features.

$$h_i^{(l+1)} = MLP\left(Concat\left(h_i^{(l)}, \sum_{j\in N_i} h_j^{(l)}\right)\right) \quad (23)$$

This formulation explicitly accounts for the presence of each node's features in the aggregation process, contributing to improved model fidelity. An alternative variant employs Eq.(24) to perform node-wise summation rather than concatenation. This variant simplifies the aggregation process while maintaining the essence of incorporating both local and neighbour information.

$$h_i^{(l+1)} = MLP\left(h_i^{(l)} + \sum_{j\in N_i} h_j^{(l)}\right) \quad (24)$$

3.1.7. Graph Embedding

The crux of graph embedding lies in transforming the discrete and complex graph structures into continuous vector representations. A fundamental approach involves leveraging random walk-based methods, such as node2vec or DeepWalk, to generate embeddings that capture the local neighbourhood information. Eq.(25) defines the probability of transitioning from node $v_i$to node $v_j$within a random walk as:

$$P\left(v_j|v_i\right) = \frac{exp\left(\theta.Sim(h_i,h_j)\right)}{\sum_{k\in N_i} exp(\theta.Sim(h_i,h_k))} \quad (25)$$

where, h_i and $h_j$denote the embeddings of nodes v_i and v_j, Sim represents a similarity function, and θ is a tunable parameter controlling the balance between breadth-first and depth-first exploration during the random walk.

Graph autoencoders provide another avenue for graph embedding, learning representations by reconstructing the input graph. The loss function for graph autoencoders is defined as the mean squared error between the input adjacency matrix A and the reconstructed adjacency matrix $\hat{A}$. Eq.(26) mathematically expresses the Graph Encoders.

$$L_{Autoencoder} = \frac{1}{|v|^2}\sum_{i,j\in v}\left(A_{ij} - \hat{A}_{ij}\right)^2 \quad (26)$$

where, $|v|$represents the number of nodes in the graph.

GraphSAGE can be extended for graph embedding, providing a scalable approach to generating node embeddings. The node embedding caused by GraphSAGE is defined as Eq.(27).

$$h_v = \sigma\left(\frac{1}{|N(v)|}\sum_{u\in N(v)} W.h_u\right) \quad (27)$$

whereh_v is the embedding for the node, W is a learnable weight matrix, and σ denotes the activation function.

Employing graph convolutional layers for graph embedding introduces more expressive power. The graph convolutional operation for embedding node $v_i$is given by Eq.(28).

$$h_i = \sigma\left(\sum_{j\in N_i} \frac{1}{\sqrt{|N_i|.|N_i|}} h_j.W\right) \quad (28)$$

where, h_i and $h_j$represent the embeddings of nodes v_i and v_j, $|N_i|$is the number of neighbors of v_i, and W is a learnable weight matrix.

Leveraging attention mechanisms in graph embedding allows the model to focus on more informative neighbours during aggregation, expressed as Eq.(29).

$$h_i = \sigma\left(\sum_{j\in N_i} softmax(a_{ij}).h_j.W\right) \quad (29)$$

The attention weight $a_{ij}$is computed based on the compatibility between the embeddings of nodes v_i and v_j.

3.1.8. Graph Augmentation and Preprocessing

Graph augmentation involves introducing controlled perturbations or modifications to the graph structure, enhancing the diversity of training data and the model's ability to handle variations. Eq.(30) (i.e., augmentation process) introduces a perturbation function P to create a modified graph $G'(v', \epsilon')$.

$$G'(v', \epsilon') = P\big(G(v, \epsilon)\big) \quad (30)$$

where, $v' and\ \epsilon'$represent the augmented set of nodes and edges, respectively.

Eq.(31) involves adding noise to the edge weights of the graph.

$$W'_{ij} = W_{ij} + \epsilon \quad (31)$$

where, $W_{ij}$is the original edge weight between nodes $v_i\ and\ v_j$,and ϵ represents a noise term.

The node dropout ratio is calculated using Eq.(32) and a fraction of nodes are randomly removed from the graph.

$$v' = Dropout(v, p) \quad (32)$$

where$p$ represents the dropout probability.

Graph preprocessing involves preparing the raw graph data for effective learning by applying various transformations. Let A be the adjacency matrix of the graph and X be the feature matrix for nodes. Feature scaling ensures that node features are within a consistent range, preventing the dominance of certain features. A standard method is Min-Max scaling, expressed as Eq.(33).

$$X_{scaled} = \frac{X - \min(X)}{\max(X) - \min(X)} \quad (33)$$

where, $\min(X)$ and $\max(X)$are the minimum and maximum values of the features, respectively.

Initializing node embeddings is crucial for effective learning, Eq.(34) involves using one-hot encoded vectors as initial node embeddings.

$$H_0 = OneHot(v) \quad (34)$$

where, $OneHot(v)$ represents the one-hot encoding of the node set v.

For models considering edge features, initializing edge embeddings can be vital. This can be achieved using Eq.(35), an encoded vector for edge types.

$$E_0 = OneHot(\epsilon) \quad (35)$$

where, $OneHot(\epsilon)$is the one-hot encoding of the edge set ϵ.

3.2. Consistent Fruitfly Optimization

Consistent Fruitfly Optimization (CFO) is a metaheuristic optimization algorithm inspired by the foraging behavior of fruit flies. It mimics the collective intelligence of fruit fly swarms to efficiently search for optimal solutions in complex problem spaces. The Consistent Fruitfly Optimization algorithm employs principles such as attraction to food sources, dispersion to explore diverse areas, and communication among individuals to maintain consistency and convergence towards global optima. CFO has been applied in various fields, including engineering,

optimization problems, and machine learning, demonstrating effectiveness in finding near-optimal solutions even in high-dimensional and nonlinear optimization landscapes. Its adaptability and robustness make Consistent Fruitfly Optimization a valuable tool for tackling diverse optimization challenges.

3.2.1. Initialization

Initialize N number of fruit flies in the population. Let x_i represent the position of the i-th fruitfly in the multidimensional space. Each dimension of x_i corresponds to a variable in the optimization problem. The initialization can be defined as follows:

$$For\ i = 1\ to\ N, initialize\ x_i(0)\ as\ a\ random\ vecto \quad (36)$$

Eq.(36) signifies the generation of the initial positions of the fruitflies, denoted as $x_i(0)$, at time $t = 0$. The subscript i enumerates the individual fruitflies in the swarm. Additionally, let $X(0)$ represent the matrix containing the initial positions of all fruitflies which are expressed as Eq.(37).

$$X(0) = \begin{bmatrix} X_1(0) \\ X_1(0) \\ \vdots \\ X_N(0) \end{bmatrix} \quad (37)$$

where $X(0)$ encapsulates the entire population's positions at the initial time step, forming the starting point for the optimization process. Introduce a function $f(x)$ to represent the objective function. The objective function is a mapping from the multidimensional solution space to fundamental values, reflecting the performance or fitness of a solution.

$$f(x): R^D \rightarrow R \quad (38)$$

where$D$ denotes the dimensionality of the solution space, and the consistency parameters $\alpha, \beta,$ and ϵ are involved in the subsequent stages of the algorithm. They are determined based on the characteristics of the optimization problem.

