UDC 621.391:004.032.26 DOI: 10.30837/0135-1710.2025.186.040

S.V. SHTANGEY, L.I. MELNIKOVA, A.V. MARCHUK, O.V. LYNNYK, O.K. SOKOLOV

MODELING AND ANALYSIS OF GRAPH NEURAL NETWORKS FOR OPTIMIZATION ROUTING IN INFOCOMMUNICATION NETWORKS

A mathematical model for routing was formulated as a graph-based problem, with key metrics introduced, including delay, number of hops, and throughput. The architectural features of graph neural networks were analyzed, focusing on message passing, attention mechanisms, aggregation, and feature updating. A comparative analysis of GCN, GAT, and GENConv was conducted, justifying the selection of GENConv as the core architecture for edge-level classification in routing tasks. A model based on GENConv with an MLP decoder was developed and trained on a large graph dataset. Its performance was evaluated in terms of accuracy, average delay, and the success rate of route construction. A comparison with a classical algorithm was also performed regarding solution quality and execution time.

1. Introduction

In modern infocommunication networks, characterized by high dynamics, complex topology, and increasing demands for transmission speed and reliability, the problem of optimal routing remains critically unresolved. Traditional routing algorithms often fail to adapt to real-time changes. They disregard contextual network features and have limited self-organizing capabilities [1]. This results in several negative theoretical consequences. There is no universal mathematical model that combines the topological flexibility of graph structures with the adaptability of neural networks. The potential of Graph Neural Networks (GNNs) in the context of routing is not sufficiently understood, especially their ability to generalize to new topologies. There is also a lack of research that integrates machine learning with classical theories of network management.

The unresolved nature of the optimal routing problem leads to decreased efficiency of data transmission in large and complex networks (such as IoT, 5G, and SDN). It results in increased delays, packet losses, and node overload caused by non-adaptive routes. It also makes it impossible to scale existing solutions to heterogeneous or dynamic network environments [2].

At the same time, modern machine learning approaches, particularly GNNs, open new possibilities for modeling network behavior, accounting for traffic characteristics, and predicting optimal routes in context. The distinctive feature of using GNNs in routing tasks lies in their ability to work directly with the graph structure of the network, where nodes represent network devices and edges represent physical or logical connections with specific metrics, such as delay. Unlike traditional methods, GNNs can learn from historical examples and generate routes that align with the global structure of the network. Although the resulting paths are not always strictly optimal in terms of the shortest route, their computation is significantly faster, which is critical for large, heavily loaded, or rapidly changing topologies. In such conditions, the priority shifts from absolute optimality to the ability to respond quickly and consistently to changes. A particularly promising approach is edge-level classification, where the neural network decides whether each edge should be included in the route between a pair of nodes.

The theoretical significance of the study lies in the development of new routing models based on Graph Neural Networks, which are capable of accounting for the structural properties of the network, the contextual relationships between nodes, and the dynamics of change. The proposed approach makes it possible to rethink classical notions of routing as a static process and transform it into an adaptive, learning system.

The applied significance is determined by the need for intelligent solutions in modern

infocommunication systems, where traditional methods no longer provide the required level of performance. The results of the study can be applied to optimize routing in SDN networks, mobile MANET networks, and IoT networks. They can also be used to improve the efficiency of real-time traffic management and to develop self-learning routing systems capable of adapting to changes without external intervention.

2. Analysis of related work and research problem formulation

In recent years, GNNs have seen active development in the field of infocommunications. In [3], the RouteNet model is examined, which employs GNNs to predict delay, jitter, and packet loss in SDN networks. It demonstrates the ability to generalize to new topologies and traffic patterns not represented in the training data. However, the analysis of scientific sources indicates a number of limitations that hinder their practical application.

In [4], the GraphSAGE model is proposed for predicting network load. Despite achieving high accuracy in static topologies, the model does not adapt to dynamic changes, which limits its applicability in mobile networks.

The authors of [5] investigate the use of Graph Attention Networks (GAT) for routing in SDN. Although the architecture is flexible, it requires significant computational resources, making real-time deployment challenging.

In [6], GNNs are applied in IoT networks. The model improves delay metrics but does not take energy consumption into account, which is a critical parameter in IoT environments.

An approach that combines GNNs with reinforcement learning demonstrates adaptability but becomes unstable when the topology changes, reducing its reliability [7].

For route classification, the authors of [8] employ graph autoencoders. The model is effective on simulation data but has not been tested on real networks.

In [9], GNNs are proposed for detecting high-traffic nodes. However, the model disregards contextual relationships between nodes, which leads to incorrect routing.

GNN-based routing in MANET networks is analyzed in [10]. Although performance improves, the model suffers from delays caused by weight coefficient calculations.

The authors of [7] apply clustering before routing based on GNNs. The clustering is performed without considering traffic dynamics, which reduces accuracy.

For traffic congestion prediction in urban networks, GNNs are used in [11]. The model is not scalable to global infocommunication systems.

An adaptive GNN model is proposed for SD-WAN in [12]. Although the performance is high, the model does not consider security aspects.

In [13], GNNs are explored for energy saving. The model does not adapt to topology changes, which limits its applicability.

In [14], a GNN-based model is developed for routing in hybrid networks. Despite its effectiveness in a test environment, the model does not account for delays caused by route changes.

The use of GNNs in the context of QoS-oriented routing is analyzed in [15]. The model does not provide sufficient accuracy under high traffic variability.

Thus, most existing solutions fail to consider the dynamics of topology changes, have limited scalability, are not adapted to real-time operation, and do not account for energy efficiency or security.

The relevant scientific problem is the development of an adaptive, scalable, and energy-efficient routing model based on Graph Neural Networks, capable of operating in real time, accounting for topology dynamics, and ensuring secure data transmission. This study is aimed at addressing this problem.

3. Research aim and objectives

The aim of this study is to model and analyze Graph Neural Networks for optimizing routing in infocommunication networks.

Achieving this goal will enable the use of Graph Neural Networks in real time within dynamic networks, where decision-making speed is critical, particularly under conditions of high topological complexity, unstable channels, and strict performance requirements. The proposed approach opens prospects for further research, especially in the areas of multi-criteria routing, adaptation to real-time metric changes, and integration with other components of network intelligence.

To achieve this goal, the following tasks must be addressed:

– construction and generation of graph data for training;

– formulation of features and target labels;

– development of the model architecture and training;

– analysis of performance and time characteristics.