3.2.2. Objective Function Evaluation

At this stage, each fruitfly's position in the multidimensional space undergoes scrutiny through the objective function $f(x)$. The objective function serves as the metric by which the algorithm gauges the quality of a solution, guiding the fruitflies in their quest for optimal configurations. The evaluation of the objective function for the i-th fruitfly at time t is expressed as Eq.(39).

$$f_i(t) = f(x_i(t)) \quad (39)$$

Eq.(39) captures the essence of the Objective Function Evaluation step. The function $f(x)$ maps the position $x_i(t)$ of the fruitfly i at time t to a real-valued fitness or performance measure $f_i(t)$. The subscript i denotes the individual fruitfly within the population. Extending this evaluation to all fruitflies in the population, this research obtains a vector of fitness values using Eq.(40).

$$F(t) = \begin{bmatrix} f_1(t) \\ f_2(t) \\ \vdots \\ f_N(t) \end{bmatrix} \quad (40)$$

The matrix $F(t)$ captures the fitness landscape of the entire population at time t, reflecting the performance of each fruitfly. In the optimization context, the goal is to minimize the objective function. Thus, the algorithm seeks to find the configuration of fruitfliesthat results in the lowest fitness values. The best fitness value in the current iteration is denoted as $f_{best}(t)$ and it is calculated using Eq.(41).

$$f_{best}(t) = min(F(t)) \quad (41)$$

Eq.(41) identifies the minimum fitness value in the population, representing the best solution found by the swarm at time t. The corresponding position of the fruitfly associated with this optimal fitness is denoted as x_{best} and calculated using Eq.(42).

$$x_{best}(t) = x_i(t)\ where\ f_i(t) = f_{best}(t) \quad (42)$$

where the index i corresponds to the fruitfly achieving the best fitness.

3.2.3. Calculation of Consistency Parameters

The third phase of the CFS algorithm involves the computation of the consistency parameters. These parameters, namely $\alpha, \beta,$and ϵ, play a pivotal role in shaping the movement and adaptability of fruitflies within the multidimensional solution space. Determining these parameters is critical for enhancing the algorithm's stability and performance across various

optimization scenarios. The calculation of the consistency parameters begins with the initialization of these values. Let α_0, β_0, andϵ_0 denote the initial values of α, β, and ϵ, respectively. These values are determined based on the characteristics of the optimization problem and can be adjusted to achieve the desired balance between exploration and exploitation. Eq.(43) expresses the same.

$$\begin{aligned} \alpha(t) &= \alpha_0 \\ \beta(t) &= \beta_0 \\ \epsilon(t) &= \epsilon_0 \end{aligned} \quad (43)$$

The stability and consistency of the algorithm are influenced by the gradual adaptation of these parameters over time. A common approach involves introducing decay mechanisms that reduce the influence of the initial values as the algorithm progresses through iterations. Exponential decay can be applied to $\alpha(t)$ over time and Eq.(44) represents the same.

$$\alpha(t) = \alpha_0 . e^{-\lambda_1 . t} \quad (44)$$

Algorithm 3: Calculate Consistency Parameters

Input:

- α_0: Initial value for the consistency parameter α
- β_0: Initial value for the consistency parameterβ
- ϵ_0: Initial value for the consistency parameter ϵ
- λ_1: Decay rate parameterfor α.
- λ_2: Decay rate parameter for β.
- λ_3: Decay rate parameter for ϵ.
- α_{min}: Minimum allowable value for α.
- β_{min}: Minimum allowable value for β.
- ϵ_{min}: Minimum allowable value forϵ.

Output:

- $\alpha(t)$: Updated value for α at iteration t.
- $\beta(t)$: Updated value for β at iteration t.
- $\epsilon(t)$: Updated value for ϵ at iteration t.

Procedure:

1. Set $\alpha(t), \beta(t)$, and $\epsilon(t)$ to their initial values: $\alpha(t) = \alpha_0, \beta(t) = \beta_0, \epsilon(t) = \epsilon_0$.
2. Apply exponential decay to $\alpha(t)$: $\alpha(t) = \alpha_0 . e^{-\lambda_1 . t}$.
3. Apply exponential decay to $\beta(t)$: $\beta(t) = \beta_0 . e^{-\lambda_2 . t}$.
4. Apply exponential decay to $\epsilon(t)$: $\epsilon(t) = \epsilon_0 . e^{-\lambda_3 . t}$.
5. Impose constraints on the minimum values:
   - If $\alpha(t) < \alpha_{min}, set \alpha(t) = \alpha_{min}$.
   - If $\beta(t) < \beta_{min}, set \alpha(t) = \beta_{min}$.
   - If $\epsilon(t) < \epsilon_{min}, set \alpha(t) = \epsilon_{min}$.
6. Return the updated values for $\alpha(t), \beta(t)$,and $\epsilon(t)$.

where, λ_1 is a decay rate parameter controlling the speed of decay, and t represents the current iteration.

Eq.(45) is applied to $\beta(t)$ and $\epsilon(t)$ to ensure a systematic reduction in their influence as the optimization process unfolds.

$$\begin{aligned} \beta(t) &= \beta_0 . e^{-\lambda_2 . t} \\ \epsilon(t) &= \epsilon_0 . e^{-\lambda_3 . t} \end{aligned} \quad (45)$$

The parameters λ_2 and $\lambda_3$govern the decay rates for $\beta(t)$ and $\epsilon(t)$ respectively. The adaptability of the algorithm is further refined by introducing constraints on the minimum values that $\alpha(t)$, $\beta(t)$, and $\epsilon(t)$ can attain. Eq.(46) prevents the parameters from diminishing too rapidly, maintaining a balance between exploration and exploitation throughout the optimization process

$$\begin{aligned} \alpha(t) &= max\big(\alpha_{min}, \alpha(t)\big) \\ \beta(t) &= max\big(\beta_{min}, \beta(t)\big) \\ \epsilon(t) &= max\big(\epsilon_{min}, \epsilon(t)\big) \end{aligned} \quad (46)$$

where, $\alpha_{min}, \beta_{min}$ and ϵ_{min}represent the minimum allowable values for $\alpha(t), \beta(t)$, and $\epsilon(t)$ respectively.

3.2.4. Fruitfly Movement

Fruitfly movement is infused with consistency enhancements, ensuring a balance between exploration and exploitation. Let $x_i(t)$ represent the position of the i-th fruitfly at iteration t in the D-dimensional solution space. Eq.(47) is designed to facilitate the transition from the current position to a new one, incorporating three distinct components: attraction towards the best solution (α), cohesion with neighbouring fruitflies (β), and a random perturbation (ϵ) for diversity.

$$x_i(t+1) = x_i(t) + \alpha(t).\big(b - x_i(t)\big) + \beta(t).\sum_{j=1}^{N}\big(x_j(t) - x_i(t)\big) + \epsilon(t).\Delta x_i(t) \quad (47)$$

where, $\alpha(t)$ controls the attraction towards the global best solution $b, \beta(t)$ governs the cohesion with neighbouring fruitflies, and $\epsilon(t)$ introduces a random perturbation for exploration. The summation term represents the influence of neighbouringfruitflies, promoting a collective movement.

The term $\Delta x_i(t)$ represents a random perturbation in the multidimensional space, injecting stochasticity into the movement. This perturbation ensures that the algorithm explores diverse regions of the solution space, preventing convergence to local optima. Eq.(48) is crucial for maintaining adaptability and preventing stagnation in the optimization process.

$$\Delta x_i(t) = \delta_i(t).(u_i(t) - I_i(t)) \quad (48)$$

The random perturbation$\Delta x_i(t)$ is calculated based on the difference between the upper $(u_i(t))$ and lower $(I_i(t))$bounds for each dimension, scaled by a random vector $\delta_i(t)$. The upper and lower bounds ($u_i(t)$and $I_i(t)$) ensure that the movement stays within the feasible region of the solution space.

$$u_i(t) = x_i(t) + \gamma.(b - x_i(t))$$
$$I_i(t) = x_i(t) + \gamma.(b - x_i(t)) \quad (49)$$

whereγ is a parameter that controls the scaling of the movement towards the global best solution.

Algorithm 4: Fruitfly Movement

Input:

- Current position of the fruitfly$x_i(t)$ in the solution space.
- Attraction parameter $\alpha(t)$
- Cohesion parameter $\beta(t)$.
- Random perturbation parameter $\epsilon(t)$.
- Upper and lower bounds for each dimension $u_i(t)$ and $I_i(t)$.
- Scaling parameter for upper and lower bounds γ.

Output:

- Updated position of the fruitfly$x_i(t+1)$ after the movement.

Procedure:

1. Calculate the attraction term towards the global best solution:
2. Calculate the cohesion term with neighbouringfruitflies
3. Calculate the random perturbation term
4. Update the position of the fruitfly
5. Ensure the updated position stays within the feasible region
6. 6. Return the updated position $x_i(t+1)$.

3.2.5. Local Search Mechanism.

The local search mechanism involves the modification of the fruitfly's position based on local information and the characteristics of the optimization landscape. Let $x_i(t)$ denote the current position of the i-th fruitfly at iteration t. The local search is guided Eq.(50) and it is a perturbation term that explores the local vicinity of the current position.

$$x_i^{local}(t+1) = x_i(t) + \lambda.\Delta_{local} \quad (50)$$

where, $x_i^{local}(t+1)$ represents the updated position after the local search, λ is a parameter controlling the step size of the search, and Δ_{local} is a random perturbation vector.

The random perturbation vector Δ_{local} introduces variability in the local search, preventing the algorithm from getting stuck in local minima. Its components are randomly generated within certain bounds and Eq.(51) expresses the same.

$$\Delta_{local} = [\delta_{local,1}, \delta_{local,2}, \dots, \delta_{local,D}] \quad (51)$$

Each $\delta_{local,i}$ is drawn from a uniform distribution within the local search bounds $[-a_{local}, a_{local}]$, where a_{local} is a user-defined parameter determining the extent of the local search. The updated position $x_i^{local}(t+1)$is then subjected to an Eq.(52) which is an evaluation to determine its fitness.

$$f_{local}(x_i^{local}(t+1)) \quad (52)$$

The fitness of the locally searched position is compared to the fitness of the original position$(f_i(t))$, and the fruitfly retains the position using Eq.(53) and it yields better fitness. This comparison ensures that the fruitfly moves to a position that improves or maintains its fitness, reinforcing the exploitation of promising regions in the local neighbourhood.

$$x_i(t+1) = \arg min\{f_i(t), f_{local}(x_i^{local}(t+1))\} \quad (53)$$

Algorithm 5: Local Search Mechanism Algorithm

Input:
- Current position of the fruitfly $x_i(t)$ in the solution space.
- Local search parameter λ controlling the step size.
- Local search bounds parameter a_{local} determining the extent of the local search.

Output:
- Updated position of the fruitfly $x_i(t+1)$ after the local search.

Procedure:
1. Generate a random perturbation vector Δ_{local} with components drawn from a uniform distribution within the bounds $[-a_{local}, a_{local}]$.
2. Perform local perturbation on the current position:
3. Evaluate the fitness of the locally searched position:
4. Compare the fitness of the locally searched position with the fitness of the original position:
   - If $f_{local}\left(x_i^{local}(t+1)\right) < f_i(t)$:
   - Update the position: $x_i(t+1) = x_i^{local}(t+1)$.
   - Otherwise, retain the original position: $x_i(t+1) = x_i(t)$.
5. Return the updated position $x_i(t+1)$ after the local search.

3.2.6. Global Search Mechanism

The global search mechanism encompasses integrating information from the entire swarm to influence the movement of each fruitfly. Let $x_i(t)$ denote the position of the i-th fruitfly at iteration t. The global search is characterized by a term that directs fruitflies towards the global best solution, denoted as G. The movement equation for the global search is formulated as Eq.(54).

$$x_i^{global}(t+1) = x_i(t+1) + \gamma.(G - x_i(t+1)) \quad (54)$$

where $x_i^{global}(t+1)$ represents the updated position after the global search, γ is a parameter controlling the influence of the global search, and G is the position corresponding to the global best solution found by the entire swarm.

The movement towards G ensures that fruitflies are guided by the collective knowledge of the swarm, promoting exploration in regions that exhibit superior performance. The position G is determined based on the fitness values of all fruitflies in the population. The global best fitness $(f_{best}(t))$ and the corresponding position $x_{best}(t)$ are identified using Eq.(55).

$$f_{best}(t) = \min(F(t)) \quad x_{best}(t) = x_i(t) where\ f_i(t) = f_{best}(t) \quad (55)$$

These values are then utilized in Eq.(56) global search equation, driving fruitflies towards the position associated with the global best fitness.

$$F = x_{best}(t) \quad (56)$$

The parameter γ regulates the extent to which fruitflies are influenced by the global best solution. Its value determines the balance between exploration and exploitation, with higher values promoting more extensive exploration.

Algorithm 6: Global Search Mechanism Algorithm

Input:
- Current position of the fruitfly $x_i(t+1)$ after the local search.
- Global search parameter γ.
- Fitness values of all fruitflies in the population $F(t)$.
- Positions of all fruitflies in the population $X(t)$.

Output:
- Updated position of the fruitfly $x_i^{global}(t+1)$ after the global search.

Procedure:
1. Identify the global best fitness $f_{best}(t)$ as the minimum fitness value in the population.
2. Determine the position $x_{best}(t)$ associated with the global best fitness.
3. Update the position of the fruitfly after the global search using the global best

solution.
4. Return the updated position $x_i^{global}(t+1)$after the global search.

3.2.7. Objective Function Re-evaluation

The objective function re-evaluation involves applying the fitness function $f(x)$ to the new positions of the fruitflies. Let $x_i^{gloabl}(t+1)$ denotes the position of the i-th fruitfly after the global search mechanism. The fitness of the fruitfly at iteration $t+1$ is given by Eq.(57).

$$f_i^{global}(t+1) = f\left(x_i^{global}(t+1)\right) \quad (57)$$

The fitness after the local search is calculated using Eq.(58).

$$f_i^{local}(t+1) = f\left(x_i(t+1)\right) \quad (58)$$

In this equation, $f_i^{local}(t+1)$represents the fitness of the i-th fruitfly after the local search. The objective function re-evaluation combines the global and local search information, providing an updated fitness landscape for the swarm.

The overall fitness vector after the objective function re-evaluation is denoted as $F^{global}(t+1)$, incorporating the global search results. Eq.(59) provides the overall fitness calculation.

$$F^{global}(t+1) = \begin{bmatrix} f_1^{global}(t+1) \\ f_2^{global}(t+1) \\ \vdots \\ f_N^{global}(t+1) \end{bmatrix} \quad (59)$$

This vector encapsulates the fitness values of all fruitflies after the global search mechanism. Similarly, the fitness vector $F^{local}(t+1)$accounts for the fitness values after the local search is calculated using Eq.(60).

$$F^{local}(t+1) = \begin{bmatrix} f_1^{local}(t+1) \\ f_2^{local}(t+1) \\ \vdots \\ f_N^{local}(t+1) \end{bmatrix} \quad (60)$$

The objective function re-evaluation ensures that the fitness values accurately reflect the performance of each fruitfly in the updated solution space. The synergy between the global and local search mechanisms is encapsulated in these fitness vectors, providing a comprehensive representation of the swarm's effectiveness in exploring and exploiting the solution landscape.