4. Materials and Methods

The subject of this research is the application of Graph Neural Networks to the routing problem in infocommunication networks.

The object of the study is the process of route construction within an infocommunication network.

The research methods include the analysis of existing routing algorithms, a theoretical examination of the operating principles of Graph Neural Networks, the design of a model architecture based on the GENConv framework, as well as its experimental training and evaluation on a generated dataset of network topologies.

GNNs are a class of models capable of efficiently processing data represented as graphs, preserving both the structural information about relationships between entities and the local features of those entities. GNNs have gained widespread adoption in tasks where topology plays a critical role – particularly in routing, where the graph reflects the physical or logical structure of a network [16].

In GNNs, a graph is typically represented as a 5-tuple $G = (V, E, X, F, A)$, where V is the set of nodes, E is the set of edges, $X \in R^{|V|\times d}$ is the node feature matrix, $F \in R^{|E|\times k}$ is the edge feature matrix, and $A \in \{0,1\}^{|V|\times|V|}$ is the adjacency matrix defining the graph structure. Here, d denotes the node feature dimension, and k denotes the edge feature dimension. In the case of a weighted graph, the binary adjacency matrix A is replaced by a weight matrix $W \in R^{|V|\times|V|}$, where each element w_{uv} corresponds to a routing metric (e.g., delay) between nodes u and v [17].

Unlike classical neural networks, where each input sample is treated independently, Graph Neural Networks (GNNs) update the representation of each node by considering not only its own features but also the information received from its neighboring nodes in the graph.

This process is known as message passing, and it involves iterative exchange of information between adjacent nodes. At each layer of the model, every node $v \in V$ updates its state h_v using the aggregated information from its neighbors $u \in \mathcal{N}(v)$:

$$h_v^{(l)} = UPDATE^{(l)}\left(h_v^{(l-1)}, m_v^{(l)}\right), \quad (1)$$

$$m_v^{(l)} = AGGREGATE^{(l)}(\{\phi\left(h_u^{(l-1)}, f_{uv}\right) : u = \overline{1,|V|}, u \in \mathcal{N}(v)\}), \quad (2)$$

where $h_v^{(l)}$ is the hidden state of node v at layer l, $l = \overline{1,L}$, with L being the number of layers in the model, $v = \overline{1,|V|}$, ϕ is a message generation function that takes into account the features of

the neighbor u and the edge (u, v), $AGGREGATE$ and $UPDATE$ are differentiable functions, which can be implemented, for instance, as mean, sum, or learnable transformations.

In addition to node features, edge features f_{uv} play a crucial role in GNNs, as they can encode routing metrics such as delay or bandwidth. These values can directly influence the message passing process, enabling the model to make more informed decisions about the relevance of information received from specific neighbors. These values can directly influence the message passing process, allowing the model to make more informed and selective decisions about the relevance of information received from individual neighbors.

Overall, GNNs provide a flexible framework for modeling dependencies in graphs, enabling each graph element (node or edge) to construct its representation based on both structure and local context.

In routing tasks, this allows models to learn from real traffic patterns or network topologies and to uncover patterns that are beyond the reach of traditional algorithms.

The first generations of Graph Neural Networks, such as Graph Convolutional Networks (GCN) and Graph Attention Networks (GAT), have served as the foundation for numerous models in graph-related tasks, including the processing of network topologies.

However, both architectures have limitations that significantly affect their effectiveness in complex, deep, or highly dynamic graphs – such as those found in telecommunications networks [18, 19].

GCN performs neighborhood feature aggregation using averaging weighted by the normalized adjacency matrix. The model updates are defined as follows [3]:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-1/2}\tilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)}\right), \tag{3}$$

where $\tilde{A} = A + I$ is the adjacency matrix with added self-loops, $\widetilde{D}$ is the diagonal degree matrix, $H^{(l)}$ are the hidden features, and $W^{(l)}$ are the layer-l learnable parameters.

Although this model is simple and effective, it does not account for differences between neighboring nodes. Moreover, deep architectures quickly encounter the problem of oversmoothing, where node representations become indistinguishable and lose their expressive power.

GAT addresses this limitation by introducing an attention mechanism – a dynamic approach that allows the model to independently determine which neighboring nodes are more important. For each edge (u, v), an attention coefficient α_{uv} is computed, which reflects the importance of the message sent from node u to node v. These coefficients are normalized via the softmax function, enabling their interpretation as weights in the aggregation process [9]:

$$\alpha_{uv} = \frac{\exp(LeakyReLU(a^T[Wh_u||Wh_v]))}{\sum_{k\in\mathrm{N}(v)}\exp(LeakyReLU(a^T[Wh_k||Wh_v]))}, \tag{4}$$

where $LeakyReLU$ is a modified version of the standard $ReLU$ activation function that allows a small negative gradient for negative inputs, typically with a slope coefficient of $\alpha \approx 0{,}2$. This helps to avoid the "dying neuron" problem where units stop learning, which is particularly important for computing attention coefficients that may take both positive and negative values.

The GENConv (Generalized Aggregation Graph Convolution) architecture, in turn, represents a more modern approach that combines the flexibility and expressiveness of attention mechanisms with the scalability and stability of classical models.

A key feature of GENConv is learnable aggregation – an aggregation function that is not fixed but is learned jointly with the rest of the model. For example, it can behave like a max

aggregator or a softmax-weighted average. This is achieved through a parameterized softmax function with temperature scaling:

$$m_{uv} = \frac{\exp(h_u/\tau)}{\sum_{k\in \mathrm{N}(v)} \exp(h_k/\tau)}, \tag{5}$$

where τ is a learnable temperature parameter. A small τ results in behavior similar to max aggregation (where a single message dominates), while a large τ leads to uniform averaging.

Thus, GENConv enables the model to autonomously adapt to the local graph structure and determine the most appropriate aggregation strategy depending on the number of neighbors, feature variability, and topological structure.

The feature update formula in GENConv is defined as:

$$h_v^{(l)} = MLP^{(l)}\left(h_v^{(l-1)} + AGG\left(\left\{ \frac{h_u^{(l-1)}}{\tau} : u \in \mathrm{N}(v) \right\} \right) \right), \tag{6}$$

where MLP denotes a nonlinear transformation with normalization and dropout, AGG is the learnable aggregator, and the addition implements a residual connection.