3.2.8. Update Swarm Information

The update of swarm information entails synthesizing the fitness vectors obtained after the global and local search mechanisms. The global fitness vector $F^{global}(t+1)$and local fitness vector $F^{local}(t+1)$are combined to create the comprehensive fitness vector $F(t+1)$ for the next iteration. Eq.(61) is applied to update the swarm information.

$$F(t+1) = min\left(F^{global}(t+1), F^{local}(t+1)\right) \quad (61)$$

This amalgamation ensures that the swarm possesses information about each fruitfly's global and local fitness values. The use of the minimum operation reflects the aspiration to retain the most promising fitness information, emphasizing the pursuit of superior solutions. Following the fitness information update, the fruitflies' positions in the swarm are adjusted based on the global and local search results. The updated position $x_i(t+1)$ is determined by Eq.(62) and it compares the fitness values after global and local search.

$$x_i(t+1) = \begin{cases} x_i^{global}(t+1) & if\ f_i^{global}(t+1) < f_i^{local} \\ x_i^{local}(t+1) & otherwise \end{cases} \quad (62)$$

This decision-making process ensures that each fruitfly adopts the position associated with the superior fitness value, aligning with the principle of seeking the most promising solutions. The global best fitness $f_{best}(t+1)$and the corresponding position $x_{best}(t+1)$ are updated using Eq.(63) and it is based on the information acquired from both global and local searches.

$$f_{best}(t+1) = min\left(f_{best}^{global}(t+1), f_{best}^{local}(t+1)\right)$$
$$x_{best}(t+1) = \begin{cases} x_{best}^{global}(t+1) & if\ f_{best}^{global}(t+1) < f_{best}^{global} \\ x_{best}^{local}(t+1) & otherwise \end{cases} \quad (63)$$

This updating mechanism ensures that the swarm retains the position associated with the globally optimal fitness value. Integrating global and local search results in updating the global best solution enhances the swarm's ability to converge towards high-quality solutions.

Algorithm 7: Update Swarm Information

Input:

- Global fitness vector $F^{global}(t+1)$.
- Local fitness vector $F^{local}(t+1)$.
- Global best fitness $f_{best}^{global}(t+1)$.
- Local best fitness $f_{best}^{local}(t+1)$.
- Global best position $x_{best}^{global}(t+1)$.
- Local best position $x_{best}^{local}(t+1)$.

Output:

- Updated position of each fruitfly$x_i(t+1)$in the swarm.
- Updated global best fitness $f_{best}(t+1)$.
- Updated global best position $x_{best}(t+1)$.

Procedure:

1. Combine the global and local fitness vectors to create the comprehensive fitness vector:
2. For each fruitfly$i$ in the swarm:
   a. Compare the fitness values after global and local searches:
      - If $f_i^{global}(t+1) < f_i^{local}(t+1)$, update the position:
      - Otherwise, update the position.
3. Update the global best fitness and position based on the global and local search outcomes:
4. Return
   - updated positions of all fruitflies in the swarm
   - updated global best fitness $f_{best}(t+1)$
   - updated global best position $x_{best}(t+1)$

3.2.9 Consistency-Driven Termination Criteria

The termination criteria are driven by the consistency of the algorithm's performance over successive iterations. Let ε represent a small positive constant, and $F(t)$ denote the fitness vector at iteration t. The consistency criterion is formulated as:

$$\left|\frac{\sum_{i=1}^{N} f_i(t)}{N} - \frac{\sum_{i=1}^{N} f_i(t-1)}{N}\right| < \epsilon \qquad (64)$$

This equation computes the absolute difference between the average fitness at the current iteration t and the previous iteration $t-1$. The termination criteria are satisfied when this difference falls below a predefined threshold ε. This approach ensures that the algorithm converges when the fitness values exhibit minimal variation, signifying consistency in the optimization process.

The termination criteria can be based on the relative improvement in the global best fitness $(f_{best}(t))$over consecutive iterations. Let δ be another small positive constant. The termination condition is expressed as:

$$\frac{f_{best}(t-1) - f_{best}(t)}{f_{best}(t-1)} < \delta \qquad (65)$$

This condition evaluates the relative improvement in the global best fitness. When the improvement falls below δ, the algorithm terminates. This criterion ensures that the algorithm halts when the improvement in the best fitness becomes negligible, indicating that further iterations may not significantly enhance the solution quality. In addition to these convergence criteria, a maximum number of iterations (T_{max}) can be defined as a termination condition:

$$t \geq T_{max} \qquad (66)$$

This criterion limits the algorithm's number of iterations, preventing excessive computational costs and ensuring a balance between exploration and exploitation.

Algorithm 8: Consistency-Driven Termination Criteria Algorithm

Input:

- Fitness vector at the current iteration $F(t)$.
- Fitness vector at the previous iteration $F(t-1)$.
- Global best fitness at the current iteration $f_{best}(t)$.
- Global best fitness at the previous iteration $f_{best}(t-1)$
- Small positive constants $\varepsilon$and δ.
- Maximum number of iterations T_{max}.

Output:

- The termination flag indicates whether the algorithm should terminate.

Procedure:

1. Compute the average fitness at the current iteration and the average fitness at the previous iteration.
2. Calculate the absolute difference between $\bar{f}(t)$ and $\bar{f}(t-1)$: $\Delta\bar{f} = \left|\bar{f}(t) - \bar{f}(t-1)\right|$.
3. Check if $\Delta\bar{f} < \varepsilon$.If true, set the termination flag to indicate that the algorithm should terminate.
4. Compute the relative improvement in the global best fitness.
5. Check if $Improvement < \delta$.If true, set the termination flag to indicate that the algorithm should terminate.
6. Check if the current iteration t is greater than or equal to the maximum number of iterations T_{max}. If true, set the termination flag to indicate that the algorithm should terminate.
7. Return the termination flag.

4. ABOUT DATASET

The compilation called the Lenskart AR Experience Sentiment Dataset is a rich reservoir of feedback garnered from Lenskart patrons. It comprises 788 entries obtained through surveys or online forms aimed at understanding customer sentiments and interactions, particularly regarding augmented reality (AR) experiences with Lenskart eyewear products. This dataset encapsulates a wide array of information, spanning demographic particulars, ratings, and qualitative expressions of sentiment. It sheds light on customer satisfaction across various aspects like product quality, pricing, customer service, and usage of the online platform. Furthermore, it scrutinizes AR experiences, gauging their level of engagement, influence on purchasing decisions, customization options, recommendations, and suggestions for enhancement. With 21 distinct categories, this dataset provides a comprehensive perspective on customer feedback and preferences, serving as a valuable tool for businesses seeking to tailor their offerings and enhance customer engagement. Researchers can utilize this dataset for sentiment analysis, gaining insights into customer experiences, and devising potential marketing strategies. The authors are willing to share the dataset with interested researchers upon request, emphasizing the importance of ethical considerations such as data privacy and informed consent in its usage.

4.1 Data Collection Process

The dataset was collected through a structured feedback acquisition process targeting users who interacted with Lenskart's augmented reality (AR)-based eyewear platform. Responses were gathered through online surveys and digital feedback forms distributed to customers after their shopping experience. The data collection focused on capturing demographic attributes, customer ratings, product satisfaction, sentiment expressions, and AR interaction experiences. To ensure reliability, incomplete or duplicate entries were removed during preprocessing. The final dataset consisted of 788 valid records, representing diverse user experiences and sentiment perspectives. Ethical considerations, including informed consent and privacy protection, were maintained throughout the data collection process to ensure responsible use of participant information.

5. PERFORMANCE METRICS

In evaluating the performance of models, several key metrics are essential: True Positive (TrP), True Negative (TrN), False Positive (FlP), and False N$egative$ (FlN).

5.1. Precision

Precision ($Prec$) delves into the accuracy of positive predictions made by a model. As shown in Eq.(66), its calculation helps determine the precision percentage.

$$Prec = \frac{TrP}{TrP + FlP} \times 100 \quad (66)$$

5.2. Recall

Recall ($Recl$) measures how effectively a model captures relevant instances of the positive class. Eq.(67) quantifies $Recl$.

$$Recl = \frac{TrP}{TrP + FlP} \times 100 \quad (67)$$

5.3. Classification Accuracy

Classification Accuracy (CA) is a crucial metric for assessing a classification model's correctness. It quantifies the proportion of correctly classified instances among the total predictions using Eq.(68).

$$CA = \left(\frac{(Number\ of\ Correct\ Predictions)}{(Total\ Number\ of\ Predictions)}\right) \times 100 \quad (68)$$

5.4. F-Measure

F-Measure (FM) combines Precision and Recall, offering a balanced assessment, especially beneficial for imbalanced datasets. Eq.(69) presents the formula for calculating FM.

$$FM = \left(2 \times \frac{(Precision \times Recall)}{(Precision + Recall)}\right) \times 100 \quad (69)$$

5.5. Matthews Correlation Coefficient

Matthews Correlation Coefficient (MCC) provides a balanced performance measure useful in imbalanced datasets. It is less affected by class imbalance compared to accuracy. Eq.(70) outlines the MCC calculation formula.

$$MCC = \frac{TrP \times TrN \times FlP \times F}{\sqrt{(TrP + FlP) \times (TrP + FlN) \times (TrN +}} \times 100 \quad (70)$$

5.6. Fowlkes-Mallows Index

Fowlkes-Mallows Index (FMI) is a metric used to evaluate the similarity between two clusters or groups, often applied in clustering algorithms or unsupervised learning scenarios. Eq.(71) defines the FMI calculation formula.

$$FMI = \left(\sqrt{Precision \times Recall}\right) \times 100 \quad (71)$$

6. RESULTS AND DISCUSSION

6.1. *Prec*and*Recl* Analysis

Figure 1 offers a comparative analysis of three distinct classification algorithms: TDAN, DELT, and CFO-GNN. This analysis is based on two vital performance metrics: Prec and Recl. Prec represents the accuracy of the positive predictions made by the algorithms, while Recl measures their ability to identify all the actual positive instances. The significance of Figure 1 lies in its capacity to provide a clear and compact comparison of these algorithms in terms of these crucial metrics. Understanding their relative strengths in Prec and Recl is essential for selecting an algorithm most suitable for specific tasks.

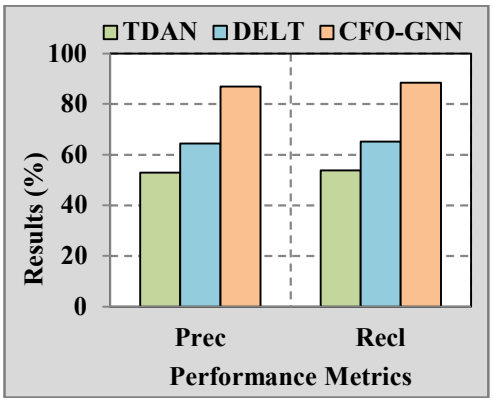

Figure 1.*Prec* and *Recl*

TDAN demonstrates a Precision score of 52.859 and a Recall score of 53.740. The mechanism underlying TDAN's performance revolves around its proficiency in handling time-dependent data. Utilizing techniques like RNNs, it identifies patterns and trends over time, significantly influencing its Prec and Recl outcomes. DELT stands out with a Prec score of 64.326 and a Recl score of 65.136. Its success can be attributed to its adeptness in feature engineering and data preprocessing. DELT likely employs various techniques to optimize features, ensuring higher Prec by accurately identifying positive instances and improving Recl by effectively capturing a more significant proportion of true positives. CFO-GNN showcases remarkable Prec at 86.864 and an impressive Recl of 88.389. The excellence of CFO-GNN can be attributed to its use of GNNs for classification tasks. By effectively leveraging the inherent relationships and dependencies within the data, CFO-GNN achieves higher Prec and Recl, making it a powerful choice.

Figure 1 presents a comparative snapshot of TDAN, DELT, and CFO-GNN performance regarding Precision and Recall. TDAN's strength lies in its ability to handle time-dependent data effectively. DELT excels in feature engineering, optimizing both Precision and Recall. CFO-GNN, leveraging Graph Neural Networks, showcases exceptional performance in both metrics. The selection of the most appropriate algorithm should align with the specific task requirements, considering factors such as the nature of the dataset and the importance of Prec and Recl. This comparative analysis equips decision-makers to make informed choices when selecting the algorithm best suited for a given application, ensuring optimal results in Prec and Recl.

Journal of Theoretical and Applied Information Technology
15st June 2026. Vol.104. No.11

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 1. Precision and Recall

Classification Algorithms	*Prec*	*Recl*
TDAN	52.859	53.740
DELT	64.326	65.136
CFO − GNN	86.864	88.389

6.2. *CA*and*FM* Analysis

Figure 2 offers a comparative analysis of three classification algorithms: TDAN, DELT, and CFO-GNN. This analysis focuses on two vital metrics: CA and FM. CA represents the algorithm's precision in classifying data accurately, while FM signifies the algorithm's ability to balance Prec and Recl. The importance of Figure 2 lies in its capacity to provide a concise yet comprehensive comparison of these algorithms in terms of these significant metrics. Understanding their relative strengths in CA and FM is crucial for making informed decisions when selecting the most suitable algorithm for specific tasks.

TDAN has a CA of 52.833 and an FM of 53.296. Its efficiency stems from its specialization in handling time-dependent data. TDAN likely employs techniques such as RNNs or temporal convolutional networks, allowing it to capture temporal patterns in the data. This contributes to its balanced performance in both CA and FM. DELT stands out with a CA of 65.003 and an FM of 64.728. Its success lies in its expertise in feature engineering and data preprocessing. DELT excels at optimizing input data and enhancing CA and FM. It achieves this by making more accurate predictions and improving the quality and relevance of features used for classification. CFO-GNN showcases exceptional performance with a CA of 87.325 and an impressive FM of 87.620. This excellence can be attributed to its use of GNNs. GNNs empower CFO-GNN to model relationships and dependencies within the data effectively, leading to outstanding scores in CA and FM.

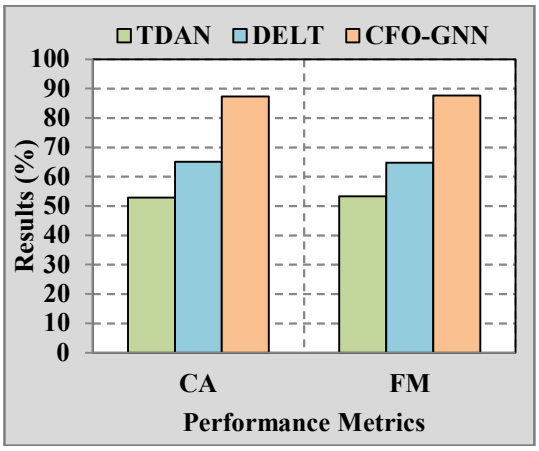

Figure 2. *CA* and*FM*

Table 2. Classification Accuracy and F-Measure

Classification Algorithms	*CA*	*FM*
TDAN	52.833	53.296
DELT	65.003	64.728
CFO − GNN	87.325	87.620

Figure 2 shows TDAN, DELT, and CFO-GNN performance concerning CA and FM. TDAN excels in handling time-dependent data, maintaining a solid balance between accuracy and feature optimization. DELT specializes in feature engineering and preprocessing, enhancing CA and FM. CFO-GNN, leveraging Graph Neural Networks, stands out with exceptional performance in both metrics. When choosing an algorithm, aligning it with the specific task requirements is essential, considering the dataset's nature and the importance of CA and FM. This comparative analysis equips decision-makers to make informed choices when selecting the algorithm best suited for a given application, ensuring optimal results regarding CA and FM.

6.3. *FMI* and *MCC* Analysis

Figure 3 is a critical visual representation that enables an in-depth comparison of three prominent classification algorithms: TDAN, DELT, and CFO-GNN. The importance of Figure 3 lies in its role as a decision support tool, helping stakeholders select the most suitable algorithm for specific machine learning tasks, thus optimizing performance outcomes.