GENConv demonstrates stable performance even with a large number of layers (30+), whereas models like GCN and GAT typically suffer from performance degradation. This is particularly important in the context of routing tasks in telecommunication networks, where relevant information about the optimal path may be located at a considerable distance within the topology, and the model must be able to accurately capture it [18].

In classical tasks addressed by Graph Neural Networks, the primary objective is often node classification or link prediction. However, in the task of constructing a route between a given source-target node pair, edge classification becomes central – specifically, determining which edges should be included in the route.

Unlike the node-level approach, where the model generates a vector for each node and makes decisions based on it, in edge-level classification, vectors are constructed for edges. In this case, each edge $(u, v) \in E$ is represented as a combination of node features h_u, h_v, edge features f_{uv}, and additional information that may be specific to the given source–target pair. The input to the edge classifier takes the form of a concatenation:

$$z_{uv} = [h_u \parallel h_v \parallel f_{uv} \parallel d_u^s \parallel d_v^t], \tag{7}$$

where h_u, h_v are the hidden representations of the nodes after passing through the GNN, f_{uv} are the edge features, and d_u^s, d_v^t are heuristic distances to the source and target nodes, respectively, e.g., precomputed shortest-path distances, which add routing-specific context to the classification task.

The edge-level approach allows the model to frame the routing task as a binary classification problem, where each edge is evaluated to determine whether it belongs to the path between a given source s and target t. Based on these predictions, the model can construct a route that aligns with the graph context, taking into account both local features and the global structure.

In the next step, the vector is passed to a decoder implemented as a Multi-Layer Perceptron (MLP). The perceptron consists of several fully connected layers, each applying a linear transformation, an activation function (e.g., ReLU), as well as dropout or normalization. The final layer typically contains a single neuron with a sigmoid activation function, which outputs the probability that the edge is part of the route between s and t.

The edge classification approach using an MLP enables the model to simultaneously account for the topological context, connection characteristics, and relevance to the routing task. Unlike

models that attempt to navigate step-by-step from node to node, the edge-level approach allows for constructing a complete view of the route in the form of either a binary edge mask or a probabilistic field.

An advantage of the edge-level approach is its flexibility and accuracy compared to the node-level paradigm. The model can take into account the specific characteristics of each edge, as well as the interaction between the connected nodes.

5. Research results

5.1. Graph data construction and generation for training

To train the Graph Neural Network (GNN) tailored for solving routing tasks in complex and dynamic telecommunication networks, a custom synthetic dataset was constructed. In total, 3000 graphs were generated, each simulating the topology of a medium- or large-scale network with a variable number of nodes, structural heterogeneity, and diverse connection characteristics.

The graphs were generated using the Python programming language and the NetworkX library [19,20,21], which provides tools for creating random graphs and working with topological structures. The number of nodes in each graph was randomly selected within the range of 50 to 100, allowing for both simple and structurally rich scenarios. Graph construction employed the networkx.gnm_random_graph generator, which creates a graph based on a fixed number of nodes and a randomly chosen number of edges in the range from N+10 to 3N, where N is the number of nodes. Connectivity was enforced as a mandatory condition and verified using nx.is_connected.

Each edge in the graph was assigned a transmission delay, represented as a random value in the range from 0.001 to 2.0 seconds. This was implemented using the random.uniform function, which ensured a uniform distribution of delays within the specified interval. Such modeling makes it possible to simulate both high-speed communication links and slow or congested lines, thereby providing sufficient diversity in training conditions.

For each graph, a random pair of nodes was selected to serve as the source and target. Based on the edge weights, the shortest path between them was computed using Dijkstra's algorithm as implemented in the NetworkX library. This path was then used to generate binary labels: edges belonging to the path were marked as positive (label 1), while all others were marked as negative (label 0). In this way, the task was formulated as an edge classification problem in the context of route construction.

All graphs (Figs. 1, 2) were converted into a format compatible with the PyTorch Geometric library [21]. Each sample includes an edge index matrix (edge_index), edge features (edge_attr), node features (x), and labels (edge_label). Node features consisted of binary indicators specifying whether the node is the source or target, the normalized degree, and the number of hops to the target node (Table 1). Edge features were represented by the delays assigned during graph generation.

Table 1

Structure in .pt for PyTorch Geometric

Field	Description	Dimension
x	Node features	[N,d]
edge_index	Indices of node pairs for each edge	[2,E]
edge_attr	Edge features (delay)	[E,k]
edge_label	Binary labels: whether the edge belongs to the route	[E]
source	Index of the source node	[1]
target	Index of the target node	[1]

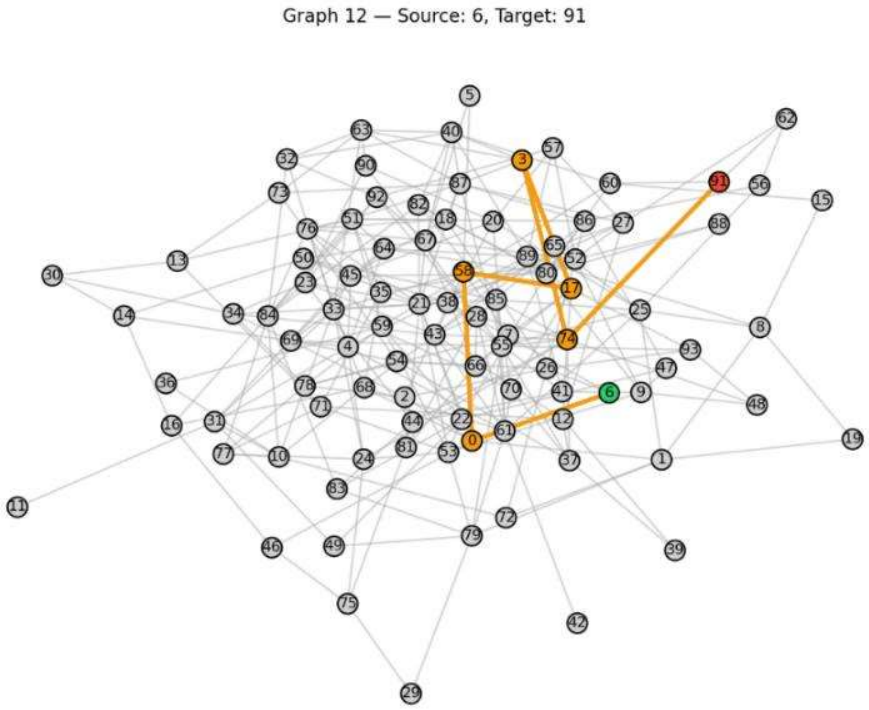

Fig. 1. Example of generated graph 1

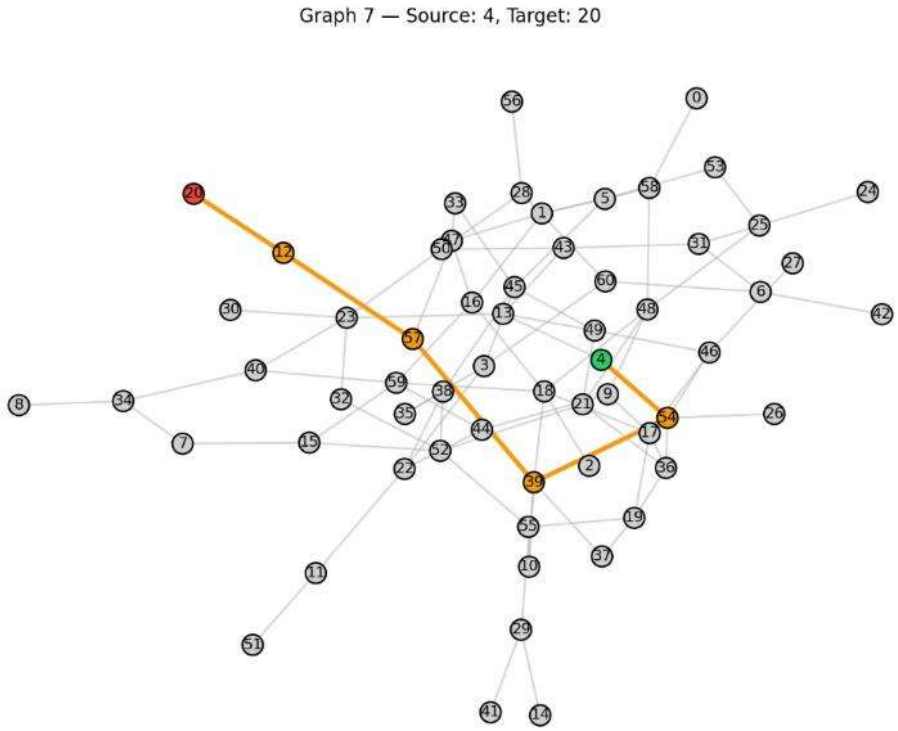

Fig. 2. Example of generated graph 2

5.2. Feature and target label definition

Since the model is tasked with classifying edges in the context of a route between a given pair of nodes, it was critically important to design input features that not only capture local properties of graph elements but also provide context specific to the routing task.

To this end, several types of features were selected for both nodes and edges, enabling the model to effectively localize the source and target, infer the routing direction, and account for the quality of connections.

Node features include binary indicators is_source and is_target, which are essential for enabling the model to distinguish the currently active source-target pair (s, t). Without this information, the routing task would be ill-defined, since the model processes the graph as a whole and must explicitly know where the route starts and ends. These features, therefore, provide task-specific addressability.

The normalized node degree is a simple yet important topological feature that provides the model with insight into the local connectivity of a node. A high degree is often associated with key routing nodes, while peripheral nodes typically have limited capacity for traffic forwarding.

Additionally, heuristic meta-information is introduced in the form of the number of hops from a node to the target. This value is computed based on the shortest path without considering edge weights and provides a general notion of the node's distance from the destination.

Although this feature is not precise in a weighted graph, it helps the model orient itself toward the target by combining local decisions with global context.

As for edges, the sole but crucial feature is latency – a value that directly corresponds to the routing metric. In real-world networks, latency or its derivatives are typically used as the primary criterion for path optimization, making the inclusion of this feature essential for effective model training.

Training labels are generated based on the shortest path between nodes s and t, computed using Dijkstra's algorithm. This approach frames the learning task as binary edge classification, where the positive class corresponds to edges that belong to the optimal path, and the negative class to all others. Such formulation scales well and enables the use of standard neural network optimization techniques for a pathfinding task that would otherwise be combinatorially complex.

Thus, the constructed system of features and labels enables the model to simultaneously account for local graph parameters, the global routing objective, and the network metric to be optimized. This establishes a logical bridge between the graph's topological structure and the practical requirements of route construction in a network.

5.3. Model architecture and training

The developed model (Table 2) implements an edge-level classification approach using a Graph Neural Network based on GENConv convolutional layers. The architecture is designed to enable efficient information propagation between graph nodes and accurate evaluation of each edge in the context of a specific routing task. At the core of the model is the EdgeEncoderOptimized module, which consists of three sequential GENConv layers and a single MLP decoder that takes as input the constructed feature vector for each edge.

Each GENConv layer performs message aggregation from neighboring nodes using softmax normalization with a learnable temperature coefficient, allowing the aggregation behavior to adapt from uniform to selective. After each convolutional layer, BatchNorm1d is applied to stabilize training, and Dropout is used to prevent overfitting. All hidden layers have a dimensionality of 64.

As a result, each node in the graph obtains a generalized representation that incorporates not only its own features but also the context of its local neighborhood over multiple hops.

Table 2

Neural network architecture

Component	Type	Parameters
conv1	GENConv	in: 4, out: 64, aggr: softmax
bn1	BatchNorm1d	64
conv2	GENConv	in: 4, out: 64, aggr: softmax
bn2	BatchNorm1d	64
conv3	GENConv	in: 4, out: 64, aggr: softmax
bn3	BatchNorm1d	64
dropout	Dropout	p = 0,3
edge_mlp	MLP decoder (3 layers)	193 → 64 → 32 → 1, ReLU + BatchNorm

As a result, a vector of size 193 is formed and passed to the MLP decoder, which consists of three fully connected layers:

– First layer: Linear(193 → 64) with ReLU activation and BatchNorm;

– Second layer: Linear(64 → 32) with ReLU;

– Third layer: Linear(32 → 1) without activation (output is a logit for subsequent sigmoid processing).

ReLU activations enable the model to learn nonlinear dependencies between structural and contextual parameters, while BatchNorm after the first layer ensures stability of the activation distribution and accelerates convergence.