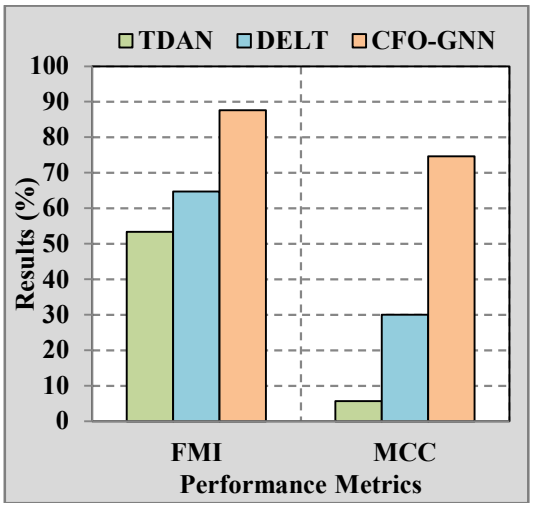

Figure 3.*FMI* and *MCC*

TDAN achieves an FMI score of 53.298% and an MCC score of 5.664%. These results are rooted in its mechanism, which specializes in handling temporal data. TDAN is typically implemented with RNNs. These architectures excel in capturing temporal dependencies within the data, allowing TDAN to identify sequential patterns effectively. However, its lower MCC score suggests that it may face challenges in scenarios involving complex data correlations and imbalanced class distributions. The limited MCC indicates a higher likelihood of false positives and negatives. DELT stands out with an FMI score of 64.730% and an impressive MCC score of 30.007%. Its success can be attributed to its feature engineering and data preprocessing expertise. DELT employs feature selection, dimensionality reduction, and data augmentation techniques. These processes enhance the quality and relevance of features used for classification. DELT achieves higher FMI by more accurately assessing data clusters and higher MCC by effectively managing class imbalances. The superior MCC indicates a better balance between precision and recall in classification. CFO-GNN exhibits outstanding performance, boasting an FMI of 87.623% and an MCC of 74.649%. This exceptional success is a direct result of its utilization of GNNs. CFO-GNN leverages GNNs to model intricate relationships and dependencies within the data effectively. GNNs are adept at capturing complex network structures, making them well-suited for complex data correlation applications. CFO-GNN's impressive MCC indicates its exceptional ability to handle class imbalances and accurately classify data instances.

Figure 3 provides an intricate analysis of TDAN, DELT, and CFO-GNN's performance regarding FMI and MCC. TDAN excels in temporal data processing, DELT's strength lies in feature engineering, and CFO-GNN leverages Graph Neural Networks for modeling complex relationships. The choice of the most suitable algorithm should align with specific task requirements, considering factors such as the nature of the dataset, temporal dependencies, and the significance of managing class imbalances and complex correlations. This comprehensive technical analysis equips decision-makers with precise insights to make informed selections, ensuring optimal results for both FMI and MCC.

Table 3. *FMI* and *MCC*

Classification Algorithms	*FMI*	*MCC*
TDAN	53.298	5.664
DELT	64.730	30.007
CFO − GNN	87.623	74.649

6.4 Comparative Performance Analysis

To evaluate the effectiveness of the proposed CFO-GNN model, its performance was compared with prominent state-of-the-art techniques, including TDAN and DELT. The experimental findings indicate that CFO-GNN consistently outperformed existing approaches across multiple performance metrics, including Precision, Recall, Classification Accuracy, Fowlkes–Mallows Index, and Matthews Correlation Coefficient. The superior performance can be attributed to the integration of graph neural representations capable of modelling relational dependencies and the optimization capability of the Consistent Fruitfly Optimization mechanism. Unlike conventional models that rely primarily on sequential or feature-engineering approaches, CFO-GNN effectively captures hidden structural relationships in sparse sentiment data, leading to improved predictive accuracy and classification robustness.

6.5 Limitations and Challenges

Despite the encouraging performance of CFO-GNN, several limitations should be acknowledged. First, the proposed model was evaluated using a dataset derived from a specific AR-enabled shopping platform, which may limit generalizability across different domains and user demographics. Second, although the model

effectively addresses sparse sentiment conditions, variations in user-generated feedback quality may still influence prediction reliability. The computational complexity associated with graph construction and optimization may also increase processing time for large-scale datasets. Another challenge involves maintaining sentiment consistency in highly dynamic AR shopping environments, where user preferences may evolve rapidly. Future studies may focus on multi-domain validation, lightweight optimization mechanisms, and real-time adaptive learning strategies to further strengthen model performance.

7. CONCLUSION

The Consistent Fruitfly Optimization-based Graph Neural Network (CFO-GNN) presents a promising solution to the challenges of sentiment analysis in AR-enabled online shopping platforms. By integrating fruitfly optimization with graph neural networks, CFO-GNN effectively addresses the issue of sparse review data while capturing the nuances of AR shopping experiences. Through empirical evaluation, CFO-GNN demonstrates its ability to provide accurate sentiment analysis, empowering businesses to make informed decisions and enhance user satisfaction. The proposed algorithm not only contributes to advancements in sentiment analysis within AR environments but also holds potential for broader applications in natural language processing and machine learning. Moving forward, further research can explore optimization techniques and model enhancements to further improve the performance and versatility of sentiment analysis tools in the dynamic landscape of AR-enabled online retail.

REFERENCES

[1] S. S. Coşkun, "Exploring the Gaussian investor sentiment process," *Borsa Istanbul Rev.*, vol. 23, no. 2, pp. 412–425, 2023, doi: 10.1016/j.bir.2022.11.012.

[2] A. E. Khedr, A. A. Almazroi, and A. M. Idrees, "Intelligent Framework for Enhancing the Quality of Online Exams based on Students' Personalization," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 7, pp. 605–614, 2022, doi: 10.14569/IJACSA.2022.0130772.

[3] Z. Wu, Q. He, J. Li, G. Bi, and M. F. Antwi-Afari, "Public attitudes and sentiments towards new energy vehicles in China: A text mining approach," *Renew. Sustain. Energy Rev.*, vol. 178, p. 113242, 2023, doi: 10.1016/j.rser.2023.113242.

[4] N. Lakshmidevi, M. Vamsikrishna, and S. S. Nayak, "Classification of sentiments on online products using deep learning model – RNN," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 1, pp. 7165–7172, 2019, doi: 10.35940/ijeat.A1910.109119.

[5] S. Wang *et al.*, "A global portrait of expressed mental health signals towards COVID-19 in social media space," *Int. J. Appl. Earth Obs. Geoinf.*, vol. 116, 2023, doi: 10.1016/j.jag.2022.103160.

[6] S. M. Sinha, V. P. Kale, R. S. Sangle, S. R. Bhoite, V. G. Kottawar, and P. B. Deshmukh, "Celestial Learning: Secure eLearning platform using Node-Express," in *2021 5th International Conference on Computer, Communication, and Signal Processing, ICCCSP 2021*, 2021, pp. 171–176. doi: 10.1109/ICCCSP52374.2021.9465500.

[7] V. Dogra, F. S. Alharithi, R. M. Álvarez, A. Singh, and A. M. Qahtani, "NLP-Based Application for Analyzing Private and Public Banks Stocks Reaction to News Events in the Indian Stock Exchange," *Systems*, vol. 10, no. 6, 2022, doi: 10.3390/systems10060233.

[8] P. J. Lee, Y. H. Hu, K. Chen, J. Michael Tarn, and L. E. Chen, "Cyberbullying detection on social network services," in *Proceedings of the 22nd Pacific Asia Conference on Information Systems - Opportunities and Challenges for the Digitized Society: Are We Ready?, PACIS 2018*, 2018. [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85088711219&partnerID=40&md5=b6699709bd1b255ca98b29bb9403d656

[9] R. Kumar, P. Kumar, and Y. Kumar, "Integrating big data driven sentiments polarity and ABC-optimized LSTM for time series forecasting," *Multimed. Tools Appl.*, vol. 81, no. 24, pp. 34595–34614, 2022, doi: 10.1007/s11042-021-11029-1.

[10] A. Solairaj, G. Sugitha, and G. Kavitha, "Enhanced Elman spike neural network based sentiment analysis of online product recommendation," *Appl. Soft Comput.*, vol. 132, p. 109789, 2023, doi: 10.1016/j.asoc.2022.109789.