For training, the generated data was divided into a training set (80% of the dataset) and a validation set (20% of the dataset). Binary cross-entropy with logits was used as the loss function during training, incorporating class imbalance through weighted scaling. Optimization was performed using the Adam algorithm with a fixed learning rate of 10^{-3}. The batch size was set to 1 (one graph per iteration), and the model was trained for 35 epochs. Performance evaluation relied on metrics such as loss, accuracy, and the success rate of correctly reconstructed routes.

Figure 3 shows the loss function, which steadily decreases on the training data and remains well-controlled on the validation set.

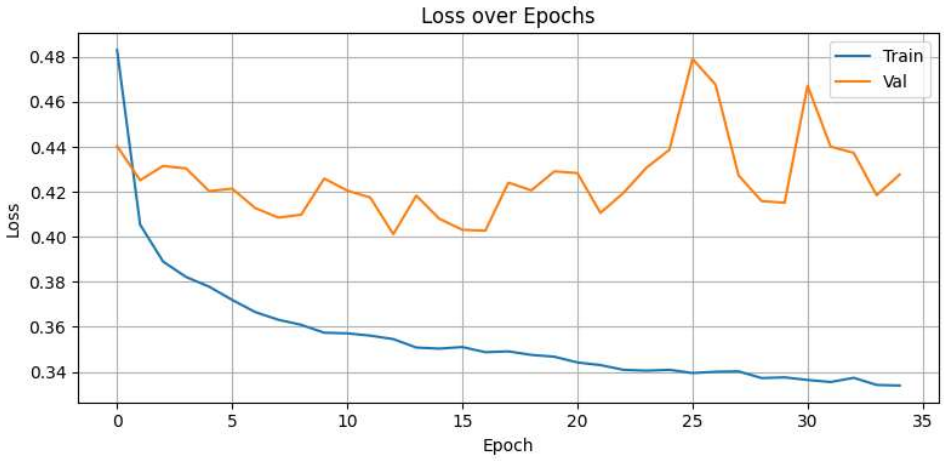

Fig. 3. Loss function plot on training and validation data

Figure 4 shows the edge classification accuracy, where the validation accuracy consistently exceeds 94%, occasionally reaching 95-96%, indicating reliable generalization. Figure 5 presents detailed results from the final epochs, including all key metrics. The most indicative are the high route construction success rate and stable accuracy.

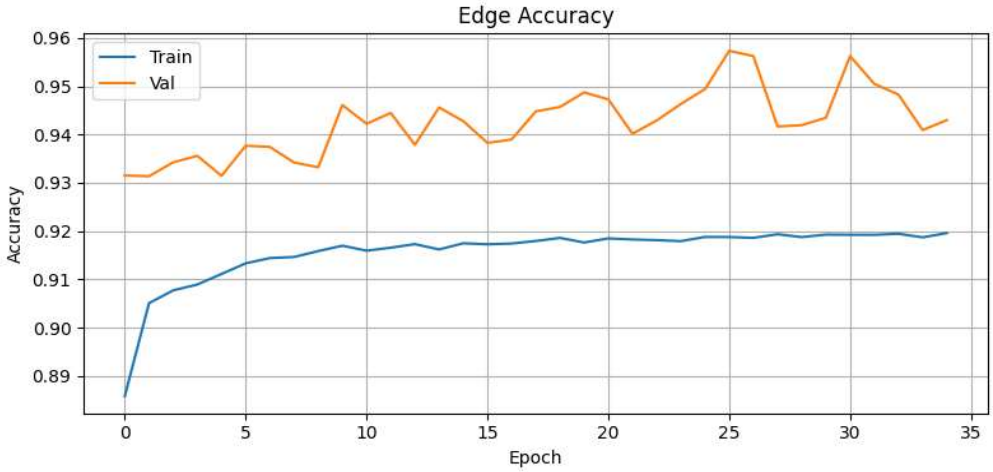

Fig. 4. Accuracy plot on training and validation data

```
Epoch 29 | Train Loss: 0.3372 | Acc: 0.919 | Lat: 2.6501 | Success: 99.6%
         | Val   Loss: 0.4159 | Acc: 0.942 | Lat: 2.3910 | Success: 91.2%
Epoch 30 | Train Loss: 0.3375 | Acc: 0.919 | Lat: 2.6550 | Success: 99.7%
         | Val   Loss: 0.4152 | Acc: 0.943 | Lat: 2.3262 | Success: 90.5%
Epoch 31 | Train Loss: 0.3364 | Acc: 0.919 | Lat: 2.6426 | Success: 99.5%
         | Val   Loss: 0.4672 | Acc: 0.956 | Lat: 2.1974 | Success: 84.5%
Epoch 32 | Train Loss: 0.3355 | Acc: 0.919 | Lat: 2.6598 | Success: 99.7%
         | Val   Loss: 0.4401 | Acc: 0.951 | Lat: 2.3071 | Success: 88.5%
Epoch 33 | Train Loss: 0.3373 | Acc: 0.919 | Lat: 2.6595 | Success: 99.8%
         | Val   Loss: 0.4374 | Acc: 0.948 | Lat: 2.2853 | Success: 87.0%
Epoch 34 | Train Loss: 0.3342 | Acc: 0.919 | Lat: 2.6294 | Success: 99.8%
         | Val   Loss: 0.4185 | Acc: 0.941 | Lat: 2.4160 | Success: 92.7%
Epoch 35 | Train Loss: 0.3339 | Acc: 0.920 | Lat: 2.6395 | Success: 99.5%
         | Val   Loss: 0.4278 | Acc: 0.943 | Lat: 2.3346 | Success: 90.3%
```

Fig. 5. Training results during the final epochs

Thus, the developed model demonstrates consistently high classification performance, is capable of constructing coherent routes under varying graph complexities, and shows effective generalization even in the presence of noise and topological variability. The combination of GENConv and the MLP decoder has proven to be effective for edge-level routing tasks.

5.4. Performance and timing analysis

To evaluate not only classification accuracy but also the quality of the constructed routes, a visualization was performed comparing the predicted paths with the reference (shortest) routes for a set of graphs.

Figures 6 and 7 show examples of constructed paths, where the predicted route is highlighted in yellow and the optimal (shortest) route in blue. In the first case (Fig. 6), the model precisely reproduced the path found by Dijkstra's algorithm, with an identical total delay of 2.2731 (Fig. 8). In the second example (Fig. 7), the model's route differs, it has fewer hops but slightly higher total delay (Fig. 9). This result indicates that the model is capable of discovering alternative paths that are close to optimal or represent a trade-off between different metrics.

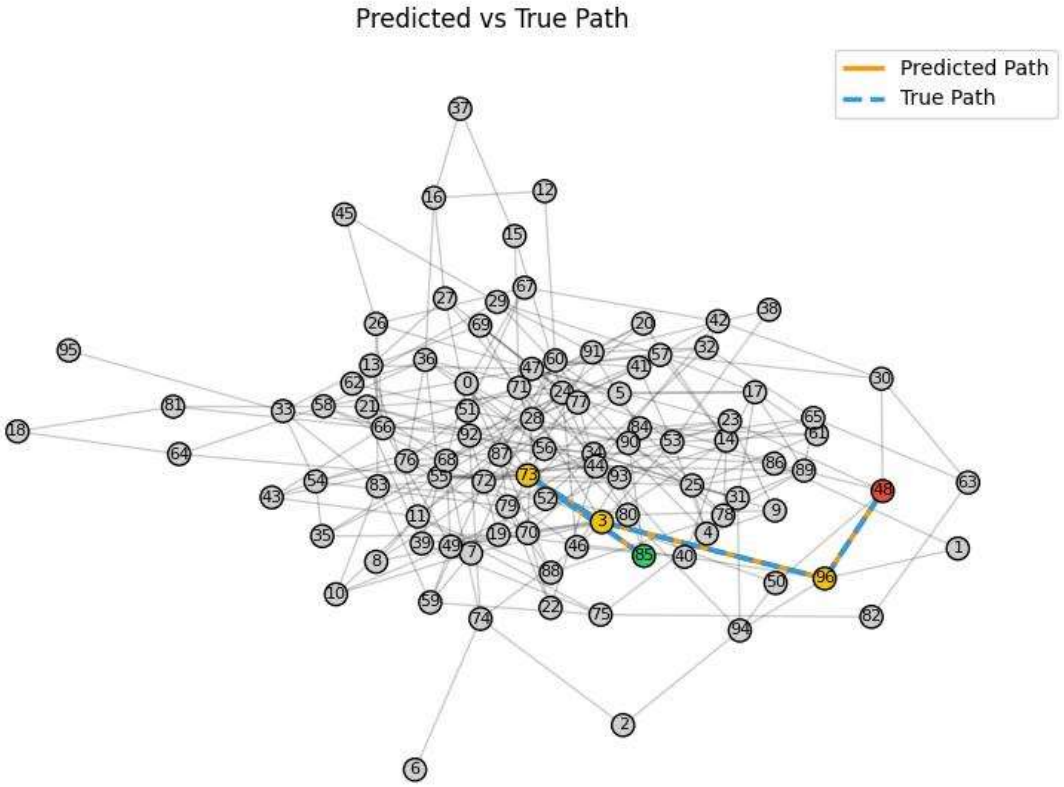

Fig. 6. Comparison of predicted and optimal routes, example 1

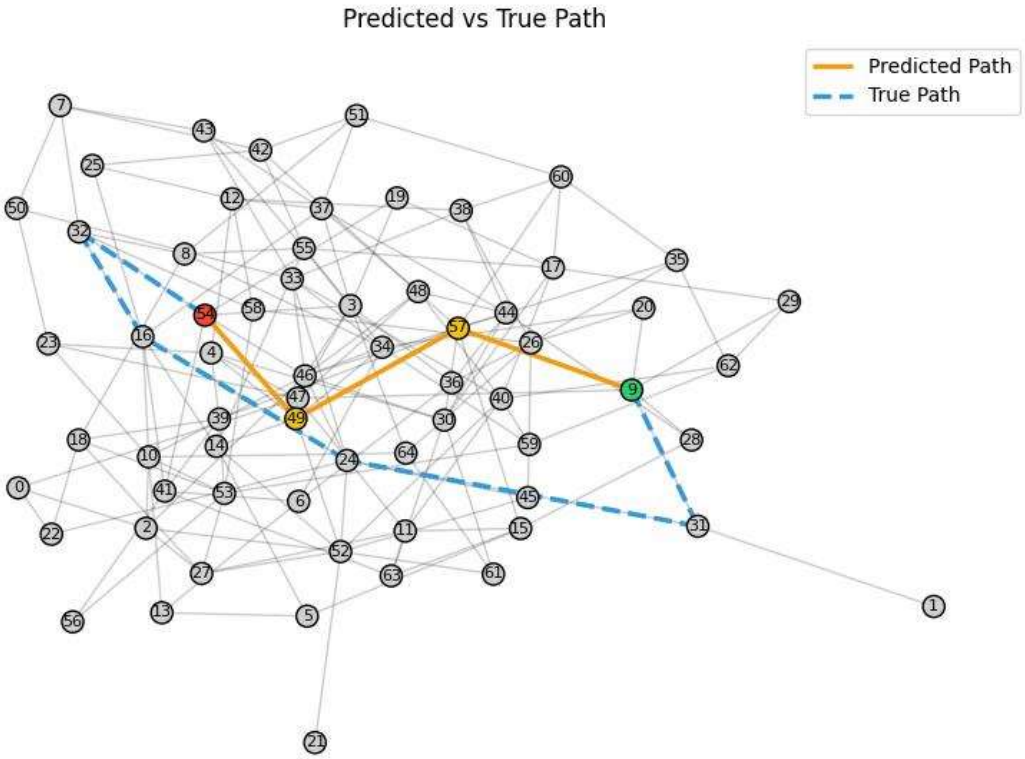

Fig. 7. Comparison of predicted and optimal routes, example 2

```
Predicted path: [85, 73, 3, 96, 48]
Ground truth : [85, 73, 3, 96, 48]
Path length  : 5 (true: 5)
Pred latency: 2.2731
True latency : 2.2731
```

Fig. 8. Comparison metrics for case 1

```
Predicted path: [9, 57, 49, 54]
Ground truth : [9, 31, 24, 16, 32, 54]
Path length  : 4 (true: 6)
Pred latency: 1.9908
True latency : 1.8471
```

Fig. 9. Comparison metrics for case 2

Figure 10 presents a comparison of delays for each of the 500 tested graphs, showing both the true latency and the latency of the routes predicted by the model.