[11] X. J. Li, G. S. Deng, X. Z. Wang, X. L. Wu, and Q. W. Zeng, "A hybrid recommendation algorithm based on user comment sentiment and matrix decomposition," *Inf. Syst.*, vol.

117, p. 102244, 2023, doi: 10.1016/j.is.2023.102244.
[12] P. Mukherjee, Y. Badr, S. Doppalapudi, S. M. Srinivasan, R. S. Sangwan, and R. Sharma, "Effect of Negation in Sentences on Sentiment Analysis and Polarity Detection," *Procedia Comput. Sci.*, vol. 185, pp. 370–379, 2021, doi: 10.1016/j.procs.2021.05.038.
[13] T. M. Omran, B. T. Sharef, C. Grosan, and Y. Li, "Transfer learning and sentiment analysis of Bahraini dialects sequential text data using multilingual deep learning approach," *Data Knowl. Eng.*, vol. 143, 2023, doi: 10.1016/j.datak.2022.102106.
[14] N. Pughazendi, P. V. Rajaraman, and M. H. Mohammed, "Graph Sample and Aggregate Attention Network optimized with Barnacles Mating Algorithm based Sentiment Analysis for Online Product Recommendation," *Appl. Soft Comput.*, vol. 145, p. 110532, 2023, doi: 10.1016/j.asoc.2023.110532.
[15] M. Agrawal and N. R. Moparthi, "An efficient multiple-word embedding-based cross-domain feature extraction and aspect sentiment classification," *Meas. Sensors*, vol. 28, p. 100851, 2023, doi: 10.1016/j.measen.2023.100851.
[16] X. Jia, C. Li, M. Zeng, L. Wang, and Q. Mi, "An improved unified domain adversarial category-wise alignment network for unsupervised cross-domain sentiment classification," *Eng. Appl. Artif. Intell.*, vol. 126, p. 107108, 2023, doi: 10.1016/j.engappai.2023.107108.
[17] Z. Yang, B. Wang, X. Li, W. Wang, and J. Ouyang, "S3MAP: Semisupervised aspect-based sentiment analysis with masked aspect prediction," *Knowledge-Based Syst.*, vol. 269, p. 110513, 2023, doi: 10.1016/j.knosys.2023.110513.
[18] Y. Zhu, Y. Qiu, Q. Wu, F. L. Wang, and Y. Rao, "Topic Driven Adaptive Network for cross-domain sentiment classification," *Inf. Process. Manag.*, vol. 60, no. 2, 2023, doi: 10.1016/j.ipm.2022.103230.
[19] C. Choudhary, I. Singh, and M. Kumar, "SARWAS: Deep ensemble learning techniques for sentiment based recommendation system," *Expert Syst. Appl.*, vol. 216, 2023, doi: 10.1016/j.eswa.2022.119420.
[20] J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," *World J. Eng.*, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.
[21] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," in *Incorporating the Internet of Things in Healthcare Applications and Wearable Devices*, IGI Global, 2019, pp. 109–121. doi: 10.4018/978-1-7998-1090-2.ch006.
[22] J. Ramkumar, R. Karthikeyan, and V. Valarmathi, "Alpine Swift Routing Protocol (ASRP) for Strategic Adaptive Connectivity Enhancement and Boosted Quality of Service in Drone Ad Hoc Network (DANET)," *Int. J. Comput. Networks Appl.*, vol. 11, no. 5, pp. 726–748, 2024, doi: 10.22247/ijcna/2024/45.
[23] R. Jaganathan, S. Mehta, and R. Krishan, *Bio-Inspired intelligence for smart decision-making*. IGI Global, 2024. doi: 10.4018/9798369352762.
[24] N. K. Ojha, A. Pandita, and J. Ramkumar, "Cyber security challenges and dark side of AI: Review and current status," in *Demystifying the Dark Side of AI in Business*, IGI Global, 2024, pp. 117–137. doi: 10.4018/979-8-3693-0724-3.ch007.
[25] J. Ramkumar, R. Karthikeyan, and M. Lingaraj, "Optimizing IoT-Based Quantum Wireless Sensor Networks Using NM-TEEN Fusion of Energy Efficiency and Systematic Governance," in *Lecture Notes in Electrical Engineering*, V. Shrivastava, J. C. Bansal, and B. K. Panigrahi, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 141–153. doi: 10.1007/978-981-97-6710-6_12.
[26] R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.
[27] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," in *ACM International Conference Proceeding Series*, Association for Computing Machinery, 2022. doi: 10.1145/3590837.3590907.
[28] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wirel. Pers. Commun.*, vol. 120, no. 2, pp. 887–909, 2021, doi: 10.1007/s11277-021-08495-z.
[29] J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "Optimal

Approach For Minimizing Delays In Iot-Based Quantum Wireless Sensor Networks Using Nm-Leach Routing Protocol," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 3, pp. 1099–1111, 2024.

[30] K. S. J. Marseline, J. Ramkumar, and D. R. Medhunhashini, "Sophisticated Kalman Filtering-Based Neural Network for Analyzing Sentiments in Online Courses," in *Smart Innovation, Systems and Technologies*, A. K. Somani, A. Mundra, R. K. Gupta, S. Bhattacharya, and A. P. Mazumdar, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 345–358. doi: 10.1007/978-981-97-3690-4_26.

[31] J. Ramkumar, R. Karthikeyan, and K. O. Nitish, "Securing Library Data With Blockchain Advantage," in *Enhancing Security and Regulations in Libraries with Blockchain Technology*, 2024, pp. 117–138. doi: 10.4018/979-8-3693-9616-2.ch006.

[32] B. Suchitra, R. Karthikeyan, J. Ramkumar, and V. Valarmathi, "Enhancing Recurrent Neural Network Performance for Latent Autoimmune Diabetes Detection (Lada) Using Exocoetidae Optimization," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 5, pp. 1645–1667, 2025.

[33] D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *Int. J. Comput. Networks Appl.*, vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.

[34] J. Joy, S. Devaraju, And J. Ramkumar, "Utilizing Fuzzy-Identity-Based Encryption With Proxy-Re-Encryption For Data Sharing," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 18, pp. 7469–7479, 2025.

[35] R. Jaganathan, S. Rajagopal, and K. Rajendran, "Cultural Intelligence in the AI Era-Enhancing Transitional Higher Education," in *Bridging Global Divides for Transnational Higher Education in the AI Era*, 2024, pp. 273–292.

[36] J. Ramkumar and V. Valarmathi, "Harnessing AI-Driven Models for Sustainable Development in Business Management," in *World Sustainability Series*, 2025, pp. 217–238.

[37] B. Suchitra, J. Ramkumar, and R. Karthikeyan, "Frog Leap Inspired Optimization-Based Extreme Learning Machine for Accurate Classification of Latent Autoimmune Diabetes in Adults (LADA)," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 2, pp. 472–494, 2025.

[38] M. P. Swapna and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," in *Communications in Computer and Information Science*, S. Rajagopal, K. Popat, D. Meva, and S. Bajeja, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 131–140. doi: 10.1007/978-3-031-59100-6_11.

[39] J. Ramkumar, V. Valarmathi, and R. Karthikeyan, "Optimizing Quality of Service and Energy Efficiency in Hazardous Drone Ad-Hoc Networks (DANET) Using Kingfisher Routing Protocol (KRP)," *Int. J. Eng. Trends Technol.*, vol. 73, no. 1, pp. 410–430, 2025, doi: 10.14445/22315381/IJETT-V73I1P135.

[40] M. P. Swapna, J. Ramkumar, and R. Karthikeyan, "Energy-Aware Reliable Routing with Blockchain Security for Heterogeneous Wireless Sensor Networks," in *Lecture Notes in Networks and Systems*, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 713–723. doi: 10.1007/978-981-97-6106-7_43.