Despite the structural variability of the topologies, the two curves exhibit a strong correlation. This indicates that even when the predicted path differs from the optimal one, the model can reproduce the delay with sufficient accuracy, staying within an acceptable margin of deviation.

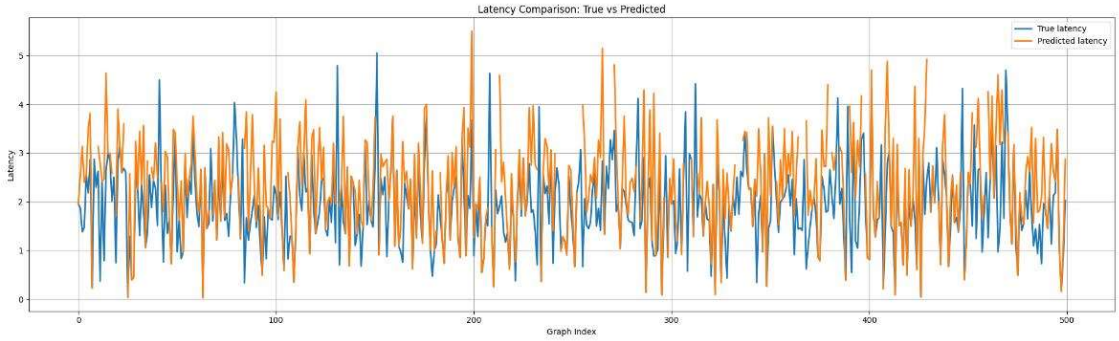

Fig. 10. Comparison of predicted and optimal paths by total latency on the test set

A performance comparison between the GNN model and the classical routing approach is presented in Figure 11. It shows the average route construction time for both Dijkstra's algorithm and the GNN model in batch inference mode. In graphs with 50–100 nodes, the model demonstrated approximately twice the throughput. The experiment was conducted in the Google Colab cloud environment with the following specifications:

- processor: Intel(R) Xeon(R) CPU @ 2.20GHz, 2 logical cores, 56 MB cache;
- RAM: ~13 GB;
- GPU: NVIDIA Tesla T4, 16 GB VRAM, CUDA 12.4.

This result is attributed to the fact that the model does not need to reconstruct the path from scratch for each query – it operates as a universal router for the entire topology.

In addition to speed, a key advantage of the neural network – based approach is its ability to instantly respond to changes in the network structure without re-running algorithmic search. This is particularly important in dynamic environments, where the number of nodes or edge weights may change in real time.

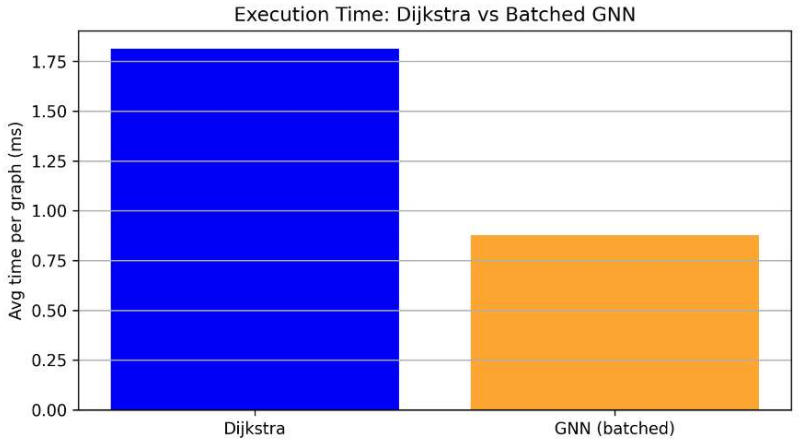

Fig. 11. Comparison of average routing time between Dijkstra's algorithm and the GNN model

6. Discussion of research results

This research has confirmed that graph neural networks based on the GENConv architecture can effectively address routing problems in complex network topologies. The model demonstrated high edge-level classification accuracy, exceeding 94% on the test sets, indicating its ability to generalize knowledge and adapt to novel graph scenarios.

Notably, in some of the tested graphs, the model generated routes that did not exactly match the classically computed shortest paths but exhibited similar or even better topological properties, particularly a reduced number of hops. This suggests that the model is capable of adaptively selecting trade-offs between metrics such as latency, path length, and reliability, which is especially relevant in dynamic network environments.

The analysis of temporal characteristics revealed the advantages of the GNN-based approach during inference. The model demonstrates twice the performance compared to Dijkstra's algorithm on graphs with 50-100 nodes, which is explained by the absence of the need to recompute the route for each query. This enables rapid response to changes in network topology – a critical requirement for modern infocommunication environments.

The GENConv architecture with learnable aggregation ensures stability at greater network depth and flexible adaptation to the local structure of the graph. Even in cases of topological variability or noise, the model demonstrates consistent performance by generating coherent routes. The inclusion of additional heuristic information (such as the number of hops to the target node) further enhances the model's ability to navigate the global context of the route.

Despite the high performance, questions remain open regarding scalability to ultra-large topologies and improving interpretability for integration into critical systems. The proposed edge-level approach opens up promising directions for further research – particularly in the areas of multi-objective routing, training on real-world networks, real-time metric adaptation, and integration with other components of SDN infrastructure.

7. Conclusions

Within the scope of the research objective, a routing model based on a Graph Neural Network with edge-level classification was developed, implemented, and experimentally investigated. The model was built on the GENConv architecture, and the following tasks were addressed.

A custom synthetic dataset was created for training the Graph Neural Network aimed at solving the routing problem in complex and dynamic telecommunication networks. In total, 3000 graphs were generated, each modeling the topology of a medium- or large-scale network with a variable number of nodes, structural heterogeneity, and diverse connection characteristics. The

graphs were generated using the Python programming language and the NetworkX library, which provides tools for generating random graphs and working with topological structures.

Several types of features were selected for both nodes and edges to enable the model to effectively localize the source and target, navigate the route direction, and account for connection quality. The node features include the binary indicators is_source and is_target, which are necessary for the model to distinguish the currently active pair of nodes.

A model was developed that implements the edge-level classification approach using a Graph Neural Network based on GENConv convolutional layers. The architecture enables efficient information propagation between graph nodes as well as accurate evaluation of each edge in the context of a specific routing task. At the core of the model lies the EdgeEncoderOptimized module, which consists of three sequential GENConv layers and one MLP decoder that processes the generated feature for each edge.

A performance comparison was conducted between the graph-based model and Dijkstra's algorithm. It was found that in inference mode, the graph model operates significantly faster, especially on large graphs, and does not require re-exploring the entire route space for each query. This makes the proposed approach suitable for real-time use in dynamic networks, where decision-making speed is critical.

The obtained results confirm that Graph Neural Networks can serve as an effective alternative to traditional routing algorithms, particularly under conditions of high topological complexity, channel instability, and strict performance requirements. The proposed approach opens new perspectives for further research, especially in the areas of multi-criteria routing, adaptation to real-time metric changes, and integration with other components of network intelligence.

References:

1. D. Medhi & K. Ramasamy, *Network routing. Algorithms, Protocols, and Architectures.* San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. doi: 10.1016/b978-0-12-088588-6.x5000-1.

2. О. В. Лемешко, О. С. Єременко, М. О. Євдокименко, А. С. Шаповалова та Б. Слейман, *Моделювання та оптимізація процесів безпечної та відмовостійкої маршрутизації в телекомунікаційних мережах: монографія.* Харків, Україна: ХНУРЕ, 2022.

3. K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros and A. Cabellos-Aparicio, "RouteNet: Leveraging Graph Neural Networks for Network Modeling and Optimization in SDN," *IEEE Journal on Selected Areas in Communications,* vol. 38, no. 10, pp. 2260-2270, Oct. 2020. doi: 10.1109/JSAC.2020.3000405.

4. J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu et al, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020. doi: 10.1016/j.aiopen.2021.01.001.

5. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 1, pp. 4-24, Jan. 2021. doi: 10.1109/TNNLS.2020.2978386.

6. H. Kim, J. Park, and Y. Lee, "Routing optimization in IoT networks using graph neural networks, " *Sensors,* vol. 22, no. 3, pp. 1–15, 2022.

7. J. Ramirez and M. Ortega, "Cluster-aware graph neural routing for dynamic networks," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2345–2357, Feb. 2024.

8. Y. Ln, X. Wang, and J. Liu, "Graph autoencoders for route classification in software-defined networks," *IEEE Access,* vol. 11, pp. 45678–45689, 2023.

9. M. Ahmed, T. Rahman, and S. Chowdhury, "Traffic hotspot detection using graph neural networks," *Journal of Network and Computer Applications,* vol. 202, pp. 1–12, 2024

10. L. Chen, K. Zhao, and Y. Sun, "GNN-based routing in mobile ad hoc networks," *Ad Hoc Networks*, vol. 145, pp. 102–115, 2024.

11. T. Tanaka, H. Saito, and K. Fujimoto, "Urban congestion prediction using graph neural networks," *IEEE Transactions on Intelligent Transportation Systems,* vol. 26, no. 4, pp. 789–798, Apr. 2025.

12. D. Ivanov and P. Smirnov, "Adaptive GNN-based routing for SD-WAN architectures," *IEEE Communications Letters*, vol. 29, no. 5, pp. 1123–1127, May 2025.

13. M. Kowalski, A. Nowak, and E. Zielinska, "Energy-efficient routing in wireless sensor networks using GNN," *Sensors*, vol. 25, no. 6, pp. 1–14, 2025.

14. Y. Zhang, L. Xu, and H. Li, "Graph neural routing in hybrid network topologies," *IEEE Transactions on Network and Service Management,* vol. 18, no. 3, pp. 456–468, Mar. 2025

15. V. Petrov, N. Koval, and D. Kravets, "QoS-aware routing with graph neural networks in heterogeneous networks," *IEEE Systems Journal,* vol. 19, no. 2, pp. 987–996, Apr. 2025

16. W. L. Hamilton, *Graph representation learning.* Springer Cham, 2020. doi: 10.1007/978-3-031-01588-5

17. Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang and P. S. Yu, "A Comprehensive Survey on Graph Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems,* vol. 32, no. 1, pp. 4-24, Jan. 2021, doi: 10.1109/TNNLS.2020.2978386

18. "A gentle introduction to graph neural networks," Distill. [Online] Available: https://distill.pub/2021/gnn-intro/ Accessed on: July, 26, 2025.

19. "NetworkX documentation," NetworkX. [Online] Available: https://networkx.org/ Accessed on: July, 26, 2025.

20. "PyG Documentation," Pytorch_geometric documentation. [Online] Available: https://pytorch-geometric.readthedocs.io/en/latest/ Accessed on: July, 26, 2025.

21. S. Shtangey, L. Melnikova and O. Sokolov, "Modeling and analysis of graph neural networks for routing optimization in infocommunication networks," Zenodo, Jul. 30, 2025. doi 10.5281/zenodo.16616697.

Надійшла до редколегії 20.08.2025 р.

Shtangey Svitlana Viktorivna, Ph.D. in Technical Sciences, Associate Professor, Associate Professor at the Department of Infocommunication Engineering V.V. Popovsky, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: Svitlana.shtanhei@nure.ua, ORCID: https://orcid.org/0000-0002-9200-3959. (Academic advisor of the higher education applicant Sokolov O.K.).

Melnikova Lubov Ivanivna, PhD in Technical Sciences, Associate Professor, Associate Professor of the Department of Infocommunication Engineering V.V. Popovsky, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: liubov.melnikova@nure.ua, ORCID: https://orcid.org/0000-0003-0439-7108

Marchuk Artem Volodymyrovych, PhD in Technical Sciences, Associate Professor, Associate Professor of the Department of Infocommunication Engineering V.V. Popovsky, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: artem.marchuk@nure.ua, ORCID: https://orcid.org/0000-0002-2720-3954

Linnyk Olena Vyacheslavivna, PhD in Technical Sciences, Associate Professor, Associate Professor of the Department of Mechanical and Biomedical Engineering, National Technical University "Dnipro Polytechnic", Dnipro, Ukraine, e-mail: linnyk.ol.o@nmu.one, ORCID: https://orcid.org/0000-0002-4906-3796

Sokolov Oleksandr Kuokovych, higher education applicant, group IST-23-1, Faculty of Infocommunications, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine, e-mail: oleksandr.sokolov1@nure.ua, ORCID: https://orcid.org/0009-0005-5382-2774