[41] M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[42] R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[43] J. Ramkumar, B. Varun, V. Valarmathi, D. R. Medhunhashini, and R. Karthikeyan, "Jaguar-Based Routing Protocol (Jrp) For Improved Reliability And Reduced Packet Loss In Drone Ad-Hoc Networks (DANET)," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 2, pp. 696–713, 2025,

[44] J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," *Int. J. Comput. Networks Appl.*, vol. 10, no. 4, pp. 668–687, 2023, doi: 10.22247/ijcna/2023/223319.

[45] J. Ramkumar and R. Vadivel, "CSIP—cuckoo

search inspired protocol for routing in cognitive radio ad hoc networks," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2017, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.

[46] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[47] J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," *Int. J. Comput. Networks Appl.*, vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[48] J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in *Lecture Notes in Electrical Engineering*, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[49] M. P. Swapna, D. Rajeev, J. Ramkumar, and S. Chandran, "Unveiling Cybercrime Patterns in Kerala: A Machine Learning Approach," in *Lecture Notes in Networks and Systems*, 2026, pp. 111–122.

[50] V. Viswanathan and R. Jaganathan, "Chapter 3 - Sustainable 6G connectivity through artificial intelligence powered energy efficiency," in *Computational Modeling Applications for Existential Risks*, P. K. Dutta, P. Raj, P. Bhattacharya, I. Budhiraja, and D. B. T.-S. P. R. Kaplun, Eds., Morgan Kaufmann, 2026, pp. 29–42. doi: https://doi.org/10.1016/B978-0-443-33172-5.00011-X.

[51] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[52] R. Jaganathan, K. Rajendran, and P. S. Ponnukumar, "Peregrine Falcon Optimization Routing Protocol (PFORP) for Achieving Ultra-Low Latency and Boosted Efficiency in 6G Drone Ad-Hoc Networks (DANET)," *Int. J. Comput. Digit. Syst.*, vol. 17, no. 1, pp. 1–18, 2025, doi: 10.12785/ijcds/1571111848.

[53] V. Valarmathi and J. Ramkumar, "Modernizing Wildfire Management Through Deep Learning and IoT in Fire Ecology," in *Machine Learning and Internet of Things in Fire Ecology*, 2024, pp. 203–229. doi: 10.4018/979-8-3693-7565-5.ch0010.

[54] J. Ramkumar and D. Ravindran, "Machine learning and robotics in urban traffic flow optimization with graph neural networks and reinforcement learning," in *Machine Learning and Robotics in Urban Planning and Management*, 2025, pp. 83–104. doi: 10.4018/979-8-3693-9410-6.ch005.

[55] A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *Int. J. Comput. Networks Appl.*, vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[56] P. S. Ponnukumar, N. I. Francis Xavier, and R. Jaganathan, "Stable Plithogenic Cubic Sets," *J. Fuzzy Ext. Appl.*, vol. 6, no. 2, pp. 410–423, 2025, doi: 10.22105/jfea.2025.449408.1422.

[57] S. P. Priyadharshini, F. Nirmala Irudayam, and J. Ramkumar, "An Unique Overture of Plithogenic Cubic Overset, Underset and Offset," in *Studies in Fuzziness and Soft Computing*, 2025, pp. 139–156.

[58] R. Jaganathan, S. Mehta, and R. Krishan, *Intelligent Decision Making Through Bio-Inspired Optimization*. IGI Global, 2024. doi: 10.4018/979-8-3693-2073-0.

[59] S. P. Geetha, N. M. S. Sundari, J. Ramkumar, and R. Karthikeyan, "ENERGY Efficient Routing In Quantum Flying Ad Hoc Network (Q-Fanet) Using Mamdani Fuzzy Inference Enhanced Dijkstra's Algorithm (MFI-EDA)," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 9, pp. 3708–3724, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85197297302&partnerID=40&md5=72d51668bee6239f09a59d2694df67d6

[60] J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICACTA54488.2022.9752899.

[61] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning

Machine for Sentiment Analysis in Big Data," in *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICACTA54488.2022.9753203.

[62] R. Suganthi, J. Ramkumar, K. K. R. Chinthala, V. Valarmathi, M. Deenathayalan, and K. R. Dhanya, "Real-Time Applications of 6G in Augmented Reality/Virtual Reality (AR/VR) Ecosystems," in *Landscaping 6G: Unlocking the Power of Ultra-Fast Communication*, 2026, pp. 175–186. doi: 10.1201/9781003683599-11.

[63] R. Jaganathan, S. Mehta, and R. Krishan, "Preface," *Bio-Inspired Intell. Smart Decis.*, pp. xix–xx, 2024, [Online].

[64] J. Ramkumar and S. Vetrivel, "Deep Learning Insights into Defending Against Adversarial Attacks in IoT Systems," in *Strategic Approaches to Intrusion Detection in Cloud-IoT Ecosystem*, 2026, pp. 227–250. doi: 10.1002/9781394341979.ch9.

[65] J. Ramkumarr, B. Narasimhan, S. P. Priyadharshini, and R. Karthikeyan, "Optimizing Intrusion Detection with Bayesian Optimization in Reactive Intrusion Detection Systems (BO-RIDS) for Enhanced Security Performance," in *Communications in Computer and Information Science*, 2026, pp. 87–104. doi: 10.1007/978-3-032-17843-5_4.

[66] B. Suchitra, J. Ramkumar, and V. Valarmathi, "Rejuvenated Particle Swarm Optimization-Based Classifier for Big Sentiment Data Classification," in *Communications in Computer and Information Science*, 2026, pp. 41–54. doi: 10.1007/978-3-032-17834-3_3.

[67] R. Jaganathan and V. V., "Advanced real-time signal processing techniques for adaptive beamforming in beyond 5G wireless networkss," in *Signal Processing Roadmap: Technologies, Applications, and Future Directions*, 2026, pp. 123–137. doi: 10.1016/B978-0-443-33172-5.00005-4.

[68] M. P. Swapna, D. Rajeev, J. Ramkumar, and S. Chandran, "Gender Dynamics of Cyber Security Awareness in Kerala, India: Statistical and Machine learning Approach," in *2025 Gender and Technology Conference, GTC 2025*, 2025. doi: 10.1109/GTC64325.2025.11478141.

[69] V. Valarmathi, J. Ramkumar, and B. Suchitra, "Classification of Cross-Domain Sentiments in Big Data Using Renewed Genetic Algorithm," in *Communications in Computer and Information Science*, 2026, pp. 25–40. doi: 10.1007/978-3-032-17834-3_2.

[70] R. Jaganathan, S. Mehta, and R. Krishan, "Preface," *Intell. Decis. Mak. Through Bio-Inspired Optim.*, pp. xiii–xvi, 2024.

[71] R. Karthikeyan, J. Ramkumar, S. P. Priyadharshini, and B. Narasimhan, "Optimized Self-adaptive Intrusion Detection System (SPIDS) for Enhanced Network Security," in *Communications in Computer and Information Science*, 2026, pp. 71–86. doi: 10.1007/978-3-032-17843-5_3.

[72] S. P. Priyadharshini and J. Ramkumar, "Mappings Of Plithogenic Cubic Sets," *Neutrosophic Sets Syst.*, vol. 79, pp. 669–685, 2025, doi: 10.5281/zenodo.14607210.

[73] S. Mall and S. Chakraverty, "Chebyshev Neural Network based model for solving Lane-Emden type equations," *Appl. Math. Comput.*, vol. 247, pp. 100–114, 2014, doi: 10.1016/j.amc.2014.08.085.

[74] B. M. Benatti *et al.*, "Machine learning for the differentiation of unipolar and bipolar depression: a deep-phenotyped multimodal approach," *Neurosci. Appl.*, vol. 5, p. 106478, 2026, doi: https://doi.org/10.1016/j.nsa.2025.106478.